BLUE TREAK
Powering Tomorrow's Vision

LH79520
User's Guide

2007 August 27

NXP founded by Philips

Content Revisions

This document contains the following changes to content, causing it to differ from previous versions.

Record of Revisions

DATE	PAGE NO.	SECTION, TABLE, OR ILLUSTRATION	SUMMARY OF CHANGES
11-15-02	1-7	Table 1-1	Pin 95 added.
	1-7	New Table 1-2	‘Operating and Test Modes’.
	2-2	Features	UART Data Rate changed to ‘Supports Data Rates up to 460.8 kbit/s’. Synchronous Serial Port line added: ‘Supports Data Rates up to 1.8452 Mbit/s’.Line added: ‘5 V Tolerant Inputs’.
	5-11	New Section	‘Selecting a Compatible Asynchronous Flash Memory Device’.
	5-15	New Section	'Using nCSx to Select an External ISA I/O Device'.
	7-5	7.5	Clarification: ‘PCLK locked to HCLK’.
	7-8	7.9	Clarification: ‘PCLK locked to HCLK’, reference to Figure 7-8.
	7-11	Table 7-1	Values clarified.
	7-18	Table 7-5	Values clarified.
	7-23	Table 7-14	Values clarified.
	7-23	Table 7-15	Values matched to Table 7-1.
	7-24	Table 7-16	Values clarified.
	7-24	Table 7-17	Values matched to Table 7-1.
	7-32	7.16.12	Clarification: ‘PCLK locked to HCLK’.
	7-33	7.16.13	
	8-3	Table 8-2	Bits [6:2] corrected
	8-4	Table 8-3	Bits [6:2] corrected; Bit [2] function corrected.
	9-10	New Section	'Important Considerations with External Level-sensitive Interrupts'.
	11-7	New Section	'LCD Power Sequencing at Turn-On and Turn-Off'.
	11-25	Table 11-14	Clarification to CPL bit description.
	11-27	Table 11-16	Values corrected.
	11-27	Table 11-17	Reference to A1:A0 corrected.
	11-28	Table 11-18	Values corrected.
	11-28	Table 11-19	Reference to A1:A0 corrected.
	13-1	13.1	Clarification: ‘PCLK locked to HCLK’.
	13-2	13.2	
	15-3	15.1.1	
	16-1	16.1	
	16-25, 16-26	16.9.8	
	17-33	Table 17-20	CTSEN and RTSEN fields deleted.
	17-34	Table 17-21	CTSEN and RTSEN descriptions deleted.

Record of Revisions (Cont'd)

DATE	PAGE NO.	SECTION, TABLE, OR ILLUSTRATION	SUMMARY OF CHANGES
11-15-02	DC Specifications, AC Specifications, Mechanical Specifications, Recommended Layout Practices		These sections moved to Data Sheet to allow for ease of maintenance and timely updates.
3-21-03	17-33	Table 17-20	Bit order corrected.
	17-34	Table 17-21	Bit order corrected.
10-20-03	3-13+	Register Bit Fields	Where Registers contain Read Only and Read/Write bits, Reserved bits now read 'Write the Reset Value'.
	7-4	7.4	Text clarified regarding source of UART clocks when the LH79520 is being driven by an external clock signal.
	7-5	7.5	
	7-6	7.5	New text to denote that Figure 7-4 illustrates signals not available outside the chip.
		Figure 7-4	Path of CLKIN signal corrected.
	7-13	Figure 7-11	
		7.12	Text clarified regarding source of UART clocks.
	7-20	Tables 7-7,8	Text and Tables added to show Chip ID.
	7-30	Table 7-28	UART Clock Source bits clarified.
	7-31	Figure 7-16	Path of CLKIN signal corrected.
	9-5	9.1.3	New section added to explain external interrupt timing.
	11-1	Chapter 11	Text rewritten to more clearly depict function of Advanced LCD Interface and CLCD Registers.
		Figure 11-1	Nomenclature of LCD interface clarified.
	11-2	Figure 11-2	New Figure: Typical Advanced LCD Panel Block Diagram.
	11-4	Figure 11-3	Signal paths clarified.
	11-6	Tables 11-1, -2	New Tables to explain frame buffer bits.
	11-9	Table 11-7	Video signals removed to more clearly highlight signal-per-panel differences.
	11-11	11.1.7	'For more information, see...' added.
		Figure 11-4	Signal paths clarified.
	11-13	11.1.9	Text added as a reminder to check the relevant LCD Data Sheet for the appropriate turn-on and turn-off times.
	11-14	11.1.10, 11.1.10.1	Interrupt conditions clarified.
	11-16	Figure 11-6	Signal paths clarified.
	11-17	11.2.1.2	Programming order corrected.
	11-38	Table 11-42	LPDEL reset bits corrected.
		Table 11-43	PSCLS and LPDEL fields clarified.
	11-39	Table 11-45	PS2CLS2 field clarified.
	11-41	Figure 11-7	Removed typo.
	11-42	Figure 11-8	Added influencing signal names and note.
	11-44	Figure 11-10	Corrected description for pin 135; removed incorrect influencing signal name from pin 137
	11-45	Figure 11-11	Corrected and added influencing signal names; corrected later states of LCDCLS and LCDPS signals.
	11-46	Figure 11-12	Corrected state of LCDSPS; corrected name of LCDCLS; corrected valid time of LCD Data; removed invalid signal.
	12-3	12.4, Table 12-7	Clarified description of Timer0.

Record of Revisions (Cont'd)

DATE	PAGE NO.	SECTION, TABLE, OR ILLUSTRATION	SUMMARY OF CHANGES
6-14-04	—	—	'Preliminary' Status removed: now Version 1.0
	—	—	Spelling corrected for signal: LCDLP.
	—	—	Pin 139 function updated to remove incorrect references.
	6-15	6-21	Resolved circular reference in bits 18/17.
	7-6	7.5	Real Time Clock path noted.
	7-12	7.11	Real Time Clock path and effects clarified.
	8-3	Table 8-3	Bit 7 of the MemMux register renamed and function clarified.
	8-6	Table 8-6	Pin 139 description updated.
	11-11	Table 11-8	Header corrected to '...16-bit Direct'
	11-12	11.1.17	Added Table describing LCD control signal muxing.
	14-1	14.1	32.768 kHz crystal connection options clarified.
5-20-05	xxxi	Preface	Text inserted regarding writes to Reserved addresses and bits.
	3-5	3.5	Text added to clarify allowable access widths for AHB.
	3-7	3.6	Text added to clarify allowable access widths for APB.
	3-7, 5-19, 6-12, 7-18, 8-2, 9-11, 10-8, 11-20, 11-36, 12-4, 13-3, 14-4, 15-10, 16-16, 17-20,18-3, 19-3, 20-1	Programmer's Model (all)	Text added to clarify allowable register access widths; whether word only or word, half-word, or byte accesses.
	3-2	3.1.2	Section added: AHB Master Endian-ness.
	3-11	3.10	JTAG functions during boundary scan clarified.
	9-27	Tables 9-33, 34	Bits 6 and 7 functions corrected.
	11-37	Tables 11-39, 40	Bits 3 (HRVE) and 2 (VRVE) added and explained.
	11-22	Table 11-15	Fields VBP, VFP, and VSW updated for STN mode.
	13-7	Tables 13-6, 7	Function of bit 6 changed to Reserved.
03-01-06	5-16	Section 5.1.4.9	Section added: Using BLE0-3 Instead of nWE.
	6-11	Section 6.1.4	Section added: Merging Write and Read Buffers.
	7-33, 11-3, 14	Table 7-29, Sections 11.1.3, 11.1.10	Text concerning changing LCD clock sources while CLCDC is enabled and operating added.
	10-25	Table 10-29	Corrected text regarding EOTx bitfields.
	7-6, 11-22, 12-4, 14-4, 15-10, 16-16, 17-20	7.5, 11.3, 12.6, 14.2, 15.2, 16.7, 17.3	Text added regrarding enabling the system clock before programming any registers.
	11-31	11.3.8	Text concerning programming output mode switches added.
	11-44	11.5	Timing diagrams moved to Data Sheet for ease of maintenance.

Record of Revisions (Cont'd)

DATE	PAGE NO.	SECTION, TABLE, OR ILLUSTRATION	SUMMARY OF CHANGES
10-31-06	7-19	Table 7-6	Table added to clarify register Reset values for different revisions of silicon.
	7-21	Table 7-9	Entries added to clarify register values for different revisions of silicon.
	8-9	Table 8-14	Description text for bits 1:0 conformed to correct signal names.
	9-3	Note	Removed "Standby" from cautionary note
	9-6	Note	Removed cautionary note
	9-8	Table 9-4	Removed "Standby" from cautionary note on bits [4:2]
	11-3, -32	11.1.3, Table 11-30	Watermark text clarified.
	11-32	Table 11-30	Bit 6 (MONO8L) added; bit is NOT reserved.
8-27-07	All	—	All references to Sharp replaced with NXP. Revision number rolled to Version 1.3.

Table of Contents

Chapter 3 – ARM Core and Buses

Chapter 4 – Internal Static RAM (SRAM)

Chapter 5 – External Static Memory Controller (SMC)

Chapter 6 – SDRAM Memory Controller (SDRC)

Chapter 7 – Reset, Clock Generation, and Power Control (RCPC)

Chapter 13 – Watchdog Timer (WDT)

Chapter 14 – Real-Time Clock (RTC)

Chapter 15 – Pulse Width Modulators (PWM)

List of Figures

Chapter 16 – Synchronous Serial Port (SSP)

Chapter 17 – Universal Asynchronous Receiver/Transmitters (UARTs)

Chapter 18 – IrDA Communications (SIR)

Chapter 19 – General Purpose I/O (GPIO)

Chapter 21 – Glossary

List of Tables

Chapter 7 – Reset, Clock Generation, and Power Control (RCPC)

Chapter 8 – I/O Control and Multiplexing (IOCON)

Chapter 9 – Exceptions and Interrupts

Chapter 10 – DMA Controller (DMAC)

Chapter 11 – Color LCD Controller

Chapter 12 – Timers

Chapter 13 – Watchdog Timer (WDT)

Chapter 14 – Real-Time Clock (RTC)

Chapter 15 – Pulse Width Modulators (PWM)

Chapter 16 – Synchronous Serial Port (SSP)

Chapter 17 – Universal Asynchronous Receiver/Transmitters (UARTs)

Chapter 18 – IrDA Communications (SIR)

Chapter 19 – General Purpose I/O (GPIO)

Chapter 20 – Status and Configuration Registers

Preface

The NXP LH79520 is a complete microcontroller. This User's Guide is intended to be the principal technical reference for this device. This document assumes that the reader is familiar with programming the ARM720T core. For more information on programming the ARM720T core, see the extensive library of methods and downloads available from ARM Ltd. at http://www.nxp.com/redirect/arm.com/.

Designers wishing an abridged version of this Guide may consult the LH79520 Data Sheet which also presents the Electrical Characteristics and Mechanical Specifications. An even more abbreviated version of this Guide is available as a single-page Product Brief. For details, contact your local NXP representative or see the NXP Semiconductors web site at http://www.nxp.com.

Application Notes and further information on connecting, programming and implementing the LH79520, along with suggestions for companion parts can be found on NXP's website. Point your browser to http://www.nxp.com.

What's in This User's Guide

Chapter 1 – Introduction

This chapter presents an illustration of the LH79520 MCU package, showing all signals for each package pin. This chapter includes a table with each signal listed by function, and a brief description of each signal. The complete LH79520 pinout is presented in tabular format.

Chapter 2 – System Overview

This chapter lists the features of the LH79520 MCU and presents a simplified block diagram of the device, with the major architectural features identified.

Chapter 3 – ARM 720T Core and Memory Systems

This chapter contains the theory of operation of the LH79520 MCU, including a brief overview of the ARM720T processor and MMU. The theory of operation covers bus architecture, bus arbitration, and the base addresses for each of the Advanced High-performance Bus (AHB) and Advanced Peripheral Bus (APB) devices, and the APB Bridge. Coverage of the memory system includes memory mapping, memory remapping, and the External Bus Interface (EBI). This chapter provides programmer's models, programmable parameters, default memory widths, address mapping, and includes a register summary and register descriptions for the core and memory map.

Chapter 4 – Internal SRAM

This chapter provides details of the 32KB of internal SRAM memory in the LH79520 MCU.

Chapter 5 – Static Memory Controller

This chapter explains the theory of operation of the LH79520 Static Memory Controller (SMC), including programmable parameters, default memory widths, and address mapping. This chapter includes a register summary and register descriptions for the SMC.

Chapter 6 – Synchronous DRAM Controller

This chapter explains the theory of operation of the LH79520 Synchronous Dynamic RAM Memory Controller (SDRC), including programmable parameters, device selection, memory widths, and address mapping. This chapter includes a register summary and register descriptions for the SDRC.

Chapter 7 – Reset, Clock Generation and Power Control

This chapter provides a short overview of the LH79520 Reset, Clock Generation and Power Control (RCPC) system, including a block diagram, a list of clock signals, power control modes, programmer's model, signal descriptions, power sequences, register summaries, register descriptions, and descriptions of interface signals.

Chapter 8 – I/O Controls and Multiplexing

This chapter includes a brief overview of the LH79520 I/O Controls and pin Multiplexing (IOCON). The chapter provides a block diagram, programmer's model, register summary, and register descriptions.

Chapter 9 – Exceptions and Interrupts

The Exceptions and Interrupts chapter introduces and explains the distinction between exceptions and interrupts, then examines the LH79520 Vectored Interrupt Controller (VIC). The chapter includes a short overview, a block diagram, programmer's model, interrupt channel list, register summaries, and register descriptions.

Chapter 10 – DMA Operations

This chapter examines the DMA operations available in the LH79520 MCU, latencies from one process to another, and the interrupts involved.

Chapter 11 – Color LCD Controller

This chapter describes the Color LCD Controller (CLCDC) and the Advanced LCD Interface (ALI) functional blocks within the LH79520. The Color LCD Controller can drive STN and TFT displays. The ALI block adds the additional control required for AD-TFT and HR-TFT displays. This chapter includes a brief overview, lists the types of panels supported, and at what bit-depths. The chapter also lists and explains the programmable parameters, presents a programmer's model, and includes a register summary. Register descriptions (with reset values), and horizontal timing restrictions are provided.

Chapter 12 – Timers

This chapter explains the four LH79520 Timers. The chapter includes a short overview and block diagram of each timer, signal descriptions, operation sequences, timer cascading, register summaries, register descriptions, and interface signals.

Chapter 13 – Watchdog Timer

This chapter discusses the LH79520 Watchdog Timer, which is a different functional block than the four timers explained in another chapter. The chapter includes a short overview, block diagram, a list of clock signals, programmer's model, signal descriptions, operating sequences, register summaries and register descriptions.

Chapter 14 – Real Time Clock

This chapter examines the LH79520 Real Time Clock. The chapter includes a short overview, a block diagram, a list of clock signals, programmer's model, signal descriptions, operating sequences, register summaries, register descriptions and interface signals.

Chapter 15 – Pulse Width Modulators

This chapter discusses the LH79520 Pulse Width Modulator functional blocks in detail. The chapter includes a brief overview, a block diagram, sample signals, programmer's model, register summaries, register descriptions, pulse frequency and duty cycle, descriptions of the normal and synchronous modes of operation and a signal interface description.

Chapter 16 – Synchronous Serial Ports

This chapter includes a brief overview of the LH79520 Synchronous Serial Ports, a block diagram, programmer's model, register summary, register descriptions, interrupts, and register locations.

Chapter 17 – UARTs

This chapter introduces the LH79520 UART blocks, and includes a brief overview, block diagram, programmer's model, programmable parameters, lists the UARTs' variations from the 16C550 UART, and includes a register summary and register descriptions.

Chapter 18 – Infrared Communications

This chapter describes the LH79520 IrDA Serial InfraRed (SIR) block, including a brief overview, block diagram, list of programmable parameters, register descriptions, and features specific to the LH79520's SIR communication system.

Chapter 19 – General Purpose Input/Output

This chapter presents the LH79520 General Purpose Input/Output (GPIO) systems, beginning with a brief overview, and including a block diagram, programmer's model, register summary, and register descriptions.

Chapter 20 – Status and Configuration Register Summary

This chapter is a complete compilation of all registers in the LH79520, listed by address, in ascending numerical sequence, with brief definitions of each register's function.

Chapter 21 – Glossary

This chapter contains an alphabetical listing of common terminology appearing in this User's Guide.

Electrical and Mechanical Specifications

Please consult the LH79520 Data Sheet for full specifications.

Terms and Conventions

For information on specific terms and acronyms see the Glossary in this User's Guide.

Multiplexed Pins

The LH79520 is manufactured in an LQFP package with 176 pins. Some pins have only one function, but others are multiplexed and may carry as many as three functions. Designers must be aware that multiplexed pins cannot simultaneously support more than one function; a choice is required.

Pin Names

Package pins are named to indicate the signal(s) or functionality available at the pin. If the signal or function is active LOW, the name is prefixed with a lower-case 'n', such as nCS2. Multiplexed pins are named to indicate all available functions, such as Pin 41: nCS6/PH7, which can function as either SMC Chip Select 6, or bit 7 of GPIO Port H.

These naming conventions help designers recognize and avoid collisions between multiplexed functions but can complicate explanatory text, so this Guide uses the name appropriate to the context. A discussion of chip selects, for example, would refer to signal nCS6, but information about PortH, bit 7 would use PH7. Readers must be aware that these are separate signals, with distinctly different functionality, which happen to be available on the same pin. Such signals are not simultaneously available at the pin.

Peripheral Devices

The LH79520 is a microcontroller built using the ARM720T RISC core as a base. Objects within the chip but external to the core processor and its support devices are referred to throughout this Guide as 'Peripheral Devices'.

The LH79520 includes two buses: an Advanced High-Performance Bus (AHB) and an Advanced Peripheral Bus (APB). The devices shown on the APB in the block diagrams are an example of 'peripheral devices' in this document. Devices that are external to the chip are referred to as 'external devices'.

Register Addresses

The LH79520 is a memory-mapped device with programmable, internal registers that control its operation. Each internal register is located at a unique address in the memory map and the registers are generally grouped in the map by subsystem.

In this Guide, the addresses for all registers are expressed as a base address and an offset from that base. The base address indicates where in the map a group of registers begins and the offset locates a particular register, relative to its base address. Thus, any register's absolute address is the sum of its base address and its offset. Programmers will find this base + offset representation convenient for creating software structures to access the registers. For convenience, this base + offset convention is echoed in the ADDR line of the Register Tables, as in the example shown in Table 1.

Reserved Addresses

Some register areas contain reserved addresses. These areas must only be written as zeroes, as writing ones may cause undesirable or unstable operation.

Reserved Bits

Some register addresses may contain reserved bits. Each Register Table (such as Table 1) will have specific instructions as to what may be written to those bits.

Register Tables

All Registers are presented in tabular format. A primary table presents each register's name, address, permissions, bit field names and the register's contents at reset. Subsequent tables detail the specific function(s) of all bit fields in the register and explain any important variations that may exist. See Table 1 and Table 2 for examples of this practice.

Table 1. 32-bit Register Example

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								ExA	ExB	///	ExC	///		Ex1	Ex2
RESET	0	0	0	0	0	0	0	0	0	0		0			0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	W	RW	RW	RW	R
ADDR	0x8000.0E00 + 0x244															

Table 2. 32-bit Register Fields

BIT	FIELD NAME	DESCRIPTION
31:8	///	**Reserved** Write the reset value.
7	ExA	**EXAMPLE A** The MCU writes a 1 to this bit to clear SETUP_END (D4).
6	ExB	**EXAMPLE B** The MCU writes a 1 to this bit to clear OUT_PKT_RDY (D1).
5	///	**Reserved** Write the reset value.
4	ExC	**EXAMPLE C** This bit controls the FIFO dump. Writing to it flushes the FIFO.
3:2	///	**Reserved** Reads are undefined. Write zeroes to these bits. Writing ones can result in unpredictable operation.
1	Ex1	**EXAMPLE 1** The MCU sets this bit after writing a packet of data into ENDPOINT 0 FIFO. The USB clears this bit once the packet has been successfully sent to the host.
0	Ex2	**EXAMPLE 2** The USB sets this bit once a valid token is written to the FIFO. This field must not be written, or erratic operation will result.

Table Conventions

In Table 1, the TYPE line shows the bit type; whether readable, writable, or any restriction upon reading and writing. Those conventions are shown here.

R

Reading returns (an automatically specified value). Values written cannot be read. It is safe as a shortcut to write the reset value.

W

Reads are undefined.

RW

Requires the most explanation. Write to program the device. Reading returns (either the value most recently written to this field by software or an automatically specified value).

Numeric Values

Binary values are prefixed with 0b, as in 0b00001000. Binary values may also be shown within quotation marks, as in '0'.

Hexadecimal values are expressed with UPPER CASE letters and prefixed with 0x, as in 0x0FBC.

All numeric values not specifically identified with the above prefixes as either binary or hexadecimal are decimal values.

Block Diagrams

The functional descriptions in this Guide include block diagrams with symbols representing logical or mathematical operations or selections, usually the result of writing a value to a Register. Figure 1 shows one such multiplexer with three inputs and one output (the result).

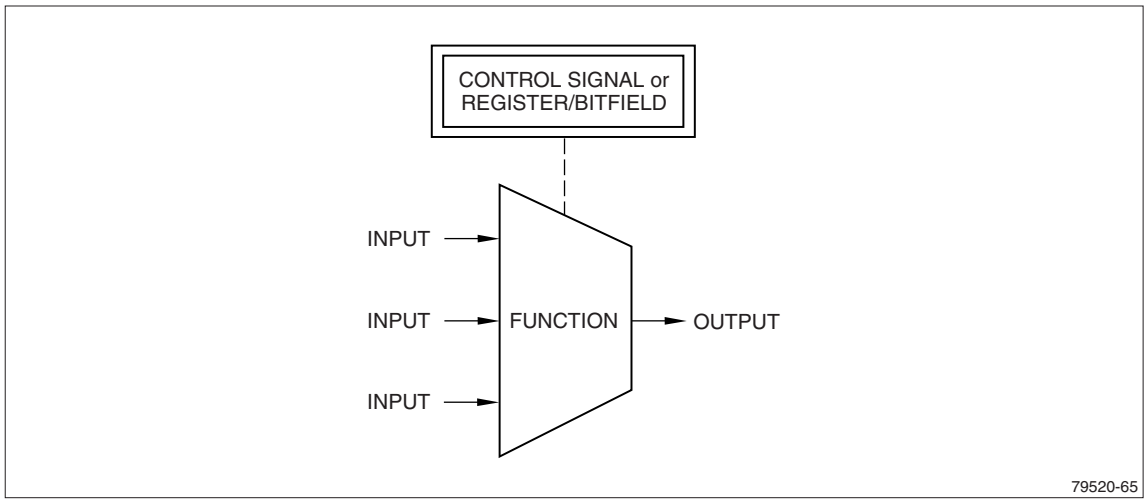


Figure 1. Multiplexer

Block diagrams can include symbols representing Registers and the bit fields within them. Figure 2 shows that the BITFIELDNAME bit field in the REGISTERNAME register enables or disables the signal named OUTPUT.

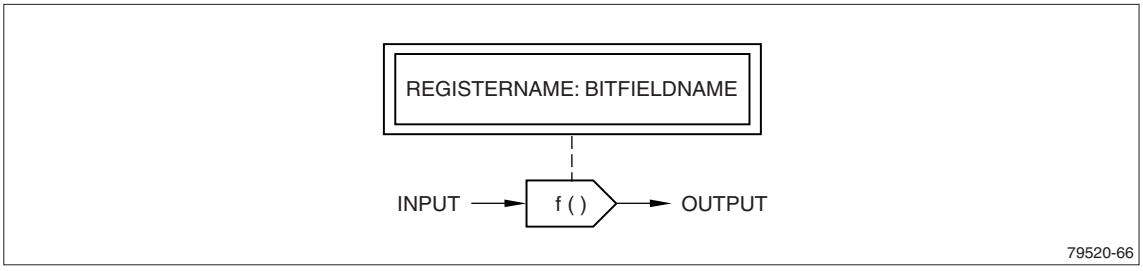


Figure 2. Register with Bit-Field Named

Figure 3 is similar to Figure 2 except that Figure 3 references multiple (different) BIT-FIELDS in the REGISTERNAME register.

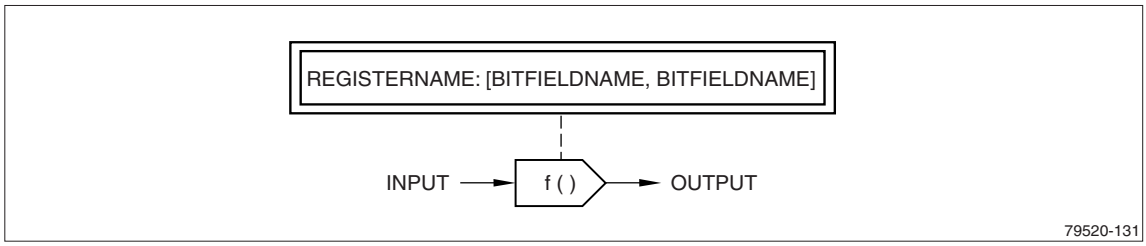


Figure 3. Register with Multiple Bit-Fields Named

Not all bit fields are named. If a bit field has no name, the Register is shown with numbers indicating the appropriate bit-positions, with the least-significant bit on the right, as in Figure 4. This bit-ordering matches that of the Register tables, shown in Table 1.

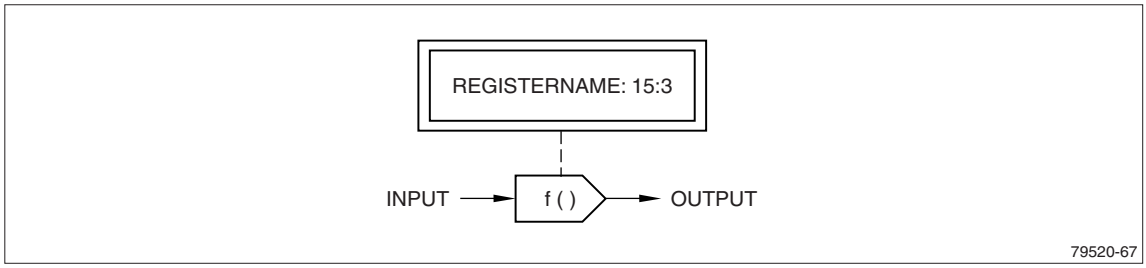


Figure 4. Register with Bit-Field Numbered

Chapter 1
Introduction

This User's Guide is a detailed technical reference for the NXP LH79520 microcontroller (MCU). This Guide is intended for use by Engineers, Programmers, and Designers. For in-depth electrical information, please see the LH79520 Data Sheet.

1.1 Description

The LH79520 is a fully-integrated 32-bit MCU based on an ARM720T core. The LH79520 includes a Color LCD Controller and high performance functional blocks that provide a glueless interface to external memory. A fully static design provides low voltage operation and a full compliment of power management features.

The LH79520 combines a 32-bit ARM RISC core with a Direct Memory Access Controller, Vectored Interrupt Controller, Color LCD Controller, 32KB of internal SRAM, and several supporting peripherals. The 32-bit ARM720T RISC core provides a powerful instruction set and includes Cache RAM, a Write buffer, Memory-Management Unit (MMU) and Translation Lookaside buffer (TLB).

Supporting functional blocks within the LH79520 include Serial and Parallel Interfaces, Counters/Timers, Real Time Clock, Watchdog Timer, Pulse Width Modulators, Infrared support and an on-chip Phase-Lock Loop. JTAG support is provided to simplify debugging.

1.2 Pin Diagram

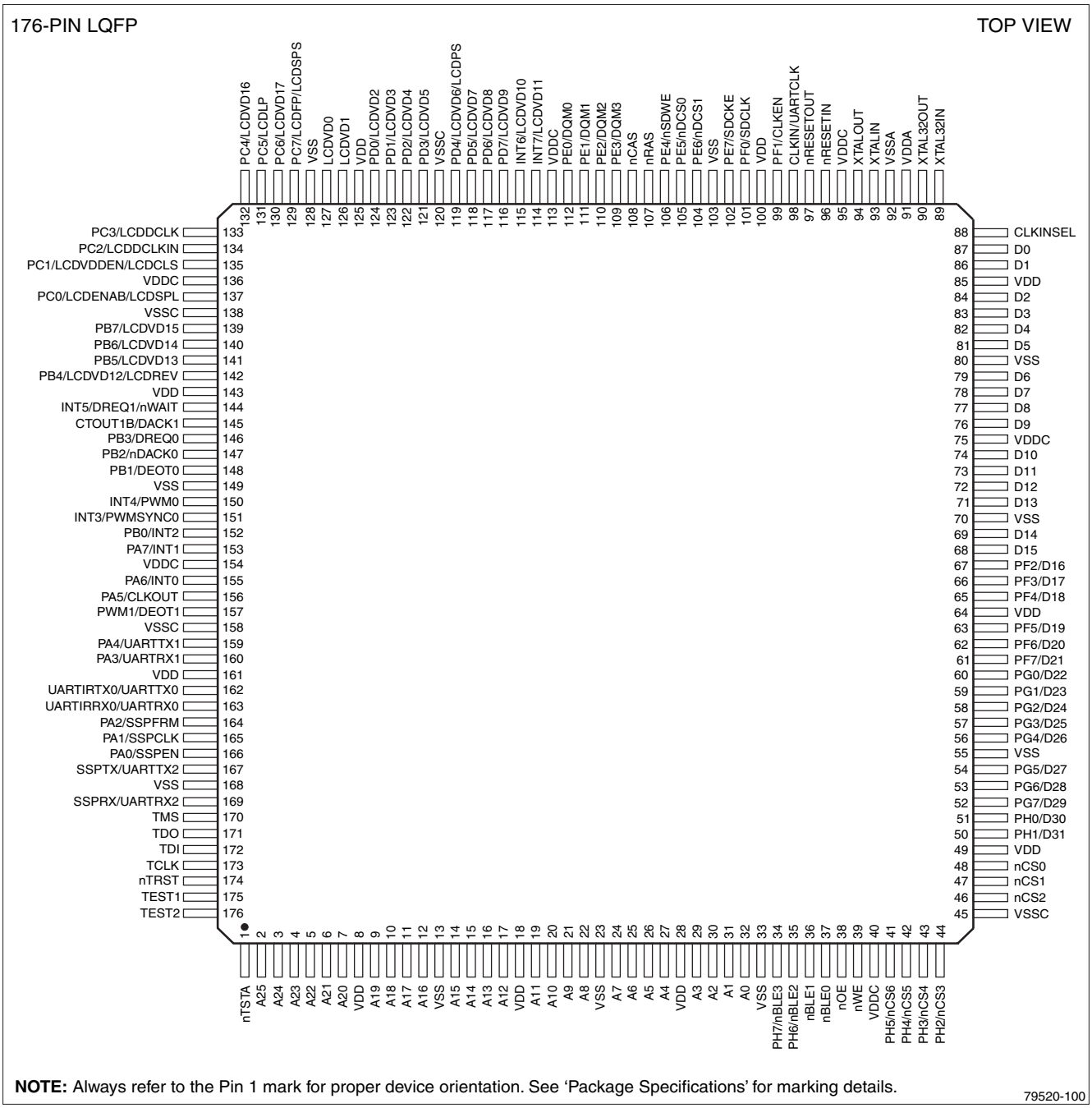


Figure 1-1. LH79520 Pin Diagram

1.3 Functional Pin List

Table 1-1. LH79520 Functional Pin List

PIN	SIGNAL NAME	TYPE	DESCRIPTION	NOTES
MEMORY INTERFACE (MI)				
2-7, 9-12 14-17, 19-22, 24-27, 29-32	A[25:0]	Output	Address Signals	
50-54, 56-63, 65-66, 67-69, 71-74, 76-79, 81-84, 86-87	D[31:0]	Input/Output	Data Input/Output Signals	1
101	SDCLK	Output	SDRAM Clock	1
109	DQM3	Output	Data Mask Output to SDRAMs	1
110	DQM2	Output	Data Mask Output to SDRAMs	1
111	DQM1	Output	Data Mask Output to SDRAMs	1
112	DQM0	Output	Data Mask Output to SDRAMs	1
102	SDCKE	Output	SDRAM Clock Enable	1
104	nDCS1	Output	SDRAM Chip Select	1
105	nDCS0	Output	SDRAM Chip Select	1
107	nRAS	Output	Row Address Strobe	
108	nCAS	Output	Column Address Strobe	
106	nSDWE	Output	SDRAM Write Enable	1
41	nCS6	Output	Static Memory Controller Chip Select	1
42	nCS5	Output	Static Memory Controller Chip Select	1
43	nCS4	Output	Static Memory Controller Chip Select	1
44	nCS3	Output	Static Memory Controller Chip Select	1
46	nCS2	Output	Static Memory Controller Chip Select	
47	nCS1	Output	Static Memory Controller Chip Select	
48	nCS0	Output	Static Memory Controller Chip Select	
38	nOE	Output	Static Memory Controller Output Enable	
34	nBLE3	Output	Static Memory Controller Byte Lane Enable/Byte Write Enable	1
35	nBLE2	Output	Static Memory Controller Byte Lane Enable/Byte Write Enable	1
36	nBLE1	Output	Static Memory Controller Byte Lane Enable/Byte Write Enable	
37	nBLE0	Output	Static Memory Controller Byte Lane Enable/Byte Write Enable	
39	nWE	Output	Static Memory Controller Write Enable	
144	nWAIT	Input	Static Memory Controller External Wait Control	1, 3

Table 1-1. LH79520 Functional Pin List (Cont'd)

PIN	SIGNAL NAME	TYPE	DESCRIPTION	NOTES
DMA CONTROLLER (DMAC)				
148	DEOT0	Output	DMA 0 End of Transfer	1
147	nDACK0	Output	DMA 0 Acknowledge	1
146	DREQ0	Input	DMA 0 Request	1
157	DEOT1	Output	DMA 1 End of Transfer	1
145	DACK1	Output	DMA 1 Acknowledge	1
144	DREQ1	Input	DMA 1 Request	1, 3
COLOR LCD CONTROLLER (CLCDC)				
130, 132, 139, 140, 141, 142, 114, 115, 116, 117, 118, 119, 121, 122, 123, 124, 126, 127	LCDVD[17:0]	Output	LCD Panel Data bus	1
137	LCDENAB	Output	LCD Data Enable	1
129	LCDFP	Output	Frame Pulse (STN), Vertical Synchronization Pulse (TFT)	1
131	LCDLP	Output	Line Synchronization Pulse (STN), Horizontal Synchronization Pulse (TFT)	1
133	LCDDCLK	Output	LCD Panel Data Clock	1
134	LCDCLKIN	Input	LCD External Clock Input	1
135	LCDVDDEN	Output	LCD Digital Supply Enable	1
135	LCDCLS	Output	LCD Clock Signal for Gate Driver (AD-TFT, HR-TFT only)	1
129	LCDSPS	Output	LCD Reset Signal for Row Display (AD-TFT, HR-TFT only)	1
142	LCDREV	Output	LCD Reverse Signal (AD-TFT, HR-TFT only)	1
137	LCDSPL	Output	LCD Line Start Pulse (Left) (AD-TFT, HR-TFT only)	1
119	LCDPS	Output	LCD Power Save (AD-TFT, HR-TFT only)	1
SYNCHRONOUS SERIAL PORT (SSP)				
164	SSPFRM	Output	SSP Serial Frame Output	1
165	SSPCLK	Output	SSP Clock	1
166	SSPEN	Output	SSP Data Enable	1
167	SSPTX	Output	SSP Data Out	1
169	SSPRX	Input	SSP Data In	1
PULSE WIDTH MODULATOR (PWM)				
150	PWM0	Output	PWM0 Output	1
151	PWMSYNC0	Input	PWM0 Synchronizing Input	1
157	PWM1	Output	PWM1 Output	1
UART0 (U0)				
163	UARTRX0	Input	UART0 Received Serial Data Input	1
162	UARTTX0	Output	UART0 Transmitted Serial Data Output	1
163	UARTIRRX0	Input	UART0 InfraRed Receive	1
162	UARTIRTX0	Output	UART0 InfraRed Transmit	1

Table 1-1. LH79520 Functional Pin List (Cont'd)

PIN	SIGNAL NAME	TYPE	DESCRIPTION	NOTES
UART1 (U1)				
160	UARTRX1	Input	UART1 Received Serial Data Input	1
159	UARTTX1	Output	UART1 Transmitted Serial Data Output	1
UART2 (U2)				
169	UARTRX2	Input	UART2 Received Serial Data Input	1
167	UARTTX2	Output	UART2 Transmitted Serial Data Output	1
GENERAL PURPOSE INPUT/OUTPUT (GPIO)				
153 155 156 159 160 164 165 166	PA7 PA6 PA5 PA4 PA3 PA2 PA1 PA0	Input/Output	General Purpose I/O Signals - Port A	1
139 140 141 142 146 147 148 152	PB7 PB6 PB5 PB4 PB3 PB2 PB1 PB0	Input/Output	General Purpose I/O Signals - Port B	1
129 130 131 132 133 134 135 137	PC7 PC6 PC5 PC4 PC3 PC2 PC1 PC0	Input/Output	General Purpose I/O Signals - Port C	1
116 117 118 119 121 122 123 124	PD7 PD6 PD5 PD4 PD3 PD2 PD1 PD0	Input/Output	General Purpose I/O Signals - Port D	1
102 104 105 106 109 110 111 112	PE7 PE6 PE5 PE4 PE3 PE2 PE1 PE0	Input/Output	General Purpose I/O Signals - Port E	1
61 62 63 65 66 67 99 101	PF7 PF6 PF5 PF4 PF3 PF2 PF1 PF0	Input/Output	General Purpose I/O Signals - Port F. GPIO PF1 is only available when CLKINSEL is '0' (i.e. the external clock source is not being used).	1

Table 1-1. LH79520 Functional Pin List (Cont'd)

PIN	SIGNAL NAME	TYPE	DESCRIPTION	NOTES
52 53 54 56 57 58 59 60	PG7 PG6 PG5 PG4 PG3 PG2 PG1 PG0	Input/Output	General Purpose I/O Signals - Port G	1
34 35 41 42 43 44 50 51	PH7 PH6 PH5 PH4 PH3 PH2 PH1 PH0	Input/Output	General Purpose I/O Signals - Port H	1
TIMER				
145	CTOUT1B	Output	Timer Output	1
RESET, CLOCK, AND POWER CONTROLLER (RCPC)				
96	nRESETIN	Input	Reset Input	
97	nRESETOUT	Output	Reset Output	
114	INT7	Input	External Interrupt Input	1
115	INT6	Input	External Interrupt Input	1
144	INT5	Input	External Interrupt Input	1, 3
150	INT4	Input	External Interrupt Input	1
151	INT3	Input	External Interrupt Input	1
152	INT2	Input	External Interrupt Input	1
153	INT1	Input	External Interrupt Input	1
155	INT0	Input	External Interrupt Input	1
93	XTALIN	Input	Crystal Input	
94	XTALOUT	Output	Crystal Output	
89	XTAL32IN	Input	32.768 kHz Crystal Oscillator Input	
90	XTAL32OUT	Output	32.768 kHz Crystal Oscillator Output	
88	CLKINSEL	Input	External Clock Select	
98	CLKIN	Input	External Clock Input (if CLKINSEL = HIGH at reset)	1
99	CLKEN	Output	External Clock Enable (if CLKINSEL = LOW at reset, then this pin functions as PF1)	1
156	CLKOUT	Output	Clock Out (selectable from the internal bus clock or 32.768)	1
98	UARTCLK	Input	External UART Clock Input (with CLKSEL = LOW)	1

Table 1-1. LH79520 Functional Pin List (Cont'd)

PIN	SIGNAL NAME	TYPE	DESCRIPTION	NOTES
TEST INTERFACE				
174	nTRST	Input	JTAG Test Reset Input	
170	TMS	Input	JTAG Test Mode Select Input	
173	TCLK	Input	JTAG Test Clock Input	
172	TDI	Input	JTAG Test Serial Data Input	
171	TDO	Output	JTAG Test Data Serial Output	
175	TEST1	Input	Tie LOW for Normal Operation (has internal pull-down).	4
176	TEST2	Input	JTAG Debug Enable: Tie LOW for Normal Operation; pull HIGH to enable JTAG Debugging (has internal pull-down).	4
1	nTSTA	Input	Tie HIGH for Normal Operation (has internal pull-up).	4
POWER AND GROUND (GND)				
40, 75, 95, 113, 136, 154	VDDC	Power	Core Power Supply	
45, 120, 138, 158	VSSC	Ground	Core GND	
8, 18, 28, 49, 64, 85, 100, 125, 143, 161	VDD	Power	Input/Output Power Supply	
13, 23, 33, 55, 70, 80, 103, 128, 149, 168	VSS	Ground	Input/Output GND	
91	VDDA	Power	Analog Power Supply for PLLs and XTAL Oscillators	
92	VSSA	Ground	Analog GND for PLLs and XTAL Oscillators	

NOTES:
1. These pin numbers have multiplexed functions.
2. Signals preceded by 'n' are Active LOW.
3. Immediately after reset, pin 144 can be programmed to function as INT5, DREQ1 or both. Software should avoid enabling both of these functions simultaneously. Pin 144 can also be programmed to function as nWAIT, rendering the INT5/DREQ1 choice unavailable.
4. See Table 1-2.

Table 1-2. Test Modes

OPERATING MODE	PIN STATES		
	nTESTA	TEST1	TEST2
Normal	HIGH	LOW	LOW
Debug with JTAG access	HIGH	LOW	HIGH
Reserved for Production Test	All other states		

NOTES:
1. Normal = No test mode selected.
2. Debug Mode = JTAG Interface enabled.
3. These modes are exclusive; e.g.: JTAG is not available in Normal mode.

1.4 Numerical Pin List

Table 1-3. LH79520 Numerical Pin List

PIN NO.	FUNCTION AT RESET	FUNCTION 2	FUNCTION 3	TYPE	OUTPUT DRIVE	NOTES
1	nTSTA			Input	None	1
2	A25			Output	8 mA	
3	A24			Output	8 mA	
4	A23			Output	8 mA	
5	A22			Output	8 mA	
6	A21			Output	8 mA	
7	A20			Output	8 mA	
8	VDD			Power	None	
9	A19			Output	8 mA	
10	A18			Output	8 mA	
11	A17			Output	8 mA	
12	A16			Output	8 mA	
13	VSS			Ground	None	
14	A15			Output	8 mA	
15	A14			Output	8 mA	
16	A13			Output	8 mA	
17	A12			Output	8 mA	
18	VDD			Power	None	
19	A11			Output	8 mA	
20	A10			Output	8 mA	
21	A9			Output	8 mA	
22	A8			Output	8 mA	
23	VSS			Ground	None	
24	A7			Output	8 mA	
25	A6			Output	8 mA	
26	A5			Output	8 mA	
27	A4			Output	8 mA	
28	VDD			Power	None	
29	A3			Output	8 mA	
30	A2			Output	8 mA	
31	A1			Output	8 mA	
32	A0			Output	8 mA	
33	VSS			Ground	None	
34	PH7	nBLE3		I/O	8 mA	
35	PH6	nBLE2		I/O	8 mA	
36	nBLE1			Output	8 mA	
37	nBLE0			Output	8 mA	

Table 1-3. LH79520 Numerical Pin List (Cont'd)

PIN NO.	FUNCTION AT RESET	FUNCTION 2	FUNCTION 3	TYPE	OUTPUT DRIVE	NOTES
38	nOE			Output	8 mA	
39	nWE			Output	8 mA	
40	VDDC			Power	None	
41	PH5	nCS6		I/O	8 mA	
42	PH4	nCS5		I/O	8 mA	
43	PH3	nCS4		I/O	8 mA	
44	PH2	nCS3		I/O	8 mA	
45	VSSC			Ground	None	
46	nCS2			Output	8 mA	
47	nCS1			Output	8 mA	
48	nCS0			Output	8 mA	
49	VDD			Power	None	
50	PH1	D31		I/O	8 mA	
51	PH0	D30		I/O	8 mA	
52	PG7	D29		I/O	8 mA	
53	PG6	D28		I/O	8 mA	
54	PG5	D27		I/O	8 mA	
55	VSS			Ground	None	
56	PG4	D26		I/O	8 mA	
57	PG3	D25		I/O	8 mA	
58	PG2	D24		I/O	8 mA	
59	PG1	D23		I/O	8 mA	
60	PG0	D22		I/O	8 mA	
61	PF7	D21		I/O	8 mA	
62	PF6	D20		I/O	8 mA	
63	PF5	D19		I/O	8 mA	
64	VDD			Power	None	
65	PF4	D18		I/O	8 mA	
66	PF3	D17		I/O	8 mA	
67	PF2	D16		I/O	8 mA	
68	D15			I/O	8 mA	
69	D14			I/O	8 mA	
70	VSS			Ground	None	
71	D13			I/O	8 mA	
72	D12			I/O	8 mA	
73	D11			I/O	8 mA	
74	D10			I/O	8 mA	
75	VDDC			Power	None	

Table 1-3. LH79520 Numerical Pin List (Cont'd)

PIN NO.	FUNCTION AT RESET	FUNCTION 2	FUNCTION 3	TYPE	OUTPUT DRIVE	NOTES
76	D9			I/O	8 mA	
77	D8			I/O	8 mA	
78	D7			I/O	8 mA	
79	D6			I/O	8 mA	
80	VSS			Ground	None	
81	D5			I/O	8 mA	
82	D4			I/O	8 mA	
83	D3			I/O	8 mA	
84	D2			I/O	8 mA	
85	VDD			Power	None	
86	D1			I/O	8 mA	
87	D0			I/O	8 mA	
88	CLKINSEL			Input	None	2
89	XTAL32IN			Input	None	8
90	XTAL32OUT			Output		3
91	VDDA			Power	None	
92	VSSA			Ground	None	
93	XTALIN			Input	None	8
94	XTALOUT			Output		3
95	VDDC			Power	None	
96	nRESETIN			Input	None	1, 4
97	nRESETOUT			Output	4 mA	
98	CLKIN	UARTCLK		Input	None	
99	PF1	CLKEN		I/O	2 mA	
100	VDD			Power	None	
101	PF0	SDCLK		I/O	8 mA	
102	PE7	SDCKE		I/O	8 mA	
103	VSS			Ground	None	
104	PE6	nDCS1		I/O	8 mA	
105	PE5	nDCS0		I/O	8 mA	
106	PE4	nSDWE		I/O	8 mA	
107	nRAS			Output	8 mA	
108	nCAS			Output	8 mA	
109	PE3		DQM3	I/O	8 mA	
110	PE2		DQM2	I/O	8 mA	
111	PE1		DQM1	I/O	8 mA	
112	PE0		DQM0	I/O	8 mA	
113	VDDC			Power	None	

Table 1-3. LH79520 Numerical Pin List (Cont'd)

PIN NO.	FUNCTION AT RESET	FUNCTION 2	FUNCTION 3	TYPE	OUTPUT DRIVE	NOTES
114	INT7	LCDVD11		I/O	8 mA	
115	INT6	LCDVD10		I/O	8 mA	
116	PD7	LCDVD9		I/O	8 mA	
117	PD6	LCDVD8		I/O	8 mA	
118	PD5	LCDVD7		I/O	8 mA	
119	PD4	LCDVD6	LCDPS	I/O	8 mA	
120	VSSC			Ground	None	
121	PD3	LCDVD5		I/O	8 mA	
122	PD2	LCDVD4		I/O	8 mA	
123	PD1	LCDVD3		I/O	8 mA	
124	PD0	LCDVD2		I/O	8 mA	
125	VDD			Power	None	
126	LCDVD1			Output	8 mA	
127	LCDVD0			Output	8 mA	
128	VSS			Ground	None	
129	PC7	LCDFP	LCDSPS	I/O	8 mA	
130	PC6	LCDVD17		I/O	8 mA	
131	PC5	LCDLP		I/O	8 mA	
132	PC4	LCDVD16		I/O	8 mA	
133	PC3	LCDDCLK		I/O	8 mA	
134	PC2	LCDDCLKIN		I/O	2 mA	
135	PC1	LCDVDDEN	LCDCLS	I/O	8 mA	
136	VDDC			Power	None	
137	PC0	LCDENAB	LCDSPL	I/O	8 mA	
138	VSSC				8 mA	
139	PB7	LCDVD15		I/O	8 mA	
140	PB6	LCDVD14		I/O	8 mA	
141	PB5	LCDVD13		I/O	8 mA	
142	PB4	LCDVD12	LCDREV	I/O	8 mA	
143	VDD			Power	None	
144	INT5/DREQ1	nWAIT		Input	None	4, 6
145	CTOUT1B	DACK1		Output	4 mA	
146	PB3	DREQ0		I/O	2 mA	4
147	PB2	nDACK0		I/O	4 mA	
148	PB1	DEOT0		I/O	4 mA	
149	VSS			Ground	None	
150	INT4	PWM0		I/O	4 mA	4
151	INT3	PWMSYNC0		Input	None	4

Table 1-3. LH79520 Numerical Pin List (Cont'd)

PIN NO.	FUNCTION AT RESET	FUNCTION 2	FUNCTION 3	TYPE	OUTPUT DRIVE	NOTES
152	PB0	INT2		I/O	2 mA	4
153	PA7	INT1		I/O	2 mA	4
154	VDDC			Power	None	
155	PA6	INT0		I/O	2 mA	4
156	PA5	CLKOUT		I/O	8 mA	
157	PWM1	DEOT1		Output	4 mA	
158	VSSC			Ground	None	
159	PA4	UARTTX1		I/O	4 mA	
160	PA3	UARTRX1		I/O	2 mA	4
161	VDD			Power	None	
162	UARTIRTX0	UARTTX0		Output	4 mA	
163	UARTIRRX0	UARTRX0		Input	None	4
164	PA2	SSPFRM		I/O	4 mA	
165	PA1	SSPCLK		I/O	4 mA	
166	PA0	SSPEN		I/O	4 mA	
167	SSPTX	UARTTX2		Output	4 mA	
168	VSS			Ground	None	
169	SSPRX	UARTRX2		Input	None	4
170	TMS			Input	None	1, 4
171	TDO			Output	4 mA	
172	TDI			Input	None	1, 4
173	TCLK			Input	None	
174	nTRST			Input	None	1, 4
175	TEST1			Input	None	2
176	TEST2			Input	None	2

NOTES:
1. Input with internal pull-up.
2. Input with internal pull-down.
3. Output is for crystal oscillator only, no drive capability.
4. Input with Schmitt Trigger.
5. I/O = Input/Output.
6. Software should avoid enabling the INT5 and DREQ1 functions simultaneously.
7. Output Drive Values shown are MAX. See the Electrical Specifications in the LH79520 Data Sheet.
8. Crystal Oscillator Inputs should be driven to a maximum of 1.8 V ± 10%. See the Electrical Specifications in the LH79520 Data Sheet.

Chapter 2
System Overview

This chapter of this User's Guide introduces the NXP LH79520 Universal MCU, lists its features, and describes its internal architecture and operation in general terms. Subsequent chapters of this Guide explain in more detail the major features and functions introduced here. Readers should familiarize themselves with this chapter, and the Terms and Conventions of this User's Guide, before reviewing the chapters concerning the individual functional blocks.

2.1 Features

The NXP LH79520, powered by an ARM720T core, is a complete MCU designed with a high level of integration for use in advanced portable devices. The LH79520 includes the following features:

- Highly Integrated Single Chip
- High Performance (77.414 MHz)
- ARM720T™ RISC Core
  - 32-bit ARM7TDMI™ RISC Core
  - 8KB Cache
  - MMU
- 32KB On-Chip SRAM
- Flexible, Programmable Memory Interface
  - SDRAM Interface
    - 15-bit External Address Bus
    - 32-bit Data Bus
    - Two Segments (128MB each)
  - SRAM/Flash/ROM Interface
    - 26-bit External Address Bus
    - 32-bit External Data Bus
    - Seven Segments (64MB Each)
- Multi-stream DMA Controller
  - Four 32-bit Burst-based Data Streams
- Clock and Power Management
  - 32.768 kHz Oscillator for Real Time Clock
  - 14.7456 MHz Oscillator and PLL for System Clocks
  - Active, Standby, Sleep and Stop Power Modes
  - External Clock Option

- Low Power Modes
  - Active Mode
  - Standby Mode
  - Sleep Mode
  - Stop Mode
- Watchdog Timer
- Vectored Interrupt Controller
  - Any interrupt source assignable to IRQ or FIQ
  - Any IRQ source assignable to 1 of 16 hardware-prioritized vectors
  - Unvectored IRQ sources assigned to a default vector (lowest hardware priority)
- Three UARTs
  - 16-byte FIFOs for Rx and Tx
  - IrDA SIR Support up to 115.2 kbit/s
  - Supports Data Rates up to 460.8 kbit/s
- Two 16-bit Pulse Width Modulators
- Two Dual Channel Timer Modules
- Real Time Clock
  - 32-bit Up-counter with Programmable Load
  - Programmable 32-bit Match Compare Register
- 64 Programmable General Purpose I/O Signals
  - Multiplexed with Peripheral I/O Signals
  - 5 V Tolerant Digital Inputs
- Programmable Color LCD Controller
  - Up to 800 × 600 Resolution
  - Supports STN, Color STN, AD-TFT, HR-TFT, TFT
  - Supports 15 Shades of Gray
  - TFT: Supports 256 Colors selected from a Palette of 64 k Colors or 64 k Direct Colors
  - Color STN: Supports 256 Colors Selected from a Palette of 3,375 Colors or 4 k Direct Colors
- Synchronous Serial Port
  - Compatible with Common Interface Schemes
    - Motorola SPI™
    - National Semiconductor Microwire™
    - Texas Instruments SSI™
    - Supports Data Rates up to 1.8452 Mbit/s
- JTAG Debug Interface and Boundary Scan

2.2 Block Diagram

The LH79520 is a 32-bit ARM-based MCU and is a very good fit for portable, battery-powered applications requiring LCDs.

The LH79520, shown in Figure 2-1, is organized around two 32-bit internal buses and operates with either an external 14.7456 MHz crystal or an external clock source (≤154.8288 MHz).

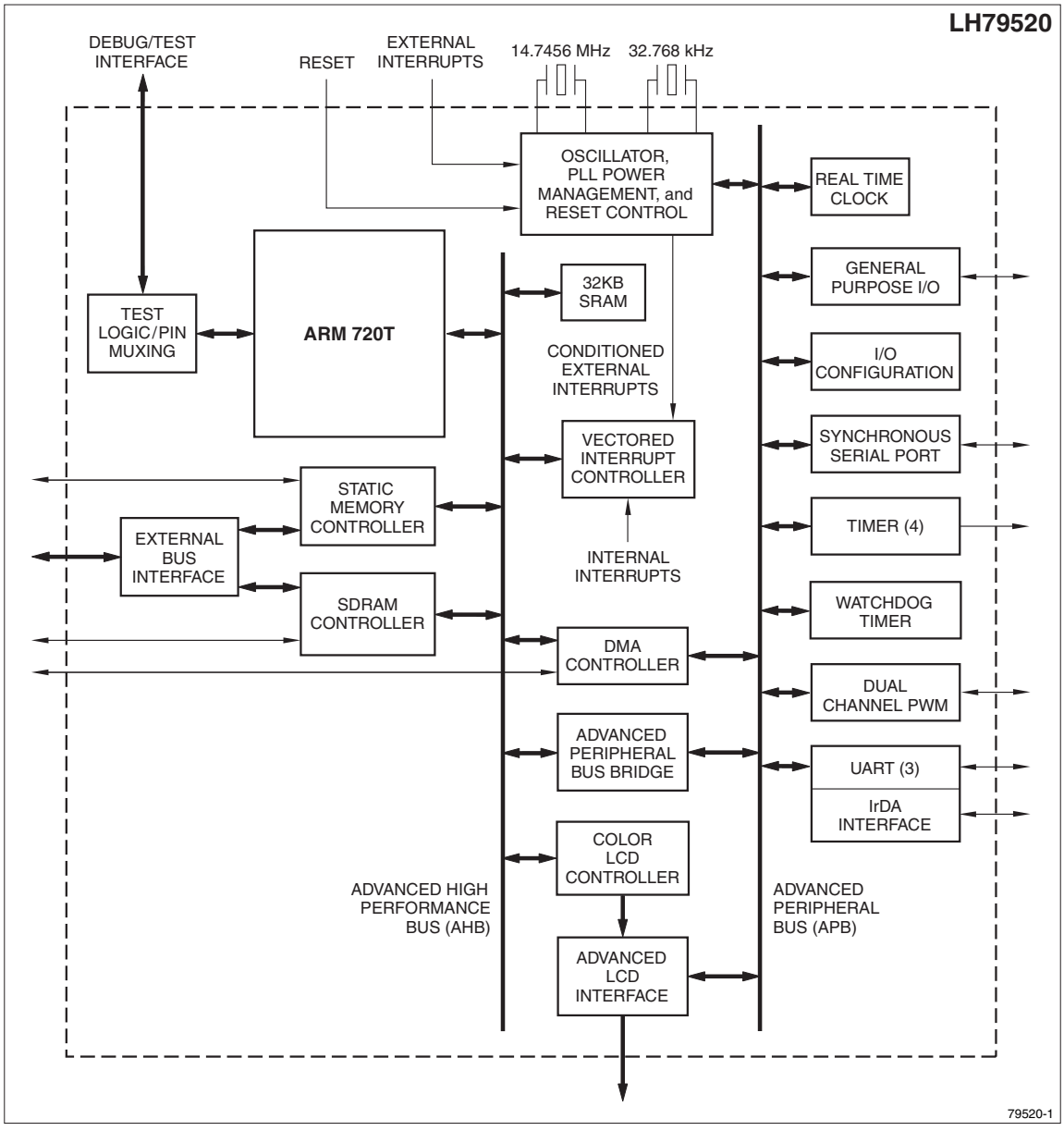


Figure 2-1. LH79520 Block Diagram

The LH79520 includes an ARM720T core, a four-channel Direct Memory Access (DMA) Controller (DMAC) and a DMA-enabled Color LCD Controller (CLCDC) on a 32-bit Advanced High-Performance Bus (AHB). This high level of integration promotes the concurrency of tasks such as LCD flat-panel refresh while the core is operating in its cache.

The LH79520 on-chip memory resources include 32KB of internal Static RAM (SRAM), a Static Memory Controller (SMC) and a Synchronous Dynamic Ram Controller (SDRC) for interfacing to external memory devices. Speed-critical memory interface control signals are pipelined to facilitate the use of the fast, local modes offered by industry-standard dynamic RAM devices. See Chapter 3 – ARM Core and Buses for more information about LH79520 core, buses, and memory resources.

Essential on-chip peripherals (see Section 2.1) are accessed via a 32-bit Advanced Peripheral Bus (APB) connected to the AHB by a 32-bit bridge. The AHB, APB and Bridge conform to the ARM Advanced Microcontroller Bus Architecture (AMBA) specification. The 4-channel on-chip DMAC can transfer data between memory and peripherals to promote efficient AHB bandwidth utilization. The DMAC is explained in Chapter 10 – DMA Controller (DMAC), and the CLCDC is covered in Chapter 11 – Color LCD Controller.

The LH79520 can respond to interrupts from 32 different sources, eight of which are external. All external interrupts can be conditioned in software for level- or edge-sensitivity. All interrupts are routed to a Vectored Interrupt Controller (VIC) that enforces a programmable hardware interrupt priority and can generate either an IRQ or an FIQ signal to the ARM720T core. This flexible interrupt structure promotes concurrency and reduces LH79520 interrupt latencies. See Chapter 9 – Exceptions and Interrupts for more information.

The LH79520 can be reset by external input, an internal signal from the Watchdog Timer (WDT), or under software control. Software can reset the entire LH79520, including the on-chip Real-Time Clock (RTC) peripheral (a global reset), or software can reset all of the LH79520 except the RTC. Global resets are presented at an output pin for coordination with external circuitry.

The LH79520 offers 5 programmable 'power modes' of operation ranging from 'fully active' to 'fully stopped'. When fully stopped, all clocks, the internal Phase-Locked Loop (PLL) system and the PLL's on-chip oscillator circuit are inactive. A fully-stopped LH79520 can only be reactivated by an external interrupt, or by the Real-Time Clock (RTC), which uses a separate 32.768 kHz crystal oscillator system (timebase). These features extend battery life. See Chapter 7 – Reset, Clock Generation, and Power Control (RCPC) for more information about these features.

Chapter 3
ARM Core and Buses

This chapter of this User's Guide presents details of the ARM720T Core, AHB and APB buses, and bus clocking modes. A discussion of the Theory of Operation is followed by descriptions of the internal registers mentioned in the discussion.

3.1 Theory of Operation

The LH79520 includes an ARM720T core, 32KB of internal SRAM, and memory resources (on-chip memory controllers) for a glueless interface to external memories. The core and memory controllers utilize an Advanced High-Performance Bus (AHB) and interface to external memory through an External Bus Interface (EBI). Essential on-chip peripherals are accessed via a 32-bit Advanced Peripheral Bus (APB) connected to the AHB by a 32-bit bridge. Figure 3-1 shows the core, buses, memory controllers, EBI, and internal SRAM.

The LH79520 includes three AHB bus masters (for the ARM core, the on-chip Color LCD Controller, and the on-chip DMA Controller), with a fixed hardware priority. Details of the ARM720T core are presented in this chapter. The DMAC is explained in Chapter 10 – DMA Controller (DMAC) and the CLCDC is explained in Chapter 11 – Color LCD Controller.

The ARM720T and the AHB may be operated (clocked) in any one of three different modes. For more information about the bus clocking modes, see Section 3.8.

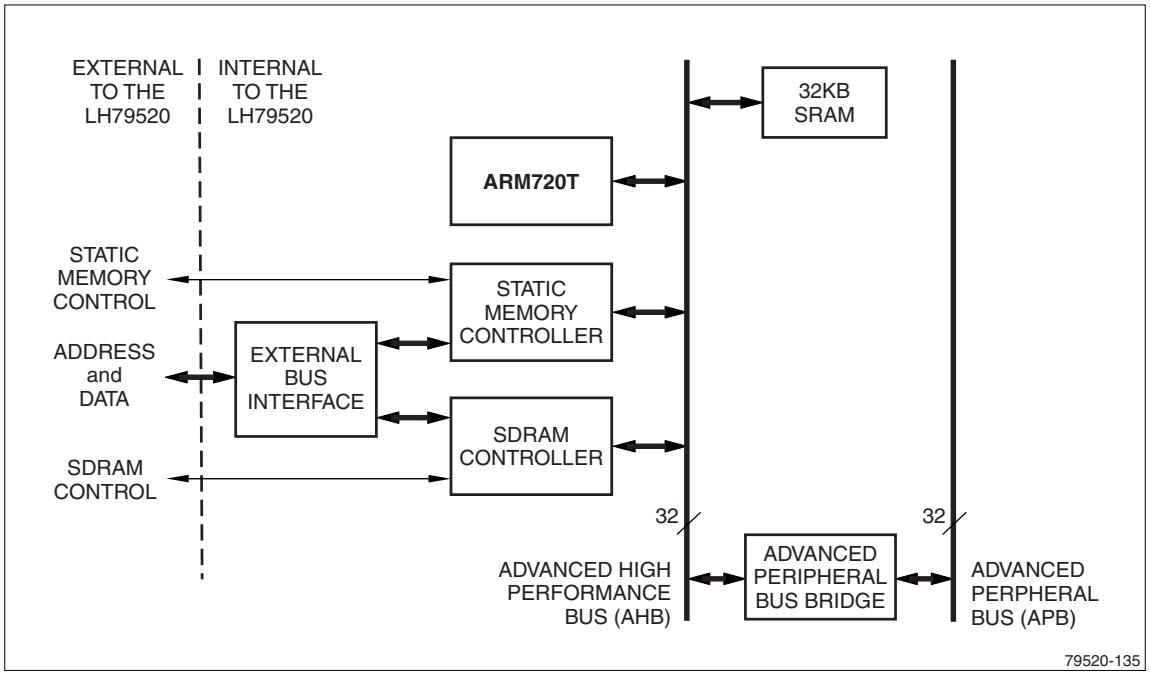


Figure 3-1. LH79520 ARM Core and Memory Interfaces

3.1.1 External Memory Interfaces

The LH79520 can be interfaced to a variety of memory technologies. Figure 3-1 shows that the LH79520 includes an External Bus Interface (EBI) and two specialized memory controllers: a Static Memory Controller (SMC) and an SDRAM Memory Controller (SDRC).

The SDRC is specialized for SDRAM devices and the SMC manages all other external memory. The SDRC provides chip selects for SDRAM memory and the SMC provides chip selects and also byte lane control signals for other types of external memory.

The SMC is for use with Flash, ROM, RAM and external peripherals accessed in similar fashion. The SDRC is for use with Synchronous Dynamic RAM devices. The control signals associated with both memory controllers are listed in the 'Memory Interface' segment of Table 1-1, which lists each of the LH79520 pins. The SMC also provides 'byte-lane' signals to interface with 8-, 16-, and 32-bit wide devices.

Although SMC banks can be composed of memory of varying technologies and widths, all devices within a bank of memory controlled by a controller must have similar access characteristics, such as bus-width, access time, and number of wait states. When memory is addressed on Word (4-byte) boundaries, both controllers provide sequential Load\Store contiguity between the external memory banks they govern. The SMC includes an nWAIT signal handshake to external memory, or software can utilize programmable wait-states to provide delays where necessary. See Chapter 5 – External Static Memory Controller (SMC) and Chapter 6 – SDRAM Memory Controller (SDRC) for additional information about these memory controllers.

Each of the memory controllers in the LH79520 is assigned to a specific portion of the memory map. For external memory, this assignment determines the range of addresses over which each chip select signal is valid. At reset, the LH79520 mirrors the external memory accessed by SMC Chip Select 0 to the lowest page, to assure that ROM'd exception-vectors are correctly positioned at addresses 0x00000000 through 0x0000001C. After reset, software can use RCPCRemapCtrl:REMAP (the REMAP bit field in the RCPCRemapCtrl register) to alter the LH79520 memory-map to mirror any one of the three memory technologies (internal SRAM, SMC-controlled memory, or SDRC-controlled memory) to the lowest page. Software can also program the core's Memory-Management Unit (MMU) to alter the LH79520 memory map. See Section 3.2 for more information.

3.1.2 AHB Master Endian-ness

The ARM720T core is dynamically configurable to little- or big-endian, via CP15 (Control) Register1:B. The reset default is little-endian. See the ARM Core documentation for further information.

The Color LCD Controller is also dynamically configurable, via the LCDControl Register:BEBO. The reset default is little-endian. See Section 11.3.8 for further information.

The DMA Controller, Static Memory Controller, and Dynamic Memory Controller are all hardware-defaulted to little-endian.

3.2 Memory Map

The LH79520 can utilize internal (on-chip) and also external memory. The internal RAM memory is fixed at 32KB, but the amount and type of external memory will depend upon the application. The LH79520 provides an on-chip External Bus Interface (EBI) and on-chip memory controllers for two different types of external memory technologies. An on-chip SDRAM Memory Controller (SDRC) provides the interface signals for external SDRAM, often a system's only external, volatile memory. An on-chip Static Memory Controller (SMC) provides the interface signals for other types of external memory (RAM, ROM, Flash, etc.), and for external devices that utilize a similar memory interface.

The LH79520 uses little-endian storage (defined in the Glossary) and the LH79520 memory-map is re-configurable, as shown in Figure 3-2. Each configuration maps (mirrors) a different type of memory to the lowest range of addresses, 0x00000000 through 0x1FFFFFFF, while also keeping the memory available at the memory's usual range of addresses. This flexibility permits the system to boot from ROM, for example, but execute from RAM after the system is initialized. Designers can use this flexibility to optimize their system's performance at low cost.

Software can reconfigure the LH79520 memory map by a write to RCPCRemapCtrl:REMAP. Figure 3-2 shows the LH79520 memory map at reset, with external static memory mirrored in the lowest page, to accommodate external memory devices containing the system boot code.

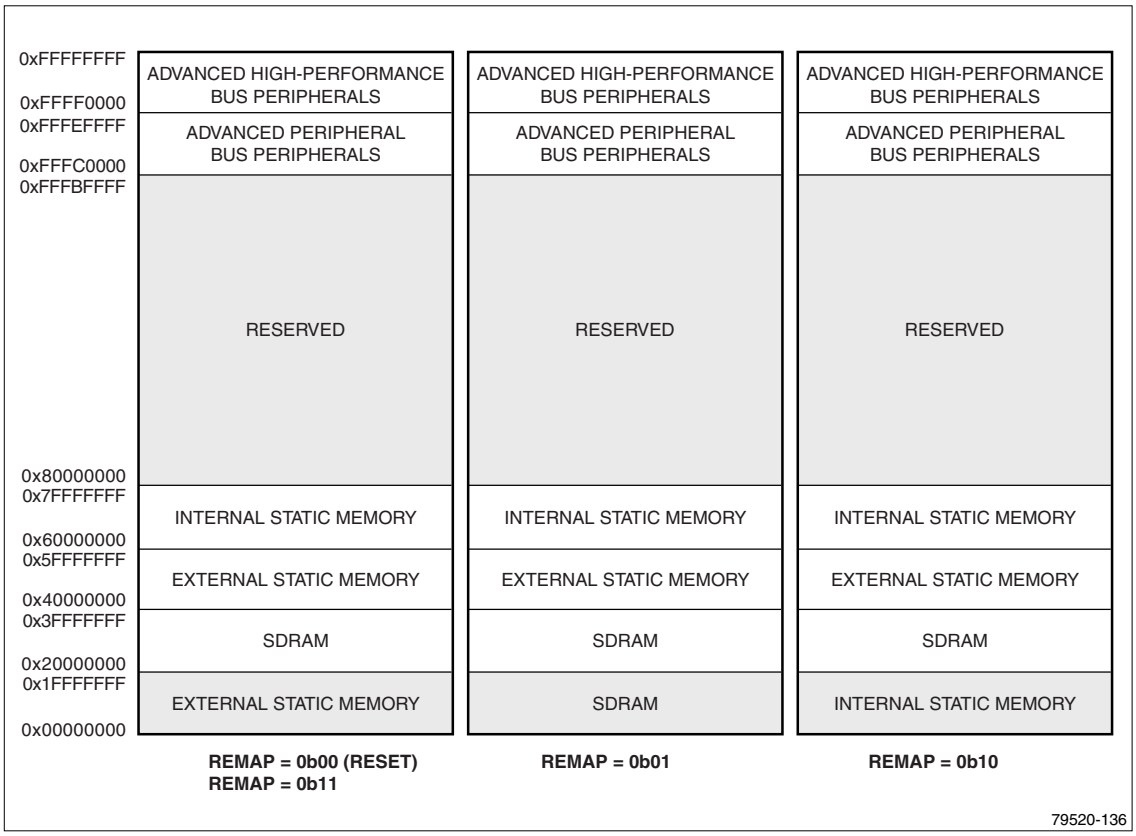


Figure 3-2. LH79520 Memory Map Variations

Figure 3-2 also shows the memory map with SDRAM mirrored to the lowest page, with RCPCRemapCtrl:REMAP = 0b01. This is useful for systems having only external boot ROM and external SDRAM memory. Alternatively, programming RCPCRemapCtrl:REMAP = 0b10 mirrors the 32KB of internal SRAM to the lowest page. This configuration can provide fast access to the system interrupt vectors.

3.3 Instruction and Data Cache

The ARM720T Core includes an 8K Cache, Cache Controller, Memory-Management Unit (MMU), and Write Buffer. One cache is used for both instructions and data. The cache is an important core feature because the LH79520 AHB carries all Core, DMA and LCD Display traffic. For best bandwidth utilization, software should be structured to ensure that the Core is running from within its cache whenever possible.

At reset, the Write Buffer, Cache, and MMU are disabled and the MMU's Translation Lookaside Buffer (TLB) is flushed.

If the MMU is utilized, then software can determine memory cachability by segment or by page. For best performance, the Color LCD frame buffer should not be located in a cachable region.

3.4 Memory Management Unit (MMU)

The ARM720T core in the LH79520 includes a Memory Management Unit (MMU), which performs three primary functions: It translates virtual addresses into physical addresses, it enables cache and write buffering for particular ranges of virtual addresses, and it controls memory access permissions. When the MMU is turned off, as it is at reset, all virtual addresses are output directly onto the physical address bus (the AHB).

The MMU supports memory accesses based on 'sections' or 'pages' of memory. Sections are 1MB blocks of memory; pages can be either small or large. Small pages consist of 4KB blocks of memory. Additional access control mechanisms are extended to 1KB subpages. Large pages consist of 64KB blocks of memory. Large pages are supported to allow mapping of a large region of memory while using only a single entry in the Translation Lookaside Buffer. Additional access control mechanisms are extended to 16KB subpages.

For more information about the core, cache and MMU, refer to the ARM document 'ARM720T Processor Data Sheet', available through ARM, Ltd., at http://www.nxp.com/redirect/arm.com/.

3.5 Advanced High-Performance Bus (AHB)

The LH79520 includes an ARM Advanced High-Performance Bus (AHB). The AHB is the 'backbone' of this MCU because it connects the core and all other on-chip systems capable of mastering the AHB with all memory resources and peripherals.

Table 3-1 shows how the Controllers are mapped to the AHB. This address scheme allows up to 4KB of register and FIFO space for each peripheral. These Controllers all support word, half-word, and byte writes to their registers.

It should be noted that while the Color LCD Controller itself supports word, half-word, and byte writes, the Advanced LCD Interface does not. Although LCD drive signals are routed through the ALI, its addressing is via the APB, and therefore only supports word accesses.

Note that certain ranges of addresses are reserved for future expansion. Accessing these reserved addresses is discouraged and may result in unpredictable behavior.

Table 3-1. AHB Device Addressing

BASE ADDRESS	BASE NAME	CONTROLLER
0xFFFFF000	VICBase	Vectored Interrupt Controller
0xFFFF5000 - 0xFFFFEFFF	///	Reserved
0xFFFF4000	CLCDBase	Color LCD Controller
0xFFFF3000 - 0xFFFF3FFF	///	Reserved
0xFFFF2000	SDRAMBase	SDRAM Controller
0xFFFF1000	SMCBase	Static Memory Controller
0xFFFF0000	VICBase (mirrored)	Vectored Interrupt Controller

NOTE: /// = Reserved.

3.5.1 AHB Masters and Arbitration

The LH79520 includes three AHB masters — the ARM720T processor, the DMA controller and the Color LCD Controller. All are capable of communicating with each other and the peripherals. The Color LCD Controller interfaces to the AHB via a slave interface for programming, and via a master interface for accessing the SDRAM Memory Controller (SDRC), Static Memory Controller (SMC), and External Bus Interface (EBI).

3.5.2 AHB Master Priority

The Color LCD Controller has the highest priority, to ensure that the LCD display is refreshed correctly during DMA transfers or when the MCU is in the Standby power mode. Programmers must be aware that the Color LCD Controller has the potential to lock out both the DMA Controller and the ARM720T Core because they have a lower priority.

The DMA Controller has middle priority, higher than the Core but lower than the Color LCD Controller. This allows DMA transfers to continue when the LH79520 is operating in the Standby Mode, when the Core is not being clocked.

The ARM720T Core is the default (lowest priority) bus master. The Core will master the bus most often and this priority minimizes latency for the core. Table 3-2 shows these priorities.

Table 3-2. Priorities Among Bus Masters

PRIORITY	BUS MASTER
Highest	Color LCD Controller
Middle	DMA Controller
Lowest	ARM720T (the default and lowest Priority)

3.5.3 AHB Errors

Three LH79520 AHB slaves are capable of issuing an AHB error response: the Static Memory Controller (SMC), the Vectored Interrupt Controller (VIC), and the AHB address decoder.

The SMC can issue an AHB error response if a write is attempted to a memory bank that is defined as write-protected.

The VIC can issue an AHB error response if a byte or half-word (16-bit) AHB access to the VIC is attempted.

The AHB address decoder can issue an AHB error response if an access is attempted to a reserved portion of the AHB memory map (listed in Table 3-1).

3.6 Advanced Peripheral Bus (APB)

Figure 2-1 (the simplified block diagram) shows that the LH79520 includes an ARM 32-bit Advanced Peripheral Bus (APB) to interface its on-chip peripherals to the AHB through a bridge. The APB is designed for low-speed peripherals that do not require the performance of a pipelined interface such as the AHB.

The APB does not support byte and half-word writes. A byte write operation at a register address will affect all four bytes in the register equally. Likewise, a half-word write will affect both half-words in the register equally (where hardware allows all bits to be written). Byte writes to registers that are 8 bits or less and half-word writes to registers that are 16 bits or less can be performed, but such operations do not save any cycles and this practice is not recommended.

The AHB and APB clocks both operate at the same frequency. Table 3-3 shows the addresses of each of the peripherals on the APB.

3.7 AHB-to-APB Bridge

The AHB-to-APB Bridge provides the ARM720T Core with transparent access to each peripheral on the APB. These peripherals can be serviced by the Core or, if the peripheral is programmed to be DMA-enabled, by the DMA Controller. Using a DMA operation to access a peripheral can improve system performance if the software is structured to ensure that the ARM720T Core is running from within its cache memory during the DMA. Note that if the Core is not running in cache, or needs to access the AHB (or a peripheral on the APB bus), the access can be blocked by the DMA Controller, which has a higher priority. Table 3-3 shows how the peripheral devices are mapped to the APB.

Table 3-3. APB Device Addressing

BASE ADDRESS	BASE NAME	PERIPHERAL
0xFFFC0000	UART0Base	UART 0
0xFFFC1000	UART1Base	UART 1
0xFFFC2000	UART2Base	UART 2

Table 3-3. APB Device Addressing

0xFFFC3000	PWMBase	Pulse Width Modulators
0xFFFC4000	TimerABase	Dual Timer A
0xFFFC5000	TimerBBase	Dual Timer B
0xFFFC6000	SSPBase	Synchronous Serial Peripheral Interface
0xFFFC7000 - 0xFFFDBFFF	///	N/A
0xFFFDC000	GPIOBase 3	General Purpose I/O (Ports G and H)
0xFFFDD000	GPIOBase 2	General Purpose I/O (Ports E and F)
0xFFFDE000	GPIOBase 1	General Purpose I/O (Ports C and D)
0xFFFDF000	GPIOBase 0	General Purpose I/O (Ports A and B)
0xFFFE0000	RTCBase	Real Time Clock
0xFFFE1000	DMABase	DMA Controller
0xFFFE2000	RCPCBase	Reset, Clock & Power Controller
0xFFFE3000	WDTBase	Watchdog Timer
0xFFFE4000	ADTFTCBase	AD-TFT/HR-TFT LCD Timing Controller
0xFFFE5000	IOCONBase	I/O Configuration Peripheral
0xFFFE6000 - 0xFFFFEFFF	///	N/A

NOTE: /// = Reserved.

3.8 Bus Clocking Modes

The clocking modes concern the relative frequencies at which the ARM720T core and the AHB are operated, and the associated AHB access considerations. The ARM720T core (including the cache) and its AHB interface can be operated (clocked) using either the Fast-bus mode of operation or one of two Standard clocking modes (Synchronous or Asynchronous). The clocking modes can have significant impact on power consumption and system throughput, depending upon the application and the speed of external memory.

The ARM720T core and the AHB are clocked by separate signals and the core is capable of operation at much higher frequency than the AHB. This higher core speed benefits applications running from the cache more than applications requiring frequent AHB access because each AHB access requires that the core and AHB be re-synchronized. Parallel core and AHB operations can continue with buffered writes to the AHB but all read accesses will stall the core until the bus access is completed. Programmers can use the three bus clocking modes to maximize throughput by reducing the re-synchronization delays (the quantity of 'WAIT' states).

The bus clocking modes discussed in this chapter are set by the value programmed to the CoreClkConfig register. The clock frequencies are set by the HClkPrescale and CpuClk-Prescale registers, discussed in Chapter 7 – Reset, Clock Generation, and Power Control (RCPC). Table 3-4 summarizes the LH79520 bus clocking modes.

3.8.1 Standard Bus Clocking Modes

The Standard bus clocking modes are useful for designs involving low-cost, low-speed memory, where operation of the core at a faster speed than the AHB is desired. These modes involve:

- A programmable choice of Synchronous or Asynchronous operation
- Two clocks: HCLK and FCLK

The AHB interface is controlled by the bus clock (HCLK), qualified by a WAIT signal. Figure 3-3 shows the Standard mode clocking arrangement. The core and cache are driven by FCLK while HCLK drives the bus. FCLK must always be ≥ HCLK, on a cycle-by-cycle basis. The WAIT signal can extend a memory access by inserting entire HCLK cycles into the bus cycle timing.

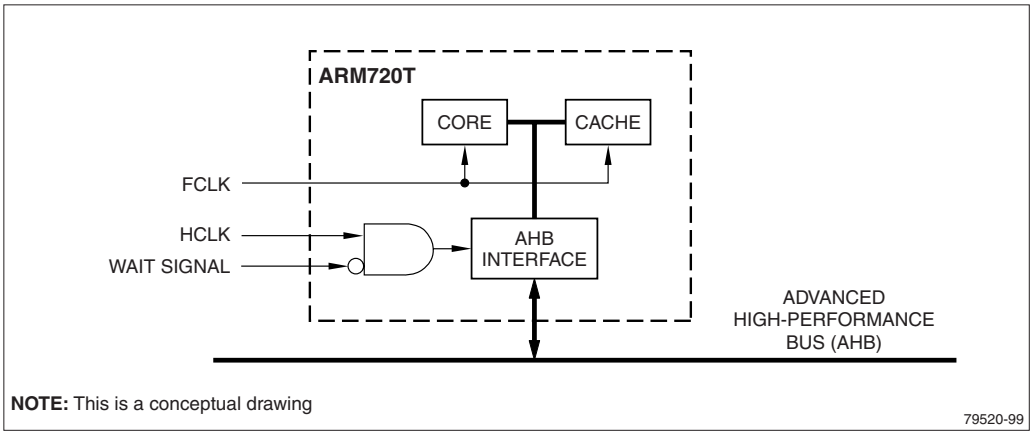


Figure 3-3. Standard Mode Clocking

3.8.1.1 Synchronous and Asynchronous Bus Clocking Modes

Although the frequency of FCLK must always be greater than or equal to HCLK, the two Standard modes vary the relationship between these two clock signals. In the Synchronous mode, the FCLK frequency must be programmed to be an even, integer multiple of the HCLK frequency; an even harmonic. Bus accesses in the Synchronous mode will require a re-synchronization delay of at least one WAIT state. In the Asynchronous mode the harmonic relationship between the clocks need not be maintained; the two clock signals may be of unrelated frequency. Bus accesses in the Asynchronous mode will require a minimum re-synchronization delay of two WAIT states.

3.8.2 Fastbus Extension Bus Clocking Mode

Designs involving frequent accesses of high-speed memory may benefit by using the Fastbus Extension clocking mode. This inherently synchronous mode clocks the core, cache and AHB at the same frequency. Where the Standard modes utilized two different clocks, the Fastbus mode operates the core, cache and AHB interface with two signals derived from the same source; essentially the same clock. Figure 3-4 shows the Fastbus Extension mode clocking arrangement. The Fastbus Extension mode does not require re-synchronization delays.

The Fastbus Extension mode is useful for applications involving frequent AHB accesses. Although the core's frequency is limited by the AHB maximum frequency, the Fastbus Extension mode avoids the WAIT state penalties imposed by the Standard modes.

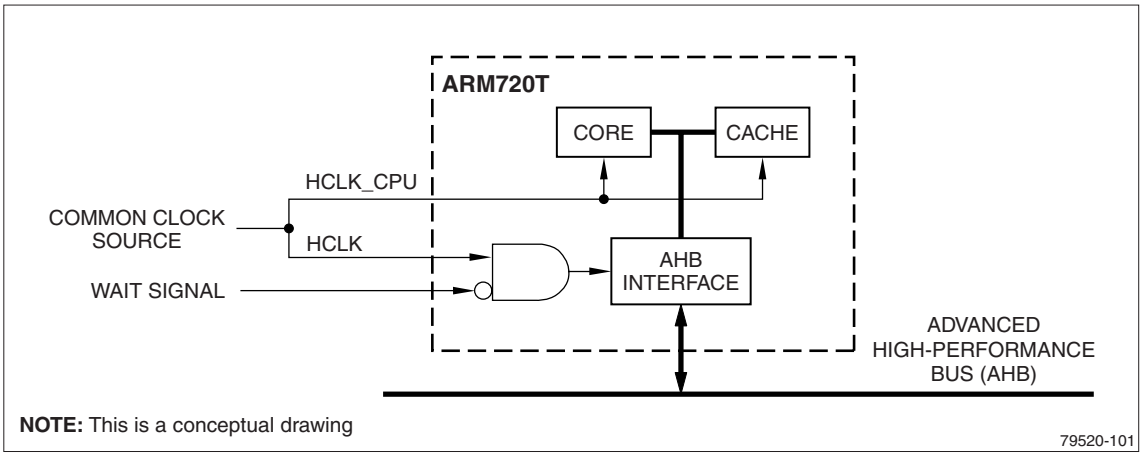


Figure 3-4. Fastbus Mode Clocking

3.8.3 Clocking Power Considerations

Because the ARM720T is a fully static design, the LH79520 core and bus clock signals can be stopped indefinitely, with no loss of internal data, in any of the bus clocking modes.

3.8.4 Bus Clocking Modes Summary

Table 3-4 summarizes the three Bus Clocking Modes. Also see the Reset, Clock Generation and Power Control chapter of this User's Guide and the sections that detail the CoreClkConfig, HClkPrescale and CpuClkPrescale registers.

Table 3-4. Standard versus Fastbus Mode Comparisons

ITEM	STANDARD		FASTBUS
	SYNCHRONOUS	ASYNCHRONOUS	
Clock signals	HCLK, FCLK	HCLK, FCLK	HCLK, HCLK_CPU
What the clock signals do	HCLK clocks the AHB, FCLK clocks the core.	HCLK clocks the AHB, FCLK clocks the core.	HCLK clocks the AHB, HCLK_CPU clocks the core.
Clock signal frequency limitations	Requires FCLK ≥ HCLK, and FCLK must be an integer multiple (a harmonic) of HCLK.	Requires FCLK ≥ HCLK but there are no other limitations.	The speed at which the core can be operated is limited by the maximum AHB speed.
Synchronization	The core and the AHB are always synchronized but may be operated at different frequencies.	The core and the AHB operate asynchronously.	The core and the AHB are inherently synchronized. No additional synchronization logic is required.
AHB access	1 wait state minimum	2 wait states minimum	No wait states
Advantages	• The core can be operated much faster than the AHB but the core speed must be an even, integer multiple (A harmonic) of the AHB frequency. • A slower HCLK can reduce the power consumption.	• The core can be operated much faster than the AHB, and the core and the AHB can be operated at different, unrelated frequencies. • A slower HCLK can reduce the power consumption. The bus frequency may be optimized for a specific peripheral. • Clocking requirements are dependant upon the CPU operating speed.	• The core and the AHB operate at the same frequency, driven by the same clock. • Inherent synchronization avoids the logic required in the other two bus-clocking modes and so requires no wait states.
Limitations	The speed at which the core can be operated is limited by the maximum core and AHB speeds. Memory accesses will impose a delay if the core and bus are operating at different speeds.	Memory accesses will impose a delay because the core and the AHB must be re-synchronized whenever the core must access the bus.	The speed at which the core can be operated is limited by the maximum AHB speed. Memory accesses will impose no delay for synchronization.
Uses	For CPU-intensive applications	For CPU-intensive applications requiring a specific source clock frequency for on-chip peripherals. The peripheral clocks are derived from the bus clock.	For memory-access intensive applications

3.9 System Performance Considerations

System performance is expressed as 'Bandwidth', a measure of the speed with which the LH79520 can transfer data across the AHB. System bandwidth is a function of the frequency at which the AHB operates and also a function of the characteristics of the endpoints of the transfers (source and destination). The access requirements of the endpoints will limit the bandwidth for transfers to and from those endpoints.

The LH79520 bus architecture places a high demand on its AHB, which carries the LCD data as well as all DMA and non-cached core instructions and data. An AHB operating at 50 MHz can produce 50,000,000 clock cycles per second. In this period of time, assuming 32-bit words and no latencies, the quantity of words (in Millions) that can be fetched are:

Theoretical Maximum Bus Bandwidth= 50 ÷ Bytes per Word = 2.5 MWords/sec

Note again that the above formula does not account for latencies.

Be sure to consider the bandwidth budget carefully. Designers must provide the appropriate memory technologies and programmers will wish to utilize DMA operations wherever possible, to optimize the bus traffic.

Systems with LCD Displays must meet the refresh requirements of the display. This performance can consume much of the available bus bandwidth, so designers often consider it first. The bandwidth is utilized by the Color LCD Controller (CLCDC) in the regular transfer of display data from the frame buffer to the Color LCD Panel.

Depending upon the size of the LCD Panel, programmers may choose to locate the frame buffer in internal SRAM or external, volatile memory devices such as SRAM or SDRAM. The LH79520 provides all necessary interface signals for each of these memory technologies.

3.10 JTAG Considerations

The LH79520 boundary scan system is not IEEE 1149.1 compliant in two different areas.

The test mode select pins nTSTA, TEST1, and TEST2 are included in the boundary scan chain. These pins, when set to 101 (respectively) before a rising edge of nRESETIN, are used to activate the JTAG interface. However, boundary scan vectors that are generated using the BSDL file float these inputs because they are part of the boundary scan chain.

As a result, the JTAG interface enters an undefined state when the vectors are exercised, causing the boundary scan to be non-functional. The only way to insure a proper Reset when using boundary scan would be to add external hardware to hold thse pins at the proper levels during that time.

Also during boundary scan operation, the LH79520 outputs change state on the rising edge of the Test Clock (TCK). The IEEE 1149.1 specification states that signals driven out of the component via the boundary scan logic must change state relative to the falling edge of TCK.

3.11 Core Configuration Programmer's Model

The registers which control the ARM Core and memory configurations described in this chapter of this User's Guide are located in the Reset, Clock and Power Control (RCPC) section of the LH79520 memory map.

The Register Base Address for the ARM720T Core and memory configuration registers is:

RCPCBase: 0xFFFE2000

A complete listing of all programmable registers in the LH79520, arranged by address, is presented in Chapter 20 – Status and Configuration Registers.

3.12 Core Configuration Register Summary

Table 3-5 summarizes the programmable registers that configure the ARM720T Core and Memory Systems. Each register is treated in more detail on the following pages.

Table 3-5. ARM Core and Memory Configuration Registers

ADDRESS	REGISTER NAME	DESCRIPTION
RCPCBase + 0x008	RCPCRemapCtrl	Configures the LH79520 Memory Map
RCPCBase + 0x088	CoreClkConfig	ARM Core Clock configuration

3.13 Core Configuration Register Descriptions

This section of this User's Guide lists each of the registers that configure the LH79520 ARM Core and Memory Systems, and describes its function.

3.13.1 RCPC Remap Control Register

The RCPCRemapCtrl register can alter the LH79520 Memory Map.

The LH79520 memory map can be set to one of three different configurations.

Table 3-6 shows the register that controls the memory map and Table 3-7 explains the register's bit fields.

Table 3-6. RCPCRemapCtrl Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///														REMAP	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	RCPCBase + 0x008															

Table 3-7. RCPCRemapCtrl Register Bit Fields

BITS	FIELD NAME	FUNCTION
15:2	///	**Reserved** Write the reset value.
1:0	REMAP	**Remap Control** 0b00 = Locates External Static Memory at address 0x00000000 (reset) 0b01 = Locates SDRAM Memory at address 0x00000000 0b10 = Locates Internal Static Memory at address 0x00000000 0b11 = (Same as 0b00)

Table 3-8 lists the available memory map variations. As shown in Table 3-8 and Figure 3-2, each configuration maps (mirrors) a different type of memory to the lowest range of addresses, 0x00000000 through 0x1FFFFFFF, while also keeping it available at its usual range of addresses.

The shaded areas in Table 3-8 show how the LH79520 physical (not virtual) memory map is altered by the state of the REMAP bit field in the RCPCRemapCtrl Register. See Section 3.4 for more information about physical-to-virtual address mapping.

Table 3-8. LH79520 Memory Map Variations

RCPCRemapCtrl Register:REMAP			
PHYSICAL ADDRESS	**REMAP = 0b00 (RESET) REMAP = 0b11**	**REMAP = 0b01**	**REMAP = 0b10**
0xFFFC0000	System/Periph. Registers	System/Periph. Registers	System/Periph. Registers
0x80000000	///	///	///
0x60000000	Internal SRAM Memory	Internal SRAM Memory	Internal SRAM Memory
0x40000000	External Static Memory	External Static Memory	External Static Memory
0x20000000	SDRAM Memory	SDRAM Memory	SDRAM Memory
0x00000000	External Static Memory (ROM, RAM, Flash, etc.)	SDRAM Memory	Internal SRAM Memory

NOTE: /// = Reserved.

3.13.2 Core Clock Configuration Register

The CoreClkConfig register selects the clocking mode for the ARM720T core, cache and the AHB.

Table 3-9. CoreClkConfig Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///														CFGVAL	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	RCPCBase + 0x088															

This register selects one of three modes of operation for the ARM720T Core and the AHB, shown in Figure 3-5. The three modes of operation are:

- Standard, Asynchronous (the mode at reset)
- Standard, Synchronous
- FastBus Extension

Table 3-10 shows the bit field that affects the mode of operation.

Table 3-10. CoreClkConfig Register Bit Fields

BIT	FIELD NAME	FUNCTION
31:2	///	**Reserved** Write the reset value.
1:0	CFGVAL	0b00 = Standard Mode, asynchronous operation (the value at reset) 0b10 = Standard Mode, synchronous operation 0b01 = FastBus Extension Mode 0b11 = FastBus Extension Mode

In the Standard mode, the FCLK signal clocks the ARM core and the HCLK signal drives the AHB. The ARM core and the AHB can be operated either Synchronously or Asynchronously, but the synchronization logic imposes a minimum delay of one bus clock cycle, even when the core and bus are operating at the same frequency.

In the FastBus Extension mode, the ARM core and AHB operate fully synchronously. This mode uses two identical clocks (HCLK and HCLK_CPU) and avoids all re-synchronization delays.

Figure 3-5 shows that the ARM720T core receives three clock signals from the LH79520 Reset, Clock and Power Control (RCPC) functional block: FCLK, HCLK, and HCLK_CPU.

CoreClkConfig:CFGVAL (the CFGVAL bit field in the CoreClkConfig register) directs the ARM core and AHB interface to use either FCLK and HCLK, or HCLK_CPU and HCLK.

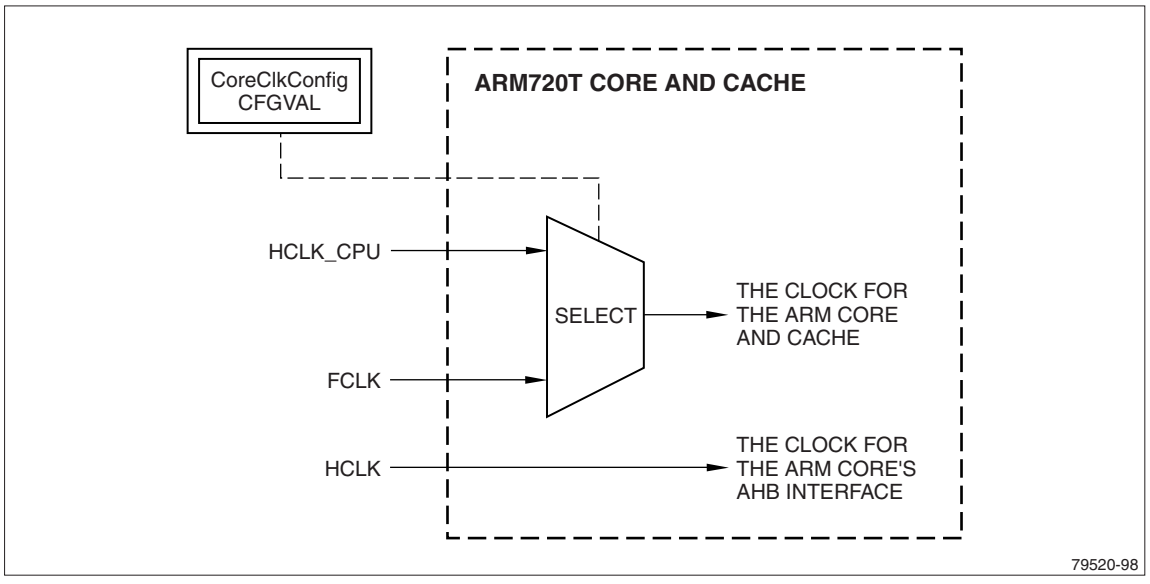


Figure 3-5. Bus Clocking Selection

Chapter 4
Internal Static RAM (SRAM)

This chapter of this User's Guide presents details of the Internal Static Ram (SRAM) included in the LH79520 Universal MCU.

4.1 Internal Static RAM (SRAM)

The LH79520 contains 32KB of internal Static RAM memory (SRAM). This memory is mapped to the AHB and is useful as scratchpad RAM, interrupt vector storage or as a frame buffer for a small LCD Display. Figure 4-1 shows the internal SRAM's initial location.

Although initially mapped as shown in Figure 4-1, this internal SRAM can be mirrored to address 0x00000000 by writing the appropriate value to RCPCRemapCtrl:REMAP (the REMAP field in the RCPCRemapCtrl register). See Chapter 3, Section 3.2 for more information about remapping the internal SRAM memory.

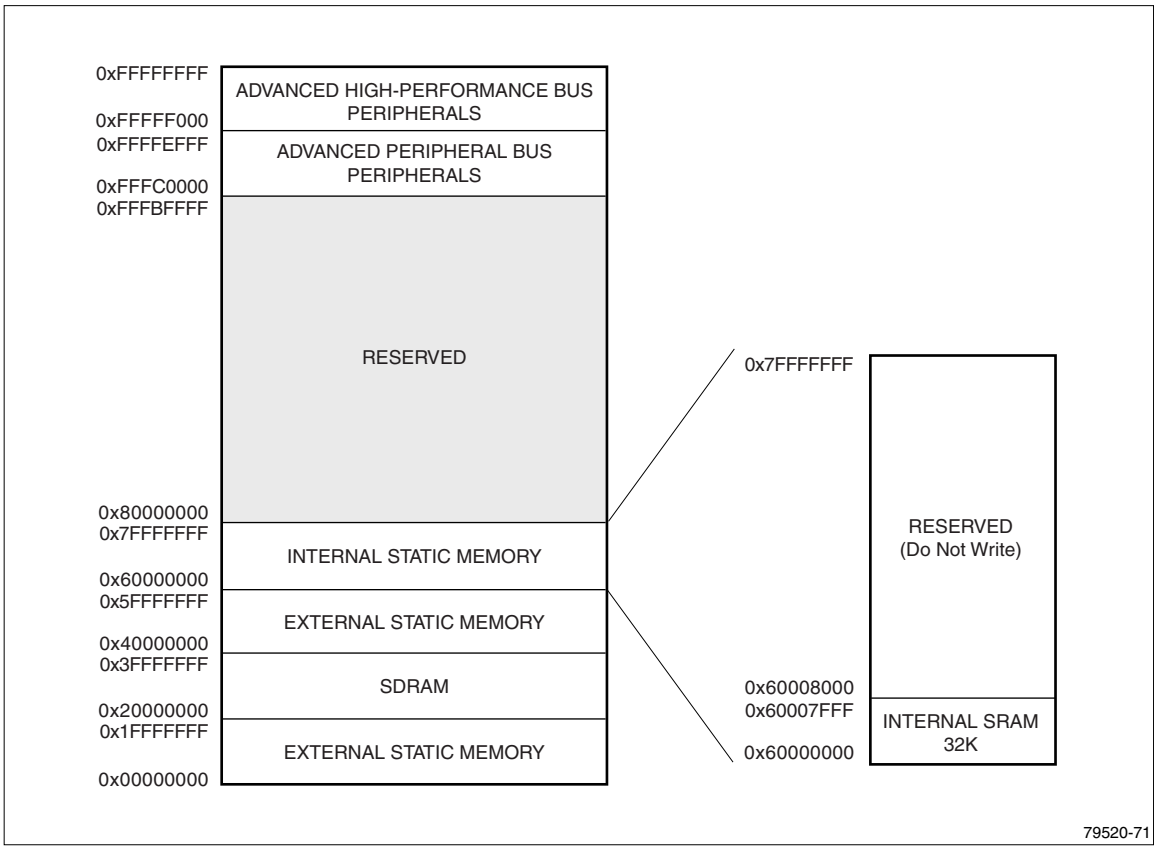


Figure 4-1. Internal Static RAM Locations

Chapter 5
External Static Memory Controller (SMC)

This chapter of this User's Guide presents details of the Static Memory Controller (SMC) in the LH79520 Universal MCU. A discussion of its Theory of Operation is followed by descriptions of the internal registers mentioned in the discussion.

5.1 Theory of Operation

The LH79520 includes a SMC to interface the LH79520 to external (off-chip) memory devices other than Synchronous Dynamic RAM. Figure 5-1 shows a simplified block diagram of the SMC.

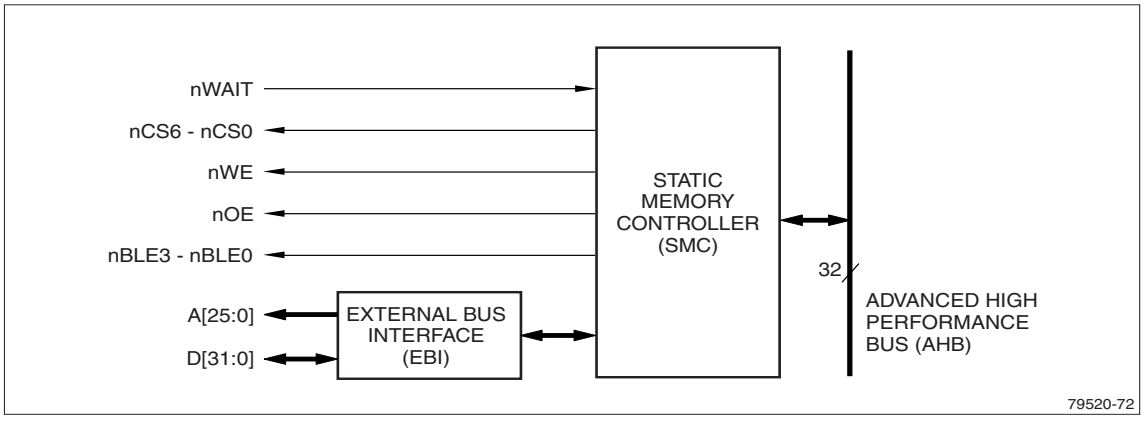


Figure 5-1. Static Memory Controller Block Diagram

The LH79520 is designed to boot from 16-bit wide external memory connected to Chip Select 0 (nCS0), which is managed by the SMC. The following parameters are programmable for each bank of external memory controlled by the SMC:

- Bus turnaround (idle) cycles
- Read and Write WAIT states, for static RAM devices
- Initial and subsequent burst read WAIT states
- Write protection
- Burst mode operation for burst ROM devices
- External memory width of 8-, 16-, or 32-bit
- Byte lane enable controls
- nWAIT handshake

The SMC does not control the 32KB of SRAM internal to the LH79520. The SMC is for use with all external memory devices except SDRAM. The LH79520 includes a separate memory controller for SDRAM. See Chapter 6 for additional information about the SDRAM controller.

The SMC supports asynchronous Page Mode and Burst Mode Read operations and allows up to 32 programmable Wait states and programmable Bus turnaround cycles. The SMC can accommodate up to seven banks of external memory, each up to 64MB in size. The base address of each bank is hard-coded by the address decoder. A separate SMC Chip Select signal (nCS0:nCS6) is provided for each bank of memory, and Byte-Lane Enable signals (nBLE0:nBLE3) are also available for use with memory devices of varying widths. External memory devices configured as 8, 16, or 32 bits wide are acceptable for use with the SMC.

Each of the SMC memory banks has an associated configuration register. A bit field in each configuration register determines the width for the memory accessed by that chip select. The memory banks associated with the nCS0 and nCS1 signals are automatically configured as 16 bits wide at reset.

Table 5-1 and Figure 5-2 show how the banks and chip select signals correspond to the LH79520 memory map. The Figure also illustrates the width of each SMC bank at reset.

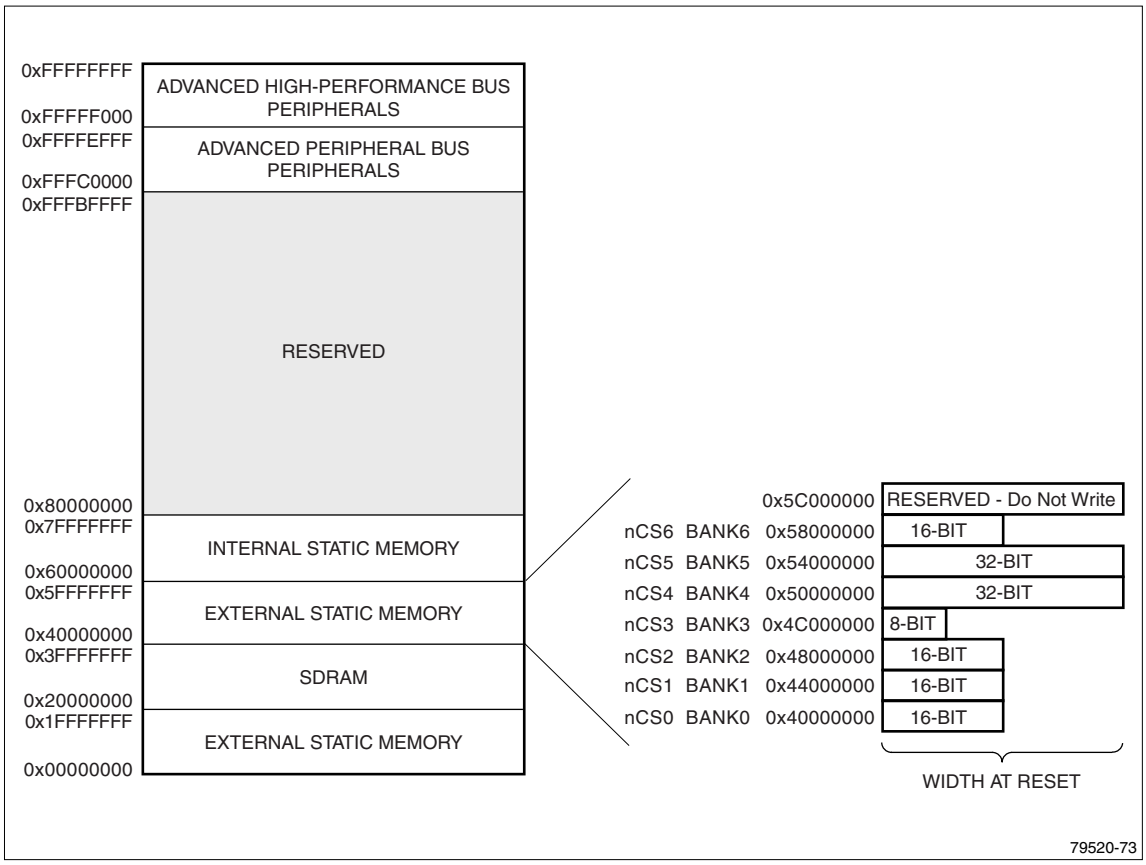


Figure 5-2. External Static Memory Banks, REMAP = 0b00

Figure 5-2 also shows a seventh segment of external static memory extending from 0x5C000000 to 0x5FFFFFFF and marked as 'Reserved'. Software should not access this segment because, although no memory exists there, a data abort will not be generated if this segment is accessed.

Although SMC banks can be composed of memory of varying technologies and widths, all devices within a bank must have the same bus-width, access time and wait states. When memory is addressed on Word (4-byte) boundaries, the SMC controller provides Load/ Store multiple (MDM/STM) contiguity between the external memory banks it governs. The SMC includes an nWAIT signal handshake to external memory, or software can utilize programmable wait-states to provide access delays where necessary.

5.1.1 Remapping SMC Memory

Each LH79520 memory controller is assigned to a specific portion of the memory map. For external memory controlled by the SMC, this assignment determines the range of addresses over which each SMC Chip Select signal (nCS0:nCS7) is valid. At reset, the LH79520 mirrors the external memory accessed by SMC Chip Select 0 (nCS0) to the lowest page, to assure that ROM'd exception-vectors are correctly positioned at addresses 0x00000000 through 0x0000001C. After reset, software can use the RCPCRemapCtrl register to alter the LH79520 memory-map to mirror any one of the three memory technologies (internal SRAM, SMC-controlled memory, or SDRC-controlled memory) to the lowest page. Software can also program the Core's Memory-Management Unit (MMU) to alter the LH79520 memory map. See Chapter 3 – ARM Core and Buses for more information about the RCPCRemapCtrl register and the MMU.

5.1.2 External Memory Bank Width

As shown in Table 5-1 and Figure 5-2, the width of each bank of external memory controlled by the SMC is programmable. Figure 5-2 shows the default widths at reset, but software can redefine each bank to match the actual hardware requirements (8-, 16-, or 32-bits wide). The LH79520 provides byte-lane enable signals (nBLE0:nBLE3) as well as SMC Chip Select signals (nCS0:nCS7) to support this feature. This flexibility is useful for interfacing the LH79520 with a broad variety of external memory devices.

The designed width of the external memory hardware does not impact the ARM conventions for data-alignment. The LH79520 can store and retrieve bytes at any address. Halfwords can be stored at any even addresses and words are stored on 4-byte boundaries. All 32-bit (word) accesses are aligned on 32-bit boundaries.

Table 5-1. SMC Memory Bank Addresses and Chip Selects

ADDRESS RANGE	SMC CHIP SELECT	DESCRIPTION	WIDTH AT RESET
0x5C000000 - 0x5FFFFFFF	///	Reserved, do not write	///
0x58000000 - 0x5BFFFFFF	nCS6	SMC Chip Select 6	16-bit
0x54000000 - 0x57FFFFFF	nCS5	SMC Chip Select 5	32-bit
0x50000000 - 0x53FFFFFF	nCS4	SMC Chip Select 4	32-bit
0x4C000000 - 0x4FFFFFFF	nCS3	SMC Chip Select 3	8-bit
0x48000000 - 0x4BFFFFFF	nCS2	SMC Chip Select 2	16-bit
0x44000000 - 0x47FFFFFF	nCS1	SMC Chip Select 1	16-bit
0x40000000 - 0x43FFFFFF	nCS0	SMC Chip Select 0	16-bit

NOTE: /// = Reserved.

5.1.3 External Static Memory Interfacing

The LH79520 Static Memory Controller (SMC) can support seven independently configurable banks of external memory, with a separate chip select for each bank. The SMC provides control signals for 8-, 16-, and 32-bit wide banks of external memory, which must be connected correctly and configured by software (by writes to registers SMCBCR0:SMCBCR6).

Note that the memory connected to SMC Chip Select nCS0 (bank 0 memory) must be connected for 16-bit operation in order to match the LH79520 boot configuration.

The SMC's control logic generates the following control signals for external memory:

- Chip Select and Write Enable signals
- Output Enable and Byte Lane Enable signals
- Data sequencing for memory reads and writes

The SMC's control logic also supports an nWAIT input for access delay.

5.1.3.1 Access Sequencing and Data Width

The data width of each external memory bank should be configured by programming the appropriate bank configuration register, SMCBCR0:SMCBCR6. When the external memory bus is more narrow than the transfer initiated from the current bus master, the internal bus transfer will require several external bus transfers to complete.

For example, if Bank 0 is configured as 8-bit wide memory and a 32-bit read is initiated, the AHB will stall while the SMC reads four consecutive bytes from external memory. During these accesses, the data path is controlled (in the external memory data path logic) to demultiplex the four bytes into one 32-bit word on the AHB.

5.1.3.2 Wait State Generation

Each bank of external memory should be configured for the appropriate number of wait states for read and write accesses. This configuration is achieved by programming the WST1 and WST2 bit fields in the SMC bank configuration registers, SMCBCR0:SMCBCR6.

Wait State control refers to external bus transfer wait states. The number of cycles required for a bus transfer is controlled by two other factors: access width and external memory width.

The WST1 field in each configuration register can be programmed to select the wait states for read memory accesses to ROM or SRAM, or the initial burst read access to burst ROM. The WST2 wait state field in each register can be programmed to select the wait states for write accesses to SRAM devices, or for each access during burst mode reads from Burst ROM devices. Each memory bank can also be configured to use external bus turnaround cycles between read and write memory accesses (the IDCY bit field in the SMCBCRx registers).

5.1.3.3 The nWAIT Function

The nWAIT function can be utilized to extend SMC memory accesses beyond the number of wait states programmed to the SMC Bank Configuration registers. See Section 5.4.1 for information about the Wait State 1 (WST1) and Wait State 2 (WST2) bit fields in the SMCB-CRx registers.

Refer to Figure 5-3 for the following discussion:

- If the nWAIT function is selected, then the SMC will monitor the nWAIT input during every memory access (read or write cycle). When the SMC is monitoring nWAIT, the SMC inspects the nWAIT input on each system clock (HCLK) edge.
- The system clock edge following the edge during which the nCSx signal is asserted is considered to be the end of the first wait state.
- The SMC will recognize an asserted (LOW) nWAIT input two system clock (HCLK) cycles after the external device asserts the nWAIT signal.
- The Chip Select must be programmed for at least two wait states. If only two wait states are programmed, the external device must assert nWAIT in the cycle immediately following the cycle during which the SMC Chip Select (nCS0:nCS7) is asserted.
- If a Chip Select (nCS0:nCS7) is programmed for *N* wait states, then the SMC will assert all accesses of the memory selected by that Chip Select for *N* system clock cycles, or until the SMC recognizes that the nWAIT signal is deasserted (HIGH), whichever occurs last.
- When the SMC recognizes that the external device has deasserted nWAIT, the SMC will complete the current access (read or write cycle) in three system clock (HCLK) cycles.

Refer to Chapter 21 for more information on the use of nWAIT.

5.1.3.4 Write Protection

Each bank of external memory controlled by an SMC Chip Select (nCS0:nCS6) may be configured for write protection. Although external SRAM will normally be unprotected, and ROM devices will normally be write-protected, the external SRAM devices may also be write protected if desired. A write access to a write-protected bank of memory will generate an error, but a write access to unprotected ROM, for example, will not cause an error.

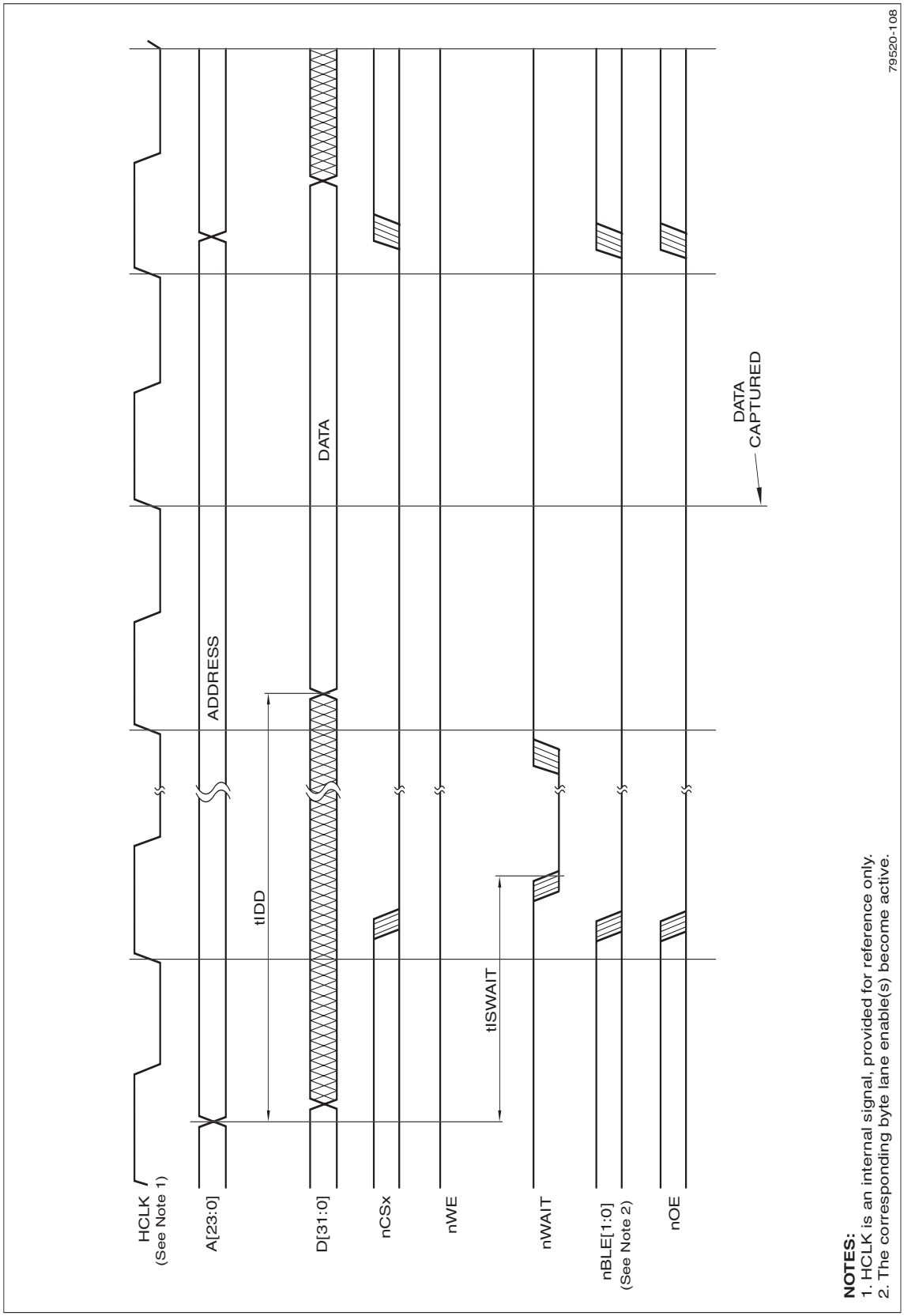


Figure 5-3. External Memory Read, nWAIT Active

5.1.4 Interfacing to SRAM, ROM and Flash Devices

The SMC provides the same read timing control for SRAM, ROM and Flash devices. Each read begins with an assertion of the appropriate SMC Chip Select signal (nCS0:nCS6) and memory address A[26:0].

Read access time is determined by the programmed quantity of wait states, SMCBRx:WST1 (the WST1 bit field in one of the SMCBCRx registers). The quantity of bus turnaround wait states that will be added between external read and write transfers is determined by SMCBCRx:IDCY. See the LH79520 Data Sheet.

5.1.4.1 Burst ROM

The SMC supports sequential access burst reads of up to four consecutive locations in 8-, 16-, or 32-bit wide external memories. This feature supports Burst mode ROM devices and increases the available system bandwidth by using a reduced (configurable) access time for three sequential reads following a quad-location boundary read.

5.1.4.1 Static Memory Write Control

Write timing for SRAM begins with assertion of the appropriate SMC memory bank Chip Select signal (nCS0:nCS6) and the external address bus signals, A[25:0]. The write access time is determined by the number of wait states programmed into SMCBCRx:WST2.

5.1.4.2 SMC Chip Select Behavior

Each of the SMC Chip Select signals (nCS0:nCS6) will be active for accesses within a specific range of physical addresses, listed in Table 5-1 and illustrated in Figure 5-2.

Deassertion of each SMC Chip Select can vary by access type (single or load/store multiple). See the LH79520 Data Sheet for SMC memory access timing diagrams.

5.1.4.3 Byte Lane Control

Each bank of external memory must be configured according to the way in which the actual hardware is wired. The wiring depends upon the actual width of each device (8, 16, or 32 bits), and the desired width of the bank. Logic within the SMC can then drive the Chip Selects (nCS0:nCS6) and the byte lane enables (nBLE0:nBLE3) appropriately for each bank, assuming that the SMC bank configuration registers (SMCBCR0:SMCBCR6) are correctly programmed. The operation of the byte lane enable signals depend upon:

- The width to which the memory bank is configured (programmed)
- The desired width of the transfer (an 8-, 16-, or 32-bit memory access)

External memory may be composed of either 8-bit devices, or devices partitioned into multiples of a byte. The type of memory devices used for each bank determine how the interface signals must be connected in order to provide byte, half-word or word-wide accesses. The write-enable signal nWE may or may not be required. For more information, see Section 5.1.4.9.

The nCSx and nBLEx signals for each memory bank are controlled by SMCBCRx:RBLE. The RBLE bit field in each SMCBCRx register also controls the behavior of the Write-Enable (nWE) when that Chip Select is asserted. If RBLE = 0 then the nWE signal will not be asserted during a write access involving that Chip Select.

The programmed width of the external memory system determines how data is transferred between the internal (AHB) and external buses.

5.1.4.4 Memory Banks Constructed with 8-bit or Non Byte-partitioned Devices

Figure 5-4 shows 8-bit wide memory devices connected to the LH79520. SMCBCRx:RBLE is programmed to '0' to ensure that the memory is not write-enabled during reads. Signal nWE is unused because it would provide insufficient granularity during write accesses. Signal nWE permits write access to the word-level but does not permit write access of only one particular byte within a word.

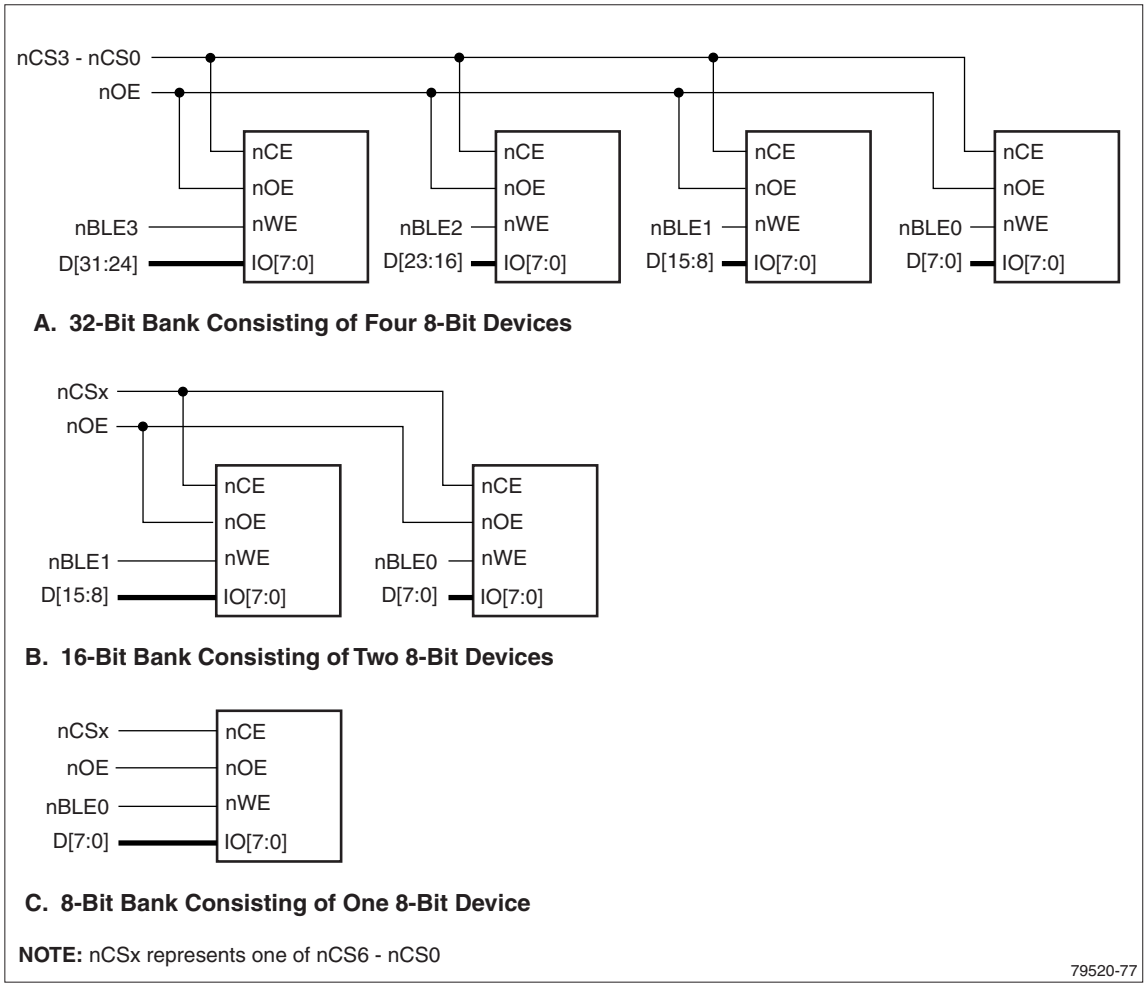
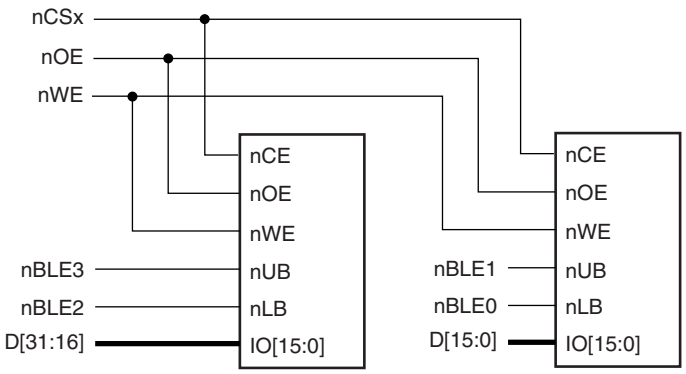


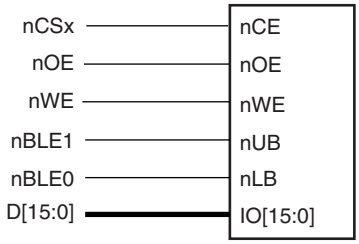
Figure 5-4. Memory Banks Constructed from 8-bit Memory

5.1.4.5 Memory Banks Constructed With 16 or 32-bit Memory devices

Figure 5-5 and Figure 5-6 show 16- and 32-bit wide memory devices connected to the LH79520. Signals nWE and nBLEx are utilized, and SMCBCRx:RBLE in the appropriate configuration registers is programmed to '1'.



A. 32-Bit Bank Consisting of Two 16-Bit Devices

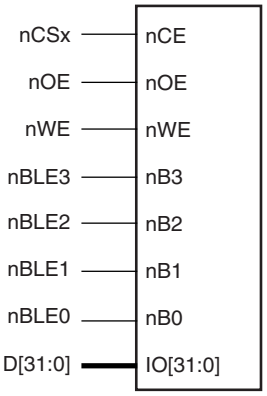


B. 16-Bit Bank Consisting of One 16-Bit Device

NOTE: nCSx represents one of nCS6 - nCS0.

79520-78

Figure 5-5. Memory Banks Constructed from 16-bit Memory



NOTE: nCSx represents one of nCS6 - nCS0

79520-79

Figure 5-6. Memory Banks Constructed from 32-bit Memory

5.1.4.6 Memory Banks Constructed with Mixed-width Memory Devices

Figure 5-7 shows a mixture of memory devices connected to the LH79520. Bank0 is a read-only device, so SMCBCR0:RBLE is moot. Bank1 devices require nWE and nBLEx signals; software should program SMCBCR1:RBLE = 1. Bank2 consists of byte-wide devices, so software should program SMCBCR2:RBLE = 0.

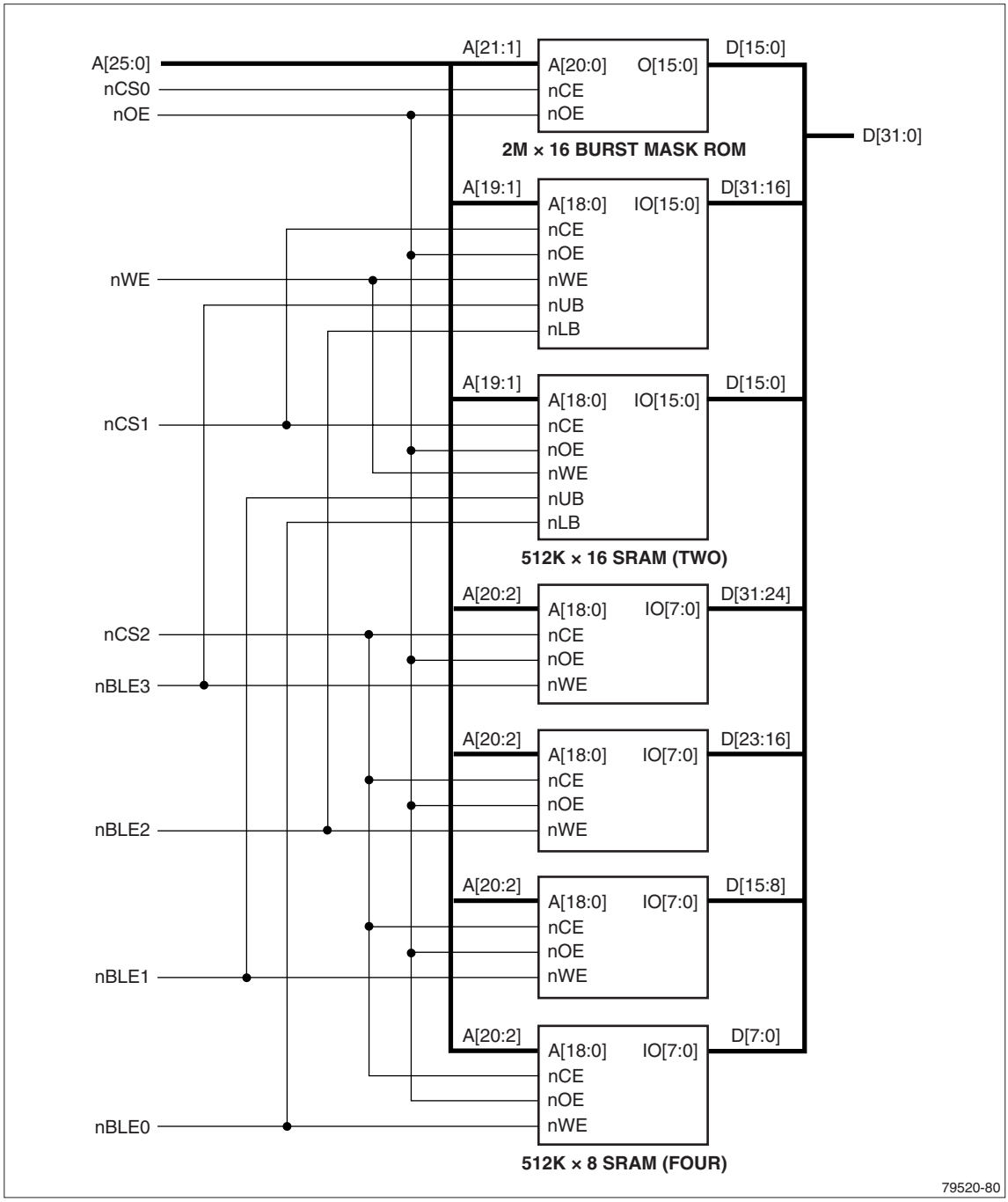


Figure 5-7. Typical Memory Connection Diagram

5.1.4.7 Selecting a Compatible Asynchronous Flash Memory Device

The LH79520 Asynchronous Memory Interface Signals AC Timing does not guarantee that the nCS[6:0] will become active prior to nBLE[3:0] or the nWE signals. Therefore, only an asynchronous Flash memory device that allows either nCE or nWE to become active first during a write cycle should be selected for use with the LH79520 MCU. One of the LH79520 nCSx outputs is typically connected to the Flash memory device nCE input. The LH79520 nWE output is typically connected to the Flash memory device nWE input.

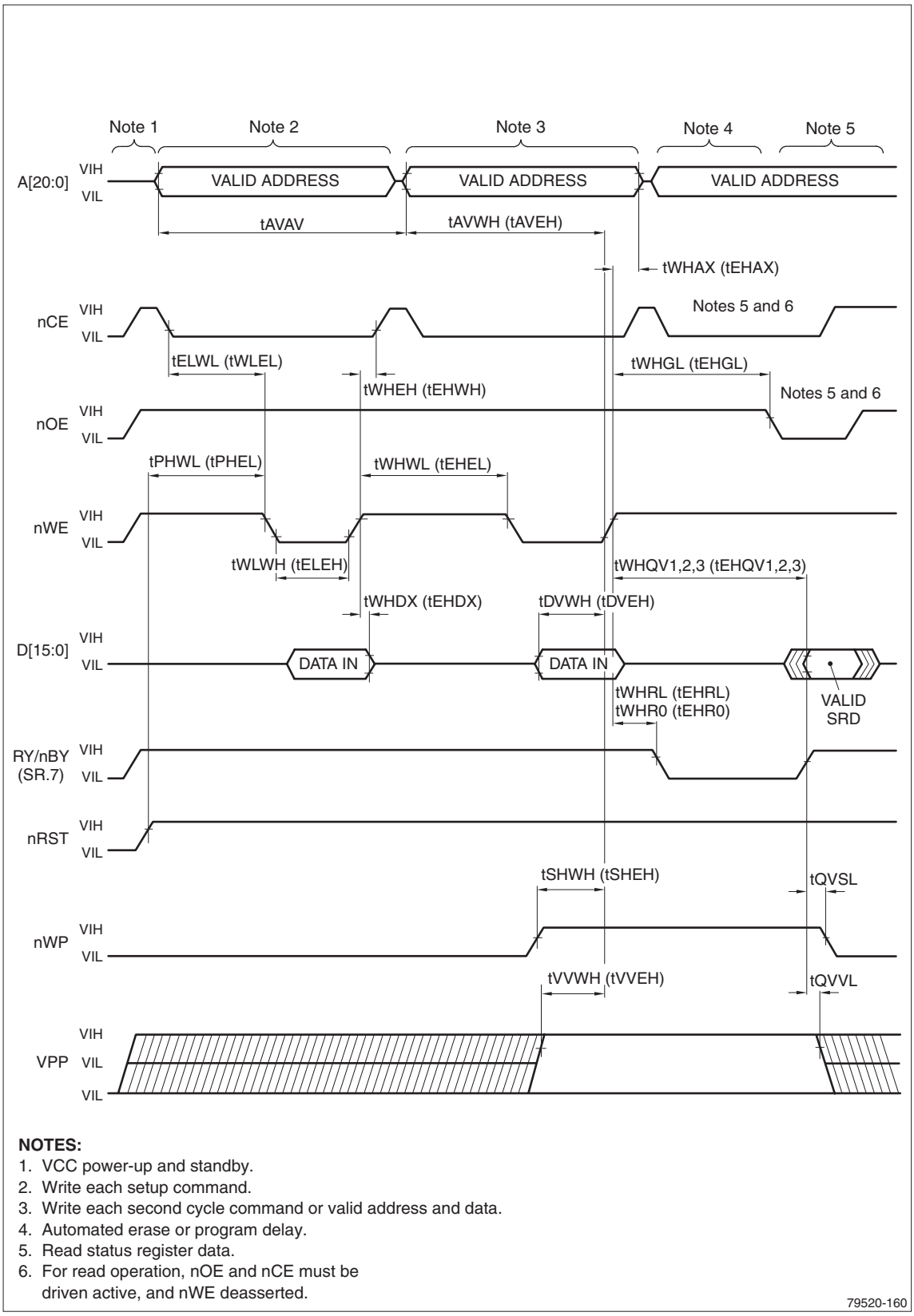
As an example of a compatible Flash device, refer to Table 5-2 and Figure 5-8, the NXP LHF32F11 Flash Memory device Write Operation AC Timing parameters and diagram. Notice the characteristic tELWL (tWLEL) is listed as a minimum of 0 ns, and that Note 4 states that no restrictions are placed on the timing relationship between nCE and nWE. The write cycle begins when both nCE and nWE are active and ends when either signal goes inactive.

Table 5-2. NXP LHF32F11 Write Parameters

SYMBOL	PARAMETER	MIN.	MAX.	UNIT	NOTES
tAVAV	Write Cycle Time	80		ns	
tPHWL (tPHEL)	nRST HIGH Recovery to nWE (nCE) Going LOW	150		ns	3
tELWL (tWLEL)	nCE (nWE) Setup to nWE (nCE) Going LOW	0		ns	4
tWLWH (tELEH)	nWE (nCE) Pulse Width	60		ns	4
tDVWH (tDVEH)	Data Setup to nWE (nCE) Going HIGH	40		ns	8
tAVWH (tAVEH)	Address Setup to nWE (nCE) Going HIGH	50		ns	8
tWHEH (tEHWH)	nCE (nWE) Hold from nWE (nCE) HIGH	0		ns	
tWHDX (tEHDX)	Data Hold from nWE (nCE) HIGH	0		ns	
tWHAX (tEHAX)	Address Hold from nWE (nCE) HIGH	0		ns	
tWHWL (tEHEL)	nWE (nCE) Pulse Width HIGH	30		ns	5
tSHWH (tSHEH)	nWP High Setup to nWE (nCE) Going HIGH	0		ns	3
tVVWH (tVVEH)	VPP Setup to nWE (nCE) Going HIGH	200		ns	3
tWHGL (tEHGL)	Write Recovery before Read	30		ns	
tQVSL	nWP HIGH Hold from Valid SRD, RY/nBY High-Z	0		ns	3, 6
tQVVL	VPP Hold from Valid SRD, RY/nBY High-Z	0		ns	3, 6
tWHR0 (tEHR0)	nWE (nCE) HIGH to SR.7 Going 0		tAVQV+50	ns	3, 7
tWHRL (tEHRL)	nWE (nCE) HIGH to RY/nBY Going LOW		100	ns	3

NOTES:

1. The timing characteristics for reading the status register during block erase, full chip erase, (page buffer) program and OTP program operations are the same as during read-only operations.
2. A write operation can be initiated and terminated with either nCE or nWE.
3. Sampled, not 100% tested.
4. Write pulse width (tWP) is defined from the falling edge of nCE or nWE (whichever goes low last) to the rising edge of nCE or nWE (whichever goes high first). Hence, tWP = tWLWH = tELEH = tWLEH = tELWH.
5. Write pulse width HIGH (tWPH) is defined from the rising edge of nCE or nWE (whichever goes high first) to the falling edge of nCE or nWE (whichever goes LOW last). Hence, tWPH = tWHWL = tEHEL = tWHEL = tEHWL.
6. VPP should be held at VPP=VPPH1/2 until determination of block erase, (page buffer) program or OTP program success (SR.1/3/4/5 = 0) and held at VPP = VPPH1 until determination of full chip erase success (SR.1/3/5 = 0).
7. tWHR0 (tEHR0) after the Read Query or Read Identifier Codes/OTP command = tAVQV+100 ns.



NOTES:
1. VCC power-up and standby.
2. Write each setup command.
3. Write each second cycle command or valid address and data.
4. Automated erase or program delay.
5. Read status register data.
6. For read operation, nOE and nCE must be driven active, and nWE deasserted.

79520-160

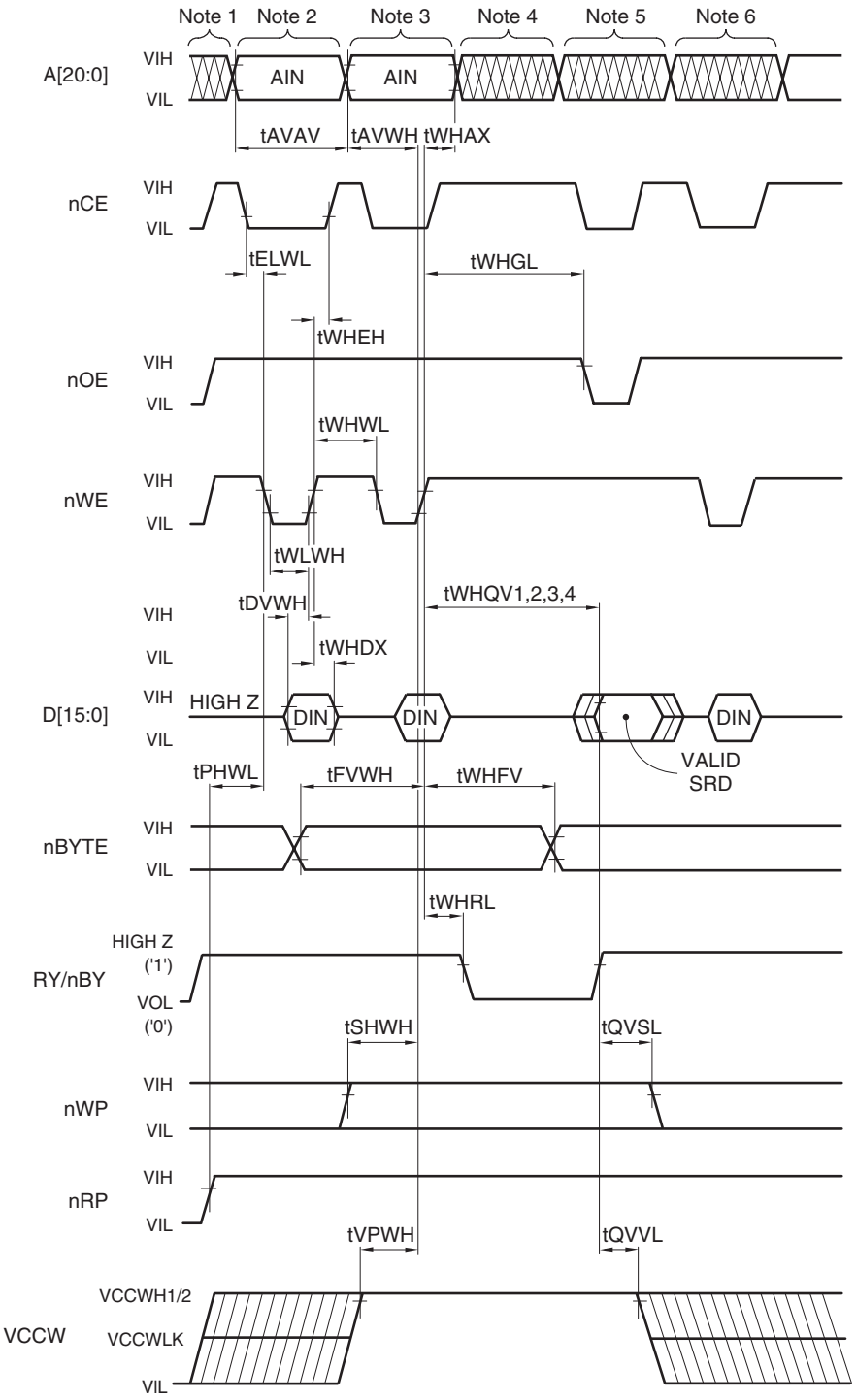
Figure 5-8. NXP LHF32F11 Write AC Timing

As an example of an incompatible Flash device, refer to Table 5-3 and Figure 5-9, the NXP LHF32J06 Flash Memory device Write Operation AC Timing Parameters and diagram. Notice the characteristic tELWL is listed as a minimum of 10 ns, requiring nCE to become active prior to nWE.

Table 5-3. NXP LHF32J06 Write Parameters

SYMBOL	PARAMETER	MIN.	MAX.	UNIT	NOTES
tAVAV	Write Cycle Time	90		ns	
tPHWL	nRP HIGH Recovery to nWE Going LOW	1		µs	2
tELWL	nCE (nWE) Setup to nWE (nCE) Going LOW	10		ns	
tWLWH	nWE Pulse Width	50		ns	
tSHWH	nWP VIH Setup to nWE Going HIGH	100		ns	2
tVPWH	VCCW Setup to nWE Going HIGH	100		ns	2
tAVWH	Address Setup to nWE Going HIGH	50		ns	3
tDVWH	Data Setup to nWE Going HIGH	50		ns	3
tWHDX	Data Hold from nWE HIGH	0		ns	
tWHAX	Address Hold from nWE HIGH	0		ns	
tWHEH	nCE Hold from nWE HIGH	10		ns	
tWHWL	nWE Pulse Width HIGH	30		ns	
tWHRL	nWE High to RY/nBY Going LOW or SR.7 Going "0"		100	ns	
tWHGL	Write Recovery before Read	0		ns	
tOVVL	VCCW Hold from Valid SRD, RY/nBY High-Z	0		ns	2, 4
tOVSL	nWP VIH Hold from Valid SRD, RY/nBY High-Z	0		ns	2, 4
tFVWH	nBYTE Setup to nWE Going HIGH	50		ns	5
tWHFV	nBYTE Hold from nWE HIGH	90		ns	5

NOTES:
1. Read timing characteristics during block erase, full chip erase, word/byte write and lock bit configuration operaions are the same as druing read-only operations.
2. Sampled, not 100% tested.
3. Refer to the Data Sheet for valid AIN and DIN for block erase, full chip erase, word/byte wrtie or lock bit configuration.
4. VCCW should be held at VDDWH1/2 until the chip indicates successful completion of the erase, write, or lock bit configuration cycle (SR. 1/3/4/5 = 0).
5. If nBYTE is switched during the Read cycle, then these timing characteristics will be different.



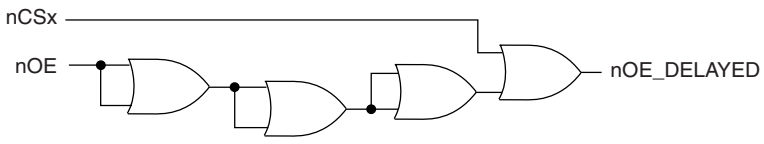
NOTES:
1. VCC power-up and standby.
2. Write each setup command.
3. Write each confirm command or valid address and data.
4. Automated erase or program delay.
5. Read status register data.
6. Write Read Array command.

79520-159

Figure 5-9. NXP LHF32J06 Write AC Timing

5.1.4.8 Using nCSx to Select an External ISA I/O Device

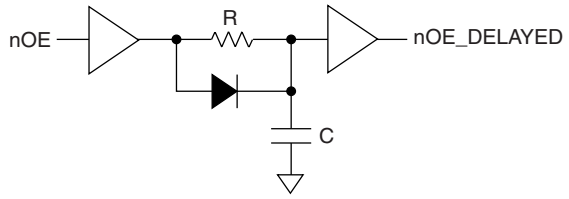
If your external I/O device has timing requirements on nCSx versus the falling edge of nOE or nWE (for example, to latch the Address and Chip Select status on the falling edge of a data strobe), then you must take care to make sure that your device gates nOE and nWE with nCSx. Furthermore, you must make sure your gate circuit provides enough setup time for your address latches to work. See Figure 5-10 through Figure 5-12. These circuits will allow you to add delay. In Figure 5-10 and Figure 5-12, more gates may be added as needed. The circuit in Figure 5-12 also allows for the adjustment of CLKOUT to meet timing requirements.



NOTE: This solution is sensitive to gate delay variation.

79520-162

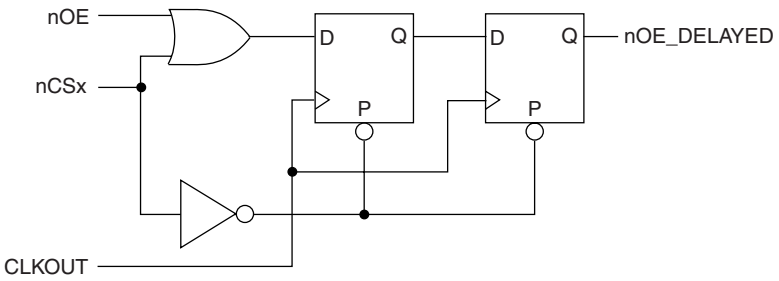
Figure 5-10. Delay Using Gates



NOTE: Adjust R&C to meet nOE timing requirements.
The delay is controlled by RC only if RC> gate delay.

79520-163

Figure 5-11. Delay Using RC Time Constant



NOTE: This design requires CLKOUT to be set to match the bus clock frequency.
The setup time will depend on the current bus clock.

79520-164

Figure 5-12. Delay Using Flip-flops

5.1.4.9 Using BLE0-3 Instead of nWE

If the external memory to be connected to the LH79520 is only one x8 memory device and there are no other memory devices connected to that same bank, the user may optionally connect nBLE0 or nWE to the external nWE pin. Referring to Table 5-4 and Table 5-5, if using nBLE0, RBLE must be set to 0. If using nWE, set RBLE = 1.

If the external memory consists of two or more x8 devices, nBLE0-3 must be connected to external devices using the nWE pin. Referring to Table 5-4 and Table 5-5, RBLE must be set to 0.

If the external memory is one x16 device, nWE must be connected to the external memory's nWE pin, nBLE0 must be connected to the memory's LB (low byte) pin, and nBLE1 must be connected to the memory's HB (high byte) pin. Referring to Table 5-4 and Table 5-5, set RBLE = 1.

If the external memory is one x32 device, nWE must be connected to the external memory's nWE pin and nBLE0-3 must be connected to the memory's byte strobe pin. Referring to Table 5-4 and Table 5-5, set RBLE = 1.

Table 5-4. RBLE = 0

ACCESS	nOE	nWE	nBLE0-3
Write	Inactive	Inactive	Active; acts as write strobe
Read	Active	Inactive	Inactive

Table 5-5. RBLE = 1

ACCESS	nOE	nWE	nBLE0-3
Write	Inactive	Active	Active
Read	Active	Inactive	Active

5.1.5 Eliminating Floating Bytes on the External Interface

The SMC uses the programmed width of each bank (SMCBCRx:MW) to determine how many bytes of the external bus it must drive in order to ensure that the external bus is never floating. For example, a byte write to a bank of external memory which is programmed to be 16 bits wide will drive only the lower 16 external interface lines. The pins for unused lines should be tied either HIGH or LOW.

5.1.6 Byte Lane Control and Steering

Table 5-6 through Table 5-11 show the relationships of the signals that map the data between the external memory data bus and the AHB.

Table 5-6. 8-bit External Bus, Read Operation

REQUESTED TRANSFER WIDTH	QUANTITY OF TRANSFERS REQUIRED	AHB ADDRESS BITS	EXTERNAL ADDRESS BITS	EXTERNAL DATA MAPPING ONTO AHB			
		AHB[1:0]	A[1:0]	31:24	23:16	15:8	7:0
Word	4	xx	11	7:0			
			10		7:0		
			01			7:0	
			00				7:0
Half-word	2	1x	11	7:0			
			10		7:0		
Half-word	2	0x	01			7:0	
			00				7:0
Byte	1	11	11	7:0			
Byte	1	10	10		7:0		
Byte	1	01	01			7:0	
Byte	1	00	00				7:0

Table 5-7. 16-bit External Bus, Read Operation

REQUESTED TRANSFER WIDTH	QUANTITY OF TRANSFERS REQUIRED	AHB ADDRESS BITS	EXTERNAL ADDRESS BITS	EXTERNAL DATA MAPPING ONTO AHB			
		AHB[1:0]	A[1:0]	31:24	23:16	15:8	7:0
Word	2	xx	1x	15:8	7:0		
			0x			15:8	7:0
Half-word	1	1x	1x	15:8	7:0		
Half-word	1	0x	0x			15:8	7:0
Byte	1	11	1x	15:8			
Byte	1	10	1x		7:0		
Byte	1	01	0x			15:8	
Byte	1	00	0x				7:0

Table 5-8. 32-bit External Bus, Read Operation

REQUESTED TRANSFER WIDTH	QUANTITY OF TRANSFERS REQUIRED	AHB ADDRESS BITS	EXTERNAL ADDRESS BITS	EXTERNAL DATA MAPPING ONTO AHB			
		AHB[1:0]	A[1:0]	31:24	23:16	15:8	7:0
Word	1	xx	xx	31:24	23:16	15:8	7:0
Half-word	1	1x	xx	31:24	23:16	15:8	7:0
Half-word	1	0x	xx	31:24	23:16	15:8	7:0
Byte	1	11	xx	31:24	23:16	15:8	7:0
Byte	1	10	xx	31:24	23:16	15:8	7:0
Byte	1	01	xx	31:24	23:16	15:8	7:0
Byte	1	00	xx	31:24	23:16	15:8	7:0

Table 5-9. 8-bit External Bus, Write Operation

REQUESTED TRANSFER WIDTH	QUANTITY OF TRANSFERS REQUIRED	AHB ADDRESS BITS	EXTERNAL ADDRESS BITS	nBLE [3:0]	AHB DATA MAPPING ONTO EXTERNAL DATA BUS			
		AHB[1:0]	A[1:0]		31:24	23:16	15:8	7:0
Word	4	xx	11	1110				31:24
			10	1110				23:16
			01	1110				15:8
			00	1110				7:0
Half-word	2	1x	11	1110				31:24
			10	1110				23:16
Half-word	2	0x	01	1110				15:8
			00	1110				7:0
Byte	1	11	11	1110				31:24
Byte	1	10	10	1110				23:16
Byte	1	01	01	1110				15:8
Byte	1	00	00	1110				7:0

Table 5-10. 16-bit External Bus, Write Operation

INTERNAL TRANSFER WIDTH	QUANTITY OF TRANSFERS REQUIRED	AHB ADDRESS BITS	EXTERNAL ADDRESS BITS	nBLE [3:0]	AHB DATA MAPPING ONTO EXTERNAL DATA BUS			
		AHB[1:0]	A[1:0]		31:24	23:16	15:8	7:0
Word	2	xx	1x	1100			31:24	23:16
			0x	1100			15:8	7:0
Half-word	1	1x	1x	1100				
Half-word	1	0x	0x	1100			15:8	7:0
Byte	1	11	1x	1101			31:24	
Byte	1	10	1x	1110				23:16
Byte	1	01	0x	1101			15:8	
Byte	1	00	0x	1110				7:0

Table 5-11. 32-bit External Bus, Write Operation

INTERNAL TRANSFER WIDTH	QUANTITY OF TRANSFERS REQUIRED	AHB ADDRESS BITS	EXTERNAL ADDRESS BITS	nBLE [3:0]	AHB DATA MAPPING ONTO EXTERNAL DATA BUS			
		AHB[1:0]	A[1:0]		31:24	23:16	15:8	7:0
Word	1	xx	xx	0000	31:24	23:16	15:8	7:0
Half-word	1	1x	xx	0011	31:24	23:16		
Half-word	1	0x	xx	1100			15:8	7:0
Byte	1	11	xx	0111	31:24			
Byte	1	10	xx	1011		23:16		
Byte	1	01	xx	1101			15:8	
Byte	1	00	xx	1110				7:0

5.2 SMC Programmer's Model

The Base Address for the registers that configure the Static Memory Controller (SMC) is:

SMCRegBase: 0xFFFF1000

All registers in the SMC will accept word, half-word, and byte accesses.

5.3 SMC Configuration Register Summary

Table 5-12 summarizes the programmable registers that configure the LH79520 Static Memory Controller (SMC). Each register is treated in more detail on the following pages.

Each SMC Configuration register affects the behavior of a different Chip Select signal (nCS0 through nCS6), as shown in Table 5-13.

Table 5-12. SMC Configuration Register Summary

ADDRESS	REGISTER NAME	DESCRIPTION
SMCRegBase + 0x000	SMCBCR0	Static Memory Bank 0 Configuration Register
SMCRegBase + 0x004	SMCBCR1	Static Memory Bank 1 Configuration Register
SMCRegBase + 0x008	SMCBCR2	Static Memory Bank 2 Configuration Register
SMCRegBase + 0x00C	SMCBCR3	Static Memory Bank 3 Configuration Register
SMCRegBase + 0x010	SMCBCR4	Static Memory Bank 4 Configuration Register
SMCRegBase + 0x014	SMCBCR5	Static Memory Bank 5 Configuration Register
SMCRegBase + 0x018	SMCBCR6	Static Memory Bank 6 Configuration Register

Table 5-13. SMC Registers and Associated Chip Selects

REGISTER NAME	CHIP SELECT
SMCBCR0	nCS0
SMCBCR1	nCS1
SMCBCR2	nCS2
SMCBCR3	nCS3
SMCBCR4	nCS4
SMCBCR5	nCS5
SMCBCR6	nCS6

5.4 SMC Configuration Register Descriptions

This section of this User's Guide lists and describes each of the SMC programmable configuration registers.

5.4.1 SMC Bank Configuration Register 0

The SMCBCR0 register must be programmed to describe the manner in which the external static memory accessed by Chip Select 0 (nCS0) is to function.

The bit fields for registers SMCBCR1 through SMCBCR6 are identical to the bit fields shown in Table 5-14, except that those registers have different offsets from the base address (SMCRegBase), and those registers affect different Chip Selects (nCS1:nCS6). See Table 5-12 for a summary of the address offsets, and Table 5-13 for the Chip Selects.

Table 5-15 explains the bit fields in registers SMCBCR0 through SMCBCR6.

Table 5-14. SMCBCR0 Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///		MW		BM	WP	WPERR	BUSERR	///							
RESET	0	0	(See Note)		0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW		RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	WST2					RBLE	WST1					///	IDCY			
RESET	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	SMCRegBase + 0x000															

NOTE: The MW bit field in each of the registers SMCBCR0 through SMCBCR6 defaults to a different value at reset because the various banks of external memory have different widths at reset. See Table 5-1 for a list of the bank-widths at reset.

Table 5-15. SMCBCR0 Register Bit Fields

BITS	FIELD NAME	FUNCTION	NOTES
31:30	///	**Reserved** Write the reset value.	
29:28	MW	**External Static Memory Width** 0b00 = 8-bit 0b01 = 16-bit 0b10 = 32-bit 0b11 = Reserved — do not write	1
27	BM	**Burst Mode** 0 = Non-burst devices (reset) 1 = Burst ROM	
26	WP	**Write Protect** 0 = Not write-protected (reset) 1 = Write-protected	
25	WPERR	**Write Protect Error Status Flag** When read: 0 = no error (reset) 1 = write protect error Writing a '1' to this bit will clear the Write Protect Error Status Flag.	
24	BUSERR	**Bus Transfer Error Status Flag** When read: 0 = No error (reset). 1 = Bus transfer error. Writing a '1' to this bit will clear the bus transfer error status flag.	
23:16	///	**Reserved** Write the reset value.	
15:11	WST2	**Wait State2** See Section 5.1.3.3. This is the write access time for SRAM, the burst access time for burst ROM, and does not apply to ROM devices. This wait state time is: (WST2 + 1) × tHCLK in the case of SRAM, or (WST2) × tHCLK in the case of burst ROM. 0b11111 = (reset)	2
10	RBLE	**Read Byte Lane Enable** 0 = All byte lane strobes nBLE[3:0] are held HIGH during system reads from memory (reset). The nBLE[3:0] signals are used as write-enables. The nWE signal will be inactive. 1 = All byte lane strobes nBLE[3:0] are held LOW during system reads from memory. The nWE signal is used as a write-enable.	3
9:5	WST1	**Wait State1** See Section 5.1.3.3. This is the read access time for SRAM and ROM, or the initial access time for burst ROM. This wait state time is (WST1 + 1) × tHCLK.	
4	///	**Reserved** Write the reset value.	
3:0	IDCY	Idle cycle memory data bus turn-around time. The turn around time is (IDCY + 1) × (the Period of the AHB clock).	

NOTES:
1. The MW bit field in each of the registers SMCBCR0 through SMCBCR6 defaults to a different value at reset because the various banks of external memory have different widths at reset. See Table 5-1 for a list of the bank-widths at reset.
2. tHCLK = the period of the AHB clock = 1 ÷ (Frequency of the AHB clock).
3. Software should write RBLE = 0 when interfacing to external 8-bit or non byte-partitioned memory devices; Software should write RBLE = 1 when using the byte-lane control signals nBLE[0:3] to interface a bank of external 16- or 32-bit memory devices to the LH79520.

5.4.2 SMC Bank Configuration Register 1

Register: SMCBCR1
Address = **SMCRegBase** + 0x004

5.4.3 SMC Bank Configuration Register 2

Register: SMCBCR2
Address = **SMCRegBase** + 0x008

5.4.4 SMC Bank Configuration Register 3

Register: SMCBCR3
Address = **SMCRegBase** + 0x00C

5.4.5 SMC Bank Configuration Register 4

Register: SMCBCR4
Address = **SMCRegBase** + 0x010

5.4.6 SMC Bank Configuration Register 5

Register: SMCBCR5
Address = **SMCRegBase** + 0x014

5.4.7 SMC Bank Configuration Register 6

Register: SMCBCR6
Address = **SMCRegBase** + 0x018

Configuration registers SMCBCR1 through SMCBCR6 are identical to register SMCBCR0, except that each register affects a different chip select signal (see Table 5-13) and each register has a different offset from the SMC base address (see Table 5-12).

Each of these registers must be programmed to describe the manner in which the external static memory bank accessed by the associated Chip Select signal is to function.

See Table 5-15 for a description of the bit fields in any of these registers. Note that the value for the MW bit field in each register can vary. The MW bit field in each of these registers describes the 'width' of the bank of external memory associated with the register's Chip Select signal. The LH79520 is designed to automatically, at reset, assign a certain width to each bank of external memory, as shown in Figure 5-2 and Table 5-1. This automatic assignment causes a variation in the reset value of the MW field in each of these registers. Following each reset, the MW bit field must be programmed appropriately for the actual memory devices in use.

Chapter 6
SDRAM Memory Controller (SDRC)

This chapter of this User's Guide presents details of the Synchronous Dynamic RAM Memory Controller (SDRC) in the LH79520 Universal MCU. A discussion of its Theory of Operation is followed by descriptions of the internal registers mentioned in the discussion.

6.1 Theory of Operation

The LH79520 includes an SDRAM Controller (SDRC). SDRAM memory technology is very different from the simpler memory technologies interfaced by the LH79520 Static Memory Controller (SMC). Figure 6-1, an expanded portion of the 'LH79520 Block Diagram', shows how the SDRC is integrated into the LH79520 MCU.

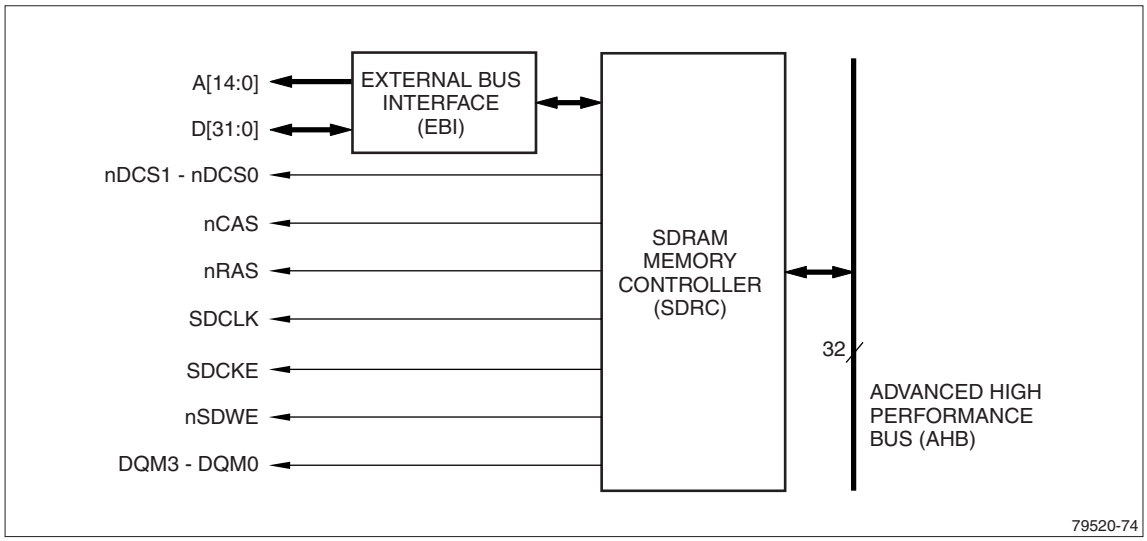


Figure 6-1. SDRAM Memory Controller Block Diagram

The LH79520 has one external address bus, shared by the Static Memory Controller (SMC) and the SDRAM Memory Controller (SDRC). This sharing is automatically coordinated by an External Bus Interface (EBI) shown in Figure 6-1.

The SDRC provides an interface between the AHB and external (off-chip) SDRAM memory devices. The SDRC provides all necessary signals to interface the LH79520 to SDRAM devices and includes a Read Buffer and a Merging Write buffer. The Read Buffer can be disabled in software.

The SDRC can accommodate two banks of external SDRAM memory, each up to 128M in size. The LH79520 SDRC provides separate SDRAM Chip Select signals (nDCS0:nDCS1) for each range of addresses accessed by the SDRC.

This chapter explains how the two SDRC Chip Select signals (nDCS0:nDCS1) can be used, along with other LH79520 memory interface signals, to access multiple SDRAM devices. Although this chapter describes the memory addressed by each of the LH79520 SDRC Chip Select signals (nDCS0:nDCS1) as a separate 'bank' of SDRAM memory, readers should not confuse these banks with the 'banks' of memory internal to SDRAM devices. Each SDRAM device may contain multiple banks of SDRAM memory, as specified by the SDRAM device manufacturer.

Figure 6-2 shows how the SDRC Chip Select signals (nDCS0:nCDS1) correspond to the LH79520 memory map. Figure 6-2 also shows a region of external SDRAM memory extending from 0x30000000 to 0x3FFFFFFF and marked as 'Reserved'. Software should not access this region because, although no memory exists there, a data abort will not be generated.

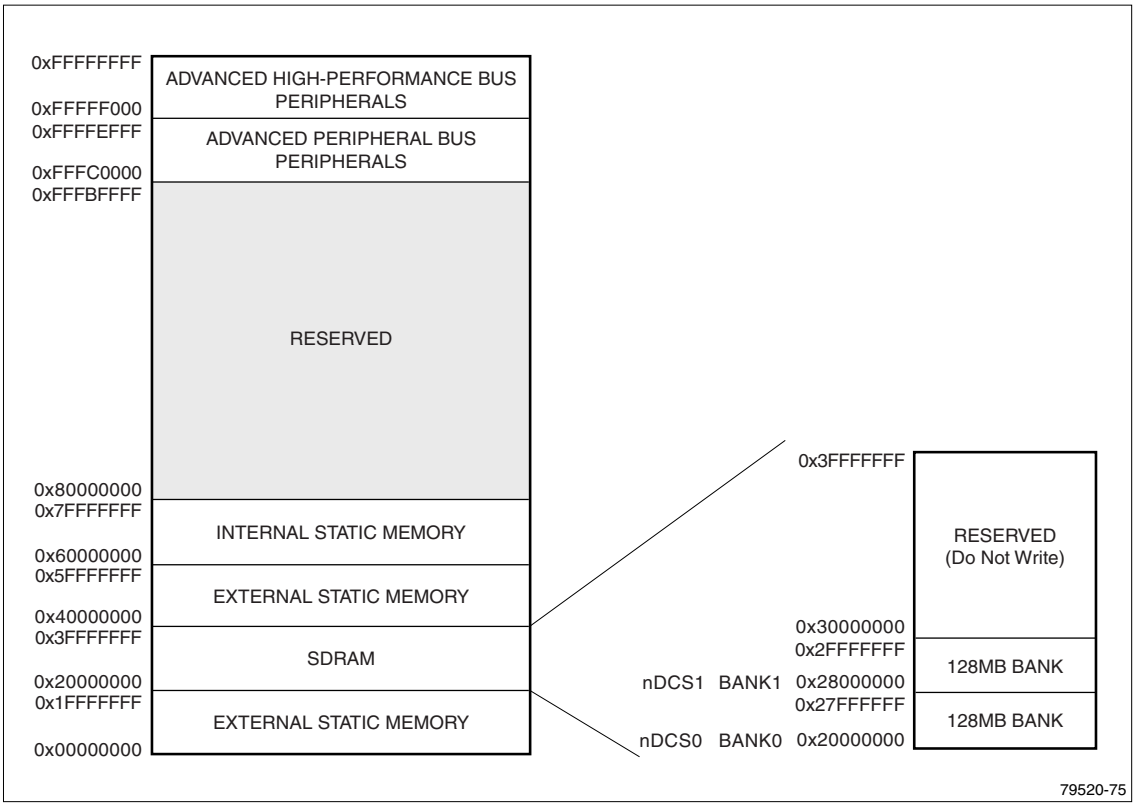


Figure 6-2. External SDRAM Memory Banks

Table 6-1 lists the SDRAM Chip Select signals (nDCS0:nDCS1) provided by the SDRC, and shows the width of the SDRAM memory accessed by both signals at reset.

Table 6-1. SDRAM Memory Bank Addresses and Chip Selects

ADDRESS RANGE	SDRC CHIP SELECT	DESCRIPTION	WIDTH AT RESET
0x30000000 - 0x3FFFFFFF	///	Reserved	
0x28000000 - 0x2FFFFFFF	nDCS1	SDRAM Chip Select 1	32-bit
0x20000000 - 0x27FFFFFF	nDCS0	SDRAM Chip Select 0	32-bit

The operation of the SDRC is governed by four programmable registers explained in this Chapter of this User's Guide. For information about each of these registers, see Section 6.4.1, Section 6.4.2, Section 6.4.3, and Section 6.4.4.

When the SDRC is controlling the external address bus, it uses the bus differently than the SMC because SDRAM devices require multiplexed address signals. Multiplexing is required because SDRAM devices use row/column/bank addressing schemes.

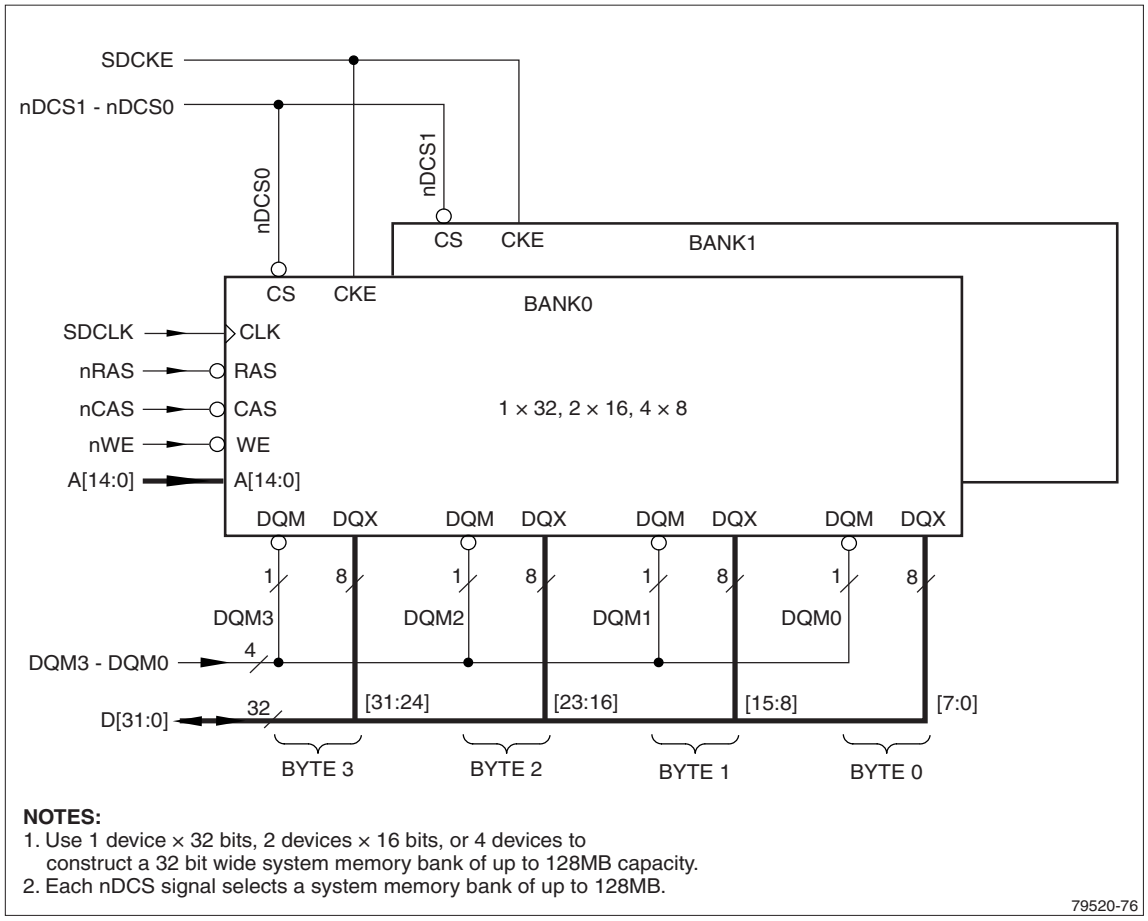
The SDRC Chip Select signals (nDCS0:nDCS1) operate according to the addressing shown in Figure 6-2, but the actual addresses placed on the external address bus during SDRAM accesses are multiplexed (encoded) by the SDRAM Controller (SDRC) for the particular SDRAM memory devices in use. The exact manner in which the addresses are multiplexed depends upon the SDRC Configuration register programming.

The encoding for various SDRAM devices is presented in Tables 6-3 through 6-18. See Section 6.4.1 and Section 6.4.2 for more information.

Designers will want to ensure that the SDRAM devices are connected according to information presented here and also in the data sheet supplied by the SDRAM device manufacturer. Programmers must ensure that the SDRCC is correctly configured according to information presented in Tables 6-3 through 6-18.

6.1.1 SDRAM Device Interfacing

Figure 6-3 illustrates how the LH79520 external memory interface signals should connect to an external SDRAM memory array to form a 32-bit-wide external data bus. By comparison, a 16-bit-wide external SDRAM data bus would utilize only Bytes 0 and 1, and signals DQM3:DQM2, and D[31:16], would not be connected.



NOTES:
1. Use 1 device × 32 bits, 2 devices × 16 bits, or 4 devices to construct a 32 bit wide system memory bank of up to 128MB capacity.
2. Each nDCS signal selects a system memory bank of up to 128MB.

79520-76

Figure 6-3. SDRAM Device Interfacing

The next section presents a series of tables for use as a guide when implementing an external SDRAM memory interface to the LH79520 MCU.

6.1.2 Selecting SDRAM Devices

The LH79520 SDRAM Controller (SDRC) can accommodate a wide variety of SDRAM memory devices, listed in Table 6-2. The 'T', 'B', 'F' and 'X' headings in the table refer to bit fields in the SDRCConfig0 register detailed in Section 6.4.1.

For each SDRAM configuration listed in Table 6-2, Tables 6-3 through 6-18 show how addresses placed on the AHB are automatically multiplexed within the LH79520 before being presented to the SDRAM device connected to the external address lines.

To use Table 6-2, first locate the desired memory organization in Table 6-2, then refer to the secondary table listed in the Reference Table column of Table 6-2 for details concerning how the selected SDRAM memory devices should be connected.

Table 6-2. Summary of SDRAM Devices

TOTAL BITS	ORGANIZATION: ADDRESSABLE LOCATIONS × DATA BITS	T: MUXING 1 = ×8 0 = ×16, ×32	B: BANKS 1 = FOUR 0 = TWO	F: 256M 1 = YES 0 = NO	X: EXTERNAL BUS WIDTH 0 = 32 BITS 1 = 16 BITS	SIGNALS TO USE AS SDRAM DEVICE BANK SELECTS*	REFERENCE TABLE
16M	1M x 16	0	0	0	0	A[11]	Table 6-3
16M	2M x 8	1	0	0	0	A[11]	Table 6-4
64M	2M × 32	0	1	0	0	A[12], A[13]	Table 6-5
64M	4M × 16	0	1	0	0	A[12], A[13]	Table 6-5
64M	8M × 8	1	1	0	0	A[12], A[13]	Table 6-6
128M	16M × 8	1	1	0	0	A[12], A[13]	Table 6-7
128M	8M × 16	0	1	0	0	A[12], A[13]	Table 6-8
256M	16M × 16	0	1	1	0	A[13], A[14]	Table 6-9
256M	32M × 8	1	1	1	0	A[13], A[14]	Table 6-10
16M	1M × 16	0	0	0	1	A[11]	Table 6-11
16M	2M × 8	1	0	0	1	A[11]	Table 6-12
64M	2M × 32	0	1	0	1	A[12], A[13]	Table 6-13
64M	4M × 16	0	1	0	1	A[12], A[13]	Table 6-13
64M	8M × 8	1	1	0	1	A[12], A[13]	Table 6-14
128M	16M × 8	1	1	0	1	A[12], A[13]	Table 6-15
128M	8M × 16	0	1	0	1	A[12], A[13]	Table 6-16
256M	16M × 16	0	1	1	1	A[13], A[14]	Table 6-17
256M	32M × 8	1	1	1	1	A[13], A[14]	Table 6-18

NOTE: *SDRAM devices have 'bank select' inputs which choose among the banks of memory within the SDRAM device. Entries in this column concern these SDRAM bank-select inputs. Avoid confusing these SDRAM inputs with the 'banks' of SDRAM memory selected by the signals nDCS0 and nDSC1.

For example, to utilize a 16M SDRAM device which is organized as 2M addressable locations × 8 data bits, Table 6-2 refers the reader to Table 6-4. A review of Table 6-4 shows that AHB address bit A[8] within the LH79520 will be presented to the external SDRAM device on external address line A[20] during RAS (Row Address Strobe) time. Table 6-4 also shows that AHB address bit A[8] will be presented to the SDRAM device on external address line A[10] during CAS (Column Address Strobe) time.

Table 6-4 and its notes also show that:

- SDRCConfig0:T (the 'T' bit field in SDRCConfig0) should be programmed to 1.
- AHB Address bits A[14] and A[12] within the LH79520 are unused with an external SDRAM device of this size.
- AHB Address bit A[13] will appear on external address line A[11], is not used as an address, and should be connected to the SDRAM device's Bank Select input.
- AHB Address bit A[9] will appear on A[25] during CAS but is unused by the SDRAM device.
- AHB Address bit A[10] will be replaced by the value programmed to SDRCConfig0:AP during CAS time.

Designers and Programmers should also follow these constraints:

- Memory selected by either SDRAM Chip Select (nDCS1 or nDCS0) must be contiguous within the range of addresses over which the Chip Select is active.
- The bank address boundary should coincide with a SDRAM page address bounder.

6.1.3 SDRAM Device Selection Tables

Use Tables 6-3 through 6-18 to select and connect SDRAM devices to the LH79520. See Table 6-2 for additional information.

Table 6-3. 16M SDRAM (1M × 16)

EXT. ADDR LINE	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Row	**	10*	**	10	11	21	20	19	18	17	16	15	14	13	12
Col	**	10*	**	10	AP	25*	10*	9	8	7	6	5	4	3	2

NOTES:
1. * = Unused bit.
2. ** = Fixed level; in this case 0.
3. T = 0, B = 0, F = 0, X = 0
4. Use A[11] as Bank Select.
5. The 'AP' entry refers to SDRCConfig0:AP (the AP bit field in the SDRCConfig0 register).

Table 6-4. 16M SDRAM (2M × 8)

EXT. ADDR LINE	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Row	**	11*	**	11	22	21	20	19	18	17	16	15	14	13	12
Col	**	11*	**	11	AP	25*	10	9	8	7	6	5	4	3	2

NOTES:
1. * = Unused bit.
2. ** = Fixed level; in this case 0.
3. T = 1, B = 0, F = 0, X = 0
4. Use A[11] as Bank Select.
5. The 'AP' entry refers to SDRCConfig0:AP (the AP bit field in the SDRCConfig0 register).

Table 6-5. 64M SDRAM (2M × 32, 4M × 16)

EXT. ADDR LINE	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Row	11*	10	11	23	22	21	20	19	18	17	16	15	14	13	12
Col	11*	10	11	23*	AP	25*	24*	9	8	7	6	5	4	3	2

NOTES:
1. * = Unused bit.
2. T = 0, B = 1, F = 0, X = 0
3. Use A[12] and A[13] as Bank Selects.
4. The 'AP' entry refers to SDRCConfig0:AP (the AP bit field in the SDRCConfig0 register).

Table 6-6. 64M SDRAM (8M × 8)

EXT. ADDR LINE	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Row	11*	12	11	23	22	21	20	19	18	17	16	15	14	13	24
Col	11*	12	11	23*	AP	25*	10	9	8	7	6	5	4	3	2

NOTES:
1. * = Unused bit.
2. T = 1, B = 1, F = 0, X = 0
3. Use A[12] and A[13] as Bank Selects.
4. The 'AP' entry refers to SDRCConfig0:AP (the AP bit field in the SDRCConfig0 register).

Table 6-7. 128M SDRAM (16M × 8)

EXT. ADDR LINE	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Row	11*	12	11	23	22	21	20	19	18	17	16	15	14	13	24
Col	11*	12	11	23*	AP	25	10	9	8	7	6	5	4	3	2

NOTES:
1. * = Unused bit.
2. T = 1, B = 1, F = 0, X = 0
3. Use A[12] and A[13] as Bank Selects.
4. The 'AP' entry refers to SDRCConfig0:AP (the AP bit field in the SDRCConfig0 register).

Table 6-8. 128M SDRAM (8M × 16)

EXT. ADDR LINE	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Row	11*	10	11	23	22	21	20	19	18	17	16	15	14	13	24
Col	11*	10	11	23*	AP	25*	24	9	8	7	6	5	4	3	2

NOTES:
1. * = Unused bit.
2. T = 0, B = 1, F = 0, X = 0
3. Use A[12] and A[13] as Bank Selects.
4. The 'AP' entry refers to SDRCConfig0:AP (the AP bit field in the SDRCConfig0 register).

Table 6-9. 256M SDRAM (16M × 16)

EXT. ADDR LINE	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Row	11	12	24	23	22	21	20	19	18	17	16	15	14	13	25
Col	11	12	24*	23*	AP	26*	10	9	8	7	6	5	4	3	2

NOTES:
1. * = Unused bit.
2. T = 0, B = 1, F = 1, X = 0
3. Use A[13] and A[14] as Bank Selects.
4. The 'AP' entry refers to SDRCConfig0:AP (the AP bit field in the SDRCConfig0 register).

Table 6-10. 256M SDRAM (32M × 8)

EXT. ADDR LINE	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Row	11	12	24	23	22	21	20	19	18	17	16	15	14	13	25
Col	11	12	24*	23*	AP	26	10	9	8	7	6	5	4	3	2

NOTES:
1. * = Unused bit.
2. T = 1, B = 1, F = 1, X = 0
3. Use A[13] and A[14] as Bank Selects.
4. The 'AP' entry refers to SDRCConfig0:AP (the AP bit field in the SDRCConfig0 register).

Table 6-11. 16M SDRAM (1M × 16)

EXT. ADDR LINE	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Row	**	9*	**	9	11	10	20	19	18	17	16	15	14	13	12
Col	**	9*	**	9	AP	24*	9*	8	7	6	5	4	3	2	**

NOTES:
1. * = Unused bit.
2. ** = Fixed level; in this case 0.
3. T = 0, B = 0, F = 0, X = 1
4. Use A[11] as Bank Select.
5. The 'AP' entry refers to SDRCConfig0:AP (the AP bit field in the SDRCConfig0 register).

Table 6-12. 16M SDRAM (2M × 8)

EXT. ADDR LINE	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Row	**	10*	**	10	11	21	20	19	18	17	16	15	14	13	12
Col	**	10*	**	10	AP	24*	9	8	7	6	5	4	3	2	**

NOTES:
1. * = Unused bit.
2. ** = Fixed level; in this case 0.
3. T = 1, B = 0, F = 0, X = 1
4. Use A[11] as Bank Select.
5. The 'AP' entry refers to SDRCConfig0:AP (the AP bit field in the SDRCConfig0 register).

Table 6-13. 64M SDRAM (2M × 32, 4M × 16)

EXT. ADDR LINE	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Row	9*	10	9	22	11	21	20	19	18	17	16	15	14	13	12
Col	9*	10	9	22*	AP	24*	23*	8	7	6	5	4	3	2	**

NOTES:
1. * = Unused bit.
2. ** = Fixed level; in this case 0.
3. T = 0, B = 1, F = 0, X = 1
4. Use A[12] and A[13] as Bank Selects.
5. The 'AP' entry refers to SDRCConfig0:AP (the AP bit field in the SDRCConfig0 register).

Table 6-14. 64M SDRAM (8M × 8)

EXT. ADDR LINE	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Row	11*	10	11	23	22	21	20	19	18	17	16	15	14	13	12
Col	11*	10	11	23*	AP	24*	9	8	7	6	5	4	3	2	**

NOTES:
1. * = Unused bit.
2. ** = Fixed level; in this case 0.
3. T = 1, B = 1, F = 0, X = 1
4. Use A[12] and A[13] as Bank Selects.
5. The 'AP' entry refers to SDRCConfig0:AP (the AP bit field in the SDRCConfig0 register).

Table 6-15. 128M SDRAM (16M × 8)

EXT. ADDR LINE	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Row	11*	10	11	23	22	21	20	19	18	17	16	15	14	13	12
Col	11*	10	11	23*	AP	24	9	8	7	6	5	4	3	2	**

NOTES:
1. * = Unused bit.
2. ** = Fixed level; in this case 0.
3. T = 1, B = 1, F = 0, X = 1
4. Use A[12] and A[13] as Bank Selects.
5. The 'AP' entry refers to SDRCConfig0:AP (the AP bit field in the SDRCConfig0 register).

Table 6-16. 128M SDRAM (8M × 16)

EXT. ADDR LINE	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Row	9*	10	9	22	11	21	20	19	18	17	16	15	14	13	12
Col	9*	10	9	22*	AP	24*	23	8	7	6	5	4	3	2	***

NOTES:
1. * = Unused bit.
2. ** = Fixed level; in this case 0.
3. *** = internally-generated address.
4. T = 0, B = 1, F = 0, X = 1
5. Use A[12] and A[13] as Bank Selects.
6. The 'AP' entry refers to SDRCConfig0:AP (the AP bit field in the SDRCConfig0 register).
7. The LH79520 Evaluation Board uses SDRAM devices of this type.

Table 6-17. 256M SDRAM (16M × 16)

EXT. ADDR LINE	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Row	11	10	24	23	22	21	20	19	18	17	16	15	14	13	12
Col	11	10	24*	23*	AP	25*	9	8	7	6	5	4	3	2	**

NOTES:
1. * = Unused bit.
2. ** = Fixed level; in this case 0.
3. T = 0, B = 1, F = 1, X = 1
4. Use A[13] and A[14] as Bank Selects.
5. The 'AP' entry refers to SDRCConfig0:AP (the AP bit field in the SDRCConfig0 register).

Table 6-18. 256M SDRAM (32M × 8)

EXT. ADDR LINE	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Row	11	10	24	23	22	21	20	19	18	17	16	15	14	13	12
Col	11	10	24*	23*	AP	25	9	8	7	6	5	4	3	2	**

NOTES:
1. * = Unused bit.
2. ** = Fixed level; in this case 0.
3. T = 1, B = 1, F = 1, X = 1
4. Use A[13] and A[14] as Bank Selects.
5. The 'AP' entry refers to SDRCConfig0:AP (the AP bit field in the SDRCConfig0 register).

6.1.4 Merging Write and Read Buffers

The LH79520 incorporates a set of merging write and read buffers which convert all read and write operations into quad-word bursts. Operation is transparent to the programmer.

Each write buffer is split into two halves. Each half holds a quad word, which is the size of the default SDRAM data burst length. Using two quad words allows a new quad word to be buffered while the content of the other quad-word buffer is transferred to memory. The quad-word buffer will merge noncontiguous word, half-word or byte writes to the same quad word address.

The buffers operate independently on the AHB. The contents of a buffer will be transferred (or flushed) to SDRAM when:

- there is a write to an SDRAM address outside of the current merging quad word address
- there is a read from the address of the merging quad word
- the write buffer times out
- the write buffer is disabled.

The halves of the quad-word write buffer are exchanged when a write to a different quad work address starts (a write buffer miss). The buffer half that was previously merging data is flushed (contents written to SDRAM) while new data writes are merged in the new input buffer space.

When a write-back is triggered by a read from the memory address of the merging quad word, the write-back operation is completed before the requested data is read from memory. Write-before-read insures the quad word returned is up to date. The buffer contents must be merged with the memory contents to insure all the bytes of the quad word read from memory are updated.

Each buffer has an associated timer. The write buffer timer forces a write-back (buffer flush) after a programmable delay. Any write to a buffer starts the timer by loading the timeout count into a down-counter. The delay time is measured from the last write to the active half of a merging write buffer.

The read buffers work independently and hold the last quad word read. Any read request to an address in the same quad word as the buffered data uses the buffered data and will not cause an external memory access. A write to the same address in the same quad word as the buffered data will invalidate the buffered data but only in the read buffer.

6.1.5 SDRAM System Initialization

At power-on reset, software must initialize the LH79520 SDRC, and then initialize each of the SDRAM devices connected to the SDRC.

SDRAM devices are configurable. SDRAM devices must be powered-up and configured in a manner predefined by the device manufacturer. The exact initialization sequence for SDRAM devices can vary by manufacturer. Sequences not specified by the manufacturer can result in undefined operation.

Programmers should review the manufacturer's data sheet for the actual SDRAM devices in use. The initialization sequences in this section are presented as general guidance.

6.1.5.1 JEDEC General SDRAM Initialization Sequence

1. Apply Power. Apply VDD/VDDQ equally, and ensure that the clock to the SDRAM device is stable.
2. Wait at least 200 µs.
3. Select the SDRAM device's PRECHARGE ALL mode.
4. Perform eight AUTO REFRESH commands.
5. Program the SDRAM device's LOAD MODE register.

The SDRAM is ready for operation.

6.1.5.2 Micron SDRAM Initialization Sequence

A stable clock is defined as a signal cycling within the timing constraints specified by the SDRAM device manufacturer for the SDRAM device's clock pin. When power has been applied and the clock is stable, SDRAM devices usually require a delay prior to issuing any commands other than a COMMAND INHIBIT or a NOP. The COMMAND INHIBIT or NOP commands should be applied to the device throughout this delay.

When the delay has expired, a PRECHARGE ALL command should be issued. All SDRAM banks must then be precharged, thereby placing the SDRAM device in the ALL BANKS IDLE state.

Once the SDRAM device is in the IDLE state, two AUTO REFRESH cycles must be performed. When the two AUTO REFRESH cycles are completed, the SDRAM device is ready for MODE REGISTER programming. The SDRAM device's MODE register powers up in an unknown state, and should be programmed before any other operational commands are issued to the SDRAM device.

Although Micron SDRAM devices can be initialized with a JEDEC sequence, Micron devices also permit a faster initialization sequence:

1. Apply power. Apply VDD/VDDQ equally, and ensure that the CLK is stable.
2. Wait at least 100 µs. Begin and continue applying NOP or COMMAND INHIBITs during this delay.
3. Issue a PRECHARGE ALL command to the SDRAM device.
4. Issue two (or more) AUTO REFRESH commands to the SDRAM device. The order of the AUTO REFRESH and LOAD MODE REGISTER commands can be reversed if preferred.
5. Program the SDRAM device's LOAD MODE REGISTER.

The SDRAM is ready for operation.

6.2 SDRC Programmer's Model

The Base Address for the registers that configure the SDRAM Controller is:

SDRAMBase: 0xFFFF2000

All registers in the SDRC will accept word, half-word, and byte accesses.

6.3 SDRC Register Summary

Table 6-19 summarizes the programmable registers that configure the LH79520 SDRC. Each register is treated in more detail on the following pages.

Table 6-19. SDRAM Controller Register Summary

ADDRESS	REGISTER NAME	DESCRIPTION
SDRAMBase + 0x000	SDRCConfig0	SDRAM Memory Bank 0 Configuration Register
SDRAMBase + 0x004	SDRCConfig1	SDRAM Memory Bank 1 Configuration Register
SDRAMBase + 0x008	SDRCRefTimer	SDRAM Refresh Timer Register
SDRAMBase + 0x00C	SDRCWBTimeout	SDRAM Write Buffer Time-Out Register

6.4 SDRC Register Descriptions

This section of this User's Guide lists the definitions for each of the programmable registers that control the SDRC.

6.4.1 SDRC Configuration Register 0

The SDRCConfig0 must be programmed to describe how the external SDRAM device(s) are to function.

Table 6-21 explains each of the bit fields in this register.

Table 6-20. SDRCConfig0 Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///							AP	R1	R0	C1	C0	X	C	E	///
RESET	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								B1	T1	F1	///	B0	T0	F0	///
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	SDRAMBase + 0x000															

Table 6-21. SDRCConfig0 Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:25	///	**Reserved** Write the reset value.
24	AP	**Auto Pre-Charge Control for SDRAM Accesses** 0 = No Auto Pre-Charge 1 = Auto Pre-Charge (reset value)
23:22	R1:R0	**RAS to CAS Latency SDRAM Mode** 0b00 = Reserved; do not write 0b01 = Reserved; do not write 0b10 = RAS to CAS latency = 2 0b11 = RAS to CAS latency = 3 (reset value)
21:20	C1:C0	**CAS Latency** 0b00 = Reserved; do not write 0b01 = CAS Latency = 1 0b10 = CAS Latency = 2 0b11 = CAS Latency = 3 (reset value)
19	X	**External Bus Width** 0 = External Bus width = 32 (reset value) 1 = External Bus width = 16
18	C	**SDRAM Clock Enable (SDCKE) Control (Shutdown Mode)** 0 = Clock Enables of IDLE devices are de-asserted, to save power (reset value) 1 = Clock Enables are driven continuously HIGH The combination of C = 1 and E = 1 is not allowed. See the 'E' bit field.

Table 6-21. SDRCConfig0 Register Bit Fields (Cont'd)

BITS	FIELD NAME	FUNCTION
17	E	**SDRAM Clock Control** 0 = SDCLK signal runs continuously (reset value) 1 = SDCLK signal stops when all SDRAMs are idle. The E bit should be set only when the SDCKE shutdown mode is active (the combination of E = 1 and C = 1 is not allowed).
16:8	///	**Reserved** Write the reset value.
7	B1	Indicates whether the SDRAM device attached to Chip Select nDCS1 is a 2- or 4-bank device. 0 = 2-bank device (reset value) 1 = 4-bank device
6	T1	Sets the address multiplexing used for Chip Select nDCS1. 0 = ×16 or ×32 memory device databus width (reset value) 1 = ×8 memory device databus width
5	F1	Sets the address multiplexing for Chip Select nDCS1, for 256M SDRAM. 0 = Not 256M device (reset value) 1 = 256M devices
4	///	**Reserved** Write the reset value.
3	B0	Indicates whether the SDRAM device attached to Chip Select nDCS0 is a 2- or 4-bank device. 0 = 2-bank device (reset value) 1 = 4-bank device
2	T0	Sets the address multiplexing used for Chip Select nDCS0. 0 = ×16 or ×32 memory device databus width (reset value) 1 = ×8 memory device databus width
1	F0	Sets the address multiplexing for Chip Select nDCS0, for 256M SDRAM. 0 = not 256M device (reset value) 1 = 256M devices
0	///	**Reserved** Write the reset value.

NOTE: Software must not write to register SDRCConfig0 when the SDRAM Controller is busy. Software should check SDRCConf1:B (the B bit field in the SDRCConf1 register) to ensure that the controller is not busy before writing to SDRCConfig0.

6.4.2 SDRC Configuration Register 1

The SDRCConfig1 register contains the (read-only) SDRC Status bit and other fields that must be programmed to configure the SDRC's read\write buffers. See Section 6.1.4 for further information on the buffers.

Table 6-23 explains each of the bit fields in this register.

Table 6-24 shows how SDRCConfig1:M and SDRCConfig1:I control the initialization of an external SDRAM memory device.

Table 6-22. SDRCConfig1 Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///										B	///	W	R	M	I
RESET	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RO	RW	RW	RW	RW	RW
ADDR	SDRAMBase + 0x004															

Table 6-23. SDRCConfig1 Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:6	///	**Reserved** Write the reset value.
5	B	**SDRAM Controller Status Bit** 0 = SDRAM Controller is idle 1 = SDRAM Controller is busy
4	///	**Reserved** Write the reset value.
3	W	**Write Buffer Enable** 0 = Merging write buffer(s) disabled 1 = Merging write buffer(s) enabled (reset value) Disabling the write buffer(s) will flush any stored value(s) to the external memory it is buffering.
2	R	**Read Buffer Enable** 0 = Read buffer(s) disabled 1 = Read buffer(s) enabled. (reset value)
1	M	Control bit for memory device initialization. See Table 6-24.
0	I	Control bit for memory device initialization. See Table 6-24.

Table 6-24. Using the M and I Bit Fields

M	I	FUNCTION
1	1	Automatically issue a NOP to the SDRAM
1	0	Enable the SDRAM MODE command
0	1	Automatically issue a PALL (Precharge All) to the SDRAM
0	0	Normal operation (reset value)

6.4.3 SDRC Refresh Timer Register

The SDRCRefTimer register establishes the SDRAM refresh interval as a function of the AHB clock.

Table 6-26 explains each of the bit fields in this register.

Table 6-25. SDRCRefTimer Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	RTVAL															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	SDRAMBase + 0x008															

Table 6-26. SDRCRefTimer Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:16	///	**Reserved** Write the reset value.
15:0	RTVAL	The quantity of AHB clock cycles that should be counted between SDRAM refresh cycles.

The SDRC Refresh Timer Register must be programmed with the quantity of AHB clock cycles that should be counted between SDRAM refresh cycles.

For example, for a refresh period of 16 μs and a AHB clock frequency of 50 MHz, the SDRCRefTimer register should be programmed to:

(16 μs) × (50 MHz) = 800 = 0x0320

The Refresh Timer in the SDRAM Controller (SDRC) is automatically initialized to a value of 128 (0x080) at power-on Reset. To ensure a refresh interval of less than 16 μs after reset, the minimum allowable AHB clock frequency is:

128 ÷ (16 μs) = 8 MHz

The SDRCRefTimer register should be programmed as early as possible in the system start-up procedure. If the AHB clock is less than 8 MHz, the register should be programmed within the first few clock cycles after reset.

6.4.4 SDRC Write Buffer Timeout Register

The SDRCWBTimeout register sets the delay time for a forced flush of the write buffer.

Table 6-28 explains each of the bit fields in this register.

Table 6-27. SDRCWBTimeout Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	TOVAL															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	SDRAMBase + 0x008															

Table 6-28. SDRCWBTimeout Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:16	///	**Reserved** Write the reset value.
15:0	TOVAL	The delay (in AHB clocks) that must occur before the SDRC's Merging Write Buffer is flushed. Writing '0' disables the timeout.

The Write Buffer Timeout Register establishes the delay that must expire before a forced flush (write to SDRAM) of the SDRC's Merging Write Buffer occurs. This is useful to ensure that video display data is correctly updated.

The SDRC contains a down-counter which is clocked by the AHB clock. This down-counter is preloaded with the value in the SDRCWBTimeout Register whenever a write to the SDRC's Merging Write Buffer occurs.

If the SDRC's Merging Write Buffer is regularly accessed (written-to), the down-counter will never be allowed to decrement to zero. If writes to the Merging Write Buffer are delayed or blocked, the down-counter will eventually decrement to zero. When the down-counter reaches zero the contents of the Merging Write Buffer are written (flushed) to the external memory.

Writing SDRCWBTimeout:TOVAL = 0 disables the Write Buffer Timeout function.

Chapter 7
Reset, Clock Generation, and Power Control (RCPC)

This chapter of this User's Guide presents details of the Reset, Clock Generation and Power Control (RCPC) systems in the LH79520 MCU. The RCPC is the control center for the LH79520. A discussion of the Theory of Operation of the RCPC is followed by descriptions of the internal registers mentioned in the discussion.

7.1 Theory of Operation

The LH79520 RCPC affects most of the other systems within the LH79520, including the ARM core, DMA Controller, Real-Time Clock, Watchdog Timer, Synchronous Serial Ports, Timers, Pulse-Width Modulators and the UARTs.

The RCPC is composed of the functional blocks shown in Figure 7-1. The Figure shows how these block relate to the Phase-Locked Loop (PLL) circuit and the on-chip circuits which, with the addition of external crystals, function as oscillators. Figure 7-1 also relates the appropriate external signals (shown with circled pin numbers, like nRESETIN) to these functional blocks.

The RCPC is programmed via an APB interface and provides power management and the ability to control system clocks and resets. This control includes enabling and disabling the clocks, the onboard Phase Lock Loop circuit (PLL) and the internal oscillator circuit for the external clock crystal. Control also includes selecting the sources for various clocks, varying their frequencies, enabling and disabling the clocks and managing the LH79520 power-down sequencing.

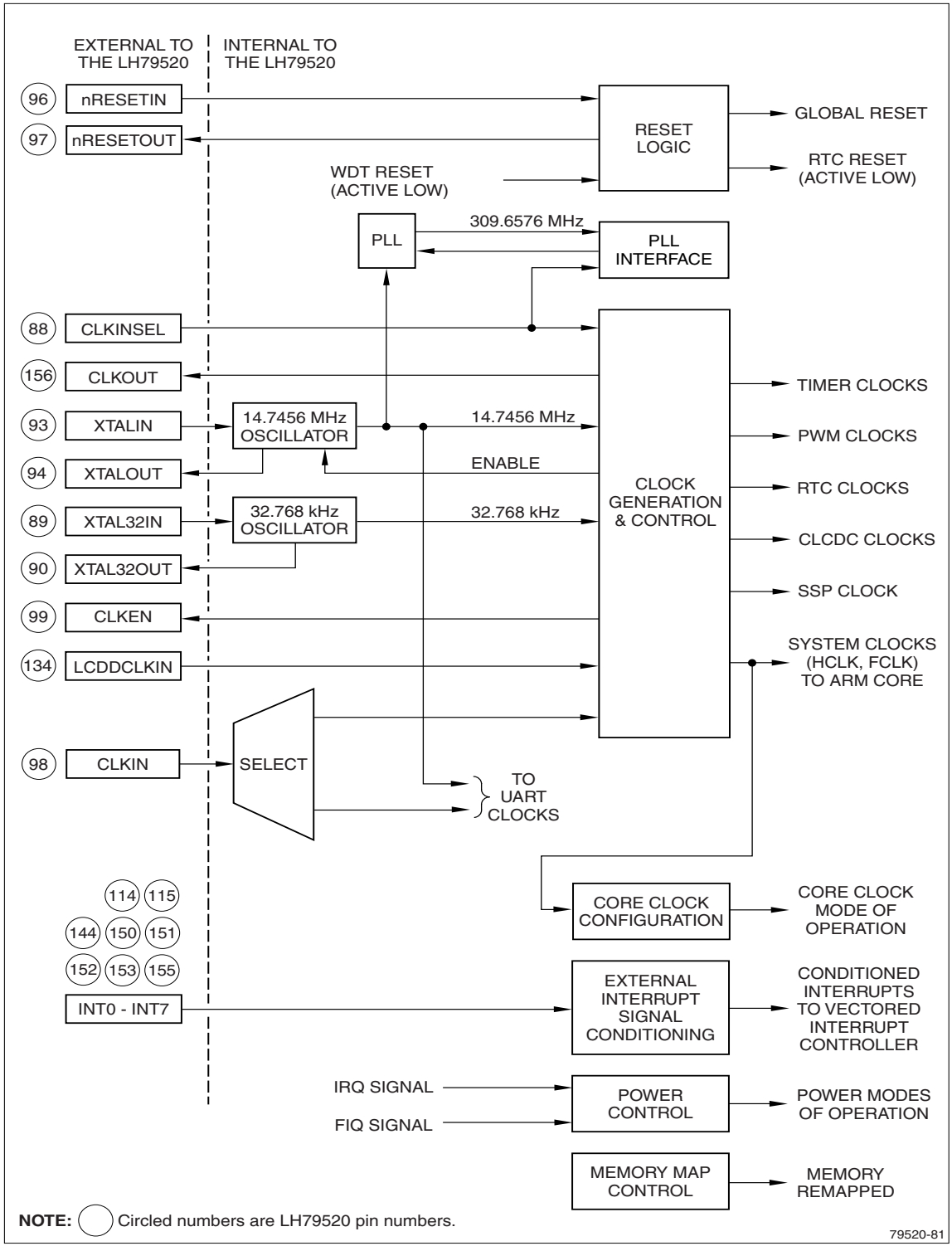


Figure 7-1. RCPC Functional Blocks

7.2 Write Locking

All RCPC programmable registers are reset to RW (Read-Write) but can be set to RO (Read-Only) by software. See Table 7-4 in Section 7.15 for a complete list of the affected registers.

7.3 Reset Logic

Two systems within the LH79520 can drive a Reset. Figure 7-2 shows that the Real-Time Clock (RTC) and the remainder of the MCU will be reset simultaneously by hardware but can be reset separately under software control. The Figure shows that resets can be generated in three ways:

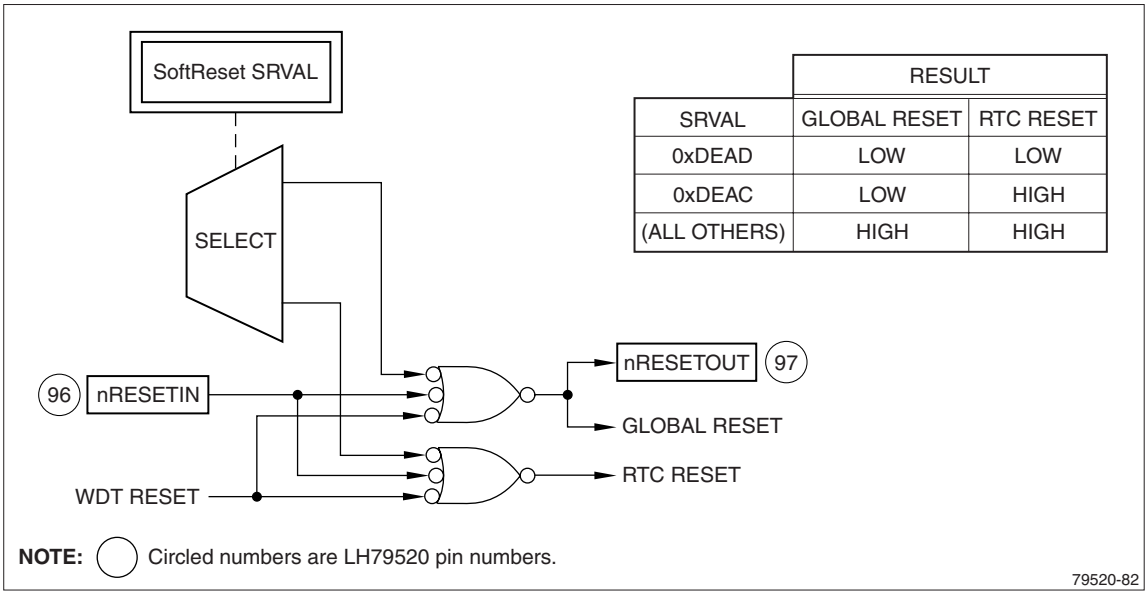
- A logic LOW signal on the external nRESETIN input pin
- A signal from the (internal) Watchdog Timer
- When a special value is written to the SoftReset register

Timing specifications associated with resets are included in the AC Characteristics section of this Guide.

System reset latency depends upon the source of the system clock. If the LH79520 is operating from its internal clock source (using a 14.7456 MHz crystal and the internal PLL system) then reset will be slightly delayed while the PLL acquires lock. If an external clock source (at the CLKIN input, with the CLKINSEL pin tied HIGH) is used, then the PLL will be reset to a disabled condition and the system reset will complete after a duration of time equal to eight cycles of the AHB clock (HCLK), which is a function of the external clock source.

A Global (system-wide) reset will return the LH79520 to its 'Active' power mode, explained in the Power Management and Power Modes section of this User's Guide. This will occur whether or not the RTC is also reset.

See the sections of this chapter of this User's Guide that detail the SoftReset, ResetStatus and ResetStatusClr registers for additional details regarding the Reset system.



	RESULT	
SRVAL	GLOBAL RESET	RTC RESET
0xDEAD	LOW	LOW
0xDEAC	LOW	HIGH
(ALL OTHERS)	HIGH	HIGH

Figure 7-2. LH79520 Resets

7.4 PLL (Phase-Locked Loop) Interface

The LH79520 includes an on-chip system to generate a 309.6576 MHz signal. This signal is divided to lower frequencies and distributed to the core, the buses and various peripherals. This system includes an oscillator, a PLL circuit and a PLL Interface. An external 14.7456 MHz crystal is required. If this internal system is used, the oscillator and the PLL are permanently enabled and cannot be disabled in software because doing so would defeat the operation of the LH79520.

The PLL Interface, shown in Figure 7-1, provides for an orderly start-up by automatically imposing a delay until the 14.7456 MHz crystal oscillator (if used) stabilizes and the PLL (if used) acquires lock. The frequency of the PLL is not adjustable.

If the LH79520 is to be driven by an external clock signal (at the CLKIN pin, with the CLKINSEL pin HIGH at reset) then the 14.7456 MHz crystal is not necessary (unless the UARTs are required) and the oscillator and PLL are automatically disabled at reset. If required, the oscillator and the PLL may be re-enabled by software for driving the UARTs, but doing so requires the connection of a 14.7456 MHz crystal to drive the UART clocks, since the input clock path is multiplexed and therefore unavailable to the UARTs when the chip is being driven by this external clock signal.

When the LH79520 is reset, the state of the CLKINSEL pin is automatically latched to RCPCCtrl:CLKSEL (the CLKSEL bit field in the RCPCCtrl register). Software can inspect the (Read-Only) CLKSEL bit field to establish the reset circumstance: the CLKSEL bit field will be HIGH if the CLKINSEL pin was HIGH at reset, LOW if the pin was LOW.

If the LH79520 is reset with the CLKINSEL pin HIGH, then the on-chip oscillator and the PLL are disabled but can be re-enabled if desired. Figure 7-3 shows how RCPCCtrl:CLKSEL is utilized. The figure also shows that the oscillator is controlled by RCPCCtrl:EX and the PLL by RCPCCtrl:EP.

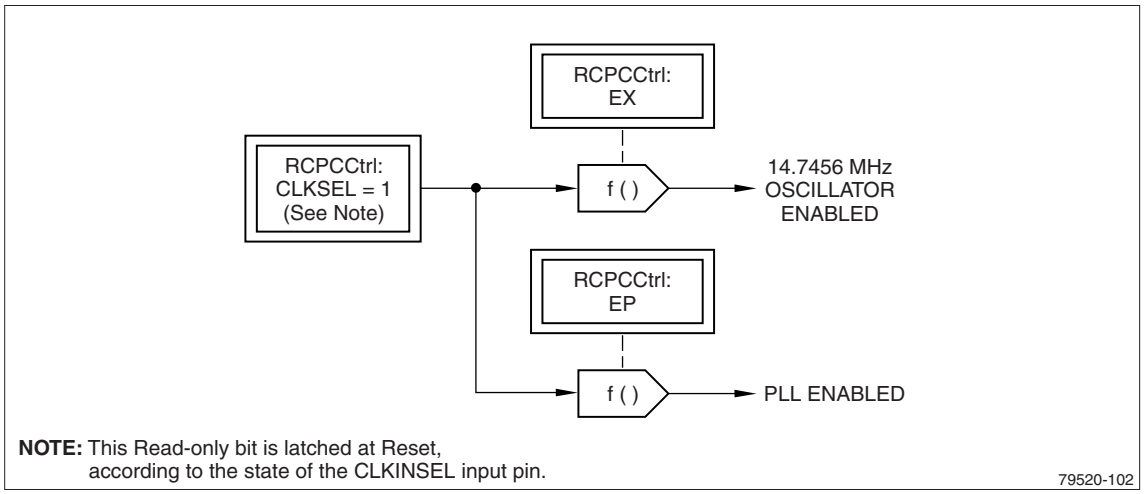


Figure 7-3. PLL Interface

7.5 Clock Generation and Control

The LH79520 can generate all internal clock signals (except the signal that drives the on-chip Real-Time Clock peripheral) from one of two sources:

- An internal 309.6576 MHz system (requires an external, user-supplied 14.7456 MHz crystal)
- An external clock signal at the CLKIN pin (requires that the CLKINSEL pin be HIGH at reset).

The internal 309.7456 MHz clock-generation system consists of an oscillator circuit, a PLL Interface and a Phase-Locked Loop (PLL) circuit optimized for use with a 14.7456 MHz external crystal. The crystal completes the oscillator circuit and the PLL multiplies the oscillator's output frequency by 21 to produce a 309.6576 MHz signal. This signal is then divided to obtain the AHB and APB clocks, and differently divided to obtain the FCLK signal for the ARM720T core. The crystal, oscillator, PLL and PLL Interface functional blocks are shown in Figure 7-1.

To use the internal clock-generation system, users must connect a 14.7456 MHz crystal to the XTALIN and XTALOUT pins and ensure that the CLKINSEL pin is LOW at reset. The CLKINSEL pin has an internal pull-down. If the CLKINSEL pin is HIGH at reset, the LH79520 will defeat the internal oscillator circuit, defeat the onboard PLL, and seek an external clock signal at the CLKIN pin.

To use an external clock signal, the external clock signal must be connected to the LH79520 CLKIN pin. In this case the CLKINSEL pin must be HIGH at reset, when its condition is latched to RCPCCtrl:CLKSEL (the CLKSEL bit in the RCPCCtrl register). See the explanation of the CPUClkPrescale and HCLKPrescale registers for additional information regarding clock division and distribution.

The LH79520 also contains a second oscillator circuit intended to drive the Real Time Clock (RTC) peripheral. If used, this 32.768 kHz system will require a second, external crystal. If not used, the RTC can be disabled by software, to avoid drawing excess power. See Chapter 14 – Real-Time Clock (RTC) for more information about this second clock system.

The Clock Generation and Control block generates several clock signals. These are the:

- Timer clocks
- Clocks for the Color LCD Controller (CLCDC)
- Clocks for the Synchronous Serial Port (SSP)
- FCLK and HCLK and its derivatives, for the CPU and other systems that access the AHB
- PCLK, for APB devices (the on-chip peripherals; locked to HCLK)
- The CLKOUT signal, available at an external pin
- Clocks for the Pulse-Width Modulators (PWMs).

If not used, each of these clocks can be disabled or programmed to a low frequency, to save power. Most clocks are disabled at reset and must be enabled by software. Programmers should ensure that the system clock to any of these peripherals is enabled before programming any registers.

It should be noted that PCLK is locked to HCLK in the LH79520. See Section 7.4.

Figure 7-4 presents a general view of the generation and distribution of all clock signals. The signals shown with circled numbers, like CLKOUT, are available at pins on the LH79520 package. The other signals are included for reference and are not directly available outside the package. The Real Time Clock passes through this block, but only for division and control; it is not used in any Reset or power-down operations.

Figures 7-6 through Figure 7-11 present additional details about each of these clock signals.

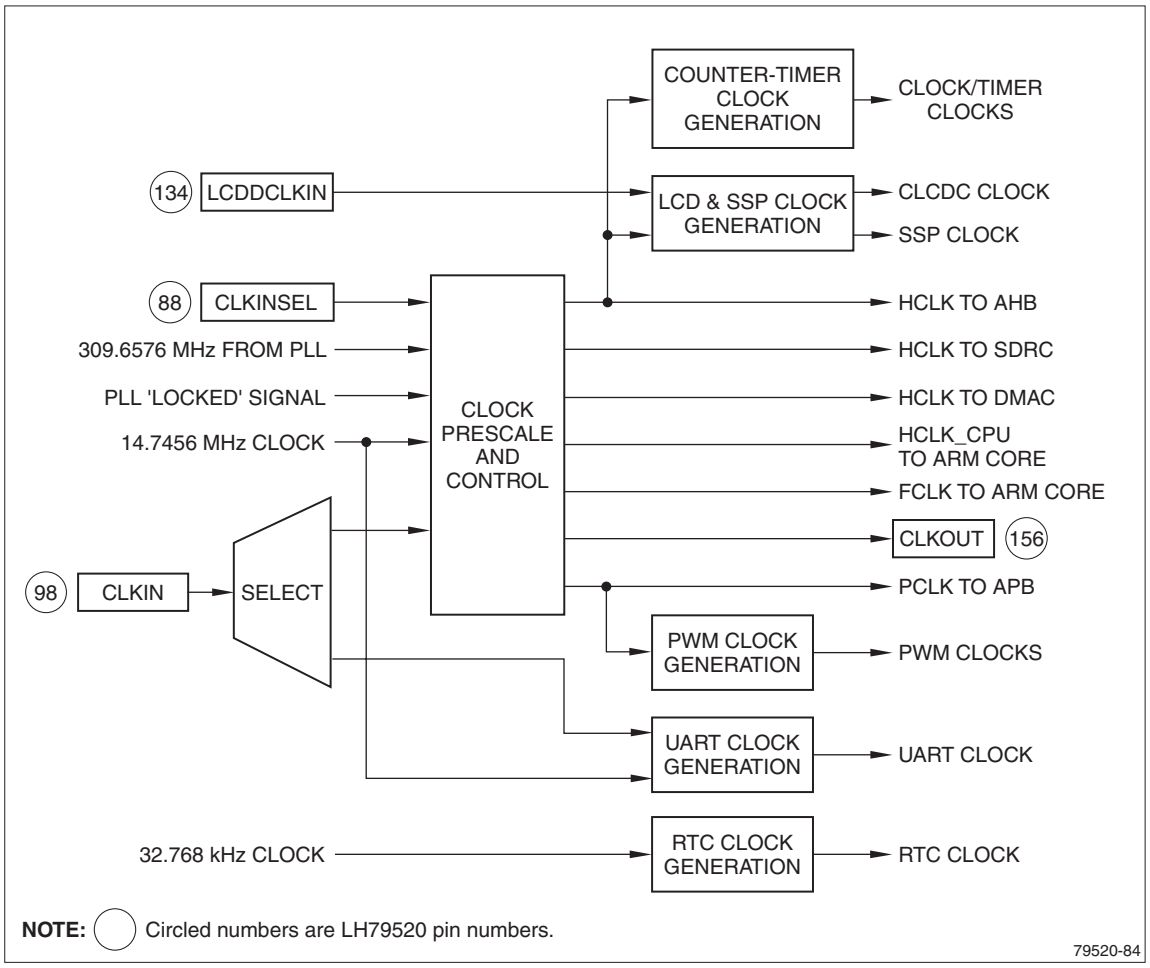


Figure 7-4. Clock Generation

7.6 Timer Clock Generation

Figure 7-5 illustrates how the clock signals for four of the on-chip Timer peripherals are generated and controlled. Timer0 and Timer1 receive a clock signal which can be enabled or disabled. Timer2 and Timer3 receive a similar signal. Both clock signals are disabled at reset. See Chapter 7 for details about programming the Timer clocks signals.

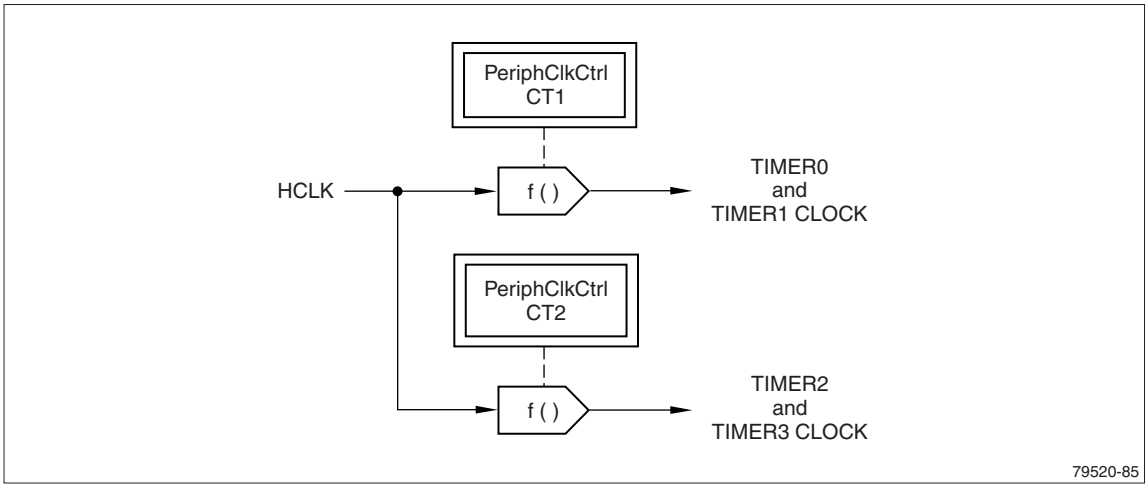


Figure 7-5. Timer Clock Generation

7.7 Color LCD Controller (CLCDC) Clock

Figure 7-6 shows the way in which the clock signal for the CLCDC peripheral is generated and controlled. PeriphClkSel2:LCSRC (the LCSRC bit field in the PeriphClkSel2 register) determines whether the CLCDC receives HCLK or an external signal connected to the LCDCLKIN pin. PeriphClkCtrl2:LCDCLK enables or disables the chosen signal and LCDClkPrescale:LCDPSVAL selects the ratio by which the signal is divided before use by the Color LCD Controller (CLCDC).

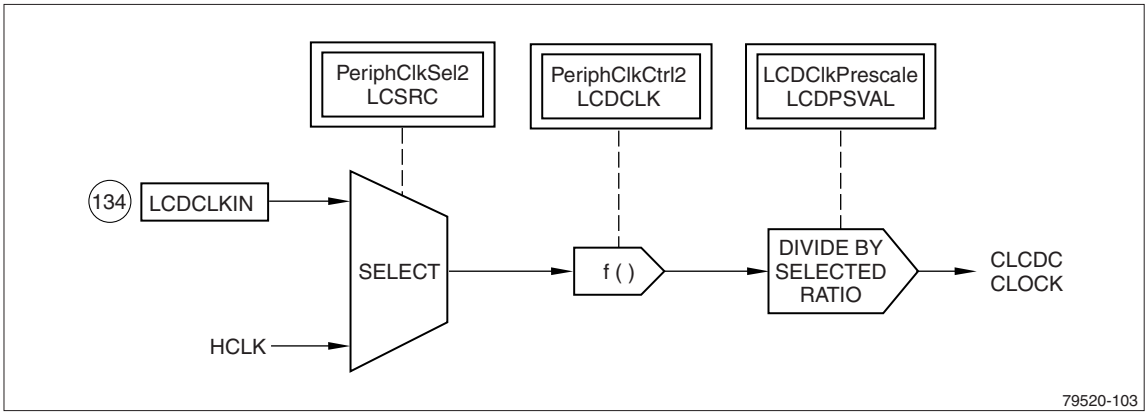


Figure 7-6. CLCDC Clocks

7.8 Synchronous Serial Port (SSP) Clock

Figure 7-7 shows the way in which the clock signal for the SSP peripheral is generated and controlled. The SSP peripheral receives HCLK, gated and pre-scaled by PeriphClkCtrl2:SSPCLK and SSPClkPrescale:SSPPSVAL, respectively.

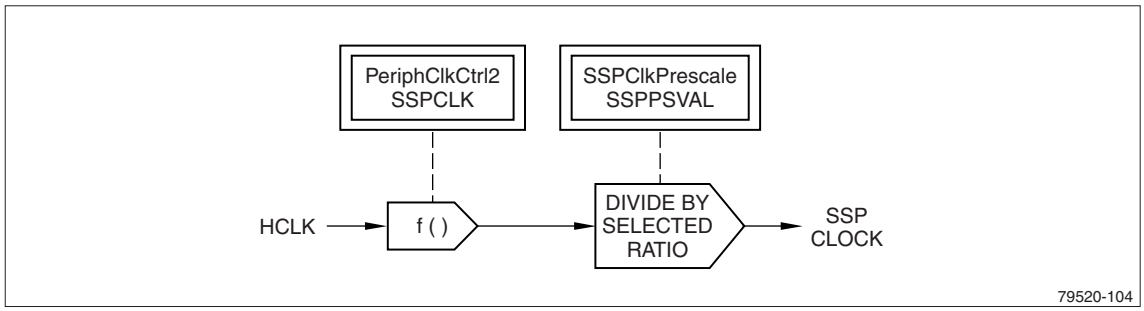


Figure 7-7. SSP Clocks

7.9 FCLK, HCLK, PCLK and CLKOUT Generation

Figure 7-8 shows the generation and distribution of the system clocks and the CLKOUT signal. HCLK is the signal that drives the AHB. The HCLK derivatives (HCLK_DMAC and HCLK_SDRC) drive the DMA Controller and the SDRAM Controller, respectively.

The core and cache are clocked by FCLK (or HCLK_CPU, depending upon the core's Mode of operation, explained in Chapter 3, Section 3.8). The numeric clock frequencies shown in Figure 7-8 are for reference only, as suggested maximum frequencies and to illustrate that FCLK must ALWAYS be ≥ HCLK and its derivatives. The core and cache must never be operated at a lower frequency than the AHB.

Figure 7-8 also shows that one of four signals may be made available at the CLKOUT pin, depending upon the value programmed to RCPCCtrl:OUTSEL (the OUTSEL bit field in the RCPCCtrl register).

PCLK is locked to HCLK (note the connections in Figure 7-8) and is the signal that drives the APB and is the basis for other programmable peripheral clocks.

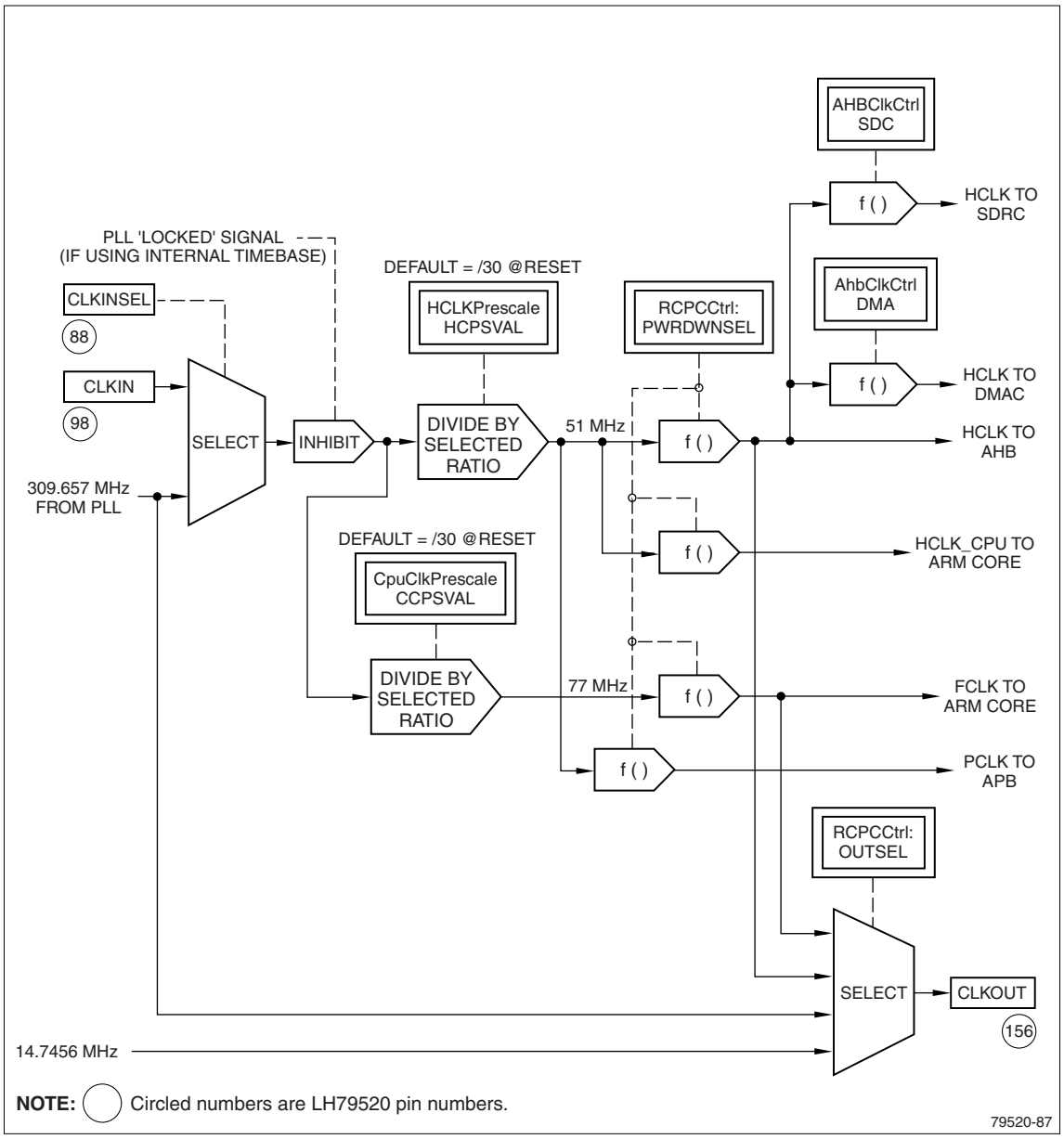


Figure 7-8. HCLK, FCLK AND PCLK Clocks

7.9.1 Programming The System Clocks

Figure 7-8 shows that the HCLK signal routed to the SDRC is controlled by the SDC bit field in the AHBClkCtrl register. Software must write AHBClkCtrl:SDC to '1' before programming any or all of the CpuClkPrescale, HCLKPrescale, or RCPCCtrl registers.

Programmers should follow this general sequence when changing the system clock frequencies:

7.9.1.1 Programming a New Prescale Value

- Program CoreClkCfg:CFGVAL to select the asynchronous mode of operation
- Program CPUClkPrescale and/or HCLKPrescale to select the desired frequency(s).
  - If programming both CPUClkPrescale and HCLKPrescale, always program CPUClkPrescale before programming HCLKPrescale.
  - When programming either CPUClkPrescale or HCLKPrescale, always ensure that FCLK ≥ HCLK and FCLK ≥ HCLK_CPU
- Program CoreClkCfg:CFGVAL to select the new mode of operation.

7.9.2 Selecting Frequencies

Table 7-1 lists the clock frequencies obtainable with a 14.7456 MHz crystal connected to the XTALIN and XTALOUT pins of the LH79520. The crystal completes the on-chip oscillator circuit to produce a 14.7456 MHz signal. The on-chip Phase-Locked Loop (PLL) circuit multiplies the 14.7456 MHz signal by 21 to obtain a higher frequency which is then divided according to the values programmed to the CPUClkPrescale and HCLKPrescale registers.

Any combination of CPUClk and HCLK Prescale values may be used within these valid ranges. They need not be programmed the same.

Table 7-1. Frequencies Obtainable with a 14.7456 MHz Crystal

FCLK or HCLK DESIRED DIVISION	CPUClkPrescale BIT-FIELD VALUE	HCLKPrescale BIT-FIELD VALUE	RESULTANT FCLK or HCLK (MHz)
1			
2			
3			
4	0b0010		77.4144
6	0b0011	0b0011	51.6096
8	0b0100	0b0100	38.7072
10	0b0101	0b0101	30.9657
12	0b0110	0b0110	25.8048
14	0b0111	0b0111	22.1184
16	0b1000	0b1000	19.3536
18	0b1001	0b1001	17.2032
20	0b1010	0b1010	15.4828
22	0b1011	0b1011	14.0734
24	0b1100	0b1100	12.9024
26	0b1101	0b1101	11.9099
28	0b1110	0b1110	11.0592
30	0b1111	0b1111	10.3219

NOTES:
1. Shaded areas indicate invalid values.
2. If the external clock-select pin, CLKINSEL, is latched HIGH at reset then the CLKSEL bit will be set to '1' and the EX and EP bit fields will be RO (Read-Only).

7.10 Pulse-Width Modulator Clocks

The clock for either PWM channel is derived from PCLK and can be enabled, disabled and pre-scaled, as shown in Figure 7-9. See the explanations of the PeriphClkCtrl, PWM0Prescale and PWM1Prescale registers for more information.

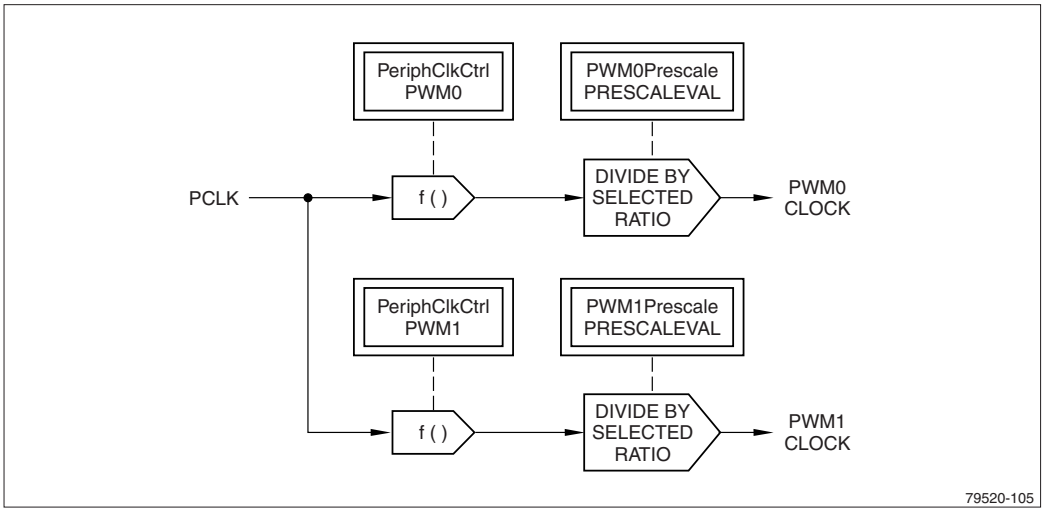


Figure 7-9. PWM Clock Generation

7.11 RTC Clock

Figure 7-11 illustrates that the clock signal routed to the Real-Time Clock (RTC) peripheral is generated by the output of the 32.768 kHz clock oscillator, or by that same signal divided internally by 32,768. The lower-frequency (divided) signal is intended for normal use. The 32.768 kHz clock passes through the RCPC only for division to 1 Hz; it does not support power-down or other RCPC operations. If the 32.768 kHz crystal oscillator is not required for the design, it may be left out to save power, the clock disabled in software, and the XTAL32IN pin pulled to VSS. XTAL32OUT should be left floating.

The input-signal selection is determined by PeriphClkSel:RTC (the RTC bit field in the PeriphClkSel register) and the output fed to the RTC peripheral can be enabled or disabled by PeriphClkCtrl:RTC (the RTC bit field in the PeriphClkCtrl register).

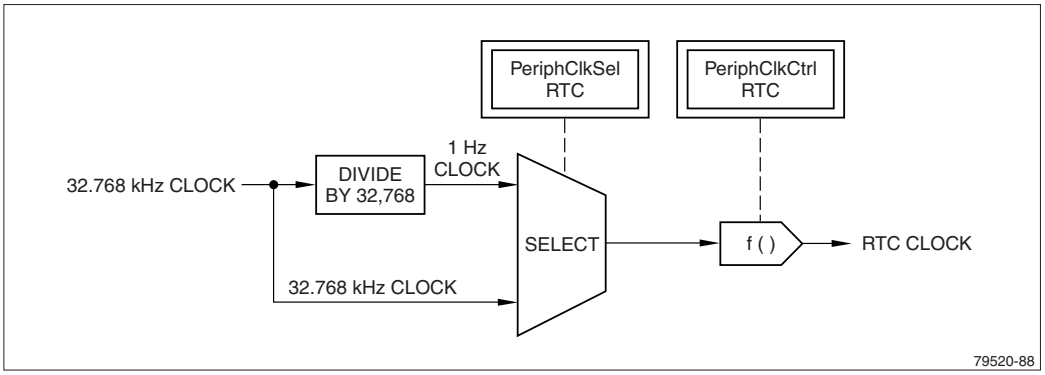


Figure 7-10. RTC Clock Generation

7.12 UART Clocks

Figure 7-11 illustrates that the UART clocks can be generated by the 14.7456 MHz clock oscillator signal or by an external signal via the CLKIN pin to generate custom baud rates while the rest of the chip is being clocked by the 14.7456 MHz oscillator. Each of the three UART clocks can be enabled or disabled by a bit field in the PeriphClkCtrl register. However, the CLKIN signal is muxed between the main clock path and the UARTs. To use the UARTs with this arrangement, an external 14.7456 MHz crystal oscillator must be added and selected for the UART clocks. It is important to note that the LH79520 will allow the selection of an external clock for the UARTs when the CLKIN signal has been routed to the main clock path. Such a selection will result in no clock being fed to the UARTs.

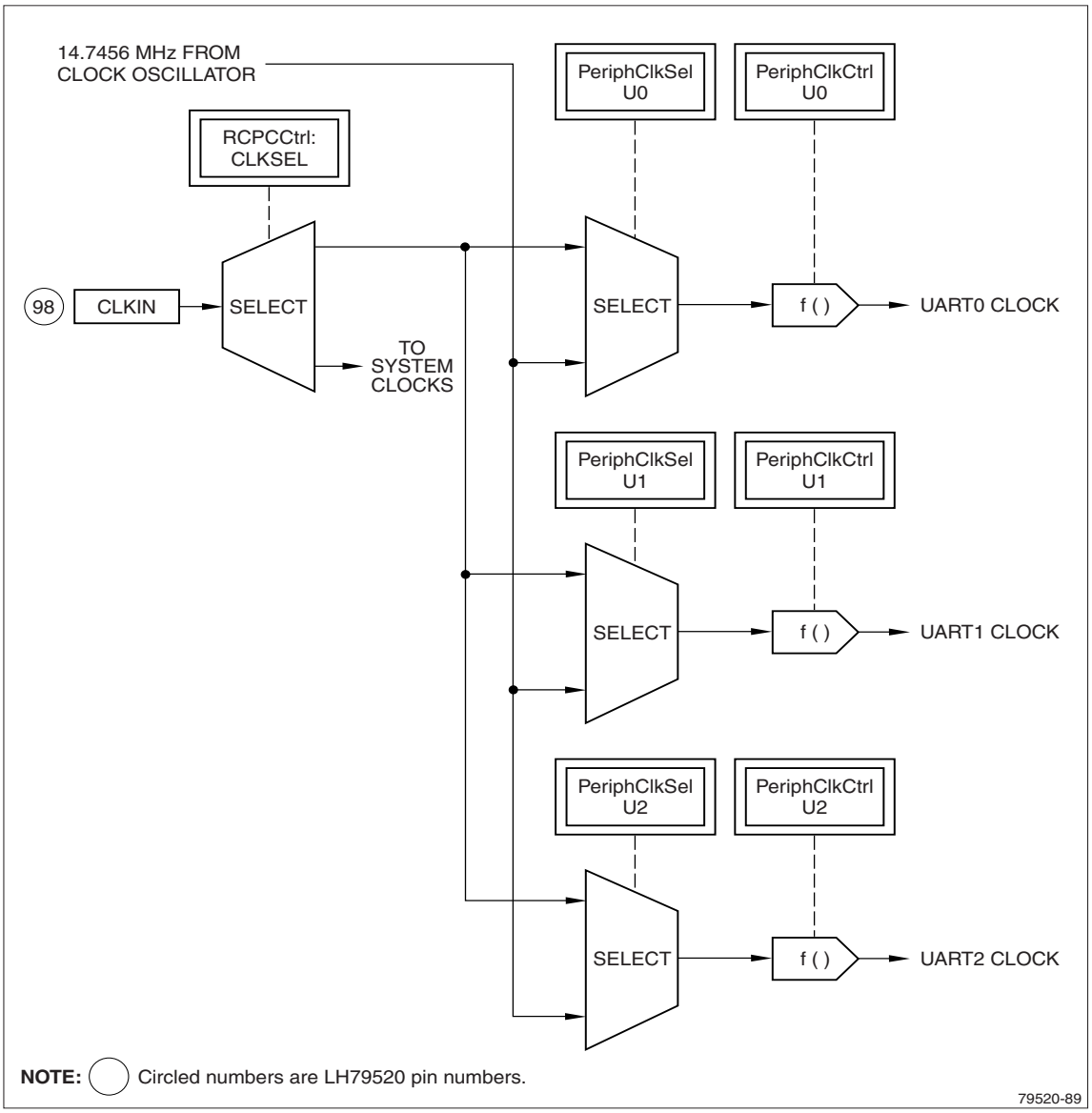


Figure 7-11. UART Clock Generation

7.13 Power Management and Power Modes

The LH79520 can be operated in any of the five different Power Modes shown in Figure 7-12. The five modes are: Active, Standby, Sleep, Stop1, and Stop2. These power modes reduce the MCU's power consumption in stages, with each successive mode providing progressively greater power savings.

The Active mode is the MCU's normal mode of operation and is also the mode capable of consuming the most power. The LH79520 transitions from the Active Mode to other modes under software control, and always returns to the Active mode on Reset or Interrupt.

Although the Power Management system includes circuitry to ensure orderly transitions between modes, the software is responsible for the overall power management strategy. Tables 7-2 and 7-3 list how the Power Modes affect the clocks and the CLKEN output. Clock signals not included in Tables 7-2 and 7-3 are individually controlled by software (and driven either by HCLK or from an external clock source).

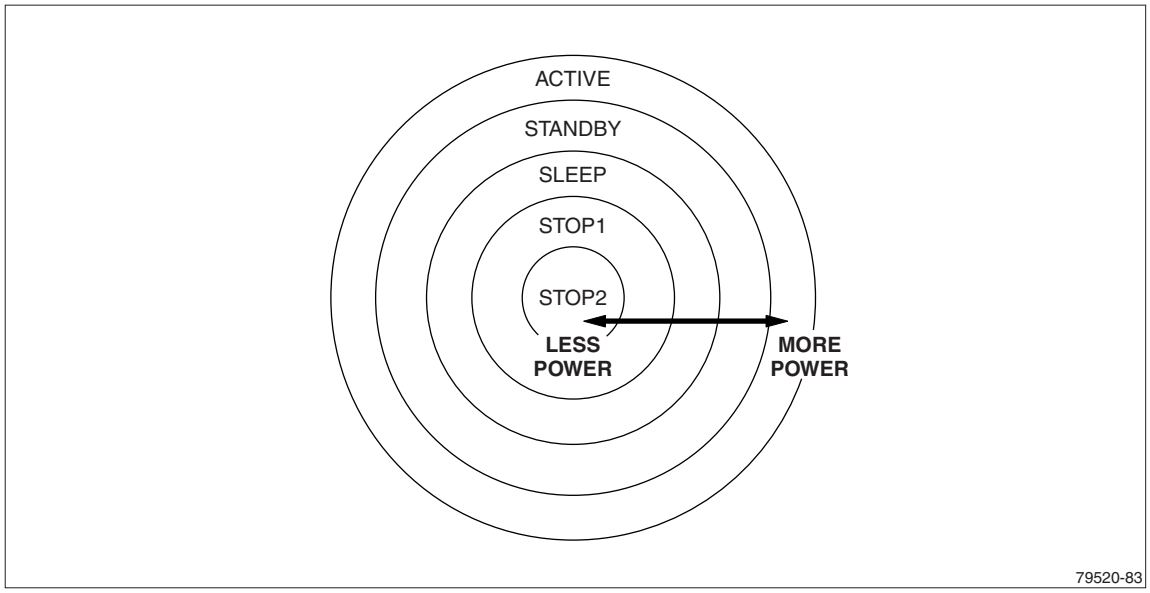


Figure 7-12. Successive Modes Reduce Power Usage

Table 7-2. Mode Control Using Internal Clock Generation (CLKINSEL pin LOW at Reset)

CLOCK	POWER MODE				
	ACTIVE	STANDBY	SLEEP	STOP1	STOP2
Internal 14.7456 MHz Oscillator Circuit	ON	ON	ON	ON	OFF
Internal 14.7456 MHz PLL Circuit	ON	ON	ON	OFF	OFF
CLKEN Output	ON	ON	ON	OFF	OFF
HCLK (the AHB Clock)	ON	ON	OFF	OFF	OFF
HCLK_CPU (Alternate ARM Core Clock)	ON	OFF	OFF	OFF	OFF
FCLK_CPU (the ARM Core Clock)	ON	OFF	OFF	OFF	OFF

Table 7-3. Mode Control Using An External Clock Source (CLKINSEL pin HIGH at Reset)

CLOCK	POWER MODE				
	ACTIVE	STANDBY	SLEEP	STOP1	STOP2
Internal 14.7456 MHz Oscillator Circuit	OFF	OFF	OFF	OFF	OFF
Internal 14.7456 MHz PLL Circuit	OFF	OFF	OFF	OFF	OFF
CLKEN Output	ON	ON	ON	OFF	OFF
HCLK (the AHB Clock)	ON	ON	OFF	OFF	OFF
HCLK_CPU (Alternate ARM Core Clock)	ON	OFF	OFF	OFF	OFF
FCLK_CPU (the ARM Core Clock)	ON	OFF	OFF	OFF	OFF

The LH79520 Power Mode is controlled by RCPCCtrl:PWRDWNSEL (the PWRDWNSEL bit field in the RCPCCtrl register). For precise control, the assembly-language instruction that writes to the PWRDWNSEL bit field should be followed by at least ten (10) NOP instructions. Fifteen (15) NOP instructions are recommended if the core is operating much faster than the AHB. Optimizing compilers can defeat this intention by removing in-line assembly code, so the NOPs' presence and quantity should be verified after the source code is compiled, assembled and linked. If undesirable optimization becomes an issue, programmers may wish to implement the NOPs within a separate assembly-language file.

The following sections of this User's Guide present details of each of the power modes.

7.13.1 Active Mode

The Active mode is the normal mode of operation for the LH79520. The LH79520 MCU enters this mode on Reset and returns to this mode when any interrupt is received, if operating in any other power mode. The LH79520 cannot transition directly between the other power modes; it will always return to the Active mode before entering any other power mode.

7.13.2 Standby Mode

The Standby mode halts the clocks to the CPU (HCLK_CPU and FCLK_CPU) while leaving the remainder of the chip active. The LH79520 transitions from the Standby mode to the Active mode when an interrupt is received.

7.13.3 Sleep Mode

The Sleep mode halts all system clocks. Only the PLL and the internal oscillators remain active. If the 32.768 kHz internal oscillator is in use, it will also remain active. The LH79520 transitions from the Sleep mode to the Active mode when an interrupt is received.

When transitioning from the Active mode to the Sleep mode, the LH79520 Power Management system automatically performs the following sequence:

- Acquires control of the AHB, to ensure that all transactions are completed.
- Ensures that all SDRAM devices are placed in the self-refresh mode of operation.
- Halts all output clocks that are driven by HCLK, HCLK_CPU, and PCLK.
- Waits for IRQ or FIQ to be asserted (which will return the LH79520 to the Active mode).

When an IRQ or FIQ occurs, the LH79520 returns to the Active mode, restarts the output clocks, resumes SDRAM refresh and then cedes control of the AHB.

7.13.4 Stop1 Mode

The Stop1 mode halts all system clocks and disables the PLL but keeps the internal oscillators active. If the 32.768 kHz internal oscillator is in use, it will remain active. The LH79520 transitions from the Stop1 mode to the Active mode when an interrupt is received.

When transitioning from the Active mode to the Stop1 mode, the LH79520 Power Management system automatically performs the following sequence:

- Acquire control of the AHB, to ensure that all transactions are completed.
- Ensure that all SDRAM devices are placed in the self-refresh mode of operation.
- Halt all clocks, disable the PLL and deassert the CLKEN output signal.
- Wait for IRQ or FIQ to be asserted (which will return the LH79520 to the Active mode).

When an IRQ or FIQ occurs, the LH79520 returns to the Active mode, restarts the output clocks (which may require a delay for the PLL to reacquire lock, if the PLL is in use), resumes SDRAM refresh and then cedes control of the AHB.

7.13.5 Stop2 Mode

The Stop2 mode halts all system clocks and disables both the PLL and the internal oscillator that feeds it. If the 32.768 kHz internal oscillator is in use, it will remain active. The LH79520 transitions from the Stop2 mode to the Active mode when an interrupt is received.

When transitioning from the Active mode to the Stop2 mode, the LH79520 Power Management system automatically performs the following sequence:

- Acquire control of the AHB, to ensure that all transactions are completed.
- Ensure that all SDRAM devices are placed in the self-refresh mode of operation.
- Halt all clocks, disable the PLL, disable the 14.7456 MHz oscillator and deassert the CLKEN output signal.
- Wait for IRQ or FIQ to be asserted (which will return the LH79520 to the Active mode).

When an IRQ or FIQ occurs, the LH79520 returns to the Active mode, restarts the output clocks (which may require a delay for the PLL to reacquire lock if the PLL is in use), resumes SDRAM refresh and then cedes control of the AHB.

7.14 RCPC Programmer's Model

The Register Base Address for the all of the LH79520 Reset, Clock Generation, and Power Control registers is:

RCPCBase: 0xFFFE2000

All registers in the RCPC must be accessed as a full word. The RCPC does not support byte and half-word writes.

7.15 RCPC Register Summary

Table 7-4 summarizes the programmable registers that configure the LH79520 Reset, Clock Generation and Power Control (RCPC) systems. Each register is treated in more detail on the following pages.

All RCPC programmable registers are reset to RW (Read-Write) but can be set to RO (Read-Only) by software. Table 7-4 presents a complete list of the affected registers. Some registers in this list are explained in other chapters of this User's Guide but included here in order to maintain a complete list of the registers which can potentially be write locked. See the RCPCCtrl register description for additional details regarding the WRT-LOCK bit field.

Table 7-4. RCPC Register Summary

ADDRESS	REGISTER NAME	DESCRIPTION
RCPCBase + 0x000	RCPCCtrl	RCPC Control
RCPCBase + 0x004	IDString	Chip ID
RCPCBase + 0x008	RCPCRemapCtrl	Memory Map Control (See Section 3.13.1)
RCPCBase + 0x00C	SoftReset	Soft Reset
RCPCBase + 0x010	ResetStatus	Reset Status
RCPCBase + 0x014	ResetStatusClr	Reset Status Clear
RCPCBase + 0x018	HCLKPrescale	HCLK Prescale
RCPCBase + 0x01C	CpuClkPrescale	ARM Core Clock Prescale
RCPCBase + 0x020	///	Reserved
RCPCBase + 0x024	PeriphClkCtrl	Peripheral Clock Control
RCPCBase + 0x028	PeriphClkCtrl2	Peripheral Clock Control 2
RCPCBase + 0x02C	AHBClkCtrl	AHB Clock Control
RCPCBase + 0x030	PeriphClkSel	Peripheral Clock Select
RCPCBase + 0x034	PeriphClkSel2	Peripheral Clock Select 2
RCPCBase + 0x038	PWM0Prescale	PWM 0 Prescale
RCPCBase + 0x03C	PWM1Prescale	PWM 1 Prescale
RCPCBase + 0x040	LCDClkPrescale	LCD Clock Prescale
RCPCBase + 0x044	SSPClkPrescale	SSP Clock Prescale
RCPCBase + 0x080	IntConfig	Interrupt Configuration (See Section 9.5.1)
RCPCBase + 0x084	IntClear	Interrupt Clear (See Section 9.8.6)
RCPCBase + 0x088	CoreClkConfig	ARM Core Clock Configuration (See Section 3.13.2)

7.16 RCPC Register Descriptions

This section of this User's Guide lists the registers that configure the LH79520 Reset, Clock Generation and Power Control (RCPC) system, and describe their function.

7.16.1 RCPC Control Register

The RCPCCtrl register enables the PLL and handles write locking, clock selection and power modes.

Table 7-5. RCPCCtrl Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///						WRTLOCK	///	CLKSEL	OUTSEL		PWRDOWNSEL			EX	EP
RESET CLKINSEL PIN HIGH	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0
RESET CLKINSEL PIN LOW	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	1
RW	RW	RW	RW	RW	RW	RW	Note1	RW	RO	RW		RW	RW	RW	Note 2	Note 2
ADDR	RCPCBase + 0x000															

NOTES:

1. The WRTLOCK bit field can render all other bits in this register, and all bits in all other RCPC registers, RO (Read-Only). The WRTLOCK bit field's value at reset is 1 (Read-Write).
2. The LH79520 is designed to protect its system clock-source from errors. If the system is using a 14.7456 MHz crystal and the internal Phase-Locked Loop (PLL) to generate the system clocks (HCLK, HCLK_CPU, FCLK), then these bits cannot be changed because doing so would defeat the signal that creates the clocks. If the system is using an external clock signal to generate the clocks, then this bit can be changed (and the PLL will be OFF at reset). Additionally, when an external clock signal is used to generate the clocks, the UARTs are disconnected from this clock source.
3. If the CLKINSEL pin was latched LOW at reset then EX and EP are RO (Read-Only).
4. If the CLKINSEL pin was latched HIGH at reset then EX and EP are RW (Read/Write).
5. This register's Reset definition will be different for different levels of silicon revision. See Table 7-6

Table 7-6. Reset Definition

ADDRESS	SILICON REV.	CLOCK SEL
0x63	A.0	0
0xE0	A.0	1
0x263	B.1	0
0x2E0	B.1	1

Table 7-7. RCPCCtrl Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:10	///	**Reserved** Write the Reset Value.
9	WRTLOCK	**Write Lock** 0 = All RCPC registers are RO (write-protected) except this bit field in this register. 1 = All RCPC registers are RW (write-enabled) (reset value).
8	///	**Reserved** Write the Reset Value.
7	CLKSEL	**HCLK Source Select Status** Reflects the status of the CLKINSEL pin at last reset. Will be set to '1' following a reset if the CLKINSEL pin is HIGH at reset. 0 = HCLK is derived from the 309.6576 MHz output of the PLL 1 = HCLK is derived from the signal at the CLKIN input pin
6:5	OUTSEL	**CLKOUT Source Select** 0b00 = 14.7456 MHz signal from the internal oscillator 0b01 = 309.6576 MHz signal from the PLL 0b10 = FCLK 0b11 = HCLK
4:2	PWRDWNSEL	**Power Down Mode Select** 0b000 = Active Mode 0b001 = Standby Mode 0b010 = Sleep Mode 0b011 = Stop1 Mode 0b100 = Stop2 Mode Other bit combinations are undefined and should not be written. These bits will always read as '000' because they are automatically cleared to '0' within one clock-cycle after they are written.
1	EX*	**Enable Internal Crystal Oscillator** Always readable; writable only if the CLKSEL bit field (bit 7 of this register) is '1'. 0 = Disable the 14.7456 MHz internal crystal oscillator 1 = Enable the 14.7456 MHz internal crystal oscillator
0	EP*	**Enable Phase-Locked Loop (PLL)** Always readable; writable only if the CLKSEL bit field (bit 7 of this register) is '1'. 0 = Disable PLL 1 = Enable PLL

NOTE: *If the external clock-select pin, CLKINSEL, is latched HIGH at reset then the CLKSEL bit will be set to '1' and the EX and EP bit fields will be RO (Read-Only).

7.16.2 Chip ID Register

The Chip ID register is Read Only, and returns the value of the part number.

Table 7-8. IDString Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0
RW	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
ADDR	RCPCBase + 0x004															

Table 7-9. IDString Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:15	///	**Reserved** Read Only
14:0	ID	**Chip ID** Returns a value, depending upon the silicon version: A.0 Silicon: 0x5200 B.1 Silicon: 0x5201

7.16.3 Soft Reset Register

Writes of 'key' values to the SoftReset register can reset the LH79520 system, the internal RTC, or both.

This register provides a mechanism for software to assert a global system reset and/or reset the RTC. Software can generate two different types of resets by writing the 'key' values listed in Table 7-8 and Table 7-9.

See Figure 7-2 in Section 7.3 and the descriptions of the 'ResetStatus' and 'ResetStatusClr' registers for more information.

Table 7-10. SoftReset Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	SRVAL															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	RCPCBase + 0x00C															

Table 7-11. SoftReset Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:16	///	**Reserved** Write the Reset Value.
15:0	SRVAL	Write one of two possible 'Key' values: 0b1101111010101101 = 0xDEAD 0b1101111010101100 = 0xDEAC Writes of other values will have no effect. Value returned on a read will be unpredictable.

Table 7-12. Key Value Resets

FIELD NAME	'KEY' VALUE	GLOBAL RESET	RTC RESET
SRVAL	0b1101111010101101 = 0xDEAD	Asserted	Asserted
	0b1101111010101100 = 0xDEAC	Asserted	Not Asserted
	Any other value	Not Asserted	Not Asserted

7.16.4 Reset Status Register

Software can read the ResetStatus register to ascertain the present status of the LH79520 reset system.

This register is read-only. Reads of this register return the reset status of the LH79520 MCU. On external reset, the EXT bit is set to '1' and the WDTO bit is cleared to '0'. On WDT time-out, only the WDTO bit is set to '1'. Once set, either bit will remain set until it is cleared by a write to the ResetStatusClr register. See the descriptions of the SoftReset and ResetStatusClr registers for more information.

Table 7-13. ResetStatus Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///														WDTO	EXT
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
ADDR	RCPCBase + 0x010															

Table 7-14. ResetStatus Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:2	///	**Reserved** Reads are unpredictable.
1	WDTO	**WDT Timeout Status** 0 = No WDT time-out and reset has occurred since this bit was last cleared 1 = At least one WDT time-out and reset has occurred since this bit was last cleared.
0	EXT	**External Reset Status** 0 = No External Reset has occurred since this bit was last cleared 1 = At least one External Reset has occurred since this bit was last cleared

7.16.5 Reset Status Clear Register

Writes to the ResetStatusClr register will clear the bits in the ResetStatus register which indicate the status of the LH79520 reset system.

A '1' written to a bit field in this register will cause the corresponding bit in the Reset Status register to be cleared. A '0' written to a bit field in this register will have no effect on the corresponding bit in the Reset Status register. This register is write-only.

See the descriptions of the SoftReset and ResetStatus registers for more information.

Table 7-15. ResetStatusClr Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///														TOCLR	EXTCLR
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
ADDR	RCPCBase + 0x014															

Table 7-16. ResetStatusClr Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:2	///	**Reserved** Write-only. Value returned on a read will be unpredictable.
1	TOCLR	Writing a '1' to this bit clears ResetStatus:WDTO. Reads of this bit are unpredictable.
0	EXTCLR	Writing a '1' to this bit clears ResetStatus:WDTO. Reads of this bit are unpredictable.

7.16.6 HCLK Prescale Register

Values written to the HCLKPrescale register select the frequency-division ratio that produces the HCLK signal.

HCLK is the signal that drives the AHB. The value written to this register determines the ratio by which the source of the HCLK signal is divided, to produce HCLK. All valid values are listed in Table 7-18. The hardware will not accept writes of invalid values.

The HCLK pre-scaling system, shown in Figure 7-13, divides the (internal or external) input clock signal by one of several different ratios to produce the HCLK signal. Also see Figures 7-4 and 7-8 for an overview of the LH79520 clock distribution system.

Table 7-17. HCLKPrescale Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///												HCPSVAL			
RESET	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	RCPCBase + 0x018															

Table 7-18. HCLKPrescale Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:4	///	**Reserved** Write the Reset Value.
3:0	HCPSVAL	0b0011 = Divider Value = 6, *f*_input/6 0b0100 = Divider Value = 8, *f*_input/8 (values in the range of 0b0100 to 0b1111 are valid; see Table 7-1) 0b1111 = Divider Value = 30, *f*_input/30

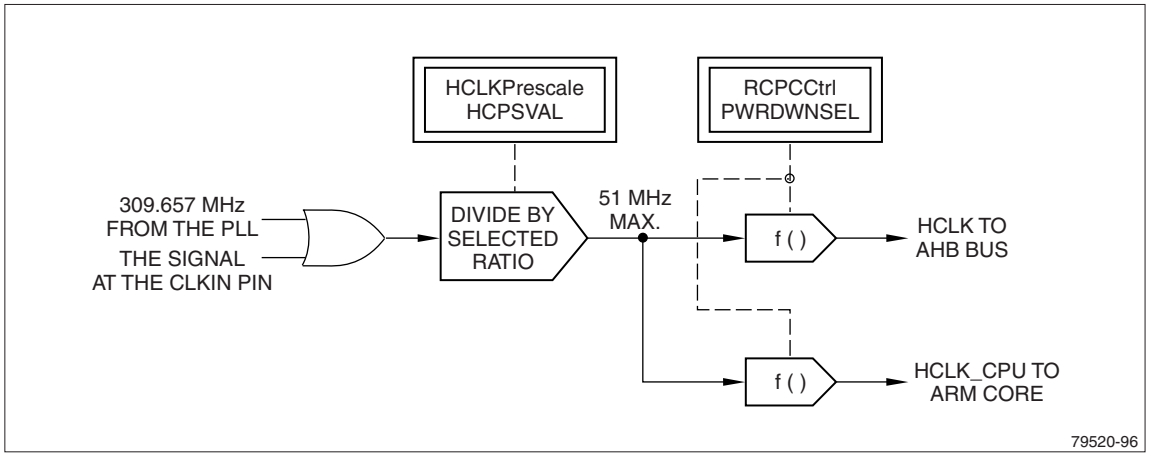


Figure 7-13. HCLK Prescaling

7.16.7 CPU Clock Prescale Register

Values written to the CPUClkPrescale register select the frequency-division ratio that produces the FCLK signal.

FCLK is the signal that clocks the ARM core. The value written to this register determines the ratio by which the source of the FCLK signal is divided, to produce FCLK. All valid values are listed in Table 7-20; all other values are invalid. The hardware will not accept writes of invalid values.

The FCLK pre-scaling system, shown in Figure 7-14, divides the (internal or external) input clock signal by one of several different ratios to produce the FCLK signal. Also see Figures 7-4 and 7-8 for an overview of the entire LH79520 clock distribution system.

Table 7-19. CPUClkPrescale Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///												CCPSVAL			
RESET	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	RCPCBase + 0x01C															

Table 7-20. CPUCLKPrescale Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:4	///	**Reserved** Write the Reset Value.
3:0	CCPSVAL	0b0010 = Divider Value = 4, *f*_input/4 0b0011 = Divider Value = 6, *f*_input/6 0b0100 = Divider Value = 8, *f*_input/8 (values in the range of 0b0100 to 0b1111are valid; see Table 7-1) 0b1111 = Divider Value = 30, *f*_input/30

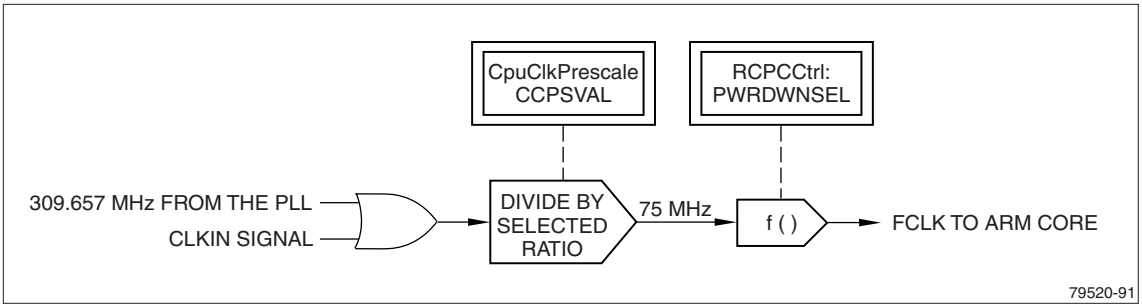


Figure 7-14. FCLK Prescaling

7.16.8 Peripheral Clock Control Register

The PeriphClkCtrl register activates or deactivates the clocks fed to various LH79520 peripherals.

The values written to this register enable or disable the clock signals to the RTC, PWMs, Timers, and UARTs. See the description of the PeriphClkSel register for more information regarding which clock is routed to each peripheral. See Figures 7-5, 7-8, 7-9, and 7-11 for information regarding the clocks controlled by this register.

Writing a '1' to a bit field in this register inhibits the associated clock, writing a '0' enables it. Table 7-22 lists the bit fields and describes their function.

Table 7-21. PeriphClkCtrl Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///						RTC	PWM1	PWM0	///	T23	T01	///	U2	U1	U0
RESET	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	SBO	RW	RW	RW
ADDR	RCPCBase + 0x024															

Table 7-22. PeriphClkCtrl Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:10	///	**Reserved** Write the Reset Value.
9	RTC	1 = Disable RTC clock 0 = Enable RTC clock
8	PWM1	1 = Disable PWM1 clock 0 = Enable PWM1 clock
7	PWM0	1 = Disable PWM0 clock 0 = Enable PWM0 clock
6	///	**Reserved** Write the Reset Value.
5	T23	1 = Disable the clock to Timer2 and Timer3 0 = Enable the clock Timer2 and Timer3
4	T01	1 = Disable the clock to Timer0 and Timer1 0 = Enable the clock to Timer0 and Timer1
3	///	**Reserved** Write the Reset Value.
2	U2	1 = Disable UART2 clock 0 = Enable UART2 clock
1	U1	1 = Disable UART1 clock 0 = Enable UART1 clock
0	U0	1 = Disable UART0 clock 0 = Enable UART0 clock

The LH79520 contains four Timers, for a total of four devices: Timer0, Timer1, Timer2, and Timer3. The RCPC can generate two clock signals for the four Timers.

The PeriphClkCtrl:CT1 bit field controls the clock signal fed to Timer0 and Timer1 (the first and second Timers). The PeriphClkCtrl:CT2 bit field controls the clock signal routed to Timer2 and Timer3, (the third and fourth Timers).

Of the four timers, only Timer3 has an output available at a pin on the LH79520 package. The remaining three Timers can be cascaded, can generate interrupts and can be accessed by software, but have no physical output pin.

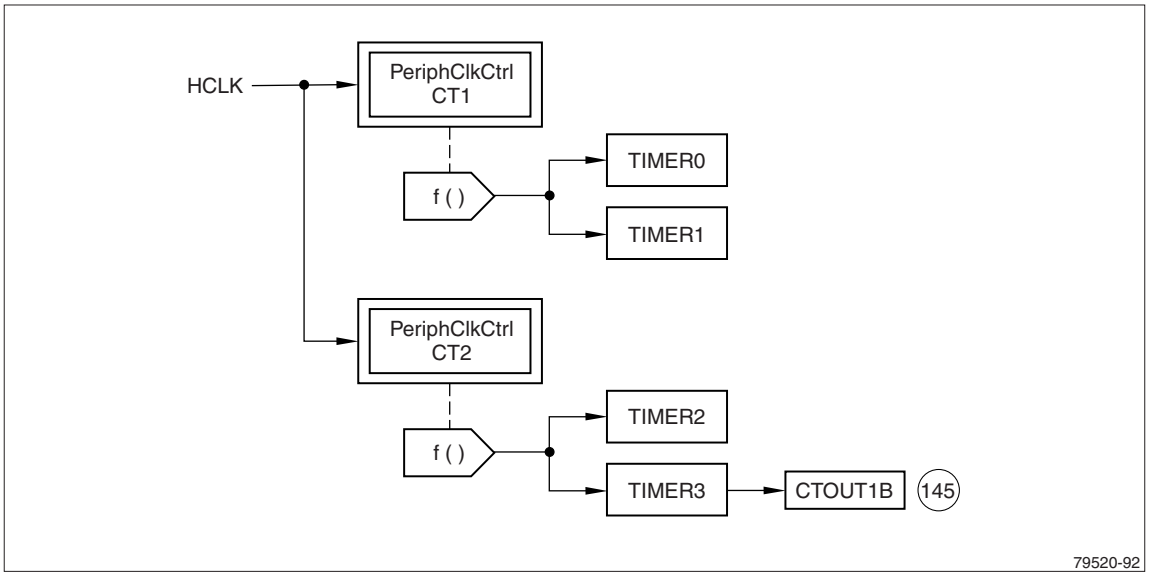


Figure 7-15. Timer Clocks and Output

7.16.9 Peripheral Clock Control Register 2

The PeriphClkCtrl2 register activates or deactivates clocks fed to the LH79520 CLCDC and the SSP peripheral.

The values written to this register enable or disable the clock signals fed to the Color LCD Controller and the Synchronous Serial Port peripheral.

Writing a '1' to a bit field in this register inhibits the associated clock, writing a '0' enables it. Table 7-24 lists the bit fields and describes their function.

See Figures 7-6 and 7-7, Chapter 11 – Color LCD Controller, and Chapter 16 – Synchronous Serial Port (SSP) for more information.

Table 7-23. PeriphClkCtrl2 Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///														SSPCLK	LCDCLK
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	RCPCBase + 0x028															

Table 7-24. PeriphClkCtrl2 Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:2	///	**Reserved** Write the Reset Value.
1	SSPCLK	1 = Disable SSP clock (reset) 0 = Enable SSP clock
0	LCDCLK	1 = Disable CLCDC clock (reset) 0 = Enable CLCDC clock

7.16.10 AHB Clock Control Register

The AHBClkCtrl register activates or deactivates clocks for devices on the AHB.

The values written to the bit fields in this register enable or disable HCLK signals fed to lthe SDRAM Controller (SDRC) and the DMA Controller (DMAC). See Section 7.9 and Figures 7-1, 7-4, and 7-8 for more information about these two signals.

Writing a '1' to a bit field in this register inhibits the associated clock, writing a '0' enables it. Table 7-26 lists the bit fields and describes their function.

Table 7-25. AHBClkCtrl Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///													///	SDC	DMA
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	RCPCBase + 0x02C															

Table 7-26. AHBClkCtrl Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:2	///	**Reserved** Write the Reset Value.
1	SDC	1 = Disable the HCLK signal fed to the SDRC (reset) 0 = Enable the HCLK signal fed to the SDRC
0	DMA	1 = Disable the HCLK signal fed to the DMAC (reset) 0 = Enable the HCLK signal fed to the DMAC

7.16.11 Peripheral Clock Select Register

The PeriphClkSel register selects which clock signals are fed to the RTC and UARTs.

Table 7-25 describes the bit fields in the PeriphClkSel register. Figure 7-16 summarizes Peripheral Clock selection and control.

See Figure 7-9 and Section 7-11 for more information about the generation of the signal that drives the on-chip Real-Time Clock (RTC) peripheral. See Chapter 17 – Universal Asynchronous Receiver/Transmitters (UARTs) and Figure 7-11, for more information on how the UART clocks are generated and controlled.

Table 7-27. PeriphClkSel Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///							RTC		///				U2	U1	U0
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	RCPCBase + 0x030															

Table 7-28. PeriphClkSel Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:9	///	**Reserved** Write the Reset Value.
8:7	RTC	**RTC Clock Source** 0b00 = 1 Hz clock (the 32 kHz clock divided by 32,768; the reset condition) 0b01 = Reserved — do not write this value 0b10 = Reserved — do not write this value 0b11 = 32 kHz clock
6:3	///	**Reserved** Write the Reset Value.
2	U2	**UART2 Clock Source** Do not set this bit to '1' if the LH79520 is running from an external oscillator. See Section 7.5. 0 = Crystal oscillator output 1 = CLKIN
1	U1	**UART1 Clock Source** Do not set this bit to '1' if the LH79520 is running from an external oscillator. See Section 7.5. 0 = Crystal oscillator output 1 = CLKIN
0	U0	**UART0 Clock Source** Do not set this bit to '1' if the LH79520 is running from an external oscillator. See Section 7.5. 0 = Crystal oscillator output 1 = CLKIN

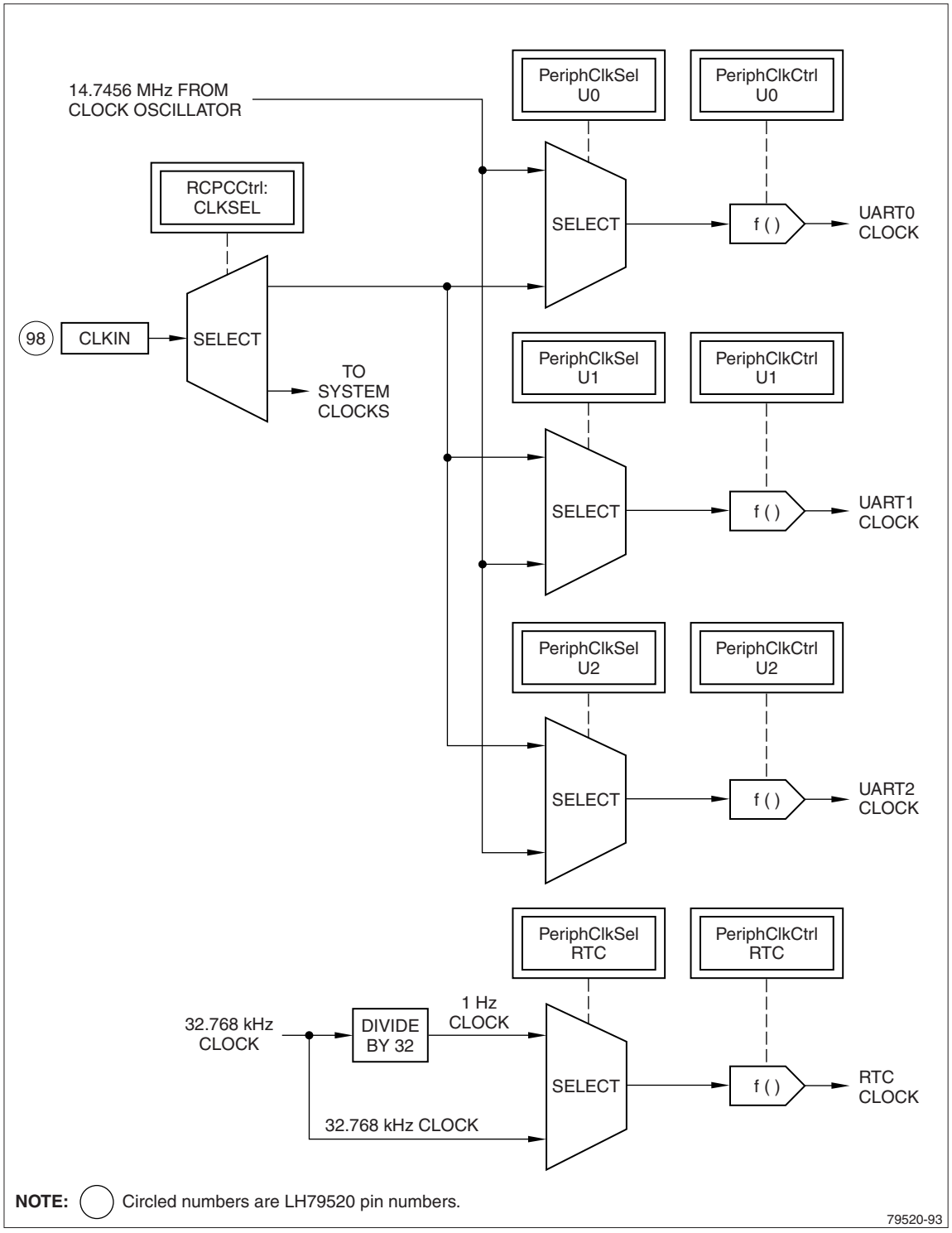


Figure 7-16. Peripheral Clock Selection and Control

7.16.12 Peripheral Clock Select Register 2

The PeriphClkSel2 register determines which clock signal is fed to the Color LCD Controller (CLCDC).

Table 7-30 lists the bit fields in the PeriphClkSel2 register and describes their functions.

See Figures 7-6 and 7-7 and the Chapter 11 – Color LCD Controller for more information.

Table 7-29. PeriphClkSel2 Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///															LCSRC
TYPE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	RCPCBase + 0x034															

Table 7-30. PeriphClkSel2 Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:1	///	**Reserved** Write the Reset Value.
0	LCSRC	**LCD Clock Source** Selects which clock signal is fed to the CLCDC. Do not change clock sources to the CLCDC when it is enabled and operating. The resultant loss of internal synchronization in the CLCDC can cause the system to hang. 0 = Select HCLK (reset) 1 = Select the signal at the LCDCLKIN pin

7.16.13 PWM0 Prescale Register

The PWM0Prescale register pre-scales the clock routed to Pulse Width Modulator 0 (PWM0).

The 15-bit value written to this register selects the amount by which the PCLK signal (locked to HCLK in the LH79520) is divided to produce the clock that drives PWM0. The available division-ratios are listed in Table 7-32.

Table 7-31. PWM0Prescale Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///	PRESCALEVAL														
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	RCPCBase + 0x038															

Table 7-32. PWM0Prescale Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:15	///	**Reserved** Write the Reset Value.
14:0	PRESCALEVAL	0b000000000000000 = Do not write 0b000000000000001 = Divider Value = 2, PCLK/2 (reset value) 0b000000000000010 = Divider Value = 4, PCLK/4 0b000000000000011 = Divider Value = 6, PCLK/6 0b000000000000100 = Divider Value = 8, PCLK/8 (values in this range are valid) 0b111111111111111 = Divider Value = 65534, PCLK/65534

NOTE: The PRESCALEVAL field cannot be programmed to obtain a Divider Value of 1.

7.16.14 PWM1 Prescale Register

The PWM1Prescale register pre-scales the clock routed to Pulse Width Modulator 1 (PWM1).

The 15-bit value written to this register selects the amount by which the PCLK signal (locked to HCLK in the LH79520) is divided to produce the clock that drives PWM1. The available division-ratios are listed in Table 7-34.

Table 7-33. PWM1Prescale Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///	PRESCALEVAL														
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	RCPCBase + 0x03C															

Table 7-34. PWM1Prescale Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:15	///	**Reserved** Write the Reset Value.
14:0	PRESCALEVAL	0b000000000000000 = Do not write 0b000000000000001 = Divider Value = 2, PCLK/2 (reset value) 0b000000000000010 = Divider Value = 4, PCLK/4 0b000000000000011 = Divider Value = 6, PCLK/6 0b000000000000100 = Divider Value = 8, PCLK/8 (values in this range are valid) 0b111111111111111 = Divider Value = 65534, PCLK/65534

NOTE: The PRESCALEVAL field cannot be programmed to obtain a Diver Value of 1.

7.16.15 LCD Clock Prescale Register

The LCDClkPrescale register pre-scales the clock routed to the Color LCD Controller (CLCDC).

The 8-bit value written to this register selects the amount by which the source signal (*f*_source) selected by PreiphClkSel2:LCDCLK (the LCDCLK bit field in the PeriphClkSel2 register) is divided to produce the clock that drives the Color LCD Controller (CLCDC). The available division-ratios are listed in Table 7-36.

See Figure 7-6 and Section 7.7 for more information about the selection and generation of the CLCDC clock signal.

Table 7-35. LCDClkPrescale Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								LCDPSVAL							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	RCPCBase + 0x040															

Table 7-36. LCDClkPrescale Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:8	///	**Reserved** Write the Reset Value.
7:0	LCDPSVAL	0b00000000 = Divider Value = 1, *f*_source/1 0b00000001 = Divider Value = 2, *f*_source/2 0b00000010 = Divider Value = 4, *f*_source/4 0b00000100 = Divider Value = 8, *f*_source/8 0b00001000 = Divider Value = 16, *f*_source/16 0b00010000 = Divider Value = 32, *f*_source/32 0b00100000 = Divider Value = 64, *f*_source/64 0b01000000 = Divider Value = 128, *f*_source/128 0b10000000 = Divider Value = 256, *f*_source/256 Other values are invalid and should not be written

7.16.16 SSP Clock Prescale Register

The SSPClkPrescale register pre-scales the clock fed to the LH79520 Synchronous Serial Port (SSP) peripheral.

The 8-bit value written to this register selects the amount by which HCLK (enabled by PeriphClkCtrl2: SSPCLK) is divided to produce the clock that drives the SSP. The available division-ratios are listed in Table 7-38.

See Figure 7-7, and also Chapter 16 – Synchronous Serial Port (SSP), for more information about the selection and generation of the SSP clock signal.

Table 7-37. SSPClkPrescale Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								SSPPSVAL							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	RCPCBase + 0x044															

Table 7-38. SSPClkPrescale Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:8	///	**Reserved** Write the Reset Value.
7:0	SSPPSVAL	0b00000000 = Divider Value = 1, HCLK/1 0b00000001 = Divider Value = 2, HCLK/2 0b00000010 = Divider Value = 4, HCLK/4 0b00000100 = Divider Value = 8, HCLK/8 0b00001000 = Divider Value = 16, HCLK/16 0b00010000 = Divider Value = 32, HCLK/32 0b00100000 = Divider Value = 64, HCLK/64 0b01000000 = Divider Value = 128, HCLK/128 0b10000000 = Divider Value = 256, HCLK/256 Values not listed here are invalid and must not be written

Chapter 8
I/O Control and Multiplexing (IOCON)

Many of the pins of the LH79520 package may be multiplexed with as many as three different functions. The multiplexing is controlled by the Input/Output Control (IOCON) registers presented in this chapter. Figure 8-1 shows a conceptual diagram of the multiplexing scheme.

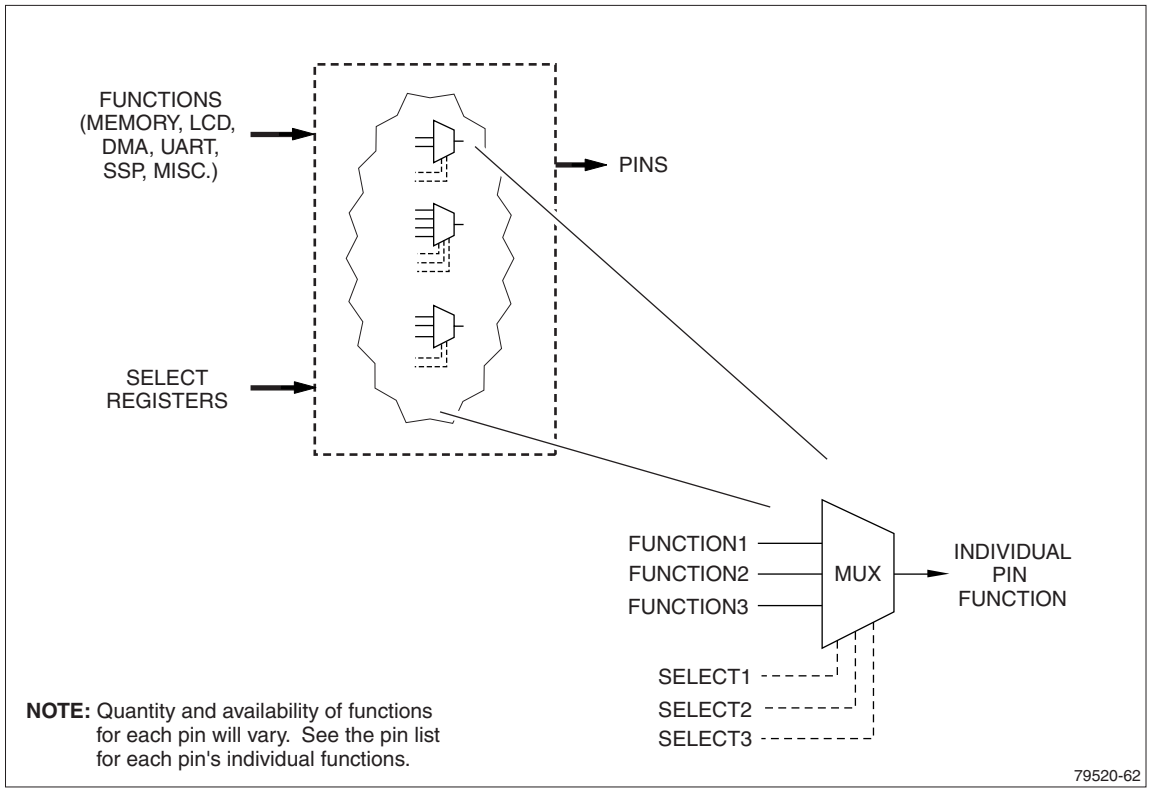


Figure 8-1. IOCON Pin-Function Multiplexing

8.1 Theory of Operation

Each LH79520 pin can perform only one function at a time. The pin functions at reset have been carefully chosen to facilitate the most common boot requirements. Most pin functions are selectable in software but some pin functions may rely on the condition of other pins at reset. Table 1-3, in Chapter 1 – Introduction, lists each pin and its possible functions. Pin functions at reset are also presented in this chapter, in the register tables.

Hardware designers must be aware of each pin's function at reset, to ensure correct wiring. Programmers must ensure that the software selects the correct pin function during initialization. Whenever software enables an on-board system (such as a UART), the software should also ensure that the correct pin functions are selected in these registers. Where the INT5 and DREQ1 functions are concerned, software should avoid enabling both the INT5 and DREQ1 functions simultaneously.

8.2 IOCON Programmer's Model

The Register Base Address for all of the LH79520 IOCON registers is:

IOCONBase: 0xFFFE5000

All registers in the IOCON must be accessed as a full word. The IOCON does not support byte and half-word writes.

8.3 IOCON Register Summary

Table 8-1 summarizes the programmable IOCON registers. Tables 8-3 through 8-14 explain the individual registers and their bit fields in detail.

Table 8-1. IOCON Register Summary

ADDRESS	REGISTER NAME	DESCRIPTION
IOCONBase + 0x000	MemMux	Memory Interface Multiplexing
IOCONBase + 0x004	LCDMux	LCD Signal Multiplexing
IOCONBase + 0x008	MiscMux	Miscellaneous Pin Multiplexing
IOCONBase + 0x00C	DMAMux	DMA Multiplexing
IOCONBase + 0x010	UARTMux	UART Multiplexing
IOCONBase + 0x014	SSPMux	SSP Multiplexing
IOCONBase + 0x018	///	Reserved — do not write

8.4 IOCON Register Descriptions

8.4.1 Memory Interface Multiplexing Register

The MemMux register reconfigures the Memory Interface signal pins as General Purpose I/O pins.

Table 8-3 and Table 8-4 explain how the bit fields in the MemMux register affect the signals available at various LH79520 pins.

Table 8-2. MemMux Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///		PIN34	PIN35	PIN41	PIN42	PIN43	PIN44	UPPER16	PIN101	PIN102	PIN104	PIN105	PIN106	PIN112: PIN109	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	IOCONBase + 0x000															

Table 8-3. MemMux Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:14	///	**Reserved** Write '0'. Read '0'
13	PIN34	1 = Pin 34 functions as nBLE3 0 = Pin 34 functions as PH7 (reset)
12	PIN35	1 = nBLE2 0 = PH6 (reset)
11	PIN41	1 = nCS6 0 = PH5 (reset)
10	PIN42	1 = nCS5 0 = PH4 (reset)
9	PIN43	1 = nCS4 0 = PH3 (reset)
8	PIN44	1 = nCS3 0 = PH2 (reset)
7	UPPER16	**Enable Upper 16 Data Bus bits** Setting this bit affects pins 67:50, enabling the upper 16 bits of the data bus (D31:16) onto these pins. 1 = Pins 67 through 50 function as data 0 = Pins 67 through 50 function as GPIO (reset)
6	PIN101	1 = SDCLK 0 = PF0 (reset)
5	PIN102	1 = SDCKE 0 = PE7 (reset)
4	PIN104	1 = nDCS1 0 = PE6 (reset)
3	PIN105	1 = nDCS0 0 = PE5 (reset)
2	PIN106	1 = nSDWE 0 = PE4 (reset)
1:0	PIN112:PIN109	Configure pins 112 through 109. See Table 8-4. 0b00 = reset

Table 8-4 shows how MemMux bits 1 and 0 select the functions available on LH79520 pins 109 through 112.

Table 8-4. Functions Selected by MemMux Bits 1:0

BIT1:0	SIGNAL ON PIN112	SIGNAL ON PIN111	SIGNAL ON PIN110	SIGNAL ON PIN109
0b11	DQM0	DQM1	DQM2	DQM3
0b10	PE0	PE1	PE2	PE3
0b01	DQM0	DQM1	PE2	PE3
0b00 (reset)	PE0	PE1	PE2	PE3

8.4.2 LCD Interface Multiplexing Register

The LCDMux register reconfigures the General Purpose I/O pins as LCD panel interface signal pins.

At reset, the pins controlled by the LCDMux register function as General Purpose I/O pins. Software must program this register to enable the appropriate LCD interface functions. Table 8-6 explains the bit fields in the LCDMux register.

For information concerning the various LCD technologies (STN, TFT and HR-TFT) and the associated LCD data formats, see Chapter 11 – Color LCD Controller.

Table 8-5. LCDMux Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///			PIN114	PIN115	PIN116	PIN117	PIN118	PIN119		PIN121	PIN122	PIN123	PIN124	PIN129	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	PIN130	PIN131		PIN132	PIN133	PIN134	PIN135		PIN137		PIN139		PIN140	PIN141	PIN142	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	IOCONBase + 0x004															

Table 8-6. LCDMux Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:29	///	**Reserved** Write the Reset value.
28	PIN114	1 = LCDVD11 0 = INT7 (reset)
27	PIN115	1 = LCDVD10 0 = INT6 (reset)
26	PIN116	1 = LCDVD9 0 = PD7 (reset)
25	PIN117	1 = LCDVD8 0 = PD6 (reset)
24	PIN118	1 = LCDVD7 0 = PD5 (reset)
23:22	PIN119	0b11 = Reserved 0b10 = Configure the pin to function as LCDPS (AD-TFT, HR-TFT panels) 0b01 = Configure the pin as LCDVD6 (STN panels) 0b00 = Configure the pin to function as PD4 (reset)
21	PIN121	1 = LCDVD5 0 = PD3 (reset)
20	PIN122	1 = LCDVD4 0 = PD2 (reset)
19	PIN123	1 = LCDVD3 0 = PD1(reset)
18	PIN124	1 = LCDVD2 0 = PD0 (reset)
17:16	PIN129	0b11 = Reserved 0b10 = Configure the pin to function as LCDSPS (AD-TFT, HR-TFT panels) 0b01 = Configure the pin to function as LCDFP (STN and TFT panels) 0b00 = Configure the pin to function as PC7 (reset)
15	PIN130	1 = LCDVD17 0 = PC6 (reset)
14:13	PIN131	0b11 = Reserved 0b10 = Configure the pin to function as LCDLP (AD-TFT, HR-TFT panels) 0b01 = Configure the pin to function as LCDLP (TFT and STN panels) 0b00 = Configure the pin to function as PC5 (reset)
12	PIN132	1 = LCDVD16 0 = PC4 (reset)
11	PIN133	1 = LCDDCLK 0 = PC3 (reset)
10	PIN134	1 = LCDCLKIN 0 = PC2 (reset)
9:8	PIN135	0b11 = Reserved 0b10 = Configure the pin to function as LCDCLS (AD-TFT, HR-TFT panels) 0b01 = Configure the pin to function as LCDVDDEN (STN and TFT panels) 0b00 = Configure the pin to function as PC1 (reset)
7:6	PIN137	0b11 = Reserved 0b10 = Configure the pin to function as LCDSPL (AD-TFT, HR-TFT panels) 0b01 = Configure the pin to function as LCDENAB (STN and TFT panels) 0b00 = Configure the pin to function as PC0 (reset)
5:4	PIN139	0b11 = Reserved 0b10 = Reserved 0b01 = Configure the pin to function as LCDVD15 0b00 = Configure the pin to function as PB7 (reset)
3	PIN140	1 = LCDVD14 0 = PB6 (reset)
2	PIN141	1 = LCDVD13 0 = PB5 (reset)
1:0	PIN142	0b11 = Reserved 0b10 = Configure the pin to function as LCDREV (AD-TFT, HR-TFT panels) 0b01 = Configure the pin to function as LCDVD12 (STN and TFT panels) 0b00 = Configure the pin to function as PB4 (reset)

8.4.3 Miscellaneous Pin Multiplexing Register

The MiscMux register reconfigures assorted LH79520 pins to their secondary functions. Table 8-8 explains the bit fields in the MiscMux register.

Table 8-7. MiscMux Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///					PIN98	PIN99	PIN144	PIN145	PIN150	PIN151	PIN152	PIN153	PIN155	PIN156	PIN157
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	IOCONBase + 0x008															

Table 8-8. MiscMux Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:11	///	**Reserved** Write the Reset value.
10	PIN98	1 = UARTCLK 0 = CLKIN (reset)
9	PIN99	1 = CLKEN 0 = PF1(reset)
8	PIN144	1 = nWAIT 0 = INT5/DREQ1 (reset) Although this bit field must be cleared to '0' to enable the use of either the INT5 or the DREQ1 function, software should avoid enabling both the INT5 and DREQ1 functions simultaneously. See Chapter 9 for more information concerning INT5. See Chapter 10 for more information concerning DREQ1.
7	PIN145	1 = DACK1 0 = CTOUT1B (reset)
6	PIN150	1 = PWM0 0 = INT4 (reset)
5	PIN151	1 = PWMSYNC0 0 = INT3 (reset)
4	PIN152	1 = INT2 0 = PB0 (reset)
3	PIN153	1 = INT1 0 = PA7 (reset)
2	PIN155	1 = INT0 0 = PA6 (reset)
1	PIN156	1 = CLKOUT 0 = PA5 (reset)
0	PIN157	1 = DEOT1 0 = PWM1 (reset)

8.4.4 DMA Interface Multiplexing Register

The DMAMux register reconfigures the DMA signal pins to their secondary functions. Table 8-10 explains the DMAMux register's bit fields.

Table 8-9. DMAMux Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///													PIN146	PIN147	PIN148
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	IOCONBase + 0x00C															

Table 8-10. DMAMux Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:3	///	**Reserved** Write the Reset value.
2	PIN146	1 = DREQ0 0 = PB3 (reset)
1	PIN147	1 = nDACK0 0 = PB2 (reset)
0	PIN148	1 = DEOT0 0 = PB1 (reset)

8.4.5 UART Interface Multiplexing Register

The UARTMux register reconfigures the UART signal pins to their secondary functions. Table 8-12 explains the UARTMux register bit fields.

Table 8-11. UARTMux Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///												PIN159	PIN160	PIN162	PIN163
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	IOCONBase + 0x010															

Table 8-12. UARTMux Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:4	///	**Reserved** Write the Reset value.
3	PIN159	1 = U1TxD 0 = PA4 (reset)
2	PIN160	1 = U1RxD 0 = PA3 (reset)
1	PIN162	1 = U0TxD 0 = U0IRTxA (reset)
0	PIN163	1 = U0RxD 0 = U0IRRxA (reset)

8.4.6 SSP Interface Multiplexing Register

The SSPMux register reconfigures the SSP interface pins to their secondary functions. Table 8-14 explains the SSPMux register bit fields.

Table 8-13. SSPMux Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///											PIN164	PIN165	PIN166	PIN167	PIN169
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	IOCONBase + 0x014															

Table 8-14. SSPMux Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:5	///	**Reserved** Write the Reset value.
4	PIN164	1 = SSPFRM 0 = PA2 (reset)
3	PIN165	1 = SSPCLK 0 = PA1 (reset)
2	PIN166	1 = SSPEN 0 = PA0 (reset)
1	PIN167	1 = U2TxD 0 = SSPTX (reset)
0	PIN169	1 = U2RxD 0 = SSPRX (reset)

Chapter 9
Exceptions and Interrupts

The LH79520 integrates a programmable Vectored Interrupt Controller (VIC) into the exception-handling hardware provided by the ARM720T core. This Chapter explains the theory and use of the LH79520 interrupt system, including conditioning external interrupt signals, ARM exceptions, and the VIC.

9.1 Theory of Operation

The LH79520 responds to ARM exceptions and to vectored Interrupts generated by the on-board VIC. The LH79520 VIC accepts inputs from 32 interrupt sources, of which 24 are internal sources and eight are external sources. The VIC also recognizes interrupts asserted under program control. The VIC is the principal user interface to the LH79520 interrupt system.

Exceptions, explained in Section 9.1.1, are a fundamental part of the ARM architecture. In the ARM architecture only the IRQ and FIQ exception vectors are available to service interrupts not related to reset or errors. If the FIQ vector is correctly utilized to provide low-latency response for a single, high-priority interrupt source then the IRQ handler must service all other interrupts from all other interrupt sources.

This burden on the IRQ vector can slow program execution and increase the complexity of the IRQ handler because the IRQ handler will be required to inspect all potential sources of interrupts and to dispatch the appropriate handler. The LH79520 provides a Vectored Interrupt Controller (VIC) to overcome this limitation.

All LH79520 interrupts, internal and external, are routed to the VIC, where the interrupt priorities are determined by hardware and the appropriate interrupt signal (nVICIRQ or nVICFIQ) is generated and dispatched to the ARM720T exception-handling hardware. The ARM720T exception-handling hardware services the interrupt as either a vectored interrupt or a non-vectored interrupt, depending upon VIC programming.

Figure 9-1, an expanded portion of the LH79520 Block diagram, shows that the VIC is programmed via the AHB and receives interrupt inputs from the DMA Controller, interrupt inputs from the on-board peripherals, and interrupt inputs from all external interrupt (INTx) sources (after the external interrupt sources are conditioned for level and edge-sensitivity). The VIC outputs two signals, nVICIRQ and nVICFIQ, to the ARM core's exception- handling circuitry.

Section 9.1.2 provides additional information regarding external interrupt signal conditioning.

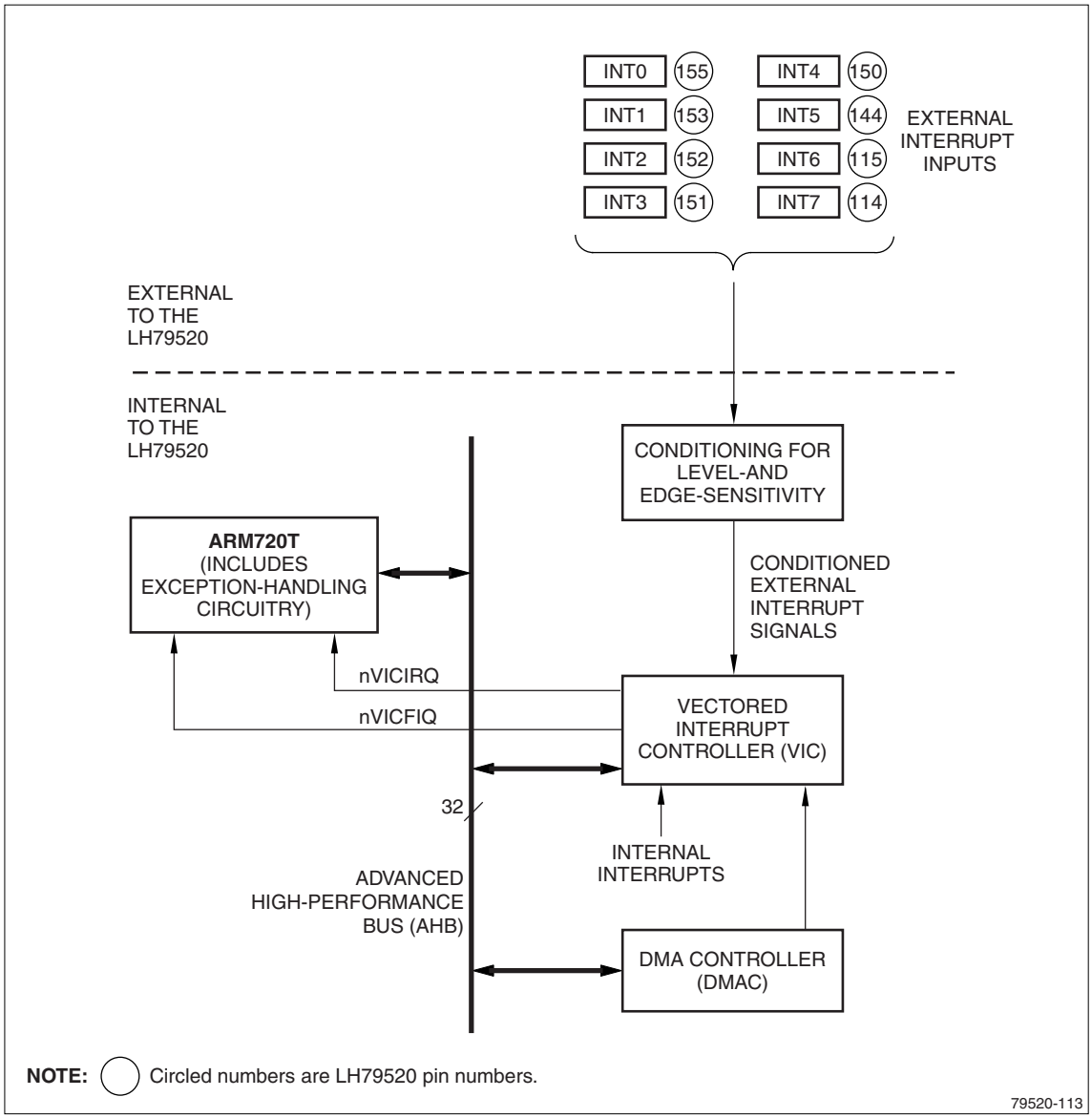


Figure 9-1. LH79520 Interrupt System

9.1.1 ARM Exceptions

The LH79520's ARM720T core recognizes ARM exceptions. Exceptions can occur synchronously with program execution and are fundamental to the ARM architecture. The ARM core's response to exceptions involves changing the core's operating mode and utilizing the exception vector addresses located in low memory and listed in Table 9-1.

Table 9-1. ARM Exception Vectors

TYPES OF EXCEPTION	CORE MODES	EXCEPTION VECTOR ADDRESS
Reset	SVC	0x00000000
Undefined Instruction	UND	0x00000004
Software Interrupt (SWI)	SVC	0x00000008
Prefetch Abort (instruction fetch memory fault)	Abort	0x0000000C
Data abort (data access memory fault)	Abort	0x00000010
IRQ (normal interrupt)	IRQ	0x00000018
FIQ (fast interrupt)	FIQ	0x0000001C

In ARM terminology, exceptions include the system reset function, the software interrupt command and undefined instruction traps. ARM exceptions can be grouped into three categories:

- Exceptions generated as a result of handling an instruction
- Exceptions generated as a side-effect of handling an instruction
- Exceptions generated externally and unrelated to the instruction flow

When an exception occurs, the ARM core completes (if possible) the execution of the present instruction and then departs from the present instruction sequence in order to service the exception. The ARM core changes its present operating mode to the operating mode associated with the exception, saves the address of the next instruction, saves the CPSR, disables IRQs and FIQs by setting bits in the CSPR to '1', and forces the Program Counter to begin executing the software located at the appropriate exception vector address.

In most cases the software located at the exception vector address is a branch to the relevant exception-handling software routine. In the case of the FIQ, the actual exception-handling routine can be located at the FIQ exception vector address because no other exception vectors are mapped above it; the memory is available for use. The FIQ (fast interrupt) vector is so named because this close location avoids the branch and permits extremely low-latency service.

9.1.2 External Interrupt Sensitivity

The eight LH79520 External Interrupt (INTx) inputs can be configured for edge-sensitivity or level-sensitivity. Figure 9-2 shows that the IntConfig register selects the desired sensitivity for each INTx input. The figure also shows that the IntClear register is utilized to clear asserted, edge-sensitive INTx interrupts. INTx inputs configured for edge-sensitivity toggle a storage-element which is cleared by a subsequent write to the appropriate bit field in the IntClear register.

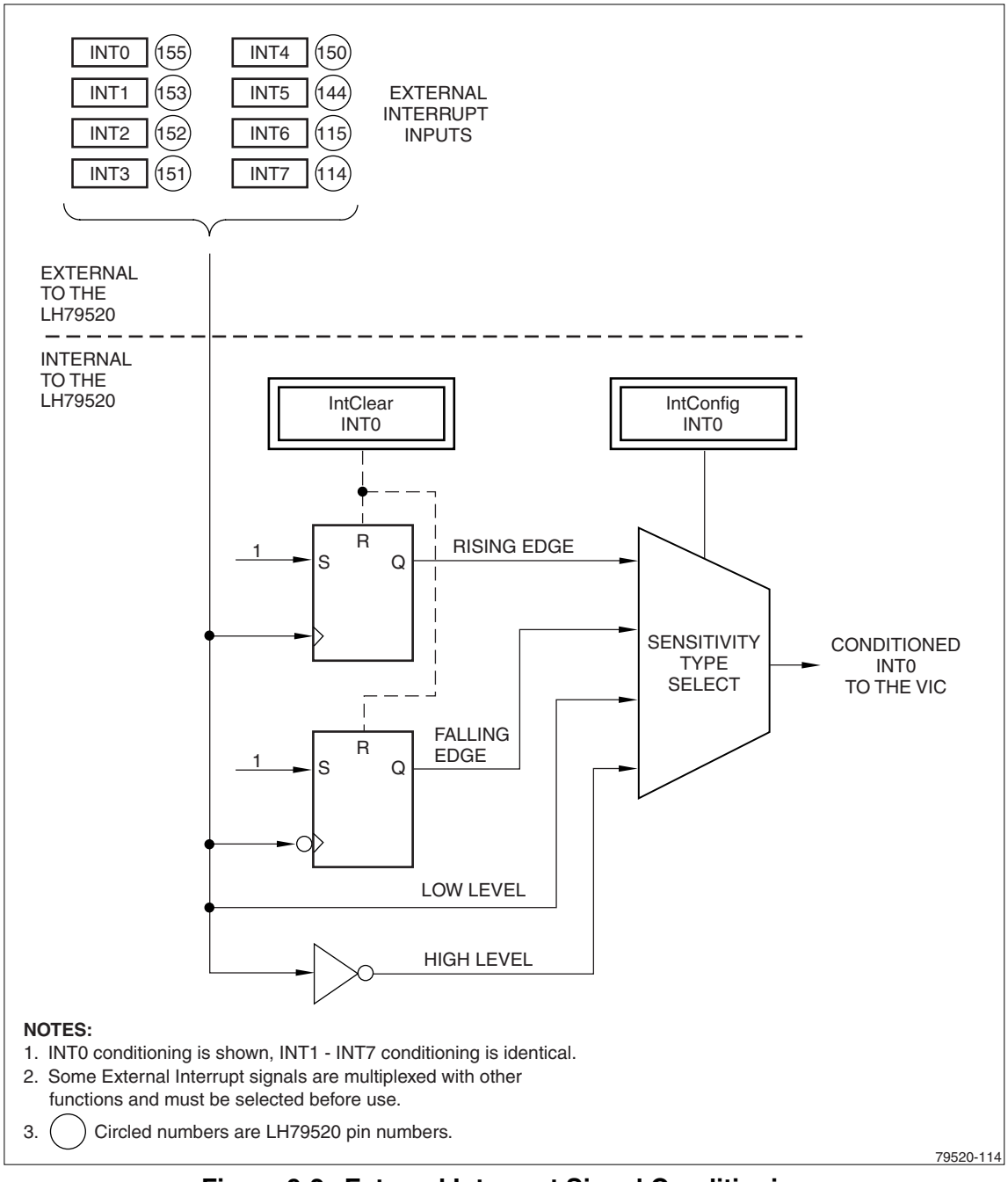


Figure 9-2. External Interrupt Signal Conditioning

All eight LH79520 INTx inputs are multiplexed with other signals and may require selection before use. The INT3 through INT7 inputs are selected at reset but the INT0 through INT2 inputs must be selected before use. See Chapter 8 – I/O Control and Multiplexing (IOCON) for additional information about selecting which function is asserted on a particular multiplexed pin.

All LH79520 INTx inputs are configured for LOW level-sensitivity at reset and the IntClear register does not affect asserted external interrupts which are configured to be level-sensitive. External interrupts configured as level-sensitive flow directly to the VIC and can generate a continuous interrupt signal which must be cleared by external hardware.

For example, if an INTx input is configured for LOW level sensitivity (as at reset), external hardware must be able to set that input to a HIGH level in order to avoid generating a continuous interrupt signal. This should be done during initialization, before interrupts are first enabled, and whenever such interrupts are serviced. See Section 9.5 for descriptions of the registers involved.

9.1.3 External Interrupt Timing

Edge-triggered interrupts must be synchronized before they pass on to the VIC. This requires at most two HCLK cycles. A level-triggered interrupt essentially goes straight through to the processor core, so there is only a few nanoseconds of delay.

However, the time it takes the processor core to respond to either an edge-triggered interrupt or a level-triggered interrupt is indeterminate. The actual response time depends on the application, the number and activity of enabled peripherals, and the arbitration that must occur between all these active signals.

9.1.4 Clearing External Interrupts

On reset, all external interrupts are configured for LOW level-sensitivity. External interrupts which are configured as edge-triggered interrupts during initialization must also be cleared during initialization. External hardware is not required in order to prevent spurious interrupts. Each of these interrupts can be cleared by writing a '1' to the appropriate bit field in the IntClear register.

See Section 9.1.2 for additional information.

9.2 Vectored Interrupt Controller (VIC)

The Vectored Interrupt Controller (VIC) provides the functionality of a hardware-vectored interrupt system and the functionality of a standard interrupt system. The VIC accepts inputs from 32 interrupt sources, eight external and 24 internal. The VIC can provide up to 16 vectored interrupts and 16 or more default-vectored interrupts, to a total of 32. Interrupts can be asserted under program control, with hardware interrupts having the highest priority. Figure 9-3 shows a simplified block diagram of the VIC, the LH79520 interrupt signals connected as inputs and the two outputs from the VIC to the ARM exception-handling hardware. These two outputs signal the core that a FIQ or IRQ interrupt has occurred.

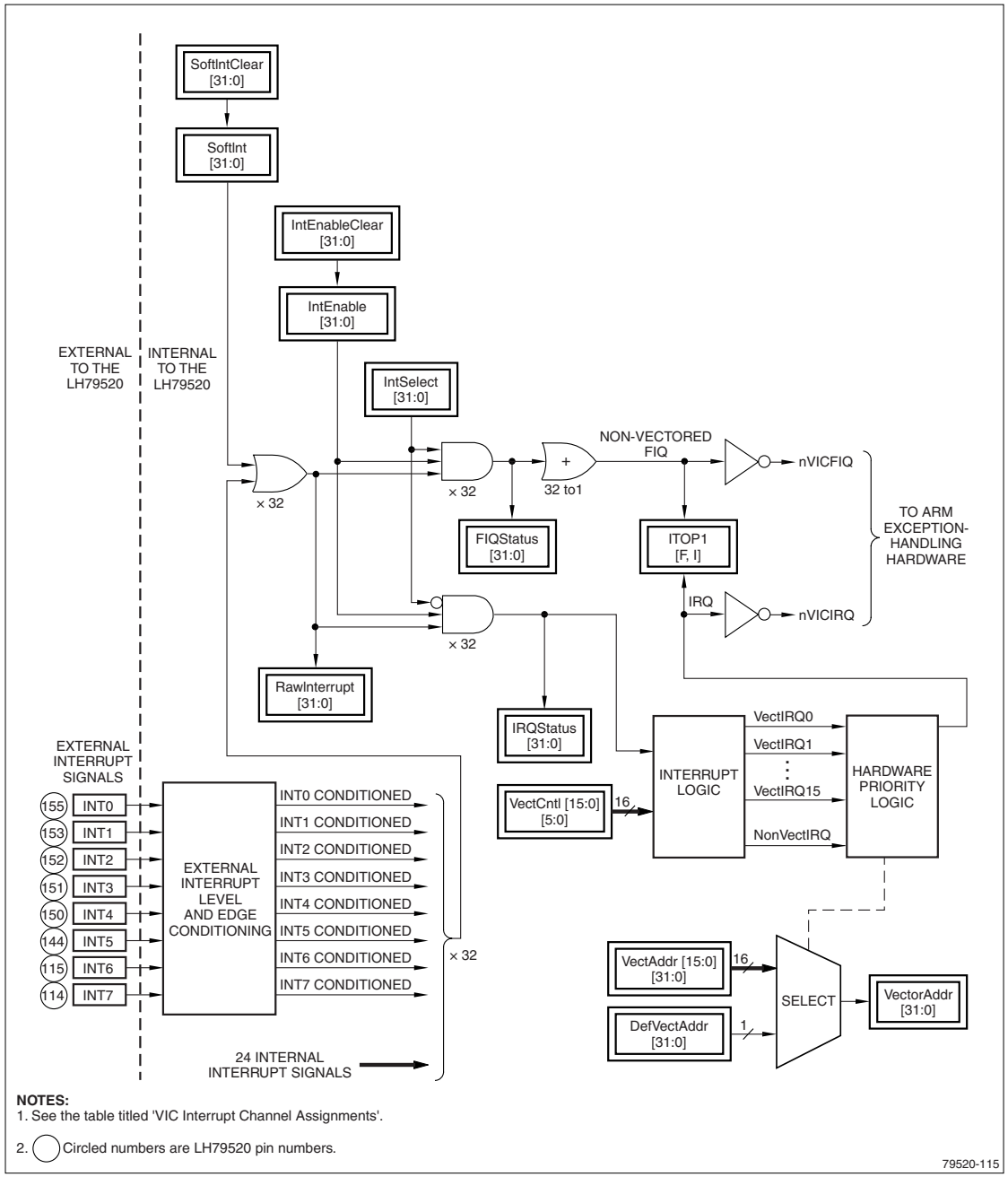


Figure 9-3. Vectored Interrupt Controller Block Diagram

9.2.1 VIC Interrupt Channels

The VIC can accept inputs from thirty-two (32) separate interrupt sources. Table 9-2 lists the interrupt channel assignments for the VIC.

Table 9-2. VIC Interrupt Channel Assignments

CHANNEL	INTERRUPT SOURCE
0	External Interrupt 0
1	External Interrupt 1
2	External Interrupt 2
3	External Interrupt 3
4	External Interrupt 4
5	External Interrupt 5
6	External Interrupt 6
7	External Interrupt 7
8	Spare Internal Interrupt 0
9	COMRx Interrupt*
10	COMTx Interrupt*
11	SSP SSPRXT0
12	CLCD Combined Interrupt
13	SSP SSPTXINTR
14	SSP SSPRXINTR
15	SSP SSPRORINTR

CHANNEL	INTERRUPT SOURCE
16	SSP SSPINTR Combined
17	Timer0
18	Timer1
19	Timer2
20	Timer3
21	UART0 Rx Combined
22	UART0 Tx Combined
23	UART0 Combined
24	UART1 Combined
25	UART2 Combined
26	DMA Combined
27	Spare Internal Interrupt 4
28	Spare Internal Interrupt 5
29	Spare Internal Interrupt 6
30	RTC
31	WDT

NOTE: *The COMRx and COMTx debugging interrupts are intended for use by the ARM RealMonitor debug software when the RealMonitor software is resident on an LH79520. In use, the ARM RealMonitor software is capable of streaming serial data to ARM debug monitor software executing on a host system. For additional information regarding ARM RealMonitor software and the use of the COMRx and COMTx debugging interrupts, refer to the literature which accompanies the ARM RealMonitor product.

9.2.2 VIC Priorities

The VIC may assert FIQ and IRQ interrupts simultaneously. If this occurs, the ARM core will give the FIQ interrupt priority over the IRQ interrupt.

Multiple IRQ interrupts may occur simultaneously. If this occurs, the VIC will arbitrate their priority in hardware, depending upon how they are programmed. The priority for IRQ interrupts is accomplished as follows:

- Vectored interrupts have priority over non-vectored interrupts
- Lower-numbered vectored interrupts have higher priority than higher-numbered vectored interrupts. Vectored interrupt 0 has highest priority; vectored interrupt 15 has lowest priority.
- Within the VIC, all non-vectored interrupts have the same priority, which is the lowest priority.

9.2.3 Interrupt Sequence of Operations

In the LH79520, interrupts are processed in the following sequence. This material refers to Figure 9-3.

1. An interrupt is asserted. If the interrupt is an external interrupt (INTx), the interrupt is conditioned for level-sensitivity or edge-sensitivity according to the way in which the IntConfig register (in the RCPC) is programmed. The resulting signal is an active HIGH level signal to the VIC. Internal interrupt signals also produce an active HIGH level signal to the VIC. If the signal is a software interrupt, the user has set a bit in the SoftInt register which causes the equivalent of an active HIGH level interrupt on the associated channel to the VIC.
2. The unmasked results of the foregoing appear in the RawInterrupt register, and are submitted to an interrupt enable masking operation. If the signal has been unmasked, that is, enabled as an interrupt which is indicated by a corresponding bit having been initialized (set) in the IntEnable register by the user, processing by the VIC continues.
3. If the signal has been enabled as an interrupt, it is routed according to whether it has been identified by the user as an FIQ or IRQ using the IntSelect register. If it has not been enabled, processing ceases. Note that the interrupt is not cleared if it has not been enabled. If the interrupt is subsequently enabled, the uncleared interrupt will assert immediately producing a spurious interrupt if global interrupts are enabled.
4. If the interrupt has been identified as an FIQ, the active HIGH signal is inverted and asserted on the CPU FIQ input line without further processing.
5. If the interrupt has been identified as an IRQ, the active HIGH signal is routed for further processing by interrupt vector and priority logic.
6. The interrupt vector logic establishes whether the interrupt has been associated with a vectored interrupt. If the signal is identified as a source calling for handling by a vectored interrupt 0-15, the signal is routed to the correct vector logic for processing.
7. If the signal is not identified as a source calling for handling by a vectored interrupt, the signal is treated as a default-vectored interrupt and is routed to the default- vectored interrupt logic for processing.
8. For the case where the signal is a vectored interrupt, the value in the associated VectAddrX register is loaded into the VectorAddr register. The VectAddrX register has been initialized by the user with the entry address of the vectored interrupt handler. The signal itself is routed to Interrupt Priority Logic within the VIC. The active HIGH output signal from the priority processing is inverted and asserted on the CPU IRQ line.
9. For the case where the signal is a default-vectored interrupt, the processing is substantially the same as for a vectored interrupt, with the difference being that the same vector is used for multiple interrupt source lines. The value in the DefVectAddr register (Default Vector Address) is loaded into the VectorAddr register. The DefVectAddr register has been initialized by the user with the entry address of the default-vectored interrupt handler. The signal itself is routed to interrupt priority logic within the VIC. The active HIGH output signal from the Interrupt Priority logic is inverted and asserted on the CPU IRQ line.
10. After an FIQ or IRQ channel into the CPU has been asserted, the CPU takes over processing of the interrupt by switching internally to either FIQ or IRQ processor mode as applicable.

11. If the interrupt is an FIQ interrupt, CPU processing should follow normal ARM conventions for processing FIQ interrupts. There are several methods for such processing which may be elected by the user. These methods are discussed in ARM literature and will not be addressed further here.
12. If the interrupt is an IRQ interrupt, low latency processing is invoked by the following instruction programmed at location 0x18 (the CPU IRQ Exception Vector) in the CPU memory map.

    0x18LDR PC, [PC,#-0xFF0]
13. If global IRQ interrupts are enabled in the CPSR of the CPU, the instruction at address 0x18 is the first instruction executed upon the invocation of any IRQ exception. Execution of this instruction will immediately load the value in register VectorAddr into the program counter as the address of the next instruction to be executed. Since VectorAddr has been loaded by the VIC with the entry address for the specific interrupt handler for the associated signal, interrupt service routine processing begins with the next instruction without any necessity to determine the interrupt source. Similarly, if the interrupt is default-vectored, the Default Vector Address has been loaded into VectorAddr by the VIC. It is incumbent on the interrupt service routine for default-vectored interrupts to determine the source of the interrupt and handle it appropriately.
14. FIQ and IRQ interrupts are globally disabled by the CPU upon the assertion of an FIQ. IRQ interrupts are globally disabled by the CPU upon the assertion of an IRQ. The timing and circumstances of re-enabling global interrupts and implementation of interrupt nesting are implementation specific and special precautions to save and restore context must be taken by the user. These subjects will not be discussed here, and it is assumed that global interrupts are re-enabled upon invoking the conventional means of returning from a single (non-nested) interrupt.
15. Before returning from an interrupt, however, the user must clear the interrupt at the source and then execute a WRITE operation to the VectorAddr register, in that order. This action clears the interrupt and notifies the interrupt and priority logic that the next interrupt may be processed by the hardware. The next interrupt may be pending, or it may be asynchronously asserted at some later time, but the sequence of interrupt processing by the VIC is the same for each.

    Note that in the event of a pending interrupt, the pending interrupt will be serviced immediately upon the return from handling the current interrupt. This is because the IRQ line to the CPU will be re-asserted immediately by the VIC hardware when the VIC VectorAddr register is written but it will not be acted upon until the global IRQ interrupts are re-enabled. This situation applies only to the case where interrupt nesting is not permitted.

9.2.4 Initializing Vectored Interrupts

To utilize the vectored feature of the VIC, initialize the following registers:

- For each interrupt source channel to be utilized, decide whether that interrupt source will be utilized as an IRQ or a FIQ type of interrupt, then program the IntSelect register accordingly.
- If any non-vectored interrupts will be enabled, initialize the DefVectAddr register to contain the entry address of the single Interrupt Service Routine (ISR) which is to service all non-vectored interrupts.
- For each interrupt channel which will be enabled as a vectored interrupt, write to the corresponding VectAddrX register the entry address of the ISR which is to service that specific interrupt. The LH79520 contains sixteen unique VectAddr registers, one for each possible vectored interrupt.
- For each interrupt channel which will be enabled as a vectored interrupt, write to the corresponding VectCntl:INTSRCE (the INTSRC bit field in one of the 16 VectCntl registers) a binary value indicating the specific channel. Also write VectCntlX:E to 1 in order to enable that interrupt source as a vectored interrupt. The LH79520 contains sixteen unique VectCntl registers, one for each possible vectored interrupt.
- For each interrupt channel to be enabled, whether vectored or non-vectored, write the appropriate bit in the IntEnable register to 1.

9.2.5 Important Considerations with External Level-sensitive Interrupts

When external interrupts are configured as level-sensitive, the interrupt service routine must ensure that there is sufficient time delay between clearing the source of the external interrupt and clearing the interrupt at the VIC. Specifically, the source of the external interrupt must have pulled the line HIGH (or LOW, depending on whether the external interrupt input pin is configured as active LOW or active HIGH) before the interrupt is cleared at the VIC, because the VIC will sample the line immediately following the Clear and will generate an interrupt again to the ARM core if the line is recognized to be still active.

When an interrupt line is shared by multiple open-collector devices in a wired-OR configuration with a pull-up resistor, the line can be especially susceptible to causing multiple interrupts if there is insufficient delay between clearing the source of the external interrupt and clearing the interrupt at the VIC. This is due to the relatively slow rise-time of the interrupt signal when being pulled to its inactive state by the pull-up resistor. The larger the resistor and load capacitance on the interrupt line, the slower the rise-time and hence more is the delay required.

In general, it is good practice to clear the source of the interrupt as early as practical in the interrupt service routine, when configuring external interrupts as level-sensitive. This will ensure a maximum delay between clearing the external interrupt and clearing the interrupt at the VIC.

9.3 External Interrupt Programmer's Model

The base address for the registers that condition the level and edge-sensitivity of external interrupts is:

RCPCBase: 0xFFFE2000

All registers in the RCPC must be accessed as a full word. The RCPC does not support byte and half-word writes.

The base address for the LH79520 Vectored Interrupt Controller (VIC) is:

VICBase: 0xFFFFF000

The Vectored Interrupt Controller registers are mirrored at a second base address:

VICBaseMirror: 0xFFFF0000

Although this User's Guide locates the VIC registers at the VICBase address, all registers shown as mapped to the VICBase address are also available at the VICBaseMirror address.

All registers in the VIC will accept word, half-word, and byte accesses.

9.4 External Interrupt Register Summary

Table 9-3 presents a summary of the programmable LH79520 registers which can be used to condition external interrupt inputs for edge-sensitivity (rising, falling) or level-sensitivity. These registers are mapped to offsets from the RCPCBase address.

Table 9-3. External Interrupt Conditioning Register Summary

ADDRESS	REGISTER NAME	DESCRIPTION
RCPCBase + 0x080	IntConfig	Interrupt Configuration Register
RCPCBase + 0x084	IntClear	Interrupt Clear Register

9.5 External Interrupt Register Descriptions

This section of this User's Guide presents descriptions of the LH79520 programmable registers that configure the external interrupt inputs. See Section 9.1.2 for additional details.

9.5.1 Interrupt Configuration Register

The IntConfig register configures external interrupt inputs (INTx) for edge-sensitivity or level-sensitivity.

At reset, all external interrupts are configured to trigger on a LOW level. Table 9-5 explains the IntConfig register bit fields.

Table 9-4. IntConfig Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	INT7		INT6		INT5		INT4		INT3		INT2		INT1		INT0	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	RCPCBase + 0x080															

Table 9-5. IntConfig Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:16	///	**Reserved** Write the Reset value.
15:14	INT7	**Configure External Interrupt 7** 0b11 = Trigger on rising edge 0b10 = Trigger on falling edge 0b01 = Trigger on HIGH level 0b00 = Trigger on LOW level (reset)
13:12	INT6	**Configure External Interrupt 6** See the INT7 bit field for details.
11:10	INT5	**Configure External Interrupt 5** See the INT7 bit field for details.
9:8	INT4	**Configure External Interrupt 4** See the INT7 bit field for details.
7:6	INT3	**Configure External Interrupt 3** See the INT7 bit field for details.
5:4	INT2	**Configure External Interrupt 2** See the INT7 bit field for details.
3:2	INT1	**Configure External Interrupt 1** See the INT7 bit field for details.
1:0	INT0	**Configure External Interrupt 0** See the INT7 bit field for details.

9.5.2 Interrupt Clear Register

The IntClear is used to individually clear asserted external interrupts.

The IntClear register is a write-only register containing bit fields which can be utilized to individually clear asserted external interrupts. Table 9-5 explains the IntClear register bit fields.

The IntClear register is located at an offset from the RCPCBase address.

Table 9-6. IntClear Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
ADDR	RCPCBase + 0x084															

Table 9-7. IntClear Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:8	///	**Reserved** Write the Reset value.
7	INT7	**Clear External Interrupt 7** 1 = Clears the interrupt 0 = Writing a '0' will have no effect on this interrupt
6	INT6	**Clear External Interrupt 6** See this register's INT7 bit field for details.
5	INT5	**Clear External Interrupt 5** See this register's INT7 bit field for details.
4	INT4	**Clear External Interrupt 4** See this register's INT7 bit field for details.
3	INT3	**Clear External Interrupt 3** See this register's INT7 bit field for details.
2	INT2	**Clear External Interrupt 2** See this register's INT7 bit field for details.
1	INT1	**Clear External Interrupt 1** See this register's INT7 bit field for details.
0	INT0	**Clear External Interrupt 0** See this register's INT7 bit field for details.

9.6 VIC Programmer's Model

The base address for the LH79520 Vectored Interrupt Controller (VIC) is:

VICBase: 0xFFFFF000

For Windows CE™ compatibility, use the MMU to map the interrupt vector table to 0xFFFF0000.

Although this User's Guide locates the VIC registers at the VICBase address, all registers shown as mapped to the VICBase base address are also available at the VICBaseMirror base address.

9.7 VIC Register Summary

Table 9-8 presents a summary of the programmable LH79520 registers which affect or report the operation of the VIC. These registers are mapped to offsets from the VICBase address.

Table 9-8. VIC Register Summary

ADDRESS	REGISTER NAME	DESCRIPTION
VICBase + 0x000	IRQStatus	VIC IRQ Status register
VICBase + 0x004	FIQStatus	VIC FIQ Status register
VICBase + 0x008	RawInterrupt	VIC Raw Interrupt register
VICBase + 0x00C	IntSelect	VIC Interrupt Select register
VICBase + 0x010	IntEnable	VIC Interrupt Enable register
VICBase + 0x014	IntEnableClear	VIC Interrupt Enable Clear register
VICBase + 0x018	SoftInt	VIC Software Interrupt register
VICBase + 0x01C	SoftIntClear	VIC Software Interrupt Clear register
VICBase + 0x020	///	Reserved — do not write*
VICBase + 0x024 - VICBase + 0x02F	///	Reserved — do not write
VICBase + 0x030	VectorAddr	Interrupt Vector address
VICBase + 0x034	DefVectAddr	Default Vector address
VICBase + 0x038 - VICBase + 0x0FF	///	Reserved — do not write
VICBase + 0x100 - VICBase + 0x13C	VectAddrX	Interrupt Vector Address X, where X = 0 to 15
VICBase + 0x140 - VICBase + 0x1FF	//	Reserved — do not write
VICBase + 0x200 - VICBase + 0x23C	VectCntlX	Interrupt Vector Control X, where X = 0 to 15
VICBase + 0x240 - VICBase + 0x30B	///	Reserved — do not write
VICBase + 0x30C	ITOP1	Returns the status of the VIC's nVICFIQ and nVICIRQ outputs to the ARM core.
VICBase + 310 - VICBase + 0xFFF	///	Reserved — do not write

NOTE: *Software must not write to this register. A write to this register will adversely affect the operation of the LH79520 and will necessitate a reset.

9.8 VIC Register Description

This section of this User's Guide presents descriptions of the LH79520 programmable registers that configure and report the status of the Vectored Interrupt Controller (VIC). These registers are located at offsets from the VICBase address.

For those VIC registers which concern all 32 VIC interrupt channels, such as the IRQStatus register, bit0 of the register corresponds to VIC interrupt channel 0, bit1 corresponds to VIC interrupt channel 1 and so on. The VIC Interrupt Channel Assignments are listed in Table 9-2.

9.8.1 IRQ Status Register

The IRQStatus register contains the status of each IRQ-type interrupt input after mask filtering and FIQ/IRQ selection by the IntEnable and IntSelect registers.

A bit which is '1' in this read-only register indicates that the associated interrupt is asserted and will generate an interrupt to the processor. See Table 9-2 for a list of interrupt channel assignments. This register is reset to '0'.

Table 9-9. IRQStatus Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	S31	S30	S29	S28	S27	S26	S25	S24	S23	S22	S21	S20	S19	S18	S17	S16
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	S15	S14	S13	S12	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
ADDR	VICBase + 0x000															

Table 9-10. IRQStatus Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:0	S31:S0	**IRQ Interrupt Status** 1 = IRQ interrupt asserted 0 = IRQ interrupt not asserted (reset) This is a read-only register

9.8.2 FIQ Status Register

The FIQStatus register contains the status of each FIQ-type interrupt input after mask filtering and FIQ/IRQ selection.

A bit which is '1' in this read-only register indicates that the associated interrupt is asserted and will generate an interrupt to the processor. See Table 9-2 for a list of interrupt channel assignments. This register is reset to '0'.

Table 9-11. FIQStatus Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	S31	S30	S29	S28	S27	S26	S25	S24	S23	S22	S21	S20	S19	S18	S17	S16
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	S15	S14	S13	S12	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
ADDR	VICBase + 0x004															

Table 9-12. FIQStatus Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:0	S31:S0	**FIQ Interrupt Status** 1 = FIQ interrupt asserted 0 = FIQ interrupt not asserted (reset) This is a read-only register

9.8.3 Raw Interrupt Register

The RawInterrupt register contains the status of all interrupt inputs prior to masking by the IntEnable register.

A bit which is '1' in this register indicates that the associated interrupt request is asserted (prior to masking). See Table 9-2 for a list of interrupt channel assignments.

Table 9-13. RawInterrupt Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	S31	S30	S29	S28	S27	S26	S25	S24	S23	S22	S21	S20	S19	S18	S17	S16
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	S15	S14	S13	S12	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0
RESET	0	0	0	0	0	0	0	0	*	*	*	*	*	*	*	*
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
ADDR	VICBase + 0x008															

NOTE: *The state of this bit field at reset (i.e.: '1' or '0') is implementation-dependent.

Table 9-14. RawInterrupt Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:0	S31:S0	**Raw Interrupt Status** 1 = Interrupt asserted 0 = Interrupt not asserted This is a read-only register. At reset, the content of bits 7:0 of this register is implementation-dependent.

The value in this read-only register at reset depends upon the logical states of external signals applied to the LH79520. Values read from this register at reset will depend upon the condition of external interrupt signals connected to the LH79520 INTx input pins.

At reset, the LH79520 interrupt system is conditioned to respond to LOW level external interrupt (INTx) inputs. If one or more external interrupt signals are LOW at reset, those signals will cause bits in this register to be set to '1'. Depending upon how the VIC is programmed, asserted external interrupt signals may also flow through the VIC to the ARM core, where they will generate continuous interrupts if interrupts are enabled.

Initialization software should clear external interrupting signals at their sources (external to the LH79520) before enabling interrupts. Section 9.2.1 lists the interrupt channel assignments, which correspond to the bits in this register. See Section 9.1.2 for additional information concerning external interrupt signal conditioning for level and edge-sensitivity.

9.8.4 Interrupt Select Register

The IntSelect register determines whether the corresponding interrupt request will generate a FIQ or an IRQ interrupt.

At reset, all VIC channels are defined as IRQ channels. Software must select those VIC channels which are to be utilized as FIQ channels.

To select a FIQ interrupt, the corresponding bit should be set to '1' in this register.

To select an IRQ interrupt, the corresponding bit should be set to '0' in this register. SeeTable 9-2 for a list of interrupt channel assignments.

This read-write register is reset to '0'.

Table 9-15. IntSelect Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	S31	S30	S29	S28	S27	S26	S25	S24	S23	S22	S21	S20	S19	S18	S17	S16
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	S15	S14	S13	S12	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	VICBase + 0x00C															

Table 9-16. IntSelect Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:0	S31:S0	Interrupt Type Selection 1 = Interrupt will be considered to be a FIQ interrupt 0 = Interrupt will be considered to be an IRQ interrupt (reset) This is a read-write register.

9.8.5 Interrupt Enable Register

The IntEnable register can enable individual interrupt request channels. This register controls the effect of individual interrupt sources for the FIQ and IRQ interrupts.

When reading this register, a bit which is '1' indicates that the interrupt is enabled and will allow an interrupt request on that interrupt channel to reach the processor. A bit which is read as '0' indicates that the interrupt is disabled and will not allow an interrupt request on that interrupt channel to reach the processor.

This is a 'modified' read-write register. This register cannot be utilized to disable individual interrupt request channels. The bits in this register can be written to '1' but cannot be written to '0' directly. See Section 9.8.6 for instructions about clearing individual bits in this register to '0'.

This register is reset to '0', which disables all interrupts. SeeTable 9-2 for a list of interrupt channel assignments.

Clearing a bit in the IntEnable register by using the IntEnableClear register disables the interrupt but does not clear the interrupt at the source nor does it clear it within the VIC. Software must separately clear the interrupt both at the source and within the VIC.

Table 9-17. IntEnable Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	IE31	IE30	IE29	IE28	IE27	IE26	IE25	IE24	IE23	IE22	IE21	IE20	IE19	IE18	IE17	IE16
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	IE15	IE14	IE13	IE12	IE11	IE10	IE9	IE8	IE7	IE6	IE5	IE4	IE3	IE2	IE1	IE0
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	VICBase + 0x010															

Table 9-18. IntEnable Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:0	IE31:IE0	**Interrupt Enable** When read: 0 = Interrupt is disabled (reset) 1 = Interrupt is enabled When written: 0 = No effect 1 = Interrupt is enabled This is a modified read-write register.

9.8.6 Interrupt Clear Register

The IntClear register is utilized to clear bits in the IntEnable register.

When writing to this register, each data bit that is written to '1' causes the corresponding bit in the IntEnable register to be cleared to '0'.

Data bits that are written as '0' to this register will have no effect on the corresponding bit in the IntEnable register. See Table 9-2 for a list of interrupt channel assignments. This is a write-only register; reads of this register return unpredictable results.

Clearing a bit in the IntEnable register by using the IntEnableClear register disables the interrupt but does not clear the interrupt at the source nor does it clear it within the VIC. Software must separately clear the interrupt both at the source and within the VIC.

Table 9-19. IntClear Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	C31	C30	C29	C28	C27	C26	C25	C24	C23	C22	C21	C20	C19	C18	C17	C16
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	C15	C14	C13	C12	C11	C10	C9	C8	C7	C6	C5	C4	C3	C2	C1	C0
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
ADDR	VICBase + 0x014															

Table 9-20. IntClear Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:0	C31:C0	**Interrupt Clear** Writing a bit in this register as '1' will cause the corresponding bit in the IntEnable register to be cleared to '0'. 0 = Has no effect on the IntEnable register. 1 = Clears the corresponding bit in the IntEnable register. This is a write-only register. Reads return unpredictable results.

9.8.7 Soft Interrupt Register

The SoftInt register is utilized to generate interrupts under program control. Writing one or more SoftInt bits to '1' forces the corresponding interrupt(s) to be asserted.

Writing a '1' to a bit in this register will generate an interrupt on the interrupt channel associated with that bit. The Interrupt channel assignments are presented in Table 9-2.

Writing a '0' to a bit in this register has no effect.

This is a 'modified' read-write register. The bits in this register can be written to '1' but cannot be written to '0' directly. See Section 9.8.8 for instructions about clearing individual bits in this register to '0'.

This register is reset to '0'. SeeTable 9-2 for a list of interrupt channel assignments.

Table 9-21. SoftInt Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	S31	S30	S29	S28	S27	S26	S25	S24	S23	S22	S21	S20	S19	S18	S17	S16
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	S15	S14	S13	S12	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	VICBase + 0x018															

Table 9-22. SoftInt Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:0	S31:S0	Interrupt generation under program control. When read: 0 = No interrupt has been asserted (reset) 1 = An interrupt has been asserted, under program control, on this interrupt channel When written: 0 = No effect 1 = Forces the generation of an interrupt on the corresponding interrupt channel This is a modified read-write register

9.8.8 Soft Interrupt Clear Register

The SoftIntClear register is utilized to clear bits in the SoftInt register.

Each data bit that is written as a '1' in this register clears the corresponding bit in the SoftInt register to '0'. A data bit written to '0' in this register has no effect on the SoftInt register.

SeeTable 9-2 for a list of interrupt channel assignments.

Table 9-23. SoftIntClear Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	C31	C30	C29	C28	C27	C26	C25	C24	C23	C22	C21	C20	C19	C18	C17	C16
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	C15	C14	C13	C12	C11	C10	C9	C8	C7	C6	C5	C4	C3	C2	C1	C0
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
ADDR	VICBase + 0x01C															

Table 9-24. SoftIntClear Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:0	C31:C0	**Interrupt Clear** Writing a bit in this register as '1' will cause the corresponding bit in the SoftInt register to be cleared to '0'. 0 = Has no effect on the SoftInt register. 1 = Clears the corresponding bit in the SoftInt register. This is a write-only register. Reads return unpredictable results.

9.8.9 Vector Address Register

The VectorAddr register is automatically loaded by the VIC with the address of the Interrupt Service Routine (ISR) for the currently asserted interrupt.

A write of '0' (the reset value) to this register clears the associated vectored interrupt within the VIC hardware (but does not clear the interrupt signal at its source).

Reads of this register will return the address of the associated vectored interrupt (or '0' if '0' has been written).

Reads of the VectorAddr register also have an effect on the VIC. Reading from the VectorAddr register provides the address of the ISR, and updates the interrupt priority hardware that masks out the current and any lower priority interrupt requests. Writing to the VectorAddr register indicates to the VIC interrupt priority hardware that the interrupt has been serviced, allowing lower priority interrupts to go active.

See Table 9-2 for a list of interrupt channel assignments. See Section 9.1.3 for additional information.

Users must not confuse the VectorAddr register with one of the sixteen (16) VectAddr registers also provided in the VIC.

Table 9-25. VectorAddr Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	A31	A30	A29	A28	A27	A26	A25	A24	A23	A22	A21	A20	A19	A18	A17	A16
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	VICBase + 0x030															

Table 9-26. VectorAddr Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:0	A31:A0	The address of the Interrupt Service Routine for the currently-asserted interrupt.

9.8.10 Default Vector Address Register

The DefVectAddr register must be programmed with the address of the default Interrupt Service Routine (ISR).

This read-write register is reset to '0'. Table 9-28 explains the bit fields in this register.

The ISR located at the address programmed to this register will be required to service all interrupts which are defined as non-vectored interrupts in the IntSelect register.

A Default Vector handler should always be installed if the VIC is used.

The IntSelect register identifies an interrupt channel as either an IRQ or an FIQ interrupt.

If the channel is identified in the IntSelect register as an FIQ, then an interrupt on the channel is handled outside the VIC, i.e., as a 'non-vectored' interrupt.

If the channel is identified in the IntSelect register as an IRQ, then interrupts on the channel will be handled as a vectored interrupt by the VIC. The vector which handles the interrupt is either the Default Vector or a specific vector, Vector0 through Vector15.

If the channel is identified in the IntSelect register as an IRQ, and the channel is NOT identified in the VectCntlX register with a specific vector, or is not enabled in the VectCntlX register, then the DefVectAddr ISR will handle that ISR.

If the VIC is programmed incorrectly, the VIC can generate multiple interrupts. If two or more specific vectors are allocated to a single specific VIC channel, the VIC hardware will assert the nVICIRQ signal for each and every vector allocated.

Table 9-27. DefVectAddr Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	A31	A30	A29	A28	A27	A26	A25	A24	A23	A22	A21	A20	A19	A18	A17	A16
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	VICBase + 0x034															

Table 9-28. DefVectAddr Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:0	A31:A0	The address of the default ISR for non-vectored interrupts.

9.8.11 Vector Address Registers

Each of the sixteen (16) VectAddr registers must be programmed with the address of an Interrupt Service Routine (ISR) if that interrupt is to be a vectored interrupt.

The sixteen (16) VectAddr registers are numbered 0 through 15. Lower-numbered registers have a higher priority in the VIC. Table 9-30 describes the bit fields in each of these registers. See Section 9.5 for additional information.

Table 9-29. VectAddrX Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	A31	A30	A29	A28	A27	A26	A25	A24	A23	A22	A21	A20	A19	A18	A17	A16
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	(VICBase + 0x100) through (VICBase + 0x13C)															

Table 9-30. VectAddrX Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:0	A31:A0	The address of an ISR for a specific vectored interrupt.

9.8.12 Vector Control Registers

Each of the sixteen (16) VectCntl registers specifies the interrupt source to use for that Vectored Interrupt.

The sixteen (16) VectCntl registers are numbered 0 through 15. SeeTable 9-2 for a list of interrupt channel assignments.

A Vectored interrupt is generated if and only if two conditions are met:

- The IntEnable register is programmed to enable the specific interrupt
- The IntSelect register is programmed to generate an IRQ interrupt.

This double requirement avoids the possibility that a single interrupt request could generate multiple interrupts if the Vectored Interrupt Controller were programmed incorrectly.

Table 9-32 explains the bit fields in each of the 16 VectCntl registers: See Section 9.5 for additional information.

Table 9-31. VectCntlX Registers

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///										E	INTSRC				
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	(VICBase + 0x200) through (VICBase + 0x23C)															

Table 9-32. VectCntlX Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:6	///	**Reserved** Write the Reset value.
5	E	0 = Disable the associated vectored interrupt. This bit is cleared to '0' on reset. 1 = Enable the associated vectored interrupt.
4:0	INTSRC	Interrupt source select. A binary value, 0b00000 through 0b11111 to denote which of the 32 possible sources to utilize for this interrupt. 0b11111 = use source #31 (all values in this range are allowed) 0b00000 = use source #0 Any of the 32 interrupt sources may be chosen. This bit field is cleared to 0b00000 on reset.

9.8.13 Interrupt Test Output Register 1

Reads of the ITOP1 register return the status of the IRQ and FIQ interrupt request outputs from the VIC to the ARM exception-handling circuitry.

The ITOP1 register is a read-only register cleared to 0 at reset.

A bit field in this register will be '1' to indicate that an interrupt request signal to the ARM exception-handling circuitry is asserted. The bit fields in this register reflect the states of the VICFIQ and VICIRQ signals before they are inverted to become nVICFIQ and nVICIRQ, respectively. See Figure 9-3 for additional information.

Table 9-34 describes the bit fields in the ITOP1 register:

Table 9-33. ITOP1 Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								VI	VF	///					
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
ADDR	VICBase + 0x30C															

Table 9-34. ITOP1 Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:8	///	**Reserved** Read '0'.
7	VI*	**VICIRO Output Status** 1 = an IRQ interrupt request to the ARM core is asserted. 0 = an IRQ interrupt request to the ARM core is not asserted. (reset)
6	VF*	**VICFIO Output Status** 1 = a FIQ interrupt request to the ARM core is asserted. 0 = a FIQ interrupt request to the ARM core is not asserted. (reset)
5:0	///	**Reserved** Read '0'.

NOTE: *This bit field is asserted as a '1'.

Chapter 10
DMA Controller (DMAC)

The LH79520 includes a Direct Memory Access (DMA) Controller (DMAC) which can transfer data between on-board peripherals and memory, or between memory and memory. This Chapter explains the theory and use of the LH79520 DMAC.

10.1 Theory of Operation

The LH79520 DMA Controller (DMAC) is connected to the AHB, the APB, the interrupt system, and signals external to the LH79520, as shown in Figure 10-1, a simplified block diagram. Readers may also wish to refer to Figure 2-1, the block diagram of the entire LH79520 MCU.

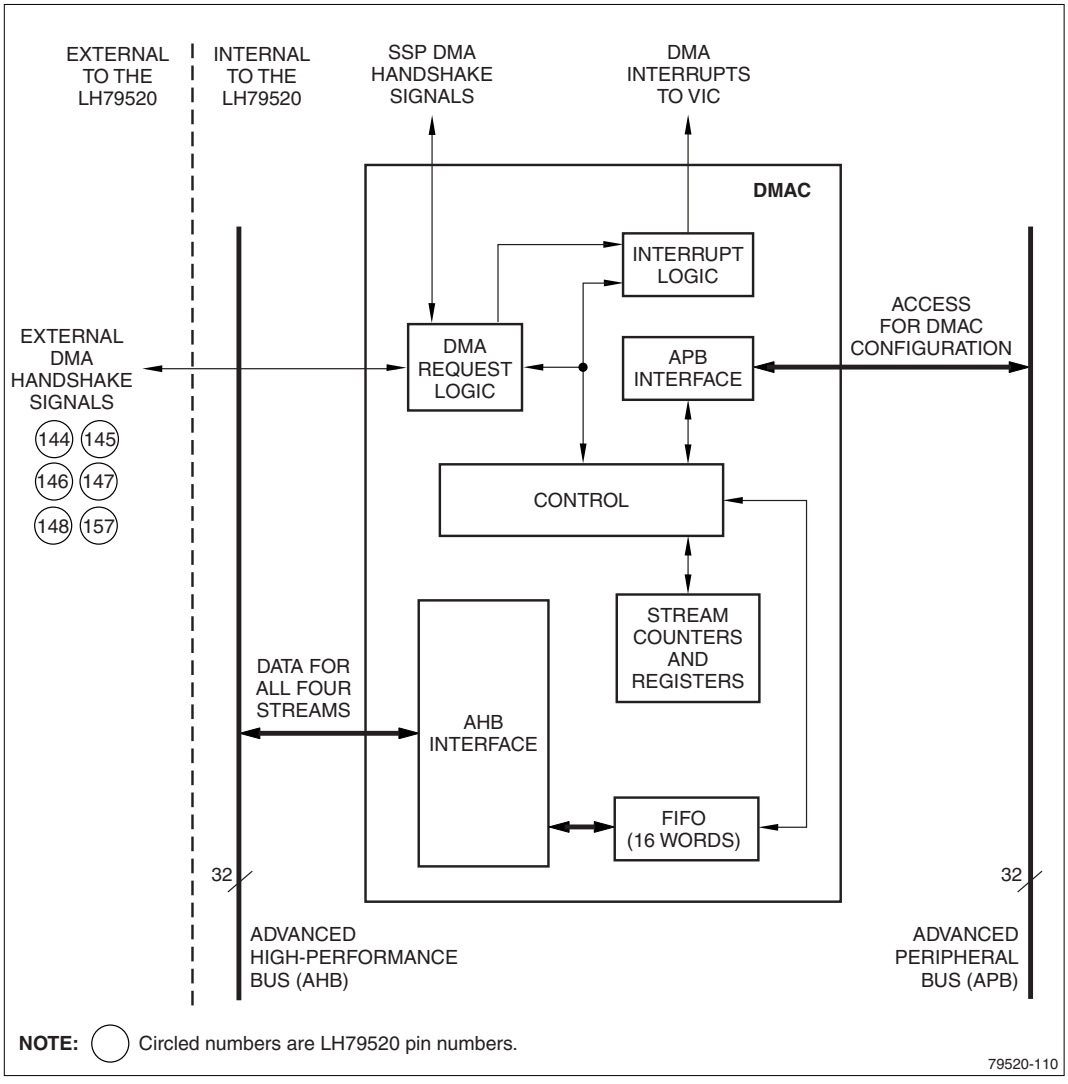


Figure 10-1. DMA Controller Block Diagram

The LH79520 DMAC supports four separate data streams (stream0, stream1, stream2, and stream3). The four data streams use a fixed-priority arbitration scheme and share a common 16-word deep FIFO for buffering burst data. Stream3 can accomplish memory-to-memory (MTM) DMA transfers under software control.

The DMAC control registers are accessed via the DMAC's 16-bit APB interface. Each of the four streams has its own independent set of DMA status and configuration registers, DMA address registers and DMA transfer count registers.

The LH79520 DMAC is programmed via the APB but all data transfers occur via the AHB. When the DMAC transfers data between the SSP and memory, the data is transferred via the AHB, the APB and the bridge between the two buses. Figure 10-2 illustrates all possible data paths.

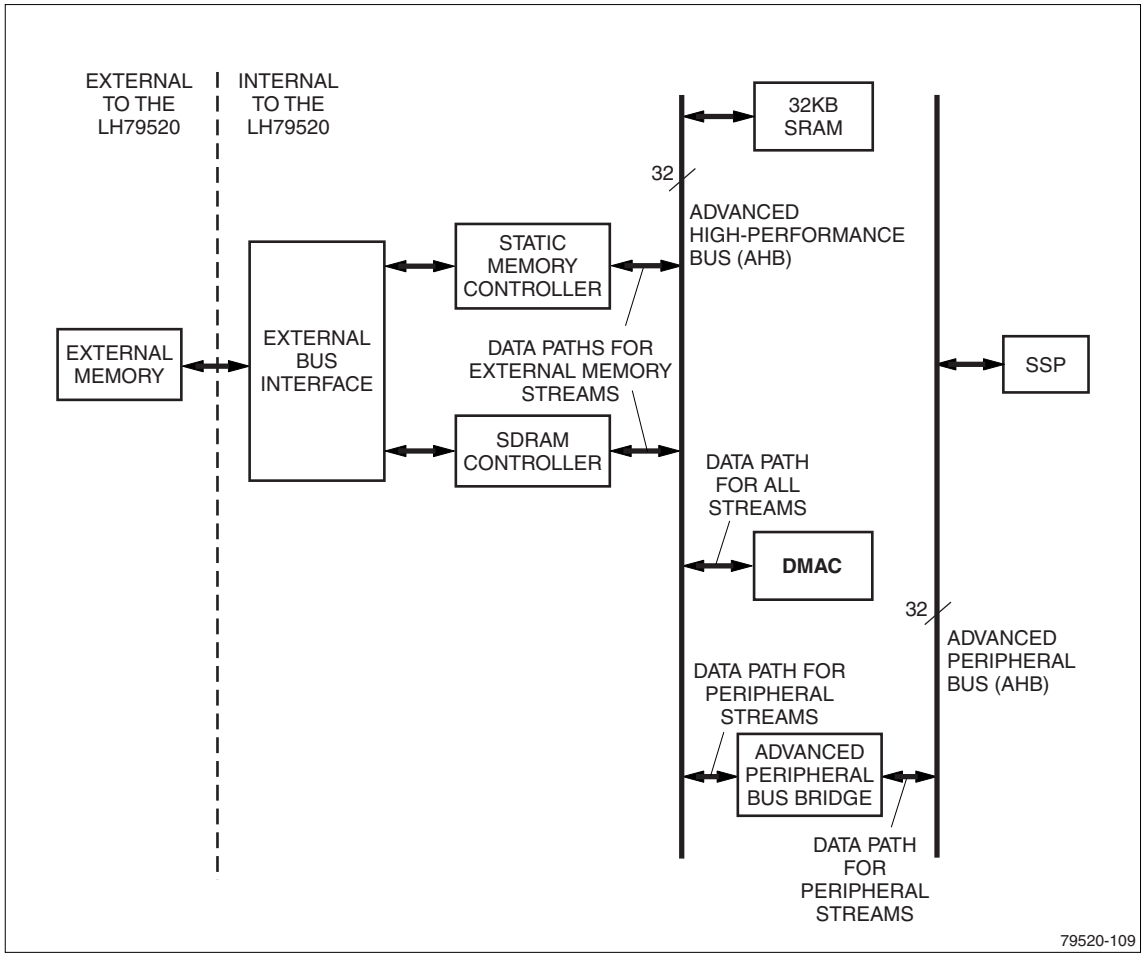


Figure 10-2. DMA Controller Data Paths

10.1.1 Device Characteristics Affect DMA Operations

DMA transfers rely upon the operating characteristics of the endpoints (the source and destination devices) and the data path between them. Endpoint characteristics such as the manner in which the device is addressed or accessed will affect how the DMAC must be programmed. Data path characteristics to be considered include the width of the path and also the way in which the path is controlled.

Endpoint addressing (at a single address or at multiple, adjacent addresses), or access (the way in which its control lines must be handled), will affect how the DMAC must be programmed. Memory, for example, is usually accessed as a series of unique, adjacent addresses. DMA transfers which access this type of endpoint will usually require an incrementing address pointer. The SSP data register, on the other hand, exists at only one unique address, so DMA transfers involving the SSP will require an unchanging address pointer.

DMA operations also rely upon the characteristics of the data path between the endpoints. The LH79520 Static Memory Controller (SMC), for example, asserts the external control signals (CSx, nWE, nOE) only when that particular block of external memory space is first accessed. These control signals remain asserted until a different block of memory is accessed. Thus, peripherals which require repeated assertions of any of these control signals (as a strobe) will be limited to a 'single' data transfer for each DMA operation, with an intervening access to a different block of memory. This can be important when transferring data to or from a peripheral which contains a FIFO and which requires a 'strobed' interface.

10.1.2 DMA Bursts

The DMAC utilizes the AHB to transfer data between memory and peripherals or between memory and memory. The DMAC must have control of the AHB in order to perform this function. In the LH79520, control of the AHB is prioritized among three Bus Masters, as explained in Chapter 3, Section 3.5.2. Only the Color LCD Controller (CLCDC) has higher AHB priority than the DMAC.

When the DMAC has control of the AHB, it transfers data in 'bursts'. A data burst is 'atomic' because the DMAC cannot lose control of the AHB during a burst. A burst represents the longest sustained period of time during which the DMAC can retain guaranteed control of the AHB. A burst also represents the largest quantity of data that can be transferred without interruption.

The DMAC can lose control of the AHB between bursts if the CLCDC requires the AHB. If the DMAC loses control of the AHB between bursts, ongoing DMA operations on all DMA data streams will be stalled until the DMAC regains control of the AHB. If a DMA transfer is interrupted because the DMAC loses control of the AHB, the DMAC will automatically resume the DMA transfer when the DMAC regains control of the AHB.

The total size of a DMA transfer, and therefore the amount of time required to accomplish the transfer at a given bus frequency, is determined by the programmed source width (DMACtrl:SOSIZE) and the programmed transfer size (DMATcnt:MAXCOUNT), in bytes.

The amount of data transferred during a burst is an important consideration for overall throughput. If the quantity of data to be transferred exceeds the capacity of the DMAC FIFO (16 words or 64 bytes) then the total DMA transfer will require more than one burst. When the DMAC is transferring data, data bursts are executed back-to-back until the requisite quantity of data has been transferred from the source to the destination, or until the CLCDC requires the use of the AHB.

The LH79520 DMAC supports all of the on-board DMA-capable peripherals except the Color LCD Controller (CLCDC), which has an independent DMA capability. See Chapter 11, Section 11.3 for more information.

10.1.3 DMA Streams

The DMA Controller (DMAC) manages four streams (channels) of data, prioritized as shown in Table 10-1. The logic within the DMAC receives DMA requests, prioritizes the requests and initiates the data transfers. The DMAC manages each stream according to the way in which the stream is programmed.

Each stream can be enabled or disabled by that stream's DMACtrl:ENABLE (the ENABLE bit field in that stream's DMACtrl register). Each stream's source base, destination base and maximum count registers must be programmed before that stream is enabled. The DMAC will automatically clear the appropriate stream's DMACtrl:ENABLE to '0' when that stream's data transfer is complete.

If, during a transfer, a stream's DMACtrl:ENABLE is written to '0' by software, then that stream's interface may be re-initialized to the values (such as addresses) programmed as its 'initial' values (depending upon the programmed mode of addressing) and the transfer will be aborted.

Table 10-1. DMA Stream Assignments

DMA STREAM	PRIORITY	DMA REQUEST SOURCE
Stream0	Highest	SSP Receive
Stream1		SSP Transmit
Stream2		External Request 0 (DREQ0)
Stream3	Lowest	External Request 1 (DREQ1)

10.1.4 DMA Addressing Modes

The LH79520 DMAC offers two different programmable modes of addressing: the 'Wrapping' mode of addressing and the 'Incremental' or 'Persistent' mode of addressing. These two modes of addressing concern the way in which the source and destination pointers for a data stream operate during a DMA transfer. The programmed addressing mode, controlled by DMACtrl:ADDRMODE, affects how the DMAC handles the source and destination address pointers. The addressing mode for a stream must be programmed before that stream is enabled.

10.1.4.1 DMA Wrapping Mode

This mode of DMAC operation must be selected before a DMA stream is enabled. When a DMA stream is enabled in the 'Wrapping' mode, the contents of the source and destination address registers for the stream are copied to the current source and current destination address registers for the stream, respectively. In this mode of operation, this copying process will be repeated each time the DMA stream is enabled. Software can force the current address pointers to the values contained in the source and destination registers by disabling and then re-enabling the stream (DMACtrl:ENABLE).

10.1.4.2 DMA Incremental Mode

This mode of DMAC operation must be selected before the DMA stream is enabled. In this mode the copying process described in Section 10.1.4.1 will occur only the first time that the DMA stream is enabled in the 'Incremental' mode. In this mode the copying process will not be repeated as long as the stream remains in the Incremental mode. The current address pointers will not be returned to their programmed 'initial' values until the stream is disabled, placed in the 'Wrapping' mode, returned to the 'Incremental' mode, and re-enabled.

10.1.5 DMA Combined Interrupt

The DMAC can generate a DMC combined interrupt to the LH79520 VIC. Generation of the DMA combined interrupt is controlled by the bit fields in the DMAMask and DMAStatus registers.

Each of the four DMAC data streams can be programmed to cause the generation of the DMAC combined interrupt when a DMA transfer is complete, or if an error occurs during a transfer on that data stream, or both. Generation of the DMA combined interrupt is disabled at reset because the DMAMask register is reset to '0'. The DMAMask register must be programmed to select which of the DMA status indications (EOT and/or error) can generate a DMA interrupt.

If any of the bit fields in the DMAMask register are programmed to '1', and if one or more of the corresponding bit fields in the DMAStatus register are set to '1' by the DMAC, then the DMAC will set the DMA combined interrupt signal HIGH. The VIC will respond to the HIGH DMA combined interrupt signal only if the DMA combined interrupt is also enabled in the VIC.

Software servicing the DMA combined interrupt must inspect the bit fields in the DMAMask register, and also the bit fields in the DMAStatus register, in order to determine which stream generated the DMA combined interrupt, and why.

The DMA combined interrupt signal to the VIC can be cleared to a LOW by clearing all of the bit fields in the DMAStatus register to '0'. To clear one or more bit fields in the DMAStatus register to '0', software must write a '1' to the corresponding bit fields in the DMAClr register.

The DMA combined interrupt signal to the VIC can be disabled by clearing the DMAMask register to '0'. For additional information about the LH79520 VIC, see Chapter 9, Section 9.2.

10.2 DMA Sequence of Operation

When a peripheral DMA transfer is initiated by an external request signal, the DMAC requests the use of the AHB. The DMAC will have guaranteed ownership of the AHB for only one 'burst'.

When the use of the bus is granted to the DMAC, the DMAC accesses the source specified by the stream's DMASoCurrHi and DMASoCurrLo registers. The DMAC fills the FIFO with 'source data units'. The size of a source data unit (in bytes) is governed by DMACtrl:SOSIZE.

If the transfer is a Memory-To-Memory (MTM) transfer (DMACtrl:MEM2MEM = 1), then the DMAC fills the FIFO with the quantity of source data units specified by the 'source burst size' (1, 4, 8, or 16 bytes) programmed in DMACtrl:SOBURST.

If the transfer is not a MTM transfer then the quantity of source data units read into the FIFO depends upon the programming of the maximum quantity of source data units to be transferred (DMAMax:MAXCOUNT). This is explained in more detail in Section 10.3.

The DMAC then empties the FIFO by writing the contents of the FIFO to the destination specified by the address contained in the DMADeCurrHi and DMADeCurrLo registers. The size of a 'destination data unit' is determined by the programmed destination data width (byte, half-word or word) in DMACtrl:DESIZE.

This filling and emptying of the FIFO is always completed for the stream currently being serviced before other DMA requests to service other streams are considered.

When stream3 is utilized for MTM transfers, the transfer must be initiated by software. To initiate a stream3 MTM transfer, software must access that stream's control register, select an MTM transfer (DMACtrl:MEM2MEM = 1) and enable the transfer (DMACtrl:ENABLE). The transfer is then carried out in bursts, as it was for the memory-to-peripheral (MTP) or peripheral-to-memory (PTM) cases. When stream3 is utilized for MTM transfers, the DMAC will disregard the external DMA control signals (DREQ1, DEOT1, DACK1), and transfer data from the stream's source to the same stream's destination as fast as possible, until DMATcnt:TCOUNT for that stream is decremented to '0'.

10.2.1 DMA Operation Limits

Memory-to-Memory (MTM) transfers, if followed by a Peripheral-to-Memory (either MTP or PTM transfers), must have the DMA disabled before setting up the Peripheral transfer. Otherwise, the PTM or MTP transfer will start immediately without a trigger.

For example, allow the MTM transfer to complete, then disable the DMA, configure the PTM transfer with the DMA disabled, then enable the transfer.

10.3 DMA Data Units and Data Packets

The DMAC hardware uses the programmed 'source' information for each stream to establish how the source data is accessed (byte, half-word or word), how it is packed into the DMAC FIFO, and the total quantity of data to be transferred (DMAMax:MAXCOUNT). The destination information for the transfer is fully described by the programmed size of the access required to write the data to the destination. The transfer continues until the programmed quantity of data (DMAMax:MAXCOUNT) has been transferred.

Each of the four data streams can be separately programmed to describe the widths (in bytes) of the sources and the destinations for DMA transfers.

source data unit = the source data-width (byte, half-word or word) = DMACtrl:SOSIZE

destination data unit = the destination data-width (byte, half-word or word) = DMACtrl:DESIZE

Each data stream can be programmed to transfer a specific quantity of source data units from the source to the destination. Each stream can transfer from 1 to 65,535 source data units.

source data packet = (quantity of bytes in a source data unit) × (quantity of data units)

destination data packets = an automatic result of the programmed destination data width

Together the quantity of bytes implied by the source data units, and the quantity of source data units, specify the amount of data, in bytes, that will be moved on a particular stream.

10.3.1 DMA Data Unit and Data Packet Examples

This section of this User's Guide lists several examples concerning streams, data units, and data packets.

10.3.1.1 DMA Example #1

If a stream is programmed to DMACtrl:SOSIZE = 0b00 (1 byte) and DMAMax:MAXCOUNT = 0x01, then the DMAC will transfer 1 byte from the source to the destination. This will require only 1 burst. Assuming that DMACtrl:DESIZE = 0b00 (byte), the DMAC will use 8-bit accesses for the read and the write operations.

10.3.1.2 DMA Example #2

If a stream is programmed to DMACtrl:SOSIZE = 0b10 (32-bit words, each 4 bytes) and DMAMax:MAXCOUNT = 0x1000 (decimal 4096) then the DMAC will transfer 4 × 4096 = 16,384 bytes from the source to the destination. Assuming that the destination programming specifies half-words (DMACtrl:DESIZE = 0b01), the DMAC will use 32-bit accesses to acquire (read) the source data and 16-bit accesses to store it to the destination. Because the DMAC FIFO can contain only 16 words (64 bytes), this transfer will require 16,384 ÷ 64 = 256 bursts. This DMA transfer will almost certainly be periodically stalled while the CLCDC utilizes the AHB to service an LCD panel.

10.3.1.3 DMA Example #3

The LH79520 DMAC (DMA Controller) can be programmed to transfer data to and from the on-chip SSP. See Chapter 16, Section 16.6 for more information.

10.4 DMAC Programmer's Model

The DMAC is arranged with five Register Base Addresses. There are four Base Addresses for the four data streams (0 through 3) and one Base Address for the DMAC's status and control registers:

DMABase = **Stream0Base** = 0xFFFE1000

Stream1Base = 0xFFFE1040

Stream2Base = 0xFFFE1080

Stream3Base = 0xFFFE10C0

DMAControlBase = 0xFFFE10F0

All registers in the DMAC must be accessed as a full word. The DMAC does not support byte and half-word writes.

10.5 DMAC Register Summary

The DMA Controller (DMAC) includes five groups of programmable registers. One group of registers configures the operation of the DMAC; the remaining four groups of registers configure the operation of the four data streams.

10.5.1 DMA Stream Configuration Registers

The DMAC supports four data streams (stream0 through stream3). Each data stream involves a group of programmable registers which define the behavior of that stream. The programmable registers for all streams are listed in Table 10-2. These registers are accessed via the APB. The APB data path to these registers is 16 bits wide.

10.5.2 DMAC Status and Control Registers

The programmable DMA registers which concern the overall operation of the DMAC are located at a different Base Address than the registers which configure the four data streams explained in Section 10.5.1. The programmable DMA status and control registers are listed in Table 10-2. Although there are four sets of DMA stream configuration registers, there is only one set of DMAC status and control registers.

Table 10-2. DMAC Register Summary

ADDRESS	REGISTER NAME	DESCRIPTION
Stream0Base+0x000	DMASourceLo	Source base address, lower 16 bits
Stream0Base+0x004	DMASourceHi	Source base address, higher 16 bits
Stream0Base+0x008	DMADestLo	Destination base address, lower 16 bits
Stream0Base+0x00C	DMADestHi	Destination base address, higher 16 bits
Stream0Base+0x010	DMAMax	Data Stream Maximum Count Register
Stream0Base+0x014	DMACtrl	DMA Control Register
Stream0Base+0x018	DMASoCurrHi	Current Source address, higher 16 bits
Stream0Base+0x01C	DMASoCurrLo	Current Source address, lower 16 bits
Stream0Base+0x020	DMADeCurrHi	Current Destination address, higher 16 bits
Stream0Base+0x024	DMADeCurrLo	Current Destination address, lower 16 bits
Stream0Base+0x028	DMATcnt	DMA Terminal Counter
Stream0Base+0x02C - Stream0Base+0x03F	///	Reserved — do not write
Stream1Base+0x000	DMASourceLo	Source base address, lower 16 bits
Stream1Base+0x004	DMASourceHi	Source base address, higher 16 bits
Stream1Base+0x008	DMADestLo	Destination base address, lower 16 bits
Stream1Base+0x00C	DMADestHi	Destination base address, higher 16 bits
Stream1Base+0x010	DMAMax	Data Stream Maximum Count Register
Stream1Base+0x014	DMACtrl	DMA Control Register
Stream1Base+0x018	DMASoCurrHi	Current Source address, higher 16 bits
Stream1Base+0x01C	DMASoCurrLo	Current Source address, lower 16 bits
Stream1Base+0x020	DMADeCurrHi	Current Destination address, higher 16 bits
Stream1Base+0x024	DMADeCurrLo	Current Destination address, lower 16 bits
Stream1Base+0x028	DMATcnt	DMA Terminal Counter
Stream1Base+0x02C - Stream1Base+0x07F	///	Reserved — do not write
Stream2Base+0x000	DMASourceLo	Source base address, lower 16 bits
Stream2Base+0x004	DMASourceHi	Source base address, higher 16 bits
Stream2Base+0x008	DMADestLo	Destination base address, lower 16 bits
Stream2Base+0x00C	DMADestHi	Destination base address, higher 16 bits
Stream2Base+0x010	DMAMax	Data Stream Maximum Count Register
Stream2Base+0x014	DMACtrl	DMA Control Register
Stream2Base+0x018	DMASoCurrHi	Current Source address, higher 16 bits

Table 10-2. DMAC Register Summary (Cont'd)

ADDRESS	REGISTER NAME	DESCRIPTION
Stream2Base+0x01C	DMASoCurrLo	Current Source address, lower 16 bits
Stream2Base+0x020	DMADeCurrHi	Current Destination address, higher 16 bits
Stream2Base+0x024	DMADeCurrLo	Current Destination address, lower 16 bits
Stream2Base+0x028	DMATcnt	DMA Terminal Counter
Stream2Base+0x02C - Stream2Base+0x0BF	///	Reserved — do not write
Stream3Base+0x000	DMASourceLo	Source base address, lower 16 bits
Stream3Base+0x004	DMASourceHi	Source base address, higher 16 bits
Stream3Base+0x008	DMADestLo	Destination base address, lower 16 bits
Stream3Base+0x00C	DMADestHi	Destination base address, higher 16 bits
Stream3Base+0x010	DMAMax	Data Stream Maximum Count Register
Stream3Base+0x014	DMACtrl	DMA Control Register
Stream3Base+0x018	DMASoCurrHi	Current Source address, higher 16 bits
Stream3Base+0x01C	DMASoCurrLo	Current Source address, lower 16 bits
Stream3Base+0x020	DMADeCurrHi	Current Destination address, higher 16 bits
Stream3Base+0x024	DMADeCurrLo	Current Destination address, lower 16 bits
Stream3Base+0x028	DMATcnt	DMA Terminal Counter
Stream3Base+0x02C - Stream3Base+0x0EF	///	Reserved — do not write
DMAControlBase+0x000	DMAMask	DMA Interrupt Mask Register
DMAControlBase+0x004	DMAClr	DMA Interrupt Clear Register
DMAControlBase+0x008	DMAStatus	DMA Status Register

See Chapter 20 – Status and Configuration Registers for a complete listing of all of the programmable registers in the LH79520.

The following section of this chapter of this User's Guide explains the DMA stream configuration registers and the DMAC status and control registers in detail. The base addresses for DMAC registers which are duplicated for each stream, such as the DMASourceLo register, are identified as StreamxBase, where x is 0, 1, 2 or 3.

10.6 DMA Stream Register Descriptions

This section of this User's Guide lists and describes each of the registers that configure the DMA streams in the LH79520.

10.6.1 DMA Source Low Register

Each RegisterDMASourceLo register must be programmed with the lower 16 bits of the source address for a particular DMA data stream.

The DMASourceLo register and the DMASourceHi register comprise a set. When the DMA Controller is enabled, the contents of the DMASourceLo register are loaded into the DMASoCurrLo register for the same stream. All four DMASourceLo registers are cleared to '0' at reset.

Table 10-3. DMASourceLo Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	StreamxBase + 0x000															

10.6.2 DMA Source High Register

Each RegisterDMASourceHi register must be programmed with the upper 16 bits of the source address for a particular DMA data stream.

The DMASourceHi register and the DMASourceLo register comprise a set. When the DMA Controller is enabled, the contents of the DMASourceHi register are loaded into the DMASoCurrHi register for the same stream. All four DMASourceHi registers are cleared to '0' at reset.

Table 10-4. DMASourceHI Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	A31	A30	A29	A28	A27	A26	A25	A24	A23	A22	A21	A20	A19	A18	A17	A16
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	StreamxBase + 0x004															

10.6.3 DMA Destination Low Register

Each RegisterDMADestLo register must be programmed with the lower 16 bits of the destination address for a particular DMA data stream.

The DMADestLo register and the DMADestHi register comprise a set. When the DMA Controller is enabled, the contents of the DMADestLo register are loaded into the DMADeCurrLo register for the same stream. All four DMADestLo registers are cleared to '0' at reset.

Table 10-5. DMADestLo Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	StreamxBase + 0x008															

10.6.4 DMA Destination High Register

Each RegisterDMADestHi register must be programmed with the upper 16 bits of the destination address for a particular DMA stream.

The DMADestHi register and the DMADestLo register comprise a set. When the DMA Controller is enabled, the contents of the DMADestHi register are loaded into the DMADeCurrHi register for the same stream. All four DMADestHi registers are cleared to '0' at reset.

Table 10-6. DMADestHi Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	A31	A30	A29	A28	A27	A26	A25	A24	A23	A22	A21	A20	A19	A18	A17	A16
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	StreamxBase + 0x00C															

10.6.5 DMA Maximum Count Register

The RegisterDMAMax register for each stream must be programmed with the maximum quantity of 'data-units' to be transferred during the next DMA transfer on that stream.

A 'data-unit' is equal to the source-to-DMA data width (byte, half-word or word) programmed to DMACtrl:SOSIZE (the SOSIZE bit field in the DMACtrl register).

When a DMAC stream is enabled (DMACtrl:ENABLE = 1), the value in the DMAMax register for that stream is copied to the DMATcnt (DMA Terminal Count) register for the same stream.

Table 10-8 explains the bit fields in the DMAMax register. The value written to DMAMax:MAXCOUNT must be less than 2^{16}.

Table 10-7. DMAMax Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	MAXCOUNT															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	StreamxBase + 0x010															

Table 10-8. DMAMax Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:16	///	**Reserved** Write the Reset value.
15:0	MAXCOUNT	**Maximum Count** The maximum quantity of source data units to transfer on this stream. 0xFFFF = the largest value that can be written to this register, and also the largest quantity of 'data-units' that can be transferred on this stream. Values from 0x0000 through 0xFFFF are allowed. 0x0001 = transfer one 'data unit' of this stream's DMACtrl:SOSIZE from the source address to the destination address and then update this stream's DMATcnt:TCOUNT. 0x0000 = perform no transfers on this stream (reset)

10.6.6 DMA Control Register

The LH79520 includes one RegisterDMACtrl register for each DMA stream, for a total of four registers. Programming each DMACtrl register configures the associated DMA stream.

Table 10-10 explains the bit fields in each DMACtrl register.

Table 10-9. DMACtrl Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///		DIR	///	MEM2MEM	///	ADDRMODE	DESIZE		SOBURST		SOSIZE		DELNC	SOLNC	ENABLE
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	StreamxBase + 0x014															

Table 10-10. DMACtrl Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:14	///	**Reserved** Write the Reset value.
13	DIR	Controls the activity of the nDACK0 or DACK1 signals. 0 = Specifies a Peripheral-to-Memory transfer (reset) 1 = Specifies a Memory-to-Peripheral transfer
12	///	**Reserved** Write the Reset value.
11	MEM2MEM	For data stream0, stream1 and stream2: This bit field has no effect on these streams. Write '0', Read '0'. For data stream3: Selects a 'memory to memory' transfer: 0 = stream3 not configured for mem to mem transfer (reset) 1 = stream3 is configured for mem to mem transfer
10	///	**Reserved** Write the Reset value.
9	ADDRMODE	0 = Wrapping mode for source and destination (reset) 1 = Incremental address mode for source and destination
8:7	DESIZE	DMAC-to-destination data width. See Table 10-11.
6:5	SOBURST	Memory-to-Memory burst size. For stream3 only. See Table 10-12.
4:3	SOSIZE	Source-to-DMAC data width. See Table 10-13.
2	DEINC	0 = Current Destination Register is to be unchanged (reset) 1 = Current Destination Register is to be incremented
1	SOINC	0 = Current Source Register is to be unchanged (reset) 1 = Current Source Register is to be incremented
0	ENABLE	0 = DMA transfers on this stream are disabled (reset) 1 = Enable a DMA transfer on this stream

The value programmed to the DESIZE bit field for each stream determines the size in bytes of a 'destination data unit' for that stream, as shown in Table 10-11.

Table 10-11. DMA Destination Data Units

DMACtrl:DESIZE	AHB DATA WIDTH UTILIZED FOR DMA TRANSFERS TO A DESTINATION
0b00	1 byte
0b01	1 half-word (2 bytes)
0b10	1 word (4 byte)
0b11	Reserved. Do not write.

The value programmed to the SOBURST bit field for each stream can affect DMAC performance because the DMAC may not retain control of the AHB between bursts. When a burst is complete, the DMAC can lose control of the AHB if another, higher-priority Master requests the AHB. In the LH79520, only the CLCDC has a higher priority than the DMAC. When stream3 is configured as a memory-to-memory transfer, SOBURST relates to the source-side burst length. All possible values for SOBURST are given in Table 10-12.

Table 10-12. DMA Burst Sizes

SOBURST	AHB BURST TYPE
0b00	1 single burst of data-unit size
0b01	4 incrementing bursts of data-unit size
0b10	8 incrementing bursts of data-unit size
0b11	16 incrementing bursts of data-unit size

The value programmed to the SOSIZE bit field for each stream determines the size in bytes of a 'source data unit' for that stream, as shown in Table 10-13.

Table 10-13. DMA Source Data Units

DMACtrl:SOSIZE	AHB DATA WIDTH UTILIZED FOR DMA TRANSFERS FROM A SOURCE
0b00	1 byte
0b01	1 half-word (2 bytes)
0b10	1 word (4 byte)
0b11	Reserved. Do not write.

When the source is a peripheral, SOBURST defines the quantity of SOSIZE source data units (bytes, half-words or words) that will be read into the FIFO before the contents of the FIFO are written to the destination.

When a peripheral is the destination, the DMAC will automatically read the correct quantity of source data units (bytes, half-words or words) to compose a burst of DESIZE data.

10.6.7 DMA Current Source High Register

The RegisterDMASoCurrHi register for each data stream will contain the upper sixteen bits of the present value of the stream's source address pointer. This is a read-only register.

Together, the values in the DMASoCurrHi and the DMASoCurrLo registers indicate the present value of a stream's source address pointer. Each register contains 16 bits of the 32-bit address.

Table 10-15 presents the bit field definitions for the DMASoCurrHi register.

If DMACtrl:SOINC = 1, then the values in these registers will be incremented as data are transferred from the source to the DMAC FIFO. The value will be incremented when each data unit has been retrieved and written to the DMAC FIFO.

If DMACtrl:SOINC = 0, then the values in these registers will not change during the entire DMA transfer.

If a DMA transfer is ongoing, then concatenating the values in the DMASoCurrHi and DMASoCurrLo registers in order to obtain the complete address of an ongoing DMA transfer cannot be accomplished in 'atomic' fashion. The concatenation requires two successive read operations and the DMAC may alter the first register while the second register is being read, rendering the result inaccurate.

If a DMA transfer is not occurring, then the addresses in the DMASoCurrHi and DMASoCurrLo registers will not be changing, and the concatenated result of two successive read operations will be accurate.

Table 10-14. DMASoCurrHi Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	A31	A30	A29	A28	A27	A26	A25	A24	A23	A22	A21	A20	A19	A18	A17	A16
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
ADDR	StreamxBase + 0x018															

Table 10-15. DMASoCurrHi Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:16	///	**Reserved** Do not write.
15:0	A31:A16	The upper sixteen bits of the present value of this stream's source address pointer. Read-only.

10.6.8 DMA Current Source Low Registers

The RegisterDMASoCurrLo register for each data stream will contain the lower sixteen bits of the present value of a stream's source address pointer. This is a read-only register.

Together, the values in the DMASoCurrHi and the DMASoCurrLo registers indicate the present value of a stream's source address pointer. Each register contains 16 bits of the 32-bit address.

Table 10-17 presents the bit field definitions for the DMASoCurrLo register.

If DMACtrl:SOINC = 1, then the values in these registers will be incremented as data are transferred from the source to the DMAC FIFO. The value will be incremented when each data unit has been retrieved and written to the DMAC FIFO.

If DMACtrl:SOINC = 0, then the values in these registers will not change during the entire DMA transfer.

If a DMA transfer is ongoing, then concatenating the values in the DMASoCurrHi and DMASoCurrLo registers in order to obtain the complete address of an ongoing DMA transfer cannot be accomplished in 'atomic' fashion. The concatenation requires two successive read operations and the DMAC may alter the first register while the second register is being read, rendering the result inaccurate.

If a DMA transfer is not occurring, then the addresses in the DMASoCurrHi and DMASoCurrLo registers will not be changing, and the concatenated result of two successive read operations will be accurate.

Table 10-16. DMASoCurrLo Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
ADDR	StreamxBase + 0x01C															

Table 10-17. DMASoCurrLo Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:16	///	**Reserved** Do not write.
15:0	A15:A0	The lower sixteen bits of the present value of this stream's source address pointer. Read-only.

10.6.9 DMA Current Destination High Register

Each stream's RegisterDMADeCurrHi register can contain the upper sixteen bits of the present value of a stream's destination address pointer. This is a read-only register.

Together, the values in the DMADeCurrHi and the DMADeCurrLo registers indicate the present value of a stream's destination address pointer. Each register contains 16 bits of the 32-bit address.

Table 10-19 presents the bit field definitions for the DMADeCurrHi register.

If DMACtrl:DEINC = 1, then the values in these registers will be incremented as data are transferred from the DMAC FIFO to the destination. The value will be incremented when each data unit has been written from the DMAC FIFO to the destination.

If DMACtrl:DEINC = 0, then the values in these registers will not change during the entire DMA transfer.

If a DMA transfer is ongoing, then concatenating the values in the DMADeCurrHi and DMADeCurrLo registers in order to obtain the complete address of an ongoing DMA transfer cannot be accomplished in 'atomic' fashion. The concatenation requires two successive read operations and the DMAC may alter the first register while the second register is being read, rendering the result inaccurate.

If a DMA transfer is not occurring, then the addresses in the DMADeCurrHi and DMADeCurrLo registers will not be changing, and the concatenated result of two successive read operations will be accurate.

Table 10-18. DMADeCurrHi Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	A31	A30	A29	A28	A27	A26	A25	A24	A23	A22	A21	A20	A19	A18	A17	A16
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
ADDR	StreamxBase + 0x020															

Table 10-19. DMADeCurrHi Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:16	///	**Reserved** Do not write.
15:0	A31:A16	The upper sixteen bits of the present value of this stream's destination address pointer. Read-only.

10.6.10 DMA Current Destination Low Register

Each stream's RegisterDMADeCurrLo register can contain the lower sixteen bits of the present value of a stream's destination address pointer. This is a read-only register.

Together, the values in the DMADeCurrHi and the DMADeCurrLo registers indicate the present value of a stream's destination address pointer. Each register contains 16 bits of the 32-bit address.

Table 10-21 presents the bit field definitions for the DMADeCurrLo register.

Table 10-20. DMADeCurrLo Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
ADDR	StreamxBase + 0x024															

Table 10-21. DMADeCurrLo Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:16	///	**Reserved** Do not write.
15:0	A15:A0	The lower sixteen bits of the present value of this stream's destination address pointer. Read-only.

If DMACtrl:DEINC = 1, then the values in these registers will be incremented as data are transferred from the DMAC FIFO to the destination. The value will be incremented when each data unit has been written from the DMAC FIFO to the destination.

If DMACtrl:DEINC = 0, then the values in these registers will not change during the entire DMA transfer.

If a DMA transfer is ongoing, then concatenating the values in the DMADeCurrHi and DMADeCurrLo registers in order to obtain the complete address of an ongoing DMA transfer cannot be accomplished in 'atomic' fashion. The concatenation requires two successive read operations and the DMAC may alter the first register while the second register is being read, rendering the result inaccurate.

If a DMA transfer is not occurring, then the addresses in the DMADeCurrHi and DMADeCurrLo registers will not be changing, and the concatenated result of two successive read operations will be accurate.

10.6.11 DMA Terminal Count Register

Each stream's RegisterDMATcnt register is a 16-bit read-only register. This register contains the quantity of 'data units' that have yet to be moved in the current DMA transfer operation.

A 'data unit' is equal to the source-to-DMAC data width (byte, half-word or word) programmed in DMACtrl:SOSIZE. Table 10-23 presents the bit field definitions for the DMATcnt register.

When a DMA transfer is enabled, the contents of DMAMax:MAXCOUNT (the MAXCOUNT bit field in the DMAMax register) for that stream is copied to this DMATcnt register for that stream. The value is copied immediately *after* the first data unit has been transferred. Thereafter, the value in DMATcnt is decremented by '1' each time a data unit has been transferred from the DMAC FIFO to the stream destination. When the value in DMATcnt reaches '0', the FIFO content has been transferred to the destination and the DMA transfer is completed.

Table 10-22. DMATcnt Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	TCOUNT															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
ADDR	StreamxBase + 0x028															

Table 10-23. DMATcnt Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:16	///	**Reserved** Do not write.
15:0	TCOUNT	**Terminal Count** The Terminal Count is the quantity of data units remaining in the current DMA transfer operation; the quantity of data units that have not yet been transferred. When a DMA transfer on this stream is enabled, this bit field is initially loaded with the value programmed to DMAMax:MAXCOUNT. As the DMA transfer operation proceeds, this value is decremented by '1' at the completion of each transfer of each data unit. If the DMA transfer operation proceeds to completion, this value will eventually be decremented to 0. If the DMA operation halts with an error, this value will indicate the quantity of data units that were not successfully transferred. Read-only.

10.7 DMAC Status and Configuration Register Descriptions

This section of this User's Guide lists each of the registers that configure the LH79520 DMAC, and describes its function.

10.7.1 DMA Mask Register

The RegisterDMAMask register controls the generation of DMA interrupts on each of the four DMAC data streams.

Each of the four DMAC data streams can be programmed to generate an interrupt when a DMA transfer is complete (End-Of-Transfer, or EOT). Each data stream can also be programmed to generate an interrupt if a DMA error occurs. The DMAMask register contains bit fields which must be programmed to select which of these DMA status indications can generate an interrupt.

The DMAC can generate a DMA combined interrupt to the LH79520 VIC. The DMA combined interrupt must be enabled in the VIC as well as in the DMAMask register.

Software servicing the DMA combined interrupt must inspect the bit fields in the DMAMask register and also the bit fields in the DMAStatus register in order to determine which DMA stream generated the DMA combined interrupt.

The LH79520 contains only one DMAMask register. The DMAMask register bit fields are defined in Table 10-25.

Table 10-24. DMAMask Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								MASKE3	MASKE2	MASKE1	MASKE0	MASK3	MASK2	MASK1	MASK0
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
ADDR	DMAControlBase + 0x000															

Table 10-25. DMAMask Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:8	///	**Reserved** Write '0'. Read as '0'.
7	MASKE3	1 = the Error interrupt for data stream 3 is enabled 0 = the Error interrupt for data stream 3 is disabled (reset)
6	MASKE2	1 = the Error interrupt for data stream 2 is enabled 0 = the Error interrupt for data stream 2 is disabled (reset)
5	MASKE1	1 = the Error interrupt for data stream 1 is enabled 0 = the Error interrupt for data stream 1 is disabled (reset)
4	MASKE0	1 = the Error interrupt for data stream 0 is enabled 0 = the Error interrupt for data stream 0 is disabled (reset)
3	MASK3	1 = the EOT interrupt for data stream 3 is enabled 0 = the EOT interrupt for data stream 3 is disabled (reset)
2	MASK2	1 = the EOT interrupt for data stream 2 is enabled 0 = the EOT interrupt for data stream 2 is disabled (reset)
1	MASK1	1 = the EOT interrupt for data stream 1 is enabled 0 = the EOT interrupt for data stream 1 is disabled (reset)
0	MASK0	1 = the EOT interrupt for data stream 0 is enabled 0 = the EOT interrupt for data stream 0 is disabled (reset)

See Section 10.7.2 and Section 10.7.3 for additional information regarding the bit fields in the DMAMask register.

See Section 10.1.5 for additional information about the DMA combined interrupt which the DMAC can issue to the LH79520 VIC.

See Chapter 9, Section 9.2 for more information about the LH79520 VIC.

10.7.2 DMA Clear Register

The RegisterDMAClr register is utilized to clear the status bits in the DMAStatus register.

Software can clear a bit field in the DMAStatus register to a LOW by writing a '1' to the corresponding bit field in the DMAClr register. Writing a bit field in the DMAClr register to '0' will have no effect on the DMAClr register and will have no effect on the corresponding bit field in the DMAStatus register.

The DMAClr register is write-only. Definitions for the DMAClr register bit fields are shown in Table 10-27.

See Section 10.7.1 and Section 10.7.3 for additional information.

Table 10-26. DMAClr Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								CLEARE3	CLEARE2	CLEARE1	CLEARE0	CLEAR3	CLEAR2	CLEAR1	CLEAR0
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
ADDR	DMAControlBase + 0x004															

Table 10-27. DMAClr Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:8	///	**Reserved** Write '0'. Reads are unpredictable.
7	CLEARE3	Write '1' to clear DMAStatus:ERROR3 to '0'. Reads are unpredictable.
6	CLEARE2	Write '1' to clear DMAStatus:ERROR3 to '0'. Reads are unpredictable.
5	CLEARE1	Write '1' to clear DMAStatus:ERROR3 to '0'. Reads are unpredictable.
4	CLEARE0	Write '1' to clear DMAStatus:ERROR3 to '0'. Reads are unpredictable.
3	CLEAR3	Write '1' to clear DMAStatus:EOT3 to '0'. Reads are unpredictable.
2	CLEAR2	Write '1' to clear DMAStatus:EOT2 to '0'. Reads are unpredictable.
1	CLEAR1	Write '1' to clear DMAStatus:EOT1 to '0'. Reads are unpredictable.
0	CLEAR0	Write '1' to clear DMAStatus:EOT0 to '0'. Reads are unpredictable.

10.7.3 DMA Status Register

Reads of the bit fields in the read-only RegisterDMAStatus Register provide information about the status of DMA operations on each of the four streams.

The bit fields in the DMAStatus register indicate the presence of activity and/or errors on each of the four DMAC data streams. The LH79520 MCU contains only one DMAStatus register. Table 10-29 explains the bit fields in the DMAStatus register.

Table 10-28. DMAStatus Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///				ACTIVE3	ACTIVE2	ACTIVE1	ACTIVE0	ERROR3	ERROR2	ERROR1	ERROR0	EOT3	EOT2	EOT1	EOT0
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
ADDR	DMAControlBase + 0x008															

Table 10-29. DMAStatus Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:12	///	**Reserved** Do not write. Reads unpredictable.
11	ACTIVE3	1 = A data transfer is in progress on stream3 0 = A data transfer is not in progress on stream3 (reset)
10	ACTIVE2	1 = A data transfer is in progress on stream2 0 = A data transfer is not in progress on stream2 (reset)
9	ACTIVE1	1 = A data transfer is in progress on stream1 0 = A data transfer is not in progress on stream1 (reset)
8	ACTIVE0	1 = A data transfer is in progress on stream0 0 = A data transfer is not in progress on stream0 (reset)
7	ERROR3	**Data Stream 3 Error Flag** 1 = stream3 data transfer has been aborted 0 = stream3 data transfer has not been aborted (reset)
6	ERROR2	**Data Stream 2 Error Flag** 1 = stream2 data transfer has been aborted 0 = stream2 data transfer has not been aborted (reset)
5	ERROR1	**Data Stream 1 Error Flag** 1 = stream1 data transfer has been aborted 0 = stream1 data transfer has not been aborted (reset)
4	ERROR0	**Data Stream 0 Error Flag** 1 = stream0 data transfer has been aborted 0 = stream0 data transfer has not been aborted (reset)

Table 10-29. DMAStatus Register Bit Fields (Cont'd)

BITS	FIELD NAME	FUNCTION
3	EOT3	**Data Stream 3 EOT (End of Transfer) Flag** 1 = stream3 data transfer has been completed 0 = stream3 data transfer is not complete (reset)
2	EOT2	**Data Stream 2 EOT Flag** 1 = stream2 data transfer has been completed 0 = stream2 data transfer is not complete (reset)
1	EOT1	**Data Stream 1 EOT Flag** 1 = stream1 data transfer has been completed 0 = stream1 data transfer is not complete (reset)
0	EOT0	**Data Stream 0 EOT Flag** 1 = stream0 data transfer has been completed 0 = stream0 data transfer is not complete (reset)

The DMAStatus:ACTIVEx bit fields indicate whether or not a data transfer is presently occurring on that data stream. A HIGH bit indicates that a data transfer is in progress. The DMAStatus:ACTIVEx bit fields have the same priority as the DMACtrl:ENABLEx bit fields. The DMA priorities are presented in Table 10-1.

The DMAStatus:ERRORx bit fields indicate whether or not an error has occurred on that data stream. A HIGH bit indicates that the data transfer operation on the corresponding data stream was aborted by an ERROR response from an AHB slave. When this occurs, the stream that has experienced the error will remain disabled until the stream is re-enabled by a write to DMACtrl:ENABLE = 1. For additional information about AHB errors, see Chapter 3, Section 3.5.3.

The DMAStatus:EOTx bit fields indicate that a transfer has been completed on that data stream. The DMAStatus:EOTx bit field for any data stream will be set HIGH when a data transfer on that stream has been completed (when the entire packet has been transferred to its destination).

To clear any or all of the bit fields in the DMAStatus register to a LOW, software must write a '1' to the corresponding bit field in the DMAClr register.

Chapter 11
Color LCD Controller

This chapter discusses the LH79520 Color LCD Controller (CLCDC) and its Advanced LCD Interface Peripheral (ALI) for AD-TFT, HR-TFT panels, and any technology of panel compatible with this signal system. The ALI-specific description begins in section 11.2.1.

For CLCDC and ALI Interface waveforms, please see the LH79520 Data Sheet.

11.1 Introduction

The CLCDC provides all necessary control and data signals to interface the LH79520 directly to a variety of color and monochrome LCD panels, including STN and TFT panels. The ALI modifies the CLCDC output to allow the LH79520 to connect directly to the Row and Column driver chips on superthin panels, including AD-TFT, HR-TFT, or any panel that supports this method of connection. Figure 11-1 shows a simplified diagram of the two controllers connected to the AHB, to the APB, and to each other.

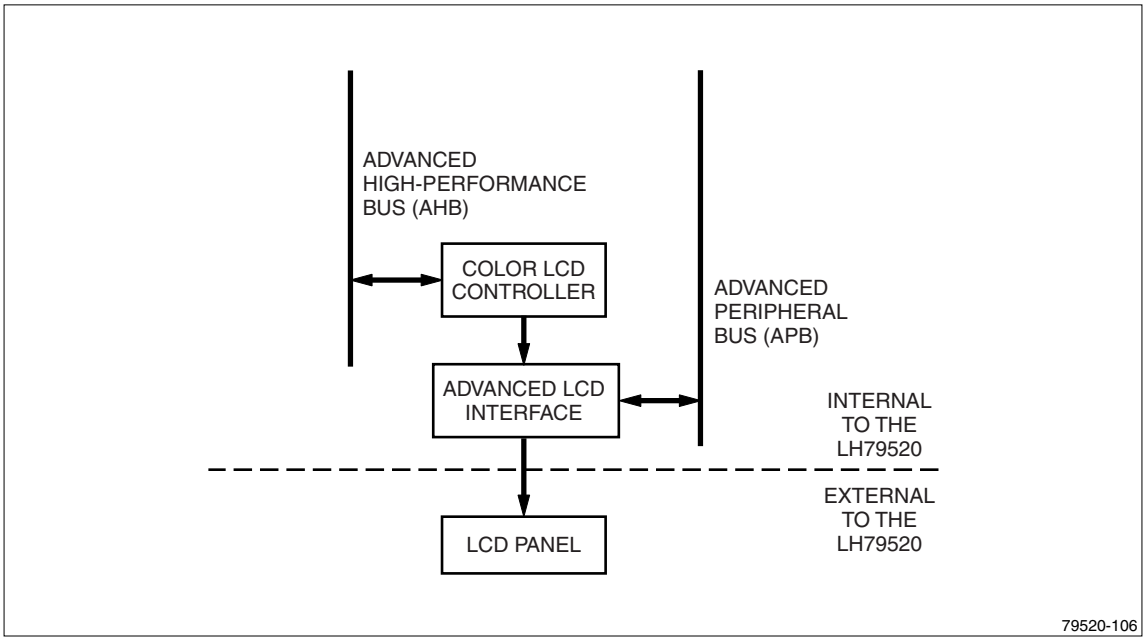


Figure 11-1. LH79520 LCD System, Simplified Block Diagram

11.1.1 LCD Panel Architecture

The CLCDC has an AHB slave interface to its registers and an AHB master interface for the LCD data. The ALI has an APB slave interface to its registers. Image data flows from the AHB, through the CLCDC and the ALI, to an external LCD panel. Although a particular LCD panel may not require the ALI, the ALI must be correctly programmed because all LCD data passes through it, even if it is set to bypass mode for STN and TFT applications.

Modern technology panels, including AD-TFT and HR-TFT panels, are thinner than ever. To achieve maximum space savings, they are manufactured without the large ASICs and DC-DC converter blocks built into STN and TFT panels. See Figure 11-2.

The ASIC in STN and TFT panels decodes input data into Row and Column information and builds the timing signals. It supplies this information to the panel's Row and Column driver chips to set the proper pixels at the proper intensity and at the proper times. The DC-DC converter runs the panel's power supplies and illuminator. Including these devices in STN and TFT panels, however, comes at the cost of bulk and weight.

The ALI eliminates the need for a separate Timing ASIC, since it is able to drive the panel's Row and Column driver chips directly. The DC-DC conversion is also handled off-panel, by a separate device operating the panel's high voltage supplies and illuminator. The DC-DC conversion must be handled by a separate device, since the LH79520 does not supply this function.

Unless the behavior is different, this User's Guide uses the term TFT to discuss all types of TFT panels whether the panel requires timing support from the ALI or not.

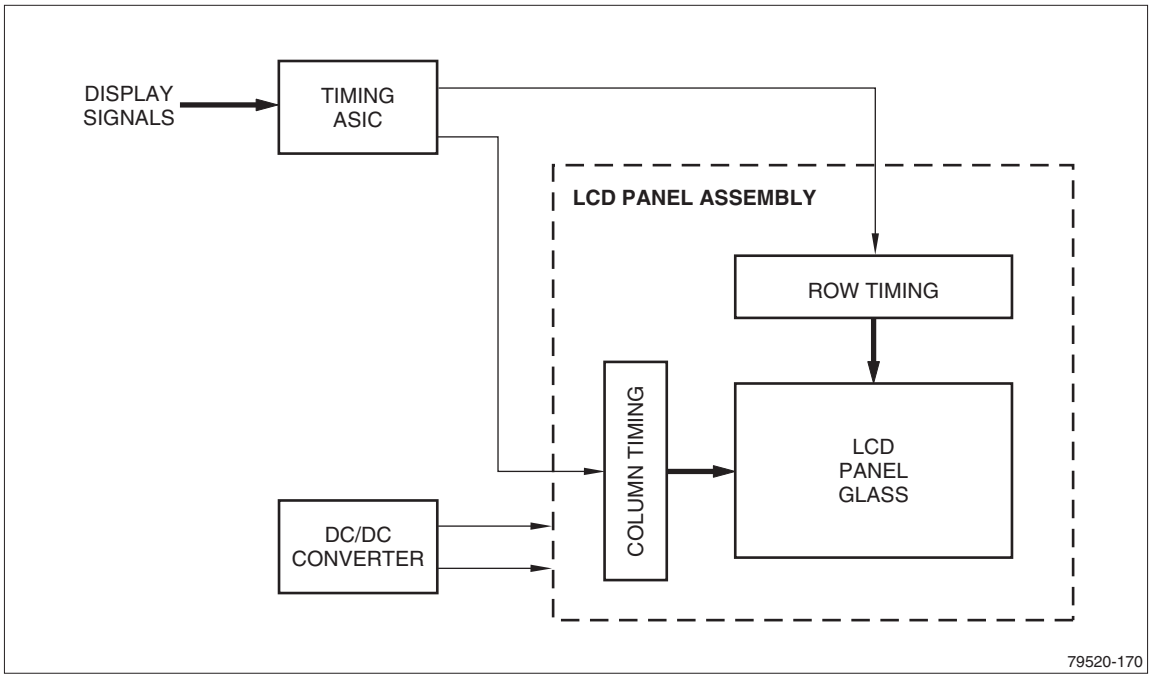


Figure 11-2. Block Diagram of a Typical Advanced LCD Panel

11.1.2 Features

The following parameters can be programmed in the CLCDC:

- Horizontal front and back porch width
- Horizontal synchronization pulse width
- Number of pixels per line
- Vertical front and back porch width
- Vertical synchronization pulse width
- Number of horizontal lines per panel
- Number of panel data clocks per line
- Programmable signal polarities, active HIGH or active LOW
- AC panel bias
- Panel data clock frequency (LCDDCLK)
- Bits-per-pixel
- Little-endian, big-endian, and WinCE™ data formatting
- Interrupt generation.

11.1.3 Theory of Operation

The CLCDC, shown in Figure 11-3, retrieves image data from a frame buffer, formats the data for the LCD panel, and writes it to the panel. The CLCDC also generates the control signals that enable the panel to display the formatted data.

Raw image data is stored in a frame buffer, which can be located in internal or external static memory, or in SDRAM. The CLCDC retrieves raw image data from the frame buffer by its own dedicated two channel DMA and formats the data as programmed. The CLCDC supports little-endian, big-endian, or WinCE pixel ordering in the frame buffer. WinCE pixel ordering is big-endian within a byte and little-endian within a word. The CLCDC can function as an AHB Master, and has the highest priority of the three LH79520 AHB Masters.

The CLCDC utilizes two clock signals operating at considerably different frequencies. The two clocks need not be synchronous. DMA operations that access the frame buffer use the AHB (fast) clock. A second, slower clock drives the logic that creates the control signals and presents formatted data to the LCD panel. Although DMA operations always occur at the internal AHB rate, the LCD panel logic can be driven by an internally-generated clock or by an external clock source. Changing sources while the CLCDC is enabled and operating is not recommended. In either case, the panel operates at a much lower frequency than the AHB. See Section 11.1.10 and Figure 11-4 for more information about the CLCDC clocks.

The CLCDC contains two DMA FIFO buffers for frame buffer data. The CLCDC requests the AHB whenever the level of data in a FIFO falls below the programmed Watermark (four or eight 32-bit words) for that FIFO. The CLCDC will not request the AHB unless a FIFO can accept at least four words. If necessary, the CLCDC will insert AHB busy cycles to ensure that the FIFOs are correctly written. When the CLCDC is granted access to the AHB, it can service either or both FIFOs. Dual panels do not double the overhead associated with mastering the AHB. An interrupt signal is asserted (if enabled), if either of the two LCD DMA FIFOs is read when they are empty.

In the 1-, 2-, 4- or 8-Bit-Per-Pixel (BPP) modes, the data provides an index into a 16-bit wide color Look-Up Table, called the Palette RAM. In the 16 BPP mode of operation the Palette RAM is bypassed and the data is the actual pixel information. The 16 BPP mode is a direct mode.

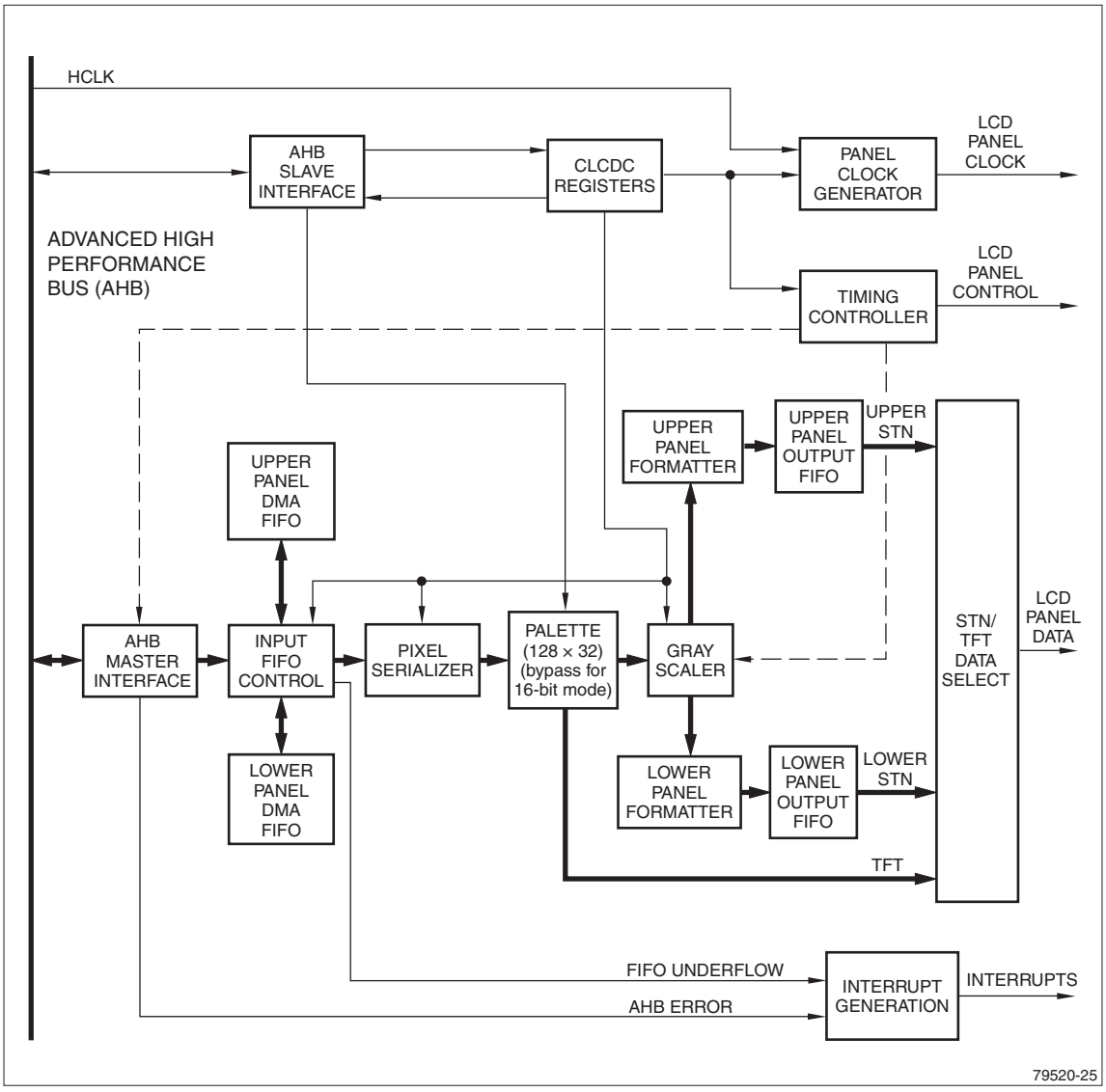


Figure 11-3. CLCDC Block Diagram

11.1.3.1 Dual-panel Color STN Operation

In the dual-panel color STN mode each FIFO (upper and lower) is fed by a separate DMA channel. The image data to be displayed on the upper and lower panels is read from the upper and lower regions of the frame buffer respectively and placed in the appropriate FIFO. Requests to each DMA channel are independently generated by each FIFO, according to the FIFO's demand. Data in each FIFO is continuously transferred to the Pixel Serializer where the data is separated into 16-, 8-, 4-, or 1-bit sections, according to the mode. The Pixel Serializer extracts the pixel data for the upper and lower panels on alternate clock cycles and passes the pixel data to the Palette RAM. The Palette RAM consists of a 128 × 32-bit dual-port RAM logically organized as 256 × 16-bits.

In the 8, 4, 2, and 1 BPP modes the Palette RAM provides the pixel's physical color value, passing the most significant 4 bits of each color bit field to the Grayscaler.

In the 16 BPP mode the Palette RAM is bypassed and the most significant 4 bits of each color bit field of the Pixel Serializer's output are passed directly to the Grayscaler. The Grayscaler generates a 1-bit pixel for each color. Each 1-bit pixel will be either HIGH or LOW for a particular pixel on the display, in a particular frame. Each color's 1-bit Grayscaler output is passed to the appropriate Formatter, to form an 8-bit data value in the Formatter's FIFO. The final data value for each pixel is output from the FIFO to the panel at 2-2/3 pixels per LCDDCLK cycle.

11.1.3.2 Dual-panel Monochrome STN Operation

In the dual-panel monochrome STN mode, image data is transferred from the frame buffer to the CLCDC input FIFOs by DMA operation, as it is for dual-panel color STN displays. Data also flows from these FIFOs, through the Pixel Serializer, to the Palette RAM in the same manner. In this mode the Palette RAM requires only 16 entries because only 1, 2, and 4 BPP formats are supported. Each Palette RAM entry represents 1 of 16 different displayable levels of gray. The Grayscaler generates 1-bit pixels, each of which will be HIGH or LOW for a particular pixel in a particular frame. The Grayscaler output is passed to the appropriate Formatter, to form a 4-bit or an 8-bit data value in the Formatter's (3 byte) FIFO, depending upon the LCD interface. The final data value is output from the Formatter's FIFO to the LCD panel at a rate of either 4 or 8 pixels per LCDDCLK cycle, depending upon the LCD interface.

11.1.3.3 Single-panel Color STN Operation

In the single-panel color STN mode, the Formatter for the lower panel is disabled and the upper and lower DMA FIFOs are utilized as a single FIFO of twice the capacity. Image data is transferred from the frame buffer to the FIFO by DMA operation. The data in the FIFO is split by the Pixel Serializer into 16-, 8-, 4-, 2-, or 1-bit sections, depending upon the programmed operating mode. The Pixel Serializer extracts the pixel data on every clock cycle and passes it to the Palette RAM or directly to the Grayscaler, depending upon the mode.

In the 8, 4, 2, and 1 BPP modes, the Palette RAM functions in the same manner as for dual-panel color STN displays.

In the 16 BPP mode, the Palette RAM is bypassed and the Pixel Serializer passes the most significant 4 bits of each color bit field to the Grayscaler.

As in dual-panel Color STN operation, the Grayscaler generates a 1-bit pixel for each color. Each 1-bit pixel will be either HIGH or LOW for a particular pixel in a particular frame. The 1-bit pixel is passed to the upper Formatter to form an 8-bit data byte in the Formatter's output (3 byte) FIFO. The final data value is output from the FIFO at a rate of 2-2/3 pixels per LCDDCLK cycle.

11.1.3.4 Single-panel Monochrome STN Operation

In the single-panel monochrome STN mode, the lower panel's Formatter is disabled and the image data is transferred by DMA operation from the frame buffer to the doubled FIFO, as it is for single-panel color STN operation. The Pixel Serializer serializes the data and passes it to the Palette RAM. The Palette RAM provides the physical gray value of the pixel for 4, 2 or 1 BPP modes.

As in dual-panel Monochrome STN operation, the Grayscaler generates 1-bit pixels, each pixel being either HIGH or LOW for a particular pixel in a particular frame. The chosen 1-bit grayscale output is passed to the Formatter to form a 4-bit or 8-bit data byte, depending upon the programmed width of the LCD interface (4 or 8-bits), and written to the (3 byte) FIFO. The final data value for the LCD panel is output at a rate of 4 or 8 pixels per LCDDCLK cycle, depending upon the programmed width of the Interface.

11.1.3.5 TFT Operation

The TFT mode is functionally similar to the single-panel color STN mode except that the Grayscaler and the Formatter are both bypassed. The image data is transferred by DMA operation from the frame buffer to the doubled FIFO. Data from the FIFO is split by the Pixel Serializer into 16-, 8-, 4-, 2-, or 1-bit sections, depending upon the programmed operating mode.

In the 8-, 4-, 2-, or 1-bit modes, the output of the Pixel Serializer is utilized as the Palette index, to extract the physical color. The Palette data is written to the LCD panel.

In the 16-bit mode, the Palette RAM is bypassed and the output of the Pixel Serializer is directly utilized as the panel data.

11.1.4 How Pixels are Stored in Memory

Table 11-1 shows the pixel arrangement on a display, with the first 32 pixels labeled p0 through p31. Table 11-2 and Table 11-3 show the data structure in each DMA FIFO word corresponding to the bpp combinations. The required data for each panel display pixel must be extracted from the data word. The first pixel value in the frame corresponds to the color value encoded in P0 (see Table 11-3), the second corresponds to P1, the third to P2, and so on (continuing in Table 11-2).

Table 11-1. Pixel Display Arrangement

p0	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14	p15	p16	p17	p18	p19	p20	p21	p22	p23	p24	p25	p26	p27	p28	p29	p30	p31

Table 11-2. Frame Buffer Pixel Storage Format [31:16]

BPP	DMA FIFO OUTPUT BITS															
1	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	p31	p30	p29	p28	p27	p26	p25	p24	p23	p22	p21	p20	p19	p18	p17	p16
2	p15		p14		p13		p12		p11		p10		p9		p8	
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
4	p7				p6				p5				p4			
	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
8	p3								p2							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
16	p1															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Table 11-3. Frame Buffer Pixel Storage Format [15:0]

BPP	DMA FIFO OUTPUT BITS															
1	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	p15	p14	p13	p12	p11	p10	p9	p8	p7	p6	p5	p4	p3	p2	p1	p0
2	p7		p6		p5		p4		p3		p2		p1		p0	
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
4	p3				p2				p1				p0			
	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
8	p1								p0							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
16	p0															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

11.1.4.1 Storing Pixels in the Frame Buffer

Tables 11-4 and 11-5 show the physical data structure in each frame buffer corresponding to the BPP combinations. The tables show only the first word of the data. In the tables 'B' represents blue, 'G' represents green, and 'R' represents red.

Table 11-4 describes color mode only. It shows the first word in the frame buffer for 16 BPP direct color, 5:5:5 + intensity bit, and the BGR bit programmed to 0 (see Section 11.3.8 for information about the BGR bit). For CSTN modes, the I bit and the least significant bit of the R, G, and B fields are unused.

Table 11-5 shows the first word in the frame buffer for 16 BPP direct color, 5:6:5 and the BGR bit programmed to 0 (see Section 11.3.8 for information about the BGR bit). It describes TFT modes, including AD-TFT and HR-TFT panels.

Table 11-4. 16 BPP Direct, 5:5:5 + Intensity, BGR = 0

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

Table 11-5. 16 BPP Direct, 5:6:5, BGR = 0 (TFT Only; includes AD-TFT and HR-TFT)

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

11.1.4.2 Pixel Serializer

The pixel serializer reads the 32-bit-wide LCD data from the output port of the LCD DMA FIFO and extracts 12, 8, 4, 2, or 1 bits per pixel (BPP) data, depending on the operating mode. In Dual Panel Mode, data alternately is read from the upper and lower LCD DMA FIFOs. Depending on the operating mode, the extracted data is either used to point to a color/gray scale value in the Palette RAM or directly applied to an LCD panel input.

11.1.5 Palette RAM

The CLCDC includes a 256 × 16 bit dual-port RAM-based palette. The least-significant bit of the serialized pixel data selects the upper or lower half of the palette RAM, based on the Byte Ordering Mode. Table 11-6 shows the representation of each word.

- One port of the dual-port palette RAM is used as a Read/Write port and is connected to the AHB slave interface. Palette entries can be written and verified through this port.
- The second port of the dual-port palette RAM is used as a Read Only port and is connected to the unpacker and grayscaler.
- In 12 and 16 bpp TFT mode, the palette is bypassed and the output of the pixel serializer is used as the TFT panel data.

Table 11-6. Palette Data Storage

BIT	NAME*	DESCRIPTION
31	I	Intensity
30:26	MB[4:0]	Most Significant Blue palette data
25:21	MG[4:0]	Most Significant Green palette data
20:16	MR[4:0]	Most Significant Red palette data
15	I	Intensity
14:10	LB[4:0]	Least Significant Blue palette data
9:5	LG[4:0]	Least Significant Green palette data
4:0	LR[4:0]	Least Significant Red palette data

NOTE: *Blue and red palette data can be swapped by programming CTRL:BGR.

11.1.6 LCD Panel Resolutions

LCD panel resolution is expressed as the addressable number of horizontal pixels with the addressable number of vertical pixels. The CLCDC supports STN, TFT, AD-TFT and HR-TFT LCD panels with a wide range of resolutions, including:

- 120 × 160
- 320 × 200, 320 × 240
- 640 × 200, 640 × 240, 640 × 480
- 800 × 600
- 1,024 × 768 (8 Bits-Per-Pixel MAX.)

11.1.6.1 Color and Gray Scale Selection

TFT, AD-TFT, and HR-TFT LCD panels utilize color palette RAM. For these panels, each 16-bit palette entry is composed of 5 Bits-Per-Pixel plus a common intensity bit. In addition, the total number of supported colors can be doubled from 32,768 to 65,536 if the Intensity bit is utilized and applied simultaneously to all three color components (R, G, and B). The CLCDC can drive up to 16 data lines for standard and thin TFT panel modules. Whether the LCD module requires the ALI to generate additional timing information does not affect the data line usage. The selection of colors that the CLCDC can drive a TFT panel to display is determined by the number of video data lines wired to the panel. If all 16 video data lines are wired to the TFT panel, 65,536 total colors are available. Possible settings are listed in Table 11-7.

The number of different colors displayable on the TFT panel at one time is programmed by the Bits-Per-Pixel (BPP) setting. Color and Monochrome STN panels are driven by a gray-scale algorithm. For Monochrome STN displays, this algorithm provides 15 different levels of gray. In the case of Color STN displays, the three color components (red, green, and blue) are simultaneously gray-scaled. The simultaneous gray-scaling provides 3,375 (15 × 15 × 15) possible levels of gray.

If the BGR bit in the CONTROL register is 0, the CLCDC outputs the color value in the palette entry (for palettized modes) or in the frame buffer entry (direct mode) onto video data lines. If the BGR bit is 1, then bits 4:0 of the palette entry or frame buffer entry are swapped with bits 14:9 before they are output.

Table 11-7. Supported TFT, AD-TFT, and HR-TFT LCD Panels

BPP	SOURCE	TFT, AD-TFT AND HR-TFT (UP TO 16-BIT BUS)
1	Palettized	2 colors selected from 65,536 available colors
2	Palettized	4 colors selected from 65,536 available colors
4	Palettized	16 colors selected from 65,536 available colors
8	Palettized	256 colors selected from 65,536 available colors
16	Direct	65,536 colors are encoded directly in the frame buffer

Table 11-9 shows the bit-depths (Bits-Per-Pixel) supported for Monochrome STN panels.

Table 11-8. Supported Color STN LCD Panels

BPP	SOURCE	COLOR STN (SINGLE AND DUAL PANEL, 8-BIT BUS)
1	Palettized	2 colors selected from 3,375
2	Palettized	4 colors selected from 3,375
4	Palettized	16 colors selected from 3,375
8	Palettized	256 colors selected from 3,375
16	Direct	4:4:4 RGB. Requires 12 data lines, 4 BPP are unused.

NOTE: 3,375 colors = (15 RED) × (15 BLUE) × (15 GREEN).

Table 11-9. Supported Mono-STN LCD Panels

BPP	SOURCE	MONO STN (SINGLE AND DUAL, 4- AND 8-BIT BUS)
1	Palettized	2 gray scales selected from 15
2	Palettized	4 gray scales selected from 15
4	Palettized	16 gray scales selected from 15

The grayscaler transforms each 4-bit grayscale value into a sequence-of-activity per pixel, over several frames. The effectiveness of this operation relies on the characteristics of STN LCDs to produce a representation of a grayscale. Table 11-10 shows the intensity that can be obtained from each of the 16 possible 4-bit palette combinations. Only 15 of the combinations are useful because the 2 middle values produce the same result.

Table 11-10. Color STN Intensities From Grayscale Modulation

4-BIT PALETTE VALUE	DUTY CYCLE[1]	RESULTING INTENSITY[2]
0b0000	0/90	00.0%
0b0001	10/90	11.1%
0b0010	18/90	20.0%
0b0011	24/90	26.7%
0b0100	30/90	33.3%
0b0101	36/90	40.0%
0b0110	40/90	44.4%
0b0111	45/90	50.0%
0b1000	45/90	50.0%
0b1001	50/90	55.6%
0b1010	54/90	60.0%
0b1011	60/90	66.6%
0b1100	66/90	73.3%
0b1101	72/90	80.0%
0b1110	80/90	88.9%
0b1111	90/90	100.0%

NOTES:
1. Duty cycle is determined by (pixels on ÷ (pixels on + pixels off)).
2. Resulting intensity: 000% = black, 100% = white.

11.1.7 CLCDC Interface Signals

The LCD interface signals generated by the CLCDC and the ALI are multiplexed with General Purpose I/O (GPIO) functions. These pins are set to GPIO functions at reset; LCD interface signals must be selected by software. LCD interface signals function differently for STN, TFT, AD-TFT, or HR-TFT panels. Table 11-11 lists only the different LCD interface signals without showing the pins carrying LCD data, and shows which signal is utilized by each of the supported types of panels.

Table 11-11. LCD Panel Interface Signals

PIN NO.	SIGNAL	STN	TFT	AD-TFT, HR-TFT
119	LCDVD6/LCDPS	LCDVD6		LCDPS (Power Save)
127	LCDVD0	LCDVD0	LCDVD0 (Intensity)	LCDVD0 (Intensity)
129	LCDFP/LCDSPS	LCDFP (Frame Pulse)	LCDFP (Vertical Sync Pulse)	LCDSPS (Row Reset)
131	LCDLP	LCDLP (Line Sync Pulse)	LCDLP (Horizontal Sync Pulse)	LCDLP (Horizontal Sync Pulse)
133	LCDDCLK	LCDDCLK (Panel Data Clock)	LCDDCLK (Panel Data Clock)	LCDDCLK (Panel Data Clock)
134	LCDCLKIN	LCDCLKIN (Optional External Clock Input)		
135	LCDVDDEN/ LCDCLS	LCDVDDEN (Digital Supply Enable)	LCDVDDEN (Digital Supply Enable)*	LCDCLS (Gate Driver Clock)
137	LCDENAB/ LCDSPL	LCDENAB (MCLK = AC Bias)	LCDENAB (Data Enable)	LCDSPL (Line Start Pulse Left)
142	LCDVD12/ LCDREV	LCDVD12	LCDVD12	LCDREV (REV = AC Bias)

NOTE: *Some TFT panels can utilize this signal to control the high-voltage supplies.

11.1.8 LCD Data Multiplexing

When LCD data is written to a LCD panel, the manner in which the LCD data is multiplexed onto the external data bus varies for STN, TFT, AD-TFT, or HR-TFT panels. Table 11-12 shows the data multiplexing for each supported panel.

For example, Table 11-12 shows that a dual-panel, color STN display receives a total of 16 bits of data for each pixel, represented here as CLSTN[7:0] and CUSTN[7:0]. The Table also shows that the display will receive the data for the lower panel on LCDVD[15:8] and the data for the upper panel on LCDVD[7:0].

Table 11-12. LCD Data Multiplexing

PIN NO.	LCD DATA SIGNAL	STN						TFT		
		MONO 4-BIT		MONO 8-BIT		COLOR		5:5:5+I	5:6:5	PALETTE DATA OR 16-BIT DIRECT
		SINGLE PANEL	DUAL PANEL	SINGLE PANEL	DUAL PANEL	SINGLE PANEL	DUAL PANEL			
130	LCDVD17							BLUE4	BLUE3	BIT 14
132	LCDVD16							BLUE3	BLUE2	BIT 13
139	LCDVD15				MLSTN7		CLSTN7	BLUE2	BLUE1	BIT 12
140	LCDVD14				MLSTN6		CLSTN6	BLUE1	BLUE0	BIT 11
141	LCDVD13				MLSTN5		CLSTN5	BLUE0	GREEN5	BIT 10
142	LCDVD12				MLSTN4		CLSTN4			
114	LCDVD11		MLSTN3		MLSTN3		CLSTN3	GREEN4	GREEN4	BIT 9
115	LCDVD10		MLSTN2		MLSTN2		CLSTN2	GREEN3	GREEN3	BIT 8
116	LCDVD9		MLSTN1		MLSTN1		CLSTN1	GREEN2	GREEN2	BIT 7
117	LCDVD8		MLSTN0		MLSTN0		CLSTN0	GREEN1	GREEN1	BIT 6
118	LCDVD7			MUSTN7	MUSTN7	CUSTN7	CUSTN7	GREEN0	GREEN0	BIT 5
119	LCDVD6			MUSTN6	MUSTN6	CUSTN6	CUSTN6			
121	LCDVD5			MUSTN5	MUSTN5	CUSTN5	CUSTN5	RED4	RED4	BIT 4
122	LCDVD4			MUSTN4	MUSTN4	CUSTN4	CUSTN4	RED3	RED3	BIT 3
123	LCDVD3	MUSTN3	MUSTN3	MUSTN3	MUSTN3	CUSTN3	CUSTN3	RED2	RED2	BIT 2
124	LCDVD2	MUSTN2	MUSTN2	MUSTN2	MUSTN2	CUSTN2	CUSTN2	RED1	RED1	BIT 1
126	LCDVD1	MUSTN1	MUSTN1	MUSTN1	MUSTN1	CUSTN1	CUSTN1	RED0	RED0	BIT 0
127	LCDVD0	MUSTN0	MUSTN0	MUSTN0	MUSTN0	CUSTN0	CUSTN0	Intensity[6]	BLUE4	BIT 15

NOTES:
1. The Intensity bit is identically generated for all three colors.
2. MUSTN = Monochrome Upper data bit for STN panel.
3. MLSTN = Monochrome Lower data bit for STN panel.
4. CUSTN = Color Upper data bit for STN panel.
5. CLSTN = Color Lower data bit for STN panel.
6. Connect to the LSB of the Red, Green, and Blue inputs of a 6:6:6 panel.
7. Recommended hookups for TFT 5:5:5 + Intensity and 5:6:5 are shown. This wiring requires the BGR bit in the CONTROL Register to be 0.

11.1.9 LCD Control Signal Multiplexing

Table 11-13 lists the control and timing signals available in the LH79520 for various panels. Some control signals are utilized universally across all types of panels, such as the Line Pulse and Data Clock signals; while some signals are utilized only in the ALI, such as LCDPS and LCDREV.

Table 11-13. Control and Timing Signal Multiplexing

PIN NO.	STN and TFT	AD-TFT, HR-TFT
119		LCDPS
129	LCDFP	LCDSPS
131	LCDLP	
133	LCDDCLK	
134	LCDDCLKIN	
135	LCDVDDEN	LCDCLS
137	LCDENAB	LCDSPL
142		LCDREV

11.1.10 CLCDC Clock Generation

The CLCDC and the ALI are driven by a common clock which can be generated internally or externally sourced. The internal source is the HCLK signal, the high frequency clock for the AHB. The external source is the LCDCLKIN input at pin 134. Figure 11-4 summarizes the LH79520 CLCDC clock generation. Some of the registers shown in the Figure are covered in Chapter 7, Reset, Clock Generation, and Power Control (RCPC).

Although the clock source for the LCD can be changed between HCLK and LCDCLKIN while the CLCDC is enabled and operating, this practice is not recomended as the resulting loss of internal synchronization can cause the system to hang.

To use an external clock input, program LCDMux:PIN134 and PeriphClkSel2:LCSRC to use an external clock input. If the LCDCLKIN function is not selected, pin 134 functions as GPIO Port C2. In addition to these selections, Figure 11-4 shows that five additional registers scale and direct the clock signal (either external or internal) to the output pin (pin 133) where hardware connects it to the LCD panel.

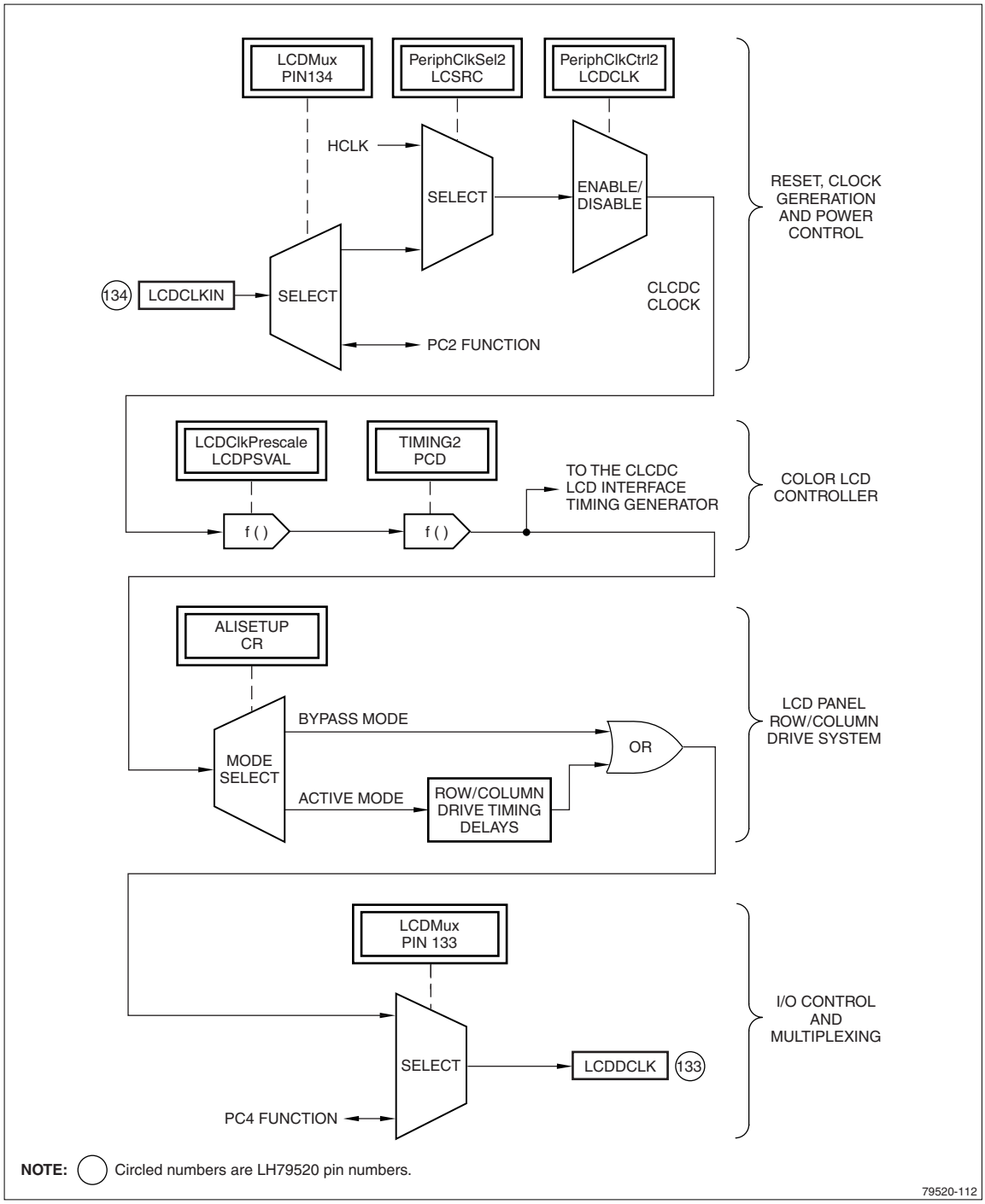


Figure 11-4. LCDDCLK Clock Generation

11.1.11 LCD Interface Timing Signals

LCD interface timing signals are categorized as either horizontal or vertical timing signals. These signals are created by the CLCDC, optionally modified by the ALI, and applied directly to an external LCD panel with no additional external hardware required, except for CGS panels.

11.1.11.1 LCD Horizontal Timing Signals

The horizontal components of LCD timing describe the process of writing one line of LCD data to a LCD panel and include programmable delays before and after the data is written to the panel. A line of data is composed of all pixel information for one displayed line. See Section for timing diagrams.

11.1.11.1.1 STN Horizontal Timing Restrictions

The CLCDC's dedicated DMA system requests new data at the start of each horizontal display line. Time must be allowed for the DMA transfer operation to occur. Time must also be allowed for the data to propagate down the FIFO path within the LCD interface. These delays constitute LCD data path latency. The data path latency imposes some restrictions on the usable minimum values for horizontal back porch width when operating in the STN modes. The value restrictions are listed in Table 11-14.

Table 11-14. Usable Minimum Values Affecting STN Back Porch Width

HORIZONTAL TIMING VALUE	SINGLE-PANEL MODE	DUAL-PANEL MODE
TIMING0:HSW	3	3
TIMING0:HBP	5	5
TIMING0:HFP	5	5
TIMING2:PCD	1 × (CLCD CLOCK/3)	5 × (CLCD CLOCK/7)

NOTE: The minimum value for PCD is 4.

11.1.11.2 LCD Vertical Timing Signals

Data is written to an LCD panel in frames. Each frame is composed of a number of horizontal lines. The vertical components of LCD timing describe the process of writing one full frame to an LCD panel.

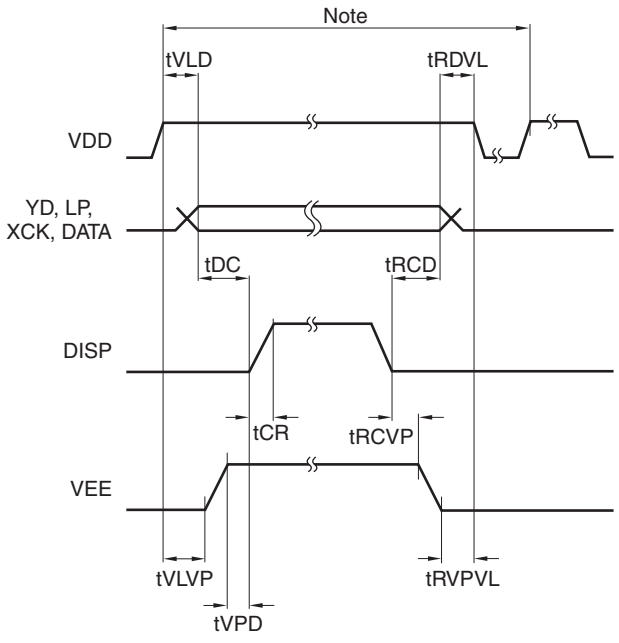
Each frame begins with a frame pulse or vertical synchronization pulse of programmable duration. Each frame pulse is followed by a programmable delay, the vertical back porch. When the vertical back porch expires, all line information for the frame is presented to the LCD panel. See Section 11.1.11.1. The line information is followed by another programmable delay, the vertical front porch.

11.1.12 LCD Power Sequencing at Turn-On and Turn-Off

Many LCD panels require ground, power for the digital logic, and high-voltage power supplies. To extend the life of these panels, the digital power must be applied before the high voltage is applied, and removed after they are removed. The logic signals driving the panel must be active before the panel voltages are applied, and the panel voltages must be removed before the logic signals are removed. This sequencing ensures that the panel is always operated with a net DC bias of 0 VDC.

Software must ensure that these conditions are met. The requisite delay is usually specified in the LCD panel's data sheet. If the proper power sequencing is not followed, the LSI drivers in the panel can latch and the display will freeze. Typically when this happens, the colors will be incorrect on STN panels. In addition the power down sequence must be followed or LCD life can be degraded.

Figure 11-5 is an example of these timing requirements for the NXP LM057QCTT03 Color STN LCD Panel, and the accompanying timing specifications. Always refer to your specific LCD panel's Data Sheet to determine the specific turn-on and turn-off requirements for the panel being used in your application.



POWER ON

SYMBOL	ALLOWABLE VALUE	
tVLD	0 ms MIN.	1 sec. MAX.
tDC	20 ms MIN.	
tCR		100 ns MAX.
tVPD	0 ms MIN.	
tVLVP	0 ms MIN.	

POWER OFF

SYMBOL	ALLOWABLE VALUE	
tRDVL	0 ms MIN.	1 sec. MAX.
tRCD	20 ms MIN.	
tRCVP	20 ms MIN.	
tRVPVL	0 ms MIN.	

NOTE: 500 ms MIN. ON/OFF cycle time. All signals and power must be switched in this sequence during power ON/OFF cycles.

79520-161

Figure 11-5. LCD Panel Power Sequencing

11.1.12.1 Minimizing a Retained Image on the LCD

While it is very important to follow the power turn-off sequence to ensure longevity of the LCD panel, this sequence alone will not ensure there is no retained image (ghosting) left on the LCD panel after the LCD has been powered down.

This ghost bleeds away slowly after powering down the LCD panel. It is most noticeable with LCD panels utilizing HR-TFT and AD-TFT technologies to light the LCD. With these types of LCD panels the ambient light alone is enough to make the retained image visible. TFT-type LCD panels also have a retained image, but it is typically not as visible once the backlight source is turned off.

To minimize the appearance of a retained image on HR-TFT and AD-TFT LCD panels, software should write a complete frame of all 1's (white) to the LCD just prior to initiating the turn-off sequence.

To minimize the appearance of a retained image on a TFT LCD panel, software should write a complete frame of all 0's (black) just prior to initiating the turn-off sequence.

In all cases the illumination source should be turned off prior to initiating the turn-off sequence.

11.1.13 CLCDC Interrupts

The CLCDC can generate an interrupt for each of four different conditions:

- Master Bus Error Interrupt
- Vertical Compare Interrupt
- LCD Next Base Address Update Interrupt
- LCD FIFO Underflow Interrupt.

Software must acknowledge the interrupt by writing a '1' to the corresponding status register bit, after which the CLCDC resumes operation from the beginning of the current frame.

Each of the four individually-maskable interrupts is enabled or disabled by the mask bits in the INTREN register. The status of the individual interrupt sources can be read from the STATUS register.

The hardware executes a logical OR of the four interrupts and asserts one combined CLCDINTR interrupt to the Vectored Interrupt Controller (VIC).

11.1.13.1 Master Bus Error Interrupt — MBEI

The Master Bus Error Interrupt is asserted when an ERROR response is received by the AHB DMA master interface during a transaction with an AHB DMA slave. For example, this error occurs when the UPBASE Register is programmed to an address that is not in the LH79520's memory map. When such an error is encountered, the master interface enters an error state and remains in this state until clearance of the error has been signalled to it. On completion of the interrupt service routine, clear the Master Bus Error Interrupt by writing a '1' to STATUS:MBEI. This action releases the master interface from its ERROR state, allowing a fresh frame of data display to be initiated.

11.1.13.2 Vertical Compare Interrupt — VCI

The Vertical Compare Interrupt is asserted when one of four vertical display regions, selected via the CONTROL register, is accessed. This interrupt can be programmed to occur at the start of:

- Vertical Synchronization
- Vertical Back Porch
- Active Video
- Vertical Front Porch.

Clear the Vertical Compare Interrupt by writing a '1' to STATUS:VCI register.

11.1.13.3 LCD Next Base Address Update Interrupt — BUI

The LCD Next Base Address Update Interrupt is asserted when either the UPBASE or the LPBASE values have been transferred to the UPCUR or LPCUR incrementers respectively. This interrupt signals that it is valid to update the UPBASE or the LPBASE registers with new frame base address, if required.

Clear this interrupt by writing a '1' to STATUS:BUI register.

11.1.13.4 LCD FIFO Underflow Interrupt — FUI

The FIFO Underflow Interrupt is asserted when data is requested by the core from an empty CLCDC input FIFO. Clear the FIFO underflow interrupt by writing a '1' to the STATUS:FUI register.

11.2 Advanced LCD Interface

The Advanced LCD Interface (ALI) provides the additional processing required to interface the LH79520 to AD-TFT, HR-TFT, or any display technology that uses this method of connection. Figure 11-6 shows the ALI between the CLCDC and the LCD output pins.

The ALI is programmed via its16-bit APB interface and receives control signals and display data from the CLCDC. The ALI converts the display data to a format suitable for direct connection to the Row and Column driver ICs in AD-TFT, HR-TFT displays, or any display using similar technology.

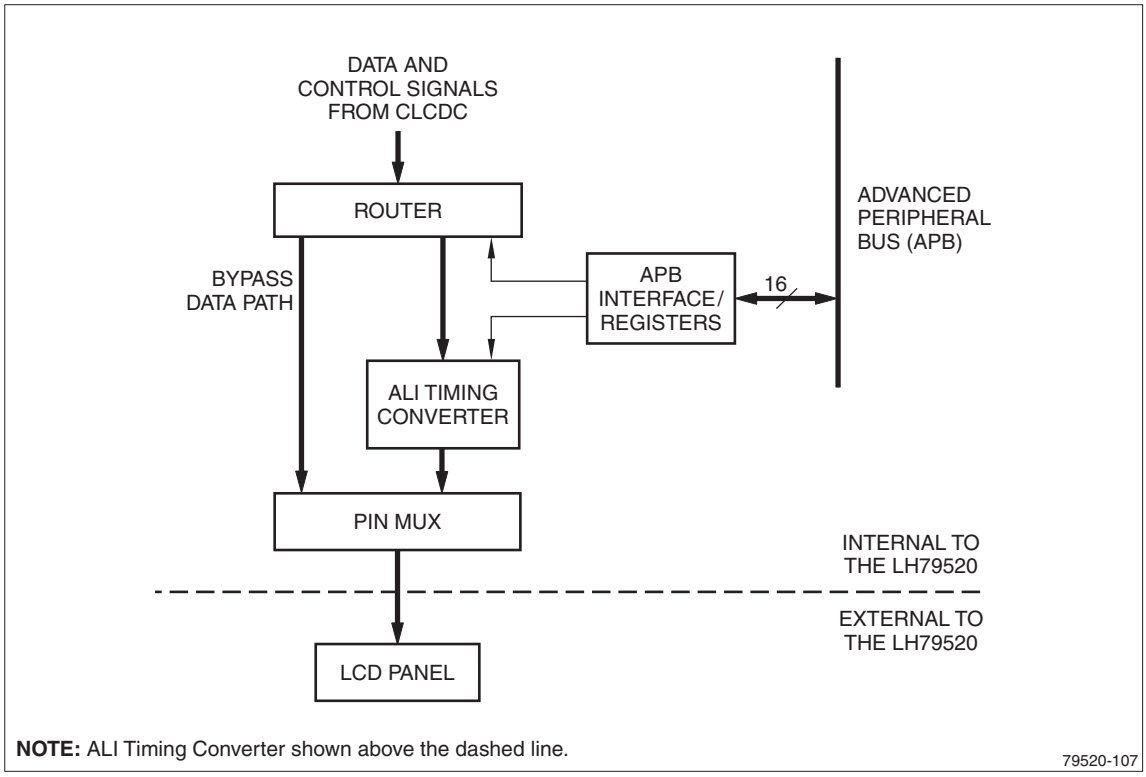


Figure 11-6. ALI Simplified Block Diagram

11.2.1 Theory of Operation

The ALI operates in two modes:

- Bypass Mode
- Active Mode

The ALI operating mode must be selected by software.

NOTE: This Chapter of the User's Guide has been updated. To aid in backward compatibility, the former register names have been included in the Register Reference. There is no change to the registers themselves; either in function or location.

11.2.1.1 Bypass Mode

The ALI defaults to Bypass mode following reset. In this mode, signals and data received from the CLCDC simply pass unaltered through the ALI to the LCD output pins. Select the Bypass mode when the CLCDC is driving STN or TFT LCD panels that contain a timing ASIC.

11.2.1.2 Active Mode

In Active mode, the ALI reformats TFT data and control signals from the CLCDC to drive the Row and Column driver circuitry on the LCD panel. When ALISETUP:CR is programmed to select the Active mode, the ALI re-times TFT data from the CLCDC so that it is output on the falling edge of the output clock (LCDDCLK). In the Active mode the ALI also generates source driver, gate driver, and voltage-preparation control signals appropriate for AD-TFT and HR-TFT LCDs.

The correct programming sequence for the CLCDC and ALI registers in Active mode is:

1. Ensure that the CLCDC is not enabled
2. Program the ALISETUP register
3. Program the ALITIMING1 Register
4. Program the ALITIMING2 register
5. Program the ALICONTROL register
6. Enable the CLCDC.

11.2.1.3 CLCDC Setup for AD-TFT or HR-TFT Operation

To supply the correct waveforms, the ALI must receive the correct signals from the CLCDC. Software must:

- Program the CLCDC to scale the internal clock signal routed to the ALI from the CLCDC to a frequency appropriate for the AD-TFT or HR-TFT panel being connected. The ALI will modify this signal's timing but not its frequency. See Section 11.1.10 and Figure 11-4 for additional information regarding the clock frequency setup.
- Program the CLCDC to operate in TFT mode.
- Program the CLCDC to provide an enable signal (LCDSPL).
- Program the CLCDC to provide a continuous clock signal (LCDDCLK).
- Program TIMING0:HSW (Horizontal Sync Width) for the AD-TFT/HR-TFT display.
- Program TIMING1:VSW (Vertical Sync Width) for the AD-TFT/HR-TFT display.
- Program TIMING2:IVS to select the appropriate polarity for the LCDSPS (Row reset) signal.
- Program TIMING2:IHS to select the appropriate polarity for the LCDLP (Horizontal Sync) signal.
- Program TIMING2:IPC to drive data on the falling edge of the LCDDCLK signal.
- Program TIMING2:IOE to '0' to select the appropriate polarity for the LCDSPL signal.

11.3 CLCDC Register Reference

The Register Base Address for the Color LCD Controller is:

CLCDC Base: 0xFFFF4000

All registers in the CLCDC will accept word, half-word, and byte accesses. Programmers should ensure that the system clock is enabled before programming any registers.

11.3.1 CLCDC Memory Map

The register offsets shown in Table 11-15 are relative to the CLCDC base address.

Table 11-15. CLCDC Register Summary

ADDRESS OFFSET	NAME	DESCRIPTION
0x000	TIMING0	Horizontal axis panel control
0x004	TIMING1	Vertical axis panel control
0x008	TIMING2	Clock and signal polarity control
0x00C	///	Reserved
0x010	UPBASE	Upper panel frame base address
0x014	LPBASE	Lower panel frame base address
0x018	INTREN	Interrupt enable
0x01C	CONTROL	LCD panel pixel parameters
0x020	STATUS	Raw interrupt status
0x024	INTERRUPT	Masked interrupt status
0x028	UPCUR	LCD upper panel current address
0x02C	LPCUR	LCD lower panel current address
0x030 - 0x1FC	///	Reserved
0x200 - 0x3FC	PALETTE	256 × 16-bit color palette
0x400 - 0x7FF	///	Reserved

11.3.2 CLCDC Register Descriptions

11.3.2.1 LCD Timing 0 Register (TIMING0)

The TIMING0 register sets the LCD panel's horizontal timing. The fields in the TIMING0 register control the Horizontal Synchronization pulse Width (HSW), the Horizontal Front Porch (HFP) period, the Horizontal Back Porch (HBP) period and the Pixels-Per-Line (PPL). Table 11-17 lists the bit assignments for the TIMING0 register.

Table 11-16. TIMING0 Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	HBP								HFP							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	HSW								PPL						///	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RO	RO
ADDR	0xFFFF4000															

Table 11-17. TIMING0 Fields

BITS	FIELD NAME	DESCRIPTION
31:24	HBP	**Horizontal Back Porch** HBP specifies the number of LCDDCLK periods between the end of the LCDLP signal and the beginning of valid data. HBP can be programmed for a delay of 1 to 256 pixel clock cycles. HBP = (LCDDCLK periods) – 1
23:16	HFP	**Horizontal Front Porch** HFP specifies the number of LCDDCLK periods between the end of valid data and the beginning of the LCDLP signal. HFP can be programmed for a delay of 1 to 256 pixel clock cycles. HFP = (LCDDCLK periods) – 1
15:8	HSW	**Horizontal Synchronization Pulse Width** HSW specifies the width of the LCDLP signal, in LCDDCLK periods. In STN modes, this signal is referred to as the 'line clock'. In TFT modes, this signal is referred to as the 'horizontal synchronization pulse'. HSW = (LCDDCLK periods) – 1
7:2	PPL	**Pixels-Per-Line** PPL specifies the number of pixels in each line of the LCD panel. The PPL value sets the number of pixel clocks that occur before the value in the HFP bit field is applied (that is, before the LCDLP signal is asserted). The PPL bit field is a 6-bit value that represents a number corresponding to the actual pixels-per-line, which can range from 16 to 1,024. At reset this field is 0, which corresponds to 16 actual pixels-per-line. PPL = (Actual pixels-per-line/16) – 1 Actual pixels-per-line = 16 × (PPL + 1)
1:0	///	**Reserved** Reading returns 0. Write Reset Value.

NOTE: *The CLCDC DMA system requests new data at the beginning of each horizontal display line. The delays implicit in this operation must be considered when establishing the horizontal display timing. See Section 11.1.11.1.1 for additional information.

11.3.3 LCD Timing 1 Register (TIMING1)

The TIMING1 register controls the LCD panel's vertical timing.

TIMING1 controls the number of Lines-Per-Panel (LPP), the Vertical Synchronization pulse Width (VSW), the Vertical Front Porch (VFP) period and the Vertical Back Porch (VBP) period. Table 11-19 lists the bit definitions for the TIMING1 register.

Table 11-18. TIMING1 Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	VBP								VFP							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	VSW						LPP									
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	0xFFFF4004															

Table 11-19. TIMING1 Fields

BITS	FIELD NAME	DESCRIPTION
31:24	VBP	**Vertical Back Porch** VBP defines the number of inactive lines at the start of a frame, after the vertical synchronization period (after the framing pulse, LCDFP, is de-asserted). The VBP bit field specifies the number of horizontal line-clocks (LCDLP) inserted at the beginning of each frame. The value in the VBP bit field can generate from 0 to 255 additional line-clock cycles. TFT modes: The VBP delay begins after the vertical synchronization signal for the previous frame (LCDFP) has been deasserted. STN modes: This field can be programmed in STN modes, but the value has no effect on the vertical back porch. The vertical back porch is zero regardless of what is written to this field. Program VBP = 0 on STN displays.
23:16	VFP	**Vertical Front Porch** VFP defines the number of inactive lines at the end of a frame, before the vertical synchronization period (before the framing pulse, LCDFP, is asserted). VFP specifies the number of horizontal line-clocks (LCDLP) inserted at the end of each frame. The value in the VFP bit field can generate from 0 to 255 line-clock cycles, thereby setting the number of lines within the vertical front porch. TFT modes: The vertical synchronization signal (LCDFP) is asserted after the VFP delay has expired. STN modes: This field can be programmed in STN modes, but the size of the vertical front porch can affect the display contrast. Typically, the larger the VFP, the worse the display contrast will be. To reduce display contrast issues, keep this value low, or program VFP = 0 on STN displays.
15:10	VSW	**Vertical Synchronization (Pulse) Width** VSW is the width of the LCDFP signal. VSW is specified in terms of the number of horizontal synchronization lines (LCDLP pulses). VSW = (LCDLP Periods) – 1 STN modes: This field can be programmed in STN modes, but the value has no effect upon the vertical sync width. The vertical sync width will always be one line regardless of what is written to this field. Programming any other value than zero causes VFP to be enlarged by whatever is programmed in this field. Therefore program VSW = 0 for STN LCDs.

Table 11-19. TIMING1 Fields (Cont'd)

BITS	FIELD NAME	DESCRIPTION
9:0	LPP	**Lines Per Panel** LPP specifies the number of active lines (rows of pixels) per panel. The LPP bit field is a 10-bit value allowing between 1 and 1,024 lines. LPP = (Active Lines) – 1 Dual-Panel displays: The two panels in dual-panel displays are assumed to be of identical sizes. For dual-panel displays, the LPP bit field should be programmed to describe either the upper or the lower panel, and not be doubled. ALI Active modes: LPP and LCDREV must be considered together. Program LPP to an odd number of lines for AD-TFT and HR-TFT panels. AD-TFT and HR-TFT displays utilize the LCDREV signal as an AC bias signal. The LCDREV signal oscillates, driven HIGH during one frame and driven LOW during the next frame. To avoid long-term LCD damage, the AC bias applied to any line of an LCD panel should average to a net 0 VDC. When correctly programmed, the LCDREV signal will be HIGH for a line during one frame and LOW for the same line during the next frame. If the LPP bit field specifies an even number of Lines Per Panel, this oscillation will be mismatched to the display's lines and the lines will not receive a net 0 VDC bias. See ALITIMING1:REVDEL for additional information.

11.3.4 LCD Timing 2 Register (TIMING2)

The TIMING2 register controls the generation of the CLCDC clocks for the LCD panel. Table 11-21 details this register's bit assignments.

Table 11-20. TIMING2 Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///					BCD	CPL									
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///	IOE	IPC	IHS	IVS	ACB					///	PCD				
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RO	RW	RW	RW	RW	RW
ADDR	0xFFFF4008															

Table 11-21. TIMING2 Fields

BITS	FIELD NAME	DESCRIPTION
31:27	///	**Reserved** Reading returns 0. Write Reset Value.
26	BCD	**Bypass Pixel Clock Divider** 1 = Bypass the pixel clock divider logic 0 = Use the pixel clock divider logic See the description of the PCD bit field, below.
25:16	CPL	**Clocks Per Line** CPL specifies the number of LCDDCLK pulses fed to the LCD panel during each horizontal line. TIMING2:CPL and TIMING0:PPL work together; both must be programmed correctly in order for the CLCDC to function correctly. TFT panels: CPL = (PPL – 1) 4-bit mono STN panels: CPL = ((PPL/4) – 1) 8-bit mono STN panels: CPL = ((PPL/8) – 1) Color STN panels: CPL = (((3 x PPL) / 8) – 1)
15	///	**Reserved** Reading returns 0. Write Reset Value.
14	IOE	**Invert Output Enable** IOE applies only to TFT modes and should be programmed to 0 for all other modes. In the TFT mode, the LCDENAB pin indicates to the LCD panel that valid display data is available. IOE selects the active polarity of this output enable signal. In the TFT mode, data is driven onto the LCD data lines at the programmed edge of LCDDCLK when LCDENAB is asserted. 1 = The LCDENAB output pin is active LOW 0 = The LCDENAB output pin is active HIGH
13	IPC	**Invert Panel Clock** IPC selects the active edge of the LCDDCLK signal. 1 = Data is driven on the LCD data lines on the falling-edge of LCDDCLK 0 = Data is driven on the LCD data lines on the rising-edge of LCDDCLK
12	IHS	**Invert Horizontal Synchronization** IHS selects the polarity of the LCDLP signal. 1 = The LCDLP pin is active LOW 0 = The LCDLP pin is active HIGH
11	IVS	**Invert the Vertical Synchronization Signal** IVS selects the polarity of the LCDFP signal. 1 = LCDFP is active LOW 0 = LCDFP is active HIGH

Table 11-21. TIMING2 Fields (Cont'd)

BITS	FIELD NAME	DESCRIPTION
10:6	ACB	**AC Bias Signal Frequency** ACB sets the frequency of the LCDENAB signal. STN modes: ACB applies to the CLCDC when it is operating in the STN mode. STN displays require periodic reversal of the pixel voltages in order to prevent damage to the STN panel due to DC charge accumulation. Program this field to select the required number of line clocks (the LCDLP signal) between each toggle of the AC bias signal (LCDENAB). ACB = (line clocks) – 1 TFT modes: This field has no effect if the CLCDC is operating in TFT mode because the LCDENAB pin is instead utilized for a Data Enable signal.
5	///	**Reserved** Reading returns 0. Write Reset Value.
4:0	PCD	**Panel Clock Divisor** Program this field to select the LCD panel clock frequency (LCDDCLK frequency) from the input CLCDC CLOCK frequency. LCDDCLK = (CLCDC CLOCK)/(PCD+2) Mono STN modes: LCDDCLK for mono STN panels with a four- (or eight-) bit interface should be programmed to be 1/4 (or 1/8) the desired individual pixel clock rate. Color STN modes: Color STN displays receive multiple pixels during each clock cycle. The pixel data for Color STN displays is stored and transferred in packed format, with each pixel represented by three bits (R,G and B). Therefore, one byte contains the pixel data for 2 2/3 pixels (RGB,RGB,RG) and three bytes contain the pixel data for eight complete pixels. For Color STN panels, each LCDDCLK cycle transfers one byte, containing 2 2/3 pixels, to the panel. LCDDCLK should be programmed to be as close as possible to 3/8 the desired individual pixel clock rate. See Table 11-22 for additional STN mode restrictions. TFT mode: For TFT displays, the pixel clock divider can be bypassed by programming TIMING2:BCD = 1.

NOTE: As shown in Figure 11-4, the CLCDC CLOCK signal is fed from the LH79520 clock-generation circuitry to the CLCDC. The CLCDC CLOCK is an input to the CLCDC. Data path latency restricts the usable minimum values of the Panel Clock Divider (PCD) bit field when operating in STN modes, shown in Table 11-22.

Table 11-22. TIMING2:PCD Restrictions in STN Modes

PANEL	TYPE	INTERFACE	PCD MINIMUM VALUE
Single	Color	(All)	PCD = 1 (LCDDCLK = CLCDC CLOCK/3)
Dual	Color	(All)	PCD = 4 (LCDDCLK = CLCD CLOCK/6)
Single	Mono	4-bit	PCD = 2 (LCDDCLK = CLCDC CLOCK/4)
Dual	Mono	4-bit	PCD = 6 LCDDCLK = CLCDC CLOCK/8)
Single	Mono	8-bit	PCD = 6 (LCDDCLK = CLCDC CLOCK/8)
Dual	Mono	8-bit	PCD = 14 (LCDDCLK = CLCDC CLOCK/16)

11.3.5 LCD Upper Panel Base Address Register (UPBASE)

The UPBASE register must be programmed with the base address of the upper panel's frame buffer. The UPBASE register is used for TFT displays, single-panel STN displays, and the upper panel of dual-panel STN displays. The value in the UPBASE register is used for Color LCD DMA operations (frame buffer-to-panel). Also see the LCD Lower Panel Base address register. Both registers contain frame buffer base addresses.

Software must initialize UPBASE (and LPBASE, for dual panels) prior to enabling the CLCDC. The value in UPBASE can be changed in mid-frame to allow double-buffered video displays to be created.

The value in the UPBASE register is copied to the UPCUR register upon each LCD vertical synchronization. This event sets STATUS:BUI and can optionally generate an interrupt. The interrupt can be used to trigger reprogramming the base address when generating double-buffered video.

Table 11-17 defines the bit fields in the UPBASE register.

Table 11-23. UPBASE Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	A31	A30	A29	A28	A27	A26	A25	A24	A23	A22	A21	A20	A19	A18	A17	A16
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	0xFFFF40010															

Table 11-24. UPBASE Fields

BITS	FIELD NAME	DESCRIPTION
31:2	A31:A2	**LCD Upper Panel Base Address** This is the start address of the upper panel frame data stored in memory and is word-aligned.
1:0	A1:A0	**A1 and A0** These two bits read as 0 and must always be written as 0.

11.3.6 LCD Lower Panel Base Address Register (LPBASE)

The LPBASE register is used for the lower panel of dual-panel STN displays and must be programmed with the base address of the lower panel's frame buffer. LPBASE is used for Color LCD DMA operations (frame buffer-to-panel). Also see the LCD Upper Panel Base address register. Both of these registers contain frame buffer base addresses.

When driving dual panels, software must initialize LPBASE prior to enabling the CLCDC. The value in LPBASE can be changed in mid-frame to allow double-buffered video displays to be created.

The value in the LPBASE register is copied to the LPCUR register upon each LCD vertical synchronization. This event sets STATUS:BUI and can optionally generate an interrupt. The interrupt can trigger reprogramming the base address when generating double-buffered video.

Table 11-19 explains the bit fields in the LPBASE register.

Table 11-25. LPBASE Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	A31	A30	A29	A28	A27	A26	A25	A24	A23	A22	A21	A20	A19	A18	A17	A16
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	0xFFFF4014															

Table 11-26. LPBASE Register Bit Fields

BITS	FIELD NAME	DESCRIPTION
31:2	A31:A2	**LCD Lower Panel Base Address** This is the start address of the lower panel frame data in memory and is word-aligned.
1:0	A1:A0	**A1 and A0** These two bits read as 0 and must always be written as 0.

11.3.7 LCD Interrupt Enable Register (INTREN)

The INTREN register enables and disables raw CLCDC interrupts.

The bits in the INTREN register are logically ANDed with the corresponding raw interrupt STATUS bit values to be passed to the INTERRUPT register. Thus, the masked interrupt states are reflected in the INTERRUPT register.

Table 11-28 shows the bit assignments for the INTREN register.

Table 11-27. INTREN Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///											MBEIEN	VCIEN	BUIEN	FUIEN	///
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RO
ADDR	0xFFFF4018															

Table 11-28. INTREN Fields

BITS	FIELD NAME	DESCRIPTION
31:5	///	**Reserved** Reading returns 0. Write Reset Value.
4	MBEIEN	**Bus Master ERROR Interrupt Enable** 1 = Interrupt enabled 0 = Interrupt disabled
3	VCIEN	**Vertical Compare Interrupt Enable** 1 = Interrupt enabled 0 = Interrupt disabled
2	BUIEN	**Next Base Update Interrupt Enable** 1 = Interrupt enabled 0 = Interrupt disabled
1	FUIEN	**FIFO Underflow Interrupt Enable** 1 = Interrupt enabled 0 = Interrupt disabled
0	///	**Reserved** Reading returns 0. Write Reset Value.

11.3.8 LCD Control Register (CONTROL)

The CONTROL register selects the CLCDC's mode of operation. Table 11-30 shows the bit assignments for the CONTROL register.

Note that the operating mode (color/mono, bits-per-pixel, etc.) can only be changed between frames to avoid corruption of the current frame data. To ensure this is done properly, use the Vertical Compare Interrupt to detect that the current frame is complete. To do this, follow these steps when changing operating mode:

1. Program the CONTROL: VCI bit to 0b00 to generate the Vertical Compare Interrupt on entry to the Sync State. Enable this interrupt by programming the INTREN:VCIEN bit to 1.
2. Program the CONTROL:LCDEN bit to 0. The controller will complete the current frame before sampling this bit.
3. Wait for the Vertical Compare Interrupt. Upon assertion of the interrupt, program the new mode (e.g. from color to monochrome using the CONTROL:BW bit).
4. Then program the CONTROL:LCDEN bit to 1 and clear the Vertical Compare Interrupt by writing a 1 to the STATUS:VCI bit. The CLCDC will resume operating, using the newly programmed parameters.

Table 11-29. CONTROL Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															WATERMARK
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///		VCI		PWR	BEPO	BEBO	BGR	DUAL	MONO8L	TFT	BW	BPP			LCDEN
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	0xFFFF401C															

Table 11-30. CONTROL Fields

BITS	FIELD NAME	DESCRIPTION
31:17	///	**Reserved** Reading returns 0. Write Reset Value.
16	WATERMARK	**LCD DMA FIFO Watermark Level** 1 = Requests data when either of the two DMA FIFOs have eight or more empty locations. 0 = Requests data when either of the two DMA FIFOs have four or more empty locations.
15:14	///	**Reserved** Reading returns 0. Write Reset Value.
13:12	VCI	**LCD Vertical Compare** Program to generate an interrupt at: 00 = start of vertical synchronization 01 = start of back porch 10 = start of active video 11 = start of front porch
11	PWR	**LCD Power Enable** 1 = LCD is on when LCDEN = 1 0 = LCD is off
10	BEPO	**Big-Endian Pixel Ordering** The BEPO bit selects between little and big-endian pixel packing for 1, 2 and 4 bpp display modes. The BEPO bit has no effect on 8 or 16 bpp pixel formats. 1 = Big-endian pixel ordering within a byte 0 = Little-endian ordering within a byte
9	BEBO	**Big-Endian Byte Ordering to the LCD** 1 = Big-endian byte order 0 = Little-endian byte order
8	BGR	**RGB or BGR Format Selection** 1 = Bits 14:10 and 4:0 swapped (blue and red swapped) 0 = Display data normal output
7	DUAL	**Dual Panel STN LCD** 1 = Dual panel LCD is in use 0 = Single panel LCD is in use
6	MONO8L	**Monochrome LCD** LCD is Monochrome with an 8-bit interface. This bit controls whether a monochrome STN LCD uses a 4 or an 8-bit parallel interface. It should be programmed to '0' for all other types of displays. 1 = Mono LCD uses 8-bit interface 0 = Mono LCD uses 4-bit interface
5	TFT	**TFT LCD** 1 = LCD is TFT — do not use grayscaler 0 = LCD is an STN display — use grayscaler
4	BW	**Monochrome STN LCD** LCD is Monochrome (Black and White) STN. This bit has no effect in TFT mode. 1 = STN LCD is monochrome 0 = STN LCD is color

Table 11-30. CONTROL Fields (Cont'd)

BITS	FIELD NAME	DESCRIPTION
3:1	BPP	**LCD Bits-Per-Pixel** 000 = 1 BPP 001 = 2 BPP 010 = 4 BPP 011 = 8 BPP 100 = 16 BPP 101 = Invalid 110 = Invalid 111 = Invalid
0	LCDEN	**Color LCD Controller Enable** LCD displays usually require that their logical signals be operating before the high voltages are applied to the display. Thus, the LCDVDDEN output signal is not asserted unless both the LCDEN and PWR bit fields have been programmed to 1. Most LCD displays require that the controller be enabled (LCDEN = 1) approximately 20 ms before power is applied to the LCD (PWR = 1). Most LCD displays also specify that the power-down sequence be the reverse of the power-up sequence. 1 = Color LCD Controller enabled 0 = Color LCD Controller disabled

11.3.9 Interrupt Status Register (STATUS)

The STATUS register provides the status of the raw LCD interrupts. The STATUS register contains 4 interrupt flags. A bit value of 1 indicates that the corresponding interrupt is asserted, even if not enabled in the INTREN register. STATUS and INTREN are logically ANDed to derive the masked interrupt values in the INTERRUPT register.

Interrupts are cleared by writing a 1 to the corresponding bit. Writing a 0 has no effect.

Table 11-32 shows the bit assignments for the STATUS register.

Table 11-31. STATUS Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///											MBEI	VCI	BUI	FUI	///
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW
ADDR	0xFFFF4020															

Table 11-32. STATUS Fields

BITS	FIELD NAME	DESCRIPTION
31:5	///	**Reserved** Reading returns 0. Write Reset Value.
4	MBEI	**AMBA AHB Master Bus Error Status** Indicates that the CLCDC AHB master has encountered a bus error response from a slave. 1 = Interrupt asserted 0 = Interrupt cleared
3	VCI	**Vertical Compare** Set to 1 when one of the four vertical regions selected in the CONTROL register is reached. 1 = Interrupt asserted 0 = Interrupt cleared
2	BUI	**LCD Next Base Address Update** Mode dependent; set to 1 when the Current Base Address registers have been successfully updated with the data from Next Address registers. Signifies that a new Next Address can be loaded if double buffering is in use. 1 = Interrupt asserted 0 = Interrupt cleared
1	FUI	**FIFO Underflow** Set to 1 when either the upper or lower DMA FIFOs have been accessed when empty, resulting in an underflow condition 1 = Interrupt asserted 0 = Interrupt cleared
0	///	**Reserved** Reading returns 0. Write Reset Value.

11.3.10 INTERRUPT Register (INTERRUPT)

The INTERRUPT register provides masked interrupt status indications.

Each of the four bits in the LCD Interrupt register represents a bit-by-bit logical AND of the corresponding bit in the STATUS register with the corresponding bit in the INTREN register. A logical OR of all interrupts is also provided to the Vectored Interrupt Controller (VIC).

Table 11-34 shows the bit field assignments for the INTERRUPT register.

Table 11-33. INTERRUPT Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///											MBEIM	VCIM	BUIM	FUIM	///
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
ADDR	0xFFFF4024															

Table 11-34. INTERRUPT Fields

BITS	FIELD NAME	DESCRIPTION
31:5	///	**Reserved** Reading returns 0. Write Reset Value.
4	MBEIM	**Masked AHB Master Error Interrupt** 1 = Interrupt asserted and enabled 0 = Interrupt cleared
3	VCIM	**Masked Vertical Compare Interrupt** 1 = Interrupt asserted and enabled 0 = Interrupt cleared
2	BUIM	**Masked LCD Next Base Address Update Interrupt** 1 = Interrupt asserted and enabled 0 = Interrupt cleared
1	FUIM	**Masked FIFO Underflow Interrupt** 1 = Interrupt asserted and enabled 0 = Interrupt cleared
0	///	**Reserved** Reading returns 0. Write Reset Value.

11.3.11 LCD Upper Panel Current Address Register (UPCUR)

The UPCUR register contains the present upper-panel LCD DMA address.

This is a read-only register; do not write to this register.

When read, The UPCUR register returns the value of the present upper panel LCD data DMA address. The contents of this register change dynamically, and therefore should be used only as a mechanism to implement a coarse delay.

Table 11-36 shows the bit assignments for the UPCUR register.

Table 11-35. UPCUR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	A31	A30	A29	A28	A27	A26	A25	A24	A23	A22	A21	A20	A19	A18	A17	A16
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
ADDR	0xFFFF4028															

Table 11-36. UPCUR Register Bit Fields

BITS	FIELD NAME	DESCRIPTION
31:0	A31:A0	A31:A0 of the current lower panel data DMA address. Values change dynamically. Read only.

11.3.12 LCD Lower Panel Current Address Register (LPCUR)

The LPCUR register contains the real-time lower-panel LCD DMA address.

This is a read-only register. Do not write to this register.

When read, The LPCUR register returns the value of the present lower panel LCD data DMA address. The contents of this register change dynamically, and therefore should be used only as a mechanism to implement a coarse delay.

Table 11-38 shows the bit assignments for LPCUR.

Table 11-37. LPCUR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	A31	A30	A29	A28	A27	A26	A25	A24	A23	A22	A21	A20	A19	A18	A17	A16
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
ADDR	0xFFFF402C															

Table 11-38. LPCUR Fields

BITS	FIELD NAME	DESCRIPTION
31:0	A31:A0	A31:A0 of the current lower panel data DMA address. Values change dynamically. Read-only; do not write.

11.3.13 LCD Palette Registers (PALETTE)

The LH79520 CLCDC includes 128 PALETTE registers. The LCD Palette registers contain 256 palette entries organized as 128 locations of two entries per register. Each palette entry occupies 16 bits and each register contains two complete palette entries. Values are stored in little-endian format.

Mono STN modes use only the red palette bits [4:1]. Color STN modes use the red palette bits [4:1], the green palette bits [4:1] and the blue palette bits [4:1]. TFT modes with fewer than 16 BPP use all of the palette bits, including the Intensity bit. The 16 BPP TFT mode bypasses the palette and routes data directly to the LCD.

Table 11-40 shows the bit assignments for a PALETTE register, when used with a 5:5:5 + Intensity TFT panel. Table 11-41 shows the arrangement for a 5:6:5 TFT. All PALETTE registers have the same bit fields.

Table 11-39. PALETTE Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	MI	MB4	MB3	MB2	MB1	MB0	MG4	MG3	MG2	MG1	MG0	MR4	MR3	MR2	MR1	MR0
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	I	LB4	LB3	LB2	LB1	LB0	LG4	LG3	LG2	LG1	LG0	LR4	LR3	LR2	LR1	LR0
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	0xFFFF4200 through 0xFFFF43FC															

Table 11-40. PALETTE Register Bit Fields, BGR = 0, 5:5:5 + Intensity TFT

BITS	FIELD NAME	DESCRIPTION
31	MI	**Most Significant Intensity Bit** See the bit 15 description in this table.
30:26	MB4:MB0	**Most Significant Blue Palette Data** See the bit 14:10 description in this table.
25:20	MG4:MG0	**Most Significant Green Palette Data** See the bit 9:5 description in this table.
19:16	MR4:MR0	**Most Significant Red Palette Data** See the bit 4:0 description in this table.
15	LI	**Least Significant Intensity Bit** Unused for STN displays. Can be used as the LSB of the R, G and B inputs to a 6:6:6 TFT display, effectively doubling the number of available colors from 32K to 64K, where each color has two different intensities.
14:10	LB4:LB0	**Least Significant Blue Palette Data** For color STN displays, only the four MSBs (bits 4:1) of each color are used.
9:5	LG4:LG0	**Least Significant Green Palette Data** For color STN displays, only the four MSBs (bits 4:1) of each color are used.
4:0	LR4:LR0	**Least Significant Red Palette Data** For color STN displays, only the four MSBs (bits 4:1) of each color are used. For monochrome STN displays, only the four MSBs (bits 4:1) of the red palette data is used.

Table 11-41. PALETTE Register Bit Fields, BGR = 0, 5:6:5 TFT

BITS	FIELD NAME	DESCRIPTION
31:27	MB4:MB0	Most Significant Blue Palette Data
26:20	MG5:MG0	Most Significant Green Palette Data
19:16	MR4:MR0	Most Significant Red Palette Data
15:11	LB4:LB0	Least Significant Blue Palette Data
10:5	LG5:LG0	Least Significant Green Palette Data
4:0	LR4:LR0	Least Significant Red Palette Data

11.4 ALI Register Reference

The Register Base Address for the ALI is:

ALI Base: 0xFFFE4000 (Formerly known as: HRTFTC Base)

All registers in the ALI must be accessed as a full word. The ALI does not support byte and half-word writes.

11.4.1 ALI Memory Map

Programming of the setup and timing registers is done via the APB interface. The ALI must be configured prior to enabling the CLCDC for the first frame. The register memory map is shown in Table 11-42.

NOTE: This Chapter of the User's Guide has been updated. To aid in backward compatibility, the former register names have been included here in the Register Reference. There is no change to the registers themselves; either in function or location.

Table 11-42. Register Summary

ADDRESS	REGISTER NAME	DESCRIPTION
ALI Base + 0x000	ALISETUP	ALI Setup Register (Formerly known as HRTFTCSetup)
ALI Base + 0x004	ALICONTROL	ALI Control Register (Formerly known as HRTFTCControl)
ALI Base + 0x008	ALITIMING1	ALI Timing Register 1 (Formerly known as HRTFTCTiming1)
ALI Base + 0x00C	ALITIMING2	ALI Timing Register 2 (Formerly known as HRTFTCTiming2)

11.4.2 ALI Register Descriptions

This section lists the definitions for each of the programmable registers that control the ALI.

11.4.3 ALI Setup Register (ALISETUP)

The ALISETUP register allows configuration of the ALI pixels-per-line and setting Bypass or Active mode. The CLCDC feeds LCD data and control signals to the ALI, as described in Section 11.2. If the ALI is operating in its Bypass mode, the panel data is transmitted unchanged to the LCD panel. If the ALI is operating in its Active mode, the CLCDC control signals and data are re-timed for the panel's Row and Column driver ASICs prior to transmission.

This register was formerly known as HRTFTCSetup.

Change the ALI mode only when the CLCDC is disabled. Software should first disable the CLCDC by writing to CONTROL:LCDEN and CONTROL:PWR. See Section 11.3.8 for more information. The bit fields for the ALISETUP register are detailed in Table 11-44.

Table 11-43. ALISETUP Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///			PPL									VRVE	HRVE	CR	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	ALI Base + 0x000															

Table 11-44. ALISETUP Register Bits

BITS	FIELD NAME	DESCRIPTION
31:13	///	**Reserved** Reading returns 0. Write Reset Value.
12:4	PPL	**Pixels Per Line** PPL = (Actual Pixels per line) – 1.
3	VRVE	**Vertical Reverse Video Enable** Reverses video display vertically. 1 = Reverse Scanning 0 = Normal Scanning
2	HRVE	**Horizontal Reverse Video Enable** Reverses video display horizontally. 1 = Reverse Scanning 0 = Normal Scanning
1:0	CR	**Conversion Mode Select** This field selects the conversion mode (on/off) for the ALI. Change the ALI mode only when the CLCDC is disabled. 11 = Invalid 10 = Invalid 01 = Active Mode 00 = Bypass Mode (for STN or TFT panels)

11.4.4 ALI Control Register (ALICONTROL)

The ALICONTROL register controls the generation of the LCDCLS and LCDSPS ALI output signals.

This register should be programmed only after the appropriate CLCDC registers and the ALISETUP register have been programmed. These registers should be programmed in specific sequence. See Section 11.2.1.2 for additional information.

This register was formerly known as HRTFTCControl.

Table 11-46 presents the bit definitions for the ALICONTROL register.

Table 11-45. ALICONTROL Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///														LCDCLSEN	LCDSPSEN
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW
ADDR	ALI Base + 0x004															

Table 11-46. ALICONTROL Register Bit Fields

BITS	FIELD NAME	DESCRIPTION
31:2	///	**Reserved** Reading returns 0. Write Reset Value.
1	LCDCLSEN	**LCDCLS Enable** STN or TFT (Bypass) modes: Reserved Reading returns 0. Values written cannot be read. Active mode: Enables or disables the generation of the LCDCLS (Gate Driver Clock) signal. 1 = LCDCLS signal enabled 0 = LCDCLS signal disabled
0	LCDSPSEN	**LCDSPS Enable** STN or TFT (Bypass) modes: Reserved Reading returns 0. Values written cannot be read. Active mode: Enables or disables the generation of the LCDSPS (Row Reset) signal. 1 = LCDSPS signal is enabled 0 = LCDSPS signal is disabled

11.4.5 ALI Timing 1 Register (ALITIMING1)

The ALITIMING1 register specifies the required delays for Row and Column driver chip control signals LCDLP, LCDREV, LCDPS and LCDCLS.

This register was formerly known as HRTFTCTiming1.

Table 11-48 gives the bit definitions for the ALITIMING1 register.

Table 11-47. ALITIMING1 Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///				PSCLS				REVDEL				LPDEL			
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	ALI Base + 0x008															

Table 11-48. ALITIMING1 Bits

BIT	FIELD NAME	DESCRIPTION
31:12	///	**Reserved** Reading returns 0. Write Reset Value.
11:8	PSCLS	**LCDPS and LCDCLS Delay** Controls the delay in LCDDCLK periods from the first rising edge of the internal CLCDC clock after the leading edge of the internal LCDLP signal (not the LCDLP pin no. 137), to the leading edge of the LCDREV signal. PSCLS = (LCDDCLK periods) – 1
7:4	REVDEL	**Polarity-Reversal Delay*** Controls the delay in LCDDCLK periods from the first assertion of the LCDLP (horizontal sync) signal, to the falling edge of the LCDREV signal. REVDEL = (LCDDCLK periods) – 1
3:0	LPDEL	**LCDLP Delay** Controls the delay in LCDDCLK periods from the first rising edge of the internal CLCDC clock after the leading edge of the internal LCDLP signal (not the LCDLP pin no. 137), to the leading edge of the LCDLP signal. LPDEL = (LCDDCLK periods) – 1

NOTE: *The LCDREV signal generated by the ALI is intended for input to a grayscaler ASIC associated with an AD-TFT or HR-TFT display. In this application, it acts as a type of AC Bias signal, switching at a horizontal-line rate, synchronized to the LCDDCLK signal. The LCDREV signal is not intended to reverse the panel's direction-of-scan. Panels utilizing the LCDREV signal must be programmed to utilize an 'odd' number of horizontal display lines. If the panel is programmed to display an 'even' number of horizontal lines, then the net AC Bias applied to each line of the panel will not average to 0 VDC and the panel can suffer long-term damage.

11.4.6 ALI Timing 2 Register (ALITIMING2)

The ALITIMING2 register specifies the required delays for the panel's Row and Column driver chip control signal LCDSPL.

This register was formerly known as HRTFTCTiming2.

Table 11-50 gives the bit definitions for this register.

Table 11-49. ALITIMING2 Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	SPLDEL							PS2CLS2								
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	ALI Base + 0x00C															

Table 11-50. ALITIMING2 Fields

BITS	FIELD NAME	DESCRIPTION
31:16	///	**Reserved** Reading returns 0. Write Reset Value.
15:9	SPLDEL	**LCDSPL Delay** Controls the delay in LCDDCLK periods of the LCDSPL signal during vertical front and back porches. This field must be programmed to a value greater than the sum of (TIMING0:HSW + TIMING0:HBP). SPLDEL = (LCDDCLK periods) – 1
8:0	PS2CLS2	**LCDSPL and LCDCLS Delay 2** Controls the delay in LCDDCLK periods from the first rising edge of the LCDSPL signal to the trailing edge of the LCDCLS and LCDPS signals. PS2CLS2 = (LCDDCLK periods) – 1

Chapter 12
Timers

The LH79520 MCU includes two Timer modules, each containing two decrementing Timers, for a total of four Timers. This Chapter explains the Timers included in the LH79520. A discussion of the theory of operation of the Timers is followed by descriptions of the programmable registers which configure and control the Timers.

12.1 Theory of Operation

The LH79520 Timers are organized as APB slave peripheral modules, and AHB accesses of the Timer registers occur via the APB, through the APB Bridge. Each APB module contains two Timers. Timer0 and Timer1 comprise one module and receive one common clock from the LH79520 RCPC functional block. Timer2 and Timer3 comprise a second module and receive a second, similar clock. Each Timer can be programmed to generate an interrupt output to the LH79520 VIC, and the output of Timer3 is available at an external LH79520 pin. Figure 12-1 shows how the Timers are integrated into the LH79520 MCU.

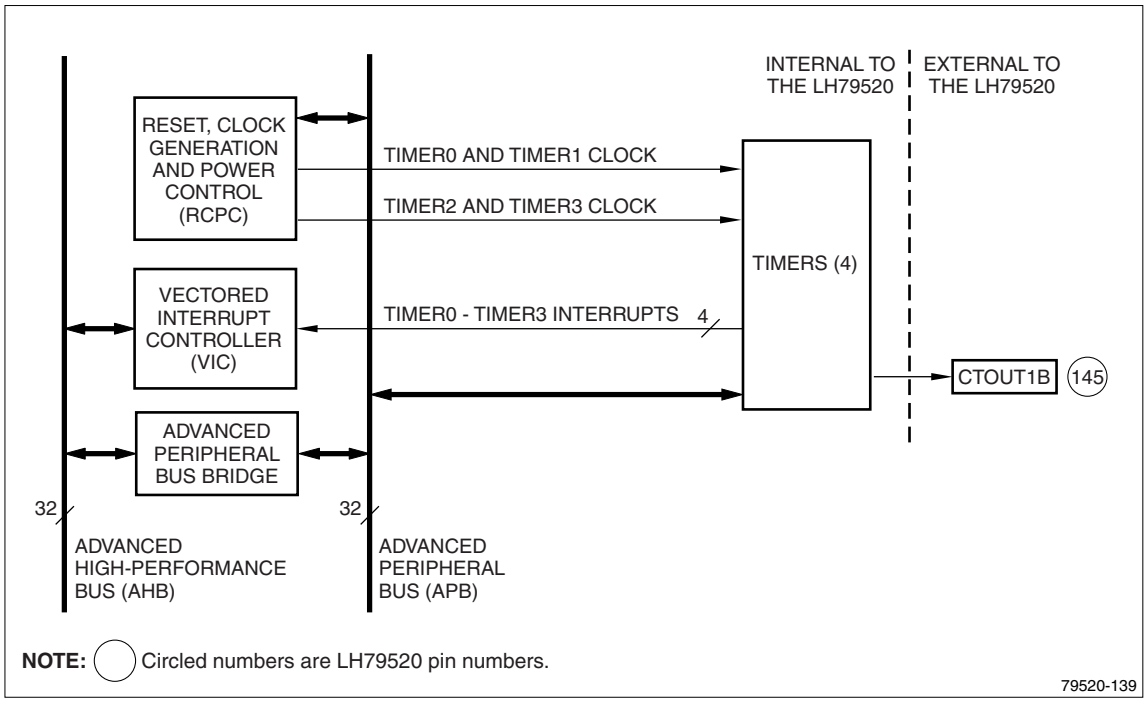


Figure 12-1. LH79520 Timers

The four Timers can be used individually, or the Timers can be cascaded to provide longer countdown periods than would be provided by an individual Timer. When the Timers are cascaded, only the last Timer in the cascaded chain can cause an interrupt. A block diagram of one Timer is shown in Figure 12-2.

Each Timer includes three programmable registers (Control, Load, and Clear), and one read-only register (Value).

Each Timer can be clocked by a signal from the LH79520 RCPC functional block, or the Timer can be clocked by the CASCADE OUT signal from the previous Timer. Generation and control of the Timer clock signals in the RCPC is explained in Chapter 7, Section 7.6.

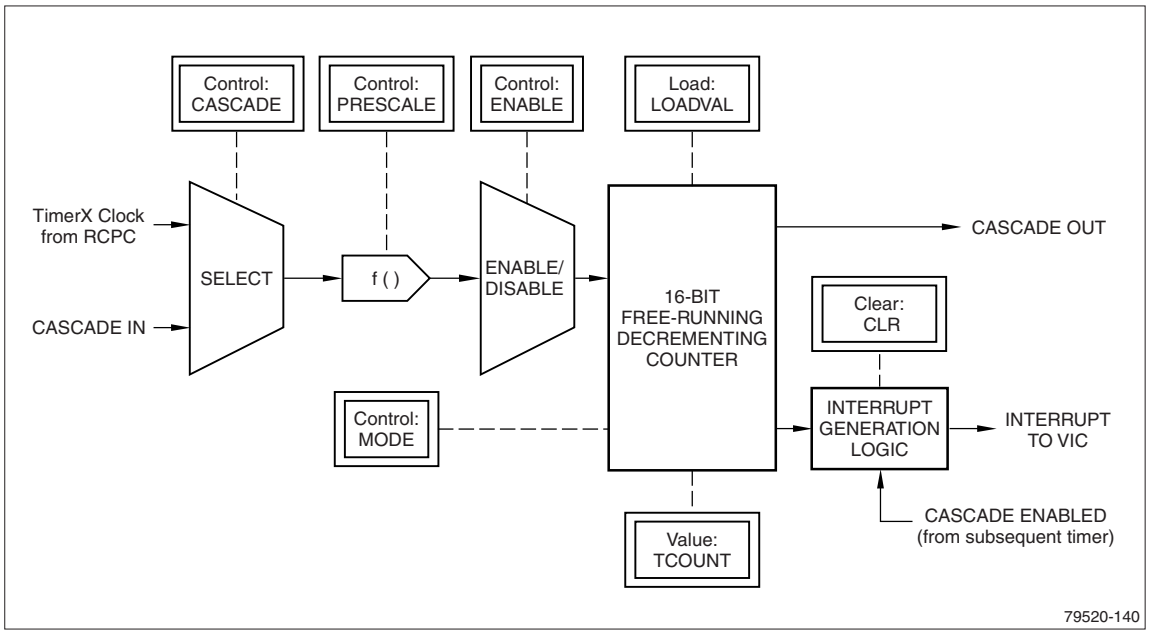


Figure 12-2. Timer Block Diagram

12.2 Timer Modes of Operation

Each Timer may be operated independently, in one of two modes:

- The (default) free-run mode allows a Timer to wrap to its maximum value after being decremented to zero. When a Timer has been decremented to 0x0000, the next cycle of the Timer clock will cause the Timer to wrap to a value of 0xFFFF and resume counting down.
- The Periodic Timer mode allows a Timer to generate an interrupt, or a cascade signal, at a constant interval. The interrupt, or cascade signal, is generated when the Timer is decremented past zero. When a Timer has been decremented to 0x0000, the next clock cycle will cause the Timer to generate an interrupt, or a cascade signal, reload the Timer with a programmed load value, and resume counting down.

12.3 Timer Output Function

The Timer output toggles each time Timer3 reaches zero. Therefore, if the output is to be generated from a count that involves more than one Timer (using cascaded Timers), then the cascade chain must utilize the highest-numbered Timers.

12.4 Cascading the Timers

Each Timer can be programmed to cascade to the subsequent Timer, shown in Figure 12-3. The cascade sequence is fixed in hardware; the Timers must be cascaded in the sequence shown; the Timers cannot be cascaded in a different sequence, such as Timer3 to Timer1.

When a Timer is programmed to operate as a stand-alone timer (to not cascade), the Timer is clocked by a signal generated in the LH79520 RCPC. When a Timer is programmed to cascade, the Timer is clocked by the output of the previous Timer. Enabling the Cascade mode for Timer0 is legal although there is no Timer preceding it; its count will proceed normally.

When a Timer is programmed to cascade, the value in the Timer will be decremented only when all preceding Timers in the cascaded chain roll under from 0x0000 to 0xFFFF (or to the value programmed to the Load register, depending on the programmed Timer mode).

Each Timer can signal the preceding Timer that cascading is enabled. When a Timer is programmed to the Cascade mode, the Timer will block the generation of the interrupt from the preceding Timer, producing only one interrupt for all cascaded Timers. Only the highest-numbered Timer in the cascaded chain will generate an interrupt.

Cascading the Timers increases their range as follows:

- $2^{16} - 1$ when using a single Timer
- $2^{32} - 1$ when cascading two Timers
- $2^{48} - 1$ when cascading three Timers
- $2^{64} - 1$ when cascading all four Timers

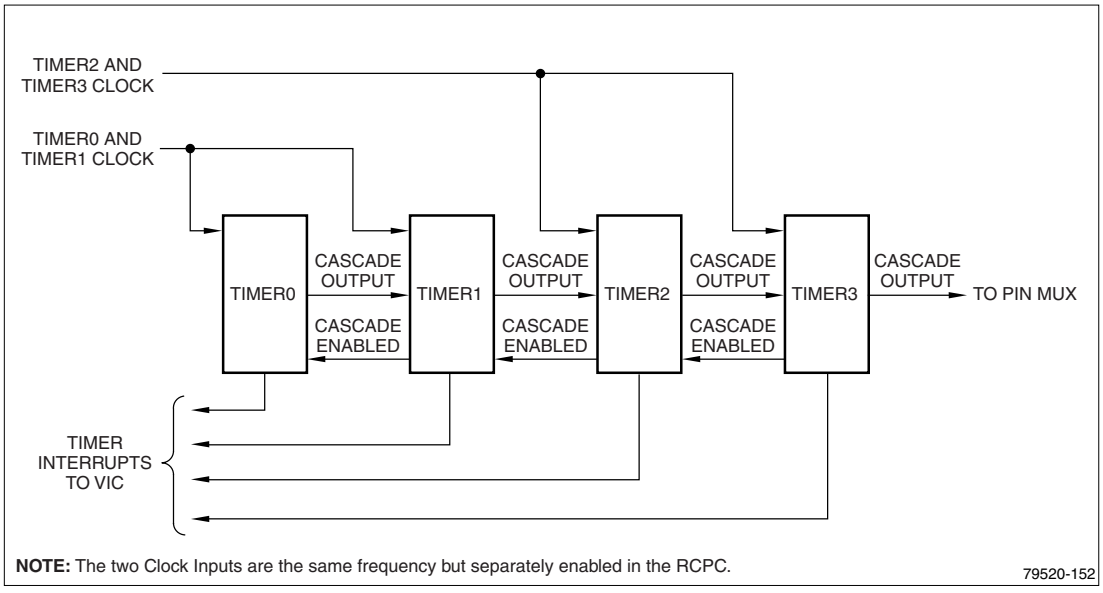


Figure 12-3. Cascaded Timers

12.5 Timer Interrupts

Each of the four LH79520 Timers can generate an interrupt output to the LH79520 VIC. Internal interrupt lines 17, 18, 19, and 20 are driven by TIMER0, TIMER1, TIMER2 and TIMER3, respectively. Any of these lines may be assigned to any of the vectored interrupts provided by the VIC, or left as a default interrupt.

While a Timer is running, the Timer always generates an interrupt signal to the VIC when wrapping past a value of zero. If a Timer interrupt is not desired, the interrupt must be masked in the VIC.

Cascading a particular Timer suppresses interrupts generated by the preceding Timer in the cascaded chain. The final Timer in a cascaded chain will generate an interrupt.

12.6 Timer Programmer's Model

The Register Base Addresses for the four LH79520 Timers are:

Timer0Base: 0xFFFC4000

Timer1Base: 0xFFFC4020

Timer2Base: 0xFFFC5000

Timer3Base: 0xFFFC5020

All registers in the Timers must be accessed as a full word. The Timers do not support byte and half-word writes. Programmers should ensure that the system clock is enabled before programming any registers.

12.7 Timer Register Summary

Table 12-1 presents a summary of all of the programmable registers for the four LH79520 Timers.

Table 12-1. Timer Register Summary

ADDRESS	REGISTER NAME	DESCRIPTION
Timer0Base + 0x000	Load	Timer0 Load
Timer0Base + 0x004	Value	Timer0 Value
Timer0Base + 0x008	Control	Timer0 Control
Timer0Base + 0x00C	Clear	Timer0 Clear
Timer1Base + 0x000	Load	Timer1 Load
Timer1Base + 0x004	Value	Timer1 Value
Timer1Base + 0x008	Control	Timer1 Control
Timer1Base + 0x00C	Clear	Timer1 Clear
Timer2Base + 0x000	Load	Timer2 Load
Timer2Base + 0x004	Value	Timer2 Value
Timer2Base + 0x008	Control	Timer2 Control
Timer2Base + 0x00C	Clear	Timer2 Clear
Timer3Base + 0x000	Load	Timer3 Load
Timer3Base + 0x004	Value	Timer3 Value
Timer3Base + 0x008	Control	Timer3 Control
Timer3Base + 0x00C	Clear	Timer3 Clear

12.8 Timer Register Descriptions

This section of this chapter of this User's Guide lists and describes the registers that configure the LH79520 Timer modules.

12.8.1 Load

The Timer Load register can be programmed with value which will be loaded to the Timer when the Timer is enabled, or when the Timer underflows, if operating in the periodic mode.

The Load register is a read-write register. Table 12-3 describes the bit fields in the Load register.

Table 12-2. Load Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	LOADVAL															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	TimerBase + 0x000															

Table 12-3. Load Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:16	///	**Reserved** Write the Reset value.
15:0	LOADVAL	**Load Value** A 16-bit value. Reset = 0.

12.8.2 Value

A read of the Timer Value register returns the current value of the Timer.

The Value register is a read-only register. Table 12-5 describes the bit fields in the Value register.

Table 12-4. Value Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	CURRENTCOUNT															
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
ADDR	TimerBase + 0x004															

Table 12-5. Value Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:16	///	**Reserved** Value returned on a read will be unpredictable.
15:0	CURRENTCOUNT	The current value of the Timer. Reset = 0xFFFF.

12.8.3 Control

Each Timer Control register controls one Timer.

Bit fields in the Control register enable/disable the Timer, set the mode, enable/disable cascading, and select the clock prescale for the Timer. Table 12-7 describes the bit fields in the Control register.

Table 12-6. Control Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								ENABLE	MODE	///	CASCADE	PRESCALE		///	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	TimerBase + 0x008															

Table 12-7. Control Register

BITS	FIELD NAME	FUNCTION
31:8	///	**Reserved** Write the Reset value.
7	ENABLE	**Enable** 0 = Timer Disabled. The Timer does not decrement (reset). 1 = Timer Enabled. The Timer decrements on the rising edge of the input clock signal, as defined by the CASCADE and PRESCALE bit fields.
6	MODE	**Mode** 0 = Free-Running mode (reset). When the Timer decrements to zero, the Timer underflows to 0xFFFF and continues to decrement. 1 = Periodic mode. When the Timer decrements to zero, the Timer is automatically reloaded with the value previously programmed to the-Load register.
5	///	**Reserved** Write the Reset value.
4	CASCADE	0 = Cascade disabled, Timer runs as stand-alone down-counter. 1 = Timer is enabled for cascading. Setting the Cascade bit on Timer0 is legal, even though there is no Timer preceding it; the timer will count normally.
3:2	PRESCALE	**Input Clock Prescale** Reset = 0b00 **Clock Division** **Stages of Prescale** 0b11 = /// /// 0b10 = 256 8 0b01 = 16 4 0b00 = 1 0
1:0	///	**Reserved** Write the Reset value.

12.8.4 Clear

A write access to the Timer Clear register clears the interrupt associated with this Timer. This is a write-only register. Table 12-9 describes the bit fields in the Clear register.

Table 12-8. Clear Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET																
TYPE	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	CLR															
RESET																
TYPE	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
ADDR	TimerBase + 0x00C															

Table 12-9. Clear Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:16	///	**Reserved** Write-only. Value returned on a read will be unpredictable.
15:0	CLR	**Clear** A write of any value to this location clears the interrupt generated by this Timer. The value written is not retained in this register.

Chapter 13
Watchdog Timer (WDT)

This Chapter explains the programmable Watchdog Timer (WDT) included in the LH79520 MCU. A discussion of the theory of operation of the WDT is followed by descriptions of the programmable registers that control the WDT.

13.1 Theory of Operation

The Watchdog Timer (WDT) is a hardware protection against malfunctions. The WDT in the LH79520 is a programmable down-counter decremented by the APB clock signal, PCLK. If enabled, the WDT down-counter must be reloaded by software at regular intervals. Failure to reload the counter before it decrements to zero can cause an interrupt or a system reset, or both. Figure 13-1 shows that the LH79520 WDT is programmed via the APB, clocked by PCLK (the APB clock and locked to HCLK in the LH79520), can issue an interrupt signal to the VIC, and can issue a WDT Reset signal to the RCPC functional block.

Figure 13-1 shows a subset of the functional blocks in the LH79520. Readers may also wish to refer to Figure 2-1, a simplified block diagram of the entire LH79520 MCU.

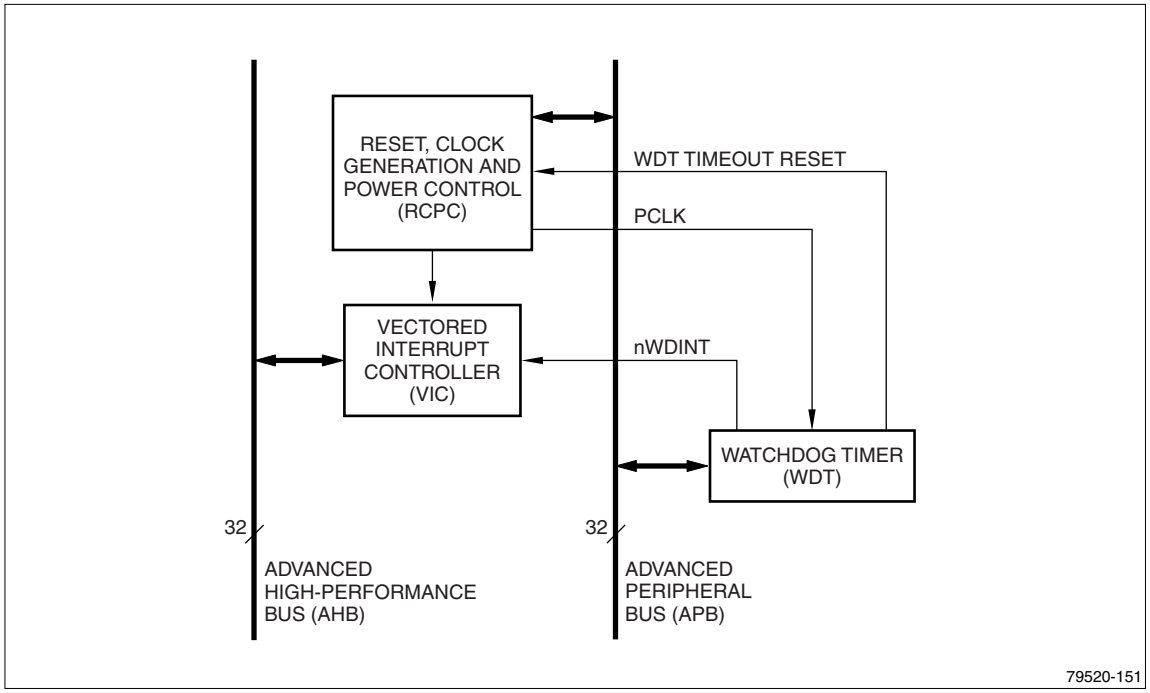


Figure 13-1. LH79520 Watchdog Timer Block Diagram

13.2 WDT Clock

The LH79520 WDT is driven by the PCLK signal, which is the clock for the APB. PCLK is locked to HCLK, the AHB clock in the LH79520. See Chapter 7, Section 7.9 for additional information about PCLK generation and control.

13.3 WDT Modes of Operation

The LH79520 WDT is disabled at system reset and offers two modes of operation when enabled:

- In the first mode of operation, WDCTLR:RSP = 0, failure to reload the WDT counter will cause a system reset if the WDT counter is permitted to decrement to zero (expire). In this mode the WDT expires only once before the reset occurs.
- The second mode of operation, WDCTLR:RSP = 1, requires that the WDT be permitted to expire twice in succession. In this mode of operation, the first expiration will generate a WDT interrupt and will also automatically reload the programmed count (WDCTLR:TOP) to the WDT counter. If the counter is permitted to expire a second time, the WDT will then generate a system reset.

13.4 WDT Resets

The WDT can be programmed to send a reset signal to the LH79520 Reset System when the WDT counter expires. Software can test a bit in a WDT register to establish whether or not this type of reset has occurred.

Applying power to the LH79520 sets a read-only bit in the WDTSTR register. This bit can be cleared only by the WDT. Applying power to the LH79520 causes WDTSTR:nWDRES (the nWDRES bit in the WDTSTR register) to be set to '1'. This bit will remain set unless the WDT generates a reset. This bit will remain set even if other LH79520 systems cause system resets for other reasons. This bit will be cleared to '0' only if the WDT generates a reset (assuming that the LH79520 is continuously powered). When the WDT generates a reset, the WDT clears WDTSTR:nWDRES to '0'. Software can test this read-only bit to establish whether or not the WDT has caused an LH79520 reset. See Chapter 7 – Reset, Clock Generation, and Power Control (RCPC), for additional information about LH79520 resets.

13.5 WDT Interrupts

The WDT can be programmed to generate a system reset or an interrupt (or both) when the WDT counter expires. The status of the WDT interrupt can be read from the WDTSTR register. The interrupt signal generated by the WDT is input to the LH79520 VIC (Vectored Interrupt Controller), where the interrupt can be used to generate either an IRQ or an FIQ interrupt. See Chapter 9 – Exceptions and Interrupts for more information about the VIC.

13.6 Reading the WDT Counter

The present value of the WDT down-counter is available in four adjacent registers, WDCNT[3:0]. Each register contains 1 byte (8-bits) of the total 32-bit value. Concatenating the values in the WDCNT3, WDCNT2, WDCNT1 and WDCNT0 registers in order to obtain the complete present count is possible but cannot be accomplished in 'atomic' fashion. The concatenation requires four successive read operations and the WDT may alter the value in the first register while subsequent registers are being read, rendering the result inaccurate.

13.7 WDT Programmer's Model

The Register Base Address for all of the programmable LH79520 WDT registers is:

WDTBase: 0xFFFE3000

All registers in the WDT must be accessed as a full word. The WDT does not support byte and half-word writes.

13.8 WDT Register Summary

Table 13-1 summarizes the programmable WDT registers. Tables 13-2 through 13-11 explain the individual registers and their bit fields in detail.

Table 13-1. WDT Register Summary

ADDRESS	REGISTER NAME	DESCRIPTION
WDTBase + 0x00	WDCTLR	Watchdog Control Register
WDTBase + 0x04	WDCNTR	Watchdog Counter Reset Register
WDTBase + 0x08	WDTSTR	Watchdog Status Register
WDTBase + 0x0C	WDCNT0	WDT Counter Section 0 (least-significant byte)
WDTBase + 0x10	WDCNT1	WDT Counter Section 1
WDTBase + 0x14	WDCNT2	WDT Counter Section 2
WDTBase + 0x18	WDCNT3	WDT Counter Section 3 (most significant byte)

13.9 WDT Register Descriptions

This section of this chapter of this User's Guide lists and describes the registers that configure the LH79520 Watchdog Timer (WDT) peripheral.

13.9.1 Watchdog Control Register

The WDTCRLR register controls the operation of the LH79520 WDT.

Bit fields in the WDTCRLR register establish the WDT timeout count (a quantity of PCLK cycles), determine the WDT response if the counter is permitted to decrement to zero (expire) and enable or disable the WDT.

Software should program the WDCTLR register with the WDT counter pre-load. The pre-load value representing the desired quantity of APB (PCLK) clock cycles with which the watchdog counter pre-load. This value is known as the timeout count.

When writing a timeout count to this register, the value written will not take effect until a counter-reset command has been issued (via the WDCNTR register) or until the count has been allowed to decrement to zero.

Table 13-2. WDCTLR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								TOP				FRZ	///	RSP	EN
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	*
ADDR	WDTBase + 0x00															

NOTE: *The EN bit is reset to RW but this permission can be altered by the FRZ bit. See Table 13-3.

Table 13-3 explains the bit fields in the WDTCTLR register.

Table 13-3. WDCTLR Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:8	///	**Reserved** Write the Reset value.
7:4	TOP	**Timeout Period** 1 of 16 possible values: 0xF = 2^{31} clock ticks 0xE = 2^{30} clock ticks 0xD = 2^{29} clock ticks 0xC = 2^{28} clock ticks 0xB = 2^{27} clock ticks 0xA = 2^{26} clock ticks 0x9 = 2^{25} clock ticks 0x8 = 2^{24} clock ticks 0x7 = 2^{23} clock ticks 0x6 = 2^{22} clock ticks 0x5 = 2^{21} clock ticks 0x4 = 2^{20} clock ticks 0x3 = 2^{19} clock ticks 0x2 = 2^{18} clock ticks 0x1 = 2^{17} clock ticks 0x0 = 2^{16} clock ticks (reset)
3	FRZ	**Freeze (Lock) the EN bit** Writing this bit to '1' when the WDT is enabled makes the EN bit read-only. FRZ = 1 stops the EN bit from being cleared. Locking the EN bit avoids the possibility that an accidental write operation could disable the WDT. Once set to '1', FRZ can be cleared to '0' only by a global reset.
2	///	**Reserved** Write the Reset value.
1	RSP	**Timeout Response** Determines how the WDT responds when the WDT is allowed to decrement to zero (to its timeout): 0 = A system reset will be generated whenever the WDT times out. 1 = An interrupt will be generated when the first timeout occurs. If the interrupt is not cleared prior to the occurrence of the second timeout, then a system reset will be generated when the second timeout occurs.
0	EN	**Watchdog Enable** 0 = WDT disabled (reset). The WDT counter does not decrement and no interrupts or system resets will be generated by the WDT. 1 = WDT enabled. The WDT counter does decrement. A reset or interrupt will be generated by the WDT if the watchdog count is not periodically re-loaded to avoid permitting the counter to be decremented to zero.

13.9.2 Watchdog Counter Reset Register

Writes to the WDCNTR register restart the WDT operation.

Writing the key value of 0x1984 to this register restarts the WDT operation. When the key value is written to this register:

- The Watchdog timer is reloaded with the count specified by WDTCLR:TOP
- Interrupts that have been generated by the WDT are cleared.

Table 13-5 describes the bit fields in the write-only WDCNTR register.

Table 13-4. WDCNTR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET																
TYPE	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	WDT_RESET															
RESET																
TYPE	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
ADDR	WDTBase + 0x04															

Table 13-5. WDCTLR Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:16	///	**Reserved** Write '0'. Write-only. Reads unpredictable.
15:0	WDT_RESET	Writing the key value 0x1984 to this register restarts the WDT. This is a write-only register. Reads are unpredictable.

13.9.3 Watchdog Status Register

Reads of the WDTSTR return the status of the WDT interrupt.

The WDTSTR register is a read-only register which is reset to 0x0040. The WDTSTR register contains the watchdog interrupt status, the system reset status and two copies of the watchdog response bit, WDCTLR:RSP.

Table 13-7 describes the bit fields in the WDTSTR register.

Table 13-6. WDTSTR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								nWDINT	///	RSP		///			
RESET	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
ADDR	WDTBase + 0x08															

Table 13-7. WDTSTR Register Bits

BITS	FIELD NAME	FUNCTION
31:8	///	**Reserved** Reads unpredictable.
7	nWDINT	**WDINT Interrupt Status** 0 = nWDINT has been triggered 1 = nWDINT has not been triggered (reset)
6	///	**Reserved** Reading this bit returns 1. Write the reset value.
5:4	RSP	**Response Status** A copy of WDCTLR:RSP
3:0	///	**Reserved** Reads unpredictable.

13.9.4 Watchdog Counter Section 0 Register

The WDCNT0 register contains the low byte of the present value of the watchdog timer.

This register is one of a set of four registers. Each register contains one byte of the present watchdog timer count. Together, these four read-only registers contain the current value of the watchdog timer count. WDCNT0:CounterSection0 is reset to 0x00.

Concatenating the values in the WDCNT0, WDCNT1, WDCNT2 and WDCNT3 registers in order to obtain the complete, present count cannot be accomplished in 'atomic' fashion. The concatenation requires four successive read operations and the WDT may alter the first register while subsequent registers are being read, rendering the result inaccurate.

Table 13-8. WDCNT0 Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								CounterSection0							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
ADDR	WDTBase + 0x00C															

13.9.5 Watchdog Counter Section 1 Register

The WDCNT1 register contains a byte of the present value of the watchdog timer.

This register is one of a set of four registers. Each register contains one byte of the present watchdog timer count. Together, these four read-only registers contain the current value of the watchdog timer count. WDCNT1:CounterSection1 is reset to 0x00.

Table 13-9. WDCNT1 Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								CounterSection1							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
ADDR	WDTBase + 0x010															

13.9.6 Watchdog Counter Section 2 Register

The WDCNT2 register contains a byte of the present value of the watchdog timer.

This register is one of a set of four registers. Each register contains one byte of the present watchdog timer count. Together, these four read-only registers contain the current value of the watchdog timer count. WDCNT2:CounterSection2 is reset to 0x01.

Table 13-10. WDCNT2 Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								CounterSection2							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
ADDR	WDTBase + 0x010															

13.9.7 Watchdog Counter Section 3 Register

The WDCNT3 register contains the high byte of the present value of the watchdog timer.

This register is one of a set of four registers. Each register contains one byte of the present watchdog timer count. Together, these four read-only registers contain the current value of the watchdog timer count. WDCNT3:CounterSection3 is reset to 0x00.

Table 13-11. WDCNT3 Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								CounterSection3							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
ADDR	WDTBase + 0x014															

Chapter 14
Real-Time Clock (RTC)

The LH79520 includes a Real-Time Clock (RTC) peripheral which can provide either a basic alarm function or a long time-base counter. This Chapter explains the theory and operation of the RTC.

14.1 Theory of Operation

The RTC is a free-running counter which can be clocked at a 1 Hz rate. The 1 Hz clocking signal is generated by an internal oscillator which utilizes an external 32.768 kHz crystal. The RTC can be programmed to issue an interrupt when the RTC count matches a programmed value. An RTC interrupt will if necessary transition the LH79520 from a sleep mode to the active power mode. Figure 14-1, an expanded portion of the LH79520 block diagram, shows that the RTC is programmed via the APB, can issue an interrupt to the LH79520 VIC and is clocked by a signal from the Reset, Clock Generation and Power Control System (RCPC).

If the LH79520 RTC is not utilized, it can remain disabled (its reset condition), the external 32.768 kHz crystal need not be provided, but the XTAL32IN input must be tied LOW to prevent spurious noise from being generated if a 32.768 kHz crystal is not connected.

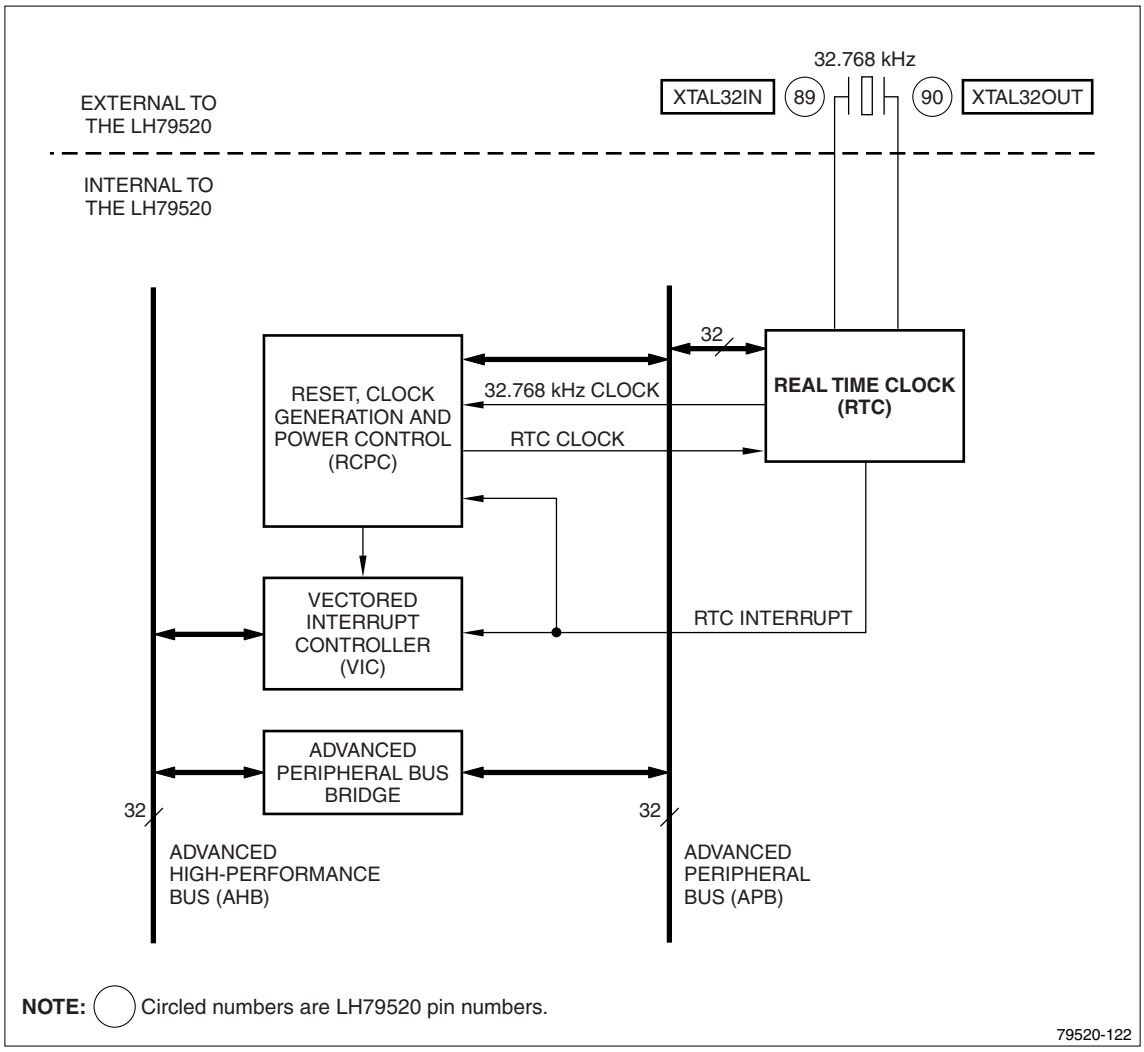


Figure 14-1. RTC Peripheral

14.1.1 RTC Clock Generation

Figure 14-2 illustrates that the clock signal routed to the Real-Time Clock (RTC) peripheral is generated by the output of an internal 32.768 kHz clock oscillator, or by that same signal divided internally by 32,768. The lower-frequency (divided) signal is intended for normal use.

The LH79520 RTC peripheral is disabled at reset. The input-signal selection is determined by PeriphClkSel:RTC (the RTC bit field in the PeriphClkSel register) and the output fed to the RTC peripheral can be enabled or disabled by PeriphClkCtrl:RTC (the RTC bit field in the PeriphClkCtrl register).

The PeriphClkSel and PeriphClkCtrl registers are explained in Chapter 7 – Reset, Clock Generation, and Power Control (RCPC). See Chapter 7, Section 7.11 for additional information about RTC clock signal generation.

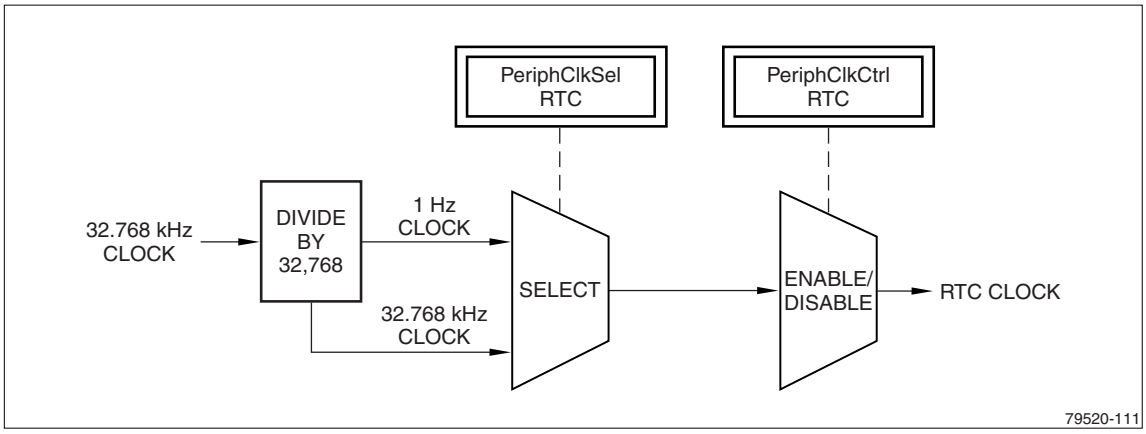


Figure 14-2. RTC Clock Generation

14.1.2 RTC Interrupts and LH79520 Power Modes

The LH79520 RTC is clocked by a 1 Hz clock signal which remains active in several of the LH79520 reduced-power modes. The 1 Hz clock drives the RTC free-running counter and the RTC can be programmed to generate an interrupt when the value in the free-running counter matches a value stored in the RTCMR register.

When an RTC Interrupt occurs, the LH79520 will transition to its Active power mode. See Section 7.13 in Chapter 7 for more information about the LH79520 power management and power modes.

14.2 RTC Programmer's Model

The Register Base Address for the LH79520 Real-Time Clock peripheral is:

RTCBase: 0xFFFE0000

All registers in the RTC must be accessed as a full word. The RTC does not support byte and half-word writes. Programmers should ensure that the system clock is enabled before programming any registers.

14.3 RTC Register Summary

The programmable RTC registers are listed in Table 14-1.

The free-running RTC counter is clocked by the 1 Hz RTC clock signal. The RTC clock signal is controlled by programmable registers in the LH79520 RCPC. The RTC will not function correctly until the RTC clock signal is enabled. See Section 14.1.1 for information about those registers.

Table 14-1. RTC Register Summary

ADDRESS	NAME	DESCRIPTION
RTCBase + 0x000	RTCDR	RTC Data Register
RTCBase + 0x004	RTCMR	RTC Match Register
RTCBase + 0x008	RTCSTAT/RTCEOI	This is a dual-purpose register: RTC Interrupt Status Register (Read) RTC Interrupt Clear Register (Write)
RTCBase + 0x00C	RTCLR	RTC Counter load Register
RTCBase + 0x010	RTCCR	RTC Interrupt Control Register
RTCBase + 0x014 - RTCBase + 0x0FF	///	Reserved. Do not write.

14.4 RTC Register Descriptions

This section of this chapter of this User's Guide lists and describes the registers that configure the LH79520 Real-Time Clock (RTC).

14.4.1 RTC Data Register

The RTCDR register contains the present value of the 32-bit free-running RTC counter.

Table 14-2. RTCDR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	C31	C30	C29	C28	C27	C26	C25	C24	C23	C22	C21	C20	C19	C19	C17	C16
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	C15	C14	C13	C12	C11	C10	C09	C08	C07	C06	C05	C04	C03	C02	C01	C00
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
ADDR	RTCBase + 0x00															

The value in the RTCDR can be read in atomic fashion by one 32-bit read operation. Table 14-3 describes the bit fields in the RTCDR.

Table 14-3. RTCDR Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:0	C31:C00	**RTC Count** Reads return the present value of the free-running RTC counter. This is a read-only register. Writes have no effect. 0x0000 = reset

14.4.2 RTC Match Register

The RTCMR register contains the count at which an RTC interrupt can be generated.

The RTC contains a free-running counter (the RTCDR register) which is clocked by a 1 Hz clock. The RTC can be programmed to issue to the LH79520 VIC an RTC interrupt when the value in the RTCDR register matches the value programmed to this RTCMR (match-count) register.

Writes to this (RTCMR) register establish the match value in this register. Reads of this (RTCMR) register return the last value written since reset.

Table 14-5 describes the bit fields in the RTCMR register.

Table 14-4. RTCMR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	M31	M30	M29	M28	M27	M26	M25	M24	M23	M22	M21	M20	M19	M19	M17	M16
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	M15	M14	M13	M12	M11	M10	M09	M08	M07	M06	M05	M04	M03	M02	M01	M00
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	RTCBase + 0x04															

Table 14-5. RTCMR Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:0	M31:M00	**RTC Match Count** Writes establish the value of the RTC Match Count. Reads return the present value of the RTC Match Count. 0x0000 = reset

14.4.3 RTC Interrupt Status and Interrupt Clear Registers

The RTCSTAT/RTCEOI register has two uses. Reads of this register return the present status of the RTC interrupt; writes to this register will clear an RTC interrupt.

A write of any value to this register will clear the RTCINTR interrupt signal routed to the LH79520 VIC and also clear the RTCINTR status bit in this register. Values written to this register will not be stored in this register but the write operation will have the effect of clearing the interrupt flag in this register.

Table 14-7 shows the bit field definitions for the RTCSTAT/RTCEOI register.

Table 14-6. RTCSTAT/RTCEOI Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0															
TYPE	RW*															
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///															RTCINTR
RESET	0															0
TYPE	RW*															RW
ADDR	RTCBase + 0x08															

NOTE: *This is a modified R/W register. Write operations have an effect but the value written is not retained. Reads of this register return the status of the RTCINTR interrupt.

Table 14-7. RTCSTAT/RTCEOI Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:1	///	**Reserved** Write '0', reads unpredictable.
0	RTCINTR	**Rtc Interrupt Status Flag** 0 = RTCINTR is not asserted (reset) 1 = RTCINTR is asserted A write of any data value to this register will clear the RTC interrupt, and will clear this bit field, but the value written will not be retained in this register. A read of this register will return the present status of this bit field.

14.4.4 RTC Load Register

The RTCLR register permits software to initialize the free-running RTC counter.

Writes to this 32-bit register will load the free-running RTC counter with an initial value within 1 RTC clock cycle. The free-running RTC counter is updated with the value written to this register on the next rising edge of the 1 Hz RTC clock signal.

This register is not the free-running RTC counter; this register is a buffer from which the RTC counter is loaded. Reads of this register do not return the present value of the free-running RTC counter. Instead, reads of this register return the last value written to this register.

Software can obtain the present value of the free-running RTC counter by reading the RTCDR register.

Table 14-9 defines the bit fields in the RTCLR register.

Table 14-8. RTCLR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	IC31	IC30	IC29	IC28	IC27	IC26	IC25	IC24	IC23	IC22	IC21	IC20	IC19	IC19	IC17	IC16
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	IC15	IC14	IC13	IC12	IC11	IC10	IC09	IC08	IC07	IC06	IC05	IC04	IC03	IC02	IC01	IC00
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	RTCBase + 0x0C															

Table 14-9. RTCLR Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:0	IC31:IC00	**Initial Count** Writes establish the initial value of the 32-bit RTC count. Reads return the present value of the 32-bit RTC count. 0x0000 = reset

14.4.5 RTC Interrupt Masking Register

The RTCCR register can be used to enable or disable the RTC interrupt.

The RTC can be programmed to issue an interrupt when the value in the RTCDR register (a free-running count) matches the value programmed in the RTCMR register.

Writing RTCCR:MIE = 1 enables the RTC interrupt; writing '0' disables the interrupt.

This is a write-only register; reads will return unpredictable results.

Table 14-11 shows the bit field definitions for the RTCCR register.

Two LH79520 systems must be correctly programmed in order for the RTC Interrupt to function correctly:

- The RTC Interrupt must be enabled by a write to this register
- The LH79520 VIC must be programmed to utilize the RTC Interrupt.

See Chapter 9 – Exceptions and Interrupts for information concerning the LH79520 VIC.

Table 14-10. RTCCR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET																
TYPE	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///															MIE
RESET																
TYPE	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
ADDR	RTCBase + 0x10															

Table 14-11. RTCCR Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:1	///	**Reserved** Write '0', reads unpredictable. This is a write-only field.
0	MIE	**Match Interrupt Enable** 0 = Match Interrupt is disabled (reset) 1 = Match Interrupt is enabled This is a write-only field.

Chapter 15
Pulse Width Modulators (PWM)

The LH79520 includes dual Pulse Width Modulator (PWM) peripherals. This Chapter explains the theory and operation of the PWM peripherals.

15.1 Theory of Operation

Two PWM peripherals are integrated into the LH79520 as shown in Figure 15-1. Both of the PWM peripherals are programmed via the APB, receive scaled clock inputs from the LH79520 RCPC and produce outputs at external pins. PWM channel 0 can receive an optional synchronizing input signal from an LH79520 pin.

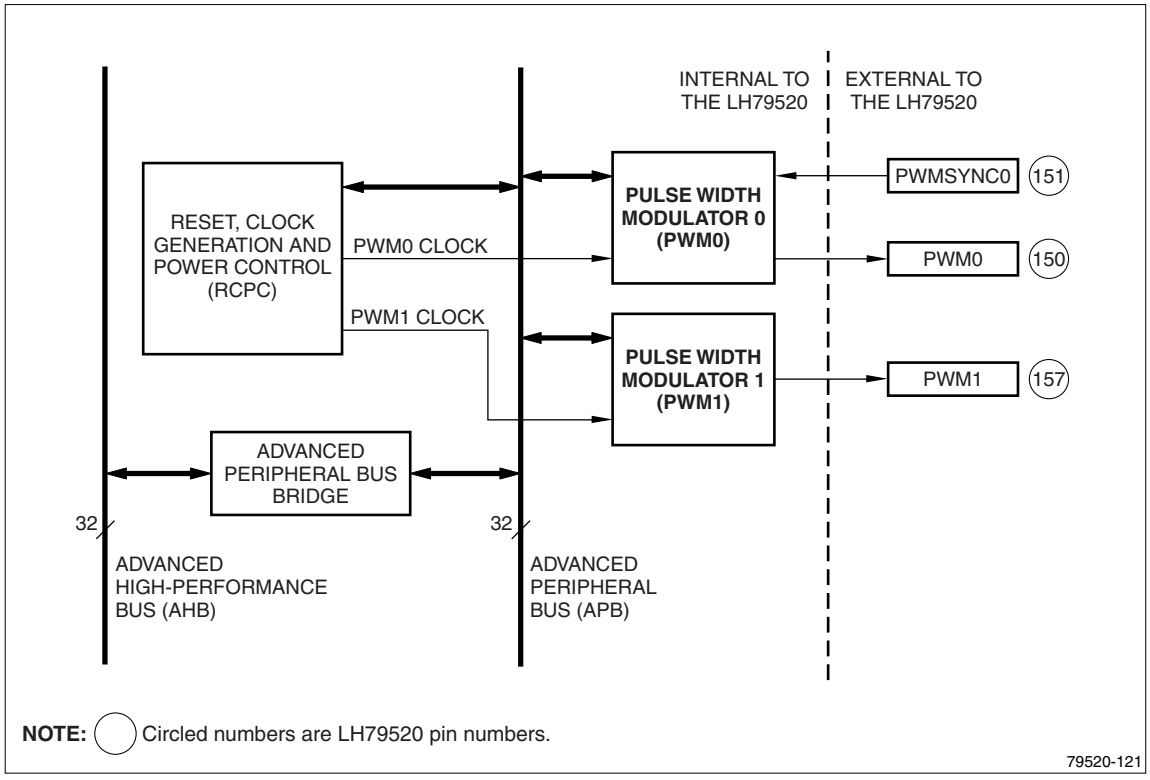


Figure 15-1. LH79520 PWM Peripherals

Either PWM channel can be utilized to create reoccurring pulses at LH79520 PWMx output pins. Depending upon how a PWM is programmed, its output can vary from a continuous level (100% duty cycle), to a square wave (50% duty cycle), to a narrow pulse approaching a 0% duty cycle. Both PWMs offer 16-bit resolution of the input clock signal.

Figure 15-2, a simplified block diagram of the LH79520 PWM peripherals, shows that the PWM peripherals are independently configurable and both PWMs are reset when the LH79520 is reset. See Chapter 7 – Reset, Clock Generation, and Power Control (RCPC) for more information about LH79520 resets.

The outputs of both LH79520 PWM channels are programmed in terms of PWM input clock cycles. Each PWM may be programmed statically (when it is halted) or dynamically (while it is running). The output of either PWM may be programmed as normal or inverted. With the exception of inversion, if a PWM is programmed statically, no change to the output will occur until the PWM is enabled. If a PWM is reprogrammed while it is running (enabled), the output is updated to the new programming (total period, total period asserted) at the beginning of the next PWM cycle. The exception for inverted operation is explained below. Both PWMs are reset to the halted condition.

The output of either LH79520 PWM can be programmed for either normal or inverted operation. Inversion affects the output pin when the PWM peripheral is halted and also when it is running. Both outputs are reset to the normal (non-inverted) configuration, which places the output pins in a LOW condition at reset. When the output is reprogrammed to be inverted (or to be normal), the new programming does not become effective until the rising edge of the PWM input clock signal.

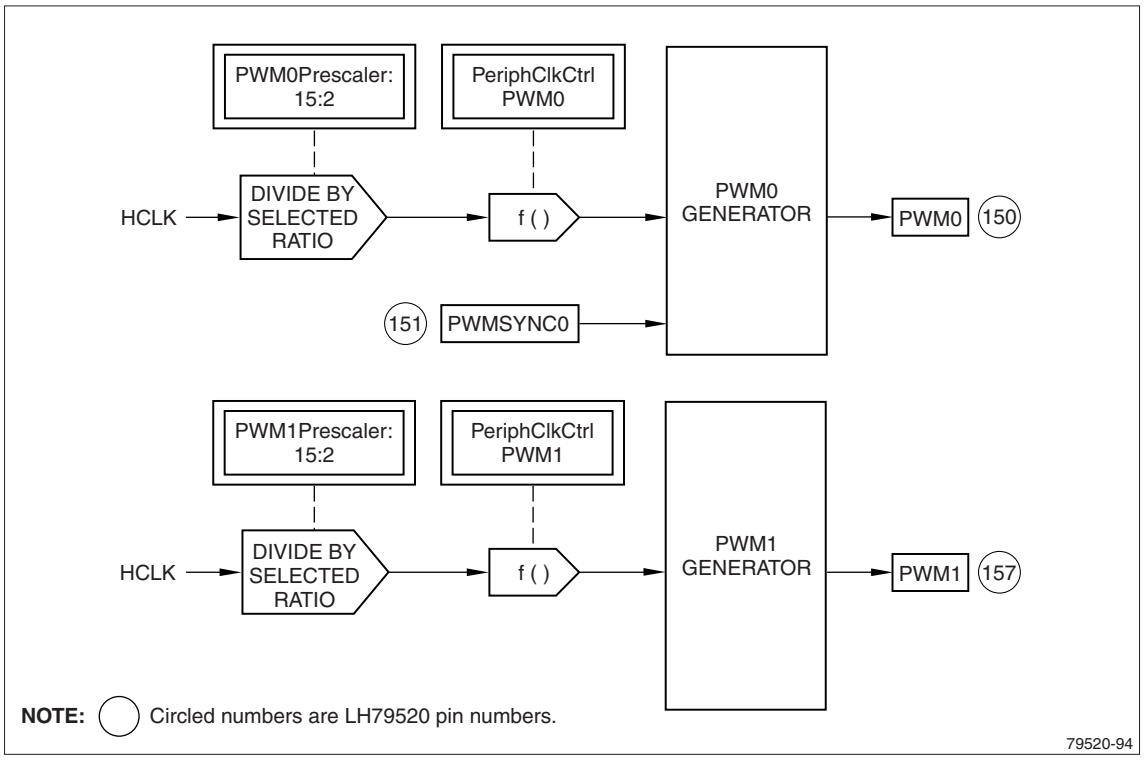


Figure 15-2. LH79520 PWM Block Diagram

15.1.1 PWM Clocks

Each PWM channel receives a clock signal from the LH79520 RCPC. Both clock signals are derived from PCLK. (PCLK is locked to HCLK in the LH79520.) The clock signals can be independently enabled, disabled and pre-scaled, as shown in Figure 15-3. The clock signals are disabled at reset and must be enabled if one or both PWM channels are to be used.

For more information about these clock signals, see the explanations of the PeriphClkCtrl, PWM0Prescale and PWM1Prescale registers in Chapter 7 – Reset, Clock Generation, and Power Control (RCPC).

When the PWM channels are operating, disabling either of the PWM clock signals shown in Figure 15-3 will freeze the output of the associated PWM channel. The output is frozen in its present condition (HIGH or LOW). If the clock signal is subsequently restored, the PWM channel will restart at the point at which it was stopped.

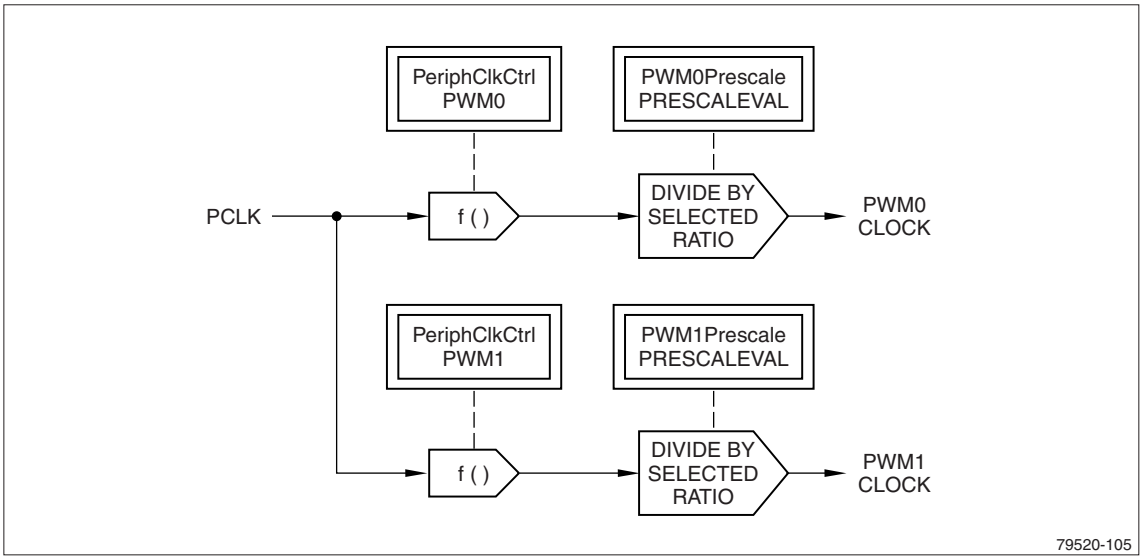


Figure 15-3. PWM Clock Generation

15.1.2 PWM Output Period and Duty Cycle

A signal's duty cycle is a measure of the percentage of time that it is asserted, relative to its total period. Duty cycle can be defined in two different (but equivalent) ways:

- Duty Cycle = Time ASSERTED/(Time ASSERTED + Time DEASSERTED)
- Duty Cycle = Time ASSERTED/Total Time

Although the first definition is convenient for manual calculations, the registers that control the LH79520 PWM channels are programmed according to the second definition.

For either PWM channel, the Total Time and the Time Asserted are determined by the values programmed to the PWMx_TC and PWMx_DC registers for that channel, respectively. When the PWM channel is initially enabled, and each time the PWM cycle repeats, the values programmed to these double-buffered registers are utilized as the initial values for down-counters. When a PWM channel is running, the down-counters for that channel are decremented by the PWM clock input for that channel. The PWM clock input can be pre-scaled in the RCPC to permit a wide range of output periods and pulse-widths. Table 15-1 lists the valid ranges for the programmed PWM values.

Table 15-1. PWMx Register Valid Values

REGISTER	BIT FIELD	DESCRIPTION	VALID RANGE	UNITS
PWMx_TC	TERMINALCOUNT	Total Count	$1 \leq \text{TERMINALCOUNT} \leq (2^{16} - 1)$	PWMx input clocks
PWMx_DC	DCCOUNT	Asserted Count	$1 \leq \text{DCCOUNT} \leq \text{TERMINALCOUNT}$	PWMx input clocks

The duration of one PWM cycle is the period of one PWM input clock cycle multiplied by the PWM channel's TERMINALCOUNT, plus 1.

PWMx Cycle Duration = PWM input clock period × (PWMx_TC:TERMINALCOUNT + 1)

The output of the PWM channel will be asserted for:

PWMx Asserted Duration = PWM input clock period × (PWMx_DC:DCCOUNT + 1)

A channel's asserted condition (HIGH or LOW) is determined by PWMx_INV:INV (the INV bit field in the channel's PWMx_INV register). The INV bit field is reset to '0', which will produce an output that is LOW when the PWM is halted.

If the PWM channel is operating in the normal output mode, the PWM output signal will be HIGH for PWMx_DC:DCCOUNT+1 clock cycles and LOW for the remainder of the PWM cycle. If the PWM channel is operating in the inverted output mode, the PWM output signal will be LOW for PWMx_DC:DCCOUNT+1 clock cycles and HIGH for the remainder of the PWM cycle.

PWMx_TC:TERMINALCOUNT and PWMx_DC:DCCOUNT values that do not meet the limits shown in Table 15-1 are illegal and the output is not guaranteed. Programming PWMx_DC:DCCOUNT to be greater than PWMx_TC:TERMINALCOUNT+1 will result in a 100% Duty Cycle.

15.1.3 PWM Output Signals

The output of either PWM channel can be programmed to be either normal or inverted. This programming affects the PWM outputs when the PWM channel is active and also when the PWM channel is halted.

15.1.3.1 Non-Inverted PWM Output Signals

Figure 15-4 illustrates the timing conventions for the LH79520 PWM output signals. The figure shows a PWM channel programmed for normal (non-inverted) operation and driven by a PWMCLK (PWM clock) signal.

The PWM channel shown in Figure 15-4 is programmed to produce a pulse which is HIGH for two complete cycles of the PWM input clock (due to the value in the PWMx_TC register, plus 1) and LOW for one cycle of the input clock (due to the value in the PWMx_DC register, plus 1). The resulting pulse is HIGH for 2/3 of the total PWM output cycle and LOW for 1/3 of the total cycle. The duty cycle of the pulse is 2/3 or 66%. The figure shows three such pulses.

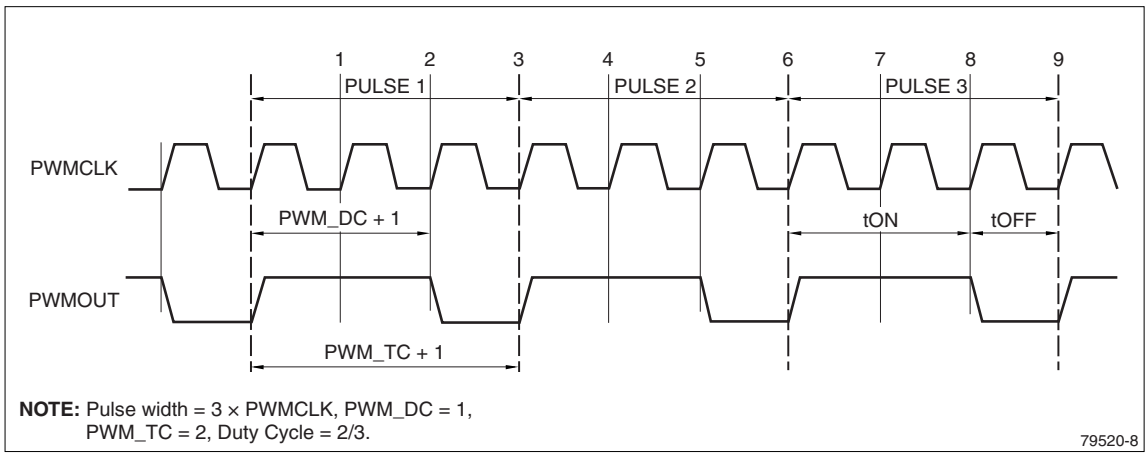


Figure 15-4. PWM Output Signal Diagram

15.1.3.2 Inverted PWM Output Signals

The output of either PWM channel can be inverted by programming PWMx_INV:INV = 1. Figure 15-5 shows examples of normal (non-inverted) and inverted PWM outputs.

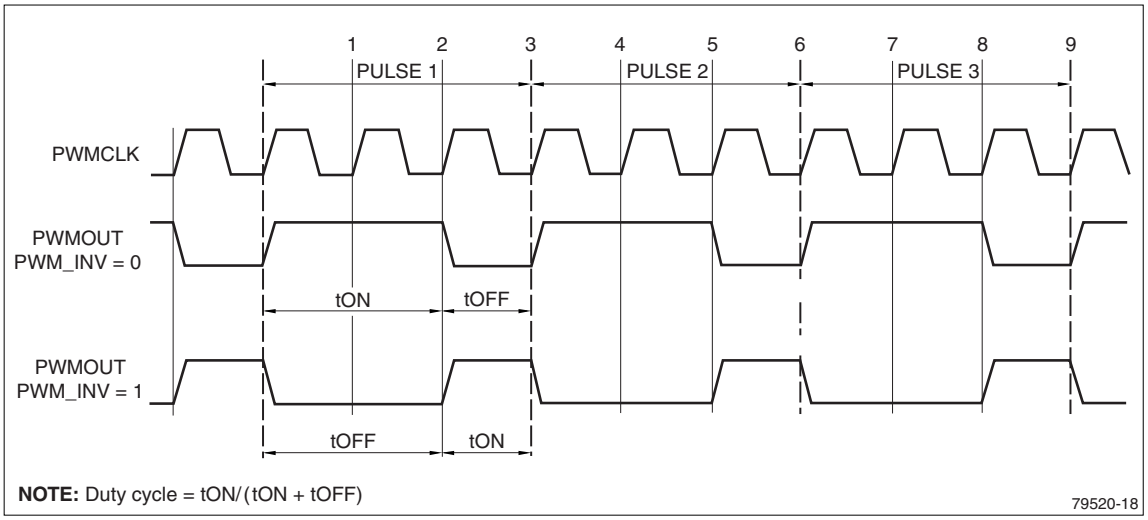


Figure 15-5. PWM Inverted Output Signal Diagram

The PWMx_INV register affects the PWM output when the PWM channel is halted and also when the channel is active. The outputs of both PWM channels are reset to the normal condition and both PWM channels are reset to a halted condition.

The normal output of a halted PWM channel is a LOW level. The inverted output of a halted PWM channel is a HIGH level.

The normal output of an active PWM channel will be HIGH for PWMx_DC:DCCOUNT+1 and LOW for the remainder of the period specified by PWMx_TC:TERMINALCOUNT+1.

The inverted output of an active PWM channel will be LOW for PWMx_DC:DCCOUNT+1 and HIGH for the remainder of the period specified by PWMx_TC:TERMINALCOUNT+1.

If PWMx_INV:INV is written when the PWM channel is halted, the new PWMx_INV:INV value will become effective on the rising edge of the next PWM input clock. If the PWM clock for that channel is disabled (as explained in Section 15.1.1) then the new value will not take effect until the clock is enabled and the first rising edge of the PWM input clock occurs.

If PWMx_INV:INV is written when the PWM channel is running, the new PWMx_INV:INV value will not become effective until the beginning of the next PWM cycle, to avoid glitching the output.

A new PWM cycle begins when the down-counters expire and are automatically reloaded from the PWMx_TC and PWMx_DC registers. When a PWM channel is first enabled, its output does not commence until the next rising edge of the PWMx input clock signal. If the PWM channel is programmed for synchronous operation (PWM channel 0 only), the new values will not be utilized until an external signal transition from LOW to HIGH occurs.

Although not recommended, software can, for example, produce an approximately square wave (50% duty cycle) on a disabled PWMx output pin merely by looping and toggling PWMx_INV:INV for that channel. This form of operation is not possible on an enabled PWM channel because the channel output is governed by the PWM logic and down-counters.

15.1.4 PWM Modes of Operation

The output of either PWM channel normally commences on the rising edge of the first PWM clock cycle after the channel is enabled. In addition to this normal mode of operation, the output of PWM channel 0 can be optionally synchronized (i.e.: triggered by) an external signal. The external signal need not remain asserted once it has triggered PWM0 to commence generating output. Synchronization has no other effect on the operation of PWM channel 0. The synchronization feature adds an extra register to the PWM0 register set.

15.1.4.1 Normal Mode

The normal mode of operation for either PWM channel is shown in Figure 15-6. In this mode the output of the PWM channel will commence on the first rising edge of the PWMx clock following the write operation which enables that PWM channel (PWMx_EN:EN = 1).

The output of an active PWM channel can be disabled by writing PWMx_EN:EN = 0. When disabled, the PWM output does not halt immediately. When the PWM output is disabled, the output will halt when the present PWM cycle is completed.

Although both PWM channels can be operated in the normal mode, software must ensure that PWM0_SYNC:MODE = 0 (the reset condition) to operate PWM0 in the normal mode.

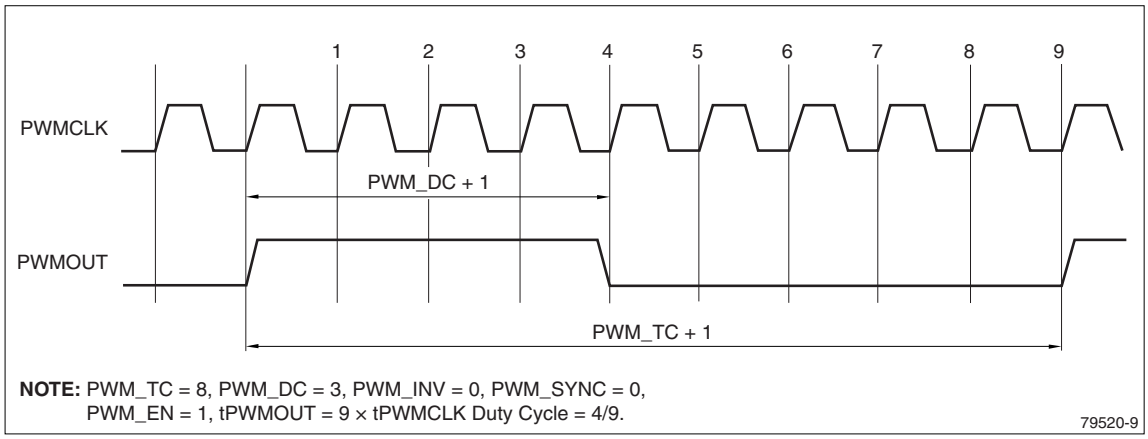


Figure 15-6. PWMx Normal Mode Operation

15.1.4.2 Synchronous Mode

PWM0 can be operated in synchronous mode by writing PWM0_SYNC:MODE = 1. When the PWM0 channel is placed in the synchronous mode and subsequently enabled (by writing PWM0_EN:EN = 1), the PWM0 output will not commence until the signal at the PWM0_SYNC input pin transitions from a LOW to a HIGH. When the transition occurs, the PWM0 output will commence at the next rising edge of the PWM0 clock.

As in the normal mode, writing PWM0_EN:EN = 0 will disable the PWM0 channel output. When disabled, the PWM0 output does not halt immediately. When the PWM0 output is disabled, the output will halt when the present PWM0 output signal cycle is completed.

The difference between the normal and synchronous modes of operation is the way in which the PWM0 output commences. If PWM0 is operating in the synchronous mode, then when the PWM0 output has been triggered (started) by the synchronizing signal, the synchronizing signal has no more effect and need not remain asserted. Figure 15-7 shows a PWM output in synchronous mode, triggered by an external sync signal.

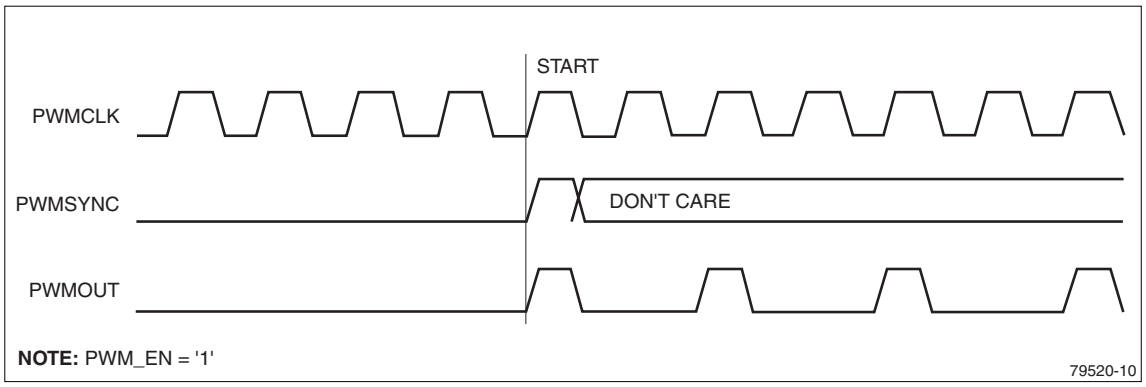


Figure 15-7. PWM1, Triggered By The PWM1_SYNC Input

15.1.5 PWM Clocks and Power Management

Each LH79520 PWM channel receives a programmable clock (PWMx clock) from the LH79520 RCPC, as shown in Figure 15-1. Both of these clocks are derived from the APB clock generated in the LH79520 RCPC functional block and explained in Chapter 7 – Reset, Clock Generation, and Power Control (RCPC).

Disabling either PWM clock at its source (in the RCPC block) will freeze the output of the associated PWM channel in its present condition (HIGH or LOW). Re-enabling the PWM clock at its source (in the RCPC) will cause the associated PWM channel to resume operation at the point at which it had stopped. See Chapter 7, Section 7.10 for more information about PWM clock generation in the RCPC.

The LH79520 can be placed in several different power modes, to conserve power. These power modes affect the generation of the two clock signals (PWMx clocks) fed to the PWM system. See Chapter 7, Section 7.13 for additional information concerning power modes and clock generation.

15.1.6 Programming Recommendations

15.1.6.1 Programming the Duty Cycle

The software cannot directly ascertain the condition of the PWMx output pin. To ensure proper operation, especially when reprogramming an active PWM channel, preserve the relationship between the values in the PWMx_TC and PWMx_DC registers by observing the programming rules listed here. Failure to observe these rules may result in a duty cycle of 100%. If a 100% duty cycle occurs and the programming is correct, the incorrect duty cycle will persist until a new PWM cycle begins. If the programming is incorrect, the incorrect duty cycle will persist until both PWMx_TC and PWMx_DC are updated with values that obey these rules.

1. Maintain PWMx_TC:TERMINALCOUNT $\leq$ PWMx_DC:DCCOUNT
2. If PWMx_TC:TERMINALCOUNT (new) $\geq$ PWMx_TC:TERMINALCOUNT (old) Write PWMx_TC first, then write PWMx_DC
3. If PWMx_TC:TERMINALCOUNT (new) < PWMx_TC:TERMINALCOUNT (old) Write PWMx_DC first, then write PWMx_TC
4. Failure to follow these rules may result in a duty cycle of 100%. This incorrect duty cycle will persist until both PWMx_TC and PWMx_DC are updated.

15.1.7 PWM Programming Example

Using a PWM input clock of 66 MHz (15 ns) to produce on an active PWM channel a non-inverted output with a 20% duty cycle which repeats at 100 kHz (10 μs):

15.1.7.1 Decimal Calculations

1. Calculate PWMx_TC:TERMINALCOUNT = ((66.0 MHz/0.1 MHz) – 1) = 659
2. Calculate PWMx_DC:DCCOUNT = ((0.2 × (659+1)) – 1) = 131

15.1.7.2 Static Programming

1. Write PWMx_EN:EN = 0 to disable the PWM channel
2. Wait for the PWM channel to complete the present PWM cycle
3. Write PWMx_TC:TERMINALCOUNT = 659
4. Write PWMx_DC:DCCOUNT = 131
5. Write PWMx_EN:EN = 1 to enable the PWM channel

15.1.7.3 Dynamic Programming

1. Write PWMx_TC:TERMINALCOUNT = 659
2. Write PWMx_DC:DCCOUNT = 131

15.2 PWM Programmer's Model

The Register Base Address for the LH79520 Pulse Width Modulators is:

PWMBase: 0xFFFC3000

All registers in the PWM must be accessed as a full word. The PWM does not support byte and half-word writes. Programmers should ensure that the system clock is enabled before programming any registers.

15.3 PWM Register Summary

Table 15-2 summarizes the programmable registers for both PWM channels. Tables 15-3 through 15-13 explain the individual registers and their bit fields in detail.

Table 15-2. PWMx Register Summary

ADDRESS	REGISTER NAME	DESCRIPTION
PWMBase + 0x000	PWM0_TC	PWM0 Terminal Count register
PWMBase + 0x04	PWM0_DC	PWM0 Duty Cycle register
PWMBase + 0x08	PWM0_EN	PWM0 Enable register
PWMBase + 0x0C	PWM0_INV	PWM0 Invert register
PWMBase + 0x10	PWM0_SYNC	PWM0 Synchronization register
PWMBase + 0x014 - PWMBase +0x1F	///	Reserved
PWMBase + 0x20	PWM1_TC	PWM1 Terminal Count register
PWMBase + 0x24	PWM1_DC	PWM1 Duty Cycle register
PWMBase+ 0x28	PWM1_EN	PWM1 Enable register
PWMBase + 0x2C	PWM1_INV	PWM1 Invert register

NOTE: At reset, both PWM channels are programmed for normal (non-inverted) operation, both PWM outputs are LOW, and both PWM channels are halted.

15.4 Register Descriptions

This section of this chapter of this User's Guide lists and describes the registers that configure both of the LH79520 PWM channels.

15.4.1 PWM0 Terminal Count Register

The 16-bit value programmed to the PWM0_TC register establishes the total period of the PWM0 output, in PWM0 input clock cycles.

The total period of the PWM0 output determines the duration of the PWM0 cycle, and therefore how often the PWM0 cycle will repeat. The PWM0_TC register provides 16-bit resolution for the PWM0 output and may be programmed when PWM channel 0 is halted (static programming) or when the channel is running (dynamic programming).

Table 15-4 describes the bit fields in the PWM0_TC register.

The PWM0_TC register is double-buffered; the value in PWM0_TC does not change as PWM0 runs. At the beginning of a PWM cycle, the PWM logic copies the contents of the PWM0_TC register to a down-counter. The PWM0 input clock signal decrements the counter to produce the PWM0 output signal. When PWM0_TC is programmed statically, the timing of the write access is not important. When PWM0_TC is programmed dynamically, the counter will be updated at the beginning of the next PWM0 cycle, to avoid glitching the PWM0 output.

The down-counter is not directly available for read or write operations. Reads of the PWM0_TC register do not return the present value of the counter. Instead, reads of the register return the value last written to the PWM0_TC register.

Table 15-3. PWM0_TC Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	TERMINALCOUNT															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	PWMBase + 0x000															

Table 15-4. PWM0_TC Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:16	///	**Reserved** Write the Reset value.
15:0	TERMINALCOUNT	**PWM0 Total Period** $1 \leq$ TERMINALCOUNT $\leq (2^{16} - 1)$ This field is reset to 0.

15.4.2 PWM0 Duty Cycle Register

The PWM0_DC register establishes the period of time (in PWM clock cycles) during which the PWM0 output is asserted (i.e.: HIGH, for non-inverted operation).

The PWM0_DC register works in conjunction with the PWM0_TC register to establish the duty cycle of the PWM0 output.

The PWM0_DC register provides 16-bit resolution for the PWM0 output and may be programmed when PWM channel 0 is halted (static programming) or when the channel is running (dynamic programming).

Table 15-6 describes the bit fields in the PWM0_DC register.

The PWM0_DC register is double-buffered; the value in PWM0_DC does not change as PWM0 runs. At the beginning of a PWM cycle, the PWM logic copies the contents of the PWM0_DC register to a down-counter. The PWM0 input clock signal decrements the counter to produce the PWM0 output signal. When PWM0_DC is programmed statically, the timing of the write access is not important. When PWM0_DC is programmed dynamically, the counter will be updated at the beginning of the next PWM0 cycle, to avoid glitching the PWM0 output.

The actual counter is not directly available for read or write operations. Reads of the PWM0_DC register do not return the present value of the counter. Instead, reads of the register return the value last written to the PWM0_DC register.

Table 15-5. PWM0_DC Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	DCCOUNT															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	PWMBase + 0x004															

Table 15-6. PWM0_DC Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:16	///	**Reserved** Write the Reset value.
15:0	DCCOUNT	**PWM0 Duty Cycle** 1 ≤ DCCOUNT ≤ PWM0_TC:TERMINALCOUNT This field is reset to 0.

15.4.3 PWM0 Enable Register

The PWM0_EN register enables (starts) and disables (halts) the output of PWM channel 0.

Table 15-8 describes the bit fields in the PWM0_EN register.

The PWM0_EN register is a read-write register.

Writing PWM0_EN:EN = 1 enables (starts) PWM channel 0. When PWM channel 0 is enabled, the contents of the PWM0_TC and PWM0_DC registers are copied to the PWM0 down-counters and the PWM0 output commences on the next rising edge of the PWM0 input clock signal. The PWM0 output level (LOW or HIGH) during the PWM0 cycle is determined by the PWM0_INV:INV.

Writing PWM0_EN:EN = 0 disables (halts) the PWM. When PWM channel 0 is disabled, the PWM0 output will halt when it reaches the end of its current PWM0 cycle (as defined by the values programmed to the PWM0_TC, PWM0_DC and PWM0_INV registers).

PWM0 is reset to a halted condition and a LOW level output.

Table 15-7. PWM0_EN Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///															EN
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	PWMBase + 0x008															

Table 15-8. PWM0_EN Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:1	///	**Reserved** Write '0'. Read '0'.
0	EN	**PWM0 Enable** 0 = PWM0 is disabled (halted) (reset) 1 = PWM0 is enabled

15.4.4 PWM0 Output Invert

The PWM0_INV register can be programmed to invert the PWM0 output.

The PWM0_INV register is double-buffered to allow it to be programmed statically (when the PWM0 channel is halted) or dynamically (when the PWM channel is running). When programmed statically, the PWM0 output is updated on the rising edge of the next PWM0 input clock. When programmed dynamically, the PWM0 output is updated at the end of a PWM cycle, to prevent output glitches. Table 15-10 describes the bit fields in the PWM0_INV register.

When PWM0_INV:INV = 0, the output is not inverted. The PWM0 output will first be HIGH for a duration of PWM0 clock cycles defined by the value programmed to PWM0_DC, then LOW for a duration of PWM0 clock cycles defined by the difference between the values programmed to PWM0_TC and PWM0_DC. When PWM0_INV:INV = 1, the output will be inverted; the PWM0 output will first be LOW, then HIGH:

When the PWM0 output is halted, the PWM0 output pin will remain at the level determined by PWM0_INV:INV. Table 15-11 shows how PWM0_INV:INV affects the PWM0 output.

Table 15-9. PWM0_INV Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///															INV
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	PWMBase + 0x00C															

Table 15-10. PWM0_INV Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:1	///	**Reserved** Write the Reset value.
1	INV	**Invert** 0 = Do not invert the output. (reset) 1 = Invert the output. Halted: Update occurs on rising edge of next PWM0 input clock. Running: Update occurs at beginning of next PWM cycle.

Table 15-11. Halted PWM0 Output Level

PWM0_EN:EN	PWM0_INV:INV	PWM0 OUTPUT PIN
0	1	HIGH
0	0	LOW (reset)

15.4.5 PWM0 Synchronization

The PWM0_SYNC register permits the output of PWM0 to be triggered (synchronized with) an external signal at the PWM0_SYNC input pin.

The PWM0 channel can be operated in two different modes: normal and synchronized.

In the normal mode of operation, the PWM0 output commences when the channel is correctly programmed and then enabled. The normal mode of operation is the reset condition.

The synchronized mode of operation is identical to the normal mode of operation except that the PWM0 output does not commence until a LOW to HIGH signal transition occurs on the PWM0_SYNC input pin. Once triggered, the PWM0 output will continue even if the signal at the PWM0_SYNC pin returns to a LOW level.

Table 15-13 describes the bit fields in the PWM0_SYNC register.

The output of the PWM0 channel operating in either mode can be halted by writing PWM0_EN:EN = 0. The PWM0 output does not immediately halt when PWM0_EN:EN is written to '0'; instead, the output halts when the present PWM0 cycle has been completed.

Table 15-12. PWM0_SYNC Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///															MODE
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	PWMBase + 0x010															

Table 15-13. PWM0_SYNC Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:1	///	**Reserved** Write the Reset value.
0	MODE	**Mode of Operation** 0 = Normal mode. (reset) PWM0 activity begins when the channel is enabled. 1 = Synchronous mode. PWM0 activity begins when the channel is enabled and the external signal at the PWM0_SYNC pin transitions from LOW to HIGH.

15.4.6 PWM1 Terminal Count Register

The 16-bit value programmed to the PWM1_TC register establishes the total period of the PWM1 output, in PWM1 clock cycles.

The PWM1_TC register is identical in function to the PWM0_TC register, except that it controls the operation of PWM channel 1 (instead of PWM channel 0) and is located at a different address offset than the PWM0_TC register. For information about the bit fields in the PWM1_TC register, see Section 15.4.1.

Table 15-14. PWM1_TC Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	TERMINALCOUNT															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	PWMBase + 0x020															

15.4.7 PWM1 Duty Cycle Register

The PWM1_DC register establishes the period of time (in PWM clock cycles) during which the PWM1 output is HIGH.

The PWM1_DC register is identical in function to the PWM0_DC register, except that it controls the operation of PWM channel 1 (instead of PWM channel 0) and is located at a different address offset than the PWM0_DC register. For information about the bit fields in the PWM1_DC register, see Section 15.4.2.

Table 15-15. PWM1_DC Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	DCCOUNT															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	PWMBase + 0x024															

15.4.8 PWM1 Enable Register

The PWM1_EN register enables (starts) and disables (halts) the output of PWM channel 1. The output of PWM channel 1 is reset to the disabled (halted) condition.

The PWM1_EN register is identical in function to the PWM0_EN register, except that it controls the operation of PWM channel 1 (instead of PWM channel 0) and is located at a different address offset than the PWM0_EN register. For information about the bit fields in the PWM1_EN register, see Section 15.4.3.

Table 15-16. PWM1_EN Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///															EN
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	PWMBase + 0x028															

15.4.9 PWM1 Output Invert Register

The PWM1_INV register can be programmed to invert the PWM1 output.

The PWM1_INV register is identical in function to the PWM0_INV register, except that it controls the operation of PWM channel 1 (instead of PWM channel 0) and is located at a different address offset than the PWM0_INV register. For information about the bit fields in the PWM1_INV register, see Section 15.4.4.

Table 15-17. PWM1_INV Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FIELD	///															INV
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	PWMBase + 0x02C															

Chapter 16
Synchronous Serial Port (SSP)

The LH79520 MCU includes a Synchronous Serial Port (SSP) peripheral to facilitate synchronous serial communications with slave peripheral devices external to the LH79520. This Chapter explains the theory and operation of the LH79520 SSP peripheral.

16.1 Theory of Operation

The LH79520 SSP peripheral is a Master-only device, with programmable clock bit-rate and prescale factors. The SSP supports devices utilizing Motorola SPI, National Semiconductor Microwire or Texas Instruments' Synchronous Serial interfaces. Figure 16-1 illustrates how the SSP is integrated into the LH79520 MCU.

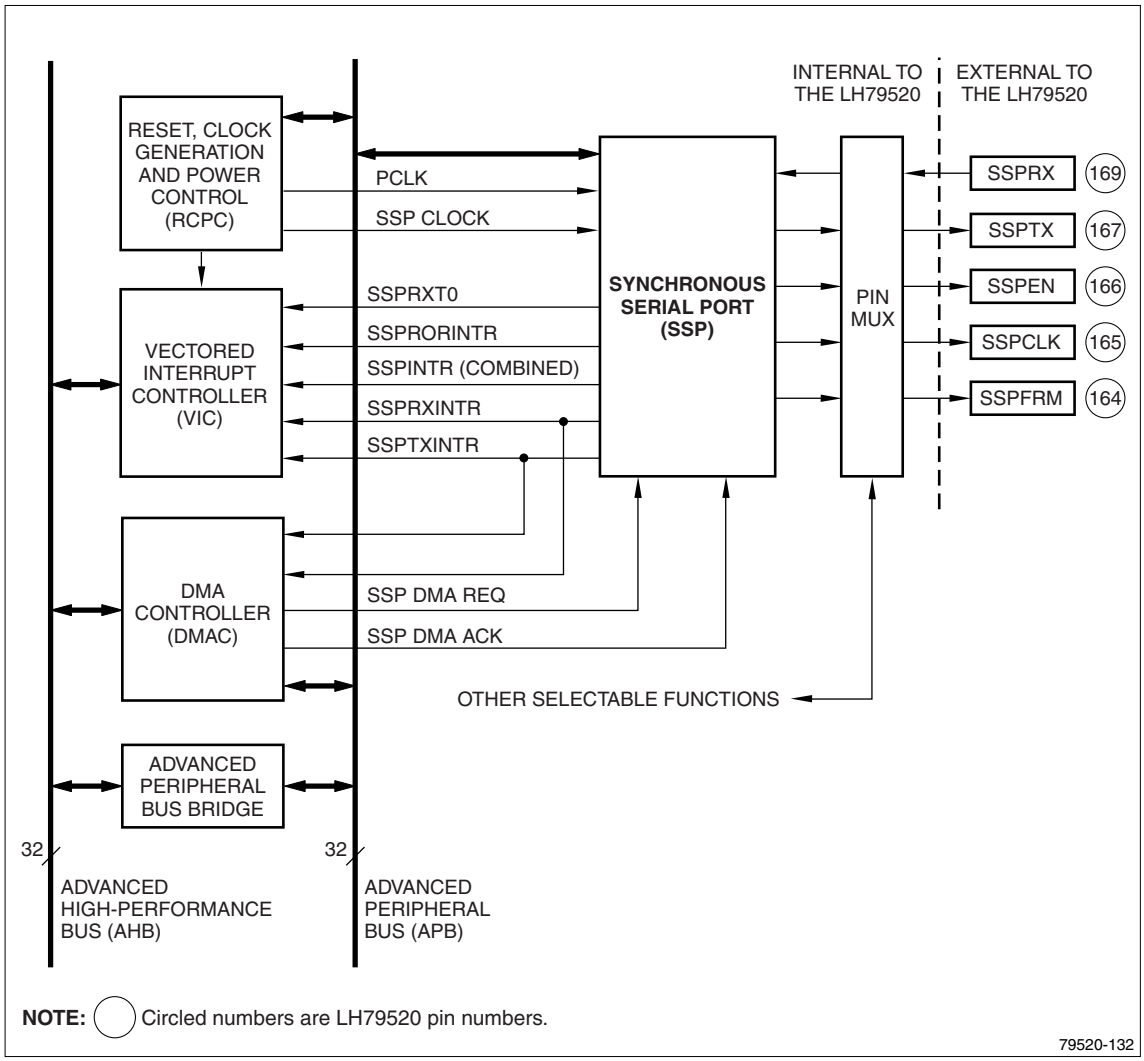


Figure 16-1. SSP Peripheral

The SSP is programmed via the APB, receives PCLK (locked to HCLK in the LH79520) and a scaled SSP Clock from the RCPC, and can issue five different interrupt signals to the LH79520 VIC. Two of the interrupt signals, SSPRXINTR and SSPTXINTR, also connect to the LH79520 DMAC.

Figure 16-1 identifies the SSP interface pins for connection to external peripheral devices. SSP interface signals are multiplexed to LH79520 pins with signals for Port A and UART2. Multiplexed SSP interface signals must be selected by software, via the SSPMux register, before use. Table 1-2, in Chapter 1 of this User's Guide, lists the multiplexed signals for each pin, and Chapter 8, Section 8.4.6 presents detailed information about the SSPMux register.

The SSP performs parallel-to-serial conversion on data written to an internal Transmit FIFO (First-In, First-Out buffer), then transmits the data, in serial fashion, to an external slave peripheral. The SSP also receives serial data from an external slave peripheral, performs a serial-to-parallel conversion on the received data, and buffers the received data to an internal Receive FIFO. Both FIFOs are 16-bits wide × 8-storage-locations deep. Data frame sizes may be programmed to be from 4 to 16 bits in length.

The SSP can assert five different interrupts. All five interrupts are individually maskable in the VIC. Separate RX, TX and Overrun interrupts are maskable in the SSP. The combined interrupt is automatically masked (disabled) at the SSP if all of the separate interrupts are masked (disabled) at the SSP. Of the five interrupts, only the RX Timeout interrupt is not maskable in the SSP. Figure 16-2 is a simplified block diagram of the SSP peripheral.

The SSP programmable bit-rate clock divider and prescaler utilize the SSP Clock signal to generate the serial output clock signal SSPCLK. The SSP supports bit-rates beyond 2 MHz, subject to the programmer's choice of SSPCLK frequency. The maximum bit rate will usually be determined by the external peripheral slave devices.

In addition to a scalable clock, the SSP includes programmable control registers to select the SSP operating mode, data frame format, and data frame size.

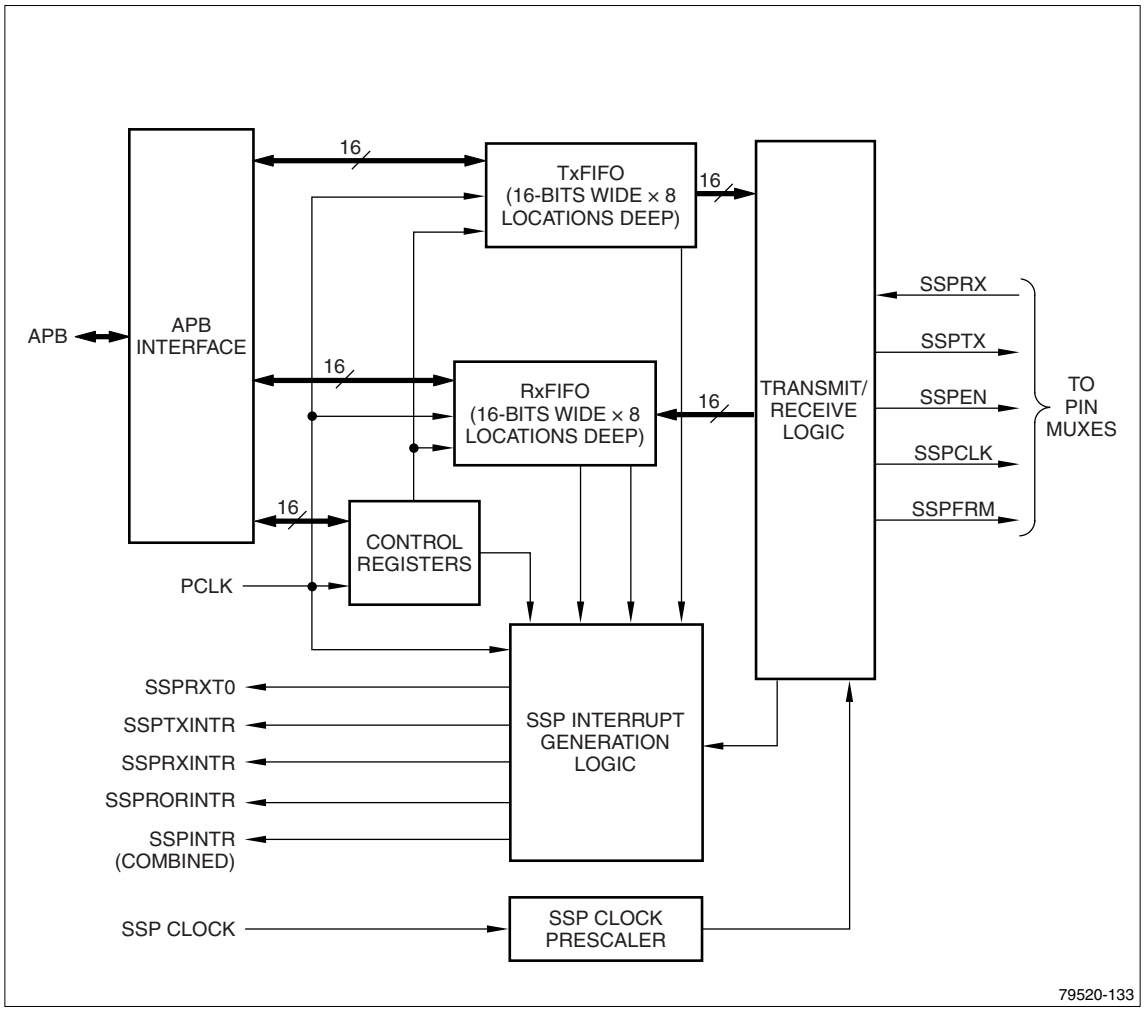


Figure 16-2. SSP Block Diagram

16.2 SSP Clock

Figure 16-1 shows that the LH79520 SSP peripheral receives the SSP CLOCK signal from the RCPC functional block. Figure 16-3 shows that SSP CLOCK is a derivative of HCLK, enabled and pre-scaled by PeriphClkCtrl2:SSPCLK and SSPClkPrescale:SSPPSVAL. The SSP CLOCK signal is disabled at reset and must be enabled for use by the SSP peripheral. For more information about programming the SSP CLOCK, see Chapter 7 – Reset, Clock Generation, and Power Control (RCPC).

Five programmable registers, shown in Figure 16-4, affect the SSP CLOCK signal. Software must condition these registers in order to obtain the desired output signal, SSPCLK. Four of the registers are in the SSP peripheral. The fifth register, SSPMux, is explained in Chapter 8, Section 8.4.3.

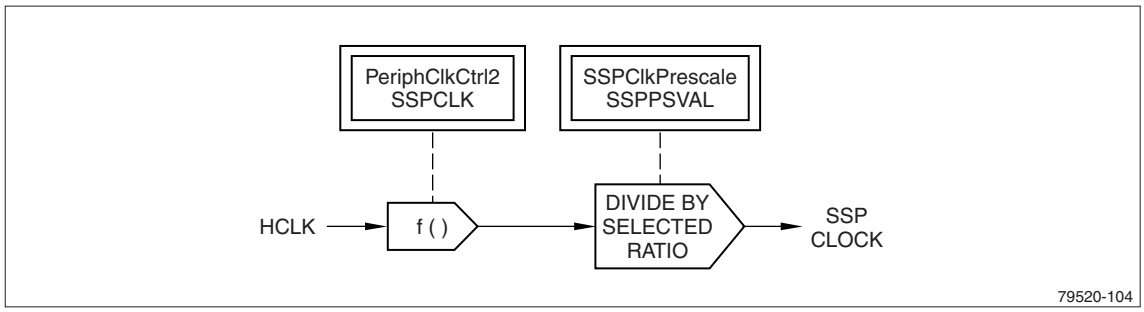


Figure 16-3. SSP Input Clock

Figure 16-4 illustrates that software must program the SSPCPSR and SSPCR0 registers to condition the SSP CLOCK signal to produce an SSPCLK output signal appropriate to the desired serial communication protocol.

Additional registers within the SSP (but not shown in Figure 16-4) must be programmed to enable the SSP peripheral, enable Tx and Rx interrupts and select the data frame format and data frame size. Depending upon the operating mode selected, the SSPFRM output signal can operate as an active HIGH data frame synchronization output for Texas Instruments' Synchronous Serial frame format or as an active LOW slave-select signal for Motorola SPI and National Semiconductor Microwire external devices.

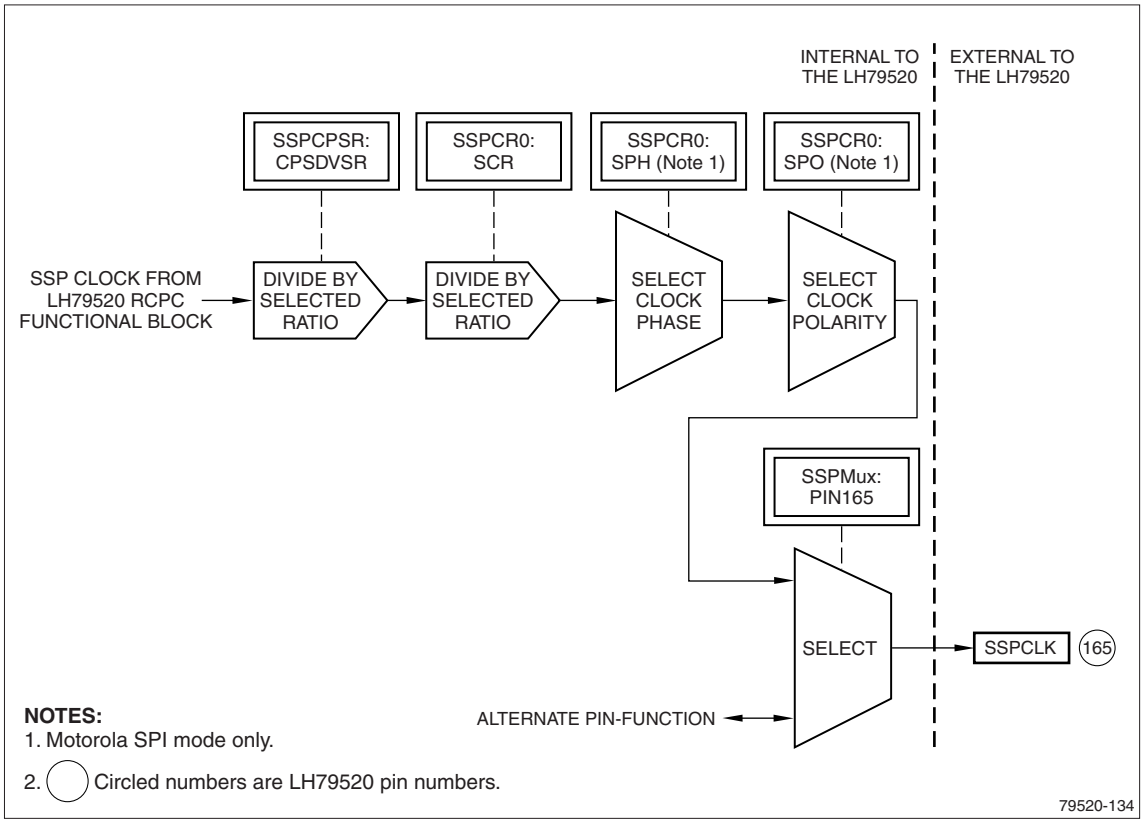


Figure 16-4. SSP Clock Conditioning

16.3 SSP FIFOs

Figure 16-2 shows that the SSP contains a Transmit FIFO and a Receive FIFO. Both of these FIFOs are 16-bits wide × 8-storage-locations deep. Neither FIFO can be flushed by hardware. Both FIFOs must be flushed by software.

To flush the Receive FIFO, software must read SSPDR until SSPSR:RNE = 0, indicating that the Receive FIFO is empty.

To flush the Transmit FIFO, software must first disable the SSP and select non-SSP functions for each of the SSP pins. Software must then re-program the SSP clock, re-enable the SSP, and poll SSPSR:BSY until the SSP is not busy.

To learn which functions are available on each LH79520 pin, see Table 1-1 and Table 1-3 in Chapter 1.

For information about selecting pin functions, see Chapter 8 – I/O Control and Multiplexing (IOCON).

16.4 SSP Data Formats

The SSP supports three data frame formats:

- Texas Instruments' synchronous serial
- Motorola SPI
- National Semiconductor Microwire

Each frame format is between 4 and 16 bits in length, depending upon the programmed data size, and each data frame is transmitted beginning with the MSB (Most Significant Bit). For all three formats, the SSP serial clock is held LOW (inactive) while the SSP is idle. The SSP serial clock transitions only during active transmission of data. The SSPFRM signal is used to mark the beginning and end of a frame. The SSPEN signal can be used to control an off-chip line driver's output enable pin.

The SSP supports both single transfer and continuous transfer operation. In single transfer operation, only one item is written to the transmit FIFO at a time; no other data items are written to the transmit FIFO until transmission is complete. In continuous operation, multiple data items are written to the transmit FIFO; a continuous transmission ends when the transmit FIFO is empty.

For Texas Instruments' synchronous serial frame format, the SSPFRM pin is pulsed for one serial clock period beginning at its rising edge, prior to each frame's transmission. For this frame format, both the SSP and the external slave device drive their output data on the rising edge of the clock and latch data from the other device on the falling edge. See Figure 16-5 and Figure 16-6.

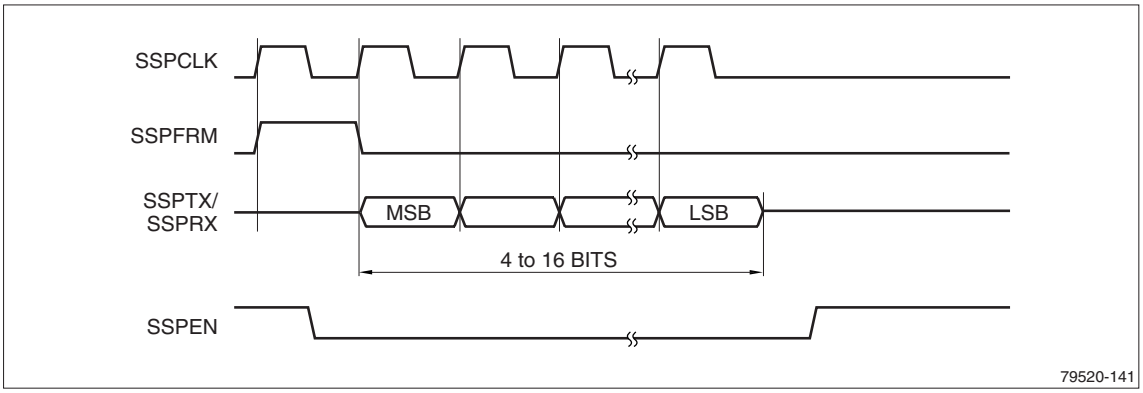


Figure 16-5. Texas Instruments Synchronous Serial Frame Format (Single Transfer)

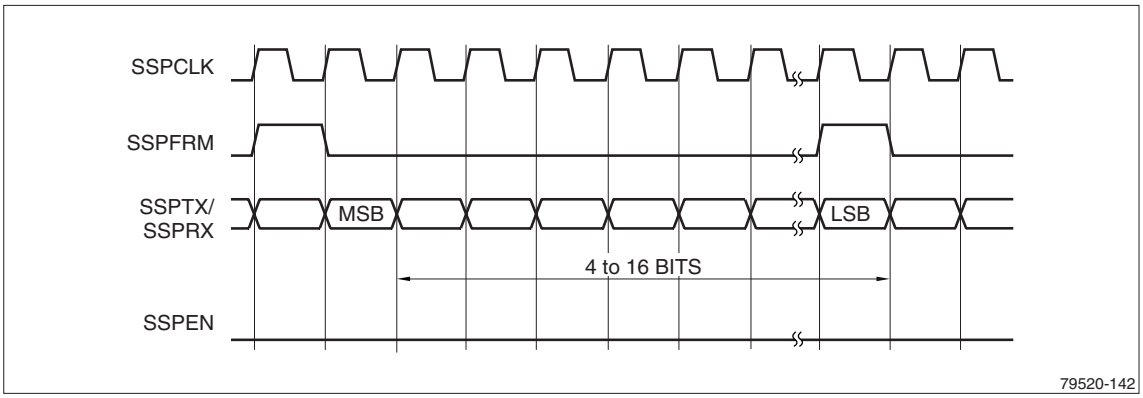


Figure 16-6. Texas Instruments Synchronous Serial Frame Format (Continuous Transfer)

For Motorola SPI format, the serial frame pin (SSPFRM) is active LOW. The SPO and SPH bits in SSP Control Register 0 (see Section 16.9.1) influence SSPCLK and SSPFRM operation in single and continuous modes. See Figures 16-7 through 16-14.

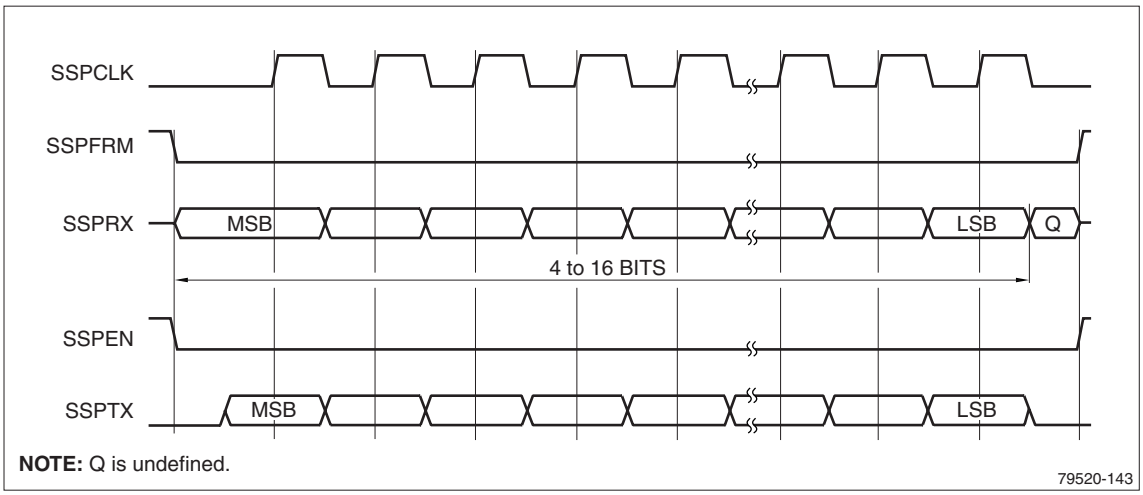


Figure 16-7. Motorola SPI Frame Format (Single Transfer) with SPO = 0 and SPH = 0

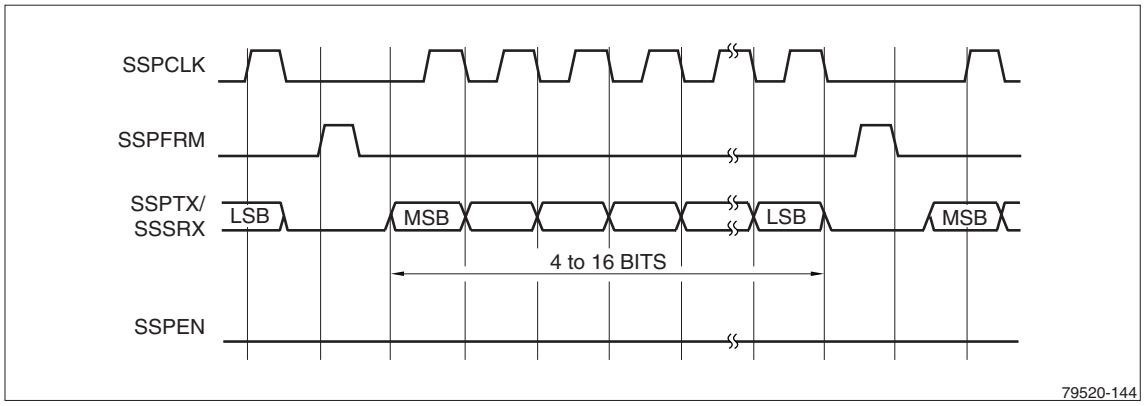


Figure 16-8. Motorola SPI Frame Format (Continuous Transfer) with SPO = 0 and SPH = 0

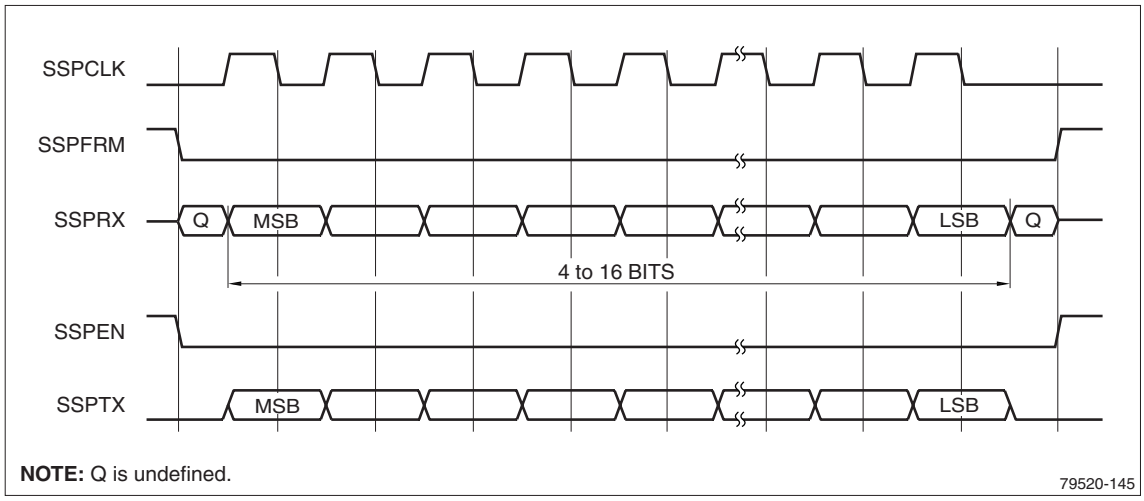


Figure 16-9. Motorola SPI Frame Format (Single Transfer) with SPO = 0 and SPH = 1

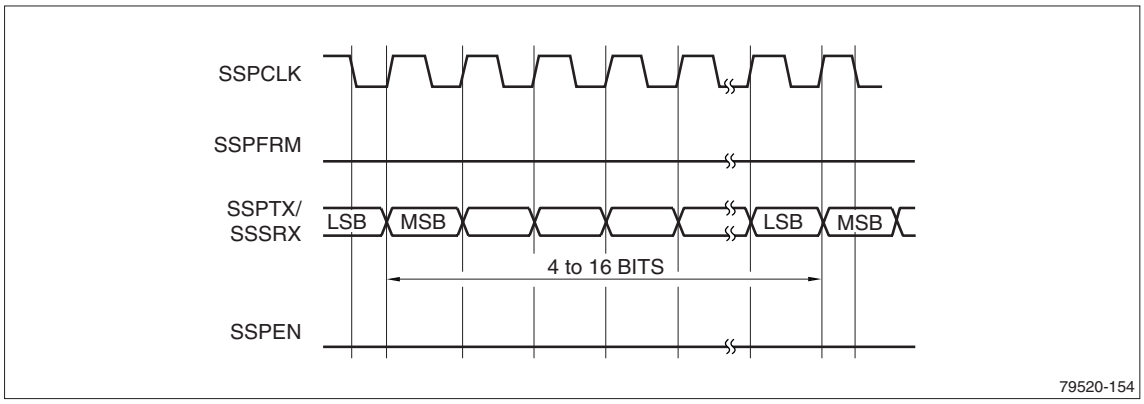


Figure 16-10. Motorola SPI Frame Format (Continuous Transfer) with SPO = 0 and SPH = 1

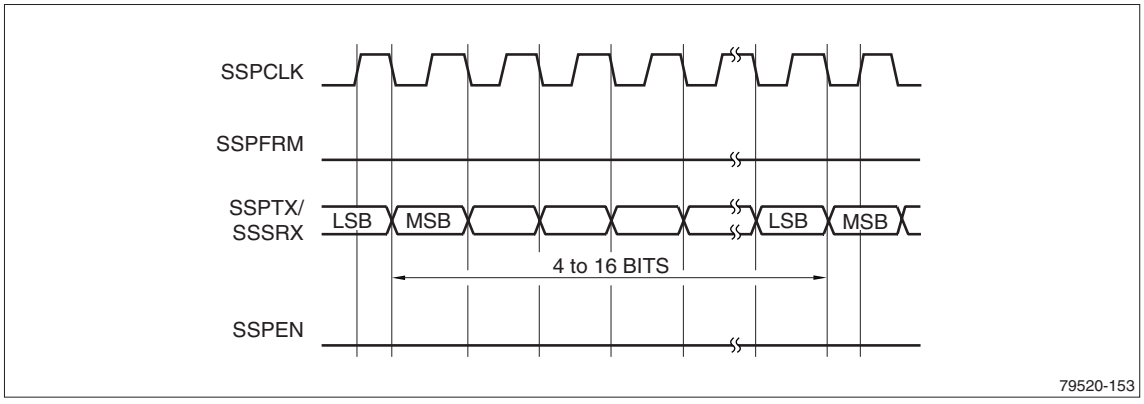


Figure 16-11. Motorola SPI Frame Format (Continuous Transfer) with SPO = 1 and SPH = 1

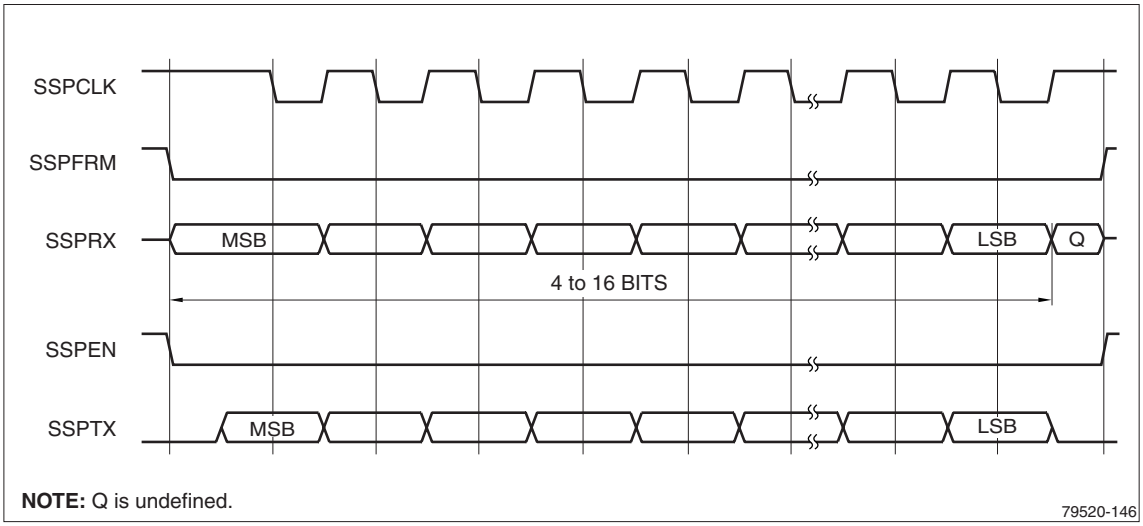


Figure 16-12. Motorola SPI Frame Format (Single Transfer) with SPO = 1 and SPH = 0

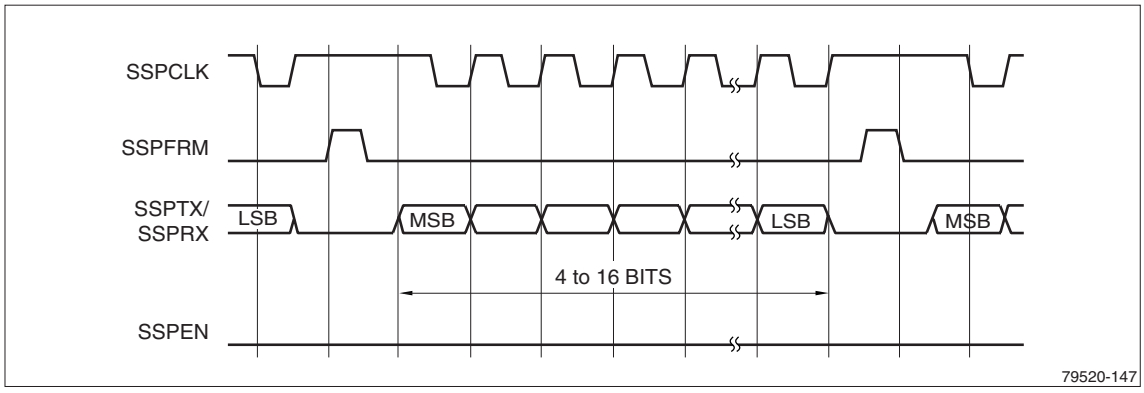


Figure 16-13. Motorola SPI Frame Format (Continuous Transfer) with SPO = 1 and SPH = 0

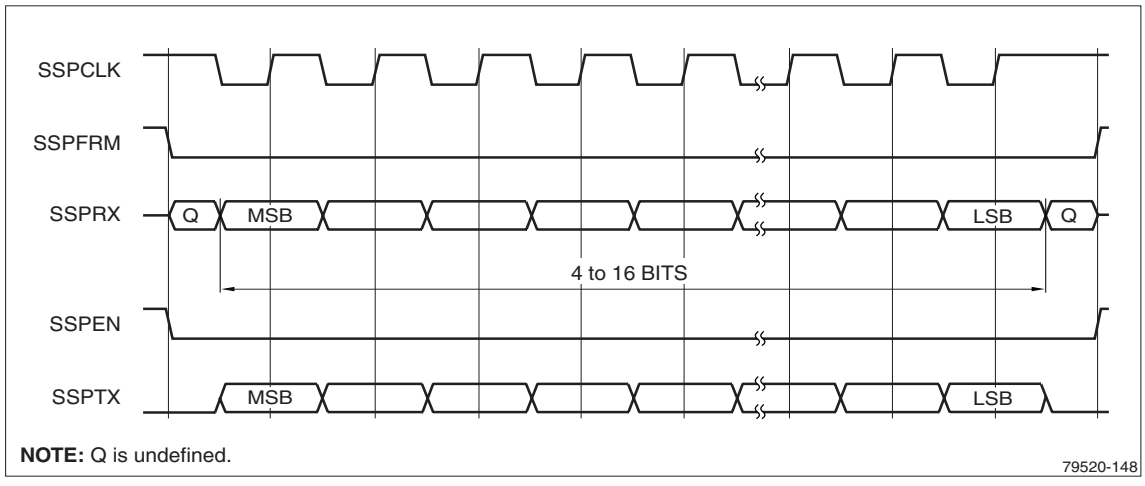


Figure 16-14. Motorola SPI Frame Format (Single Transfer) with SPO = 1 and SPH = 1

For National Semiconductor Microwire format, the serial frame pin (SSPFRM) is active LOW. Both the SSP and the external slave device drive their output data on the falling edge of the clock, and latch data from the other device on the rising edge of the clock. Unlike the full-duplex transmission of the other two frame formats, the National Semiconductor Microwire format utilizes a special master-slave messaging technique which operates at half-duplex. In this mode, when a frame begins, an 8-bit control message is transmitted to the off-chip slave. During this transmission, no incoming data is received by the SSP. After the message has been sent, the external slave device decodes the message and, after waiting one serial clock period after the last bit of the 8-bit control message was received, responds by returning the requested data. The returned data can be 4 to 16 bits in length, making the total frame length anywhere from 13 to 25 bits. See Figure 16-15 and Figure 16-16.

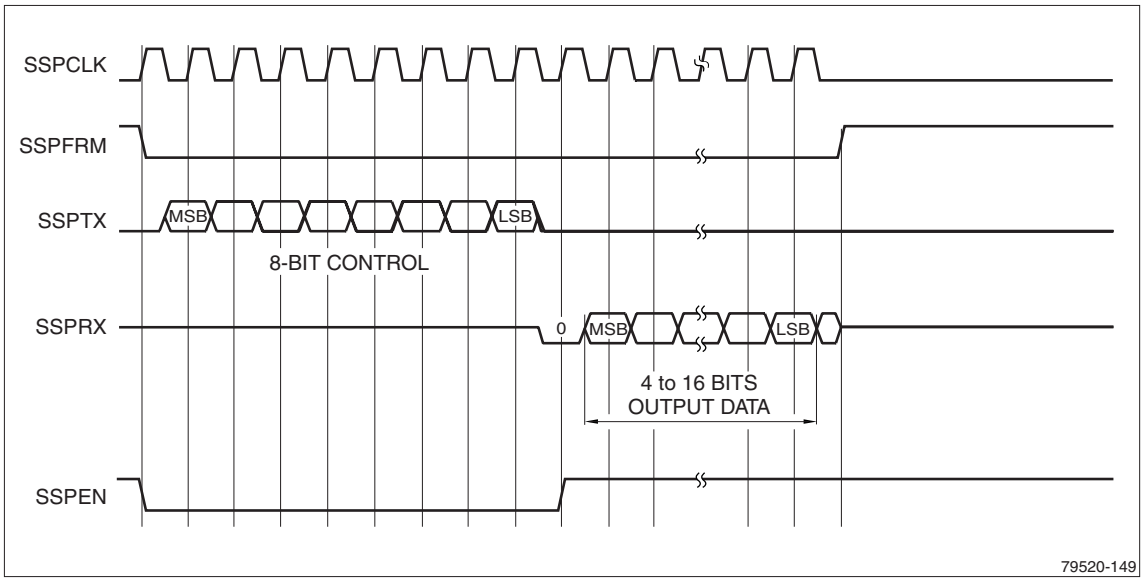


Figure 16-15. Microwire Frame Format (Single Transfer)

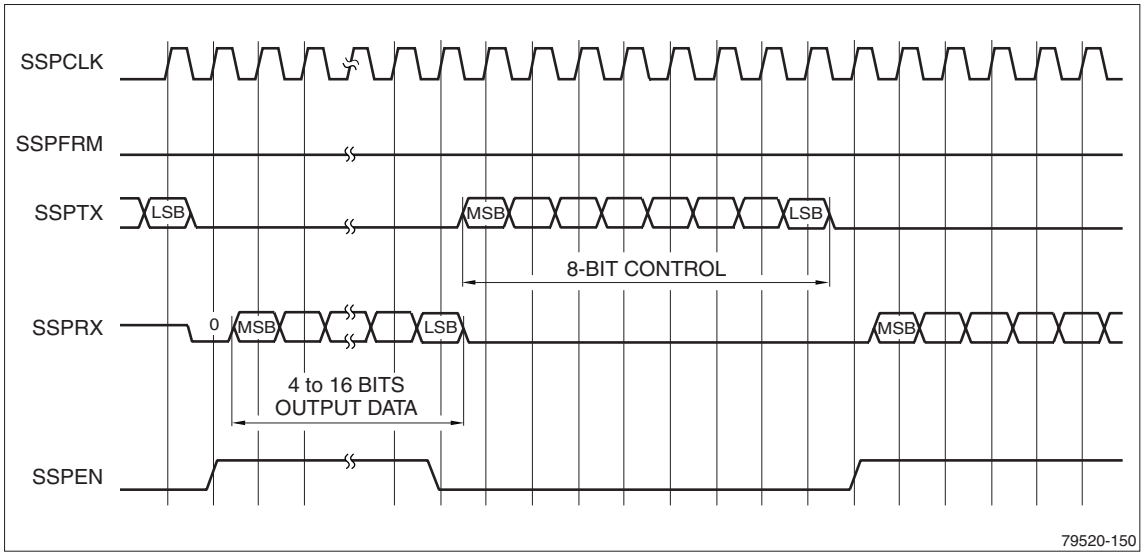


Figure 16-16. Microwire Frame Format (Continuous Transfers)

16.5 SSP Interrupts

The SSP can assert a total of five different interrupts. Figure 16-1 shows that all of the SSP interrupts are routed to the LH79520 VIC. Two of the interrupts, SSPTXINTR and SSPRXINTR are also routed to the LH79520 DMAC, to facilitate SSP DMA operations.

The SSP includes a Transmit FIFO and a Receive FIFO. The SSP can assert interrupts to request service for either FIFO or to indicate that an overrun condition exists in the Receive FIFO. The SSP can also assert interrupts to indicate, for SSP DMA operations, that a programmed period of time has elapsed since the Receive FIFO was first serviced by SSP DMA operation. The status of any of the interrupts which are maskable in the SSP can be read from the SSPIIR register.

Table 16-1 lists each of the SSP interrupts and explains masking restrictions. All SSP interrupts can be masked at the VIC, and the status of all SSP interrupts can be polled at the VIC.

The SSP Combined interrupt is the logical OR of the SSP interrupts connected to VIC channels 13, 14, and 15. Table 16-1 shows that the SSP Combined interrupt can be masked at the SSP by individually masking all three of the interrupts which form the SSP Combined interrupt. The SSP interrupt mask bits are located in the SSPCR1 register.

The SSPRXTO interrupt can be utilized to avoid leaving orphaned data in the SSP Receive FIFO when servicing the FIFO by DMA operation. The SSPRXTO interrupt can be masked only at the VIC. The status of the SSPRXTO interrupt can be read only from the VIC.

Table 16-1. SSP Interrupt Summary

SSP INTERRUPT NAME	VIC INTERRUPT CHANNEL	SSP INTERRUPT DESCRIPTION	SSP INTERRUPT TYPE
SSPRXTO	11	SSP Rx Timeout for SSP DMA Transfers	Maskable at the VIC but non-maskable at the SSP
SSPTXINTR	13	SSP Tx Interrupt	Maskable at the VIC and maskable at the SSP
SSPRXINTR	14	SSP Rx Interrupt	Maskable at the VIC and maskable at the SSP
SSPRORINTR	15	SSP ROR Interrupt	Maskable at the VIC and maskable at the SSP
SSPINTR	16	SSP Combined Interrupt (the logical OR of SSPTXINTR, SSPRXINTR, and SSPRORINTR)	Maskable at the VIC and maskable at the SSP by masking at the SSP each of the constituent SSP interrupts

16.5.1 SSPRXINTR Interrupt

The SSP will assert the SSP Receive Interrupt whenever the SSP Receive FIFO contains four or more frames of data. To clear this interrupt, software must enable the SSP and then repeatedly read from the SSPDR until the SSP Receive FIFO contains three or fewer frames of data. To utilize this interrupt to read data from the SSP Receive FIFO, software must unmask this interrupt at the SSP and mask this interrupt at the VIC.

16.5.2 SSPTXINTR Interrupt

The SSP will assert the SSP Transmit Interrupt whenever the SSP Transmit FIFO contains four or fewer frames of data. The SSP will also assert the SSP Transmit Interrupt if the SSP is disabled.

To clear this interrupt, software must enable the SSP and then repeatedly write to the SSPDR until the SSP Transmit FIFO contains five or more frames of data.

To utilize this interrupt for DMA transfers, software must unmask this interrupt at the SSP and mask this interrupt at the VIC.

Data can be written to the SSP Transmit FIFO prior to enabling both the SSP and the SSPTXINTR interrupt, or the SSP and SSPTXINTR interrupt can be enabled so that data can be written to the FIFO by an Interrupt Service Routine.

16.5.3 SSPRORINTR Interrupt

The SSP Receive FIFO can contain a maximum of eight frames of data. The SSP will assert the Receive Overrun Interrupt, SSPRORINTR, whenever the SSP attempts to write a frame of data to a Receive FIFO which is full (because it contains eight data frames). If the SSP attempts a write to a full FIFO, the attempt will fail and the received data will be lost.

To clear this interrupt, software must either perform a write access of the SSPICR register, or write SSPCR1:RORIE = 0.

16.5.4 SSPINTR Interrupt

The SSPINTR interrupt is the logical OR of three other individually maskable SSP interrupts: SSPTXINTR, SSPRXINTR, and SSPRORINTR. This combined interrupt will be asserted if any of the three individual interrupts are unmasked and asserted.

To clear this interrupt, software must clear any of the three individual SSP interrupts which are not masked. This interrupt must be masked at the VIC if either the SSP Transmit interrupt, SSPTXINTR, or the SSP Receive interrupt, SSPRXINTR, or both, is being utilized for SSP DMA transfers. See Table 16-1 for a summary of the SSP interrupts.

16.5.5 SSPRXTO Interrupt

During a SSP DMA operation, the SSP can assert the SSPRXTO interrupt to indicate that there are 3 or fewer (perhaps 0) data frames in the SSP Receive FIFO that could not be retrieved by the SSP DMA operation. An ISR should complete the data transfer by reading these items manually. To utilize this interrupt, software must enable the SSPRXTO interrupt at the LH79520 VIC. To clear this interrupt, software must write any value to the SSPRXTO register in the SSP.

The LH79520 DMA function cannot be used to receive three or fewer data frames from the SSP Receive FIFO because three or fewer data frames are insufficient to trigger a DMA transfer; the receive timeout function will never be armed by a first transfer.

See Section 16.9.8 for more information about the SSPRXTO interrupt.

16.6 Programming DMA Transfers via the SSP

The LH79520 DMAC (DMA Controller) can be programmed to transfer data to and from the on-chip SSP. Readers may wish to refer to Chapter 9 – Exceptions and Interrupts, and Chapter 10 – DMA Controller (DMAC), for additional information regarding the following programming sequence.

In the following programming sequence, the source address is a pointer to data which is to be transmitted from the LH79520 SSP to a destination which is external to the LH79520.

In the following programming sequence, the destination address is a pointer to the memory utilized by the LH79520 software to store data received by the SSP from a source which is external to the LH79520.

16.6.1 The SSP DMA Programming Sequence

1. Initialize the source address pointer to point to the first data which is to be transmitted from the SSP.
2. Initialize the destination address pointer to point to the first memory location at which data received by the SSP is to be stored.
3. Disable DMA Interrupts at the VIC (Vectored Interrupt Controller).
4. Disable all SSP interrupts to the VIC. This includes the SSPINTR Combined, SSPRXINTR, SSPTXINTR, SSPRORINTR, and SSPRXTO interrupts to the VIC. See Chapter 9, Table 9-2 for a list of the VIC Interrupt Channel Assignments.
5. Ensure that the DMA clock and SSP clock signals are enabled. Software may need to unlock the RCPC to enable these clock signals. Software can unlock the RCPC by writing to RCPC:WRTLOCK.
6. Program the SSP bit rate, data size, and data frame formats.
7. Configure the IOCON to disconnect the SSP from the SSP output pins.
8. Enable the SSP.
9. Flush the SSP Transmit FIFO and the SSP Receive FIFO, if these FIFOs are not already flushed.
10. Configure the IOCON to connect the SSP to the SSP output pins.

11. Program DMA Stream1 (the SSP Transmit stream):

    DMASourceLo = the lower 16 bits of the physical source address

    DMASourceHi = the upper 16 bits of the physical source address

    DMADestLo = the lower 16 bits of the SSP data register physical address

    DMADestHi = the upper 16 bits of the SSP data register physical address

    DMAMax = the number of SSP data frames (i.e.: the quantity of data units to transfer)

    DMACtrl = the amount by which to increment the source address. Use a 2-byte source size, a 2-byte destination size, select the wrapping mode of addressing, consider the destination to be a peripheral, and select an incrementing burst length of 4.

    Note that it is necessary to specify 2-byte source and destination sizes (i.e.: to use 16-bit transfers), even if the data being transferred is less than 16 bits in size).

    Note that the DMA is not yet enabled.

12. Program DMA Stream0 (the SSP Receive stream)

    DMASourceLo = the lower 16 bits of the SSP data register physical address

    DMASourceHi = the upper 16 bits of the SSP data register physical address

    DMADestLo = the lower 16 bits of the physical destination address

    DMADestHi = the upper 16 bits of the physical destination address

    DMAMax = the number of SSP data frames (i.e.: the quantity of data units to transfer)

    DMACtrl = the amount by which to increment the destination address. Use a 2-byte source size, a 2-byte destination size, select the wrapping mode of addressing, consider the source to be a peripheral, and select an incrementing burst length of 4.

    Note that is necessary to specify 2-byte source and destination sizes (i.e.: to use 16-bit transfers), even if the data being transferred is less than 16 bits in size).

    Note that the DMA is not yet enabled.

13. Software must enable, then disable, the Stream0 DMA by a write to Stream0 DMACtrl:ENABLE.
14. Clear any previous SSP DMA complete interrupts.
15. Program the SSP DMA receiver timeout appropriately for the transmit time and the number of bits per data frame.
16. Enable the SSP transmit and receive interrupts at the SSP. This permits the SSP FIFO interrupts to request DMA transfers from the DMAC.
17. Install an SSP interrupt-handler to service the SSP Receive Timeout interrupt. This handler must transfer the final few data frames from the SSP Receive FIFO to the destination addresses. The handler will need to be able to transfer from 0 to 3 data frames from the SSP Receive FIFO to the final destination addresses.
18. Enable the SSP receive timeout interrupt at the VIC.
19. Write to the Stream0 DMACtrl:ENABLE. On completion of this write operation, the DMA is ready to transfer received data.
20. Write to the Stream1 DMACtrl:ENABLE. On completion of this write operation, the DMA will load the SSP transmit FIFO and begin to transfer data.

16.7 SSP Programmer's Model

The Register Base Address of the LH79520 Synchronous Serial Port is:

SSPBase: 0xFFFC6000

All registers in the SSP must be accessed as a full word. The SSP does not support byte and half-word writes. Programmers should ensure that the system clock is enabled before programming any registers.

16.8 SSP Register Summary

The programmable SSP registers are shown in Table 16-2.

Table 16-2. SSP Register Summary

ADDRESS	NAME	DESCRIPTION
SSPBase + 0x000	SSPCR0	SSP Control Register 0
SSPBase + 0x004	SSPCR1	SSP Control Register 1
SSPBase + 0x008	SSPDR	SSP Receive FIFO Register (Read)/ SSP Transmit FIFO Data Register (Write)
SSPBase + 0x00C	SSPSR	SSP Status Register
SSPBase + 0x010	SSPCPSR	SSP Clock Prescale Register
SSPBase + 0x014	SSPIIR/SSPICR	SSP Interrupt Identification Register (Read)/ SSP Interrupt Clear Register (Write)
SSPBase + 0x018	SSPRXTO	SSP DMA Receive Timeout Register
SSPBase + 0x01C - SSPBase + x0FF	///	Reserved

16.9 SSP Register Descriptions

This section of this chapter of this User's Guide lists and describes the registers that configure the LH79520 SSP peripheral.

16.9.1 SSP Control Register 0

The SSPCR0 register establishes the Transmit and Receive data frame formats and controls various other functions within the SSP. Table 16-4 describes the bit fields in the SSPCR0 register.

Table 16-3. SSPCR0 Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	SCR															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	SCR								SPH	SPO	FRF		DSS			
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	PWMBase + 0x00															

Table 16-4. SSPCR0 Register Bits

BITS	FIELD NAME	FUNCTION
31:8	SCR	**Serial Clock Rate** SCR generates the SSP's transmit and receive bit rates. The bit rate is: fSSPCLK ÷ (CPSDVSR × (1 + SCR)), where CPSDVSR is an even value from 2 to 254 and SCR is a value from 0 - 255. 0 = reset
7	SPH	**SCLK Phase** Applicable to Motorola SPI data frame format only. 0 = The SSPFRM signal is LOW for each data frame, but HIGH between data frames (reset) 1 = The SSPFRM signal is continuously LOW for all data frames. The SSPFRM signal does NOT return HIGH between data frames.
6	SPO	**SCLK Polarity** Applicable to Motorola SPI data frame format only. 0 = reset
5:4	FRF	**Data Frame Format** 0b11 = Reserved. Write '0'. Reads unpredictable. 0b10 = National Microwire data frame format 0b01 = TI synchronous serial data frame format 0b00 = Motorola SPI data frame format (reset)
3:0	DSS	Data Size Select 0000 = Reserved, do not write 0001 = Reserved, do not write 0010 = Reserved, do not write 0011 = 4-bit data 0100 = 5-bit data 0101 = 6-bit data 0110 = 7-bit data 0111 = 8-bit data 1000 = 9-bit data 1001 = 10-bit data 1010 = 11-bit data 1011 = 12-bit data 1100 = 13-bit data 1101 = 14-bit data 1110 = 15-bit data 1111 = 16-bit data

16.9.2 SSP Control Register 1

The SSPCR1 register establishes the Transmit/Receive handling characteristics for the SSP. Table 16-6 describes the bit fields in SSPCR1.

Table 16-5. SSPCR1 Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///											SSE	LBM	RORIE	TIE	RIE
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	PWMBase + 0x04															

Table 16-6. SSPCR1 Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:5	///	**Reserved** Write the Reset value.
4	SSE	**Synchronous Serial Port Enable** 0 = SSP operation disabled. (reset) 1 = SSP operation enabled.
3	LBM	**Loopback Mode** 0 = Normal serial port operation enabled. (reset) 1 = Output of the transmit serial shifter is connected internally to the input of the receive serial shifter.
2	RORIE	**Receive FIFO Overrun Interrupt Enable** 0 = Overrun detection is disabled. An overrun condition will not generate the SSPRORINTR interrupt (reset). Clearing this bit to zero also clears the SSPRORINTR interrupt if the SSPRORINTR interrupt is already asserted. 1 = Overrun detection is enabled. An overrun condition will generate the SSPRORINTR interrupt.
1	TIE	**Transmit FIFO Interrupt Enable** 0 = Transmit FIFO half-full (or less) does not generate the SSPTXINTR interrupt (reset). 1 = Transmit FIFO half-full (or less) generates the SSPTXINTR interrupt.
0	RIE	**Receive FIFO Interrupt Enable** 0 = Receive FIFO half-full (or more) condition does not generate the SSPRXINTR interrupt (reset). 1 = Receive FIFO half-full (or more) condition generates the SSPRXINTR interrupt.

16.9.3 SSP Data Register

The SSPDR register contains the 16-bit SSP data word. Table 16-8 describes the bit fields in the SSPDR register.

Reads of the SSPDR access the entry in the receive FIFO pointed-to by the current FIFO read pointer. As data values are removed by the SSP's receive logic from the incoming data frame, the data values are placed into the receive FIFO location pointed-to by the current FIFO write pointer.

Writes of the SSPDR place data in the transmit FIFO location pointed-to by the write pointer. The transmit logic removes data values from the transmit FIFO one-value-at-a-time. Data removed from the transmit FIFO is loaded to the transmit serial shifter and shifted serially onto the SSPOUT pin at the programmed bit rate.

When a data size of less than 16 bits is selected, the software must right-justify data written to the transmit FIFO; the transmit logic will ignore the unused bits. Received data of less than 16 bits is automatically right-justified in the receive buffer.

When the SSP is programmed for the National MICROWIRE data frame format, the default size for transmitted data is eight bits (the most significant byte is ignored). The received data size is controlled by software. The transmit FIFO and the receive FIFO are not cleared even when SSPCR1:SSE = 0. This allows the software to fill the transmit FIFO before enabling the SSP.

Table 16-7. SSPDR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	DATA															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	PWMBase + 0x08															

Table 16-8. SSPDR Register Bits

BITS	FIELD NAME	FUNCTION
31:16	///	**Reserved** Write the Reset value.
15:0	DATA	**Transmit/Receive FIFO** Read = Receive FIFO Write = Transmit FIFO Software must right-justify data when the SSP peripheral is programmed for a datasize that is less than 16 bits. Unused bits at the top are ignored by Transmit logic. The Receive logic automatically right-justifies incoming data.

16.9.4 SSP Status Register

Reads of the SSPSR register return the SSP FIFO fill status and the SSP busy status.

The SSPSR is a read-only register which is reset to a value of 0x0003, which indicates that the Transmit FIFO is not full and the Transmit FIFO is not half-full.

Table 16-10 defines the bit fields in the SSPSR.

Table 16-9. SSPSR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///											BSY	RFF	RNE	TNF	TFE
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
ADDR	PWMBase + 0x0C															

Table 16-10. SSPSR Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:5	///	**Reserved** Read-only. Reads unpredictable.
4	BSY	**SSP Busy Flag** (read only) 0 = The SSP is idle (reset) 1 = The SSP is currently transmitting and/or receiving a frame of data, or the transmit FIFO is non-empty
3	RFF	**Receive FIFO Full** (read only) 0 = the Receive FIFO is not full (reset) 1 = the Receive FIFO is full
2	RNE	**Receive FIFO Not Empty** 0 = the Receive FIFO is empty (reset) 1 = the Receive FIFO is not empty
1	TNF	**Transmit FIFO Not Full** 0 = the Transmit FIFO is full 1 = the Transmit FIFO is not full (reset)
0	TFE	**Transmit FIFO Empty** 0 = the Transmit FIFO is not empty 1 = the Transmit FIFO is empty (reset)

16.9.5 SSP Clock Prescale Register

The SSPCPSR register specifies the division factor for the input SSPCLK.

The value programmed to this register should be an even number between 2 and 254 (decimal). The least significant bit of the value written must be zero.

The LSB of this register is fixed at '0'; it cannot be written to '1'. If an odd number is written to this register, the least-significant bit will be ignored. Data read back from this register will have the least significant bit = 0.

Table 16-12 shows the bit assignments for SSPCPSR.

Table 16-11. SSPCPSR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								CPSDVSR							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RO
ADDR	PWMBase + 0x10															

Table 16-12. SSPCPSR Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:8	///	**Reserved** Write the Reset value.
7:0	CPSDVSR	**Clock Prescale Divisor** Must be an even number from 2 to 254 (decimal), depending on the frequency of SSPCLK. The least significant bit always returns '0' on reads. This bit field is reset to 0.

16.9.6 SSP Interrupt Identification Register

The read-only SSPIIR register is mapped to the same address as the write-only SSPICR register. Reads of the SSPIIR register return interrupt status information.

The SSPIIR register is a read-only register. See Section 16.9.7 for additional information.

Table 16-14 defines the bit fields in the read-only SSPIIR register.

Table 16-13. SSPIIR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///													RORIS	TIS	RIS
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
ADDR	PWMBase + 0x14															

Table 16-14. SSPIIR Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:3	///	**Reserved** Read only. Read '0'.
2	RORIS	**SSP Receive FIFO Overrun Interrupt Status** 0 = SSPRORINTR is not asserted (reset) 1 = SSPRORINTR is asserted
1	TIS	**SSP Transmit FIFO Service Request Interrupt Status** 0 = SSPTXINTR is not asserted (reset) 1 = SSPTXINTR is asserted
0	RIS	**SSP Receive FIFO Service Request Interrupt Status** 0 = SSPRXINTR is not asserted (reset) 1 = SSPRXINTR is asserted

16.9.7 SSP Interrupt Clear Register

The write-only SSPICR register is mapped to the same address as the read-only SSPIIR register. A write to the SSPICR register clears the SSP Receive FIFO Overrun Interrupt.

The SSPICR register is a write-only register. No values are retained. See Section 16.9.6 for additional information.

A write of any value to this register clears the SSP receive FIFO overrun interrupt (SSPCR1:RORIE) to '0'. A write to this register does not clear any other SSP interrupt. SPSCR1:RORIE may also be cleared directly by writing a '0' to SPSCR1:RORIE.

Table 16-16 shows the bit assignments for the write-only SSPICR register.

Table 16-15. SSPICR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET																
TYPE	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	CLEAR															
RESET																
TYPE	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
ADDR	PWMBase + 0x14															

Table 16-16. SSPICR Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:16	///	**Reserved** Write only. Value written does not matter.
15:0	CLEAR	**Interrupt Clear** Any write access of this register clears the Receive FIFO Overrun Interrupt, SSPCR1:RORIE, regardless of the data value written. This is a write only bit field.

16.9.8 SSP DMA Receive Timeout Register

The SSPRXTO register can be programmed with a 16-bit value which determines the delay, in PCLK cycles, before the SSPRXTO interrupt will be generated. PCLK is locked to HCLK in the LH79520.

DMA transfers to and from the SSP are triggered by the SSP transmit and receive interrupts, SSPTXINTR and SSPRXINTR, respectively. If the end of a DMA transfer leaves 3 or fewer frames of data in the SSP Receive FIFO, the LH79520 DMAC will be unable to transfer these orphaned data frames from the SSP Receive FIFO because the SSPRXINTR interrupt is not asserted. The LH79520 SSP provides the SSPRXTO register and the SSPRXTO interrupt to handle this situation.

Prior to the first DMA transfer from the SSP receive FIFO, the SSP Receive Timer is stopped. After the first DMA transfer from the SSP Receive FIFO, the SSP Receive Timer is loaded with a the value from the SSPRXTO register. The control logic in the SSP will decrement the SSP Receive Timer by one count at each APB (PCLK) cycle. When the SSP Receive Timer is decremented to a value of zero, the SSP Control logic will assert the SSPRXTO interrupt. If the timer is already zero, then the receive timer will never decrement and will never assert the SSPRXTO interrupt. If the SSPRXTO interrupt does assert, then there are three or fewer (perhaps 0) entries in the SSP receive FIFO that DMA could not retrieve. An interrupt handler should complete the DMA transfer by retrieving these items manually.

The SSPRXTO register is a read-write register which is reset to 0. Writing any value to this register will clear an asserted SSPRXTO interrupt. Reads of this register will return the current value of the timer. Table 16-8 describes the bit fields in the SSPRXTO register.

Table 16-17. SSPRXTO Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	TOCOUNT															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	PWMBase + 0x018															

Table 16-18. SSPDR Register Bits

BITS	FIELD NAME	FUNCTION
31:16	///	**Reserved** Write the Reset value.
15:0	TOCOUNT	**Timeout Count** A quantity of PCLK cycles of delay. 0 = Generation of the SSPRXTO interrupt is disabled (Reset) 65,535 = MAX.

The SSPRXTO register can be utilized as a guard timer during DMA transfers from the SSP Receive FIFO, to avoid leaving orphaned data in the FIFO. The SSPRXTO interrupt will be asserted when a DMA transfer from the SSP Receive FIFO is ongoing, the value in the SSPRXTO register was non-zero when the DMA operation began, and the value in the SSPRXTO register is decremented to zero by PCLK.

The period of time which must elapse before the SSPRXTO Interrupt will be asserted is determined by the value programmed to the SSPRXTO register, in PCLK cycles. The value in the SSPRXTO register is not pre-scaled, which allows a maximum PCLK count of 65,535 cycles. For a PCLK frequency (locked to HCLK in the LH79520) of 50 MHz, the maximum period of delay that can be programmed is approximately 1.3 μs.

Chapter 17
Universal Asynchronous Receiver/Transmitters (UARTs)

The LH79520 includes three UART peripherals, identified as UART0, UART1, and UART2. UART0 includes a Serial InfraRed (SIR) ENDEC (Encoder/Decoder), and is capable of generating two interrupts not available from UART1 or UART2. This Chapter explains the theory and operation of the LH79520 UART peripherals.

17.1 Theory of Operation

The three UART peripherals are integrated into the LH79520 MCU as shown in Figure 17-1. Each LH79520 UART is capable of simultaneously sending and receiving serial data. Each UART can receive data which is transmitted asynchronously to the LH79520 clock system. UART0 can also be utilized for IrDA-compatible Serial InfraRed (SIR) communications.

The ARM720T core writes data and control information to a UART through the UART's APB interface, via the Advanced Peripheral Bus Bridge. The core reads UART status information through the same APB interface.

Each UART receives a scaled clock signal and a global reset signal from the LH79520 RCPC functional block. Each UART can generate interrupts from seven different sources and assert one combined interrupt (representing any of the seven sources) to the LH79520 VIC. In addition to issuing a combined interrupt, UART0 can generate two discrete interrupt signals concerning UART0 FIFO-levels.

UART peripherals receive serial data from external devices, check the received data for errors, and convert the data to parallel format for storage. UART peripherals also convert parallel data to serial format and transmit the data serially to an external device. Each LH79520 UART is functionally similar to the industry-standard 16C550 UART device.

Figure 17-2 shows a simplified block diagram of one LH79520 UART. Each UART includes programmable control registers, a baud rate generator, and FIFOs to buffer the received (i.e. inbound), and the transmitted (i.e. outbound), serial data.

The UART transmit and receive data paths are each buffered with internal FIFO memories. The receive FIFO provides up to 16 storage locations for received data and the transmit FIFO can buffer up to 16 bytes of outgoing data. The FIFOs may also be programmed to be 1-byte deep, providing a conventional double-buffered UART interface.

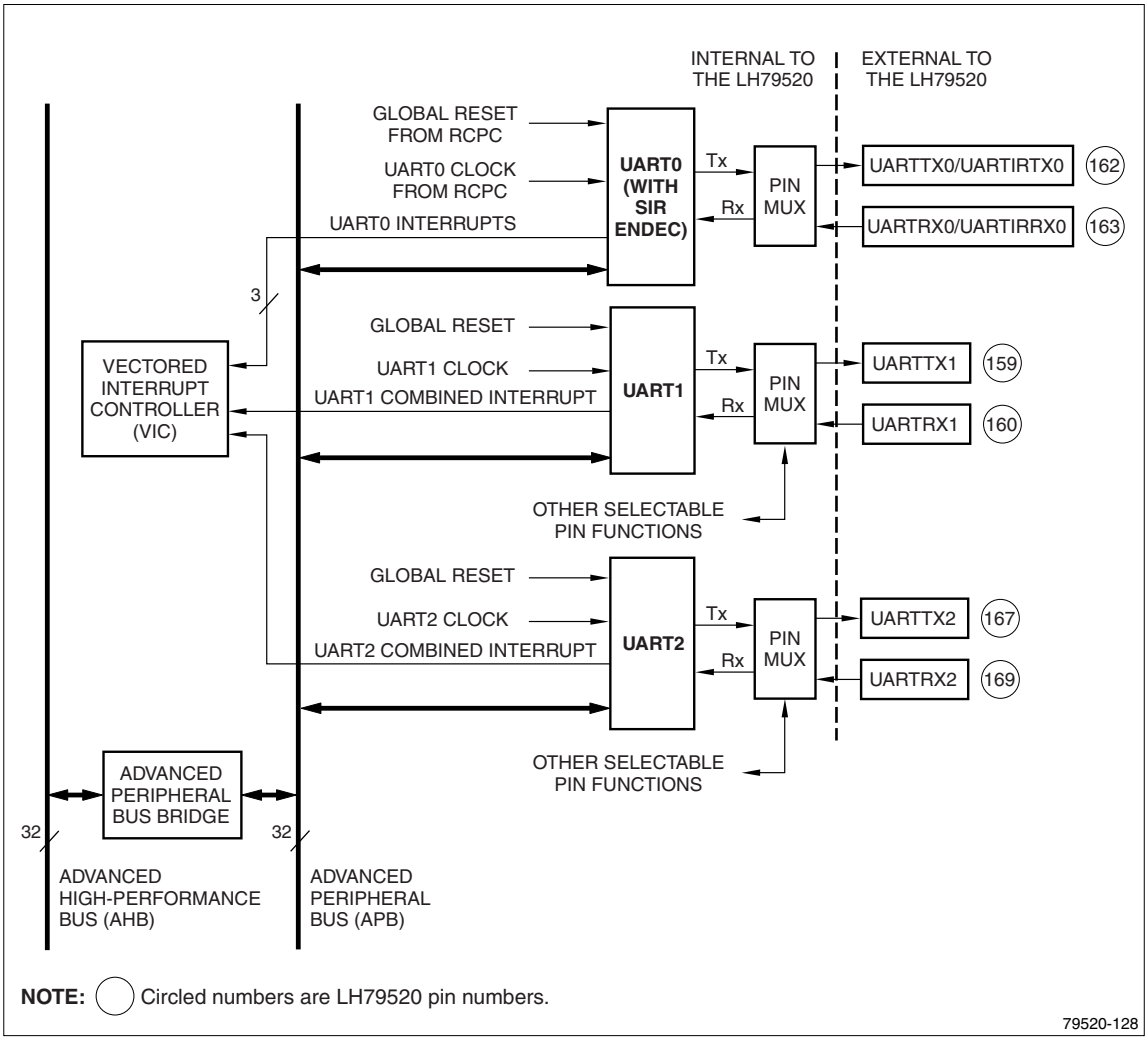


Figure 17-1. LH79520 UART Peripherals

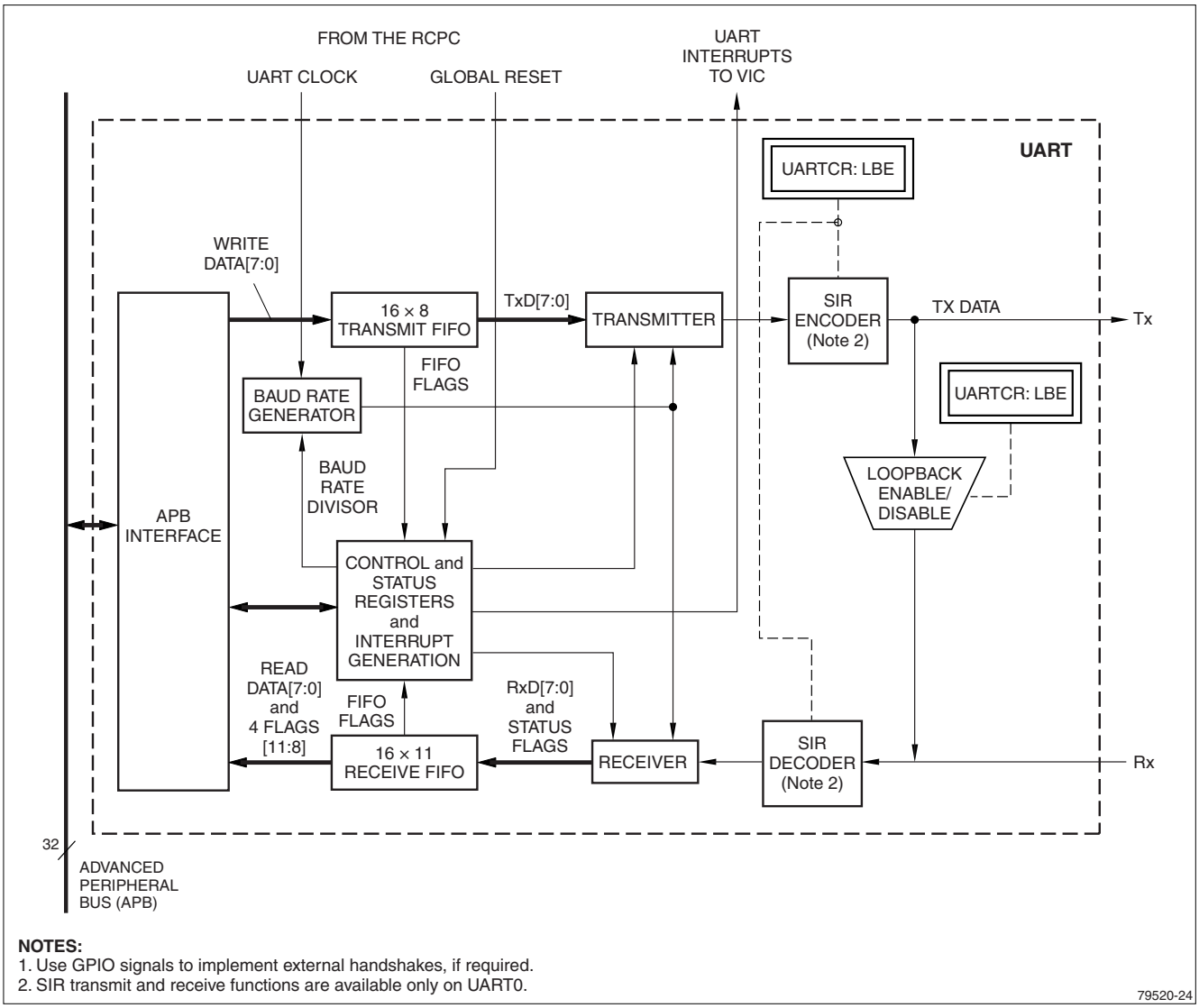


Figure 17-2. LH79520 UART Simplified Block Diagram

If a FRAMING, PARITY, or BREAK error occurs while serial data is being received, the appropriate error bit is stored in the FIFO, along with the data. If an OVERRUN condition (error) occurs while serial data is being received, the OVERRUN error is not stored in the FIFO. Instead, the OVERRUN error is immediately flagged by a bit available to software in the UARTDR and UARTRSR registers. Data already in the FIFO is protected (not overwritten).

The LH79520 includes in UART0 an IrDA-compatible Serial InfraRed (SIR) Encoder/ Decoder (ENDEC) block which supports serial infrared communications protocol and can be operated in two modes: a normal mode or a low-power mode. The SIR ENDEC shown in Figure 17-2 is available only on UART0. UART1 and UART2 do not include a SIR ENDEC. Programming UART0 to function as a SIR interface involves enabling the SIR ENDEC and programming an additional baud rate division if the SIR is to be operated in the low-power mode. The SIR ENDEC can be enabled for serial IR communication through pins UARTIRTX0 and UARTIRRX0. The SIR ENDEC is half-duplex only, so it cannot receive while transmitting, or transmit while receiving. The IrDA SIR physical layer specifies a minimum 10 μs delay between transmission and reception. This delay must be provided in software. For additional information regarding the SIR, refer to section Section 17.5.11, and also see Chapter 18 – IrDA Communications (SIR).

The operation of each UART, including the UART's baud rate, is controlled by the UART's line control register (UARTLCR_H) and the UART's two baud rate divisor registers (UARTIBRD and UARTFBRD). The baud rate at which each UART operates is scaled from a UART Clock signal generated in the LH79520 RCPC.

17.1.1 Data Transmission or Reception

Each UART is equipped with two FIFOs, for Transmit and Receive data. Both FIFOs are 16 locations in size, but the Receive FIFO has an additional four bits per location, to store status information about the received byte.

Data to be transmitted is written to the transmit FIFO. Data entering the Transmit FIFO causes a data frame to begin being transmitted, according to the parameters programmed to the UARTLCR_H register. Outgoing data will continue to be transmitted until the transmit FIFO is empty. UARTFR:BUSY be set to '1' when data is written to the transmit FIFO, and UARTFR:BUSY will remain '1' until the Transmit FIFO is empty and the last bit of the last character has been transmitted from the transmit shift register, including the STOP bits. If the UART is transmitting data, UARTFR:BUSY can remain '1' even though software has disabled the UART.

To detect incoming data, the UART receiver samples the incoming data at a rate determined by UART clock frequency and the UART's programmable parameters, such as the baud rate. For each sample, the receiver takes three readings and retains the majority value.

When the receiver is idle (the RxD input is continuously a '1'), and a '0' is detected on the RxD input (signalling that a START bit has been received), the receive counter, with the clock enabled, begins running and the data is sampled on the eighth cycle of a counter (half way through its bit period).

The START bit for the incoming data is valid if the input RxD is still '0' on the eighth cycle of the counter. If the RxD input is no longer '0' on the eighth cycle of the counter, the START bit is ignored.

If the START bit is found to be valid, successive data bits are sampled on every 15th cycle of the clock (one bit period later), according to the programmed length of the data characters. If PARITY is enabled, then the PARITY bit is also checked.

When the START, DATA and PARITY bits have been determined, a valid STOP bit is confirmed if the RxD input is '1' at the appropriate time, otherwise a FRAMING error has occurred. When a complete byte is received, including the START, DATA, PARITY and STOP bits, then the data is stored in the Receive FIFO, with any error bits generated during the reception of that byte.

17.1.2 UART Pin Multiplexing

Each of the LH79520 pins utilized for a UART function is multiplexed with another function. Figure 17-3 shows a typical UART and demonstrates that, except in the case of UART0, the pin multiplexing is external to the UART peripheral.

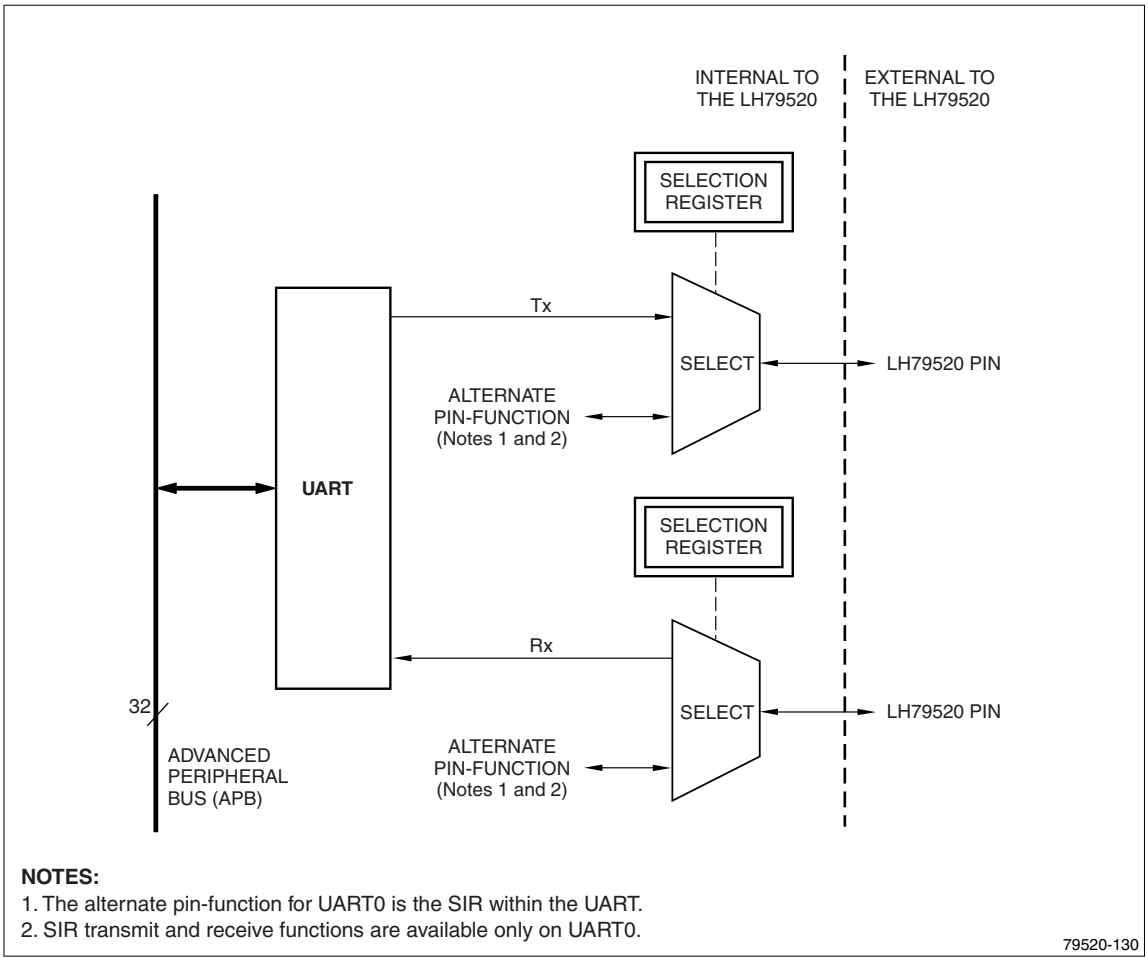


Figure 17-3. LH79520 UART Pin Multiplexing

Table 17-1 lists the LH79520 pins associated with the UART peripherals, the functions available on each pin, and the register which selects the alternate function.

Only UART0 is available at reset; UART1 and UART2 must be selected by software.

Table 17-1. UART Signal Multiplexing

LH79520 PIN NUMBER	PIN FUNCTION AT RESET	ALTERNATE PIN FUNCTION	SELECTION REGISTER	REGISTER BASE ADDRESS
159	PA4	UARTTX1	UARTMux	IOCONBase
160	PA3	UARTRX1	UARTMux	IOCONBase
162	UARTTX0	UARTIRTX0	UART0CR	UART0Base
163	UARTTX0	UARTIRTX0	UART0CR	UART0Base
167	SSPTX	UARTTX2	SSPMux	IOCONBase
169	SSPRX	UARTRX2	SSPMux	IOCONBase

17.1.3 UART Programmable Parameters

The following parameters are programmable for each UART:

- Communication baud rate (as integer and fractional parts)
- Quantity of data bits and quantity of STOP bits
- PARITY
- Tx and Rx FIFO size, 1-byte depth or 16-byte depth
- FIFO interrupt trigger levels are selectable at 1/8, 1/4, 1/2, 3/4, and 7/8 full

UART0 includes an IrDA-compatible Serial InfraRed (SIR) module:

- Internal nominal SIR internal clock frequency of 1.8432 MHz clock (1.42 MHz - 2.12 MHz) to generate shorter bit durations in SIR low-power mode

17.1.4 Variations from the 16C550 UART

This section of this User's Guide lists the ways in which the UART peripherals in the LH79520 MCU differ from the industry-standard 16C550 UART device:

- Receive FIFO trigger levels are 1/8, 1/4, 1/2, 3/4, 7/8
- Receive errors are stored in the FIFO, and do not generate an interrupt
- The internal register map address space, and each register's bit function, differ from those of the 16C550 UART.
- UART0 in the LH79520 includes a SIR (Serial InfraRed) ENDEC (Encoder\Decoder).

The following 16C550 UART features are *not* supported:

- 1.5 STOP bits (1 or 2 STOP bits only are supported)
- Modem control signals

17.1.5 UART Transmit and Receive FIFOs

Each LH79520 UART includes a 16-byte transmit FIFO and a 16-location receive FIFO. The receive FIFO is wider than the transmit FIFO because the receive FIFO includes 3 bits of status information concerning the received data. The FIFOs are set to a size of 1 at reset but can be enlarged to 16 by software control. The FIFO sizes must be changed together; the FIFOs cannot be separately resized.

Data intended for transmission is written to UARTDR:DATA. Each UART can be individually programmed to transmit 5, 6, 7 or all 8 bits of the data. Received data is read from UARTDR and includes 3 bits of status associated with the received data. A fourth bit of status is available in UARTDR, to indicate to software that an OVERRUN condition exists.

If the FIFOs are programmed to a size of 1, data written to UARTDR is placed in a transmit holding register which becomes the first-out location of the transmit FIFO when the FIFOs are programmed to a size of 16. If the FIFOs are programmed to a size of 16, data written to UARTDR is placed at the first-in location of the transmit FIFO and automatically removed from the first-out location of the FIFO as it is transmitted.

If the FIFOs are programmed to a size of 1, received data and status information is placed in a holding register which becomes the first-in location of the receive FIFO if the FIFOs are programmed to a size of 16.

If the FIFOs are programmed to a size of 16, received data and status information is placed at the first-in location of the receive FIFO and removed from the first-out location of the FIFO as it is read by software.

17.1.6 Bit Sequence Variations

UART peripherals send and receive data in identical formats, at a programmable baud rate. When a UART transmits data, the data is transmitted in a serial bit sequence which includes 1 START bit, the DATA, an optional PARITY bit, and 1 or 2 STOP bits. The width of the DATA can be programmed to be 5, 6, 7 or 8 bits. The PARITY is programmable as ODD, EVEN or NONE, and the UART can be programmed to send either 1 or 2 STOP bits. This programming also determines the bit sequence the UART expects to receive.

Received bit sequences which do not match a UART's programmed parameters generate errors indicated by flags which can be read with the received data.

17.1.7 Receiving Data

This section of this User's Guide describes the characteristics of the incoming data and the errors which the LH79520 UART peripherals can detect. This section assumes that the sending UART and the receiving UART are programmed to operate with the same parameters. Both ends of the serial communication link must be operating at the same baud rate, with the same quantity of data bits and the same type of parity.

Serial data is received by a UART in bit sequences described in Section 17.1.6. The structure and timing of the bit sequences will have predictable characteristics. The UART hardware utilizes the UART Clock signal and the UART's programmable parameters, such as the baud rate and the quantity of data bits, to establish the expected structure of the incoming data and the expected duration of each incoming bit of data. The UART is designed to determine whether or not the individual bits of incoming data have the correct logic levels for the correct durations of time.

17.1.7.1 BREAK Conditions

The bit sequences received by a UART may be separated by BREAK conditions imposed by the sender. A receiving UART declares that a BREAK condition exists when the UART's RxD (received data) pin is held LOW by the sender for a period of time which exceeds the time required receive a valid bit sequence. The way in which a UART is programmed determines the period of time which should be required to receive a valid bit sequence.

The incoming flow of information is controlled by the sender and is asynchronous to the operation of the LH79520. A BREAK condition can be imposed by the sender and may arise unexpectedly, at any time. Thus, BREAK conditions are categorized by the LH79520 UART peripherals as errors. The terms BREAK error and BREAK condition are used interchangeably in this Chapter of this User's Guide.

Each LH79520 UART can automatically sense BREAK errors and condition a flag which software can use to control the incoming flow of serial information.

17.1.7.2 FRAMING Errors

Each UART in the LH79520 automatically monitors incoming data to determine if the data is correctly framed. A UART is designed to receive incoming data in the bit sequence described in section Section 17.1.6. Correctly framed data will exhibit predictable characteristics. For example, if the incoming data is correctly framed, the UART will detect a START bit at the beginning of the bit sequence, and at least one STOP bit at the end of the bit sequence.

Each UART can automatically sense FRAMING errors and condition a flag which software can use to control the incoming flow of serial information.

17.1.7.3 PARITY Errors

Each UART in the LH79520 automatically monitors incoming data to determine if the data has the correct PARITY. Each UART can be programmed to send (and to expect to receive) EVEN parity, ODD parity or NO parity. A UART can detect a situation in which the PARITY of the received bit sequence does not match the UART's programmed PARITY.

Each UART can automatically sense PARITY errors and condition a flag which software can use to control the incoming flow of serial information.

17.1.7.4 OVERRUN Errors

Each UART can automatically detect an OVERRUN condition (error) and condition a flag which software can use to control the incoming flow of serial information.

An OVERRUN error will occur if the UART receives new incoming data and cannot store the new data to the receive FIFO because the FIFO is full. Each UART in the LH79520 automatically monitors its receiver to determine if an OVERRUN error has occurred.

When an OVERRUN condition is detected, the UART will immediately generate an OVERRUN error. If the OVERRUN interrupt is enabled, the UART will immediately assert a HIGH level to the LH79520 VIC. The data presently in the UART's receive FIFO will be protected and the data which generated the OVERRUN condition will be lost.

The UART indicates an OVERRUN error by setting the OE bit field to '1'. The OE bit field can be read by software in two different registers: UARTDR:OE and UARTRSR:OE. Although the OE bit field is available on a read of either register, the OE bit field is not stored in the receive FIFO.

17.1.8 Transmitting Data

Writing data to UARTDR:DATA initiates serial transmission of that data from the UART. The data is assembled in a specific bit sequence (described in Section 17.1.6), and automatically transmitted serially from the UART's TxD pin to a remote receiver. See Section 17.1.7 for additional information.

17.1.9 UART Clocks

Each LH79520 UART receives a scaled clock signal from the LH79520 RCPC functional block, as shown in Figure 17-1. The frequency of each UART Clock signal is configurable within the RCPC, and should be programmed to a frequency which is compatible with the additional scaling available in each UART peripheral, to produce the desired baud rate.

Figure 17-4 shows how the UART Clock signals are generated within the RCPC. The UART Clock signals can be generated by the 14.7456 MHz clock oscillator signal or by an external signal input at the CLKIN pin, to get custom baud rates. Each of the three UART Clock signals can be enabled or disabled by a bit field in the PeriphClkCtrl register.

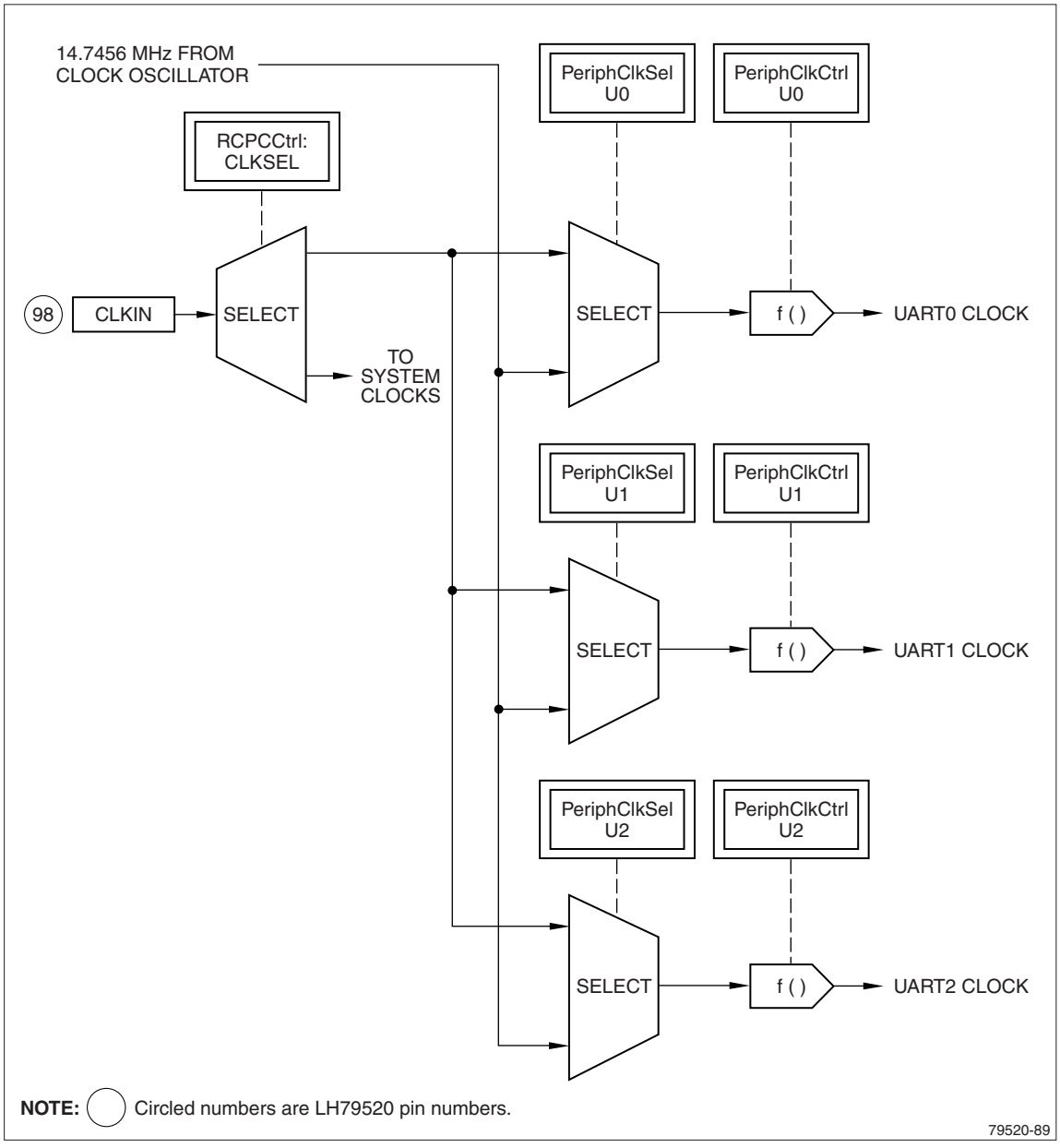


Figure 17-4. UART Clock Generation (RCPC)

For example, programming the UART Clock signals to a frequency of 3.6864 MHz, 7.3728 MHz, or 14.7456 MHz will allow the three LH79520 UART peripherals to produce standard baud rates. See Chapter 7 – Reset, Clock Generation, and Power Control (RCPC) for more information about the origins and programming (scaling) of the UART Clock signals.

Software utilizes programmable registers within each UART to further scale the UART Clock signals to produce a BAUD Clock which is utilized by the Baud Rate Generator within each UART peripheral. Figure 17-5 shows the registers that control the scaling of the UART Clock to produce a BAUD Clock for each UART peripheral.

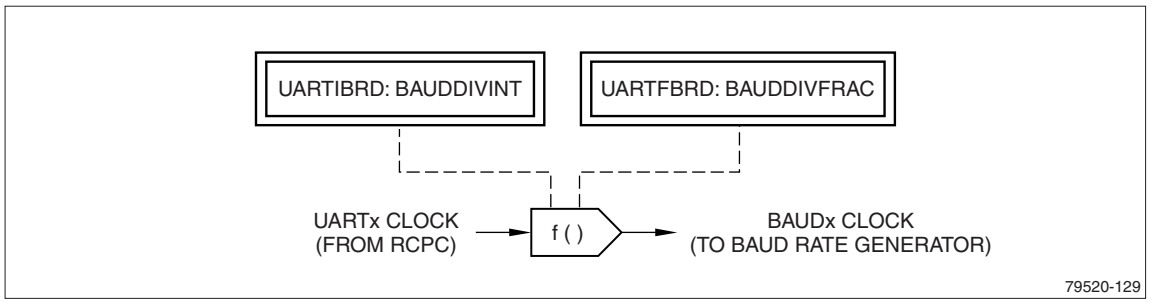


Figure 17-5. LH79520 UART Clock Scaling

17.1.10 UART Baud Rates

Each LH79520 receives a scaled clock signal input from the RCPC functional block. The input clock signal is further divided within the UART peripheral to produce the desired baud rate for that UART. Within each UART the input clock signal is divided by Baud Rate Divisor values programmed to the UARTIBRD and UARTFBRD registers.

The UARTIBRD register in each UART should be programmed with the integer (whole) part of the Baud Rate Divisor for that UART. The UARTFBRD register in each UART should be programmed with the fractional part of the Baud Rate Divisor for that UART.

17.1.10.1 Calculating the Baud Rate Divisor

1. Calculate the total value of the Baud Rate Divisor:
   Baud Rate Divisor = (fUARTCLK / (16 × Desired Baud Rate)),
   where fUARTCLK is the frequency of the UART Clock signal generated in the LH79520 RCPC.
2. Consider the integer and fractional parts of the total value separately.
3. Program UARTIBRD:BAUDDIVINT with the integer part of the total value.
4. Use the fractional part of the total value to calculate an integer, m:
   m = integer ((fractional_part x 64) + 0.5)
5. Program UARTFBRD:BAUDDIVFRAC with the integer, m.

17.1.10.2 Example Baud Rate Divisor Calculation

This example shows the calculations required to achieve 230,400 bps with a 4.0 MHz UART Clock. The 4.0 MHz signal in this example is assumed to be provided by an external clock source because the LH79520 internal clock oscillator cannot operate with a 4.0 MHz crystal:

1. The Baud Rate Divisor = $(4 \times 10^6) \div (16 \times 230{,}400) = 1.085$
2. The whole part is 1 and the fractional part is 0.085
3. Program UARTIBRD:BAUDDIVINT = 1
4. m = integer $((0.085 \times 64) + 0.5) = 5$
5. Program UARTFBRD:BAUDDIVFRAC = 5

In this example, the actual baud rate division will be:

$1 + 5/64 = 1.078$

Therefore, the baud rate actually generated will be:

$(4 \times 10^6) \div (16 \times 1.078) = 231{,}911$ bps

with an error of:

$(231{,}911 - 230{,}400) \div 230{,}400 \times 100 = 0.656\%$

The maximum error for the UARTFBRD:BAUDDIVFRAC register is

$1/64 \times 100 = 1.56\%$.

This maximum error will occur when m = 1. The error is cumulative over 64 clock ticks. One or the other Baud Rate Divisor values should always be programmed to be non-zero. Programming both values to zero will result in no transmission or reception of serial data.

17.1.10.3 Baud Rate Examples, UART Clock = 14.7456 MHz.

Table 17-2 lists the integer and fractional divisors required to obtain various UART bit rates with a UART input clock (from the RCPC) of 14.7456 MHz. In these cases, the value written to UARTFBRD:BAUDDIVFRAC (the BAUDDIVFRAC bit field in the UARTFBRD register) is zero.

Table 17-2. Typical Baud Rates and Divisors

UARTIBRD: BAUDDIVINT (INTEGER DIVISOR)	UARTFBRD: BAUDDIVFRAC (FRACTIONAL DIVISOR)	GENERATED BIT RATE (BPS)
0x0002	0x000	460,800
0x0004	0x000	230,400
0x0008	0x000	115,200
0x000C	0x000	76,800
0x0010	0x000	57,600
0x0018	0x000	38,400
0x0030	0x000	19,200
0x0040	0x000	14,400
0x0060	0x000	9,600
0x0180	0x000	2,400
0x0300	0x000	1,200
0x20BA	0x000	110

17.1.10.4 Baud Rate Examples, UART Clock = 4.0 MHz.

Table 17-3 lists the integer and fractional divisors required to obtain various UART bit rates with a UART input clock (from the RCPC) of 4.0 MHz. In these cases, the value written to UARTFBRD:BAUDDIVFRAC is non-zero.

Table 17-3. Typical Baud Rates, Integer and Fractional Divisors

UARTIBRD: BAUDDIVIN (INTEGER DIVISOR)	UARTFBRD: BAUDDIVFRAC (FRACTIONAL DIVISOR)	DESIRED BIT RATE (BPS)	GENERATED BIT RATE (BPS)	ERROR (%)
0x0001	0x005	230,400	231,911	0.656
0x0002	0x00B	115,200	115,101	0.086
0x0003	0x010	76,800	76,923	0.160
0x0006	0x021	38,400	38,369	0.081
0x0011	0x017	14,400	14,401	0.007
0x0068	0x00B	2,400	2,400	~0000
0x08E0	0x02F	110	110	~0000

17.1.11 UART Interrupt Overview

Each of the three LH79520 UART peripherals can generate maskable interrupts which are routed to the LH79520 Vectored Interrupt Controller (VIC). All UART interrupts are disabled at reset and must be individually enabled by software. Each UART interrupt must be enabled in the UART peripheral and also in the VIC. Figure 17-6 diagrams the structure of the UART interrupts and the associated UART registers.

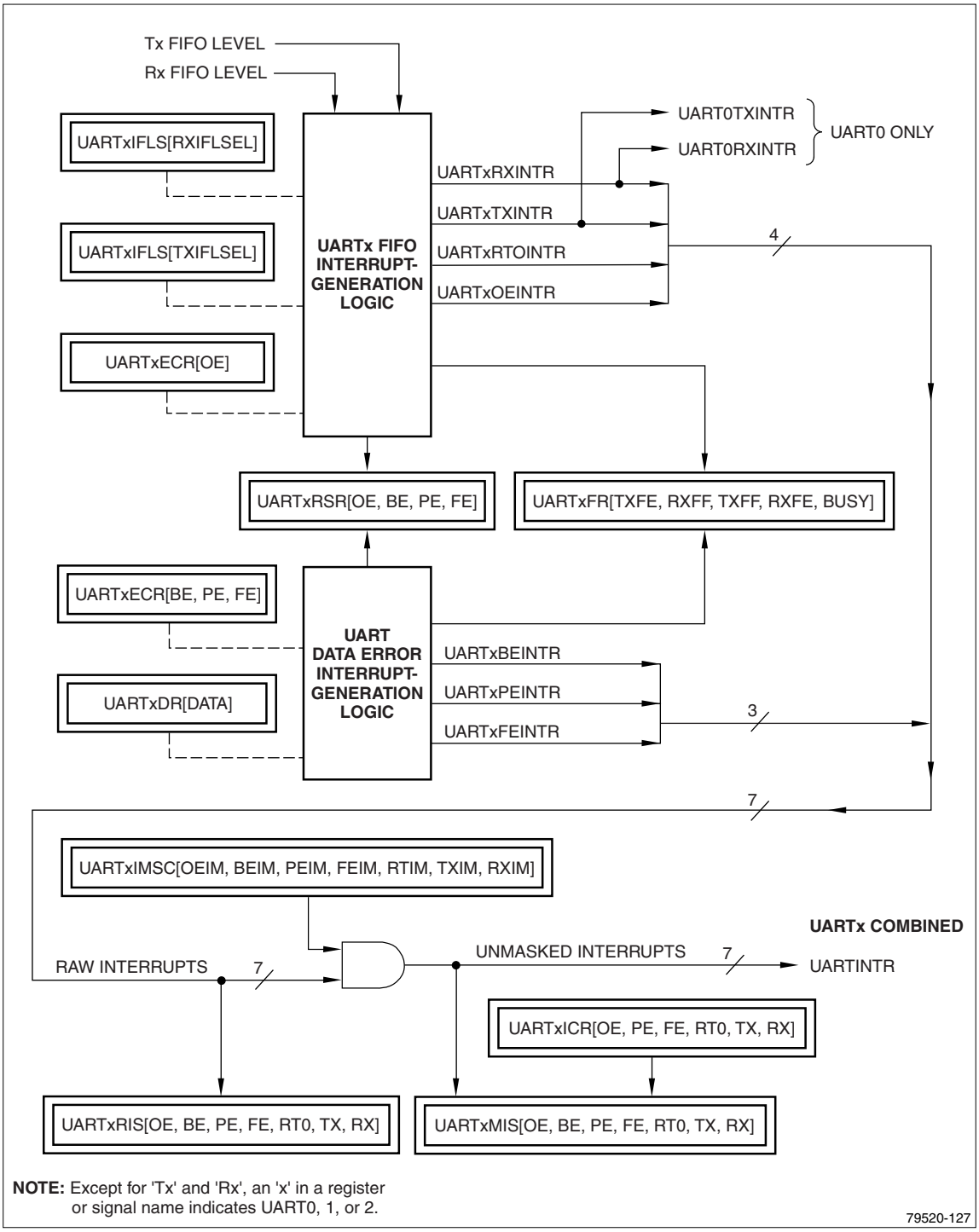


Figure 17-6. UART Interrupts

If a UART has been correctly initialized, operating the UART involves servicing the FIFOs in the UART and handling errors which may arise during operation. Figure 17-6 shows the LH79520 UART interrupts grouped in this manner. The UART FIFO Interrupt Generation Logic block can issue four of the seven possible UART interrupts. The other three UART interrupts are generated by the UART Data Error Interrupt Generation Logic block. After masking in the UARTMIS register, these seven sources of UART interrupt are logically OR'd to form one combined interrupt output, UARTINTR, which is routed to the LH79520 VIC. Two of the UART0 FIFO interrupts, UART0TXINTR and UART0RXINTR, although included in the UART0 combined interrupt, are also routed directly to the LH79520 VIC, to facilitate low-latency FIFO service for UART0.

Software can selectively enable or disable each of the individual interrupts by writing the appropriate mask bits in a UART's UARTIMSC register. Writing the appropriate mask bit to '1' enables the interrupt. Software can read either of two registers to establish the status of each individual interrupt source. The status of the raw interrupts can be read from the UARTRIS register and the status of the masked interrupts can be read from UARTMIS register. The seven individual interrupts are logically OR'd within each UART to form one combined interrupt which is routed from that UART to the LH79520 VIC, as shown in Figure 17-6.

The UART0 peripheral is a special case because two of the seven sources of interrupt included in its combined interrupt are also routed directly to the VIC.

17.1.11.1 FIFO-Related Interrupts

The FIFO-related interrupt group concerns the quantity of data in the UART's transmit and receive FIFOs. Software can utilize these interrupts to transmit data when space is available in the transmit FIFO, and to receive data when data arrives in the receive FIFO.

FIFO-related interrupts can be generated by FIFO levels (i.e.: programmable watermarks), received data OVERRUN errors, or receive time-outs. These interrupts are generated in the UART FIFO Interrupt Generation Logic and unmasked (enabled) by bit fields in the UARTIMSC register. The raw status of each FIFO interrupt is indicated by a flag in the UARTRIS register and a masked status flag for each FIFO interrupt is included in the UARTMIS register. When an enabled FIFO interrupt occurs, the interrupt service routine (ISR) should clear the status flag in the UARTMIS register by a write to the UARTICR register. Although the operation of the FIFOs is programmable, software can access the data in the FIFOs only via the UARTDR register. Software writes outgoing (i.e.: transmitted) data to UARTDR:DATA and software reads incoming (i.e.: received) data from UARTDR:DATA. A read of the UARTDR register also returns status flags associated with the data in UARTDR.

17.1.11.2 Data Error Interrupts

The Data Error interrupt group concerns errors encountered by the UART while transmitting or receiving data. Software can utilize these interrupts to correct data errors which the UART detects during operation.

Figure 17-6 shows that the LH79520 UART peripherals can generate interrupts when data errors occur. Data error interrupts include BREAK errors, FRAMING errors, and PARITY errors.

17.1.12 UART Interrupt Descriptions

Each of the three UART peripherals in the LH79520 is capable of generating seven different, individually-maskable interrupts. This section lists and describes each of the seven sources of interrupt which can be generated by a UART peripheral.

17.1.12.1 UARTRXINTR

The UART Receive interrupt is asserted by the UART when one of the following events occurs:

- If the FIFOs are programmed to a size of 16, and the quantity of data in the receive FIFO exceeds the programmed trigger level, then the receive interrupt is asserted. The receive interrupt can be cleared by repeatedly reading data from UARTDR until the quantity of data in the receive FIFO is below the programmed trigger level. The receive interrupt can also be cleared by writing a '1' to UARTICR:RXIC or by disabling the interrupt in the UARTIMSC register.
- If the FIFOs are programmed to a size of 1, and data is received to the FIFO, the FIFO is considered to be full. When a FIFO is full, the UART asserts the receive interrupt. The receive interrupt can be cleared in the same manner as if the FIFOs were programmed to a size of 16.

17.1.12.2 UARTTXINTR

UARTTXINTR is the UART Transmit interrupt. In the case of UART0, this interrupt is included in the combined interrupt and also routed directly to the LH79520 VIC. For UART1 and UART2, this interrupt is included in the combined interrupt but not routed directly to the VIC.

Data written to the UARTDR is automatically placed in the Transmit FIFO. The conditions under which the UART Transmit interrupt will be asserted depend upon the size of the Transmit FIFO (1 or 16 bytes) and the movement of data through the Transmit FIFO. The Transmit interrupt is not asserted when the UART and the interrupt are first enabled, but before any data is written to the UARTDR register. The Transmit interrupt will only be asserted when the UART is enabled, and the Transmit interrupt is enabled, and data has entered, and then exited, the Transmit FIFO.

When the FIFOs are programmed to a size of 1, the FIFO's programmed watermark is effectively ignored. The Transmit interrupt will be asserted when the FIFO becomes empty because data which was in the FIFO has been transmitted. Software can clear the Transmit interrupt by performing a single write to UARTDR, which makes the FIFO full. Alternatively, software can clear the Transmit interrupt by writing a '1' to UARTICR:TXIC or by disabling the interrupt in the UARTIMSC register.

When the FIFOs are programmed to a size of 16, the programmed watermark determines when the Transmit interrupt will be asserted. The Transmit interrupt will be asserted when the amount of data in the FIFO first exceeds the watermark, then falls to below the watermark (because the data which was in the FIFO has been transmitted). Software can clear the Transmit interrupt by writing data to UARTDR until the FIFO is again filled to above the FIFO's programmed watermark. Alternatively, software can clear the transmit interrupt by writing a '1' to UARTICR:TXIC or by disabling the interrupt in the UARTIMSC register.

17.1.12.3 UARTRTINTR

The UART Receive timeout interrupt is asserted when the receive FIFO is not empty, and no further data is received for a duration of time equal to the time required by the UART's present programming, to send or receive 32 bits. The receive timeout interrupt status flag will be cleared to '0' either when the FIFO becomes empty (by repeated reads of the data), or when a '1' is written to the corresponding bit of the UARTICR register.

17.1.12.4 UARTEINTR

The error interrupt is asserted immediately when an error occurs in the reception of data by the UART. The interrupt can be caused by a number of different error conditions:

- FRAMING error
- PARITY error
- BREAK error
- OVERRUN error

Although the general reason for this interrupt can be determined by reading the UARTRIS or UARTMIS registers, software may need to retrieve data from the FIFO to make a final determination.

FRAMING, BREAK and PARITY errors concern data which has been received to the FIFO. If a FRAMING, PARITY or BREAK error occurs during reception, the appropriate error bit is stored in the FIFO, with the affected data. Software must examine the data in the FIFO to determine which data was affected. The affected data will be the newest data in the FIFO.

OVERRUN conditions (errors) concern incoming data which cannot be saved to the FIFO because the FIFO is full. The OVERRUN error status is not stored in the FIFO, with the data, because there is no room in the FIFO for the data. Instead, the data presently in the FIFO is protected (not overwritten), the data which caused the OVERRUN error condition is lost, and the OVERRUN error is reported in both the UARTDR:OE and UARTRSR:OE bit fields. Software must make room in the FIFO by reading data from the FIFO.

The cause of this interrupt can be cleared by writing to the relevant bit field of the UARTICR register.

17.1.12.5 UARTINTR

The interrupts generated by each UART are combined into one interrupt output for that UART. The combined output is the logical OR of the individually masked sources of interrupts for that UART. The UART asserts the combined interrupt if any of the individual interrupts are unmasked and asserted.

The combined output is routed to the LH79520 VIC, where it can be masked on an individual peripheral basis.

17.1.13 UART Interrupt Outputs to the VIC

Together, the three UART peripherals generate five interrupt signals to the VIC, as shown in Table 17-4. Each of the interrupts input on VIC channels 23, 24, and 25 is the combined output of all interrupt sources for one UART. Software utilizing these interrupts must examine the UARTMIS register in the appropriate UART in order to determine the cause of the interrupt. The interrupts assigned to VIC channels 21 and 22 concern only UART0. These interrupts are diagrammed in Figure 17-6 and discussed in Section 17.1.

Table 17-4. UART Interrupt Outputs

UART INTERRUPT SOURCE	VIC INTERRUPT CHANNEL	INTERRUPT NAME	INTERRUPT DESCRIPTION
UART0 ONLY	21	UART0RXINTR	UART0 Rx FIFO Watermark
UART0 ONLY	22	UART0TXINTR	UART0 Tx FIFO Watermark
UART0	23	UART0INTR	UART0 Combined Interrupt
UART1	24	UART1INTR	UART1 Combined Interrupt
UART2	25	UART2INTR	UART2 Combined Interrupt

17.1.14 Loopback Mode

Each LH79520 UART can be operated in a loopback mode to permit testing of the serial data transmission and reception entirely within the UART. This loopback mode does not require external interconnection of the LH79520 UART pins.

When a UART is operating in the loopback mode, the UART's transmit data path (the output) is connected directly to the UART's receive data path (the input). Connecting the UART peripheral's output to its input causes the UART to immediately receive the same serial data the UART is transmitting.

The loopback mode for each LH79520 UART is disabled at reset but software can utilize UARTCR:LBE (the LBE bit field in each UART's UARTCR register) to place any LH79520 UART in a loopback mode.

When a UART is operated in the loopback mode, the data transmitted by the UART is also available at the UART's Tx pin.

Operating UART0 in the loopback mode requires additional considerations because UART0 includes a SIR (Serial InfraRed) ENDEC to facilitate IrDA-compatible serial communications. If the UART0 SIR is disabled, then the loopback mode for UART0 functions like the loopback mode for UART1 or UART2. If the UART0 SIR is enabled, then additional programming is required.

In normal use, the SIR ENDEC included with UART0 operates in a half-duplex mode. The SIR ENDEC is a half-duplex device and therefore cannot under normal circumstances receive while it is transmitting. If the UART0 SIR is to be tested, software must condition UARTTCR:SIRFDME, to place the SIR in a full-duplex mode. When the UART0 SIR is enabled, and the SIR is placed in the full-duplex mode, and UART0 is operated in the loop-back mode, the output of the SIR is inverted before being routed back to the input of the SIR, to permit loopback testing. The SIR full-duplex mode is provided for testing only.

17.2 UART Programming Guidelines

This section of this User's Guide discusses several important issues concerning the programming and use of the UART peripherals in the LH79520. This section lists, for example, UART operations which should be accomplished by software in a specific sequence.

17.2.1 Disable a UART while Reprogramming

Always disable a UART before reprogramming any of its control registers. Enable the UART after its control registers have been reprogrammed.

17.2.2 Write the UARTLCR_H Register Last

The operation of any UART is based on the contents of three programmable registers associated with that UART:

- UARTIBRD
- UARTFBRD
- UARTLCR_H.

Although software may write new values to these three registers, those new values will not be utilized by the UART until a write to the UART's UARTLCR_H register is completed. The sequence in which the UARTIBRD and UARTFBRD registers are written is not important but software should always perform a write to the UARTLCR_H register after writing to either (or both) of these registers.

17.2.3 Enabling a UART

To enable a data transmit operation by a particular UART, both UARTCR:UARTEN and UARTCR:TXE for that UART must be programmed to '1'. The sequence in which these two bit fields are written to '1' is not important.

To enable a data receive operation by a particular UART, both UARTCR:UARTEN and UARTCR:RXE for that UART must be programmed to '1'. The sequence in which these two bit fields are written to '1' is not important.

The sequence in which the transmit and receive operations are enabled is not important.

17.2.4 Disabling a UART

When a UART is disabled while it is transmitting a character, the UART will complete the transmit operation before becoming disabled. When a UART is disabled while receiving a character, the UART will complete the receive operation before becoming disabled.

17.2.5 Reading Received Data and Associated Status Flags

As a UART receives serial data, the UART also monitors the serial communications link for FRAMING errors, BREAK errors, PARITY errors, and OVERRUN errors. When an error occurs, the error is indicated by a status flag in the UARTDR register. The same flag is also available in the UARTRSR register.

If software elects to utilize the flags in UARTRSR, the software should always read UARTDR and then immediately read UARTRSR. This read sequence cannot be reversed because UARTRSR is updated only when UARTDR is accessed by a read operation.

17.3 UART Programmer's Model

The Register Base Addresses for the three UART peripherals in the LH79520 are:

UART0Base: 0xFFFC0000

UART1Base: 0xFFFC1000

UART2Base: 0xFFFC2000

All registers in the UARTs must be accessed as a full word. The UARTs do not support byte and half-word writes. Programmers should ensure that the system clock is enabled before programming any registers.

17.4 UART Register Summary

The programmable registers for each LH79520 UART peripheral are summarized in Table 17-5 and explained in more detail on the pages that follow the table.

Table 17-5. UART Register Summary

ADDRESS	REGISTER NAME	DESCRIPTION
UARTxBase + 0x000	UARTDR	Data read from or written to the UART.
UARTxBase + 0x004	UARTRSR/UARTECR	Receive Status Register (when Read). Error Clear Register (when Written).
UARTxBase + 0x008 - UARTxBase + 0x017	///	Reserved. Do not write.
UARTxBase + 0x018	UARTFR	Flag Register (read only)
UARTxBase + 0x01C	///	Reserved. Do not write
UARTxBase + 0x020	UARTILPR*	IrDA Low-Power Counter
UARTxBase + 0x024	UARTIBRD	Integer Baud Rate Divisor Register
UARTxBase + 0x028	UARTFBRD	Fractional Baud Rate Divisor Register
UARTxBase + 0x02C	UARTLCR_H	Line Control Register, HIGH byte
UARTxBase + 0x030	UARTCR	UART Control Register
UARTxBase + 0x034	UARTIFLS	Interrupt FIFO Level-Select
UARTxBase + 0x038	UARTIMSC	Interrupt Mask Set/Clear
UARTxBase + 0x03C	UARTRIS	Raw Interrupt Status Register
UARTxBase + 0x040	UARTMIS	Masked Interrupt Status Register
UARTxBase + 0x044	UARTICR	Interrupt Clear Register
UARTxBase + 0x048 - UARTxBase + 0x07F	///	Reserved. Do not write
UARTBase + 0x080	UARTTCR*	SIR Test Control Register
UARTxBase + 0x084 - UARTxBase + 0xFFF	///	Reserved. Do not write.

NOTE: *These registers concern the UART0 IrDA (SIR) functionality. The corresponding registers at UART1Base and UART2 Base should be considered 'Reserved — do not write'. See Chapter 18 – IrDA Communications (SIR) for more information about programming the SIR.

17.5 UART Register Descriptions

This section of this User's Guide lists and describes each of the registers that configure the three LH79520 UART peripherals.

A UART peripheral must be disabled before any of its control registers are reprogrammed.

UART0 in the LH79520 includes a Serial InfraRed (SIR) ENDEC which can be utilized to provide SIR functionality. The UART1 and UART2 peripherals in the LH79520 do not include this SIR capability. Registers located at the UART1Base and UART2Base addresses, and which concern SIR functionality, should be considered 'Reserved — write the reset value'. See Chapter 18 – IrDA Communications (SIR) for additional information regarding the UART0 SIR function.

UART0 in the LH79520 provides two additional interrupt outputs not implemented for either UART1 or UART2 in the LH79520. Although these two interrupt sources are included in the combined interrupt output of all three UART peripherals, for UART0 the two sources are also routed directly to the LH79520 VIC. For UART1 and UART2 these interrupts are included in each UART's combined interrupt output but not routed directly to the VIC.

The register addresses in this section of this User's Guide include an 'x', where x = 0, 1 or 2, in reference to UART0, UART1 or UART2, respectively.

17.5.1 UART Data Register

The UARTDR register for each UART has a dual use: data to be transmitted by a UART is written to UARTDR and data received by a UART is read from UARTDR.

The UARTDR register is a read-write register, except for the OE, BE, PE, and FE flags, which are read-only. These flags are associated with received data, and are conditioned by the UART as the data is received. Reads of UARTDR return the received data, and also return the status flags associated with that particular data.

Table 17-6. UARTDR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///				OE	BE	PE	FE	DATA							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	UARTxBase + 0x000															

17.5.1.1 Transmitting Data

Software does not have direct access to the transmit FIFO. Software access to the transmit FIFO is via the UARTDR register. If the UART FIFOs are programmed to a size of 16, data written to UARTDR:DATA is pushed onto the first-in location of the transmit FIFO. If the UART FIFOs are programmed to a size of 1, data written to UARTDR:DATA is placed directly into the UART transmitter's holding register, which is the transmit FIFO's single location.

Writing data to UARTDR:DATA initiates serial transmission of that data from the UART. The PARITY of the data-to-be-transmitted is determined by the way in which the UARTLCR_H:SPS, UARTLCR_H:PEN, and UARTLCR:PES bit fields are programmed. The width of the data-to-be-transmitted is determined by UARTLCR_H:LEN, and is aligned with the least-significant bit of UARTDR_DATA. When the UART is composing the DATA portion of the bit sequence to be transmitted, the UART begins with the least-significant bit of the value in UARTDR:DATA and transmits the quantity of DATA bits determined by the value programmed to UARTLCR_H:LEN.

For example, if the data width is programmed to 6 bits (i.e.: UARTLCR_H:LEN = 0b01), and 0b01100000 is written to UARTDR:DATA, then the UART will transmit a DATA value of 0b100000.

When the UART is composing the bit sequence to be transmitted, the UART automatically prefixes the DATA field with a START bit, appends the appropriate PARITY bit (if PARITY is enabled) to the end of the DATA field, and appends the programmed quantity of STOP bits following the PARITY bit. The resulting bit sequence (START bit, DATA bits, PARITY bit, STOP bits) is transmitted from the UART's TxD pin in serial fashion, beginning with the START bit. See Section 17.5.7 for more information about establishing the UART peripheral's transmit parameters such as DATA width and PARITY.

17.5.1.2 Receiving Data

Software does not have direct access to the receive FIFO. Software access to received data located in the receive FIFO is via reads of the UARTDR register.

If the FIFOs are programmed to a size of 1, then the receive FIFO is 1 location deep. If the FIFOs are programmed to a size of 16, then the receive FIFO is 16 locations deep.

If the FIFOs are programmed to a size of 16, and data is received, the data and 3 bits of status (BREAK, FRAMING, and PARITY) are pushed onto the 11-bit wide receive FIFO. If the FIFOs are programmed to a size of 1, and data is received, the data and 3 bits of status are stored in the single location which is the receive FIFO.

Reads of the UARTDR return the data, the three bits of status discussed above, and a fourth bit of status, the OVERRUN status bit. When OVERRUN errors occur, the received data is lost, and so cannot be placed in the receive FIFO. Instead, the present contents of the receive FIFO are protected (not overwritten), and the OVERRUN status bit is set to '1'.

When the received data is read from UARTDR:DATA, the received data is automatically aligned to the least-significant bit position within the UARTDR:DATA bit field, and the bits above the received data are read as '0'.

For example, if the UART is programmed to transmit and receive five bits of data, and five bits of data, such as 0b11100, are correctly received, then a read of UARTDR:DATA will return 0b00011100.

A read of UARTDR can provide the data and also the status information associated with that data. Alternatively, the status information associated with the data can be obtained by a subsequent read of the UARTRSR register. If Programmers elect to obtain the status information from the UARTRSR register, then each read of UARTDR should be followed immediately by a read of UARTRSR, to ensure that the status information is associated with the appropriate data. This sequence cannot be reversed.

If error interrupts are enabled and an error in the received data is detected, the error interrupt will be generated immediately. Software called to handle the error interrupt must consider the type of error and also the depth of the FIFO. If the error is an OVERRUN error, then the data which actually caused the error is lost, but the FIFO should be serviced by reading UARTDR. If the error is a FRAMING, BREAK, or PARITY error, then the data which caused the error will be the newest data in the receive FIFO. Software handling the error interrupt will need to perform repeated reads of the UARTDR (or the UARTDR/ UARTRSR sequence outlined above), in order to access the status information associated with the data which caused the error interrupt.

Table 17-7 describes the bit fields in the UARTDR register.

Table 17-7. UARTDR Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:12	///	**Reserved** Write the reset value.
11	OE	**OVERRUN Error** This bit is set by the UART to '1' if the UART receives a byte which cannot be placed in the receive FIFO because the receive FIFO is full. OE = 1 is an error condition. This bit is cleared to '0' if the UART receives a byte which can be placed in the receive FIFO because the FIFO has space available for the byte. OE = 0 is the normal (desirable) operating condition. Reset = 0
10	BE	**BREAK Error** This bit is set to '1' when the UART detects a BREAK condition. This bit is cleared to '0' by a write to UARTECR:BE. A BREAK condition indicates that the RxD input pin for this UART was held LOW for longer than one complete byte-reception time. The byte-reception time is defined as the period of time (at the present baud rate) required for the UART to receive the START, DATA, PARITY, and STOP bits. When a BREAK condition is detected, the UART sets this bit to '1', and places a NULL byte (0b0) into the UART's receive register. If the FIFO is programmed to a size of 16, the NULL byte is instead placed in the first-in location of the FIFO. The BREAK error is considered to be associated with this NULL byte. Only one NULL byte (0b0) is loaded into the FIFO for each instance of a BREAK condition detected by the UART. When a BREAK condition has been detected, the UART will make no attempt to receive subsequent bytes until the BREAK condition is cleared. The BREAK will be considered to have been cleared only after the RxD input for this UART has gone a '1' (marking state), and the next valid START bit has been received. Reset = 0.
9	PE*	**PARITY Error** The UART sets this bit field to '1' to indicate that the PARITY of the received data byte does not match the PARITY selections programmed to UARTLCR_H:EPS and UARTLCR_H:SPS. Reset = 0.
8	FE*	**FRAMING Error** The UART sets this bit field to '1' to indicate that the received byte lacked a valid STOP bit. A valid STOP bit = 1. Reset = 0.
7:0	DATA	**Data** Write a byte-to-be-transmitted to this bit field. Reads of this bit field return a received data byte. If the UART's FIFOs are programmed to a size of 1, the returned byte will be the contents of the UART's receive register. If the UART's FIFOs are programmed to a size of 16, the returned byte will be the oldest byte from the receive FIFO. In this case, the returned byte will have been removed from the UART's receive FIFO. Reset = 0.

NOTE: *When the FIFOs are programmed to a size of 1, this error is associated with the data byte located in the UART's receive register. When the FIFOs are programmed to a size of 16, this error is associated with the newest data in the FIFO.

17.5.2 UART Receive Status Register

Reads of the UARTRSR register return the status flags associated with the data just read from the UARTDR register.

The UARTRSR register should be read immediately after a read of the UARTDR register. The status information returned on a read of the UARTRSR register includes FRAMING, BREAK, and PARITY error flags associated with the data just read from the UARTDR register. The status information returned on a read of the UARTRSR register also includes an OVERRUN error flag, which will be '1' to signal that incoming data could not be placed in the FIFO because the FIFO is full.

The status information returned on a read of the UARTRSR register is available in two different locations: the UARTDR register and the UARTRSR register. The status information returned on a read of the UARTRSR register is a duplicate of the status information returned, along with the received data, on a read of the UARTDR register.

If Programmers elect to obtain this status information from the UARTRSR register instead of from the UARTDR register, each read of the UARTDR register should be followed immediately by a read of the UARTRSR register, to ensure that the status information obtained is associated with the appropriate data.

The UARTRSR register is a read-only register. All bits of the UARTRSR register are cleared to '0' at reset.

See Section 17.5.3, UARTECR, for information regarding clearing the bit fields in the UARTRSR register to '0'.

The UARTRSR and UARTECR registers are mapped to the same memory address but function differently on read accesses than on write accesses.

Table 17-8. UARTRSR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///												OE	BE	PE	FE
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
ADDR	UARTxBase + 0x004															

Table 17-9 describes the bit fields in the UARTRSR register.

Table 17-9. UARTRSR Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:4	///	**Reserved** Reads as 0.
3	OE	**OVERRUN Error** The OE bit field concerns new data in the incoming (Rx) shift register; this bit does not concern data already transferred from the shift register to the receive FIFO. The OE bit field is set to '1' if a new data character arrives in the incoming shift register but the receive FIFO is full. The OE bit field set to '1' because the new data in the incoming shift register cannot be transferred from the incoming shift register to the receive FIFO, because the receive FIFO is full. The OE bit field must be cleared to '0' by a write to the UARTECR register. When the receive FIFO is full, and new data arrives in the incoming shift register, only the previous contents of the shift register are overwritten. The contents of the receive FIFO remain valid when an OVERRUN error occurs because the receive FIFO will not accept additional data when the receive FIFO is full. The receive FIFO does not automatically discard data. When an OVERRUN error occurs, the present contents of the receive shift register are lost and are not recoverable. To recover from an OVERRUN error, software must read data from the receive FIFO and write to UARTECR. The software must read data from the receive FIFO in order to make space available in the receive FIFO for new data. The software must write to the UARTECR register in order to clear UARTECR:OE to '0'.
2	BE	**BREAK Error** The BE bit field is set to '1' if the UART detects a BREAK error condition. A BREAK condition indicates that the Rx input line was held LOW by the sender (external to the LH79520) for a period of time longer than the time required to receive one complete character according to the UART's present programming. The time required to receive one complete character includes the time required to receive a start bit, the data bits, a PARITY bit, and the STOP bit(s). The BE bit field must be cleared to '0' by a write to the UARTECR register. If the UART's FIFOs are programmed to a size of 16, the BREAK error is associated with the character at the first-in location of the receive FIFO. When a BREAK error is detected by the UART, only one '0' character is loaded to the receive FIFO. Characters subsequently received on the UART's Rx input line will be ignored until the BREAK condition is cleared. The BREAK condition is considered to be cleared only when the UART's Rx input line has transitioned from a LOW to a HIGH (from the SPACING to the MARKING state), and the next valid start bit has been received by the UART.
1	PE*	**PARITY Error** The PE bit field is set to '1' if the UART determines that the PARITY of the received data character does not match the PARITY specified by UARTLCR_H:PEN. The PE bit field must be cleared to '0' by a write to UARTECR. If the UART's FIFOs are programmed to a size of 16, the PARITY error is associated with the character at the first-in location of the FIFO.
0	FE*	**FRAMING Error** The FE bit field is set to '1' by the UART to indicate that the received data character did not have a valid STOP bit. A valid STOP bit = 1. The FE bit must be cleared to '0' by a write to UARTECR. If the UART's FIFOs are programmed to a size of 16, a FRAMING error is associated with the character at the first-in location of the FIFO.

NOTE: *When the FIFOs are programmed to a size of 1, this error is associated with the data byte located in the UART's receive holding register. When the FIFOs are programmed to a size of 16, this error is associated with the newest data in the FIFO.

17.5.3 UART Receive Error Clear Register

Writes to the UARTECR register clear the error status flags in the UARTRSR register.

Table 17-10. UARTECR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET																
TYPE	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///												OE	BE	PE	FE
RESET																
TYPE	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
ADDR	UARTxBase + 0x004															

The UARTRSR and UARTECR registers are mapped to the same memory address but function differently on read accesses than on write accesses.

The UARTECR register is a write-only register. Reads of the UARTECR register return the value in the UARTRSR register.

Software can use the UARTECR register to clear the error flags in the UARTRSR register. One write to the UARTECR register clears the entire UARTRSR register. A single write of any value to the UARTECR register will clear to '0' all non-zero bits in the UARTRSR register. A single write to the UARTECR register will clear all FRAMING, BREAK, PARITY, and OVERRUN errors presently flagged in the UARTRSR register. The value of the data written to the UARTECR register is not important, and will not be retained.

The bit fields in the UARTECR register are organized in the same manner as the bit fields in the UARTRRSR register.

Section 17.5.2 for descriptions of the bit fields in the UARTRSR register. Table 17-9 in that section describes the bit fields in the UARTECR register.

17.5.4 UART Flag Register

The UARTFR register contains status flags associated with UART and FIFO operation. UARTFR:TXFF, UARTFR:RXFF, and UARTFR:BUSY are reset to '0'. UARTFR:TXFE and RXFE are reset to '1'. Table 17-12 describes the bit fields in the UARTFR register.

Table 17-11. UARTFR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								TXFE	RXFF	TXFF	RXFE	BUSY	///		
RESET	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
ADDR	UARTxBase + 0x018															

Table 17-12. UARTFR Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:8	///	**Reserved** Read as 0.
7	TXFE	**Transmit FIFO Empty** The meaning of this bit depends on the state of the FEN bit in the register. If the FIFO is programmed to a size of 1, this bit is set when the transmit holding register is empty. If the FIFO is programmed to a size of 16, the TXFE bit is set when the transmit FIFO is empty. Reset = 1.
6	RXFF	**Receive FIFO Full** The meaning of this bit depends on the state of the FEN bit in the UARTLCR_H register. If the FIFO is programmed to a size of 1, this bit is set when the receive holding register is full. If the FIFO is programmed to a size of 16, the RXFF bit is set when the receive FIFO is full. Reset = 0.
5	TXFF	**Transmit FIFO Full** The meaning of this bit depends on the state of the FEN bit in the UARTLCR_H register. If the FIFO is programmed to a size of 1, this bit is set to '1' when the transmit holding register is full. If the FIFO is programmed to a size of 16, this bit is set to '1' when the transmit FIFO is full. Reset = 0.
4	RXFE	**Receive FIFO Empty** The meaning of this bit depends on the state of the FEN bit in the UARTLCR_H register. If the FIFO is programmed to a size of 1, this bit is set to '1' when the receive holding register (the first-out location of the FIFO) is empty. If the FIFO is programmed to a size of 16, the RXFE bit is set to '1' when the receive FIFO is empty. Reset = 1.
3	BUSY	**UART Busy** The UART sets this bit to '1' when the UART is busy transmitting data. The UART sets this bit to '1' when the transmit FIFO becomes non-empty, even if the UART is not enabled. This bit will remain set to '1' until the complete bit sequence, including the byte being transmitted and the STOP bits, has been shifted from the shift register to the TxD pin. Reset = 0.
2:0	///	**Reserved** Read as 0.

17.5.5 UART Integer Baud Rate Divisor Register

The UARTIBRD register for each UART must be programmed with the integer part of the desired Baud Rate Divisor for that UART.

The baud rate for each UART is achieved by dividing the UART's Clock input signal by a Baud Rate Divisor. Parts of the Baud Rate Divisor must be written to the UART's UARTIBRD and UARTFBRD registers. The integer part of the Baud Rate Divisor is programmed to the UARTIBRD register. The fractional part of the Baud Rate Divisor is programmed to the UARTFBRD register.

UARTIBRD:BAUDDIVINT is reset to '0'. Table 17-14 describes the bit fields in the UARTIBRD register.

The UARTIBRD:BAUDDIVINT and UARTFBRD:BAUDDIVFRAC bit fields are both reset to '0'. The UART will not produce a bit clock when both bit fields are zero. To produce a bit clock, one or both of these bit fields must be programmed to a non-zero value. The UART must be disabled when it is reprogrammed.

See Section 17.1.10.1 for information about the values which must be programmed to the UARTIBRD and UARTFBRD registers.

Table 17-13. UARTIBRD Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	BAUDDIVINT															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	UARTxBase + 0x024															

Table 17-14. UARTIBRD Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:16	///	**Reserved** Write the reset value.
15:0	BAUDDIVINT	**Baud Rate Divider Integer** The integer part of the Baud Rate Divisor. This bit field is cleared to '0' at reset.

17.5.6 UART Fractional Baud Rate Divisor Register

The UARTFBRD register for each UART must be programmed with an integer representing the fractional part of the desired Baud Rate Divisor for that UART.

Each UART's baud rate is achieved by dividing the UART's Clock input signal by a Baud Rate Divisor. Parts of the Baud Rate Divisor must be written to each UART's UARTIBRD and UARTFBRD registers. The integer part of the Baud Rate Divisor is programmed to the UARTIBRD register. The fractional part of the Baud Rate Divisor is programmed to the UARTFBRD register.

UARTFBRD:BAUDDIVFRAC is reset to '0'.

Table 17-16 describes the bit fields in the UARTFBRD register.

The UARTIBRD:BAUDDIVINT and UARTFBRD:BAUDDIVFRAC bit fields are both reset to '0'. The UART will not produce a bit clock when both bit fields are zero. To produce a bit clock, one or both of these bit fields must be programmed to a non-zero value. The UART must be disabled when it is reprogrammed.

See Section 17.1.10.1 for information about the values which must be programmed to the UARTIBRD and UARTFBRD registers.

Table 17-15. UARTFBRD Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///										BAUDDIVFRAC					
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	UARTxBase + 0x028															

Table 17-16. UARTFBRD Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:6	///	**Reserved** Write the reset value.
5:0	BAUDDIVFRAC	**Baud Rate Fraction** The fractional portion of the Baud Rate Divisor. Reset = 0.

17.5.7 UART Line Control Register, High Byte

Bit fields in each UART's UARTLCR_H register control that UART's operation. The UARTLCR_H register is reset to '0'. Table 17-18 describes the bit fields in the UARTLCR_H register.

Table 17-17. UARTLCR_H Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								SPS	WLEN		FEN	STP2	EPS	PEN	BRK
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	UARTxBase + 0x02C															

Table 17-18. UARTLCR_H Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:8	///	**Reserved** Write the reset value.
7	SPS	**Stuck PARITY Select** When PEN, EPS, and SPS set to '1', the PARITY bit is transmitted and checked as a '0'. When PEN and SPS are set, to '1', and EPS is cleared to '0', the PARITY bit is transmitted and checked as a '1'. When SPS is cleared to '0', stuck PARITY is disabled. Refer to Table 17-19 for a truth table showing SPS, EPS, and PEN. (Reset = 0)
6:5	WLEN	**Length** This bit field indicates the quantity bits of data transmitted or received: 0b11 = 8 bits 0b10 = 7 bits 0b01 = 6 bits 0b00 = 5 bits (reset)
4	FEN	**Enable FIFOs** If this bit is set to '1', the UART's transmit FIFO is 16 bytes deep and the UART's receive FIFO is 16 location deep (FIFO mode). If this bit is cleared to '0', the UART's transmit and receive FIFO buffers are each 1 location deep. When the FIFOs are not 16 bytes deep, the FIFOs become single-character holding registers. (Reset = 0)
3	STP2	**Two STOP Bits Select** If this bit is set to '1', two STOP bits are transmitted at the end of the bit sequence. The UART's receive logic does not check for two STOP bits being received. (Reset = 0)
2	EPS	**Even PARITY Select** If this bit is set to '1', even PARITY generation and checking is performed during transmission and reception, which checks for an even number of 1s in data and PARITY bits. See Table 17-19. When cleared to '0' then odd PARITY is performed which checks for an odd number of 1s. This bit has no effect when PARITY is disabled by PARITY Enable (bit 1) being cleared to '0'. (Reset = 0)
1	PEN	**PARITY Enable** If this bit is set to '1', PARITY checking and generation is enabled, else PARITY is disabled and no PARITY bit added to the data frame. See Table 17-19. (Reset = 0)
0	BRK	**Send BREAK** If this bit is set to '1', a LOW level is continually output on the UARTTXD output, after completing transmission of the current character. This bit must be asserted for at least one complete frame transmission time in order to generate a BREAK condition. The transmit FIFO contents remain unaffected during a BREAK condition. For normal use, this bit must be cleared to '0'. (Reset = 0)

Table 17-19 shows how the programming of the SPS, EPS, and PEN bit fields in the UARTLCR_H register affect the PARITY of characters that are transmitted and received by the UART.

The write operation that programs the UARTLCR_H register also latches the contents of other UART registers to the UART's internal logic. The UARTLCR_H register should be programmed last, after the other registers in the UART have been reprogrammed. See Section 17.2 for additional information.

Table 17-19. Parity Truth Table

STUCK PARITY SELECT (SPS)	EVEN PARITY SELECT (EPS)	PARITY ENABLE (PEN)	PARITY BIT	
			TRANSMITTED	RECEIVED
x	x	0	(Not transmitted)	Not checked on receipt
0	1	1	Transmitted as Even PARITY	Checked on reciept for Even PARITY
0	0	1	Transmitted as Odd PARITY	Checked on receipt for Odd PARITY
1	0	1	Transmitted as 1	Assumed to be 1
1	1	1	Transmitted as 0	Assumed to be 0

NOTE: 'x' = don't care.

17.5.8 UART Control Register

Each UARTCR register can enable or disable the associated UART peripheral. The UARTCR register for UART0 can also enable the UART0 SIR ENDEC.

All bit fields in the UARTCR register are reset to '0' except for the RXE and TXE bit fields, which are reset to '1'. Table 17-21 shows the bit field definitions for the UARTCR register.

Table 17-20. UARTCR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///						RXE	TXE	LBE	///				SIRLP	SIREN	UARTEN
RESET	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	UARTxBase + 0x030															

Table 17-21. UARTCR Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:10	///	**Reserved** Write the reset value.
9	RXE	**Receive Enable** This bit field can be programmed to enable or disable the receive section of the UART. When the UART is disabled while receiving a character, the UART will finish receiving the character before becoming disabled. 1 = Receive section enabled (reset) 0 = Receive section disabled
8	TXE	**Transmit Enable** This bit field can be programmed to enable or disable the transmit section of the UART. When the UART is disabled while transmitting a character, the UART will finish transmitting the character before becoming disabled. 1 = Transmit section enabled (reset) 0 = Transmit section disabled
7	LBE	**Loop Back Enable** This bit field enables a UART peripheral's loopback mode. When a UART is operating in the loopback mode, bit sequences transmitted by the UART are also received by the UART; the output of the UART is internally connected or 'looped-back' to the input of the UART. The loopback mode is provided for the purpose of testing a UART, since UART peripherals are normally operated with the loopback mode disabled. 1 = Loopback enabled 0 = Loopback disabled (reset)
6:3	///	**Reserved** Write the reset value.

Table 17-21. UARTCR Register Bit Fields (Cont'd)

BITS	FIELD NAME	FUNCTION
2	SIRLP	**IrDA SIR Low Power Mode** This bit field selects the IrDA SIR encoding mode. When this bit field is cleared to '0', LOW-level bits are transmitted as an active HIGH pulse with a width of 3/16th of the bit period. When this bit field is set to '1', LOW-level bits are transmitted with a pulse width which is three times the period of the IrLPBaud16 input signal, regardless of the selected bit rate. Enabling the IrDA SIR Low Power mode causes the LH79520 to consume less power, but may reduce the optical transmission distance. UART1 and UART2 do not utilize this function, as they do not have SIR functionality. 1 = IrDA Low Power mode enabled 0 = IrDA SIR Low Power mode disabled (reset)
1	SIREN	**SIR Enable** This bit field is utilized only for UART0. UART1 and UART2 do not have SIR functionality, so this bit field in their corresponding registers should be considered 'Reserved — Write the reset value'. This bit field enables or disables the UART0 SIR function. When the UART0 SIR ENDEC is enabled, data is transmitted and received on the UARTIRTX0 and UARTIRRX0 pins of the LH79520, respectively. When the SIR ENDEC is enabled, data transmitted and/or received by UART0 is converted to the SIR format by the SIR ENDEC. When the UART0 SIR ENDEC is disabled, the UARTIRTX0 output remains cleared to '0', and signal transitions on the UARTIRRX0 input are not detected by UART0. 1 = SIR enabled (UART0 only) 0 = SIR disabled (UART0 only) (reset)
0	UARTEN	**UART Enable** This bit field can be programmed to enable or disable the UART. The UART must be disabled in order to reprogram the UART. UART0 must be enabled in order to utilize the UART0 SIR function. When the UART is disabled while transmitting or receiving a character, the UART will finish transmitting or receiving the character before becoming disabled. 1 = UART enabled 0 = UART disabled (reset)

NOTE: To enable a UART's transmission, both TXE and UARTEN, must be set to '1'.
To enable a UART's reception, both RXE and UARTEN must be set to '1'.

Only UART0 includes an SIR ENDEC. If the SIR is enabled, and UART0 is to be tested in the loopback mode, then UART0TCR:SIRFDME must be set to '1' in order to switch the SIR from half-duplex to full-duplex mode of operation. This is necessary in order to permit the system to simultaneously send and receive data. See Section 17.1.14 for additional considerations regarding the UART0 SIR and the loopback mode.

17.5.9 UART Interrupt FIFO Level Select Register

The UARTIFLS register determines the FIFO level at which the UARTTXINTR and UARTRXINTR interrupts for the UART will be triggered.

The UARTTXINTR and UARTRXINTR interrupts are not generated when the contents of a UART FIFO merely attain a certain level. Instead, these interrupts are generated when the contents of a FIFO exceed a programmed level, then fall back to below the level.

The UART's transmit and receive FIFOs can be programmed to be one location deep, or 16 locations deep. The FIFOs are resized together; it is not possible to resize one FIFO without resizing other FIFO.

If the FIFOs are programmed to a size of 16, then the UARTIFLS register determines the FIFO watermark. If the FIFOs are programmed to a size of 1, then the capacity of the transmit FIFO is effectively reduced to one byte and the capacity of the receive FIFO is effectively reduced to one location (received data plus status flags). When the FIFOs are programmed to a size of 1, the appropriate interrupt will be generated when the FIFO is filled, then emptied.

At reset, the RXIFLSEL and TXIFLSEL bit fields in the UARTIFLS register are conditioned so that the trigger level is satisfied when the FIFOs are at the half-way mark.

Table 17-23 defines the bit fields in the UARTIFLS register.

Table 17-22. UARTIFLS Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///										RXIFLSEL			TXIFLSEL		
RESET	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	UARTxBase + 0x034															

Table 17-23. UARTIFLS Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:6	///	**Reserved** Write the reset value.
5:3	RXIFLSEL	**Receive Interrupt FIFO Level Select** The interrupt trigger points for the Receive interrupt are: 0b000 = Receive FIFO becomes ≥ 1/8 full 0b001 = Receive FIFO becomes ≥ 1/4 full 0b010 = Receive FIFO becomes ≥ 1/2 full 0b011 = Receive FIFO becomes ≥ 3/4 full 0b100 = Receive FIFO becomes ≥ 7/8 full 0b101 through 0b111 = Reserved. Do not write.
2:0	TXIFLSEL	**Transmit Interrupt FIFO Level Select** The interrupt trigger points for the Transmit interrupt are: 0b000 = Transmit FIFO becomes ≤ 1/8 full 0b001 = Transmit FIFO becomes ≤ 1/4 full 0b010 = Transmit FIFO becomes ≤ 1/2 full 0b011 = Transmit FIFO becomes ≤ 3/4 full 0b100 = Transmit FIFO becomes ≤ 7/8 full 0b101 through 0b111 = Reserved. Do not write.

17.5.10 UART Interrupt Mask Set/Clear Register

Software can use the UARTIMSC register to enable or disable the interrupts which can be generated by a UART.

The UARTIMSC register is a read\write register cleared to '0' at reset. All interrupts are disabled at reset. Reads of this register return the present mask for each interrupt. Writing a bit field to '1' enables (masks) the corresponding interrupt. Writing a bit field to '0' disables (unmasks) the corresponding interrupt. Table 17-25 defines the register bit fields.

Table 17-24. UARTIMSC Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///					OEIM	BEIM	PEIM	FEIM	RTIM	TXIM	RXIM	///			
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	UARTxBase + 0x038															

Table 17-25. UARTIMSC Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:11	///	**Reserved** Write the reset value.
10	OEIM	**OVERRUN Error Interrupt Mask** 1 = enable this interrupt 0 = disable this interrupt (reset)
9	BEIM	**BREAK Error Interrupt Mask** 1 = enable this interrupt 0 = disable this interrupt (reset)
8	PEIM	**PARITY Error Interrupt Mask** 1 = enable this interrupt 0 = disable this interrupt (reset)
7	FEIM	**FRAMING Error Interrupt Mask** 1 = enable this interrupt 0 = disable this interrupt (reset)
6	RTIM	**Receive Timeout Interrupt Mask** 1 = enable this interrupt 0 = disable this interrupt (reset)
5	TXIM	**Transmit Interrupt Mask** 1 = enable this interrupt 0 = disable this interrupt (reset)
4	RXIM	**Receive Interrupt Mask** 1 = enable this interrupt 0 = disable this interrupt (reset)
3:0	///	**Reserved** Write the reset value.

17.5.11 UART Raw Interrupt Status Register

Reads of the UARTRIS register return the present raw status value of interrupts which can be generated by the UART.

The UARTRIS register is a read-only register; writes to this register have no effect; the value written is not retained.

Bits 31:4 are cleared to '0' at reset. The conditions of bits 3:0 are unpredictable after reset.

Table 17-27 shows the definitions for each of the bit fields in the UARTRIS register.

Table 17-26. UARTRIS Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///					OERIS	BERIS	PERIS	FERIS	RTRIS	TXRIS	RXRIS	///			
RESET	0	0	0	0	0	0	0	0	0	0	0	0	–	–	–	–
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
ADDR	UARTxBase + 0x03C															

Table 17-27. UARTRIS Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:11	///	**Reserved** Read as 0.
10	OERIS	**OVERRUN Error Interrupt Status** The state (prior to masking) of the UARTOEINTR interrupt. 1 = Interrupt active 0 = Interrupt not active (reset)
9	BERIS	**BREAK Error Interrupt Status** The state (prior to masking) of the UARTBEINTR interrupt. 1 = Interrupt active 0 = Interrupt not active (reset)
8	PERIS	**PARITY Error Interrupt Status** The state (prior to masking) of the UARTPEINTR interrupt. 1 = Interrupt active 0 = Interrupt not active (reset)
7	FERIS	**FRAMING Error Interrupt Status** The state (prior to masking) of the UARTFEINTR interrupt. 1 = Interrupt active 0 = Interrupt not active (reset)

Table 17-27. UARTRIS Register Bit Fields (Cont'd)

BITS	FIELD NAME	FUNCTION
6	RTRIS	**Receive Timeout Interrupt Status** The state (prior to masking) of the UARTRTINTR interrupt. This bit field indicates the state (prior to masking) of the UARTRTINTR interrupt. The UART cannot set this bit field to '1' unless the corresponding interrupt mask (UARTIMSC:RTIM) has been written to '1'. The interrupt mask UARTIMSC:RTIM acts an enable, to save power. This bit field can also be read from UARTMIS:RTMIS. 1 = Interrupt active 0 = Interrupt not active (reset)
5	TXRIS	**Transmit Interrupt Status** The state (prior to masking) of the UARTTXINTR interrupt. 1 = Interrupt active 0 = Interrupt not active (reset)
4	RXRIS	**Receive Interrupt Status** The state (prior to masking) of the UARTRXINTR interrupt. 1 = Interrupt active 0 = Interrupt not active (reset)
3:0	///	**Reserved** Reads are unpredictable.

17.5.12 UART Masked Interrupt Status Register

Reads of the UARTMIS register return the present masked status value of the UART interrupts.

This is a read-only register; writes have no effect; the value written is not retained. Bits 31:4 are cleared to '0' at reset. The conditions of bits 3:0 are unpredictable after reset. Table 17-29 shows the definitions of the bit fields in the UARTMIS register.

Table 17-28. UARTMIS Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///					OEMIS	BEMIS	PEMIS	FEMIS	RTMIS	TXMIS	RXMIS	///			
RESET	0	0	0	0	0	0	0	0	0	0	0	0				
TYPE	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
ADDR	UARTxBase + 0x040															

Table 17-29. UARTMIS Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:11	///	**Reserved** Reads as 0.
10	OEMIS	**OVERRUN Error Masked Interrupt Status** The state (after masking) of the UARTOEINTR interrupt. 1 = Interrupt active 0 = Interrupt not active (reset)
9	BEMIS	**BREAK Error Masked Interrupt Status** The state (after masking) of the UARTBEINTR interrupt. 1 = Interrupt active 0 = Interrupt not active (reset)
8	PEMIS	**PARITY Error Masked Interrupt Status** The state (after masking) of the UARTPEINTR interrupt. 1 = Interrupt active 0 = Interrupt not active (reset)
7	FEMIS	**FRAMING Error Masked Interrupt Status** The state (after masking) of the UARTFEINTR interrupt. 1 = Interrupt active 0 = Interrupt not active (reset)
6	RTMIS	**Receive Timeout Masked Interrupt Status** The state (after masking) of the UARTRTINTR interrupt. 1 = Interrupt active 0 = Interrupt not active (reset)
5	TXMIS	**Transmit Masked Interrupt Status** The state (after masking) of the UARTTXINTR interrupt. 1 = Interrupt active 0 = Interrupt not active (reset)
4	RXMIS	**Receive Masked Interrupt Status** The state (after masking) of the UARTRXINTR interrupt. 1 = Interrupt active 0 = Interrupt not active (reset)
3:0	///	**Reserved** Reads are unpredictable.

17.5.13 UART Interrupt Clear Register

Writes to the UARTICR register can individually clear each of the interrupts which can be generated by the associated UART.

A write of '1' to a bit field in the UARTICR register clears the corresponding interrupt. A write of '0' has no effect. This is a write-only register; reads of this register will return unpredictable values. Table 17-31 shows the bit assignments for the UARTICR register.

Table 17-30. UARTICR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET																
TYPE	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///					OEIC	BEIC	PEIC	FEIC	RTIC	TXIC	RXIC	///			
RESET																
TYPE	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
ADDR	UARTxBase + 0x044															

Table 17-31. UARTICR Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:11	///	**Reserved** Write 0.
10	OEIC	**OVERRUN Error Interrupt Clear** 1= Clears the UARTOEINTR interrupt 0 = No effect
9	BEIC	**BREAK Error Interrupt Clear** 1 = Clears the UARTBEINTR interrupt 0 = No effect
8	PEIC	**PARITY Error Interrupt Clear** 1 = Clears the UARTPEINTR interrupt 0 = No effect.
7	FEIC	**FRAMING Error Interrupt Clear** 1 = Clears the UARTFEINTR interrupt 0 = No effect
6	RTIC	**Receive Timeout Interrupt Clear** 1 = Clears the UARTRTINTR interrupt 0 = No effect
5	TXIC	**Transmit Interrupt Clear** 1 = Clears the UARTTXINTR interrupt 0 = No effect
4	RXIC	**Receive Interrupt Clear** 1 = Clears the UARTRXINTR interrupt 0 = No effect
3:0	///	**Reserved** Write 0.

Chapter 18
IrDA Communications (SIR)

The LH79520 includes an IrDA-compatible Serial InfraRed (SIR) Encoder/Decoder (ENDEC) which supports serial infrared communications protocol on UART0. This Chapter explains the theory and operation of the LH79520 SIR peripheral.

18.1 Theory of Operation

The LH79520 includes three UARTs. UART0 includes a SIR ENDEC and can be programmed to implement IrDA Serial InfraRed (SIR) communications protocols. Figure 18-1 shows the way in which the SIR is integrated into the LH79520.

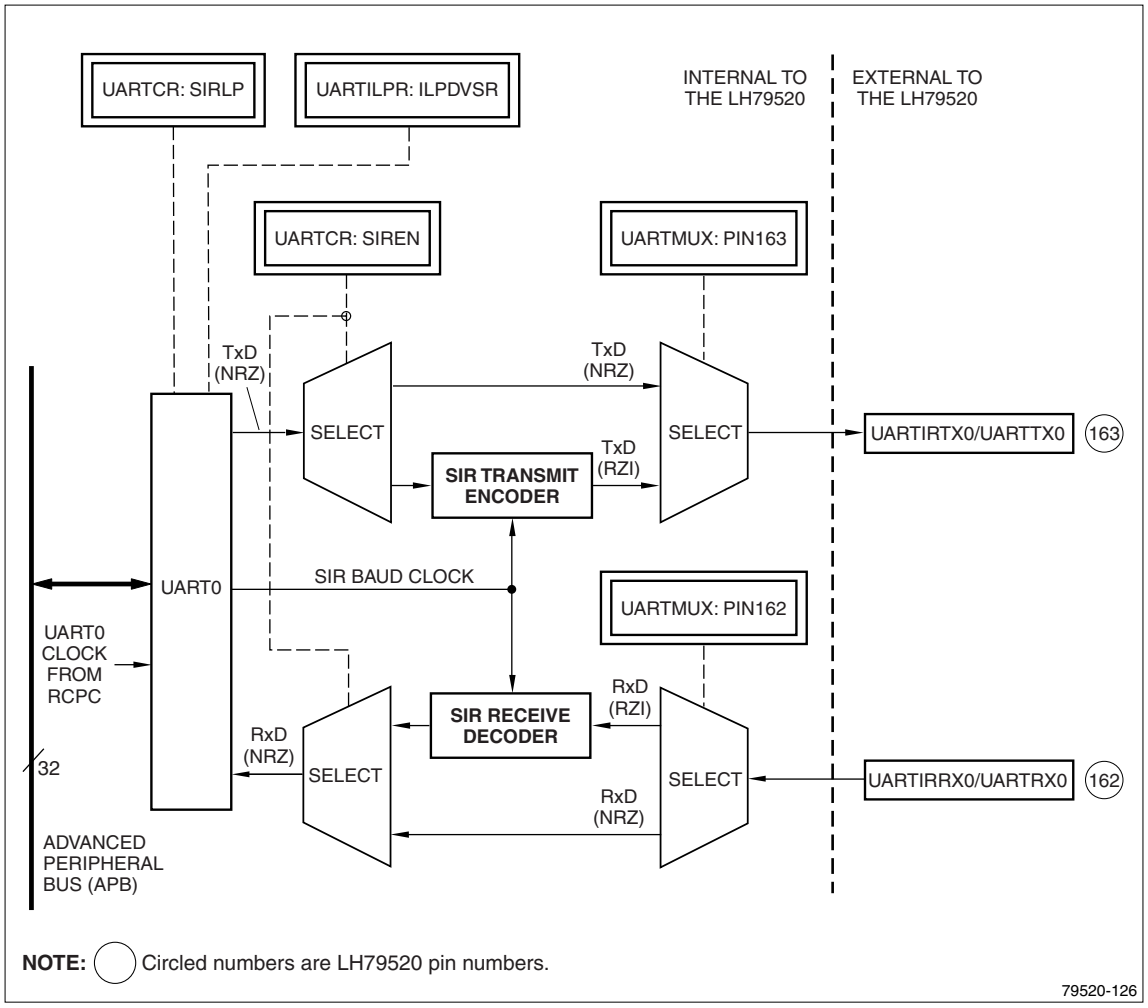


Figure 18-1. SIR Peripheral

Data formatted for a UART is not compatible with SIR requirements; the UART data must be reformatted for SIR use. UARTs transmit and receive data encoded to the NRZ (Non-Return to Zero) format but the IrDA physical layer specifies the use of the RZI (Return to Zero, Inverted) format. The RZI format represents a logical '0' as a discrete pulse, typically a pulse of light.

Figure 18-1 shows that the LH79520 SIR includes a Serial InfraRed Transmit Encoder which can convert serial data transmitted by UART0 from NRZ to RTI format. The LH79520 also includes a Serial InfraRed Receive Encoder, to convert data received serially in RTI format to the NRZ format required by UART0.

18.2 Programming the SIR

The LH79520 SIR ENDEC supports SIR functions to speeds of 115.2 kbps, half-duplex. The SIR ENDEC also supports normal 3/16 bit durations and low power (1.41 μs to 2.33 μs) bit durations.

For low-power mode bit durations, the reference clock which is input to UART0 can be divided by values from 1 to 512 (decimal), for use as the SIR baud clock.

The SIR system in the LH79520 is a half-duplex system; the SIR cannot receive while it is transmitting. This is to accomodate IrDA devices which include both a transmitter and a reciever; since the reciever can 'see' the transmitter, it will output false reception data unless a delay is included to allow the transmit pulses to complete and decay.

The IrDA SIR physical layer specifies a minimum 10 μs delay between transmission and reception. This delay must be implemented in software.

Programming the LH79520 SIR involves first programming UART0, then enabling the SIR ENDEC to modify the data on the serial side of the UART. Each of the registers shown in Figure 18-1 must be correctly programmed in order to implement SIR communications.

- Program UARTCR:SIRLP to select the desired IrDA encoding mode.
- Program UARTILPR:ILPDVSR to specify the SIR baud rate divisor.
- Program UARTMUX:PIN162 and UARTMUX:PIN162 to select the SIR function for those LH79520 pins.
- Program UARTCR:SIREN to enable the SIR encoder and decoder.

UART0 Clock generation is documented in Chapter 7 – Reset, Clock Generation, and Power Control (RCPC).

The UARTCR and UARTILPR registers, and information about the LH79520 UART peripherals is documented in Chapter 17 – Universal Asynchronous Receiver/ Transmitters (UARTs).

The UARTMUX register is documented in Chapter 8 – I/O Control and Multiplexing (IOCON).

18.3 Loopback Testing

When a UART is operating in the loopback mode, the UART's transmit data path (the output) is connected directly to the UART's receive data path (the input). Connecting the UART's output to the UART's input causes the UART to immediately receive the data the UART transmits. The data may also be available at the UART's output pins.

Operating UART0 in the loopback mode requires additional considerations because UART0 includes a SIR ENDEC. If the SIR is disabled, then the loopback mode for UART0 functions like the loopback mode for UART1 or UART2. If the SIR is enabled, then additional programming is required.

The SIR ENDEC is a half-duplex device and therefore cannot under normal circumstances receive while it is transmitting. When the SIR is enabled and UART0 is operated in the loopback mode, the output of the SIR is inverted before being routed back to the input of the SIR, to permit loopback testing.

See Section 18.3 and Chapter 17 – Universal Asynchronous Receiver/Transmitters (UARTs) for additional information about utilizing the UART0 SIR during loopback testing.

18.4 SIR Programmer's Model

The SIR is utilized in conjunction with UART0. The registers which affect the SIR are located at offsets from the UART0 Register Base Addresses:

UART0Base: 0xFFFC0000

All registers must be accessed as a full word. The UART does not support byte and half-word writes.

18.5 SIR Register Summary

The programmable SIR registers are summarized in Table 18-1 and explained in more detail on the pages that follow the table.

See Table 17-5 in Chapter 17 – Universal Asynchronous Receiver/Transmitters (UARTs) for a complete listing of the UART registers.

Table 18-1. SIR Register Summary

ADDRESS	REGISTER NAME	DESCRIPTION	NOTES
UART0Base + 0x020	UARTILPR	IrDA low-power counter Register.	
UART0Base + 0x030	UARTCR	UART Control Register	1, 2
UARTBase + 0x080	UARTTCR	SIR Test Control Register.	2

NOTES:
1. The UART0CR register concerns both UART0 and the SIR ENDEC included in UART0. The Programmer's Model for the UART0CR register is included in Chapter 17 – Universal Asynchronous Receiver/Transmitters (UARTs).
2. IrDA (SIR) functionality is available only for UART0. UART1 and UART2 do not offer SIR functionality. This Register Summary lists the UART0 registers to illustrates how the programmable SIR registers are incorporated into the UART0 memory map. See Chapter 17 – Universal Asynchronous Receiver/Transmitters (UARTs) for more information about programming UART0, UART1 and UART2.

18.6 SIR Register Descriptions

This section of this chapter of this User's Guide lists and describes each of the registers that configure the SIR in the LH79520 UART0 peripheral. The UART0 peripheral must be disabled before any of its control registers are reprogrammed.

18.6.1 IrDA Low-Power Counter Register

The UART0ILPR register should be programmed with a divisor value utilized to generate the SIR Baud Clock signal.

The UART0ILPR register is reset to '0' and must be reprogrammed with a non-zero divisor value for use. Programming a zero value will result in no SIR Baud Clock pulses being generated.

The SIR Baud Clock is generated by dividing the UART0 Clock input signal (from the LH79520 RCPC) by the Low Power Divisor Value written to this register.

The Low Power Divisor value is calculated as follows:

UARTILPR:ILPDVSR = (fUART0 Clock/fSIR Baud Clock) –1,

where fSIR Baud Clock is nominally 1.8432 MHz.

The divisor value written to this register must be chosen so that

1.42 MHz < SIR Baud Clock < 2.12 MHz.

The divisor value must produce a low power pulse duration of 1.41 μs - 2.11 μs (three times the period of SIR Baud Clock).

The minimum frequency of SIR Baud Clock ensures that pulses less than one period of SIR Baud Clock are rejected, but that pulses greater in duration than 1.4 μs are accepted as valid pulses.

Table 18-3 describes the bit fields in the UARTILPR register.

Table 18-2. UART0ILPR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								ILPDVSR							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	UART0Base + 0x020															

Table 18-3. UART0ILPR Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:8	///	**Reserved** Write the Reset value.
7:0	ILPDVSR	**InfraRed Low Power Divisor** See text for value to be programmed. Reset = 0

18.6.2 UART0 Test Control Register

The UART0TCR register permits loopback testing of the UART0 SIR functional block.

When enabled, the UART0 SIR normally operates in the half-duplex mode. The SIR must be placed in the full-duplex mode for loopback testing of the SIR function. Software can use UART0TCR:SIRFDME to place the SIR in a full-duplex mode. This mode is provided for testing only; for normal operation, the SIR should be configured for half-duplex operation.

The UART0TCR register is a read-write register which is reset to 0. Table 18-5 describes the bit fields in the UART0TCR register.

UART1 and UART2 do not include SIR capability. The registers located at UART1Base + 0x080 and UART2Base + 0x080 should be considered 'Reserved — do not write'.

See Chapter 17 – Universal Asynchronous Receiver/Transmitters (UARTs) for more information about the LH79520 SIR ENDEC.

See Section 17.1.14 in Chapter 17 for more information about loopback testing.

Table 18-4. UART0TCR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///													SIRFDME	///	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	UARTxBase + 0x080															

Table 18-5. UART0TCR Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:3	///	**Reserved** Write the Reset value.
2	SIRFDME	**SIR Full-Duplex Mode Enable for Loopback Testing** This bit field is provided to facilitate loopback testing of the UART0 SIR. Writing this bit field to '1' enables the SIR receive data path during SIR transmission. 1 = SIR receive data path is enabled during SIR transmission 0 = SIR receive data path is disabled during SIR transmission (reset)
1:0	///	**Reserved** Write the Reset value.

Chapter 19
General Purpose I/O (GPIO)

The LH79520 MCU includes four GPIO modules which provide a total of eight GPIO ports. This Chapter explains the theory and operation of the General Purpose I/O (GPIO) ports.

19.1 Theory of Operation

Each LH79520 GPIO module provides two I/O ports. The four GPIO modules in the LH79520 provide a total of 64 bits of programmable input/output, organized as eight 8-bit ports. The GPIO ports are designated PortA through PortH and each pin of each port is programmable as either an input or an output. Control words may be read back from each port and all GPIO pins not otherwise utilized at Reset default to inputs.

Figure 19-1 illustrates how the GPIO modules are integrated into the LH79520 MCU. Each GPIO module is interfaced to the APB, and AHB accesses of the registers in the GPIO modules occur through the APB Bridge.

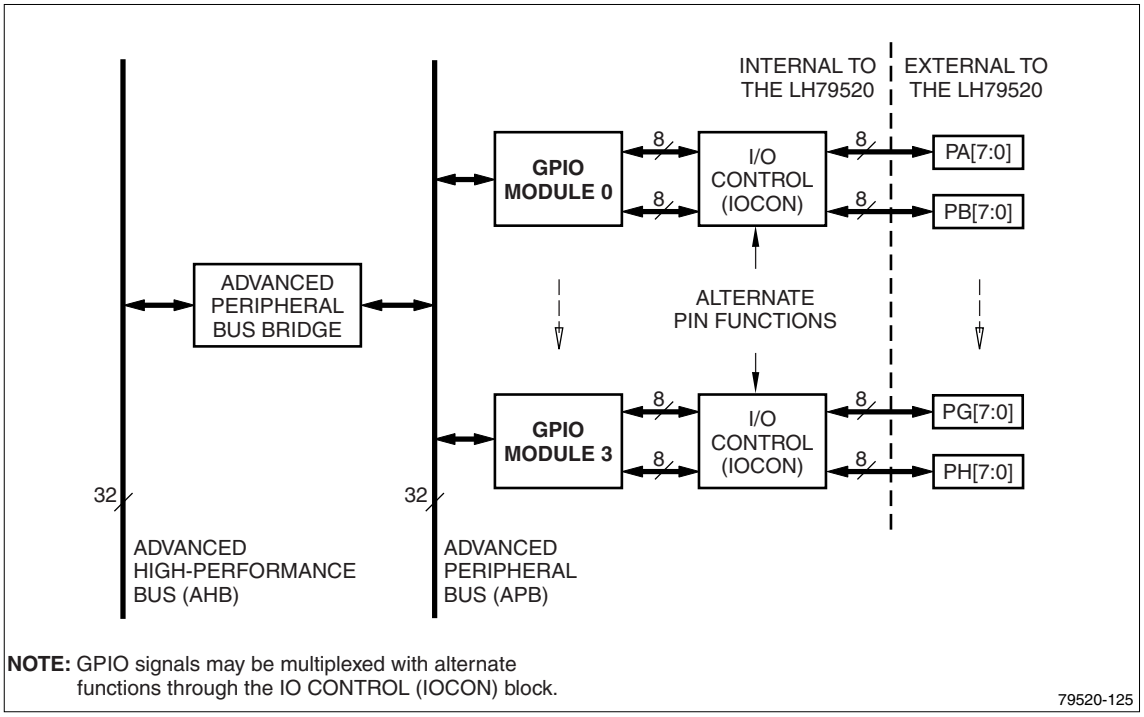


Figure 19-1. LH79520 GPIO Peripherals

All GPIO signals are routed through multiplexers before connection to the external pins of the LH79520 package. The multiplexers are controlled by the I/O Control (IOCON) functional block. Because of this multiplexing, the LH79520 pins associated with GPIO signals also support up to two alternate functions. In most cases the multiple functions on a pin cannot be utilized simultaneously.

The functions assigned to each LH79520 pin at Reset have been carefully chosen to support the majority of designs, but multiplexing must be considered when designing a system around the LH79520. See Chapter 1 – Introduction and Chapter 8 – I/O Control and Multiplexing (IOCON) for additional information regarding the alternate functions supported by the LH79520 pins which also support the GPIO signals.

Figure 19-2 shows a simplified block diagram of one LH79520 GPIO module. Each module contains two 8-bit ports, and each bit of each port can function as either an input or an output. The direction of each bit of each port can be selected by a write to the data direction register associated with the port. All bits in all GPIO ports are Reset as inputs except for those GPIO port bits which are Reset to alternate functions. Table 1-3 in Chapter 1 lists the function of each LH79520 pin at Reset.

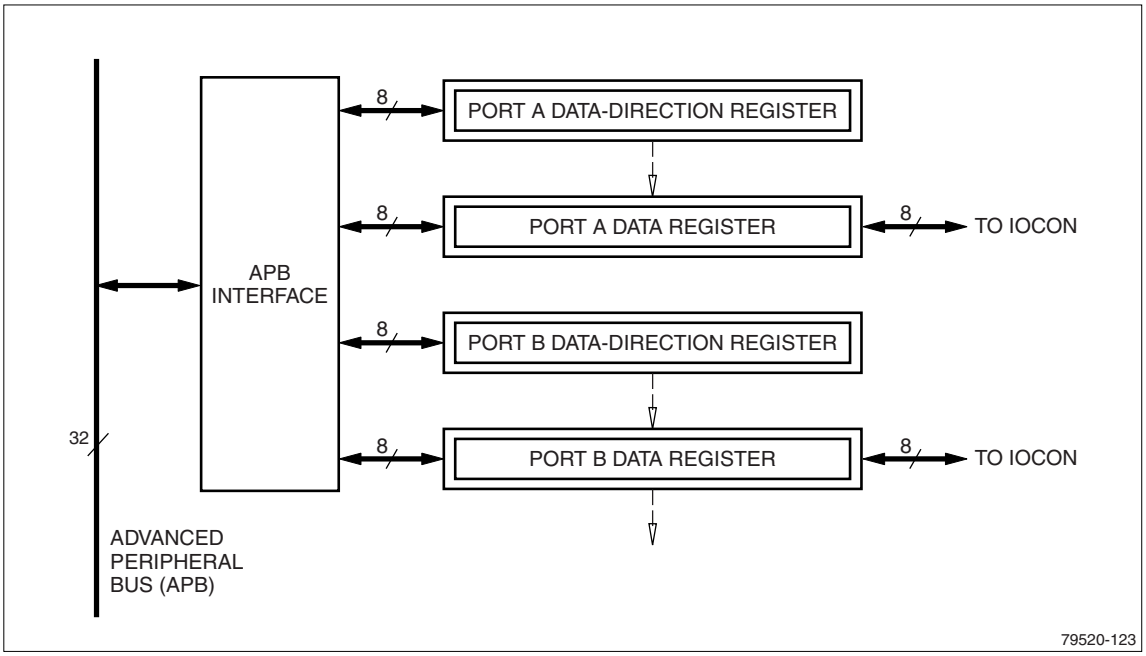


Figure 19-2. GPIO Block Diagram

19.2 GPIO Programmer's Model

The Register Base Addresses for the four LH79520 GPIO modules are:

GPIOBase0: 0xFFFDF000

GPIOBase1: 0xFFFDE000

GPIOBase2: 0xFFFDD000

GPIOBase3: 0xFFFDC000

All registers in the GPIO must be accessed as a full word. The GPIO does not support byte and half-word writes.

19.3 GPIO Register Summary

The LH79520 includes four GPIO modules. Table 19-1 summarizes the programmable registers in each module. Table 19-2 through Table 19-33 describe each register in detail.

Table 19-1. GPIO Register Summary

ADDRESS	REGISTER NAME	DESCRIPTION
GPIOBase0 + 0x000	GPIOPADR	Port A data register
GPIOBase0 + 0x004	GPIOPBDR	Port B data register
GPIOBase0 + 0x008	GPIOPADDR	Port A data direction register
GPIOBase0 + 0x00C	GPIOPBDDR	Port B data direction register
GPIOBase0 + 0x010 - 0x0FF	///	Reserved — do not write
GPIOBase1 + 0x000	GPIOPCDR	Port C data register
GPIOBase1 + 0x004	GPIOPDDR	Port D data register
GPIOBase1 + 0x008	GPIOPCDDR	Port C data direction register
GPIOBase1 + 0x00C	GPIOPDDDR	Port D data direction register
GPIOBase1 + 0x010 - 0x0FF	///	Reserved — do not write
GPIOBase2 + 0x000	GPIOPEDR	Port E data register
GPIOBase2 + 0x004	GPIOPFDR	Port F data register
GPIOBase2 + 0x008	GPIOPEDDR	Port E data direction register
GPIOBase2 + 0x00C	GPIOPFDDR	Port F data direction register
GPIOBase2 + 0x010 - 0x0FF	///	Reserved — do not write
GPIOBase3 + 00x00	GPIOPGDR	Port G data register
GPIOBase3 + 0x004	GPIOPHDR	Port H data register
GPIOBase3 + 0x008	GPIOPGDDR	Port G data direction register
GPIOBase3 + 0x00C	GPIOPHDDR	Port H data direction register
GPIOBase3 + 0x010 - 0FF	///	Reserved — do not write

19.4 GPIO Register Descriptions

This section of this chapter of this User's Guide lists and describes the registers that configure the LH79520 GPIO.

19.4.1 GPIO Port A Data Register

Values written to GPIOPADR will be output on the PORT A pins if the corresponding GPIOPADDR data direction bits are set HIGH (the port is programmed to be an output).

The values read from this register are determined, for each bit, by the value of the corresponding bit in the data direction register GPIOPADDR. A read of this register will return the last bit value written when the bit is configured as an output, or it will return the current value on the corresponding port input bit if the bit is configured as an input.

All bits are cleared to '0' at Reset. Table 19-3 shows the bit assignments for the GPIOPADR register.

Table 19-2. GPIOPADR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								PADATA							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	GPIOBase0 + 0x00															

Table 19-3. GPIOPADR Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:8	///	**Reserved** Write the Reset value.
7:0	PADATA	**Port A Data** Port A input or output data.

19.4.2 GPIO Port B Data Register

Values written to GPIOPBDR will be output on the PORTB pins if the corresponding GPIOPBDDR data direction bits are set HIGH (the port is programmed to be an output).

The values read from this register are determined, for each bit, by the value of the corresponding bit in the data direction register GPIOPBDDR. A read of this register will return the last bit value written when the bit is configured as an output, or it will return the current value on the corresponding port input bit if the bit is configured as an input.

All bits are cleared by a Reset. Table 19-5 shows the bit assignments for GPIOPBDR.

Table 19-4. GPIOPBDR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								PBDATA							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	GPIOBase0 + 0x04															

Table 19-5. GPIOPBDR Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:8	///	**Reserved** Write the Reset value.
7:0	PBDATA	**Port B Data** Port B input or output data.

19.4.3 GPIO Port A Data Direction Register

Bits set HIGH in GPIOPADDR will set the corresponding pin in PORTA to be an output.

Clearing a bit to '0' configures the pin to be an input. All bits are cleared to '0' by a Reset. Table 19-7 shows the bit assignments for GPIOPADDR.

Table 19-6. GPIOPADDR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								PADDR							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	GPIOBase0 + 0x08															

Table 19-7. GPIOPADDR Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:8	///	**Reserved** Write the Reset value.
7:0	PADDR	**Port A Data Direction** Bits set = Port A output. Bits cleared = Port A input.

19.4.4 GPIO Port B Data Direction Register

Bits set HIGH in the GPIOPBDDR will set the corresponding pin in PORTB to be an output.

Clearing a bit to '0' configures the pin to be an input. All bits are cleared to '0' by a Reset. Table 19-9 shows the bit assignments for the GPIOPBDDR register.

Table 19-8. GPIOPBDDR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								PBDDR							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	GPIOBase0 + 0x0C															

Table 19-9. GPIOPBDDR Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:8	///	**Reserved** Write the Reset value.
7:0	PBDDR	**Port B Data Direction** Bits set = Port B output. Bits cleared = Port B input.

19.4.5 GPIO Port C Data Register

Values written to GPIOPCDR will be output on the PORTC pins if the corresponding GPIOPCDDR data direction bits are set HIGH (the port is programmed to be an output).

The values read from this register are determined, for each bit, by the value of the corresponding bit in the data direction register GPIOPCDDR. A read of this register will return the last bit value written when the bit is configured as an output, or it will return the current value on the corresponding port input bit if the bit is configured as an input.

All bits are cleared by a Reset. Table 19-11 shows the bit assignments for GPIOPCDR.

Table 19-10. GPIOPCDR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								PCDATA							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	GPIOBase1 + 0x00															

Table 19-11. GPIOPCDR Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:8	///	**Reserved** Write the Reset value.
7:0	PCDATA	**Port C Data** Port C input or output data.

19.4.6 GPIO Port D Data Register

Values written to GPIOPDDR will be output on the PORTD pins if the corresponding GPIOPDDDR data direction bits are set HIGH (the port is programmed to be an output).

The values read from this register are determined, for each bit, by the value of the corresponding bit in the data direction register GPIOPDDDR. A read of this register will return the last bit value written when the bit is configured as an output, or it will return the current value on the corresponding port input bit if the bit is configured as an input.

All bits are cleared by a Reset. Table 19-13 shows the bit assignments for GPIOPDDR.

Table 19-12. GPIOPDDR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								PDDATA							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	GPIOBase1 + 0x04															

Table 19-13. GPIOPDDR Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:8	///	**Reserved** Write the Reset value.
7:0	PDDATA	**Port D Data** Port D input or output data.

19.4.7 GPIO Port C Data Direction Register

Bits set HIGH in GPIOPCDDR will set the corresponding pin in PORTC to be an output. Clearing a bit configures the pin to be an input. All bits are cleared by a Reset.

Table 19-15 shows the bit assignments for the GPIOPCDDR register.

Table 19-14. GPIOPCDDR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								PCDDR							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	GPIOBase1 + 0x08															

Table 19-15. GPIOPCDDR Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:8	///	**Reserved** Write the Reset value.
7:0	PCDDR	**Port C Data Direction** Bits set = Port C output. Bits cleared = Port C input.

19.4.8 GPIO Port D Data Direction Register

Bits set HIGH in the GPIOPDDDR will set the corresponding pin in PORTD to be an output. Clearing a bit configures the pin to be an input. All bits are cleared by a Reset.

Table 19-17 shows the bit assignments for the GPIOPDDDR register.

Table 19-16. GPIOPDDDR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								PDDDR							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	GPIOBase1 + 0x0C															

Table 19-17. GPIOPDDDR Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:8	///	**Reserved** Write the Reset value.
7:0	PDDDR	**Port D Data Direction** Bits set = Port D output. Bits cleared = Port D input.

19.4.9 GPIO Port E Data Register

Values written to GPIOPEDR will be output on the PORTE pins if the corresponding GPIOPEDDR data direction bits are set HIGH (the port is programmed to be an output).

The values read from this register are determined, for each bit, by the value of the corresponding bit in the data direction register GPIOPEDDR. A read of this register will return the last bit value written when the bit is configured as an output, or it will return the current value on the corresponding port input bit if the bit is configured as an input.

All bits are cleared by a Reset. Table 19-19 shows the bit assignments for the GPIOPEDR register.

Table 19-18. GPIOPEDR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								PEDATA							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	GPIOBase2 + 0x00															

Table 19-19. GPIOPEDR Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:8	///	**Reserved** Write the Reset value.
7:0	PEDATA	**Port E Data** Port E input or output data.

19.4.10 GPIO Port F Data Register

Values written to GPIOPFDR will be output on the PORTF pins if the corresponding GPIOPFDDR data direction bits are set HIGH (the port is programmed to be an output).

The values read from this register are determined, for each bit, by the value of the corresponding bit in the data direction register GPIOPFDDR. A read of this register will return the last bit value written when the bit is configured as an output, or it will return the current value on the corresponding port input bit if the bit is configured as an input.

All bits are cleared by a Reset. Table 19-21 shows the bit assignments for the GPIOPFDR register.

Table 19-20. GPIOPFDR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								PFDATA							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	GPIOBase2 + 0x04															

Table 19-21. GPIOPFDR Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:8	///	**Reserved** Write the Reset value.
7:0	PFDATA	**Port F Data** Port F input or output data.

19.4.11 GPIO Port E Data Direction Register

Bits set HIGH in GPIOPEDDR will set the corresponding pin in PORTE to be an output. Clearing a bit configures the pin to be an input. All bits are cleared by a Reset.

Table 19-23 shows the bit assignments for the GPIOPEDDR register.

Table 19-22. GPIOPEDDR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								PEDDR							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	GPIOBase2 + 0x08															

Table 19-23. GPIOPEDDR Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:8	///	**Reserved** Write the Reset value.
7:0	PEDDR	**Port E Data Direction** Bits set = Port E output. Bits cleared = Port E input.

19.4.12 GPIO Port F Data Direction Register

Bits set HIGH in the GPIOPFDDR will set the corresponding pin in PORTF to be an output. Clearing a bit configures the pin to be an input. All bits are cleared by a Reset.

Table 19-25 shows the bit assignments for the GPIOPFDDR register.

Table 19-24. GPIOPFDDR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								PFDDR							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	GPIOBase2 + 0x0C															

Table 19-25. GPIOPFDDR Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:8	///	**Reserved** Write the Reset value.
7:0	PFDDR	**Port F Data Direction** Bits set = Port F output. Bits cleared = Port F input.

19.4.13 GPIO Port G Data Register

Values written to GPIOPGDR will be output on the PORTG pins if the corresponding GPIOPGDDR data direction bits are set HIGH (the port is programmed to be an output).

The values read from this register are determined, for each bit, by the value of the corresponding bit in the data direction register GPIOPGDDR. A read of this register will return the last bit value written when the bit is configured as an output, or it will return the current value on the corresponding port input bit if the bit is configured as an input.

All bits are cleared by a Reset. Table 19-27 shows the bit assignments for the GPIOPGDR register.

Table 19-26. GPIOPGDR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								PGDATA							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	GPIOBase3 + 0x00															

Table 19-27. GPIOPGDR Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:8	///	**Reserved** Write the Reset value.
7:0	PGDATA	**Port G Data** Port G input or output data.

19.4.14 GPIO Port H Data Register

Values written to GPIOPHDR will be output on the PORTH pins if the corresponding GPIOPHDDR data direction bits are set HIGH (the port is programmed to be an output).

The values read from this register are determined, for each bit, by the value of the corresponding bit in the data direction register GPIOPHDDR. A read of this register will return the last bit value written when the bit is configured as an output, or it will return the current value on the corresponding port input bit if the bit is configured as an input.

All bits are cleared by a Reset. Table 19-29 shows the bit assignments for the GPIOPHDR register.

Table 19-28. GPIOPHDR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								PHDATA							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	GPIOBase3 + 0x04															

Table 19-29. GPIOPHDR Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:8	///	**Reserved** Write the Reset value.
7:0	PHDATA	**Port H Data** Port H input or output data.

19.4.15 GPIO Port G Data Direction Register

Bits set HIGH in GPIOPGDDR will set the corresponding pin in PORTG to be an output. Clearing a bit configures the pin to be an input. All bits are cleared by a Reset.

Table 19-31 shows the bit assignments for the GPIOPGDDR register.

Table 19-30. GPIOPGDDR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								PGDDR							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	GPIOBase3 + 0x08															

Table 19-31. GPIOPGDDR Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:8	///	**Reserved** Write the Reset value.
7:0	PGDDR	**Port G Data Direction** Bits set = Port G output. Bits cleared = Port G input.

19.4.16 GPIO Port H Data Direction Register

Bits set HIGH in the GPIOPHDDR will set the corresponding pin in PORTH to be an output. Clearing a bit configures the pin to be an input. All bits are cleared by a Reset.

Table 19-33 shows the bit assignments for the GPIOPHDDR register.

Table 19-32. GPIOPHDDR Register

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	///															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	///								PHDDR							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TYPE	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
ADDR	GPIOBase3 + 0x0C															

Table 19-33. GPIOPHDDR Register Bit Fields

BITS	FIELD NAME	FUNCTION
31:8	///	**Reserved** Write the Reset value.
7:0	PHDDR	**Port H Data Direction** Bits set = Port H output. Bits cleared = Port H input.

Chapter 20
Status and Configuration Registers

This Chapter lists all of the programmable Status and Configuration Registers within the LH79520 MCU. The registers are grouped by individual subsystems and listed in ascending numeric sequence, by Base Address and Address Offset from the Base Address.

20.1 Register Addressing

Any LH79520 register's absolute address is the sum of its Base Address plus its Address Offset from that Base Address. Individual registers are listed with their Address Offset value from a Base Address. Where multiple similar subsystems exist, the Base Addresses are listed together for clarity.

For example, the absolute address of the UARTFR register for UART0 (the first of three UART modules) is:

0xFFFC0018 = 0xFFFC0000 + 0x018

The absolute address for the register with the same function (i.e.: UARTFR) in UART2 is:

0xFFFC2018 = 0xFFFC2000 + 0x018

This presentation encourages efficient software data structures which facilitate access to each individual register.

Where a register has a different function depending on whether it is being read or written, the register may be shown with two different names, as if it is two different registers. One of the two registers will be described as a read-only register; the other register will be described as a write-only register. In such cases, Programmers should consider these two registers to be a read-only register and a write-only register, mapped to the same address. Programmers should use 32-bit operations to access all registers. The only devices that will accept word, half-word, and byte accesses are the SMC, SRAMC, VIC, and CLCDC.

For more information about the functions of a particular register, and the bit fields within it, refer to the chapter of this User's Guide which explains the functional block containing that register. The UARTFR register, for example, is explained in Chapter 17, Section 17.5.4.

20.2 Register Addresses

Table 20-1 is a summary of all registers in the LH79520.

Table 20-1. LH79520 Status and Configuration Registers

OFFSET FROM BASE ADDRESS	REGISTER NAME	REGISTER DESCRIPTION
UARTx BASE ADDRESS FOR UART 0, UART0Base = 0xFFFC0000 BASE ADDRESS FOR UART 1, UART1Base = 0xFFFC1000 BASE ADDRESS FOR UART 2, UART2Base = 0xFFFC2000		
0x000	UARTDR	UARTx Data Register: Read: Received Data + Status Write: Data to be Transmitted
0x004	UARTRSR/UARTECR	UARTx Receive Status Register (Read) UARTx Error Clear Register (Write)
0x008 - 0x017	///	Reserved — do not write
0x018	UARTFR	UARTx Flag Register (Read only)
0x01C - 0x01F	///	Reserved — do not write
0x020	UARTILPR	UART0: UART0 SIR Low-Power Counter Register UART1: Reserved — do not write. UART2: Reserved — do not write.
0x024	UARTIBRD	UARTx Integer Baud Rate Divisor Register
0x028	UARTFBRD	UARTx Fractional Baud Rate Divisor Register
0x02C	UARTLCR_H	UARTx Line Control Register, HIGH byte
0x030	UARTCR	UARTx Control Register
0x034	UARTIFLS	UARTx Interrupt FIFO Level-Select Register
0x038	UARTIMSC	UARTx Interrupt Mask Set/Clear
0x03C	UARTRIS	UARTx Raw Interrupt Status
0x040	UARTMIS	UARTx Masked Interrupt Status
0x044	UARTICR	UARTx Interrupt Clear Register
0x04C - 0x07F	///	Reserved — do not write.
0x080	UARTTCR	UART0: SIR Test Control Register UART1: Reserved — do not write UART2: Reserved — do not write
0x084 - 0xFFF	///	Reserved — do not write.
PULSE WIDTH MODULATORS (PWM) BASE ADDRESS PWMBase = 0xFFFC3000		
0x000	PWM0_TC	PWM 0, Terminal Count
0x004	PWM0_DC	PWM 0, Duty Cycle
0x008	PWM0_EN	PWM 0, Enable
0x00C	PWM0_INV	PWM 0, Invert
0x010	PWM0_SYNC	PWM 0, Synchronization
0x014 - 0x01F	///	Reserved — do not write
0x020	PWM1_TC	PWM 1, Terminal Count
0x024	PWM1_DC	PWM 1, Duty Cycle
0x028	PWM1_EN	PWM 1, Enable
0x02C	PWM1_INV	PWM 1, Invert
0x030	///	Reserved — do not write

Table 20-1. LH79520 Status and Configuration Registers (Cont'd)

OFFSET FROM BASE ADDRESS	REGISTER NAME	REGISTER DESCRIPTION
TIMERS BASE ADDRESS FOR TIMER0, TIMER0Base = 0xFFFC4000 BASE ADDRESS FOR TIMER1, TIMER1Base = 0xFFFC4020 BASE ADDRESS FOR TIMER2, TIMER2Base = 0xFFFC5000 BASE ADDRESS FOR TIMER3, TIMER3Base = 0xFFFC5020		
0x000	Load	Timer Load
0x004	Value	Timer Value
0x008	Control	Timer Control
0x00C	Clear	Timer Clear
0x010 - 0x01F	///	Reserved — do not write.
SYNCHRONOUS SERIAL PORT (SSP) BASE ADDRESS SSPBase = 0xFFFC6000		
0x000	SSPCR0	Control Register 0
0x004	SSPCR1	Control Register 1
0x008	SSPDR	Receive FIFO Data (Read) Transmit FIFO Data (Write)
0x00C	SSPSR	Status Register
0x010	SSPCPSR	Clock Prescale Register
0x014	SSPIIR/SSPICR	Interrupt Identification (Read) / Interrupt Clear (Write)
0x018	SSPRXT0	SSP Receive Timeout Register
0x018 - 0x0FF	///	Reserved — do not write.
GENERAL PURPOSE I/O MODULES (GPIOx) BASE ADDRESS GPIOBase3 = 0xFFFDC000		
0x000	GPIOPGDR	Port G Data Register
0x004	GPIOPHDR	Port H Data Register
0x008	GPIOPGDDR	Port G Data Direction Register
0x00C	GPIOPHDDR	Port H Data Direction Register
0x010 - 0x0FF	///	Reserved — do not write.
BASE ADDRESS GPIOBase2 = 0xFFFDD000		
0x000	GPIOPEDR	Port E Data Register
0x004	GPIOPFDR	Port F Data Register
0x008	GPIOPEDDR	Port E Data Direction Register
0x00C	GPIOPFDDR	Port F Data Direction Register
0x010 - 0x0FF	///	Reserved — do not write.
BASE ADDRESS GPIOBase1 = 0xFFFDE000		
0x000	GPIOPCDR	Port C Data Register
0x004	GPIOPDDR	Port D Data Register
0x008	GPIOPCDDR	Port C Data Direction Register
0x00C	GPIOPDDDR	Port D Data Direction Register
0x010 - 0x0FF	///	Reserved — do not write.
BASE ADDRESS FOR GPIOA, GPIOBase0 = 0xFFFDF000		
0x000	GPIOPADR	Port A Data Register
0x004	GPIOPBDR	Port B Data Register
0x008	GPIOPADDR	Port A Data Direction Register
0x00C	GPIOPBDDR	Port B Data Direction Register
0x010 - 0x0FF	///	Reserved — do not write.

Table 20-1. LH79520 Status and Configuration Registers (Cont'd)

OFFSET FROM BASE ADDRESS	REGISTER NAME	REGISTER DESCRIPTION
REAL TIME CLOCK REGISTERS BASE ADDRESS RTCBase = 0xFFFE0000		
0x000	RTCDR	RTC Data Register
0x004	RTCMR	RTC Match Register
0x008	RTCSTAT / RTCEOI	Interrupt Status (Read) / Interrupt Clear (Write) Register
0x00C	RTCLR	RTC Counter Load Register
0x010	RTCCR	RTC Interrupt Control Register
0x014 - 0x0FF	///	Reserved — do not write
DMA CONTROLLER BASE ADDRESS DMABase = Stream0Base = 0xFFFE1000 (SSP RECEIVE)		
0x000	DMASourceLo	Source base address, lower 16 bits
0x004	DMASourceHi	Source base address, higher 16 bits
0x008	DMADestLo	Destination base address, lower 16 bits
0x00C	DMADestHi	Destination base address, higher 16 bits
0x010	DMAMax	Data Stream 0 Register, Maximum Count Register
0x014	DMACtrl	Control Register
0x018	DMASoCurrHi	Current Source address, higher 16 bits
0x01C	DMASoCurrLo	Current Source address, lower 16 bits
0x020	DMADeCurrHi	Current Destination address, higher 16 bits
0x024	DMADeCurrLo	Current Destination address, lower 16 bits
0x028	DMATcnt	DMA Terminal Counter
0x02C - 0x03F	///	Reserved — do not write
BASE ADDRESS Stream1Base = 0xFFFE1040 (SSP TRANSMIT)		
0x000	DMASourceLo	Source base address, lower 16 bits
0x004	DMASourceHi	Source base address, higher 16 bits
0x008	DMADestLo	Destination base address, lower 16 bits
0x00C	DMADestHi	Destination base address, higher 16 bits
0x010	DMAMax	Data Stream 0 Register, Maximum Count Register
0x014	DMACtrl	Control Register
0x018	DMASoCurrHi	Current Source address, higher 16 bits
0x01C	DMASoCurrLo	Current Source address, lower 16 bits
0x020	DMADeCurrHi	Current Destination address, higher 16 bits
0x024	DMADeCurrLo	Current Destination address, lower 16 bits
0x028	DMATcnt	DMA Terminal Counter
0x02C - 0x07F	///	Reserved — do not write
BASE ADDRESS Stream2Base = 0xFFFE1080 (EXTERNAL REQUEST 0, DREQ0)		
0x000	DMASourceLo	Source base address, lower 16 bits
0x004	DMASourceHi	Source base address, higher 16 bits
0x008	DMADestLo	Destination base address, lower 16 bits
0x00C	DMADestHi	Destination base address, higher 16 bits
0x010	DMAMax	Data Stream 0 Register, Maximum Count Register
0x014	DMACtrl	Control Register
0x018	DMASoCurrHi	Current Source address, higher 16 bits
0x01C	DMASoCurrLo	Current Source address, lower 16 bits

Table 20-1. LH79520 Status and Configuration Registers (Cont'd)

OFFSET FROM BASE ADDRESS	REGISTER NAME	REGISTER DESCRIPTION
0x020	DMADeCurrHi	Current Destination address, higher 16 bits
0x024	DMADeCurrLo	Current Destination address, lower 16 bits
0x028	DMATcnt	DMA Terminal Counter
0x02C - 0x0BF	///	Reserved — do not write
BASE ADDRESS Stream3Base = 0xFFFE10C0 (EXTERNAL REQUEST 1, DREQ1)		
0x000	DMASourceLo	Source base address, lower 16 bits
0x004	DMASourceHi	Source base address, higher 16 bits
0x008	DMADestLo	Destination base address, lower 16 bits
0x00C	DMADestHi	Destination base address, higher 16 bits
0x010	DMAMax	Data Stream 0 Register, Maximum Count Register
0x014	DMACtrl	Control Register
0x018	DMASoCurrHi	Current Source address, higher 16 bits
0x01C	DMASoCurrLo	Current Source address, lower 16 bits
0x020	DMADeCurrHi	Current Destination address, higher 16 bits
0x024	DMADeCurrLo	Current Destination address, lower 16 bits
0x028	DMATcnt	DMA Terminal Counter
0x02C - 0x0EF	///	Reserved — do not write
BASE ADDRESS DMAControlBase = 0xFFFE10F0		
0x000	DMAMask	DMA Interrupt Mask Register
0x004	DMAClr	DMA Interrupt Clear Register
0x008	DMAStatus	DMA Status Register
RESET, CLOCK AND POWER CONTROLLER BASE ADRESS RCPCBase = 0xFFFE2000		
0x000	RCPCCtrl	RCPC Control
0x004	///	Reserved — do not write.
0x008	RCPCRemapCtrl	Remap Control
0x00C	SoftReset	Soft Reset
0x010	ResetStatus	Reset Status
0x014	ResetStatusClear	Reset Status Clear
0x018	HclkPrescale	HCLK Prescaler
0x01C	CpuClkPrescale	ARM Core Clock Prescaler
0x020	///	Reserved — do not write.
0x024	PeriphClkCtrl	Peripheral Clock Control
0x028	PeriphClkCtrl2	Peripheral Clock Control 2
0x02C	AHBClkCtrl	AHB Clock Control
0x030	PerpihClkSelect	Peripheral Clock Select
0x034	PeriphClkSelect2	Peripheral Clock Select 2
0x038	PWM0Prescale	PMW0 Prescaler
0x03C	PWM1Prescale	PWM1 Prescaler
0x040	LCDClkPrescale	LCD Clock Prescaler (CLCDC
0x044	SSPClkPrescale	SSP Clock Prescaler (SSP)
0x048 - 0x07F	///	Reserved — do not write.
0x080	IntConfig	External Interrupt Configuration

Table 20-1. LH79520 Status and Configuration Registers (Cont'd)

OFFSET FROM BASE ADDRESS	REGISTER NAME	REGISTER DESCRIPTION
0x084	IntClear	External Interrupt Clear
0x088	CoreClkConfig	ARM Core Clock Configuration
WATCHDOG TIMER BASE ADDRESS WDTBase = 0xFFFE3000		
0x000	WDCTLR	Watchdog Control Register
0x004	WDCNTR	Watchdog Counter Reset
0x008	WDTSTR	Watchdog Register
0x00C	WDCNT0	WDT Counter Section 0
0x010	WDCNT1	WDT Counter Section 1
0x014	WDCNT2	WDT Counter Section 2
0x018	WDCNT3	WDT Counter Section 3
HR-TFT LCD TIMING CONTROLLER BASE ADDRESS HRTFTCBase = 0xFFFE4000		
0x000	HRTFTCSetup	HR-TFT LCD Timing Controller Setup Register
0x004	HRTFTCControl	HR-TFT LCD Timing Controller Control Register
0x008	HRTFTCTiming1	HR-TFT LCD Timing Controller Timing Register 1
0x00C	HRTFTCTiming2	HR-TFT LCD Timing Controller Timing Register 2
INPUT/OUTPUT CONTROL BASE ADDRESS IOCONBase = 0xFFFE5000		
0x000	MemMux	Memory Interface Multiplexing
0x004	LCDMux	LCD Pin Multiplexing
0x008	MiscMux	Miscellaneous Multiplexing
0x00C	DMAMux	DMA Multiplexing
0x010	UARTMux	UART Multiplexing
0x014	SSPMux	SSP Multiplexing
0x018	///	Reserved — do not write.
MIRRORED BASE ADDRESS FOR VECTORED INTERRUPT CONTROLLER REGISTERS BASE ADDRESS VICBaseMirror = 0xFFFF0000 (SEE VICBase)		
0x000 - 0xFFF	(mirrored)	(The registers at VICBase are mirrored here)
STATIC MEMORY CONTROLLER BASE ADDRESS SMCREGBase = 0xFFFF1000		
0x000	SMCBCR0	Static Memory Bank 0 Configuration Register
0x004	SMCBCR1	Static Memory Bank 1 Configuration Register
0x008	SMCBCR2	Static Memory Bank 2 Configuration Register
0x00C	SMCBCR3	Static Memory Bank 3 Configuration Register
0x010	SMCBCR4	Static Memory Bank 4 Configuration Register
0x014	SMCBCR5	Static Memory Bank 5 Configuration Register
0x018	SMCBCR6	Static Memory Bank 6 Configuration Register
0x01C	///	Reserved — do not write.
SYNCHRONOUS DYNAMIC RAM CONTROLLER BASE ADDRESS SDRAMBase = 0xFFFF2000		
0x000	SDRCConfig0	SDRAM Memory Bank 0 Configuration Register 0
0x004	SDRCConfig1	SDRAM Memory Bank 1Configuration Register 1
0x008	SDRCRefTimer	SDRAM Refresh Timer Register
0x00C	SDRCWBTimeout	SDRAM Write Buffer Time-Out Register

Table 20-1. LH79520 Status and Configuration Registers (Cont'd)

OFFSET FROM BASE ADDRESS	REGISTER NAME	REGISTER DESCRIPTION
COLOR LCD CONTROLLER BASE ADDRESS CLCDCBase = 0xFFFF4000		
0x000	LCDTiming0	Horizontal Axis Panel Control
0x004	LCDTiming1	Vertical Axis Panel Control
0x008	LCDTiming2	Clock and Signal Polarity Control
0x00C	///	Reserved — do not write.
0x010	LCDUPBASE	Upper Panel Frame Base address
0x014	LCDLPBASE	Lower Panel Frame Base address
0x018	LCDINTRENABLE	Interrupt Enable Mask
0x01C	LCDControl	Panel Pixel Parameters
0x020	LCDStatus	Raw interrupt Status
0x024	LCDInterrupt	Final Masked Interrupt Status
0x028	LCDUPCURR	Upper Panel Current address Value
0x02C	LCDLPCURR	Lower Panel Current address Value
0x030 - 0x1FF	///	Reserved — do not write.
0x200 - 0x3FC	LCDPalette	256 × 16-bit Color Palette
VECTORED INTERRUPT CONTROLLER (ALSO SEE VICBaseMirrored) BASE ADDRESS VICBase = 0xFFFFF000		
0x000	IRQStatus	Status of Interrupts 0-31 after IRQ masking
0x004	FIQStatus	Status of Interrupts 0-31 after FIQ masking
0x008	RawInterrupt	Status of Interrupts 0-31 to the Interrupt Controller
0x00C	IntSelect	FIQ/IRQ Select for each Interrupt Source
0x010	IntEnable	Interrupt Source mask
0x014	IntEnableClear	Use to Clear bits in the IntEnable Register
0x018	SoftInt	Use to generate interrupts under program control
0x01C	SoftIntClear	Use to clear bits in the SoftInt Register
0x020 - 0x02F	///	Reserved — do not write.
0x030	VectorAddr	Interrupt Vector address
0x034	DefVectAddr	Default Vector address
0x038 - 0x0FF	///	Reserved — do not write.
0x100	VectAddr0	Interrupt Vector address 0
0x104	VectAddr1	Interrupt Vector address 1
0x108	VectAddr2	Interrupt Vector address 2
0x10C	VectAddr3	Interrupt Vector address 3
0x110	VectAddr4	Interrupt Vector address 4
0x114	VectAddr5	Interrupt Vector address 5
0x118	VectAddr6	Interrupt Vector address 6
0x11C	VectAddr7	Interrupt Vector address 7
0x120	VectAddr8	Interrupt Vector address 8
0x124	VectAddr9	Interrupt Vector address 9
0x128	VectAddr10	Interrupt Vector address 10
0x12C	VectAddr11	Interrupt Vector address 11
0x130	VectAddr12	Interrupt Vector address 12
0x134	VectAddr13	Interrupt Vector address 13

Table 20-1. LH79520 Status and Configuration Registers (Cont'd)

OFFSET FROM BASE ADDRESS	REGISTER NAME	REGISTER DESCRIPTION
0x138	VectAddr14	Interrupt Vector address 14
0x13C	VectAddr15	Interrupt Vector address 15
0x104 - 0x1FF	///	Reserved — do not write.
0x200	VectCntl0	Interrupt Vector Control 0
0x204	VectCntl1	Interrupt Vector Control 1
0x208	VectCntl2	Interrupt Vector Control 2
0x20C	VectCntl3	Interrupt Vector Control 3
0x210	VectCntl4	Interrupt Vector Control 4
0x214	VectCntl5	Interrupt Vector Control 5
0x218	VectCntl6	Interrupt Vector Control 6
0x21C	VectCntl7	Interrupt Vector Control 7
0x220	VectCntl8	Interrupt Vector Control 8
0x224	VectCntl9	Interrupt Vector Control 9
0x228	VectCntl10	Interrupt Vector Control 10
0x22C	VectCntl11	Interrupt Vector Control 11
0x230	VectCntl12	Interrupt Vector Control 12
0x234	VectCntl13	Interrupt Vector Control 13
0x238	VectCntl14	Interrupt Vector Control 14
0x23C	VectCntl15	Interrupt Vector Control 15
0x240 - 0x30B	///	Reserved — do not write.
0x30C	ITOP1	Interrupt Test Output 1
0x310 - 0xFFC	///	Reserved — do not write.

NOTE: /// = Reserved

Chapter 21
Glossary

AHB
Advanced High-Performance Bus. The main bus in the LH79520.

AMBA
Advanced Microprocessor Bus Architecture.

APB
Advanced Peripheral Bus.

Big-endian byte order
See Little-endian byte order.

Binary
A system of two. A term for a numbering scheme in which there are only two possible values for any digit in a number: 0 and 1.

Block
In Flash Memory terminology, a Block (or sector) is a group of memory cells that need to be erased together. Blocks are usually 64KB in size, but other sizes exist (8KB boot blocks for example). Once a byte within a block has been programmed, it is necessary to erase the entire Block containing that byte before the byte can be programmed again.

BPP
Bits-Per-Pixel.

BW
Bandwidth.

B/W
Black-and-white, or grayscale.

Byte
In most computer systems, a byte is a unit of data containing eight binary digits (bits), Always shown with the MSB at the left and the LSB on the right.

Byte Lane
A data path that is one byte (8 bits) wide, analogous to a highway with eight lanes of traffic.

Chip
A packaged integrated circuit device.

CPSR

Current Program Status Register. In ARM architecture, the CPSR is the register that stores the condition code bits.

Dice, more commonly: Die

An unpackaged integrated circuit device.

DNM (Do Not Modify fields)

Means the value must not be altered by software. DNM fields read as unpredictable values and must only be written with the same value read from the same field.

ENDEC

ENCoder/DECoder.

Endianness

See Big-endian byte order.

Half-Word

In the context of 32-bit SoCs such as the LH79520, a half-word is an ordered pair of bytes totaling 16-bits. Half-words are always shown with the MSB at the left and the LSB on the right.

IrDA

A serial, half-duplex optical communications protocol sponsored by the InfraRed Data Association.

ISR

Interrupt Service Routine.

k

'kilo-' prefix. Used with a modifier, such as kHz or kW.

K

Kilobit. A unit of measurement signifying 1,024 bits of memory, as in 64K of memory

KB

In binary numbering systems, a unit of measurement representing a quantity of 1024 Bytes. In common parlance this is approximately one thousand bytes.

LPP

Lines Per Panel. The quantity of active lines per LCD screen, or 'panel'.

Little-endian byte order

Little-endian and big-endian byte ordering describe the sequence in which bytes are stored to computer memory. The LH79520 uses little-endian byte ordering (storage). Assuming that the storage operations proceed upward in memory (from lower- to higher-numbered addresses), big-endian storage means that the most-significant byte is stored at a lower-numbered memory location, followed by the next byte at the next higher-numbered location. In little-endian systems, this sequence is reversed and the least-significant byte is stored first. Figure 21-1 illustrates little-endian and big-endian byte ordering.

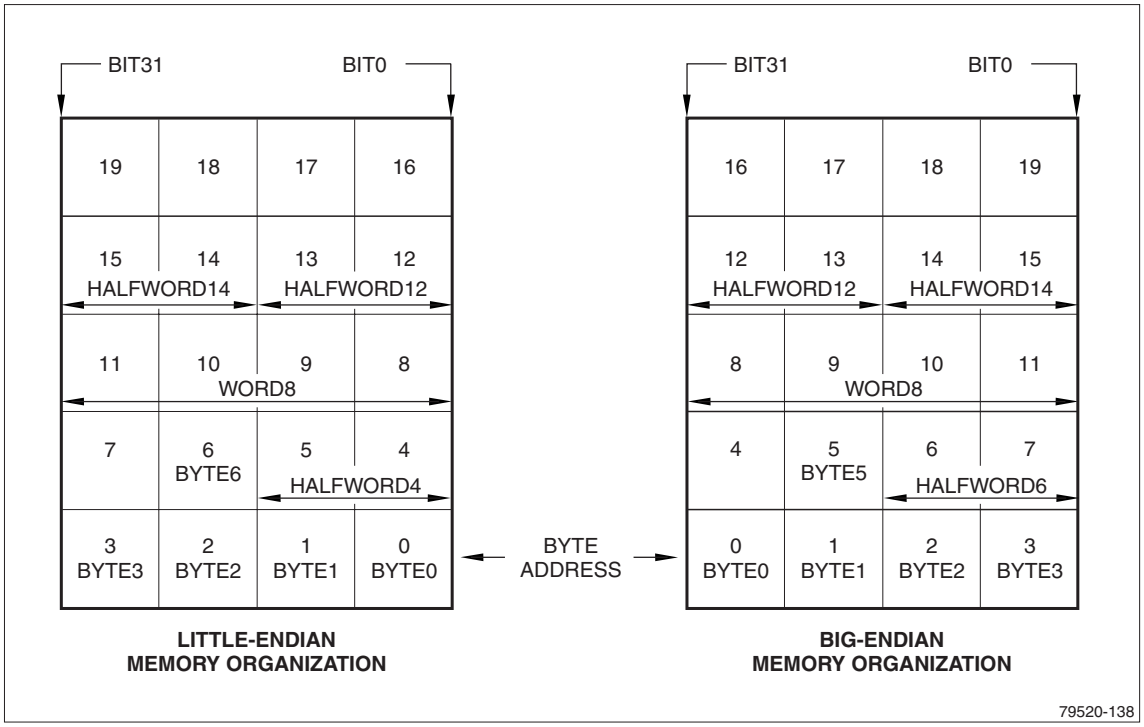


Figure 21-1. Byte Ordering

LSB

Least-significant bit of a byte, half-word or word.

LUT

Look-Up Table.

M

Megabit. A unit of measure.

Maskable

Having a provision for masking. Capable of being blocked or disabled. Maskable interrupts can be masked (blocked, disabled), or unmasked (unblocked, enabled). See Nonmaskable.

MB

Megabyte. A unit of measure.

MCU

Microcontroller. A single-chip microprocessor system of sufficient complexity so as to require only minimal additional support devices in order to perform its intended function.

Merging Write Buffer

A merging write buffer compacts writes of all widths (byte, half-word, and word) into quad-word bursts which can be efficiently transferred to SDRAM.

MSB

Most-significant bit of a byte, half-word or word.

MWords

Mega-Words, millions of words.

Nonmaskable

Having no provision for masking; not capable of being blocked or disabled. Nonmaskable interrupts cannot be masked (blocked, disabled). See Maskable.

NRZ

Non-Return-to-Zero. A method of encoding binary serial data.

Non-Volatile Memory

A memory technology that retains its contents when power is removed. Examples are ROM and Flash. Also see Volatile Memory.

PC

The Program Counter in the ARM720T core.

Pixel

Picture Element. The smallest controllable unit of a matrix LCD display.

RO

Read Only. Writes will have no effect.

RW

Read or Write. May be read or written.

RZI

Return-to-Zero, Inverted. A method of encoding binary serial data.

SIR

Serial InfraRed.

SWI

Software Interrupt.

Volatile memory

A general term for any memory technology that loses its contents when power is removed. Examples are RAM, SRAM and SDRAM. See Non-Volatile Memory.

VBP

Vertical Back Porch. The quantity of inactive lines at the start of a frame, after the vertical synchronization period.

VFP

Vertical Front Porch. The quantity of inactive lines at the end of a frame, before the vertical synchronization period.

VSW

Vertical Synchronization (Pulse) Width. The quantity of horizontal synchronization lines fed to an LCD panel.

Word

In the context of 32-bit SoCs such as the LH79520, a word is an ordered set of four bytes, totaling 32 bits. Words are always shown with the MSB at the left and the LSB on the right.

WO

Write Only. Reads will return unpredictable results.

Index

D

E

F

G

H

N

O

P

R

S

T

U

V

W

ANNEX A: Disclaimers (11)

1. t001dis100.fm: General (DS, AN, UM, errata)

General — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

2. t001dis101.fm: Right to make changes (DS, AN, UM, errata)

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

3. t001dis102.fm: Suitability for use (DS, AN, UM, errata)

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in medical, military, aircraft, space or life support equipment, nor in applications where failure or malfunction of a NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

4. t001dis103.fm: Applications (DS, AN, UM, errata)

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

5. t001dis104.fm: Limiting values (DS)

Limiting values — Stress above one or more limiting values (as defined in the Absolute Maximum Ratings System of IEC 60134) may cause permanent damage to the device. Limiting values are stress ratings only and operation of the device at these or any other conditions above those given in the Characteristics sections of this document is not implied. Exposure to limiting values for extended periods may affect device reliability.

6. t001dis105.fm: Terms and conditions of sale (DS)

Terms and conditions of sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, including those pertaining to warranty, intellectual property rights infringement and limitation of liability, unless explicitly otherwise agreed to in writing by NXP Semiconductors. In case of any inconsistency or conflict between information in this document and such terms and conditions, the latter will prevail.

7. t001dis106.fm: No offer to sell or license (DS)

No offer to sell or license — Nothing in this document may be interpreted or construed as an offer to sell products that is open for acceptance or the grant, conveyance or implication of any license under any copyrights, patents or other industrial or intellectual property rights.

8. t001dis107.fm: Hazardous voltage (DS, AN, UM, errata; if applicable)

Hazardous voltage — Although basic supply voltages of the product may be much lower, circuit voltages up to 60 V may appear when operating this product, depending on settings and application. Customers incorporating or otherwise using these products in applications where such high voltages may appear during operation, assembly, test etc. of such application, do so at their own risk. Customers agree to fully indemnify NXP Semiconductors for any damages resulting from or in connection with such high voltages. Furthermore, customers are drawn to safety standards (IEC 950, EN 60 950, CENELEC, ISO, etc.) and other (legal) requirements applying to such high voltages.

9. t001dis108.2.fm: Bare die (DS; if applicable)

Bare die (if applicable) — Products indicated as Bare Die are subject to separate specifications and are not tested in accordance with standard testing procedures. Product warranties and guarantees as stated in this document are not applicable to Bare Die Products unless such warranties and guarantees are explicitly stated in a valid separate agreement entered into by NXP Semiconductors and customer.

10. t001dis109.fm: AEC unqualified products (DS, AN, UM, errata; if applicable)

AEC unqualified products — This product has not been qualified to the appropriate Automotive Electronics Council (AEC) standard Q100 or Q101 and should not be used in automotive critical applications, including but not limited to applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is for the customer’s own risk.

11. t001dis110.fm: Suitability for use in automotive applications only (DS, AN, UM, errata; if applicable)

Suitability for use in automotive applications only — This NXP Semiconductors product has been developed for use in automotive applications only. The product is not designed, authorized or warranted to be suitable for any other use, including medical, military, aircraft, space or life support equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer’s own risk.