OS8805

MOST System On Chip

Final Product Data Sheet

DS8805FP5

Apr. 2003

MOST®

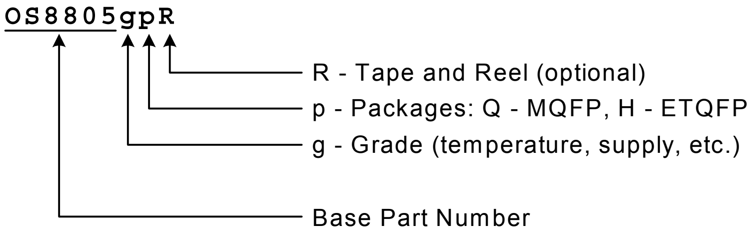
Media **O**riented **S**ystems **T**ransport

Multimedia and Control Networking Technology

RESTRICTED ACCESS

OASIS SiliconSystems

Ordering Information


OS8805gpR
R - Tape and Reel (optional)
p - Packages: Q - MQFP, H - ETQFP
g - Grade (temperature, supply, etc.)
Base Part Number

Valid Part Numbers:

Order Number	Grade		Package
	Temperature	Supply	
OS8805AQ	T_J = -40 to +150 °C	3.3 V ±5 %	128-pin MQFP
OS8805AQR	T_J = -40 to +150 °C	3.3 V ±5 %	128-pin MQFP, Tape and Reel
OS8805AH	T_J = -40 to +150 °C	3.3 V ±5 %	128-pin ETQFP
OS8805AHR	T_J = -40 to +150 °C	3.3 V ±5 %	128-pin ETQFP, Tape and Reel

This table represents valid part numbers at the time of printing and may not represent parts that are currently available. For the latest list of valid ordering numbers for this product, please contact the nearest sales office (as listed on back page).

Intellectual Property

© Copyright 2001-2003 Oasis SiliconSystems. The information within this document is Oasis SiliconSystems intellectual property. Duplication of this document without the expressed written permission from Oasis SiliconSystems is prohibited. The information in this document is considered "Restricted Access", and must not be sent to third parties without written permission from Oasis SiliconSystems.

Trademarks

MOST is a registered trademark of Oasis SiliconSystems. All other trademarks used in this document are proprietary of their respective owners.

Patents

There are a number of patents and patents pending on the MOST technology. The rights to these patents are not granted without any specific Agreement between the users and the patent owners.

Conventions

Within this manual, the following abbreviations and symbols are used to improve readability.

Example	Description
BIT	Name of a single bit within a register
REG.BIT	Name of a single bit (BIT) in register REG
x..y	Range from x to y, inclusive
BITS[m:n]	Groups of bits (or pins) from m to n, inclusive
PIN	Pin Name
0xzzz	Hexadecimal number (value zzz)
zzh	Hexadecimal number (value zz)
rsvd	Reserved memory location. Must write 0, read value indeterminate
`code`	Instruciton code
Multi Word Name	Used for multiple words that are considered a single unit, such as: *Resource Allocate* message, or *Connection Label*, or *Decrement Stack Pointer* instruction.
Section Name	Section or Document name.
$\overline{\text{VAL}}$	Over-bar indicates active low pin or register bit
bREG	Single byte MOST register
mBUF	Multi-byte MOST buffer
x	Don’t care

Data Sheet Versions

Doc. Number	Date	Description
DS8805AP2	Aug. 2001	First released Data Sheet
DS8805AP3	Mar. 2002	Second released Data Sheet
DS8805AP4	July 2002	Third released Data Sheet
DS8805FP5	Apr. 2003	First Final Product Data Sheet

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Overview

The OS8805 is a flexible hardware platform that can be programmed for a wide variety of signal processing system solutions. Software is available from Oasis SiliconSystems for applications including a digital radio, amplifiers, and active speakers. Programming tools such as emulators, assemblers, and C compilers are available to support user-developed applications.

The OS8805 is a monolithic CMOS integrated circuit, which combines high-speed micro-controllers and digital signal processors, with high quality analog circuitry. The OS8805 has four processors, eleven ADCs and DACs, and a variety of digital I/O. The Host Controller has 128k bytes of on-chip Flash Program memory, supporting easy updates during the design process as well as in the field. The programmable Host Controller and fixed-point DSPs enable the OS8805 to be a cost-effective solution for a wide variety of applications.

The MOST Processor provides a standard interface to the MOST Network. It generates the timing for all components on the chip, allowing the entire chip to operate synchronously to the Network. Even when not connected to the Network, the MOST Processor efficiently routes real-time data between resources, such as the DSPs, Source Ports, and Source Converters.

The OS8805 is a complete MOST (Media Oriented Systems Transport) Network Transceiver device capable of more than 24 Mbit/s data throughput when interfacing to a MOST optical Network. All relevant Network management functions are handled on-chip providing a complete system interface to the Physical layer components. Minimal additional components are required due to the high-level of integration on chip. An ultra-low jitter on-chip PLL guarantees high-quality audio and video transmission, and clock recovery over a wide frequency range.

A typical MOST Network node would consist of the OS8805 and the Physical layer devices. The MOST transceiver handles the Data-Link layer protocols and the Host Controller handles the Network layer and higher. Oasis SiliconSystems also offers a NetServices software stack, written in ANSI C, to ease implementation of the MOST Specification software protocol; thereby, drastically shrinking development time. NetServices can be ported to the Host Controller, along with the application's code for the particular node. All Network management functions can be off-loaded from the programmer allowing full concentration on the application being developed. Figure 1-1 illustrates the OS8805, with the matching software-protocol stack implementation. This Figure illustrates the OS8805 streaming synchronous data between the MOST Network and the on-chip ADCs, and DACs, and DSPs. Source Ports even support streaming data to off-chip high-speed controllers.

To minimize costs, the MOST Network supports a peer-to-peer methodology, eliminating excessive hardware overhead such as a hub (although hub-based architectures can also be implemented). In addition to handling Network interface and communication management functions, the MOST transceiver also handles all of the important low-level Network management functions; such as node position sensing (Plug-and-Play), network delay detection, start up and shut down, as well as error reporting, fail-safe operation, and channel allocation. Placing all these features in hardware, frees up much of the Host Controller MIPS to focus on higher-level Network functions.

The MOST Network actually consists of three simultaneous networks operating across a single low-cost plastic fiber:

- a Control network to manage the Network and communicate control data across the network,
- a Packet-based network to support nodes that communicate data,
- and a Streaming network that supports high-speed synchronous data with extremely low overhead (such as audio and video).

Each of these three networks, or *data transfer methods*, operates independently (not affecting the other two), providing a robust, dependable, and deterministic system architecture. The OS8805 supports the Control and Streaming portions of the MOST Network, by default. The OS8805 also supports Packet messages if enabled, otherwise it will not acknoledge reception of asynchronous Packets.

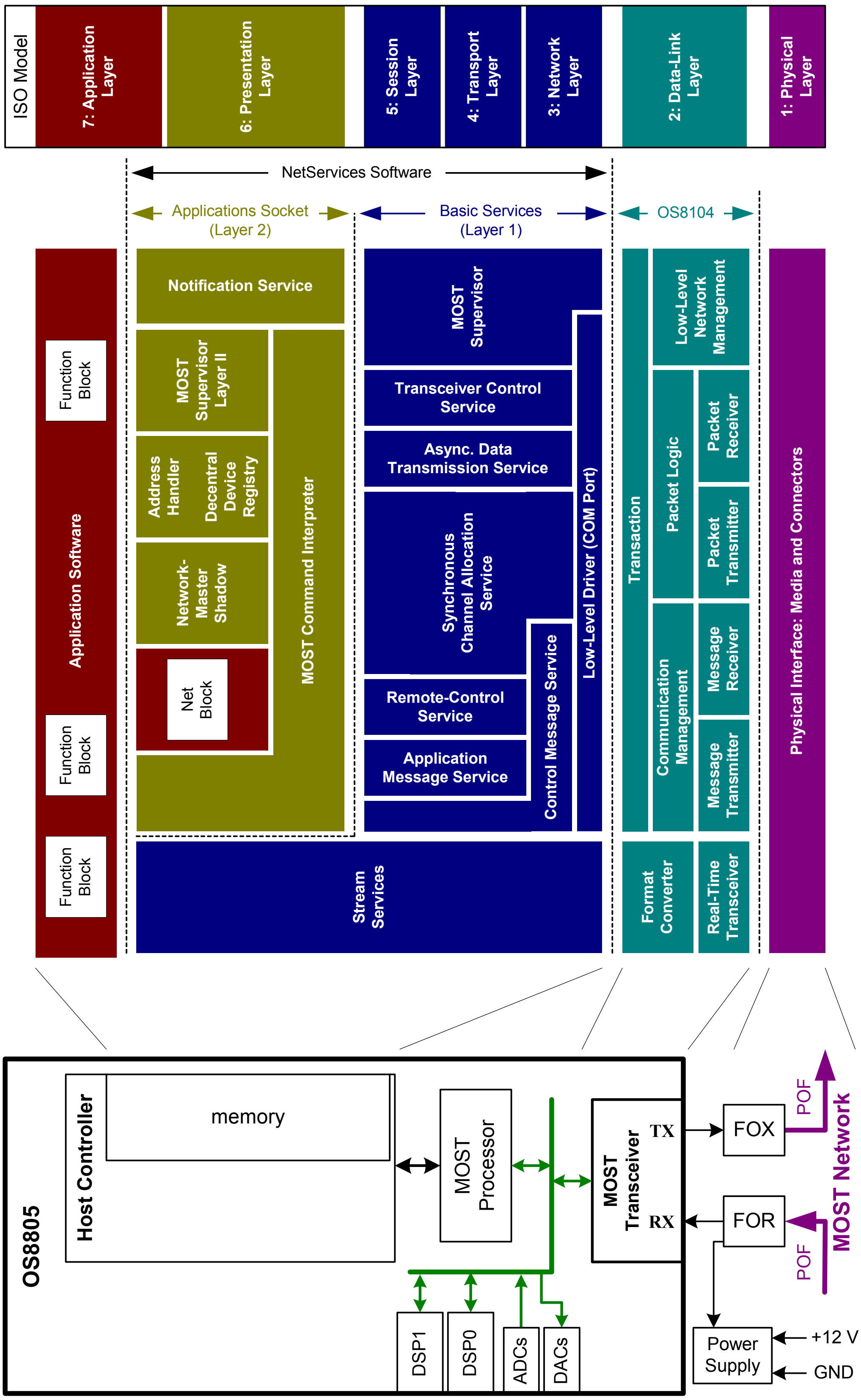


Figure 1-1: MOST Hardware/Software System Overview

As shown in Figure 1-2, the chip contains four main processors: the Host Controller, the MOST Processor, and two DSPs. The Host Controller operates on the Control bus and communicates with the external Host System. This Controller manages the peripherals on the Control Bus, loads programs into the DSP memories, and configures the MOST Processor and connection to the MOST Network. The data crossing the Control bus is generally low-speed data such as configuration information and messages to or from the MOST Network. The Controller interfaces to the external system through a Control Port. For external systems with intelligence, the Control Port (and the chip) can be configured as a serial device. For stand-alone applications, the Control Port can be configured as a master, wherein the Controller manages the external peripherals in a similar fashion as the peripherals on the Control bus.

The Host Controller generally operates out of on-chip Flash memory, but also supports operation from external memory through the external memory port. If the Controller uses external memory, then DSP0 is limited to the internal data memory. If the Controller runs out of internal Flash, DSP0 can use the external memory port for data-memory expansion.

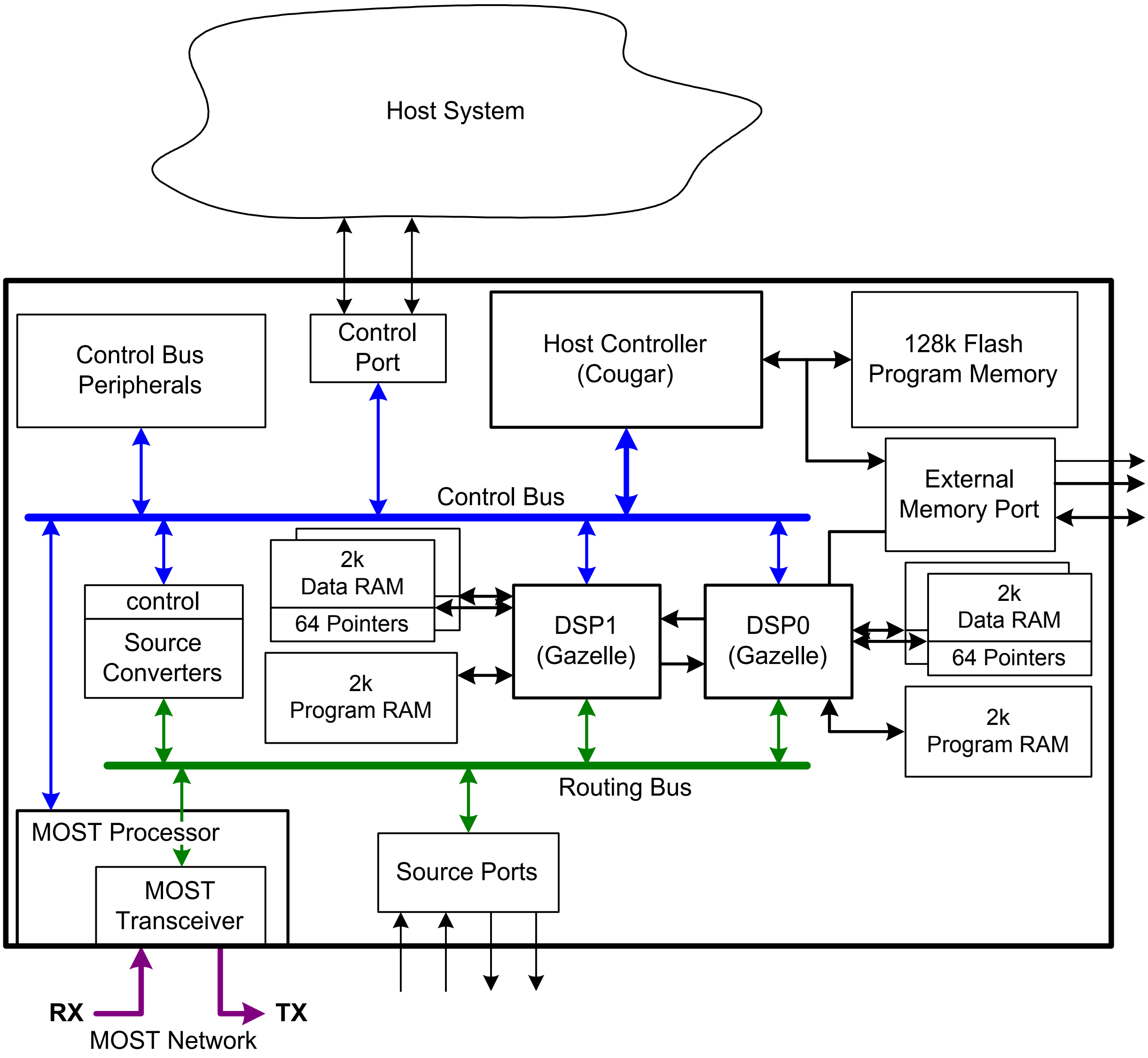


Figure 1-2: OS8805 Block Diagram

The MOST Processor manages real-time high-speed data traversing the Routing bus and includes a MOST Network transceiver. The high-speed data can be exchanged between the MOST Network and among the devices connected to the Routing bus; such as either DSP, the Source Ports, or the Source Converters (ADCs and DACs). The MOST Processor can transfer up to 15 high-fidelity stereo-audio streams onto or off of the MOST Network, per Fs period. The MOST Routing Table (MRT), programmed by the Host Controller, determines how the high-speed data is routed by the MOST Processor.

The two DSPs are Oasis SiliconSystem's *Gazelle* DSP cores and are identical, with the exception of extra peripherals on DSP0's I/O bus. Each DSP Program RAM must be loaded from the Host Controller. Each DSP can transfer up to 32-bytes per Fs onto and off of the Routing bus. The DSPs also have a FIFO port between them, allowing processing to be split between the DSPs without using Routing-bus bandwidth. Each DSP has an I/O bus with peripherals. Each DSP also has an Asynchronous Source Port with flexible clocking options. This port can be clocked asynchronously to the OS8805 and includes a timer that can reference the asynchronous clock to the internal chip timing and the MOST Network.

The Host Controller manages COM Ports that provide bi-directional communications to each DSP and the MOST Processor. Through the Control bus, the Host Controller also has access to the program controllers of each DSP. This enables the Host Controller to download programs to the DSPs.

Peripherals such as the **GPIO** pins, Clock Manager, a dual-USART, and the DC measurement ADC reside on the Control bus. The Control Port and Debug Port peripherals are also connected to the Control bus and allow the Host Controller to communicate with external devices. The Source Converters consist of ADCs and DACs that are connected to both the Control and Routing busses. The Host Controller can enable the Converters and adjust the volume, while the high-speed MOST Processor routes the Converter's data across the Routing bus. The Source Ports provide a gateway for high-speed data between the OS8805 and the external system.

Most digital pins that are not used for their primary function, may be configured as GPIO.

1.1 Host Controller

The Host Controller has multiple-byte instructions and a CISC architecture, which minimizes the amount of program code required to perform control applications. A byte/word-mode switch enables it to efficiently process byte and word variables without wasting data memory. The Controller operates at a maximum of 8.5 MIPS. Instructions can be one to four bytes long.

The Host Controller addresses 128k bytes of internal Flash or external Program memory, and 4k bytes of internal Data memory for variables. NetServices software is available from Oasis SiliconSystems which runs on the Host Controller and manages both the low- and high-level network protocol processing for the MOST Network.

The Host Controller is the main interface to the external system for control and configuration information. As illustrated in Figure 1-3, the external Host System communicates with the OS8805 through the Control Port. The Host Controller (Controller) reads the Control Port and reacts directly, or passes the information to one of the peripheral processors such as a DSP or the MOST Processor. The Controller also manages all the peripherals that reside on the Control bus.

In some systems, the OS8805 contains all the intelligence and the external Host System is only comprised of peripherals that are managed via the Control Port, in Master mode.

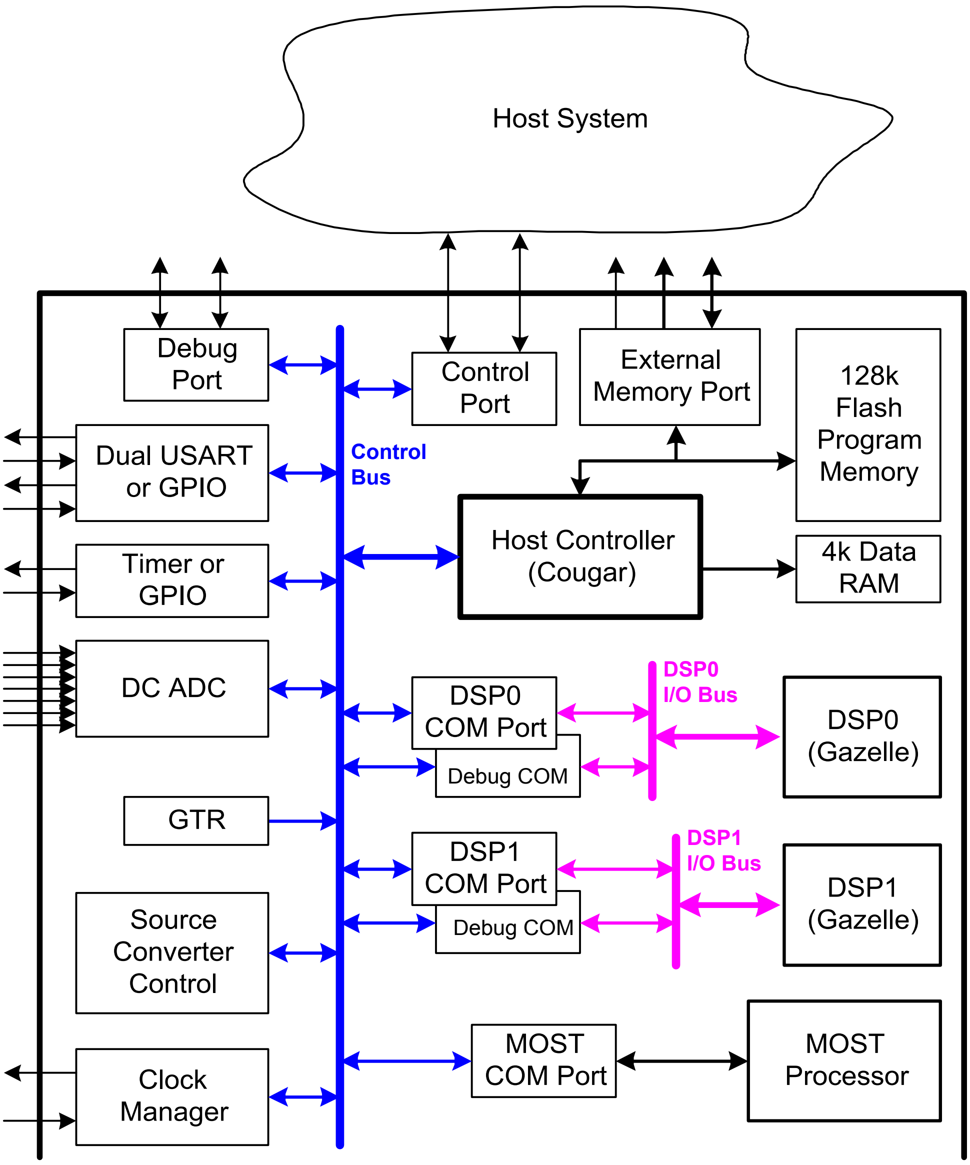


Figure 1-3: Host Controller and Control Bus

1.2 Control Bus and Peripherals

The Host Controller manages all Control bus peripherals. The Control Port can be a slave or a master and provides a serial interface between the Host Controller and the external system. The Control Port can operate in one of four formats: I^{2}C, Oasis-specific SPI, generic SPI format, or USART format. In both I^{2}C and Oasis-specific SPI (OSPI), the data exchange supports efficient transfer of blocks of data. The generic SPI (GSPI) format does not interpret the data, leaving the data format up to the user. The Control Port can be a master or slave in I^{2}C or GSPI formats, and only a slave in the OSPI format.

The Debug Port is a second serial interface between the Host Controller and the external system. Although listed as a "Debug" Port, the port is generic and has a unique interrupt vector. The Debug Port shares pins with three of the GPIO pins and supports the same four formats that the Control Port supports. The Debug Port can also be a master in the I^{2}C or GSPI formats only.

The Control bus also contains three COM Ports, which support inter-processor communications between the Host Controller and the two DSPs, as well as the MOST Processor. The MOST Processor communications utilize a format similar to the Control and Debug Ports in I^2C mode. The DSPs should be programmed to support the same format used by the MOST Processor to maximize code reuse. Through the MOST COM Port, the Host Controller can initialize the MOST Processor and transmit and receive messages across the MOST Network. The DSP COM Ports can be utilized in the same manner.

The Global Timer flag register is visible to all I/O busses and supplies timing flags, at up to 8xFs, for synchronized communications across the processors. The Global Timer also provides a periodic time interval, based on the sample frequency Fs.

The GPIO pins are multiplexed with the General Purpose Timer, Control Port and the Debug Port. The Timer is a 16-bit timer that counts up at a 64Fs rate and resets when the count reaches the modulo value. Two output-compare registers are provided that can toggle a pin when the Timer reaches a programmed value. Two capture registers save the timer value when a pin changes state and can generate an interrupt.

The Clock Manager peripheral contains a PLL and generates all the clocks needed by the OS8805. It also provides an external programmable clock (RMCK), to synchronize external system devices. When the OS8805 is configured as a MOST Network timing-slave device, the timing source is the MOST Network receive pin **RX**. When the OS8805 is configured as a MOST Network master, the timing source can be an external crystal, the external Source Port clock **SCK**, or the external Source Port data pin **SR0** when configured as an SPDIF input.

The DC Measurement ADC peripheral consists of an eight-to-one mux and an over-sampling ADC. The resolution is programmable from 5 to 12 bits. The conversion process takes approximately 1 ms to achieve 12-bit resolution, and 250 µs to achieve 10-bit resolution. The **POT[7:0]** pins support up to eight analog inputs.

1.3 MOST Processor and the Routing Bus

The MOST Processor interfaces to the MOST optical Network and manages the transfer of real-time data between on-chip resources. Included in the MOST Processor is a MOST Network transmitter and receiver. The MOST receiver, in conjunction with the clock manager, recovers the clock, decodes the data, and passes the information to the MOST Processor. The MOST Processor sends data to the transmitter which, encodes the data, and transmits it on the MOST Network.

The MOST transceiver uses the **TX** and **RX** pins for transmitting and receiving MOST optical network data. The MOST Processor, Routing bus, and Routing bus peripherals are illustrated in Figure 1-4.

The MOST Processor is a hard-coded RISC micro-controller, which is responsible for routing data internally as well as onto the optical Network. For high-speed source data, the MOST Processor acts like a cross-point switch using the MOST Routing Table (MRT) to connect sources and destinations across the Routing bus. For low speed data, such as control messages, the Processor communicates with the Host Controller through the COM port. This port enables the Host Controller to instruct the MOST Processor to send and receive messages and to route the appropriate source data between the MOST Network and the on-chip resources.

The MOST Network can communicate 60 bytes of synchronous data (Source data) at the audio sample rate (Fs). The MOST Processor can route the received data back onto the network, or (using the Routing bus) to the internal DSPs, Source Converters, or Source Ports. The transmitted MOST data can also come from the DSPs, Source Ports, and Source Converters. Each DSP can sink and source up to 32 bytes per Fs period (sixteen 16-bit channels in each direction). Each Source Port can sink and source up to 8 bytes per Fs period (four 16-bit channels).

The Source Converters consist of: the MPX ADC (16-bit samples at four times the audio sample rate), the stereo audio ADCs (two 16-bit samples at the audio sample rate), the microphone ADC (16-bit samples at ¼ the audio sample rate), and the audio DACs (four 16-bit samples at the audio sample rate).

The Global Timer peripheral provides inter-processor synchronization. The MOST Processor uses the flags in this register to transfer data across the Routing bus. The DSPs can also use this register to synchronize with the MOST Processor when transferring data across the MOST Routing port.

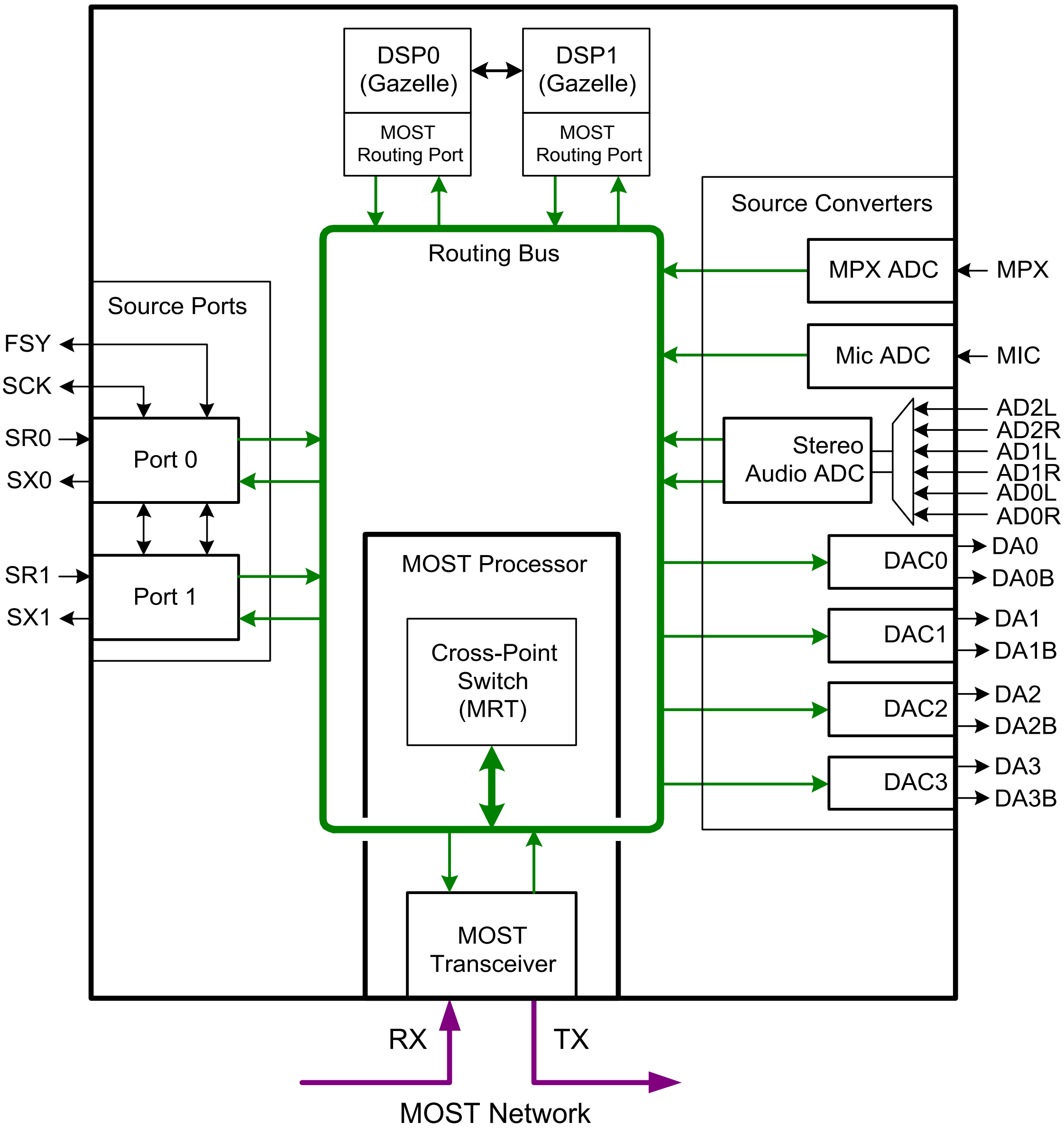


Figure 1-4: MOST Processor Overview

1.3.1 Source Data Ports

The Source Data Ports move high-speed data between the Routing bus and external devices. The external devices could be extra DACs, DSPs for further processing, or CD transports. Each Source Data Port can transfer up to eight bytes per sample frequency period. Therefore, a maximum of four 16-bit channels can be input to or sent out each Source Port.

Two serial-input and two serial-output ports communicate source data between internal and external components. The data format is programmable via the SDC1 register in the MOST Processor I/O space. Data is clocked through all four ports in the same data format with the same clock and frame sync.

Data is serially shifted into and out of the Source Ports through the serial receive **SR[1:0]** and the serial transmit **SX[1:0]** pins. Source port 0 can be configured to transmit and receive SPDIF data (**SX0** and **SR0**). If the OS8805 is configured as a MOST Network master, then the PLL can recover a clock from the **SR0** bit stream, when configured for SPDIF. If the chip is a timing-slave, which recovers the clock from the MOST Network **RX** pin, the SPDIF data and Source Port data must be entered synchronously to the Network **RX** pin. External synchronization can be achieved by using the OS8805 **RMCK** output as the master clock for the external system.

1.3.2 Source Data Converters

The Source Converters are high-speed ADCs and DACs that reside on the Routing bus. The Source Converters consists of four audio-band DACs, two audio-band ADCs, a 4x audio-bandwidth MPX ADC, and a ¼x audio bandwidth microphone ADC. The MPX, audio, and microphone ADCs are over-sampling delta-sigma converters which provide wide dynamic range and excellent linearity. An analog MUX, in front of the audio ADCs, selects one of three stereo input pairs. All ADCs have input gain stages and all DACs have output attenuators. The Host Controller controls the Audio ADC input MUX and the gain and attenuation settings through the Control bus. The Source Converter volume can also be set by either DSP.

The inputs to the ADCs are single-ended and must be AC coupled. The **MPX** pin is the input to the MPX ADC and the **MIC** pin is the input to the microphone ADC. The two audio ADC inputs are **AD0L-AD0R**, **AD1L-AD1R**, or **AD2L-AD2R** pins, depending on the state of the input multiplexer.

The ADCs output their data to the Routing bus and MOST Processor. The MOST Processor can then route the data to the MOST transmitter to go out on the Network, route the data to the DSPs for further processing, or route the data out the Source Data Port. Since the MPX ADC runs at 4xFs, the MOST Processor sends the 16-bit audio data to the MOST Network four times per frame. Since the Mic ADC runs at ¼xFs, the MOST Network sees the same copy of Mic data four times before new data is available.

The four DACs can be programmed for single-ended or differential outputs, which are available on the **DA[3:0]** and **DA[3:0]B** pins. The DACs are fed data across the Routing bus from the MOST Processor. The MOST Processor can get the data from the MOST Network receiver, the DSPs, or the Source Data Ports.

1.4 DSP Processors

The two DSPs have a RISC architecture, which provides the optimum cost performance trade-off for a wide variety of signal-processing applications. The RAM-based architecture allows software to be developed and downloaded for different system requirements. The DSP processor performs 18x14 bit operations with single-cycle instructions that produce 60 MIPs at 60 MHz. A combination hardware/software stack provides high-speed interrupt servicing for high-priority interrupts and die-area-efficient servicing for low priority interrupts. Special I/O instructions during high-priority interrupts provide low overhead I/O processing.

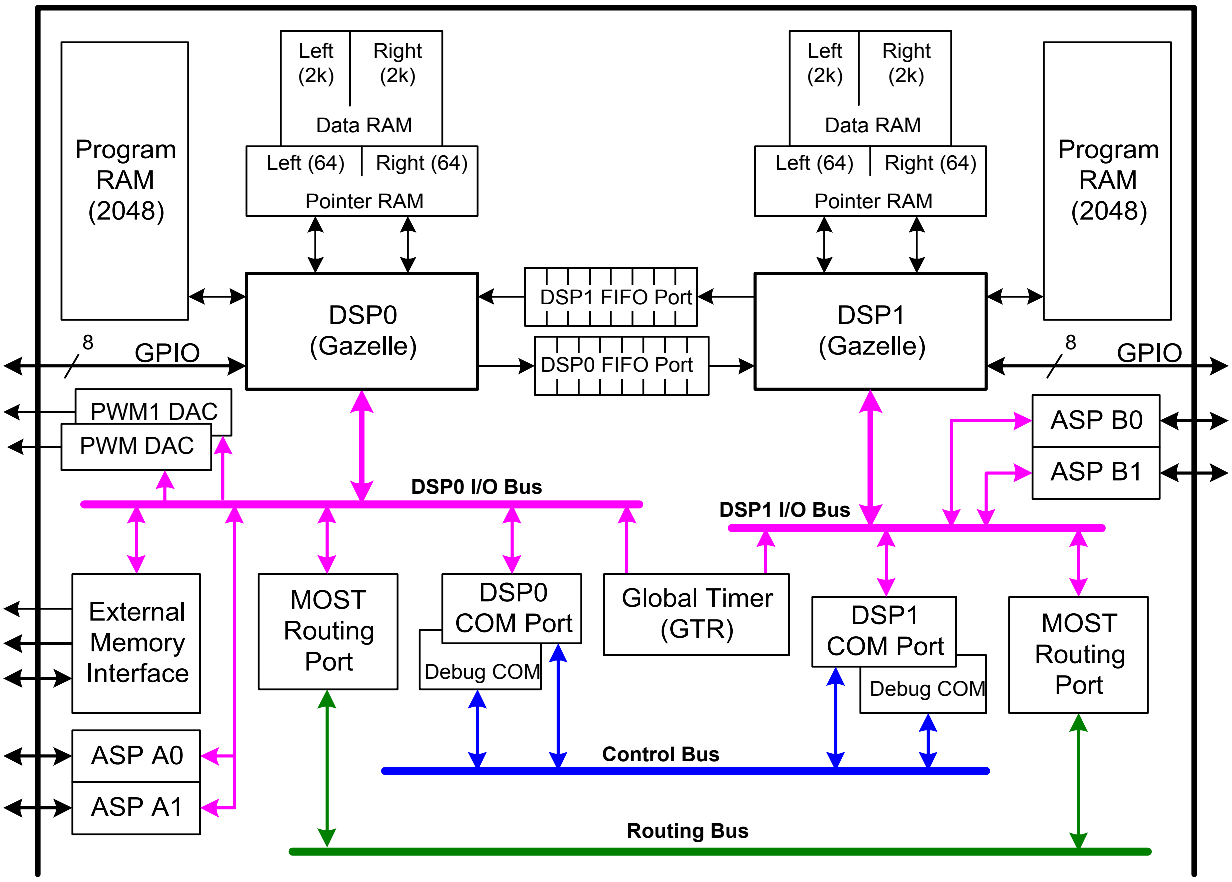


Figure 1-5: DSP Overview

The Host Controller connects to the DSP's program controller and has read/write access to the program counter, the program control register, and Program memory. This feature allows downloading programs as well as visibility of the DSP internal operation.

All the memories associated with the DSP processor are on-chip. The Program memory is 26 bits wide and 2048 locations deep. The Data memory section is divided into left and right sides with each side having a Pointer memory and a Vector memory. Each Pointer memory is 25 bits wide and both left and right Pointer memories are 64 locations deep. Left Vector memory is 14 bits wide and 2048 locations deep. Right Vector memory is 18 bits wide and 2048 locations deep.

The left and right Pointer memories store pointer values which consist of a 11-bit address, 6-bit update, and 8-bit modulo field. The accumulator consists of the low accumulator (14 bits), the high accumulator (18 bits), and the guard bits (4 bits).

Each DSP can control up to 8 GPIO pins that are multiplexed with the Asynchronous Source Ports. If the Async. Source Ports are not used, the DSP (or the Host Controller) can utilize the pins as GPIO.

The MOST Routing Port peripheral supports the transfer of data between the DSP and the Routing bus. The MOST Processor can be programmed to transfer data (in either direction) between the Source Ports, Source Converters, or the MOST Network and the DSPs.

Each DSP has an Asynchronous Source Port (ASP) peripheral with flexible clocking options. ASP A is associated with DSP0 and ASP B is associated with DSP1. Each DSP Asynchronous Source Port is divided in two Source Ports: 0 and 1. Each Port (0 or 1) has a serial data in, serial data out, framing clock, and bit clock. The Async. Source Ports can be clocked off an internal reference or externally clocked, independent of the internal chip clock. A timer connected to each Async. Source Port can reference the external clock to the internal system clock.

The DSPs also have a FIFO Port between the two DSPs supporting the off-loading of tasks from one DSP to the other. The FIFO Port can support simultaneous transfers of 8 words in each direction or up to 16 words in a single direction or time-division multiplexed (not at the same time) scheme.

The DSP I/O bus contains the Global Timer flag register that provides periodic flags at up to 8xFs. These flags support inter-processor synchronization since the same Global Timer is available to the Host Controller and the MOST Processor. As an example, the MOST Processor reads and writes the DSP's MOST Routing port eight times per sample period. The DSPs can use the Global timer **GTR.FS8** flag to update the MOST Routing port. This process synchronizes data flow between the DSPs and the MOST Processor.

DSP0 has access to the external memory port, if not used by the Host Controller. The external memory port supports direct connection of DRAM or SRAM memory chips. This expands DSP0's data memory up to 512k 16-bit words, when using DRAM, and 64k 16-bit words, when using SRAM. External data communication occurs through 16-bit I/O registers and includes modulo and post-incrementing addressing.

DSP0's I/O bus also includes two pulse-width-modulation (PWM) DAC peripherals for sub-woofer applications or low frequency volume control.

1.5 MOST NetServices API

For accelerating development of applications using the OS8805, Oasis SiliconSystems offers the MOST NetServices API. The MOST NetServices API provides software access to the MOST Network. All services that are relevant for MOST Network are available as a software library. This includes basic services like initialization, up to high-level communication tasks. MOST NetServices is modular and can be customized for the OS8805. The MOST NetServices API is implemented in ANSI C, which can be adapted to individual requirements through configuration files.

The MOST NetServices API is organized into two layers: Basic Services (Layer 1) and the Applications Socket (Layer 2). The Basic Services provides low-level services such as network initialization, Control message management through the COM Port, Source Port configuration, and synchronous channel allocation on the network

The Applications Socket (Layer 2) operates on top of Layer 1 and provides a command interpreter and the NetBlock function required on all network devices. The command interpreter provides a simple API for developing new functions within a node. It also supports the MOST Specification Notification Services and functional addressing.

With respect to the ISO communications model, the OS8805 chip, with its on-chip Host Controller supports all network layers except the Physical layer, with MOST Network transceiver containing the Data-Link layer and the Host Controller managing all the upper layers. MOST NetServices Layer 1 supports the Network layer through the Session layer. MOST NetServices Layer 2 supports the Presentation and part of the Application layer. The ISO network model and MOST NetServices is depicted in Figure 1-1.

More information on MOST NetServices is available on the Oasis SiliconSystems web page:

http://www.oasis.com

2 Host Controller and Control Bus Peripherals

The Host Controller (Controller) manages data movement across the Control bus. The Host Controller also handles internal and external communication and control between on-chip resources and the Control Port. When power is initially applied to the OS8805, the **CMCS.PD** bit is set, placing the OS8805 in power-down mode. A chip reset, $\overline{\textbf{RST}}$, or a transition on **RX** or **GPA0-GPA2** is needed to clear **CMCS.PD** and place the Host Controller in normal operation. The Debug Port provides a second serial port that provides on-chip debug capability, with the appropriate debug code programmed into the Host Controller. Although listed as the Debug Port, it can be programmed to serve as an additional control port or as GPIOs. The Controller is also responsible for program downloading into the DSPs' memories. Figure 2-1 highlights the Control Bus section of the OS8805.

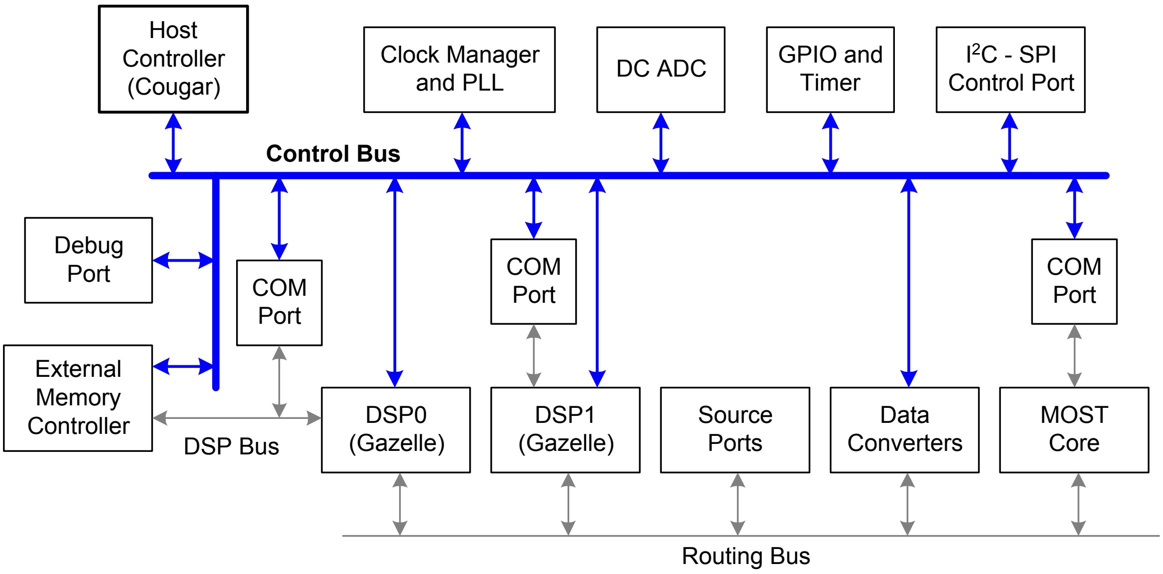


Figure 2-1: Control Bus

2.1 Architecture

The Host Controller conforms to the Oasis SiliconSystems Cougar 16-bit architecture and is a general purpose CISC architecture processor. The multiple-byte instructions and CISC architecture is optimized to reduce Program memory requirement of control applications. A byte/word mode switch enables it to efficiently process byte and word variables allowing optimal use of data memory. Special I/O instructions and a maximum performance of 8.5 MIPS (2 cycles/instruction at 384Fs) enable it to properly process real-time applications. For programming information on the Controller, see the *Cougar User's Manual*. The OS8805 has 128k bytes of on-chip Flash Program memory and 4k bytes of on-chip Data memory.

The Cougar Host Controller instructions can be one to four bytes in length. The first byte consists of the opcode, and indicates the addressing mode for the instruction. The following bytes provide an 8- to 16-bit direct memory address or immediate data, or up to an 17-bit jump address. The Controller can access one byte of Program memory every two clock cycles, and operates at a clock frequency of 384×Fs.

The addressing mode determines the number of bytes in the instruction. For a two-byte instruction, the second byte contains 8-bit immediate data or an 8-bit address. For a three-byte instruction, the second and third bytes together form 16-bit immediate data or a 16-bit address. Four-byte instructions are used in jump and jump to subroutine instructions as absolute addresses that span the entire Program memory space.

When an instruction is executed, the first byte of the instruction (the opcode) is loaded into the Instruction Register (IR). The lower three or four bits of the byte indicate the addressing mode to be used by the instruction. When an extended jump is fetched, its fourth byte is written into the M2 register.

A three-byte instruction specifies the Data memory address (far-direct addressing mode), while a two-byte instruction specifies only the low 8 bits of the Data memory address (near-direct addressing mode). In near-direct addressing mode, the high bits of the Data memory address are specified by the Address Page (AP) bits in SR.

Jump instructions can be short, long, or extended. Short jumps are two-byte instructions. Long jumps are three-byte instructions. Extended jumps are four-byte instructions. With the exception of short subroutine jumps, a short jump instruction provides an 8-bit two's complement number that is added to the current Program Counter value. This provides PC-relative addressing to +127 and -128 Program memory locations. Short subroutine jumps are absolute jumps when the upper jump address bits A[16:8] are 0x000 (subroutines on first Program memory page). A long jump instruction specifies a 16-bit Program memory address and is a relative address. An extended jump instruction is the only four-byte instruction and specifies an absolute Program memory address.

Two- and three-byte instructions provide 8- and 16-bit immediate data, respectively. When immediate data is 16 bits, the Arithmetic Logic Unit (ALU) performs 16-bit operations. Eight-bit immediate data is zero-extended during arithmetic operations. During logical operations and moves, the 8-bit ALU result is stored in the low byte of the destination. The high byte is unaffected by the operation, except when ACC is the destination and a carry from an arithmetic operation occurs. Figure 2-2 shows how the opcode is constructed.

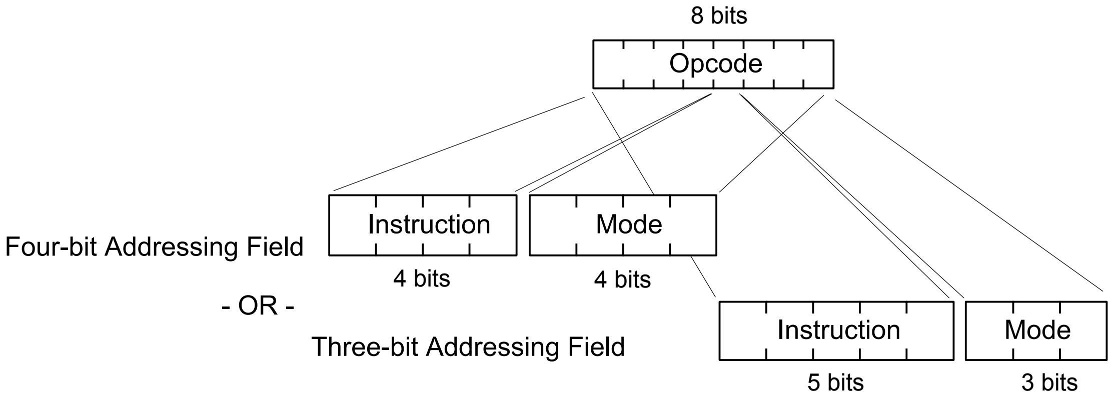


Figure 2-2: Host-Controller Opcode Architecture

Single-operand instructions use the 3-bit address field and dual-operand instructions use the 4-bit address field. For more information, see Section 2, in the *Cougar User's Manual*. Table 2-1 lists the instructions that use the 3-bit addressing field. The Bit Move instruction also uses a 3-bit address field; however, it doesn't support Register addressing, only Data memory direct or indirect.

Opcode (bits 7:0)	Instruction
1000 0yyy	Increment
1000 1yyy	Test
1001 0yyy	Shift right
1001 1yyy	Exchange
1010 0yyy	Decrement
1011 0yyy	Shift left

Table 2-1: Single-Operand Instructions

In dual-operand instructions, the 4-bit addressing modes is used, where the added bit specifies whether the source or the destination is the ACC register. In dual-operand instructions, one of the operands must be the ACC register. Table 2-2 lists the instructions that use the 4-bit addressing field.

Opcode (bits 7:0)	Instruction
0000 xxxx	Add
0001 xxxx	Add with carry
0010 xxxx	Subtract
0011 xxxx	Subtract with carry
0100 xxxx	Exclusive or
0101 xxxx	Or
0110 xxxx	And
0111 xxxx	Move

Table 2-2: Dual-Operand Instructions

The Host Controller communicates with the DSPs and peripherals through the Control bus. Each processor and peripheral has registers mapped into the I/O space of the Controller. The processors and many of the peripherals can interrupt the Controller when service is required.

The Controller's Program memory contains programs that are downloaded into the Program RAM of the DSPs. The external Program memory can be Flash, EPROM, or SRAM (generally used for emulators in a debug environment).

2.1.1 Program Memory

The Host Controller can operate from internal or external Program memory. The external Program memory enable, **RGEN.$\overline{\text{XME}}$** along with the **MMPC.XMQ** bits, select between the two options (see the description of the RGEN and MMPC registers for more details). Figure 2-3 illustrates the memory interface connected to the Controller's program memory interface. The $\overline{\text{XME}}/\overline{\text{PCS}}$ pin is sampled at the rising edge of $\overline{\text{RST}}$ or power-up and the value is stored in the **RGEN.$\overline{\text{XME}}$** bit. Changing the **RGEN.$\overline{\text{XME}}$** bit causes an internal reset.

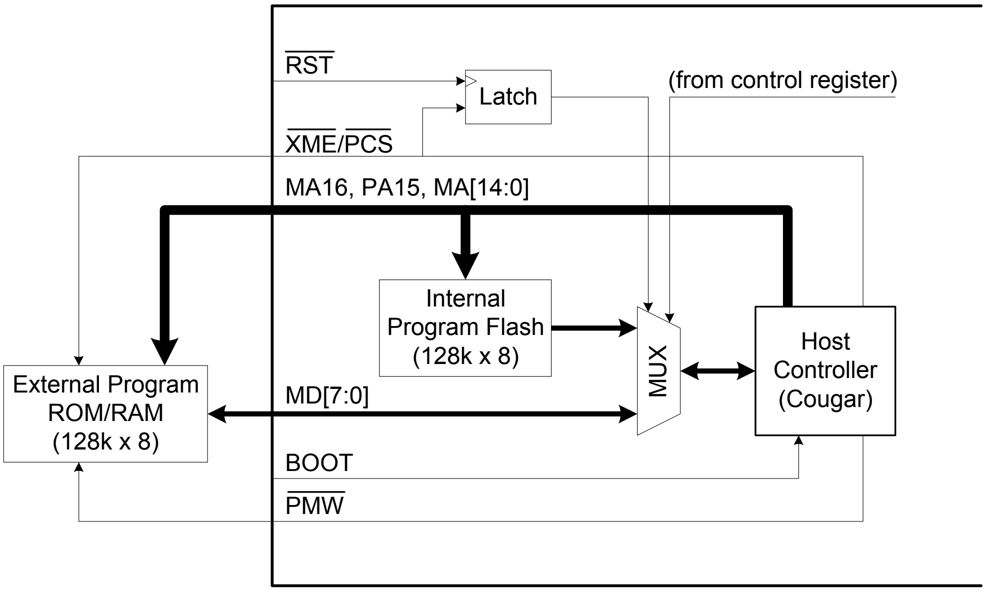


Figure 2-3: Host Controller External Program Memory Diagram

37h MMPC Mode Control Register Host

Bit	Label	Description	Default
15..12	ID[3:0]	OS8805 part ID (read only)	0010
11..8	REV[3:0]	OS8805 part revision: D (read only)	0100
7	rsvd	Reserved, Write to 0	0
6	RFS1	MOST Processor High Frequency Select - bit 1	0
5	RSTD	Host Controller Reset Disable	0
4	rsvd	Reserved, Write to 0	0
3	$\overline{\text{XMQ}}$	External program memory interface quiet mode	0
2	rsvd	Reserved, Write to 0	0
1	PCSC	$\overline{\text{PCS}}$ timing control	1
0	rsvd	Reserved, Write to 0	0

Table 2-3: MMPC Register

ID[3:0] Part ID (read-only)
0001 – OS8804. See the *OS8804 Data Sheet*.
0010 - OS8805. This Data Sheet.

REV[3:0] OS8805 Revision (read only)
0110 - F.
1000 - H. This Data Sheet reflects revision H silicon.

RFS1 MOST Processor High Frequency Select. When set, operates the MOST Processor at a higher speed. **RFS1** must be set when Packet Data is supported (**bNC.APREN** set).

RSTD Host Controller Reset Disable.
0 - The reset vector is loaded in the program counter when the `RESET` instruction is executed or when **RGEN.$\overline{\text{XME}}$** is changed. All Host Controller registers are reset just like all the peripheral circuits/registers.
1 - The Host Controller is NOT reset when a `RESET` instruction is executed or when **RGEN.$\overline{\text{XME}}$** is changed. In addition all Host Controller internal registers are NOT reset: PC (program counter) SPC, DSPC, SR IFL, IER, SP, SPCH, DSPCH, and PGMP. Peripheral registers/circuits are still reset.

$\overline{\text{XMQ}}$ External Program Memory interface Quiet. This bit, in combination with the **RGEN.$\overline{\text{XME}}$** bit, define how the external memory interface pins function:

RGEN.$\overline{\text{XME}}$	MMPC.$\overline{\text{XMQ}}$	External Memory Interface		
		DSP0	Host Cont. Read	Host Cont. Write
1	0	* Data memory	† internal Flash	‡ internal Flash
1	1	none	† internal Flash	† external memory
0	x	none	† external memory	† external memory

* DSP Data memory is only enabled when **GCTL.EDMEN** is set. Otherwise, the pins are EGPIO.
† Read or written a byte at a time, through ACCL (ACC[7:0]).
‡ Written a word at a time. ACCH to the lower address and ACCL to the higher address.

PCSC Controls the phase timing of the Host Controller clocks. Also affects the $\overline{\text{PCS}}$ pin timing when the external memory interface is enabled for the Host Controller. When **PCSC** is clear, the $\overline{\text{PCS}}$ high and low times are equivalent. When **PCSC** is set, the $\overline{\text{PCS}}$ low time is longer that the high time. See the *External Program Memory Interface* tables in the *Electrical Characteristics* section for more details.

Internal Program memory is 128k bytes of Flash programmable memory. If all the DSPs programs are to be downloaded from the Host Controller, the Controller's Program memory must contain code to download these programs, along with the DSP's main code and the Controller's main code. The Host Controller executes out of internal Program memory whenever the **RGEN.$\overline{\text{XME}}$** bit is set. The initial state of **RGEN.$\overline{\text{XME}}$** is captured from the **$\overline{\text{XME}}$/$\overline{\text{PCS}}$** pin (same polarity) at the rising edge of **$\overline{\text{RST}}$** or when power is initially applied to the device.

If external Program memory is enabled, the 17-bit address is output through the Memory Address pins **MA[16:0]**. The Memory Data pins are **MD[7:0]**. The Program Memory Chip Select (**$\overline{\text{PCS}}$**) provides the read timing for the external SRAM, EEPROM or Flash. If the Host Controller uses internal memory, then DSP0 can use the external bus for data memory (configured through the XMC and GCTL registers).

2.1.1.1 Flash Memory

The 128k Flash memory is divided into 16 blocks and 128 partitions. Each block can be "protected" from direct reading and writing. Flash Block 16 is further divided into half-partition (512 bytes) segments for individual protection from reading and writing. Each Flash memory partition (1k bytes) can be erased and then written. Flash partitions are erased to all ones. Flash memory can be programmed on a word-by-word basis, through the ACC register, where ACCH (ACC[15:8]) is written to the lower address and ACCL (ACC[7:0]) is written to the upper address. Once a zero bit is programmed into a word, the only way to change it to a one, is to erase the entire partition.

A 256-byte separate Flash partition, designated *Info Block*, is used by the on-chip Flash Handler to manage the programming of the Flash memory. The Flash Handler resides in the lowest 8k Flash block, and the *Info Block* is addressed by setting the **PGMP.A17** bit and clearing the **PGMP.A16** bit. Then AR1 can indirectly access the lower 256 memory addresses for the *Info Block*. For more information on the Flash Handler, see the *OS8805 Flash Handler User's Manual*. Since the Flash Handler resided in the lowest 8k Flash memory block, all the interrupt vectors not used by the Flash Handler (all except **BOOT**), are redirected as illustrated in Figure 2-4.

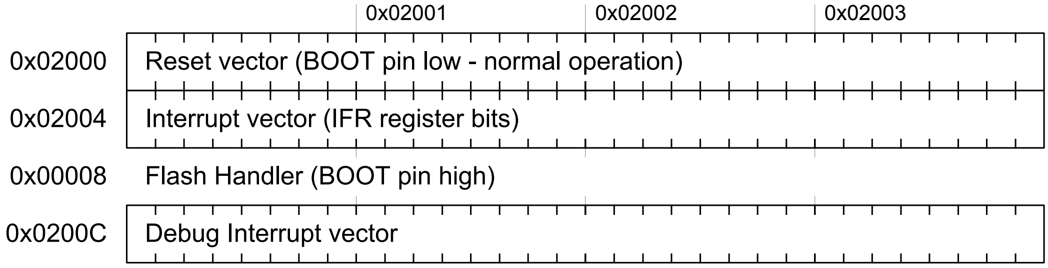

0x02001
0x02002
0x02003
0x02000
Reset vector (BOOT pin low - normal operation)
0x02004
Interrupt vector (IFR register bits)
0x00008
Flash Handler (BOOT pin high)
0x0200C
Debug Interrupt vector

Figure 2-4: User Interrupt Vector Table

Internal Flash memory can only be written (a word at a time) when **RGEN.$\overline{\text{XME}}$** is set and **MMPC.$\overline{\text{XMQ}}$** is clear (DSP data memory selected for external memory port), and Flash protection for the particular memory location is disabled.

When Programming internal Flash, data is written a word at a time; therefore, the LSB in AR1 must be 0. ACCH is written to the lower byte of memory and ACCL is written to the upper byte. This byte-order is opposite to words written into Data memory.

5Bh FMC Flash Memory Control Host

Bit	Label	Description	Default
15..	rsvd	Reserved. Write to 0.	00h, 0
6, 5	FCP[1:0]	Flash Clock Prescaler	00
4, 3	FPT[1:0]	Flash Program Time	00
1	PER	Partition Erase. 0 = write Program memory, 1 = erase partition	0
0	FPD	Flash Powerdown. 0 = enabled, 1 = standby mode	0

Table 2-4: FMC Register

FCP[1:0] Flash Clock Prescaler. Determines the prescaler for the Flash programming time (delay counter clock).
00 - Divide by 1.
01 - Divide by 2.
10 - Divide by 3.
11 - Divide by 4.

FPT[1:0] Flash Program Time. Programming Cycle time. The Flash programming time should be 20 µs minimum. See Table 2-5 for programming time details.
00 - 384 cycles
01 - 272 cycles
10 - 156 cycles
11 - 116 cycles

FCP[1:0]	FPT[1:0] (Fs = 44.1 kHz)				FPT[1:0] (Fs = 48 kHz)			
	384	272	156	116	384	272	156	116
1	22.6	16.0	9.2	6.8	20.8	14.6	8.5	6.3
2	45.3	32.1	18.4	13.7	41.7	29.5	16.9	12.6
3	68.0	48.2	27.6	20.5	62.5	44.3	25.4	18.9
4	90.7	64.2	36.8	27.4	83.3	59.0	33.9	25.2

Table 2-5: Flash Memory Programming Times (in µs)

PER Partition Erase. When clear, Flash Program memory is written. When set, the Flash memory partition that is selected during a Program memory write instruction, `(rom)*ar1 = acc;`, is erased (data used in write instruction is ignored). Flash protection for the particular location or partition must be disabled in order to write to or erase internal Flash memory.

FPD Flash Powerdown. When clear, Flash memory is powered up and operating normally. When set, the Flash memory is powered down and not accessible. Since this disables further instructions (halts Host Controller operation), it should only be set to minimize power when the Host Controller is operating out of external Program memory.

The Flash programming time must be kept greater than or equal to 20 µs and less than or equal to 40 µs. When the PLL is unlocked, the sample frequency can drift down to its minimum frequency. However, if the PLL unlocks during a Program memory write, the duration of the write is short enough that the PLL frequency will not drift too far from its nominal locked value. Therefore, software that writes to Program memory must check the state of the PLL and, if unlocked (**CMCS.LOCK** clear), software must either not initiate any more Program memory writes, or shorten the programming time through the **FMC.FPT[1:0]** bits.

5Ch FPBK Flash Protection 8K Blocks Host

Bit	Label	Description	Default
15	FB15	Address Range 0x1C000 - 0x1DFFF	0
14	FB14	Address Range 0x1A000 - 0x1BFFF	0
13	FB13	Address Range 0x18000 - 0x19FFF	0
12	FB12	Address Range 0x16000 - 0x17FFF	0
11	FB11	Address Range 0x14000 - 0x15FFF	0
10	FB10	Address Range 0x12000 - 0x13FFF	0
9	FB9	Address Range 0x10000 - 0x11FFF	0
8	FB8	Address Range 0x0E000 - 0x0FFFF	0
7	FB7	Address Range 0x0C000 - 0x0DFFF	0
6	FB6	Address Range 0x0A000 - 0x0BFFF	0
5	FB5	Address Range 0x08000 - 0x09FFF	0
4	FB4	Address Range 0x06000 - 0x07FFF	0
3	FB3	Address Range 0x04000 - 0x05FFF	0
2	FB2	Address Range 0x02000 - 0x03FFF	0
1	FB1	Address Range 0x00000 - 0x01FFF	0
0	FBIB	Address Range 0x20000 - 0x200FF (Info Block)	0

Table 2-6: FPBK Register

FB[15:0] Flash Block protection. When a bit is set, the 8K Flash memory block cannot be read using the instruction `acc = (rom)*ar1;` or written/erased using `(rom)*ar1 = acc;`. The value read back will be indeterminate. When **FBn** is clear, the Program read instruction reads a byte constant from Program memory, and the Program write instruction either writes a word from ACC or erases a partition (based on **FMC.PER**). The Flash Handler resides in **FB1**.

FBIB Info Block protection. When set, the Flash 256-byte *Info Block* cannot be read, written, or erased. The value read back will be indeterminate. When **FBIB** is clear, the Program read instruction reads a byte constant from the *Info Block*, and the Program write instruction either writes a word from ACC or erases a partition.

5Dh FPB16 Flash Protection - Flash Block 16 Host

Bit	Label	Description	Default
15	FP7B	Address Range 0x1FE00 - 0x1FFFF; Flash Block 16, partition 7b	0
14	FP7A	Address Range 0x1FC00 - 0x1FDFF; Flash Block 16, partition 7a	0
13	FP6B	Address Range 0x1FA00 - 0x1FBFF; Flash Block 16, partition 6b	0
12	FP6A	Address Range 0x1F800 - 0x1F9FF; Flash Block 16, partition 6a	0
11	FP5B	Address Range 0x1F600 - 0x1F7FF; Flash Block 16, partition 5b	0
10	FP5A	Address Range 0x1F400 - 0x1F5FF; Flash Block 16, partition 5a	0
9	FP4B	Address Range 0x1F200 - 0x1F3FF; Flash Block 16, partition 4b	0
8	FP4A	Address Range 0x1F000 - 0x1F1FF; Flash Block 16, partition 4a	0
7	FP3B	Address Range 0x1EE00 - 0x1EFFF; Flash Block 16, partition 3b	0
6	FP3A	Address Range 0x1EC00 - 0x1EDFF; Flash Block 16, partition 3a	0
5	FP2B	Address Range 0x1EA00 - 0x1EBFF; Flash Block 16, partition 2b	0
4	FP2A	Address Range 0x1E800 - 0x1E9FF; Flash Block 16, partition 2a	0
3	FP1B	Address Range 0x1E600 - 0x1E7FF; Flash Block 16, partition 1b	0
2	FP1A	Address Range 0x1E400 - 0x1E5FF; Flash Block 16, partition 1a	0
1	FP0B	Address Range 0x1E200 - 0x1E3FF; Flash Block 16, partition 0b	0
0	FP0A	Address Range 0x1E000 - 0x1E1FF; Flash Block 16, partition 0a	0

Table 2-7: FPB16 Register

FP[7:0]A Flash partition protection, lower-half of a partition. When set, the 512 byte Flash memory segment cannot be read using the instruction `acc = (rom)*ar1;` or written/erased using `(rom)*ar1 = acc;`. The value read back will be indeterminate. When **FPnA** is clear, the program read instruction reads a byte constant from Program memory. If Program memory in **FPnA** needs to be erase to re-program, **FPnB** will also be erased. Only whole partitions can be erased. If the write instruction is used to erase a partition and the address points to an unprotected memory location in **FPnA**, the **FPnB** portion will also be erased regardless of the **FPnB** setting.

FP[7:0]B Flash partition protection, upper-half of a partition. When set, the 512 byte Flash memory segment cannot be read using the instruction `acc = (rom)*ar1;` or written/erased using `(rom)*ar1 = acc;`. The value read back will be indeterminate. When **FPnB** is clear, the program read instruction reads a byte constant from Program memory. If Program memory in **FPnB** needs to be erase to re-program, **FPnA** will also be erased. Only whole partitions can be erased. If the write instruction is used to erase a partition and the address points to an unprotected memory location in **FPnB**, the **FPnA** portion will also be erased regardless of the **FPnA** setting.

Bits in FPBK and FPB16 are protected from arbitrary writes by only allowing certain bits to be updated from certain Program memory locations. Therefore, an IO Write instruction from the following PC locations will allow the listed bits to be updated. **FB1** contains the Flash Handler and is always protected.

PC location for IO Write Instruction		Register bits that can be changed	
Block	**PC Address**	**Register**	**Bits**
FB2	0x03862	FPBK	FB[15:2]
	0x03867	FPB16	FP[7:0]A, FP[7:0]B (all bits)
FB3	0x048DF	FPBK	FB[15:3]
	0x048E4	FPB16	FP[5:0]A, FP[5:0]B
FB5	0x0996E	FPBK	FB[15:5]
	0x09973	FPB16	FP[3:0]A, FP[3:0]B

Table 2-8: IO Write Locations for FPBK and FPB16 Bits

2.1.2 Data Memory

The Host Controller has 4096 bytes of internal Data memory. Byte and word variables can be stored in Data memory, the size indicated in the **SR.W** status flag (controlled via the `SB` and `SW` instructions). Up to 2048 sixteen-bit variables or 4096 byte variables can be stored.

2.1.3 Program and Interrupt Controller

The program controller maintains the Program Counter (PC) for sequential instruction fetching, change of flow instructions, and interrupts. A Shadow Program Counter (SPC and SPCH) speeds subroutine calls, and a dedicated Debug Shadow Program Counter (DSPC and DSPCH) handles the debug interrupt.

00h SPC Shadow Program Counter Host

Bit	Label	Description	Default
15..0	SPC[15:0]	Lower 16 bits of the Shadow Program Counter. Used during interrupts and subroutine calls	0000h

Table 2-9: SPC Register

58h SPCH Shadow Program Counter - High Host

Bit	Label	Description	Default
15..	rsvd	Reserved. Write to 0.	000h, 0
0	SPC16	Upper bit of the Shadow Program Counter. Used during interrupts and subroutine calls	0

Table 2-10: SPCH Register

01h DSPC Debug Shadow Program Counter Host

Bit	Label	Description	Default
15..0	DSPC[15:0]	Lower 16 bits of the Debug Shadow Program Counter. Used during the Debug interrupt. Writing DSPC causes the next `RET`/`RETI` instruction to use the DSPC/DSPCH registers to retrieve the return address.	0000h

Table 2-11: DSPC Register

59h DSPCH Debug Shadow Program Counter - High Host

Bit	Label	Description	Default
15..	rsvd	Reserved. Write to 0.	000h, 0
0	DSPC16	Upper bit of the Debug Shadow Program Counter. Used during the Debug interrupt.	0

Table 2-12: DSPCH Register

02h SR† Status Register Host

Bit	Label	Description	Default
15..12	rsvd	Reserved. Write to 0.	0000
11..8	AP[3:0]	Address Page bits for direct Data memory access instructions	0000
7	AC	Auxiliary Carry flag	*
6, 5	rsvd	Reserved, Write to 0	00
4	GIE	Global Interrupt Enable Status (read-only)	0
3	W	Byte/word mode flag. Set is Word mode, clear is Byte mode	0
2	N	Negative flag	0
1	C	Carry flag	0
0	Z	Zero flag	0

† The SR register must not be written when **GIE** is set. **GIE** is disabled by executing the `DISABLE` instruction.

Table 2-13: SR Register

03h IFL† Interrupt Flag Register Host

Bit	Label	Description	Default
15..12	rsvd	Reserved, Write to 0	0000
11	IPLL	PLL out of lock interrupt (**CMCS.LOCK** going low – edge sensitive)	0
10	rsvd	Reserved, Write to 0	0
9	IDSP1	DSP1 COM port (rising edge of **D1CS.INT** bit)	0
8	IDSP0	DSP0 COM port (rising edge of **D0CS.INT** bit)	0
7	IMST	MOST COM port (rising edge of **RCS.INT** bit)	0
6	ITMR0	Timer - CMP0 equals timer value (edge sensitive)	0
5	ICP	Control Port data ready – sending and receiving data (level sensitive)	0
4	IADC	DC Measurement ADC ready (edge sensitive)	0

† The IFL register should not be written when **SR.GIE** is set; otherwise edge-sensitive interrupts could be lost.
* If programmed as level sensitive (**GPC.LVn**), then **IDn** bit is read-only – cannot clear by writing a 0 to it.

Table 2-14: IFL Register

3	ID3*	**GPD3** pin, pos/neg edge/level triggered (GPC)	0
2	ID2*	**GPD2** pin, pos/neg edge/level triggered (GPC)	0
1	ID1*	**GPD1** pin, pos/neg edge/level triggered (GPC), CAP1 saves timer value	0
0	ID0*	**GPD0** pin, pos/neg edge/level triggered (GPC), CAP0 saves timer value	0

† The IFL register should not be written when **SR.GIE** is set; otherwise edge-sensitive interrupts could be lost.
* If programmed as level sensitive (**GPC.LVn**), then **IDn** bit is read-only – cannot clear by writing a 0 to it.

Table 2-14: IFL Register (Continued)

04h IER Interrupt Enable Register Host

Bit	Label	Description	Default
15	DRI	Disable `RETI` Interrupt enable. When set, `RETI` instructions do NOT set **SR.GIE**. When clear, executing `RETI` causes **SR.GIE** to be set.	0
14, 13	rsvd	Reserved, Write to 0	00
12	DIEN	Debug Port Interrupt Enable. 0 is disabled, 1 is enabled.	0
11	IEPLL	When set, enables **IFL.IPLL** Interrupt (PLL out of lock)	0
10	rsvd	Reserved, Write to 0	0
9	IEDSP1	When set, enables **IFL.IDSP1** Interrupt (DSP1 COM port)	0
8	IEDSP0	When set, enables **IFL.IDSP0** Interrupt (DSP0 COM port)	0
7	IEMST	When set, enables **IFL.IMST** Interrupt (MOST COM port **RCS.INT** only)	0
6	IETMR0	When set, enables **IFL.ITMR** Interrupt (Timer from TMR0 compare)	0
5	IECP	When set, enables **IFL.ICP** Interrupt (Control Port)	0
4	IEADC	When set, enables **IFL.IADC** Interrupt (DC measurement ADC)	0
3	IED3	When set, enables **IFL.ID3** Interrupt (**GPD3** pin)	0
2	IED2	When set, enables **IFL.ID2** Interrupt (**GPD2** pin)	0
1	IED1	When set, enables **IFL.ID1** Interrupt (**GPD1** pin)	0
0	IED0	When set, enables **IFL.ID0** Interrupt (**GPD0** pin)	0

Table 2-15: IER Register

05h SP Stack Pointer Host

Bit	Label	Description	Default
15..0	D[15:0]	Points to the next entry of the software stack	0000h

Table 2-16: SP Register

Program data is byte-wide, where an instruction is made up of one, two, three, or four program data bytes. The first byte of the instruction always contains the op-code and indicates the addressing mode for the instruction. For two-byte instructions, the second byte contains either immediate data or an 8-bit direct address. For three-byte instructions, the second and third bytes contain either 16-bit immediate data or a 16-bit direct address. The only four-byte instructions are absolute jumps, either conditional or bit-set/clear, that cover the entire Program memory space.

The interrupt controller handles eleven interrupt sources (IFL register bits), the Debug Port interrupt, and reset (start-up). Interrupts divert program flow to an interrupt vector address. The return-from-interrupt instruction returns execution back to the interrupted instruction, and execution continues as expected. The hardware registers associate with the interrupt controller are the interrupt flag register IFL, interrupt enable register IER, and the stack pointer SP.

The reset vector can be directed to two different locations, depending on the status of the **BOOT** pin. When the **BOOT** pin is high, the master reset initializes the host program counter to 0x00008, which is used by the Flash Handler. When the **BOOT** pin is low, the master reset initializes the host program counter to 0x00000 to start normal program execution. The Host Controller's Vector table is illustrated in Figure 2-5. The BOOT interrupt is used by the on-chip Flash Handler software which resides in the lowest 8k Flash memory block. Since the interrupt vectors are located in the Flash Handler's memory space, the other three interrupt vectors (not the BOOT vector) are redirected to the second 8k Flash memory block, as illustrated in Figure 2-4.

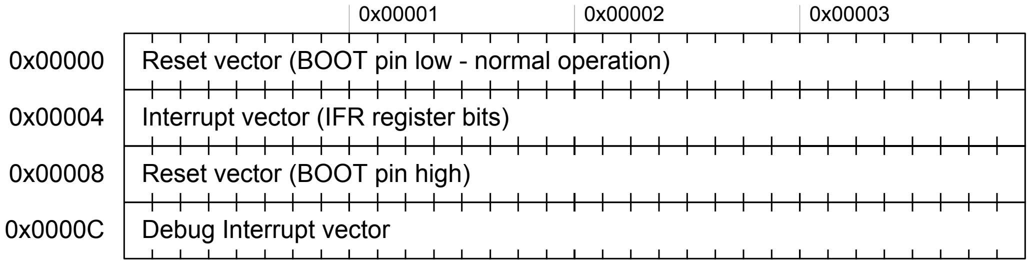

0x00001
0x00002
0x00003
0x00000
Reset vector (BOOT pin low - normal operation)
0x00004
Interrupt vector (IFR register bits)
0x00008
Reset vector (BOOT pin high)
0x0000C
Debug Interrupt vector

Figure 2-5: Host Controller Hardware Vector Table

The Watchdog timer powers up enabled. Therefore, if its counter reaches the terminal count, it will reset the chip. If it is not used, it must be explicitly disabled by setting **RGEN.WDD**.

The Controller has 11 different interrupt sources generated from the two DSPs, the GPIO pins (configured as interrupts), the timer, the DC measurement ADC, and the Control Port. All IFL interrupts use the same interrupt vector location at Program memory address 0x00004. For interrupts to occur, the Global Interrupt Enable status bit **SR.GIE** must be high (using the `ENABLE` instruction) and the appropriate bit in IER must be set. The Interrupt Flag register IFL identifies the source of the interrupt. To clear an edge-sensitive interrupt, a zero must be written to the respective bit that is set in the IFL register. Level-sensitive interrupt bits in IFL are read-only and cannot be cleared through the IFL register. The Interrupt Enable register IER provides independent enables for each interrupt. Other than the Debug Port interrupt, which has the highest priority, the other 11 interrupts have the same priority. The interrupt priority and servicing must be handled through software. An interrupt can also be generated by writing a one to the corresponding bit in the interrupt flag register IFL, if the interrupt is enabled.

If an edge-trigger interrupt condition occurs while **SR.GIE** is cleared, the interrupt remains pending until **SR.GIE** is set. When **SR.GIE** is set high, a pending interrupt will then set the corresponding bit in IFL bit. The IER bits can block a IFL bit from causing an interrupt; however, the IFL bit will still be set and can be polled, if desired.

The IFL register should not be written when the **SR.GIE** bit is set; otherwise, an edge-sensitive interrupt occurring during the I/O write instruction could be lost. The following code could lose an interrupt:

```
                // assume SR.GIE set
acc = 0xFFFD;
ifl &= acc;     // read-modify-write to IFL. Could lose an incoming interrupt
```

The following code will not miss an interrupt, since pending logic stores all incoming edge-sensitive interrupts when **SR.GIE** is clear:

```
                // assume SR.GIE set
disable;        // clear SR.GIE, any incoming interrupts will now be pending
acc = 0xFFFD;
ifl &= acc;     // IFL can be written without concern for losing interrupts
enable;         // re-set SR.GIE, if any interrupts came in while it was clear, they
                //      will happen now.
```

When an interrupt occurs, **SR.GIE** is cleared by hardware, and the Program Counter (PC) value is written to the Shadow Program Counter registers SPC/SPCH. SPC/SPCH must be stored on a software stack before the interrupt service routine calls any subroutines. If SPC/SPCH is not saved, its contents will be lost. The ISR must restore SPC/SPCH from the stack, before leaving the interrupt service routine. Then, a return-from-interrupt instruction (`RETI`) copies the contents of the SPC/SPCH to the PC register, and if **IER.DRI** is clear, sets the global interrupt enable flag **SR.GIE** high.

The Debug Port interrupt vectors to location 0x0000C. The Debug Port interrupt is enabled by setting the **IER.DIEN** bit. The Debug Port interrupt can also be caused by one of the DSPs executing a `TRAP` instruction (which is not masked by **IER.DIEN**). The `TRAP` instruction also causes the particular DSP to vector to the Debug Interrupt vector.

When a debug interrupt occurs, the PC value is written to the Debug Shadow PC registers DSPC/DSPCH, and the Debug Interrupt Vector is loaded into the PC. The Debug Interrupt does not affect **SR.GIE** and cannot be interrupted. The `RET` instruction is used to exit the Debug Interrupt since it does not affect **SR.GIE.**

The first `RET` instruction inside the Debug ISR fetches the return address from DSPC/DSPCH; therefore, special attention is needed when using subroutines from within the Debug ISR. If this issue is not addressed, the first subroutine's `RET` instruction will exit the entire Debug ISR instead of returning to the calling routine. In addition, this first `RET` restores/unblocks **SR.GIE**, therefore, **SR.GIE** must be cleared before the first `RET`, and restored before exiting the debug ISR.

2.1.4 Execution Unit

The execution unit consists of the arithmetic logic unit, the shift unit, and the I/O unit. The condition code bits, stored in the Status Register are updated based on the value of the result. The type of operation is encoded in the instruction byte.

2.1.5 Address Generation Unit

The address generation unit calculates the addresses of the source and destination operands. The address generation unit supports five different operand addressing modes: immediate, register, direct, indirect and program memory indirect. Immediate addressing supports 8- and 16-bit immediate data, as one byte or two bytes, directly following the instruction. When the operand resides in the accumulator or one of the two address registers, the register identifier is encoded in the instruction. Memory operands can be either 1-byte or 2-bytes long. With near-direct addressing, the upper four bits of the memory operand address comes from the Address Page pointer **SR.AP[3:0]**, and the lower 8-bits of the memory operand address come from the instruction word (2-byte instruction). With far-direct addressing, the entire Data memory can be accessed using sixteen bits in the instruction word (3-bytes instruction) for the memory operand address. Indirect addressing utilizes one of two address registers, AR0 and AR1, to obtain the address for the data memory operand. Program Memory Indirect Addressing uses the address page register PGMP and address register AR1 for the Program memory operand address and supports efficient storage of constant tables.

Instructions support either zero, one, or two source operands where the destination operand is the same as one of the source operands. An operand can be an immediate value, a value stored in the accumulator, a value stored in an address register, a value stored in the Data memory, or a value stored in the Program memory.

An instruction is limited to one memory operand. For instructions requiring a source memory operand and a destination memory operand, the source memory operand and the destination memory operand must be the same. This reduces the size of the instruction, since a separate destination does not have to be explicitly specified in the instruction.

2.1.5.1 Immediate Data

Immediate data supports 8- and 16-bit immediate data, expressed as one byte or two bytes directly following the instruction. When an instruction utilizes immediate data, the byte or word decision is embedded in the instruction encoding, and the **SR.W** Word flag is ignored. The assembler uses the data size to determine which encoding to use. To force a constant to 16 bits, `(int)` should preface the constant. Byte constants affect only ACCL. The SR bits are affected only by ACCL. ACCH remains unchanged, unless a carry or borrow occurs because of an arithmetic operation.

```
acc = 0x12;         // ACCL loaded with 0x12, ACCH remains unchanged. The SR flags are
                    //   based on ACCL contents ONLY.
acc = (int)18;      // ACC = 0x0012. The SR flags are affected by all of ACC.
acc += 0x1234;      // 0x1234 is added to ACC. The SR flags are affected by all of ACC.
```

2.1.5.2 Inherent

In Inherent mode, the instruction does not specify data or a data location. Instead, the opcode is explicitly stated by the instruction, and performs a discrete task. All Inherent mode instructions are 1-byte instructions.

```
reti;               // Returns from an interrupt handling state.
enable;             // Set SR.GIE
carry = 0;
nop;
```

2.1.5.3 Register Addressing

In Register Addressing Mode, operands are explicitly encoded within the instruction. The address registers and the I/O registers can be register operands. I/O registers are read and written using separate I/O read and I/O write instructions. In Register Addressing mode, the operand size is always 16-bits, and the **SR.W** Word flag is ignored. When ACC is used in all single-operand instructions (except Test), it is also considered a 16-bit register operand. In dual-operand instructions and the Test instruction, the other operand determines whether ACC or ACCL is used (word or byte operands).

```
acc += ar0;         // Adds the address register AR0 to the accumulator ACC
io_reg &= acc;      // io_reg is AND'd with ACC, with the result stored in io_reg
acc = io_reg;       // ACC is loaded with 16-bit value from io_reg
acc++;              // ACC is incremented
acc >>= 1;          // ACC is shifted right by 1
*sp = acc;          // push ACC onto stack
sp -= 2;            // decrement the stack pointer
```

2.1.5.4 Direct Addressing

The Host Controller supports near-direct (NDIR) addressing and far-direct (FDIR) addressing of Data memory variables. With near-direct addressing, the upper bits of the Data memory operand address (memory page) come from the Address Page pointer **SR.AP[3:0]**, and the lower 8 bits of the memory operand address come from the instruction word (2-byte instruction). With far-direct addressing, the entire Data memory address is included in the instruction word (3-byte instruction). When accessing Data memory, the **SR.W** Word flag determines whether operands are bytes or words. When `near` or `far` are not explicitly stated, the compiler or assembler will determine the default usage.

```
sw;                      // set word mode
(near)var += acc;        // Adds ACC to var, where the lower 8 addr. bits of var are in
                         //  the instruction & SR.AP[3:0] supplies the upper addr. bits.
(far)var += acc;         // Adds ACC to var, where the entire address of var is included
                         //  in the instruction. SR.AP[3:0] are not used.
sb;                      // set byte mode
acc -= var1;             // Subtracts var1 byte from ACCL. The SR flags are updated
                         //  based on ACCL. ACCH is unaffected unless a carry or borrow
                         //  occurs
var1 &= acc;             // var1 byte is AND'd with ACCL, The SR flags are updated based
                         //  on the resultant var1 byte stored in memory.
```

2.1.5.5 Indirect Addressing

When using Indirect Addressing Mode, an instruction does not directly specify the address of the data memory operand. Instead, one of the address registers AR0 or AR1 provides the address. These instructions are one byte in length, since no address bits are needed with the instruction. Since this addressing mode accesses Data memory, the **SR.W** Word flag determines whether operands are bytes or words.

```
int var(0x100);   // word variables must start on a word boundary.
char var1;        // byte variables can start anywhere
acc = &var1;
ar0 = acc;        // load AR0 with byte variable var1 address
acc = &var;
ar1 = acc;        // load AR1 with word variable var address.

acc = 0xF234;
sw;               // set word mode
*ar1 = acc;       // Loads the variable pointed to by AR1 from ACC. The SR flags are
                  //   updated based on the word result stored at *AR1. Since var =
                  //   0xF234, SR.N would be set, and SR.Z would be clear.
acc -= *ar1;      // Subtracts variable pointed to by AR1 from ACC. The SR flags are
                  //   updated based on the entire ACC result. Since ACC is zero in
                  //   this example, SR.N would be clear, and SR.Z would be set.

sb;               // set byte mode
acc = 0x01FF;
*ar0 = acc;       // ACCL is loaded in var1. The SR flags are updated based on
                  //   byte var1. ACCH is unaffected unless a carry or borrow occurs.
                  //   Since var1 = 0xFF, SR.N is set and SR.Z is cleared.
acc--;            // ACC is decremented. ACC = 0x01FE
acc -= *ar1;      // var1 is subtracted from ACC, which makes ACC = 0x00FF. The SR bits
                  //   are updated on ACCL: SR.N set and SR.C clear. ACCH is affected
                  //   due to arithmetic borrow.
```

2.1.5.6 Program Memory Indirect Addressing Mode

The Program memory of the Controller can be the data operand. The Program memory is accessed through the program memory indirect addressing mode. The program memory indirect addressing mode uses the address register AR1 (and the PGMP page register) to indirectly point to the location in program memory where the data is stored. To access Flash memory, the Flash Protection for the particular memory location must be disabled (set to 0). The data read from a protected partition will be indeterminate. Internal Flash memory can only be written when **RGEN.$\overline{\text{XME}}$** is set and **MMPC.$\overline{\text{XMQ}}$** is clear (DSP data memory selected for external memory port), and Flash protection for the location is disabled. In addition, internal Flash memory is written a word (two bytes) at a time, from ACC, with ACCH going to the lower location and ACCL going to the upper location. Due to this constraint, AR1 must always point to even addresses (bit 0 = 0) when writing to Program memory.

```
acc = 1;          // Example writes a word to the upper half of Program memory.
pgmp = acc;       // Set Program memory page to upper half of program memory.
acc = 0x8002;
ar1 = acc;        // Program memory address is 0x18002
acc = 0x1234      // Data to write to Program memory: ACCH = 12h, and ACCL = 34h.
(rom)*ar1 = acc;  // Moves the accumulator ACC value to Program memory location
                  //   indirectly addressed by PGMP and AR1.
                  //   Program location 0x18002 = 12h, and
                  //   Program location 0x18003 = 34h. Protection for this location
                  //     must be disabled.
```

```
acc = 0x07F2;           // Example adds a byte of Program memory to ACC.
acc = acc + (rom)*ar1;  // Adds a byte of Program memory at the location indirectly
                        // addressed by PGMP and AR1 to ACC. If PGMP and AR1 still point
                        // to 0x18002, then 12h will be added to ACC, producing the
                        // result ACC = 0x0804. To read this Program memory location,
                        // the Flash block it resides in must NOT be protected.
                        // 0x18002 is in FPBK.FB13 (must be clear).
```

5Ah PGMP Program Memory Page register Host

Bit	Label	Description	Default
15..	rsvd	Reserved. Write to 0.	000h, 0
1	A17	256-byte Information block only. The lower 16 address bits are in AR1.	0
0	A16	Selects the upper or lower half of the 128k Program memory during Program memory reads and writes. The lower 16 address bits are in AR1.	0

Table 2-17: PGMP Register

A17 17th address bit of Program memory. When set, the 256-byte information block of the Flash memory is selected. **A16** must be clear when **A17** is set.

A16 16th address bit of Program memory. Selects either the upper (when set) or lower 64k of Program memory to read from or write to.

2.1.6 Instruction Summary

Instruction	Opcode	Example	Bytes	†Cycles
Add	0000 xxxx	acc = acc + *ar0;	1/2/3	1/2/3
Add with Carry	0001 xxxx	acc = acc + *ar0 + carry;	1/2/3	1/2/3
And (logical)	0110 xxxx	acc = acc & var;	1/2/3	1/2/3
Bit move	1100 dzzz	var.4 = carry;	1/2/3	1/2/3
Clear ACC	1100 0000	clr acc;	1	1
Clear Carry	1011 1111	carry = 0;	1	1
Complement Carry	1010 1111	carry = !carry;	1	1
Decrement	1010 0yyy	var--;	1/2/3	1/2/3
Decrement Stack Pointer	1010 1001	sp -= 2;	1	1
Disable interrupts	1011 1001	disable;	1	1
Enable interrupts	1011 1000	enable;	1	1
Exchange	1001 1yyy	xch(acc, var);	1/2/3	1/2/3
Exclusive OR (logical)	0100 xxxx	var = var | acc;	1/2/3	1/2/3
Increment	1000 0yyy	var++;	1/2/3	1/2/3
Increment Stack Pointer	1010 1000	sp += 2;	1	1
I/O Access	0zzz 0111	ifl = ifl & acc;	2/3	
Jump on ACC bit set/clear	111c bbbb	if (3) jmp <label>;	2	3
Jump on ACC bit set/clear - extended	1100 1110	if (!5) jmp <label>;	4	4
Jump Conditional	1101 sccc	if (!N) jmp <label);	2/3	3
Jump Conditional - extended	1100 1111	if (Z) jmp label;	4	4
Jump to Subroutine	1101 s100	<label>();	2/3	3
Jump to Subroutine - extended	1100 1111	<label>();	4	4
Move	0111 xxxx	ar1 = acc;	1/2/3	1/2/3
NOP	1010 1110	nop;	1	1
Or (logical)	0101 xxxx	var = var | acc;	1/2/3	1/2/3
Pop	1010 1011	acc = *sp;	1	1
Push	1010 1010	*sp = acc;	1	1
Reset	1011 1011	reset;	1	1
Return	1010 1100	ret;	1	2
Return from interrupt	1010 1101	reti;	1	2
Set byte	1011 1101	sb;	1	1
Set carry	1011 1110	carry = 1;	1	1
Set word	1011 1100	sw;	1	1
Shift left	1011 0yyy	var = var << 1;	1/2/3	1/2/3
Shift right	1001 0yyy	var >>= 1;	1/2/3	1/2/3
Subtract	0010 xxxx	acc = acc - var;	1/2/3	1/2/3
Subtract with carry	0011 xxxx	var = var - acc - carry;	1/2/3	1/2/3
Test	1000 1zzz	dummy = acc - var;	1/2/3	1/2/3
Trap	1011 1010	trap;	1	1
Write Program Memory	1001 1110	(rom) *ar1 = acc;	1	

† One cycle is equal to two 384Fs Host Controller clocks.

Table 2-18: Host Controller Instruction Set Summary

2.2 Inter-Processor Communications

The inter-processor communication ports consist of the COM ports between the Host Controller and the slave processors: the two DSPs and MOST Processor. The MOST Processor's COM port is hard-coded to use the protocol described below. The DSPs should be programmed to match this protocol to maximize reusable code.

The COM ports from the MOST and each DSP consist of a status register and two one-way data registers. The status register contains the start (**STR**), read (**RD**), and write (**WR**) flags. It also contains the interrupt (**INT**) flag, which informs the Controller that the slave needs servicing. The DSP data registers are 16-bits wide and the MOST data register is 8-bits wide. The data registers transfer data in either direction.

> The data registers are one-way (read-only and write-only), that share a common address. Since the data registers for each direction share the same I/O address, the value written cannot be read. The value read is from the other device.

For example, data written by the Host Controller to a DSP can be read by the DSP, but NOT by the Host Controller. When the Host Controller reads the register, the data read is the last data written by the DSP. On the Control bus, each data register has two I/O addresses. The Host Controller signifies the first and last word of a transfer by accessing one address. The Controller uses the other address to transfer all other data.

The **STR**, **RD**, and **WR** flags are read-only. They support handshaking between the Controller and each slave during data transfers. They are set when the Controller accesses the data register and are cleared when the slave accesses the data register. A low-to-high transition of the **STR**, **WR**, or **RD** flags can generate the COM port interrupt in the DSPs. A low-to-high transition of **INT** can generate the COM port interrupt in the Host Controller. Since the status flags have to cross clock boundaries, a one-cycle delay is incurred between when a status flag is changed, and when it becomes readable from the register. For example, if the DSP writes data into the COM data register, a `NOP` instruction should precede reading the **RD** bit low from the COM status register. Figure 2-6 is a block diagram of a generic COM port (although the DSP Slave Processor register names are shown for clarity).

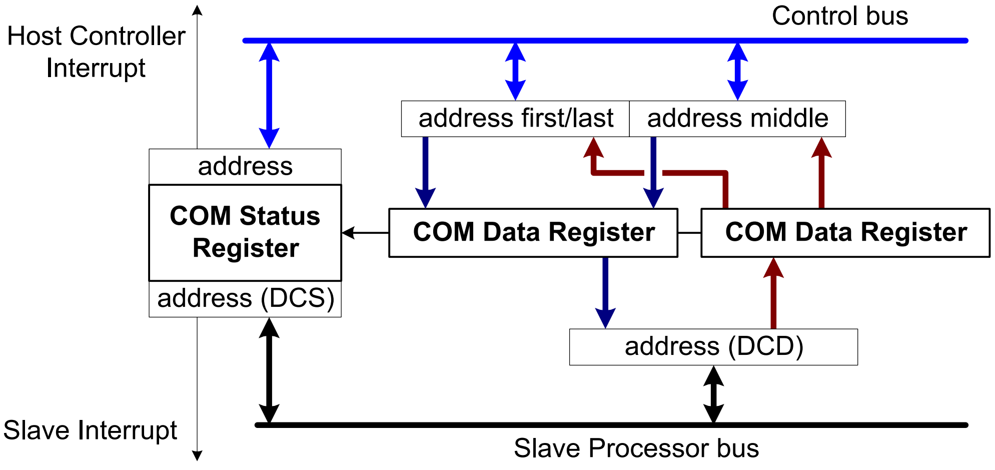


Figure 2-6: Generic COM Port

From the software's perspective, the transfer of data through the COM ports is much like that of the Control Port. The Host Controller operates as the master and the slave processor operates as the slave.

The slave manages a memory address pointer (MAP) and a software control register. The first word of a write transfer is written to the MAP. The MAP should then be incremented. Successive words should then be written to the location pointed to by the MAP pointer. The MOST processor is hard-coded to operate in this fashion; however, the DSPs and Host Controller must be user-programmed for this protocol.

To write a message to a slave, the Host initiates the transfer by writing the first data word of the message to the data register address that indicates the first/last data word. This sets the **STR** and the **WR** flags in the COM status register. The COM status register can be polled by the DSP or the DSP can be interrupted. When a slave reads the data register (there is only one data register address on the slave bus), **WR** is cleared. The DSP should interpret the first transfer as a memory pointer and load the data register with the memory location pointed to by the memory pointer when the transfer is a read. The DSP should write (arbitrary) data to the data register to clear **STR**. In this way, the next time the DSP reads the COM status register, **STR** will be low indicating that the transfer is data and not the memory address pointer (MAP).

To write more than one word to a slave, the Host Controller monitors the **STR** flag until it goes low. The Controller then writes the next word to the data register address specifying middle data words. This causes the WR flag to go high again. The DSP either monitors the **WR** flag or is interrupted when it goes high. When a slave reads the data register again, the WR flag is cleared.

Successive words can be transferred by the Controller by monitoring **WR** and writing new words when **WR** goes low (indicating the slave read the previous data). A message of any length can be transferred from the Controller to a slave. The Controller simply stops sending data when the message is complete. Figure 2-7 depicts a write transfer from the Controller to a slave (MOST Processor or a DSP).

For a DSP, the Controller's write transfers can be supported by copying the MAP data to a DSP pointer variable `map_ptr`, and using the following command when a byte is received:

```
*map_ptr++ = dcd;
```

This will store the DSP's DCD COM port register data at the address pointed to by `map_ptr`, and then post-increment the MAP pointer `map_ptr` to be ready for the next transfer.

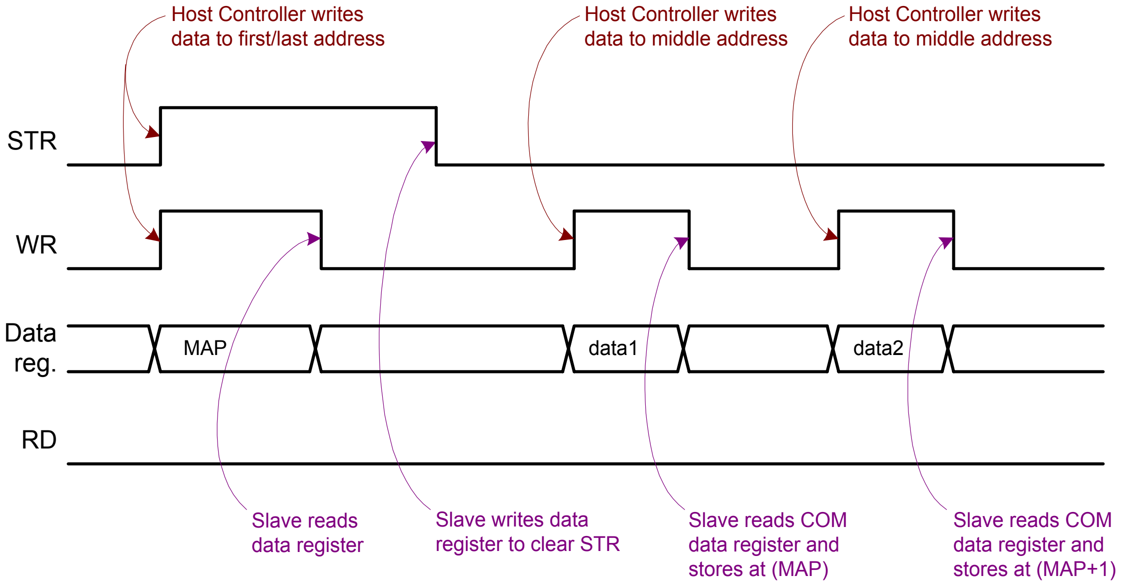

Host Controller writes data to first/last address
Host Controller writes data to middle address
Host Controller writes data to middle address
STR
WR
Data reg.
MAP
data1
data2
RD
Slave reads data register
Slave writes data register to clear STR
Slave reads COM data register and stores at (MAP)
Slave reads COM data register and stores at (MAP+1)

Figure 2-7: Generic COM Port Write Sequence

The Controller begins reading a message from a slave by writing the read address (MAP) to the data register that signifies the first/last data word. This sets the **STR** and **WR** flags in the status register. The DSP should read the address from the data register (which clears the **WR** bit), get the data pointed to by the MAP, and writes that data into the COM data register (which clears the **STR** bit).

To read additional words, the Controller should read the data register address that signifies middle data words. This read sets the **RD** flag. The DSP can either monitor this flag or be interrupted. The DSP then loads the data register with the memory location pointed to by the memory address pointer. This clears the **RD** flag. The Controller monitors the **RD** flag for this high-to-low transition. Then the Controller reads the data register.

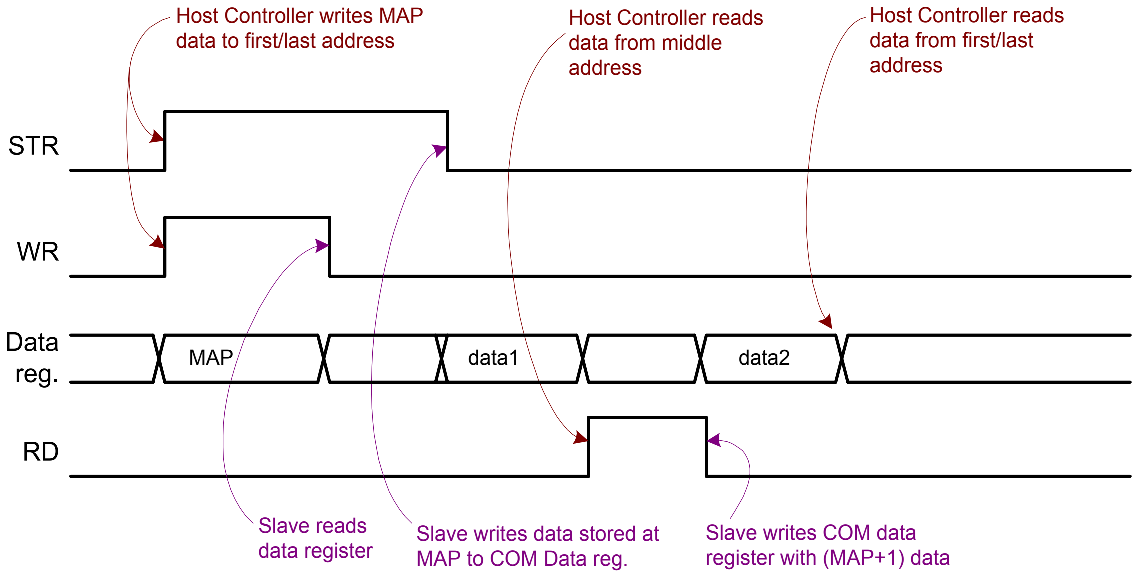

Host Controller writes MAP data to first/last address
Host Controller reads data from middle address
Host Controller reads data from first/last address
STR
WR
Data reg.
MAP
data1
data2
RD
Slave reads data register
Slave writes data stored at MAP to COM Data reg.
Slave writes COM data register with (MAP+1) data

Figure 2-8: Generic COM Port Read Sequence

The Controller can terminate a read transfer by reading from the first/last data register address. A read from this address does not cause **RD** to go high. Figure 2-8 is a timing diagram illustrating a read transfer.

For a DSP, the Controller's read transfers can be supported by copying the MAP data to a DSP pointer variable `map_ptr`, and using the following command when a byte is received:

```
dcd = *map_ptr, map_ptr++;
```

This will store the data, pointed to by `map_ptr`, into the DSP's DCD COM Port register, and then post-increment the MAP pointer `map_ptr` to be ready for the next transfer.

The DSP can set the **INT** flag to inform the Controller that it requires servicing. This flag is read/writable by both the DSP and the Controller. Once the Controller services the slave, it should clear the **INT** flag.

2.2.1 MOST Processor COM Port

10h RCS MOST COM Status Register Host

Bit	Label	Description	Default
15..5	rsvd	Reserved. Write to 0	00h, 000
4	AINT	Asynchronous (or Packet) Data Flag. Set whenever a Packet has been received or is finished being transmitted by the MOST Processor. This bit does not cause an interrupt to the Host Controller, so it must be polled. (read-only)	0
3	INT	Interrupt flag. Set based on the error bits active in the MOST register bMSGS, enabled by the corresponding bits in the MOST bIE register.	0
2	STR	Start transfer (read-only)	0
1	WR	Write data available (read-only)	0
0	RD	Read data request (read-only)	0

Table 2-19: RCS Register

06h RCF MOST COM First/Last Data

07h RCM MOST COM Middle Data Host

Bit	Label	Description	Default
15..8	rsvd	Reserved. Write to 0	00h
7..0	D[7:0]	bit 7 is MSB. Data written to the MOST Processor (write only) Data read from the MOST Processor (read only)	00h

Table 2-20: RCF, RCM Registers

The MAP value is 8 bits, which spans 256 bytes of the MOST Processor register space. The MOST Processor register space spans 1024 bytes, or four pages of 256 bytes each. The last byte in each page is reserved for switching pages. Therefore, if the Host Controller writes the MAP to FFh (RCF = FFh), and then writes RCM to 0, 1, or 3 (page 2 is reserved); then the MOST Processor switches to the respective memory page. When the MAP is auto-incrementing and reaches the end of a page, it wraps to the beginning of the same page.

2.2.2 DSP0 COM Port

11h D0CS DSP0 COM Status Register Host

Bit	Label	Description	Default
15..4	rsvd	Reserved. Write to 0	000h
3	INT	Interrupt flag	0
2	STR	Start transfer (read-only)	0
1	WR	Write data available (read-only)	0
0	RD	Read data request (read-only)	0

Table 2-21: D0CS Register

08h D0CF DSP0 COM First/Last Data

09h D0CM DSP0 COM Middle Data Host

Bit	Label	Description	Default
15..0	D[15:0]	bit 15 is MSB. Data written to DSP0 (write only) Data read from DSP0 (read only)	0000h

Table 2-22: D0CF, D0CM Registers

2.2.3 DSP1 COM Port

12h D1CS DSP1 COM Status Register Host

Bit	Label	Description	Default
15..4	rsvd	Reserved. Write to 0	000h
3	INT	Interrupt flag	0
2	STR	Start transfer (read-only)	0
1	WR	Write data available (read-only)	0
0	RD	Read data request (read-only)	0

Table 2-23: D1CS Register

0Ah D1CF DSP1 COM First/Last Data
0Bh D1CM DSP1 COM Middle Data Host

Bit	Label	Description	Default
15..0	D[15:0]	bit 15 is MSB. Data written to DSP1 (write only) Data read from DSP1 (read only)	0000h

Table 2-24: D1CF, D1CM Registers

2.2.4 DSP Debug Interface

The DSP debug interface consists of debug COM Ports and a DSP Trap Shadow Program Counter registers.

The debug COM port is identical to the regular COM ports between the DSPs and the Host Controller, except that the DSPs do not have the capability to interrupt the Host Controller through this interface. The data register can be read or written by the Host Controller using DD0CF for the first or last transfer and DD0CM for middle word transfer to DSP0. Similarly, DD1CF and DD1CM are for DSP1. The Host Controller can read the status register using DD0CS and DD1CS for DSP0 and DSP1 respectively.

1Eh DD0CS DSP0 Debug COM Status Register Host

Bit	Label	Description	Default
15..6	rsvd	Reserved. Write to 0	00h, 00
5	TIP	Trap In Progress status (read only)	0
4	TRINT	Trap Interrupt	0
3	rsvd	Reserved, Write to 0	0
2	STR	Start transfer (read-only)	0
1	WR	Write data available (read-only)	0
0	RD	Read data request (read-only)	0

Table 2-25: DD0CS Register

13h DD0CF DSP0 Debug COM First/Last Data
14h DD0CM DSP0 Debug COM Middle Data Host

Bit	Label	Description	Default
15..0	D[15:0]	bit 15 is MSB. Data written to DSP0 (write only) Data read from DSP0 (read only)	0000h

Table 2-26: DD0CF, DD0CM Registers

1Fh DD1CS DSP1 Debug COM Status Register Host

Bit	Label	Description	Default
15..6	rsvd	Reserved. Write to 0	00h, 00
5	TIP	Trap In Progress status (read only)	0
4	TRINT	Trap Interrupt	0
3	rsvd	Reserved, Write to 0	0
2	STR	Start transfer (read-only)	0
1	WR	Write data available (read-only)	0
0	RD	Read data request (read-only)	0

Table 2-27: DD1CS Register

15h DD1CF DSP1 Debug COM First/Last Data
16h DD1CM DSP1 Debug COM Middle Data Host

Bit	Label	Description	Default
15..0	D[15:0]	bit 15 is MSB. Data written to DSP1 (write only) Data read from DSP1 (read only)	0000h

Table 2-28: DD1CF, DD1CM Registers

48h D0TSPC DSP0 Trap Shadow Program Counter (read-only) Host

Bit	Label	Description	Default
15..11	rsvd	Reserved. Write to 0	00000
10..0	D[10:0]	Program Counter Copy. Loaded when the DSP0 executes a `TRAP` instruction or when the Host Controller sets **DD0CS.TRINT**.	00h, 000

Table 2-29: D0TSPC Register

49h D1TSPC DSP1 Trap Shadow Program Counter (read-only) Host

Bit	Label	Description	Default
15..11	rsvd	Reserved. Write to 0	00000
10..0	D[10:0]	Program Counter Copy. Loaded when the DSP1 executes a `TRAP` instruction or when the Host Controller sets **DD1CS.TRINT**.	00h, 000

Table 2-30: D1TSPC Register

2.3 Control Bus Peripherals

The Control Bus peripherals consist of peripherals managed by the Host Controller; as well as enables and volume controls for peripherals controlled by other processors. The data for the PWM DACs come from DSP0, but the enable is controlled by the Host Controller. The data for the Source Converters (MPX, MIC, Audio ADCs and Quad DACs) is transferred by the MOST Processor; whereas, the enables and volume controls are in the Controller's I/O space. Volume controls are also in the DSP's I/O space.

2.3.1 Clock Manager

The Clock Manager generates all the clocks required by the chip and outputs a clock (**RMCK**) to support synchronizing external devices. As shown in Figure 2-9, the Clock Manager consists of an input multiplexer, an analog PLL, and output dividers. The input multiplexer allows the chip to be configured as a network timing-master or timing-slave device. As a network timing-slave device, the PLL is clocked from the network receiver interface, **RX**. As a network timing-master device, the PLL is clocked from a crystal, the source data port clock, or **SR0** (configured as an SPDIF input). The analog PLL generates a 3072×Fs clock which is then divided down to create all the necessary internal clocks.

While reset is asserted, the crystal oscillator is disabled and the PLL is pulled to a frequency below the normal operating frequency. When reset goes away, the crystal is automatically enabled and the PLL begins to lock to the crystal. If the external crystal is actually 512×Fs, the PLL will lock and the **CMCS.LOCK** bit will go high. If the crystal is actually 384×Fs or 256×Fs, the PLL may not be able to lock depending on the pull range of the PLL. In either case, the Control Port will still operate, thereby allowing the Clock Manager parameters to be changed from the external system.

The external system should first load the proper **CMCS.XTL[1:0]** value. Then the PLL should lock, indicated via the **CMCS.LOCK** bit, and the PLL input mux. select bits can be changed so the PLL locks to **SCK**, **SRO** (SPDIF), or **RX** data.

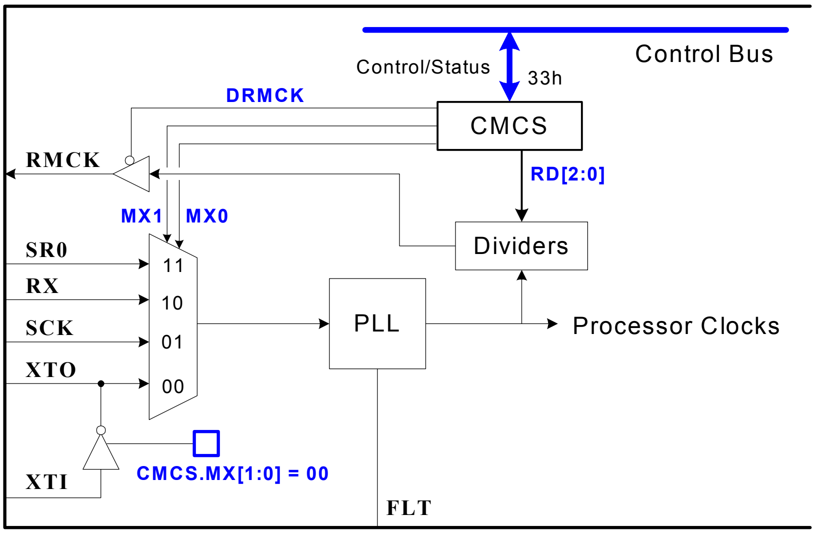


Figure 2-9: Clock Manager

33h CMCS Clock Manager Control/Status Host

Bit	Label	Description	Default
15	LOCK	PLL lock status (read only)	0
14	ASC	Automatic switch to crystal	0
13	rsvd	Reserved, Write to 0.	0
12	rsvd	Reserved, Write to 0	0
11	rsvd	Reserved, Write to 0.	0
10	rsvd	Reserved, Write to 0	0
9	PLD	PLL disable	0
8	PD	Power down	1
7	DRMCK	When set, **RMCK** output is disabled and can be used as **IOG0**	0
6..4	RD[2:0]	**RMCK** divider	000
3..2	XTL[1:0]	Crystal oscillator divider	00
1..0	MX[1:0]	PLL input multiplexer select	00

Table 2-31: CMCS Register

LOCK PLL Locked. When set, indicates that the PLL is properly locked onto the source. **LOCK** will go low immediately when the source stops toggling. **LOCK** will also go low if an out-of-lock condition is detected on the source.
When out of lock, the Control Port still operates; however, Source data is not transferred. Once lock is reestablished, three frames are required for Source data to be transferred properly.

ASC Automatic Switch. When set and the PLL source is not the crystal, causes the **MX[1:0]** bits to change to the crystal (00) when the PLL loses lock (**LOCK** = 0). When **MX[1:0]** are changed to 00, the crystal starts to oscillate. The crystal start-up time can be quite long and will cause the PLL to drop in frequency until the oscillator stabilizes.

PLD PLL Disable. When set, the PLL is pulled to its nominal low frequency.

PD Power Down. When set, powers down the entire chip by turning off all clocks to the digital section, placing the FLASH memory in standby mode, and shutting down all analog bias currents. Before setting **PD**, the PLL input mux should be set to any setting except the crystal (**XTL[1:0]** = 00) to insure that power through the crystal oscillator is minimized. RAM contents are preserved while **PD** is set. **PD** is cleared through a hardware reset or any of the conditions enabled through the RGEN register. The part goes through an under-voltage condition on initial power-up causing the **PD** bit to be set at power-up. The chip is brought out of power-down by a low-to-high edge on **$\overline{\text{RST}}$**, or toggling **RX**, **GPA0-GPA2**; as configured through the RGEN register.

DRMCK Disable **RMCK**. When clear, **RMCK** is enabled and has a frequency based on **RD[2:0]**. When set, **RMCK** output can be used as GPIO pin **IOG0**.

RD[2:0] **RMCK** Divider. When **RMCK** is enabled (**DRMCK** = 0), these bits determine the output frequency as a ratio to the locked sample frequency.
000 – 384×Fs
001 – 256×Fs
010 – 128×Fs
011 – 64×Fs
100 – 1536×Fs
101 – 1024×Fs (33 % duty cycle)
110 – 768×Fs
111 – 512×Fs

XTL[1:0] Crystal oscillator divider. These bits allow the crystal oscillator to be one of three frequencies for a given network sample frequency (Fs).
00 - 512×Fs
01 - 384×Fs
10 - 256×Fs
11 - Reserved

MX[1:0] PLL Multiplexer input select. Selects the source to the PLL. When using NetServices software, a crystal is required even in timing-slave nodes since the node can be set as a timing-master for diagnostics if the ring goes down.
00 – Crystal oscillator. If not selected, the oscillator is automatically disabled to save power.
01 – **SCK** (Source Port serial bit clock)
10 – **RX** (network receiver – timing-slave device)
11 – **SR0** (SPDIF Source Port input)

Power can be minimized during normal operation by turning off the chip resources that are not being used. The DSP processor has a run bit which, when cleared, turns off the DSP processor's clock. This reduces the power consumption of the processor to the DC power of the memories. Each data converter has an enable bit. When not enabled, the converter is in a low-power state.

38h CM4 Clock Manager 4 Host

Bit	Label	Description	Default
15..8	rsvd	Reserved. Write to 0.	00h
7..0	D[7:0]	Affects the PWD tolerance. Must be set to C3h for proper operation.	C3h

Table 2-32: CM4 Register

2.3.1.1 Crystal Pins

The crystal oscillator is used by the timing-master node to set the timing for the entire ring. Timing-slave nodes generally have a crystal oscillator to allow the node to run diagnostics when the ring is down.

The crystal oscillator should be fundamental mode, parallel resonant with a load capacitor specified as mentioned below. Figure 2-10 depicts the nominal oscillator circuit, where the left side illustrates the amplifier's small-signal output impedance and the right side illustrates the amplifier's large-signal output impedance. Since the internal inverter/amplifier is operated in its linear region, external series resistors should not be used, as they will lower the gain and could cause start-up problems.

When the crystal oscillator is not selected as the timing source for the node (**CMCS.MX[1:0]** not equal to 00), the oscillator is powered down to minimize noise and power consumption. If an external signal or a crystal is not used, **XTI** should be grounded to minimize current draw.

If an external clock is used in lieu of a crystal oscillator, it must support CMOS drive levels and be connected to **XTI** (see Figure 2-10), and **XTO** must have a total capacitance of less than 10 pF. For the timing-master node, this clock must be jitter free. For a timing-slave node, the clock must also be jitter free to support ring-down diagnostics where the timing-slave might be the timing-master for the rest of a broken ring.

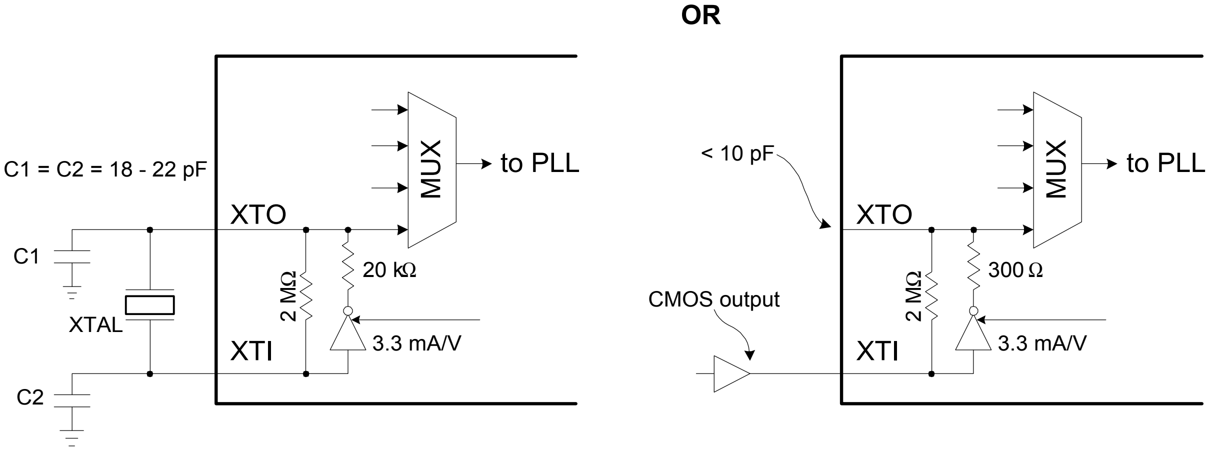


Figure 2-10: Crystal Oscillator Input

The crystal frequency can be 256xFs, 384xFs, or 512xFs; selected via the **CMCS.XTL[1:0]** bits. The load capacitors should typically be in the range of 18 pF to 22 pF. For more information on crystals, see Section 8.4.

CMCS.XTL[1:0]	10	01	00	Units
Fs	256x	384x	512x	
38 kHz	9.728	14.592	19.456	MHz
44.1kHz	11.2896	16.9344	22.5792	MHz
48 kHz	12.288	18.432	24.576	MHz

Table 2-33: Crystal Oscillator Frequencies

2.3.2 Global Timer

The Global Timer consists of an 8-bit counter clocked off of SCK at 16xFs, and can be read by the Controller, DSPs, and MOST Processor. Even if the Source Ports are not used, SCK must be configured (set as an output, or an input with the proper external clock applied) through bSDC1 for GTR to operate. Each bit of the counter oscillates at different rates which are used for inter-processor synchronization. The rates available are 8, 4, 2, 1, 1/2, 1/4, 1/8, and 1/16xFs. Some of the bits of the Global Timer can generate periodic interrupts on the rising edge in various processors. The rising edge of bits **FS1** through **FS8** are aligned; however, the other bits have skewed rising edges as depicted in Figure 2-12.

36h GTR Global Timer (read-only) Host

Bit	Label	Description	Default
15..8	rsvd	Reserved, Write to 0	00h
7	FS16TH	one rising edge every 16 Fs periods	0
6	FS8TH	one rising edge every 8 Fs periods	0
5	FS4TH	one rising edge every 4 Fs periods	0
4	FSHALF	one rising edge every other Fs period	0
3	FS1	1 rising edge every Fs period	0
2	FS2	2 rising edges every Fs period	0
1	FS4	4 rising edges every Fs period	0
0	FS8	8 rising edges every Fs period	0

Table 2-34: GTR Register

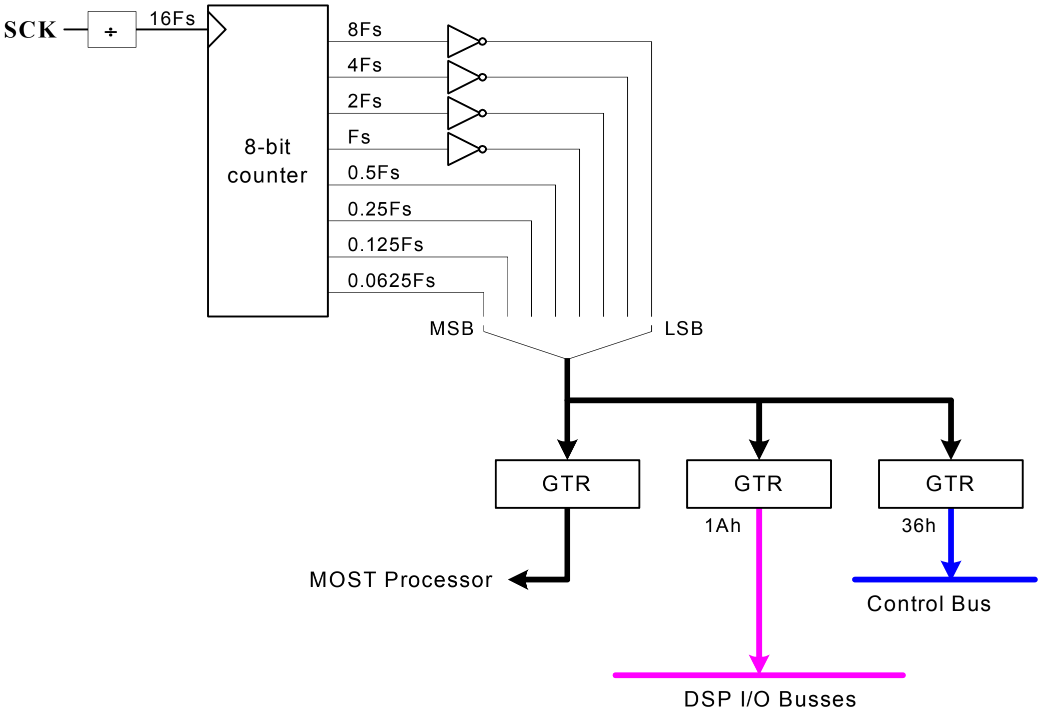


Figure 2-11: Global Timer

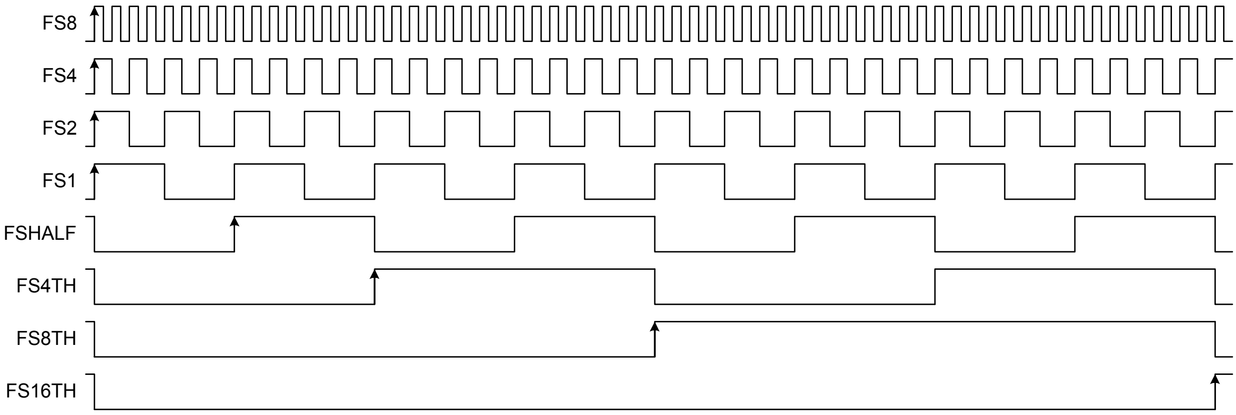


Figure 2-12: Global Timer Timing

2.3.3 Control Port

The Control Port is the interface between the external system and the OS8805, and provides an access portal to the Controller's RAM and I/O registers. The Control Port operates in an I^{2}C-compatible, a USART, or an SPI format. The Control Port hardware supports an I^{2}C data format (even in SPI mode) where the first byte contains the address and read/write bit, followed by the rest of the data. A generic SPI mode is available (**CPS.GSPI** set) where no interpretation of the data is performed; however, the standard I^{2}C data format is preferred as it is similar, in format, to the internal COM Port formats (allowing sharing of code), and is the standard format used by Oasis SiliconSystem's design tools. In USART mode, the Control Port interface is through the USART0 registers. The Control Port can operate the serial port as a master or slave device for all formats except OSPI, which is slave only. The USART format and register interface are described in the separate *USARTs* Section.

When $\overline{\textbf{RST}}$ is de-asserted or at initial power-up, the initial Control Port format is selected by a pull-up (I^{2}C format) or pull-down (OSPI format) on the **SCL/SCLK** pin. The Control Port format can also be changed after power-up via the **CPS.I2CF** bit. When the **SCL/SCLK** pin is pulled high at startup (default I^{2}C format), the **SDA** pin will be driven low for one 128Fs period immediately on exiting the reset condition. This produces a start/stop condition on the bus; however, no address is output..

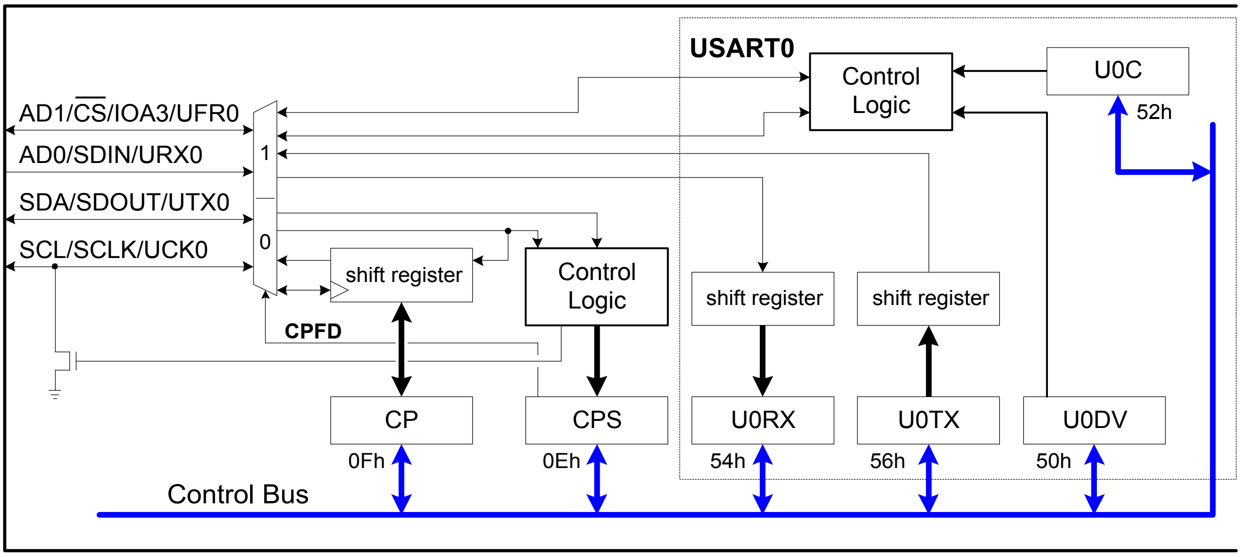


Figure 2-13: Control Port

The Control Port, when not in USART mode, is accessed by the Host Controller through the Control Port Status register and the Control Port data register. The Status register also includes configuration options. The Control Port interrupt bit **IFL.ICP** goes high whenever a rising edge occurs of any of the CPS status bits: **STOP** (not in OSPI mode), **STR**, **WR**, **NACK**, **RD**. Since **IFL.ICP** is level-sensitive, the interrupt service routine must clear the condition causing the interrupt (clear the **CPS** bit as described below).

0Eh CPS Control Port Status Host

Bit	Label	Description	Default
15..12	rsvd	Reserved, Write to 0	0000
11	CPFD	Control Port Formats Disable	0
10	FAST	Fast Mode enable	0
9	NACK	No Acknowledge	0
8	CPMM	Control Port Master Mode. (must be clear in OSPI format)	0
7	STOP	Stop Condition (not valid in OSPI mode)	0
6	I2CF	0 is SPI mode, 1 is I^2C mode	*
5	GSPI	0 is Oasis-specific SPI mode (OSPI), 1 is generic SPI mode (GSPI)	0
4	CPOL	For generic SPI mode, polarity of clock	0
3	CPHA	For generic SPI mode, phase of clock to data	0
2	STR	Start transfer (read-only when CPMM clear)	0
1	WR	Write data available	0
0	RD	Read data request	0

* Initial state determined by state of **SCL/SCLK** pin at power-up.

Table 2-35: CPS Register

CPFD Control Port Format Disable. When clear, the Control Port formats listed in this section are enabled. If set, the Control Port formats are disabled and the Control Port can be configured for USART0 (**U0C.USEN** set) or the four pins can be used as EGPIO port **IOA[3:0]** (**U0C.USEN** clear)

FAST Fast mode enable. When the Control Port is a master (**CPMM** set), **FAST** determines the clocking serial port clocking frequency as shown below:

With **FAST** clear the approximate clocking frequency is 88.2 kHz when Fs = 44.1 kHz, and 96 kHz when Fs = 48 kHz.

With **FAST** set, the approximate clocking frequency is 295.8 kHz when Fs = 44.1 kHz, and 310.6 kHz when Fs = 48 kHz.

NACK No Acknowledge. Reading this bit does not read what was written. The read bit going high can generate an interrupt, if enabled. Writing **NACK** to zero clears the interrupt condition.

When the Control Port is configured as a slave port (**CPMM** clear):

NACK is set when the external master device is performing a read operation and does not acknowledge the transfer. Software can write this bit to zero **after** reading CP. When configured as a slave port, software should never write **NACK** to one.

When the Control Port is configured as a master port (**CPMM** set):

NACK is set when writing an external device and it does not acknowledge the transfer. In I^2C format, software should set **NACK** to one to keep the Control Port from acknowledging the next (last) transfer. Once the “no acknowledge” is transmitted, **NACK** will read back one and must be cleared in software. The data in CP must be read before clearing **NACK** or another eight clocks will be sent out. In GSPI format, **NACK** should be set for full-duplex communications, since it will keep CP reads from generating 8 clocks (only writes will generate 8 clocks). When using GSPI in one direction, **NACK** is used to stop the last transfer from generating clocks.

CPMM Control Port Master Mode enable. When clear, the Control Port is a slave serial device. When set, the Control Port is a master port and **FAST** sets the clocking speed used. OSPI master mode is not supported. In addition, I^{2}C multi-master mode is not supported.

STOP When reading, **STOP** set indicates a stop condition has been detected on the bus. Writing to 0 clears the bit and writing to 1 (when in I^{2}C master mode) causes a Stop condition. If a **STOP** condition is detected on the bus, an interrupt can be generated. The **STOP** bit is set at power-up and when switching to master mode, and should be cleared before enabling the port. The **STOP** bit also respond to any bus stop bits until the first start bit and address is sent. Then the **STOP** bit only responds after the correct address is sent.

When the Control Port is configured as a slave port (**CPMM** clear):

> In I^{2}C format, **STOP** is set when an I^{2}C Stop bit is detected on the bus. In either SPI format, **STOP** high indicates that the $\overline{\textbf{CS}}$ pin rose. **STOP** is cleared by writing **STOP** to zero. When configured as a slave port, software should never write **STOP** to one.

When the Control Port is configured as a master port (**CPMM** set):

> In I^{2}C format, **STOP** is set only after an I^{2}C Stop bit is detected on the bus. In GSPI format, **STOP** is always low. **STOP** is cleared by writing **STOP** to zero. In I^{2}C format, writing **STOP** to one causes a Stop bit to be transmitted onto the bus. Once the bit is transmitted, **STOP** will be read as one, indicating a Stop condition was seen on the bus.

I2CF SPI or I^{2}C format. When set, the Control Port uses the I^{2}C format. When clear, the Control Port uses the SPI format indicated by the **GSPI** bit. **I2CF** is initially set by the value of the **SCL/SCLK** pin when $\overline{\textbf{RST}}$ is de-asserted or at initial power-up.

GSPI Generic SPI. When set, and **I2CF** is clear, the Control Port interface uses a generic SPI format (GSPI). When **GSPI** is clear (Oasis-specific SPI, OSPI), the data format matches the I^{2}C format and the **WR** and **RD** bits work as in I^{2}C format. The Control Port must be a slave in OSPI mode.

CPOL Clock Polarity, in generic SPI format only. When **GSPI** is set and **I2CF** is clear, this bit determines the **SCLK** clock polarity as indicated in Figure 2-24.

CPHA Clock Phase, in generic SPI format only. When **GSPI** is set and **I2CF** is clear, this bit determines the **SCLK** clock phase as indicated in Figure 2-24. When in slave mode and **CPHA** is clear, $\overline{\textbf{CS}}$ must be brought high between each data byte transferred.

STR Start. Reading indicates if a start condition has been detected on the bus. In master mode, writing to 0 clears the bit and writing to 1 causes a Start condition to be transmitted. If a **STR** condition is detected on the bus, an interrupt can be generated in I^{2}C and GSPI-slave formats.

When the Control Port is configured as a slave port (**CPMM** clear):

> In I^{2}C format, **STR** is set when an I^{2}C Start bit is detected on the bus followed by an address matching 01000xy, where xy is set via the **AD1** and **AD0** pins, respectively. In both SPI formats, **STR** going high indicates that the $\overline{\textbf{CS}}$ pin fell. **STR** is cleared by writing to the CP register (**STR** is read-only and cannot be cleared by writing to 0). Writing **STR** to 1 has no effect.

When the Control Port is configured as a master port (**CPMM** set):

> In I^{2}C format, **STR** is set only after an I^{2}C Start bit is detected on the bus. In GSPI format, **STR** is always low. **STR** is cleared by writing **STR** to zero. In I^{2}C format, writing **STR** to one causes a Start bit to be transmitted onto the bus. Once the bit is transmitted, **STR** will be read as one, indicating a Start condition was seen on the bus.

WR Write data bit. When the Control Port is not in generic SPI mode (**GSPI** clear):

When in slave mode, **WR** is set when data is transferred from the serial input shift register to the CP register; and **WR** is cleared when the Controller reads the CP register or explicitly writes **WR** to 0. Writing **WR** to 1 has no effect.

When in master mode, **WR** is set when data is transferred from CP to the serial shift register, to be shifted out; and **WR** is cleared when the Controller writes CP with the next word, or explicitly writes **WR** to 0.

RD Read data bit. When the Control Port is not in generic SPI mode (**GSPI** clear):

When in slave mode, **RD** is set when data is transferred from the CP register to the shift register, to be shifted out of the chip; and **RD** is cleared when the Controller writes the CP register or explicitly writes **RD** to 0. Writing **RD** to 1 has no effect.

When in master mode, **RD** is set when data is transferred from the shift register to the CP register, and **RD** is cleared when the Controller reads CP or explicitly writes **RD** to 0.

0Fh CP Control Port Data Host

Bit	Label	Description	Default
15..8	rsvd	Reserved, Write to 0	00h
7..0	D[7:0]	data, bit 7 is MSB	00h

Table 2-36: CP Register

D[7:0] Control Port bi-directional data register when the Control Port is enabled and in any format except USART0.

When the Control Port is configured as a slave port (**CPS.CPMM** clear):

Reading or writing the CP register after **CPS.WR** or **CPS.RD** is set, clears the particular status bit, clearing the interrupt condition.

When the Control Port is configured as a master port (**CPS.CPMM** set):

Writing the CP register causes 8 clocks to be transmitted in GSPI and 9 clocks in I^{2}C. Reading the CP register also causes 8 clocks to be transmitted in GSPI and 9 clocks in I^{2}C format. However in I^{2}C format, **CPS.NACK** should be written immediately after reading the second to last transfer from CP. This will cause the part to not-acknowledge the last transfer. Once the last byte is in CP, it must be read BEFORE clearing **CPS.NACK** (or another 9 clocks will be transmitted). For GSPI format, if **CPS.NACK** is set before reading the last word in CP, no more clocks will be generated. Then CP should be read, followed by clearing **CPS.NACK**.

The following slave protocol is recommended for data transfers in the I^{2}C and Oasis-specific SPI (OSPI) formats, as the Control Port hardware is optimized to support this protocol. A generic SPI format is also provided to allow custom data formats.

When the part is a Control Port slave device (**CPS.CPMM** clear), the first data word written by the external system (after the chip address byte) is a Memory Address Pointer (MAP), which determines where the second data byte is to be stored/retrieved. The MAP should be incremented after the second-byte data transfer to be ready for the next byte. If more data is to be transferred, the bytes are transferred to/from memory sequentially with the MAP being incremented after every transfer. This scenario provides efficient transfer of a byte, or a block of bytes by only providing the MAP once, initially.

Figure 2-14 illustrates an external system write sequence that writes data to the Controller's memory. Using this protocol, the external system (System) first writes the address byte where the LSB indicates a read or a write sequence. For the I^{2}C format the 7-bit address must match the Control Port address for the **CPS.STR** bit to go high. For the OSPI format, the address is ignored and **CPS.STR** goes high when the $\overline{\text{CS}}$ pin goes low. The Controller, through either polling the CPS register or through the Control Port interrupt, reads CPS. Since **STR** is high, the Controller knows that the System is accessing the Control Port but doesn't know yet whether it's a read or a write sequence. In case it's a read sequence, the Controller takes the data stored at the previous MAP and stores it in the CP register. Writing CP causes the **CPS.STR** bit to fall. When the System writes the next byte, the **CPS.WR** bit goes high, and the controller should interpret this first byte to be a new MAP byte, and store the data as the new Memory Address Pointer. When CP is read

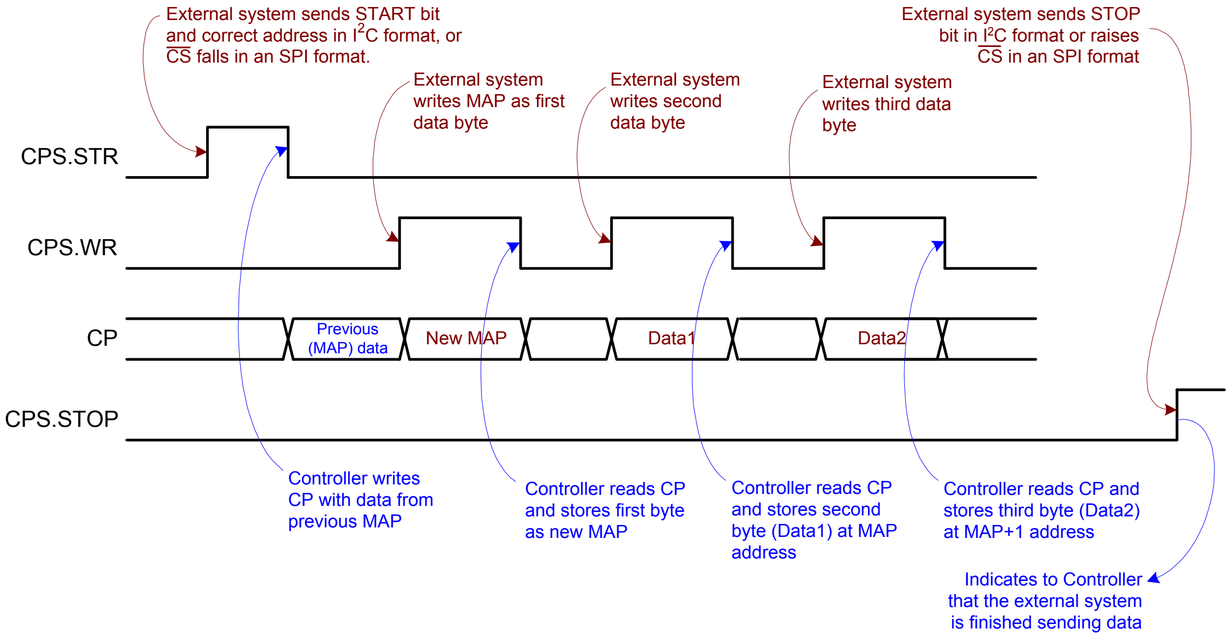

External system sends START bit and correct address in I2C format, or CS falls in an SPI format.
External system sends STOP bit in I2C format or raises CS in an SPI format
External system writes MAP as first data byte
External system writes second data byte
External system writes third data byte
CPS.STR
CPS.WR
CP
Previous (MAP) data
New MAP
Data1
Data2
CPS.STOP
Controller writes CP with data from previous MAP
Controller reads CP and stores first byte as new MAP
Controller reads CP and stores second byte (Data1) at MAP address
Controller reads CP and stores third byte (Data2) at MAP+1 address
Indicates to Controller that the external system is finished sending data

Figure 2-14: Control Port I^{2}C and OSPI Write Sequence

by the Controller, **CPS.WR** is cleared. When the System writes the next byte, **CPS.WR** goes high again and the Controller should interpret this new byte as actual data and store it at the location pointed to by the MAP byte. The Controller should then increment the MAP byte to be ready for the next byte received. When the System finishes writing all the data, it sends a **STOP** bit for the I^{2}C format, or raises the $\overline{\text{CS}}$ pin for the OSPI format. This causes the **CPS.STOP** bit to go high signaling the Controller that this sequence is over. The Controller should write the **CPS.STOP** bit to zero clearing it.

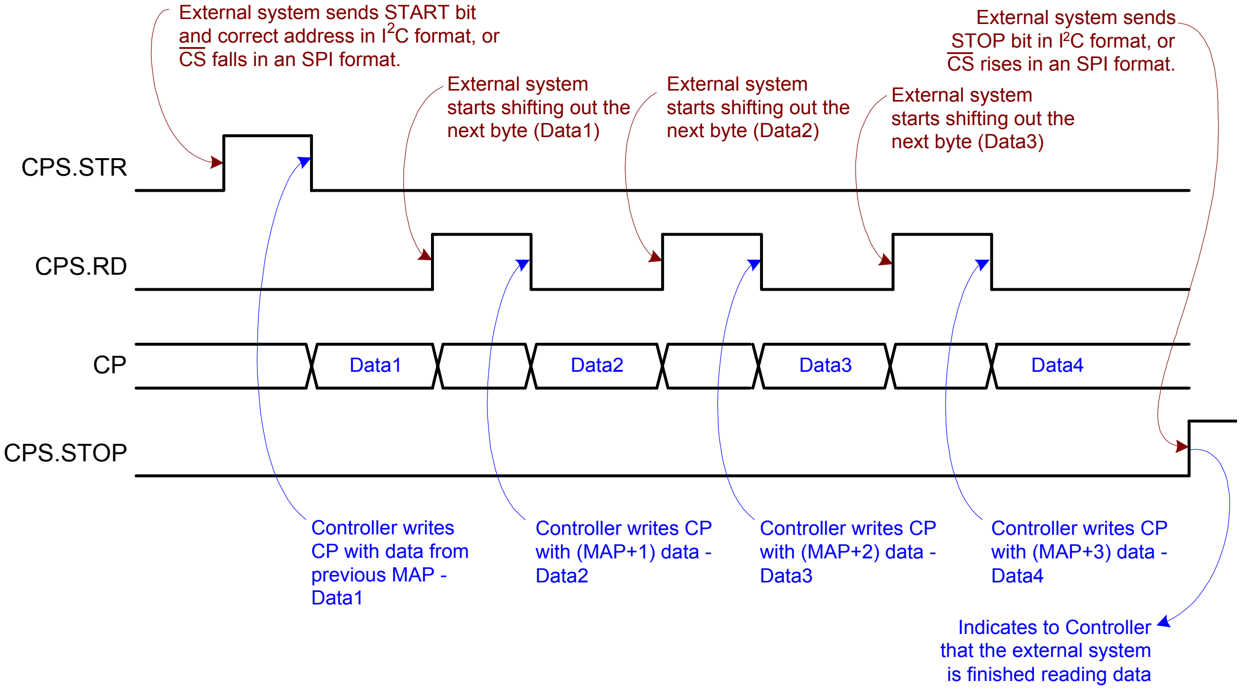

External system sends START bit and correct address in I2C format, or CS falls in an SPI format.
External system sends STOP bit in I2C format, or CS rises in an SPI format.
External system starts shifting out the next byte (Data1)
External system starts shifting out the next byte (Data2)
External system starts shifting out the next byte (Data3)
CPS.STR
CPS.RD
CP
Data1
Data2
Data3
Data4
CPS.STOP
Controller writes CP with data from previous MAP - Data1
Controller writes CP with (MAP+1) data - Data2
Controller writes CP with (MAP+2) data - Data3
Controller writes CP with (MAP+3) data - Data4
Indicates to Controller that the external system is finished reading data

Figure 2-15: Control Port I^{2}C and OSPI Read Sequence (slave)

For a read sequence (illustrated in Figure 2-15), the System should always precede the read by a write to the memory address pointer. This write is accomplished as describe above, but without the actual data bytes (just the MAP byte). Then the System sends another **START** bit for the I^{2}C format, or cycles the $\overline{\text{CS}}$ pin for the OSPI format, either of which causes the **CPS.STR** bit to go high again. The Controller should respond to this as a new sequence. The System then sends the address byte again, but this time the LSB is a 1, indicating a read sequence. As with the write sequence, the Controller does not know yet whether this sequence is a write or a read, so the Controller should get the data pointed to by the previously written MAP, and store that data into the Control Port data register CP. The Controller should then increment MAP to be ready for another read request. After sending the address byte with R/W bit, the System starts shifting out the next byte. This causes the CP register to be loaded into the Control Port shift register, to be shifted out of the chip. This transfer between the CP and the shift register, causes the **CPS.RD** bit to go high again, which tells the Controller that CP is ready for another byte. The Controller then reads the (MAP+1) location and stores it in CP. Then the Controller should increment MAP. This sequence continues as long as **CPS.RD** goes high. When the System finishes getting all the data it needs, it sends a **STOP** bit for the I^{2}C format, or raises the $\overline{\text{CS}}$ pin for the OSPI format. This causes **CPS.STOP** bit to go high signaling the Controller that this sequence is over. The Controller should write **CPS.STOP** = 0 to clear it for the next transfer.

2.3.3.1 I^{2}C Slave Format

When the I^{2}C format is selected (**CPS.I2CF** set or pull-up on **SCL/SCLK** pin), the **AD1** and **AD0** pins select the lower two address bits for the I^{2}C port with the upper five bits hard-coded to 01000. If configured as a slave device and the external system sends an address that matches the programmed chip address, the **CPS.STR** bit goes high (which can interrupt the Controller) indicating that a correct address has been received. The R/W bit, in the address byte, indicates to the Control Port hardware how the **CPS.RD** and **CPS.WR** should respond. Figure 2-16 illustrates the write sequence for data transferred across the **SDA** pin.

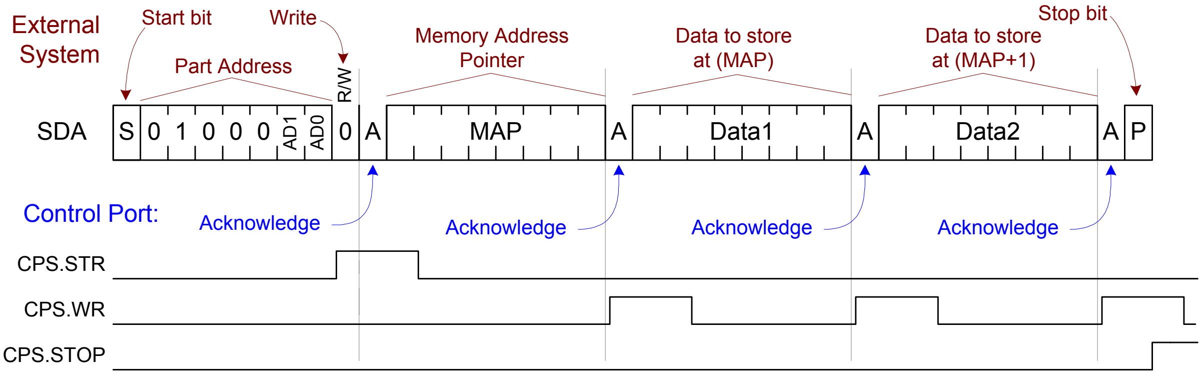


Figure 2-16: Control Port I^{2}C Slave Write Sequence

If the external system is clocking the data too fast, the Control Port hardware will stretch the clock (**SCL** pin) while **CPS.STR** is high, or if **CPS.WR** remains high and a new byte is in the shift register waiting to be transferred to the CP register. This prevents the external system I^{2}C master from continuing until **SCL** is released. Prior to an initial start bit and address byte, the **CPS.STOP** bit will be set when a stop bit is seen on the bus (even though the chip has not been addressed yet). Once a start bit and address byte is seen by the chip, the **CPS.STOP** bit will only respond after the address byte matches the chip address.

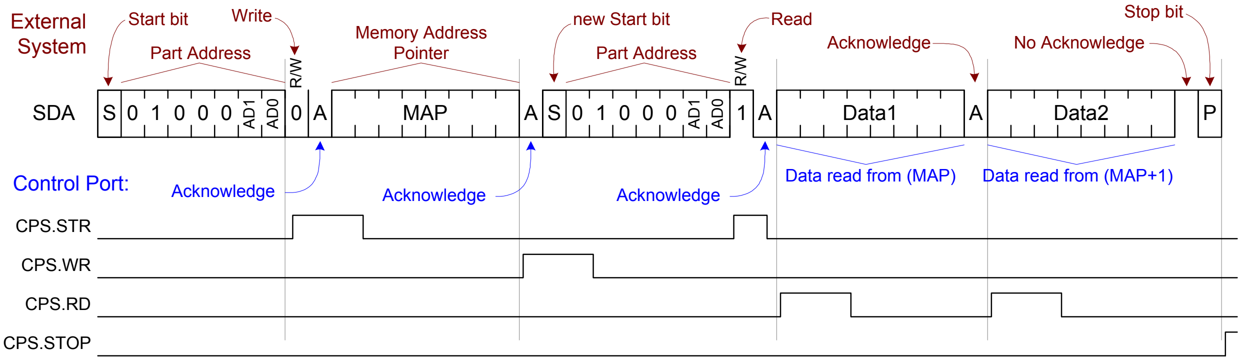


Figure 2-17: Control Port I²C Slave Read Sequence

2.3.3.1.1 I²C Master Mode

The Control Port is a serial port master when **CPS.CPMM** is set. Multi-master I²C format is not supported. As the serial port master, clock generation is controlled by the OS8805 (although slave peripherals can stretch the clock if needed). Once in master mode, **CPS.FAST** determines the Ports clocking frequency. When in master mode, an additional clock pulse on SCL can occur before the start bit is generated. Since slave devices are looking for a start bit, this extra clock should have no adverse effect.

The following is a brief description of how the Host Controller software interacts with the Control Port in I²C master mode:

- To start a transfer, the **CPS.STR** bit is set, causing the hardware to generate an I²C start bit on the bus.
- When the start condition is actually generated on the bus, **CPS.STR** will be set; thereby generating an interrupt.
- In the ISR, **CPS.STR** is written to zero, and the first byte (containing the I²C peripheral address and read/write bit) is written to the CP register. A write to the CP register causes the generation of 9 SCL clock pulses and the first byte will be shifted out on the SDA line.
- A **CPS.RD**, **CPS.WR**, or **CPS.NACK** interrupt (depending on the read/write bit in the first byte and whether the slave acknowledges the transfer or not) will occur after the 9 SCL clock cycles. Additional read/write operations can occur by reading or writing the CP register, which clears the **CPS.RD** or **CPS.WR** bits, thereby clearing the interrupt condition.
- To terminate a write transfer, writing **CPS.WR** to zero without writing the CP register clears the interrupt condition.
- To generate a Stop bit on the bus, **CPS.STOP** should be set. When the stop bit is actually driven on the bus, a stop interrupt will occur and the software should clear the **CPS.STOP** bit.
- To terminate a read transfer, the **CPS.NACK** bit should be set immediately after the reading of the second to last byte to stop the Control Port hardware from generating an acknowledge bit.
- When the last byte is shifted in, the read flag will not be set (**CPS.RD** = 0) and a no-acknowledge interrupt (**CPS.NACK**) will occur. The CP register must be read before the **CPS.NACK** bit is cleared, to prevent the generation of another 9 SCL clock pulses.
- To generate a Stop bit on the bus, **CPS.STOP** should be set. When the stop bit is actually driven on the bus, a stop interrupt will occur and the software should clear the **CPS.STOP** bit

In I^{2}C master mode, the status bits **STR**, **STOP**, and **NACK** are NOT the same bits when read and written. Therefore, reading, masking, and writing should be done with extreme caution. Writing these bits control the master-mode port operation; whereas, reading these bits indicate the status on the I^{2}C bus. For example, writing **NACK** to one causes the Control Port to NOT generate clocks when reading CP. But reading **NACK** = 1, indicates that a “no-acknowledge” condition occurred on the bus.

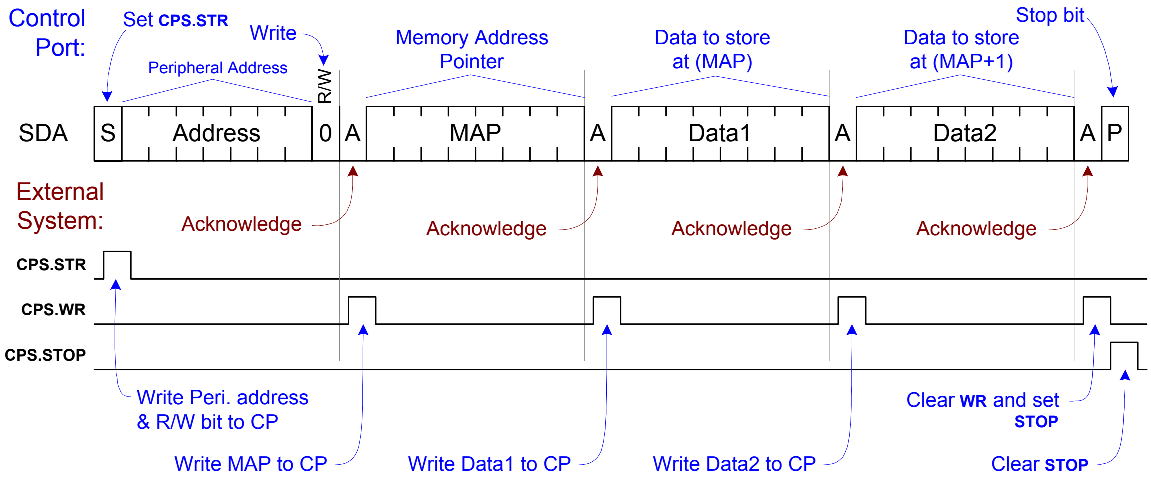


Figure 2-18: Control Port I^{2}C Master Write Sequence

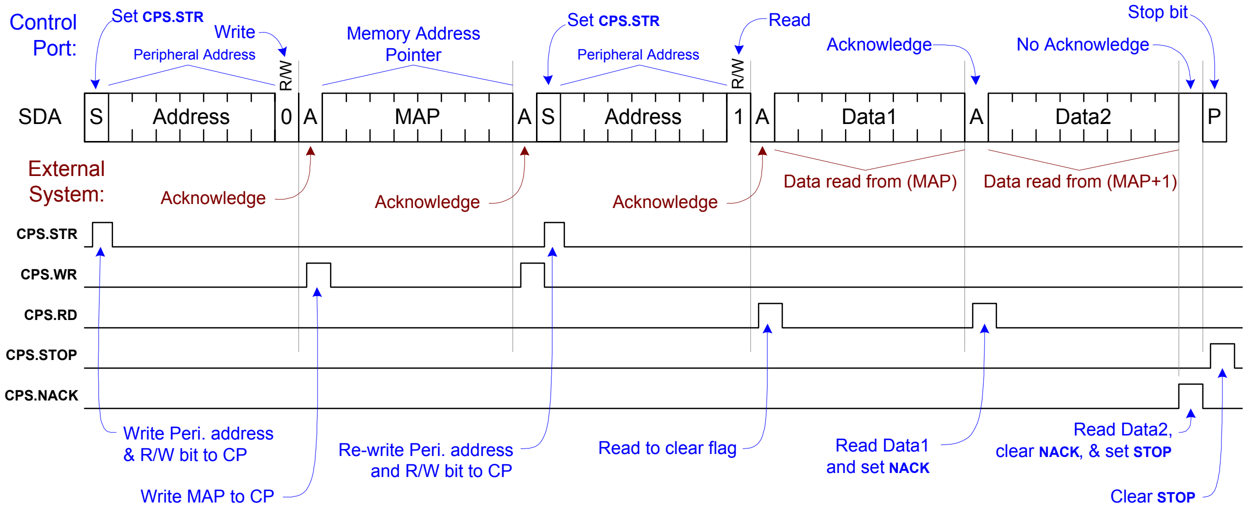


Figure 2-19: Control Port I^{2}C Master Read Sequence

2.3.3.2 Oasis-Specific Slave SPI Format

The Oasis-specific SPI format (OSPI) follows the same data format as the I^{2}C format, where a 7-bit address is sent (actual address is arbitrary) followed by a R/W bit. Then the data should continue to follow the I^{2}C format of MAP followed by write data, if any. However, the hardware pins used follow standard SPI conventions, with a chip select $\overline{\text{CS}}$, serial data in **SDIN**, separate serial data out **SDOUT**, and a serial clock **SCLK**. The OSPI format is selected at start-up by placing a pull-down on the **SCL/SCLK** pin. The Control Port can also be configured after start-up for OSPI by clearing **CPS.I2CF** and clearing **CPS.GSPI**. The advantage of OSPI over the generic SPI (GSPI) is that the **CPS.RD** and **CPS.WR** bits work properly in OSPI mode, and the *MAP followed by data* format is similar to the COM Port format used for inter-processor communications (allowing the sharing of code). Master mode is not supported for this format (**CPS.CPMM** clear). When the Control Port is operating in an SPI slave format, **IOA3** must be configured as an input (**EDD1.GPOA3** clear).

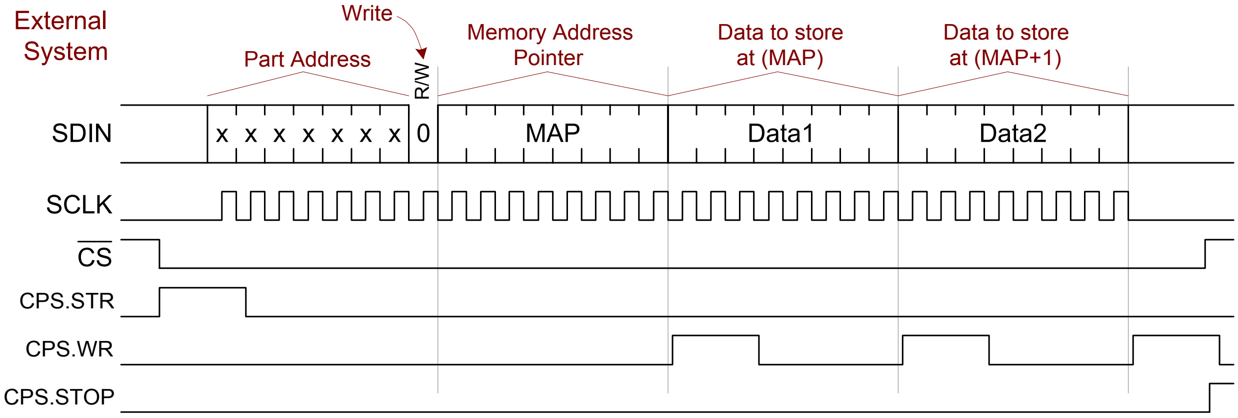


Figure 2-20: Control Port OSPI Write Sequence

The **CPS.STR** bit is set when the $\overline{\text{CS}}$ pin goes low, and is cleared when the Controller writes the CP register. The address, in the first byte sent by the external system, is arbitrary and not checked by the Control Port hardware. In the address byte, only the R/W bit has any significance, and determines how the Control Port **RD** and **WR** bits respond. If the OSPI slave device does not have enough time (or does not respond) between $\overline{\text{CS}}$ going low and the first byte shifting out, the first byte received by the OSPI master will be old data. Since the SPI format has no provision for stretching the clock, interrupt latencies should be minimized, and external system code should be written with this issue in mind. When the external System drives $\overline{\text{CS}}$ high, the **CPS.STOP** bit rises indicating the end of a transfer. An OSPI write sequence is illustrated in Figure 2-20. The **SCLK** polarity and phase are fixed as depicted in the Figure (in the GSPI format, **SCLK** is configurable). An OSPI read sequence is illustrated in Figure 2-21.

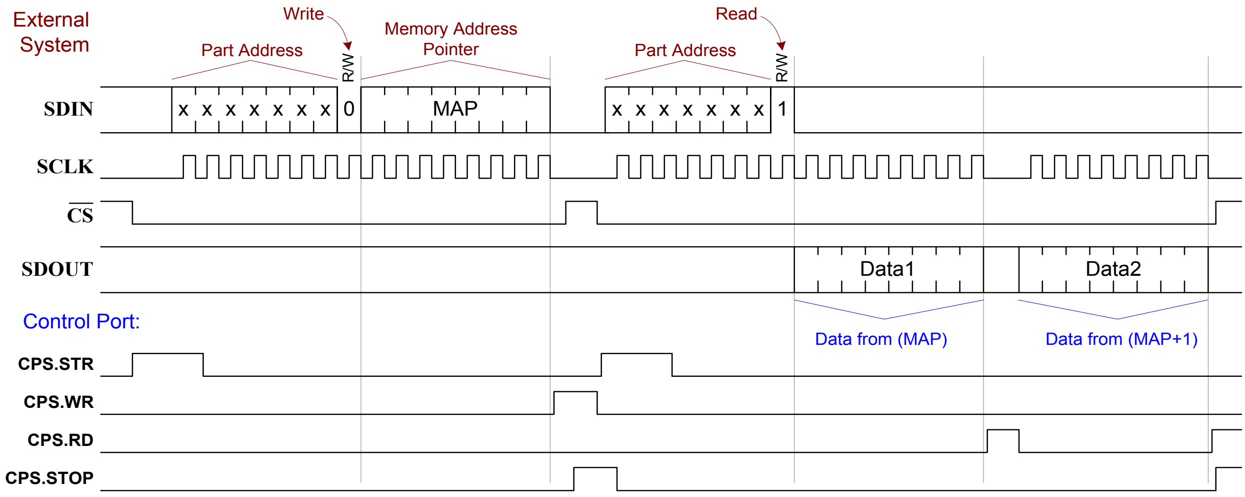


Figure 2-21: Control Port OSPI Read Sequence

2.3.3.3 Generic Slave SPI Format

In the Generic SPI format (GSPI), no interpretation of the data is made (no R/W bit); therefore, the **CPS.WR** and **CPS.RD** bits are not useful. The GSPI format can only be selected after start-up by clearing the **CPS.I2CF** bit and setting the **CPS.GSPI** bit. The Control Port can be configured at startup for the OSPI format by placing a pull-down on the **SCL/SCLK** pin.

When configured as a slave device (**CPS.CPMM** clear) and the $\overline{\text{CS}}$ pin goes low, the **CPS.STR** bit is set, and is cleared when the Controller writes the CP register. The **CPS.STOP** bit is set when the $\overline{\text{CS}}$ pin goes high and is cleared when the Controller writes the **STOP** bit to zero. If **CPS.CPHA** is clear, then $\overline{\text{CS}}$ must be brought high between each byte transferred. Since the SPI format has no provision for stretching the clock, interrupt latencies should be minimized, and external system code should be written with this issue in mind. A GSPI data transfer is illustrated in Figure 2-22. Since **CPS.CPHA** is set in this figure, $\overline{\text{CS}}$ can remain low for the duration of the transfer. When the Control Port is operating in an SPI slave format, **IOA3** must be configured as an input (**EDD1.GPOA3** clear).

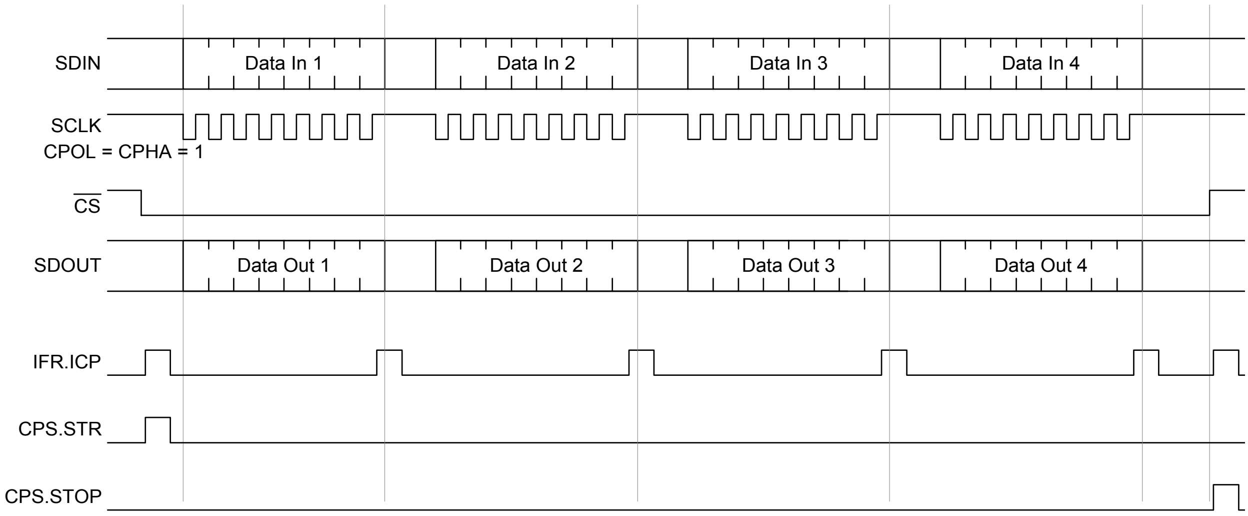


Figure 2-22: Control Port Generic SPI Sequence (CPHA set)

Figure 2-23 illustrates a transfer where **CPS.CPHA** is clear, and $\overline{\text{CS}}$ must be toggled between each byte transferred. One to three interrupts can occur after each byte transferred, based on how fast the external system raises and lowers $\overline{\text{CS}}$, relative to the last bit clocked in.

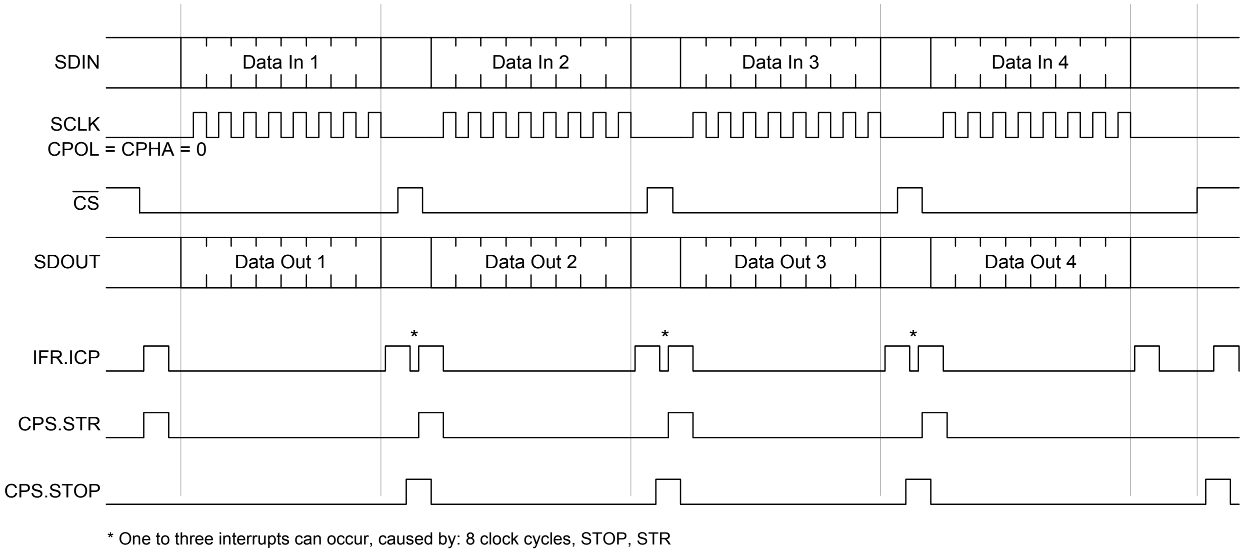


Figure 2-23: Control Port Generic SPI Sequence (CPHA clear)

The relationship between SCLK and the data can be programmed using two bits in the CPS register: **CPOL** and **CPHA**. SCLK must be stable and the correct polarity, as illustrated in Figure 2-24, when $\overline{\text{CS}}$ transitions.

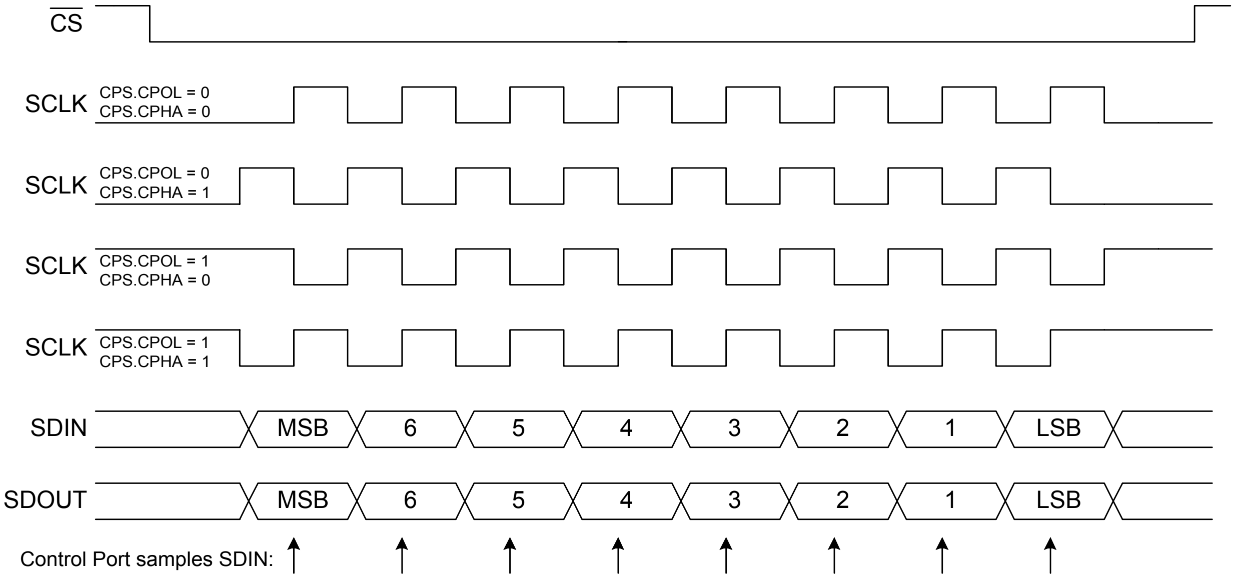


Figure 2-24: Control Port Generic SPI SCLK Format

In GSPI format, an internal counter counts 8 SCLK bit periods and then sets the Control Port interrupt bit **IFL.ICP**. Reading or writing CP clears this interrupt. The counter is reset when $\overline{\text{CS}}$ is high. The Controller can use **IFL.ICP** to service the Control Port. The data protocol for this format is left entirely to the user. If polling is desired, Controller software can be written to logically OR the **STR**, **RD** and **WR** bits in the CPS register, and when found high, the Control port needs servicing. Using the **RD** and **WR** bits in this manner requires the Controller write CP each time the Control Port is serviced.

2.3.3.3.1 GSPI Master Mode

The Control Port is a serial port master when **CPS.CPMM** is set. As the serial port master, clock generation is controlled by the OS8805. Once in master mode, **CPS.FAST** determines the Ports clocking frequency. The $\overline{\textbf{CS}}$ pin is not used and is available, as EGPIO, for chip select signal of the external device, if desired. To support single devices, the **IOA3** pin (same pin as $\overline{\textbf{CS}}$) can be configured as an output to drive the external device. The **IOA3** pin is configured as an output by setting **EDD1.GPOA3**. Then **EGPD1.GPDA3** controls the **IOA3/**$\overline{\textbf{CS}}$ output pin. If multiple external peripheral devices exist (multiple GPIOs).

In GSPI master mode, the Start and Stop bits are not active (constantly low). When transferring bytes (reads or writes), the **CPS.RD** and **CPS.WR** bits are not useful; however, **IFL.ICP** is set after every 8 clocks are generated, indicating a completed transfer. If **IER.IECP** is set, an interrupt is generated, when **IFL.ICP** is set.

In GSPI format, the SPI port is always full duplex and does not classify an operation as a read or write (no bit interpretation as with I^2C). In every transfer, 8-bits are shifted out the shift register onto the **SDOUT** pin, while 8-bit are shifted in the shift register via the **SDIN** pin. Since reading CP and writing CP each generates 8 clocks, when used for full-duplex operation **CPS.NACK** should be set (and not cleared), which will inhibit the generating of 8 clock pulses when CP is read. Only writing CP will generate eight clocks. Software would then read CP to get the data from the peripheral, and then write CP, which would cause the 8 clocks to be generated, swapping another byte with the external peripheral.

If only reading the external peripheral, then **CPS.NACK** should be clear. Then writing CP would start a transfer to the peripheral and the peripheral would send a byte. Next, reading CP would retrieve the byte and cause another 8 clocks to be generated, which would transfer the next byte. Before reading the last byte from CP, **CPS.NACK** should be set. Then reading CP will not cause any more clocks to be generated. Lastly, the chip select signal (**GPIO** pin) would be set high.

If only writing to the external peripheral, the first step is to bring the chip select for the external device low. Then writing CP would load the byte into a shift register and generate the 8 clocks needed to transmit the byte. On the next interrupt, the next byte would be loaded into CP. When the last byte is loaded into CP, the software should wait until the following interrupt (to allow the previous byte to be fully transmitted), then bring the chip select high. This interrupt can be cleared by setting **CPS.NACK** and then reading CP. Lastly, **CPS.NACK** should be cleared by writing it to zero (it will read zero since the value read is status from the SPI port, and SPI mode never generates NACK signals). Since the $\overline{\textbf{CS}}$ pin is ignored by the Control Port, and writing or reading CP will cause clocks to be generated, software timing should be considered to guarantee that the “chip-select low to SCLK changing state” time for the external device is met.

2.3.4 Debug Port

The Debug Port provides a secondary port (besides the Control Port) for the external system to access the Controller. Although listed as a Debug Port, this interface is generic. When enabled, the Debug Port uses the EGPIO **IOB[3:0]** port. The Debug Port is enabled by default; however, the Controller can disable the Debug Port by setting the **DCPS.DPFD** bit, and not enabling USART1 (**U1C.USEN** clear), which configures the EGPIO **IOB[3:0]** port as extended general purpose I/O pins.

The Debug Port operation and control is very similar to the Control Port. The major differences are that the Debug Port can only set the LSB of the address in I^{2}C format, and the Debug Port has its own interrupt vector (doesn't use the IFL register) at memory location 0x0000C. In addition, when the Debug Port is configured for USART format, it connects to USART1. The Debug Port can operate the serial port as a master or slave device for all formats except OSPI, which is slave only. When configured as an GSPI master, the $\overline{\textbf{DCS}}$ pin can be controlled via the **IOB3** EGPIO control, if only one chip select is needed. The Debug Port interrupt vector has a higher priority than the Control Port interrupt vector. Other than these differences, the Debug Port operates similar to the Control Port. See the *Control Port* description for more details on the three Port formats and their differences. The USART format and register interface are described in the *USARTs* Section.

When the Debug Port is operating in a slave format, **IOB3** must be configured as an input (**EDD1.GPOB3** clear).

When $\overline{\textbf{RST}}$ is de-asserted or at initial power-up, the Debug Port format is selected by a pull-up (I^{2}C format) or pull-down (SPI format) on the **DSCL/DSCLK** pin. The Debug Port format can also be changed after power-up via the **DCPS.I2CF** bit. When the I^{2}C format is selected (**DCPS.I2CF** set or pull-up on **DSCL/DSCLK** pin), the **DAD0** pin selects the least significant address for the I^{2}C port with the upper six bits hard-coded to 010001. When the **DSCL/DSCLK** pin is pulled high at startup (default I^{2}C format), the **DSDA** pin will be driven low for one 128Fs period immediately on exiting the reset condition. This produces a start/stop condition on the bus; however, no address is output.. Data is always transferred MSB first on the data lines (**DSDA** for I^{2}C, and **DSDIN/DSDOUT** for SPI format).

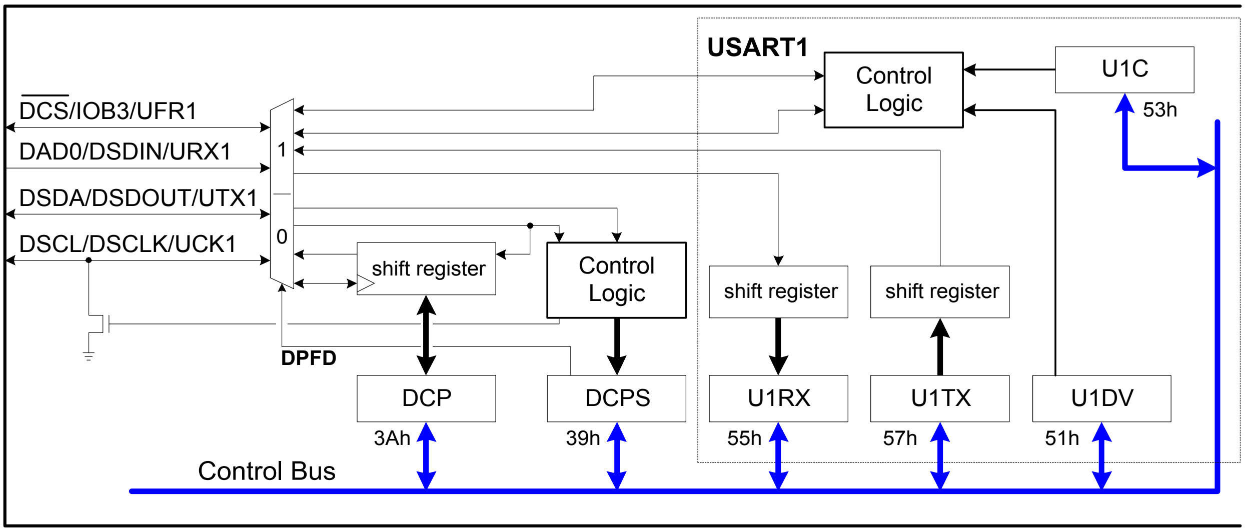


Figure 2-25: Debug Port

Program control is diverted to the Debug Port interrupt vector (location 0x0000C) whenever any of the status bits (DCPS bits **STOP**, **STR**, **WR**, **NACK**, **RD**) goes high. If the debug interrupt is enabled (**IER.DIEN** set), the debug interrupt service routine must clear the condition causing the interrupt (clear the DCPS bit as described below), before exiting the ISR.

39h DCPS Debug Port Status Host

Bit	Label	Description	Default
15..12	rsvd	Reserved, Write to 0	0000
11	DPFD	Debug Port Formats Disable	0
10	FAST	Fast Mode enable	0
9	NACK	No Acknowledge	0
8	DPMM	Debug Port Master Mode. (must be clear in OSPI format)	0
7	STOP	Stop Condition (not valid in OSPI mode)	0
6	I2CF	0 in SPI mode, 1 is I^{2}C mode	*
5	GSPI	0 is Oasis-specific SPI mode (OSPI), 1 is generic SPI mode (GSPI)	0
4	CPOL	For generic SPI mode, polarity of clock	0
3	CPHA	For generic SPI mode, phase of clock to data	0
2	STR	Start transfer (read-only when DPMM clear)	0
1	WR	Write data available	0
0	RD	Read data request	0

* Initial state determined by **DSCL/DSCLK** pin at power-up.

Table 2-37: DCPS Register

DPFD Debug Port Formats Disable. When clear, the Debug Port formats listed in this section are enabled. If set, the Debug Port formats are disabled and the Debug Port can be configured for USART1 (**U1C.USEN** set) or the four pins can be used as EGPIO port **IOB[3:0]** (**U1C.USEN** clear).

FAST Fast mode enable. When the Debug Port is a master (**DPMM** set), **FAST** determines the clocking serial port clocking frequency as shown below:

With **FAST** clear the approximate clocking frequency is 88.2 kHz when Fs = 44.1 kHz, and 96 kHz when Fs = 48 kHz.

With **FAST** set, the approximate clocking frequency is 295.8 kHz when Fs = 44.1 kHz, and 310.6 kHz when Fs = 48 kHz.

NACK No Acknowledge. Reading this bit does not read what was written. The read bit going high can generate an interrupt, if enabled. Writing **NACK** to zero clears the interrupt condition.

When the Debug Port is configured as a slave port (**DPMM** clear):

NACK is set when the external master device is performing a read operation and does not acknowledge the transfer. Software can write this bit to zero **after** reading DCP. When configured as a slave port, software should never write **NACK** to one.

When the Debug Port is configured as a master port (**DPMM** set):

NACK is set when writing an external device and it does not acknowledge the transfer. In I^{2}C format, software should set **NACK** to one to keep the Debug Port from acknowledging the next (last) transfer. Once the “no acknowledge” is transmitted, **NACK** will read back one and must be cleared in software. The data in DCP must be read before clearing **NACK** or another eight clocks will be sent out. In GSPI format, **NACK** should be set for full-duplex communications, since it will keep DCP reads from generating 8 clocks (only writes will generate 8 clocks).

DPMM Debug Port Master Mode enable. When clear, the Debug Port is a slave port. When set, the Debug Port is a master port and **FAST** selects the clocking speed used. The OSPI format is not supported. In addition, I^{2}C multi-master format is not supported.

STOP

When reading, **STOP** set indicates a stop condition has been detected on the bus; whereas writing to 0 clears the bit and writing to 1 (when in I^2C master mode) causes a Stop condition to be transmitted. If a **STOP** condition is detected on the bus, an interrupt can be generated. The **STOP** bit is set at power-up and when switching to master mode, and should be cleared before enabling the port. The **STOP** bit also respond to any bus stop bits until the first start bit and address is sent. Then the **STOP** bit only responds after the correct address is sent.

When the Debug Port is configured as a slave port (**DPMM** clear):

In I^2C format, **STOP** is set when an I^2C Stop bit is detected on the bus. In either SPI format, **STOP** high indicates that the $\overline{\text{DCS}}$ pin rose. **STOP** is cleared by writing **STOP** to zero. When configured as a slave port, software should never write **STOP** to one.

When the Debug Port is configured as a master port (**DPMM** set):

In I^2C format, **STOP** is set only after an I^2C Stop bit is detected on the bus. In GSPI format, **STOP** is always low. **STOP** is cleared by writing **STOP** to zero. In I^2C format, writing **STOP** to one causes a Stop bit to be transmitted onto the bus. Once the bit is transmitted, **STOP** will be read as one, indicating a Stop condition was seen on the bus.

I2CF

SPI or I^2C format. When set, the Debug Port uses the I^2C format. When clear, the Debug Port uses the SPI format indicated by the **GSPI** bit. **I2CF** is initially set by the value of the **DSCL/DSCLK** pin when $\overline{\text{RST}}$ is de-asserted or at initial power-up.

GSPI

Generic SPI. When set, and **I2CF** is clear, the Debug Port interface uses a generic SPI format (GSPI). When GSPI is clear (Oasis-specific SPI, OSPI), the data format matches the I^2C format and the **WR** and **RD** bits work as in I^2C format. The Debug Port must be a slave port when configured for OSPI mode.

CPOL

Clock Polarity, in generic SPI format only. When GSPI is set and **I2CF** is clear, this bit determines the **SCLK** clock polarity as indicated in Figure 2-24.

CPHA

Clock Phase, in generic SPI format only. When GSPI is set and **I2CF** is clear, this bit determines the **DSCLK** clock phase as indicated in Figure 2-24. When in slave mode and **CPHA** is clear, $\overline{\text{DCS}}$ must be brought high between each data byte transferred.

STR

Start. Reading indicates if a start condition has been detected on the bus. When in master mode, writing to 0 clears the bit and writing to 1 causes a Start condition. If a **STR** condition is detected on the bus, an interrupt can be generated in I^2C and GSPI-slave formats.

When the Debug Port is configured as a slave port (**DPMM** clear):

In I^2C format, **STR** is set when an I^2C Start bit is detected on the bus, followed by an address matching 010001z, where z is set via the **DAD0** pin. In both SPI formats, **STR** going high indicates that the $\overline{\text{DCS}}$ pin fell. **STR** is cleared by writing to the DCP register (**STR** is read-only and cannot be cleared by writing to 0). Writing **STR** to 1 has no effect.

When the Debug Port is configured as a master port (**DPMM** set):

In I^2C format, **STR** is set only after an I^2C Start bit is detected on the bus. In GSPI format, **STR** is always low. **STR** is cleared by writing **STR** to zero. In I^2C format, writing **STR** to one causes a Start bit to be transmitted onto the bus. Once the bit is transmitted, **STR** will be read as one, indicating a Start condition was seen on the bus.

WR

Write data bit. When the Debug Port is not in generic SPI mode (**GSPI** clear):

When in slave mode, **WR** is set when data is transferred from the serial input shift register to the DCP register; and **WR** is cleared when the Controller reads the DCP register or explicitly writes **WR** to 0. Writing **WR** to 1 has no effect.

When in master mode, **WR** is set when data is transferred from DCP to the serial shift register, to be shifted out; and **WR** is cleared when the Controller writes DCP with the next word, or explicitly writes **WR** to 0.

RD Read data bit. When the Debug Port is not in generic SPI mode (**GSPI** clear):

When in slave mode, **RD** is set when data is transferred from the DCP register to the shift register, to be shifted out of the chip; and **RD** is cleared when the Controller writes the DCP register or explicitly writes **RD** to 0. Writing **RD** to 1 has no effect.

When in master mode, **RD** is set when data is transferred from the shift register to the DCP register, and **RD** is cleared when the Controller reads DCP or explicitly writes **RD** to 0.

3Ah DCP Debug Port Data Host

Bit	Label	Description	Default
15..8	rsvd	Reserved, Write to 0	00h
7..0	D[7:0]	data, bit 7 is MSB	00h

Table 2-38: DCP Register

D[7:0] Debug Port bi-directional data register when the Debug Port is enabled and in any format except USART1.

When the Debug Port is configured as a slave port (**DCPS.DPMM** clear):

Reading or writing the DCP register after **DCPS.WR** or **DCPS.RD** is set, clears the particular status bit, clearing the interrupt condition.

When the Debug Port is configured as a master port (**DCPS.DPMM** set):

Writing the DCP register causes 8 clocks to be transmitted in GSPI and 9 clocks in I^{2}C. Reading the DCP register also causes 8 clocks to be transmitted in GSPI and 9 clocks in I^{2}C format. However in I^{2}C format, **DCPS.NACK** should be written immediately after reading the second-to-last transfer from DCP. This will cause the part to not-acknowledge the last transfer. Once the last byte is in DCP, it must be read BEFORE clearing **DCPS.NACK** (or another 9 clocks will be transmitted). For GSPI format, if **DCPS.NACK** is set before reading the last word in DCP, no more clocks will be generated. Then DCP should be read, followed by clearing **DCPS.NACK**.

See the *Control Port* Section for an functional description of how the Debug Port operates. The differences are that the Debug port can only set the LSB of the port address in the I^{2}C format, the interrupt vector is unique for the Debug Port, the registers are DCPS and DCP instead of CPS and CP, and the pins used have the same name as the Control Port with a ‘D’ prefix attached. In addition, the Debug Port is attached to USART1; whereas the Control Port is attached to USART0. Since the interrupt is non-maskable and the Debug Port pins powering-up can generate initial conditions, (generally the **DCPS.STOP** bit is set) a Debug Interrupt Routine is required, even if the Debug Port is unused. The following is an example of a dummy interrupt routine that will clear out any initial interrupt flags that get set.

```
porg (0x09);Debug Interrupt Vector
jmp Dummy_Debug_Int_Routine
. . .

//-------------------------------------
// Dummy Debug Interrupt Routine
//-------------------------------------
Dummy_Debug_Int_Routine:
  *sp = acc;           // push ACC on stack
  sp += 2;

  acc = sr;            // push SR on stack
  *sp = acc;

  sb;                  // set byte mode
  acc = (int) 0x0078;  // set zeros to mask out all status bits
  dcps &= acc;         // clear all four status bits in Debug Port Status register

  acc = *sp;           // restore SR
  sr = acc;
  sp -= 2;             // restore ACC
  acc = *sp;
  ret;                 // End of Debug Port Interrupt
//-------------------------------------
```

2.3.5 USARTs

The Control Port and the Debug Port can be configured for USART operation. USART0 is connected to the Control Port and USART1 is connected to the Debug Port. Through a programmable divider, standard BAUD rates are supported (1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400).The clock source for the USARTs is configurable, based on the **UnC.CKSL[1:0]** bits: 256xFs, 384xFs, 512xFs. For timing-slave devices, the entire chip is frequency locked to the Network Fs; therefore, the BAUD rate generated is relative to Fs. If the chip is out-of-lock, the BAUD rate will drift with the PLL, which can cause errors when the USART is configured for asynchronous operation (assuming the device connected to the USART is not also frequency locked to Fs).

The USART can be configured for asynchronous or synchronous operation. Asynchronous operation supports BAUD rates of up to 230400; whereas, synchronous operation supports BAUD rates up to 1 Mbit. The synchronous format uses start and stop bits to delineate the bytes.

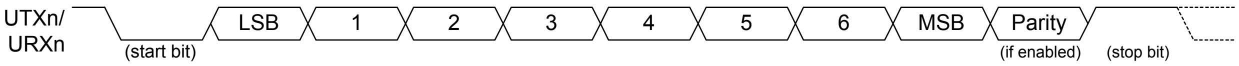


Figure 2-26: USART Asynchronous Format (UnC.SYEN = 0)

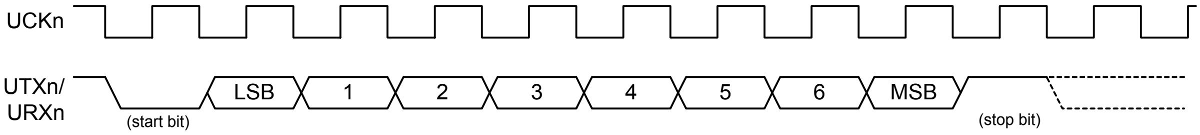


Figure 2-27: USART Synchronous Format

Regardless of whether in synchronous or asynchronous formats, software must read the data out of **UnRX.URD[7:0]** before the next byte is fully received, or an overrun error (**UnRX.OE**) will occur. The default, no data, state for the **URXn** an **UTXn** pins is the mark state, where the actual pins are high. An all-space state, where the pins are consistently low, denotes a USART break condition.

52h U0C USART0 Configuration (Control Port) Host

Bit	Label	Description	Default
15, 14	rsvd	Reserved. Write to 0	00
13	TXI	Transmit status	0
12	RXI	Receive status	0
11	TXEN	Transmit Interrupt Enable	0
10	RXEN	Receive Interrupt Enable	0
9	SYTX	Synchronous Transmit signal.	0
8	USEN	USART0 Enable	0
7, 6	CKSL[1:0]	Clock Select: 00 - 256Fs, 01 - 384Fs, 10 - 512Fs	00
5	CKMTR	When in Synchronous Mode, enables synchronous clock output on the **UCK0** pin.	0
4, 3	PMD[1:0]	Parity Mode: 00 - even, 01 - odd, 10 - space, 11 - mark	00
2	PEN	Parity Enable: 0 - no parity bit, 1 - parity enabled	0
1	SYSL	Synchronous select:	0
0	SYEN	Synchronous enable: 0 - asynchronous, 1 - synchronous mode	0

Table 2-39: U0C Register

TXI Transmit Status. When set, The **U0TX.UTD[7:0]** bits are free to be loaded with the next byte to transmit. **TXI** can be cleared by writing **U0TX**. **TXEN** set can also set **IFL.ICP** and can generate an interrupt if **IER.IECP** is set. When initially coming out of reset, the **TXI** bit will be clear even though the **U0TX.UTD[7:0]** bits will be empty. After the initial data is written to **U0TX.UTD[7:0]**, and the UART starts transmitting the data, **TXI** will be set.

RXI Receive Status. When set, The **U0RX.URD[7:0]** bits contain a received byte. **RXI** can be cleared by reading **U0RX**. **RXEN** set can also set **IFL.ICP** and can generate an interrupt if **IER.IECP** is set.

TXEN Transmit Interrupt Enable. When set, and **USEN** is set, the USART0 transmit port empty bit **TXI** sets **IFL.ICP** and can generate an interrupt if **IER.IECP** is set. When clear, the transmit Interrupt will not set **IFL.ICP** and **TXI** can be used in a polling fashion.

RXEN Receive Interrupt Enable. When set, and **USEN** is set, the USART0 receive port full bit **RXI** sets **IFL.ICP** and can generate an interrupt if **IER.IECP** is set. When clear, the receive Interrupt will not set **IFL.ICP** and **RXI** can be used in a polling fashion.

SYTX Synchronous Transmit signal. Must be 0.

USEN USART0 enable. When set, and the regular Control Port formats are disabled (**CPS.CPFD** set), USART0 is enabled. When clear, USART0 is disabled and the pins can be used as **IOA[3:0]** if **CPS.CPFD** is set.

CKSL[1:0] Clock Select. Selects the clock source used for the divider chain U0DV.
00 - 256Fs
01 - 384Fs
10 - 512Fs
11 - Reserved.

CKMTR Clock Master. When in synchronous mode (**SYEN** set), **CKMTR** set enables the synchronous clock (set by U0DV) to be output on **UCK0**. When **CKMTR** is clear (clock slave), **UCK0** is an input for the synchronous clock, and the data direction bit for **IOA0** (**EDD1.GPOA0**) must be clear (set to input).

PMD[1:0] Parity Mode. Only valid when parity is enabled (**PEN** set) and in asynchronous mode (**SYEN** clear).
00 - Even Parity. Number of transmitted ones is even.
01 - Odd Parity. Number of transmitted ones is odd.
10 - Space. Parity is always 0.
11 - Mark. Parity is always 1.

PEN Parity Enable. When set and in asynchronous mode (**SYEN** clear), parity is enabled and a parity character is added between the last data bit and the stop bit. The type of parity is selected via **PMD[1:0]**.

SYSL Synchronous Select. Must be 0.
0 - One Start and Stop bit delineate each synchronous byte.

SYEN Synchronous Enable. When set, the data format is synchronous. **CKMTR** determines whether this device is the synchronous master or slave device. Synchronous data is output on the falling edge of **UCK0** and valid on the rising edge of **UCK0**. In synchronous mode, no parity is generated. When **SYEN** is clear, USART0 operates as a UART (asynchronously), and the **UCK0** and **UFR0** pins are free to used as EGPIO pins **IOA0** and **IOA3**.

50h U0DV USART0 Divider Host

Bit	Label	Description	Default
15..0	D[15:0]	16-bit clock divider. The value loaded in this register should be the desired divide value - 1.	0000h

Table 2-40: U0DV Register

D[15:0] The following table indicates the dividers needed to support standard BAUD Rates. The maximum BAUD rate for asynchronous operation is 230400, and the maximum BAUD rate for synchronous operation is 1000000 (1 MBAUD). The formula is **U0DV[15:0]** $= \frac{\text{CKSL} \times \text{Fs}}{\text{Target BAUD}} - 1$.

Target BAUD Rate	UnDV[15:0]					
	Fs = 44.1 kHz			Fs = 48 kHz		
	256xFs	384xFs	512xFs	256xFs	384xFs	512xFs
1200	24BFh	371Fh	497Fh	27FFh	3BFFh	4FFFh
2400	125Fh	1B8Fh	24BFh	13FFh	1DFFh	27FFh
4800	092Fh	0DC7h	125Fh	09FFh	0EFFh	13FFh
9600	0497h	06E3h	092Fh	04FFh	077Fh	09FFh
19200	024Bh	0371h	0497h	027Fh	03BFh	04FFh
38400	0125h	01B8h	024Bh	013Fh	01DFh	027Fh
57600	00C3h	0125h	0187h	00D4h	013Fh	01A9h
115200	0061h	0092h	00C3h	0069h	009Fh	00D4h
230400	0030h	0048h	0061h	0034h	004Fh	0069h

Table 2-41: UnDV Values for Standard BAUD Rates

54h U0RX USART0 Receive register Host

Bit	Label	Description	Default
15..13	rsvd	Reserved. Write to 0	000
12	RB	Receive Break	0
11	OE	Overrun Error	0
10	PE	Parity Error	0
9	FE	Framing Error	0
8	RDF	Receive Data Full (read-only)	0
7..0	URD[7:0]	USART Receive Data (read only)	00h

Table 2-42: U0RX Register

RB Received break. When set, indicates that a break was detected (all space). Cleared by writing **RB** to 0.

OE Overrun Error. When set, indicates that a new data byte was lost due to the **URD[7:0]** being full. This bit is sticky and cleared by writing to 0.

PE Parity Error. Only possible in Asynchronous mode, **U0C.SYEN** clear. When **PE** is set, indicates that parity was in error since the last time **PE** was cleared. Cleared by writing **PE** to 0.

FE Framing Error. **FE** set indicates a stop bit was not received at the expected bit time.

RDF When set, **URD[7:0]** contain a received data byte. Cleared by reading U0RX.

URD[7:0] USART0 received data. **RDF** set indicates new data is available.

56h U0TX USART0 Transmit register Host

Bit	Label	Description	Default
15..10	rsvd	Reserved. Write to 0	000000
9	TSB	Transmit Send Break	0
8	rsvd	Reserved. Write to zero.	1
7..0	UTD[7:0]	USART Transmit Data	00h

Table 2-43: U0TX Register

TSB Transmit Send Break. When set, the **UTX0** pin continually sends the space character. (Sets the **FE** and **RB** bits in receiving device.)

UTD[7:0] USART0 Transmit Data. **U0C.TXI** set indicates that new data can be loaded into **UTD[7:0]**.

53h U1C USART1 Configuration (Debug Port) Host

Bit	Label	Description	Default
15, 14	rsvd	Reserved. Write to 0	00
13	TXI	Transmit status	0
12	RXI	Receive status	0
11	TXEN	Transmit Interrupt Enable	0
10	RXEN	Receive Interrupt Enable	0
9	SYTX	Synchronous Transmit signal.	0
8	USEN	USART1 enable	0
7, 6	CKSL[1:0]	Clock Select: 00 - 256Fs, 01 - 384Fs, 10 - 512Fs, 11 - crystal	00
5	CKMTR	Enables synchronous clock output on the **UCK1** pin.	0
4, 3	PMD[1:0]	Parity Mode: 00 - even, 01 - odd, 10 - space, 11 - mark	00
2	PEN	Parity Enable: 0 - no parity bit, 1 - parity enabled	0

Table 2-44: U1C Register

Bit	Label	Description	Default
1	SYSL	Synchronous select:	0
0	SYEN	Synchronous enable: 0 - asynchronous, 1 - synchronous mode	0

Table 2-44: U1C Register (Continued)

TXI Transmit Status. When set, The **U1TX.UTD[7:0]** bits are free to be loaded with the next byte to transmit. **TXI** can be cleared by writing U1TX. If **TXEN** is set when **TXI** goes high, a Debug Interrupt will occur if **IER.DIEN** is set. When initially coming out of reset, the **TXI** bit will be clear even though the **U1TX.UTD[7:0]** bits will be empty. After the initial data is written to **U1TX.UTD[7:0]**, and the UART starts transmitting the data, **TXI** will be set.

RXI Receive Status. When set, The **U1RX.URD[7:0]** bits contain a received byte. **RXI** can be cleared by reading U1RX. If **RXEN** is set when **RXI** goes high, a Debug Interrupt will occur if **IER.DIEN** is set.

TXEN Transmit Interrupt Enable. When set (and **USEN** is set), the USART1 transmit port empty bit **TXI** causes a Debug Port interrupt if **IER.DIEN** is set. When clear, **TXI** going high does not cause a Debug Interrupt and can be used in a polling fashion.

RXEN Receive Interrupt Enable. When set, and **USEN** is set, the USART1 receive port full bit **RXI** causes a Debug Port interrupt if **IER.DIEN** is set. When clear, **RXI** going high does not cause a Debug Interrupt and can be used in a polling fashion.

SYTX Synchronous Transmit signal. Must be 0.

USEN USART1 enable. When set, and **DCPS.DPFD** set (Debug Port Formats disabled), enables USART1. When clear, USART1 is disabled and the pins can be used as **IOB[3:0]** if **DCPS.DPFD** is set.

CKSL[1:0] Clock Select. Selects the clock source used for the divider chain U1DV.
00 - 256Fs
01 - 384Fs
10 - 512Fs
11 - Reserved.

CKMTR Clock Master. When in synchronous mode (**SYEN** set), **CKMTR** set enables the synchronous clock (set by U1DV) to be output on **UCK1**. When **CKMTR** is clear (clock slave), **UCK1** is an input for the synchronous clock, and the data direction bit for **IOB0** (**EDD1.GPOB0**) must be clear (set to input).

PMD[1:0] Parity Mode. Only valid when parity is enabled (**PEN** set), and in asynchronous mode (**SYEN** clear).
00 - Even Parity. Number of transmitted ones is even.
01 - Odd Parity. Number of transmitted ones is odd.
10 - Space. Parity is fixed as a space.
11 - Mark. Parity is fixed as a mark.

PEN Parity Enable. When set and in asynchronous mode (**SYEN** clear), parity is enabled and a parity character is added between the data and the stop bit. The type of parity is selected via **PMD[1:0]**.

SYSL Synchronous Select. Must be 0.
0 - One Start and Stop bit delineate each synchronous byte.

SYEN Synchronous Enable. When set, the data format is synchronous. **CKMTR** determines whether this device is the synchronous master or slave device. Synchronous data is output on the falling edge of **UCK1** and valid on the rising edge of **UCK1**. In synchronous mode, no parity is generated. When **SYEN** is clear, USART1 operates as a UART (asynchronously), and the **UCK1** and **UFR1** pins are free to used as EGPIO pins **IOB0** and **IOB3**.

51h U1DV USART1 Divider Host

Bit	Label	Description	Default
15..0	D[15:0]	16-bit clock divider. The value loaded in this register should be the desired divide value - 1.	0000h

Table 2-45: U1DV Register

U1DV[15:0] See Table 2-41 for the divider needed to support standard BAUD rates. The maximum BAUD rate for asynchronous operation is 230400, and the maximum BAUD rate for synchronous operation is 1000000 (1 MBAUD). The formula is $\mathbf{U1DV[15:0]} = \frac{\text{CKSL} \times \text{Fs}}{\text{Target BAUD}} - 1$

55h U1RX USART1 Receive register Host

Bit	Label	Description	Default
15..13	rsvd	Reserved. Write to 0	000
12	RB	Receive Break	0
11	OE	Overrun Error	0
10	PE	Parity Error	0
9	FE	Framing Error	0
8	RDF	Receive Data Full (read only)	0
7..0	URD[7:0]	USART Receive Data (read only)	00h

Table 2-46: U1RX Register

RB Received break. When set, indicates that a break was detected (all space). Cleared by writing **RB** to 0.

OE Overrun Error. When set, indicates that a new data byte was lost due to the **URD[7:0]** being full. This bit is sticky and cleared by writing to 0.

PE Parity Error. Only possible in Asynchronous mode, **U1C.SYEN** clear. When **PE** is set, indicates that parity was in error since the last time **PE** was cleared. Cleared by writing **PE** to 0.

FE Framing Error. **FE** set indicates a stop bit was not received at the expected bit time.

RDF When set, **URD[7:0]** contain a received data byte. Cleared by reading U1RX.

URD[7:0] USART1 received data. **RDF** set indicates new data is available.

57h U1TX USART1 Transmit register Host

Bit	Label	Description	Default
15..10	rsvd	Reserved. Write to 0	000000
9	TSB	Transmit Send Break	0
8	rsvd	Reserved. Write to zero.	1
7..0	UTD[7:0]	USART Transmit Data	00h

Table 2-47: U1TX Register

TSB Transmit Send Break. When set, the **UTX1** pin continually sends the space character. (Sets the **FE** and **RB** bits in receiving device.)

UTD[7:0] USART1 Transmit Data. **U1C.TXI** set indicates that new data can be loaded into **UTD[7:0]**.

2.3.6 GPIO Pins and GP Timer

The GP Timer (not to be confused with the Global Timer flags GTR register) peripheral consists of pins that can output a value when the timer reaches a certain value, or pins that can capture a timer value when an external event occurs. Four of these pins can also generate interrupts (**GPD[3:0]**). When not using these pins for timer functions, they can be used as general purpose IO. The GPA through GPD GPIO ports differ from the **EGPIO** pins mentioned in the next section. The GPA through GPD ports are backwards compatible with the OS8804, whereas the new **EGPIO** pins have “enhanced” functionality/control.

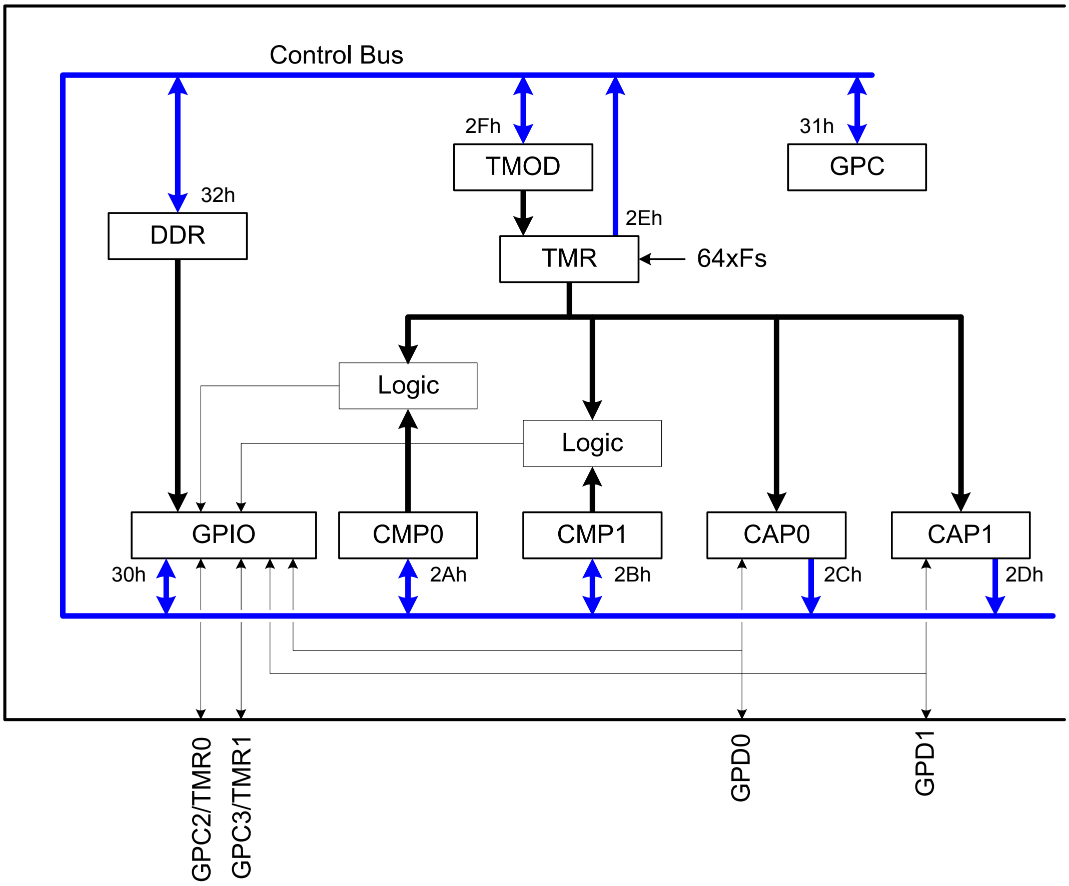


Figure 2-28: GP Timer

To save pins, all the GPIO are multiplexed with other functions. The **GPA[3:0]** and **GPB[3:0]** can be used by either the Host Controller or DSP0. For the Host Controller, they are just GPIO and serve no other purpose. For DSP0, the pins can be either enhanced GPIO or Asynchronous Source Port A. For DSP0 to have control over the **GPA[3:0]** and **GPB[3:0]** pins, the Host Controller must leave them configured as inputs. In rev. G or greater, when the GPIO is configured as an output by either DSP0 or the Host Controller, DSP0’s IPOT register controls the output type (open-drain or driven in both directions). In revisions prior to G, the Host Controller’s GPIO outputs are always open-drain.

The **GPC[3:0]** and **GPD[3:0]** pins can be used by either the Host Controller or DSP1. For the Host Controller, **GPC[3:2]** can be GPIO or **TMR[1:0]**, and the **GPD[3:0]** pins can generate interrupts. For DSP1, the pins can be either enhanced GPIO or Asynchronous Source Port B. For DSP1 to have control over the **GPC[3:0]** and **GPD[3:0]** pins, the Host Controller must leave them configured as inputs. In rev. G or greater, when the GPIO is configured as an output by either DSP1 or the Host Controller, DSP1’s IPOT register controls the output type (open-drain or driven in both directions). In revisions prior to G, the Host Controller’s GPIO outputs are always open-drain.

The general purpose timer (GP Timer) is a 16-bit timer clocked at 64xFs. The modulo of the counter is programmable via the Timer Modulo TMOD register. The phase of this timer, the time when the counter is loaded from the TMOD register, can be adjusted by writing TMOD to zero and then writing the correct value. For example, the TMOD register is loaded with 0x0B06 to get a 1 ms timer tick when the part is in the lock state at a network rate of 44.1 kHz. If the part goes into an unlocked state, it may be desirable to reprogram the timer, while in an unlock state, to 0x0280 to maintain the approximately 1 ms ticks. If the timer counter has counted past 0x0280 when TMOD is written, then the counter will count all the way to 0xFFFF and rollover - causing a very long time interval. Writing TMOD to zero first (which causes the counter to reset) forces the timer to re-initialize. When TMOD is then written to a non-zero value, the timer's counter starts counting up again.

There are two timer output pins: **TMR1-TMR0**. The rising edge of **TMR0** can generate a Host-Controller timer interrupt (**IFL.ITMR0**). Each output has an associated compare register: CMP1 and CMP0. The **TMRn** pins have one state when the GP Timer count is less than or equal to the CMPn register value and another state when the GP Timer count is greater than CMPn. The Polarity of the **TMRn** pins is determined by the **GPC.OP[1:0]** bits.

GPD0 and **GPD1** have associated capture registers: CAP0 and CAP1. When the pin is configured to be edge sensitive (**GPC.LVn** clear), the corresponding capture register is loaded with the contents of the GP Timer when the interrupt occurs. The capture registers are not affected when **GPDn** is configured as level-sensitive (**GPC.LVn** set).

2Ah CMP0 Timer Compare 0 Host

Bit	Label	Description	Default
15..0	D[15:0]	When the timer value is less than or equal to CMP0, the **TMR0** pin has one state, and **TMR0** is in the other state when the timer value is greater than CMP0. (if enabled).	0000h

Table 2-48: CMP0 Register

2Bh CMP1 Timer Compare 1 Host

Bit	Label	Description	Default
15..0	D[15:0]	When the timer value is less than or equal to CMP1, the **TMR1** pin has one state, and **TMR1** is in the other state when the timer value is greater than CMP1. (if enabled).	0000h

Table 2-49: CMP1 Register

2Ch CAP0 Timer Capture 0 (read only) Host

Bit	Label	Description	Default
15..0	D[15:0]	When (edge-sensitive) **GPD0** goes active, TMR value is loaded into CAP0.	0000h

Table 2-50: CAP0 Register

2Dh CAP1 Timer Capture 1 (read only) Host

Bit	Label	Description	Default
15..0	D[15:0]	When (edge-sensitive) **GPD1** goes active, TMR value is loaded into CAP1.	0000h

Table 2-51: CAP0 Register

2Eh TMR GP Timer Value (read only) Host

Bit	Label	Description	Default
15..0	D[15:0]	GP Timer value. Counts up at a 64xFs rate. When TMR equals TMOD, TMR is reset to 0 and counting up begins again.	0000h

Table 2-52: TMR Register

2Fh TMOD GP Timer Modulo Host

Bit	Label	Description	Default
15..0	D[15:0]	GP Timer Modulo value. When TMR reaches this value, TMR is reset to 0 and continues counting up. TMOD should be set to the desired count value - 1. Setting TMOD to 0000h, disables/resets the GP Timer.	0000h

Table 2-53: TMOD Register

30h GPIO General Purpose I/O Data Host

Bit	Label	Description	Default
15..12	GPD[3:0]	I/O data for **GPD[3:0]** pins. As inputs, these pins can also generate interrupts. These pins can also be used by DSP1.	*
11..8	GPC[3:0]	I/O data for **GPC[3:0]** pins. **GPC[3:2]** can also be configured as timer compare outputs. These four pins can also be used by DSP1.	*
7..4	GPB[3:0]	I/O data for **GPB[3:0]** pins. These pins can also be used by DSP0.	*
3..0	GPA[3:0]	I/O data for **GPA[3:0]** pins. These pins can also be used by DSP0.	*

* Initial value is determined by the respective **GPxn** pin.

Table 2-54: GPIO Register

GPD[3:0] I/O data for the **GPD[3:0]** pins. **DDR.GPDOE[3:0]** enables the pin as an output. These pins can also interrupt the Host Controller. As an interrupt, the polarity, and whether its level- or edge-sensitive, is controlled via **GPC.IP[3:0]** and **GPC.LV[3:0]** bits, respectively. The interrupt is enabled via **IER.IED[3:0]** with the status in **IFL.ID[3:0]**. If these pins are left as inputs, then DSP1 can use the GPD port as its own GPIO or as Source Port B1.

GPC[3:0] I/O data for the **GPC[3:0]** pins. **DDR.GPCOE[3:0]** enables the pin as an output. **GPC3** can also be the **TMR1** output, enabled via **GPC.TOE1**. **GPC2** can also be the **TMR0** output, enabled via **GPC.TOE0**. The timer interrupt **IFL.ITMR0** is generated by the rising edge of **TMR0**. If these pins are left as inputs, then DSP1 can use the GPC port as its own GPIO or as Source Port B0.

GPB[3:0] I/O data for the **GPB[3:0]** pins. **DDR.GPBOE[3:0]** enables the pin as an output. If these pins are left as inputs, then DSP0 can use the GPB port as its own GPIO or as Source Port A1.

GPA[3:0] I/O data for the **GPA[3:0]** pins. **DDR.GPAOE[3:0]** enables the pin as an output. If these pins are left as inputs, then DSP0 can use the GPA port as its own GPIO or as Source Port A0.

31h GPC General Purpose I/O Control Host

Bit	Label	Description	Default
15..13	rsvd	Reserved. Write to 0	000
12	rsvd	Reserved. Write to 0	0
11	TOE1	Timer 1 Output Enable. When set, **GPC3** pin functions as **TMR1** output	0
10	TOE0	Timer 0 Output Enable. When set, **GPC2** pin functions as **TMR0** output	0
9, 8	OP[1:0]	TMR[1:0] output polarity	00
7..4	LV[3:0]	**GPD[3:0]** interrupt type. When set, **GPDn** is level-sensitive. When clear, **GPDn** is edge sensitive.	0000
3..0	IP[3:0]	**GPD[3:0]** interrupt polarity. When set, interrupt is active high.	0000

Table 2-55: GPC Register

TOE1 Timer 1 Output Enable. When set, changes the **GPC3** pin to the **TMR1** output. The **DDR.GPCOE3** bit must still be set to use **TMR1**. Once enabled, **OP1** sets the output polarity. When the GP Timer value matches CMP1, **TMR1** changes state.

TOE0 Timer 0 Output Enable. When set, changes the **GPC2** pin to the **TMR0** output. The **DDR.GPCOE2** bit must still be set to use **TMR0**. Once enabled, **OP0** sets the output polarity. When the GP Timer value matches CMP0, **TMR0** changes state. A rising edge of **TMR0** can also generate an interrupt (**IFL.ITMR0**).

OP[1:0] Timer Output Polarity for **TMR1** and **TMR0** pins. When clear, the corresponding pin is low while the GP Timer value is less than or equal to the corresponding CMPn register value. When set, the corresponding pin is high while the GP Timer value is less than or equal to the corresponding CMPn value.

LV[3:0] Level-sensitive Interrupt type. When set, the corresponding **GPDn** interrupt is level sensitive and the **IFL.IDn** bit is read-only. The interrupt must be cleared externally. When **LVn** is clear, the **IFL.IDn** bit is edge-sensitive and cleared by writing a 0 to the **IFL.IDn** bit.

IP[3:0] Interrupt polarity. When set, the **GPDn** pin is active high (rising edge, or high, generates interrupt). When **IPn** is clear, the **GPDn** pin is active low (falling edge, or low, generates interrupt).

32h DDR GPIO Data Direction Register Host

Bit	Label	Description	Default
15..12	GPDOE[3:0]	**GPD[3:0]** pins output enables	0000
11..8	GPCOE[3:0]	**GPC[3:0]** pins output enables	0000
7..4	GPBOE[3:0]	**GPB[3:0]** pins output enables	0000
3..0	GPAOE[3:0]	**GPA[3:0]** pins output enables	0000

Table 2-56: DDR Register

GPDOE[3:0] **GPD[3:0]** output enables. When set, the corresponding **GPDn** pin is an output and, in rev. G or greater, DSP1's **IPOT.GPPTDn** bits control the output driver type. When clear, the **GPDn** pin is an input if DSP1 hasn't configured it as an output through the corresponding **EDD.GPODn** bit.

GPCOE[3:0] **GPC[3:0]** output enables. When set, the corresponding **GPCn** pin is an output and, in rev. G or greater, DSP1's **IPOT.GPPTCn** bits control the output driver type. When clear, the **GPCn** pin is an input if DSP1 hasn't configured it as an output through the corresponding **EDD.GPOCn** bit.

GPBOE[3:0] **GPB[3:0]** output enables. When set, the corresponding **GPBn** pin is an output and, in rev. G or greater, DSP0's **IPOT.GPPTBn** bits control the output driver type. When clear, the **GPBn** pin is an input if DSP0 hasn't configured it as an output through the corresponding **EDD.GPOBn** bit.

GPAOE[3:0] **GPA[3:0]** output enables. When set, the corresponding **GPAn** pin is an output and, in rev. G or greater, DSP0's **IPOT.GPPTAn** bits control the output driver type. When clear, the **GPAn** pin is an input if DSP0 hasn't configured it as an output through the corresponding **EDD.GPOAn** bit.

2.3.7 Global Control

The GCTL register manages Network pins, external DSP0 memory enable, and an advanced software-controlled jitter tolerance circuit for the timing-master node.

3Bh GCTL Global Control register Host

Bit	Label	Description	Default
15	RFPR	RX FIFO Pointers Release	0
14..7	rsvd	Reserved. Write to 0.	00h
6	MJCE	Timing-Master Jitter Control Enable for MOST timing-master node	0
4, 5	rsvd	Reserved. Write to 0.	00
3	NETD	Network Disable. When set, **TX** and **RX** can be used as GPIO.	0
2	ERRD	**ERR** pin Disable. When set, **ERR** can be used as EGPIO **IOG1**	0
1	EDMEN	External Data Memory Enable for DSP0.	0
0	rsvd	Reserved. Write to 0.	0

Table 2-57: GCTL Register

RFPR **RX** FIFO Pointer Release. Only valid when **MJCE** is set. When **RFPR** is clear, the **RX** FIFO Pointers are held in reset, where the pointers are equidistant. When **RFPR** is set, the **RX** FIFO Pointers are free to track the data, where the incoming FIFO pointer is clocked from the **RX** recovered clock and the outgoing FIFO pointer is clocked from an internal non-jittered clock. When **MJCE** is set, software must toggle **RPFR** after any unlock-to-lock sequence to maximize the distance between the read and write pointers. Toggling this bit may or may not cause a momentary unlock event (which should be ignored). Leaving this bit low also may not cause unlock events.

MJCE Timing-Master Jitter Control Enable. When **MJCE** is set in a timing-master (**bXCR.MTR** set), a receive FIFO is used to recover data from the **RX** pin, and software must manage the reset of the pointers to maximize the peak-to-peak jitter tolerance of a master node. When **MJCE** is set, software must use **RFPR** to reset the FIFO pointers every time lock is achieved. When **MJCE** is clear, the MOST transceiver operates in a normal automatic jitter tolerance mode that requires no software intervention.

NETD Network Disable. When set, **TX** and **RX** can be used as GPIO **IOG2** and **IOG3**, respectively.

ERRD **ERR** pin Disable. When set, **ERR** can be used as EGPIO **IOG1**.

EDMEN External Data Memory Enable for DSP0, when **RGEN.**$\overline{\textbf{XME}}$ is set and **MMPC.**$\overline{\textbf{XMQ}}$ is clear. Then **EDMEN** set enables the external data memory port. **EDMEN** clear configures the **IOE[15:0]** and **IOF[10:0]** pins to be used as EGPIO.

2.3.8 EGPIO

The OS8805 has the ability to configure almost all digital IO pins for its regular functions or as Enhanced GPIO pins. There are a total of 64 GPIO pins **GPA[3:0]**, **GPB[3:0]**, **GPC[3:0]**, **GPD[3:0]**, **IOA[3:0]**, **IOB[3:0]**, **IOC[5:0]**, **IOD[1:0]**, **IOE[15:0]**, **IOF[10:0]**, and **IOG[4:0]**, however they are multi-functions pins.

Although the GPIO pins of OS8805 (**GPA[3:0]**, **GPB[3:0]**, **GPC[3:0]**, and **GPD[3:0]**) are multiplexed with the new DSP source ports, the default is that the GPIO pins of OS8805 function the same in the OS8804. The remaining GPIO pins operate as the extended GPIO (EGPIO) where the function of EGPIO is described in this section.

Each EGPIO pin is controlled by three control signals: the data direction **GPOxn** bit, the data polarity or driver type **GPPTxn** bit, and the **EGPIO** sticky or output disable **GPSDxn** bit.

Each EGPIO pin has a corresponding data direction **GPOxn** bit. If the **GPOxn** bit is set and the pin is configured as EGPIO, the pin is configured as an output pin for the Host Controller. If the **GPOxn** bit is low and the pin is configured as EGPIO, the pin is configured as an input pin for the Host Controller.

Each EGPIO pin has a corresponding polarity or driver type **GPPTxn** bit. If the **GPPTxn** bit is set and **GPOxn** bit is clear, the pin serves as an active low input, where the register bit is set if the input is low. If the **GPPTxn** bit is cleared and the **GPOxn** bit is clear, the pin serves as an input where the register bit is set if the input is high.

When the EGPIO pin is configured as an output (**GPOxn** set), the **GPPTxn** bit controls the selection of CMOS or open-drain output level on the pin. If the **GPOxn** bit is set and **GPPTxn** bit is set, the pin is configured as a CMOS output. If the **GPOxn** bit is set and **GPPTxn** bit is clear, the output level is an open-drain driver.

Each EGPIO pin has a corresponding sticky or output disable bit **GPSDxn**. When the **EGPIO** pin is configured to be an input, and the **GPSDxn** bit is set, the corresponding pin will be sticky. If the **GPSDxn** bit is high and the **GPPTxn** bit is low, a high pulse input on the pin will set the corresponding EGPIO register. The EGPIO register remains set until the input goes low and a zero is written to the corresponding **EGPIO** register bit (**GPDxn**). If the data bit is cleared, but the input is still high, the data bit is immediately set again. The combination of the **GPPTxn** and the **GPSDxn** bits supports detection of high or low pulses.

When the EGPIO pin is configured to be an output (**GPOxn** set), the **GPSDxn** bit functions as an output disable.

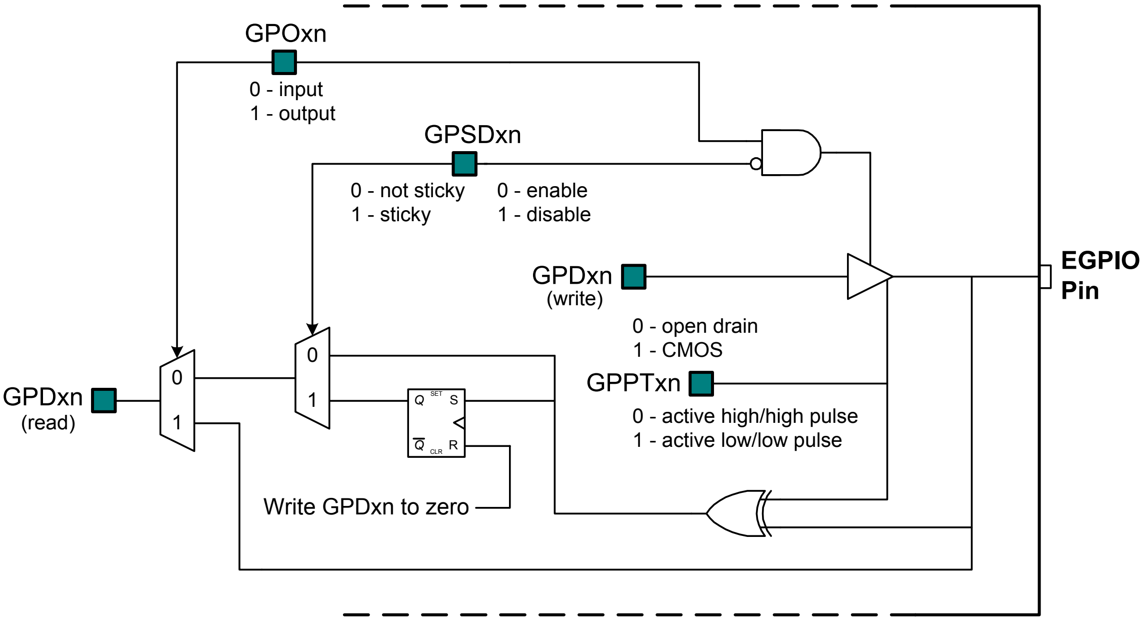


Figure 2-29: EGPIO Conceptual Logic

The table below summarizes the conditions in which each GPIO and EGPIO port can be configured and the default function of each port.

The **GPA[3:0]** and **GPB[3:0]** are shared between the Host Controller and DSP0. The **GPC[3:0]** and **GPD[3:0]** are shared between the Host Controller and DSP1. The DSPs have their own set of registers which control the functionality of these pins and they are implemented as extended GPIOs. When the DDR bit of the Host Controller of the corresponding GPIO pin is low, the DSPs has control of the direction of the GPIO pin.

In the event that both the DDR of the corresponding pin of the Host Controller and the DSPs are set, the Host Controller has priority of the GPIO pin over the DSPs. In other words, the data of the Host Controller is presented on the GPIO pin.

Pin	Function Name	Power-Up State	GPIO Name	GPIO Enable	Default Function
95 94 93 91	**SCKA0** **FSYA0** **SXA0** **SRA0**	Inputs	**GPA0** **GPA1** **GPA2** **GPA3**	DSP0 **DS0C.BPI[6:0]** = 0000000	GPA Port
90 89 88 87	**SCKA1** **FSYA1** **SXA1** **SRA1**	Inputs	**GPB0** **GPB1** **GPB2** **GPB3**	DSP0 **DS1C.BPI[6:0]** = 0000000	GPB Port
86 85 84 81	**SCKB0** **FSYB0** **TMR0/SXB0** **TMR1/SRB0**	Inputs	**GPC0** **GPC1** **GPC2** **GPC3**	DSP1 **DS0C.BPI[6:0]** = 0000000	GPC Port
80 79 78 77	**INT0/SCKB1** **INT1/FSYB1** **INT2/SXB1** **INT3/SRB1**	Inputs	**GPD0** **GPD1** **GPD2** **GPD3**	DSP1 **DS1C.BPI[6:0]** = 0000000	GPD Port
14 15 12 13	**SCL/SCLK/UCK0** **SDA/SDOUT/UTX0** **AD0/SDIN/URX0** **AD1/$\overline{\text{CS}}$/UFR0**	Input - Config. pin Input/Output Input Input	**IOA0** **IOA1** **IOA2** **IOA3**	**CPS.CPFD** = 1 and **U0C.USEN** = 0	Control Port
73 74 99 31	**DSCL/SCLK/UCK1** **DSDA/DSDOUT/UTX1** **DAD0/DSDIN/URX1** **$\overline{\text{DCS}}$/UFR1**	Input - Config. pin Input/Output Input Input	**IOB0** **IOB1** **IOB2** **IOB3**	**DCPS.DPFD** = 1 and **U1C.USEN** = 0	Debug Port
23 22 18 24 19 25	**SCK** **FSY** **SX0** **SR0** **SX1** **SR1**	Input Input Output - low Input Output - low Input	**IOC0** **IOC1** **IOC2** **IOC3** **IOC4** **IOC5**	**bSDC1.MOD[1:0]** = 11	Source Port
98	**PWM0**	Input	**IOD0**	**ACR.ENPWM** = 0	**IOD1**
17	**MA16**	*	**IOD1**	**RGEN.$\overline{\text{XME}}$** = 1*, **MMPC.$\overline{\text{XMQ}}$** = 0, and **GCTL.EDMEN** = 0	***MA16**
110 112 many	**$\overline{\text{MRD}}$/PA15** **MA14/$\overline{\text{MCAS}}$** **MA[13:0]**	*	**IOE15** **IOE14** **IOE[13:0]**	**RGEN.$\overline{\text{XME}}$** = 1*, **MMPC.$\overline{\text{XMQ}}$** = 0, and **GCTL.EDMEN** = 0	*
117 111 109 many	**$\overline{\text{XME}}$/$\overline{\text{PCS}}$/SA15** **$\overline{\text{MRAS}}$/$\overline{\text{MCS}}$** **$\overline{\text{MWR}}$/PMW** **MD[7:0]**	* - Config. Pin * * *	**IOF10** **IOF9** **IOF8** **IOF[7:0]**	**RGEN.$\overline{\text{XME}}$** = 1*, **MMPC.$\overline{\text{XMQ}}$** = 0, and **GCTL.EDMEN** = 0	*
28	**RMCK**	Output - toggling	**IOG0**	**CMCS.DRMCK** = 1	**RMCK**
29	**ERR**	Output	**IOG1**	**GCTL.ERRD** = 1	**ERR**
26 27	**TX** **RX**	Output - **RX** value Input	**IOG2** **IOG3**‡	**GCTL.NETD** = 1	**TX** **RX**
30	**BOOT/PWM1**	Input - **BOOT**	**IOG4**	**ACR2.ENPWM1** = 0	†**BOOT/IOG4**

* Initial value is determined by the **$\overline{\text{XME}}$/$\overline{\text{PCS}}$** pin. If **$\overline{\text{XME}}$/$\overline{\text{PCS}}$** is high at power-up, all pins default to EGPIO inputs, otherwise the pins power-up as external Program memory for the Host Controller.

† At power-up, **BOOT** is the initial function and determines the reset vector for the Host Controller.

‡ When IOG3 is configured as an output, it can only be open-drain (not CMOS).

Table 2-58: EGPIO Enable Summary

4Bh EGPD1 Enhanced GPIO Data register 1 Host

Bit	Label	Description	Default
15, 14	GPDD[1:0]	Data bits for **IOD[1:0]** pins	*
13..8	GPDC[5:0]	Data bits for **IOC[5:0]** pins	*
7..4	GPDB[3:0]	Data bits for **IOB[3:0]** pins	*
3..0	GPDA[3:0]	Data bits for **IOA[3:0]** pins	*

* Initial value determined by state of corresponding pin.

Table 2-59: EGPD1 Register

GPDxn GPIO Data registers. When configured as an output, the data written to this bit is output to the respective pin. The data read is the value at the pin and may not be the value written. When configured as an input, displays the data from the pin, either directly or captured, based on the polarity selected.

3Dh EGPD2 Enhanced GPIO Data register 2 Host

Bit	Label	Description	Default
15..0	GPDE[15:0]	Data bits for **IOE[15:0]** pins	*

* Initial value determined by state of corresponding pin.

Table 2-60: EGPD2 Register

GPDEn GPIO Data registers. When configured as an output, the data written to this bit is output to the respective pin. The data read is the value at the pin and may not be the value written. When configured as an input, displays the data from the pin, either directly or captured, based on the polarity selected.

3Eh EGPD3 Enhanced GPIO Data register 3 Host

Bit	Label	Description	Default
15..12	GPDG[3:0]	Data bits for **IOG[3:0]** pins	*
11	GPDG4	Data bit for **IOG4** pin	†
10..0	GPDF[10:0]	Data bits for **IOF[10:0]** pins	*

* Initial value determined by state of corresponding pin.
† Initial value set by **BOOT** pin function, which determines the reset vector for the Host Controller.

Table 2-61: EGPD3 Register

GPDxn GPIO Data registers. When configured as an output, the data written to this bit is output to the respective pin. The data read is the value at the pin and may not be the value written. When configured as an input, displays the data from the pin, either directly or captured, based on the polarity selected.

4Ch EDD1 Enhanced GPIO Data Direction register 1 Host

Bit	Label	Description	Default
15, 14	GPOD[1:0]	Output enable for **IOD[1:0]** pins	00
13..8	GPOC[5:0]	Output enable for **IOC[5:0]** pins	000000
7..4	GPOB[3:0]	Output enable for **IOB[3:0]** pins	0000
3..0	GPOA[3:0]	Output enable for **IOA[3:0]** pins	0000

Table 2-62: EDD1 Register

GPOxn Output Enable for corresponding **IOxn** pin. When **IOxn** is configured as a GPIO, this corresponding bit determines whether the pin is a general purpose input or output.

0 - Corresponding **IOxn** pin is an input.

IPOT1.GPPTxn determines input polarity.

ISOD1.GPSDxn determines whether input is sticky or not.

1 - Corresponding **IOxn** pin is an output.

IPOT1.GPPTxn determines output driver type (CMOS or open-drain).

ISOD1.GPSDxn is an output disable/enable.

If the Control Port formats are enabled (**CPS.CPFD** clear), **IOA3** can be configured as an output when using GSPI master mode to act as the external device chip select, and must be configured as an input in all other cases. If the Control Port is used in USART synchronous slave mode (**CPS.CPFD** clear, **U0C.USEN** and **U0C.SYEN** set and **U0C.CKMTR** clear), **IOA0** must be configured as input. If the Control Port is used in USART asynchronous mode (**CPS.CPFD** and **U0C.SYEN** clear, and **U0C.USEN** set), **IOA0** and **IOA3** can be used as GPIO.

Likewise, if the Debug Port formats are enabled (**DCPS.DPFD** clear), **IOB3** can be configured as an output when using GSPI master mode to act as the external device chip select, and must be configured as an input in all other cases. If the Debug Port is used in USART synchronous slave mode (**DCPS.DPFD** clear, **U1C.USEN** and **U1C.SYEN** set and **U1C.CKMTR** clear), **IOB0** must be configured as input. If the Debug Port is used in USART asynchronous mode (**DCPS.DPFD** and **U1C.SYEN** clear, and **U1C.USEN** set), **IOB0** and **IOB3** can be used as GPIO.

3Fh EDD2 Enhanced GPIO Data Direction register 2 Host

Bit	Label	Description	Default
15..0	GPOE[15:0]	Output enable for **IOE[15:0]** pins	0000h

Table 2-63: EDD2 Register

GPOEn Output Enable for corresponding **IOEn** pin. When **IOEn** is configured as a GPIO, this corresponding bit determines whether the pin is a general purpose input or output.

0 - Corresponding **IOEn** pin is an input.

IPOT2.GPPTEn determines input polarity.

ISOD2.GPSDEn determines whether input is sticky or not.

1 - Corresponding **IOEn** pin is an output.

IPOT2.GPPTEn determines output driver type (CMOS or open-drain).

ISOD2.GPSDEn is an output disable/enable.

40h EDD3 Enhanced GPIO Data Direction register 3 Host

Bit	Label	Description	Default
15..12	GPOG[3:0]	Output enable for **IOG[3:0]** pins	0h
11	GPOG4	Output enable for **IOG4** pin	0
10..0	GPOF[10:0]	Output enable for **IOF[10:0]** pins	00h, 000

Table 2-64: EDD3 Register

GPOxn Output Enable for corresponding **IOxn** pin. When **IOxn** is configured as a GPIO, this corresponding bit determines whether the pin is a general purpose input or output.

0 - Corresponding **IOxn** pin is an input.
IPOT3.GPPTxn determines input polarity.
ISOD3.GPSDxn determines whether input is sticky or not.

1 - Corresponding **IOxn** pin is an output. When **IOG3** is an output, it can only be configured as open-drain, **IPOT3.GPPTG3** doesn't affect the pin.
IPOT3.GPPTxn determines output driver type (CMOS or open-drain).
ISOD3.GPSDxn is an output disable/enable.

41h IPOT1 EGPIO Input Polarity/Output Type register 1 Host

Bit	Label	Description	Default
15, 14	GPPTD[1:0]	Input polarity or output driver type for **IOD[1:0]** pins	00
13..8	GPPTC[5:0]	Input polarity or output driver type for **IOC[5:0]** pins	000000
7..4	GPPTB[3:0]	Input polarity or output driver type for **IOB[3:0]** pins	0000
3..0	GPPTA[3:0]	Input polarity or output driver type for **IOA[3:0]** pins	0000

Table 2-65: IPOT1 Register

GPPTxn Input polarity or output driver type for corresponding **IOxn** pin when configured as a GPIO.

When **IOxn** pin is configured as an input (**EDD1.GPOxn** clear), this bit sets the polarity:
0 - active high input (non-inverting) or high-pulse capture if sticky
1 - active low input (inverting) or low-pulse capture if sticky

When **IOxn** pin is configured as an output (**EDD1.GPOxn** set), this bit sets the output driver type:
0 - Open-drain output (only driven low)
1 - CMOS output (driven both high and low)

42h IPOT2 EGPIO Input Polarity/Output Type register 2 Host

Bit	Label	Description	Default
15..0	GPPTE[15:0]	Input polarity or output driver type for **IOE[15:0]** pins	0000h

Table 2-66: IPOT2 Register

GPPTEn Input polarity or output driver type for corresponding **IOEn** pin when configured as a GPIO.

When **IOEn** pin is configured as an input (**EDD2.GPOEn** clear), this bit sets the polarity:
0 - active high input (non-inverting) or high-pulse capture if sticky
1 - active low input (inverting) or low-pulse capture if sticky

When **IOEn** pin is configured as an output (**EDD2.GPOEn** set), this bit sets the output driver type:
0 - Open-drain output (only driven low)
1 - CMOS output (driven both high and low)

43h IPOT3 EGPIO Input Polarity/Output Type register 3 Host

Bit	Label	Description	Default
15..12	GPPTG[3:0]	Input polarity or output driver type for **IOG[3:0]** pins	0h
11	GPPTG4	Input polarity or output driver type for **IOG4** pin	0
10..0	GPPTF[10:0]	Input polarity or output driver type for **IOF[10:0]** pins	00h, 000

Table 2-67: IPOT3 Register

GPPTxn Input polarity or output driver type for corresponding **IOxn** pin when configured as a GPIO.

When **IOxn** pin is configured as an input (**EDD3.GPOxn** clear), this bit sets the polarity:

0 - active high input (non-inverting) or high-pulse capture if sticky

1 - active low input (inverting) or low-pulse capture if sticky

When **IOxn** pin is configured as an output (**EDD3.GPOxn** set), this bit sets the output driver type.

When **IOG3** is an output, it can only be configured as open-drain, **IPOT3.GPPTG3** doesn't affect the pin.

0 - Open-drain output (only driven low)

1 - CMOS output (driven both high and low)

44h ISOD1 EGPIO Input Sticky/Output Disable register 1 Host

Bit	Label	Description	Default
15, 14	GPSDD[1:0]	Input sticky or output disable for **IOD[1:0]** pins.	00
13..8	GPSDC[5:0]	Input sticky or output disable for **IOC[5:0]** pins	000000
7..4	GPSDB[3:0]	Input sticky or output disable for **IOB[3:0]** pins	0000
3..0	GPSDA[3:0]	Input sticky or output disable for **IOA[3:0]** pins	0000

Table 2-68: ISOD1 Register

GPSDxn Input sticky or output disable for corresponding **IOxn** pin when configured as a GPIO.

When **IOxn** pin is configured as an input (**EDD1.GPOxn** clear), this bit sets sticky or not.

0 - **EGPD1.GPDxn** reflects the **IOxn** pin, after polarity (**IPOT1.GPPTxn**).

1 - **EGPD1.GPDxn** is sticky and captures a high or low pulse (based on **IPOT1.GPPTxn**) on **IOxn**. Once set, cleared by writing **EGPD1.GPDxn** to zero (assuming pin is in inactive state).

When **IOxn** pin is configured as an output (**EDD1.GPOxn** set), this bit controls output enable:

0 - The **IOxn** pin is enabled and driven, based on **IPOT1.GPPTxn**.

1 - The **IOxn** pin is disabled, high impedance.

45h ISOD2 EGPIO Input Sticky/Output Disable register 2 Host

Bit	Label	Description	Default
15..0	GPSDE[15:0]	Input sticky or output disable for **IOE[15:0]** pins	0000h

Table 2-69: ISOD2 Register

GPSDEn Input sticky or output disable for corresponding **IOEn** pin when configured as a GPIO.

When **IOEn** pin is configured as an input (**EDD2.GPOEn** clear), this bit sets sticky or not.

0 - **EGPD2.GPDEn** reflects the **IOEn** pin, after polarity (**IPOT2.GPPTEn**).

1 - **EGPD2.GPDEn** is sticky and captures a high or low pulse (based on **IPOT2.GPPTEn**) on **IOEn**. Once set, cleared by writing **EGPD2.GPDEn** to zero (assuming pin is in inactive state).

When **IOEn** pin is configured as an output (**EDD2.GPOEn** set), this bit controls output enable:

0 - The **IOEn** pin is enabled and driven, based on **IPOT2.GPPTEn.**

1 - The **IOEn** pin is disabled, high impedance.

46h ***ISOD3*** ***EGPIO Input Sticky/Output Disable register 3*** ***Host***

Bit	Label	Description	Default
15..12	GPSDG[3:0]	Input sticky or output disable for **IOG[3:0]** pins	0h
11	GPSDG4	Input sticky or output disable for **IOG4** pin	0
10..0	GPSDF[10:0]	Input sticky or output disable for **IOF[10:0]** pins	00h, 000

Table 2-70: ISOD3 Register

GPSDxn Input sticky or output disable for corresponding **IOxn** pin when configured as a GPIO.
When **IOxn** pin is configured as an input (**EDD3.GPOxn** clear), this bit sets sticky or not.
0 - **EGPD3.GPDxn** reflects the **IOxn** pin, after polarity (**IPOT3.GPPTxn**).
1 - **EGPD3.GPDxn** is sticky and captures a high or low pulse (based on **IPOT3.GPPTxn**) on **IOxn**. Once set, cleared by writing **EGPD3.GPDxn** to zero (assuming pin is in inactive state).
When **IOxn** pin is configured as an output (**EDD3.GPOxn** set), this bit controls output enable:
0 - The **IOxn** pin is enabled and driven, based on **IPOT3.GPPTxn**.
1 - The **IOxn** pin is disabled, high impedance.

2.3.9 DC Measurement ADC

The DC Measure ADC consists of an eight-to-one mux followed by a charge-balanced analog-to-digital converter with a first-order modulator. The analog input range is from 0 V to twice the **VREF** voltage. The output is a two's complement number where 000h represents center scale or the **VREF** voltage, which is typically 1.3 V.

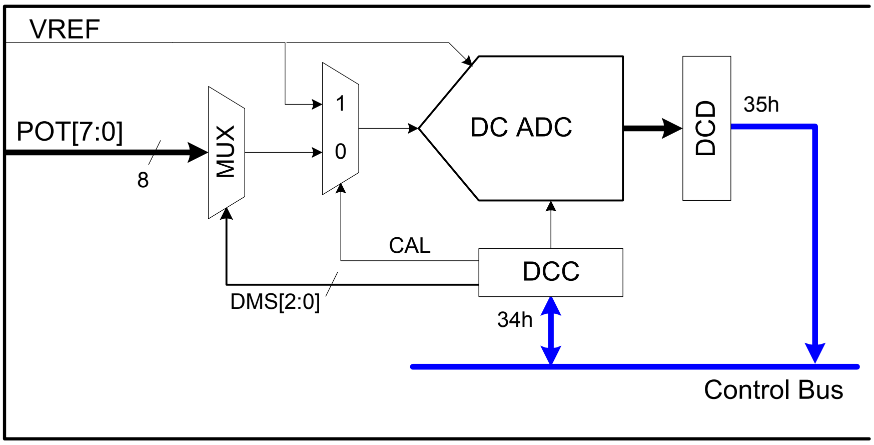


Figure 2-30: DC Measurement ADC

The DCC register contains three bits which control the analog multiplexer and a sample bit (**SAM**). The output of the multiplexer is sampled onto a capacitor approximately 1 µs after **DCC.SAM** goes high. This sampled value is converted to a 12-bit value in approximately another 1 ms. After the conversion is complete, the ready signal **DCC.RDY** goes high. The ready signal sets the DC ADC data ready interrupt bit **IFL.IADC**.

Since there is a delay between the time **DCC.SAM** goes high and the output of the mux is sampled, **DCC.SAM** and the mux control bits **DCC.DMS[2:0]** can be changed at the same time. In addition, the mux can be changed after the sampling process is complete, but before the conversion process is complete.

The interrupt resolution control bits **DCC.IRES[2:0]** specify the resolution at which **DCC.RDY** goes high, the conversion is stopped, and the interrupt occurs.

34h DCC DC Measurement ADC Control Host

Bit	Label	Description	Default
15	RDY	Data Ready. Cleared by reading DCD register	0
14	CAL	Calibration	0
13..8	rsvd	Reserved. Write to 0	000000
7	PDN	Power down.	0
6..4	IRES[2:0]	Interrupt Resolution	000
3	SAM	Sample (write only)	0
2..0	DMS[2:0]	DC ADC Mux Select	000

Table 2-71: DCC Register

RDY Data Ready. When set, indicates that the DCD register has valid data in it. The resolution which causes **RDY** to go high is specified by **IRES[2:0]**. When **RDY** goes high, an interrupt can be generated (**IFL.IADC**) if the interrupt enable bit **IER.IEADC** is set. When the DCD register is read, **RDY** is cleared.

CAL Calibration. When set, the voltage reference is fed directly to the DC ADC input so that the VREF error can be determined. Since setting **CAL** only selects the **VREF** voltage, subtracting out the offset error must be done in Controller software.

PDN Power-down. When set, the DC ADC is powered down.

IRES[2:0] Interrupt resolution. Sets the resolution at which the interrupt **IFL.IADC** is generated.

000 – 12 bit
001 – 11 bit
010 – 10 bit
011 – 9 bit
100 – 8 bit
101 – 7 bit
110 – 6 bit
111 – 5 bit

SAM Start Sampling. Approximately 1 µs after **SAM** is set, the selected input is sampled. **SAM** is automatically cleared. The conversion must finish (**RDY** high) before another conversion can be started. If **SAM** is set before a conversion is complete, the current conversion is lost, and a new conversion is started.

DMS[2:0] DC Measurement ADC Mux Select. Selects the DC input to be converted. Since the sampling occurs approximately 1 µs after **SAM** is set, **SAM** and **DMS[2:0]** can be changed in the same register write.

000 – **POT0** pin
001 – **POT1** pin
010 – **POT2** pin
011 – **POT3** pin
100 – **POT4** pin
101 – **POT5** pin
110 – **POT6** pin
111 – **POT7** pin

35h DCD DC Measurement ADC Data (read only) Host

Bit	Label	Description	Default
15	rsvd	Reserved.	0
14..12	RES[2:0]	Data Resolution. Between 5 and 12, indicating the current resolution.	000
11..0	DCD[11:0]	DC ADC measurement value. Bit 11 (**DCD11**) is the MSB. Data is two's complement format.	000h

Table 2-72: DCD Register

RES[2:0] Resolution of the **DCD[11:0]** bits. The conversion cycles through the resolutions until it reaches the resolution set in **DCC.RES[2:0]** bits. **RES[2:0]** are updated as the **DCD[11:0]** bits are updated when the **DCC.IRES[2:0]** resolution is reached, conversion is stopped and **DCC.RDY** is set. When DCD is read, the **DCC.RDY** bit is cleared.

000 – 12 bit
001 – 11 bit
010 – 10 bit
011 – 9 bit
100 – 8 bit
101 – 7 bit
110 – 6 bit
111 – 5 bit

DCD[11:0] Two's complement DC ADC measurement result. Resolutions below 12 bits are LSB-justified (smaller numbers) and sign-extended to the full 12 bits, as shown in the Table below. Figure 2-31 illustrates the transfer function when configured for 12-bit resolution.

Resolution	approx. 96Fs cycles	RES[2:0]	DCD[11:0]	
			Positive Number	Negative Number
5 bits	31	111	00000000RRRR	11111111RRRR
6 bits	63	110	0000000RRRRR	1111111RRRRR
7 bits	127	101	000000RRRRRR	111111RRRRRR
8 bits	255	100	00000RRRRRRR	11111RRRRRRR
9 bits	511	011	0000RRRRRRRR	1111RRRRRRRR
10 bits	1023	010	000RRRRRRRRR	111RRRRRRRRR
11 bits	2047	001	00RRRRRRRRRR	11RRRRRRRRRR
12 bits	4095	000	0RRRRRRRRRRR	1RRRRRRRRRRR

The 'R' characters are the non-sign resolution binary bits

Table 2-73: DC ADC Resolution

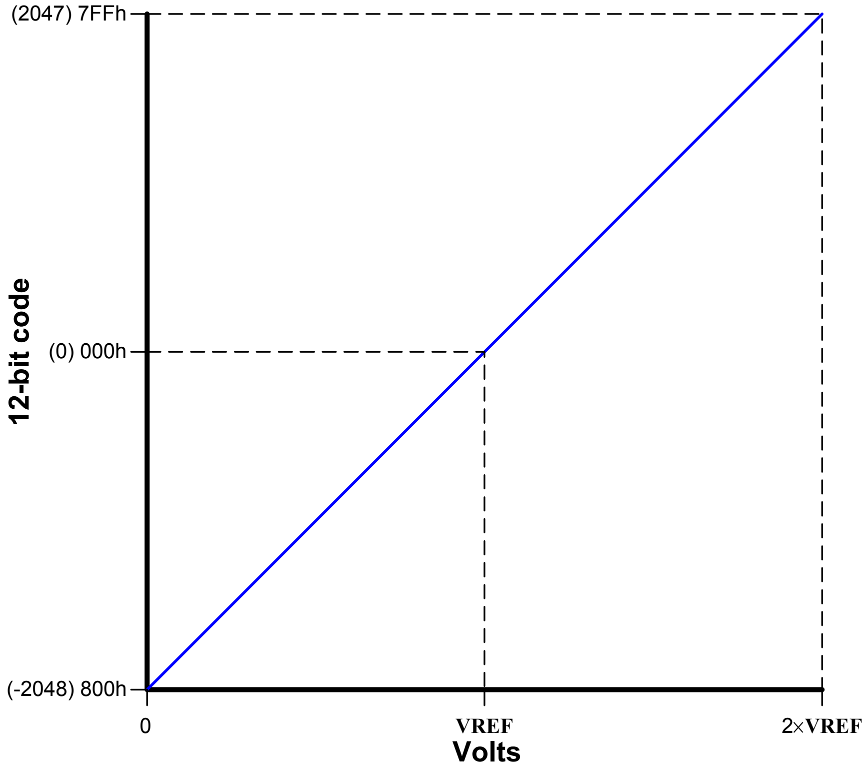


Figure 2-31: DC ADC Transfer Function (12-bits)

2.3.10 Source Converter Control

The Source Converters consist of the high-speed ADCs and DACs that are connected through the MOST Routing bus. The MOST Routing Table determines how to access the Converter data. The PWM DACs are controlled by DSP0. The DC Measurement ADC is controlled directly by the Host Controller. For all converters, the enables and volume controls are accessed by the Host Controller through the Control bus. All volume controls use zero-crossing to change volume, thereby minimizing pops and clicks.

Each ADC has a programmable input gain stage before the actual ADC, accessed via the Control bus. Each DAC has a programmable output attenuator and associated control register, which specifies the amount of attenuation.

The gain and attenuation bits in the ADACn registers, GADCs, GMIC and GMPX registers are mapped to both DSPs as well as the Host Controller. Since both the Host Controller and DSPs can write to the register to change the setting, the last write of the register will remain and that is the value to be read back. Therefore, the Host Controller may not read back what it wrote to the register if one of the DSPs writes to it after the Host Controller write. In addition, these registers can only be written when **FPCR.RUN** is set, and the Source Port clocks (**FSY** and **SCK**) are operating (are outputs and enabled, or are inputs and driven by external clocks).

1Dh FPCR Source Converter Control Register Host

Bit	Label	Description	Default
15..4	rsvd	Reserved. Write to 0.	000h
3	RUN	Global enable for Source Converters. When set, the enabled Source Converters data is processed and transferred across the Routing bus. When **RUN** is clear, no data is transferred across the Routing bus.	0
2..0	rsvd	Reserved. Write to 0.	000

Table 2-74: FPCR Register

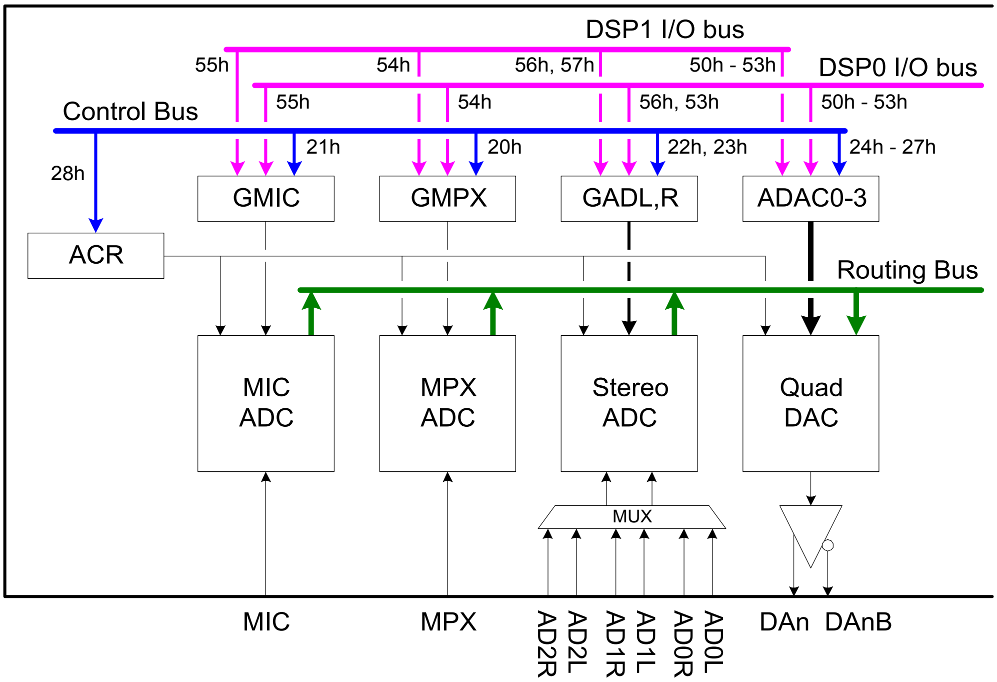


Figure 2-32: Source Converters

Each data converter channel can be independently enabled and disabled through the enable bits in the Analog Control Register (ACR). When the enable bit is high, the corresponding converter operates normally. When the enable bit is low, the corresponding converter enters a low power state. In the low power state, the digital data out of the ADCs is all zero's. DSP0, DSP1, and the Host Controller all have access to the converter's volume controls. The last value written is what is read by any of the three processors. Therefore, if two processors are writing a converter's volume register, a processor may not read the same value written (if the other processor made the last write).

Each audio ADC has a three-to-one input mux. Both muxes are controlled by the same two bits in the analog control register.

28h ACR Analog Control Register Host

Bit	Label	Description	Default
15..13	rsvd	Reserved. Write to 0	000
12	PWMOE	PWM0 DAC Output enable. When clear, the **PWM0** output is muted.	0
11	ENPWM	PWM0 DAC enable. When clear, the PWM0 DAC is powered down.	0
10	rsvd	Reserved. Write to 0	0
9, 8	AMS[1:0]	Audio ADC Mux input select	00
7	ENMPX	MPX ADC enable. When clear, MPX ADC is powered down.	0
6	ENMIC	Mic ADC enable. When clear, Mic ADC is powered down.	0
5	ENADL	Left Audio ADC enable. When clear, Left Audio ADC is powered down.	0
4	ENADR	Right Audio ADC enable. When clear, Right Audio ADC is powered down.	0
3	ENDAC0	DAC0 enable. When clear, DAC0 is powered down.	0
2	ENDAC1	DAC1 enable. When clear, DAC1 is powered down.	0
1	ENDAC2	DAC2 enable. When clear, DAC2 is powered down.	0
0	ENDAC3	DAC3 enable. When clear, DAC3 is powered down.	0

Table 2-75: ACR Register

PWMOE PWM DAC Output Enable. When set, the **PWM0** pin outputs a PWM signal representative of the data sent to the PWMDR register by DSP0. When clear, the **PWM0** output is muted (AC ground).

ENPWM Enable PWM DAC. When set, the PWM0 DAC is enabled. When clear, the PWM0 DAC is powered down and the **PWM0** pin can be used as GPIO **IOD0** pin.

AMS[1:0] Audio ADC Mux Select. Selects stereo analog inputs to the Audio ADC.
00 – **AD0L, AD0R** pins
01 – **AD1L, AD1R** pins
10 – **AD2L, AD2R** pins
11 – Reserved.

ENMPX Enable MPX ADC. When set, the MPX ADC is enabled. When clear, MPX is powered down.

ENMIC Enable Mic ADC. When set, the Mic ADC is enabled. When clear, Mic is powered down.

ENADL Enable Left Analog ADC. When set, the left Analog ADC is enabled. When clear, the left Analog ADC is powered down.

ENADR Enable Right Analog ADC. When set, the right Analog ADC is enabled. When clear, the right Analog ADC is powered down.

ENDAC[3:0] Enables for DAC3 through DAC0. When set, the corresponding DACx is enabled. When clear, DACx is powered down and the output pin is at analog ground (DC zero).

4Ah ACR2 Analog Control Register 2 Host

Bit	Label	Description	Default
15..13	rsvd	Reserved. Write to 0.	000
12	PWM1OE	PWM1 DAC Output enable. When clear, the **PWM1** output is muted.	0
11	ENPWM1	PWM1 DAC enable. When clear, the PWM1 DAC is powered down.	0
10..0	rsvd	Reserved. Write to 0.	00, 00h

Table 2-76: ACR Register

PWM1OE PWM1 DAC Output Enable. When set, the **PWM1** pin outputs a PWM signal representative of the data sent to the PWM1DR register by DSP0. When clear, the **PWM1** output is muted (AC ground).

ENPWM1 Enable PWM1 DAC. When set, the PWM1 DAC is enabled. When clear, the PWM1 DAC is powered down and the **PWM1** pin can be used as GPIO **IOG4**.

2.3.10.1 MPX ADC

20h GMPX MPX ADC Volume Host

Bit	Label	Description	Default
15..5	rsvd	Reserved. Write to 0.	00h, 000
4..0	GAIN[4:0]	MPX ADC Gain. The least significant bit represents 1 dB, with 00000 = 0 dB and 11010 = 26 dB (maximum value).	00000

Table 2-77: GMPX Register

2.3.10.2 Microphone ADC

21h GMIC MIC ADC Volume Host

Bit	Label	Description	Default
15..4	rsvd	Reserved. Write to 0.	000h
3..0	GAIN[3:0]	Mic ADC Gain. The least significant bit represents 1 dB, with 0000 = 0 dB and 1111 = 15 dB.	0000

Table 2-78: GMIC Register

2.3.10.3 Stereo Audio ADCs

22h GADL Left Audio ADC Volume Host

Bit	Label	Description	Default
15..4	rsvd	Reserved. Write to 0.	000h
3..0	GAIN[3:0]	Left Audio ADC Gain. The least significant bit represents 1 dB, with 0000 = 0 dB and 1111 = 15 dB.	0000

Table 2-79: GADL Register

23h GADR Right Audio ADC Volume Host

Bit	Label	Description	Default
15..4	rsvd	Reserved. Write to 0.	000h
3..0	GAIN[3:0]	Right Audio ADC Gain. The least significant bit represents 1 dB, with 0000 = 0 dB and 1111 = 15 dB.	0000

Table 2-80: GADR Register

2.3.10.4 Quad Audio DACs

For the DACs, common-mode rejection can be improved by connecting 0.1 µF and 1 µF bypass capacitors between the **VREFS** pin and AGND, and setting the **ADACn.SEDE** bit for the particular DAC. Although these capacitors help in differential mode, significant dynamic range improvements are seen when using the DACs in single-ended mode.

24h ADAC0 Audio DAC0 Volume Host

Bit	Label	Description	Default
15..7	rsvd	Reserved. Write to 0.	00h, 0
6	SEDE	Set when **VREFS** has capacitors connected to improve dynamic range.	0
5	MUTE	Mute. When set, DAC0 output is muted.	0
4..0	ATTN[4:0]	DAC0 Attenuation. The least significant bit represents -1 dB, with 00000 = 0 dB and 11111 = -31 dB.	00000

Table 2-81: ADAC0 Register

25h ADAC1 Audio DAC1 Volume Host

Bit	Label	Description	Default
15..7	rsvd	Reserved. Write to 0.	00h, 0
6	SEDE	Set when **VREFS** has capacitors connected to improve dynamic range.	0
5	MUTE	Mute. When set, DAC1 output is muted.	0
4..0	ATTN[4:0]	DAC1 Attenuation. The least significant bit represents -1 dB, with 00000 = 0 dB and 11111 = -31 dB.	00000

Table 2-82: ADAC1 Register

26h ADAC2 Audio DAC2 Volume Host

Bit	Label	Description	Default
15..7	rsvd	Reserved. Write to 0.	00h, 0
6	SEDE	Set when **VREFS** has capacitors connected to improve dynamic range.	0
5	MUTE	Mute. When set, DAC2 output is muted.	0
4..0	ATTN[4:0]	DAC2 Attenuation. The least significant bit represents -1 dB, with 00000 = 0 dB and 11111 = -31 dB.	00000

Table 2-83: ADAC2 Register

27h ADAC3 Audio DAC3 Volume Host

Bit	Label	Description	Default
15..7	rsvd	Reserved. Write to 0.	00h, 0
6	SEDE	Set when **VREFS** has capacitors connected to improve dynamic range.	0
5	MUTE	Mute. When set, DAC3 output is muted.	0
4..0	ATTN[4:0]	DAC3 Attenuation. The least significant bit represents -1 dB, with 00000 = 0 dB and 11111 = -31 dB.	00000

Table 2-84: ADAC3 Register

2.3.11 DSP Program Control

The Host Controller has access to each DSP's Program Counter (DxPC), Program Control Register (DxPCR), and Program Data Register - Lower bit (DxPDL). Since DSP memory is RAM, the Host Controller must load the RAM on start-up. This is accomplished by setting the DxPC to the starting address, and downloading data through the DxPDL and DxPCR registers. Once the DSP memory is filled, the Controller can program DxPC to the start of memory and set the **DxPCR.RUN** bit to start DSP execution.

Since DSP memory is 26 bits wide, two Controller registers are needed to access one DSP word. The DxPDL register provides access to the lower 16 bits of DSP memory (**DM[15:0]**). The DxPCR register provides access to the upper 10 bits of DSP memory through the **DM[25:16]** bits.

When the **DxPCR.AI** bit is set, the DxPC increments after data is written to (or read from) Program memory. This allows blocks of data to be loaded efficiently.

17h D0PCR DSP0 Program Control Host

Bit	Label	Description	Default
15, 14	*DFS[1:0]	DSP0 frequency select	11
13..4	DM[25:16]	High 10 bits of DSP0 Program memory. Bit 13 (**DM25**) is MSB.	00h, 00
3	*RUN	DSP0 Run enable. When clear, DSP0 is idle.	0
2	AI	Auto-Increment enable	0
1	WR	Write program memory (write only)	0
0	RD	Read program memory (write only)	0

* **RUN** must be clear to access the **DFS[1:0]** bits. In addition, **RUN** must not be enabled in the same cycle where the **DFS[1:0]** bits are configured.

Table 2-85: D0PCR Register

DFS[1:0] DSP0 Frequency Select bits. Selects the operating speed of the DSP. These bits must be configured before (in a different I/O cycle) **RUN** is set.
00 - 768×Fs
01 - 960×Fs
10 - 1536xFs
11 - 1344×Fs (default)

DM[25:16] The upper 10 bits of DSP0's Program memory. Used, in conjunction with **D0PDL.DM[15:0]**, as the data portal when the Controller is accessing DSP Program memory.

RUN Set for normal operation. When clear, the DSP is in a low power state and the operating speed can be selected through **DFS[1:0]**. In addition, Program memory can be filled by the Controller. Once the programs are downloaded, D0PC should be programmed to 0, **DFS[1:0]** should be configured, and (in a different cycle) **RUN** should be set to start DSP execution.

AI When set, the D0PC increments after a write (**WR**) or a read (**RD**).

WR Write DSP0 Program memory (write only). When **WR** is set, the **DM[25:0]** data (**D0PCR.DM[25:16]** + **D0PDL.DM[15:0]**) is written to the address specified by **D0PC**. **WR** is auto-cleared after the operation is complete.

RD Read DSP0 Program memory (write only). When **RD** is set, the **DM[25:0]** data (**D0PCR.DM[25:16]** + **D0PDL.DM[15:0]**) is read from the address specified by **D0PC**. **RD** is auto-cleared after the operation is complete.

18h D0PDL DSP0 Program memory Data Low Host

Bit	Label	Description	Default
15..0	DM[15:0]	DSP0's lower 16 bits of Program memory used, in conjunction with **D0PCR.DM[25:16]**, for Controller reading or writing DSP memory.	0000h

Table 2-86: D0PDL Register

19h D0PC DSP0 Program Counter Host

Bit	Label	Description	Default
15..11	rsvd	Reserved. Write to 0.	00000
10..0	D[10:0]	DSP0 Program Counter. Bit 10 (**D10**) is the MSB	00h, 000

Table 2-87: D0PC Register

1Ah D1PCR DSP1 Program Control Host

Bit	Label	Description	Default
15, 14	*DFS[1:0]	DSP1 frequency select	11
13..4	DM[25:16]	High 10 bits of DSP1's Program memory. Bit 13 (**DM25**) is MSB.	00h, 00
3	RUN	DSP1 Run enable. When clear, DSP1 is in low-power mode.	0
2	AI	Auto-Increment enable	0
1	WR	Write program memory (write only)	0
0	RD	Read program memory (write only)	0

* **RUN** must be clear to access the **DFS[1:0]** bits. In addition, **RUN** must not be enabled in the same cycle where the **DFS[1:0]** bits are configured.

Table 2-88: D1PCR Register

DFS[1:0] DSP1 Frequency Select bits. Selects the operating speed of the DSP. These bits must be configured before (in a different I/O cycle) **RUN** is set.

00 - 768×Fs
01 - 960×Fs
10 - 1536xFs
11 - 1344×Fs (default)

DM[25:16] The upper 10 bits of DSP1's Program memory. Used, in conjunction with **D1PDL.DM[15:0]**, as the data portal when the Controller is accessing DSP memory.

RUN Set for normal operation. When clear, the DSP is in a low power state and the operating speed can be selected through **DFS[1:0]**. In addition, Program memory can be filled by the Controller. Once the programs are downloaded, D1PC should be programmed to 0, **DFS[1:0]** should be configured, and (in a different cycle) **RUN** should be set to start DSP execution.

AI When set, the D1PC increments after a write (**WR**) or a read (**RD**).

WR Write DSP1 Program memory (write only). When **WR** is set, the DM[25:0] data (**D1PCR.DM[25:16]** + **D1PDL.DM[15:0]**) is written to the address specified by D1PC. **WR** is auto-cleared after the operation is complete.

RD Read DSP1 Program memory (write only). When **RD** is set, the DM[25:0] data (**D1PCR. DM[25:16]** + **D1PDL. DM [15:0]**) is read from the address specified by D1PC. **RD** is auto-cleared after the operation is complete.

1Bh D1PDL DSP1 Program memory Data Low Host

Bit	Label	Description	Default
15..0	DM[15:0]	DSP1's lower 16 bits of Program memory used, in conjunction with **D1PCR. DM [25:16]**, for Controller reading or writing DSP memory.	0000h

Table 2-89: D1PDL Register

1Ch D1PC DSP1 Program Counter Host

Bit	Label	Description	Default
15..11	rsvd	Reserved. Write to 0.	00000
10..0	D[10:0]	DSP1 Program Counter. Bit 10 (**D10**) is the MSB	00h, 000

Table 2-90: D1PC Register

2.3.12 DSP0 External Memory Configuration

Only when the Host Controller is operating out of internal memory (**RGEN.$\overline{\text{XME}}$** set and **MMPC.$\overline{\text{XMQ}}$** clear), can DSP0 use the external memory port. When the external memory port is configured for DSP0 external memory, **GCTL.EDMEN** set enables the external memory port for DSP0 data memory expansion. If **GCTL.EDMEN** is clear, then the external memory port can be used as GPIO. Section 4.6.1 on page 174, *DSP0 External Memory Configuration*, describes the software interface to DSP0's external data memory port.

29h XMC DSP0 External Data Memory Control Host

Bit	Label	Description	Default
15..8	RCD[7:0]	Refresh Counter Divider	00h
7	rsvd	Reserved. Write to 0.	0
6, 5	RCP[1:0]	Refresh Counter Prescaler	00
4	ARE	Automatic Refresh Enable (DRAM only, MMS = 1)	0
3	MT	Memory Timing: 0 = fast, 1 = slow	0
2	BW	external data Bus Width: 0 = 8 bits, 1 = 4 bits	0
1	MWW	Memory Word Width: 0 = 8 bits, 1 = 16 bits	0
0	MMS	Memory Mode Select: 0 = SRAM, 1 = DRAM	0

Table 2-91: XMC Register

RCD[7:0] Refresh Counter Divider. When **ARE** is set, these bits form a divider of 1 to 256 which, along with the **RCP[1:0]** bits, determine the refresh rate in DSP clock cycles (typically 1344xFs).

RCP[1:0] Refresh Counter Prescaler. When **ARE** is set, **RCP[1:0]**, along with the **RCD[7:0]** bits, divide down the DSP clock to generate an automatic DRAM refresh (CAS-before-RAS) rate.
00 – prescale divider of 64
01 – prescale divider of 8
10 – prescale divider of 1
11 – Reserved.

ARE Automatic Refresh Enable. When set, enables automatic CAS-before-RAS DRAM refresh. The refresh rate is the DSP clock rate (typically 1344Fs), divided by the Refresh Counter Divider bits **RCD[7:0]** + 1 then divided by the prescale bits **RCP[1:0]**.

The formula is: $\frac{(\text{RCD[7:0]} + 1) \times \langle\text{RCP[1:0]}\rangle}{\langle\text{DnPCR.DFS[1:0]}\rangle}$ s. Once a refresh request is made, the refresh will occur after the current external memory cycle has finished. When **MMS** is clear, **ARE** must be clear.

MT Memory Timing. External Data Memory timing control. For exact timings, see the *External Data Memory Interface* tables in the *Electrical Characteristics* Section.
0 – Fast Timing
1 – Slow Timing

BW External Data Bus Width. Along with the **MWW** bit, determines the number of external chips needed for a given data memory size.
0 – 8-bit bus width. For a 16-bit word, two accesses are made; whereas once access is made for an 8-bit word.
1 – 4-bit bus width. The External Memory Controller will make two accesses to get an 8-bit word and four accesses to get a 16-bit word.

MWW Memory Word Width. Along with the **BW** bit, determines the number of external chips needed to support a given memory size.
0 – word size is 8 bits
1 – word size is 16 bits

MMS Memory Mode Select. Determines the type of external memory used.
0 – SRAM (**ARE** must be clear)
1 - DRAM

The Host Controller uses internal Program memory, DSP0 can use the external data memory port to expand Data memory. The port can be configured as an 8-bit wide or 16-bit wide port, through the **XMC.MWW** bit. In addition, the external data accesses can be a nibble or a byte at a time. The advantage of nibble-wide data is lower costs, and the disadvantage is that accesses are twice as slow since double the accesses are needed to get a given data word width. Likewise, the advantage of byte-wide data is that accesses for a given word-width are twice as fast, and twice the memory size is supported.

Figure 2-33 illustrates 16-bit words (**XMC.MWW** set) when using nibble-wide and byte-wide external memory (SRAM is used in this example). The lower external address bits, **MA[1:0]** for nibble-wide and **MA0** for byte-wide, are generated by on-chip hardware and are not considered part of DSP0 Base+Offset address.

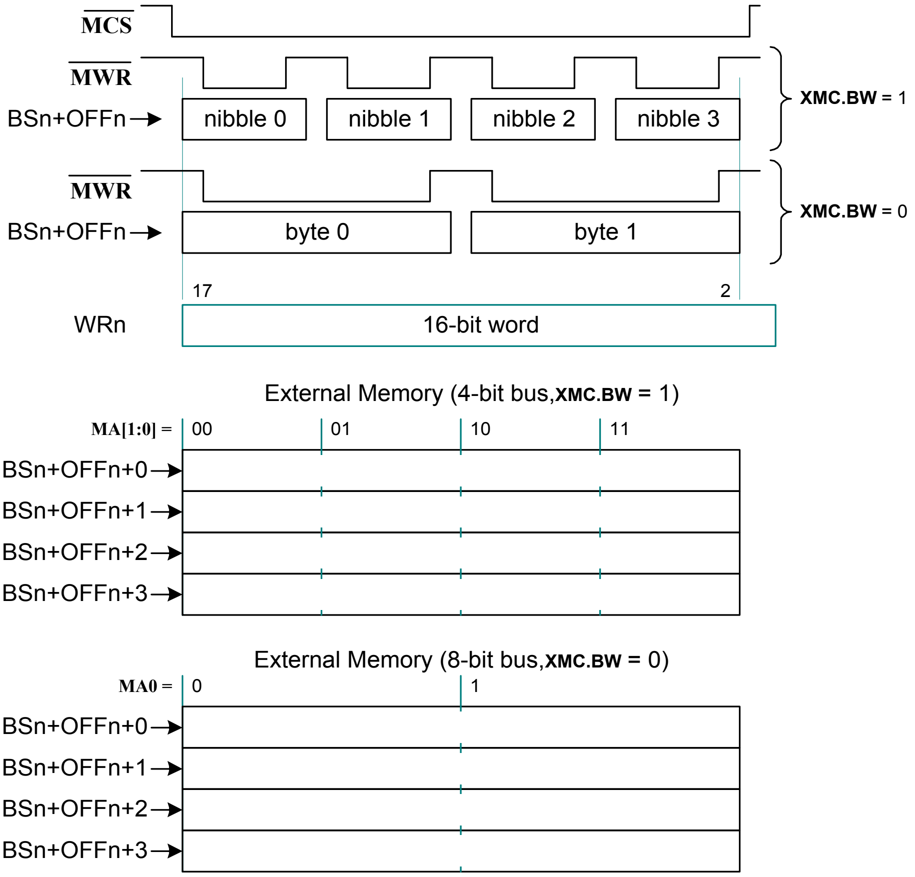


Figure 2-33: External Data Memory Port with 16-bit words (XMC.MWW set)

Figure 2-34 illustrates 8-bit words (**XMC.MWW** clear) when using nibble-wide and byte-wide external memory (SRAM is used in this example). The lower external address bit **MA0,** for nibble-wide transfers, is generated by on-chip hardware and is not considered part of DSP0 Base+Offset address. For byte-wide transfers, the Base+Offset registers generate all the external memory address bits.

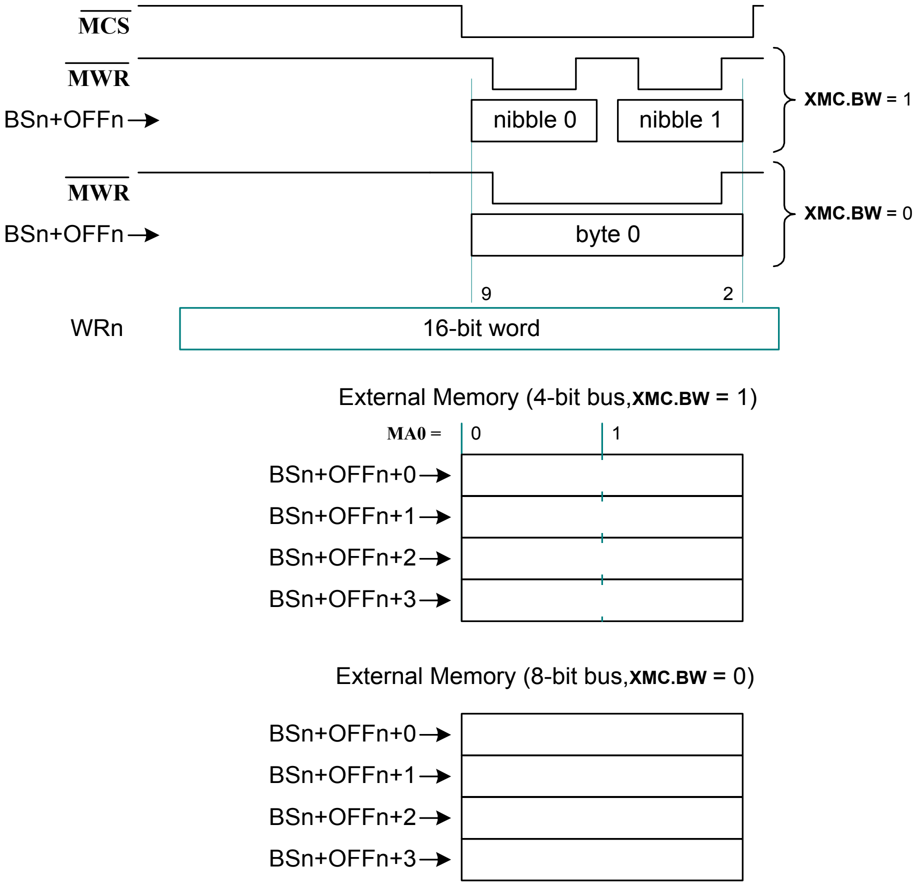


Figure 2-34: External Data Memory Port with 8-bit words (XMC.MWW clear)

2.3.13 Watchdog Timer

The Watchdog timer consists of a pre-scaler, clocked at 3072Fs, and a 16-bit counter clocked by the pre-scaler. The pre-scaler is 13 bits (divide of 8192) which produces a Watchdog timer count of 0.375Fs, or 60 µs at a Network Fs rate of 44.1 kHz. The Watchdog timer counts up and resets the chip when it reaches its terminal count of all ones. The Watchdog timer register is reset by $\overline{\text{RST}}$, a Host Controller software reset, and the Watchdog timer reset. Host Controller software must write the Watchdog timer register periodically to keep the Watchdog timer from expiring. Assuming the Host Controller writes all zeros into the register, the time-out period is 65535×60 µs = 3.96 s, when Fs = 44.1 kHz.

> The Watchdog timer powers up enabled. Therefore, if the counter reaches its terminal count, it will reset the chip. If it is not used, it must be explicitly disabled by setting **RGEN.WDD**.

4Fh WDT Watchdog Timer Host

Bit	Label	Description	Default
15..0	D[15:0]	Watchdog timer value. The watchdog timer counts up from the value written in this register at a 1/(0.375Fs) rate. If the timer hits all ones, a chip reset is produced. The timer can be disabled by setting **RGEN.WDD**.	0000h

Table 2-92: WDT Register

If the PLL is not locked, the time-out period will depend on the state of the **FLT** pin. If it is low, the time-out period can be up to four times longer than normal.

2.3.13.1 Reset Generator

The following conditions can generate the reset signal when the power supply voltage is above 2.7 to 3.0 V:

- $\overline{\text{RST}}$ pin low
- two transitions (falling edge followed by a rising edge) on **RX**, **GPA0**, **GPA1**, or **GPA2** when **CMCS.PD** is active/set. The two transitions are only detected after the power supply has stabilized.
- the Watchdog timer times out

The $\overline{\text{RST}}$ pin is level sensitive and must be low when power is applied to the device. If $\overline{\text{RST}}$ is low and the power supply is above 2.7 to 3.0 V, the internal chips reset is active. This clears the **CMCS.PD** bit and takes the chip out of the zero-power state.

The reset generator has a control and status register, RGEN, which allows the Host Controller to select which features can cause a reset and indicates the source of the previous reset. Five control bits determine which events can cause a reset, and six status bits indicate the cause of the previous reset. The status bits are only reset by writing the bits to zero (sticky) and they are not cleared by any reset source.

4Eh RGEN Reset Generator Control/Status Host

Bit	Label	Description	Default
15	GPA2D	**GPA2** toggle reset disable	0
14	GPA1D	**GPA1** toggle reset disable	0
13	GPA0D	**GPA0** toggle reset disable	0
12	RXD	**RX** toggle reset disable	0
11	WDD	Watchdog Timer disable. 0 = enabled, 1 = disabled	0
8	rsvd	Reserved. Write to 0.	0
7	XME	External Program memory enable	*
6	RSHC	Reset status - Host Controller `RESET` instruction	†
5	RSGPA2	Reset status - **GPA2** pin	†
4	RSGPA1	Reset status - **GPA1** pin	†
3	RSGPA0	Reset status - **GPA0** pin	†
2	RSRX	Reset status - **RX** pin	†
1	RSWD	Reset status - Watchdog timer	†
0	RSRST	Reset status - **RST** pin	†

* Initial state determined by the **XME/PCS** pin.
† Initial state determined by state of reset generator.

Table 2-93: RGEN Register

GPA2D **GPA2** reset disable. When **CMCS.PD** is set and **GPA2D** is clear, two transitions (falling then rising edge) on the **GPA2** pin will cause an internal reset thereby clearing the **CMCS.PD** bit and powering up the part. When **GPA2D** is set, transitions on **GPA2** have no effect and will not wake up the device. This bit is only reset by **RST** or by the power-supply monitor if the voltage is below specification.

GPA1D **GPA1** reset disable. When **CMCS.PD** is set and **GPA1D** is clear, two transitions (falling then rising edge) on the **GPA1** pin will cause an internal reset thereby clearing the **CMCS.PD** bit and powering up the part. When **GPA1D** is set, transitions on **GPA1** have no effect and will not wake up the device. This bit is only reset by **RST** or by the power-supply monitor if the voltage is below specification.

GPA0D **GPA0** reset disable. When **CMCS.PD** is set and **GPA0D** is clear, two transitions (falling then rising edge) on the **GPA0** pin will cause an internal reset thereby clearing the **CMCS.PD** bit and powering up the part. When **GPA0D** is set, transitions on **GPA0** have no effect and will not wake up the device. This bit is only reset by **RST** or by the power-supply monitor if the voltage is below specification.

RXD **RX** reset disable. When **CMCS.PD** is set and **RXD** is clear, two transitions (falling then rising edge) on the **RX** pin will cause an internal reset thereby clearing the **CMCS.PD** bit and powering up the part. When **RXD** is set, transitions on **RX** have no effect and will not wake up the device. This bit is only reset by **RST** or by the power-supply monitor if the voltage is below specification.

WDD Watchdog timer disable. When clear, the Watchdog timer, WDT, counts up and resets the device when it reaches all ones. The Host Controller must write WDT before all ones is reached to prevent a reset. If **WDD** is set, the Watchdog timer is disabled.

XME External memory enable. This bit is initialized by a pull-up/pull-down on the **XME/PCS** pin. When clear, the Host Controller uses external Program memory and the internal memory is disabled. When set, the Host Controller uses internal memory, allowing DSP0 to use the external memory port if desired (**MMPC.XMQ** and **GCTL.EDMEN** bits). Changing **XME** causes an internal reset to be generated, which resets all the on-chip peripherals, and will reset the Host Controller if the **MMPC.RSTD** bit is clear.

RSHC Reset Status - Host Controller. When set, indicates that a previous reset was due to the Host Controller executing a software `RESET` instruction. This bit is only reset by writing it to zero.

RSGPA2 Reset Status - **GPA2** pin. When set, indicates that a previous reset was due to a transition on the **GPA2** pin. This reset source can be disabled by setting the **GPA2D** bit. **RSGPA2** is only reset by writing it to zero.

RSGPA1 Reset Status - **GPA1** pin. When set, indicates that a previous reset was due to a transition on the **GPA1** pin. This reset source can be disabled by setting the **GPA1D** bit. **RSGPA1** is only reset by writing it to zero.

RSGPA0 Reset Status - **GPA0** pin. When set, indicates that a previous reset was due to a transition on the **GPA0** pin. This reset source can be disabled by setting the **GPA0D** bit. **RSGPA0** is only reset by writing it to zero.

RSRX Reset Status - **RX** pin. When set, indicates that a previous reset was due to a transition on the **RX** pin. This reset source can be disabled by setting the **RXD** bit. **RSRX** is only reset by writing it to zero.

RSWD Reset Status - Watchdog timer time-out. When set, indicates that a previous reset was due to the Watchdog timer timing out (reaching all ones). This reset source can be disabled by setting the **WDD** bit. **RSWD** is only reset by writing it to zero.

RSRST Reset Status - $\overline{\textbf{RST}}$ pin. When set, indicates that a previous reset was due to a transition on the $\overline{\textbf{RST}}$ pin. This reset source cannot be disabled. **RSRST** is only reset by writing it to zero. The $\overline{\textbf{RST}}$ reset clears the upper five control bits in this register.

2.3.14 Power Supply Monitor

The power supply monitor consists of a voltage reference and a comparator. The comparator determines whether or not a divided version of the power supply voltage is above or below the reference voltage. The reference voltage and the dividers are set to detect whether the supply voltage is above or below 2.85 V, nominally. The reference voltage has a tolerance of ±5 %, producing a power supply trip point in the range of 2.7 V to 3.0 V.

When the power supply voltage is out of range, the **CMCS.PD** bit is set and cannot be cleared. When the power supply comes back in range, the reset generator is enabled and waiting for an event to generate a reset pulse. When the $\overline{\textbf{RST}}$ pin is pulsed or the toggle detector senses activity (as specified in the RGEN register), the reset generator generates an internal reset pulse which initializes the chip and clears the **CMCS.PD** bit. Although the part goes through an under-voltage condition on initial power-up, the **CMCS.PD** default state is zero since $\overline{\textbf{RST}}$ must be asserted at power-up (clearing **CMCS.PD**).

2.4 Control Bus I/O Register Summary

The Control Bus I/O registers include the Host Controller's registers as well as peripherals and DSP Control registers, and inter-processor communication ports.

Name	Label	Addr.	Description	Page
Host Controller:				
Shadow Program Counter	SPC	00h	Swaps with PC during interrupts and subroutines	32
Shadow PC High	SPCH	58h	Swaps with the high portion of the PC, similar to SPC	33
Debug Shadow PC	DSPC	01h	Swaps with PC during debugger interrupt	33
Debug Shadow PC High	DSPCH	59h	Swaps with the high portion of the PC, similar to DSPC	33
Status	SR	02h	Controller status, page address, byte/word flag	33
Interrupt Flag	IFL	03h	Controller interrupts source status	33
Interrupt Enable	IER	04h	Enables individual interrupt sources	34
Stack Pointer	SP	05h	Identifies current stack position	34
Program Memory Page	PGMP	5Ah	Upper 2 bits when accessing Program Memory	39
Flash Memory Control	FMC	5Bh	Flash Program memory control (erase partition/pgm byte)	30
Flash Protection 8K blocks	FPBK	5Ch	Flash Protection for blocks FB15-FB1 and Info Block	31
Flash Protection FB16	FPB16	5Dh	Flash Protection for block FB16	31
COM (Internal) Ports:				
MOST COM Data First	RCF	06h	First/last address of MOST COM data reg.	44
MOST COM Data Mid	RCM	07h	Middle address of MOST COM data register	44
MOST COM Status	RCS	10h	Flags for MOST bus COM port transfers	43
DSP0 COM Data First	D0CF	08h	First/last address of control-DSP0 COM data reg.	44
DSP0 COM Data Mid	D0CM	09h	Middle address of control-DSP0 COM data register	44
DSP0 COM Status	D0CS	11h	Flags for control-DSP0 bus COM port transfers	44
DSP0 Debug COM Data First	DD0CF	13h	First/last address of control-DSP0 Debug COM data reg.	45
DSP0 Debug COM Data Mid	DD0CM	14h	Middle address of control-DSP0 Debug COM data register	45
DSP0 Debug COM Status	DD0CS	1Eh	Flags for control-DSP0 bus Debug COM port transfers	45
DSP1 COM Data First	D1CF	0Ah	First/last address of control-DSP1 COM data reg.	45
DSP1 COM Data Mid	D1CM	0Bh	Middle address of control-DSP1 COM data register	45
DSP1 COM Status	D1CS	12h	Flags for control-DSP1 bus COM port transfers	44
DSP1 Debug COM Data First	DD1CF	15h	First/last address of control-DSP1 Debug COM data reg.	46
DSP1 Debug COM Data Mid	DD1CM	16h	Middle address of control-DSP1 Debug COM data register	46
DSP1 Debug COM Status	DD1CS	1Fh	Flags for control-DSP1 bus Debug COM port transfers	45
External Ports:				
Control Port Status	CPS	0Eh	Control Port flags and control signals	52
Control Port Data	CP	0Fh	Control Port bi-directional data register	54
Debug Control Port Status	DCPS	39h	Debug Port flags and control signals	64
Debug Port Data	DCP	3Ah	Debug Port bi-directional data register	66

Table 2-94: Control Bus Register Summary

Name	Label	Addr.	Description	Page
DSP Program Control:				
DSP0 Program Control	D0PCR	17h	Control Flags and upper 10 bits of DSP0 Program memory instruction	92
DSP0 Program Data Low	D0PDL	18h	Lower 16 bits of DSP0 Program memory instruction	92
DSP0 Program Counter	D0PC	19h	DSP0 Program Counter	92
DSP0 Trap Shadow PC	D0TSPC	48h	DSP0 `TRAP` instruction Program Counter copy	46
DSP1 Program Control	D1PCR	1Ah	Control Flags and upper 10 bits of DSP1 Program memory instruction	93
DSP1 Program Data Low	D1PDL	1Bh	Lower 16 bits of DSP1 Program memory instruction	93
DSP1 Program Counter	D1PC	1Ch	DSP1 Program Counter	93
DSP1 Trap Shadow PC	D1TSPC	49h	DSP1 `TRAP` instruction Program Counter copy	46
Source Converter Control:				
Source Converter Control	FPCR	1Dh	Source Converter Routing RUN bit	88
MPX ADC Gain	GMPX	20h	Analog Gain setting for MPX ADC	90
Microphone ADC Gain	GMIC	21h	Analog gain setting for microphone ADC	90
Left Audio ADC Gain	GADL	22h	Analog gain setting for Left Audio ADC	90
Right Audio ADC Gain	GADR	23h	Analog gain setting for Right Audio ADC	90
DAC0 Attenuation	ADAC0	24h	Analog attenuation setting for DAC0 output	91
DAC1 Attenuation	ADAC1	25h	Analog attenuation setting for DAC1 output	91
DAC2 Attenuation	ADAC2	26h	Analog attenuation setting for DAC2 output	91
DAC3 Attenuation	ADAC3	27h	Analog attenuation setting for DAC3 output	91
Analog Control	ACR	28h	Data Converter enables and Audio ADC mux. select	89
Analog Control 2	ACR2	4Ah	PWM1 DAC control	90
Global Control, Memory, and Clock Manager Peripherals:				
External Memory Port Ctrl.	XMC	29h	DSP0 external data memory configuration	94
Mode Control	MMPC	37h	OS8805 revision and global control	28
Clock Manager 4	CM4	38h	Affects PWD tolerance	48
Clock Manager Ctrl./Status	CMCS	33h	PLL control, lock status, **RMCK** divider	47
Global Sync. Timer	GTR	36h	Global timing flags for processor synchronization	50
Global Control	GCTL	3Bh	GPIO Enables/Network disable	77
Watch Dog Timer	WDT	4Fh	Watchdog timer	97
Reset Generator	RGEN	4Eh	Determines cause and enables wakeup generators	98
DC Measurement ADC				
DC Meas. ADC Control	DCC	34h	Mux. select, ready flag, timing control	85
DC Meas. ADC Data	DCD	35h	ADC converstion results and resolution information	86
General Purpose Timer:				
Compare 0	CMP0	2Ah	Timer value when the **TMR0** pin toggles	74
Compare 1	CMP1	2Bh	Timer value when the **TMR1** pin toggles	74
Capture 0	CAP0	2Ch	Saves the timer value when interrupt 0 occurs	74
Capture 1	CAP1	2Dh	Saves the timer value when interrupt 1 occurs	74
Timer	TMR	2Eh	16-bit GP Timer clocked at 64×Fs	75
Timer Modulo	TMOD	2Fh	Modulo value for GP Timer	75

Table 2-94: Control Bus Register Summary (Continued)

Name	Label	Addr.	Description	Page
General Purpose I/O:				
GPIO Data	GPIO	30h	Read/write GPIO pin data for **GPA[3:0], GPB[3:0], GPC[3:0], GPD[3:0]**	75
GPIO Control	GPC	31h	Enables timer outputs and interrupt ctrl.	76
GPIO Data Direction	DDR	32h	Output enables for **GPA[3:0], GPB[3:0], GPC[3:0], GPD[3:0]**	76
EGPIO Data 1	EGPD1	4Bh	Read/write EGPIO pin data for **IOA[3:0], IOB[3:0], IOC[3:0], IOD[3:0]**	80
EGPIO Data 2	EGPD2	3Dh	Read/write EGPIO pin data for **IOE[15:0]**	80
EGPIO Data 3	EGPD3	3Eh	Read/write EGPIO pin data for **IOF[10:0], IOG[4:0]**	80
EGPIO Data Direction 1	EDD1	4Ch	Data direction for **IOA[3:0], IOB[3:0], IOC[3:0], IOD[3:0]**	81
EGPIO Data Direction 2	EDD2	3Fh	Data direction for **IOE[15:0]**	81
EGPIO Data Direction 3	EDD3	40h	Data direction for **IOF[10:0], IOG[4:0]**	82
EGPIO In Pol., Out Type 1	IPOT1	41h	Input polarity or output driver type for **IOA[3:0], IOB[3:0], IOC[3:0], IOD[3:0]**	82
EGPIO In Pol., Out Type 2	IPOT2	42h	Input polarity or output driver type for **IOE[15:0]**	82
EGPIO In Pol., Out Type 3	IPOT3	43h	Input polarity or output driver type for **IOF[10:0], IOG[4:0]**	83
EGPIO In Sticky, Out Dis. 1	ISOD1	44h	Input sticky or output disable for **IOA[3:0], IOB[3:0], IOC[3:0], IOD[3:0]**	83
EGPIO In Sticky, Out Dis. 2	ISOD2	45h	IInput sticky or output disable for **IOE[15:0]**	83
EGPIO In Sticky, Out Dis. 3	ISOD3	46h	Input sticky or output disable for **IOF[10:0], IOG[4:0]**	84
USARTS:				
USART0 Divider	U0DV	50h	Sets BAUD rate for the Control Port in USART format	69
USART0 Configuration	U0C	52h	Control Port USART0 enable, config., and interrupt status	68
USART0 Receive	U0RX	54h	Control Port USART0 receive data and status	70
USART0 Transmit	U0TX	56h	Control Port USART0 transmit data and status	70
USART1 Divider	U1DV	51h	Sets BAUD rate for the Debug Port in USART format	72
USART1 Configuration	U1C	53h	Debug Port USART1 enable, config., and interrupt status	70
USART1 Receive	U1RX	55h	Debug Port USART1 receive data and status	72
USART1 Transmit	U1TX	57h	Debug Port USART1 transmit data and status	72

Table 2-94: Control Bus Register Summary (Continued)

3 MOST Interface and Routing Bus

The MOST Processor contains a fully-compliant interface to the MOST Network and provides an efficient means to communicate streaming data between on-chip resources and the Network. The Processor consists of a ROM-coded RISC micro-controller and the appropriate logic to interface directly with the on-chip MOST fiber-optic transceiver. The MOST Processor transfers all its data across the Routing bus. The Source Converters are comprised of the mixed-signal ADCs and DACs and are described in the *Source Converters* Section.

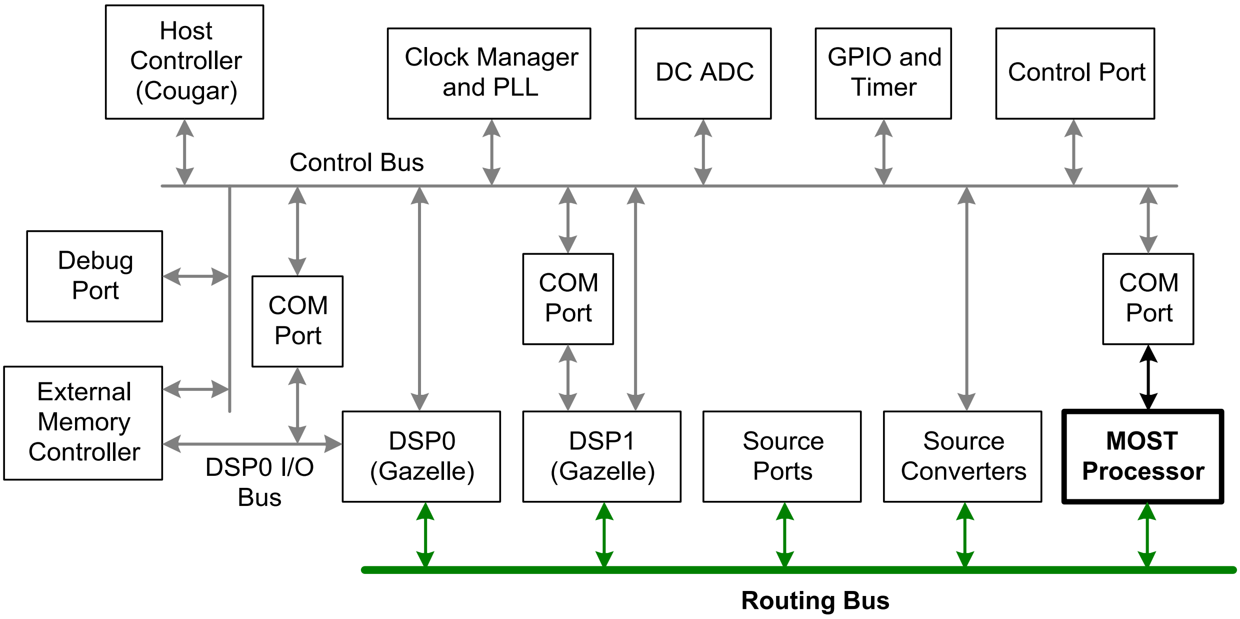


Figure 3-1: Routing Bus

3.1 Host Controller COM Port

The Host Controller COM port is an 8-bit parallel interface with some handshaking flow control signals. The MOST Processor expects a Memory Address Pointer (MAP), and then data during a write operation. During a read operation, it reads from the internal memory location pointed to by the MAP value previously written then increments the MAP pointer.

When the Controller writes the MAP byte to the COM port First/Last address register (RCF), the MOST Processor reads the corresponding MOST register and writes it back to the COM port. If the Host Controller reads this data from the Middle Address Register (RCM), MAP will automatically be incremented and the value from the new register location pointed to by MAP will be read by the MOST Processor and written to the COM port. If the Host Controller writes to the Middle Address (RCM), the MOST Processor will read the data and write it to the MOST register pointed to by MAP. Then, MAP will be incremented to point to the next register. See the Host Controller's *Inter-processor Communications* Section for more details on transferring data across the COM Port.

The MAP value is 8 bits which spans 256 bytes of the MOST Processor register space. The MOST Processor register space spans 1024 bytes or four pages of 256 bytes each. The last byte in each page is reserved for switching pages. Therefore, if the Host Controller writes the MAP to FFh (RCF = FFh), and then writes RCM to 0, 1, or 3 (page 2 is reserved); then the MOST Processor switches to the respective memory page. When the MAP is auto-incrementing and reaches the end of a page, it wraps to the beginning of the same page.

3.2 MOST Transceiver

The MOST Transceiver interfaces to the MOST Optical Network. The receiver section, in conjunction with the Clock Manager peripheral of the Host Controller, recovers the clock, decodes the data, and passes the information to the Source Ports or DACs. The transmitter accepts data from the Source Ports or ADCs, or the MOST receiver, encodes the data, and transmits it across the Network. The transceiver can be configured as a Network timing-master or timing-slave. As a timing-slave, the transmitter (and the rest of the chip) is clocked by the receiver. The received bit stream is delayed by 2.5 bi-phase bit periods and retransmitted.

As a timing-master, the timing of the transmitter (and the rest of the chip) is determined by **SCK**, the external crystal, or an SPDIF channel entered through the source data port. The receiver PLL multiplies this timing source up to 3072xFs. The PLL-control resides in the Clock Manager peripheral on the Control bus. The receiver input pin, **RX**, is over-sampled by a high frequency clock and data is recovered by a digital state machine. Source data can only be transferred while the system is locked (**CMCS.LOCK** set). When the node goes from unlock to the lock state, three frames are required to synchronize the internal buffers before source data is correctly transferred by the MOST Processor.

3.2.1 MOST Routing Table (MRT) and Routing Ports

The MOST Processor uses the MOST Routing Table (MRT) to link Source Peripherals. Routing synchronous data between on-chip peripherals and the MOST Network is accomplished by filling the MOST Routing Table (MRT) with the addresses of where data should come from (the sources). The MRT specifies destinations where data is to be sent. Source Peripheral destinations include the MOST Network (transmit), DACs, DSPs (DR register in the DSP I/O space), and Source Ports (**SX0** and **SX1**). Sources are defined by MOST Routing Addresses (MRA). Source Peripheral sources include the MOST Network (receive), ADCs, DSPs (DX register in the DSP I/O space), and Source Ports (**SR0** and **SR1**). Routing is accomplished by placing the address of a source (MRA) into the MRT location for a particular destination.

The peripheral routing is synchronized to the **SCK** pin when the Source Ports are enabled. Therefore, for the upper MRT to work properly when the Source Ports are enabled (default state), **SCK** must be configured as an output, or when **SCK** and **FSY** are configured as inputs, proper external clocks must be applied.

The following Tables, determine how the MOST Processor transfers data across the Routing bus. The MRT is a memory location for outgoing data to the MOST Network, DSPs, Source Converter DACs, or Source Ports. The Tables also list the MOST routing addresses (MRA) for data coming from the MOST Network, DSPs, Source Converter ADCs, or Source Ports. For routing incoming data to an outgoing location, the incoming data address is placed in the outgoing MRT memory location.

Source Device	MOST Routing Locations (outputs) and Addresses (inputs) (Hex)
MOST Network	00-3B
SR0, SX0 port pins *	40, 48, 50, 58, 60, 68, 70, 78
SR1, SX1 port pins	41, 49, 51, 59, 61, 69, 71, 79
DSP0 Routing port - low byte	42, 4A, 52, 5A, 62, 6A, 72, 7A
DSP0 Routing port - high byte	43, 4B, 53, 5B, 63, 6B, 73, 7B
DSP1 Routing port - low byte	44, 4C, 54, 5C, 64, 6C, 74, 7C
DSP1 Routing port - high byte	45, 4D, 55, 5D, 65, 6D, 75, 7D
Source Converters - low byte	46, 4E, 56, 5E, 66, 6E, 76, 7E
Source Converters - high byte	47, 4F, 57, 5F, 67, 6F, 77, 7F

* When configured as SPDIF, left audio (most to least significant) is 50, 48, 40 and right audio is 70, 68, 60.

Table 3-1: MOST Processor MRA and MRT

The words within the Source Converters are arranged as follows:

Source Converters	Addresses and Locations (Hex)	
	Upper Byte	Lower Byte
Inputs (MRA address to be placed in MRT):		
Left Audio ADC	47	46
MPX	4F	4E
Right Audio ADC	57	56
MPX	5F	5E
Mic ADC	67	66
MPX	6F	6E
reserved	77	76
MPX	7F	7E
Outputs (MRT locations):		
DAC0	47	46
reserved	4F	FE
DAC1	57	56
reserved	5F	5E
DAC2	67	66
reserved	6F	6E
DAC3	77	76
reserved	7F	7E

Table 3-2: Source Converter Routing MRA and MRT

The Source Converters are routed in a similar fashion. Table 3-2 lists the MOST Processor addresses for the Source Converters. The MPX ADC is listed four times since it runs at 4xFs. The Mic ADC only runs at 0.25xFs; therefore, the data is repeated four times when transferring to the MOST Network (always at Fs).

Table 3-2 lists the MRT and defines the Source Peripheral destinations. Routing is accomplished by filling this table with the source address (MRA), which defines where the data comes from. The Host Controller fills this table through the MOST COM port. The MOST Processor then uses this table to route source data to the destinations. The first half of the table supports the 60 bytes of Synchronous data on the MOST Network (transmit data) and the last half of the table is peripherals that sink data (destinations).

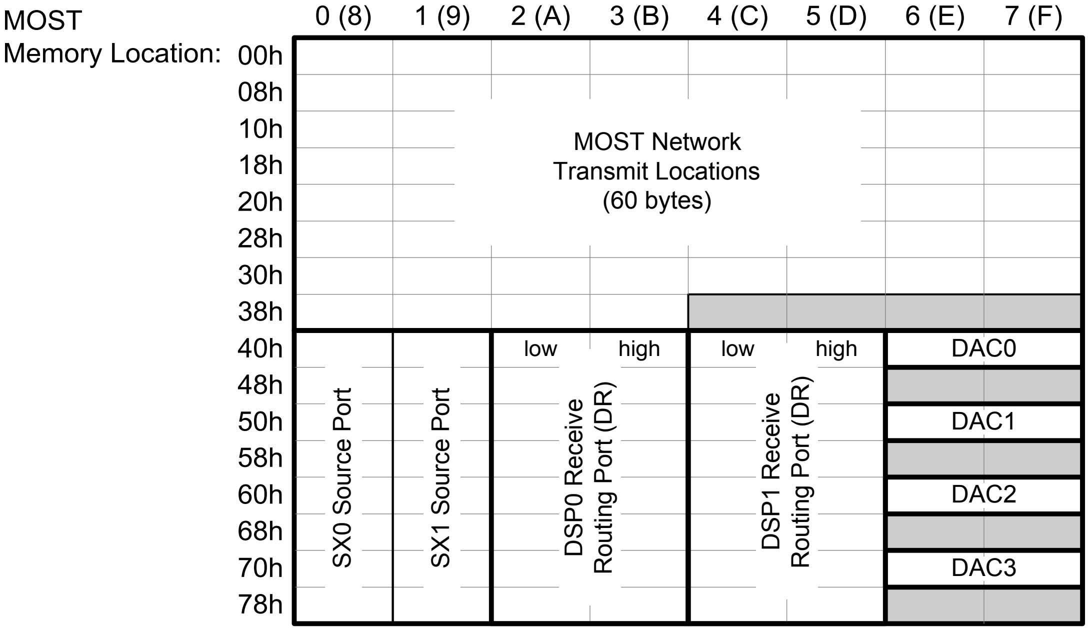


Figure 3-2: MOST Routing Table (Destinations)

Figure 3-3 illustrates the addresses for Source data. These addresses should be placed in the MRT to connect a source with a destination. The first half of the addresses are the received data from the MOST Network and the last half of the addresses are the peripherals that are data sources.

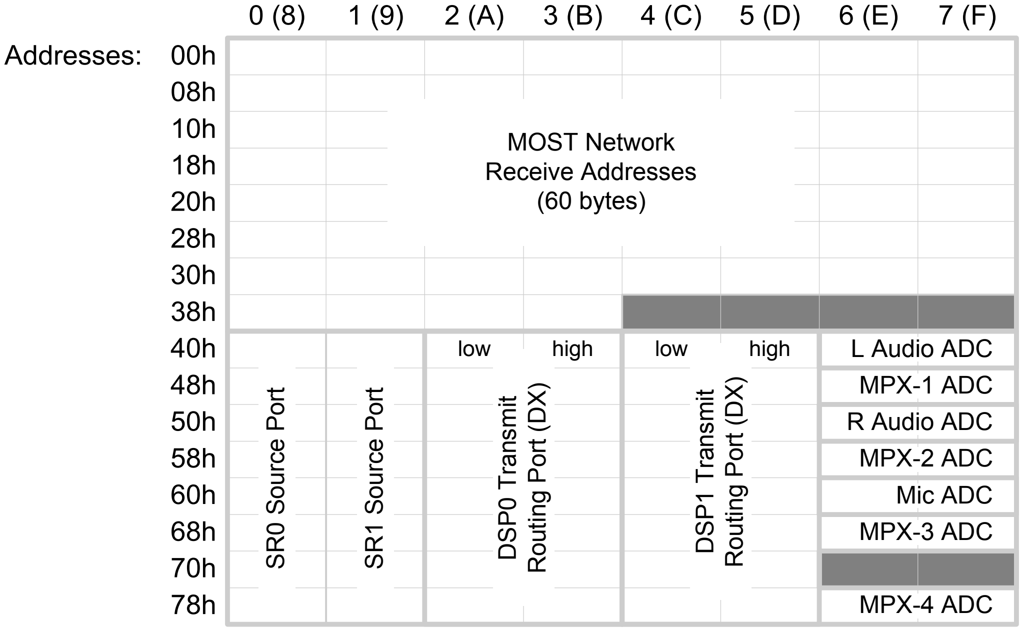


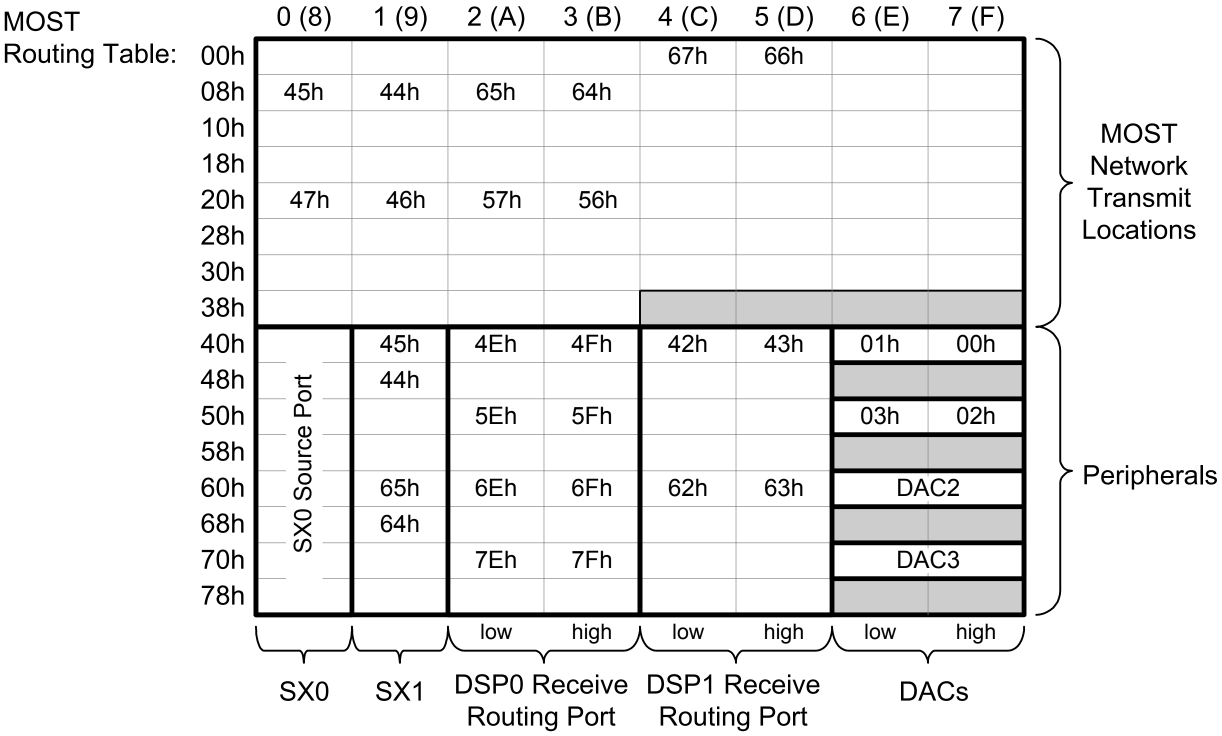
Figure 3-3: MRA Routing Addresses (Sources)

The special address reference F8h forces zero's to any desired output channel. Placing F8h in an MRT register tells the MOST Processor to output zero (0x00) for that particular destination byte. This may be used to force unused channels to zero (mute) or as a temporary remedy to mute all output peripheral channels when lock is lost.

To illustrate how to use the MRT, the following example includes:

- Sending the Mic (source) to the MOST Network Synchronous bytes 04h and 05h.
- Sending Stereo audio data from the MOST Network received synchronous bytes 00h-03h to DAC0 (for left) and DAC1 (for right).
- Sending the MPX ADC data to DSP0, who then processes the data and outputs a stereo audio signal (4 bytes), which is sent to DSP1 for further processing.
- DSP1 takes the stereo audio data from DSP0, does its processing, and then outputs the data on both the MOST Network, synchronous bytes 08h-0Bh, and Source Port **SX1**. This illustrates that source addresses can go to multiple destinations.
- The audio ADCs are sent to the MOST Network, Synchronous bytes 20h-23h
- Lastly, any MOST Transmit bytes not used by the peripherals are connected to the same byte from the MOST Receiver. This allows flow-through of channels not used by this chip.

The MRT filled with the source addresses for this example is illustrated in Figure 3-4. The addresses to fill in the MRT are gotten from Figure 3-3.



MOST Routing Table:

	0 (8)	1 (9)	2 (A)	3 (B)	4 (C)	5 (D)	6 (E)	7 (F)
00h					67h	66h		
08h	45h	44h	65h	64h				
10h								
18h								
20h	47h	46h	57h	56h				
28h								
30h								
38h					▨	▨	▨	▨
40h	SX0 Source Port	45h	4Eh	4Fh	42h	43h	01h	00h
48h		44h					▨	▨
50h			5Eh	5Fh			03h	02h
58h							▨	▨
60h		65h	6Eh	6Fh	62h	63h	DAC2	
68h		64h					▨	▨
70h			7Eh	7Fh			DAC3	
78h							▨	▨
			low	high	low	high	low	high
	SX0	SX1	DSP0 Receive Routing Port		DSP1 Receive Routing Port		DACs	

MOST Network Transmit Locations
Peripherals

Figure 3-4: MRT Example - Step 1

To send the Mic data to the MOST Network, the Host Controller would have to write, through the MOST COM port, location 04h as the MAP (Memory Address Pointer). Then the Controller would write 67h, which goes to location 04h, and 66h, which goes to location 05h. For MOST Network synchronous data greater than one byte (channel), the most significant byte generally goes to the lowest location.

The second step is to get the MOST Network received stereo audio data from synchronous channels 00h-03h and send them to DAC0 and DAC1. The MOST Network receive addresses are placed in the MRT at the DAC0 and DAC1 destinations. Therefore, MRT locations for DAC0, 47h and 46h, must be programmed with the left channel addresses from the MOST received synchronous data, 00h and 01h, respectively. Likewise, the MRT locations for DAC1, 57h and 56h, must be programmed with the right channel addresses from the MOST Network, 02h and 03h, respectively.

The third step is to send the MPX ADC data to DSP0. The MPX ADC outputs 16-bit samples at 4xFs. The DSP0 receive routing port (DR) may be sampled at up to 8xFs. Which four of the eight DSP locations are used is entirely up to the programmer. For this example, the four MPX samples are spaced evenly in the DSP0 receive routing port: locations 42h, 43h for the first MPX sample (addresses 4Eh, 4Fh) through the last MPX sample (addresses 7Eh, 7Fh) at DSP0 receive locations 72h, 73h. DSP0 could use the **GTR.FS4** flag (through interrupt or polling) to synchronize to the MPX data sent by the MOST processor.

DSP0 processes the MPX ADC data and outputs a stereo stream that is sent to DSP1 for further processing. DSP1 gets DSP0's data by placing DSP0's transmit addresses 42h, 43h into DSP1's receive MRT locations 44h and 45h; and DSP0's transmit addresses 62h, 63h into DSP1's MRT locations 64h, 65h. As with the previous sequence, the particular two sampled used out of the eight DSP samples is purely arbitrary. For this example, each sample is spaced halfway through the DSP ports allowing the **GTR.FS2** flag to be used for synchronization between the two DSPs.

In the next step, DSP1 processes DSP0's output data and then sends the data to the MOST Network and the **SX1** Source Port. This illustrates that source data can be used for multiple destinations. Since DSP1 is outputting stereo data, the 2-of-8 samples selected are the same as the incoming data (spaced halfway through the 8xFs DSP routing port so **GTR.FS2** may be used to load the DSP's DX register). To output these two DSP1 samples onto the MOST transmit Network, the MOST MRT locations 08h-0Bh get 45h, 44h for the first sample, and 65h, 64h for the second. To send the same data to the **SX1** Source Port, the **SX1** Source Port MRT locations 41h, 49h get the DSP1 addresses 45h, 44h; and the **SX1** MRT locations 61h, 69h, get the DSP1 addresses 65h, 64h. This spaces the Source Port stereo data at the edges of the **FSY** signal (MSB-justified). Assuming the Source port is using 64-bits per frame, the Source Port timing is illustrated in Figure 3-5.

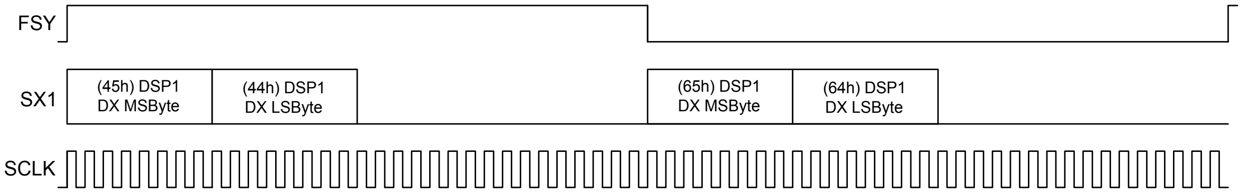
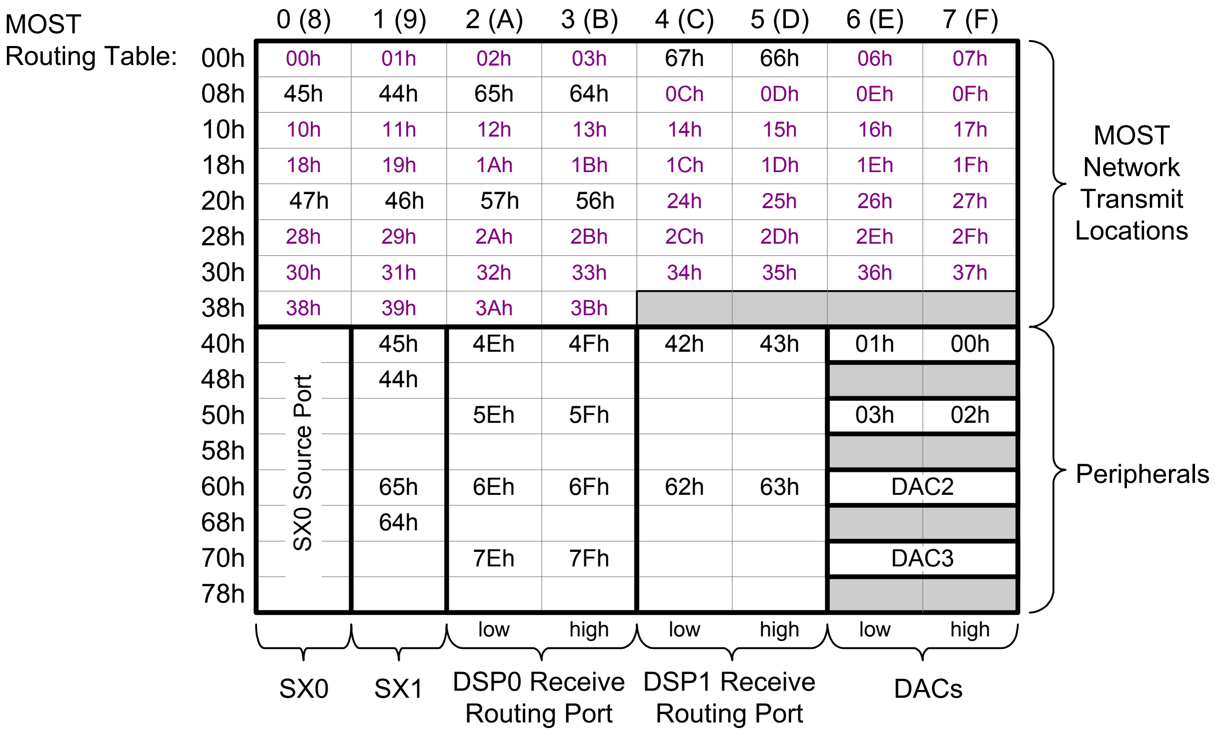


Figure 3-5: MRT Example - Source Port Routing

The last peripheral routing involves sending the left and right audio ADCs out on the MOST Network. This example uses the MRT locations 20h-23h. The standard ordering calls for left before right and most-significant byte first (at lowest address). Therefore, using Figure 3-3 addressing, the Left Audio ADC MS-Byte address (47h) is placed at MRT 20h and the LS-Byte address (46h) is placed at MRT 21h. For the Right Audio ADC, the MS-Byte address (57h) goes in MRT location 22h, and the LS-Byte address (56h) goes in MRT location 23h.

Now that all the routing is finished (and illustrated in Figure 3-4), the rest of the MOST Transmit synchronous data is generally filled with the address of the corresponding MOST receive synchronous data - which is the power-up default value. This supports flow-through data from other devices in the MOST Network. The finished MRT is illustrated in Figure 3-6. To further explain, the MOST received synchronous data channels 00h-03h are sent to DAC0 and DAC1. The same data is also passed through to the MOST transmit synchronous network (MRT locations 00h-03h) for other devices on the MOST Network to use.



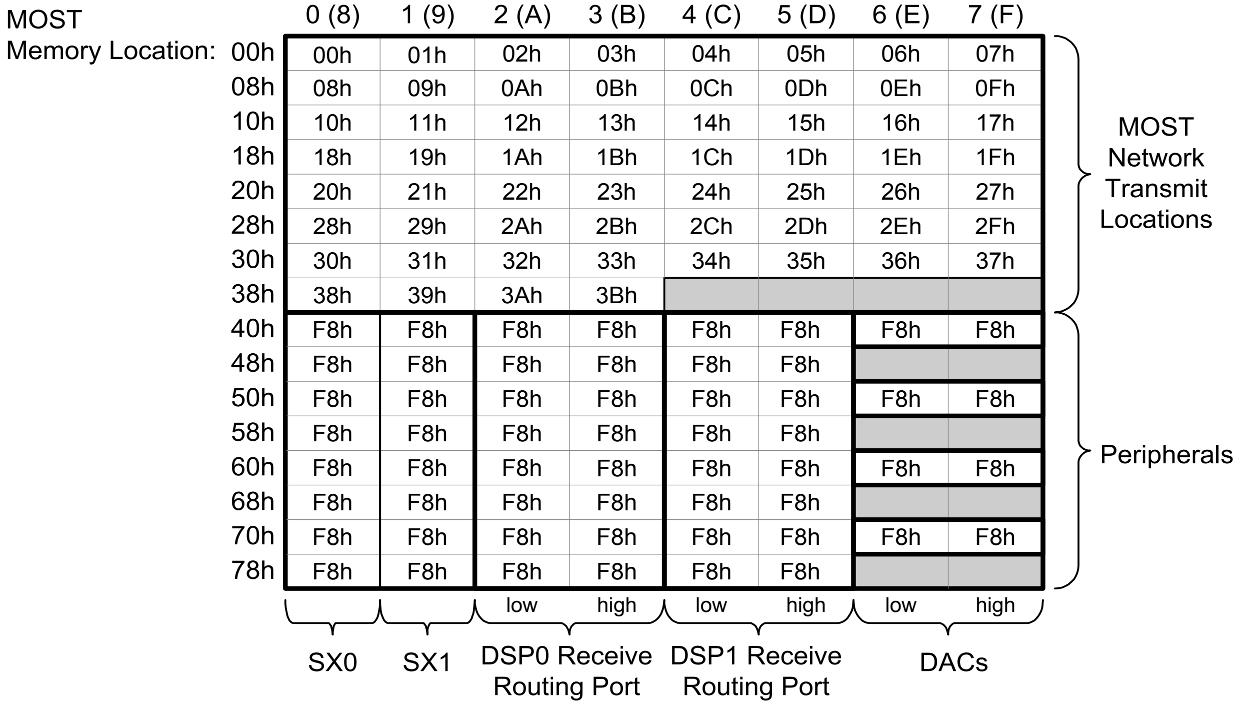
MOST Routing Table:

	0 (8)	1 (9)	2 (A)	3 (B)	4 (C)	5 (D)	6 (E)	7 (F)
00h	00h	01h	02h	03h	67h	66h	06h	07h
08h	45h	44h	65h	64h	0Ch	0Dh	0Eh	0Fh
10h	10h	11h	12h	13h	14h	15h	16h	17h
18h	18h	19h	1Ah	1Bh	1Ch	1Dh	1Eh	1Fh
20h	47h	46h	57h	56h	24h	25h	26h	27h
28h	28h	29h	2Ah	2Bh	2Ch	2Dh	2Eh	2Fh
30h	30h	31h	32h	33h	34h	35h	36h	37h
38h	38h	39h	3Ah	3Bh	▨	▨	▨	▨
40h	SX0 Source Port	45h	4Eh	4Fh	42h	43h	01h	00h
48h		44h					▨	▨
50h			5Eh	5Fh			03h	02h
58h							▨	▨
60h		65h	6Eh	6Fh	62h	63h	DAC2	
68h		64h					▨	▨
70h			7Eh	7Fh			DAC3	
78h							▨	▨
			low	high	low	high	low	high

MOST Network Transmit Locations
Peripherals
SX0
SX1
DSP0 Receive Routing Port
DSP1 Receive Routing Port
DACs

Figure 3-6: MRT Example - Finished Table

The power-up default values for the MRT are illustrated in Figure 3-7. The MRT section for the MOST Network transmit section is filled with the corresponding MOST receive bytes. Therefore, when the transceiver is taken out of *All-Bypass* mode, all the bytes received from the MOST Network will be retransmitted. The stream peripheral section of the MRT (upper half) contains F8h by default, which sends zeros to any of the peripherals that are enabled.



MOST Memory Location:	0 (8)	1 (9)	2 (A)	3 (B)	4 (C)	5 (D)	6 (E)	7 (F)	
00h	00h	01h	02h	03h	04h	05h	06h	07h	MOST Network Transmit Locations
08h	08h	09h	0Ah	0Bh	0Ch	0Dh	0Eh	0Fh	
10h	10h	11h	12h	13h	14h	15h	16h	17h	
18h	18h	19h	1Ah	1Bh	1Ch	1Dh	1Eh	1Fh	
20h	20h	21h	22h	23h	24h	25h	26h	27h	
28h	28h	29h	2Ah	2Bh	2Ch	2Dh	2Eh	2Fh	
30h	30h	31h	32h	33h	34h	35h	36h	37h	
38h	38h	39h	3Ah	3Bh					
40h	F8h	F8h	F8h	F8h	F8h	F8h	F8h	F8h	Peripherals
48h	F8h	F8h	F8h	F8h	F8h	F8h			
50h	F8h	F8h	F8h	F8h	F8h	F8h	F8h	F8h	
58h	F8h	F8h	F8h	F8h	F8h	F8h			
60h	F8h	F8h	F8h	F8h	F8h	F8h	F8h	F8h	
68h	F8h	F8h	F8h	F8h	F8h	F8h			
70h	F8h	F8h	F8h	F8h	F8h	F8h	F8h	F8h	
78h	F8h	F8h	F8h	F8h	F8h	F8h			
			low	high	low	high	low	high	
	SX0	SX1	DSP0 Receive Routing Port		DSP1 Receive Routing Port		DACs		

Figure 3-7: MRT Power-Up Defaults

Figure 3-2 and Figure 3-3 depict the Source Port signals supporting the maximum of eight bytes each. The full eight bytes are used when the Source Ports are programmed for 64 bits per frame (**bSDC1.NBR[1:0]** = 00). Table 3-3 depicts the MOST Routing Table locations for all the different bit-per-frame formats supported on **SX0** and **SX1** and Table 3-4 depicts the MOST Routing Addresses for **SR0** and **SR1**.

Signal	Bits per Frame	Byte							
		First							**Last**
SX0	*64 - in/out	40h	48h	50h	58h	60h	68h	70h	78h
	48 - out	40h	48h		58h	60h	68h		78h
	32 - out	40h		58h		60h		78h	
	*32 - in	40h		50h		60h		70h	
	SPDIF	40h	48h	50h	58h	60h	68h	70h	78h
SX1	*64 - in/out	41h	49h	51h	59h	61h	69h	71h	79h
	48 - out	41h	49h		59h	61h	69h		79h
	32 - out	41h		59h		61h		79h	
	*32 - in	41h		51h		61h		71h	

* When **SCK** is an input (in), the clock is assumed continuous (not a gated clock).

Table 3-3: Source-Port MOST Routing Table for SX0, SX1

Signal	Bits per Frame	Byte							
		First							Last
SR0	*64 - in/out	40h	48h	50h	58h	60h	68h	70h	78h
	48 - out	40h	50h		58h	60h	70h		78h
	32 - out	50h		58h		70h		78h	
	*32 - in	48h		58h		68h		78h	
	SPDIF	40h	48h	50h	58h	60h	68h	70h	78h
SR1	*64 - in/out	41h	49h	51h	59h	61h	69h	71h	79h
	48 - out	41h	51h		59h	61h	71h		79h
	32 - out	51h		59h		71h		79h	
	*32 - in	49h		59h		69h		79h	

* When **SCK** is an input (in), the clock is assumed continuous (not a gated clock).

Table 3-4: Source-Port MOST Routing Addresses for SR0, SR1

The MRT locations for **SX0** (**SX1**) and the MRA addresses for **SR0** (**SR1**) with respect to timing on the Source Ports is shown pictorially in Figure 3-8.

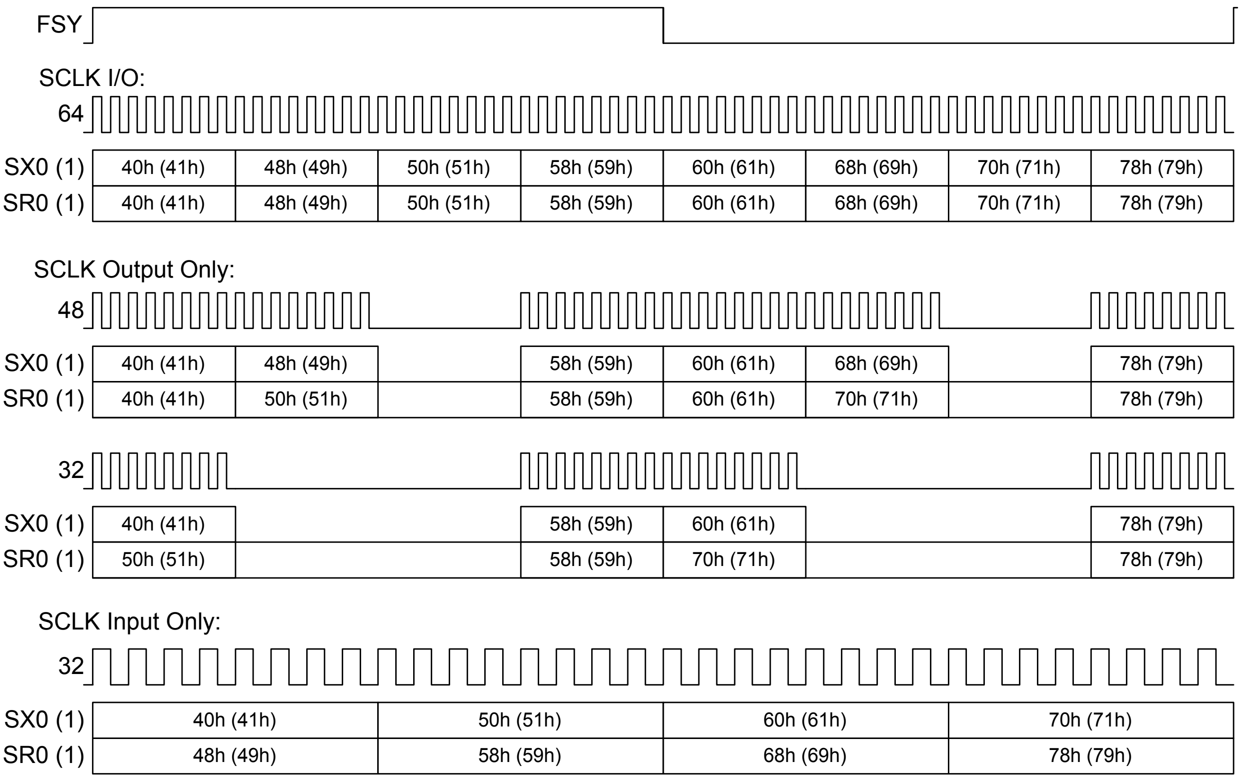


Figure 3-8: MRT Addresses with respect to Source Port Timing

3.2.1.1 Routing Limitations

Certain routing limitations exist when routing from peripheral data sources to sinks. No limitations exist when routing peripherals to or from the MOST Network, or routing from the receive to transmit portions of the MOST Network. These routing limitations exist due to the asynchronous nature of an **FSY** (input or output) that all the Source Peripherals are synchronized to, and the MOST Network receive and transmit start of frame.

To guarantee that all transmitted bytes are delayed the same amount (entered the chip at the same time - or are coherent) when entering MRA Source Peripherals into MRT Source Peripherals (destinations) MRT locations 40h-47h should not contain any of the MRA values between 78h and 7Fh. In addition, for systems that do not comply with the *MOST Specification of Physical Layer*, the delay from transmit start-of-frame (**TX**) to receive start-of-frame (**RX**) should be limited to less than 9/16ths of a frame. This condition only applies to the timing-master node. If the MOST Physical Layer specification is met, then a network delay this large cannot occur.

As an example, if DSP0 transferred a stereo 16-bit data to DAC0/1 and its DX register is loaded at times corresponding to MRA 7Ah (left sample) and 5Ah (right sample); then MRT46 = 7Ah, MRT47 = 7Bh for left, and MRT56 = 5Ah, MRT57 = 5Bh for right. The stereo pair would not be coherent, since:

- DAC0 resides in the MRT area of 40h-47h, and
- the DSP0 output routing of 7Ah/7Bh are within the MRA values (78h - 7Fh) that should not be routed to the MRT values above.

However, if the left and right samples are swapped, so MRA 5Ah is the left sample and 7Ah is the right sample, then MRT46/47 = 5Ah/5Bh and MRT65/57 = 7Ah/7Bh, the routing limitation is avoided, and the stereo sample is coherent. Therefore, this routing limitation can be avoided by judicious use of the MRA values when routing to MRT locations 40h to 47h.

3.2.2 SPDIF Mode (SR0/SX0)

The Source Port pins **SX0** and **SR0** can be configured to support SPDIF by setting **bSDC1.MOD[1:0]** = 10. In addition, if the chip is the Network timing-master, the master clock source can be the SPDIF data stream on **SR0** by setting **CMCS.MX[1:0]** = 11.

An SPDIF frame consists of two sub-frames: the left channel and the right channel sub-frame. Each sub-frame consists of a pre-amble for identification, 24 audio bits, three control bits (V, U, C), and a parity bit P. Three MRT addresses support the 24 audio bits per sub-frame. The fourth address contains the V, U, and C control bits. The MRT addresses for SPDIF data are illustrated in Figure 3-9.

A synchronous block bit, **SBK** is added to the control byte, and is high in the right sub-frame just prior to the start of an SPDIF channel status block boundary (the Z preamble), on the received SPDIF data (**SR0**).

	7			4		2	1	0
Left MRT = 58h Right MRT = 78h				SBK		C	U	V

Table 3-5: SPDIF Sub-frame Control Byte

For the transmitted SPDIF data on **SX0**, the Z preamble is transmitted every 192 SPDIF frames. If the control byte received from the Routing bus contains the SBK bit, then the transmitted data will be resynchronized to the SBK bit, by transmitting the Z preamble in the next transmitted frame on **SX0**.

Source data is transmitted across the Most Network MSB-first. Since S/PDIF data is transmitted LSB-first, each byte is bit-reversed before transmission. Therefore, data bytes received from **SR0** are bit-reversed before transmitting across the Routing bus. Likewise, bytes received from the Routing bus are bit-reversed before transmitting out the **SX0** pin.

The SPDIF receiver checks for parity and bi-phase coding errors. If an error occurs, the validity bit (V) of the erred sub-frame is automatically set.

MOST Network synchronous data that is greater than one byte (channel) generally places the most significant byte in the lowest MRT location. To place the 16 most significant bits of SPDIF left data from the received **SR0** pin on the MOST Network, channels 0 and 1 and the right SPDIF data on channels 2 and 3; the received-SPDIF most-significant-byte address 50h would be placed in the MRT table location 00h and the middle-byte address 48h would be placed in MRT location 01h. For the right channel, the received-SPDIF address 70h would be placed in MRT location 02h, and SPDIF address 68h in MRT location 03h.

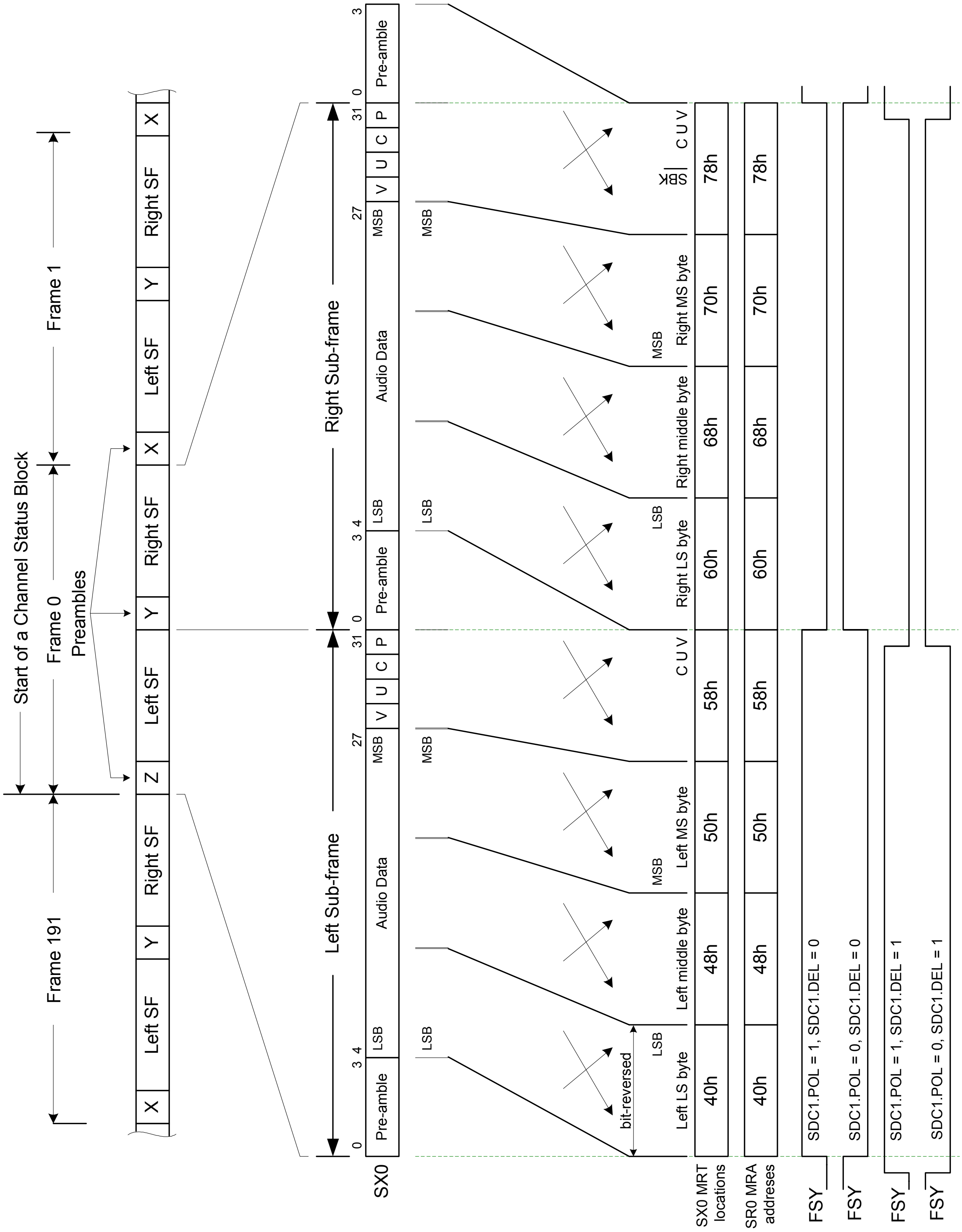


Figure 3-9: Source Port S/PDIF Format MRT/MRA

3.2.3 MOST Configuration Registers

For the MOST Processor registers, a prefix is added to the register name to help discern between byte and word registers, as well as buffers. A prefix of ‘b’ indicates a byte-wide register. A prefix of ‘w’ indicates a 16-bit word-wide register, and a prefix of ‘m’ indicates a multi-byte buffer.

80h bXCR Transceiver Control Register MOST

Bit	Label	Description	Default
7	MTR	timing-Master/Slave select	0
6	OE	Transmitter output enable	0
5	NMEN	MOST Network enable. 1 is Network enabled. 0 is factory test mode.	1
4	rsvd	Reserved, Write to 0.	0
3	rsvd	Reserved, Write to 0.	0
2	SBY	Source data bypass	0
1	$\overline{\text{ABY}}$	All Bypass	0
0	rsvd	Reserved, Write to 0	0

Table 3-6: bXCR Register

MTR Timing Master/Slave select.

0 – Device is MOST Network timing-salve node. **RX** must be set as timing source (**CMCS.MX[1:0]** = 10).

1 – Device is MOST Network timing-master node. Only one device can be timing-master in the Network. The timing source, set through **CMCS.MX[1:0]** bits, cannot be **RX**. When set to timing-master, bSBC must be configured and a *Deallocate All* message must be sent.

OE Transmitter Output Enable. Only has affect when $\overline{\textbf{ABY}}$ = 1.

0 – Transmitter output disabled. Output held low.

1 – Transmitter output enabled.

NMEN Network Mode Enable.

0 - A factory test mode.

1 - Normal Mode operation.(must be set for normal operation).

SBY Source data Bypass for timing-slave nodes (**MTR** clear).

0 – Synchronous source data can be exchanged with the Network. The node increments the system’s delay counter (bNDR) by one and the delay through the node for synchronous source data is two frames. Routing to and from the MOST Network is managed by the MRT.

1 – Source data directly bypassed within the transceiver. No routing to/from the MOST Network is possible, and bNDR is not incremented.

$\overline{\text{ABY}}$ All Bypass.

0 – Transceiver is bypassed. Part not visible to the Network, and bNPR is not incremented. Signal at **RX** electrically connected to **TX**. The delay from **RX** to **TX** is approximately 7 ns.

1 – OS8805 interacts with the Network. Data can be sourced and sinked. Control messages can be sent. The delay from **RX** to **TX** is one bit period $\left(\frac{1}{512 \times Fs}\right)$ plus 7 ns. The node increments the system’s position counter (bNPR).

81h bXSR *Transceiver Status Register* MOST

Bit	Label	Description	Default
7	rsvd	Reserved, Write to 0.	0
6	MSL	Mask SPDIF lock error	0
5	MXL	Mask transceiver lock error	0
4	ME	Mask coding error	0
3	ERR	All error capture	0
2	rsvd	Reserved, Write to 0	0
1	ESL	Error capture SPDIF	0
0	EXL	Error capture transceiver	0

Table 3-7: bXSR Register

MSL Mask SPDIF lock error.
0 – SPDIF lock error captured by **bXSR.ERR** and **ERR** pin.
1 – SPDIF lock error ignored.

MXL Mask transceiver lock error.
0 – Transceiver lock error captured by **bXSR.ERR** and **ERR** pin.
1 – Transceiver lock error ignored.

ME Mask coding error. A coding error is a bi-phase violation or parity error from the Network bit-stream (does not apply to S/PDIF input).
0 – Coding error captured by **bXSR.ERR** and **ERR** pin.
1 – Coding error ignored.

ERR All error capture. Errors not masked by **MSL**, **MXL**, or **ME**. When set, indicates an error occurred in an unmasked error condition, or a power-on ready condition. This bit is sticky and cleared by writing a 0 to it. This bit is OR'ed with **bMSGS.ERR.** If **bMSGS.ERR** is set, and the interrupt enable **bIE.IERR** is set, the MOST Routing Processor will set **RCS.INT**, which can interrupt the Host Controller. See Figure 3-10.

ESL Error capture SPDIF. When set, indicates an error occurred on the SPDIF incoming source data port. This bit is sticky and cleared by writing a 0 to it.

EXL Error capture transceiver. When set, indicates that an error occurred on the network transceiver. This bit is sticky and is cleared by writing a 0 to it.

84h bNC *Network Control* MOST

Bit	Label	Description	Default
7..2	rsvd	Reserved, Write to 0.	000000
1	APREN	Asynchronous Packet Receive Enable.	0
0	RWD	*Remote Write* disable	0

Table 3-8: bNC Register

APREN Asynchronous Packet Receive Enable. This bit must be clear on revisions prior to G, which do not support asynchronous packets.
0 – MOST Processor ignores received Asynchronous Packets.
1 – Enables reception of Asynchronous Packets. MOST Processor high-speed bit **MMPC.RFS1** must be set to meet the proper timing when **APREN** is set.

RWD *Remote Write* Disable.

0 – *Remote Write* Control messages (transmit message type bXTYP = 02h) are supported. Other nodes are allowed to write to page 0 registers of the MOST Processor.

1 – *Remote Write* Control messages (transmit message type bXTYP = 02h) are blocked. Other nodes cannot write to any of the MOST Processor registers; however, *Remote Reads* are allowed. Nodes sending *Remote Write* messages will get the bXTS response of 0x11 (transmit message type not supported).

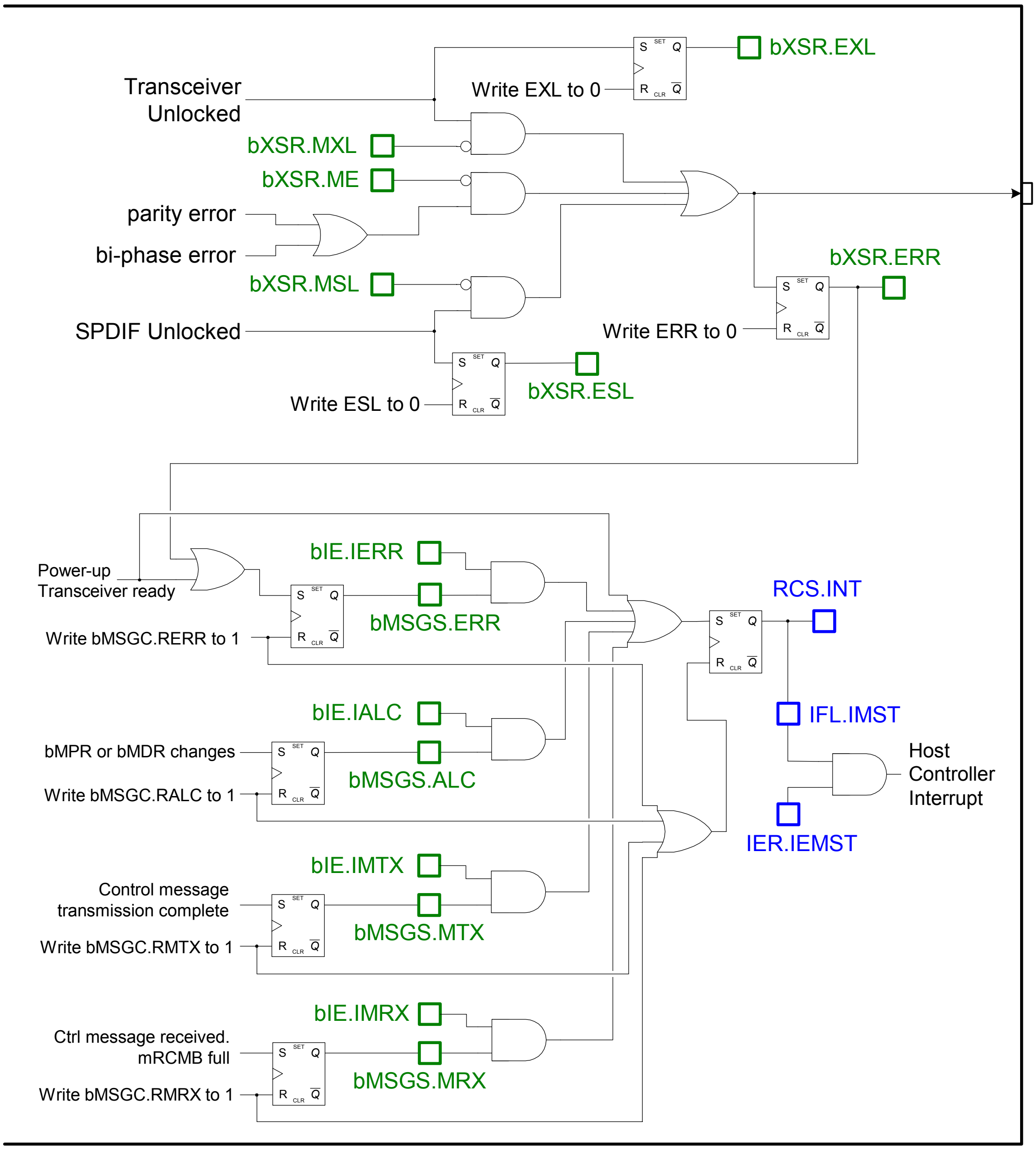


Figure 3-10: MOST Transceiver Errors and ERR Pin

87h bNPR *Node Position Register* MOST

Bit	Label	Description	Default
7..0	NP[7:0]	Node Position Register.	00h

Table 3-9: bNPR Register

NP[7:0] Node Position. Represents the chip’s physical node position in the network. Enumeration starts with 00h at the timing-master node. The timing-slave node connected to the timing-masters TX pin, has node position 01h, and so on. Nodes in all-bypass mode (**bXCR.$\overline{\text{ABY}}$** clear) are invisible to the Network and do not increment **NP[7:0]**. Sets the node’s physical address to 0x400 + **NP[7:0]**. Not valid until after Network lock (**bCM2.$\overline{\text{LOK}}$**) is established.

88h bIE *Interrupt Enable* MOST

Bit	Label	Description	Default
7..4	rsvd	Reserved. Write to 0.	0000
3	IALC	Allocation change interrupt enable. When set, allows interrupts when **bMSGS.ALC** gets set.	0
2	IERR	MOST Network Error interrupt enable. When set, allows interrupts when **bMSGS.ERR** gets set. The Transceiver power-on ready is not masked by this bit.	0
1	IMTX	Message transmitted interrupt enable. When set, allows interrupts when **bMSGS.MTX** gets set.	0
0	IMRX	Message received interrupt enable. When set, allows interrupts when **bMSGS.MRX** gets set.	0

Table 3-10: bIE Register

89h bGA *Group Address* MOST

Bit	Label	Description	Default
7..0	GA[7:0]	Group Address for Control messages sent out as Group-cast.	*

* At power-up the default is indeterminate. Once written, the value is retained even if the chip is reset.

Table 3-11: bGA Register

GA[7:0] Group Address. Sets the nodes group address (group-cast) to 0x300 + **GA[7:0]**. A Control message sent to the group address is received by all nodes that have their bGA register set to the same value. C8h is not allowed in bGA as this is the Broadcast address (3C8h) for all nodes.

8Ah bNAH *Node Address High* MOST

Bit	Label	Description	Default
7..0	NA[15:8]	Node Address High.	0Fh

Table 3-12: bNAH Register

NA[15:8] Node Address High. The high portion of the Logical address, bits 11 through 8. Together with bNAL, it forms the 12-bit Logical address used for Control messages. The default address after reset is 0FFFh. Valid addresses are 0001h - 02FFh.

8Bh bNAL Node Address Low MOST

Bit	Label	Description	Default
7..0	NA[7:0]	Node Address Low.	FFh

Table 3-13: bNAL Register

NA[7:0] Node Address low. The lower byte of the Logical address, bits 7 through 0. Together with bNAH, it forms the 12-bit Logical address used for Control messages. The default address after reset is 0FFFh. Valid addresses are 0001h - 02FFh.

8Eh bCM2 Clock Manager 2 (read only) MOST

Bit	Label	Description	Default
7	$\overline{\text{LOK}}$	PLL Lock status. When set, indicates that the PLL is NOT locked. When clear, the PLL is locked.	1
6	NNAC	No Network Activity status. When set, activity is NOT detected on the MOST Network. When clear, activity is detected on the Network.	1
5..0	rsvd	Reserved. Write to 0.	00010

Table 3-14: bCM2 Register

8FH bNDR Node Delay Register MOST

Bit	Label	Description	Default
7..0	D[7:0]	Network Node delay.	00h

Table 3-15: bNDR Register

D[7:0] Node Delays. Indicates the number of Synchronous Source data node delays between this node and the timing-master node. If this node is not in Source data bypass mode (**bXCR.SBY** clear), it will increment **bNDR** and pass the value to the next node. The time delay per node is two frames. Only valid after lock (**bCM2.$\overline{\text{LOK}}$**) is established.

90h bMPR Maximum Position Register MOST

Bit	Label	Description	Default
7..0	D[7:0]	Total number of nodes in the Network.	00h

Table 3-16: bMPR Register

D[7:0] Maximum Position. Indicates the total number of nodes in the network. A change of this register sets **bMSGS.ALC** and generates an interrupt if **bIE.IALC** is set. Nodes in all-bypass mode (**bXCR.$\overline{\text{ABY}}$** clear) are invisible to the Network and not counted. Only valid after lock (**bCM2.$\overline{\text{LOK}}$**) is established and Network initialization is complete. Valid bMPR values are 0-63, where 0 indicates 64 nodes in the ring.

91h bMDR Maximum Delay Register MOST

Bit	Label	Description	Default
7..0	D[7:0]	Maximum Delay through the network. .	00h

Table 3-17: bMDR Register

D[7:0] Maximum Delay. Indicates the total number of node delays for synchronous source data within the Network. A change of this register sets **bMSGS.ALC** and generates an interrupt if **bIE.IALC** is set. Nodes in Source-Data bypass mode (**bXCR.SBY** set) are not counted as they add no source delay. Only valid after lock (**bCM2.$\overline{\text{LOK}}$**) is established and Network initialization is complete.

96h bSBC Synchronous Bandwidth Control MOST

Bit	Label	Description	Default
7..4	rsvd	Reserved. Write to 0.	0000
3..0	SAC[3:0]	Number of synchronous quadlets. Valid range is 06h through 0Fh.	0000

Table 3-18: bSBC Register

SAC[3:0] Synchronous Area Count – in quadlets. Can only be modified in the timing-master device (**bXCR.MTR** set), and only valid after lock (**bCM2.$\overline{\text{LOK}}$**) is established. Specifies the number of quadlets reserved for synchronous data. The rest of the 15 quadlets may be used for asynchronous data. Therefore, this value specifies the boundary between synchronous and asynchronous data. In a timing-slave device, this register is read-only. Valid values are from 06h to 0Fh inclusive, where:

06h – 6 quadlets for synchronous data and 9 for asynchronous data (minimum value).

0Fh – All 15 quadlets for synchronous data and no asynchronous data. Although the chip supports 0Fh (no asynchronous data), the MOST Specification requires at least one quadlet be reserved for asynchronous data, making the maximum 0Eh.

The default value is NOT valid. This register MUST be programmed to a valid value for proper Network operation.

Whenever bSBC is changed, the timing-master node ***must*** send the *De-allocate All* system Control message to initialize the mCRA properly.

97h bXSR2 Network Transceiver Status Register 2 MOST

Bit	Label	Description	Default
7..2	rsvd	Reserved. Write to 0.	000000
1	INV	Bi-phase Inversion control. When set, inverts the bi-phase data sent to the PLL and XCR, which affects the polarity of pulse-width distortion tolerated by the device. This value should be optimized for the respective FOR device used.	0
0	rsvd	Reserved. Write to 0.	0

Table 3-19: bXSR2 Register

380h mCRA Channel Resource Allocation table MOST

Byte	Label	Description	Default
00h	bCRA0	Channel Allocation Status for MOST Synchronous data byte 0	70h
01h	bCRA1	Channel Allocation Status for MOST Synchronous data byte 1	70h
...	...	to	...
3Ah	bCRA58	Channel Allocation Status for MOST Synchronous data byte 58	70h
3Bh	bCRA59	Channel Allocation Status for MOST Synchronous data byte 59	70h

Table 3-20: mCRA Table

The mCRA contains the current allocation status for each MOST Network synchronous byte, or channel. The valid table size is based on the bSBC value (how many quadlets are reserved for synchronous data), where the last valid byte is calculated as follows:

last mCRA byte = 380h + (bSBC $\times$ 4) - 1

In each byte, the MSB (most significant bit) is interpreted separately from the lower seven bits. Timing-slave nodes (**bXCR.MTR** clear) must ignore the MSB, which can only be interpreted by the timing-master node. In the timing-master, the MSB set indicates a channel is in use, and the MSB clear indicates a channel is not in use. The lower seven bits contain the Connection Label, which is loaded into each byte (physical synchronous channel) associated with the logical channel. The value of 70h indicates that a channel is not allocated. The Connection Label is the physical channel address of the first byte in the logical grouping. Therefore, if the four physical channels allocated to a request are 00, 01, 02, and 03; then the Connection Label for the whole group is 00. At reset, all values in the mCRA are set to 70h (bytes are not allocated and are not in use).

Once lock is achieved and bSBC is set properly, the mCRA must be initialized by the timing-master node by sending a *De-allocate All* system Control message to itself. The *De-allocate All* message is derived from the *Resource De-allocate* Control system message, with the Connection Label set to 7Fh.

3.2.4 MOST Control Message Registers

In the *MOST Specification*, the Control messages have the appearance of:

```
SrcAdr -> TrgAdr: FBlockID.InstID.FktID.OpType.Length(Parameter)
```

The `SrcAdr` is the Source Address (the Node Address of the device sending the message), the `TrgAdr` is the Target Address. The MOST Control message `Length` parameter supports lengths of up to 4095 bytes, which is greater than the eleven bytes of the MOST Spec. Control message. The Length parameter is indirectly coded in the TelID and TelLen fields.

In addition, the Length field is generally hidden when CF messages are described. Since only one of a particular FBlock exists in a typical system, the *Instance ID* can also be omitted, and is coded as 0. This would make the MOST Spec Control message appear as follows:

```
SrcAdr -> TrgAdr: FBlockID.FktID.OpType(Parameter)
```

A variant listed in the *MOST Specification* lists functions which are in a device. The actual device is termed `DeviceID` and would be the target address (**`TrgAdr`**) when sending a command to that function in that particular device. The format follows:

```
DeviceID.FBlockID.InstID.FktID.OpType.Length(Parameter)
```

A Control message is transmitted through the mXCMB buffer and received through the mRCMB buffer. The relationship between the normal Control message above, and the buffers are illustrated in Figure 3-11. This Figure illustrates the alignment of the Control message to the buffers, but does not indicate all the information required to send an actual message.

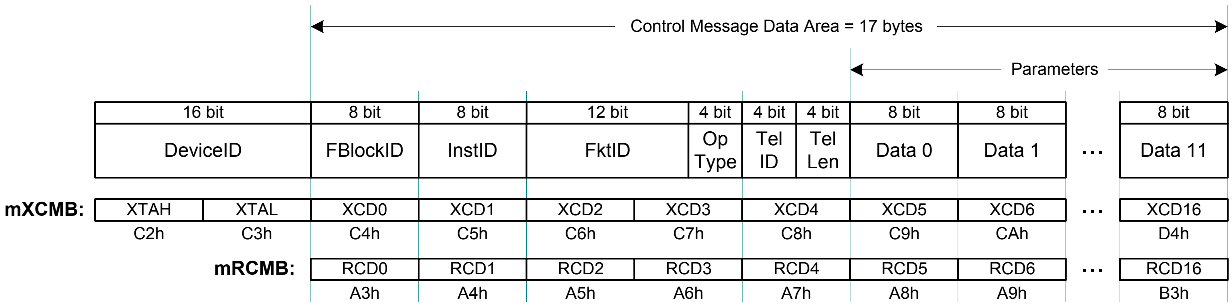


Figure 3-11: Control Message Buffers

The following registers are used to send and receive Control messages. Control messages send are divided into two groups: normal - where the transmit message type is 00h, and system - where the transmit message type is anything other than 00h.

85h bMSGC Message Control MOST

Bit	Label	Description	Default
7	STX	Start Transmission	0
6	RBE	Receiver buffer enable	0
5	RES	MOST Processor reset	0
4	SAI	Start address initialization	0
3	RALC	Reset Allocation change interrupt	0
2	RERR	Reset Error and Power-on after start-up interrupt	0
1	RMTX	Reset Message transmitted interrupt	0
0	RMRX	Reset Message received interrupt	0

Table 3-21: bMSGC Register

STX Start Transmission. When set, starts transmission of a control message previously written into the Transmit Control Message Buffer, mXCMB. When the transmission starts, **STX** is cleared automatically. The result is indicated by the **bMSGS.TXR** bit.

RBE Receive Buffer Enable. When set, the Receive Control Message Buffer, mRCMB, is free to receive a new message. **RBE** is automatically cleared when the MOST Processor finishes the releasing procedure. **RBE** should only be set after the current message has been read or if the received message is to be ignored.

RES MOST Processor reset. When set, the MOST Processor resets all of the MOST Processor, including bXCR, the COM ports, and the MOST Routing Table (MRT).

SAI Start Address Initialization. When set, starts address verification using the address previously written into the mXCMB as target address. After the address is verified as unique in the network, it is stored in the Node Address register bNAH, bNAL. The result of address initialization is indicated by the **bMSGS.TXR** bit. SAI is automatically cleared.

RALC Reset Allocation change interrupt. When set, the Allocation Change Interrupt bit, **bMSGS.ALC** is cleared. **RALC** is automatically cleared.

RERR Reset Error and Power-on after start-up interrupt. When set, the **bMSGS.ERR** bit is cleared. **RERR** is automatically cleared.

RMTX Reset Message Transmit interrupt. When set, the **bMSGS.MTX** bit is cleared. **RMTX** is automatically cleared.

RMRX Reset Message Receive interrupt. When set, the **bMSGS.MRX** bit is cleared. **RMRX** is automatically cleared.

86h bMSGS Message Status MOST

Bit	Label	Description	Default
7	RBS	Receive buffer status	0
6	TXR	Transmit message status	0
5, 4	rsvd	Reserved, Write to 0	00
3	ALC	Allocation change	0
2	ERR	Error or Power-on event	0
1	MTX	Message transmitted	0
0	MRX	Message received	0

Table 3-22: bMSGS Register

RBS Receive buffer status.
0 – mRCMB buffer is available for the next received message.
1 – mRCMB buffer is locked with a received message.

TXR Transmission Result.
0 – Last transmission failed. An error code is in bXTS.
1 – Last transmission was successful.

ALC Allocation Change. When set, indicates that one or more of the allocation registers bMPR or bMDR has changed. **ALC** is cleared by setting **bMSGC.RALC**. **ALC** can generate a COM Port interrupt to the Host Controller if **bIE.IALC** is set.

ERR Error. When set, indicates a power-on ready event or an error event that set **bXSR.ERR**. **bMSGS.ERR** is cleared by setting **bMSGC.RERR**. **ERR** can generate a COM Port interrupt to the Host Controller if **bIE.IERR** is set. See Figure 3-10.

MTX Message Transmitted. When set, message transmission is complete with TXR indicating the status. **MTX** is cleared by setting **bMSGC.RMTX**. **MTX** can generate a COM Port interrupt to the Host Controller if **bIE.IMTX** is set.

MRX Message Received. When set, a message is in the receive buffer mRCMB. **MRX** is cleared by setting **bMSGC.RMRX**. **MRX** can generate a COM Port interrupt to the Host Controller if **bIE.IMRX** is set.

A0h mRCMB Receive Control Message Buffer MOST

Byte	Label	Description	Default
00h	bRTYP	Received Message Type	00h
01h	bRSAH	Source Address High	00h
02h	bRSAL	Source Address Low	00h
03h 04h ... 12h 13h	RCD0 RCD1 ... RCD15 RCD16	Received control byte 00h Received control byte 01h to Received control byte 0Fh Received control byte 10h	00h 00h ... 00h 00h

Table 3-23: mRCMB Buffer

bRTYP Received Control message type. bRTYP indicates the type of address used to send the message.
00h – Logical addressing (address in bNAH/bNAL).
01h – Physical addressing (0x400 + bNPR).
02h – Broadcast addressing (0x3C8).
03h – Groupcast addressing (0x300 + bGA).

bRSAH Source Address high. High-nibble of the logical address of the device which sent the message.

bRSAL Source Address low. Low-byte of the logical address of the device which sent the message.

RCD[16:0] Received Control message bytes. When a message is received, **bMSGS.RBS** is set which locks this buffer from further writing and indicates that a message was received. To unlock this buffer for the next message, **bMSGC.RBE** must be set.

BEh bXTIM Transmit Retry Time Register MOST

Bit	Label	Description	Default
7..0	D[7:0]	Time between transmission retries.	0Bh

Table 3-24: bXTIM Register

D[7:0] Time between transmit Control message retries. Specifies the number of Control message periods to wait before attempting a retry. The minimum value is 03h. The time for one control message is (approximately):
363 µs at Fs = 44.1 kHz
333 µs at Fs = 48 kHz.

BFh bXRTY Transmit Retry Register MOST

Bit	Label	Description	Default
7..0	D[7:0]	Total transmission attempts. Specifies the number of transmission attempts/retries. The minimum value is 01h, the maximum value FFh.	06h

Table 3-25: bXRTY Register

D[7:0] Transmission Retry. Total number of attempts at transmitting a Control message, if the transmission fails. The minimum value is 01h. The default value of 06h indicates 5 retries (plus the initial attempt).

C0h mXCMB Transmit Control Message Buffer MOST

Byte	Label	Description	Default
00h	bXPRI	Transmit Priority	00h
01h	bXTYP	Transmit Control message type	00h
02h	bXSAH	Target Address High	00h
03h	bXSAL	Target Address Low	00h
04h 05h ... 13h 14h	XCD0 XCD1 ... XCD15 XCD16	Transmit control byte 00h Transmit control byte 01h to Transmit control byte 0Fh Transmit control byte 10h	00h 00h ... 00h 00h

Table 3-26: mXCMB Buffer

bXPRI Transmit Priority. Specifies the priority, used for Network arbitration, for the transmit message.
00h – lowest priority
0Fh – highest priority

bXTYP Transmit message type.
00h – Normal message, using any of the addressing options (logical, group, etc.).
01h – *Remote Read* system message.
02h – *Remote Write* system message. Only valid for receiving nodes with **bNC.RWD** bit clear.
03h – *Resource Allocate* system message.
04h – *Resource De-allocate* system message.
05h – *Remote GetSource* system message.

bXSAH Target Address high. High portion of the destination device address.

bXSAL Target Address low. Low byte of the destination device address. Through bXSAH/bXSAL, different addressing modes are supported (0000h is not valid):
0001h - 02FFh: Logical addressing. The receiving node's logical address, bNAH/bNAL.
03C8h: Broadcast addressing to all nodes. Use of Broadcast addressing should be minimized since all nodes stop transmission attempts until all nodes have received the broadcast message, thereby lowering the overall Network Control message bandwidth.
0300h-03FhF: Groupcast addressing. The receiving node's bGA register.
0400h-043Fh: Physical addressing. The receiving node's position register, bNPR.
0440h-0FFFh: reserved.

XCD[16:0] Transmit Control message. When this transmit buffer is loaded, then **bMSGC.STX** should be set, causing the message to be transmitted. The transmission result is indicated by **bMSGS.TXR** with the status in bXTS.

D5h bXTS Transmit Status Register MOST

Bit	Label	Description	Default
7..0	XTS[7:0]	Transmission Results.	00h

Table 3-27: bXTS Register

XTS[7:0] Transmission results. Reflects the results of the last control message transmission.
00h – Transmission failed. Target device did not respond.
10h – Transmission successful.
11h – Transmission successful, but receiving device doesn't support the message type used. Nodes with **bNC.RWD** set will return this response to a *Remote Write* system message.
20h – Transmission failed. Wrong CRC.
21h – Transmission failed. Target device receive buffer full.

3.2.4.1 System Control Messages

A non-zero transmit message type (bXTYP in mXCMB) is defined as a system Control message. Normal Control messages (bXTYP = 0) are sent to nodes which receive the message and store the message in its mRCMB receive message buffer. When a system message is sent, the target node responds in the same transmitted message with its receive buffer remaining untouched. The responses to the system message are stored in the ***sender's*** transmit buffer, mXCMB. Using this method, system messages are handled in the background and have no effect on normal Control message flow.

System messages are handled by on-chip hardware and do not require extra software in the Host Controller. In contrast, normal Control messages are sent to a target node's receive buffer, where the message is stored until Host Controller software reads the message and reacts. Table 3-28 illustrates the values transmitted (Tran) and the responses (Rec) for each system message. The '=' sign indicates that the received data is the same as the transmitted data, and blank cells are reserved. The transmit priority, bXPRI is generally left at the default value of 01h.

mXCMB	*Remote Read*		*Remote Write*		*Resource Allocate*		*Resource De-allocate*		*Remote GetSource*	
Addr.	**Tran**	**Rec**	**Tran**	**Rec**	**Tran**	**Rec**	**Tran**	**Rec**	**Tran**	**Rec**
C0h	bXPRI	=	bXPRI	=	bXPRI	=	bXPRI	=	bXPRI	=
C1h	**01h**	=	**02h**	=	**03h**	=	**04h**	=	**05h**	=
C2h	bXSAH	=	bXSAH	=	04h	=	04h	=	03h	=
C3h	bXSAL	=	bXSAL	=	00h	=	00h	=	C8h	=
C4h										
C5h	MAP	=	MAP	=	RQST	=	CL	=	†CL	=
C6h			LEN							
C7h		D0	D0			ANS1		ANS1		
C8h		D1	D1			ANS2				
C9h		D2	D2			CL (MRT0)				
CAh		D3	D3			MRT1				NP
CBh		D4	D4			MRT2				
CCh		D5	D5			MRT3				GA
CDh		D6	D6			MRT4				NAH
CEh		D7	D7			MRT5				NAL
CFh						MRT6				
D0h						MRT7				
D1h-D4h										

† For the *Remote GetSource* message, this can be the *Connection Label* or a particular physical channel.

Table 3-28: System Control Messages - mXCMB

When the mXCMB buffer is filled, the message is sent by setting the **bMSGC.STX** bit (**bMSGC.RMTX** should also be set to clear the previous transmit message status). When the **bMSGS.MTX** gets set, the message has been transmitted. If **bMSGS.TXR** is set, the results are in the mXCMB transmit buffer, as listed in Table 3-28, column *Rec*. If **bMSGS.TXR** is clear, the message was not sent successfully, and the error status is in bXTS.

3.2.4.1.1 Remote Read and Write System Messages

The *Remote Read* system message (bXTYP = 01h) always reads eight bytes on memory page 0 of the target node. The memory location/address to read data from is sent in MAP. Upon successful transmission (**bMSGS.MTX** = **bMSGS.TXR** = 1), the eight bytes read from the target node, starting at addressed MAP, are located in mXCMB, labels D0-D7.

The *Remote Write* system message (bXTYP = 02h) can write up to eight bytes on memory page 0 of the target node - only if the target node allows *Remote Write* commands (**bNC.WRD** clear). If **bNC.WRD** is set, no remote writes are possible. The memory location/address to write data to is sent in MAP, the number of bytes to write is in LEN, and the actual data is in D0-D7. Upon successful transmission (**bMSGS.MTX** = **bMSGS.TXR** = 1), LEN bytes are written to the target node, starting at addressed MAP.

3.2.4.1.2 Resource Allocate System Message

The *Resource Allocate* system message (bXTYP = 03h) is always sent to the timing-master node. Therefore, the simplest target address to use is the physical address, which is always 0x400 for the timing-master.

Before data is placed on the MOST Network (filling the lower half of the MRT), the physical synchronous channels used should be allocated by the timing-master node. Therefore, to place stereo audio data from the ADCs on the MOST Network, the four physical synchronous channels needed (two for left audio data and two for right audio data) are requested from the timing-master using the *Resource Allocate* system message. The number of channels needed (up to eight) are sent in the mXCMB RQST byte. Upon successful transmission (**bMSGS.MTX** = **bMSGS.TXR** = 1), ANS1 and ANS2 bytes indicate the response from the timing-master node. The ANS1/2 responses are listed below.

ANS1	ANS1 Description	ANS2	ANS2 Description
01	Allocation Granted. The MRT locations are in MRT0-MRT7, with MRT0 being the Connection Label (CL). CL is used to deallocate the group of channels when finished.	Free Chan.	Number of free channels - including current request
02	Timing-master is busy. Resend the message later.	Free Chan.	Total free channels available
03	Request denied. Not enough free channels available. ANS2 indicates how many free channels exist.	Free Chan.	Total free channels available
04	The RQST value is not valid (0 or greater than 8).	Free Chan.	Total free channels available
05	Message sent to the wrong node (a timing-slave node) - not sent to the timing-master.	00	

Table 3-29: Resource Allocate Responses

The physical synchronous channels allocated are considered a *logical channel*, with the logical channel identifier, or Connection Label, CL. CL is the MRT location of the first allocated channel, and used to deallocate, or give up the channels, when the logical channel is no longer needed. The CL is also stored in the mCRA in the physical channel positions for each physical channel that is part of the logical channel. Therefore, if the request for four physical channels was granted, and the returned channels were 04h, 05h, 06h, and 07h (MRT0-MRT3); then 04h would be the Connection Label CL, and 04h would be stored in the mCRA in the four physical channel positions. After masking off the MSB, 0x384 = 0x385 = 0x386 = 0x387 = 04h.

3.2.4.1.3 Resource De-allocate and De-allocate All System Messages

The *Resource De-allocate* system message (bXTYP = 04h) is always sent to the timing-master node. Therefore, the simplest target address to use is the physical address, which is always 0x400 for the timing-master.

When the node is finished using the logical synchronous channel, the MRT locations are freed-up (by placing the Network receive MRA address in the MRT location), and the channels should be de-allocated so other nodes can use them. To free up the channels, the *Resource De-allocate* system message is sent with the mXCMB CL byte contain the Connection Label received when the physical channels were allocated. Upon successful transmission (**bMSGS.MTX** = **bMSGS.TXR** = 1), ANS1 indicates the response from the timing-master node.

ANS1	ANS1 Description
01	De-allocation successful
02	Timing-master is busy. Resend the message later.
04	The CL value is not valid (greater than 7Fh).
05	Message sent to the wrong node (a timing-slave node) - not sent to the timing-master.

Table 3-30: Resource De-allocate Responses

The special Connection Label of 0x7F is defined as the *De-allocate All* system message, and must be sent by the timing-master node after any change to the bSBC register. This command properly initializes the mCRA table.

3.2.4.1.4 Remote GetSource System Message

The *Remote GetSource* system message (bXTYP = 05h) is always sent as a broadcast message (address 0x3C8). This system Control message locates the node that is transmitting data onto the MOST Network using the physical channel in the sent message. Assuming that nodes use all physical channels within a logical channel as a group, the Connection Label (CL) can be used to find the whole group. Upon successful transmission (**bMSGS.MTX** = **bMSGS.TXR** = 1), the mXCMB buffer contains the node's addresses (physical, group, and logical) that is transmitting data using the physical channel. If **bMSGS.MTX** is set, but **bMSGS.TXR** is clear, no node in the current network is using the physical channel.

mXCMB	Successful Response Description (bMSGS.MTX = bMSGS.TXR = 1)
NP	Node Position (part of physical address) of node using the physical channel
GA	Group Address (part of group-cast address) of node using the physical channel
NAH	High byte of Node Address (high byte of logical address) of node using the physical channel
NAL	Low byte of Node Address (low byte of logical address) of node using the physical channel

Table 3-31: Remote GetSource Response

3.2.5 Packet Data Transfer

The *Packet Data Transfer* service uses the portion of the MOST Network reserved for the asynchronous channel, and is useful for applications that can transfer data in bursts (e.g,. Internet data, GPS map data, email), instead of in a continuous data stream (e.g., video or audio content). The maximum packet size supported is 48 data bytes and is protected by a trailing CRC (Cyclic Redundancy Check), which is automatically generated/checked by the OS8805. Support for *Packet Data Transfer* was added in revision G silicon, and is not supported on previous revisions.

To enable Async. Packet reception on the OS8805, the **bNC.APREN** and the **MMPC.RFS1** bits must both be set.

When sending a packet, the target address can either be the target node's logical address (specified in the bNAH/bNAL registers), or it can be the address stored in the target node's alternate packet address registers. Sending packets to the logical address uses the same addressing as used for the Control messages. If separate addresses for Control and Packet messages are required, the alternate packet address registers (bAPAH and bAPAL) can be used.

The Host Controller's asynchronous/packet flag **RCS.AINT** indicates either the reception or transmission of packet data. For transmitted packets, **RCS.AINT** is set when the packet is completely transferred, the transmit status is available, and the Packet Transmit Buffer (mAXP) is available. For received packets, **RCS.AINT** is set when a valid packet is received, with the entire packet available in the Packet receive buffer mARP. A valid received packet is one for which the logical address (bNAH/bNAL) or the alternate packet address (bAPAH/bAPAL) is correct, and the message has a valid CRC.

0xE8 bAPAH Alternate Packet Address High MOST

Bit	Name	Description	Default
7..0	APA[15:8]	Alternate Packet Address High. This value cannot be the same as bNAH.	0x0F

Table 3-32: bAPAH Register

The bAPAH register keeps the higher address part (bits 15 through 8) of the alternate address for packets received. The default alternate packet address after reset is 0x0F0F. The bAPAH register cannot be the same value as bNAH. When a node receives a packet, it checks the target address high byte against its bNAH value. If they do not match, the alternate packet address registers are checked. bAPAH and bAPAL can be used as an asynchronous packet groupcast address to allow multiple nodes to receive the same message. However, using this register for groupcast addressing does not have the same protection and acknowledges that are associated with the Control message groupcast addressing.

0xE9 bAPAL Alternate Packet Address Low MOST

Bit	Name	Description	Default
7..0	APA[7:0]	Alternate Packet Address Low	0x0F

Table 3-33: bAPAL Register

0xEC bPLDT Transmit Packet Length MOST

Bit	Name	Description	Default
7..0	D[7:0]	Packet length for data transfer in quadlets	0x00

Table 3-34: bPLDT Register

The value in this register specifies the number of data bytes (in quadlets) sent when transmitting a packet, and includes the data bytes along with the two source address bytes. Therefore, the length value is calculated as follows:

$$\mathrm{bPDLT} = \mathrm{roundup}\left(\frac{\text{number of data bytes} + 2}{4}\right)$$

Valid values for bPLDT are 0x01 to 0x0D. If the length value is rounded up (fractional), then the extra filler bytes at the end of the packet should be set to 0x00. For example, sending three data bytes takes two quadlets, because two source address bytes are used. Three filler bytes exist at the end of the packet, which should be filled with zeros. bPLDT does not include the CRC bytes, which are automatically generated by the part.

0xF2 bPPI Transmit Packet Priority MOST

Bit	Name	Description	Default
7..0	D[7:0]	Packet Priority. Valid values are 0x01 (highest) to 0x07 (lowest)	0x01

Table 3-35: bPPI Register

The value in bPPI determines the priority of a transmitted packet, used in arbitration of the asynchronous portion of the MOST Network source data. Valid values are 0x01 to 0x07, where 0x01 stands for the highest priority level.

0xE2 bPCTC Packet Control MOST

Bit	Name	Description	Default
7..5	rsvd	Reserved; Write as 0	0
4	RAF	Reset *Packet rejected status (mARP full)* bit **bPCTS.AF**	0
3	RAC	Reset *Packet rejected status (CRC failed)* bit **bPCTS.AC**	0
2	rsvd	Reserved. Write as 0	0
1	RATX	Clear *Packet Transmitted* interrupt	0
0	RARX	Unlock the Asynchronous Receive Packet Buffer mARP	0

Table 3-36: bPCTC Register

RAF Reset *Packet Rejected status (mARP full)* bit **bPCTS.AF**. When the **RAF** bit is set, the **bPCTS.AF** bit is cleared. **RAF** must not be set unless **bPCTS.AF** is set. The **RAF** bit is cleared automatically.

RAC Reset *Packet Rejected status (CRC failed)* bit **bPCTS.AC**. When the **RAC** bit is set, the **bPCTS.AC** bit is cleared. **RAC** must not be set unless **bPCTS.AC** is set. The **RAC** bit is cleared automatically.

RATX Clear *Packet Transmitted* bit. When the **RATX** bit is set, the **bPCTS.ATX** bit is cleared. If both **bPCTS.ATX** and **bPCTS.ARX** are clear, the Host Controller's **RCS.AINT** bit is cleared. **RATX** must not be set unless **bPCTS.ATX** is set. The **RATX** bit is cleared automatically.

RARX Unlock the Asynchronous Receive Packet Buffer mARP, and clear packet received interrupt. Setting the **RARX** bit clears the **bPCTS.ARX** bit and unlocks the Asynchronous Receive Packet Buffer mARP. If both **bPCTS.ATX** and **bPCTS.ARX** are clear, the Host Controller's **RCS.AINT** bit is cleared. When the **bPCTS.ARX** bit is set, mARP contains a packet and is locked until the **RARX** bit unlocks it. While locked, mARP will not receive any other packets. **RARX** must not be set unless **bPCTS.ARX** is set. The **RARX** bit is cleared automatically.

0xEA bPSTX Transmit Packet Start MOST

Bit	Name	Description	Default
7	ASTX	Start packet transmission	0
6..0	rsvd	Reserved. Write as 0	0

Table 3-37: bPSTX Register

ASTX Start packet transmission. When the **ASTX** bit is set, the part starts arbitrating for the asynchronous channel to transmit the current packet in mARP. The **ASTX** bit is cleared automatically when the transmission is finished.

0xE3 bPCTS Packet Status MOST

Bit	Name	Description	Default
7..5	rsvd	Reserved	0
4	AF	Packet rejected — mARP full	0
3	AC	Packet rejected — CRC failed	0
2	rsvd	Reserved	0
1	ATX	Packet transmitted	0
0	ARX	Packet received	0

Table 3-38: bPCTS Register

AF Packet rejected, mARP full. When the **AF** bit is set, it indicates that the last reception failed because the packet receive buffer mARP is full (locked). mARP is unlocked by setting the **RARX** bit to 1 after having read its contents. The **AF** bit is cleared by setting the **bPCTC.RAF** bit.

AC Packet rejected, CRC failed. When the **AC** bit is set, it indicates that the last reception failed due to a bad CRC value. The **AC** bit is cleared by setting the **bPCTC.RAC** bit.

ATX Packet transmitted event. When the **ATX** bit is set, it indicates that a packet transmission is completed. **ATX** will be set after the last byte of a packet is processed. The Host Controller's **RCS.AINT** bit will be set shortly after the **ATX** bit is set. The **ATX** bit is cleared by setting the **bPCTC.RATX** bit.

ARX Packet received event. When the **ARX** bit is set, it indicates that a packet has been received and that mARP is locked. Shortly after the **ARX** bit is set, the Host Controller's **RCS.AINT** bit will be set. The **ARX** bit is cleared by setting the **bPCTC.RARX** bit.

As long as the **bPCTS.ARX** bit is set, no reception of further packets is possible. Clearing the **bPCTS.ARX** bit unlocks mARP. The contents of mARP can then be overwritten by the next packet.

The Asynchronous Receive Packet buffer contains the last packet which was successfully received.

0x180 mARP Asynchronous Receive Packet Buffer MOST

Byte	Name	Description	Default
0x00	bARTH	Received Target address high	0x00
0x01	bARTL	Received Target address low	0x00
0x02	bASAH	Source address high	0x00
0x03	bASAL	Source address low	0x00
0x04	bARD0	Asynchronous receive data byte 0	0x00
0x05 ... 0x32	bARD1 ... bARD46	Asynchronous receive data byte 1 to 46	0x00
0x33	bARD47	Asynchronous receive data byte 47	0x00

Table 3-39: mARP Buffer

bARTH/bARTL Received Target Address High/Received Target Address Low. The target address to which the received packet was sent. Either the logical node address (bNAH/bNAL) or the alternate packet address (bAPAH/bAPAL).

bASAH/bASAL Source Address High/Source Address Low. The logical address of the sending node. The origin of the packet data in the MOST Network.

bARD[0:47] Asynchronous Receive Data Bytes 0 to 47. These bytes contain the actual packet data that was received.

0x1C0 mAXP Asynchronous Transmit Packet Buffer MOST

Byte	Name	Description	Default
0x00	bATAH	Target address high	0x0F
0x01	bATAL	Target address low	0xFF
0x02	bAXD0	Asynchronous Transmit data byte 0	0x00
0x03 ... 0x30	bAXD1 ... bAXD46	Asynchronous Transmit data byte 1 to 46	0x00
0x31	bAXD47	Asynchronous Transmit data byte 47	0x00

Table 3-40: mAXP Buffer

bATAH/bATAL

Target Address High/Target Address Low. These bytes contain the target address of the node to send the packet to. This should be the logical node address or the alternate packet address of the receiving/target node.

bAXD[0:47] Asynchronous Transmit Data Bytes 0 to 47. These bytes contain the packet data to be sent. If the number of bytes to be sent is not divisible by 4, it is recommended to add filler bytes (generally 0x00).

3.2.5.1 Packet Data Handling

The chip transmits packet data using the portion of the MOST frame reserved for asynchronous packet data transfer. The amount of MOST Network bandwidth reserved for asynchronous packet data is defined in the bSBC register. If there is not sufficient space for sending a packet within a single frame, data is segmented automatically and sent in smaller portions until the transmission is finished.

When sending packet data, a maximum of 48 bytes can be sent per packet,. When receiving a packet, the actual data length in the buffer is not provided. Therefore, the number of user data bytes per packet must be reported from the sending node to the receiving node. This is the task of the controlling application software and should be done before starting transmission. Alternately, the packet length can be included in the first few bytes when using a predefined packet protocol, as used by the MOST High Protocol.

After length is calculated and written to the bPLDT register, user data can be written to the Asynchronous Transmit Packet Buffer mAXP.

If the Host Controller's **RCS.AINT** bit is set, indicating that status in bPCTS has changed, the Packet Control Status register is read to determine if **bPCTS.ATX** and/or **bPCTS.ARX** bits are set. In addition, error information (the **bPCTS.AF** and **bPCTS.AC** bits) can be stored for later use, if needed.

If the **bPCTS.ARX** bit is set, then a packet has been received, and is available in the Asynchronous Receive Packet Buffer mARP. After reading mARP, the buffer must be released (by setting the **bPCTC.RARX** bit) to allow further packet reception.

If the **bPCTC.RAF** and **bPCTC.RAC** error bits have been set, they should also be cleared to prepare them for the next message. This ends the handling of a packet received event.

If the **bPCTS.ATX** bit is set, the packet in mAXP has been transmitted, and a new packet can be loaded into mAXP. To start transmission, the **bPSTX.ASTX** bit must be set. The **bPSTX.ASTX** bit is cleared automatically after the transmit packet is sent.

3.2.6 Configurable Routing Registers

If some of the Source Peripherals are not used, their routing addresses can be used by one of the DSPs to double the routing throughput between the DSP and the Routing bus. By default, each DSP can transfer up to eight words (16 bytes) between the DSP and the Routing bus, each Fs period. On the DSP side, these transfers occur through the DX and DR registers. This transfer rate can be doubled by disabling one

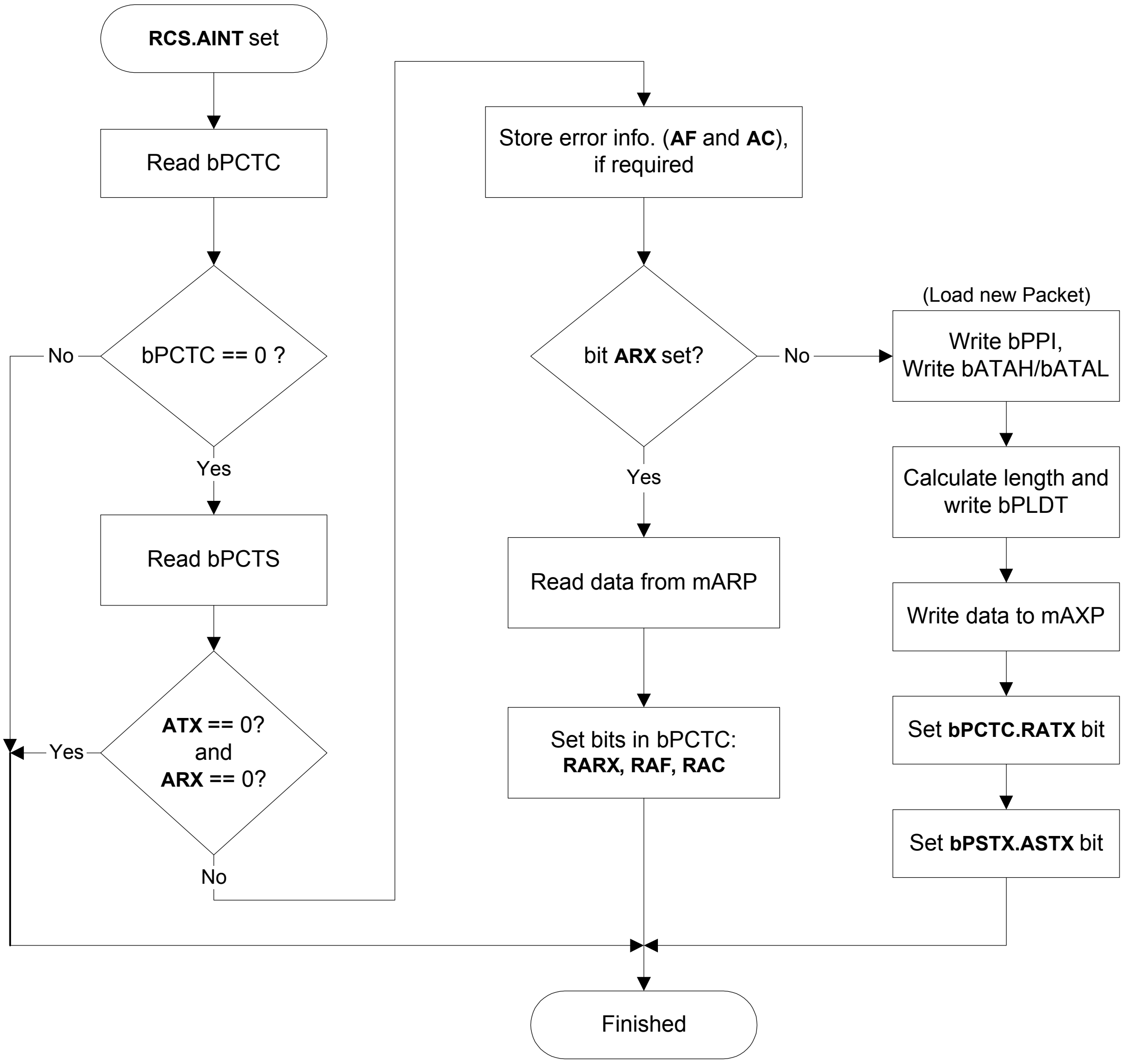


Figure 3-12: Packet Data Transfer Polling

of the other Source Peripheral routing blocks. The available blocks are the Source Ports, the Source Converters, or the other DSP. Therefore, to double the bandwidth through one of the DSPs, one of the other blocks is disabled, and the DSP uses the disabled block's Routing addresses by enabling the DSP's Secondary Routing Port. Then the DSP can use its Secondary Routing Port through DR1 and DX1.

98h bSPR Source Port Routing MOST

Bit	Label	Description	Default
7, 6	rsvd	Reserved. Write to 0	00
5..0	SPR[5:0]	Source Port Routing.	17h

Table 3-41: bSPAR Register

SPR[5:0] Source Port Routing. The only valid values are listed below.

17h - Source Port Routing Enabled.

3Dh - Source Port Routing Disabled. DSP0 or DSP1 port can use the Source Ports Routing locations/addresses to double the bandwidth between the Routing bus and the DSP.

9Dh bFPR Source Converter Routing MOST

Bit	Label	Description	Default
7, 6	rsvd	Reserved. Write to 0	00
5..0	FPR[5:0]	Source Converter Routing (ADCs and DACs).	1Dh

Table 3-42: bFPR Register

FPR[5:0] Source Converter (ADCs and DACs) Routing. The only valid values are listed below.
1Dh - Source Converter Routing Enabled.
3Dh - Source Converter Routing Disabled. DSP0 or DSP1 port can use the Source Converter Routing locations/addresses to double the bandwidth between the Routing bus and the DSP.

99h bD0RP DSP0 Primary MOST Port Routing MOST

Bit	Label	Description	Default
7, 6	rsvd	Reserved. Write to 0	00
5..0	D0RP[5:0]	DSP0 Primary Routing Port Address.	19h

Table 3-43: bD0RP Register

D0RP[5:0] DSP0 Primary MOST Routing Port Address. The only valid values are listed below.
19h - DSP0 primary MOST Routing Port, DR/DX registers, enabled.
3Dh - DSP0 primary MOST Routing Port disabled. DSP1's secondary MOST Routing port, registers DX1/DR1, can use DSP0's Routing locations/addresses to double the bandwidth between the Routing bus and the DSP1.

9Ah bD0RS DSP0 Secondary MOST Port Routing MOST

Bit	Label	Description	Default
7, 6	rsvd	Reserved. Write to 0	00
5..0	D0RS[5:0]	DSP0 Secondary Routing Port Address.	3Dh

Table 3-44: bD0RS Register

D0RS[5:0] DSP0 Secondary MOST Routing Port address. The only valid values are listed below.
3Dh - DSP0 secondary MOST Routing Port disabled. To change to any other value, one of the other routing ports must be disabled first.
17h - DSP0 secondary MOST Routing Port, DR1/DX1 registers, enabled. **SPR[5:0]** must be set to 3Dh (both Source Ports disabled).
1Bh - DSP0 secondary MOST Routing Port, DR1/DX1, enabled. **D1RP[5:0]** must be set to 3Dh (DSP1's primary MOST Routing Port disabled).
1Dh - DSP0 secondary MOST Routing Port, DR1/DX1 registers, enabled. **FPR[5:0]** must be set to 3Dh (all Source Converter, ADCs and DACs, routing disabled).

9Bh bD1RP DSP1 Primary MOST Port Routing MOST

Bit	Label	Description	Default
7, 6	rsvd	Reserved. Write to 0	00
5..0	D1RP[5:0]	DSP1 Primary Routing Port Address.	1Bh

Table 3-45: bD1RP Register

D1RP[5:0] DSP1 Primary MOST Routing Port Address. The only valid values are listed below.

1Bh - DSP1 primary MOST Routing Port, DR/DX registers, enabled.

3Dh - DSP1 primary MOST Routing Port disabled. DSP0's secondary MOST Routing port, registers DX1/DR1, can use DSP1's Routing locations/addresses to double the bandwidth between the Routing bus and the DSP0.

9Ch bD1RS DSP1 Secondary MOST Port Routing MOST

Bit	Label	Description	Default
7, 6	rsvd	Reserved. Write to 0	00
5..0	D1RS[5:0]	DSP1 Secondary Routing Port Address.	3Dh

Table 3-46: bD1RS Register

D1RS[5:0] DSP1 Secondary MOST Routing Port address. The only valid values are listed below.

3Dh - DSP1 secondary MOST Routing Port disabled. To change to any other value, one of the other routing ports must be disabled first.

17h - DSP1 secondary MOST Routing Port, DR1/DX1 registers, enabled. **SPR[5:0]** must be set to 3Dh (both Source Ports disabled).

19h - DSP1 secondary MOST Routing Port, DR1/DX1, enabled. **D0RP[5:0]** must be set to 3Dh (DSP0's primary MOST Routing Port disabled).

1Dh - DSP1 secondary MOST Routing Port, DR1/DX1 registers, enabled. **FPR[5:0]** must be set to 3Dh (all Source Converter, ADCs and DACs, routing disabled).

As an example, to double DSP0's Routing bus bandwidth, one of the following must be disabled:

- DSP1's Primary Routing bus bandwidth (bD1RP = 3Dh),
- the Source Converter's Routing bus bandwidth (bFPR = 3Dh), or
- the Source Port's Routing bus bandwidth (bSPR = 3Dh).

For this example, it is assumed that the Source Converters (ADCs and DACs) are not used; therefore, their Routing bus bandwidth is disabled (bFPR = 3Dh). Then DSP0's Secondary MOST Routing Port is enabled by connecting bD0RS to the Source Converter's Routing addresses (bD0RS = 1Dh). This changes the MOST Routing Registers from those listed in Figure 3-2 to those illustrated in Figure 3-13, where the DAC Routing Table locations are replaced by DSP0's Secondary MOST Routing Port DR1. Now DSP0 has two Routing Ports to transfer data from the Routing bus to DSP0, thereby doubling the Routing bandwidth.

Similarly, the MOST Routing Addresses change from those listed in Figure 3-3 to those listed in Figure 3-14, where the ADC Routing Addresses are replaced by DSP0's Secondary MOST Routing Port DX1. This supports the other direction (DSP0 to Routing bus) for the Secondary Routing port, DX1.

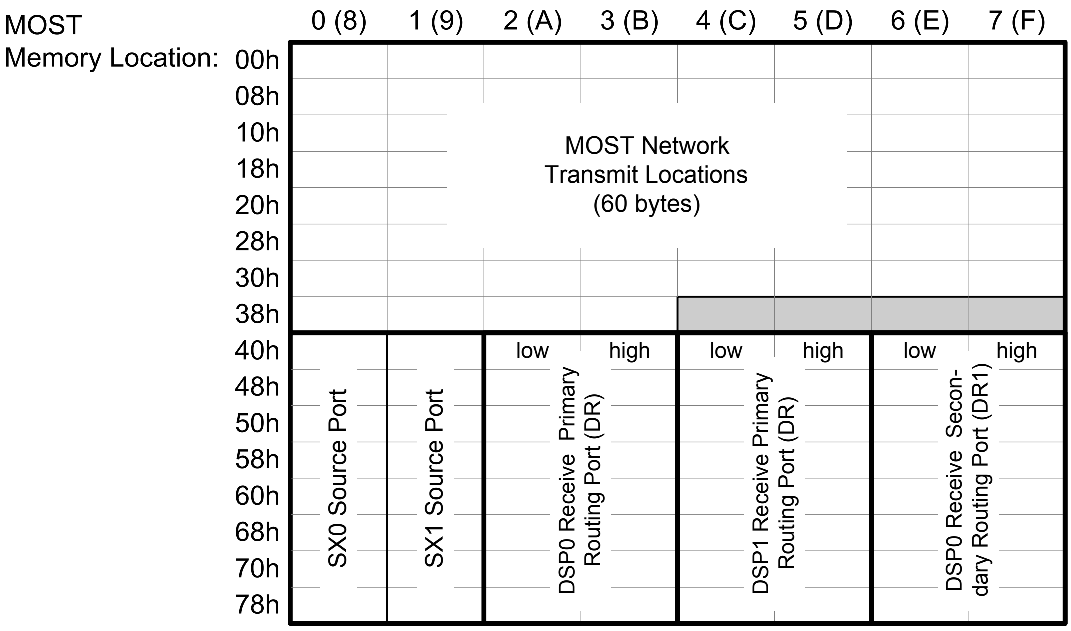


Figure 3-13: ReRoute Example - MRT

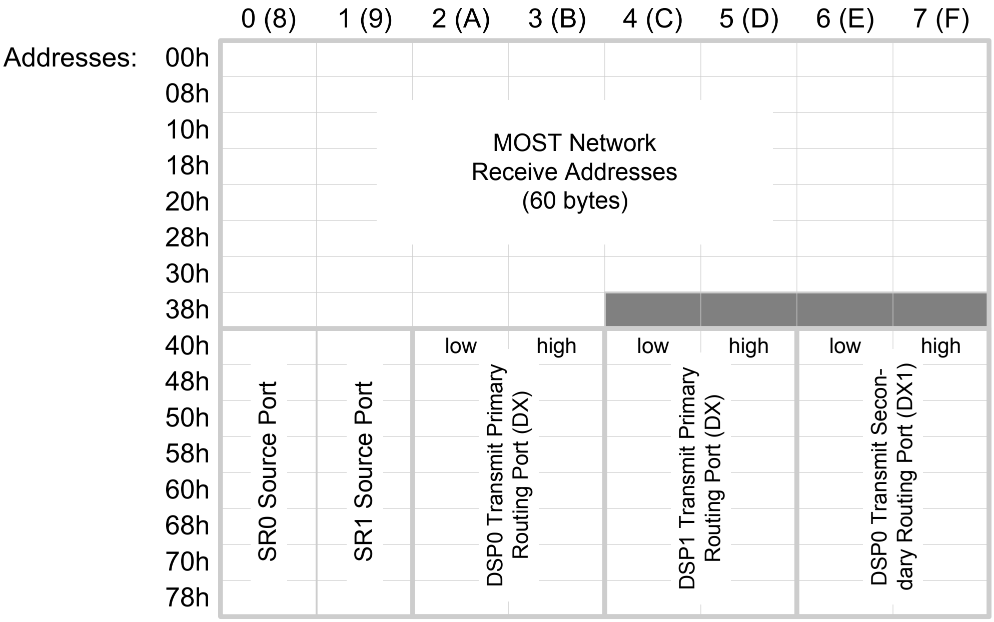


Figure 3-14: ReRoute Example - MRA

3.3 Source Ports

The Source Ports provide external access to the source data (synchronous channel data) in the MOST network. Data can be input through two serial inputs (**SR0** and **SR1**) and output through two serial outputs (**SX0** and **SX1**). The data format on all four pins is the same and they are all clocked by the **SCK** and **FSY** pins. A variety of different data formats can be selected through the Source Port Control register bSDC1.

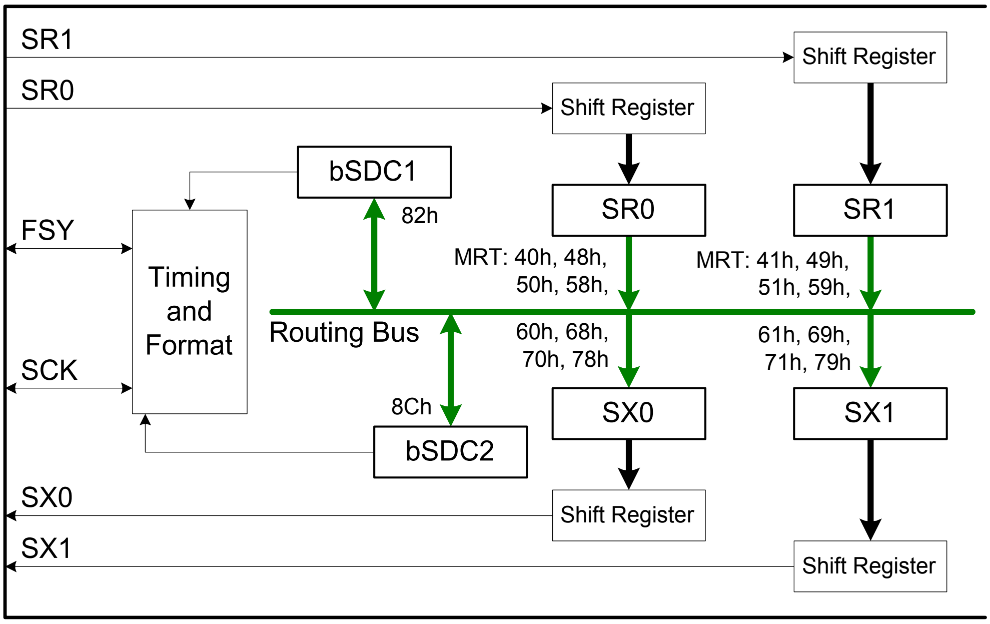


Figure 3-15: MOST Source Ports

The Global Timer GTR and peripheral routing are synchronized to the **SCK** signal. Therefore, **SCK** must be configured properly for GTR and peripheral routing to operate. If the **bSDC1.MOD[1:0]** bits are not set to 11 (Source Port enabled), then **SCK** must be configured as an output, or **SCK** and **FSY** can remain inputs as long as external clocks of the proper frequency are applied.

SR0-SX0 can be configured as an SPDIF transceiver. When configured as the timing-master (**bXCR.MTR** set), the SPDIF data can be clocked in synchronously, or the PLL can recover a clock from it. The **CMCS.MX[1:0]** bits select the source of the PLL and determine the clocking for the entire chip. As a timing-slave node, the PLL recovers a clock from the MOST data; therefore, SPDIF data must be clocked in synchronously.

82h bSDC1 Source Port Control 1 MOST

Bit	Label	Description	Default
7	EDG	Active edge of **SCK**. 0 is data valid at falling edge.	0
6	DEL	Delay from **FSY** edge – one **SCK** period	0
5	POL	Polarity of **FSY**	0
4..3	NBR[1:0]	Number of bits in audio frame	00
2..1	MOD[1:0]	Source Port mode select	00
0	$\overline{\text{MUT}}$	Mute source data outputs (**SX[1:0]**)	0

Table 3-47: bSDC1 Register

EDG Active edge of **SCK**. Indicates the edge where inputs are sampled and output data is valid.
0 – **SCK** falling edge is where inputs are sampled and outputs are valid.
1 – **SCK** rising edge is where inputs are sampled and outputs are valid.

DEL Delay. Data delayed one bit from **FSY** edge.
0 – First bit period after **FSY** edge is MSB.
1 – Second bit period after **FSY** edge is MSB (I^2S format).

POL Polarity of **FSY**.
0 – **FSY** high indicates right sample and low indicates left.
1 – **FSY** high indicates left sample and low indicates right.

NBR[1:0] Number of bits in audio frame. When **SCK** is configured as an output, the **SCK** output is a gated version of the 64-bit period clock, as illustrated in Figure 3-16. When **SCK** is an input, the chip supports either 64 or 32 bits per frame. When the chip is a timing-master and the chip's clock source is the **SCK** input, **SCK** must be a continuous clock for the PLL to lock properly. If the chip is a timing-slave or if **SCK** is not the chip's master clock, then the **SCK** input can be a gated clock.
00 – 64 bit periods per **FSY** frame (input or output)
01 – 48 bit periods per **FSY** frame (output only)
10 – 32 bit periods per **FSY** frame (input or output)
11 – Reserved.

MOD[1:0] Source Port mode select.
00 – **FSY** and **SCK** are inputs.
01 – **FSY** and **SCK** are outputs.
10 – **FSY** and **SCK** are outputs. **SR0, SX0** are SPDIF format, see Figure 3-9.
SR1, SX1 are normal audio format (format controlled by **EDG**, **DEL**, and **POL**).
11 – Disables the Source Ports, allowing **IOC[5:0]** to be used as EGPIO.

$\overline{\text{MUT}}$ Mute. When clear, **SX0** and **SX1** outputs are digitally muted (signal at ground).

When configured as an SPDIF transceiver, **FSY** is synchronized to the SPDIF data. The edge of **FSY** that identifies a left sample, as determined by **bSDC1.POL**, occurs between the left preamble and the LSB of the left sample. Data is transferred in bytes starting at this point.

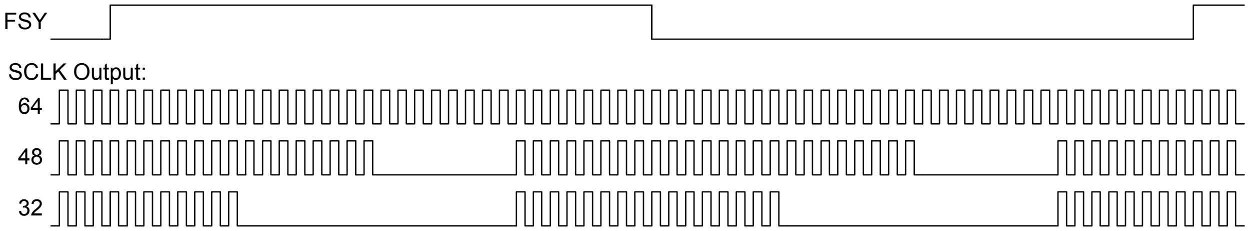


Figure 3-16: Source-Port SCLK Output Timing (bSDC1.EDG = 0)

8Ch bSDC2 Source Port Control 2 MOST

Bit	Label	Description	Default
7..4	rsvd	Reserved. Write to 0.	0000
3	MUTE	When set, mutes (holds at 0) **FSY** and **SCK**, when configured as outputs.	0
2..0	rsvd	Reserved. Write to 0.	000

Table 3-48: bSDC2 Register

3.4 MOST Register Summary

For additional details on how to use these registers, refer to the *OS8104 MOST Transceiver* Data Sheet.

Name	Label	Addr.	Description	Page
MOST Routing Table (MRT):				
	bRE0 … bRE3B	00h … 3Bh	MOST Routing Table Locations for Network Transmitted Synchronous data.	106
	bRE40 … bRE7E	40h … 7Eh	MOST Routing Table Locations for Source Port outputs (**SX0/1**), DAC outputs as well as DSP input Routing ports.	106
Hardware Control Section:				
Transceiver Control	bXCR	80h	timing-Master/slave select, and bypass control	114
Transceiver Status	bXSR	81h	**ERR** pin error conditions and masks	115
Source Port Control	bSDC1	82h	Source Port Configuration	136
Network Control	bNC	84h	*Remote Write* disable and Packet Rec. enable	115
Message Control	bMSGC	85h	Control mes. status resets, start TX, clear Rec.	121
Message Status	bMSGS	86h	Control message status	122
Node Position	bNPR	87h	Lower byte of Physical address	117
Interrupt Enable	bIE	88h	Interrupt enables for different sources	117
Group Address	bGA	89h	Lower byte of Group address for group cast	117
Node Address High	bNAH	8Ah	Logical address, high byte	117
Node Address Low	bNAL	8Bh	Logical address, low byte	118
Source Port Control 2	bSDC2	8Ch	**FSY/SCK MUTE** (disable outputs) bit	137
Clock Manager 2	bCM2	8Eh	PLL lock status bit	118
Node Delay	bNDR	8Fh	Node delays from timing-master node	118
Maximum Position	bMPR	90h	Total number of nodes in the network	118
Maximum Delay	bMDR	91h	Total delay around the network	118
Synchronous Bandwidth Control	bSBC	96h	Source data allocated to synchronous data	119
Transceiver Status Register 2	bXSR2	97h	**INV** (invert) bit for incoming **RX** data	119
Source Port Routing	bSPR	98h	Enable/disable for Source Port routing	132
DSP0 Primary MOST Port Routing	bD0RP	99h	Enable/disable for DSP0 Primary Routing port	133
DSP0 Second. MOST Port Routing	bD0RS	9Ah	Enable/disable for DSP0 Secondary Routing port	133
DSP1 Primary MOST Port Routing	bD1RP	9Bh	Enable/disable for DSP1 Primary Routing port	134
DSP1 Second. MOST Port Routing	bD1RS	9Ch	Enable/disable for DSP1 Secondary Routing port	134
Source Converter Routing	bFPR	9Dh	Enable/disable for Source Converter routing	133
Receive Control Message Buffer: mRCMB				123
Received Message Type	bRTYP	A0h	Addressing mode used in received message	
Source Address - High byte	bRSAH	A1h	Sending devices logical address, high byte	
Source Address - Low byte	bRSAL	A2h	Sending devices logical address, low byte	
Received Control Data 0 Received Control Data 1 to Received Control Data 15 Received Control Data 16	bRCD0 bRCD1 ... bRCD15 bRCD16	A3h A4h ... B2h B3h		
Control Message Transmit Control:				
Transmit Retry Time	bXTIM	BEh	Waiting time between Control mes. retries	123
Transmit Retries	bXRTY	BFh	Total number of Control message retries	123

Table 3-49: MOST Routing Bus Register Summary

Name	Label	Addr.	Description	Page
Transmit Control Message Buffer: mXCMB				124
Transmit Priority	bXPRI	C0h	00 - lowest priority, 0Fh - highest priority	
Transmit Message Type	bXTYP	C1h	Normal, Remote, Resource alloc./dealloc.	
Target Address - High byte	bXTAH	C2h	high byte of target node address	
Target Address - Low byte	bXTAL	C3h	low byte of target node address	
Transmit Control Data 0 Transmit Control Data 1 to Transmit Control Data 15 Transmit Control Data 16	bXCD0 bXCD1 ... bXCD15 bXCD16	C4h C5h ... D3h D4h		
Transmit Status	bXTS	D5h	Received okay,or failed and why	124
Packet Data Control:				
Packet Control	bPCTC	E2h	Controls clearing of status flags	129
Packet Status	bPCTS	E3h	Indicates transmit and receive Packet status	130
Alternate Packet Address High	bAPAH	E8h	Receive Packet Alternate address high	128
Alternate Packet Address Low	bAPAL	E9h	Receive Packet Alternate address low	128
Transmit Packet Start	bPSTX	EAh	Start bit to send a packet in mAXP to Network	129
Transmit Packet Length	bPLDT	ECh	Transmit Packet length in quadlets	128
Transmit Packet Priority	bPPI	F2h	Transmit Packet priority - 01 to 07 (highest)	129
Memory Page change		FFh	Can write to change memory pages to 1 or 3	
Receive Packet Message Buffer: mARP				130
Receive Address high	bARTH	180h	Either Logical or Alternate Packet Address high	
Receive Address low	bARTL	181h	Either Logical or Alternate Packet Address low	
Source Address - High byte	bASAH	182h	Sending devices logical address, high byte	
Source Address - Low byte	bASAL	183h	Sending devices logical address, low byte	
Received Packet Data 0 Received Packet Data 1 to Received Packet Data 46 Received Packet Data 47	bARD0 bARD1 ... bARD46 bARD47	184h 185h ... 1B4h 1B5h		
Transmit Packet Message Buffer: mAXP				131
Target Address - High byte	bXTAH	1C0h	high byte of target node address or alternate packet address	
Target Address - Low byte	bXTAL	1C1h	low byte of target node address or alternate packet address	
Transmit Packet Data 0 Transmit Packet Data 1 to Transmit Packet Data 46 Transmit Packet Data 47	bAXD0 bAXD1 ... bAXD46 bAXD47	1C2h 1C3h ... 1F2h 1F3h		
Memory Page change		1FFh	Can write to change memory pages to 0 or 3	
Channel Resource Allocation Table: mCRA				119
Synchronous Channel 0 Synchronous Channel 1 to Synchronous Channel 58 Synchronous Channel 59	bCRA0 bCRA1 ... bCRA58 bCRA59	380h 381h ... 3BAh 3BBh		
Memory Page change		3FFh	Can write to change memory pages to 0 or 1	

Table 3-49: MOST Routing Bus Register Summary (Continued)

Below is the MOST Hardware Control Section, bit summary.

I/O Addr.	Mnemonic	B7	B6	B5	B4	B3	B2	B1	B0	Page
80h	bXCR	MTR	OE	NMEN			SBY	$\overline{\text{ABY}}$		114
81h	bXSR		MSL	MXL	ME	ERR		ESL	EXL	115
82h	bSDC1	EDG	DEL	POL	NBR1	NBR0	MOD1	MOD0	$\overline{\text{MUT}}$	136
84h	bNC							APREN	RWD	115
85h	bMSGC	STX	RBE	RES	SAI	RALC	RERR	RMTX	RMRX	121
86h	bMSGS	RBS	TXR			ALC	ERR	MTX	MRX	122
87h	bNPR	Physical Address byte								117
88h	bIE					IALC	IERR	IMTX	IMRX	117
89h	bGA	Group Address byte								117
8Ah	bNAH	Logical Address - high byte								117
8Bh	bNAL	Logical Address - low byte								118
8Ch	bSDC2					MUTE				137
8Eh	bCM2	$\overline{\text{LOC}}$	NNAC							118
8Fh	bNDR	Node Delay byte								118
90h	bMPR	Total number of nodes in the network byte								118
91h	bMDR	Total network delay byte								118
96h	bSBC					SAC3	SAC2	SAC1	SAC0	119
97h	bXSR2							INV		119
E2h	bPCTC				RAF	RAC		RATX	RARX	129
E3h	bPCTS				AF	AC		ATX	ARX	130
E8h	bAPAH	Alternate Packet Address - high byte								128
E9h	bAPAL	Alternate Packet Address - low byte								128
EAh	bPSTX	ASTX								129
ECh	bPLDT	Packet Length, in quadlets								128
F2h	bPPI	Packet Priority, 01-07 (highest)								129

Table 3-50: MOST Hardware Control Registers - Bit Summary

4 Digital Signal Processors

The OS8805 has two on-chip digital signal processors, DSP0 and DSP1. The DSPs are both 18x14-bit Gazelle DSP cores which are customized to provide the optimum resources for a wide variety of signal processing applications. The default operating frequency for the DSPs is 1344xFs, which provides 59 MIPS. DSP0 and DSP1 are identical and have the same amount of Program memory, Vector memory, Pointer memory, and computation power. Both DSP's have a COM port interface to the Host Controller, a MOST Routing Port to the MOST Processor, and a global timing register. However, the DSP0 has additional I/O peripherals: two pulse width modulation (PWM) DACs, and an external data memory interface.

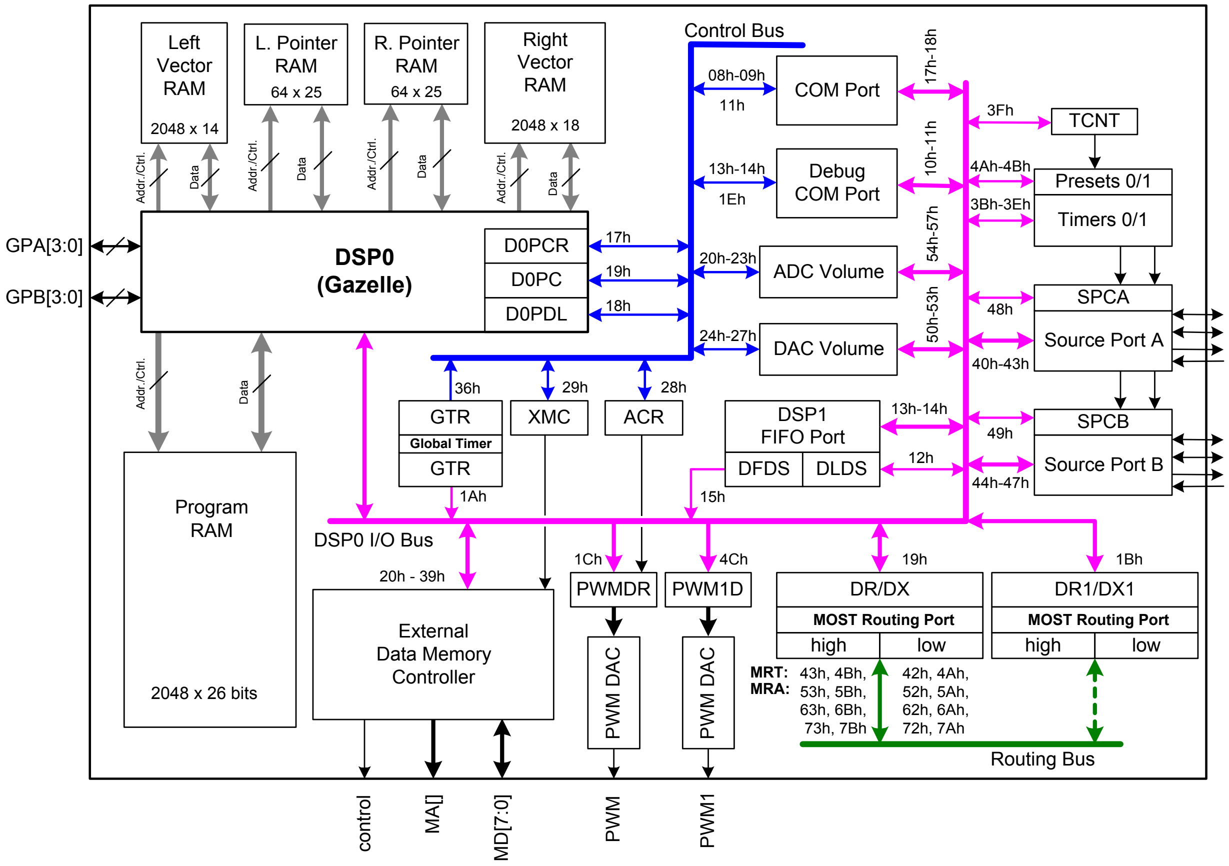


Figure 4-1: DSP0 I/O Bus

The DSPs are controlled by the Host Controller through the COM port interface and the program control interface. The DSP's access MOST Network data, Source Converters, and the Source Port through the MOST Routing Port. The data can be processed with a digital signal processing algorithm and the resulting data returned through the MOST Routing Port to the MOST Network, the Source Port, or the Source Converters.

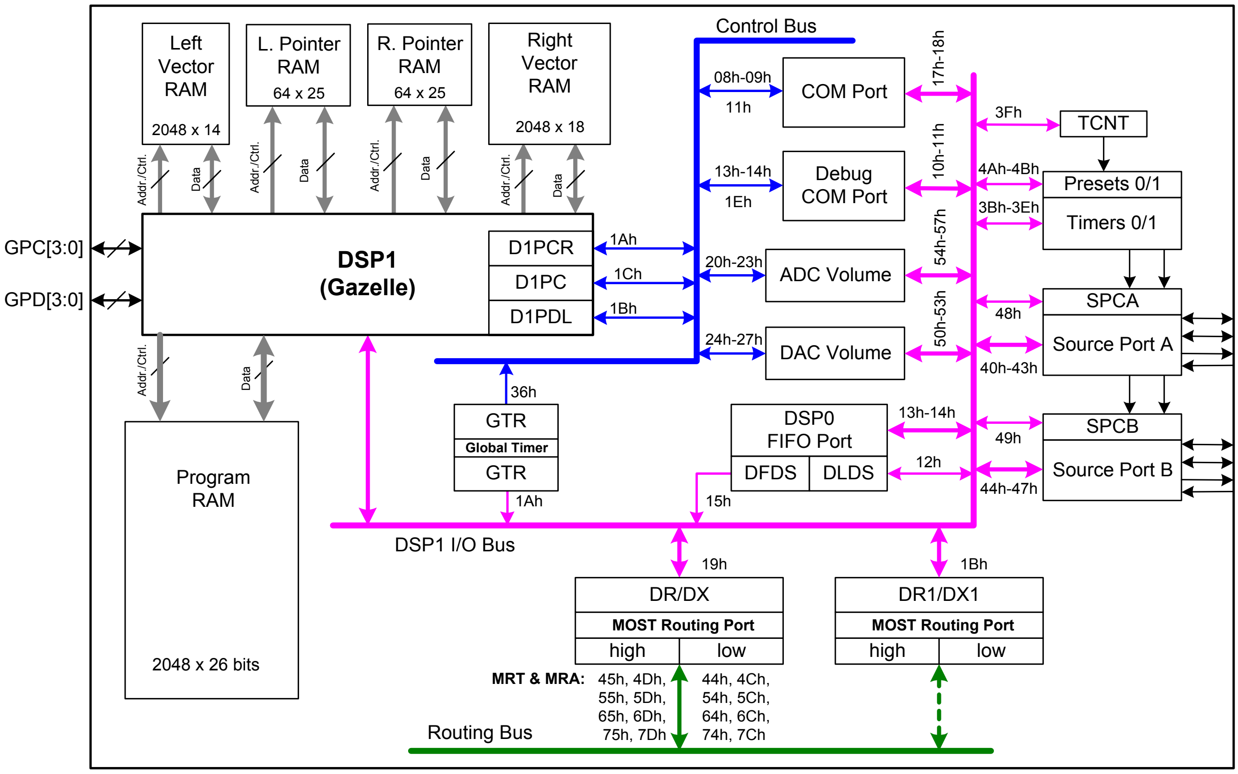


Figure 4-2: DSP1 I/O Bus

4.1 Architecture

The OS8805 DSP core consists of a Program memory, two Vector (data) memories, two Pointer memories, a program controller, an interrupt controller, two address generation units, and an execution unit. The Program memory is 26 bits wide and 2048 locations deep. The Vector memory is divided into left and right sides. The left Vector memory is 14 bits wide and 2048 locations deep and the right Vector memory is 18 bits wide and 2048 locations deep. The Pointer memory is also divided into left and right sides. Both Pointer memories are 25 bits wide and 64 locations deep

Program memory stores the application program and the Data memories store data and coefficients for digital signal processing. As illustrated in Figure 4-3, a 26-bit instruction word in Program memory is comprised of an 8-bit opcode, followed by two 9-bit operands. The 9-bit operand is further divided into a 3-bit memory type field (Control), and a 6-bit address field. The left and right Pointer memories store address pointer values, which consists of an 11-bit address field, a 6-bit update field, and an 8-bit modulo field.

Different functional blocks in the execution unit are provided to support the digital signal processing operation. A fast single-cycle multiply-and-accumulate supports arithmetic-intensive digital signal processing algorithms. A bit manipulation unit and a shifter support data packing, extraction and scaling. Two data address generation units support normal and specialized address modes, and calculate data memory addresses. Lastly, a DSP I/O data bus interfaces to the on-chip peripherals.

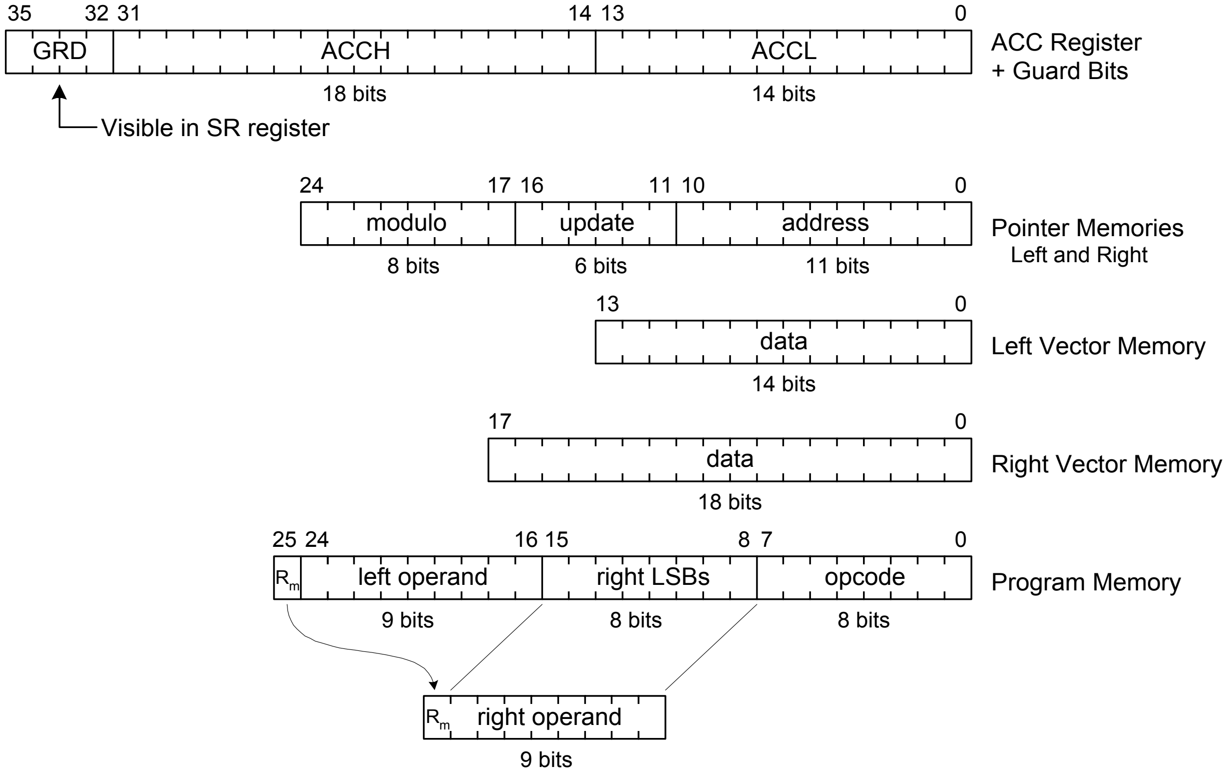


Figure 4-3: DSP Memory Architecture

The Gazelle DSP employs a four-stage pipeline: a fetch stage, a read pointer stage, a read data stage and an execute stage. Since long sequences of operations are broken down into smaller portions, higher performance of the processor can be achieved.

The following sections supply a general overview and describe the variations of Gazelle that are specific to the OS8805. The differences are restricted to memory, I/O registers, condition codes, and interrupts. For detailed description and programming information on the Gazelle DSP, see the *Gazelle User's Manual*.

4.1.1 Program Memory

The DSP has 2048x26 bits of Program memory. Program memory is accessed by the Host Controller through the program control registers. Since the DSP Program memory is RAM, the Host Controller must download programs into the DSP's Program memory during initialization.

4.1.2 Vector Memory

The Gazelle DSP employs a dual-data memory architecture: left Vector and right Vector memory. Therefore, two data operands can be fetched in a single instruction: one on the left data bus and the other on the right data bus. The left and right data address buses are obtained from either the opcode for direct addressing, or the left and right Pointer memories for indirect addressing. Direct addressing is supported for the first 64 words of each Vector memory.

4.1.3 Pointer Memory

The Gazelle DSP also employs a dual-pointer memory architecture: the left Pointer and the right Pointer memory. Therefore, two indirect addresses can be manipulated in a single instruction. The Pointer memories are 25-bit wide and consist of three fields: an 11-bit address field, an 8-bit modulo field and a 6-bit update field. The Pointer memories are generally used as address registers since they support post incrementing and modulo arithmetic. Special instructions support reading and writing the Pointer memories.

The 11-bit address field specifies the indirect address of the corresponding 2048 words of Vector memory. Both the left and right Pointer memories contain 64 pointers.

The 6-bit update field specifies the post-update value of the indirect address. Pointer addresses can be updated by this update value or one. The pointer ALU's are only 10 bits; therefore, the upper pointer address bit selects one of two 1K pages. Vector memory data arrays should not cross the 1K boundary since Pointer memory post-updates will not transition across the page boundary, but will wrap to the start of the page. For example, if a pointer with the value 0x00003FF is post-incremented by 1, the value would be 0x00000000, not 0x0000400.

The 8-bit modulo field specifies the size of the circular buffer. An 8-bit modulo field supports modulo buffer sizes of 2 to 256 words (modulo value plus one). Since the upper address bit selects between two pages as mentioned previously, a 1024 byte (half the memory) modulo buffer also exists by setting the modulo value to zero. The modulo buffer size can be an arbitrary length, starting on a 2^N block boundary, where N is the number of bits required to represent the modulo value. The highest bit set in the modulo field determines N. For example, if the modulo value is 5, then the start address is xxxxxxxx000 and the upper address is xxxxxxxx101. Except when using the entire memory page as a modulo buffer (1024 words), a modulo value of zero defines no modulo buffer.

4.1.4 Program Controller

The program controller contains the 11-bit program counter (PC) and its associated increment and decrement logic. Additional logic includes an interrupt shadow program counter, a subroutine shadow program counter, and the instruction loop control logic which includes the start register, the end register, and the count register.

The program controller is responsible for the sequencing of the program flow. The program counter always contains the address of the next instruction to be fetched. The interrupt shadow program counter, the subroutine shadow program counter and the shadow strategic registers provide fast context switching for critical code. For non-critical code, the DSP supports creation of software stacks.

All registers can be read by the `I/O read` instruction and written by the `I/O-write` instruction. The program counter has two I/O addresses to differentiate between a return-from-interrupt and a return-from-subroutine.

Two types of Interrupt Service Routines (ISRs) are defined: short and long. Short ISRs are seven or less instructions and execute while interrupts are disabled. These routines typically use the `RETI` instruction to return from an interrupt. Long ISRs are greater than seven instructions and can be interrupted (if desired) by a higher priority interrupt. To support nested interrupts, the ISPC register must be stored on a software stack so the return address of the main program can be restored. Once the ISR has completed, the ISPC value stored on the software stack can be directly written to the PCI register. When the PCI register is written, the SR, ACCH, and ACCL are swapped with their shadow registers. Writing PCI directly provides a more code-efficient method of returning than popping the stored ISPC value back to the ISPC register and executing a `RETI` instruction.

Two types of subroutines are also defined: short and long. Short subroutines are not nested; whereas long subroutines support subroutine nesting. Short subroutines generally use the `RET` instruction to exit from the subroutine. For long subroutines, the SSPC return value must be stored on a software stack before the next (nested) subroutine is entered. When exiting from a long subroutine, the PC can be written directly from the SSPC value stored on the software stack. This provides a more code-efficient method of returning than popping the stored SSPC value back to the SSPC register and executing a `RET` instruction.

00h PC Program Counter DSPs

Bit	Label	Description	Default
17..11	rsvd	Reserved, Write to 0	0000000
10..0	D[10:0]	Program Counter. Typically written to return from long (nested) subroutines by popping the PC value off a software stack directly to this register.	00h, 000

Table 4-1: PC Register

01h PCI Program Counter Interrupt DSPs

Bit	Label	Description	Default
17..11	rsvd	Reserved, Write to 0	0000000
10..0	D[10:0]	Program Counter Interrupt. Typically written to return from a long interrupt service routine by popping the PC value off a software interrupt stack directly to this register. Writing PCI causes the SR, ACCH, and ACCL registers to be swapped with their shadow counterparts.	00h, 000

Table 4-2: PCI Register

02h ISPC Interrupt Shadow Program Counter DSPs

Bit	Label	Description	Default
17..11	rsvd	Reserved, Write to 0	0000000
10..0	D[10:0]	Interrupt Shadow Program Counter. Swapped with PC during an interrupt that is not the Debug Interrupt. Restored (swapped again) to PC on a return from interrupt (`RETI`) instruction when not in the Debug Interrupt. Generally used in short ISRs.	00h, 000

Table 4-3: ISPC Register

03h SSPC Subroutine Shadow Program Counter DSPs

Bit	Label	Description	Default
17..11	rsvd	Reserved, Write to 0	0000000
10..0	D[10:0]	Subroutine Shadow Program Counter. Swapped with PC during a subroutine call (`JMPS` or `D_JMPS`). Restored to PC on a return from a subroutine (`RET`) which is generally used in short (non-nested) subroutines.	00h, 000

Table 4-4: SSPC Register

The loop control logic contains: the start register which stores the 11-bit start address location of the loop, the end register which stores the 11-bit end address location of the loop, and the count register which stores the 16-bit repeat value. There are no shadow registers for CNT, STRT, and END. They must be saved on the general purpose stack to support nested loops and loops within multiple ISR’s.

09h CNT Repeat Count DSPs

Bit	Label	Description	Default
17,16	rsvd	Reserved, Write to 0	00
15..0	D[15:0]	Count value for loops and repeats	0000h

Table 4-5: CNT Register

0Ah END End Address DSPs

Bit	Label	Description	Default
17..11	rsvd	Reserved, Write to 0	0000000
10..0	D[10:0]	End address for hardware looping	00h, 000

Table 4-6: END Register

0Bh STRT Start Address DSPs

Bit	Label	Description	Default
17..11	rsvd	Reserved, Write to 0	0000000
10..0	D[10:0]	Start address for hardware looping	00h, 000

Table 4-7: STRT Register

The count, start, and end registers (and associated logic) enable blocks of program code to be looped and single word instructions to be repeated without any cycles lost to overhead. If the PC value is equal to the end address and the count value is greater than zero, the PC will jump to the start address. A “repeat” instruction is formed by setting the start and end addresses, of a loop instruction, to the same value.

4.1.5 Interrupt Controller

The interrupt controller contains the interrupt priority logic, the interrupt issuing logic, and the interrupt queuing logic. The interrupt controller is also responsible for providing the interrupt vector, corresponding to the active interrupt, to the program controller for fetching of instructions. Some of the interrupt sources are the global timing register, the COM port to the Host Controller, the FIFO Port to the other DSP, the Async. Source Ports, and a GPIO pin. The global timing register flags, which can interrupt the DSP, are **FS8**, **FS4**, **FS2**, **FS1**, **FS4TH**, and a Fs/64 (which is not visible in GTR). The interrupts occur on the rising edge of the corresponding global timing register bit. The Fs/64 interrupt occurs on every fourth rising-edge of the Fs/16 flag **GTR.FS16TH**.

The GTR flags and peripheral routing are synchronized to **SCK**. If the Source Port is enabled (**bSDC1.MOD[1:0]** not set to 11), then **SCK** must be configured as an output, or **SCK** and **FSY** must have the proper external clocks applied when configured as inputs.

The interrupt vectors are spaced eight words apart starting with the **FS8** vector at 008h. The reset vector is 000h. The Vector table is illustrated in Figure 4-4. The interrupt enable register contains the bits to independently enable each interrupt (except the COM and FIFO Ports). If an interrupt occurs when it is disabled, it becomes pending until that interrupt is enabled.

The FIFO Port has a unique interrupt vector; however, it shares the interrupt enable and priority settings with the COM Port interrupt. For the FIFO Port to generate an interrupt, the **IER.IECP** must be set and the **DFLS.IMSK** bit must be clear. The interrupt can be used solely for the FIFO Port by setting the **DCS.CIM** mask bit, which masks COM Port interrupts. This allows the COM Port to be polled, while the DSP FIFO port uses the interrupt line.

When Async. Source Port (SP) 0 is disabled, the GPIO interrupt vector is shared with the GPIO pin and Async. SP 1. When Async. SP 0 is enabled, the interrupt vector only services the Async. Source Ports as the GPIO pin is part of SP 0. The GPIO priority affects both the GPIO pin and the Async. Source Ports.

The Host Controller setting **DDnCS.TRINT** or a `TRAP` instruction execution can cause the Debug interrupt.

All interrupts, except the GPIO and COM Port, have fixed priority. The GPIO interrupt priority is programmable. The GPIO interrupt services the Asynchronous Source Ports, when enabled. When the Async. Source Port 0 is disabled, the GPIO interrupt is used for **GPA0** on DSP0 or **GPC0** on DSP1. The COM port interrupt priority can be programmed as either the second-highest or lowest. When the COM port interrupt is the second-highest priority, it is higher than any other interrupt including GPIO, but not the Debug inter-

		0x001	0x002	0x003
0x000	Reset vector			
0x004				
0x008	8xFs Global Timer vector			
0x00C				
0x010	4xFs Global Timer vector			
0x014				
0x018	2xFs Global Timer vector			
0x01C				
0x020	Fs Global Timer vector			
0x024				
0x028	1/4Fs Global Timer vector			
0x02C				
0x030	1/64Fs Timer vector			
0x034				
0x038	COM Port vector			
0x03C				
0x040	GPIO and Async. Source Ports vector			
0x044				
0x048	FIFO Port vector			
0x04C				
0x050	Debug vector			
0x054				

Figure 4-4: DSP Vector Table

rupt vector. When the COM port interrupt is the lowest priority, it is lower than any other interrupt including GPIO. The COM Port interrupt enable and priority are shared with the FIFO Port, however, the interrupt vector is unique. Interrupt priority should not be changed without first disabling the associated interrupt and servicing any pending interrupts; otherwise, the wrong interrupt vector could be selected when reenabling the interrupt. The Debug interrupt vector is non-maskable and has the highest priority.

Interrupt	Interrupt Vector	Priority
Debug	50h	0 (highest)
8Fs	08h	1
4Fs	10h	2
2Fs	18h	3
Fs	20h	4
4th	28h	5
64th	30h	6
COM and FIFO Port	38h and 48h	second-highest or lowest
GPIO pin and Async. Source Ports	40h	programmable

Table 4-8: DSP Interrupt Priority

04h *IER* *Interrupt Enable Register* *DSPs*

Bit	Label	Description	Default
17,16	rsvd	Reserved, Write to 0	00
15	GPOL	Polarity of GPIO interrupt, low is rising edge or high level	0
14	LEV	GPIO level or edge sensitive, low is edge	0
13	rsvd	Reserved, Write to 0	0
12	rsvd	Reserved, Write to 0	0
11..9	PRI[2:0]	GPIO interrupt priority	000
8	CPHI	COM port and FIFO port interrupt priority	0
7	IEGP	GPIO and Async. Source Ports interrupt enable	0
6	IECP	COM port and FIFO port interrupt enable	0
5	IE64TH	64th interrupt enable †	0
4	IE4TH	4th interrupt enable †	0
3	IE1FS	Fs interrupt enable †	0
2	IE2FS	2Fs interrupt enable †	0
1	IE4FS	4Fs interrupt enable †	0
0	IE8FS	8Fs interrupt enable †	0

† Requires the proper **SCK** freqnency or the Source Port to be disabled

Table 4-9: IER Register

GPOL GPIO Polarity. When clear, the GPIO is rising-edge or active high. This bit provided for backwards compatibility. Newer software should use register IPOT to set the polarity of **GPA0** for DSP0 or **GPC0** for DSP1. See Figure 4-15.

LEV GPIO Level Sensitive. When clear, GPIO is edge-sensitive. Cleared by writing a zero to **EGPD.GPDA0** for DSP0 and **EGPD.GPDC0** for DSP1. This bit provided for backwards compatibility. Newer software should use register ISOD to set level or edge-sensitive configuration of **GPA0** for DSP0 or **GPC0** for DSP1. See Figure 4-15.

PRI[2:0] Priority level of GPIO. Sets the priority at which the GPIO interrupt occurs. The GPIO interrupt must be disabled (**IEGP** clear), and any pending interrupt cleared, before changing these bits.

000 – Lower than 7
001 – between 7 and 6
010 – between 6 and 5
011 – between 5 and 4
100 – between 4 and 3
101 – between 3 and 2
110 – between 2 and 1
111 – higher than 1 but lower than 0

CPHI COM Port and FIFO port Priority High. When clear, the COM port and FIFO Port have the lowest priority. When set, the COM Port and FIFO Port have the second-highest priority. The COM Port interrupt must be disabled (**IECP** clear), and any pending interrupt cleared, before changing this bit.

IEGP Interrupt Enable for GPIO and Async. Source Ports.
When Async. Source Port 0 is enabled, **IEGP** set enables the interrupt with the status available in register DTC.
When Async. Source Port 0 is disabled, **IEGP** allows **GPA0** for DSP0 or **GPC0** for DSP1 to generate an interrupt, or Async. Source Port 1. For GPIO, the level/edge and polarity should be set via the EGPIO IPOT and ISOD registers (see Figure 4-15). The interrupt priority is programmable through the **PRI[2:0]** bits.

IECP Interrupt Enable for COM Port and FIFO Port. When set, the COM Port generates an interrupt that vectors to memory location 038h, on the rising edge of the **STR**, **RD** or **WR** bits in the DCS register. Also when **IECP** is set and **DFLS.IMSK** is clear, the FIFO port generates an interrupt that vectors to memory location 048h, on the rising edge of **DFFS.STR** or **DFFS.WR**. The interrupt priority is programmable to highest (**CPHI** set) or lowest.

IE64TH Interrupt Enable at one 64th Fs. Although this bit is not visible in the GTR register, the bit is synchronous with the GTR bits and causes an interrupt on every fourth rising-edge of the Fs/16 flag **GTR.FS16TH**. The interrupt vectors to memory location 030h and has a priority of 6.

IE4TH Interrupt Enable at one 4th Fs. When set, causes an interrupt on the rising-edge of the **GTR.FS4TH** bit, which vectors to memory location 028h and has a priority of 5.

IE1FS Interrupt Enable at Fs rate. When set, causes an interrupt on the rising-edge of the **GTR.FS1** bit, which vectors to memory location 020h and has a priority of 4.

IE2FS Interrupt Enable at two times Fs. When set, causes an interrupt on the rising-edge of the **GTR.FS2** bit, which vectors to memory location 018h and has a priority of 3.

IE4FS Interrupt Enable at four times Fs. When set, causes an interrupt on the rising-edge of the **GTR.FS4** bit, which vectors to memory location 010h and has a priority of 2.

IE8FS Interrupt Enable at eight times Fs. When set, causes an interrupt on the rising-edge of the **GTR.FS8** bit, which vectors to memory location 008h and has a priority of 1.

The `TRAP` instruction is a software interrupt and can be used for debugging. The `TRAP` instruction cannot be placed in the three instructions after a delay jump, delay jump to subroutine, return, return from interrupt, I/O write to PC or PCI. `TRAP` can also not be the second word of a two-word instruction. During the execution phase of the `TRAP` instruction, the PC is loaded with the Debug interrupt vector and the DSPn Trap Shadow PC register, on the Control bus, is loaded with the `TRAP` instruction address location. In addition, SR is loaded into TSSR. Software must restore SR when returning from a debug interrupt. The first `RETI` after a Debug Interrupt copies the Trap Shadow PC (visible on the Control bus) value back to the PC; however, no other registers are swapped. All interrupts are disabled once the `TRAP` instruction is executed and while in the Debug interrupt service routine. The Host Controller can also cause the Debug interrupt vector to be fetched by setting the **DDnCS.TRINT** bit.

When an interrupt occurs (other than the Debug Interrupt), the ISPC is loaded with the interrupt vector. The instruction fetched during the interrupt is not executed. For the first three cycles of an interrupt service routine, the SPC will specify the program memory location. At the end of the third cycle of the interrupt service routine, the PC and ISPC are swapped, and ACCH, ACCL, and SR are swapped with their respective shadow registers. When returning from an interrupt, the contents of PC are swapped with ISPC, and the contents of ACCH, ACCL, and SR are swapped with their respective shadow registers.

Interrupts can be long or short. Short interrupts cannot be interrupted. They are either the highest priority ISR or they are seven or less instructions long. Short interrupts use the shadow registers as a hardware stack. A short interrupt is exited by executing a return from interrupt instruction (`RETI`).

4.1.6 Execution Unit

The Execution Unit operations have a four-stage pipeline. During the first stage, the instruction is read from Program memory and stored in the Instruction Register (IR). During the second stage, a pointer value is read from Pointer memory. During the third stage, Vector memory is read. During the fourth stage, the ALU operation is performed and the result is written to the specified destination.

The internal data bus is 18 bits wide. Since data needs to be transferred between the ACC and memories of different width, alignment issues must be addressed. For Sources going to the data bus, Figure 4-5 illustrates the alignment. The ACC and data memories are MSB aligned and the address/Pointer memories are LSB aligned.

For data going to destinations, Figure 4-6 illustrates the data alignment. The ACC and Vector memories are MSB aligned and the different sections of the Pointer memories are LSB aligned. When writing one section of the Pointer memory, the other sections are not affected.

The execution unit updates the necessary status bits in the status register SR according to the result of the operation. The execution unit also provides shadow registers for the low accumulator, high accumulator, and the status register, to facilitate fast context switching for timing-critical interrupt service routines.

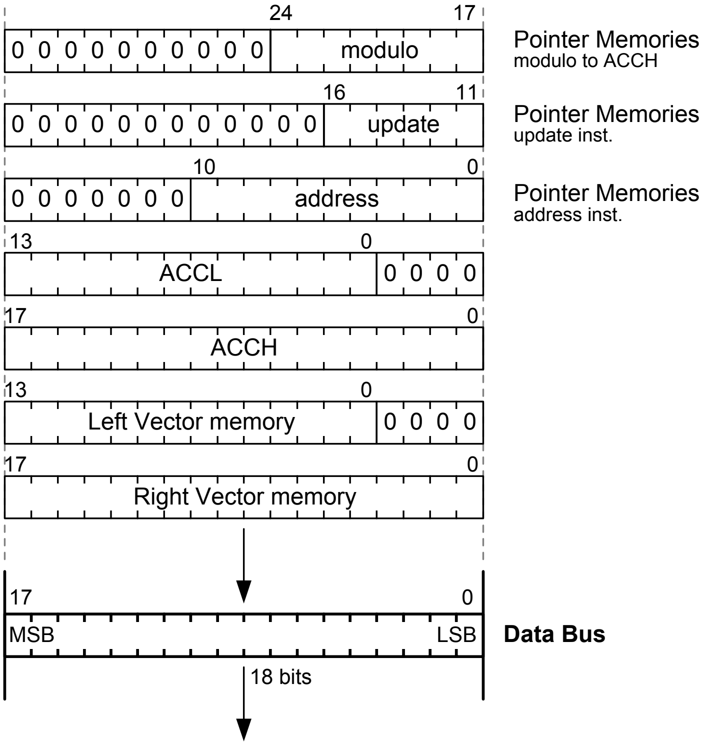


Figure 4-5: DSP Source Data Alignment

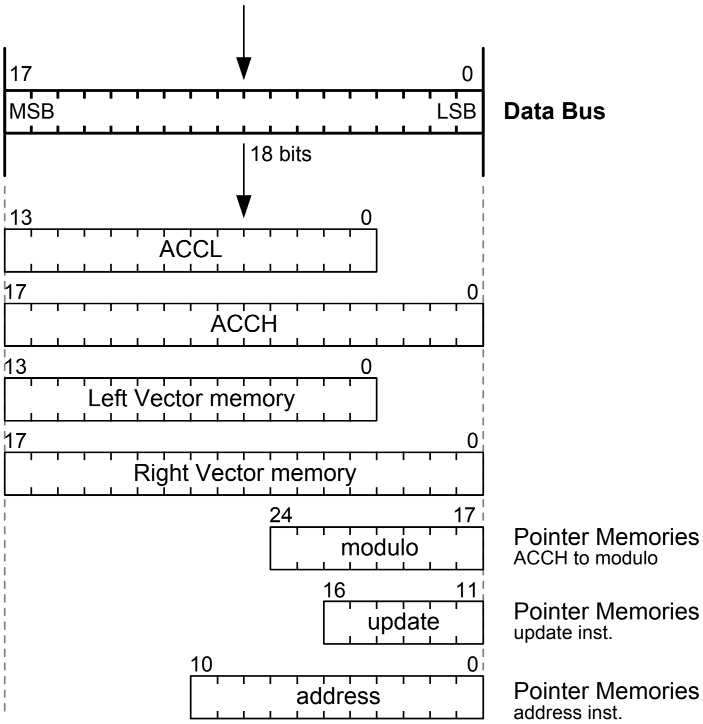


Figure 4-6: DSP Destination Data Alignment

05h SR Status Register DSPs

Bit	Label	Description	Default
17,16	rsvd	Reserved, Write to 0	00
15	C	Carry flag	0
14	N	Negative flag	0
13	Z	Zero flag	0
12	O	Overflow flag	0
11	L	Limit flag	0
10..5	rsvd	Reserved, Write to 0	000000
4	rsvd	Reserved, Write to 0	0
3..0	G[3:0]	Guard bits	0000

Table 4-10: SR Register

C Carry flag. Set when an addition operation produces a carry out or when a subtraction operation requires a borrow. Used in the conditional jump and some arithmetic instructions.

N Negative flag. Set when the MSB of an arithmetic operation is set. Used in the conditional jump instruction.

Z Zero flag. Set when the result of an arithmetic operation is zero. Used in the conditional jump instruction.

O Overflow flag. Set when a two's complement overflow occurs as the result of an operation.

L Limit flag. Set when saturation occurs as the result of a shift operation with saturation.

G[3:0] Guard bits. Upper four bits of the accumulator, used to guard against overflow of intermediate results. Used with the entire accumulator (ACC) in MAC operations or just the higher portion of the accumulator (ACCH).

06h SSR Shadow Status Register DSPs

Bit	Label	Description	Default
17,16	rsvd	Reserved, Write to 0	00
15..0	SSR[15:0]	Shadow Status Register.	0000h

Table 4-11: SSR Register

SSR[16:0] Shadow Status Register. Loaded with SR during an interrupt that is not the Debug Interrupt. Restored to SR on a return from an interrupt (`RETI`), except when in a Debug Interrupt.

07h SACCL Shadow Accumulator Low DSPs

Bit	Label	Description	Default
17..4	SACCL[14:0]	Shadow Accumulator Low.	000h, 00
3..0	rsvd	Reserved, Write to 0	0000

Table 4-12: SACCL Register

SACCL[14:0] Shadow Accumulator Low. Loaded with ACCL during a non-Debug interrupt. Restored to ACCL on a return from an interrupt (`RETI`), except when in a Debug Interrupt.

08h SACCH Shadow Accumulator High DSPs

Bit	Label	Description	Default
17..0	SACCH	Shadow Accumulator High.	0000h, 00

Table 4-13: SACCH Register

SACCH[17:0] Shadow Accumulator High. Loaded with ACCH during a non-Debug interrupt. Restored to ACCH on a return from an interrupt (`RETI`), except when in a Debug Interrupt.

0Ch TSSR Trap Shadow Status Register DSPs

Bit	Label	Description	Default
17,16	rsvd	Reserved, Write to 0	00
15..0	TSSR[15:0]	Trap Shadow Status Register.	0000h

Table 4-14: TSSR Register

TSSR[15:0] Trap Shadow Status Register. The Debug Interrupt is fetched and TSSR is loaded with SR when a `TRAP` instruction is executed, or the Host Controller sets **DDnCS.TRINT**. The programmer must restore SR manually when leaving the Debug Interrupt (`RETI`).

4.1.7 Address Generation Unit

The processor supports three different operand addressing modes: register, direct, and indirect. When the operand resides in the accumulator or an I/O register, the register identifier is encoded in the instruction. Indirect addressing reads one of 64 Pointer memory locations to obtain the address for the data memory operand. Direct addressing is available for the first 64 words of Vector data memory.

The indirect addressing mode also supports post incrementing and decrementing of the data address stored in the Pointer memory. The address is incremented and decremented by either one or an update value stored along with the data address in the Pointer memory. The address calculation can also be performed using modulo arithmetic, where the modulo value is also stored along with the data address in Pointer memory. See Figure 4-3.

4.1.7.1 Register Addressing Mode

Register operands are explicitly encoded within the instruction. The high accumulator and the low accumulator are hardware registers which can be register direct operands. I/O registers are read and written using separate `I/O read` and `I/O write` instructions.

4.1.7.2 Direct Addressing Mode

The DSP can directly address the first 64 words of both the left and right Vector data memory in direct addressing mode.

4.1.7.3 Indirect Addressing Mode

When using the indirect addressing mode, an instruction does not directly specify the address of the Vector memory operand, but instead supplies the location, in Pointer memory which contains the Vector memory address. The Vector memory operand address is read from the Pointer memory before the Vector data memory is read from or written to.

Along with the Vector memory address, the Pointer memory also stores a modulo and an update value, which can be used during the post incrementing or decrementing of the address. See Figure 4-3.

4.1.8 Instruction Summary

Instruction	Syntax	Words	Cycles
absolute value	acc = abs(acc), <ls update>, <rs update>;	1	1
add	d = ls + rs, <ls update>, <rs update>;	1	1
add with carry	d = ls + rs + carry, <ls update>,<rs update>;	1	1
and	d = ls & rs, <ls update>, <rs update>;	1	1
bit clear	s.# = 0;	1	1
bit set	s.# = 1;	1	1
clear accumulator and status	clr, <ls update>, <rs update>;	1	1
compare	dummy = ls – rs, <ls update>, <rs update>;	1	1
io read	d = ioreg;	1	1
io write	ioreg = d;	1	1
jump bit clear	if (!#) jmp label;	1	1 to 4
jump bit set	if (#) jmp label;	1	1 to 4
delay jump bit clear	if (!#) d_jmp label;	1	1
delay jump bit set	if (#) d_jmp label;	1	1
jump conditional	if (cc) jmp label;	1	1 to 4
delay jump conditional	if (cc) d_jmp label;	1	1
jump condition subroutine	if (cc) jmps label;	1	1 to 4
delay jump condition subroutine	if (cc) d_jmps label;	1	1
load	d = imm, <s update>;	2	2
loop	loop count, start, end;	2	2
move	d = s, <ls update>, <rs update>;	1	1
move acch to modulo	d.m = acch, <s update>;	1	1
move modulo to acch	acch = s.m, <s update>;	1	1
multiply	d = ls * rs, <ls update>, <rs update>;	1	1
multiply and accumulate	d = acc + ls * rs, <ls update>, <rs update>;	1	1
multiply and subtract	d = acc – ls * rs, <ls update>, <rs update>;	1	1
negate accumulator	acc = -acc, <ls update>, <rs update>;	1	1
no operation	nop;	1	1
or	d = ls | rs, <ls update>, <rs update>;	1	1
delay return from interrupt	reti, <ls update>, <rs update>;	1	1
delay return from subroutine	ret, <ls update>, <rs update>;	1	1
arithmetic shift left	d = acc << imm, <s update>;	1	1
logical shift left	d = acc < imm, <s update>;	1	1
arithmetic shift left and saturate	d = saturate(acc << imm), <s update>;	1	1
arithmetic shift left and round	d = round(acc << imm), <s update>;	1	1
arithmetic shift left, round and saturate	d = round_saturate(acc << imm), <s update>;	1	1
arithmetic shift right	d = acc >> imm, <s update>;	1	1
logical shift right	d = acc > imm, <s update>;	1	1
arithmetic shift right and saturate	d = saturate(acc >> imm), <s update>;	1	1
arithmetic shift right and round	d = round(acc >> imm), <s update>;	1	1
arithmetic shift right, round and saturate	d = round_saturate(acc >> imm), <s update>;	1	1
software reset	reset, <ls update>, <rs update>;	1	1
subtract	d = ls - rs, <ls update>, <rs update>;	1	1
subtract with carry	d = ls – rs – carry, <ls update>,<rs update>;	1	1
trap	trap;	1	4
exclusive or	d = ls ^ rs, <ls update>, <rs update>;	1	1

Table 4-15: DSP Instruction Set

4.2 Global Timer Peripheral

The Global Timer consists of an 8-bit counter, clocked at 16Fs, which can be read by the Controller, DSPs, and MOST processor. Each bit of the counter oscillates at different rates, which support inter-processor synchronization. The rates available are 8, 4, 2, 1, 1/2, 1/4, 1/8, and 1/16Fs. The **FS8**, **FS4**, **FS2**, **FS1** and **FS4TH** bits can generate a DSP interrupt. The Global Timer is illustrated in Figure 2-11 and Figure 2-12 on page 51.

1Ah GTR Global Timer Register (read only) DSPs

Bit	Label	Description	Default
17..8	rsvd	Reserved, Write to 0	00h, 00
7	FS16TH	one rising edge every 16 Fs periods	0
6	FS8TH	one rising edge every 8 Fs periods	0
5	FS4TH	one rising edge every 4 Fs periods	0
4	FSHALF	one rising edge every other Fs period	0
3	FS1	1 rising edge every Fs period	0
2	FS2	2 rising edges every Fs period	0
1	FS4	4 rising edges every Fs period	0
0	FS8	8 rising edges every Fs period	0

Table 4-16: GTR Register

4.3 Inter-Processor Communications

The inter-processor communication ports consist of the COM ports between the DSPs and the Host Controller, and the MOST Routing Ports between the DSPs and the MOST Processor. The COM ports are normally used for communicating control information between the DSPs and the Host Controller. The MOST Routing Ports allow source data to be exchanged between the DSPs and the on-chip Source Peripherals as well as the MOST Network. Each DSP has its dedicated set of inter-processor communication ports. The DSPs also have a FIFO Port between the two DSPs to allow direct inter-DSP communications.

4.3.1 MOST Routing Port

The MOST Routing Port has bi-directional data registers which enable source data to communicate between the MOST Processor (hence the MOST Network) and the DSP. Each port consists of two 16-bit registers, one for sending data in each direction. The registers co-exist at the same I/O address where DR receives data from the MOST Processor and DX transmits data to the MOST Processor.

The MOST Processor reads and writes each Routing Port (DR and DX) 8 times per audio sample. Eight 16-bit channels can be transferred into and out of each DSP. The DSP can use the Global Timer (GTR) and timer interrupts to accesses as many channels as are needed for a given application. The GTR bits and the MOST Processor peripheral routing are synchronized to the Source Port SCK pin. Therefore, for the GTR flags and peripheral routing to operate properly, the Source Ports must be disabled, SCK configured as an output, or SCK and FSY must be configured as inputs with the proper external clocks applied.

19h DR MOST Routing Port Receive Data (read only) DSPs

Bit	Label	Description	Default
17..2	DR	MOST Routing Port receive data	0000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-17: DR Register

19h DX MOST Routing Port Transmit Data (write only) DSPs

Bit	Label	Description	Default
17..2	DX	MOST Routing Port transmit data	0000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-18: DX Register

A second set of MOST Routing Ports exist which can double the bandwidth between the DSP and the Routing bus; however, other Routing bus ports must be disabled to enable this port. This Routing port is enabled through the MOST Processor registers by first disabling the external Source Ports, the Source Converters, or the other DSP's routing port.

1Bh DR1 Second MOST Routing Port Receive Data (read only) DSPs

Bit	Label	Description	Default
17..2	DR1	Second MOST Routing Port receive data. This routing port is disabled by default and must be enabled through the MOST Processor before using.	0000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-19: DR1 Register

1Bh DX1 Second MOST Routing Port Transmit Data (write only) DSPs

Bit	Label	Description	Default
17..2	DX1	Second MOST Routing Port transmit data. This routing port is disabled by default and must be enabled through the MOST Processor before using.	0000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-20: DX1 Register

4.3.2 Host Controller COM Port

The COM Port, from the DSP to the Control bus, consists of a status register (DCS) and a data register (DCD). Bits in the status register can interrupt the DSP or allow the DSP to interrupt the Host Controller for service (**INT**). The data register is 16 bits wide and communicates data bi-directionally between the DSP bus and the Control bus. For additional information on the DSP COM ports, see the Host Controller *Interprocessor Communication* Section.

The **STR**, **RD**, and **WR** flags are read-only by the Controller and DSP. They provide handshaking between the Controller and the DSP during data transfers. They are set when the Controller accesses the data register and are cleared when the DSP accesses the data register. A low-to-high transition of any of these flags can generate the DSP COM port interrupt. Since the status flags must cross clock boundaries between the DSP and the Host Controller, a one-cycle delay exists between when the DSP causes a status bit to change and when the change can be read from the status register. As an example, if the DSP is writing the data register to clear the **DCS.RD** bit, a `NOP` should be inserted before reading DCS to compensate for the clock-boundary delay. To illustrate the example:

```
                  // assumes DCS.RD is set due to Host Controller reading COM register.
dcd = acch;         // DSP clears the DCS.RD bit (after one-cycle delay) by writing DCD
nop;                // Compensates for the one-cycle delay
acch = dcs;         // Read the correct status bits. RD should now be clear.
```

The DSP alerts the Host Controller that it needs servicing by setting the **DCS.INT.** A low-to-high transition of **INT** can generate the DSP COM Port interrupt in the Controller, if the **IER.IEDSPn** bit is set. For the DSP, when **DCS.CIM** is set, the COM Port interrupt from the Host Controller is masked, which allows the COM Port to be used in a polling fashion while the DSP FIFO port uses the shared interrupt. When **DCS.CIM** is clear, the COM port can generate an interrupt to the DSP, when service is required.

17h DCS COM Port Status DSPs

Bit	Label	Description	Default
17..5	rsvd	Reserved, Write to 0	000h, 00
4	CIM	COM Port Interrupt Mask	0
3	INT	Interrupt flag to Host Controller	0
2	STR	Start transfer (read-only)	0
1	WR	Write data available (read-only)	0
0	RD	Read data request (read-only)	0

Table 4-21: DCS Register

18H DCD COM Port Data DSPs

Bit	Label	Description	Default
17..2	DCD	Bit 17 is MSB: Data from the Host Controller (read only) Data to the Host Controller (write only)	0000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-22: DCD Register

4.3.3 Host Controller Debug COM Port

The Debug COM port is identical to the regular COM port between DSPs and the Host Controller, except that the DSPs do not have the capability to interrupt the Host Controller through this port. The data register can be read or written by the Host Controller using DD0CF for the first or last transfer and DD0CM for middle word transfer to DSP0. Similarly, DD1CF and DD1CM are for DSP1. The Host Controller can read the status register using DD0CS and DD1CS for DSP0 and DSP1 respectively. Similar to the regular COM port, the status flags must cross clock boundaries between the DSP and the Host Controller, a one-cycle delay exists between when the DSP causes a status bit to change and when the change can be read from the status register. As an example, if the DSP is writing the DDCD data register to clear the **DDCS.RD** bit, a `NOP` should be inserted before reading DDCS to compensate for the clock-boundary delay.

10h DDCS Debug COM Port Status DSPs

Bit	Label	Description	Default
17..3	rsvd	Reserved, Write to 0	000h, 000
2	STR	Start transfer (read-only)	0
1	WR	Write data available (read-only)	0
0	RD	read data request (read-only)	0

Table 4-23: DDCS Register

11H DDCD Debug COM Port Data DSPs

Bit	Label	Description	Default
17..2	DCD	Bit 17 is MSB: Data from the Host Controller (read only) Data to the Host Controller (write only)	0000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-24: DDCD Register

Debug functions in the DSP are supported through two methods. One method is to insert a `TRAP` instruction in the DSP Program memory, and the other is when the Host Controller interrupts the DSP using a non-maskable highest priority DSP debug interrupt, **DDnCS.TRINT**. In the first case the `TRAP` is typically put in the DSP Program memory by the Host Controller, when the external debugger places a break point at that location in the DSP program. The `TRAP` instruction could also be a part of the DSP program in this case. Typically the first time the user puts a break point in the DSP program, the Host Controller interrupts DSP using the debug interrupt and puts a `TRAP` instruction at that location.

4.3.4 Inter-DSP FIFO Port

The two DSPs have a communication port between them that allows the direct sharing of data without going through the Routing bus, utilizing an 18-bit by 16-deep FIFO between the two DSPs. Each DSP has its own status and data registers mapped to its I/O space. One of two status bits will be set depending on the register written to by the DSP, with the option of generating an interrupt in the other DSP when the FIFO is full. The two status bits perform the same function, but allow the receiving DSP to discern between two different message types (defined by the user). Both DSPs have the following registers to initiate and control a data transfer with the other DSP.

12h DFLS FIFO Port Local Status register DSPs

Bit	Label	Description	Default
17..8	rsvd	Reserved, Write to 0	00h, 00
7..4	DEPTH[3:0]	Local FIFO port Depth (FIFO depth = DEPTH[3:0] + 1)	0000
3	IMSK	Local FIFO port Interrupt Mask	0
2	STR	Local FIFO port start status	0
1	WR	Local FIFO port write status	0
0	rsvd	Reserved, Write to 0	0

Table 4-25: DFLS Register

DEPTH[3:0] Sets the depth of the FIFO for sending data to the other DSP. When the sum of this depth and the other DSP depth, set by **DFFS.DEPTH[3:0]**, is greater than 15, software handshaking is required to prevent overwriting of data. Valid values are 0 to 15 (depth of 1 to 16 words).

IMSK When **IMSK** is clear and **IER.IECP** is set, an interrupt is generated when either **DFFS.STR** or **DFFS.WR** go high, which indicates that the other DSP has filled the FIFO to a depth of **DFFS.DEPTH[3:0]**. The interrupt priority is set by **IER.CPHI** and the interrupt vector is 048h.

STR Set when the last word is written to the DFSD register, filling the FIFO to the **DEPTH[3:0]** setting. When **STR** is set, an interrupt is generated on the other DSP, if its interrupt mask is clear (**DFFS.IMSK** clear) and **IER.IECP** is set. **STR** is cleared when the other DSP reads the last word out of the FIFO (reading its DFRD register the number of times specified by its **DFFS.DEPTH[3:0]**).

WR Set when the last word is written to the DFWD register, filling the FIFO to the **DEPTH[3:0]** setting. When **WR** is set, an interrupt is generated on the other DSP, if its interrupt mask is clear (**DFFS.IMSK** clear) and **IER.IECP** is set. **WR** is cleared when the other DSP reads the last word out of the FIFO (reading its DFRD register the number of times specified by its **DFFS.DEPTH[3:0]**).

15h DFFS FIFO Port Far Status register (read only) DSPs

Bit	Label	Description	Default
17..8	rsvd	Reserved, Write to 0	00h, 00
7..4	DEPTH[3:0]	Far end FIFO port Depth	0000
3	IMSK	Far end FIFO port Interrupt Mask	0
2	STR	Far end FIFO port start status	0
1	WR	Far end FIFO port write status	0
0	rsvd	Reserved, Write to 0	0

Table 4-26: DFFS Register

DEPTH[3:0] The depth of the FIFO set by the other DSP. This depth defines how many reads are required to empty the FIFO, thereby clearing **DFFS.WR** or **DFFS.STR** which caused the interrupt (if enabled). Since the DSPs are running off independent clocks, these bits are not reliable if the other DSP is in the process of writing them.

IMSK Indicates whether the other DSP will receive interrupts (**IMSK** clear, assuming **IER.IECP** is set) when the FIFO is filled.

STR Set when the other DSP writes the last word to its DFSD register, filling the FIFO to the **DFFS.DEPTH[3:0]** setting. When **STR** is set, an interrupt is generated on this DSP, if its interrupt mask is clear (**DFLS.IMSK** clear) and **IER.IECP** is set. **STR** is cleared when the DFRD register is read the number of times specified by **DFFS.DEPTH[3:0]**, thereby emptying the FIFO.

WR Set when the other DSP writes the last word to its DFWD register, filling the FIFO to the **DFFS.DEPTH[3:0]** setting. When **WR** is set, an interrupt is generated on this DSP, if its interrupt mask is clear (**DFLS.IMSK** clear) and **IER.IECP** is set. **WR** is cleared when the DFRD register is read the number of times specified by **DFFS.DEPTH[3:0]**, thereby emptying the FIFO.

13h DFSD FIFO Port Start Data register (write only) DSPs

Bit	Label	Description	Default
17..0	DFSD[17:0]	FIFO Port Start Data. Writing to this register loads a word into the FIFO. If the write fills the FIFO to the depth indicated by the **DFLS.DEPTH[3:0]** bits, the **DFLS.STR** bit is set, and can generate an interrupt to the other DSP if its FIFO Port interrupt is enabled (**DFFS.IMSK** clear and **IER.IECP** set).	00h, 00

Table 4-27: DFSD Register

14h DFWD FIFO Port Write Data register (write only) DSPs

Bit	Label	Description	Default
17..0	DFWD[17:0]	FIFO Port Write Data. Writing to this register loads a word into the FIFO. If the write fills the FIFO to the depth indicated by the **DFLS.DEPTH[3:0]** bits, the **DFLS.WR** bit is set, and can generate an interrupt to the other DSP if its FIFO Port interrupt is enabled (**DFFS.IMSK** clear and **IER.IECP** set).	00h, 00

Table 4-28: DFWD Register

14h DFRD FIFO Port Read Data register (read only) DSPs

Bit	Label	Description	Default
17..0	DFRD[17:0]	FIFO Port Read Data. Reading this register removes a word from the FIFO. When the last word in the FIFO is read (based on **DFFS.DEPTH[3:0]**), the **DFFS.WR** or **DFFS.STR** bit (which ever was high) will then be cleared.	00h, 00

Table 4-29: DFWD Register

4.4 Asynchronous Source Ports

Each DSP has two Asynchronous Source Ports: 0 and 1. These ports differ from the Source Ports directly connected to the routing bus, in that the DSP Source Ports are asynchronous to the Network (or global timer **GTR.FS1**). All four pins can be configured as a block of GPIO or as a Source Port. The Source Ports communicate data externally to the OS8805, as opposed to the Routing Port which communicates data internally. DSP0 communicates through Async. Source Port A and DSP1 through Source Port B. For DSP0, Source Port A0 shares pins with **GPA[3:0]** and Source Port A1 shares pins with **GPB[3:0]**. For DSP1, Source Port B0 shares pins with **GPC[3:0]** and Source Port B1 shares pins with **GPD[3:0]**. These GPIO pins can also be controlled via the Host Controller. Enabling the Source Port (**DSnC.BPI[6:0]** > 0) has precedence over any GPIO function.

Each Source Port consists of a transmit pin **SXxn** which transmits data out of the OS8805, a receive pin **SRxn** which receives data from an external source, a bit-clock pin **SCKxn**, and a framing pin **FSYxn**. For DSP0, the **x** stands for **A** (Source Port A pins) and for DSP1, the **x** stands for **B** (Source Port B pins). Since each DSP has two Source Ports, the **n** stands for 0 or 1. Table 4-30 lists the pins for each Source Port.

Function	DSP0				DSP1			
	Async. Source Port A0		Async. Source Port A1		Async. Source Port B0		Async. Source Port B1	
	SP	GPIO	SP	GPIO	SP	GPIO	SP	GPIO
Bit Clock	SCKA0	GPA0	SCKA1	GPB0	SCKB0	GPC0	SCKB1	GPD0
Framing Clock	FSYA0	GPA1	FSYA1	GPB1	FSYB0	GPC1	FSYB1	GPD1
Transmit Data	SXA0	GPA2	SXA1	GPB2	SXB0	GPC2	SXB1	GPD2
Receive Data	SRA0	GPA3	SRA1	GPB3	SRB0	GPC3	SRB1	GPD3

Table 4-30: DSP Async. Source Port Pins

The framing and bit clocks can be inputs or outputs. Each DSP has two timers that may be connected to the Source Ports, where one timer outputs the bit clock **SCKxn** and the other outputs the framing clock **FSYxn**. The second Source Port's clocks can be chained to the first, or input externally. If the timers are not used to output clocks, they can capture the counter time when **FSYxn** changes.

4.4.1 Async. Source Port Timer

The timer on each DSP consists of two 15-bit programmable dividers and a control register. The timer control register has 3 bits, which enable the timer divider outputs and control the divide-by-2 multiplexer. If the S/PDIF format is enabled, DDIV0 and DDIV1 must be set properly for the S/PDIF format regardless of whether the **SCKxn** and **FSYxn** pins are inputs or outputs.

3Bh DDIV0 DSP Divider 0 register DSPs

Bit	Label	Description	Default
17	rsvd	Reserved, Write to 0	0
16..2	DDIV0[14:0]	Divider value for Timer 0. Clocked from a 3072xFs signal.	000, 000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-31: DDIV0 Register

DDIV0[14:0] Divider value for DSP Timer 0. The Timer 0 counter counts up at a 3072xFs rate, and when the counter reaches the **DDIV0[14:0]** value, the counter generates a pulse and resets the counter. To achieve a clock that is a certain ratio to Fs, $\mathbf{DDIV0[14:0]} = \frac{3072}{\text{ratio}} - 1$. To achieve a 50 % duty cycle, the **DDIV0[14:0]** value should be set to half the required ratio (twice the frequency) minus one, and then set **DTC.T0D2E**, which will divide the result by two. Timer 0 can be output as the Async Source Port bit clock **SCKxn** by setting **DSnC.CKOE**. When used as **SCKxn** output, the valid values (assuming **DTC.T0D2E** is set) are listed in Table 4-36. When Source Port 0 is set for S/PDIF, **DDIV0[14:0]** must be set to twice the S/PDIF bit clock frequency if **DTC.T0D2E** is clear, or four times the bit-clock frequency if **DTC.T0D2E** is set.

3Ch DDIV1 DSP Divider 1 register DSPs

Bit	Label	Description	Default
17	rsvd	Reserved, Write to 0	0
16..2	DDIV1[14:0]	Divider value for Timer 1. Clocked from the output of Timer 0.	000, 000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-32: DDIV1 Register

DDIV1[14:0] Divider value for DSP Timer 1. The Timer 1 counter counts up at a rate set by the output of Timer 0, and when the counter reaches the **DDIV1[14:0]** value, the counter generates a pulse and resets the counter. To achieve a clock that is a certain ratio to Fs,

$\mathbf{DDIV1[14:0]} = \frac{\text{Timer 0 output}}{\text{ratio}} - 1$. To achieve a 50 % duty cycle, the **DDIV1[14:0]** value should be set to half the required ratio (twice the frequency) minus one, and then set **DTC.T1D2E**, which will divide the result by two. Timer 1 can be output as the Async Source Port bit clock **FSYxn** by setting **DSnC.CKOE**. When used as **FSYxn** output, the valid values (assuming **DTC.T1D2E** is set) are listed in Table 4-36.

3Dh DCAP0 DSP Capture 0 register DSPs

Bit	Label	Description	Default
17..2	DCAP0[15:0]	Capture value. Captures the timer 0 value when one of four events occurs, where the selection is made via the **DTC.CSEL[1:0]** bits.	0000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-33: DCAP0 Register

3Eh DCAP1 DSP Capture 1 register DSPs

Bit	Label	Description	Default
17..2	DCAP1[15:0]	Capture value. Captures the timer 1 value when one of four events occurs, where the selection is made via the **DTC.CSEL[1:0]** bits.	0000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-34: DCAP1 Register

3Fh DTC DSP Timer Control register DSPs

Bit	Label	Description	Default
17	SP1TXI	Source Port 1 Transmit Interrupt Status	0
16	SP0TXI	Source Port 0 Transmit Interrupt Status	0
15	SP1RXI	Source Port 1 Receive Interrupt Status	0
14	SP0RXI	Source Port 0 Receive Interrupt Status	0
13..11	rsvd	Reserved. Write to 0	000
10	T0RSD	Timer 0 S/PDIF Data Transition Reset Enable	0
9, 8	CSEL[1:0]	Capture Select	00
7	T1CAP	Trigger Capture for Timer 1	0
6	T0CAP	Trigger Capture for Timer 0	0
5	T1D2E	Timer 1 Divide by 2 enable	0
4	T0D2E	Timer 0 Divide by 2 enable	0
3	T1EN	Timer 1 Enable	0
2	T0EN	Timer 0 Enable	0
1,0	rsvd	Reserved, Write to 0	00

Table 4-35: DTC Register

SP1TXI Source Port x1 Transmit Interrupt Status. When set, this bit indicates that Source Port x1's Transmit registers DS1X0 - DS1X3 are empty and ready for loading. The rising edge of **SP1TXI** generates an interrupt to the GPIO interrupt vector when **IER.IEGP** is set. **SP1TXI** is cleared by writing any one of the DS1X0 - DS1X3 registers or directly clearing **SP1TXI**.

SP0TXI Source Port x0 Transmit Interrupt Status. When set, this bit indicates that Source Port x0's Transmit registers DS0X0 - DS0X3 are empty and ready for loading. The rising edge of **SP0TXI** generates an interrupt to the GPIO interrupt vector when **IER.IEGP** is set. **SP0TXI** is cleared by writing any one of the DS0X0 - DS0X3 registers or directly clearing **SP0TXI**.

SP1RXI Source Port x1 Receive Interrupt Status. When set, this bit indicates that Source Port x1's Receive registers DS1R0 - DS1R3 are full and ready to be read. The rising edge of **SP1RXI** generates an interrupt to the GPIO interrupt vector when **IER.IEGP** is set. **SP1RXI** is cleared by reading any one of the DS1R0 - DS1R3 registers or directly clearing **SP1RXI**.

SP0RXI Source Port x0 Receive Interrupt Status. When set, this bit indicates that Source Port x0's Receive registers DS0R0 - DS0R3 are full and ready to be read. The rising edge of **SP0RXI** generates an interrupt to the GPIO interrupt vector when **IER.IEGP** is set. **SP0RXI** is cleared by reading any one of the DS0R0 - DS0R3 registers or directly clearing **SP0RXI**.

T0RSD Timer 0 S/PDIF Data Transition Reset enable.Must be set when in S/PDIF mode.

CSEL[1:0] Capture Select. Selects the trigger mechanism for loading the DCAP0 and DCAP1 registers from their respective timers.
00 - Setting the respective trigger bit in DTC: **T0CAP** for DCAP0, and **T1CAP** for DCAP1.
01 - Start of a Async. Source Port Frame on **FSYx0**.
10 - Start of a Async. Source Port Frame on **FSYx1**.
11 - Rising edge of **GTR.FS1**.

T1CAP Capture 1 trigger. When **CSEL[1:0]** = 00, setting this bit causes the Timer 1 value to be loaded into DCAP1.

T0CAP Capture 0 trigger. When **CSEL[1:0]** = 00, setting this bit causes the Timer 0 value to be loaded into DCAP0.

T1D2E Timer 1 Divide-by-2 Enable. When set, the output generated by DDIV1 is divided by 2 to generate a 50 % duty cycle clock.

T0D2E Timer 0 Divide-by-2 Enable. When set, the output generated by DDIV0 is divided by 2 to generate a 50 % duty cycle clock.

T1EN Timer 1 Enable. When set, Timer 1 is enabled. When clear, Timer 1 is powered down.

T0EN Timer 0 Enable. When set, Timer 0 is enabled. When clear, Timer 0 is powered down.

4.4.2 Source Port Registers

The Asynchronous Source Port is a generic programmable Source Port which supports I^{2}S and S/PDIF operation. One instantiation of the Source Port block provide two independent Source Ports, port 0 and port 1. Within the Source Port block, there are two programmable timers which can generate **SCKxn** and **FSYxn** signals. Therefore, if both ports are operating in output mode, port 0 and port 1 can be of the same frequency and synchronous to each other. Also, an S/PDIF block is attached to port 0. One of the timers (timer 0) can generate the 50% duty cycle of **SCKxn**, the other timer (timer 1) can generate the **FSYxn**.

The Source Ports are capable of generating four independent interrupts, port 0 receive interrupt, port 1 receive interrupt, port 0 transmit interrupt and port 1 transmit interrupt. Depending on the register setting, all 4 interrupts can be totally independent of each other. However, depending on the register setting and configuration, port 0 and port 1 can be serviced by the same receive and transmit interrupt. In some configurations, only one interrupt is needed when port 0 and port 1, transmit and receive are synchronized.

In addition, **DTC.SP0TXI**, **DTC.SP1TXI**, **DTC.SP0RXI**, and **DTC.SP1RXI** indicate the interrupt status. If interrupts are not possible, then these four bits can be polled through software. The **DTC.SP0TXI** and **DTC.SP1TXI** bits go high when their corresponding DS0Xn or DS1Xn registers are ready to be loaded with the next transmit data. The **DTC.SP0TXI** and **DTC.SP1TXI** bits can be cleared by writing to any of the corresponding DS0Xn or DS1Xn registers. The **DTC.SP0RXI** and **DTC.SP1RXI** bits go high when their corresponding DS0Rn or DS1Rn registers are ready to be unloaded. The **DTC.SP0RXI** and **DTC.SP1RXI** bits can be cleared by reading from any of the corresponding DS0Rn or DS1Rn registers.

If Asynchronous Source Port interrupts are enabled, transmit and receive interrupts **must both** be cleared before existing the ISR; otherwise, new interrupts will be blocked from occurring. Since all four interrupts are OR'd, any **one** set when exiting the ISR will block a new rising edge from occurring, thereby blocking further interrupts. Software must verify that all four are simultaneously zero sometime during servicing, or new interrupts might not be generated.

The **DSnC.DEL**, **DSnC.FPOL**, and **DSnC.EDG** bit combinations allow the support of different formats on the Source Ports.

For the case where **SCKxn** and **FSYxn** are outputs, the following tables lists the valid combinations of timer and Source Port settings to get the desired number of bits per Fs period. When set for more than one interrupt per Fs, the GPIO input bit attached to the particular **FSYxn** pin indicates when **FSYxn** changes state. The DSP can use this data to determine which of the interrupts is occurring during an Fs period. For DSP0, **FSYA0** is connected to **GPA1** and, for DSP1, **FSYB0** is connected to **GPC1**.

SCLKxn		FSYxn		Interrupts		Registers Used	
Bits/Fs	**DDIV0 ***	**Periods per Fs**	**DDIV1 ***	**Interrupts per Fs**	**DSnC. BPI**	**TX**	**RX**
32	002Fh	1	000Fh	1	1Fh	DSnX3-DSnX2	DSnR1-DSnR0
64	0017h	1	001Fh	1	3Fh	DSnX3-DSnX0	DSnR3-DSnR0
128	000Bh	1	003Fh	2	3Fh	DSnX3-DSnX0	DSnR3-DSnR0
256	0005h	1	007Fh	4	3Fh	DSnX3-DSnX0	DSnR3-DSnR0
512	0002h	1	00FFh	8	3Fh	DSnX3-DSnX0	DSnR3-DSnR0
128 †	000Bh	1	003Fh	1	7Fh	DS0X3-DS0X0 + DS1X3-DS1X0	DS0R3-DS0R0 + DS1R3-DS1R0
256 †	0005h	1	007Fh	2	7Fh	DS0X3-DS0X0 + DS1X3-DS1X0	DS0R3-DS0R0 + DS1R3-DS1R0
512 †	0002h	1	00FFh	4	7Fh	DS0X3-DS0X0 + DS1X3-DS1X0	DS0R3-DS0R0 + DS1R3-DS1R0

* For **SCLKxn** assumes **DTC.T0D2E** set, and for **FSYxn** assumes **DTC.T1D2E** set. To load the DDIVn registers, the values listed must be shifted up by four as the register bits are MSB-aligned in the DDIVn registers.

† 128-bit mode, where Source Port x1 registers are cascaded/used; therefore, Source Port x1 is not available.

Table 4-36: Valid Output Clocking Combinations for Async. Source Ports

Each of the two Source Ports are independent from each other. Source Port x0 and Source Port x1 have their respective **DSnC.DEL**, **DSnC.FPOL**, **DSnC.EDG**, **DSnC.CKOE**, and **DSnC.BPI[6:0]** controls. However, if both Source Ports are configured to have **SCKxn** and **FSYxn** as outputs, the **SCKxn** and **FSYxn** frequency and the **DSnC.BPI[6:0]** bits need to be the set the same for both ports, since there is only one set of timers. When both sets of Source Port clocks are configured as output, only one interrupt is needed since transmit and receive are synchronized. In addition, **DSnC.CHAIN** can chain the **SCKxn** and **FSYxn** inputs together for both Source Ports, to minimize the number of interrupt. When **DS1C.CHAIN** is set, the **SCKx0** and **FSYx0** signals are used for both Source Ports.

The maximum number of bits per interrupt is 128. When the **DSnC.BPI[6:0]** is set to 7Fh, Source Port x0 and Source Port x1 are cascaded. When configured to output **SCKxn** and **FSYxn**, the timers outputs are used for both ports. When the Source Port clocking are configured to be inputs, **SCKxn** and **FSYxn** control each port (unless chained). All four interrupts occur simultaneously when in 128-bit mode (Source Ports cascaded). For received data, each must be cleared by reading at least one register in each of the received data blocks (DS0R3-DS0R0 and DS1R3-DS1R0 blocks). Likewise, for transmitted data, each must be cleared by writing at least one register in each of the received data blocks (DS0X3-DS0X0 and DS1X3-DS1X0 blocks). In addition DS0xn block contains the most significant set of data for both transmit and receive directions

The Source Port data outputs are powered up disabled to reduce EMI. The **DSnC.SDOEN** bit unmutes the SX outputs.

The two 64-bit output shift registers accept data from eight DSP IO registers, DS0X0-3 and DS1X0-3. These registers are loaded during the interrupt mentioned above and parallel loaded into the output shift registers at the same time the input shift registers are loaded into their parallel registers. Data is then shifted out the two data output pins as data is shifted in from the two data input pins.

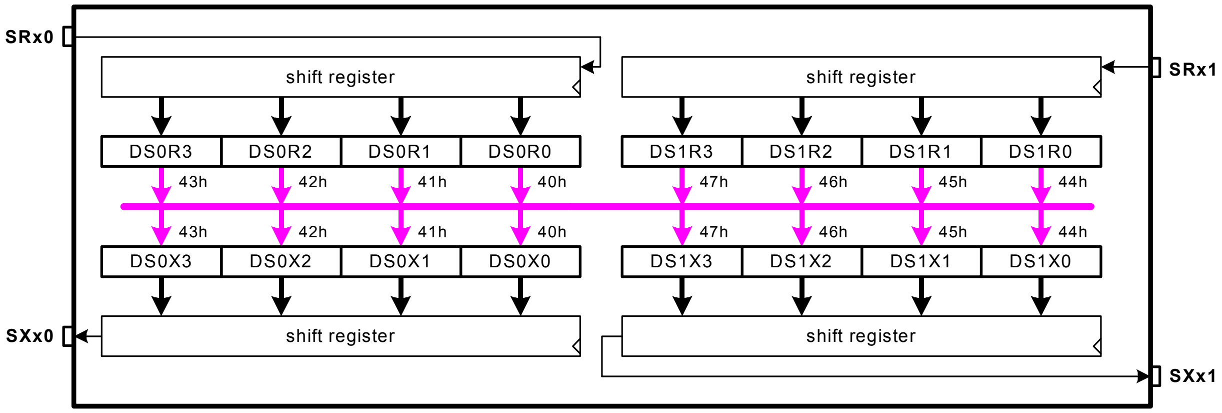


Figure 4-7: DSP ASP 64-bit Mode

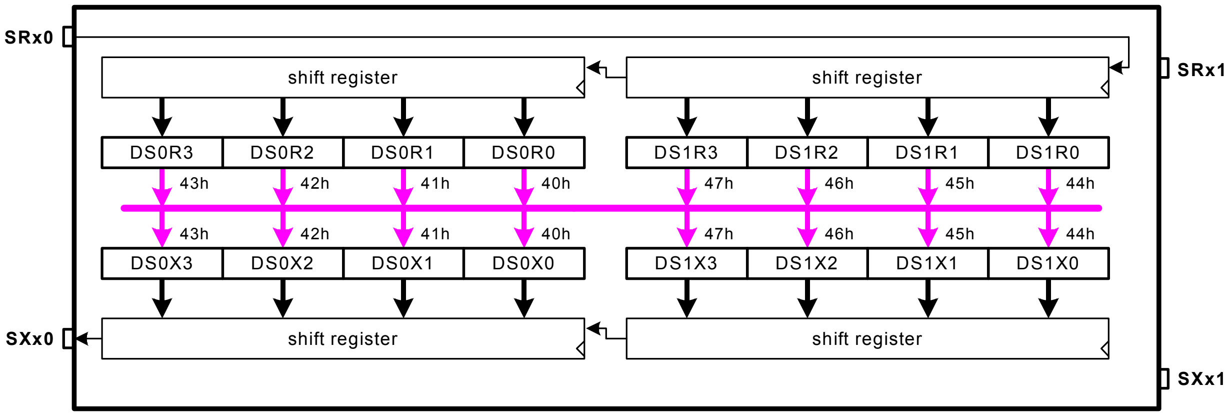


Figure 4-8: DSP ASP 128-bit Mode

In the OS8805, the **DTC.SP0TXI**, **DTC.SP1TXI**, **DTC.SP0RXI**, and **DTC.SP1RXI** bits are OR'd with the GPIO interrupt. Therefore, when a DSP GPIO interrupt occurs, software must examine the EGPIO and the DTC registers to determine the source of the interrupt. If a DTC interrupt exists, at the end of the interrupt routine **BOTH** the transmit and receive interrupts should be checked and cleared, as ***either one*** set could block further interrupts from occurring.

40h DS0R0 DSP Source Port 0, Receive word 0 (read only) DSPs

Bit	Label	Description	Default
17..2	DS0R[15:0]	DSP Source Port 0: In 32-bit mode, second of two words in, In 64- and 128-bit modes, fourth of four/eight words in, from pin **SRA0** for DSP0 and pin **SRB0** for DSP1.	0000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-37: DS0R0 Register

41h DS0R1 DSP Source Port 0, Receive word 1 (read only) DSPs

Bit	Label	Description	Default
17..2	DS0R[31:16]	DSP Source Port 0: In 32-bit mode, first of two words in, In 64- and 128-bit modes, third of four/eight words in, from pin **SRA0** for DSP0 and pin **SRB0** for DSP1.	0000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-38: DS0R1 Register

42h DS0R2 DSP Source Port 0, Receive word 2 (read only) DSPs

Bit	Label	Description	Default
17..2	DS0R[47:32]	DSP Source Port 0: (not used in 32-bit mode) In 64- and 128-bit modes, second of four/eight words in, from pin **SRA0** for DSP0 and pin **SRB0** for DSP1	0000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-39: DS0R2 Register

43h DS0R3 DSP Source Port 0, Receive word 3 (read only) DSPs

Bit	Label	Description	Default
17..2	DS0R[63:48]	DSP Source Port 0: (not used in 32-bit mode) In 64- and 128-bit modes, first of four/eight words in, from pin **SRA0** for DSP0 and pin **SRB0** for DSP1	0000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-40: DS0R3 Register

44h DS1R0 DSP Source Port 1, Receive word 0 (read only) DSPs

Bit	Label	Description	Default
17..2	DS1R[15:0]	DSP Source Port 1: In 32-bit mode, second of two words in, In 64-bit mode, fourth of four words in, In 128-bit mode, eighth of eight words in, from pin **SRA1** for DSP0 and pin **SRB1** for DSP1.	0000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-41: DS1R0 Register

45h DS1R1 DSP Source Port 1, Receive word 1 (read only) DSPs

Bit	Label	Description	Default
17..2	DS1R[31:16]	DSP Source Port 1: In 32-bit mode, first of two words in, In 64-bit mode, third of four words in, In 128-bit mode, seventh of eight words in, from pin **SRA1** for DSP0 and pin **SRB1** for DSP1	0000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-42: DS1R1 Register

46h DS1R2 DSP Source Port 1, Receive word 2 (read only) DSPs

Bit	Label	Description	Default
17..2	DS1R[47:32]	DSP Source Port 1: (not used in 32-bit mode) In 64-bit mode, second of four words in, In 128-bit mode, sixth of eight words in, from pin **SRA1** for DSP0 and pin **SRB1** for DSP1	0000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-43: DS1R2 Register

47h DS1R3 DSP Source Port 1, Receive word 3 (read only) DSPs

Bit	Label	Description	Default
17..2	DS1R[63:48]	DSP Source Port 1: (not used in 32-bit mode) In 64-bit mode, first of four words in, In 128-bit mode, fifth of eight words in, from pin **SRA1** for DSP0 and pin **SRB1** for DSP1	0000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-44: DS1R3 Register

40h DS0X0 DSP Source Port 0, Transmit word 0 (write only) DSPs

Bit	Label	Description	Default
17..2	DS0X[15:0]	DSP Source Port 0: (not used in 32-bit mode) In 64- and 128-bit modes, fourth of four/eight words out, from pin **SXA0** for DSP0 and pin **SXB0** for DSP1	0000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-45: DS0X0 Register

41h DS0X1 DSP Source Port 0, Transmit word 1 (write only) DSPs

Bit	Label	Description	Default
17..2	DS0X[31:16]	DSP Source Port 0: (not used in 32-bit mode) In 64- and 128-bit modes, third of four/eight words out, from pin **SXA0** for DSP0 and pin **SXB0** for DSP1	0000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-46: DS0X1 Register

42h DS0X2 DSP Source Port 0, Transmit word 2 (write only) DSPs

Bit	Label	Description	Default
17..2	DS0X[47:32]	DSP Source Port 0: In 32-bit mode, second of two words out In 64- and 128-bit modes, second of four/eight words out, pin **SXA0** for DSP0 and pin **SXB0** for DSP1	0000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-47: DS0X2 Register

43h DS0X3 DSP Source Port 0, Transmit word 3 (write only) DSPs

Bit	Label	Description	Default
17..2	DS0X[63:48]	DSP Source Port 0: In 32-, 64-, and 128-bit modes, first of two/four/eight words out from pin **SXA0** for DSP0 and pin **SXB0** for DSP1.	0000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-48: DS0X3 Register

44h DS1X0 DSP Source Port 1, Transmit word 0 (write only) DSPs

Bit	Label	Description	Default
17..2	DS1X[15:0]	DSP Source Port 1: (not used in 32-bit mode) In 64-bit mode, fourth of four words out, In 128-bit mode, eighth of eight words out, from pin **SXA1** for DSP0 and pin **SXB1** for DSP1	0000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-49: DS1X0 Register

45h DS1X1 DSP Source Port 1, Transmit word 1 (write only) DSPs

Bit	Label	Description	Default
17..2	DS1X[31:16]	DSP Source Port 1: (not used in 32-bit mode) In 64-bit mode, third of four words out, In 128-bit mode, seventh of eight words out, from pin **SXA1** for DSP0 and pin **SXB1** for DSP1	0000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-50: DS1X1 Register

46h DS1X2 DSP Source Port 1, Transmit word 2 (write only) DSPs

Bit	Label	Description	Default
17..2	DS1X[47:32]	DSP Source Port 1: In 32-bit mode, second of two words out In 64-bit mode, second of four words out, In 128-bit mode, sixth of eight words out, from pin **SXA1** for DSP0 and pin **SXB1** for DSP1	0000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-51: DS1X2 Register

47h DS1X3 DSP Source Port 1, Transmit word 3 (write only) DSPs

Bit	Label	Description	Default
17..2	DS1X[63:48]	DSP Source Port 1: In 32- and 64-bit modes, first of two/four words out, In 128-bit mode, fifth of eight words out, from pin **SXA1** for DSP0 and pin **SXB1** for DSP1.	0000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-52: DS1X3 Register

48h DS0C DSP Source Port 0 Control DSPs

Bit	Label	Description	Default
17..11	BPI[6:0]	Number of bits per interrupt. Also enables/disables Source Port x0.	0000000
10	SPLK	S/PDIF Lock status (read only)	0
9	EDG	Active edge of **SCKx0**. 0 = valid on falling and changing on rising edge.	0
8	DEL	Delay from **FSYx0** edge.	0
7	FPOL	Polarity of **FSYx0**. When 0, the start of frame is **FSYx0** low.	0
6	rsvd	Reserved. Write to 0	0
5	rsvd	Reserved. Write to 0	0
4	SPEN	S/PDIF mode enable	0
3	CKOE	Clock Output enable. When set, **FSYx0**/**SCKx0** are outputs.	0
2	SDOEN	Source Data Output Enable. When clear **SXx0** is always 0.	0
1,0	rsvd	Reserved. Write to 0	00

Table 4-53: DS0C Register

BPI[6:0] Number of bits per interrupt. The settings for **SCLKx0** and **FSYx0** are dependent on the **BPI[6:0]** settings. Valid **BPI[6:0]** values are:

00h - Source Port x0 is disabled. The pins can be used for GPIO. For DSP0 **BPI[6:0]** set to zero enables GPIO pins **GPA[3:0]**. For DSP1 **BPI[6:0]** set to zero enables pins **GPC[3:0]**.

1Fh - Source Port x0 enabled, with 32 bits per interrupt. Registers DS0X3-DS0X2 and DS0R1-DS0R0 are used. Not allowed when in S/PDIF mode (**SPEN** enabled).

3Fh - Source Port x0 enabled, with 64 bits per interrupt. Registers DS0X3-DS0X0 and DS0R3-DS0R0 are used.

7Fh - Source Port x0 enabled, with 128 bits per interrupt (128-bit mode). **DS1C.BPI[6:0]** must equal 7Fh as Source Port x1's registers are cascaded to get all 128 bits per interrupt. For transmitting, registers DS0X3- DS0X0 contain the most significant 64 bits and DS1X3-DS1X0 contain the least significant 64 bits. For receiving, registers DS0R3-DS0R0 contain the most significant and DS1R3-DS1R0 contain the least significant 64 bits.

SPLK S/PDIF Lock status. When Source Port x0 is configured for S/PDIF format (**SPEN** set), **SPLK** is set, indicating *in lock,* when three consecutive left preambles are correctly received. **SPLK** is cleared when two consecutive left preambles are not received correctly.

EDG Active edge of **SCKA0** for DSP0 or **SCKB0** for DSP1. **EDG** clear is data valid on the falling edge and changing on the rising edge. **EDG** set is data valid on the rising edge and changing on the falling edge.

DEL Delay from **FSYx0** edge. When set, the MSB starts one **SCKx0** clock period after **FSYx0** edge (Philips I^{2}S style). When clear, the MSB starts at the **FSYx0** edge.

FPOL Polarity of **FSYx0**. When 0, the start of frame is **FSYx0** going low. When **FPOL** is set, the start of a frame is **FSYx0** going high.

SPEN S/PDIF enabled. When set, Source Port x0 is configured to input and output S/PDIF data. **BPI[6:0]** must be set to 64 or 128 bits per interrupt. When SPEN is set, the DSP timers are used for transmitting the S/PDIF signal; therefore, Source Port x1 is internally forced to use the same timing (**DS1C.CHAIN** must be clear), or must use externally derived timing (**DS1C.CKOE** clear).

CKOE Clock Output enable. When set, the DSP timers are output on the **FSYx0**/**SCKx0** pins, where DSP timer 0 goes to the **SCKx0** pin and timer 1 goes to the **FSYx0** pin. If the **CHAIN** bit is clear, then **SRx0**/**SXx0** are clocked using the DSP timers. If **CHAIN** is set when **CKOE** is set, **SRx0**/**SXx0** are clocked from **SCKx1**/**FSYx1** and **SCKx0**/**FSYx0** just output the DSP timer clocks. When **CKOE** is clear, **FSYx0**/**SCKx0** are inputs.

SDOEN Source Data Output Enable. When clear, the **SXx0** pin is high impedance. When set, **SXx0** outputs data from the DS0X3-DS0X0 registers, starting with the MSB of DS0X3.

49h DS1C DSP Source Port 1 Control DSPs

Bit	Label	Description	Default
17..11	BPI[6:0]	Number of bits per interrupt. Also enables/disables Source Port x1.	0000000
10	rsvd	Reserved. Write to 0	0
9	EDG	Active edge of **SCKA1** for DSP0 or **SCKB1** for DSP1. 0 is data valid on the falling edge and changing on the rising edge.	0
8	DEL	Delay from **FSYx1** edge. When set, the MSB starts one **SCKx1** clock period after **FSYx1** edge. When clear, the MSB starts at the **FSYx1** edge.	0
7	FPOL	Polarity of **FSYx1**. When 0, the start of frame is **FSYx1** low. For DSP0 the pin is **FSYA1** and for DSP1 the pin is **FSYB1**.	0
6	CHAIN	Chain **FSYx1/SCKx1** with other port (**FSYx0/SCKx0**).	0
5	rsvd	Reserved. Write to 0	0
4	SPEN	S/PDIF mode enable. Only set when **DS0C.SPEN** set and **BPI[6:0]** = 7Fh.	0
3	CKOE	Clock Output enable. When set, **FSYx1/SCKx1** are outputs. When clear, **FSYx1/SCKx1** are inputs.	0
2	SDOEN	Source Data Output Enable. When clear **SXx1** is always 0. When set, **SXx1** outputs data.	0
1,0	rsvd	Reserved. Write to 0	00

Table 4-54: DS1C Register

BPI[6:0] Number of bits per interrupt. The settings for **SCLKx1** and **FSYx1** are dependent on the **BPI[6:0]** settings. Valid **BPI[6:0]** values are:

00h - Source Port x1 is disabled. The pins can be used for GPIO. For DSP0 **BPI[6:0]** set to zero enables GPIO pins **GPB[3:0]**. For DSP1 **BPI[6:0]** set to zero enables pins **GPD[3:0]**.

1Fh - Source Port x1 enabled, with 32 bits per interrupt. Registers DS1X3- DS1X2 and DS1R1-DS1R0 are used.

3Fh - Source Port x1 enabled, with 64 bits per interrupt. Registers DS1X3- DS1X0 and DS1R3-DS1R1 are used.

7Fh - Source Port x1 enabled, with 128 bits per interrupt. **DS0C.BPI[6:0]** must equal 7Fh as Source Port x0's registers are cascaded to get all 128 bits per interrupt. For transmitting, DS0X3- DS0X0 and DS1X3-DS1X0 are used as mentioned in the **DS0C.BPI[6:0]** bit description. For receiving, DS0R3-DS0R0 and DS1R3-DS1R0 are used.

EDG Active edge of **SCKA1** for DSP0 or **SCKB1** for DSP1. **EDG** clear is data valid on the falling edge and changing on the rising edge. **EDG** set is data valid on the rising edge and changing on the falling edge.

DEL Delay from **FSYx1** edge. When set, the MSB starts one **SCKx1** clock period after **FSYx1** edge (Philips I^{2}S style). When clear, the MSB starts at the **FSYx1** edge.

FPOL Polarity of **FSYx1**. When 0, the start of frame is **FSYx1** going low. When **FPOL** is set, the start of a frame is **FSYx1** going high.

CHAIN Chain **FSYx1/SCKx1** with other port (**FSYx0/SCKx0**). Source Port x0 clocks (**FSYx0/SCKx0**) are also the timing source for Source Port x1. Therefore, either the internal DSP timers must be output on the Source Port x0 clocks, or externally derived clocks must be input on **FSYx0/SCKx0**. **CHAIN** must also be cleared when cascading the two Source Ports in 128-bit mode (**BPI[6:0]** = 7Fh).

SPEN S/PDIF enabled. Must only be set when Source Port x0 is configured for 128-bit per interrupt S/PDIF operation (**DS0C.BPI[6:0]** set to 7Fh).

CKOE Clock Output enable. When set, the DSP timers are output on the **FSYx1**/**SCKx1** pins, where DSP timer 0 goes to the **SCKx1** pin and timer 1 goes to the **FSYx1** pin. If the **CHAIN** bit is clear, then **SRx1**/**SXx1** are clocked using the DSP timers. If **CHAIN** is set when **CKOE** is set, **SRx1**/**SXx1** are clocked from **SCKx0**/**FSYx0**, and **SCKx1**/**FSYx1** just output the DSP timer clocks. When **CKOE** is clear, **FSYx1**/**SCKx1** are inputs. When Source Port x0 is configured for S/PDIF (**DS0C.SPEN** set), then Source Port x0 uses the DSP timers. In this case, Source Port x1 can only set this bit if Port x1 uses the same timing as Source Port x0.

SDOEN Source Data Output Enable. When clear, the **SXx1** pin is high impedance. When set, **SXx1** outputs data from the DS1X3-DS1X0 registers, with the MSB of DS1X3 transmitted first.

The Source Port interrupt is OR'd with the GPIO interrupt; therefore, the Source Port interrupt priority is configurable via the **IER.PRI[2:0]** bits. Since the Source Port pins for both DSPs are shared with the Host Controller, the corresponding interrupt enables for the GPIO function must be disabled when the Source Ports are in use. For both DSP0 and DSP1, when Source Port x0 is in use, the DSP GPIO interrupt is automatically disabled in hardware since the **SCKx0** pin of the Source Port x0 is shared as the EGP0 of the DSPs, which can triggered the DSP GPIO interrupt. For the Host Controller, when DSP1 is using Source Port x0, the Host Controller timer interrupts must be disabled in software.

4.4.2.1 S/PDIF Format

The S/PDIF operation is only available on Source Port x0. However, a 128-bit S/PDIF format links both Source Ports together, using both sets of registers to transfer 128 bits per interrupt. To select S/PDIF mode, **DSnC.SPEN** must be set. To select 128-bit S/PDIF mode, both **DS0C.SPEN** and **DS1C.SPEN** must be set, and both **DSnC.BPI[6:0]** must be set to 7Fh.

When S/PDIF format is selected, both DSP Timers must be configured for S/PDIF operation regardless of whether **SCKxn** and **FSYxn** are inputs or outputs. DSP Timer 0 must be configured for two times the bits per S/PDIF frame when the **DTC.T0D2E** is not set, and four times the bits per S/PDIF frame when the **DTC.T0D2E** is set. Since both DSP Timers are used in S/PDIF mode, if Source Port x1 is configured to output **SCKx1** and **FSYx1**, it must be the same frequency and bit rate as Source Port x0. **SCKx1** output will be the Timer 0 setting divided by two (same bit rate as Port 0). If Source Port x1 is configured to input **SCKxn** and **FSYxn**, it can be totally independent from Source Port x0. If both Ports are independent from each other, two interrupts will be needed.

When setting Source Port x0 to S/PDIF format, Timer 0 and Timer 1 must be set properly. The transmitted bit stream on **SXx0** is set by the combination of Timer 0 settings (set to twice the bit period of the S/PDIF stream), and Timer 1 settings for the preambles as listed in Table 4-55. In S/PDIF mode, the output of Timer 0 is internally divided by two before sending the output to Timer 2 or the output pin.

S/PDIF		Timer 0		Timer 1		Interrupts		Registers Used	
Mul.	SXx0 Bits/Fs	Clocks /Fs	DDIV0	Periods per Fs	DDIV1*	Ints. per Fs	DSnC. BPI	TX	RX
1x	64	128	000Bh*	1	001Fh	1	3Fh	DSnX3-DSnX0	DSnR3-DSnR0
2x	128	256	0005h*	1	003Fh	2	3Fh	DSnX3-DSnX0	DSnR3-DSnR0
4x	256	512	0002h*	1	007Fh	4	3Fh	DSnX3-DSnX0	DSnR3-DSnR0
2x	128	256 †	0005h*	1	003Fh	1	7Fh	DS0X3-DS0X0 + DS1X3-DS1X0	DS0R3-DS0R0 + DS1R3-DS1R0
4x	256	512 †	0002h*	1	007Fh	2	7Fh	DS0X3-DS0X0 + DS1X3-DS1X0	DS0R3-DS0R0 + DS1R3-DS1R0
8x	512	1024 †	0002h	1	00FFh	4	7Fh	DS0X3-DS0X0 + DS1X3-DS1X0	DS0R3-DS0R0 + DS1R3-DS1R0

* For Timer 0, assumes **DTC.T0D2E** set (except the 8x case); and for Timer 1, assumes **DTC.T1D2E** set.

† 128-bit mode, where Source Port x1 registers are cascaded/used; therefore, Source Port x1 is not available. When in 128-bit mode, all four interrupts occur and must be cleared by reading or writing the appropriate block of registers.

Table 4-55: Valid S/PDIF Clocking Combinations for Async. Source Port 0

The first three columns in Table 4-55 use the Source Port in 64-bit per interrupt mode, and the last three columns use 128-bit per interrupt mode by cascading the two source ports together.

Figure 4-9 illustrates the S/PDIF data stream alignment with respect to the Async. Source Port data registers for 64-bit per interrupt mode. For the transmitted bit stream on **SXx0**, the required preambles are generated automatically. The preamble portion of the data must be zero for proper preamble transmission:

- Left preamble bits: **DS0X[7:4]** = 0000 (bits 9-6 of register DS0X0).
  To force a Block preamble, set bits: **DS0X[7:4]** = 0110, which is also resync the internal block counter that generates a Block preamble once every 192 frames.
- Right preamble bits: **DS0X[39:36]** = 0000 (bits 9-6 of register DS0X2).

For the received bit stream on **SRx0**, the left preamble indicates when a new CS block starts. The preamble portion of the received data is:

- Left preamble bits: **DS0R[7:4]** (bits 9-6 of register DS0R0) =
  1010 - left preamble
  0110 - block preamble
- Right preamble bits: **DS0R[39:36]** = 0110 (bits 9-6 of register DS0R2).

Figure 4-10 illustrates the S/PDIF data stream alignment with respect to the Async. Source Port data registers for 128-bit per interrupt mode, with the two Source Ports cascaded. For the transmitted bit stream on **SXx0**, the four preamble portions of the data must be zero for proper preamble transmission:

- Left preamble bits: **DS0X[7:4]** = **DS1X[7:4]** = 0000 (bits 9-6 of registers DS0X0 and DS1X0).
  To force a Block preamble, set the bits to 0110, which is also resync the internal block counter.
- Right preamble bits: **DS0X[39:36]** = **DS0X[39:36]** = 0000 (bits 9-6 of registers DS0X2 and DS1X2).

For the received bit stream on **SRx0**, the left preamble indicates when a new CS block starts. The preamble portion of the received data is:

- Left preamble bits: **DS0R[7:4]** or **DS1R[7:4]** (bits 9-6 of register DS0R0 or DS1R0) =
  1010 - left preamble
  0110 - block preamble
- Right preamble bits: **DS0R[39:36]** = **DS1R[39:36]** = 0110 (bits 9-6 of register DS0R2 and DS1R2).

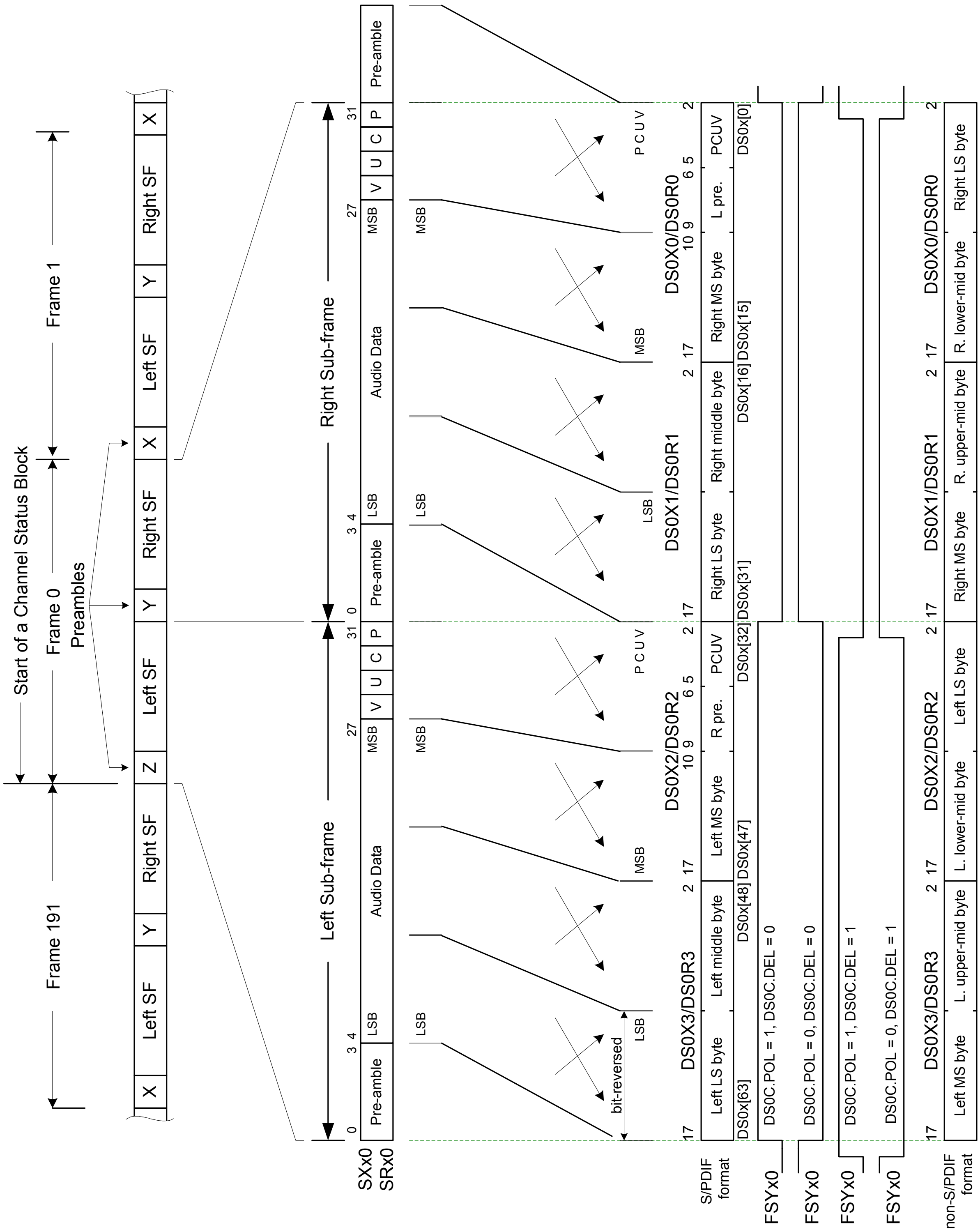


Figure 4-9: DSP Async. Source Port S/PDIF Alignment - 64 Bits/Interrupt

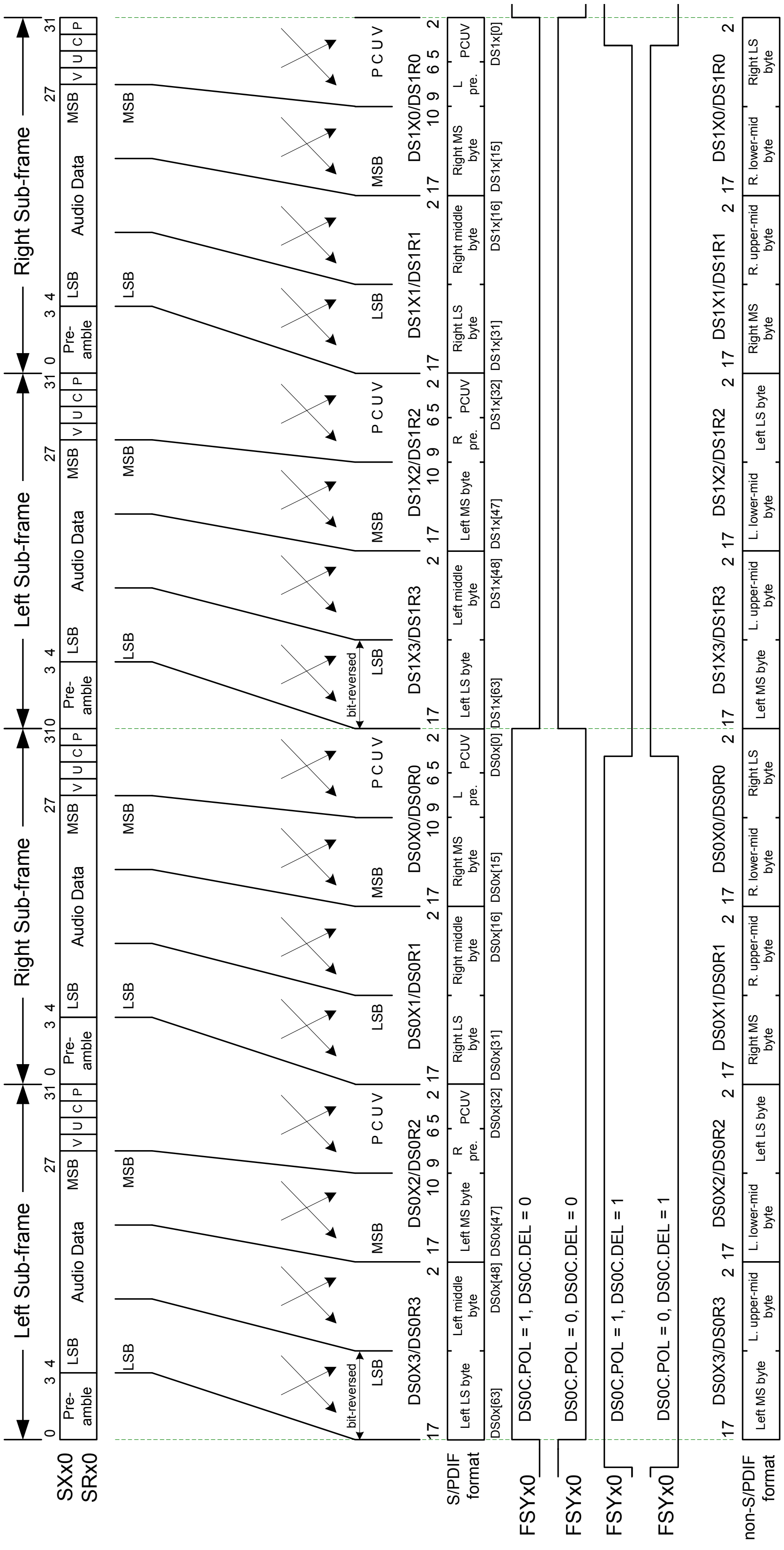


Figure 4-10: DSP Async. Source Port S/PDIF Alignment - 128 Bits/Interrupt

When transmitting complete S/PDIF data (including the preambles) to the Source Port SX0, the Left and Right preambles must be set to 0000 and the Block preamble must be set to 0001. These values also hold true when transmitting complete S/PDIF data across the MOST Network to an OS8104 or OS8804 Source Port. When receiving complete S/PDIF data from any of these sources, the preambles must be converted to the values listed above for transmitting out the DSP's Asynchronous Source Port.

When transmitting and receiving S/PDIF data, the interrupts for each direction cannot be aligned; therefore, independent ISRs are needed for transmit and receive.

4.5 Source Converter Volume Control

The ADACn registers, GADCs, GMIC and GMPX registers are mapped to both DSPs as well as the Host Controller. Since both the Host Controller and DSPs can write to the register to change the setting, the last write of the register will remain and that is the value to be read back. Therefore, a DSP may not read back what it wrote to the register if the Host Controller (or other DSP) writes to it after the DSP write. These registers are only available if the particular Source Converter is enabled by the Host Controller, **FPCR.RUN** is set, and the Source Port clocks (FSY and SCK) are operating (are outputs and enabled, or are inputs and driven by external clocks).

50h ADAC0 Audio DAC0 Volume DSPs

Bit	Label	Description	Default
17..6	rsvd	Reserved. Write to 0.	000h
5	MUTE	Mute. When set, DAC0 output is muted.	0
4..0	ATTN[4:0]	DAC0 Attenuation. The least significant bit represents -1 dB, with 00000 = 0 dB and 11111 = -31 dB.	00000

Table 4-56: ADAC0 Register

51h ADAC1 Audio DAC1 Volume DSPs

Bit	Label	Description	Default
17..6	rsvd	Reserved. Write to 0.	000h
5	MUTE	Mute. When set, DAC1 output is muted.	0
4..0	ATTN[4:0]	DAC1 Attenuation. The least significant bit represents -1 dB, with 00000 = 0 dB and 11111 = -31 dB.	00000

Table 4-57: ADAC1 Register

52h ADAC2 Audio DAC2 Volume DSPs

Bit	Label	Description	Default
17..6	rsvd	Reserved. Write to 0.	000h
5	MUTE	Mute. When set, DAC2 output is muted.	0
4..0	ATTN[4:0]	DAC2 Attenuation. The least significant bit represents -1 dB, with 00000 = 0 dB and 11111 = -31 dB.	00000

Table 4-58: ADAC2 Register

53h ADAC3 Audio DAC3 Volume DSPs

Bit	Label	Description	Default
17..6	rsvd	Reserved. Write to 0.	000h
5	MUTE	Mute. When set, DAC3 output is muted.	0
4..0	ATTN[4:0]	DAC3 Attenuation. The least significant bit represents -1 dB, with 00000 = 0 dB and 11111 = -31 dB.	00000

Table 4-59: ADAC3 Register

54h GMPX MPX ADC Volume DSPs

Bit	Label	Description	Default
17..5	rsvd	Reserved. Write to 0.	000h, 0
4..0	GAIN[4:0]	MPX ADC Gain. The least significant bit represents 1 dB, with 00000 = 0 dB and 11010 = 26 dB (maximum value).	00000

Table 4-60: GMPX Register

55h GMIC Mic ADC Volume DSPs

Bit	Label	Description	Default
17..4	rsvd	Reserved. Write to 0.	000h, 00
3..0	GAIN[3:0]	Mic ADC Gain. The least significant bit represents 1 dB, with 0000 = 0 dB and 1111 = 15 dB.	0000

Table 4-61: GMIC Register

56h GADL Left Audio ADC Volume DSPs

Bit	Label	Description	Default
17..4	rsvd	Reserved. Write to 0.	000h, 00
3..0	GAIN[3:0]	Left Audio ADC Gain. The least significant bit represents 1 dB, with 0000 = 0 dB and 1111 = 15 dB.	0000

Table 4-62: GADL Register

57h GADR Right Audio ADC Volume DSPs

Bit	Label	Description	Default
17..4	rsvd	Reserved. Write to 0.	000h, 00
3..0	GAIN[3:0]	Right Audio ADC Gain. The least significant bit represents 1 dB, with 0000 = 0 dB and 1111 = 15 dB.	0000

Table 4-63: GADR Register

4.6 Unique DSP Peripherals

The DSP0 interfaces to two additional peripherals: a external data memory interface and two pulse width modulation (PWM) DACs. The external data memory interface allow the DSP0 to access external memory for data storage. The DSP0 can optionally implement a crossover function or serve as an interpolation filter for the source data before transmitting to the PWM DACs. Each DSP controls up to eight EGPIO pins, shared with the Host Controller (a different eight for each DSP).

4.6.1 DSP0 External Data Memory Interface

The external data memory port is an I/O peripheral which provides a zero glue logic interface to external DRAM or SRAM. The port can be configured as an 8-bit wide or 4-bit wide data bus. The SRAM interface supports up to 128k bytes, and the DRAM interface supports up to 1M bytes. In DRAM mode, 10 address pins (**MA[9:0]**) are used (with the upper address bits being indeterminate) and in SRAM mode 17 address pins (**MA[14:0]**, **SA15**, and **MA16**) are available.

The memory row address select ($\overline{\textbf{MRAS}}$), memory column address select ($\overline{\textbf{MCAS}}$), memory read ($\overline{\textbf{MRD}}$), and memory write ($\overline{\textbf{MWR}}$) pins control the data transfer with external DRAM. The $\overline{\textbf{MRAS}}$ pin is a multifunction pin which becomes the memory chip select ($\overline{\textbf{MCS}}$) when communicating with external SRAM. The memory read and write signals are used for both DRAM and SRAM. The $\overline{\textbf{MCAS}}$ pin becomes **MA14** and **XME/**$\overline{\textbf{PCS}}$ pin becomes **SA15** with external SRAM.

The external data memory port is internally connected to the DSP0 I/O bus, as illustrated in Figure 4-13. The XMC register (page 94) configures the external memory port for use as Program memory for the Host Controller, or data memory for DSP0. In addition, **GCTL.EDMEN** must be set to enable the external data

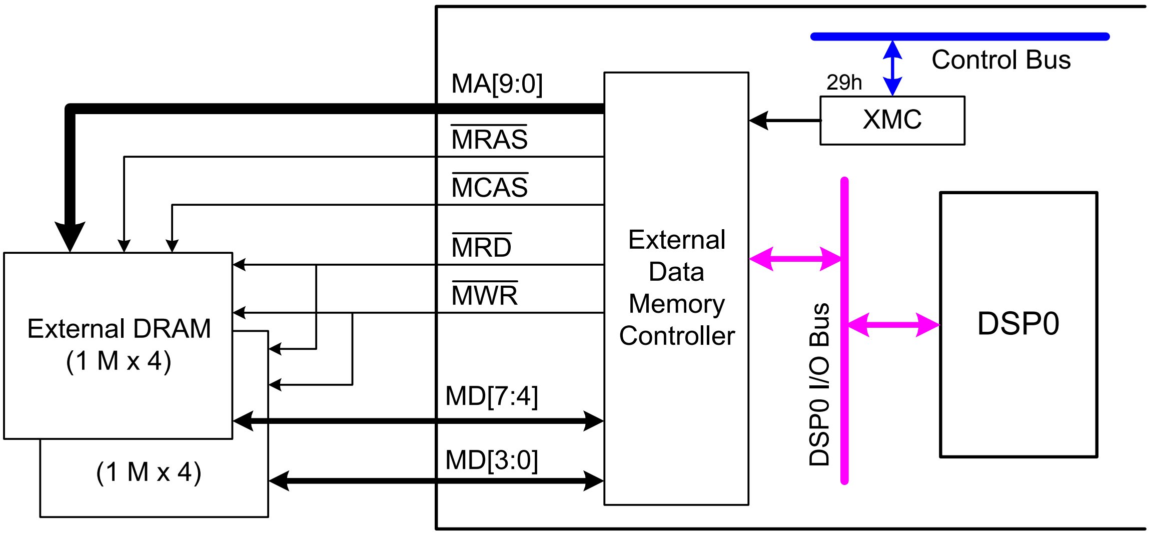


Figure 4-11: DSP0 External DRAM Interface

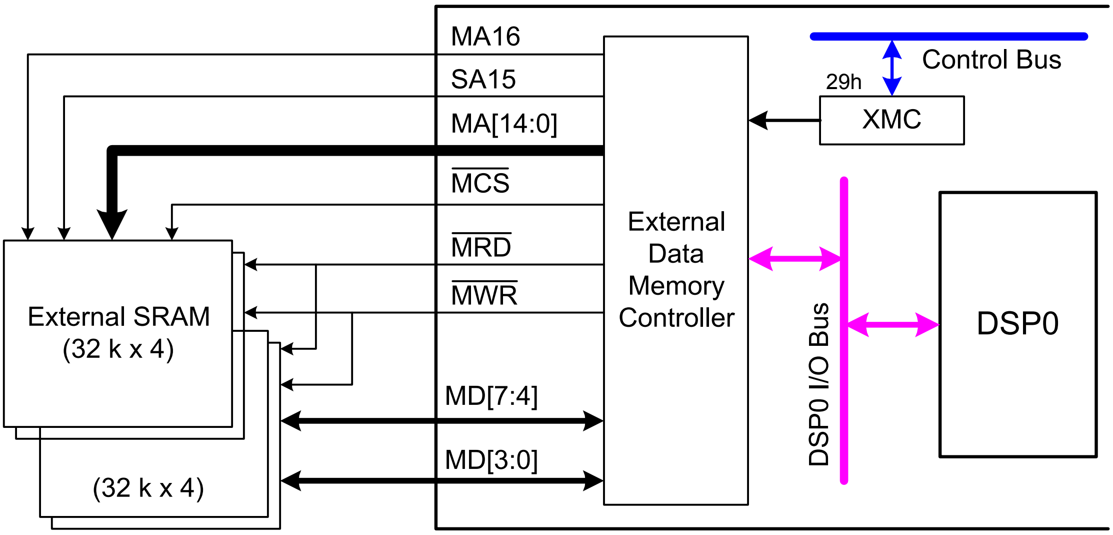


Figure 4-12: DSP0 External SRAM Interface

memory interface. If **GCTL.EDMEN** is clear, the interface can be used as EGPIO by the Host Controller. The XMC and GCTL registers reside on the Control bus and must be configured by the Host Controller. See the XMC register in the *Host Controller* section for more information on configuring the external memory port for DSP0 data memory.

The DSP accesses the external data memory port through a set of 16-bit I/O registers. The port communicates with external memory through either a 4- or 8-bit data bus. The external data memory port divides 8- or 16-bit data from the DSP into bytes or nibbles before writing to external memory. Likewise, the port assembles 8- or 16-bit words for the DSP from byte or nibble data from external memory. See Figure 2-33 on page 95 and Figure 2-34 on page 96, for examples of how external memory is addressed.

Up to 512k 16-bit words can be stored in two external DRAM's and up to 64k 16-bit words can be stored in external SRAM's. External DRAM can be automatically refreshed. The refresh and the read/write timing can be adjusted.

The memory timing (**XMC.MT**) bit specifies fast and slow memory accesses. When **XMC.MT** is low, memory accesses are fast. The DSP0 data memory timing can be found in *External Data Memory Interface* under the *Electrical Characteristics* Section.

DRAM can be automatically refreshed by setting the Automatic Refresh Enable (**XMC.ARE**) bit high. The Refresh Clock Divider (**XMC.RCD[7:0]**) bits and the Refresh Clock Prescaler (**XMC.RCP[1:0]**) bits specify the time in DSP0 clock cycles (typically 1344xFs), between CAS before RAS refresh cycles. The maximum time between refresh cycles is 483 µs. Refresh cycles have a higher priority than read or write operations. If successive reads or writes are being performed and the refresh timer requests a refresh cycle, the reads and writes will be stopped for the refresh cycle to complete.

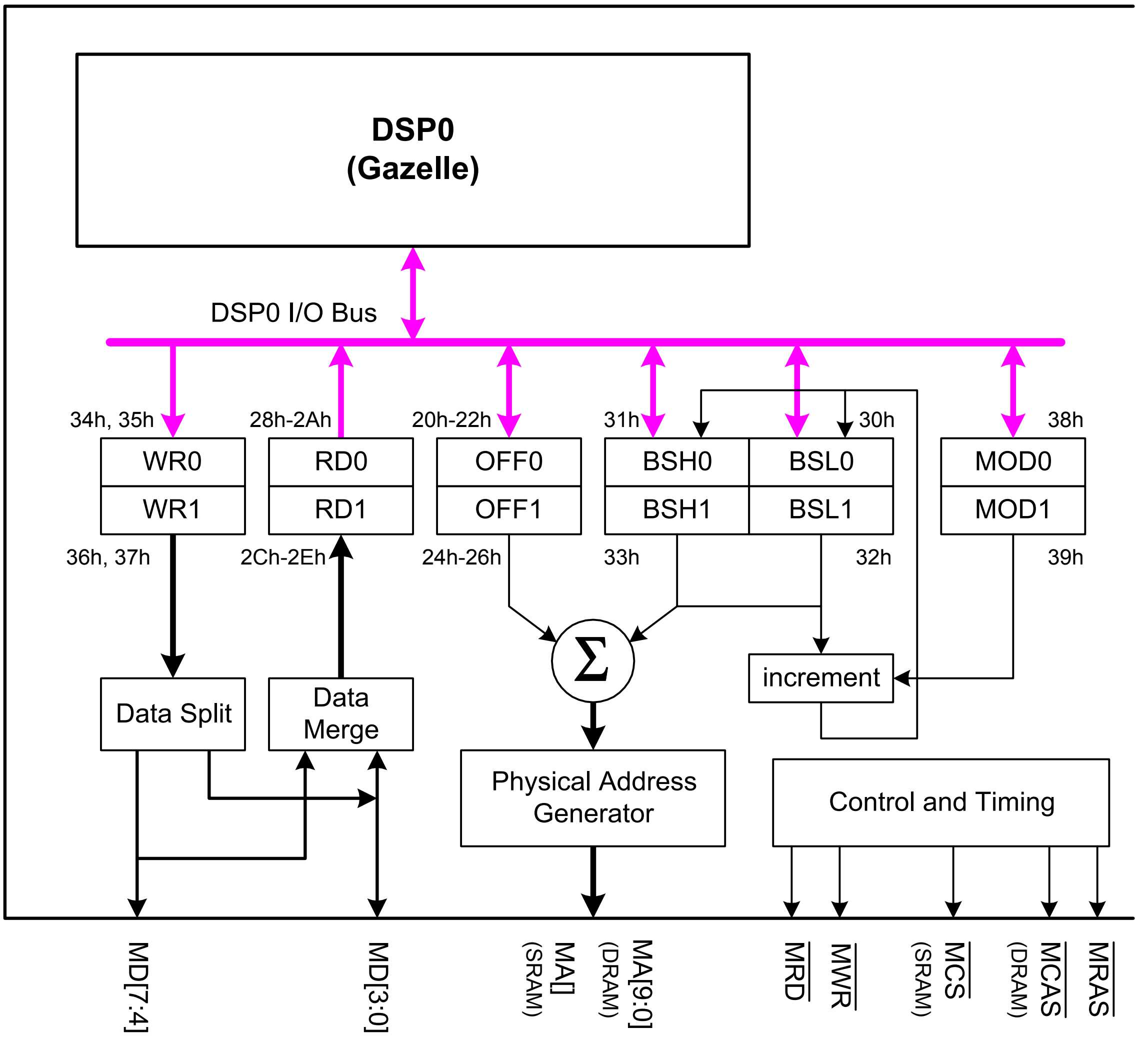


Figure 4-13: DSP0 External Memory Controller

An access to external memory can take many DSP0 instruction cycles to complete, and can be interrupted by refresh cycles when using DRAM. Once an access is started (read or write), trying to read the results (read from 28h, 29h, 2Ah, 2Ch, 2Dh, or 2Eh) or start a new access (write to 21h, 22h, 25h, 26h, 34h, 35h, 36h, or 37h) will stall the DSP until the memory cycle completes. Once stalled, the DSP cannot be interrupted. Table 4-64 indicates the number of DSP cycles required to complete a transaction once the execute cycle of the instruction that caused the operation occurs. Two cycles are needed to calculate the address, and two cycles are needed at the end of the operation (both included in the table below). Therefore, if the chip is configured for fast DRAM mode and one access per operation, then a read operation will take nine cycles to complete. If the read instruction execute cycle occurs in the tenth cycle after the execute cycle that started the access, then the DSP will not be stalled.

External Memory Port Accesses per Operation	SRAM		DRAM	
	Fast	Slow	Fast	Slow
1 Access (8-bit word and bus)	5	6	10	13
2 Accesses (8-bit word on 4-bit bus or 16-bit word on 8-bit bus)	7	9	13	17
4 Accesses (16-bit word on 4-bit bus)	11	15	19	25

Table 4-64: DSP Cycles for EMI Access

4.6.1.1 External Memory Addressing

The External Memory port has two sets of Base (BSH/L1-0), Offset (OFF1-0), Modulo (MOD1-0), Read (RD1-0), and Write (WR1-0) registers. The Base, Offset, and Modulo registers generate the high physical address of external memory. The low bits are automatically generated by the port hardware.

When storing 16-bit words through the 8-bit data bus, one address LSB is created by hardware to map the 16-bit data from the DSP to 2-byte locations in external memory. Likewise, when storing bytes through the 4-bit bus, one address LSB is created by hardware. When storing words through the 4-bit bus, the two address LSBs are created by hardware. When storing bytes through the 8-bit bus, no address LSBs are created.

The Base, Offset, and Modulo registers determine the high physical address bits for the read or write access. The values in the Base and Modulo registers together define a circular buffer of arbitrary length starting at a 2^N block boundary, where N is the number of bits required to represent the buffer length. The highest bit set in the MOD register determined N. The *lower bound* for the modular buffer it determined by masking off the lower N bits from the base address. The *upper bound* is then determined by adding the MODn value to the lower bound. As an example, if BnAD (base address) = 013A7h and MODn contained 000Bh, then the circular buffer lower bound = 013A0h and the upper bound = 013ABh.

The start address of the circular buffer is given by adding the lower bound to the OFFn register. For no modulo, set the Modulo value to maximum. A Modulo value of 0 is a circular buffer of length one.

To generate the physical address, the hardware adds the Base and Offset registers and, if the result exceeds the upper bound for the circular buffer, subtracts the (Modulo register plus one). If the OFFn value is less than or equal to the Modulo value, the generated address always lies within the circular buffer. To continue the above example, if **OFn[15:0]** = 000Ah is added to BnAD = 013A7h, then the address would be 013B1h, which is outside the upper bound of 013ABh. Therefore the physical address generated would be 013B1h - (MOD+1), or 013B1h - (000Bh + 1), which is 013A5h.

By specifying an Offset value greater than the Modulo value, it is possible for the generated address to lie in a different buffer. Using the above example, but with **OFn[15:0]** = 0100h, which is larger than the Modulo value, generates an address of 023A7h, which is also outside the upper bound of 013ABh. Therefore the physical address generated would be 023A7h - (MOD+1), or 023A7h - (000Bh + 1), which is 0239Bh. This physical address is outside the buffer due to the large offset, and subtracting MOD+1 cannot bring the physical address back inside the buffer.

The base address can be post incremented by one after a read or write. The post increment is performed modulo the value in the Modulo register so that the base address always remains in the circular buffer.

Data from external memory is loaded into a Read register at the end of a read operation (which takes several cycles). Data to be written to external memory is loaded by DSP0 into one of the Write register.

4.6.1.2 DSP0 External Memory Register Description

20h OFF0 Offset 0 DSP0

Bit	Label	Description	Default
17,16	rsvd	Reserved, Write to 0	00
15..0	D[15:0]	Provides offset from base address **B0AD[19:0]**. With 4 MSBs set to 0, OFF0 is added to **B0AD[19:0]** to generate the physical address. If the physical address is greater than the upper bound of the circular buffer, (MOD0+1) is subtracted before the physical address is generated.	0000h

Table 4-65: OFF0 Register

21h OFF0+- Offset 0, Start Read DSP0

Bit	Label	Description	Default
17,16	rsvd	Reserved, Write to 0	00
15..0	D[15:0]	A write to OFF0+- initiates a read operation.	0000h

Table 4-66: OFF0+- Register

22h OFF0++ Offset 0, Start Read with Post Increment DSP0

Bit	Label	Description	Default
17,16	rsvd	Reserved, Write to 0	00
15..0	D[15:0]	A write to OFF0++ initiates a read operation, and post-increments **B0AD[19:0]**.	0000h

Table 4-67: OFF0++ Register

24h OFF1 Offset 1 DSP0

Bit	Label	Description	Default
17,16	rsvd	Reserved, Write to 0	00
15..0	D[15:0]	Provides offset from base address **B1AD[19:0]**. With 4 MSBs set to 0, OFF1 is added to **B1AD[19:0]** to generate the physical address. If the physical address is greater than the upper bound of the circular buffer, (MOD1+1) is subtracted before the physical address is generated.	0000h

Table 4-68: OFF1 Register

25h OFF1+- Offset 1, Start Read DSP0

Bit	Label	Description	Default
17,16	rsvd	Reserved, Write to 0	00
15..0	D[15:0]	A write to OFF1+- initiates a read operation.	0000h

Table 4-69: OFF1+- Register

26h OFF1++ Offset 1, Start Read with Post Increment DSP0

Bit	Label	Description	Default
17,16	rsvd	Reserved, Write to 0	00
15..0	D[15:0]	A write to OFF1++ initiates a read operation, and post-increments **B1AD[19:0]**.	0000h

Table 4-70: OFF1++ Register

28h RD0 Read 0 (read only) DSP0

Bit	Label	Description	Default
17..2	D[15:0]	RD0 contains data from a previous read operation. Reading this register does not start a new read.	0000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-71: RD0 Register

29h RD0+- Read 0, Start Read (read only) DSP0

Bit	Label	Description	Default
17..2	D[15:0]	RD0+- contains data from a previous read operation, and a new read operation is started.	0000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-72: RD0+- Register

2Ah RD0++ Read 0, Start Read with Post Increment (read only) DSP0

Bit	Label	Description	Default
17..2	D[15:0]	RD0++ contains data from a previous read operation, a new read operation is started, and **B0AD[19:0]** is post-incremented.	0000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-73: RD0++ Register

2Ch RD1 Read 1 (read only) DSP0

Bit	Label	Description	Default
17..2	D[15:0]	RD1 contains data from a previous read operation. Reading this register does not start a new read.	0000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-74: RD1 Register

2Dh RD1+- Read 1, Start Read (read only) DSP0

Bit	Label	Description	Default
17..2	D[15:0]	RD1+- contains data from a previous read operation, and a new read operation is started.	0000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-75: RD1+- Register

2Eh RD1++ Read 1, Start Read with Post Increment (read only) DSP0

Bit	Label	Description	Default
17..2	D[15:0]	RD1++ contains data from a previous read operation, a new read operation is started, and **B1AD[19:0]** is post-incremented.	0000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-76: RD1++ Register

30h BSL0 Base Address 0 Low DSP0

Bit	Label	Description	Default
17,16	rsvd	Reserved, Write to 0	00
15..0	B0AD[15:0]	Low 16 bits of base address 0. When post-incremented, **B0AD[19:0]** is incremented and, if greater than the circular buffer upper bound, (MOD0 +1) is subtracted from **B0AD[19:0]** before storing back into BSL0/BSH0.	0000h

Table 4-77: BSL0 Register

31h BSH0 Base Address 0 High DSP0

Bit	Label	Description	Default
17..4	rsvd	Reserved, Write to 0	000h, 00
3..0	B0AD[19:16]	High 4 bits of base address 0, used for external memory access with the other base-0 registers.	0000

Table 4-78: BSH0 Register

32h BSL1 Base Address 1 Low DSP0

Bit	Label	Description	Default
17,16	rsvd	Reserved, Write to 0	00
15..0	B1AD[15:0]	Low 16 bits of base address 1. When post-incremented, **B1AD[19:0]** is incremented and, if greater than the circular buffer upper bound, (MOD1 +1) is subtracted from **B1AD[19:0]** before storing back into BSL1/BSH1.	0000h

Table 4-79: BSL1 Register

33h BSH1 Base Address 1 High DSP0

Bit	Label	Description	Default
17..4	rsvd	Reserved, Write to 0	000h, 00
3..0	B1AD[19:16]	High 4 bits of base address for external memory access	0000

Table 4-80: BSH1 Register

34h WR0+- Write 0, Start Write (write only) DSP0

Bit	Label	Description	Default
17..2	D[15:0]	Writing WR0+- starts a write operation using the base-0 registers with the data in WD0.	0000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-81: WR0+- Register

35h WR0++ Write 0, Start Write with Post Increment (write only) DSP0

Bit	Label	Description	Default
17..2	D[15:0]	Writing WR0++ starts a write operation using the base-0 registers with the data in WD0 and then post-increments **B0AD[19:0]**.	0000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-82: WR0++ Register

36h WR1+- Write 1, Start Write (write only) DSP0

Bit	Label	Description	Default
17..2	D[15:0]	Writing WR1+- starts a write operation using the base-1 registers with the data in WD1.	0000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-83: WR1+- Register

37h WR1++ Write 1, Start Write with Post Increment (write only) DSP0

Bit	Label	Description	Default
17..2	D[15:0]	Writing WR1++ starts a write operation using the base-1 registers with the data in WD1, and then post-increments **B1AD[19:0]**.	0000h
1,0	rsvd	Reserved, Write to 0	00

Table 4-84: WR1++ Register

38h MOD0 Modulo 0 DSP0

Bit	Label	Description	Default
17,16	rsvd	Reserved, Write to 0	00
15..0	D[15:0]	Highest bit set defined the number of bits used for the circular buffer and along with **B0AD[19:0]**, the lower bound. Then MOD0, added to the lower bound, define the upper bound.	0000h

Table 4-85: MOD0 Register

39h MOD1 Modulo 1 DSP0

Bit	Label	Description	Default
17,16	rsvd	Reserved, Write to 0	00
15..0	D[15:0]	Highest bit set defined the number of bits used for the circular buffer and along with **B1AD[19:0]**, the lower bound. Then MOD1, added to the lower bound, define the upper bound.	0000h

Table 4-86: MOD1 Register

4.6.1.3 Write Operation

To write data to external memory, DSP0 loads a Base, Offset, and Modulo register with the appropriate values and then writes the data word to WR0+- or WR1+-. The act of writing to the Write register initiates the write process. Either set of Base, Offset, and Modulo registers can generate the address for the stored data by writing to the different Write register I/O addresses. The selected Base register can be optionally post-incremented by writing to the Write register ending in “++”. The post-increment value can only be read by the DSP five instruction cycles after the post-increment instruction (read or write). To read the offset after the post-increment, the following delay would be needed:

```
wr0++ = var;    // start a write, and do a post increment.
nop;
nop;
nop;
nop;
accl = bsl0;    // must wait 5 inst. cycles to read post-incremented value
```

Likewise, a delay is needed on updating the base registers after post-incrementing or new data could be overwritten by the post-increment result. Since the post-increment takes five cycles to complete the following is errored:

```
wr0++ = var;    // starts a write and do a post-increment
nop;
bsl0 = acch;    // update the base address BEFORE post increment happens
nop;
nop;            // at this point bsl0 updated with post increment and
                //   ACCH in BSL0 is OVER-WRITTEN
```

When DSP0 writes to either Write register (WR0+- or WR0++), the data stored in **WD0[15:0]** is stored in external memory using the address generated by the Base 0, Offset 0, and Modulo 0 registers. Writing to WR1++ initiates the same procedure as writing to WR0+- except **B0AD[19:0]** is post incremented by one. Writing to WR1+- or WR1++ initiates the same procedure as writing to WR0+- or WR0++, except the Base 1, Offset 1, and Modulo 1 registers are used.

4.6.1.4 Read Operation

Using “0” suffix registers as an example, a read operation is initiated by reading from Read registers RD0+- or RD0++, or writing to OFF+- or OFF++ registers. Writing to OFF0 or reading from RD0 does not initiate a read. Writing to OFF0+- initiates a read process. The result is stored in RD0 when available. Writing to OFF0++ initiates the same read process and **B0AD[19:0]** is post-incremented. The post-increment value can only be read by the DSP five instruction cycles after the post-increment instruction. If the post-incremented **B0AD[19:0]** value is greater than MOD0, then **B0AD[19:0]** = **B0AD[19:0]** - (MOD0 +1).

Writing to Offset 1 I/O addresses 24h, 25h, and 26h start the same processes as writing to 20h, 21h, and 22h respectively, except Base 1, Offset 1, and Modulo1 generate the address and RD1 contains the result when the read operation finishes.

Data can be read, without initiating another read, by reading RD0. Reading RD0+- start a read, but does not post-increment the corresponding Base register. Reading RD0++ start a read, and post-increments the corresponding Base register.

4.6.1.5 Programming Examples

The following is an example of how a block of data can be written to external memory.

```
bsl0 = low_base_value;  // set low base address
bsh0 = high_base_value; // set high base address
off0 = offset_value;    // set offset
mod0 = modulo_value;    // set modulo
wr0++ = data0;          // write 1st word and post increment
wr0++ = data1;          // write 2nd word and post increment
wr0++ = data2;          // write 3rd word and post increment
wr0 = data3;            // write last word
```

Since the processor will be waiting between writes, this time could be filled with useful instructions. The following is an example of how to read a block of data.

```
bsl1 = low_base_value;  // set low base address
bsh1 = high_base_value; // set high base address
mod1 = modulo_value;    // set modulo
off1++ = offset_value;  // set offset, start read, and post increment
data0 = rd1++;          // read 1st word, start next read, and post increment
data1 = rd1++;          // read 2nd word, start next read, and post increment
data2 = rd1++;          // read 3rd word, start next read, and post increment
data3 = rd1;            // read last word
```

External memory can be used to create circular delay buffers. Each sample period a new data word overwrites the oldest sample in the buffer. Each sample period many samples with fixed delays are read from the buffer. The following is an example.

```
                  // during initialization set base and modulo
bsl1 = low_base_value;  // set low base address
bsh1 = high_base_value; // set high base address
mod1 = modulo_value;    // set modulo

                  // In sample rate interrupt
off1+- = offset_value0; // set offset, start read
data0 = rd1;            // read 1st word
off1+- = offset_value1; // set offset, start read
data1 = rd1;            // read 2nd word
off1+- = offset_value2; // set offset, start read
data2 = rd1;            // read 3rd word
off1+- = offset_value3; // set offset, start read
data3 = rd1;            // read last word
off1 = offset_value1;   // set write offset
wr1++ = new_sample;     // write new sample and post-increment base
```

4.6.1.6 Interrupts

There are 2 sets of Base, Offset, Modulo, and Read registers to provide easy context switching in the highest priority interrupt. One set of registers can be used in the highest priority interrupt and the other set in all other interrupt routines.

If multiple levels of interrupts use the same set of registers, a software stack must be maintained. The base, offset, modulo, and read registers must be pushed onto the stack at the beginning of the ISR and popped off at the end. The following illustrates how a medium level interrupt service routine manages the external memory interface registers.

```
          // push registers onto stack
push bsl1;
push bsh1;
push off1;
push mod1;
push rd1;                // waits until data is available

          // load registers with new values
bsl1 = *low_base_address;
bsh1 = *high_base_address;
mod1 = modulo_value;
off1 = offset_value;

          // interrupt service routine
*low_base_address = bsl1; // save low base register
*high_base_address = bsh1;// save high base register

          // pop registers
pop rd1;
pop mod1;
pop off1;
pop bsh1;
pop bsl1;
reti;
```

If an interrupt occurs while the processor is waiting for a read or write to complete, control immediately jumps the ISR. After an ISR completes, the processor will not return to a wait state. The ISR must be long enough to ensure that a read or write, that started before the interrupt, will complete by the time the ISR finishes.

4.6.2 DSP0 PWM DACs

A PWM DAC supports high efficiency sub-woofer applications or can be a volume control for low-power applications. The PWM DACs consist of a noise-shaping quantizer and a PWM modulator. A DSP low-pass filter should filter signal components in the audible range before feeding the data to the noise shaper. The noise shaper processes data at an 8Fs rate. If DSP0 sends data at a lower rate, the data will be digitally up-sampled to 8Fs (sample and hold). A third-order error-feedback noise shaper quantizes and shapes the noise prior to PWM. The PWM modulator converts the PCM data to a pulse-width modulated data waveform at 8Fs.

A mute signal (**ACR.PWMOE** or **ACR2.PWM1OE** clear) forces the modulator output to AC ground. A zero-crossing detector prevents clicks and pops due to mute/de-mute actions. A power down signal (**ACR.ENPWM** or **ACR2.ENPWM1** clear) forces the output to DC ground when entering a low power state. The signal path for PWM0 and PWM1 is shown in Figure 4-14. PWM1 is identical with the exception of a separate register interface.

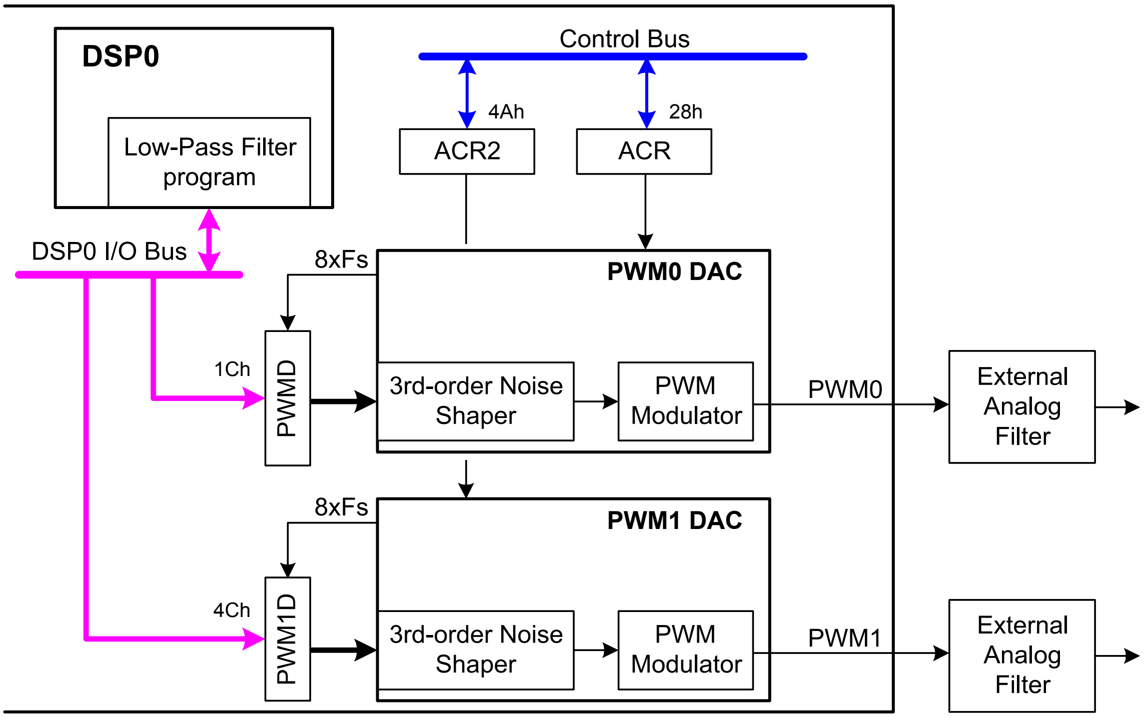


Figure 4-14: DSP0 PWM DACs

1Ch PWMD PWM DAC Data (write only) DSP0

Bit	Label	Description	Default
17..0	D[17:0]	PWM data	0000h, 00

Table 4-87: PWMD Register

4Ch PWM1D PWM1 DAC Data (write only) DSP0

Bit	Label	Description	Default
17..0	D[17:0]	PWM data	0000h, 00

Table 4-88: PWM1D Register

4.6.3 DSP0 EGPIO

The **GPA[3:0]** and **GPB[3:0]** are shared between the Host Controller and the DSP0. The DSP has its own set of registers which control the functionality of these pins and they are implemented as extended GPIOs. On revision G and greater, when the DDR bit of the Host Controller of the corresponding GPIO pin is low, the DSP has control of the direction of the GPIO pin. When the GPIO is configured as an output by either the DSP or the Host Controller, the IPOT register controls the output type (open-drain or driven in both directions).

The GPIO have the alternate function of Asynchronous Source Ports. Each set of four GPIO are grouped and can be an Asynchronous Source Port, or GPIO. If **DSnC.BPI[6:0]** is set to all zeros, then the Asynchronous Source Port is disabled and the four pins may be used as GPIO. Therefore, **DS0C.BPI[6:0]** enables Source Port A0 or **GPA[3:0]**, and **DS1C.BPI[6:0]** enables Source Port A1 or **GPB[3:0]**. For DSP1, **DS0C.BPI[6:0]** enables Source Port B0 or **GPC[3:0]**, and **DS1C.BPI[6:0]** enables Source Port B1 or **GPD[3:0]**.

In the event that both the DDR of the corresponding pin of the Host Controller and the DSP are set, the Host Controller has priority of the GPIO pin over the DSP. In this case, the data of the Host Controller is presented on the GPIO pin. DSP0's Asynchronous Source Ports have higher priority than EGPIO logic (either the Host Controller's or the DSPs).

1Dh EGPD Enhanced GPIO Data DSP0

Bit	Label	Description	Default
17..8	rsvd	Reserved. Write to 0.	00h, 00
7..4	GPDB[3:0]	Data bits for **GPB[3:0]** pins.	*
3..0	GPDA[3:0]	Data bits for **GPA[3:0]** pins	*

* Initial value determined by state of corresponding pin.

Table 4-89: EGPD Register

GPDxn General Purpose IO bits for the **GPxn** pins. These bits are GPIO when **DSnC.BPI[6:0]** = 000000. The Host Controller can also control these pins and override the DSP for output control. **GPDA0** can generate an interrupt if enabled (**IER.IEGP** set).

When configured as an output (**EDD.GPOxn** set), the value written to this bit is driven out the corresponding pin when the output is enabled (**ISOD.GPSDxn** clear), based on output driver type (**IPOT.GPPTxn**). Data read is directly from the pin.

When configured as an input (**EDD.GPOxn** clear), the polarity is controlled by **IPOT.GPPTxn** and the data can be sticky (**ISOD.GPSDxn** set) or not. When sticky, the register bit is cleared by writing it to 0 (assuming the pin is inactive).

1Eh EDD Enhanced GPIO Data Direction DSP0

Bit	Label	Description	Default
17..8	rsvd	Reserved. Write to 0.	00h, 00
7..4	GPOB[3:0]	Output enable for **GPB[3:0]** pins	0000
3..0	GPOA[3:0]	Output enable for **GPA[3:0]** pins	0000

Table 4-90: EDD Register

GPOxn Output Enable for corresponding **GPxn** pin. When **GPxn** is configured as a GPIO and the corresponding Host Controller DDR bit is clear, this corresponding bit determines whether the pin is a general purpose input or output. The Host Controller can override the DSP for output control through its DDR register.

0 - Corresponding **GPxn** pin is an input.
IPOT.GPPTxn determines input polarity.
ISOD.GPSDxn determines whether the input is sticky or not.

1 - Corresponding **GPxn** pin is an output.
IPOT.GPPTxn determines output driver type (CMOS or open-drain).
ISOD.GPSDxn is an output disable/enable.

1Fh IPOT EGPIO Input Polarity/Output Type DSP0

Bit	Label	Description	Default
17..8	rsvd	Reserved. Write to 0.	00h, 00
7..4	GPPTB[3:0]	Input polarity or output driver type for **GPB[3:0]** pins	0000
3..0	GPPTA[3:0]	Input polarity or output driver type for **GPA[3:0]** pins	0000

Table 4-91: IPOT Register

GPPTxn Input polarity or output driver type for corresponding **GPxn** pin when configured as a GPIO.
When **GPxn** pin is configured as an input (**EDD.GPOxn** clear), this bit sets the polarity:
0 - Active high input (non-inverting) or high-pulse capture, if sticky
1 - Active low input (inverting) or low-pulse capture, if sticky
When **GPxn** pin is configured as an output for DSP0 (**EDD.GPOxn** set) or the Host Controller (**EDD1.GPOxn** set on rev. G or greater), this bit sets the output driver type:
0 - Open-drain output (only driven low)
1 - CMOS output (driven both high and low)

3Ah ISOD EGPIO Input Sticky/Output Disable DSP0

Bit	Label	Description	Default
17..8	rsvd	Reserved. Write to 0.	00h, 00
7..4	GPSDB[3:0]	Input sticky or output disable for **GPB[3:0]** pins	0000
3..0	GPSDA[3:0]	Input sticky or output disable for **GPA[3:0]** pins	0000

Table 4-92: ISOD Register

GPSDxn Input sticky or output disable for corresponding **GPxn** pin when configured as a GPIO.
When **GPxn** pin is configured as an input (**EDD.GPOxn** clear), **GPSDxn** sets sticky or not.
0 - **EGPD.GPDxn** reflects the **GPxn** pin, after polarity (**IPOT.GPPTxn**).
1 - **EGPD.GPDxn** is sticky and captures a high or low pulse (based on **IPOT.GPPTxn**) on **GPxn**. Once set, cleared by writing **EGPD.GPDxn** to zero when input pin is in the inactive state.
When **GPxn** pin is configured as an output (**EDD.GPOxn** set), this bit controls output enable:
0 - The **GPxn** pin is enabled and driven, based on **IPOT.GPPTxn**.
1 - The **GPxn** pin is disabled, high impedance.

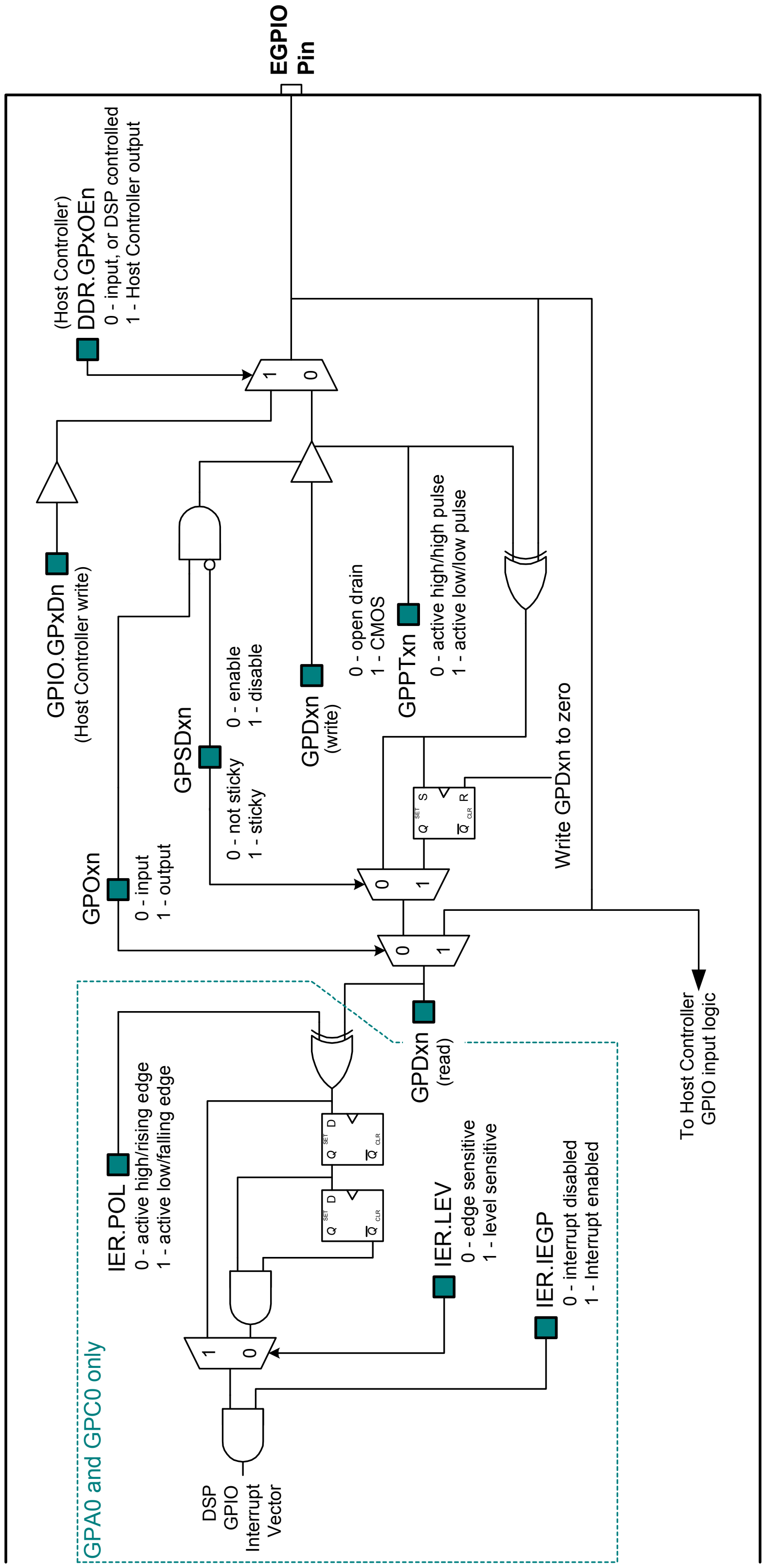


Figure 4-15: DSP EGPIO Conceptual Logic

4.6.4 DSP1 EGPIO

The **GPC[3:0]** and **GPD[3:0]** are shared between the Host Controller and the DSP1. The DSP has its own set of registers which control the functionality of these pins and they are implemented as extended GPIOs. On revision G or greater, when the DDR bit of the Host Controller of the corresponding GPIO pin is low, the DSP has control of the direction of the GPIO pin. When the GPIO is configured as an output by either the DSP or the Host Controller, the IPOT register controls the output type (open-drain or driven in both directions).

The GPIO have the alternate function of Asynchronous Source Ports. Each set of four GPIO are grouped and can be an Asynchronous Source Port, or GPIO. If **DSnC.BPI[6:0]** is set to all zeros, then the Asynchronous Source Port is disabled and the four pins may be used as GPIO. Therefore, **DS0C.BPI[6:0]** enables Source Port B0 or **GPC[3:0]**, and **DS1C.BPI[6:0]** enables Source Port B1 or **GPD[3:0]**.

In the event that both the DDR of the corresponding pin of the Host Controller and the DSP are set, the Host Controller has priority of the GPIO pin over the DSP. In this case, the data of the Host Controller is presented on the GPIO pin. DSP1's Asynchronous Source Ports have higher priority than EGPIO logic (either the Host Controller's or the DSPs).

1Dh EGPD Enhanced GPIO Data DSP1

Bit	Label	Description	Default
17..8	rsvd	Reserved. Write to 0.	00h, 00
7..4	GPDD[3:0]	Data bits for **GPD[3:0]** pins.	*
3..0	GPDC[3:0]	Data bits for **GPC[3:0]** pins	*

* Initial value determined by state of corresponding pin.

Table 4-93: EGPD Register

GPDxn General Purpose IO bits for the **GPxn** pins. These bits are GPIO when **DSnC.BPI[6:0]** = 000000. The Host Controller can also control these pins and override the DSP for output control. **GPDC0** can generate an interrupt if enabled (**IER.IEGP** set).

When configured as an output (**EDD.GPOxn** set), the value written to this bit is driven out the corresponding pin when the output is enabled (**ISOD.GPSDxn** clear), based on output driver type (**IPOT.GPPTxn**). Data read is directly from the pin.

When configured as an input (**EDD.GPOxn** clear), the polarity is controlled by the **IPOT.GPPTxn** bit and the data can be sticky (**ISOD.GPSDxn set**) or not. When sticky, the register bit is cleared by writing it to 0 (assuming the pin is inactive).

1Eh EDD Enhanced GPIO Data Direction DSP1

Bit	Label	Description	Default
17..8	rsvd	Reserved. Write to 0.	00h, 00
7..4	GPOD[3:0]	Output enable for **GPD[3:0]** pins	0000
3..0	GPOC[3:0]	Output enable for **GPC[3:0]** pins	0000

Table 4-94: EDD Register

GPOxn Output Enable for corresponding **GPxn** pin. When **GPxn** is configured as a GPIO and the corresponding Host Controller DDR bit is clear, this corresponding bit determines whether the pin is a general purpose input or output. The Host Controller can override the DSP for output control through its DDR register.

0 - Corresponding **GPxn** pin is an input.
IPOT.GPPTxn determines input polarity.
ISOD.GPSDxn determines whether the input is sticky or not.

1 - Corresponding **GPxn** pin is an output.
IPOT.GPPTxn determines output driver type (CMOS or open-drain).
ISOD.GPSDxn is an output disable/enable.

1Fh IPOT EGPIO Input Polarity/Output Type DSP1

Bit	Label	Description	Default
17..8	rsvd	Reserved. Write to 0.	00h, 00
7..4	GPPTD[3:0]	Input polarity or output driver type for **GPD[3:0]** pins	0000
3..0	GPPTC[3:0]	Input polarity or output driver type for **GPC[3:0]** pins	0000

Table 4-95: IPOT Register

GPPTxn Input polarity or output driver type for corresponding **GPxn** pin when configured as a GPIO.

When **GPxn** pin is configured as an input (**EDD.GPOxn** clear), this bit sets the polarity:
0 - active high input (non-inverting) or high-pulse capture, if sticky
1 - active low input (inverting) or low-pulse capture, if sticky

When **GPxn** pin is configured as an output for DSP1 (**EDD.GPOxn** set) or the Host Controller (**EDD1.GPOxn** set on rev. G or greater), this bit sets the output driver type:
0 - Open-drain output (only driven low)
1 - CMOS output (driven both high and low)

3Ah ISOD EGPIO Input Sticky/Output Disable DSP1

Bit	Label	Description	Default
17..8	rsvd	Reserved. Write to 0.	00h, 00
7..4	GPSDD[3:0]	Input sticky or output disable for **GPD[3:0]** pins	0000
3..0	GPSDC[3:0]	Input sticky or output disable for **GPC[3:0]** pins	0000

Table 4-96: ISOD Register

GPSDxn Input sticky or output disable for corresponding **GPxn** pin when configured as a GPIO.

When **GPxn** pin is configured as an input (**EDD.GPOxn** clear), **GPSDxn** sets sticky or not.
0 - **EGPD.GPDxn** reflects the **GPxn** pin, after polarity (**IPOT.GPPTxn**).
1 - **EGPD.GPDxn** is sticky and captures a high or low pulse (based on **IPOT.GPPTxn**) on **GPxn**. Once set, cleared by writing **EGPD.GPDxn** to zero when input pin is in the inactive state.

When **GPxn** pin is configured as an output (**EDD.GPOxn** set), this bit controls output enable:
0 - The **GPxn** pin is enabled and driven, based on **IPOT.GPPTxn**.
1 - The **GPxn** pin is disabled, high impedance.

4.7 Register Summary

DSP Processor registers are internal Gazelle registers as well as all registers connected to the DSP bus. They include the control and data registers for the COM Port to the Host Controller, and the data registers to the Routing bus (via the MOST Routing Port), the local timers, and the global timing register. The DSP0 also connects to the data registers of the external data memory port and the PWM data registers.

4.7.1 DSP0 I/O Registers

Name	Label	Addr.	Description	Page
DSP0:				
Program Counter	PC	00h	Program Counter, I/O address for subroutine pop	145
Interrupt Program Ctr.	PCI	01h	Program Counter, I/O address for interrupt pop	145
Interrupt Shadow PC	ISPC	02h	PC stack register for interrupts	145
Subroutine Shadow PC	SSPC	03h	PC stack register for subroutines	145
Interrupt Enable	IER	04h	Enables individual interrupt signals	148
Status Register	SR	05h	Contains DSP Status and ACC guard bits	151
Shadow Status Register	SSR	06h	SR stack register for interrupts and subs	151
Trap Shadow Status Register	TSSR	0Ch	SR stack register for `TRAP` instructions	152
Shadow ACC Low	SACCL	07h	ACC low stack register for interrupts and subs	151
Shadow ACC High	SACCH	08h	ACC high stack register for interrupts and subs	151
Count	CNT	09h	Count value for loops and repeats	145
End address	END	0Ah	End address for hardware looping	146
Start address	STRT	0Bh	Start address for hardware looping	146
DSP0 Peripherals:				
DSP Com Status	DCS	17h	Flags for Control/DSP inter-bus transfers	156
DSP Com Data	DCD	18h	Bi-directional data between Control and DSP buses	156
DSP Debug Com Status	DDCS	10h	Flags for Control/DSP inter-bus transfers	156
DSP Debug Com Data	DDCD	11h	Bi-directional data between Control and DSP buses	156
MOST Routing Port Receive	DR	19h	Data from MOST processor (read only)	154
MOST Routing Port Transmit	DX	19h	Data to MOST processor (write only)	155
MOST Routing Port Receive 1	DR1	1Bh	2nd Data from MOST processor (read only)	155
MOST Routing Port Transmit 1	DX1	1Bh	2nd Data to MOST processor (write only)	155
Global Timer	GTR	1Ah	Global timing flags for processor synchronization	154
PWM DAC Data	PWMD	1Ch	Output data for PWM DAC	184
PWM1 DAC Data	PWM1D	4Ch	Output data for PWM1 DAC	184
DAC0 Volume Control	ADAC0	50h	DAC0 attenuation control	173
DAC1 Volume Control	ADAC1	51h	DAC1 attenuation control	173
DAC2 Volume Control	ADAC2	52h	DAC2 attenuation control	173
DAC3 Volume Control	ADAC3	53h	DAC3 attenuation control	173
MPX Volume Control	GMPX	54h	MPX ADC gain control	174
MIC Volume Control	GMIC	55h	MIC ADC gain control	174
Audio ADC Left Volume Control	GADL	56h	Audio ADC left gain control	174
Audio ADC Right Volume Control	GADR	57h	Audio ADC right gain control	174

Table 4-97: DSP0 Register Summary

Name	Label	Addr.	Description	Page
Inter-DSP FIFO Port:				
FIFO Port Local Status	DFLS	12h	DSP0's FIFO Status and Control	157
FIFO Port Start Data Write	DFSD	13h	Start write FIFO data port (write only)	157
FIFO Port Write Data	DFWD	14h	write FIFO data port (write only)	158
FIFO Port Read Data	DFRD	14h	read FIFO data port (read only)	158
FIFO Port Far Status	DFFS	15h	DSP1's FIFO status (read only)	157
Async. Source Port Timer:				
DSP Timer Control	DTC	3Fh	DSP0's Async. Source Port Timer control	160
DSP Divider 0	DDIV0	3Bh	Async Source Port Timer divider 0	159
DSP Divider 1	DDIV1	3Ch	Async Source Port Timer divider 1	159
DSP Capture 0	DCAP0	3Dh	Async Source Port Timer capture 0	160
DSP Capture 1	DCAP1	3Eh	Async Source Port Timer capture 1	160
DSP Preset 1	DPRE1	4Bh	Positioning of S/PDIF preambles	160
Async. Source Ports:				
DSP Source Port 0 Control	DS0C	48h	Async. Source Port (ASP) A0 control	167
DSP Source Port 0, Receive 0	DS0R0	40h	ASP A0 receive LS word	163
DSP Source Port 0, Transmit 0	DS0X0	40h	ASP A0 transmit LS word	165
DSP Source Port 0, Receive 1	DS0R1	41h	ASP A0 receive lower middle word	164
DSP Source Port 0, Transmit 1	DS0X1	41h	ASP A0 transmit lower middle word	165
DSP Source Port 0, Receive 2	DS0R2	42h	ASP A0 receive upper middle word	164
DSP Source Port 0, Transmit 2	DS0X2	42h	ASP A0 transmit upper middle word	165
DSP Source Port 0, Receive 3	DS0R3	43h	ASP A0 receive MS word	164
DSP Source Port 0, Transmit 3	DS0X3	43h	ASP BA0 transmit MS word	165
DSP Source Port 1 Control	DS1C	49h	Async. Source Port A1 control	168
DSP Source Port 1, Receive 0	DS1R0	44h	ASP A1 receive LS word	164
DSP Source Port 1, Transmit 0	DS1X0	44h	ASP A1 transmit LS word	166
DSP Source Port 1, Receive 1	DS1R1	45h	ASP A1 receive lower middle word	164
DSP Source Port 1, Transmit 1	DS1X1	45h	ASP A1 transmit lower middle word	166
DSP Source Port 1, Receive 2	DS1R2	46h	ASP A1 receive upper middle word	165
DSP Source Port 1, Transmit 2	DS1X2	46h	ASP A1 transmit upper middle word	166
DSP Source Port 1, Receive 3	DS1R3	47h	ASP A1 receive MS word	165
DSP Source Port 1, Transmit 3	DS1X3	47h	ASP A1 transmit MS word	166
EGPIO:				
DSP0 EGPIO Data	EGPD	1Dh	**GPA[3:0]** and **GPB[3:0]** data	185
DSP0 EGPIO Data Direction	EDD	1Eh	**GPA[3:0]** and **GPB[3:0]** data direction (in/out)	185
EGPIO Input Polarity, Output Type	IPOT	1Fh	**GPA[3:0], GPB[3:0]** input polarity or output type	186
EGPIO In Sticky, Out Disable	ISOD	3Ah	**GPA[3:0], GPB[3:0]** input sticky or output disable	186

Table 4-97: DSP0 Register Summary (Continued)

Name	Label	Addr.	Description	Page
DSP0 External Data Port:				
Offset 0	OFF0	20h	Provides offset from base0	178
Offset 0, start read	OFF0+-	21h	A write loads off0 and initiates read	178
Offset 0, str rd, post inc	OFF0++	22h	A wr loads off0, starts read with post inc	178
Offset 1	OFF1	24h	Provides offset from base1	178
Offset 1, start read	OFF1+-	25h	A write loads off1 and initiates read	178
Offset 1, str rd, post inc	OFF1++	26h	A wr loads off1, starts read with post inc	178
Read 0	RD0	28h	Contains data read from external memory	179
Read 0, start read	RD0+-	29h	Read external data and start a new read	179
Read 0, start rd, post inc	RD0++	2Ah	Read data, start a new read with post inc	179
Read 1	RD1	2Ch	Contains data read from external memory	179
Read 1, start read	RD1+-	2Dh	Read external data and start a new read	179
Read 1, start rd, post inc	RD1++	2Eh	Read data, start a new read with post inc	179
Base Address 0 Low	BSL0	30h	Low base address for external memory access	179
Base Address 0 High	BSH0	31h	High base address for external memory access	180
Base Address 1 Low	BSL1	32h	Low base address for external memory access	180
Base Address 1 High	BSH1	33h	High base address for external memory access	180
Write 0	WR0+-	34h	Write data, start a new write	180
Write 0, post inc	WR0++	35h	Write data, start a new write with post inc	180
Write 1	WR1+-	36h	Write data, start a new write	180
Write 1, post inc	WR1++	37h	Write data, start a new write with post inc	180
Modulo 0	MOD0	38h	Modulo value for post increment of base0	181
Modulo 1	MOD1	39h	Modulo value for post increment of base0	181

Table 4-97: DSP0 Register Summary (Continued)

4.7.2 DSP1 I/O Registers

Name	Label	Addr.	Description	Page
DSP1:				
Program Counter	PC	00h	Program Counter, I/O address for subroutine pop	145
Interrupt Program Ctr.	PCI	01h	Program Counter, I/O address for interrupt pop	145
Interrupt Shadow PC	ISPC	02h	PC stack register for interrupts	145
Subroutine Shadow PC	SSPC	03h	PC stack register for subroutines	145
Interrupt Enable	IER	04h	Enables individual interrupt signals	148
Status Register	SR	05h	Contains DSP Status and ACC guard bits	151
Shadow Status Register	SSR	06h	SR stack register for interrupts and subs	151
Trap Shadow Status Register	TSSR	0Ch	SR stack register for `TRAP` instructions	152
Shadow ACC Low	SACCL	07h	ACC low stack register for interrupts and subs	151
Shadow ACC High	SACCH	08h	ACC high stack register for interrupts and subs	151
Count	CNT	09h	Count value for loops and repeats	145
End address	END	0Ah	End address for hardware looping	146
Start address	STRT	0Bh	Start address for hardware looping	146
DSP1 Peripherals:				
DSP Com Status	DCS	17h	Flags for Control/DSP inter-bus transfers	156
DSP Com Data	DCD	18h	Bi-directional data between Control and DSP buses	156
DSP Debug Com Status	DDCS	10h	Flags for Control/DSP inter-bus transfers	156
DSP Debug Com Data	DDCD	11h	Bi-directional data between Control and DSP buses	156
MOST Routing Port Receive	DR	19h	Data from MOST Processor (read only)	154
MOST Routing Port Transmit	DX	19h	Data to MOST Processor (write only)	155
MOST Routing Port Receive 1	DR1	1Bh	2nd Data from MOST processor (read only)	155
MOST Routing Port Transmit 1	DX1	1Bh	2nd Data to MOST processor (write only)	155
Global Timer	GTR	1Ah	Global timing flags for processor synchronization	154
DAC0 Volume Control	ADAC0	50h	Analog attenuation setting for DAC0 output	173
DAC1 Volume Control	ADAC1	51h	Analog attenuation setting for DAC1 output	173
DAC2 Volume Control	ADAC2	52h	Analog attenuation setting for DAC2 output	173
DAC3 Volume Control	ADAC3	53h	Analog attenuation setting for DAC3 output	173
MPX Volume Control	GMPX	54h	Analog gain setting for MPX ADC input	174
MIC Volume Control	GMIC	55h	Analog gain setting for MIC ADC input	174
Audio ADC Left Volume Control	GADL	56h	Analog gain setting for Left Audio ADC input	174
Audio ADC Right Volume Control	GADR	57h	Analog gain setting for Right Audio ADC input	174
Inter-DSP FIFO Port:				
FIFO Port Local Status	DFLS	12h	DSp1’s FIFO Status and Control	157
FIFO Port Start Data Write	DFSD	13h	Start write FIFO data port (write only)	158
FIFO Port Write Data	DFWD	14h	Write FIFO data port (write only)	158
FIFO Port Read Data	DFRD	14h	Read FIFO data port (read only)	158
FIFO Port Far Status	DFFS	15h	DSP0’s FIFO status (read only)	157

Table 4-98: DSP1 Register Summary

Name	Label	Addr.	Description	Page
Async. Source Port Timer:				
DSP Timer Control	DTC	3Fh	DSP1’s Async. Source Port Timer control	160
DSP Divider 0	DDIV0	3Bh	Async. Source Port Timer Divider 0	159
DSP Divider 1	DDIV1	3Ch	Async. Source Port Timer Divider 1	159
DSP Capture 0	DCAP0	3Dh	Async. Source Port Timer Capture 0	160
DSP Capture 1	DCAP1	3Eh	Async. Source Port Timer Capture 0	160
Async. Source Ports:				
DSP Source Port 0 Control	DS0C	48h	Async. Source Port (ASP) B0 control	167
DSP Source Port 0, Receive 0	DS0R0	40h	ASP B0 receive LS word	163
DSP Source Port 0, Transmit 0	DS0X0	40h	ASP B0 transmit LS word	165
DSP Source Port 0, Receive 1	DS0R1	41h	ASP B0 receive lower middle word	164
DSP Source Port 0, Transmit 1	DS0X1	41h	ASP B0 transmit lower middle word	165
DSP Source Port 0, Receive 2	DS0R2	42h	ASP B0 receive upper middle word	164
DSP Source Port 0, Transmit 2	DS0X2	42h	ASP B0 transmit upper middle word	165
DSP Source Port 0, Receive 3	DS0R3	43h	ASP B0 receive MS word	164
DSP Source Port 0, Transmit 3	DS0X3	43h	ASP B0 transmit MS word	165
DSP Source Port 1 Control	DS1C	49h	Async. Source Port B1 control	168
DSP Source Port 1, Receive 0	DS1R0	44h	ASP B1 receive LS word	164
DSP Source Port 1, Transmit 0	DS1X0	44h	ASP B1 transmit LS word	166
DSP Source Port 1, Receive 1	DS1R1	45h	ASP B1 receive lower middle word	164
DSP Source Port 1, Transmit 1	DS1X1	45h	ASP B1 transmit lower middle word	166
DSP Source Port 1, Receive 2	DS1R2	46h	ASP B1 receive upper middle word	165
DSP Source Port 1, Transmit 2	DS1X2	46h	ASP B1 transmit upper middle word	166
DSP Source Port 1, Receive 3	DS1R3	47h	ASP B1 receive MS word	165
DSP Source Port 1, Transmit 3	DS1X3	47h	ASP B1 transmit MS word	166
EGPIO:				
DSP1 EGPIO Data	EGPD	1Dh	**GPC[3:0]** and **GPD[3:0]** data	188
DSP1 EGPIO Data Direction	EDD	1Eh	**GPC[3:0]** and **GPD[3:0]** data direction (in/out)	189
EGPIO Input Polarity, Output Type	IPOT	1Fh	**GPC[3:0], GPD[3:0]** input polarity or output type	189
EGPIO In Sticky, Out Disable	ISOD	3Ah	**GPC[3:0], GPD[3:0]** input sticky or output disable	189

Table 4-98: DSP1 Register Summary (Continued)

5 Source Converters

The Source Converters consist of the ADCs and DACs that reside on the Routing bus. The MOST Processor transfers Source Converter data between the MOST Network and the other Source Peripherals.

The Stereo Audio ADCs have an input 3-to-1 mux; whereas the MIC and MPX ADCs are mono and have a single input. The Programmable Gain of each ADC is controlled by the Host Controller or the DSPs. The Quad Audio DAC outputs can be single-ended or differential with a drive capability of 2 kΩ. The Programmable Attenuation of each DAC is controlled by the Host Controller or either DSP. The value read from the volume control register will be the last one written. The individual ADC gain registers and DAC attenuation registers are described under *Source Converter Control* in the *Host Controller* Section and under *Source Converter Volume Control* in the *Digital Signal Processor* Section.

5.1 Quad Audio DACs

Four audio-bandwidth differential DACs are included on the chip. The DACs accept 16-bit, two's complement numbers. The typical connections are DAC0 for front left, DAC1 for front right, DAC2 for rear left, DAC3 for rear right. The programmable attenuator stage after the DACs can attenuate the analog output from 0 to -31 dB in 1 dB steps. Analog attenuation attenuates the noise with the signal, which mimics analog potentiometers.

The base-band source data is interpolated to 96xFs. This over-sampled source data is modulated into a 1-bit data stream by the 3rd-order delta-sigma modulator. The switch-capacitor filter performs a low-pass function on the 1-bit data and converts it into an analog signal. The resultant analog signal passes through the programmable attenuation and output stage. The DAC output stage has a drive capability of 2 kΩ, when referenced to ground.

Each DAC can be powered down independently by the Host Controller. For additional information on the power down mode and programmable attenuation control, see the *Source Converter Control* in the *Host Controller* Section.

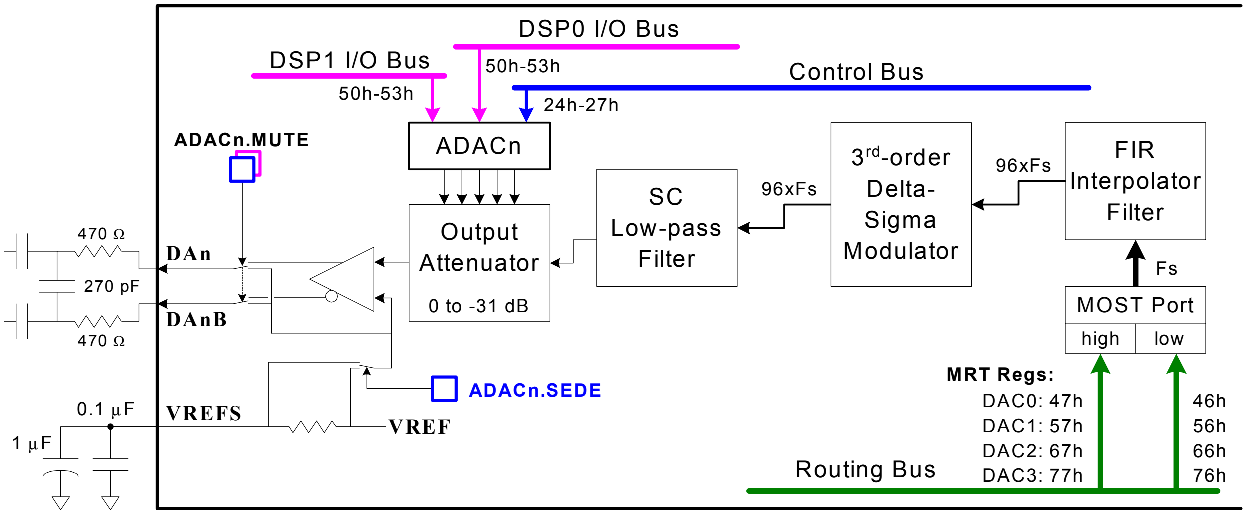


Figure 5-1: DAC Block Diagram

Common-mode rejection can be improved by connecting 0.1 µF and 1 µF bypass capacitors between the **VREFS** pin and AGND, and setting the **ADACn.SEDE** bit for the particular DAC. Although these capacitors help in differential mode, significant dynamic range improvements are seen when using the DACs in a single-ended fashion.

5.2 Microphone ADC

The microphone input signal is amplified with the programmable input gain stage. The amplified signal is over-sampled by a 2nd-order delta-sigma modulator operating at 96xFs with an over-sampling ratio of 384. The decimation filter decimates and low-pass filters the digital signal to 1/4xFs. The output is a MSB-aligned, 12-bit, two's complement number in a 16-bit field.

The programmable gain stage for the Microphone ADC can amplify the analog input from 0 to 15 dB in 1 dB steps. The Microphone ADC can be powered down to conserve power. For additional information on the power down mode and programmable gain control, see the *Source Converter Control* in the *Host Controller* Section.

The Figure below illustrates the block diagram of the Mic ADC.

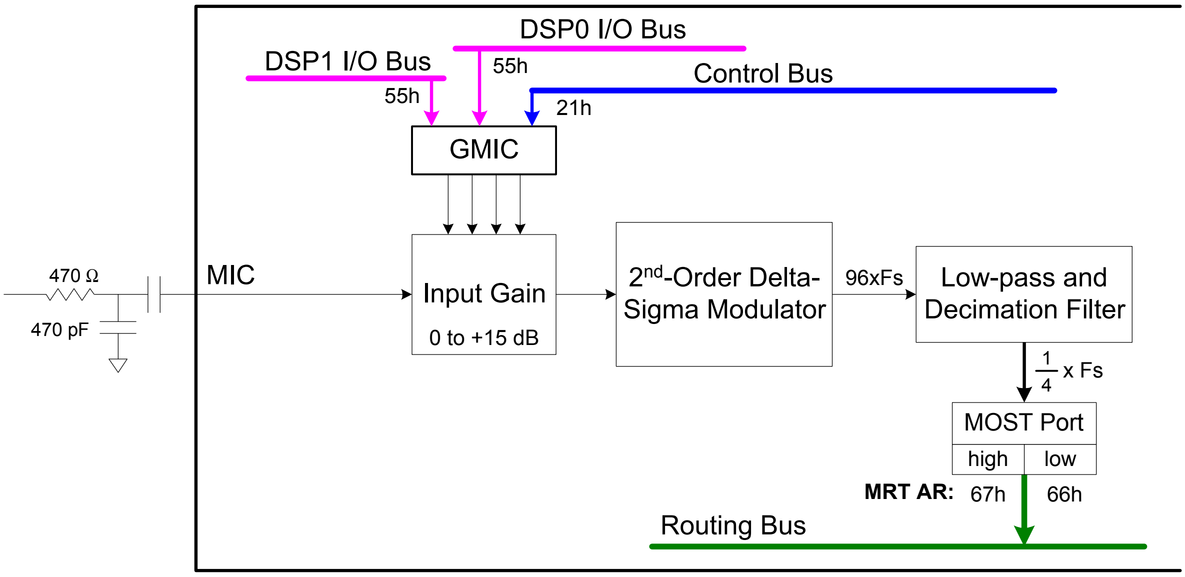


Figure 5-2: Mic ADC

5.3 MPX ADC

The MPX input signal is amplified with the programmable input gain stage. The amplified signal is over-sampled by a 4th-order cascaded delta-sigma modulator operating at 96xFs with an over-sampling ratio of 24. The decimation filter decimates and low-pass filters the digital signal to 4xFs. The output is a 16-bit two's complement number.

The programmable gain stage for the MPX ADC can amplify the analog input from 0 to 26 dB in 1 dB steps. The MPX ADC can be powered down to conserve power. For more information on the power-down mode and programmable gain control, see the *Source Converter Control* in the *Host Controller* Section.

The Figure below illustrates the block diagram of the MPX ADC.

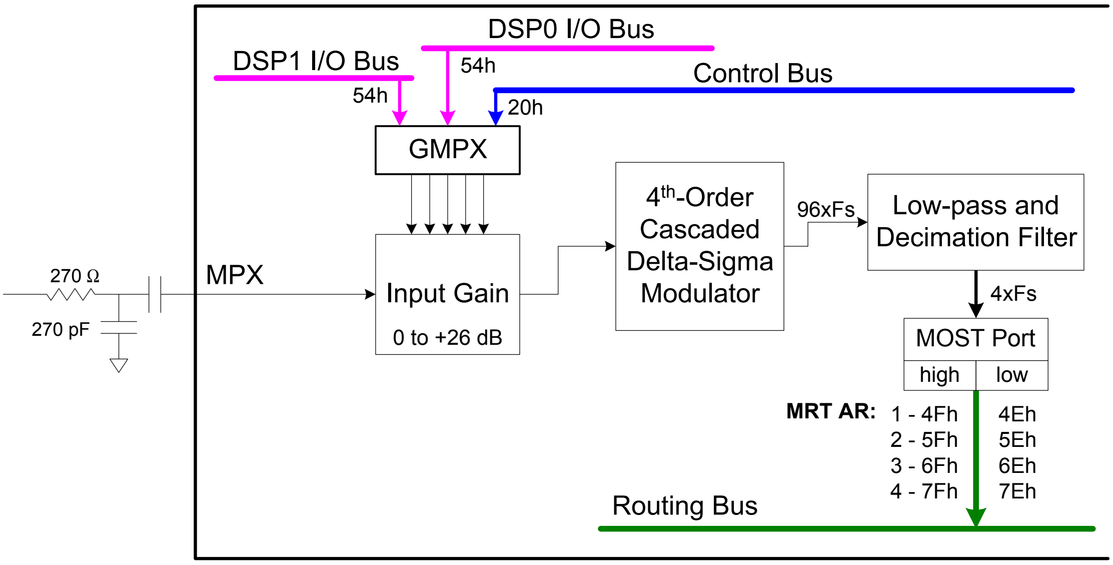


Figure 5-3: MPX ADC

5.4 Stereo Audio ADCs

An input mux supports a selection of three external stereo audio sources: **AD0L/AD0R**, **AD1L/AD1R**, and **AD2L/AD2R**. The stereo audio input signals are amplified with the programmable input gain stages. The amplified signals are over-sampled by two 3rd-order cascaded delta-sigma modulators operating at 96xFs. The decimation filters decimate and low-pass filter the digital signals to Fs. The output is a 16-bit two's complement number.

The programmable gain stages for the Stereo Audio ADC can amplify the analog inputs from 0 to 15 dB in 1 dB steps. The Stereo Audio ADC can be powered down to conserve power. For additional information on the power down mode and programmable gain control, see the *Source Converter Control* in the *Host Controller* Section.

The Figure below illustrates the block diagram of the Stereo Audio ADCs.

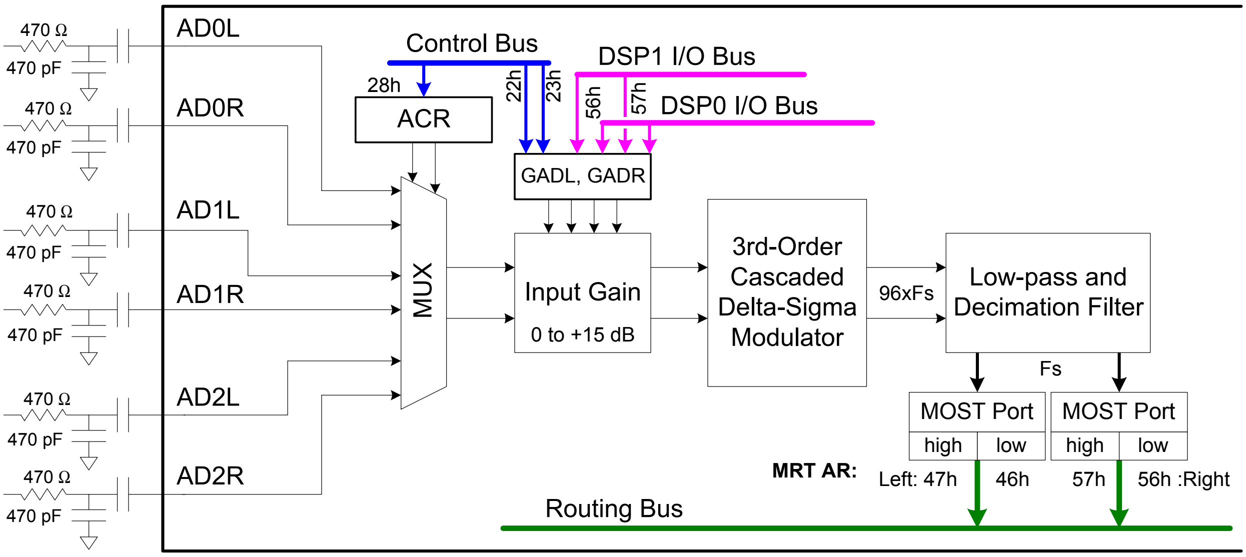


Figure 5-4: Stereo Audio ADCs

6 Electrical Characteristics

All pins configured as digital inputs or must not be left floating; therefore, they must be driven, have pull-ups or pull-downs, or be directly attached to power or ground.

6.1 Absolute Maximum Ratings

Parameter (Note 1)	Min	Max	Unit
Storage Temperature	-65	150	°C
Junction Temperature		165	°C
Power Supply Voltage	-0.5	4.0	V
DC Current to any Pin Except Power		±10	mA
Maximum input voltage: (Note 2)			
Digital I/O pins ($D_{I/O}$ and $D_{I/OD}$) - configured as outputs	-0.3	VDDD + 0.3	V
Digital I/O pins ($D_{I/O}$ and $D_{I/OD}$) - configured as inputs	-0.3	6	V
Digital Input pins (D_{IN} pins)	-0.3	6	V
Analog Input pins	-0.3	VDDA + 0.3	V

Notes:

1. Operation at or above these limits may damage the device
2. Table 7-1 lists which pins are input-only (D_{IN}) and which are not ($D_{I/O}$ or $D_{I/OD}$).

6.2 Recommended Operating Conditions

Parameter	Symbol	Min	Max	Unit
Junction Temperature (Note 1)	T_J	-40	150	°C
Power Supply Voltage (Note 2)		3.135	3.465	V
Recommended input voltage:				
Digital I/O pins ($D_{I/O}$ and $D_{I/OD}$)		0	5.5	V
Digital Input pins (D_{IN} pins)		0	5.5	V
Analog Input pins		0	VDDA	V
Operating Sample Frequency		38	48	kHz
Flash Program/Erase Cycles: T_J = 100 °C			20000	cycles
T_J = 125 °C			1000	cycles

Note:

1. When using the ETQFP package on a properly designed board, the OS8805 can support ambient temperatures of up to 120 °C. See Section 8.2 for more information.
2. All power pins, including VDDD and VDDA

6.3 Thermal Characteristics

Parameter	Symbol	Value	Unit
Typical Junction to Package (MQFP package):			
two-layer PCB (no power/ground planes)	Ψ_{JT}	3.0	°C/W
multi-layer PCB (with power and ground planes)		4.0	°C/W
Typical Junction to Package (ETQFP package): multi-layer PCB (with power and ground planes)	Ψ_{JT}	0.85	°C/W
Typical Junction to Ambient (MQFP package):			
two-layer PCB (no power/ground planes)	θ_{JA}	40	°C/W
multi-layer PCB (with power and ground planes)		38	°C/W
Typical Junction to Ambient (ETQFP package): multi-layer PCB (with power and ground planes)	θ_{JA}	15	°C/W

6.4 DC Characteristics

T_J = -40 to 150 °C; VDDD,VDDA = 3.3 V ±5 %; GNDD,GNDA = 0.0 V; PLL locked at 44.1 kHz; unless otherwise noted.

Parameter	Symbol	Min	Typ	Max	Unit	Test Conditions
Low-Level Input Voltage: XTI pin All other digital pins	V_{IL}			0.1×VDDD 0.8	V V	
High-Level Input Voltage: XTI pin All other digital pins	V_{IH}	0.9×VDDD 2.0			V V	
Input Leakage Current	I_L			±10	µA	0 < Vin < VDDD
Slew-Limited Outputs: (Note 1)						
Low Level Output Voltage	V_{OL}			0.4	V	I_{OL} = 2.4 mA
High Level Output Voltage	V_{OH}	VDDD-1.0			V	I_{OH} = -2.4 mA
High-Drive Outputs: (Note 2)						
Low Level Output Voltage	V_{OL}			0.4	V	I_{OL} = 4.8 mA
High Level Output Voltage	V_{OH}	VDDD-1.0			V	I_{OH} = -4.8 mA
Power Supply Current:						
Total Power Supply Current	$I_D + I_A$		370	610	mA	(Note 3)
Digital Supply current: Each DSP enabled ADCs/DACs enabled Host Controller w/o DSPs	I_D		95 35 85		mA mA mA	Outputs unloaded **FPCR.RUN** set
Digital Supply variation: (Note 4)			0.18 133 ±5		mA/°C mA/V %	over temperature over supply over process
Analog Supply current: All ADCs/DACs on All ADCs/DACs off	I_A		90 15		mA mA	Outputs unloaded
Analog Supply variation: (Note 4)			0.03 5 ±9		mA/°C mA/V %	over temperature over supply over process
VDDA0 (PLL supply) current: PSRR:			9 60		mA ps/mV	induced jitter
Power Down current	$I_D + I_A$		4	25	mA	**CMCS.PD** set, outputs unloaded
Digital Input Pin Capactiance: RX Other Digital Inputs				5 7	pF pF	

Notes:

1. **GPA[3:0], GPB[3:0], GPC[3:0], GPD[3:0], IOA[3:0], IOB[3:0], IOC[5:0], IOG1, IOG3**. For pins configured as open-drain outputs, V_{OH} does not apply.
2. **IOD[1:0], IOE[15:0], IOF[10:0], IOG0, IOG2, IOG4**. For pins configured as open-drain outputs, V_{OH} does not apply.
3. Both DSPs running at 1536Fs; all ADCs and DACs enabled; Fs = 48.1 kHz, and VDDD = VDDA = 3.465, outputs unloaded. The current seen in a particular application will vary significantly based on peripherals running, their speed of operation, and the external connections to the chip. For a given application, the current should be measured, and then the variation for analog and digital supplies over voltage and temperature should be added. Lastly, the value should be multiplied by (one plus) the process variation to determine the maximum current for a given application.
4. The total supply current variation over process will not exceed ±6 %.

6.5 Switching Characteristics

T_J = -40 to 150 °C; VDDD,VDDA = 3.3 V ±5 %; GNDD,GNDA = 0.0 V; PLL locked at 44.1 kHz,
Load Capacitance = 30 pF. unless otherwise noted.

Parameter	Symbol	Min	Typ	Max	Unit	Test Conditions
Slew-Limited Outputs: (Note 1)						
Rise Time	t_{slr}			20	ns	10 % to 90 %
Fall Time	t_{slf}			20	ns	90 % to 10 %
High-Drive Outputs: (Note 2)						
Rise Time	t_{hdr}			5	ns	10 % to 90 %
Fall Time	t_{hdf}			5	ns	90 % to 10 %
Clocks and Reset:						
Sample Frequency: PLL Locked PLL Unlocked	Fs	37.9 5	44.1	48.1	kHz kHz	
RMCK Master Clock Output	f_{rmck}	2.4256		73.8816	MHz MHz	64×(Fs = 37.9 kHz) 1536×(Fs = 48.1 kHz)
Crystal Oscillator	f_{osc}	9.7024		24.6272	MHz MHz	256×(Fs = 37.9 kHz) 512×(Fs = 48.1 kHz)
Pulse-Width Variation, RX on timing-master (**bXCR.MTR** set)	t_{pwmn} t_{pwmx}	0.70		1.40	UI UI	**bXSR2.INV**=0 CM4 = C3h (note 4)
Average Pulse Width Distortion, RX on timing-slave (**bXCR.MTR** clear)	t_{apwd}	-0.18		0.35	UI	**bXSR2.INV**=0, CM4=C3h (Notes 3, 4) t_{js} = 7 ns (pp)
Pulse Width Distortion, SR0	t_{pwds}	0.9		1.1	UI	(Note 5)
Jitter Tolerance (timing-master): XTI/O, SR0, SCK RX RX	t_{jm}	 0.8 5 7.38			 ns (pp) UI ns (pp)	**bXCR.MTR** set (Note 6) (Note 7) **GCTL.MJCE** cleared
RX Rise and Fall time	t_{rxt}			10	ns	recommended
TX Rise and Fall time	t_{txt}		7	10	ns	C_L = 20 pF
$\overline{\text{RST}}$ pulse width	t_{rspw}	150			ns	(Note 8)
Configuration pin setup to $\overline{\text{RST}}$ rising	t_{cpsrs}	30			ns	(Note 8)
Configuration pin hold from $\overline{\text{RST}}$ high	t_{cphrs}	30			ns	(Note 8)

Notes:

1. **GPA[3:0], GPB[3:0], GPC[3:0], GPD[3:0], IOA[3:0], IOB[3:0], IOC[5:0], IOG1, IOG3**. For pins configured as open-drain outputs, t_{hdr} does not apply.
2. **IOD[1:0], IOE[15:0], IOF[10:0], IOG0, IOG2, IOG4**. Pins configured as open-drain outputs, V_{OH} does not apply.
3. When the MOST Network frequency is 44.1 kHz, one UI is 22.1 ns, which is $t_{rxbp}/2$. The pulse width variation is defined as the sum of the average pulse-width distortion plus high-frequency jitter.
4. The average (APWD) spec, defined as $t_{apwd} = \frac{t_{pwmx} + t_{pwmn} - t_{rxbp}}{2}$, is illustrated in Figure 6-2. The **FLT** pin is a high-impedance node; therefore, leakage current should be kept below 1 µA, or average pulse-width distortion tolerance could be adversely affected.
5. **SR0** configured for S/PDIF where node is a timing-slave. One UI (Unit Interval) is defined as a single bi-phase period (one half of a bit period). Therefore, with Fs = 44.1 kHz, one UI is 177 ns.
6. The MOST Specification requires each node have a crystal-based master-clock source to support ring-break diagnostics, where nodes normally configured as timing-slaves can be re-configured as the timing-master.
7. Assumes software has enabled **GCTL.MJCE** and has toggled **GTCL.RFPR** whenever the part enters the lock state.
8. Power-up configuration pins include: **IOA3, IOB0, IOG4, IOF10**. The $\overline{\textbf{RST}}$ pulse width indicates the minimum amount of time required to reset the OS8805. However, the configuration pins can take longer to settle to their default state, based on the trace capacitance and size of the pull-up or pull-down resistor. Therefore, the $\overline{\textbf{RST}}$ pulse width must be long enough to allow the external configuration pins to achieve their default state.

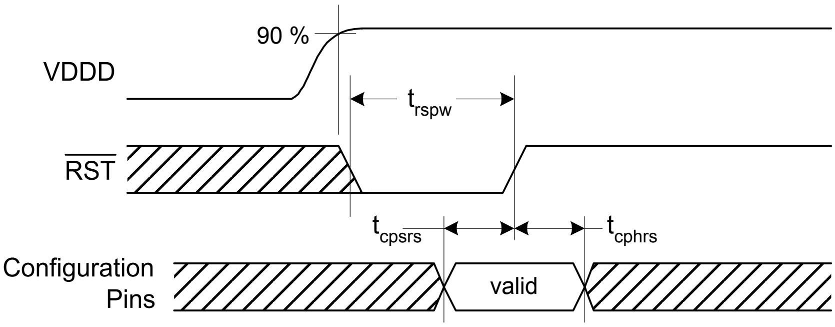


Figure 6-1: Reset and Configuration Pins

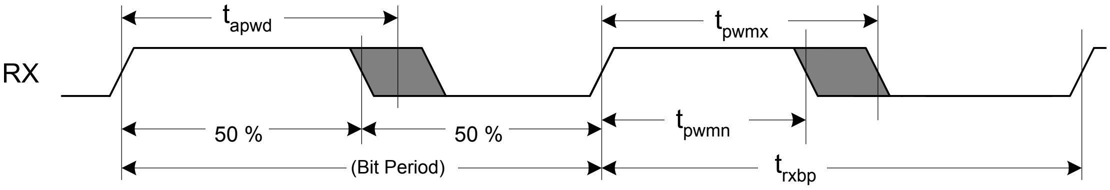


Figure 6-2: RX Pulse-Width Distortion Timing

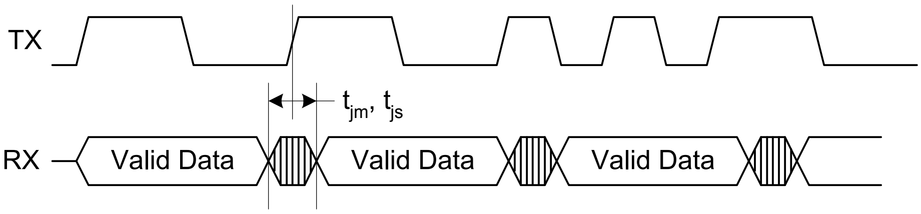


Figure 6-3: RX Jitter Tolerance Timing

6.6 External Program Memory Interface

T_J = -40 to 150 °C; VDDD,VDDA = 3.3 V ±5 %; GNDD = GNDA = 0.0 V; PLL locked at 44.1 kHz, Load Capacitance = 30 pF.

384xFs	Symbol	MMPC.PCSC = 0		MMPC.PCSC = 1		Unit
Parameter		Min	Max	Min	Max	
$\overline{PCS}$ high time	t_{pcsh}	56		41		ns
$\overline{PCS}$ low time	t_{pcsl}	56		70		ns
$\overline{PCS}$ cycle time	t_{pcsc}	115		115		ns
$\overline{PMW}$ set up to $\overline{PCS}$ low	t_{pmws}	26		19		ns
$\overline{PMW}$ hold from $\overline{PCS}$ high	t_{pmwh}	15		15		ns
MA[16:0] set up to $\overline{PCS}$ low	t_{pas}	26		19		ns
MA[16:0] hold from $\overline{PCS}$ high	t_{pah}	15		15		ns
Read Operation:						
MD[7:0] set up to $\overline{PCS}$ high	t_{rpds}	15		15		ns
MD[7:0] hold from $\overline{PCS}$ high	t_{rpdh}	0		0		ns
Write Operation:						
MD[7:0] set up to $\overline{PCS}$ low	t_{wpds}	15		15		ns
MD[7:0] hold from $\overline{PCS}$ high	t_{wpdh}	15		15		ns

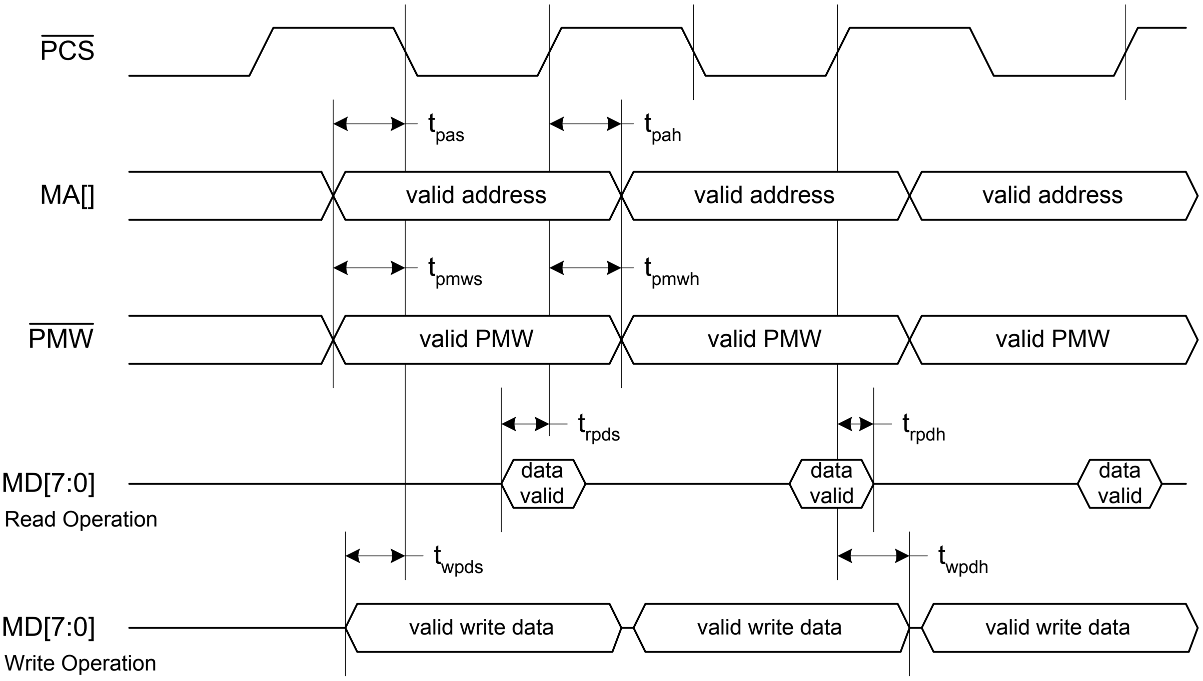


Figure 6-4: External Program Memory Timing (MMPC.PCSC = 0)

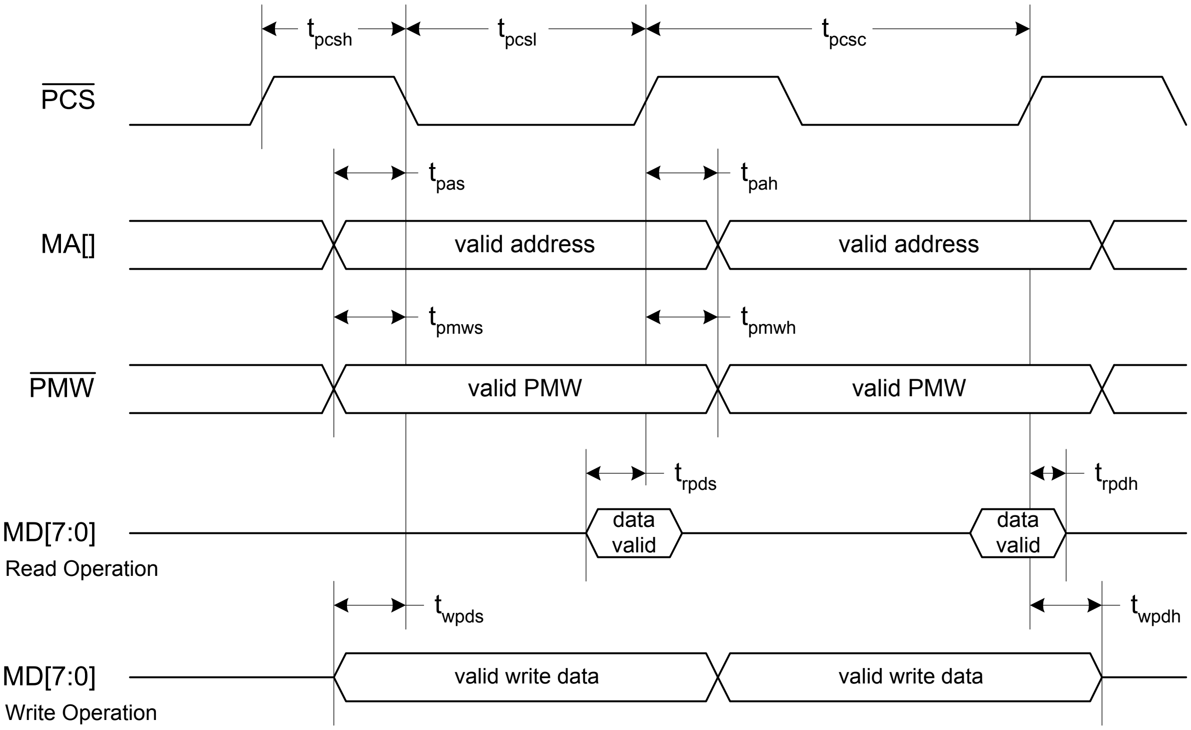


Figure 6-5: External Program Memory Timing (MMPC.PCSC = 1)

6.7 External Data Memory Interface

T_J = -40 to 150 °C; VDDD,VDDA = 3.3 V ±5 %; GNDD,GNDA = 0.0 V; PLL locked at 44.1 kHz.;
Load Capacitance = 30 pF (Note 1).

DRAM Interface (XMC.MMS = 1)	Symbol	Slow (XMC.MT = 1)		Fast (XMC.MT = 0)		Unit
Parameter		**Min**	**Typ**	**Min**	**Typ**	
DSP Clock cycle (Note 2)	tc		$\frac{1}{1536Fs}$		$\frac{1}{1536Fs}$	s
MA[9:0] set up to $\overline{\text{MRAS}}$ low	t_{mars}	0	0.5tc	0	0.5tc	ns
MA[9:0] hold from $\overline{\text{MRAS}}$ low	t_{marh}	48	3.5tc	34	2.5tc	ns
MA[9:0] set up to $\overline{\text{MCAS}}$ low	t_{macs}	0	0.5tc	0	0.5tc	ns
MA[9:0] hold from $\overline{\text{MCAS}}$ low	t_{mach}	48	3.5tc	34	2.5tc	ns
$\overline{\text{MRAS}}$ low to $\overline{\text{MCAS}}$ low	t_{rascas}	50	4tc	34	3tc	ns
$\overline{\text{MRAS}}$ low time 1 access 2 accesses 4 accesses	t_{rasl}	100	7tc 11tc 19tc	70	5tc 8tc 14tc	ns
$\overline{\text{MRAS}}$ high time	t_{rash}	70	5tc	65	4tc	ns
$\overline{\text{MCAS}}$ low time	t_{casl}	38	3tc	24	2tc	ns
$\overline{\text{MCAS}}$ high time ($\overline{\text{MRAS}}$ is low)	t_{cash}	12	tc	11	tc	ns
$\overline{\text{MCAS}}$ low to $\overline{\text{MRAS}}$ high	$t_{caslrash}$	38	3tc	24	2tc	ns
Read Operation:						
MD[7:0] set up to $\overline{\text{MCAS}}$ high	t_{mdrs}	10		10		ns
MD[7:0] hold from $\overline{\text{MCAS}}$ high	t_{mdrh}	0		0		ns
Write Operation:						
MD[7:0] set up to $\overline{\text{MCAS}}$ low	t_{mdws}	0	0.5tc	0	0.5tc	ns
MD[7:0] hold from $\overline{\text{MCAS}}$ high	t_{mdwh}	5	0.5tc	5	0.5tc	ns
Refresh Operation:						
$\overline{\text{MRAS}}$ low time	t_{rrasl}	80	6tc	50	4tc	ns
$\overline{\text{MCAS}}$ low time	t_{rcasl}	80	7tc	65	5tc	ns
$\overline{\text{MCAS}}$ low to $\overline{\text{MRAS}}$ low	$t_{rcslrsl}$	10	tc	10	tc	ns

Notes:

1. The following DRAM timing figure is illustrated with two memory accesses. One to four access can occur based on the Data memory size (**XMC.MWW**) and the external memory data width (**XMC.BW**). In DRAM mode, the address bits above **MA9** are indeterminate.
2. The DSP clock frequency is dependent on the **DnPCR.DFS[1:0]** bits. However, each frequency is achieved by dropping clocks from a 1536Fs continuous clock. Therefore, the minimum time is always based on a 1536Fs clock period, but the maximum could be up to four times the minimum value.

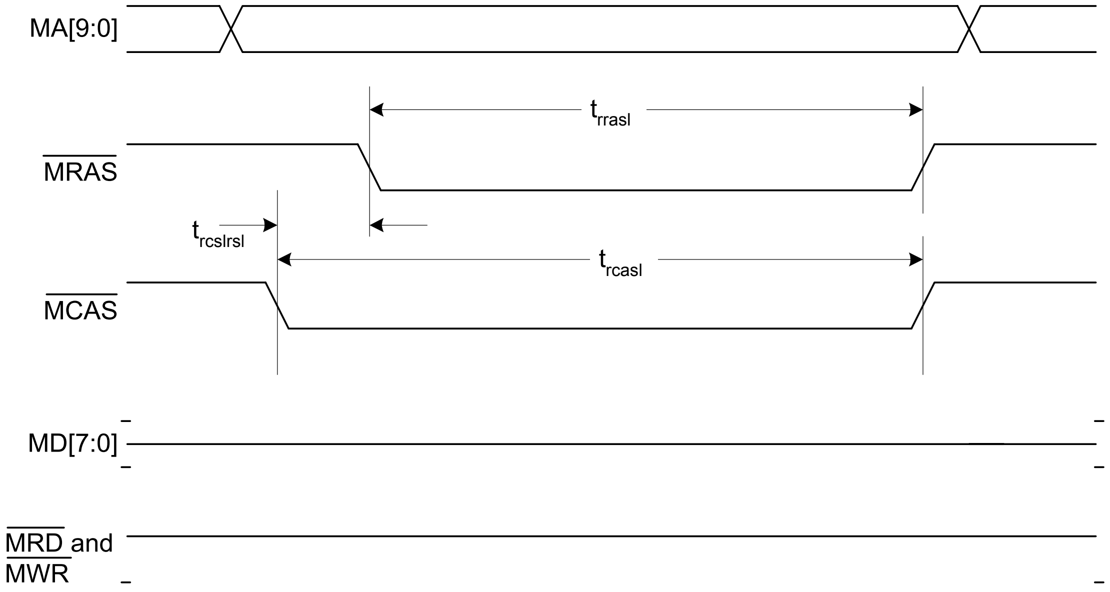


Figure 6-6: DRAM Refresh (CAS-before-RAS) Timing (XMC.MMS = 1)

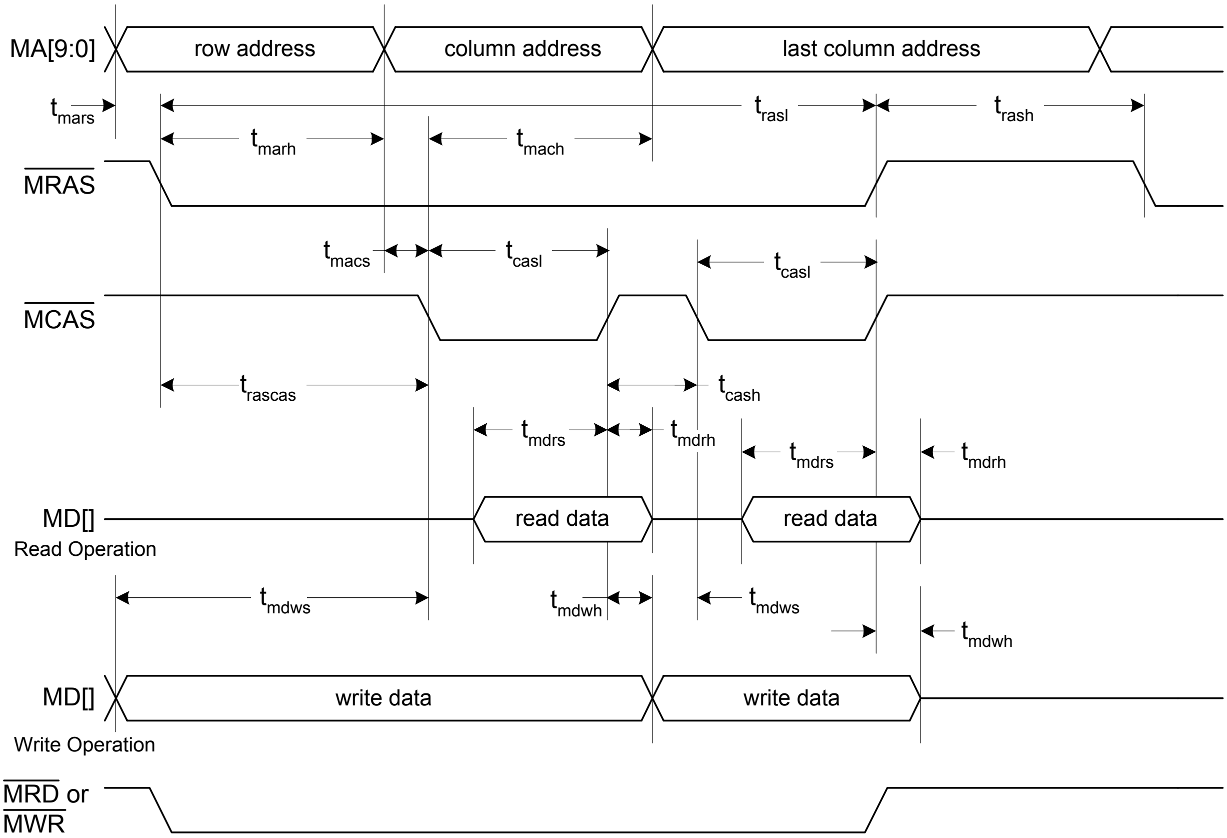


Figure 6-7: DRAM Timing (XMC.MMS = 1)

T_J = -40 to 150 °C; VDDD,VDDA = 3.3 V ±5 %; GNDD,GNDA = 0.0 V; PLL locked at 44.1 kHz.;
Load Capacitance = 30 pF. (Note 1)

SRAM Interface (XMC.MMS = 0)	**Symbol**	**Slow (XMC.MT = 1)**			**Fast (XMC.MT = 0)**			**Unit**
Parameter		**Min**	**Typ**	**Max**	**Min**	**Typ**	**Max**	
DSP Clock cycle (Note 2)	tc		$\frac{1}{1536Fs}$			$\frac{1}{1536Fs}$		s
$\overline{\text{MCS}}$ low 1 access 2 accesses 4 accesses	t_{mcsl}	41	4tc 7tc 13tc		27	3tc 5tc 9tc		ns ns ns
$\overline{\text{MCS}}$ high	t_{mcsh}	38	3tc		38	3tc		ns
Read Operation:								
MA[16:0] set up to $\overline{\text{MRD}}$ low	t_{mars}	5	1.5tc		5	1.5tc		ns
MA[16:0] hold from $\overline{\text{MRD}}$ high	t_{marh}	5	0.5tc		5	0.5tc		ns
$\overline{\text{MCS}}$ set up to $\overline{\text{MRD}}$ low	t_{mcsrs}	5	tc		5	tc		ns
$\overline{\text{MCS}}$ hold from $\overline{\text{MRD}}$ high	t_{mcsrh}	5	tc		5	tc		ns
$\overline{\text{MRD}}$ low to MD[7:0] valid	t_{mrldv}			17			7	ns
MA[16:0] valid to MD[7:0] valid	t_{mavdv}			26			12	ns
MD[7:0] set up to $\overline{\text{MRD}}$ high	t_{mdrs}	5			5			ns
MD[7:0] hold from $\overline{\text{MRD}}$ high	t_{mdrh}	0			0			ns
$\overline{\text{MRD}}$ low 1 access 2 accesses 4 accesses	t_{rdl}	24	2tc 5tc 11tc		8	1tc 3tc 7tc		ns ns ns
$\overline{\text{MRD}}$ high	t_{rdh}	60	5tc		60	5tc		ns
Write Operation:								
MA[16:0] set up to $\overline{\text{MWR}}$ low	t_{maws}	0	0.5tc		0	0.5tc		ns
MA[16:0] hold from $\overline{\text{MWR}}$ high	t_{mawh}	5	0.5tc		5	0.5tc		ns
$\overline{\text{MCS}}$ set up to $\overline{\text{MWR}}$ low	t_{mcsws}	5	tc		5	tc		ns
$\overline{\text{MCS}}$ hold from $\overline{\text{MWR}}$ high	t_{mcswh}	5	tc		5	tc		ns
MD[7:0] set up to $\overline{\text{MWR}}$ low	t_{mdws}	0	0.5tc		0	0.5tc		ns
MD[7:0] hold from $\overline{\text{MWR}}$ high	t_{mdwh}	5	0.5tc		5	0.5tc		ns
$\overline{\text{MWR}}$ low	t_{mwl}	22	2tc		8	tc		ns
$\overline{\text{MWR}}$ high	t_{mwh}	10	tc		10	tc		ns

Notes:

1. The following SRAM timing figure is illustrated with two memory accesses. One to four access can occur based on the Data memory size (**XMC.MWW**) and the external memory data width (**XMC.BW**)
2. The DSP clock frequency is dependent on the **DnPCR.DFS[1:0]** bits. However, each frequency is achieved by dropping clocks from a 1536Fs continuous clock. Therefore, the minimum time is always based on a 1536Fs clock period, but the maximum could be up to four times the minimum value.

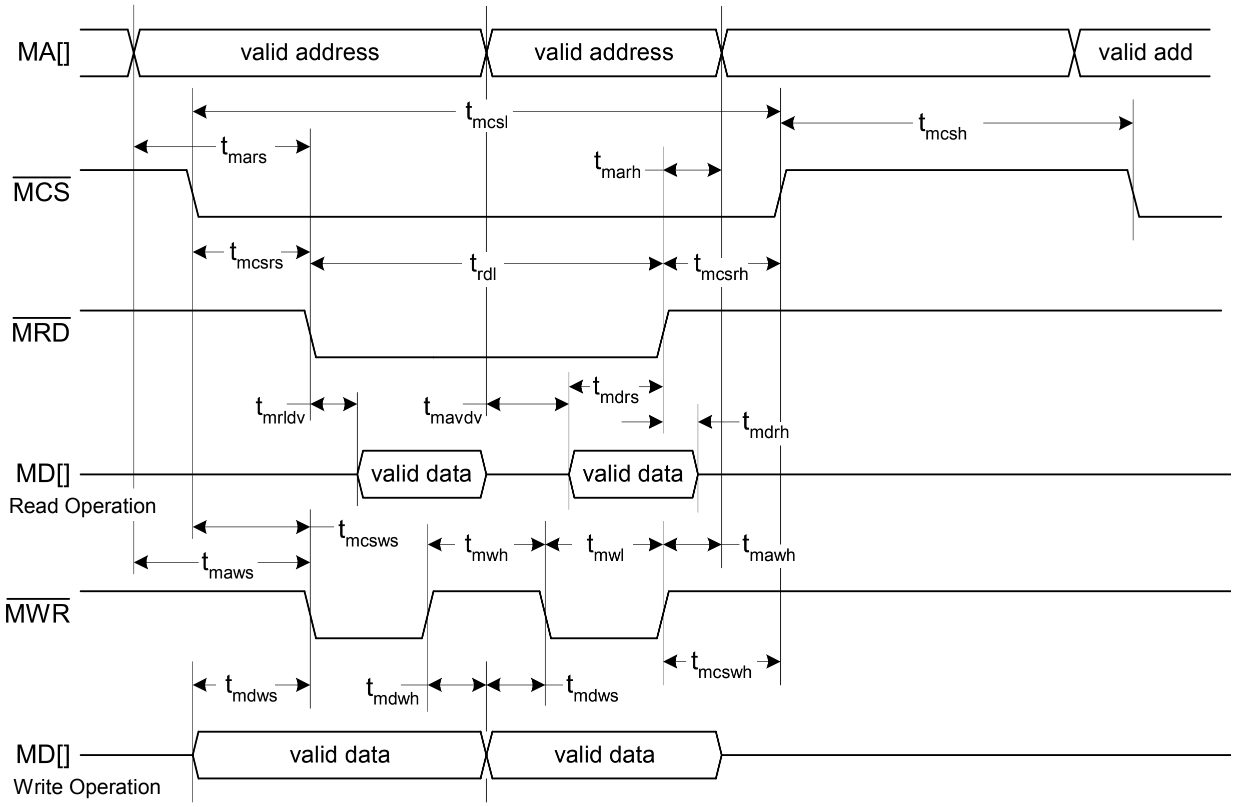


Figure 6-8: SRAM Timing (XMC.MMS = 0)

6.8 Source Data Ports

T_J = -40 to 150 °C; VDDD,VDDA = 3.3 V ±5 %; GNDD,GNDA = 0.0 V; PLL locked at 44.1 kHz;
Load Capacitance = 30 pF.

Parameter	Symbol	Min	Typ	Max	Unit	Comment
FSY frequency (Note 1)	f_{fsy}	37.9	44.1	48.1	kHz	
SCK frequency (Note 1)	f_{sck}	1.2128		3.0784	MHz MHz	32×Fs at 37.9 kHz 64×Fs at 48.1 kHz
SCK and FSY Outputs* (bSDC1.MOD[1:0] *= 01 or 10):						
SCK low time	t_{sckl}	25			ns	
SCK high time	t_{sckh}	25			ns	
FSY/SCK rise and fall time	t_{rsp}, t_{fsp}		5		ns	
SCK falling to FSY valid	t_{fsyv}	-25		25	ns	(Notes 2, 3)
SR[1:0] valid to SCK rising	t_{srs}	25			ns	(Notes 2, 4)
SR[1:0] hold from SCK rising	t_{srh}	25			ns	(Notes 2, 4)
SX[1:0] valid from SCK falling	t_{sxv}			30	ns	(Notes 2, 4)
SCK and FSY Inputs* (bSDC1.MOD[1:0] *= 00):						
SCK low time	t_{sckl}	25			ns	
SCK high time	t_{sckh}	25			ns	
FSY valid to SCK rising	t_{fsys}	25			ns	(Notes 2, 3)
FSY hold from SCK rising	t_{fsyh}	50			ns	(Notes 2, 3)
SR[1:0] valid to SCK rising	t_{srs}	25			ns	(Notes 2, 4)
SR[1:0] hold from SCK rising	t_{srh}	75			ns	(Notes 2, 4)
SX[1:0] valid from SCK falling	t_{sxv}			80	ns	(Notes 2, 4)

Notes:

1. If **SCK** and **FSY** are inputs, they must be frequency locked to the master clock (**RMCK** output clock)
2. **SCK** active edge (edge where data is stable and not changing) is determined by the **bSDC1.EDG** bit. The parameters and are illustrated with **EDG** set. If **EDG** is cleared, reverse the edge in the parameters above and invert **SCK** in the diagram below.
3. **FSY** polarity is determined by the **bSDC1.POL** bit.
4. The MSB of **SR[1:0]** and **SX[1:0]** is the first or the second bit after **FSY** changes, based on the **bSDC1.DEL** bit.

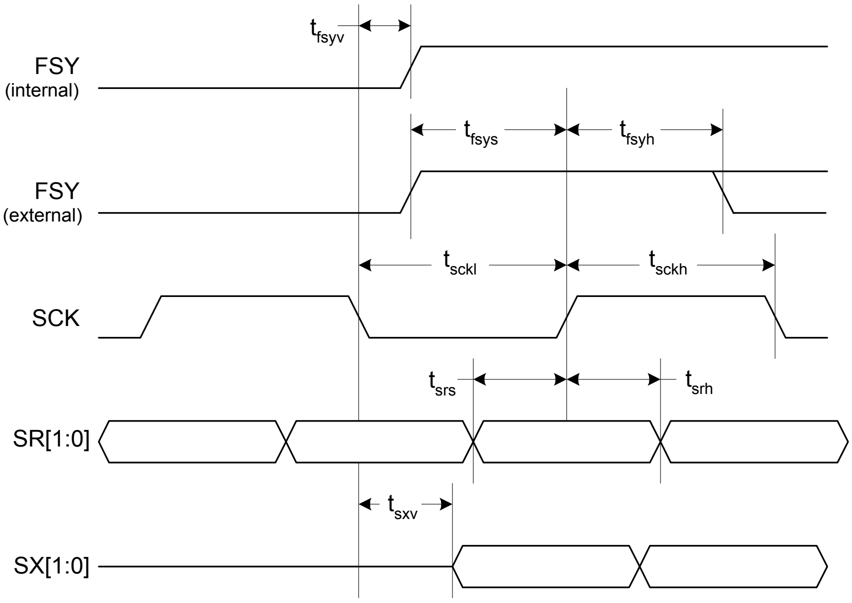


Figure 6-9: Source Port Timing

6.9 DSP Async. Source Ports

T_J = -40 to 150 °C; VDDD,VDDA = 3.3 V ±5 %; GNDD,GNDA = 0.0 V; PLL locked at 44.1 kHz;
Load Capacitance = 30 pF.

Parameter	Symbol	Min	Typ	Max	Unit	Comment
FSYnx frequency	f_{fsy}	37.9	44.1	48.1	kHz	
SCKxn frequency	f_{sck}	1.2128		24.6272	MHz MHz	32×Fs at 37.9 kHz 512×Fs at 48.1 kHz
***SCKxn and FSYxn Outputs* (DSnC.CKOE *set*):**						
SCKxn low time	t_{sckl}	15			ns	
SCKxn high time	t_{sckh}	15			ns	
FSYxn/SCKxn rise and fall time	t_{rsp}, t_{fsp}		5		ns	
SCKxn falling to FSYxn valid	t_{fsyv}	-15		15	ns	(Notes 1, 2)
SRxn valid to SCKxn rising	t_{srs}	15			ns	(Notes 1, 3)
SRxn hold from SCKxn rising	t_{srh}	15			ns	(Notes 1, 3)
SXxn valid from SCKxn falling	t_{sxv}			15	ns	(Notes 1, 3)
***SCKxn and FSYxn Inputs* (DSnC.CKOE *clear*):**						
SCKxn low time	t_{sckl}	15			ns	
SCKxn high time	t_{sckh}	15			ns	
FSYxn valid to SCKxn rising	t_{fsys}	15			ns	(Notes 1, 2)
FSYxn hold from SCKxn rising	t_{fsyh}	15			ns	(Notes 1, 2)
SRxn valid to SCKxn rising	t_{srs}	15			ns	(Notes 1, 3)
SRxn hold from SCKxn rising	t_{srh}	15			ns	(Notes 1, 3)
SXxn valid from SCKxn falling	t_{sxv}			15	ns	(Notes 1, 3)

Notes:

1. **SCKxn** active edge (edge where data is stable and not changing) is determined by the **DSnC.EDG** bit. The parameters and are illustrated with **EDG** set. If **EDG** is cleared, reverse the edge in the parameters above and invert **SCKxn** in the diagram below.
2. **FSYxn** polarity is determined by the **DSnC.POL** bit.
3. The MSB of **SRxn** and **SXxn** is the first or the second bit after **FSYxn** changes, based on the **DSnC.DEL** bit.

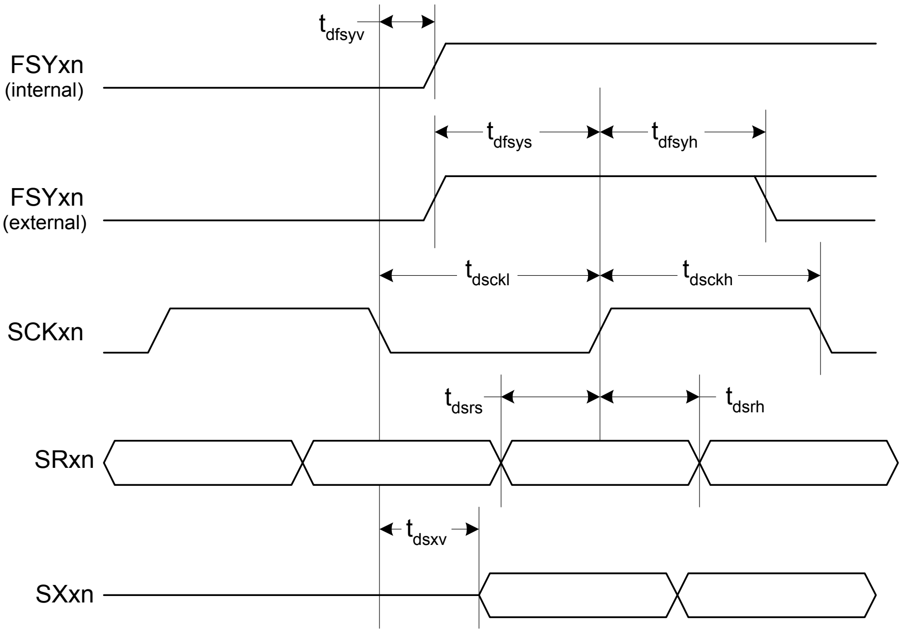


Figure 6-10: DSP Async. Source Port Timing

6.10 Control and Debug Ports - SPI Mode

T_J = -40 to 150 °C; VDDD,VDDA = 3.3 V ±5 %; GNDD,GNDA = 0.0 V; Load Capacitance = 30 pF.

Parameter	Slave mode (CPS.CPMM/DCPS.DPMM clear)				
	Symbol	Min	Typ	Max	Unit
SCLK frequency	f_{scl}			100	kHz
SCLK low time	t_{scll}	2			µs
SCLK high time	t_{sclh}	2			µs
SCLK low to $\overline{\text{CS}}$ falling setup	t_{sklcsl}	500			ns
SCLK low to $\overline{\text{CS}}$ high setup	t_{sklcsh}	2			µs
$\overline{\text{CS}}$ low to SCLK rise (Note 1)	t_{css}	2			µs
$\overline{\text{CS}}$ low to SDOUT valid	t_{cdv}			2	µs
$\overline{\text{CS}}$ high time	t_{cht}	2			µs
SDIN valid to SCLK rising (Note 1)	t_{sds}	500			ns
SDIN hold from SCLK rising (Note 1)	t_{sdh}	500			ns
SCLK falling to SDOUT valid (Note 1)	t_{sdv}			3	µs
$\overline{\text{CS}}$ high to SDOUT Hi-Z	t_{sdz}			2	µs

Note:

1. In GSPI mode the SCLK polarity and phase are programmable. The parameters and are illustrated for GSPI mode with **CPS.CPOL** = **CPS.CPHA** = 0. See Section 2.3.3.3 on page 60 for more information.

T_J = -40 to 150 °C; VDDD,VDDA = 3.3 V ±5 %; GNDD,GNDA = 0.0 V; PLL locked at 44.1 kHz;
Load Capacitance = 30 pF

Parameter	GSPI Master mode (CPS.CPMM/DCPS.DPMM set)					
	Symbol	Slow Speed CPS/DCPS.FAST clear		Fast Speed CPS/DCPS.FAST set		Unit
		Min	Max	Min	Max	
SCLK frequency (Note 2)	f_{scl}		100		350	kHz
SCLK low time	t_{scll}	4		1		µs
SCLK high time	t_{sclh}	4		1		µs
SCLK low to $\overline{\text{CS}}$ high setup	t_{sklcsh}	1		1		µs
$\overline{\text{CS}}$ low to SCLK rise (Note 1)	t_{css}	1		1		µs
$\overline{\text{CS}}$ low to SDOUT valid	t_{cdv}		250		250	ns
$\overline{\text{CS}}$ high time	t_{cht}	1		0.25		µs
SDIN valid to SCLK rising (Note 1)	t_{sds}	500		250		ns
SDIN hold from SCLK rising (Note 1)	t_{sdh}	500		250		ns
SCLK falling to SDOUT valid (Note 1)	t_{sdv}		1		0.5	µs
$\overline{\text{CS}}$ high to SDOUT Hi-Z	t_{sdz}		1		1	µs

Notes:

1. In GSPI mode the SCLK polarity and phase are programmable. The parameters and are illustrated for GSPI mode with **CPS.CPOL** = **CPS.CPHA** = 0. See Section 2.3.3.3 on page 60 for more information.
2. The typical clock frequency when **FAST** is clear is 88.2 kHz when Fs = 44.1 kHz, and 96 kHz when Fs = 48 kHz. When **FAST** is set, the typical clock frequency is 295.8 kHz when Fs= 44.1 khz, and 310.6 kHz when Fs = 48 kHz.

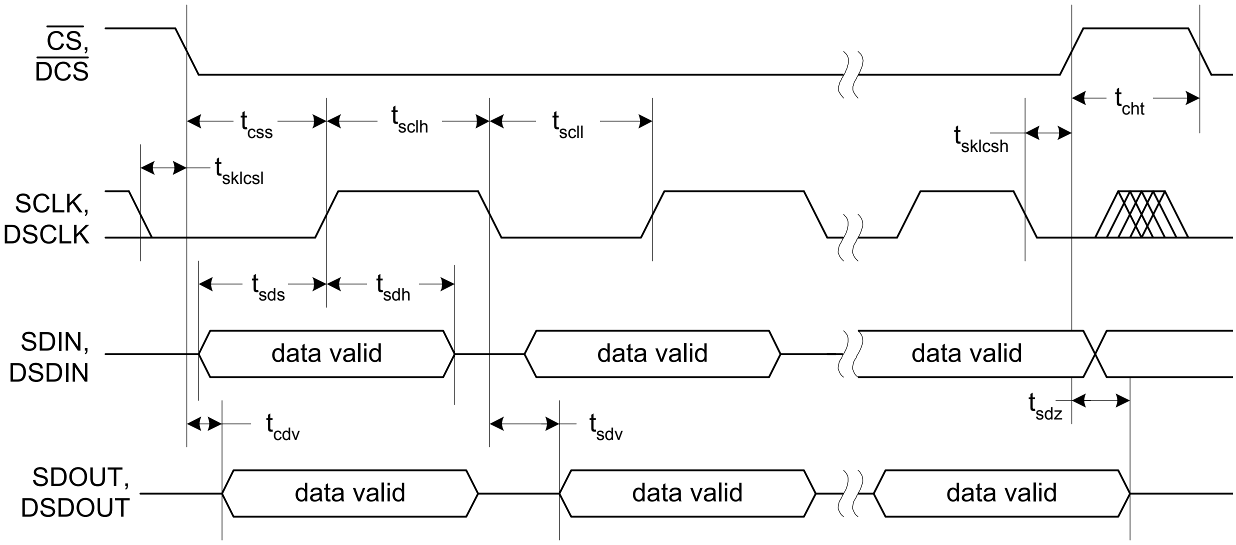


Figure 6-11: Control/Debug Port - SPI Timing

6.11 Control and Debug Ports - I^{2}C Mode

T_J = -40 to 150 °C; VDDD,VDDA = 3.3 V ±5 %; GNDD,GNDA = 0.0 V; PLL locked at 44.1 kHz.

Parameter	Slave mode (CPS.CPMM/DCPS.DPMM clear)				
	Symbol	Min	Typ	Max	Unit
SCL frequency	f_{scl}			100	kHz
Bus free between transmissions (SDA high time between start and stop)	t_{buf}	4.7			µs
Start condition hold time (SDA falling to SCL falling)	t_{stah}	4			µs
SCL low	t_{scll}	4.7			µs
SCL high	t_{sclh}	4			µs
SDA input hold from SCL falling	t_{sdah}	0			µs
SDA input valid to SCL rising	t_{sdas}	500			ns
(repeated) start condition setup time	t_{stas}	4.7			µs
SDA and SCL rise time	t_r			1	µs
SDA and SCL fall time	t_f			300	ns
Stop condition setup time (SCL rising to SDA rising)	t_{stps}	4			µs

T_J = -40 to 150 °C; VDDD, VDDA = 3.3 V ±5 %; GNDD,GNDA = 0.0 V; PLL locked at 44.1 kHz.

Parameter	Master mode (CPS.CPMM/DCPS.DPMM set)					
	Symbol	Slow Speed CPS/DCPS.FAST clear		Fast Speed CPS/DCPS.FAST set		Unit
		Min	Max	Min	Max	
SCL frequency (Note 1)	f_{scl}		100		350	kHz
Bus free between transmissions (SDA high time between start and stop)	t_{buf}	4.7		1.3		µs
Start condition hold time (SDA falling to SCL falling)	t_{stah}	4		0.6		µs
SCL low	t_{scll}	4.7		1.3		µs
SCL high	t_{sclh}	4		0.6		µs
SDA input hold from SCL falling	t_{sdah}	0		0		µs
SDA input valid to SCL rising	t_{sdas}	500		300		ns
Repeat Start condition setup time	t_{stas}	4.7		0.6		µs
SDA and SCL rise time	t_r		1		0.3	µs
SDA and SCL fall time	t_f		300		300	ns
Stop condition setup time (SCL rising to SDA rising)	t_{stps}	4		0.6		µs

Note:

1. The typical clock frequency when **FAST** is clear is 88.2 kHz when Fs = 44.1 kHz, and 96 kHz when Fs = 48 kHz. When **FAST** is set, the typical clock frequency is 295.8 kHz when Fs= 44.1 khz, and 310.6 kHz when Fs = 48 kHz.

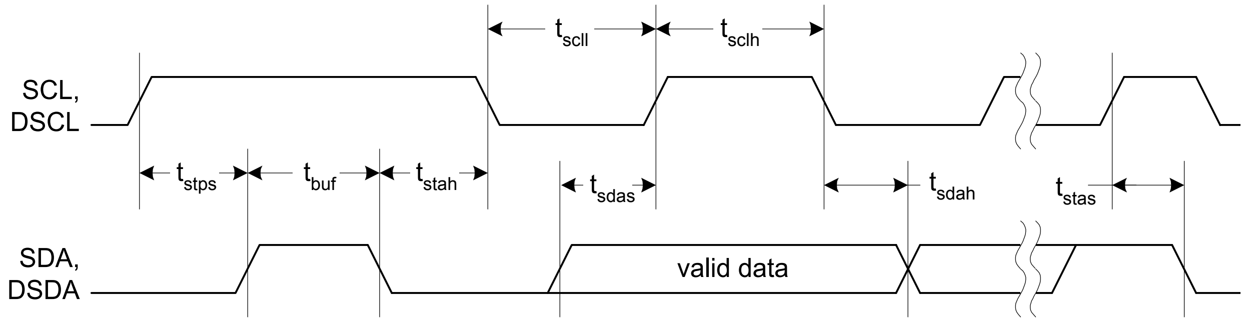


Figure 6-12: Control/Debug Port - I^{2}C Timing

6.12 Control and Debug Ports - USART mode

T_J = -40 to 150 °C; VDDD,VDDA = 3.3 V ±5 %; GNDD,GNDA = 0.0 V; PLL locked at 44.1 kHz;
Load Capacitance = 30 pF.

Parameter	Symbol	Min	Typ	Max	Unit	Comments
UCKn frequency	f_{usart}			1	MHz	
UCKn low time	t_{uckl}	300			ns	
UCKn high time	t_{uckh}	300			ns	
URXn valid to UCKn rising	t_{ucrxs}	150			ns	
URXn hold from UCKn rising	t_{ucrxh}	150			ns	
UTXn valid from UCKn falling	t_{uctxv}			200	ns	

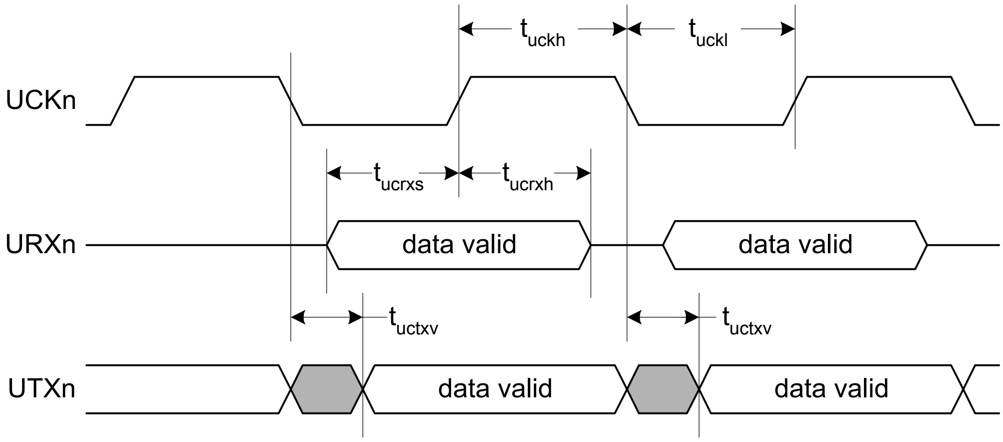


Figure 6-13: Control/Debug Port - USART Timing

6.13 Analog Performance

6.13.1 MPX ADC

T_J = -40 to 150 °C; VDDD,VDDA = 3.3 V ±5 %; GNDD,GNDA = 0.0 V; PLL locked at 44.1 kHz; Gain set to 0 dB; unless otherwise noted.

Parameter	Min	Typ	Max	Unit
Analog Characteristics:				
Resolution	16			bits
Sample Rate		4Fs		Hz
Dynamic Range: 20 Hz to 20 kHz bandwidth 20 Hz to 60 kHz bandwidth		 87 82		 dB FS dB FS
THD+N (Note 1)		-80		dB FS
Full-scale Input		2.5		Vpp
Maximum Gain		24		dB
Gain Step Size	0.7	0.9	1.3	dB
Offset Error (Note 2)		3		codes
Input Resistance (Note 3)	10		332	kΩ
Input Capacitance			15	pF
Inter-channel Isolation: MPX to Audio MPX to MIC MPX to DACs		 90 100 100		 dB dB dB
Digital Characteristics:				
Passband	0.0001Fs		1.34Fs	Hz
Passband Ripple			±0.01	dB
Transition Band	1.34Fs		2.66Fs	Hz
Stop-band	2.66Fs			Hz
Stop-band Rejection	80			dB
Group Delay		constant		

Notes:

1. MPX ADC measured with –3 dB sine wave at 997 Hz, 20 Hz – 60 kHz bandwidth.
2. Independent of Gain setting.
3. Minimum when gain set to 26 dB; and maximum when gain set to 0 dB.

6.13.2 Audio ADCs

T_J = -40 to 150 °C; VDDD,VDDA = 3.3 V ±5 %; GNDD,GNDA = 0.0 V; PLL locked at 44.1 kHz;
Gain set to 0 dB; unless otherwise noted.

Parameter	Min	Typ	Max	Unit
Analog Characteristics:				
Resolution	16			bits
Sample Rate		Fs		Hz
Dynamic Range, 20 Hz to 20 kHz bandwidth		89		dB FS
THD+N (Note 1)		-88		dB FS
Full-scale Input		2.5		Vpp
Maximum Gain		13.7		dB
Gain Step Size	0.7	0.9	1.3	dB
Offset Error (Note 2)		3		codes
Input Resistance (Note 3)	10		80	kΩ
Input Capacitance			15	pF
Inter-channel Isolation: Left to Right Audio to MPX Audio to MIC Audio to DACs		 90 100 100 100		 dB dB dB dB
Inter-channel Gain mismatch			0.5	dB
Digital Characteristics:				
Passband: ±0.01 dB ±1.00 dB	0.00005Fs 0.0002Fs		0.36Fs 0.41Fs	Hz
Passband Ripple			±0.01	dB
Transition Band	0.4Fs		0.6Fs	Hz
Stop-band	0.6Fs			Hz
Stop-band Rejection	70			dB
Group Delay		constant		
ADCL to ADCR Group Delay		$\frac{1}{6.8Fs}$		s

Notes:

1. Audio ADC measured with -3 dB FS sine wave at 997 Hz, 20 Hz – 20 kHz bandwidth.
2. Independent of Gain setting.
3. Minimum when gain set to 15 dB; and maximum when gain set to 0 dB.

6.13.3 Microphone ADC

T_J = -40 to 150 °C; VDDD,VDDA = 3.3 V ±5 %; GNDD,GNDA = 0.0 V; PLL locked at 44.1 kHz; Gain set to 0 dB; unless otherwise noted.

Parameter	Min	Typ	Max	Unit
Analog Characteristics:				
Resolution	16			bits
Sample Rate		0.25Fs		Hz
Dynamic Range, 20 Hz to 0.25Fs/2 bandwidth		87		dB FS
THD+N (Note 1)		-86		dB FS
Full-scale Input		2.5		Vpp
Maximum Gain		13.6		dB
Gain Step Size	0.7	0.9	1.3	dB
Offset Error (Note 2)		3		codes
Input Resistance (Note 3)	10		80	kΩ
Input Capacitance			15	pF
Inter-channel Isolation: MIC to MPX MIC to Audio MIC to DACs		 100 100 100		 dB dB dB
Digital Characteristics:				
Passband: ±0.01 dB	0.0002Fs		0.1Fs	Hz
Transition Band	0.1Fs		0.15Fs	Hz
Stop-band	0.15Fs			Hz
Stop-band Rejection	60			dB
Group Delay		constant		

Notes:

1. Mic ADC measured with -3 dB FS sine wave at 997 Hz, 20 Hz – 0.25Fs/2 bandwidth.
2. Independent of Gain setting.
3. Minimum when gain set to 15 dB; and maximum when gain set to 0 dB.

6.13.4 Audio DACs

T_J = -40 to 150 °C; VDDD,VDDA = 3.3 V ±5 %; GNDD,GNDA = 0.0 V; PLL locked at 44.1 kHz; Attenuation set to 0 dB; Differential ouput; unless otherwise noted.

Parameter	Min	Typ	Max	Unit
Analog Characteristics:				
Resolution	16			bits
Sample Rate		Fs		Hz
Dynamic Range (Differential): Instantaneous (**ADACn.ATTN[4:0]** = 00h) Total (**ADACn.ATTN[4:0]** = 1Fh)		 88 108		 dB FS dB FS
Dynamic Range (Single-ended): (Note 1) Instantaneous (**ADACn.ATTN[4:0]** = 00h) Total (**ADACn.ATTN[4:0]** = 1Fh)		 88 101		 dB FS dB FS
THD+N: unloaded (Note 2) R_L = 2 kΩ		-88 -86		dB FS dB FS
Full-Scale Output (Note 3)		3.6		Vpp
Minimum Attenuation		31		dB
Gain Step Size	0.7	1	1.3	dB
Offset Error		10	20	mV
Output Load Resistance (Note 4)	2			kΩ
Output Load Capacitance			100	pF
Inter-channel Isolation: DAC to DAC DAC to MPX DAC to MIC DAC to Audio		 100 110 110 110		 dB dB dB dB
Inter-channel Gain mismatch			0.5	dB
Out-of-Band Noise (20 kHz to 80 kHz)		56		dB FS
Digital Characteristics:				
Passband: ±0.01 dB ±1.00 dB	0 0		0.42Fs 0.45Fs	Hz
Passband Ripple			±0.01	dB
Transition Band	0.42Fs		0.58Fs	Hz
Stop-band	0.58Fs			Hz
Stop-band Rejection	60			dB
Group Delay		constant		
DAC[n] to DAC[n+1] Group Delay		$\frac{1}{4Fs}$		s

Notes:

1. **ADACn.SEDE** set and capacitors on **VREFS** as mentioned in the Data Sheet.
2. DACs measured with a -3 dB FS sine wave at 997 Hz, 20 Hz – 20 kHz bandwidth
3. Under no-load condition.
4. Measured between each DAC output pin and AGND.

6.13.5 DC ADC

T_J = -40 to 150 °C; VDDD,VDDA = 3.3 V ±5 %; GNDD,GNDA = 0.0 V; PLL locked at 44.1 kHz; 12-bit conversions; unless otherwise noted

Parameter	Symbol	Min	Typ	Max	Unit
Resolution	N	5		12	Bits
Differential Non-Linearity (Note 1)	DNL			±0.50	LSB
Center-scale (offset) Error (Note 2)			±3		codes
Gain Error (Note 3)			1.4		%
Gain Drift (Note 3)			56		ppm/ºC
Peak-to-Peak Noise			3		codes
VREF (common mode and center voltage)		1.2	1.3	1.4	V
VREF Drift			100		ppm/ºC
Conversion Range		0		2×VREF	V
On-Channel input resistance:			10		MΩ
Off-Channel leakage current			1		µA
Acquition Time		$\frac{1}{192Fs}$		$\frac{1}{96Fs}$	s
Conversion Time			$\frac{2^N+2}{96Fs}$		s

Notes:

1. Monotanicity guaranteed.
2. The Center-scale error is the value produced when **DCC.CAL** is set.
3. Assumes center-scale error is subtracted out.

7 Pinout and Packaging

All pins configured as digital inputs or must not be left floating; therefore, they must be driven, have pull-ups or pull-downs, or be directly attached to power or ground.

7.1 Pin List

Pins listed as Type EGPIO can be programmed for D_{IN}, D_{OUTZ}, or D_{OUTD}.

Pin	Name	Type	Pin Description
1	MA5	D_{OUT}	External memory address bus bit 5
	IOE5	EGPIO	
2	MA4	D_{OUT}	External memory address bus bit 4
	IOE4	EGPIO	
3	GNDS0		Connect to digital ground
4	GNDD0		
5	VDDD0		
6	MA3	D_{OUT}	External memory address bus bit 3
	IOE3	EGPIO	
7	MA2	D_{OUT}	External memory address bus bit 2
	IOE2	EGPIO	
8	MA1	D_{OUT}	External memory address bus bit 1
	IOE1	EGPIO	
9	MA0	D_{OUT}	External memory address bus bit 0
	IOE0	EGPIO	
10	GNDD1		
11	VDDD1		
12	AD0	D_{IN}	Control Port Address bit 0 input in I^2C mode
	SDIN	D_{IN}	Control Port Serial data input in SPI mode
	URX0	D_{IN}	Control Port data input in USART mode
	IOA2	EGPIO	
13	AD1	D_{IN}	Control Port Address bit 1 input in I^2C mode
	$\overline{CS}$	D_{IN}	Control Port Chip select input in SPI mode
	UFR0	$D_{I/O}$	Control Port frame input/output in USART (synchronous) mode
	IOA3	EGPIO	
*14	SCL	$D_{I/OD}$	Control Port serial clock input in I^2C mode
	SCLK	D_{IN}	Control Port serial clock input in SPI mode
	UCK0	$D_{I/O}$	Control Port clock input/output in USART (synchronous) mode
	IOA0	EGPIO	
15	SDA	$D_{I/OD}$	Control Port serial data input and output in I^2C mode
	SDOUT	D_{OUT}	Control Port serial data output in SPI mode
	UTX0	D_{OUT}	Control Port data output in USART mode
	IOA1	EGPIO	
16	GNDD7		

* Requires a pull-up or pull-down as this pin sets an initial state on power-up (release of $\overline{RST}$).

Table 7-1: Pinout List

Pin	Name	Type	Pin Description
17	MA16	D_{OUT}	External memory address bus bit 16
	IOD1	EGPIO	
18	SX0	D_{OUT}	Host Controller Source Port serial data output 0 (S/PDIF output also)
	IOC2	EGPIO	
19	SX1	D_{OUT}	Host Controller Source Port serial data output 1
	IOC4	EGPIO	
20	GNDD2		
21	VDDD2		
22	FSY	$D_{I/O}$	Host Controller Source Port frame sync
	IOC1	EGPIO	
23	SCK	$D_{I/O}$	Host Controller Source Port serial bit clock
	IOC0	EGPIO	
24	SR0	D_{IN}	Host Controller Source Port serial data input 0 (S/PDIF input also)
	IOC3	EGPIO	
25	SR1	D_{IN}	Host Controller Source Port serial data input 1
	IOC5	EGPIO	
26	TX	D_{OUT}	MOST network transmitter output
	IOG2	EGPIO	
27	RX	D_{IN}	MOST network receiver input
	IOG3	EGPIO	
28	RMCK	D_{OUTZ}	Recovered master clock output
	IOG0	EGPIO	
29	ERR	D_{OUT}	Error flag output
	IOG1	EGPIO	
*30	IOG4	EGPIO	
	BOOT	D_{IN}	Boot-strapping reset vector (selects reset vector address)
	PWM1	D_{OUT}	DSP0 PWM1 DAC output
31	IOB3	EGPIO	
	$\overline{\text{DCS}}$	D_{IN}	Debug Port chip select in SPI mode
	UFR1	$D_{I/O}$	Debug Port frame input/output in USART (synchronous) mode
32	$\overline{\text{RST}}$	D_{IN}	Hardware reset input
33	GNDA0		
34	GNDA1		
35	FLT	A_{IO}	PLL loop filter output
36	GNDS1		Connect to analog ground
37	VDDA0		PLL Power Supply
38	POT7	A_{IN}	DC Measurement ADC analog input 7
39	POT6	A_{IN}	DC Measurement ADC analog input 6
40	POT5	A_{IN}	DC Measurement ADC analog input 5
41	POT4	A_{IN}	DC Measurement ADC analog input 4
42	POT3	A_{IN}	DC Measurement ADC analog input 3
43	POT2	A_{IN}	DC Measurement ADC analog input 2
44	POT1	A_{IN}	DC Measurement ADC analog input 1

* Requires a pull-up or pull-down as this pin sets an initial state on power-up (release of $\overline{\text{RST}}$).

Table 7-1: Pinout List (Continued)

Pin	Name	Type	Pin Description
45	POT0	A_{IN}	DC Measurement ADC analog input 0
46	MIC	A_{AI}	Microphone ADC analog input
47	GNDA2		
48	MPX	A_{AI}	MPX ADC analog input
49	VDDA1		
50	AD2R	A_{AI}	Right analog 2 input to audio ADC
51	AD2L	A_{AI}	Left analog 2 input to audio ADC
52	GNDA3		
	VREFS	A_{AI}	Reference voltage bypass for single-ended DAC operation. Requires two capacitors to ground.
53	VREF	A_{IO}	Voltage reference output
54	GNDA4		
55	AD1R	A_{AI}	Right analog 1 input to audio ADC
56	AD1L	A_{AI}	Left analog 1 input to audio ADC
57	VDDA2		
58	AD0R	A_{AI}	Right analog 0 input to audio ADC
59	AD0L	A_{AI}	Left analog 0 input to audio ADC
60	GNDA5		
61	DA3B	A_{AO}	Inverted analog output of DAC3
62	DA3	A_{AO}	Analog output of DAC3
63	DA2B	A_{AO}	Inverted analog output of DAC2
64	DA2	A_{AO}	Analog output of DAC2
65	DA1B	A_{AO}	Inverted analog output of DAC1
66	DA1	A_{AO}	Analog output of DAC1
67	GNDS2		Connect to analog ground
68	DA0B	A_{AO}	Inverted analog output of DAC0
69	DA0	A_{AO}	Analog output of DAC0
70	GNDA6		
71	TST0	D_{IN}	Test Mode enable. Must be grounded for normal operation.
72	TST1	D_{IN}	Test Mode enable. Must be grounded for normal operation.
*73	IOB0	EGPIO	
	DSCL	$D_{I/OD}$	Debug Port clock input in I^2C mode
	DSCLK	D_{IN}	Debug Port clock input in SPI mode
	UCK1	$D_{I/O}$	Debug Port clock input in USART (synchronous) mode
74	IOB1	EGPIO	
	DSDA	$D_{I/OD}$	Debug Port data I/O in I^2C mode
	DSDOUT	D_{OUTD}	Debug Port data output in SPI mode
	UTX1	D_{OUT}	Debug Port data output in USART mode
75	XTO	A_{IO}	Crystal oscillator output
76	XTI	A_{IO}	Crystal oscillator input - or external CMOS clock input

* Requires a pull-up or pull-down as this pin sets an initial state on power-up (release of $\overline{\text{RST}}$).

Table 7-1: Pinout List (Continued)

Pin	Name	Type	Pin Description
77	GPD3	$D_{I/OD}$ EGPIO	For Host Controller, with Interrupt capability For DSP1
	SRB1	D_{IN}	DSP1, Source Port B1 - SR input
78	GPD2	$D_{I/OD}$ EGPIO	For Host Controller, with Interrupt capability For DSP1
	SXB1	D_{OUT}	DSP1, Source Port B1 - SX output
79	GPD1	$D_{I/OD}$ EGPIO	For Host Controller, with Interrupt capability For DSP1
	FSYB1	$D_{I/O}$	DSP1, Source Port B1 - Frame Sync
80	GPD0	$D_{I/OD}$ EGPIO	For Host Controller, with Interrupt capability For DSP1
	SCKB1	$D_{I/O}$	DSP1, Source Port B1 - Serial Clock
81	GPC3	$D_{I/OD}$ EGPIO	For Host Controller For DSP1
	TMR1	D_{OUTD}	Timer 1 output
	SRB0	D_{OUT}	DSP1, Source Port B0 - SR input
82	VDDD3		
83	GNDD3		
84	GPC2	$D_{I/OD}$ EGPIO	For Host Controller For DSP1
	TMR0	D_{OUTD}	Timer 0 output
	SXB0	D_{OUT}	DSP1, Source Port B0 - SX output
85	GPC1	$D_{I/OD}$ EGPIO	For Host Controller For DSP1
	FSYB0	$D_{I/O}$	DSP1, Source Port B0 - Frame Sync
86	GPC0	$D_{I/OD}$ EGPIO	For Host Controller For DSP1, with Interrupt capability
	SCKB0	$D_{I/O}$	DSP1, Source Port B0 - Serial Clock
87	GPB3	$D_{I/OD}$ EGPIO	For Host Controller For DSP0
	SRA1	D_{IN}	DSP0, Source Port A1 - SR input
88	GPB2	$D_{I/OD}$ EGPIO	For Host Controller For DSP0
	SXA1	D_{OUT}	DSP0, Source Port A1 - SX output
89	GPB1	$D_{I/OD}$ EGPIO	For Host Controller For DSP0
	FSYA1	$D_{I/O}$	DSP0, Source Port A1 - Frame Sync
90	GPB0	$D_{I/OD}$ EGPIO	For Host Controller For DSP0
	SCKA1	$D_{I/O}$	DSP0, Source Port A1 - Serial Clock
91	GPA3	$D_{I/OD}$ EGPIO	For Host Controller For DSP0
	SRA0	D_{IN}	DSP0, Source Port A0 - SR input
92	GNDD4		

* Requires a pull-up or pull-down as this pin sets an initial state on power-up (release of $\overline{RST}$).

Table 7-1: Pinout List (Continued)

Pin	Name	Type	Pin Description
93	GPA2	$D_{I/OD}$ EGPIO	For Host Controller For DSP0
	SXA0	D_{OUT}	DSP0, Source Port A0 - SX output
94	GPA1	$D_{I/OD}$ EGPIO	For Host Controller For DSP0
	FSYA0	$D_{I/O}$	DSP0, Source Port A0 - Frame Sync
95	GPA0	$D_{I/OD}$ EGPIO	For Host Controller For DSP0, with interrupt capability
	SCKA0	$D_{I/O}$	DSP0, Source Port A0 - Serial Clock
96	MD7	$D_{I/O}$	External memory data bus bit 7
	IOF7	EGPIO	
97	MD6	$D_{I/O}$	External memory data bus bit 6
	IOF6	EGPIO	
98	PWM0	D_{OUT}	Pulse-width modulation DAC output
	IOD0	EGPIO	
99	IOB2	EGPIO	
	DAD0	D_{IN}	Debug Port address bit 0 in I^2C mode
	DSDIN	D_{IN}	Debug Port serial data input in SPI mode
	URX1	D_{IN}	Debug Port data input in USART mode
100	GNDS3		Connect to digital ground
101	MD5	$D_{I/O}$	External memory data bus bit 5
	IOF5	EGPIO	
102	MD4	$D_{I/O}$	External memory data bus bit 4
	IOF4	EGPIO	
103	MD3	$D_{I/O}$	External memory data bus bit 3
	IOF3	EGPIO	
104	MD2	$D_{I/O}$	External memory data bus bit 2
	IOF2	EGPIO	
105	GNDS4		Connect to digital ground
106	VDDD4		
107	MD1	$D_{I/O}$	External memory data bus bit 1
	IOF1	EGPIO	
108	MD0	$D_{I/O}$	External memory data bus bit 0
	IOF0	EGPIO	
109	$\overline{\text{MWR}}$	D_{OUT}	External Data memory write signal. Valid SRAM Data write cycle when low.
	$\overline{\text{PMW}}$	D_{OUT}	External Program memory write signal. Low is write cycle, high is read cycle.
	IOF8	EGPIO	
110	$\overline{\text{MRD}}$	D_{OUT}	External Data memory read signal. Valid SRAM Data read cycle when low.
	PA15	D_{OUT}	External Program memory address bus bit 15
	IOE15	EGPIO	
111	$\overline{\text{MRAS}}$	D_{OUT}	DRAM row address select
	$\overline{\text{MCS}}$	D_{OUT}	External data memory (SRAM) chip select
	IOF9	EGPIO	

* Requires a pull-up or pull-down as this pin sets an initial state on power-up (release of $\overline{\text{RST}}$).

Table 7-1: Pinout List (Continued)

Pin	Name	Type	Pin Description
112	MA14	D_{OUT}	External memory address bus bit 14
	$\overline{MCAS}$	D_{OUT}	DRAM column address select
	IOE14	EGPIO	
113	GNDS5		Connect to digital ground
114	GNDD5		
115	GNDD6		
116	VDDD5		
*117	$\overline{XME}$	D_{IN}	External Program memory enable (at Reset)
	$\overline{PCS}$	D_{OUT}	External Program memory chip select
	SA15	D_{OUT}	External Data memory address bus bit 15
	IOF10	EGPIO	
118	GNDS6		Connect to digital ground
119	MA13	D_{OUT}	External memory address bus bit 13
	IOE13	EGPIO	
120	MA12	D_{OUT}	External memory address bus bit 12
	IOE12	EGPIO	
121	MA11	D_{OUT}	External memory address bus bit 11
	IOE11	EGPIO	
122	MA10	D_{OUT}	External memory address bus bit 10
	IOE10	EGPIO	
123	MA9	D_{OUT}	External memory address bus bit 9
	IOE9	EGPIO	
124	MA8	D_{OUT}	External memory address bus bit 8
	IOE8	EGPIO	
125	GNDS7		Connect to digital ground
126	GNDS8		Connect to digital ground
127	MA7	D_{OUT}	External memory address bus bit 7
	IOE7	EGPIO	
128	MA6	D_{OUT}	External memory address bus bit 6
	IOE6	EGPIO	

* Requires a pull-up or pull-down as this pin sets an initial state on power-up (release of $\overline{RST}$).

Table 7-1: Pinout List (Continued)

7.2 Equivalent Schematics for Pins

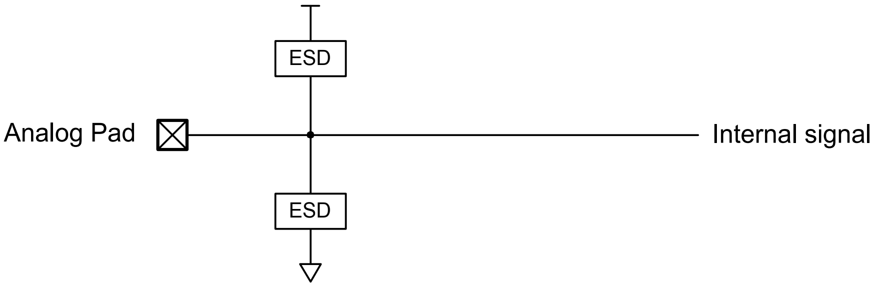


Figure 7-1: Pin-equivalent for Analog IO pin - A_{IO}

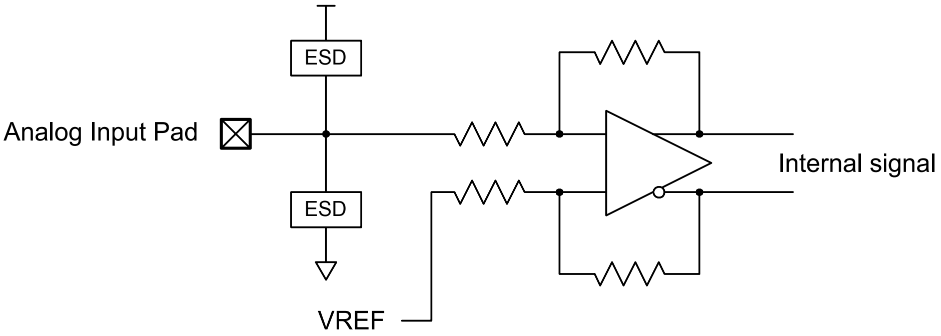


Figure 7-2: Pin-equivalent for Analog Audio Input pin - A_{AI}

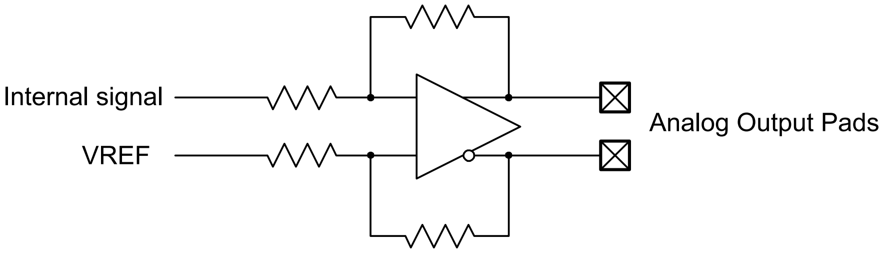


Figure 7-3: Pin-equivalent for Analog Audio Output pin - A_{AO}

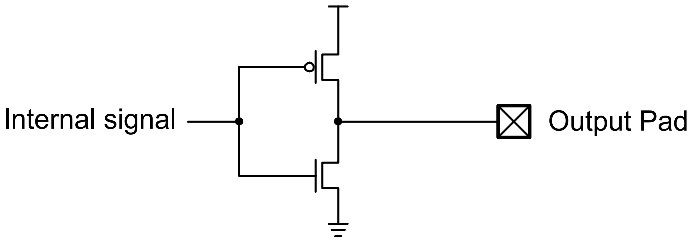


Figure 7-5: Pin-equivalent for Digital Output pin - D_{OUT}

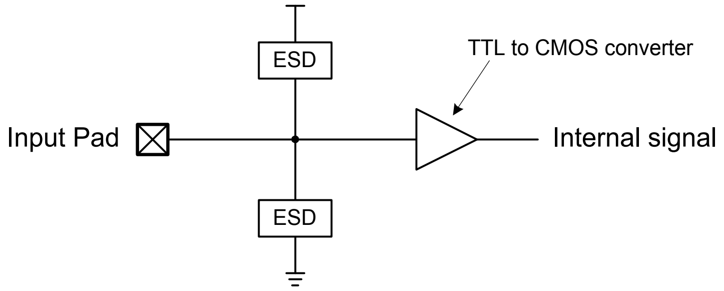


Figure 7-4: Pin-equivalent for Digital Input pin - D_{IN}

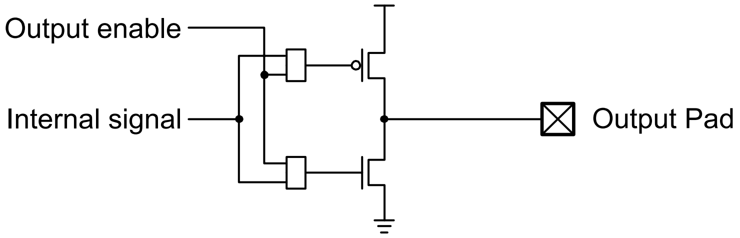


Figure 7-6: Pin-equivalent for Digital Output with high-Z control - D_{OUTZ}

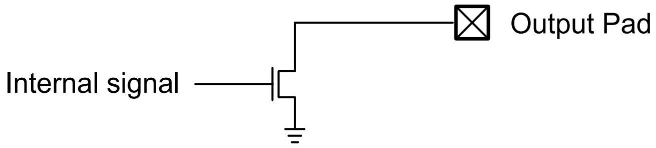


Figure 7-7: Pin-equivalent for Open-Drain Digital Output pin - D_{OUTD}

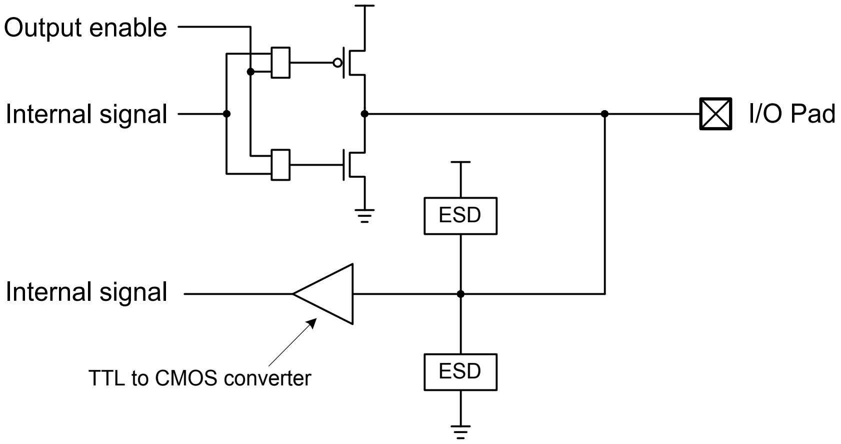


Figure 7-8: Pin-equivalent for Digital I/O pin - $D_{I/O}$

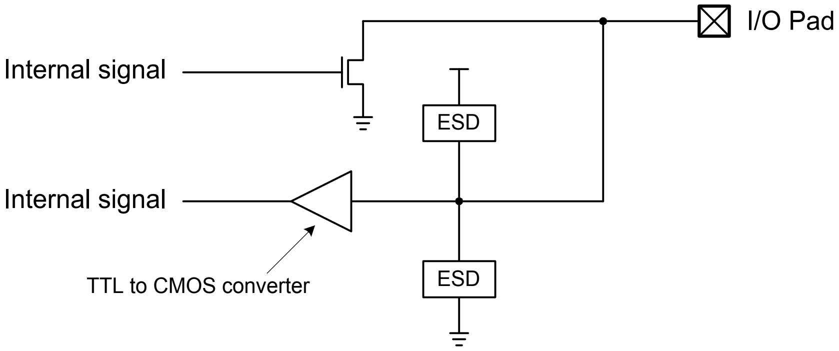


Figure 7-9: Pin-equivalent for Digital Input/Open-Drain Output pin - $D_{I/OD}$

7.3 Pinout

The package designators are:

- lll - Lot Sequence Code
- r - Chip Revision Letter (also see **MMPC.REV[3:0]**)
- yy - last two digits of Assembly Year
- ww - Assembly Work Week

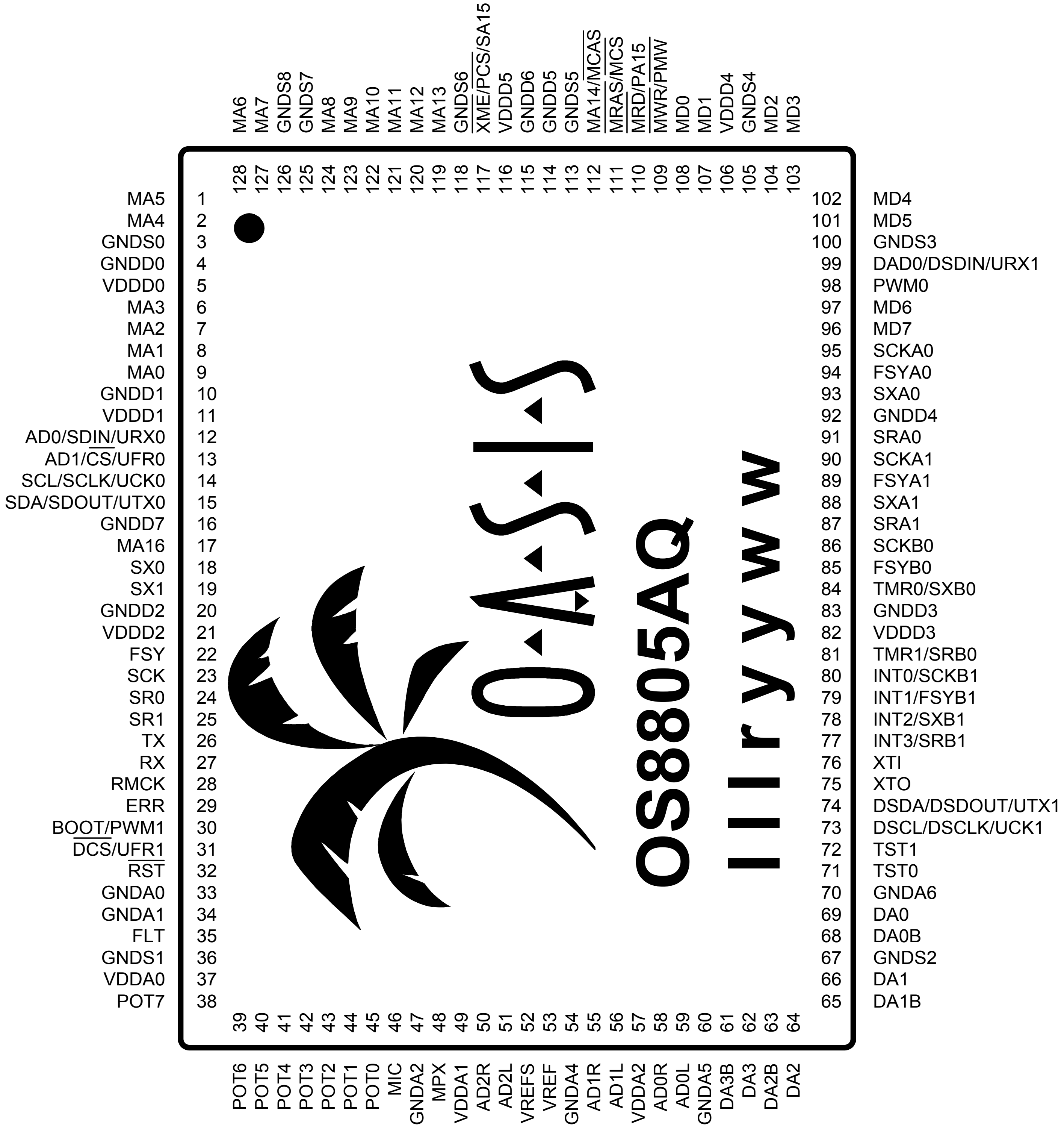


Figure 7-10: OS8805 Functional Pinout

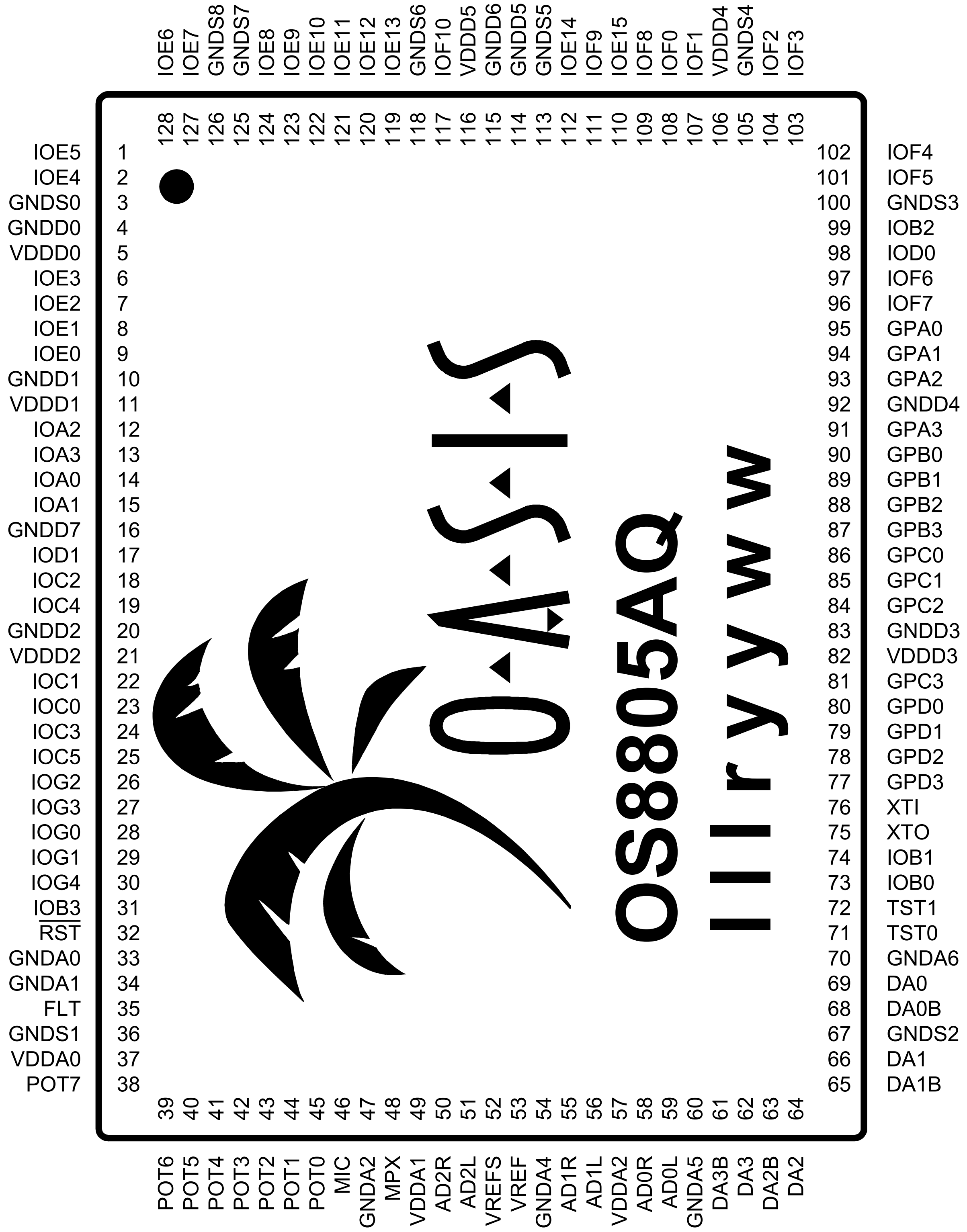
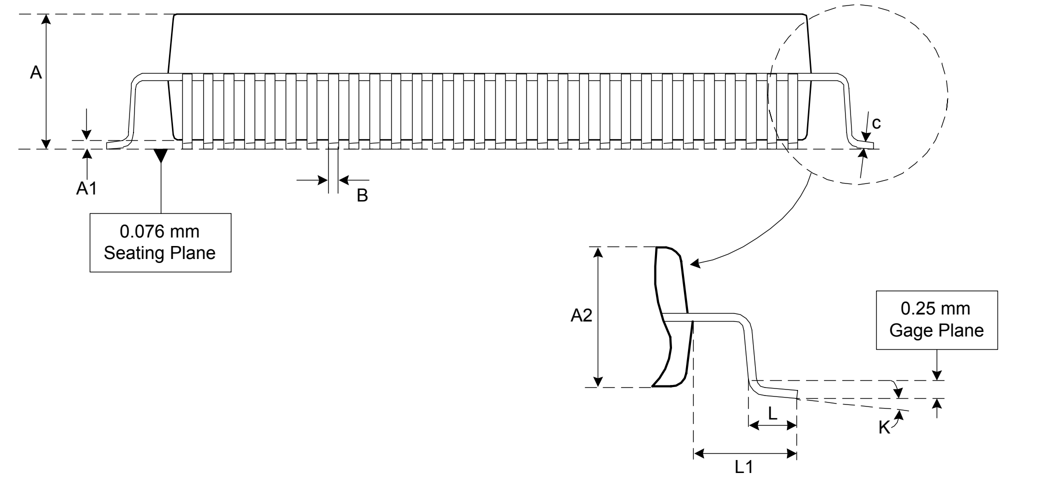
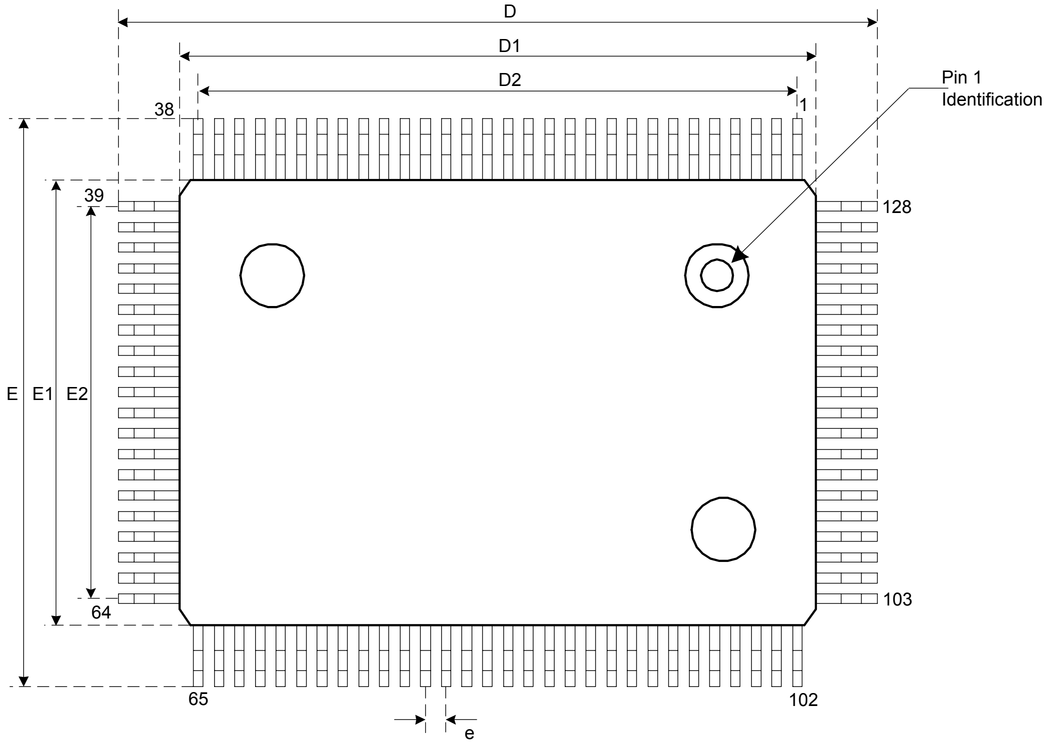


Figure 7-11: OS8805 General Purpose I/O Pinout

7.4 Package Outline, 128-pin MQFP

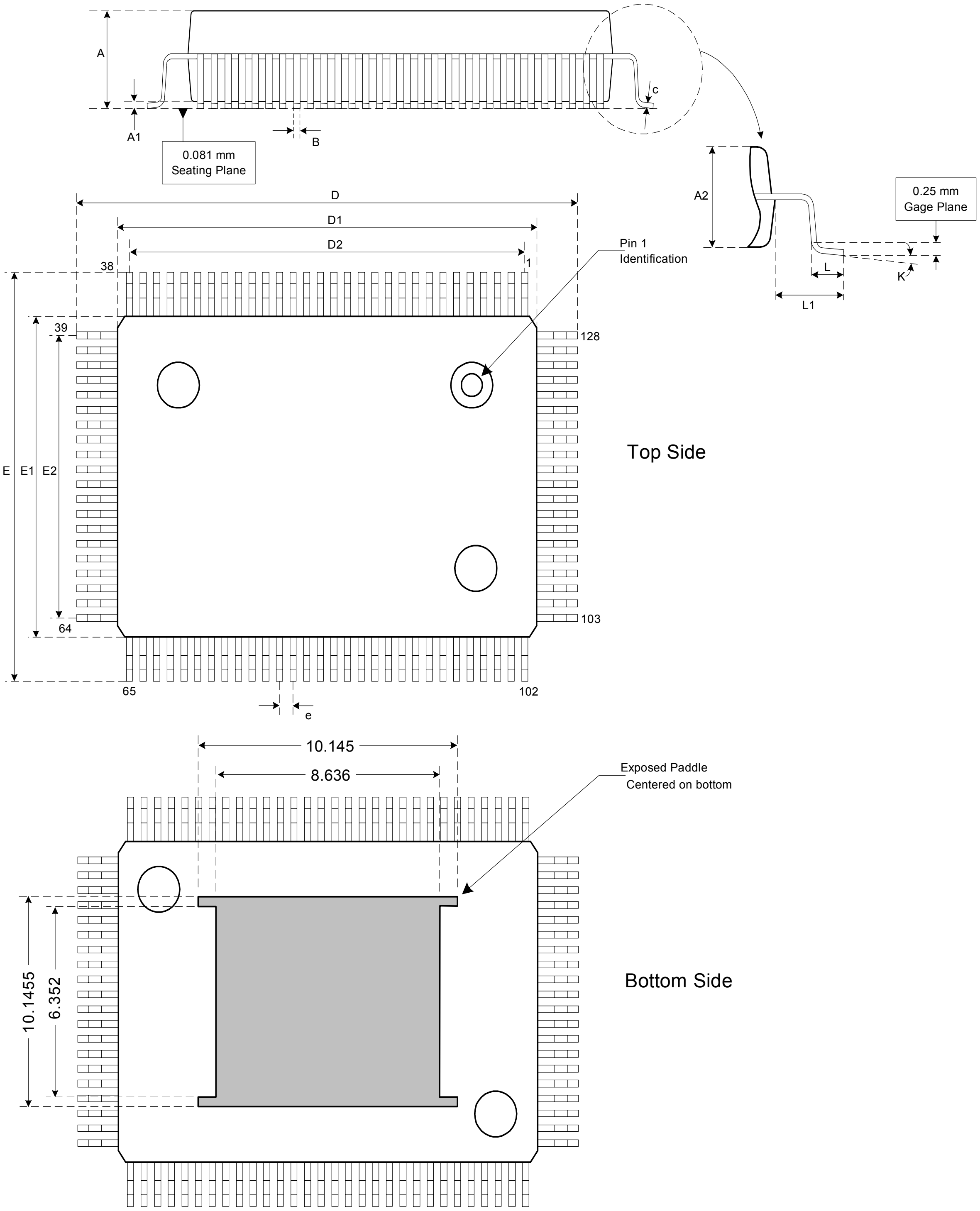




	A	A1	A2	B	c	D	D1	D2	e	E	E1	E2	L	L1	K
Min		0.25	2.60	0.17	0.11	23.0	19.9			17.0	13.9		0.78		0 °
Typ		0.33	2.70	0.22		23.2	20.0	18.5	0.50	17.2	14.0	12.5	0.88	1.60	3.5 °
Max	3.40	0.50	2.80	0.27	0.17	23.4	20.1			17.4	14.1		1.03		7 °

Table 7-2: Package Outline Dimensions (mm)

7.5 Package Outline, 128-pin ETQFP



	A	A1	A2	B	c	D	D1	D2	e	E	E1	E2	L	L1	K
Min		0.05	1.35	0.19		21.80	19.90			15.80	13.90		0.50		0 °
Typ			1.40			22.00	20.00	18.5	0.50	16.00	14.00	12.5	0.60	1.00	
Max	1.60	0.15	1.45	0.27		22.20	20.10			16.20	14.10		0.75		7 °

Table 7-3: Package Outline Dimensions (mm)

8 Application Information

8.1 Power Supplies and Analog Components

The following Figure illustrates the standard power arrangement for the OS8805. The 0.1 µF capacitors should be placed as close as possible to the appropriate power supply pins.

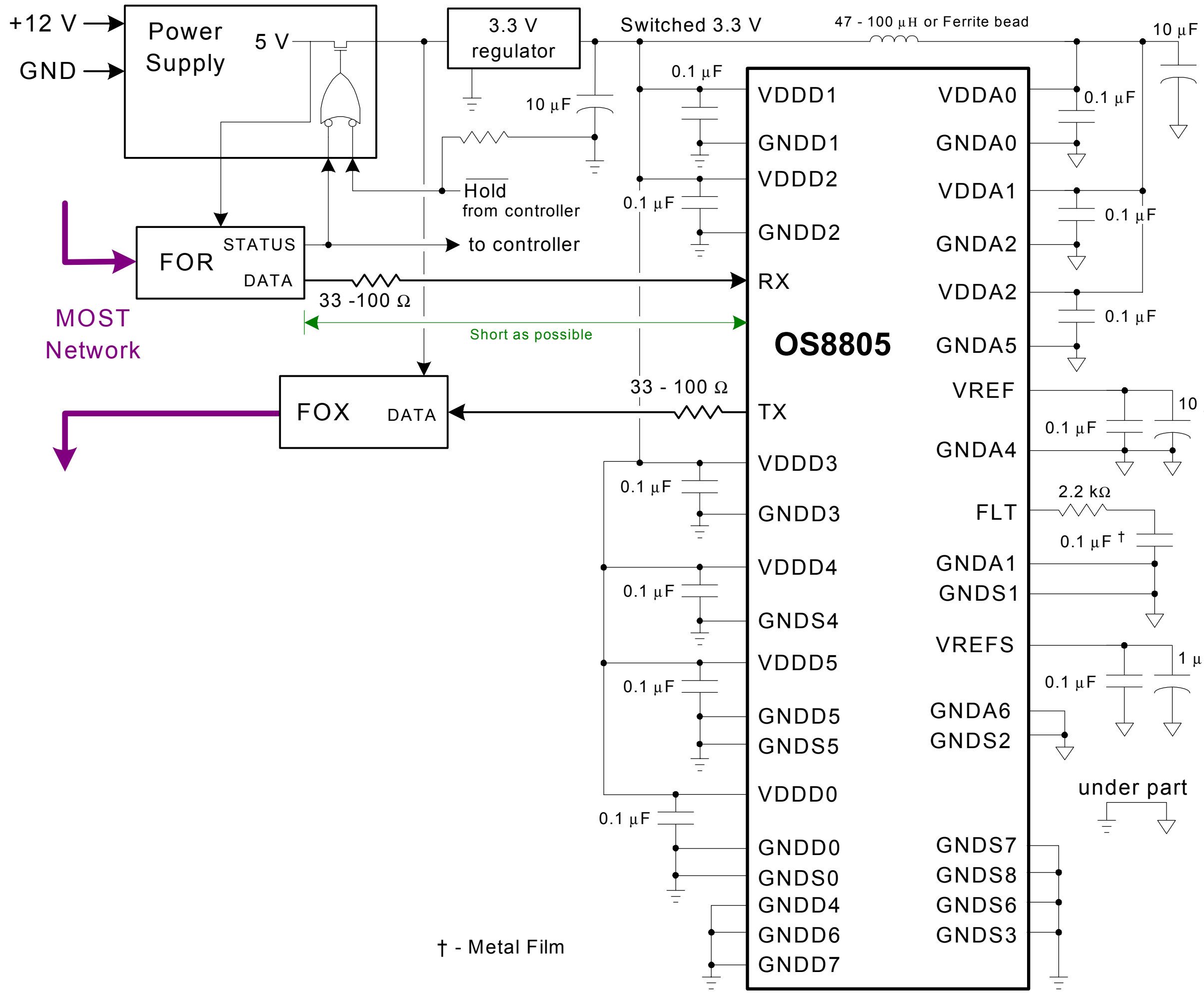


Figure 8-1: Power Supply Overview Diagram

The **FLT**, **VREFS**, and **VREF** components should be placed as close as possible to the **GNDAn** pins to minimize loop currents. To minimize vibration and shock effects on PLL locking, a metal-film capacitor (such as Panasonic ECPU 16VDC / 0805 / ECPU1C104MA5) should be connected to the **FLT** pin through the series resistor. Ceramic capacitors are more sensitive to shock and could cause unlock events in high-vibration environments. In addition, the **FLT** pin is a high-impedance node; therefore, leakage current should be kept below 1 µA, or average pulse-width distortion tolerance could be adversely affected. Conformal coating is recommended for systems where condensation can occur. The analog and digital ground planes, if separated, should be connected at one point on the board, under the part.

The distance between the Fiber Optic Receiver (FOR) unit and the OS8805 should be as short as possible to minimize capacitance on the DATA line. Minimizing capacitance will shorten transition times out of the FOR, thereby minimizing pulse width distortion and jitter.

The series resistor in the **TX** and **RX** paths are for series-termination and should be as close as possible to the transmitting end. Since the **RX** input is 5 V tolerant, even when power is not applied, the series resistor value should be selected based on the board layout and should match the line impedance, which will help minimize reflections and lower EMI.

In the Figure above, the FOR unit controls the power to the node. All power supplies, except the FOR are on the *Switched 3 V* circuit. When there is no light at the FOR input, the STATUS output turns off the switched power supply to minimize power drain. The OS8805 has the option to delay power-down to allow an orderly shut-down of the local node. The OS8805 can apply power control (Hold) once normal operation has commenced. When the FOR loses light, STATUS goes high telling the OS8805 to perform a shut-down sequence. When the OS8805 has finished an orderly shut-down, it releases the Hold power control, causing the node to go into full power-down. The circuit illustrated is a conceptual schematic and does not include the decoupling needed between powered-off and powered devices.

Figure 8-2 illustrates a more detailed typical power supply arrangement for the OS8805. This diagram is provided as an example to illustrate how the OS8805 can be utilized to meet the *MOST Specification, Power Supply Area* (V2.1, Section 4.6), and is not guaranteed to meet all the requirements of a specific OEM.

Similar to the *Power Supply Overview Diagram*, Figure 8-1, the FOR controls power to the OS8805 and the rest of the node. The MIC5236 regulator supplies the 5 V continuous power (5Vc) to the FOR as well as the power-up circuit for the rest of the system. The power-up circuit controls the power supply for a 5 V supply (5Vs), the OS8805 3 V supply (3Vs), and a 12 V switched supply (12Vs) that can be used for high-powered peripherals. The 5 V switched supply powers the FOX as well as any analog circuitry that cannot run from the 3 V supply. The time between the FOR STATUS line going low and the 3Vs/5Vs power being stable should be well within t_{WakeUp} time (currently 6 ms), as listed in the MOST Spec. (V2.1, Section 3.8) for *light at FOR input to light out at FOX output* time.

The **IOxn** pin, selected for the HOLD signal, must power up as an input. Since the OS8805 IOxn pins are 5 V tolerant (even with no applied power), no extra protection is needed while the OS8805 is powered down. Once the part is powered up, the IOxn should be configured as a digital output (CMOS-drive) and driven high to "hold" the power-supply from turning off when light disappears at the FOR. The **bCM2.NNAC** bit indicates when Network activity is lost, and should be periodically checked during normal operation. When **bCM2.NNAC** goes high, the node should perform an orderly shut-down and then drive HOLD low to allow the entire node to power-down.

The **POTn** input uses the DC ADC to measure the 12 V supply and react appropriately to the MOST-defined levels (MOST Spec. V2.1, Section 4.7):

- Normal Operation
- Super Voltage
- Critical Voltage
- Low Voltage

The reaction to different power levels varies with the application but could include powering down power-amps or motors in over- or under-voltage conditions to protect the devices.

The OS8805 internal Watchdog timer supports the Watchdog requirements of the MOST Spec (V2.1, Section 4.6). If the Watchdog timer register, WDT, isn't written before the timer counts down to 0, then the OS8805 is reset, causing the MOST Transceiver to switch to all-bypass mode (**bXCR.$\overline{\text{ABY}}$** clear). In addition, assuming the IOxn pin selected is an input at power-up, the Watchdog reset will release the HOLD signal.

The **$\overline{\text{RST}}$** pin illustrated is controlled by a MAX821 Voltage monitor. This circuit is only required if the node must power up and initialize without any external stimulus on the **RX** or one of the **GPAn** pins. When the OS8805 initially powers up, the **cMCS.PD** bit will be set by the internal Voltage monitor circuit. The part will remain in power-down mode until two transitions occur on the **RX** or **GPAn** pins, as configured in the RGEN register. If the node must do some initialization at power-up, a transition on the **$\overline{\text{RST}}$** line will clear **cMCS.PD** causing the OS8805 to power up. The MAX821 illustrated has a nominal threshold of 2.93 V. The SRT pin

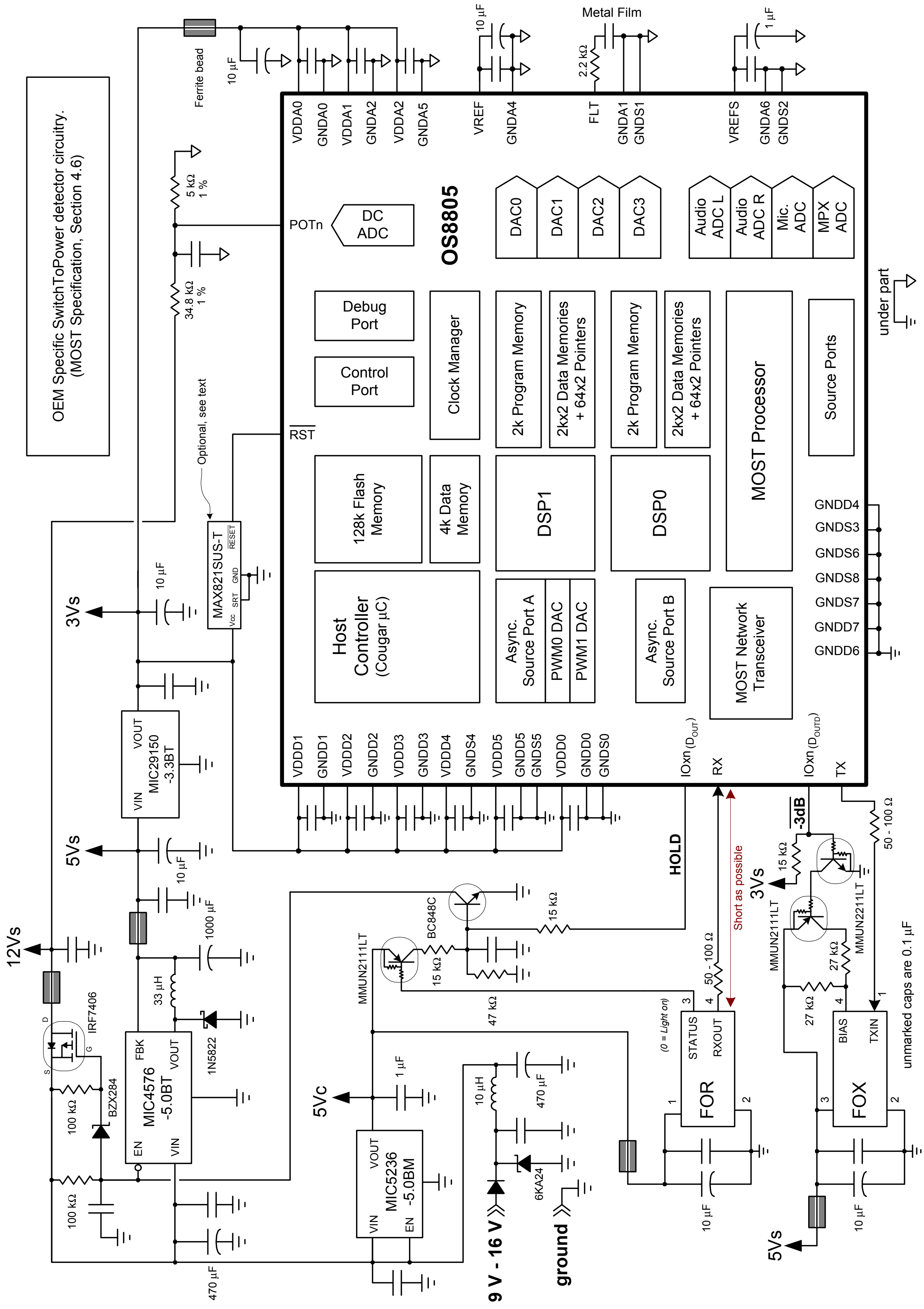


Figure 8-2: Detailed Power Supply Arrangement

is set for a 1 ms $\overline{\text{RST}}$ low time. The $\overline{\text{RST}}$ low time should be small to allow the maximum amount of software initialize time (and power-supply ramp time) to meet the *FOR light-on to* **bXCR.$\overline{\text{ABY}}$** *set* ($t_{\text{WaitNodes}}$ in the MOST Spec. V2.1, Section 3.8 - currently 100 ms) time period.

The Switch-To-Power circuitry is not included since it varies with OEMs. The OEM should be contacted to determine the proper method to support Switch-To-Power. The Switch-To-Power detector is a diagnostic mode used to force the node into ring-break diagnostics (MOST Spec. V2.1, Section 3.2.2.4). If the ring is broken, then light will not be received at the node immediately down-stream from the break. No light causes the node to power-down. When diagnostics needs to be performed on the broken ring, a method is needed to override the power-down and cause the node to power-up even though no light is received at the FOR. This override event is labeled Switch-To-Power.

8.2 Power Supply Current

The power supply current can vary widely with different applications. The *DC Characteristics* table on page 200 gives typical and maximum values for the power supply current. The difference between the typical and maximum values shows how the power supply current depends on the application running on the device. The maximum figure was generated using a program that exercises all portions of the device with the maximum number of bit transitions per cycle. The typical value shows the power supply current when the device is running firmware that represents an actual application, using the Host Controller, both DSPs, the ADCs and DACs. This value is further broken down into the current used by each DSP, by the ADCs and DACs, and by the Host Controller.

The expected current for a given application can be calculated in either of two methods: the first is to add the supply current for each module used, and account for the temperature, power supply, and process factors given in *DC Characteristics*. For example, an application that uses one DSP, the ADCs and DACs, running at 85 °C and a supply voltage of 3.465 V would have an expected maximum digital supply current of:

$$((95\text{ mA} + 35\text{ mA} + 85\text{ mA}) + (0.18\text{ mA/°C}) \times (85\text{ °C} - 70\text{ °C}) + (133\text{ mA/V}) \times (3.465\text{ V} - 3.3\text{ V})) \times 1.05 = 252\text{ mA}$$

plus an expected maximum analog supply current of:

$$(90\text{ mA} + (0.03\text{ mA/°C}) \times (85\text{ °C} - 70\text{ °C}) + (5\text{ mA/V}) \times (3.465\text{ V} - 3.3\text{ V})) \times 1.09 = 100\text{ mA}$$

for a total supply current of: 252 mA + 99 mA = 352 mA

The second approach uses the supply current measured in the user's application, running in the user's target system or on an Oasis SiliconSystems evaluation board. After subtracting the current used by any peripheral circuitry, such as amplifiers and memories, the measured digital and analog currents should be multiplied by the factors above to account for temperature and process variables to give the maximum expected supply current for that application. If the application firmware should change, then Table 8-1 can be used to predict the effect that the changes will have on the power supply current. Table 8-1 gives factors for the relative power supply current due to changes in the device configuration. For example, if the user measures the supply current for a given application, then changes the sample rate from 48 kHz to 44.1 kHz, the new supply current would be the measured current times (0.94). In another application, the if the user measures the supply current, then changes the DSP clock rate from 1344Fs to 960Fs, the new supply current would be the measured supply current times (0.81/0.94 = 0.86).

<table>
<tr><th>Parameter</th><th colspan="12">Power Factor</th><th>Test Conditions</th></tr>
<tr><td>Fs</td><td colspan="6">44.1 kHz = 0.94</td><td colspan="6">48 kHz = 1</td><td>3.3 V, 1344Fs</td></tr>
<tr><td>VDDD, VDDA</td><td colspan="6">3.36 V = 0.94</td><td colspan="6">3.465 V = 1</td><td>1344Fs, 44.1 kHz</td></tr>
<tr><td>DSP Clock Rate</td><td colspan="3">768Fs = 0.75</td><td colspan="3">960Fs = 0.81</td><td colspan="3">1344Fs = 0.94</td><td colspan="3">1536Fs = 1</td><td>3.3 V, 44.1 kHz</td></tr>
<tr><td>DSPs Enabled</td><td colspan="4">0 = 0.47</td><td colspan="4">1 = 0.74</td><td colspan="4">2 = 1</td><td>3.3 V, 1344Fs</td></tr>
</table>

Table 8-1: Power Supply Current Derating Factors

Note that these are approximations based on a given application. Actual results will depend on the application firmware.

8.3 Estimating Junction Temperature

Chip specifications and reliability are based on the junction temperature (T_J) of the device; that is, the temperature of the die. Because the die is in a package on a circuit board, the junction temperature can't be known exactly, but it can be estimated based on system measurements. For plastic packages, the most accurate methods are based either on the ambient temperature, or on the temperature of the top of the package. The heat flow from junction to case (θ_{JC}) is no longer used, because the measurement technique assumes that most heat flow is out of the top of the package. This isn't true for plastic packages: most of the heat flows out through the leads.

If ambient temperature is measured, the familiar constant θ_{JA} is used to estimate T_J. This parameter is measured with the part mounted on a standard board, with a thermal probe below the device. The board is mounted in a 1 m^3 enclosure with a measured air flow. θ_{JA} is calculated from the equation:

$$\theta_{JA} = \frac{T_J - T_A}{P}$$

where T_A is the ambient temperature and P is the total of the analog and digital power dissipation of the device. Solving for T_J gives:

$$T_J = \theta_{JA} \times P + T_A$$

For the OS8805 in a typical application, the total power supply voltage and current might be:

V_{DD} = 3.465 V, I_{DD} = 352 mA

giving a power of:

$P = I_{DD} \times V_{DD} = 1.2\ W$

For an OS8805 in a standard package at an ambient temperature of 85^{o}C on a four-layer board, the estimated junction temperature would be (referring to *Thermal Characteristics* on page 199):

$T_J = 38\ °C/W \times 1.2\ W + 85\ °C = 131\ °C$

Estimating T_J with this technique in an application may be inaccurate due to differences between the test environment and the system being measured. The heat from other devices, the size and shape of the board and of the enclosure all corrupt the measurement. A more accurate measure of T_J can be made through the parameter Ψ_{JT}. The formula for Ψ_{JT} is:

$$\Psi_{JT} = \frac{T_J - T_T}{P}$$

where T_T is the temperature measured on the top of the package. The power dissipation of the device is calculated as above. Solving for T_J gives:

$$T_J = \Psi_{JT} \times P + T_T$$

For the OS8805 in the example above, the total power was:

$P = I_{DD} \times V_{DD} = 1.2\ W$

If T_T were measured at 128 °C, the estimated junction temperature would be (referring to *Thermal Characteristics*):

$T_J = 2.5\ °C/W * 1.2\ W + 128\ °C = 131\ °C$

It's possible to get a more accurate estimate of the junction temperature using Ψ_{JT} because the thermal probe is fastened to the device, so variations in the surrounding environment are less significant. Taking the actual measurement can be more difficult since it has to be done for every device for which an estimate of T_J is needed.

For both parameters, the number of layers in the PCB makes a significant difference. Both parameters are given for two-layer and four-layer boards to lessen this effect. In addition, there is a special board for devices in an exposed paddle that simulates the ground pad that would be underneath the device.

For more information, please refer to EIA/JEDEC standards JESD51-1, -2, -3, -5 and -7.

8.4 Crystal Oscillator Selection

Several factors must be considered when selecting a crystal. These include load capacitance, oscillator margin, cut, and operating temperature.

Oscillator margin is a measure of the stability of an oscillator circuit, and is defined as the ratio of the oscillator's negative resistance (R_{NEG}) to the crystal's ESR (R_{ESR}), or:

$$\text{Margin} = \frac{|R_{NEG}|}{R_{ESR}} = \frac{|R_{VAR}| + R_{ESR}}{R_{ESR}}$$

The negative resistance can be measured by placing a variable resistor (R_{VAR}) in series with the crystal and finding the largest resistor value where the crystal still starts up properly. This point would be just below where the oscillator does not start-up or where the start-up time is excessively long.

Ideally oscillator margin should be greater than 10, and should be at least 5. Smaller oscillator margin can affect the ability of the oscillator to start up.

The load capacitor, specified when ordering the crystal, is the series combination of the capacitance on each leg of the crystal. This capacitance includes not only the added capacitors, but also PCB trace (shunt) capacitance and chip pin capacitance. Larger capacitors also have a negative affect on oscillator margin. In general, the external capacitors on each leg (C1 and C2) should be in the 12 to 22 pF range.

Name	Value	Description
Correlation	Parallel Resonant	Mode of oscillation
Osc. Mode	Fundamental	Oscillation mode or Operation mode.
C_L	16-20 pF	Recommended Load Capacitance.
ESR	 40 Ω 20 Ω	Recommended Maximum Equivalent Series Resistance: When crystal frequency is 256Fs or 384Fs When crystal frequency is 512Fs
Drive Level	50 µW	Typical Drive Level
T_A	-40 to 85 °C	Operating temperature range
cut	AT	AT cut produces the best temperature stability.
Tolerance	±50 ppm	Frequency tolerance at 25 °C. Typical value.

Table 8-2: Crystal Oscillator Specifications

The crystal cut and tolerance value listed in Table 8-2 are typical values and may be changed to suit differing system requirements. Higher ESR values, than those listed in the Table, run the risk of having start-up problems and should be thoroughly tested before being used. Contact the crystal manufacturer for more information.

Appendix A: OS8805 vs. OS8804

This section is divided into two parts: Variances to the OS8804 chip and new features on the OS8805.

A.1 Variances to the OS8804

The following are differences between the OS8804 and the OS8805 that may cause changes in code or PCB development.

- Host Controller:
  - Debug Port Interrupt is maskable on the OS8805 - and powers up disabled (**IER.DIEN**).
  - The OS8805 has a Watchdog timer that powers up enabled. Therefore, if an application does not want to use the Watchdog timer, it must explicitly set **RGEN.WDD**. The OS8804 did not have an internal Watchdog timer.
  - The OS8805 part ID **MMPC.ID** is 0010 (the OS8804 is 0001).
  - When using external Program memory with the Host Controller, the default setting for **MMPC.PCSC** is set in the OS8805 and clear in the OS8804.
  - The External Memory select bit $\overline{\textbf{XME}}$ has moved to the RGEN register (was in XMC on the OS8804).
  - PGMP, which holds the most significant address bits, must be set properly when accessing Program memory using AR1 indirectly.
  - When pushing or popping registers from the stack, the upper memory bits for SPC and DSPC are contained in new registers SPCH and DSPCH, respectively; which must also be pushed/popped.
  - When reading Program memory, PGMP must be programmed with the upper address (page) bits.
  - Interrupt Vector table changed from 3-byte to 4-byte vectors
  - The 3-byte long jump conditional instruction (11010xxx) changed from absolute to relative addressing
  - **GP16-GP19**: These GPIO pins are converted to EGPIO and supported through different registers
    GP16 is **IOB1**, **GP17** is **IOB0**, **GP18** is **IOB3**, and **GP19** is **IOG4**
    GPC.GPHIEN is no longer supported/needed.
    GPIO2 register (3Ch) no longer exists - see EGPD1 and EGPD3.
    DDR2 register (3Bh) has changed functionality and is now GCTL - see EDD1 and EDD3.
  - When using the Debug Port, the $\overline{\textbf{DINT}}$ pin does not exist on the OS8805.
  - When using external Program memory for the Host Controller, the OS8805 has 17 address bits, whereas the OS8804 had 16 address bits (**A16** is new).

- DSPs:
  - Pointer memory changed from 24 to 25 bits wide, so the Address field supports the full address range of Data memory. Therefore, the Mod and Update fields are shifted up by one bit.
  - Although the Pointer memory address field increased to 11 bits, the pointer update ALU's are only 10 bits; therefore, the upper address bit selects between two 1K pages of Vector memory and pointer updates will not cross this page boundary. Vector memory data arrays should not cross the 1K page boundary if the pointers used to access the array use pointer updates to move the pointer.
  - Instruction change. The non-delayed jump to subroutine returns to the instruction immediately after the sub. call (in the OS8804 it returned to the fourth instruction after the call).
  - Instruction change. The `TRAP` instruction changed to call the (new) Debug interrupt vector instead of clearing the **DnPCR.RUN** bit.
  - For DSP0, **GP8** (**GPC0**) is no longer connected, and **IER.DD** and **IER.GPIO** are not supported. For DSP0 GPIO interrupt capability, **GPA0** can be used. See Figure 4-15.
  - For DSP1, although **GP9** (**GPC1**) is connected to DSP1, it can no longer generate an interrupt, and **IER.DD** and **IER.GPIO** are not supported. For DSP1 GPIO interrupt capability, **GPC0** can be used. See Figure 4-15.
  - Once **RGEN.**$\overline{\textbf{XME}}$ and **MMPC.**$\overline{\textbf{XMQ}}$ are set for DSP0 external memory port, the port must be explicitly enabled by setting **GCTL.EDMEN**.

A.2 OS8805 New Features

The following are features that are new to the OS8805 (not in the OS8804) and can be used, if desired.

- Host Controller:
  - Host Controller Program memory size is expanded to 128k Flash (was 64k ROM).
  - Host Controller Data memory size is expanded to 4k bytes (was 2k).
  - Internal Program memory can be written by the Host Controller when **RGEN.$\overline{\text{XME}}$** and **MMPC.$\overline{\text{XMQ}}$** are set for DSP0 external memory port.
  - EGPIO support added to most digital pins.
  - Program Counter based registers have new high registers to support the added address range: SPCH and DSPCH, and PGMP for Program memory accesses.
  - Added instruction - new 4-byte long conditional jump (1100111x) which supports absolute addressing over the entire address range.
  - Added instruction - Exchange ACCH/ACCL (10011111).
  - Added instruction - Clear ACC (11000000).
  - The Control and Debug Ports can be operated as serial port masters for I^{2}C and GSPI formats.
  - The Control and Debug Ports can be operated in UART or USART formats.
  - Added Debug COM ports to each DSP.
  - Added a Watchdog timer (WDT).
  - Added Reset Generation logic for low-power wake-up support (RGEN).
  - Added a Power Supply Voltage monitor.
  - When **RGEN.$\overline{\text{XME}}$** and **MMPC.$\overline{\text{XMQ}}$** are set for DSP0 external memory port, Host Controller writes to Program memory write the internal Flash memory or erase the entire page, based on **FMC.PER**.
  - An optional software-controlled timing-master jitter tolerance circuit that can improve jitter tolerance of a timing master node dramatically (using bits in GTCL).
  - Added support for Packet Data transfer across the MOST Network (and added **RCS.AINT**).
  - When executing the `RETI` instruction, **SR.GIE** optionally gets set, based on the **IER.DRI** bit.

- DSPs:
  - Increased Program memory to 2k (from 1280 words).
  - Increased Left/Right Vector memories to 2k (from 512 words).
  - Increased Left/Right Pointer memories to 64 (from 32).
  - The External Data memory port is expanded to support 128k of SRAM memory.
  - Added support for doubling the throughput between the DSPs and the Routing bus, by remapping other ports.
  - Added a FIFO port between DSPs.
  - Added Dual Async. Source Ports between each DSP and the external system.
  - Added an extra PWM DAC (PWM1) to DSP0.
  - Added a Debug Interrupt vector (0x050) for the `TRAP` instruction as well as Host Controller debug interrupts (**DDnCS.TRINT**).
  - Each DSP has access to eight EGPIO pins, if not used for the Async Source Ports.
  - Both DSPs can control the volume for the ADCs and DACs.

- Source Converters:
  - Each Audio DAC can be operated in a single-ended mode (**ADACn.SEDE**). If any DAC is set for single-ended mode, **VREFS** must be decoupled properly.
  - Each Audio DAC can be software muted (**ADACn.MUTE**).

Appendix B: Revision History

Sections	Description of Changes
DS8805AP1d4	
1d4	1. First Data Sheet. Draft Version
DS8805AP2	
	1. ERROR: DSP0 External Data Memory Port. **XMC.RCP[1:0]** decoding was wrong 2. Clarified DSP0 External Data Memory Port timing 3. Clarified Host Controller IFL writes - only when **SR.GIE** is clear. 4. Changed Flash 1K segments from “pages” to “partitions”, to differentiate from Program or Data memory pages (256 bytes), such as **SR.AP[3:0]** 5. ERROR: Corrected Control/Debug Port UART Parity **UnC.PMD[1:0]** bits. 6. Documented Flash Power-down bit **FMC.FPD**. 7. Documented **GCTL.RFPR** and **GCTL.MJCE** for advanced timing-master software jitter control. 8. Documented DSPs **DTC.T1RSP** and DPRE1 - for aligning S/PDIF transmit and receive interrupts 9. ERROR: Table 4-30 on page 159 was incorrect. The **FSYxn** and **SCKxn** pins were swapped with respect to the associated GPIO pins. 10. Clarified I^2C master operation, and GPIO usage as master chip select. 11. Removed Debug Port $\overline{\textbf{DINT}}$ pin from documentation. 12. ERROR: **RGEN.WDD** bit can be written to 1 or 0. 13. Flash protection covers reading and writing, not just reading Flash memory. 14. ERROR: When the DSP **DSnC.SDOEN** bit is low, the pin is high impedance, not always low. 15. Fixed DSP ASP tables with respect to configuration settings. 16. ERROR - Control and Debug Port SPI master mode - Chip select not needed or used, and STR and STOP bits not active. 17. Added timing for the Control/Debug Ports - master mode, USARTs, and DSP ASPs. 18. Changed Control Port SPI slave timing to include unlock conditions.
DS8805AP3	
	1. Added Detroit’s and changed Austin’s OSS office address
Chapter 2	2. Added that the part powers up in power-down mode, **CMCS.PD** set. 3. EGPIO sticky bits changed to indicate that it does not capture edges, but captures high/low pulses. 4. Added that when the Host Controller’s GPIO are configured as outputs, the corresponding DSP’s IPOT register bits determine whether the output is open-drain or driven in both directions. 5. Added note in SR register description that SR cannot be written when **SR.GIE** is set. 6. Control Port I^2C master operation clarified. 7. Added Packet Data support bits **RCS.AINT**, and **MMPC.RFS1** 8. Added *EGPIO Enable Summary* table which includes power-up state and how to enable GPIO. 9. Clarified that two transitions (falling then rising) are required to clear the power-down state (**CMCS.PD** set), based on the RGEN settings.
Chapter 3	10. MOST registers reordered and MOST Control Message section expanded to include System Control message data 11. Added Packet Data Transfer section and associated bits, and **bNC.APREN**.
Chapter 4	12. Added COM Port interrupt mask (bit 4 - **DCS.CIM**). 13. EGPIO sticky bit modified to indicate that it does not capture edges, but captures high/low pulses, and that IPOT reg. controls output type for DSP or Host Controller. 14. Added note in IER register that Interrupt Priority bits cannot be changed when interrupts are enabled.
Chapter 8	15. Added more complete power supply description and an example power supply schematic.

Table B-1: Data Sheet Revision Summary

Sections	Description of Changes
Section A.1	16. Added OS8804 variance that SPCH/DSPCH must be pushed or popped from the stack when SPC/DSPC are pushed or popped.
DS8805AP4	
Chapter 2	1. Added **IER.DRI** bit 2. Added that **IOG3** (**RX**), when configured as a GPIO output, only supports open-drain (CMOS not supported). The **IPOT3.GPPTG3** bit has no effect on **IOG3**. 3. Changed the **UnC.CKSL[1:0]** = 11 setting to reserved. 4. Clarified **VREFS** pin and **DACn.SEDE** usage.
Chapter 3	5. Clarified usage of Control Message Transmit and Receive buffers. 6. Added Routing Limitations Section to MOST Processor chapter
Chapter 4	7. The DDIV0 and DDIV1 DSP timer divider registers contain only 15 usable bits, **DDIVn[14:0]**. The MSB must be set to 0.
Chapter 5	8. Clarified **VREFS** pin and **DACn.SEDE** usage.
Chapter 6	9. Changed operating temperature range from Ambient to Junction 10. Clarified Maximum Input Voltage in Section 6.1 for Digital I/O pins configures for inputs vs. outputs. 11. Updated thermal data 12. Changes operating supply variation to 5 % 13. Updated power consumption and PWD/Jitter numbers 14. Added Flash program/erase cycle numbers 15. Updated Analog specifications
DS8805FP5	
Chapter 2	1. When the Control or Debug Ports are configured for I^{2}C mode, added comment about initial power-up glitch on **SDA**, and added comment that **CPS.STOP** bits can occur before receiving the first start bit and address byte. 2. Added comments about the **SCK** requirements for GTR to work properly.
Chapter 3	3. Added comments about the **SCK** requirements for GTR and Peripheral Routing to work properlyCorrected bGA default value.
Chapter 4	4. Added comments about the **SCK** requirements for GTR and MOST routing to work properly 5. Removed **DS0C.CHAIN** (bit 6) as it does nothing
Chapter 6	6. Changed Ψ_{JT}, θ_{JA} specs. 7. Changed jitter tolerance spec. from the max to the min column; changed values for Pulse Width Variation and Average Pules Width Distortion. 8. Changed External Program Memory Interface, **MMPC.PCSC** = 1, t_{PCSC} to 115 from 126 ns; changed minimum values for t_{wpds} for **MMPC.PCSC** = 0. 9. Changed External Data Memory Interface minimum timing for t_{mars}, t_{macs}, t_{rascas}, t_{rash}, t_{casl}, t_{cash}, $t_{caslrash}$, t_{mdws}, t_{rrasl}, t_{rcasl}; changed typical timing for t_{rash}. 10. Changed minimum timing for t_{mcsh}, t_{rdl}, t_{rdh}, t_{maws}, t_{mdws}, t_{mwl}, t_{mwh} ; changed t_{mrldv} and t_{mavdv} to maximums; changed typical timing for t_{maws}. 11. Split timing of Source Port and Asynchronous Source Port into separate specs for FSY/SCK and FSYxn/SCKxn as outputs and inputs; changed minimums for t_{fsyh}, t_{srh}, t_{sxv}.
Chapter 8	12. Added section on calculating power dissipation from the device configuration 13. Added section on estimating junction temperature from system variables

Table B-1: Data Sheet Revision Summary (Continued)

INDEX

Notes:

Notes:

Notes:

Further Information

For more information on Oasis SiliconSystems products, including integrated circuits, software, and MOST development tools and modules, contact one of our offices below, or visit our web site:

www.oasis.com

Oasis SiliconSystems, Inc.
1120 Capital of Texas Highway South
Building 2, Suite 100
Austin, Texas 78746 USA

Tel: (+1) 512 306-8450
Fax: (+1) 512 306-8442

OSS@oasis.com

Oasis SiliconSystems, Inc.
38600 Van Dyke Avenue
Suite 220
Sterling Heights, Michigan 48312 USA

Tel: (+1) 586 795-0545
Fax: (+1) 586 795-8950

Detroit@oasis.com

Oasis SiliconSystems AG
Bannwaldallee 48
D-76185 Karlsruhe
Germany

Tel: (+49) (0) 721 6 25 37 - 0
Fax: (+49) (0) 721 6 25 37 - 119

OSS@oasis.de

Oasis SiliconSystems AG Japan
Shin-Yokohama UU Bldg. 5F
2-5-2 Shin-Yokohama, Kohoku-ku
Yokohama 222-0033, Japan

Tel: (+81) 45-470 2240
Fax: (+81) 45-470 2242

Japan@oasis.com

Technical Support

For technical support please refer to one of the following e-mail addresses:

support@oasis.de

support@oasis.com

O·A·S·I·S
SiliconSystems