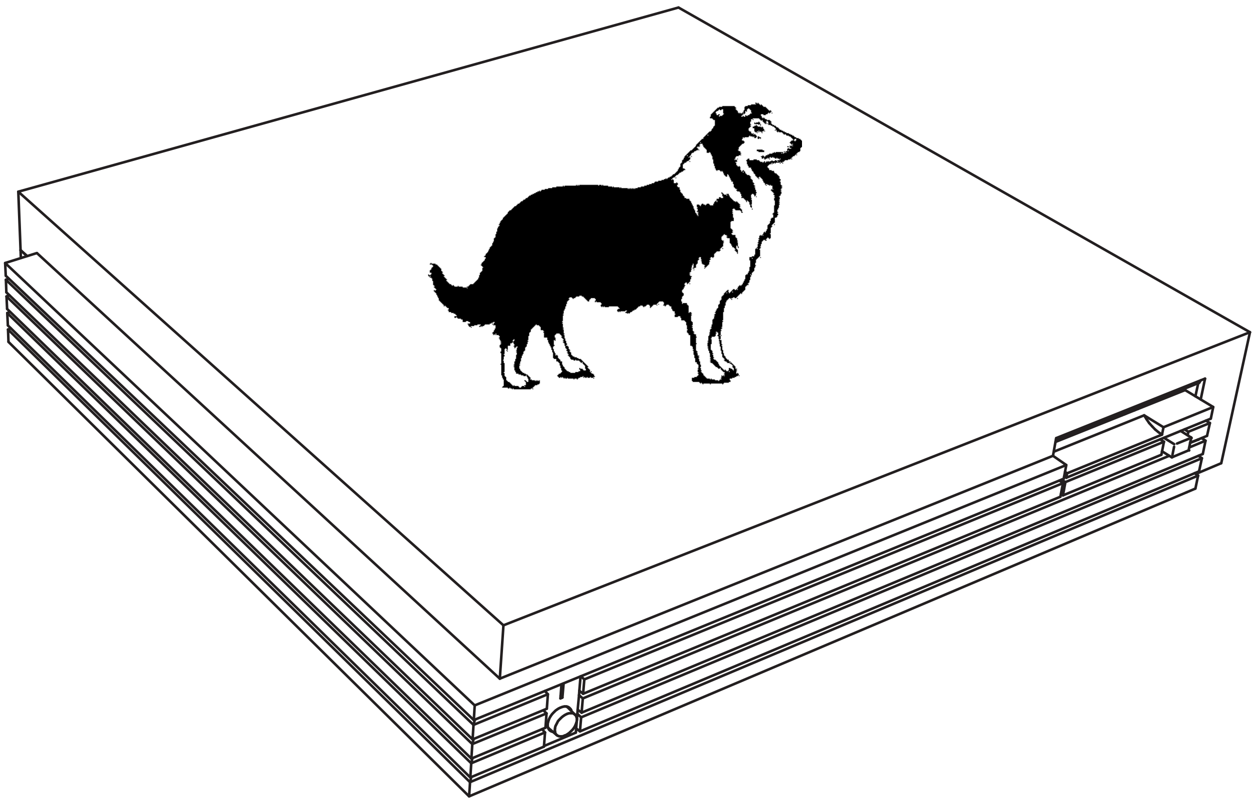
For:billk
Printed on:Tue, May 18, 1999 16:18:14
From book:lasi_ers
Document:cover_page
Last saved on:Tue, May 18, 1999 16:08:55
Document:TOC
Last saved on:Tue, May 18, 1999 16:11:27
Document:LOT
Last saved on:Tue, May 18, 1999 16:11:25
Document:LOF
Last saved on:Tue, May 18, 1999 16:11:25
Document:intro
Last saved on:Wed, Mar 17, 1999 16:18:18
Document:overview
Last saved on:Wed, Mar 17, 1999 16:20:32
Document:scsi
Last saved on:Wed, Mar 17, 1999 16:25:23
Document:lan
Last saved on:Wed, Mar 17, 1999 16:32:40
Document:Parallel
Last saved on:Wed, Mar 17, 1999 16:52:13
Document:audio
Last saved on:Wed, Mar 17, 1999 17:30:43
Document:Serial
Last saved on:Wed, Mar 17, 1999 17:33:39
(...)

June 17, 1992



712 I/O Subsystem ERS

Revision 1.1
12 February 1993

Notice

The information contained in this document is subject to change without notice.

HEWLETT–PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THE MATERIAL. INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Hewlett–Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with furnishing, performance, or use of this material.

Hewlett–Packard sassumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett–Packard.

1 INTRODUCTION

1.1 Objectives

The primary objective for the 712 I/O subsystem is to provide a core set of I/O functionality that is consistent with the cost and performance goals of the 712 box.

The primary objective for this document is to describe the software interface to 712 I/O with enough detail to allow driver development. This document (along with referenced documents) contains enough detail to ascertain the intended functionality of each block on the Lasi chip down to the register level.

This version of the document (1.1) should contain an accurate description of the 712 I/O Subsystem functionality for Lasi 1.0.

1.2 Feature Summary

The following is a list of the features of the 712 I/O Subsystem:

- SCSI (DMA)
- LAN (DMA)
- Parallel Printer Interface (DMA)
- CD quality Stereo Audio (DMA)
- serial RS232 port
- PS/2 Keyboard and mouse interface
- Real Time Clock
- 256K bytes of Flash EPROM
- Power system support
- PC floppy disk drive (DMA–optional)
- telephone interfaces (optional)
- Personality Card (optional)

1.3 Design Team

<Deleted from public version of document>

2 OVERVIEW

2.1 Functional Organization

The heart and soul of the 712 I/O Subsystem is the Lasi chip. The vast majority of the I/O functionality in 712 is either completely implemented on Lasi or accessed through Lasi. The one exception to this is the optional personality card which has its own GSC interface. The personality card is intended for IBM Token Ring or an additional I/O device implemented with a second Lasi chip. The Block diagram in figure 1 shows the organization of the 712 I/O Subsystem.

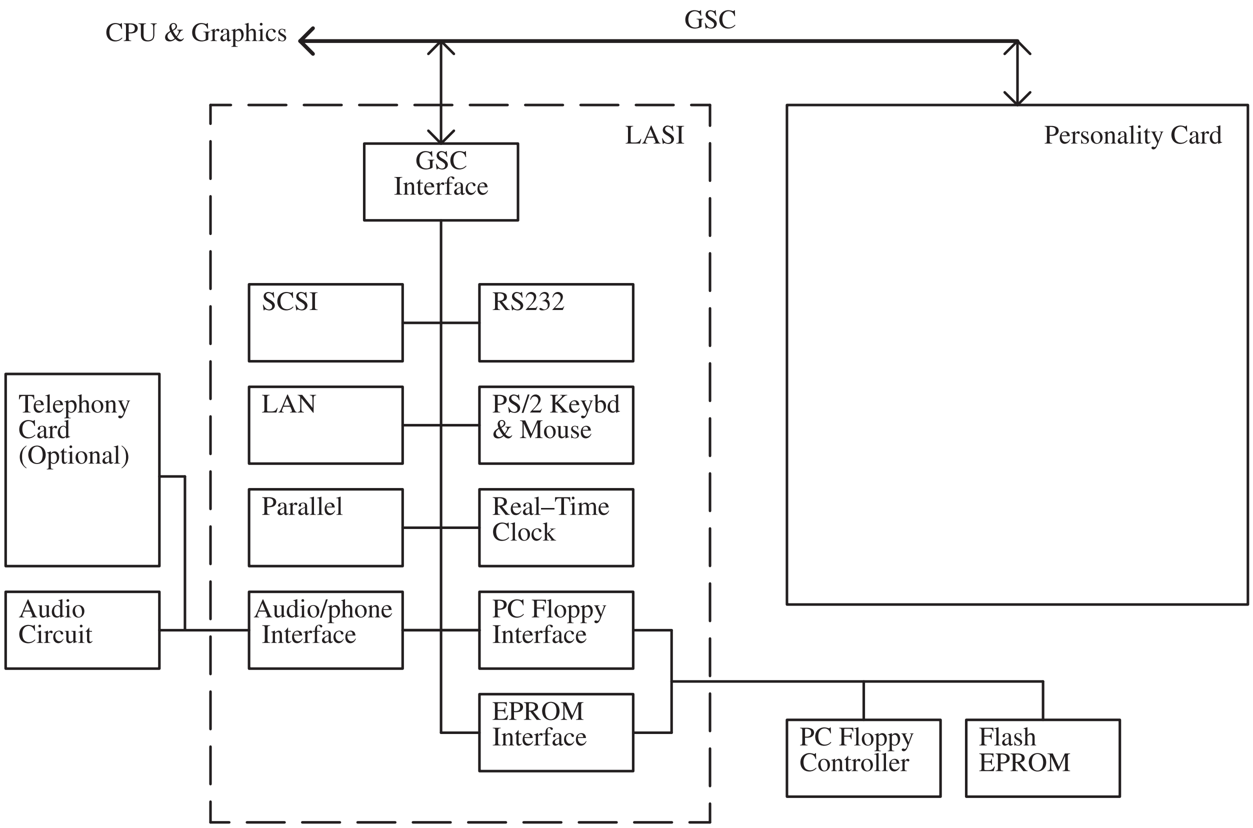


Figure 1. I/O Subsystem Block Diagram

2.2 Personality Card

The personality card allows some flexibility in Gecko I/O while maintaining the lowest possible factory cost for the base system. The current plan is to have two different flavors of the personality card. The first card would use an ASIC (Wax) to interface GSC to the TI Token Ring chip set. The second would use another Lasi chip to implement a second copy of one or more of the I/O devices implemented in Lasi.

2.2.1 Lasi Configurability

The Lasi chip has four configuration bits that allow enough flexibility to use a second Lasi on the personality card. The SPACE[1:0] bits control which of four possible base addresses are decoded by Lasi. The RESET_SLAVE bit allows to another GSC device to drive the RESETL signal. The ARB_SLAVE bit allows another device control GSC arbitration. These bits are all set during reset by pulling configuration pins either high or low through a resistor. This configuration flexibility allows more than one Lasi to reside on the same GSC. It also provides the flexibility required to make Lasi useable by some non–Gecko systems. The configuration registers in the second Lasi can be read by PDC to identify which of Lasi's I/O functionality can be used (ie. have connectors) on the second Lasi chip.

2.3 Telephone Support

Gecko supports an optional board that provides simultaneous access to two telephone lines. One can be used for voice, and the other for either voice or data. This functionality is provided on a separate board to reduce the cost of the base configuration and to accommodate the differences in international phone systems. This board has a dedicated interface that is not intended to support any other functionality.

3 SCSI

3.1 Introduction

SCSI (Small Computer System Interface) is a system level interface bus used to connect disc drives, tape drives, and other I/O devices to a computer system. Numerous workstations today support this bus standard, as SCSI is becoming the defacto disc interface standard. SCSI–1 consists of 8 bits of data at up to a 4 MB/sec for synchronous transfer rate. SCSI–2, however, allows up to a 10 MB/sec transfer rate with an 8 bit bus, up to 20 MB/sec with a 16 bit bus and up to 40 MB/sec with an extended 32 bit data bus.

3.2 Overview

The 712 I/O subsystem will support SCSI–2 specification and HPCS (HP Common SCSI) command set. However, the 712 I/O subsystem will only support the 8 bit data bus SCSI configuration and will support neither the 16 nor 32 bit data bus configuration.

The 712 built–in SCSI–2 port is implemented using a NCR 53C710 SCSI I/O Processor macrocell inside the LASI chip, referred to as NCR 710 Macrocell. The NCR 710 Macrocell will be configured in 8 bit single–ended SCSI–2 mode and, with a 40MHz SCSI clock, can transfer data at a maximum data rate of 5 MB/sec on the SCSI bus.

The NCR 53C710 Host Bus interface contains a 32b DMA engine which supports burst transfer rates up to 66 MB/sec, and a ”script processor,” which fetches its own commands and performs SCSI transactions with minimal host processor intervention.

For a detailed description, refer to the NCR 53C710 SCSI I/O Processor Data Manual rev 1.0.

3.3 Features

- Full NCR 710 SCSI and ”SCRIPT” Compatibility
- SCSI–2 Single–Ended direct connection without external transceivers
- 40 MHz SCSI Clock (SCLK)
- Supports system frequencies up to 66MHz
- Snoop bit 1 SC[1] is connected for run time interrupt

3.4 Signal Definitions

LASI supports the 8 bit Single–Ended SCSI–2 bus signals: scsd_L[7:0], scsdbp_L, scscd_L, scsio_L, scsmsg_L, scsreq_L, scsack_L, scsbsy_L, scssel_L, scsatn_L, and scsrst_L.

The NCR 710 Differential Support Lines (14 signals) are not supported on the GECKO built–in LASI pins.

Please see Appendix B: LASI Pin Definitions for detailed LASI signal list and LASI naming.

3.5 SCSI–2 Fast Single–Ended Support

The NCR 710 macrocell itself is capable of data transfer rates up to 10 MB/sec, however, LASI will not support SCSI–2 Single–Ended timing at greater than a 5 MB/sec transfer rate. It may be possible for very restricted configurations to run faster, but we have no plans to characterize or ensure this operation.

3.6 SCSI–2 Fast Differential Support

The NCR 710 Differential Support Lines (14 signals) are not supported on the GECKO built–in LASI pins. However, the NCR 710 Macrocell will contain the internal circuitry to generate these signals for future macrocell applications. The NCR 710 Macrocell itself is capable of data transfer rates up to 10 MB/sec in 8 bit Fast Differential mode. The rest of LASI and the GSC will be designed to support 10 MB/sec data rates to allow for future LASI variants.

3.7 Clock Rates

Unlike the NCR 53C700, the 53C710 has separate clocks for the Host Bus Interface and the SCSI Bus Interface. The host–side Bus Clock (BCLK) operates up to 33MHz. Lasi has a fixed 1/2 ratio between its internal C710's BCLK and the system clock, allowing system clock frequencies up to 66 MHz.

The SCSI Clock (SCLK) will operate at a fixed 40MHz.

3.8 NCR 710 Mode Bits

The NCR 710 Bus Mode Select (BS_L) signal will be hardwired to logic 0 (Synchronous (68040–like) host bus mode). The Big/Little Endian Select (BIG_LIT_L) signal will be hardwired to logic 1 (Big endian byte order).

3.9 NCR 710 Revision Code

The LASI chip NCR 710 Macrocell implementation is based on NCR 53C710 Rev. D (Rev. 4). The chip revision field reported by the chip should be "2".

3.10 Interrupts

LASI implements the Running Interrupt (or Interrupt–on–the–Fly Interrupt) in addition to the normal NCR 710 IRQ line. The logic equation for the NCR 710 Macrocell interrupt is:

NCR710_IRQ_L ==> IPR[9] (”normal” interrupt)
NCR710_SC[1]_L ==> IPR[21] (Interrupt–on–the–Fly)

IRQ is enabled through the NCR710 SIEN register.

3.11 Register Set

The SCSI Register Set as viewed from the GSC Bus is shown in **Table 1.**

All addresses in the memory map are specified relative to the relocatable Lasi base address. Byte addresses are ”Big Endian” byte numbering mode (Byte 0 is most significant and on the left). Note that this byte numbering is **NOT** consistent with the ”Little Endian” byte addressing used in the EISA SCSI Host Adapter card (25525A) using the NCR 53C710. Bit numbering is ”Little Endian” (bit 0 is least significant and on the right).

The GSC may access registers using byte or 32 bit word addresses. LASI does not require byte swapping for registers which have more than one byte; the ASP chip used in the 720/730/750 systems did.

Table 1. SCSI Register Set

Address (Hex)	31 Bit Numbering 0			
	Byte 0	Byte 1	Byte 2	Byte 3
000	SCSI Reset (W)			
004	RESERVED			
⋮				
0FF	RESERVED			
100	SIEN (R/W)	SDID (R/W)	SCNTL1 (R/W)	SCNTL0 (R/W)
104	SOCL (R/W)	SODL (R/W)	SXFER (R/W)	SCID (R/W)
108	SBCL (R)	SBDL (R)	SIDL (R)	SFBR (R)
10C	SSTAT2 (R)	SSTAT1 (R)	SSTAT0 (R)	DSTAT (R)
110	DSA (R/W)			
114	CTEST3 (R)	CTEST2 (R)	CTEST1 (R/W)	CTEST0 (R/W)
118	CTEST7 (R)	CTEST6 (R)	CTEST5 (R)	CTEST4 (R)
11C	TEMP (R/W)			
120	LCRC (R/W)	CTEST8 (R/W)	ISTAT (R/W)	DFIFO (R/W)
124	DCMD (R/W)	DBC (R/W)		
128	DNAD (R/W)			
12C	DSP (R/W)			
130	DSPS (R/W)			
134	SCRATCH (R/W)			
138	DCNTL (R/W)	DWT (R/W)	DIEN (R/W)	DMODE (R/W)
13C	ADDER (R)			
⋮				
1FF	RESERVED			

3.11.1 SCSI Reset Register

Name: SCSI Reset
Address: Word 000, Byte 0
Access: Write Only

A write to this register causes a directed reset to the SCSI subsystem.

3.11.2 NCR 710 Internal Registers

Name: See table.
Address: Words 100 to 13C
Access: See table.

Registers 100 to 13F are implemented inside the NCR53C710. See the NCR 53C710 Data Manual for the definitions of each field within each register.

4 LAN

4.1 Introduction

The712 I/O Subsystem implements a local area network (LAN) to the 802.3/Ethernet standard. Ethernet is a 10 Mbit/s packet–switched serial interface employing Carrier Sense Multiple Access/ Collision Detection (CSMA/CD).

4.2 References

- Intel 82596CA High–Performance 32–Bit Local Area Network Coprocessor
- Intel 82596 User's Manual (Order Number: 296443–0001)
- Intel 82596 Data Sheet Supplement November 1989
- Intel Microcommunications Applications, vol. 1 & 2 (Order Number: 231658)

4.3 Nomenclature and Conventions

Note than an Intel "word" is 16 bits, and a PA–RISC word is 32 bits. For both Intel and GSC, bits are numbered 31 (most significant) to 0 (least significant) from left to right.

4.4 Overview

The 712 built–in LAN is divided into two main physical blocks: the 82596 controller megacell and the 82C503 (or equivalent) interface. The controller is a megacell implementation of the stand–alone 82C596CA LAN controller from Intel. Please refer to the 82C596 documentation for detailed explanations of megacell functionality.

Lasi will typically be used with the 82C503, an intelligent LAN interface transceiver which provides both the MAU interface for conventional coaxial Ethernet and a direct interface to a twisted pair 10BASE-T network. This device will automatically detect whether a connection has been made to the MAU or twisted pair port and auto–configure to talk only to that port. No programmable interface will be provided in 712 to override the automatic selection. The automatic port selection capability will require the hub to be fully IEEE 802.3i–1990 compliant, providing the link integrity function as defined in the spec.

The LAN is also divided into two main functional blocks: the backplane interface to GSC and the frontplane interface to the network cable. The 82596 megacell depends on the Lasi GSC Interface to provide a proper backplane interface between the 32 bit multiplexed GSC bus and the non–multiplexed address and data busses on the megacell. The frontplane interface to the MAU and

10BASE-T connectors is provided by the 82C503. We advise the reader to consult the specification for this device for further information.

The 82596CA has a four channel DMA controller which allows it to communicate directly with the main memory via the Lasi internal bus and the GSC interface. The four channels are: CU (transmit header), TXD (transmit data), RU (receive header) and RXD (receive data). Following is a brief description of the shared memory model, repeated without permission from the Intel 82596 User's Manual, section 2.4.3. We strongly encourage the reader to consult this book for information about the chip:

> "To off–load the CPU the 82596 implements a shared memory communication system with the host CPU. The 82596 and CPU do not communicate directly, but rather through a shared system memory 'mailbox.' This allows the CPU to place commands in the mailbox, activate **Channel Attention** to notify the 82596 of delivery, and return to its other processing chores. The 82596 checks its mailbox in response, retrieves and interprets the commands, and executes them without further CPU interaction. After the 82596 completes its tasks it places the results in system memory and updates the mailbox. Then the 82596 uses its **interrupt** line to notify the CPU of the presence of return mail..."

For normal DMA operations, the Lasi GSC interface arbitrates for GSC on the 82596 megacell's behalf, manages the address valid/ready handshake, and synchronizes the address and data buses via the transceivers.

4.5 Transaction Types

Apart from normal communication via shared memory, a limited number of operations directly to the megacell are available:

Note: all addresses are stated relative to the relocatable Lasi base address.

4.5.1 Channel Attention

To issue a Channel Attention to the 82596CA, do a word write to the LAN Channel Attention Register, at 008. No data is associated with this operation.

4.5.2 Port Access

The 82596CA's "CPU Port" provides 4 functions:

- alternate System Configuration Pointer (SCP) address
- Dump command
- software reset
- self–test

Even though the megacell is in CA mode, the port accesses look like the part is in DX mode: the data needs to be on the lower 16 bits of the data bus for both port accesses. This change should be sufficient to make the CA mode look like DX mode to software, maintaining software compatibility with previous 700 implementations.

The port is memory mapped. It is accessed via two consecutive 32–bit word writes to 004. This is due to the fact that only 16 bits are read at a time on this port when operating in big endian mode. See the 82596 User's Manual, pp. 5–18:5–20 for instructions on how to encode the data.

4.6 Reset

The 82596 has a hardware reset, CPU Port reset (see Port Access section above), and a software reset. In addition, software can control the lan bit in the IO_RESET register at address offset 0x10C00C to force a hard reset of the LAN.

4.6.1 Consequences

Resetting the megacell does NOT trigger self–test.

4.6.1.1 Hardware Reset

The hardware reset causes the megacell to immediately cease all activity; the CU and RU become IDLE and clear all internal requests.

4.6.1.2 CPU Port Reset

The CPU Port reset causes the megacell to immediately cease all activity and execute a software reset.

4.6.1.3 Software Reset

The CU performs the following on recognition of a software reset:

- Terminates DMA activity.
- Writes zeros to the SCB Command word.
- Triggers a hardware reset.

4.6.2 Protocol

4.6.2.1 Hardware Reset

After power up, the 82596 requires a hardware reset. Lasi will automatically perform the reset as a result of the PON signal (for a primary Lasi) or the GSC RESETL signal (for a secondary Lasi), which by definition is asserted on power up. Code must wait for 10 system clocks and 5 transmit clocks (20 processor clocks + 0.5 microseconds for 10 Mbit/sec LAN) before doing a Channel Attention after a hardware reset. The LAN subsystem hardware will *not* check for the proper interval. **Code can cause a hardware reset by writing to 000.** This must be implemented by the Lasi GSC Interface block. A Channel Attention following a hardware reset will cause the 82596 to access the

SCP, which is located by default at 0x00FFFFF4 (or at an alternative address selected via the CPU Port). After Channel Attention the 82596 will read the sysbus byte and begin the initialization process.

4.6.2.2 CPU Port Reset

For information on CPU Port operations, see Port Access section above. A *Channel Attention* following a *CPU Port reset* will cause the 82596 to access the SCP, which is located at 0x00FFFFF4 (or at an alternative address selected via the CPU Port). After Channel Attention the 82596 will read the sysbus byte and begin the initialization process. The CPU must wait for 10 system clocks and 5 transmit clocks (20 processor clocks + 0.5 microseconds for 10 Mbit/sec LAN) before issuing another Channel Attention to the 82596. The LAN subsystem hardware will *not* check for the proper interval.

4.6.2.3 Software Reset

A software reset is available through bit 7 of the control command word in the SCB. It can be used after the 82596 has been initialized and has the ISCP and SCP addresses.

4.7 Interrupts

Lasi provides a uniform interface for all I/O interrupts, including LAN (see interrupt section of this document). Interrupts are generated by one or more of the following events (from page 3–46 of the User's Manual):

- Execution of a Command Block with its *I* bit set (*CX* interrupt).
- Reception of a frame (*RU* interrupt).
- The CU becoming not active (*CNA* interrupt).
- The RU becoming not ready (*RNR* interrupt).

For more information about what these actually mean, consult section 5.3 of the User's Manual.

For compatibility with existing drivers, LASI assumes that the interrupt line driven by the megacell will be active low. It follows that the interrupt bit in the SYSBUS byte should be a 1. Also, since the default for the interrupt line is active high after reset, interrupts should be disabled via the LASI interrupt mask register during reset. They should be enabled only after the LAN has a chance to read the SYSBUS byte in response to a CA. Otherwise, a spurious interrupt may be generated.

4.8 Programming Considerations

Note: Several errata have been discovered in the Intel chip and documentation. Some are mentioned below, but firmware, driver and software writers are strongly encouraged to consult the errata sheet.

Note: the 82596 megacell will be utilized in CA mode within Lasi, as opposed to the DX mode operation for 720/730/750. Utilization of CA mode was chosen to allow access to the data burst mode supported by the 82596 in CA mode only.

4.8.1 Endian Mode

The 82596 will be set to operate in big endian mode to be compatible with GSC. Programmers should take care to consult the big endian sections of the various references.

4.8.2 Bus Size

The Lasi GSC Interface provides a 32 bit data bus to the LAN controller.

4.8.3 System Configuration Pointer

The SCP defaults to 0x00FFFFF4. If this is unacceptable given memory configuration and other system parameters, firmware/software must write an acceptable value to the LAN Port Select (see above) between reset and first Channel Attention. The reader is reminded that alternate SCP addresses must be divisible by 16. In addition, *the SCP must reside in system memory.* The Intel book suggests putting the SCP in ROM, but in the 712, the ROM is on the Flash EPROM bus, and this bus is only readable by the host. All data structures must reside in main memory.

Please make sure that any bits marked "x" in the SCP description are set to 0. Intel informs us that the chip will not work otherwise.

As always, see the manual for more information.

4.8.4 Sysbus Byte

The sysbus byte must be located at 0x00FFFFF7, or at location *n+1* if an alternative SCP is used. Following is a discussion of the sysbus configuration byte. Please refer to section 5.4 of the manual.

4.8.4.1 Mode

Presumably, the 82596CA will be used in the *linear* mode. Set the bits accordingly.

4.8.4.2 Bus Throttle and Arbitration

In brief, this is how arbitration works: When the LAN needs the bus, it pulls its HOLD line. Based on the core I/O priority scheme, Lasi arbitrates for GSC on the LAN's behalf, then grants it the bus by asserting HLDA. The 82596 then has the bus for as long as it wants, which in general should be less than 5 μs.

By default, we will not use the bus throttle features. To do this, **set the TRG bit of the sysbus byte to 1** for external bus release triggering. The external trigger is hard wired inactive. These two measures in tandem with the Lasi arbitration mechanism allow the 596 to have the bus for as long as it needs to finish all pending work. However, if during system testing it is discovered that the 596 hangs onto the bus to the extent that it degrades system performance, it would be desirable to have an easy (i.e. run–time) way of (1) enabling the internal bus throttle trigger by setting the TRG bit to 0 and (2) configuring the 596 throttle registers.

4.8.4.3 Locked Cycles

The LOCK bit of the sysbus byte should be set to 0 to enable "semaphore" operations on UPDATE_ERR_CNTRS and RCV_RBD_PREFETCH.

4.8.4.4 Interrupt Pin Polarity

The INT bit of the sysbus byte should be set to 1 to force the INT pin to be **active low.**

4.8.4.5 Channel Switch Algorithm

Intel informs us that the CSW bit of the sysbus byte must be set to 1 for correct chip operation.

4.8.5 Performance Considerations

4.8.5.1 FIFO Vector

Although the 82596CA has large FIFOs, 128 bytes on receive and 64 on transmit, certain system configurations may impose long bus access latencies on the LAN. The FIFO has a programmable threshold. If the FIFO *vector* (note this is *not* the same as the threshold *value*) is set too high, the megacell will frequently request the bus, thus causing inefficiency due to arbitration overhead. Refer to section 7.3, *Setting the FIFO Thresholds,* in the User's Manual. If it is set too low, the FIFO may overrun or underrun before getting the bus. It is probable that some tuning will be required to fully optimize performance.

4.8.5.2 Memory Organization

In general, simplified, linear memory structures aligned on 4 byte boundaries, and larger size for data buffers, will tend to help LAN hardware performance, but at a cost to efficient memory utilization.

4.9 Station Nodal Address

A permanent copy of the LAN Station Nodal Address is kept in Flash EPROM.

4.10 LAN Memory Map

The memory map below does not include the address block referenced by the System Configuration Pointer (SCP).

All addresses in the memory map are Write Only, are are specified relative to the relocatable Lasi base address. To read internal state, execute the Dump command.

	31	Bit Numbering		0
Address (Hex)	Byte 0	Byte 1	Byte 2	Byte 3
000	Reset (no data)			

004	CPU PORT_L Access – see sec. 5.4 of the *82596 User Manual*
008	Channel Attention (no data)

5

5 PARALLEL INTERFACE

5.1 Introduction

The Parallel Interface is an 8 bit parallel, synchronous interface commonly used for printers. The 712 hardware implementation has bidirectional capabilities compatible with PS2 standards, also known to the world as Centronics(tm). The 712 hardware is also capable of interfacing to BiTronics type printers which transmit status information back to the workstation. The Apollo CP300 (alias Tektronics 4693D) raster copier (printer) is also supported. The HP Scanjet parallel port interface is NOT supported on the 712 and cannot be supported, even through software handshaking.

This chapter is divided into two sections: the first covers the parallel port slave interface and the second describes the DMA controller used by the parallel port.

5.2 Overview

The PS2 and AT compatible features are controlled through Lasi. Lasi emulates most of the functionality of the WD16C522 parallel to provide a 720/730 compatible device driver/receiver interface. Handshaking functionality required by CP300 products is provided by special hardware built into Lasi. Lasi drives and receives all signals to/from the parallel port connector without intermediate buffers.

5.3 Features

- Supports outbound host DMA
- Bidirectional non–DMA interface
- Supports Bitronics interface through software.
- Supports Apollo CP300 (aka Tektronics 4693) raster copier.
- 25–pin female DB25 connector (same as IBM PS/2)
- Recognizes both NACK and BUSY handshakes, independently or together
- Pull–up resistors on all lines

5.4 Addressing

All addresses within this chapter are offsets from the base address for each 4K block. The Parallel slave registers, the DMA registers, and the DMA reset register each have a separate 4K block. The abbreviations listed in the following table show the base address to be used for a given register. Please note that the 4K base addresses listed in the table need to be OR'd with the Lasi Base Address determined by the SPACE[1:0] bits.

Abbreviation	4K Base Address
dma	10_1000
slv	10_2000
drst	10_3000

Addresses in this document are always byte addresses. All addresses should be accessed with byte transactions from the host. This allows backwards compatability with previous interfaces which supported only byte accesses and also prevents unintended register modification. Lasi's parallel interface will properly handle 16-bit and 32-bit writes and reads but, because sparse decoding is used, unintended actions could result.

For example, a 32-bit write to address slv+0x800 would put byte 0 data out on the parallel data lines (and potentially start a full handshake depending on the mode selected), write byte 1 data to the status register (this would be ignored), write byte 2 data to the Parallel Device Control register, and byte 3 data would be ignored. However, a 32-bit write to dma+0x400 would put byte 1 data into the Current Count High Byte register as expected, but would also write to three other registers because many registers have multiple addresses. In this case, byte 0 data would be written to the Current Address register, byte 2 data would be written to the DMA Interrupt Logging register, and byte 3 data would be written to the Current Address High Page register.

5.5 Power Up Reset

The following conditions are set during power up reset:

- All parallel interface state machine controllers in Lasi are reset.
- NSTB is High.
- NAFD (alias WRnRD, WR/nRd) is High.
- NINIT (alias NRESET, nRESET) is Low.
- NSLIN (alias NSLCT_IN, NSLCTIN) is Low.
- All Parallel Port Interrupts will be disabled.
- TDC1 is loaded with 1 μs count value.

5.6 Quick Reference for Parallel Slave Registers

Register Abrev. Name	Bit / Bit Value / Address	7	6	5	4	3	2	1	0
		$80	**$40**	**$20**	**$10**	**$8**	**$4**	**$2**	**$1**
ParReset	slv+000	Reset (data value ignored).							
ParData	slv+800	DATA							
ParStatus	slv+801	Nbusy	Nack	pe	slct	Nerr	Nint	1	1
ParDevCtl	slv+802	1	1	NwrRd	IrqEnb	NSlin	Ninit	Autofd	Strobe
—	slv+803	Not Used (undefined)							
ModeCtl	slv+804	Mode[2]	Mode[1]	Mode[0]	Biden	fNstb	0	0	0
IECtlStat	slv+805	dmaInt	0	NbsyInt	Nack-Int	ack-Nbsy	peInt	slctInt	errInt
TDC0	slv+806	Timing Delay Value 0 (obsolete)							
TDC1	slv+807	Timing Delay Value 1							

5.7 Slave Register Descriptions

5.7.1 Directed Master Reset (Address slv+0x000)

Bit Field	Name	Write Effect	Value on Read
7–0	Reset	This has the same effect on the Parallel Interface as the power up reset described before except that the NSTB, NAFD, NINIT, and NSLIN lines are not affected. Software must initialize those outputs by writing the appropriate bits in ParDevCtl (slv+0x802) after a directed reset is invoked.	May be any value.

5.7.2 Write/Read Data (Address slv+0x800)

Bit Field	Name	Write Effect	Value on Read
7–0	ParData	Write data direct to parallel port according to handshake mode selected. There are two ways of setting the port to write mode: 1) Set the direction bit, Biden (slv+0x804 bit 4), to 0. 2) Set slv+0x804 bit 4 to 1 and slv+0x802 bit 5 to 0. Note: Write to this address only when in handshake Mode 0, 1, or 2.	Read data direct from parallel port. Biden (slv+0x804 bit 4) must = 1 and NwrRd (slv+0x802 bit 5) must = 1. Note: Read used only when in handshake Mode 0

NOTE: Writes to this register while DMA is in progress will be ignored, but reads will work properly.

5.7.3 Parallel Port Status (Address slv+0x801)

Bit Field	Name	Write Effect	Value on Read
7	Nbusy	Ignored.	returns 0 if the BUSY signal is asserted (high).
6	Nack	Ignored.	returns 0 if the NACK signal is asserted (low).
5	pe	Ignored.	returns 1 if the PE signal is asserted (high).
4	slct	Ignored.	returns 1 if the SLCT signal is asserted (high).
3	error	Ignored.	returns 0 if the NERR signal is asserted (low).
2	NINT	Ignored.	Returns 1.
1		Ignored.	Returns 1.
0		Ignored.	Returns 1.

Bit 2 was defined as NINT in the 720/730 documentation. With 720/730 this bit returns a 0 if NACK had a low to high (trailing edge) transition, and reading the status register will set this bit to 1. However, on 720/730 products this bit cannot generate a system interrupt to the processor. A complete set of interrupt choices, including a NACK interrupt with this functionality, is available in the "IE Control/Interrupt Status" register (Byte Address slv+0x805), so this function is redundant. Furthermore, the "IE Control/Interrupt Status" can be used to generate a system interrupt.

5.7.4 Parallel Device Control (Address slv+0x802)

Bit Field	Name	Write Effect	Value on Read
7,6		Ignored.	Returns 1 for both bits.
5	NwrRd	Direction control. Only significant if bi-den = 1 (see Byte Address 4, bit 4). If 1, direction is "input from device". If 0, direction is "output to device".	Returns 1.
4	IrqEnb	Ignored. (See note under status register definition.)	Returns 0.
3	NSlin	If 1, NSLIN = H. If 0, NSLIN = L.	If signal at connector is high, returns 1. Otherwise returns 0.
2	Ninit	If 1, NINIT = H. If 0, NINIT = L.	If signal at connector is high, returns 1. Otherwise returns 0.
1	Autofd	If 1, NAFD = L. If 0, NAFD = H.	If signal at connector is low, returns 1. Otherwise returns 0.
0	Strobe	If 1, NSTB = L. If 0, NSTB = H	

With ASP, this register exists on a separate WD16C522 part. When ASP is in mode 1, 2,3, or 4, it writes to the WD part to change NSTB. This write forces NwrRd to 0, IrqEnb to 0, Nslin to 0, Ninit to 1, Nafd to 1. At the end of the transfer, Nstb is 1. With Lasi, NwrRd, IrqEnb, Nslin, Ninit, and Nafd will not change if an automatic handshake mode is being used: their value will change only by writing this register. Nstb will be dependent on the mode selected in the Mode Control register. If the mode is 1, 2, or 4, Nstb will be high when no transfer is in progress. If mode 0 is used, Nstb will be dependent on this register. Care should be taken to insure NSTB=H (bit 0=0) when the mode value is changed, so no stray pulses occur on NSTB.

5.7.5 Mode Control (Address slv+0x804)

Hardware supports both DMA and non–DMA transfers for automatic handshake modes 1, 2, and 4. Mode 3—the Scanjet automatic handshake mode—is not implemented in Lasi.

CAUTION: Changing modes before a previous parallel data transaction is complete may cause unpredictable results. Software must always make sure previous transactions are complete before changing modes.

Bit Field	Name	Write Effect	Value on Read
7–5	Mode	Sets current mode. See the next table for mode encodings.	Returns value written
4	Biden	Bidirectional enable. If 1, data buffers will accept data from device. If 0, data buffers will drive data to the device.	
3	Fstb	Force NSTB pulse. When 1, forces NSTB pulse to be low for the programmed delay period, i.e. NSTB H to L transition is not conditional on BUSY=H. Prevents deadlock condition for devices which only return BUSY=1 when NSTB makes a L to H transition (such as the CP300 device in non–stream mode).	
2–0		Ignored.	May return either a 0 or 1.

Mode Reg Bits			Handshake Mode	Mode Description
Bit 7	Bit 6	Bit 5		
0	0	0	Mode 0	No automatic hardware handshake. Handshaking is under software control.
0	0	1	Mode 1	NACK pulse with not BUSY handshake. Handshake is complete if NACK pulse has completed and then BUSY=L. DMA or non–DMA transfers are allowed.
0	1	0	Mode 2	BUSY only handshake
0	1	1	Mode 3	Reserved
1	0	0	Mode 4	Stream mode. Automatic NSTB generation with no NACK or BUSY handshakes. Hardware will make the data setup, strobe duration, and data hold times the same as determined by the delay value programmed via TDC1 (address slv+0x807).
1	0	1	Mode 0	see above
1	1	0	Mode 0	see above
1	1	1	Mode 0	see above

5.7.6 IE Control/Interrupt Status (Address slv+0x805)

Bit Field	Name	Write Effect (Interrupt Enables)	Read (Interrupt Requests)
7	DMA done	Enable interrupts for when DMA completes its block transfer. (Interrupt when the whole pipeline is empty.)	DMA done has made a 0 to 1 transition, so DMA has completed its block transfer. This bit is cleared only by writing a zero into it.
6		No effect.	Returns 0.
5	Nbusy Intr	Enable interrupts on BUSY trailing edge H to L transition.	Busy H to L transition (not busy). This bit is cleared only by writing a zero into it.
4	Nack Intr	Enable interrupts on NACK trailing edge L to H transition.	NACK L to H transition occurred. This bit is cleared by writing a zero into it.
3	ack-Nbusy Intr	Enable interrupts when NACK transitions from L to H and BUSY = L (not busy).	NACK L to H transition occurred and BUSY = L. This bit is cleared only by writing a zero into it.
2	PE Intr	Enable interrupts on any PE transition.	The PE line has made any transition. This bit is cleared only by writing a zero into it.
1	Select Intr	Enable interrupts on any SLCT transition.	The SLCT line has made any transition. This bit is cleared only by writing a zero into it.
0	Error Intr	Enable interrupts on any NERR transition.	The NERR line has made any transition. This bit is cleared only by writing a zero into it.

Logic exists to prevent losing interrupts. The following sequence demonstrates the interrupt logic behavior:

1. Two or more bits of this register are set.
2. Two or more interrupts which are enabled in this register become pending.
3. The enabled Lasi global parallel interrupt register bit becomes active causing the interrupt to be serviced by the host.
4. The host reads this register, clears the Lasi global parallel interrupt bit, and then chooses to clear only one of the pending interrupts by writing that bit to zero in this register and writing one(s) for the other pending interrupt bit(s).
5. Another interrupt edge for the other pending and enabled interrupt(s) will be generated to the global parallel pending interrupt bit in Lasi immediately following the write to this register.

Also, if an interrupt was pending but not enabled, and then the host enables that pending interrupt, an interrupt edge will be sent to the Lasi global parallel interrupt register bit.

5.7.7 Timing Delay Counter 0 (Address slv+0x806)

This register provides backwards compatibility with products using the Asp chip. On these products, a bug fix required loading this register with a value based on the bus speed. For compatibility with kernel I/O drivers written for these systems, this register exists in the register map, but doesn't have any function.

Bit Field	Name	Write Effect	Value on Read
7–0	TDC0	Ignored.	May return any value.

5.7.8 Timing Delay Counter 1 (Address slv+0x807)

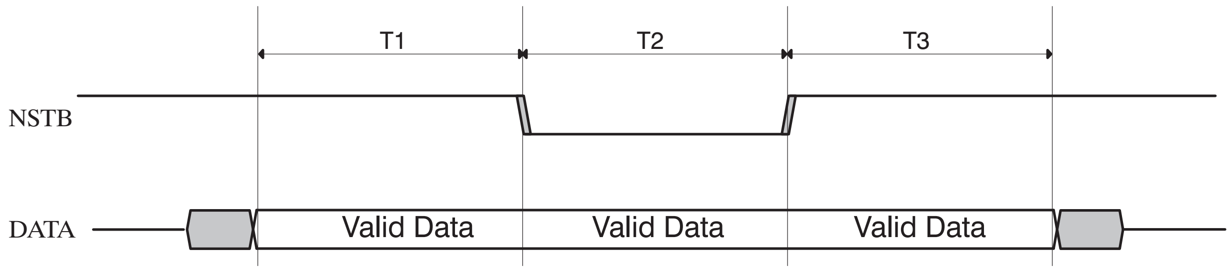


Figure 2. Parallel Data Timing

Bit Field	Name	Write Effect	Value on Read
7–0	TDC1	A count value which establishes the minimum data setup (T1) and NSTB pulse (T2) times. Also establishes the data hold time (T3) when in mode 4 (stream mode). The same count value is used for the T1, T2, and T3 periods. Note: The NSTB pulse will be low for the delay time programmed or the time it takes for BUSY to become true, whichever comes last.	Returns value written.

To compute the count value for this register, use the following formula:

<count_value> = (<desired_delay> – 200ns) / 50 ns

The "standard" for desired delay is typically equal to or greater than 1 x 10E–6 second (1 μs). However, other values may be chosen depending on peripheral/cable restrictions. Be sure to round up to the next higher integer value to ensure that the timing is not under the minimum delay time required. Some examples follow:

calculated value = 8.6 ... round up to 9
calculated value = 8.3 ... round up to 9
calculated value = 8.0 ... use 8

At power up, the register will have a value necessary to insure a delay of at least 1 μs for each period (T1, T2, and T3).

5.8 Parallel Port DMA Controller

The 712's parallel port DMA controller emulates an EISA DMA controller by providing the same register map and counter behavior. To simplify system verification and minimize area, the 712's parallel port DMA reads one 32–bit word from memory at a time and then releases the bus.

5.9 Parallel DMA Register Map

DMA Controller Register Map			
Address	Type	Size (Bytes)	Description
drst+000	write only	1	DMA Reset Register
dma+000	read/write	1	Current Address Register
dma+001	read/write	1	Current Count register
dma+008	read only	1	Status Register
dma+00A	write only	1	Write single mask bit
dma+00B	write only	1	Mode register
dma+00C	write only	1	Clear byte pointer
dma+00D	write only	1	Master Clear
dma+00E	write only	1	Clear Mask register
dma+00F	read/write	1	Mask register
dma+010	read/write	1	Fifo limit register (not used)
dma+087	read/write	1	Current Address low page register
dma+401	read/write	1	High Current Count register
dma+40A	read/write	1	Interrupt Pending register
dma+487	read/write	1	Current Address High Page register

5.10 How Parallel Port DMA Works

The 712's Parallel Port DMA controller transfers data from memory to the parallel port without disturbing the CPU until the transfer sequence is complete. To start a sequence the DMA channel needs to have a beginning address and byte count placed into the proper registers. Given that the mode (read or write) is set up properly, DMA will start once the mask bit is reset. After the sequence is complete, an interrupt will happen, the mask bit will be set, and the address and count registers will be at their final value. This controller does not support chaining, so after each sequence the count and address registers need to be reinitialized to their starting values.

The DMA controller does transactions by arbitrating for the bus, reading one 32-bit word, giving up the bus, and then handshaking each needed byte out to the parallel device. The DMA controller never produces multi–word GSC transactions or writes to memory.

5.11 Parity Errors and Bus Timeouts

If the 712's Parallel Port DMA controller gets a parity error or bus timeout while mastering a transaction, the transaction causing the error will be completed normally, but arbitration will be disabled so no more DMA can be done. The Current Address register will point to the next address to be read after the error. All of the data in the error transaction will be handshaked to the parallel device and the Current Count register will have the value indicating the data just read had been processed; this should help in the debug process.

5.12 Detailed Register Descriptions

5.12.1 Parallel DMA Reset Register

drst+000 (DMA_Reset) wo

This reset will reset all DMA state machines and clear all DMA registers to their "init" value. The value written doesn't matter: all that needs to happen is a byte write to this address. This register is on a separate 4K page so the Current Address register can have an offset of zero in the "dma" page.

5.12.2 Current Address Registers

dma+000 (Current_Address) rw init: 0

This is a 16 bit register. The byte pointer indicates which byte is accessed on a read or write. On the first access after reset, the byte pointer indicates to access the low byte. On subsequent accesses, the byte pointer indicates to access the high byte. The byte pointer is reset by power on, by a write to dma+00C (Clear_Byte_Pointer), or by a write to dma+00D (Master_Clear).

dma+087 (Current_Address_Low_Page) rw init: 0
8 bit read/write

dma+487 (Current_Address_High_Page) rw init: 0
8 bit read/write

The DMA current address is a 32–bit physical address constructed as follows:

32–bit Current Address Register		
High Page– 8 bits	Low Page– 8 bits	Current Address– 16 bits

Each EISA DMA channel also has a 32–bit read–only Current Address register. The DMA controller automatically increments the address after each transfer. The intermediate values of the address are stored in the Current Address register during the transfer. This register is cleared (set to 0x00000000) after reset.

In EISA, the way to access the Base or Current Address register is very cumbersome. The address register includes a 16 bit 8237 compatible segment (address 0000) which combines with the low page segment (address 0087) and the high page segment (address 0487) to provide a 32–bit EISA DMA address. Listed below is the procedure to access the address register:

1. CPU performs a write to the Clear Byte Pointer register (00C).
2. CPU performs an 8 bit read/write to the least significant byte (bit 7–0) of the register 000.
3. CPU performs an 8 bit read/write to the most significant byte (bit 15–8) of the register 000.
4. CPU performs a bit read/write to the low page segment (address 087) for Address register bits [23:16].
5. CPU performs a bit read/write to the high page segment (address 487) for Address register bit [31:24].

5.12.3 Byte Count Registers

dma+001 (Current_Count) rw init: FFFF

This is a 16 bit register. The byte pointer indicates which byte is accessed on a read or write. On the first access after reset, the byte pointer indicates to access the low byte. On subsequent accesses, the byte pointer indicates to access the high byte. The byte pointer is reset by power on, a write to dma+00C (Clear_Byte_Pointer), or a write to dma+00D (Master_Clear).

Note that this register counts down to –1, so (Current_Count + 1) bytes are transferred during a DMA transfer.

dma+401 (Current_Count_High_Byte) rw init:FF

DMA transfer length is specified by a 24 bit register:

24–bit Current Count Register	
High Count– 8 bits	Low Count– 16 bits

As a DMA transfer progresses, the transfer length is decremented by the number of bytes transferred so far, and the physical address increments by the number of bytes transferred so far.

For the same reason as in the case of the Base and Current Address register, we fold the Base and Current Word Count register into a single Byte Count register. This register is cleared (set to 0x000000) after reset.

The Byte Count register consists of two parts, the 16–bit 8237 compatible segment, and the 8–bit high byte count segment. The two segments are mapped at different I/O address and must be programmed separately. Listed below is the procedure to access the Byte Count register:

1. CPU performs a write to the Clear Byte Pointer register. (000C)
2. CPU performs an 8 bit read/write to the least significant byte (bit 7–0) of the register 001.
3. CPU performs an 8 bit read/write to the most significant byte (bit 15–8) of the register 001.
4. CPU performs an bit read/write to the high byte count segment (address 401) for Byte Count register bit 23–16.

5.12.4 DMA Status register

dma+008 (Status_Register)   ro init: 00000001

Bit 0 is set whenever the terminal count is reached.
Bit 4 is set whenever the DMA channel is requesting servicing
the other bits are hardwired to zero.
Simulator notes: bit 4 can be wired to zero for simulation.

The DMA Status register contains status information about the DMA channels that may be read by the CPU. The information includes which channels have reached a terminal count and which channels have pending DMA requests. In Lasi, we only support one DMA channel, thus only the status bits corresponding to channel 0 will be reported.

bit 0 – This bit is set every time terminal count is reached. It is set whenever the current–count register is FFFFFF. The reset state for bit 0 is 1.

bit 4 – This bit is set whenever the DMA channel is requesting servicing. This bit will be zero when the requesting unit (the parallel port) is reset.

bits 1, 2, 3, 5, 6, and 7 are hardwired to zero.

5.12.5 DMA Write single mask bit

dma+00A (Write_Single_Mask) wo init: 04

This register may be used to set or clear any mask register bit.

bit[1:0] – must both be zero to write this bit.
bit[2] – 0: Clear DMA mask bit
1: Set DMA mask bit (STATE AFTER RESET IS BIT 2 IS SET)

Examples:

A write of XXXXX000 clears the dma_mask_bit.
A write of XXXXX100 sets the dma_mask_bit and starts DMA transaction.
A write of XXXXXX10 has no effect.
A write of XXXXXX01 has no effect.
A write of XXXXXX11 has no effect.

5.12.6 DMA Mode register

dma+00B (Mode_Register)   wo init: 01

bit[1:0] – must both be zero to write these bits.
For example, 00001000 sets read transfer mode
00001011 changes nothing

bit[3:2] – Data Transfer Type
00 No transfer DMA Disabled
01 Write transfer
10 No transfer DMA Disabled
11 No transfer DMA Disabled

bit 4 – Lasi hardwired to 0: Disable Auto–initialization

bit 5 – Lasi hardwired to 0: Address increment select.

bit[7:6] – Lasi hardwired to 00: Demand Mode DMA transfer.

State after reset is 0x01: DMA is in write mode.

Examples:

A write of XXXX0000 disables DMA.
A write of XXXX0100 sets transfer type to write (Gecko to peripheral).
A write of XXXX1100 disables DMA.
A write of XXXXXX01 has no effect.
A write of XXXXXX10 has no effect.
A write of XXXXXX11 has no effect.
Bits 7–4 always read as zeros.

5.12.7 DMA Interrupt Logging Register

dma+40A (Interrupt_Log_Bit) rw init: 0

This single bit register is not implemented. Writes to this register will be ignored and reads will return data from another register.

5.12.8 DMA Mask Register

dma+00F (Mask_Register)   rw init: 00001111

This register is similar to DMA write single mask bit since we only have one DMA channel on Lasi.

Bit Field	Name	Write Effect	Value on Read
7–4	–	Ignored.	Always read 0.
3–1	–	Ignored.	Always read 1 because that's what EISA would do if only DMA channel zero were being used.
0	dma mask bit	0: Clear DMA mask bit. This will start a pending DMA transaction. 1: Set DMA mask bit (after reset this bit is set). Writing a one will stop an ongoing DMA transaction.	Reads current mask value.

5.12.9 FIFO Limit Register

dma+010 (FIFO_Limit_Register)   rw

Writes to this register are ignored and reads may return any value on the 712. This register is defined but not implemented, allowing compatibility with the 720/730 and with any future products that use a FIFO.

5.12.10 DMA Clear byte pointer

dma+00C (Clear_byte_Pointer)   wo init: byte pointer = 0.

The Clear Byte Pointer command clears the internal latch used to address the upper or lower byte of the 16–bit address and word count register. The latch is also cleared at power–on and by DMA controller Master Clear command. For details, please reference to EISA spec 3.1.8. The value written doesn't matter: all that needs to happen is a byte write to this address.

5.12.11 DMA Master Clear

dma+00D (Master_Clear)   wo init: no value

A write to this address clears the byte_pointer and sets the dma_mask_bit.

The Master Clear instruction clears the command, Status, and Request registers, sets the Mask register to disable DMA requests and executes a Clear Byte Pointer command. The value written doesn't matter: all that needs to happen is a byte write to this address.

5.12.12 Clear Mask register

dma+00E (Clear_Mask_Register)   wo init: mask register = 1.

A write to this address clears the dma_mask_bit. The clear mask register command enables the DMA channel by clearing the mask bit. The value written doesn't matter: all that needs to happen is a byte write to this address.

5.13 Typical DMA Sequence

This is what needs to be done to start a DMA sequence. Remember all the writes are byte writes to DMA control registers in I/O space.

```
;
;  Assemble 32 bit physical address for base of transfer
;
Write Clear_Byte_Pointer
Write Current_Address            (low eight bits)
Write Current_Address            (high eight bits)
Write Page_Low
Write Page_High
;
;  Assemble 24 bit transfer length
;
Write Clear_Byte_Pointer
Write Current_Count  (low eight bits)
Write Current_Count  (high eight bits)
Write High_Count
;
;  Set DMA direction
;
Write Mode_Register
;
;  Clear DMA mask bit
;
Write Mask_Register
```

When the DMA mask bit is cleared, dma+008{0} is set to 1. If the parallel port DMA enable bit is set, DMA transfer starts. If the parallel port DMA enable bit is clear, the DMA transfer waits for the parallel port DMA enable bit to be set before starting.

Once the DMA transfer is complete:

- dma+008{0} is cleared to 0 (DMA in progress)
- dma+40A{0} is set to 1 (DMA mask bit)
- dma+40A{0} is set to 1 (DMA interrupt bit)
- dma+805{7} is set to 1 (parallel DMA terminal count)
- If dma+008{7} is clear, dma+008{7} is set and an external interrupt request is forwarded from Lasi to Bigbird.

Also, the parallel port handshake interrupt bits ($\overline{\text{BUSY}}$, Nack 0–>1, etc) may be set as appropriate for the current handshake.

5.14 EISA Compatibility

The 712's DMA controller used the EISA specification as a guide for its original definition. This DMA controller acts like DMA channel zero of an EISA system. Using this model results in an indirect register model for directed reads and writes to control ports. An example is the byte pointer used to access the most significant byte of the address and count registers: the byte pointer allows 8–bit reads and writes to access a 16–bit register from one byte address. EISA is uses this scheme because the 8–bit Intel 8237 DMA controller was popular in the evolution of the PC–AT. Unsupported reads will return junk data with correctly set parity, while unsupported writes may do unpredictable things.

Here is a summary of known differences between the EISA specification and Gecko's internal DMA controller:

The FIFO:

THe 712's DMA channel does *not* have a FIFO integrated with the DMA controller. However, for compatibility, a FIFO limit register is placed onto the same 4K page as the EISA like registers of the DMA controller. Because this register exists on this page, the register address map is not exactly the same as that of an EISA system.

No Chaining:

To support DMA buffer chaining and autoinitialization, each EISA DMA channel has a 32–bit write–only Base Address register that is programmed with the base address for DMA transfer. The register does not decrement or increment.

In order to simplify the design, no attempt will be made to support the DMA buffer chaining and autoinitialization for Parallel Printer Interface DMA due to its low data transfer rate (maximum at ~400Kbytes/sec). Thus, inside Lasi, the Base Address register and the Current Address register are folded into a single address register.

The write–only Base Address register and the read–only Current Address register are accessed through the same address. When you write to the combined Current/Base address, you write to the base address register. When you read from that address, the value you get is the Current Address register. The same holds for the Current Count registers.

No Auto–reset of Address/Count registers:

Writing to the less significant Address/Count register bytes automatically clears the high bytes on an EISA system. Gecko's DMA controller doesn't do this; it's address and count registers are straight read and write registers.

Read/Write mode:

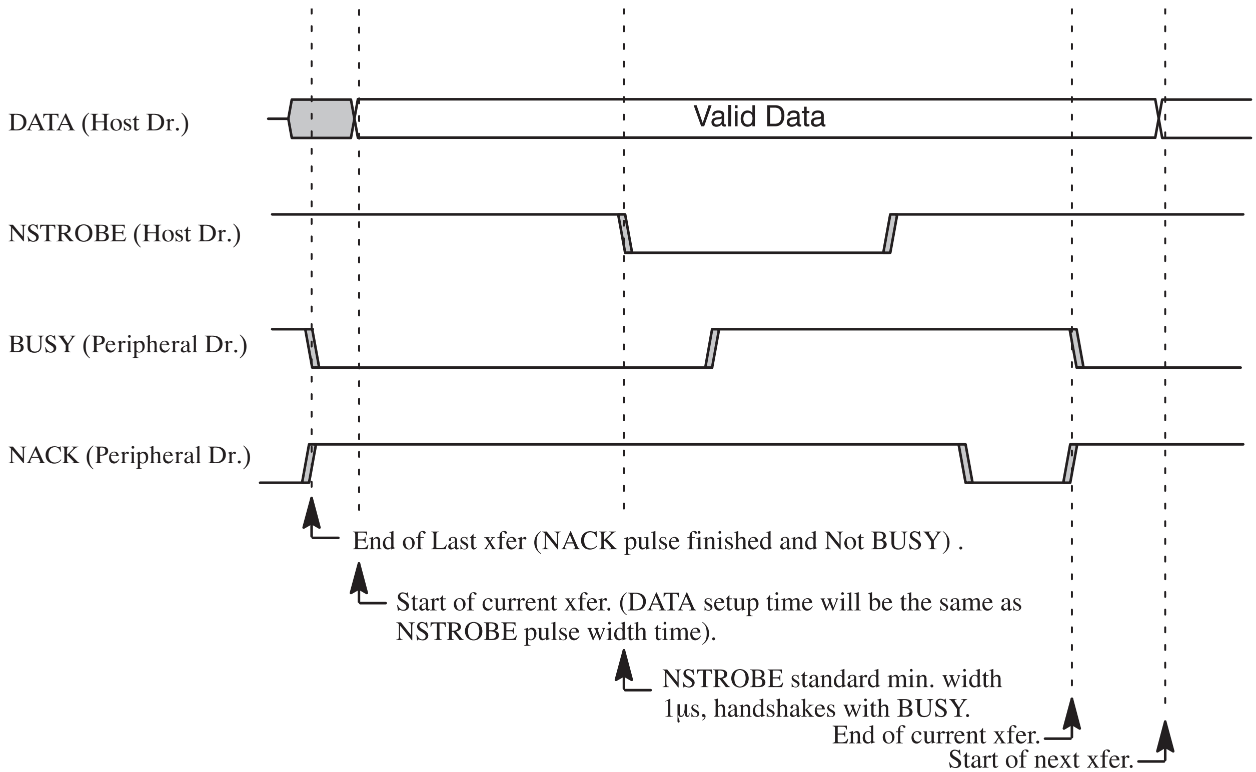
For the 712's parallel port, "write mode" is defined as writing to the parallel port and reading from memory. In an EISA DMA controller, "write mode" means write to memory, and "read mode" means read from memory.

5.15 Testing

Testing the Parallel Printer Interface can be accomplished by attaching a Centronics Test Hood Box.

5.16 Timing Examples

The below diagram shows output to the peripheral. This wave pattern could have been generated by host writes or by DMA.



6 AUDIO/TELEPHONE

6.1 Introduction

Harmony is a Time Domain Multiplexed (TDM) subsystem that communicates with the audio and telephony circuitry of the 712. Audio for the 712 is similar to, and software compatible, with the audio circuitry of early 715s. Therefore, the driver and API developed for past products will be leveraged in the development of the Harmony driver and API.

6.1.1 Audio Description

Although the audio for the 712 is similar to the features of other products, the 712 is the first to incorporate telephony into the workstation. This option allows the user complete and simultaneous access to two telephone lines for the purpose of voice or data applications. Harmony is designed to support both analog as well as ISDN telephone. Analog telephone is supported through the use of a character channel TTY interface, and ISDN is enabled with the addition of an ISDN DMA channel. The heart of the audio system is the CODEC (coder–decoder). The 712 will be able to use either the CS4215 or the AD1849. Both of these CODECs are similar in functionality, but not identical. However, Harmony is designed to work with either interchangably. One of the main points of difference between the two parts is in the CODEC loopback mode (not to be confused with either audio loopback mode, ISDN loopback mode or any one of the three TTY loopback modes available). This is a feature that will be avoidable through software fixes.

The CODEC combines CD and DAT quality stereo A/D converters for microphone and line input levels, as well as D/A converters for driving headset and line outputs. The input sampling rate and format are programmable, as are the input gain control (used for software control of recording levels) and output attenuation. The output attenuation is suitable for headset volume control. It is also possible to mix incoming audio with the outputs, thus allowing the user to monitor whatever information they may be recording. In terms of the physical audio connections, Harmony supports the following audio inputs and outputs, but the user must recognize that only one input may be used at a time:

- Mono microphone input
- Stereo line input
- Mono speaker output (not externally available)
- Stereo headphone output

There will be a single output jack provided with this product that provides headset output. Although this output is capable of driving 8 Ohms, it can also be used for higher impedance devices with little or no additional distortion, thus a line level input can be driven by the headset output. Playback and recording of audio are real-time processes, and they must operate at a constant rate. They are sometimes described as isochronous, which means they are constant with respect to time. Most processes running on workstations need not be isochronous, and if the system is more heavily loaded they will run slower. That can not be allowed to happen to audio, because users will not tolerate random pauses (or stutters) in audio. To guarantee that audio will get data in an uninterrupted flow, Harmony uses DMA to get data to and from memory. The audio driver will lock a portion of physical memory as a buffer for audio data and then inform Harmony where to get and store the data. Harmony will only receive one physical page address at a time for both playback and record, so an interrupt driven process is used to ensure that the driver will provide a new page address when Harmony needs it. Harmony initiates the interrupt in advance so the new address will become available before the data from the current page has been exhausted, therefore, Harmony has a register for the current page as well as one for the next page. The driver will always write the new page address in the register that contains the address for the next page.

Harmony is designed so that playback and record function simultaneously. Unless the DMA interrupts are turned off entirely, the system is continuously sourcing and sinking data for the audio. If audio is being used in only one direction, say to record, the driver will be written so it preprocesses data and gives Harmony an address of memory that contains null information so that known data is being sent to the CODEC.

6.1.2 Telephony Description

The telephony subsystem will be composed of two complete telephone interfaces that could possibly be configured with one being devoted to voice applications, and the other could be equipped with V.32bis data modem and fax. A separate card will be available that offers the user basic-rate ISDN. In addition to the previously features mentioned, the released version of Teleshare will include

- Caller ID
- Call Waiting
- Call Forwarding
- Conference Calling
- Data Rate Conversion and mixing
- Call Recording and Voice Mail
- A multitude of DTMF-based applications

The flexibility and robustness of this system are based upon a Digital Signal Processor (DSP) architecture. Each phone channel is equipped with an AD2105 fixed-point DSP and SRAM, which lends to such flexibility. If both phone channels are loaded with sufficient memory, they will both be enabled to perform the data modem applications. The majority of software that will be used by the telephony card is tightly coupled with the audio, in fact, all audio data will pass through the telephony data path before it is presented to the CS4215.

Communication between Harmony and the audio/telephony subsystem will take place over a modified Concentration Highway (CHI) bus whose basic structure is defined in the CS4215 data sheet. Essentially, this bus is composed of a transmit wire, receive wire, frame clock, bit clock and data/

ncontrol wire. When the telephony card is not inserted into the system the sole communication is between Harmony and the CS4215, however, once Teleshare is in place, data flows to both Teleshare and the CS4215 depending upon the mode of operation. During the time when control information is being sent to the CS4215, Harmony sends information directly to the CODEC and Teleshare is temporarily disabled. When the mode of operation changes from control to data, Teleshare becomes the sole communicator with Harmony and the CS4215 receives all of its data indirectly through Teleshare. This multi–mode operation is necessary and implies that the audio server always maintains control over the CODEC and does not allow the telephony subsystem to arbitrarily change the sample rate and width on its own. The reason Teleshare is not allowed to have independent control over the CODEC due to the fact that Lasi generates the two primary clock frequencies that are sent to the CODEC. If the correct frequency is not provided to the CODEC, unexpected results may occur. The only way to guarantee proper operation is to force the audio server to alter the control information held by the CODEC. In addition, the multi–mode operation also forces buffers to be inserted between Lasi and the CODEC that always connect the two paths together during control mode and that conditionally disconnect the paths during data mode when Teleshare is present.

6.2 CHI Communication

Viewing Harmony as a software model, the audio section is composed of two DMA channels, one going to and one coming from the CODEC. This portion of the circuitry provides the real-time audio data. The telephony portions of Harmony are seen as two TTY ports and two DMA channels that are similar to the audio DMA channels. The TTY channels independently send and receive data and commands to and from Teleshare. The TTY ports emulate the majority of the functions present in the description of a WD16552, except for a few line status functions (i.e. parity error, framing error and overrun error), but the TTY ports are complete with dual 16–byte fifos. Contrary to the audio DMA channels, the two character channels are NOT isochronous connections but are interrupt driven, indicating much system overhead is required to service every TTY–generated interrupt. The ISDN DMA channels are similar to the audio DMA channels in that they utilize two addresses simultaneously in addition to storing the next input and output addresses for future use. The ISDN DMA channels do ***not*** have fifos associated with them but rely on the fact that a DMA must be completed within 125uSec within the time it is requested.

As described in the previous section, the CHI bus operation depends on the mode of the CODEC and the presence of Teleshare. If it is in control mode the CHI bus operates as defined in the CS4215 data sheet. If the CODEC is in data mode and Teleshare is not present the same holds true. However, once Teleshare is inserted in the system and the CODEC is in data mode the CHI bus seen by the CODEC is directly connected to Teleshare and disconnected from Harmony, and the CHI bus seen by Harmony is also a separate connection to Teleshare and disconnected from Harmony. This allows completely independent operation of the two busses.

A description of the CHI bus seen by Harmony is given in this section for data mode when Teleshare is inserted in the system. The CHI bus is composed of a frame clock, bit clock, transmit wire and receive wire. The transmit and receive data is time multiplexed onto their appropriate wires based on the occurrence of a frame sync. The frame sync rate is equal to the sample rate of the CODEC and the bit clock is independent and sourced by Teleshare. The redefinition of the CHI bus seen by Harmony is as follows:

- The first 64-bits immediately following a frame sync is as defined in the CS4215 specification; it contains information for the CS4215.
- All subsequent information preceding the next frame sync is either TTY or ISDN related and is transmitted in groups of 16 bits that are referred to from here on as Tframes. Each Tframe is specified by an address and a type field that indicates its purpose and destination.
- During data mode Teleshare generates the frame sync and clock signals seen by Harmony. The CS4215 receives its data directly from Teleshare and not from Harmony, consequently, it may operate at an independent clock rate from that of Harmony.
- In order to alleviate extra crystal requirements for Teleshare, the frequency of the bit clock that is sourced by Teleshare and seen by Harmony will be 13.824 MHz because it has this frequency already available to it, however the bit clock used for communication between the CODEC and Teleshare will be strictly determined by the sampling rate and is sourced by the CODEC.
- Since the data rate of the CHI bus and the CODEC will not coincide, a frame sync does not have to be issued at prespecified intervals but is issued based on the data demand of the CODEC. Therefore, fewer frame clocks are seen while the CODEC is operating at 8KHz than while it is operating at 48KHz.
- Tframes are continuously transmitted until data is required by the CODEC. At this time a frame sync is issued and CODEC data is transmitted.
- The Tframes are sent most significant bit first and their makeup is as follows:

TTY Frame Format

15–13	12–10	9	8	7–0
addr	**type**	**N/C**	**RDL**	**Data/Control**

- **addr:** Three address bits indicate which port is sourcing the forthcoming data/control byte.
- **type:** Three bits of type indicate what type of data is being sent for the current address.
- **RDL:** Active low Ready for Data. One bit is sent by each end of the CHI bus indicating if it is able to accept data for the address previously sent. Recall that a frame is being received at the same time this frame is being transmitted. Consequently, Harmony will receive a bit indicating if it can continue to send the frame shown above. Harmony also transmits a bit to Teleshare indicating whether or not it is able to accept the byte being sent by Teleshare.
- **Data**: Eight bits of data or control compose this field whose definition is determined by the first two fields sent in the transaction.

List of CHI addresses supported by Harmony

Address	Data Src/Dest
000	TTY Port 0

001	**TTY Port 1**
010	**ISDN Byte**
111	**Telebyte**

List of CHI types supported by Harmony

Type	Function
000	**TTY Control Telebyte LSB**
001	**TTY Data Telebyte MSB**
010	**Divisor LSB**
011	**Divisor MSB**
111	**Idle**

CHI Address Desciptions

- **TTY Ports 0 and 1:** Information sourced or sunk and can be either data or control type. The information is in reference to one of the two TTY ports resident inside of Harmony.
- **ISDN Byte:** A bidirectional character that originates from or is going to system memory. An ISDN transaction is *always* originated by Teleshare. Harmony transmits one ISDN byte in response to every ISDN byte received.
- **Telebyte:** This transmitted byte of data is for use by Teleshare. The definition of these bits is not known by Harmony and does not affect its design. These two bytes of data are accessed by writing to the location defined as the scratch pad register in a 16C552 data sheet.

CHI Type Descriptions

- **TTY Control / Telebyte LSB:** This type applies to addresses 0, 1, and 7. In the context of a TTYaddress it is control information as described in a paragraph below. When this type is associated with address 7, it is originated by TTY port zero.
- **TTY Data /Telebyte MSB:** Similar to the above definition only it refers to data for TTY channels and indicates a Telebyte from TTY port one.
- **Divisor LSB:** Least significant divisor byte from either TTY address 0 or 1.
- **Divisor MSB:** Most significant divisor byte from either TTY address 0 or 1
- **Idle:** Transaction is invalid but does contain a code identifying the version of the installed telephony card.

Makeup of Control Byte Sourced by Teleshare

10	9	8	7	6	5	4	3
CTSL	**DSRL**	**RIL**	**RLSDL**	**N/C**	**N/C**	**N/C**	**BRK**

- **CTSL:** Active low Clear To Send bit received from Teleshare.
- **DSRL:** Active low Data Set Ready bit received from Teleshare.
- **RIL:** Active low Ring Indicator bit received from Teleshare.
- **RLSDL:** Active low Received Line Signal Detect received from Teleshare.

- ☐ **BRK:** Break Signal received from Teleshare.

Format of Control Information Sourced by Harmony

9	8	7	6	5	4	3 1
N/C	N/C	RING_ASRTL	DTRL	RTSL	BRK	N/C

- ☐ **RING_ASRTL**: Ring Assert is sent to Teleshare, sourced by Modem Control Register.
- ☐ **DTRL:** Data Transmit Ready sent to Teleshare, sourced by Modem Control Register.
- ☐ **RTSL:** Request To Send sent to Teleshare, sourced by Modem Control Register.
- ☐ **BRK**: Break Interrupt signal sent to Teleshare. sourced by the Line Control Register.

The CHI bus definition has been reorganized to take full advantage of the bandwidth that is available on the CHI bus. If a data addressing scheme were not used, the definition would have to revert back to time slot addressing where one device has a predefined time slot for communication. This is a viable alternative and conforms to the true definition of the CHI bus as defined by AT&T, but it also potentially wastes a large portion of the bandwidth that is available on the bus. A data addressing scheme theoretically can utilize all of the available bandwidth if either none of the other devices have information to provide or if their information is not needed.

6.3 Harmony Architecture

The following block diagram gives the reader a clearer picture of the Harmony architecture. This diagram details the internals of Lasi and does not include any description of the telephony subsystem. There are two DMA channels and two TTY channels. All data intended for the Audio DMA channel flows through the CODEC control block, FIFO, DMA-channel pathway, while all TTY and ISDN DMA related data is parsed by each individual TTY/ISDN block in conjunction with the TTY Serial Output Control section.

6.4 Audio Software Interface

6.4.1 Base_Offset

Since Harmony uses DMA to transfer data, it masters the Lasi internal bus for that operation. However, it must be initialized in slave mode. In slave mode, Harmony is hard wired to the following address range:

0xF010 4000 – 0xF010 4FFF

In the rest of this document, the start of that address range will be referred to as Base_Offset. All addresses pertaining to Harmony in slave mode will be relative to this Base_Offset, even if that is not explicitly specified.

In this document, the term *word* will mean a 32 bit quantity. A half word is 16 bits and a byte, as always, is 8 bits. All registers in Harmony are 32 bits wide, although some of the bits may be unused. Byte and half word accesses to all registers are supported by Harmony.

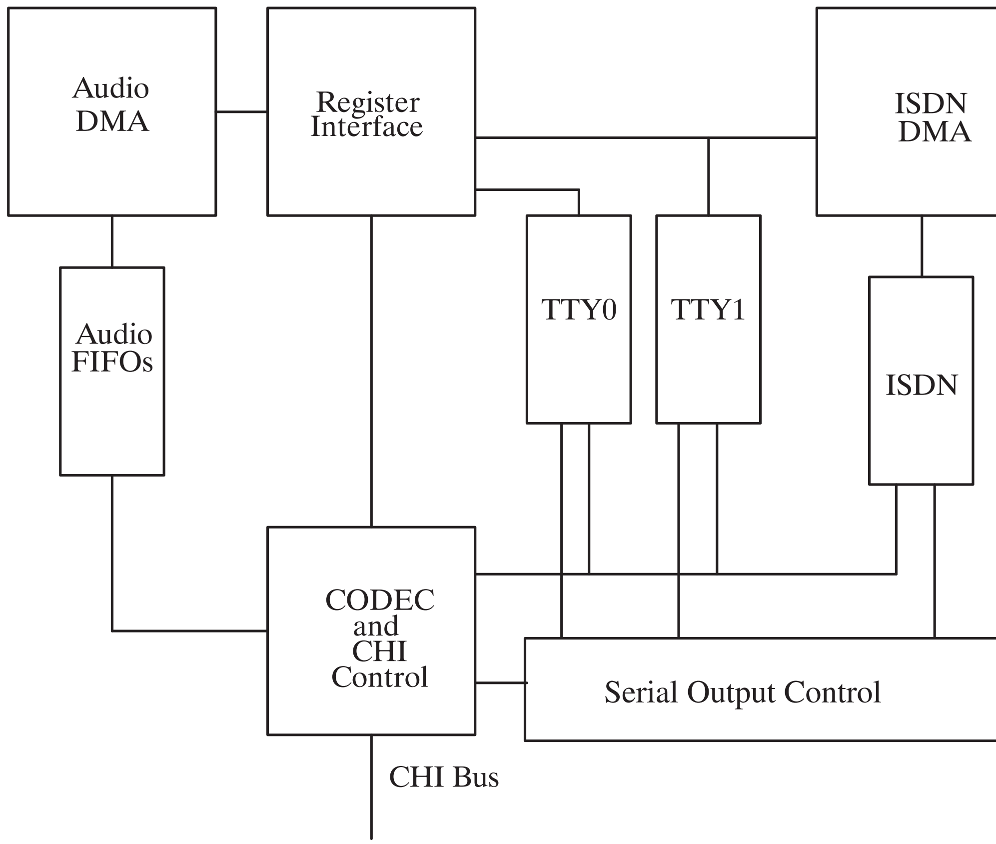


Figure 3. Harmony Block Diagram

6.4.2 ID register (Address: 0x000)

This register allows software to distinguish Harmony from other audio sub--systems, specifically the voice quality audio subsystem on the 710. The 710 only implements a single byte ID at address 0x001, so it is recommended that the driver should distinguish between the two audio devices using that byte alone. Byte 2 is read back as the Teleshare identification. These bits will become valid following the completion of the first audio frame generated by the CODEC after powerup.

ID register

	31 … 0
0x000	**ID**

- **ID**: Reads 0x0014 0000 when Teleshare is ***not*** installed. Writes are ignored.
- **ID**: Reads 0x0015 X000 when Teleshare is installed. Where X are Teleshare identification bits sent across the CHI bus on every idle transaction following the first audio frame. In order to obtain the Teleshare identification bits one must first write a control word to offset address 0x008 and then wait for control mode to complete and for the first audio frame to be transmitted. At this time the identification bits are transmitted by Teleshare.

6.4.3 Initializing the CODEC

The CS4215 CODEC requires a 50 ms reset once after power up to guarantee proper operation. Since the Gecko reset signal is not nearly this long, help from software is required to create a 50 ms reset, using the **RESET** register. At power up, the LSB of the **RESET** register will be 1, which asserts reset to the CODEC. When at least 50 ms have elapsed since power up, software must write a 0 to this bit. Obviously, no audio can be played until this time. Reading any register and writing **RESET** and **GAINCTL** registers are the only legal operations while the reset bit is 1. If it is necessary to reset the CODEC at any other time, software can write a 1 to the **RESET** bit, wait 50 ms, and then write a 0 to the **RESET** bit. Note that it is best to initialize **GAINCTL** register before clearing the **RESET** register.

RESET register

	31 – 1	0
0x004	**Unused**	**RESET**

- **RESET**: Powers up as 1, which resets the CODEC. Writing a 0 brings the CODEC out of reset. This bit can be read back.
- **Unused**: Writes to these bits are ignored. Always read back as 0s.

The next step is to put the CODEC in the control mode to setup operating parameters such as the sampling rate and the data format by writing the desired parameters into the control (**CNTL**) register. A write to this register will automatically put the CODEC in control mode. The part indicates that condition by setting the MSB of this register, which is a status bit. It will remain set as long as the CODEC needs to stay in control mode for proper initialization. This status bit will be cleared by the hardware when the CODEC gets out of control mode and is ready to record and playback. Software may write to this register as often as necessary, but **playback and recording must be stopped before this register is written. This means that PC and RC bits of DSTATUS register must be 0s when this register is written. (DSTATUS register is defined later in this document.)** Also, to avoid creating any pops or clicks, it is best to reduce the volume of the audio outputs to the minimum.

CNTL (Control) register

0x008

31	30 – 28	27 – 20	19 – 18	17 – 14	13	12	11	10	9 – 8	7 – 6	5	4 – 3	2 – 0
C	**unused**	**Rev[7:0]**	**unused**	**Exp[3:0]**	**nAC**	**nAD**	**OLB**	**ITS**	**LS**	**DF**	**ST**	**XS**	**SR**

- **C**: Control status bit. Indicates that the CODEC is still in control mode, and not ready for data. A write to any portion of this register will always set this bit, regardless of the data. It will be cleared when the CODEC is ready for data. **When this bit is set, register reads are the only legal operation on Harmony. In other words, after you write this register (thus setting this bit), you must wait for this bit to be cleared before you start playback / record or even write any register.** Power up value: 0.
- **Rev[7:0]**: Revision number of the CODEC that is being used. These bits are read only.
- **Unused**: Writes to these bits are ignored. They always read back as 0s.
- **Exp[3:0]:** Used for future definition expansion of the CODEC and correspond to control bits 17, 9, 8, and 3 respectively.

- **nAC:** Inverted value of this bit sent to CODEC to establish compatibility with AD1849 CODEC. This bit induces automatic calibration immediately following positive assertion of dnc bit.
- **nAD:** Inverted value of this bit sent to CODEC to establish compatibility with AD1849 CODEC. This part requires this bit of the control word to be high.
- **OLB:** Output Level Bit. When set reduces headphone full scale output to 2.0 Vpp.
- **ITS:** The Immediate Tristate Bit forces the CODEC to immediately tri–state its clock lines when dnc goes low.
- **LS**: Loopback select. Used for diagnostics only. The table below describes the types of loopback possible. Recall the loopback functionality varies between the two CODECs

LS	type of loopback
0	No loopback
1	Loopback within Harmony
2	Digital loopback in the CODEC
3	Analog loopback in the CODEC

- **DF**: Selects one of three data format listed below.

DF	format selected
0	16 bit linear
1	8 bit μ-law
2	8 bit A-law
3	Reserved (never use this value)

- **ST**: Stereo select. A 1 selects stereo, a 0 selects mono. In mono mode playback, both channels will be driven with the same data. For mono mode recording, the signal on the left channel will be recorded. When recording from the microphone in stereo mode, both channels will contain similar data since the microphone is monophonic.
- **XS**: Crystal select. The only valid values for these bits are 01 and 10. This field, in conjunction with the next field, determines the sampling rate.The least significant **XS** bit is sent to the Lasi phase–locked loops for determining the CODEC crystal frequency.
- **SR**: Sampling rate. The following table shows which sampling rate (in kHz) is selected for a given 5 bit value of **XS** and **SR**. Remember that values beginning with 00 or 11 are illegal. Values 01100 and 01101 are also illegal. That leaves 14 valid sampling rates however, software is strongly discouraged from supporting them all. For future hardware compatibility, only the rates that are required by our customers should be made available through the API.

XS, SR	Sample rate	XS, SR	Sample rate
01000	8 kHz	10000	5.5125 kHz
01001	16 kHz	10001	11.025 kHz
01010	27.42857 kHz	10010	18.9 kHz
01011	32 kHz	10011	22.05 kHz
01100	Illegal	10100	37.8 kHz
01101	Illegal	10101	44.1 kHz
01110	48 kHz	10110	33.075 kHz
01111	9.6 kHz	10111	6.615 kHz

6.4.4 Playback and Recording

The basic idea is that the driver will start the audio hardware on one physical page and chain to new pages as the hardware goes along. The chaining will be accomplished by hardware interrupts. To guarantee that the hardware does not run out of pages of memory, the software must provide the address of the next page ahead of time. Therefore, the hardware will keep two physical page addresses, one for the current page, and another for the next page. Every time the hardware finishes a page, it will jump to the next page address and interrupt the host so that it can provide a new next page. When hardware jumps to the next page, its address will be transferred from the next page register to the current page register. Thus, the driver will only write to the next page register. The current page register will be read only. When the circuit is inactive, interrupts will be disabled. To start the audio process from that state, software will first enable interrupts. An interrupt will occur immediately since the next page address register does not contain a valid address. The interrupt service routine will write the next page address, which will immediately be transferred to the current page address register. Another interrupt will immediately take place, because the next page address needs to be setup again. When that interrupt is serviced, everything will be in *steady state* and things will progress normally. To end this process, software will simply disable interrupts after the addresses for the last page of playback data **and** the last page of recorded data have been supplied to Harmony.

The above discussion applies to both playback and recording. The only distinction between the two is that in playback the pages supplied to the hardware will have been initialized with data to be played, whereas in recording the pages will not be initialized by software. The hardware will write the recorded data in these pages and software will read them later. Of course, there are two pairs of address registers, one for playback and one for recording.

The format of the DMA data in memory will depend on the values chosen in the **Control** register. The specific formats will be described later in the document.

1.1.1 Playback and recording are synchronous

To minimize complexity in hardware, playback and recording are synchronous in Harmony. However, the software interface is designed to allow a future design in which playback and recording are asynchronous. In this context, synchronous means that the DMAs for playback and recording will be started and stopped simultaneously. Also, when Harmony masters the bus, it will usually transfer the data for playback as well as recording in the same mastership. **To initiate DMAs in Harmony, software must write the address for playback first, and then the address for recording.** The second transaction will initiate DMA for both playback and recording. On an interrupt, the **DMA status** register (described later) should be checked to see if a new address is required for playback or record or both. In Harmony, since playback and recording are synchronous, both will require a new address at nearly the same time, however, it is recommended that software check the bits individually and provide data only for the bit that is asserted. This will ensure that existing software will be compatible with any future hardware that allows playback and recording to be asynchronous. Such hardware would start each DMA independently when the associated address is written. Obviously, the old software can not use the extra flexibility, but new software could then be written to do so. In the meanwhile, the old software would still run on the new hardware.

1.1.2 Metering

The system software for audio will implement a metering function for audio that is being recorded. The purpose of this function is to allow the user to adjust the recording level. Metering works well only if the latency for the entire system, from the analog input to the actual display, is small. Otherwise, users will notice that metering is out of sync. Ordinarily, in a DMA based approach, when software asks hardware to write data into a page of memory, the software can not read from it until hardware is done with that page. Any software accesses to that page would normally run into coherency problems. Therefore, the latency for metering will include the latency of filling one page of data. At low sampling rates the delay will be excessive.

To solve that problem, the current address register is defined as a pointer to the next memory address that hardware will write to, or read from. Reading this register will allow software to tell exactly how far the hardware has progressed in that page. Since that address is always incremented, not decremented, software can be guaranteed that no coherency problems will occur if it reads the data before that address in that page. **Caution: The location pointed to by the current address register should not be accessed by software. The previous word (located at current address – 4) is the last word guaranteed to contain fresh data. There is a corner case to worry about. If the current address register points to the first word of a page, that page has not been written at all, but the previous page has been written completely. However, (current address – 4) does not necessarily give you the last word of the previous page, since pages need not be allocated sequentially in memory.**

Even though metering is a concern for recording only, playback implements the same protocol for consistency.

1.1.3 Register interface

The current and next address registers for playback are shown below. Under normal operation, software should only write to the next address register. The current address should be treated as a read only register. Actually, the current address register is writable for test purposes, but software is **not** allowed to write it. Also remember that the value in the current address register can change dynamically. **To get coherent data, the current address register must be read in a single, 32-bit word operation.** Hardware designers please note: If Harmony is interfaced to a bus where word accesses are broken up into half word or byte accesses, you will break this feature.

PNXTADD (playback next address) register

	31 … 12	11 … 0
0x010	**PNXTADD**	**0x000**

- **PNXTADD**: Full 32 bit physical address for the next page of playback data. That page must be locked. Since this address points to the first location of a 4K byte page, the 12 LSBs must be 0s. This does *not* mean that any bit shifting is required on the address. During a write, the 12 LSBs of address are ignored by Harmony and the value for these bits is forced to 0. This means that Harmony is hard wired for 4K pages.

PCURADD (playback current address) register

	31 … 0
0x014	**PCURADD**

- **PCURADD**: Full 32 bit physical address for the next location to be read for playback data. Writes are legal for test purposes only. **This register must be read with a single 32 bit operation to get coherent data.**

The current and next address registers for recording are shown below. **The caveats regarding PCURADD apply to RCURADD as well.**

RNXTADD (recording next address) register

	31 … 12	11 … 0
0x018	**RNXTADD**	**0x000**

- **RNXTADD**: Full 32 bit physical address of the next page for recorded data. That page must be locked. Since this address points to the first location of a 4K byte page, the 12 LSBs must be 0s. This does *not* mean that any bit shifting is required on the address. During a write, the 12 LSBs of address are ignored by Harmony and the value for these bits is forced to 0.

RCURADD (recording current address) register

	31 … 0
0x01C	**RCURADD**

- **RCURADD**: Full 32 bit physical address for the next location to be written with recorded data. Writes are legal for test purposes only. **This register must be read with a single 32 bit operation to get coherent data.**

Note that **PNXTADD** must be written before **RNXTADD**.

1.1.4 DMA and Interrupt status

This status register allows software to determine whether a DMA is in progress or not, and whether playback and record processes need new page addresses. It also lets software enable or disable interrupts. The data in this register is dynamic, in that it can change spontaneously. If that is not taken into account by software, some race conditions might be created.

DSTATUS (DMA status) register

- **IE**: A 1 in this bit enables DMA–based interrupts, a 0 disables them. This bit does not clear an interrupt; a DMA–based interrupt is cleared by writing the **PNXTADD** and **RNXTADD** registers. Powers up as 0.

	31	30 … 10	9	8	7 … 2	1	0
0x020	**IE**	**Unused**	**PN**	**PC**		**RN**	**RC**

- **PN**: Playback New address request. It is set when a new value for **PNXTADD** is needed, and reset when **PNXTADD** is written. This bit is read only, writes are ignored. Powers up as 1.
- **PC**: Playback DMA currently active. This bit is set when DMA is started, cleared when DMA finishes. This bit is read only, writes are ignored. Powers up as 0.
- **RN**: Recording New address request. It is set when a new value for **RNXTADD** is needed, and reset when **RNXTADD** is written. This bit is read only, writes are ignored. Powers up as 1.
- **RC**: Recording DMA currently active. Set when DMA is started, cleared when DMA finishes. This bit is read only, writes are ignored. Powers up as 0.

In Harmony, since playback and record are synchronous, **PN** and **PC** simply duplicate the function of **RN** and **RC**. Generally, **PN** and **RN** will be in the same state, and **PC** and **RC** will be in the same state. However, they don't transition at exactly the same time. This redundancy allows a future implementation to provide asynchronous playback and record. To allow compatibility between software for Harmony and future hardware, software should write **PNXTADD** only when **PN** is true, and write **RNXTADD** only when **RN** is true (even though checking just one bit would be sufficient).

The interrupt line is asserted when **IE** is true and both **PN** and **RN** are true. In Harmony, **PN** and **RN** will be asserted at *almost* the same time, so the interrupt line will be asserted when both playback and record need new addresses. In future hardware, **PN** and **RN** could change asynchronously. In such hardware, an interrupt must be issued when either **PN** or **RN** is true. For example, the interrupt could be caused by the assertion of **RN**, so the software will read **PN** as false. However, right after the read, **PN** could go true, before the interrupt has been serviced. This would keep the interrupt line constantly asserted, even after software is done servicing the interrupt. In many systems, the CPU hardware for interrupts is edge sensitive. Therefore, in the case just outlined, no interrupt would take place from the assertion of **PN**, so that event would go un–serviced. To avoid such a problem, it is recommended that the interrupt service routine should first disable interrupts, and then write new addresses as necessary. The interrupt should be re–enabled at the end of the routine. This re–enabling will ensure that a new edge will be created if another cause of interrupt had just shown up. The re–enabling also helps avoid a potential problem in Harmony itself, so it is essential to follow this guideline.

Note that **PN** is asserted somewhat earlier than **RN**. Assertion of either signal will cause an interrupt, so the CPU will see an interrupt as soon as **PN** is asserted. If the CPU responds to the interrupt very quickly, **RN** may not be asserted when the CPU reads **DSTATUS**. In that case, the CPU should write **PNXTADD** only. Another interrupt is guaranteed to occur when **RN** eventually becomes true, as long as the software follows the protocol of writing 0 to **IE** during the interrupt service routine.

1.1.5 Data Formats

If stereo mode is on, twice as much data will be needed on every sample as in mono mode. Similarly, 16 bit linear mode requires twice as much data as 8 bit A-law or μ-law modes. There are four possible combinations of these modes, requiring anywhere from 8 to 32 bits per sample. The four modes are shown below, with a diagram showing how the data is packed within a word of memory. Remember, these are not register definitions. **Note that Harmony can only operate on word aligned data. If**

you offset the format shown below with 1, 2 or 3 bytes, the results will be very disconcerting. Also note, negative numbers will be in 2's complement format.

8 bit A-law or μ-law, mono

31 24	23 16	15 8	7 0
sample n	sample n + 1	sample n + 2	sample n + 3

Each sample is byte sized, and these bytes are packed into a word with the first sample in the most significant byte.

8 bit A-law or μ-law, stereo

31 24	23 16	15 8	7 0
sample n, left	sample n, right	sample n+1, left	sample n+1, right

Each sample contains a byte for the left channel and a byte for the right channel. The byte for left data is more significant than the byte for right data. The resulting half word samples are packed into a word with the earlier sample occupying the more significant half word.

16 bit linear, mono

31 16	15 0
sample n	sample n+1

Each sample is a half word. The half word samples are packed into a word with the earlier sample occupying the more significant half word.

16 bit linear, stereo

31 16	15 0
sample n, left	sample n, right

Each sample contains a half word for the left channel and a half word for the right channel. They are packed into a word with the half word for left data in more significant position than the half word for right data.

6.4.5 Gain Control

The CODEC provides control over gain (or attenuation) of each analog channel. It also provides on/off control over each type of input and output. All of these parameters are controlled through the **Gain Control** register.

GAINCTL (Gain Control) register

	31 28	27	26	25	24	23 20	19 16	15 12	11 6	5 0
0x00C	**Unused**	**HE**	**LE**	**SE**	**IS**	**MA**	**LI**	**RI**	**LO**	**RO**

- □ **HE**: Headphones output enable. A 1 turns on headphones, 0 turns them off.
- □ **LE**: Line output enable. A 1 turns on line outputs.
- □ **SE**: Speaker enable. A 1 turns on the speaker.
- □ **IS**: Input select. A 0 selects line inputs, while a 1 selects microphone inputs. Both line and microphone inputs can *not* be active simultaneously.

- **MA**: Monitor attenuation. This 4 bit value selects the amount of attenuation applied to the input signal before it is mixed with the output. A value of 0 provides no attenuation, resulting in maximum level. Each step in the value increases the attenuation by 6 dB. The maximum value, 0xF, attenuates by 90 dB, which effectively turns off the monitor function.
- **LI**: Input gain for the left channel (applies to either microphone or line input, whichever has been selected). Gain is adjusted in steps of 1.5 dB, ranging from 0 dB for 0x0 to 22.5 dB for 0xF.
- **RI**: Analogous to **LI**, but applies to the right channel.
- **LO**: Left output attenuation (applies to headphones, speaker and line outputs). Attenuation is adjusted in steps of 1.5 dB, ranging from 0 dB for 0x00 to 94.5 dB for 0x3F.
- **RO**: Right output attenuation. Analogous to **LO**, but applies to the right channel.

Warning: LI and RI are gain numbers, i.e., the largest value results in the highest level. On the contrary, MA, LO and RO are attenuation values. For these, the largest value results in the lowest level.

1.1.6 Guidelines for avoiding audible artifacts

When the control register is written, the analog outputs of the CODEC may produce a spike, which is an artifact perceived by the user as a click or a pop. To avoid such an artifact, write **GAINCTL** register with maximum attenuation value in **LO** and **RO** fields. While this practice is not required for proper operation of Harmony, it is strongly recommended. Also, when first initializing Harmony after power-up, it is best to write **GAINCTL** first, with maximum attenuation in **LO** and **RO**. This will help avoid any artifacts when the CODEC is brought out of reset.

6.4.6 Over-range Indication

If the amplitude of the input signal is too large, an over-range condition will occur. This is communicated to software with the OV register.

OV (Over-range) register

	31 … 1	0
0x024	**Unused**	**OV**

- **OV**: Indicates that an input over-range condition took place. After such an event, this bit will remain set until software clears it by writing a 0 to it. Software should never write a 1 to this bit. Clearing **OV** via software takes precedence over setting it in hardware, so it is possible that as software clears this bit it may miss another over-range condition that took place simultaneously. Power up value: 0.

6.4.7 PIO register

The **PI** and **PO** fields of the control register are used to communicate with the PIO lines of the CODEC. Currently, no use for these bits has been identified, however, a mechanism to use them exists in Harmony. The **PIO** register is used for that purpose. It is suggested that software initialize this register with 0x0000000f so that a future use is identified for these lines, the hardware can count on these lines starting in a known state.

PIO register

	31 – 4	3 – 2	1 – 0
0x028	**Unused**	**PO**	**PI**

- **PO**: Parallel output. Used to communicate with **PIO** lines of the CODEC.
- **PI**: Parallel input. Read only, the write data is ignored. Used to communicate with **PIO** lines of the CODEC.

There are 2 **PIO** lines, and 2 bits each in **PI** and **PO** registers. If a given **PIO** bit is to be used as an output, the value to be output should be written in the corresponding bit of the **PO** register. If a bit is to be used as an input, the corresponding bit in **PO** must be set to 1. Then, the corresponding bit in **PI** will reflect the state of that input. Even if a bit is being used as an output, the **PI** bit will reflect the state of that line, i.e., it will simply mirror the state of **PO** bit, but with some delay. Note that the hardware connected to the a **PIO** line will be designed to use it either as an input or an output, so that is what determines what protocol to use.

6.4.8 DIAG register

The **DIAG** register is used for diagnostics only and is of limited use. It essentially reads the signal that is present on the CHI bit clock and depending on the mode, will yield a one on all consecutive reads. However, if the CODEC is in data mode consecutive reads will provide different values at random times. This bit definition is different from that of Vivace for Lasi was not able to support one more pin for this purpose. However, if the CODEC is in data mode, this bit will provide a good indication of whether or not it is alive for the signal should be evidently toggling.

DIAG register

	31 – 1	0
0x03c	**Unused**	**CO**

- **CO**: SCLK from the CODEC. Toggles when the CODEC has been initialized properly and is in data mode.

6.5 TTY Software Interface

The following table is a concise description of the TTY registers contained in each character channel used in communication with the telephony subsystem. These register descriptions correspond to those in the WD16552 data sheet.

Description	R/W	D7	D6	D5	D4	D3	D2	D1	D0
Receiver Buffer Register (RBR)	R	Data Bit 7	Data Bit 6	Data Bit 5	Data Bit 4	Data Bit 3	Data Bit 2	Data Bit 1	Data Bit 0
Transmitter Holding Register (THR)	W	Data Bit 7	Data Bit 6	Data Bit 5	Data Bit 4	Data Bit 3	Data Bit 2	Data Bit 1	Data Bit 0
Interrupt Enable Register (IER)	R/W	0	0	0	0	Enable MSI	Enable RLSI	Enable THREI	Enable RDAI
Interrupt Ident Register (IIR)	R	Fifos Enabled	Fifos Enabled	0	0	Int ID Bit 2	Int ID Bit 1	Int ID Bit 0	Int Not Pending
Fifo Control Register (FCR)	W	Rx Trig MSB	Rx Trig LSB	X	X	DMA Mode	Tx Fifo Reset	Rx Fifo Reset	Fifo Enable
Line Control Register (LCR)	R/W	DLAB Bit	Set Break	0	0	0	0	0	0
Modem Control Register (MCR)	R/W	TBRdy	RingAsrt	Rsrvd	Loop Back	Unused	*See Note 1*	RTS	DTR
Line Status Register (LSR)	R	0	Txmitter Empty	Tx Hold Reg Emp	Break Interrupt	0	0	0	Rx Data Avail
Modem Status Register (MSR)	R/W	RLSD	RI	DSR	CTS	Delta RLSD	Trail Edge RI	Delta DSR	Delta CTS
Divisor Latch Reg LSB (DLL)	R/W	Divisor Bit 7	Divisor Bit 6	Divisor Bit 5	Divisor Bit 4	Divisor Bit 3	Divisor Bit 2	Divisor Bit 1	Divisor Bit 0
Divisor Latch Reg MSB (DLM)	R/W	Divisor Bit 15	Divisor Bit 14	Divisor Bit 13	Divisor Bit 12	Divisor Bit 11	Divisor Bit 10	Divisor Bit 9	Divisor Bit 8
TeleByte Register	R/W	Data Bit 7	Data Bit 6	Data Bit 5	Data Bit 4	Data Bit 3	Data Bit 2	Data Bit 1	Data Bit 0

Note 1 – Bit 2 of the MCR is involved in Hardware Handshaking control.

6.5.1 Receive Buffer and Transmit Hold Register

The Receive Buffer Register (**RBR**) is a temporary buffer used to hold incoming data previously transferred to it from the Receive Buffer Shift Register (**RBSR**). The **RBSR**, as well as all other TTY-related registers, is byte-wide and this particular byte is accessible in byte lane 0. If the TTY channel is operating in character mode, whenever a new word is transferred into the **RBR** an interrupt is generated and is not cleared until the **RBR** is read by the CPU. However, an interrupt is generated during fifo mode only after the number of received bytes has met a predetermined trigger level.

The Transmit Holding Register (**THR**) is located at the same address as is the **RBR**, but is for use in the transmit direction instead of the receive direction. In character mode when a byte is accepted by the **THR** it is transferred into the Transmit Shift Register (**TSR**)and sent to Teleshare serially via the CHI bus. An interrupt is generated when the **THR** is empty. This interrupt is cleared once either the

THR is written, or the Interrupt Identification Register (**IIR**) is read. The transmit fifo resides at the same location as does the **THR** and during fifo mode the fifo accepts any incoming bytes which automatically transfers its contents to the **THR** when the **THR** is able to accept data. A system interrupt is generated once the transmit fifo is empty and this interrupt is cleared when at least one byte is written into the transmit fifo.

Notice the fifos are placed in series with the transmit and receive holding registers, consequently, the effective fifo depth is eighteen bytes and not sixteen bytes as stated in the WD16552 data sheet. These registers may be accessed only when bit 7 of the **LCR** (**DLAB**) is zero.

RBR0

	31–24
0x040	**RBR0**

RBR1

	31–24
0x060	**RBR1**

- **RBR0, RBR1**: Software accessible register used to read received data from corresponding TTY port. If fifo mode is enabled, a read from this address will pull data from the receive fifo.

THR0

	31–24
0x040	**THR0**

THR1

	31–24
0x060	**THR1**

- **THR0, THR1**: Software accessible register used to write transmit data to corresponding TTY port. If fifo mode is enabled, a write to this address will insert data into the transmit fifo.

6.5.2 Interrupt Enable Register

The Interrupt Enable Register (**IER**) is able to selectively enable or disable each of the interrupts. A logical one written to a given bit enables the corresponding interrupt, while a zero disables the interrupt from alerting the CPU to its status. This register must be accessed with bit 7 of the **LCR** (**DLAB**) is zero.

IER0

	23–20	19	18	17	16
0x040	0	**MSI0**	**RLS0**	**THRE0**	**RDA0**

IER1

	23–20	19	18	17	16
0x060	0	**MSI1**	**RLS1**	**THRE1**	**RDA1**

- **RDA0, RDA1**: Enables Received Data Available Interrupt or Character Time out Interrupt when set to logic one. In fifo mode, RDA is asserted when the fifo reaches its trigger level.
- **THRE0, THRE1:** Enables Transmit Holding Register Empty Interrupt when set to logic one. The THRE interrupt is activated when the transmit holding register is empty and every time this interrupt bit is taken from a zero to a one. During fifo mode it is asserted when the transmit fifo is empty.
- **RLS0, RLS1:** Enables the highest priority interrupts, Overrun Error and Break Interrupt, when set to a logic one.
- **MSI0, MSI1:** Enables Modem Status Interrupts of CTS, DSR, RI and RLSD when set to a logic one.

6.5.3 Interrupt Identification Register

The Interrupt Identification Register (**IIR**) is a read only register that returns a value indicating which prioritized interrupt is pending. During the time the **IIR** is addressed, the highest priority pending interrupt is frozen and no other interrupts are acknowledged until the particular interrupt is serviced by the CPU. Each **IIR** resides in byte lane 2. The following Interrupt Identification Table describes which interrupt is pending based upon the contents of the **IIR**.

IIR0

	15	14	13–12	11	10	9	8
0x040	**FCR0[0]**	**FCR0[0]**	0	**bit3**	**bit2**	**bit1**	**bit0**

IIR1

	15	14	13–12	11	10	9	8
0x060	**FCR1[0]**	**FCR1[0]**	0	**bit3**	**bit2**	**bit1**	**bit0**

Interrupt Identification Register					Interrupt Set and Reset Functions		
Bit 3	Bit 2	Bit 1	Bit 0	Priority Level	Interrupt Flag	Interrupt Source	Interrupt Reset Control
0	0	0	1	–	None	None	–
0	1	1	0	HIghest	Receiver Line Status	Break Interrupt	Reading Line Status
0	1	0	0	Second	Received Data Available	Receiver Data Available, Character Time Out	Reading the Receiver Buffer Register
0	0	1	0	Third	THRE Empty	Transmitter Holding Register Empty	Reading IIR or Writing to Transmitter Holding Register
0	0	0	0	Fourth	Modem Status	CTS or DSR or RI or RLSD	Reading the Modem Status Register

6.5.4 Fifo Control Register

The Fifo Control Register controls the activity of both the transmit and receive fifos. It is a write only register located in the same address space as the **IIR**. Software is able to determine whether or not fifo mode is currently active, for the **FifoEn** bit is read back as the upper two bits of the **IIR**.

FCR0

	15	14	13 12	11	10	9	8
0x040	**Trigger0[1]**	**Trigger0[0]**	0	**DMS0**	**TxRst0**	**RxRst0**	**FifoEn0**

FCR1

	15	14	13 12	11	10	9	8
0x060	**Trigger1[1]**	**Trigger1[0]**	0	**DMS1**	**TxRst1**	**RxRst1**	**FifoEn1**

- **Trigger0, Trigger1:** Determines the trigger level of the receive fifo. In progressive order the levels are 1, 4, 8, 14.
- **DMS0, DMS1:** DMA Mode Select together with the trigger level determine the value of RxrdyL which is transported to the peripheral through RTS if it enabled to do so in the Modem Control Register.
- **TxRst0, TxRst1:** Resets the transmit fifo counters and clears THRE and TEMT. Contrary to what is stated in WD16552 data sheet, this bit does not clear the contents of the transmit fifo to zero.
- **RxRst0, RxRst1:** Resets the receive fifo counters, RDR and Rdy4data state machine signals as well as all Break signals stored in the receive fifos. Similar to TxRst, RxRst does not clear the contents of the receive fifo.
- **FifoEn0, FifoEn1:** This signal enables the operation of the fifos. This bit must be set before any of the other bits in this register may be written. Consequently, in order to set the trigger level it will take a minimum of two writes to this register; the first to enable the fifo and the second to set the trigger level.

6.5.5 Line Control Register

The Line Control Register is used for sending a Break signal to the telelphony subsystem and for enabling a write to the divisor latch bytes. The other bits normally defined by the line control register dealing with parity information are not used and are read back as zero.

LCR0

	7	6	5 … 0
0x040	**DLAB0**	**SetBrk0**	0

LCR1

	7	6	5 … 0
0x060	**DLAB1**	**SetBrk1**	0

- **DLAB0, DLAB1:** Divisor Latch Access Bit enables the CPU to perform writes to the upper and lower bytes of the Divisor Latch.
- **SetBrk0, SetBrk1:** Set Break forces the TTY channel to transmit a Break to the telephony subsystem the entire time this bit is set. It takes precedence over all other data or control that may need to be sent.

6.5.6 Modem Control Register

The Modem Control Register (**MCR**) controls the interface with the modem and is used to either initiate a modem connection, for flow control, to enable interrupts, or for loopback testing. In the case of loopback testing, all interrupts are disabled. When the loopback bit is set the following occur:

- Transmit Shift Register is connected to the Receiver Shift Register's input.
- Four modem status inputs (**CTS, DSR, RLSD, RI**) are disconnected and the four Modem Control bits are connected to the Modem Status inputs. These signals are inverted from what is stored in the **MCR.** Therefore, to assert each one as active low as seen by Teleshare, the corresponding bit in the register should be written as a one.
- The interrupt output pin is disabled, but the receiver and transmitter interrupts are still fully operational and controlled by the Interrupt Enable Register. However, during diagnostics, the interrupt sources are now the lower four bits of the Modem Control Register.

MCR0

	31	30	29	28	27	26	25	24
0x044	**TBRdy0**	**RingAsrt0**	**Rsrvd**	**LOOP0**	**N/C0**	**Hw_Hs0**	**RTS0**	**DTR0**

MCR1

	31	30	29	28	27	26	25	24
0x064	**TBRdy1**	**RingAsrt1**	**Rsrvd**	**LOOP1**	**N/C1**	**Hw_Hs1**	**RTS1**	**DTR1**

- **DTR0, DTR1**: Data Terminal Ready (**DTR**) informs the associated modem that TTY port has data to send.
- **RTS0, RTS1**: Request to Send (**RTS**) output informs the associated modem it is able to receive data. This bit incorporates the function of **RxRdyL** if **Hw_Hs** is zero, which allows hardware handshake to control data flow.
- **Hw_Hs0, Hw_Hs1:** Used in conjunction with **RTS** and **RxrdyL** to generate a hardware flow control version of **RTS**. **RxrdyL** is defined in the WD16552 data sheet. The flow control version of **RTS** is dictated by the following equation:

  RTS' = !(Hw_Hs | RxrdyL) | RTS

  Essentially **Hw_Hs** either enables or masks the operation of **RxrdyL** as does **RTS.** If **RTS** and **Hw_Hs** are both zero, **RTS'** will be seen as the inverted form of **RxrdyL**. Under these conditions if **RxrdyL** is asserted, it will inform the peripheral device to cease from sourcing data. In loopback mode this bit is connected to bit 6 of the **MSR**.
- **N/C0, N/C1**: No connect. In loopback mode this bit is connected to bit 7 of the **MSR**.
- **LOOP0, LOOP1**: This bit provides a loopback feature for diagnostic testing of the TTY. It is important to note that in order to insure proper operation the LOOP bits *must* be set before data is sent to either channel. In addition, contiguous back to back writes are not guaranteed to function properly while the TTY port is in character mode and the CODEC is in control mode. For best results always operate the TTY port in loopback mode. This loopback mode is contained inside of Harmony and does not require the assistance of Teleshare.
- **Rsrvd**: These bits are for reserved usage.

- **RingAsrt0, RingAsrt1:** These bits are intended to be used for TTY driver testing where the output from one TTY port is routed back to the input of the other TTY port inside of Teleshare. This particular bit will be connected to the **RI** bit in the Modem Status Register and will simulate a ring being detected.
- **TBRdy0, TBRdy1**: These bits indicate when a telephony byte has been written but not yet sent to Teleshare.

6.5.7 Line Status Register

This 8-bit register provides status information to the CPU concerning the data transfer. Departing from the definition of the WD16552, Harmony does not provide the status of Parity Error, Framing Error and LSR7, for they are not deemed to be necessary for this application.

LSR0

	23	22	21	20	19	18	17	16
0x044	0	**TEMT0**	**THRE0**	**BI0**	0	0	0	**RDR0**

LSR1

	23	22	21	20	19	18	17	16
0x064	0	**TEMT1**	**THRE1**	**BI1**	0	0	0	**RDR1**

- **RDR0, RDR1:** Receive Data Ready indicator. Set to logic one whenever a complete incoming character has been received and transferred into the Receiver Buffer Register or the receive fifo. This bit will be reset to logic 0 when either the CPU has either read the Receive Buffer Register in character mode, or when the receive fifo is empty, or when a zero has been written into this bit position.
- **BI0, BI1:** Break Interrupt indicates that a break character has been received in the corresponding TTY channel. The reception of this character forces the creation of the highest priority interrupt. This bit is cleared when the Line Status Register is read. In fifo mode **BI** is written to the fifo and is not seen by the system until it is located at the top of the fifo. During fifo mode it is required that immediately following the transmission of a Break bit, Teleshare transmit a valid word, preferably a null character, which provides the necessary write for the Break bit to be inserted into the fifo. Otherwise the Break signal may not be seen by the system for an indefinite period of time.
- **THRE0, THRE1**: Transmit Holding Register Empty indicates that the Transmit Holding Register is able to accept a new character for transmission during character mode and it indicates when the transmit fifo is empty during fifo mode.
- **TEMT0, TEMT1**: Transmitter Empty indicates that both the Transmit Holding Register or the transmit fifo as well as the Transmit Shift Register are empty, and that both can accept a new character. It is reset to 0 upon loading of the Transmit Holding Register.

6.5.8 Modem Status Register

The Modem Status Register provides the current state of the Modem Control Lines. The four least significant bits provide change information and are set to logic one whenever a control input from the modem changes. They are reset to logic 0 whenever the CPU reads the Modem Status Register.

MSR0

	15	14	13	12	11	10	9	8
0x044	**RLSD0**	**RI0**	**DSR0**	**CTS0**	**DRLSD0**	**TERI0**	**DDSR0**	**DCTS0**

MSR1

	15	14	13	12	11	10	9	8
0x064	**RLSD1**	**RI1**	**DSR1**	**CTS1**	**DRLSD1**	**TERI1**	**DDSR1**	**DCTS1**

- **DCST0, DCTS1:** Delta Clear To Send indicates that **CTS** on corresponding TTY has changed states since the last time it was read by the CPU.
- **DDSR0, DDSR1:** Delta Data Set Ready indicates that **DSR** on corresponding TTY has changed states since the last time it was read by the CPU.
- **TERI0, TERI1:** Trailing Edge Ring Indicator informs the CPU that the **RI** input on the corresponding TTY has changed from a logic 1 to a logic 0.
- **DRLSD0, DRLSD1:** Delta Received Line Signal Detect indicates that the **RLSD** on the corresponding TTY has changed state.
- **CTS0, CTS1:** Clear To Send is the complement of the active low Clear To Send input of the corresponding TTY which indicates the receiving end of the TTY is ready to accept data. This bit becomes equivalent to **RTS** during loopback testing.
- **DSR0, DSR1:** Data Set Ready is the complement of the active low Data Set Ready input of the corresponding TTY and indicates that the other end of this TTY is ready to send data. This bit becomes equivalent to **DTR** during loopback testing.
- **RI0, RI1:** Ring Indicator is the complement of the active low Ring Indicator input of the corresponding TTY and indicates that a ringing signal is being received by this TTY port. This bit becomes equivalent to bit 2 of the Modem Control Register during loopback testing.
- **RLSD0, RLSD1:** Received Line Signal Detect is the complement of the active low Receive Line Signal Detect input of the corresponding TTY and indicates that the TTY port is receiving a signal which meets its signal quality conditions. This bit becomes equivalent to bit 4 of the Modem Control Register during loopback testing.

6.5.9 Divisor Latch Register LSB

The LSB Divisor Latch Register is used to hold the least significant byte of divisor information that is sent to Teleshare. This register may be written only when DLAB (bit 7 of the Line Control Register) is set. It resides in the same address space as transmit and receive registers.

DLL0

	31 … 24
0x040	**DLL0[7:0]**

DLL1

	31 … 24
0x060	**DLL1[7:0]**

- **DLL0, DLL1:** Least significant byte of Divisor Latch Register. In order to access this register DLAB must be set. The information contained by this register is sent to Teleshare every time a write to this location occurs.

6.5.10 Divisor Latch Register MSB

The MSB Divisor Latch Register is used to hold the most significant byte of divisor information that is sent to Teleshare. This register may be written only when DLAB (bit 7 of the Line Control Register) is set. It resides in the same address space as interrupt enable register.

DLM0

	23 ... 16
0x040	**DLM0[7:0]**

DLM1

	23 ... 16
0x060	**DLM1[7:0]**

- **DLM0, DLM1:** Most significant byte of Divisor Latch Register. In order to access this register DLAB must be set. The information contained by this register is sent to Teleshare every time a write to this location occurs.

6.5.11 Telephony Information Byte

The telephony information bytes are accessed by writing and reading the Telebyte registers. Each channel has an independent register whose information is passed directly to Teleshare whenever this register is written. The content of this register has no effect on Harmony but is used by Teleshare to inform it of various modes of operation such as when it is to load code into the DSP's memory and where to direct data from various sources. Another key function that one of these bits enables is a special loopback mode that will be used for one TTY to communicate with the other TTY. The basic need for this loopback mode is for driver development and is available only when Teleshare is present.

The telephony bits are a convenient method of enabling flexibility within Teleshare. A note for software accessibility, bit seven in each modem control register indicates when the telebyte has been written and waiting to be sent to Teleshare. **DO NOT** write another byte of information into the telebyte register until bit seven of the corresponding **MCR** is clear.

TELEBYTE0

	7 ... 0
0x044	**TELEBYTE0[7:0]**

TELEBYTE1

	7 ... 0
0x064	**TELEBYTE1[7:0]**

6.6 ISDN Interface

The ISDN DMA circuitry is valued as being a high bandwidth connection to the telephony interface and may be used for ISDN as well as analog telephone applications. It is initiated by clearing the ISDN reset bit in the ISDN control register, setting the ISDN interrupt enable signal and writing to both of the Next Address registers. This will immediately invoke a single word Output ISDN DMA transaction from memory into Harmony. However, all further transactions are triggered off the actions of Teleshare. ***Teleshare initiates** all **ISDN transactions across the CHI bus by first sending an ISDN byte***, waiting until Harmony responds with an ISDN byte and the process is repeated. If the order is ever interrupted, the whole process will be hung and the ISDN system will have to be reset in order to regain synch. There is not a fifo provided by Harmony for the purpose of ISDN for it does not need one. The protocol for Basic Rate ISDN (BRI) indicates a new byte is required every 125uSec, which is plenty of time for a DMA transaction to be completed. For a 712 system Harmony is guaranteed to obtain bus ownership every 20uS. As with the Audio DMA interface the ISDN channels utilize 4KByte page sizes of memory, however, it operates a little bit differently. If the system had to wait for an entire 4K buffer to be filled up before it could be informed there is new data available, this would be a real time delay of approximately .5 Sec. In order to shorten this delay, the ISDN DMA will interrupt the system every 1K of buffer space, which drops the delay to approximately .1 Sec and will be reasonably acceptable for real time applications that retransmit or reutilize this captured data for voice–related ISDN applications. The interrupt is removed when the ISDN Status register is read.

The ISDN DMA Software interface is very much similar to the Audio DMA interface and the concept of operation is comparable. It is composed of four address registers and a Status/Control register. System software should only access the Next address registers during normal operation, and the system will be interrupted when a new address is required. For a more detailed explanation please refer back to the Audio DMA register section.

OUTNXTADD (ISDN output next address) register

	31 … 12	11 … 0
0x070	**OUTNXTADD**	**0x000**

- **OUTNXTADD**: Full 32 bit physical address of the next page for input ISDN data. That page must be locked. Since this address points to the first location of a 4K byte page, the 12 LSBs must be 0s. This does *not* mean that any bit shifting is required on the address. During a write, the 12 LSBs of address are ignored by Harmony and the value for these bits is forced to 0.

OUTCURADD (output ISDN current address) register

	31 … 0
0x074	**OUTCURADD**

- **OUTCURADD**: Full 32 bit physical address for the next location to be written with recorded data. Writes are legal for test purposes only. **This register must be read with a single 32 bit operation to get coherent data.**

INNXTADD (ISDN input next address) register

	31 … 12	11 … 0
0x078	**INNXTADD**	**0x000**

- **INNXTADD**: Full 32 bit physical address of the next page for input ISDN data. That page must be locked. Since this address points to the first location of a 4K byte page, the 12 LSBs must be 0s. This does *not* mean that any bit shifting is required on the address. During a write, the 12 LSBs of address are ignored by Harmony and the value for these bits is forced to 0.

INCURADD (input ISDN current address) register

	31 … 0
0x07C	**INCURADD**

- **INCURADD**: Full 32 bit physical address for the next location to be written with recorded data. Writes are legal for test purposes only. **This register must be read with a single 32 bit operation to get coherent data.**

ISDN_STATUS / CONTROL register

	31	30	29 18	17	16	15 10	9	8	7 2	1	0
0x050	**IE**	**1K_Int**	**N/C**	**LOOP**	**RST**	**N/C**	**OUTN**	**OUTC**	**N/C**	**INN**	**INC**

- **IE**: A 1 in this bit enables ISDN DMA–based interrupts, a 0 disables them. This bit does not clear an interrupt; an ISDN DMA–based interrupt is cleared by either writing the **OUTNXTADD** and **INNXTADD** registers or by reading the ISDN Status register. Powers up as 0.
- **1K_INT:** Is active after every 1K bytes of ISDN data have been transferred to memory. It is cleared upon reading this register.
- **LOOP:** Induces ISDN loopback mode. During this time all data that is transferred from memory will be sent back to the INCURADD of memory. This loopback takes place within Harmony. It is important to note that in order to guarantee valid results the LOOP bit *must* be asserted before the ISDN DMA is initiated. This loopback mode is also totally contained inside of Harmony.
- **RST:** Resets the ISDN portion of Harmony.
- **OUTN**: ISDN Output New address request. It is set when a new value for **OUTNXTADD** is needed, and reset when **OUTNXTADD** is written. This bit is read only, writes are ignored. Powers up as 1.
- **OUTC**: ISDN Output DMA currently active. This bit is set when DMA is started, cleared when DMA finishes. This bit is read only, writes are ignored. Powers up as 0.
- **INN**: ISDN Input New address request. It is set when a new value for **INNXTADD** is needed, and reset when **INNXTADD** is written. This bit is read only, writes are ignored. Powers up as 1.
- **INC**: ISDN Input DMA currently active. This bit is set when DMA is started, cleared when DMA finishes. This bit is read only, writes are ignored. Powers up as 0.

7 RS–232 SERIAL INTERFACE

7.1 Introduction

The Serial interface emulates the National Semiconductor NS16550A device with some updates. The 712 implementation is based on a design done for the Stiletto chip used in the 742 VME computer

7.2 Feature Summary

This implementation includes all of the features of a standard NS16550A. See a commercial data sheet for detailed information. Virtually all of the interactions for the part are the same, so there is no need to rewrite already existing software. This implementation will have Receive and Transmit FIFOs each 16 bytes deep. Supported baud rates range from 50 to 454k. The hardware handshaking option will allow data input at up to 227k baud.

7.3 Register Definitions

Description	Offset	R/W	D7	D6	D5	D4	D3	D2	D1	D0
Reset Register	0x000	W	X	X	X	X	X	X	X	X
Undefined	0x001–0x7FF									
Receiver Buffer Register (RBR)	0x800 DLAB = 0	R	Data Bit 7	Data Bit 6	Data Bit 5	Data Bit 4	Data Bit 3	Data Bit 2	Data Bit 1	Data Bit 0
Transmitter Holding Register (THR)	0x800 DLAB = 0	W	Data Bit 7	Data Bit 6	Data Bit 5	Data Bit 4	Data Bit 3	Data Bit 2	Data Bit 1	Data Bit 0
Interrupt Enable Register (IER)	0x801	R/W	0	0	0	0	Enable MSI	Enable LSI	Enable THREI	Enable RDAI
Interrupt Ident Register (IIR)	0x802	R	Fifos Enabled	Fifos Enabled	0	0	Int ID Bit 2	Int ID Bit 1	Int ID Bit 0	Int Not Pending
Fifo Control Register (FCR)	0x802	W	Rx Trig MSB	Rx Trig LSB	X	X	DMA Mode	Tx Fifo Reset	Rx Fifo Reset	Fifo Enable
Line Control Register (LCR)	0x803	R/W	DLAB Bit	Set Break	Stick Parity	Even Parity	Parity Enable	Num of Stop Bits	Wrd Len Bit 1	Wrd Len Bit 0
Modem Control Register (MCR)	0x804	R/W	0	0	0	Loop Back	Unused	*See Note 1*	RTS	DTR
Line Status Register (LSR)	0x805	R	Error In Rx Fifo	Txmitter Empty	Tx Hold Reg Emp	Break Interrupt	Framing Error	Parity Error	Overrun Error	Rx Data Avail
Modem Status Register (MSR)	0x806	R/W	DCD (RLSD)	RI	DSR	CTS	Delta DCD	Trail Edge RI	Delta DSR	Delta CTS

Description	Offset	R/W	D7	D6	D5	D4	D3	D2	D1	D0
Scratch Register (SCR)	0x807	R/W	Scratch Bit 7	Scratch Bit 6	Scratch Bit 5	Scratch Bit 4	Scratch Bit 3	Scratch Bit 2	Scratch Bit 1	Scratch Bit 0
Divisor Latch Reg LSB (DLL)	0x800 DLAB = 1	R/W	Divisor Bit 7	Divisor Bit 6	Divisor Bit 5	Divisor Bit 4	Divisor Bit 3	Divisor Bit 2	Divisor Bit 1	Divisor Bit 0
Divisor Latch Reg MSB (DLM)	0x801 DLAB = 1	R/W	Divisor Bit 15	Divisor Bit 14	Divisor Bit 13	Divisor Bit 12	Divisor Bit 11	Divisor Bit 10	Divisor Bit 9	Divisor Bit 8
Undefined	0x808–0xFFF									

Table 2. RS232 Register Definitions

Note 1—Bit 2 of the MCR is a read/write bit which is used as the control bit for the Hardware Handshaking functionality, where 1 implies normal RTS operation and 0 implies hardware protocol. Note that the reset state for bits in this register is 0, which needs to be overridden for normal RTS operation.

7.4 Differences from NS16550A

The Lasi serial port is intended to function just like the real National NS16550A. This includes behavior that often seems rather stupid. The National NS16550A was chosen over the WD16C552 because the National part came first, and the WD part is supposed to be compatible. There are a few minor differences, however, between the NS16550A and this implementation.

Baud clock generation is an area of slight differences from the NS16550A. The baudrate reference frequency is taken from the 40 MHz IO system clock. The frequency 7.2727 MHz is generated by dividing 40 MHz by 5.5. This generates a slightly asymmetrical waveform (6:5 :: low:high). When changing baud rates, the NS16550A will glitch *baudclk*, but it does update to the new baud rate immediately. The 712 implementation will never glitch *baudclk*, but it takes as many as 5 baudrate reference clock periods for the new baud rate to be reflected on *baudclk*. The worst case result is that a transmitter or receiver could see as many as 3 extra, or 3 fewer 16x baudclk pulses than with the real NS16550A. If the baud rate is changed in the middle of a character transmission, the results are unpredictable. If the baud rate is changed and then a character is written into the transmitter buffer, the character will still come out properly, and meet NS16550A specs.

A few clarifications of hardware operations are also in order. The NS16550A supplies 2 status line, *Ntxrdy* and *Nrxrdy*. *Ntxrdy* is not used at all and will not be generated. *Nrxdy* is generated and used internally to qualify *rts_L* for hardware handshaking.

8 REAL–TIME CLOCK

8.1 Introduction

The Real–Time Clock (RTC) keeps track of time and date information when the system is powered down. A battery powered crystal oscillator operating at 32.768 kHz is used to keep an accurate record of elapsed time.

8.2 Feature Summary

The external crystal runs at 32.768 kHz and is divided down to 1 Hz by a 15 bit pre–counter. After the frequency of the crystal has been divided down to 1 Hz, this signal is then used to increment the 32 bit RTC register which keeps track of elapsed time in seconds. Under UNIX, this number represents the cumulative seconds that have elapsed since January 1, 1970. The register may be loaded with a known value representing the current time. This action will reset the 15 bit pre–counter to 0.

The standard Unix system call 'time' returns the current time as the number of seconds since 1970 and other routines map this number into various formats and national standards for date and time.

8.3 Register Definition

There is one 32–bit read/write register located at address offset 0x000. It does not make sense to reset the clock (equivalent to writing 0x00000000 to its register), so no means of reset is provided. Since the external 32 kHz clock is independent of the 712 system clock, it is possible to read or write the RTC register while it is changing. The write implementation will clear the 15–bit divide chain so that a full second will pass before the register is updated again. To read the Real_Time register, we suggest that several readings be taken; when 2 successive readings match, use that value.

9 PS2 INTERFACE FOR KEYBOARD/MOUSE

Introduction

LASI implements the keyboard and mouse interfaces as simple serial ports conforming to the de facto industry standard PS/2 specification. Each user input device has a dedicated serial port of its own. LASI includes two ports, one for keyboard and one for mouse. The interface ports rely on the software to provide all of their intelligence, therefore, they do not interpret the characters passing through them in either direction. The interface to the host processor is through 6 one–byte registers for each port.

Registers

Register Label	Register Name	Address offset (word aligned)	Reset Value	Access
ID	ID Register	0x00	*note 1*	R
RESET	Interface reset register	0x00	0xXX	W
RCVDATA	Received data register	0x04	0xXX	R
XMTDATA	Transmit data register	0x04	0xXX	W
CONTROL	Control register—read/write	0x08	0x00 *note 2*	R/W
STATUS	Status register—read only	0x0c	0x00	R

Note 1: Each PS2 device returns a unique hardwired ID code in bits 3:0 of ID.
Note 2: Resetting the block disables it (see Table 6).

Table 3. PS2 Interface Registers

ID Register

Bit	Symbol	Name	Description
3:0	ID	ID Code	Hardwired physical identification bits. (0=keyboard;1=mouse)
7:4	Reserved		

Table 4. PS2 ID Register

Reset Register

Any write to this register will cause the interface to be reset: all buffers will be emptied and the receive/transmit state machines will be reset. The value of the data written will be discarded. Note that this does not cause the external device to be reset; it must be reset explicitly through a command sent from the host. A reset will leave the interface in the disabled state.

Rcvdata Register

This register provides access to the received data buffer in the interface. Each read operation from this register removes one character from the received data buffer. Receive Buffer Not Empty (bit 0 of the STATUS register) is set to 1 whenever there is one or more characters in the receiver buffer and returns to 0 when the buffer is empty. The port asserts its interrupt line when the receive buffer goes from empty to not empty. It is the responsibility of the host to read the rcvdata register until the Receive Buffer Not Empty bit returns to a 0 value. Note that a port does not generate an interrupt for each received character, only when the first character is placed in an empty buffer. LASI implements the Received Data Buffer with 4 characters of storage.

Xmtdata Register

This register provides data to the transmitter section of the interface. Since each character transmitted requires an acknowledge character, the transmit buffer is only one character deep. The PS2 protocol allows for outgoing data to interrupt and override any incoming data. LASI will receive any incoming character before starting the transmit process. The transmitter will assert control as soon as a received character is finished and send the character in the transmit buffer. Transmit Buffer Not Empty (bit 1 of STATUS register) is set to 1 when there is a character in the transmit buffer. It is the responsibility of the host to determine when the buffer is empty by monitoring TBNE, which will go to 0 when the buffer is empty. A PS2 port will ignore an attempt to write to the Xmtdata register while TBNE is 1. No interrupts are generated in the transmit process, except for acknowledge characters returned by the external device.

Control Register (R/W)

Bit	Symbol	Name	Description
0	ENBL	Enable	Set status of interface: 0 = disabled, 1 = enabled. When disabled, port will neither receive data from external device nor generate interrupts to host.
1	LPBXR	Loopback Xmt/Rcv mode	Set to 1 for loopback diagnostic mode: transmitter output is connected to receiver input. In this case, a much faster internal clock is used to transfer the data (2 MHz).
4:2	Reserved		
5	DIAG	Diagnostic mode	When set to 0, bits 6 and 7 have no effect on the external Data and Clock lines. When set to 1, bits 6 and 7 directly control the value of the external Data and Clock lines for diagnostic purposes.
6	DATDIR	External data line direct control	Provides direct control of value of external Data line while in diagnostic mode.
7	CLKDIR	External clock line direct control	Provides direct control of value of external Clock line while in diagnostic mode.

Table 5. PS2 Control Register

Status Register (Read only)

Bit	Symbol	Name	Description
0	RBNE	Receive buffer not empty	0 = receive buffer empty, 1 = receive buffer not empty
1	TBNE	Transmit buffer not empty	0 = transmit buffer empty, 1 = transmit buffer not empty
2	TERR	Timeout Error	Normally set to 0. See Note 2
3	PERR	Parity Error	Normally set to 0. See Note 1
4	CMPINTR	Composite interrupt	OR of interrupt lines of all PS2 ports
5	Reserved		
6	DATSHD	Data line shadow	Copy of current value of external data line
7	CLKSHD	Clock line shadow	Copy of current value of external clock line

Table 6. PS2 Status Register

Note 1:

When the receiver detects a parity error, bit 2 of STATUS register is set to 1. The incorrect character is placed in the receive buffer. Busy is asserted to the external device to halt any further transmissions. The host must recognize the parity bit and empty the receive buffer since there may be valid characters in the buffer ahead of the incorrect one. The last character in the buffer is always the incorrect one. In order to clear the parity error, the interface must be reset. It is the responsibility of the host to request a resend from the external device if desired.

Addressing

One 4Kbyte block of the 712 IO space is allocated for both PS2 interfaces. Each interface is assigned a 256 byte block of its own. The keyboard port is located a offset 0x000 within the 4K block and the mouse is located at offset 0x100. There is no logical difference between the keyboard and mouse interface other than their address and the contents of the ID register. All registers are 1–byte wide and are aligned on word boundaries. All addresses are multiply–mapped so that a read/write to any address in the 4k block will access a legitimate register.

Interrupt processing

The LASI interrupt handler supports only one interrupt line for both PS2 ports. The host must poll both ports to determine which ones have data. Each port asserts its interrupt line until its buffer is empty. Both of the interrupt lines are OR'd together to make a single interupt signal. An interrupt will be issued when it sees a positive edge on this signal. Thus there is an interrupt for the first character to arrive at either port, but none for following characters until both of the ports have been emptied. The host must cycle through both ports, emptying the data from each until both are empty, including data which arrives during the service process. The host determines that both ports are empty by the Composite Interrupt signal (*cmp_intr*), which is copied into bit 4 of the STATUS register of both ports. Since the host must continue to read data from ports until *cmp_intr* is 0, the next arriving character forces *cmp_intr* to 1, which causes an interrupt. It is anticipated that in most cases each arriving character will cause an interrupt and will be serviced before the next character arrives.

Timing

The PS2 specification maximum transfer rate is about 1 character per millisecond (80 microseconds per serial bit for 11 bits plus some overhead). The fastest input expected from a keyboard would be about 16 characters per second. If the keyboard generates 3 characters per keypress (character code on downstroke, up code on release, followed by character code again), then we can expect characters about every 20 ms. A mouse is programmable for sample spacing (20 to 200 samples per second, default 100) and sends 3 characters for each XY sample. Thus a mouse might push close to the 1 ms character spacing. There should be no problems responding to interrupts at this rate.

10 PC FLOPPY

10.1 Introduction

The 712 supports an optional floppy disk drive. This drive is for use with soft PC and allows a means for transferring data between machines (eg. between a 712 and a portable). The floppy disk drive (FDD) is controlled by a Western Digital WD37C65C Floppy Disk Controller chip (FDC). The processor accesses the FDC registers through Lasi. Lasi provides a restricted form of DMA for moving data between the FDC and main memory.

10.2 DMA Operation

Floppy data is transferred in the 712 by moving data between a 1 page (4K byte) circular buffer in main memory and the FDC. The floppy interface in Lasi provides a 4–byte FIFO to deal with latency associated with accessing main memory without causing data overrun errors in the FDC. The Floppy DMA address register (FDAR) serves as the ”head pointer” for the circular buffer. The FDAR always contains the memory address of the next word of data to be moved between main memory and the FDC. The bits of the FDAR are defined in figure 4. The circular buffer is located in main memory at a page address specified by the 20 most significant bits of the FDAR. The data in the buffer is stored as un–packed bytes. Word transactions are used on GSC, but only the most significant byte contains floppy data. Therefore, only 1K of data can be stored at a time in the circular buffer.

31 12	11 2	1	0
Page Address	Page Offset	0	DMA_EN

Figure 4. Floppy DMA Address Register

Bits 11:2 of the FDAR provide the page offset of the head pointer. The page offset will increment after every data transfer. Bit 0 of the FDAR is the DMA enable bit (DMA_EN). This bit is read only in the FDAR. Data will only be transferred if the DMA_EN bit is set to 1. The DMA interrupt signal will be true any time the page offset is 0x3ff or 0x1ff. This signal will cause an interrupt to be issued if the DMA_IE bit is set to 1 in the Floppy Control Register and bit 20 of the Interrupt Mask Register (in the interrupt block) is set to 1. The DMA interrupt is cleared by setting the DMA_IE bit to 0 while the page offset in the FDAR is any value other than 0x3ff or 0x1ff.

The direction of the DMA is determined by the read enable (RE) and write enable (WE) bits in the Floppy Control Register. When the RE bit is set to 1, data will be moved from the FDC to main memory (provided DE=1). When the WE bit is set to 1, data will be moved from main memory to the FDC (provided DE=1). RE and WE should not be set to 1 at the same time.

The DMA controller in Lasi has no indication of how many bytes of data to transfer. It simply attempts to keep the 4 byte FIFO in Lasi's floppy interface either full (WE=1) or empty (RE=1) depending on the direction of the data transfer.

Note: The FIFO will most likely be full following a DMA operation from main memory to the FDC. The FIFO can be emptied with slave reads of the FIFO data registers to avoid using the data on subsequent transfers.

The FDC controls the amount of data that is actually transferred. It can be programmed to move one or more sectors and will interrupt upon completion. The FDC must be used in a mode that does DMA (see the WD37C65C for details). When the FDC tells Lasi that it is ready to transfer data, Lasi transfers the data between the FDC and the 4–byte FIFO. Lasi's floppy DMA mechanism completes the transfer by moving data between the circular buffer in main memory and the FIFO.

10.2.1 Servicing the Circular Buffer

It is up to software to keep the circular buffer from overflowing or underflowing when transferring more than 1K bytes of data. This can be accomplished by using the two interrupts that can be generated by the circular buffer. One interrupt is generated as the page offset rolls over the page boundary and the other is generated as the page offset rolls over the half–page boundary. This allows software to fill/empty one half of the page while DMA is working on the other half of the page. This scheme allows for very long interrupt response times (> 6mS) without data overrun errors. If there should be a data overrun error, it will be indicated in the FDC and the sector operation will need to be repeated.

10.3 PC Floppy Registers

The PC Floppy contains three types of registers. The FDC registers are byte registers and are physically located on the WD37C65C. They are accessed through Lasi in the 4K block at address offset 0xa000. The DMA registers are both 32–bit registers that are in the 4K block at address offset 0xd000. The remaining registers are byte registers and are also in the 4K block at address offset 0xa000. Each register is described below.

10.3.1 FDC Registers

All of the FDC registers can be accessed directly with byte reads and writes. Table 7 lists the FDC registers and their address. See the WD37C65C specification for more detail as to the exact bit definition and functionality of each register.

FDC Register	Address Offset	R/W
Master Status Register	0xa008	R
Control Register	0xa020	R/W
Operations Register	0xa040	W
Data Register	0xa004	R/W

Table 7. Floppy Disk Controller Registers

Floppy Registers	Address Offset	R/W
Control Register	0xa018	R/W
Status Register	0xa01C	R
FIFO Data Register	0xa014	R/W

Table 8. Floppy Registers

DMA Registers	Address Offset	R/W
DMA Address Reg	0xd000	R/W
DMA enable Reg	0xd004	R/W

Table 9. DMA Registers

Bit	Symbol	Description
0	WE	write enable – Setting this bit, establishes the direction of DMA as from main memory to the FDC[1].
1	RE	read enable – Setting this bit, establishes the direction of DMA as from the FDC to main memory.
2	FDC_IE	Allows interrupts generated by the FDC to be forwarded to the processor.
3	DMA_IE	Allows the DMA address register to generate an interrupt when rolls–over the page and half–page boundary.

Table 10. Floppy Control Register bit Definition

Bit	Symbol	Description
0	FULL	Indicates that the FIFO is full.
1	EMPTY	Indicates that the FIFO is empty.
2	DMA_IRQ	Indicates that the DMA address register generated an interrupt. This bit can only be cleared by clearing the DMA_IE bit and setting the PAGE_OFF bits of the DMA address register to a non–interrupting values (anything but 0.

Table 11. Floppy Status Register bit Definition

1. The write enable bit must be set in order to do slave writes from the processor directly into the FIFO.

Bit	Symbol	Description
0	DMA_EN	DMA enable bit– **Read Only**
1		This bit is always 0 – **Read Only**
11:2	PAGE_OFF	These bits specify the word offset within the memory page specified by PAGE_ADR – R/W
31:12	PAGE_ADR	These bits specify the memory page to use for floppy DMA – R/W

Table 12. Floppy DMA Address Register bit Definition

Bit	Symbol	Description
0	DMA_EN	DMA enable bit– R/W
1		This bit is always 0 – **Read Only**
11:2	PAGE_OFF	These bits specify the word offset within the memory page specified by PAGE_ADR – **Read Only**
31:12	PAGE_ADR	These bits specify the memory page to use for floppy DMA – **Read Only**

Table 13. Floppy DMA Enable Register bit Definition

11 FLASH EPROM

11.1 Overview

One half of the Lasi address space (1 Mbyte) maps to the 8–bit external bus. The 712 uses this space only for processor dependent code and the stable–store. Both will be implemented using flash EPROM technology. Past implementations required two EPROMS with sockets and an EEPROM to implement PDC and stable–store. The 712 will use two un–socketed Flash EPROM parts instead. This is a lower cost implementation with additional features such as remote PDC updates and additional flexibility during turn–on and test.

The Flash EPROM resides on Lasi's external 8–bit bus. The processor accesses the Flash EPROM via GSC. Reads from PDC space can be either double–word, word, or byte accesses. All writes to the Flash EPROM must be byte accesses. Software must provide all the control for erasing and programming new data.

11.2 Memory Map

Lasi's external 8–bit bus provides 1M byte of address space with 19 address signals and three chip select signals. One chip select signal decodes the lower half of the 1M address space. The second chip select signal decodes the next 256K bytes of the address space. The third chip select signal decodes the last 256K bytes. The third chip select signal is multiplexed with the address latch signal. This allows up to 1M byte of EPROM space in three different devices without external decode logic or configuration bits. External logic could be added to the external 8–bit bus to support more than three devices. The Gecko implementation will provide 256K bytes of storage in two Flash EPROM devices.

The target Flash EPROM device for the 712 is the AM29F010 from AMD. Using this device, the address space is segmented into 16 different areas. All of these segments can be modified by software. Figure 5 is a memory map for the Flash EPROMs. If other devices are used, the segment locations will most likely be different. Notice that the EPROM space in the 712 is not contiguous.

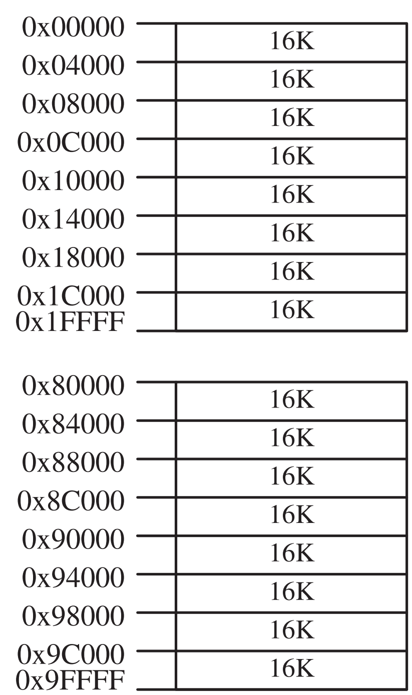


Figure 5. Flash EPROM Memory Map

11.3 Performance

11.3.1 Reading Data

The Flash EPROM devices are relatively slow and are accessed via an 8–bit bus. As a result, the request for a double word of data from the Flash Eprom will be very slow. Data will be returned to the processor approximately 75 GSC cycles after it is requested.

11.3.2 Writing Data

Writing data to the Flash EPROM requires the software to erase the entire segment and then write the data back a byte at a time. The timing required for the writes is built into the the Flash EPROM device. Software must write a command sequence to the device to get it to erase a sector or program a byte. The device is thenpolled to determine when the write is complete. Erasing and re–programming the entire Flash EPROM device is a relatively slow process and may take a couple of seconds. Refer to the Am29F010 specification for more detailed information.

12 POWER SYSTEM SUPPORT

12.1 Overview

The 712 I/O system provides support for the power system in two areas.

- The Lasi chip receives the asynchronous PON signal from the power supply and drives that signal to the other GSC devices as a synchronous RESET signal.
- Lasi provides an interface between the CPU and the power supply allowing a smart power switch. When the switch is turned off, the CPU will shut the system down gracefully before power is actually removed.

12.2 Reset

In the 712, the primary Lasi chip receives the asynchronous PON signal directly from the power supply. The primary Lasi chip then must synchronize the rising edge of this signal to the system clock and drive the signal as the RESETL signal on GSC. Note that Lasi must drive this signal at the full processor frequency to insure that all GSC devices are in phase.

12.2.1 RESETL Configurability

The RESETL signal can be configured on Lasi as either an input or an output by setting the RST_SLAVE bit (bit 0 of the external 8–bit bus) at power–up. If this bit is high, Lasi will use the RESETL signal as an input. This is necessary if another device in the system is used to control reset as would be the case for a second Lasi used on the personality card. To make this work without drive fights, Lasi must always drive the RESETL signal low when its PON input is driven low. The PON input on the second Lasi must also be tied to PON from the power supply. The RESETL signal will be an output if the RST_SLAVE bit is low during reset and will be an input if it is high. Both Lasi's will try to drive the RESETL signal for a short time during power–up, but they will both be driving it low. Bit 0 of the external 8–bit bus will be latched in as the RST_SLAVE bit on the rising edges of the RESETL signal.

12.2.2 I/O Reset

When RESETL is low, the I/O reset register will be cleared and all Lasi internal devices will be reset (except RTC).

Software can cause a hard reset to any most of the internal I/O devices by writting to the I/O reset register located at address offset 0x10C00C.

The I/O reset register will be cleared, (all I/O reset register outputs = “0”), when RESETL is “0”. Before accessing an I/O reset register device, PDC will have set the appropriate I/O reset register device bit to a “1”.

Table 14 defines each of the bits in the I/O reset register.

Bit	Symbol	Description
31–9		Undefined
8	RST_ARTIST_L	Reset Artist chip
7	RST_KBD_L	Reset keyboard
6	RST_RS232_L	Reset RS232
5	RST_EXT	Reset FDC, LAN Buf, and tri–stat main memory
4	RST_INTR_L	Reset Interrupt
3	RST_AUD_L	Reset Audio
2	RST_PAR_L	Reset Parallel
1	RST_LAN_L	Reset LAN
0	RST_SCSI_L	Reset SCSI

Table 14. I/O Reset Register

The RS232, LAN, and parallel I/O devices can also be reset for 15 GSC clock cycles by writing to address offset 0x000 of their respective 4K page.

The parallel port dma registers will be reset by writing to address offset 0x10300.

12.3 Smart Power Switch

The front panel power switch on Gecko is a logical switch that is qualified by the pwr_on_L signal from Lasi. The diagram in figure 6 shows the logical representation of the power switch circuit.

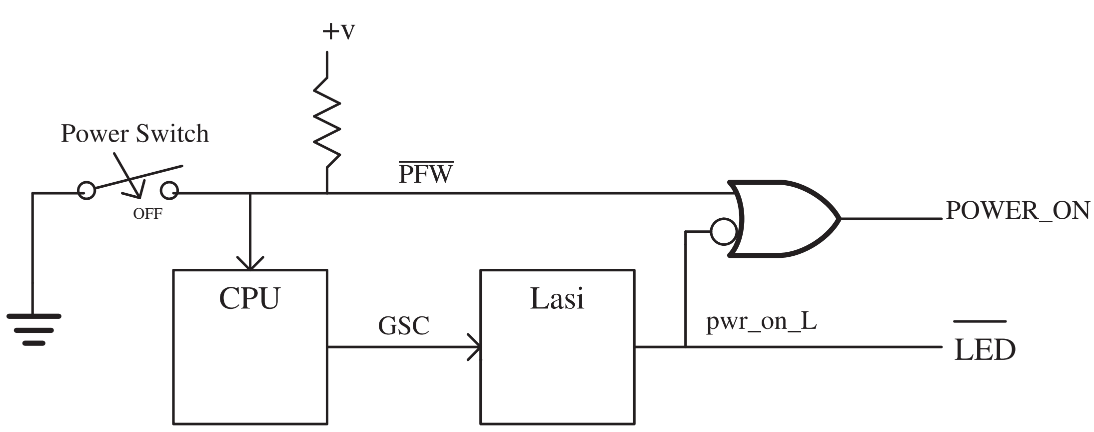


Figure 6. Power–down circuit logic

When the power switch is turned on, the logical POWER_ON signal will activate the primary switcher in the power supply, causing the DC supplies to come into spec. When the power switch is turned off, the power fail warning interrupt will be issued to the CPU. This power fail warning interrupt signal will come directly from the switch and will therefore be completely asynchronous to the system clock. The signal will not be "de–bounced" before it is sent to the CPU so it will most likely make several transitions before settling to a low level. The software must be able to handle this situation.

The CPU will do all the tasks necessary for a clean shut down of the system and will then set the PWR_ON_L bit to a "1" ,in the power control register (PCR). The PCR is located at address F010 C000. Table 15 shows the bit definitions for the PCR.

12.4 LED Control

The pwr_on_L signal is used for LED control to indicate possible system conditions. If PWR_ON_L is "0" and FLASH_LED is "0" then the LED is on. If FLASH_LED is "1" and PWR_ON_L is "0" then the LED will flash at 2Hz.

Bit	Symbol	Description
31–2		Undefined.
1	PWR_ON_L	The bit is set by software to shut down the power supply. It is set at reset to a "0".
0	FLASH_LED	The bit is set by software to gate a 2Hz signal to the LED. Set to a "1" at power up.

Table 15. Power Control Register bit Definition

13 MISCELLANEOUS REGISTERS

13.1 I/O Configuration Registers

The I/O configuration registers are two 8–bit read–only registers provided in Lasi to allow PDC to identify which I/O devices in the second Lasi are enabled (have connectors). PDC can read these registers and determine which I/O functionality can be used in the second Lasi. These registers are only present when there is a second Lasi in the 712 system (SPACE[1:0]=01).

13.2 Setting the Registers

The configuration registers are two read–only byte registers located at address offset 0x7FFFE and 0x7FFFF. The second register is optional. This is in the address space that would normally be occupied by the Flash EPROM devices in Lasi. If only the first register is needed, it can be implemented by using resistors to bias the signals. If both registers are required, some active logic is required to implement the second register.

13.2.1 Primary I/O Configuration Register

The first configuration register bits are set by using resistors on Lasi's 8–bit external bus to pull the un–driven signal corresponding to each bit either high or low. The resistors on bits 7–4 are only required on the second Lasi because the primary Lasi will always have a Flash EPROM device on this bus. The resistors on bits 0–3 are required for both the primary and the secondary Lasi for hardware configuration at power–up. These resistors will normally be pulled low on the primary Lasi. The second configuration register (at offset 0x7FFFF) is optional and is defined to exist only when bit 7 is a one. If bit 7 of the first configuration register is a 0, it can be assumed that all bits in the second configuration register are 0.

The table 16 defines each of the bits in the primary configuration register.

Bit	Symbol	Description
7	IO_CONF_REG2	Indicates that there is a second I/O configuration register.
6	LAN	Indicates that the secondary LAN can be used.
5	X.25	Indicates that X.25 can be used with the second Lasi.
4	RS232	Indicates that the secondary RS232 can be used.
3	SPACE[1]	Indicates Lasi base address.
2	SPACE[0]	Indicates Lasi base address.

1	ARB_SLAVE	Disables the arbitration circuit in Lasi and forces it to arbitrate through the EREQUESTL and EGRANTL pins on Lasi.
0	RST_SLAVE	Causes the RESETL to be an input.

Table 16. I/O Configuration Primary Register Bit Definition

13.2.2 Secondary I/O Configuration Register

The secondary I/O configuration register is at address offset 0x7FFFF. This register is a read–only byte register defined to exist only if bit 7 of the primary I/O configuration register is set. This register is implemented using a discrete component (such as a simple PAL or a ROM) that lives on the 8–bit bus.

Table 17 defines each of the bits in the secondary configuration register.

Bit	Symbol	Description
7–5	Undefined.	
4	TOC	Indicates that the card is being used for TOC functionality.
3	SCSI	Indicates that the secondary SCSI interface can be used.
2	AUDIO	Indicates that the secondary audio/phone interface can be used.
1	FLOPPY	Indicates that the secondary floppy interface can be used.
0	PAR	Indicates that the secondary parallel port can be used.

Table 17. I/O Configuration Secondary Register Bit Definition

13.3 Error Logging Register

The only error detection capability built in to Lasi is GSC parity and time–out. If Lasi detects an error when some other device is bus master, Lasi will pull the GSC ERRORL line low and no other action will be taken by Lasi. If the ERRORL signal is pulled low when Lasi is the bus master, an interrupt will be generated by Lasi.

If a bus error occurs while Lasi is the bus master, a bit is set in the Error Logging Register that corresponds to the Lasi internal device that was active when the data error occurred. Table 18 defines the bits of the Error Logging Register. This read/write register is at address offset 0x10C004. This register must be cleared by software following an error.

The Error Logging Register is accessed as a word register.

Bit	Description
31–7	Undefined.
6	Indicates that Floppy was the bus master when the error occurred.
5	Indicates that Parallel was the bus master when the error occurred.

4	Indicates that ISDN was the bus master when the error occurred.
3	Indicates that Audio/Phone was the bus master when the error occurred.
2	Indicates that Interrupt was the bus master when the error occurred.
1	Indicates that LAN was the bus master when the error occurred.
0	Indicates that SCSI was the bus master when the error occurred.

Table 18. Error Logging Resister

13.4 Lasi Version Control Register

The Lasi Version Control Register is a read–only 4bit register, (bits 3–0), at address offset 0x10C008. This register is accessed as a word register.

This register will always return a zero in the initial Lasi chip. The register's value will be incremented if there are additional revisions of the Lasi chip.

14 INTERRUPTS

14.1 Overview

External interrupts in the 712 are always sent to the processor via a GSC write transaction to the IO_EIR. The IO_EIR is a five bit register physically located inside the processor. The five bit value written to the IO_EIR indicates which bit of the EIR (CR23) will be set. The address of the IO_EIR is fully programmable in Lasi. Either Lasi or the personality card can master the write transaction. The personality card will contain either a second Lasi chip or a Wax chip (Token Ring).

14.2 Register Definitions

There are five registers in Lasi associated with interrupts. The registers are defined below in table 19.

Register	Symbol	Address Offset	R/W	Description
Interrupt Request Register	IRR	10 0000	R	The IRR contains the status of all requesting interrupts. A 1 in an IRR bit indicates that the corresponding interrupt is pending and enabled. When an IRR bit is set it will cause Lasi to generate an interrupt transaction.
Interrupt Mask Register	IMR	10 0004	R/W	The IMR is used to mask pending interrupts. A 1 in an IMR bit enables the corresponding pending interrupt to create an interrupt request.
Interrupt Pending Register	IPR	10 0008	R/W	The IPR is used to latch incoming interrupts and indicate them as pending. An active edge on an internal interrupt signal causes the corresponding IPR bit to be set to 1. Writes to this register are intended for diagnostic use only and will cause the entire register to be cleared.

Register	Symbol	Address Offset	R/W	Description
Interrupt Control Register	ICR	10 000C	R/W	The ICR is used to indicate if the system is running HP–UX or HP–RT.
Interrupt Address Register	IAR	10 0010	R/W	Bits 31:5 specify the address of the IO_EIR register (4:0=0). Bits 4:0 specify the data to be written. The power–up value is 0xfffbe003.

Table 19. Interrupt Registers

The interrupt registers appear to be 32–bits and are accessed as such. However, not all of the bits are implemented for each register. The un–implemented bits are not affected by writes and are always read as zeros.

14.3 Interrupt Operation

The interrupt mechanism in Lasi operates in two different modes. The mode is determined by the RT bit. The RT bit is set by software writing to the ICR. Both modes will cause at least one write transaction to the address specified by bits 31:5 of the IAR when a bit is set in the IRR.

If an interrupt source in Lasi wants to interrupt the processor, the corresponding bit in the IPR will be set to 1. If that interrupt is enabled (IMR bit=1), the same bit of the IRR will be set to 1 and a write to the location specified by bits 31:5 of the IAR will be initiated. The data written will be specified by bits 4:0 of the IAR.The contents of the IRR and unmasked bits (IMR bit=1) of the IPR are cleared on the clock cycle following a read of the IRR.

NOTE: RT mode is broken in current silicon.

14.4 Interrupt Register Bit Assignments

Table 20 shows the implemented bits for the ICR in Lasi. The interrupt controller will never request mastership of GSC when the bus_error bit is set. The interrupt controller can effectively be disabled by setting this bit. Also, this bit must be cleared by software if normal interrupt operation is desired following a bus error. The TOC bit can be set to turn the external interrupt signal into a TOC signal. External interrupts must be enabled for a TOC to be issued.

Bit	Symbol	Description
0	Reserved	All writes to this bit location must be a 0
1	TOC	This bit re–maps the external interrupt signal to TOC when set. This bit is write–only and will always return a 0 when read.
8	bus_error	Indicates the interrupt controller received a bus error during a write to the IO_EIR. This bit disables any additional interrupts when set.

Table 20. Interrupt Control Register Bit Definition

Table 21 shows the implemented bits for the IRR, IMR and IPR in Lasi. This table shows GSC bit definitions. Note that these are not the same bit numbers as are seen in a PA register.

Bit	Interrupt Source
5	RS–232
7	Parallel Printer Interface
8	LAN
9	SCSI
13	Audio
14	Bus Error
16	Telephone 0
17	Telephone 1
18	ISDN
19	SCSI Snoop
20	PC Floppy
21	External Interrupt
26	Keyboard/mouse

Table 21. IPR, IMR, and IRR Bit Definition

14.5 Error Handling

If a GSC bus error occurs while Lasi is the bus master and the bus error interrupt is enabled, an interrupt will be issued and the bus error bit will be set in the IRR.

If a bus error occurs while the interrupt block is doing a write to the IO_EIR, the bus_error bit will be set in the ICR and no further interrupts will be issued. In this case, the CPU is never informed by Lasi that an error occurred. PCXL will, however, issue an HPMC when it sees the error.

15 GSC INTERFACE

15.1 Interface Overview

Lasi, the PA7100LC CPU, Artist graphics, and an expansion slot all reside on the Gecko System Connection (GSC). This section describes the hardware Lasi uses to interface to GSC and the connection features Lasi uses.

15.2 Connection Control Path

The Lasi GSC interface can be either a connection master or a slave. When it is a slave it will receive addresses and respond with a ready signal. When it is a master it generates addresses and receives the ready signal.

While in slave mode, the GSC interface will derive internal control signals from external GSC signals using combinational logic and a simple state machine. The interface will assume that all LASI internal slave devices have a common timing behavior. The number of GSC states needed to do a slave transaction varies depending on the internal device accessed.

As a master, Lasi's GSC interface depends heavily on its internal devices for the generation of addresses and internal control signals. In this mode the interface serves as a means of translating signals to those used by GSC, generating parity, dealing with bus errors, and providing the correct timing. The GSC interface will be built to maximize SCSI performance.

15.2.1 Control Simplification Ideas

In order to simplify the control design Lasi's GSC interface will not use the GSC LSL signal to split GSC transactions. Lasi avoids using the split signal by not granting its internal bus before getting the external bus. Doing this will increase the amount of time between bus grant and the first address valid of a DMA transaction.

Lasi will run its internal bus at the GCLK frequency to avoid synchronizing to signals on the GSC interface. All internal Lasi devices must be able to meet the timing requirements imposed by the internal bus. The GSC interface does not fifo the data going to the GSC bus so internal devices need to transfer data fast enough to keep up.

15.3 Lasi's Data /Address Path

As a slave, GSC data and addresses will be separated within the GSC interface. The interface latches addresses and presents them to the rest of Lasi for decoding. Depending on the GSC transaction type, the interface generates data cycles for internal slaves. Since Lasi needs to support up to 2–word GSC slave reads, the interface will provide a two word buffer.

Only byte, word, and double word transactions are currently being considered for internal slave devices, however, Lasi still needs to respond to other transactions, even illegal ones. The interfaces response to an unsupported transaction will be to generate valid data and data parity, but the data may not be predictable.

While in master mode, the Lasi GSC interface will support all necessary GSC transactions. Lasi's internal bus interface will be synchronous to GSC; this forces all internal master devices to generate data at a rate equal to the imaginary GCLK frequency.

15.4 LASI GSC Behavior Summary

For writes to which Lasi is a slave READYL will always be asserted, if the address is valid. Since Lasi is the data sink for this type of transaction it will assert ERRORL if bad data parity is detected. Lasi properly responds to all types of GSC write transactions for which it receives a valid address

When Lasi is a slave to a read transaction READYL will always be asserted, if the address is valid. Since Lasi is the data source for this type of transaction it is supposed to look at ERRORL , however, it doesn't do anything differently if an error happens. Lasi properly responds to all types of GSC read transactions for which it receives a valid address.

As a bus master for write transactions data Lasi needs to watch for READYL and ERRORL while writing data to the slave. Since the slave is the data receiver it will signal data parity errors using ERRORL , and the CPU will generate ERRORL if no devices respond with READYL within the timeout period. Lasi distinguishes between timeouts and data parity errors by knowing when to expect a data parity error for each data word written. In the case of a timeout Lasi uses ERRORL as a ready indication, this gets the state machine unstuck. In either case Lasi goes through the motions of transferring data as if the error had not occurred.

When Lasi is the bus master for read transactions it looks for the assertion of READYL as an indication that valid data is on the bus. If READYL does not happen (because of a timeout) the CPU will assert ERRORL which in this case is a substitute for READYL, and the transaction will complete by pretending to transfer the correct amount of data. Bus–holders in the CPU keep valid data and parity on the GSC interface.

Lasi's GSC interface looks at the ERRORL signal much the way it looks at READY. Timed–out transactions complete normally except that an error is logged. Transactions with data parity errors also complete normally except that an error is logged. ERRORL is only asserted by Lasi as the result of a data parity error when Lasi is the data sink (Master Reads & Slave Writes). To summarize, errors help get the GSC interface unstuck if there is a timeout but do not directly affect interface operation, rather they affect the internal devices attached to the interface and the processor.

16 ARBITRATION

16.1 712 Arbitration Overview

The 712 System Connection (GSC) uses a central arbitration scheme. The arbitration controller is located inside the Lasi chip for the 712. There are eight potential bus masters in the 712. Six of the eight are internal to the Lasi chip. All devices serviced by the arbitration controller are given equal priority. The CPU is the only device that will be granted the bus without having requested it. Table 22 identifies the potential bus masters in the 712.

Device	Location	Signal Sense
CPU	outside Lasi	negative true
Expansion Slot	outside Lasi	negative true
SCSI	inside Lasi	negative true
LAN	inside Lasi	positive true
Parallel	inside Lasi	positive true
Audio	inside Lasi	positive true
Interrupts	inside Lasi	positive true
floppy	inside Lasi	positive true

Table 22. 712 Bus Masters

16.2 Arbitration Mask Register

The arbitration mask register (AMR) is a read/write register located at address offset 0x10C010. The AMR provides a means to prevent a device from being granted the bus. Table 23 defines the AMR bits. If a bit in the AMR is set to a 1, the request is disabled and the associated device(s) will never be granted the bus. All of the internal Lasi devices (except interrupts) are masked with the IRM bit. The CPU's bus request signal can never be masked. Interrupt request signals are not masked because the interrupts can be masked by writing to the interrupt mask register (IMR).

Bit	Symbol	Description
31–2		Undefined

1	ERM	External request mask – if ERM=1, the expansion slot will never be granted the bus.
0	IRM	Internal request mask – if IRM=1, none of the devices internal to the Lasi chip will be granted the bus (except interrupts).

Table 23. Arbitration Mask Register

All non–CPU devices power–up with arbitration disabled. Software must clear the AMR bits before any other device can request the bus.

The IRM bit will be set by Lasi if a bus error occurs on GSC when Lasi is the bus master. No additional internal bus grants will be issued after the IRM bit is set. Devices that already own the bus may issue new transactions after the bit is set if they already are the bus master.

16.3 Disabling the Arbitration Controller

The Lasi chip can be configured to work on a GSC bus when it is not the arbitration controller. The configuration bit ARB_SLAVE can be set high if Lasi is not the controller. The ARB_SLAVE bit must be 1 on the second Lasi in a 712 system.

When Lasi is not the arbitration controller, the CGRANTL signal becomes the GSC bus request signal and the CREQUESTL signal becomes the GSC grant signal. In this mode, Lasi will only grant its internal bus to a device if Lasi has been granted the bus via its GSC grant signal. All internal devices will have equal priority when Lasi is not the arbitration controller. If all devices in both Lasi chips requested the bus at the same time, all devices would get the bus exactly once before any device received it twice.

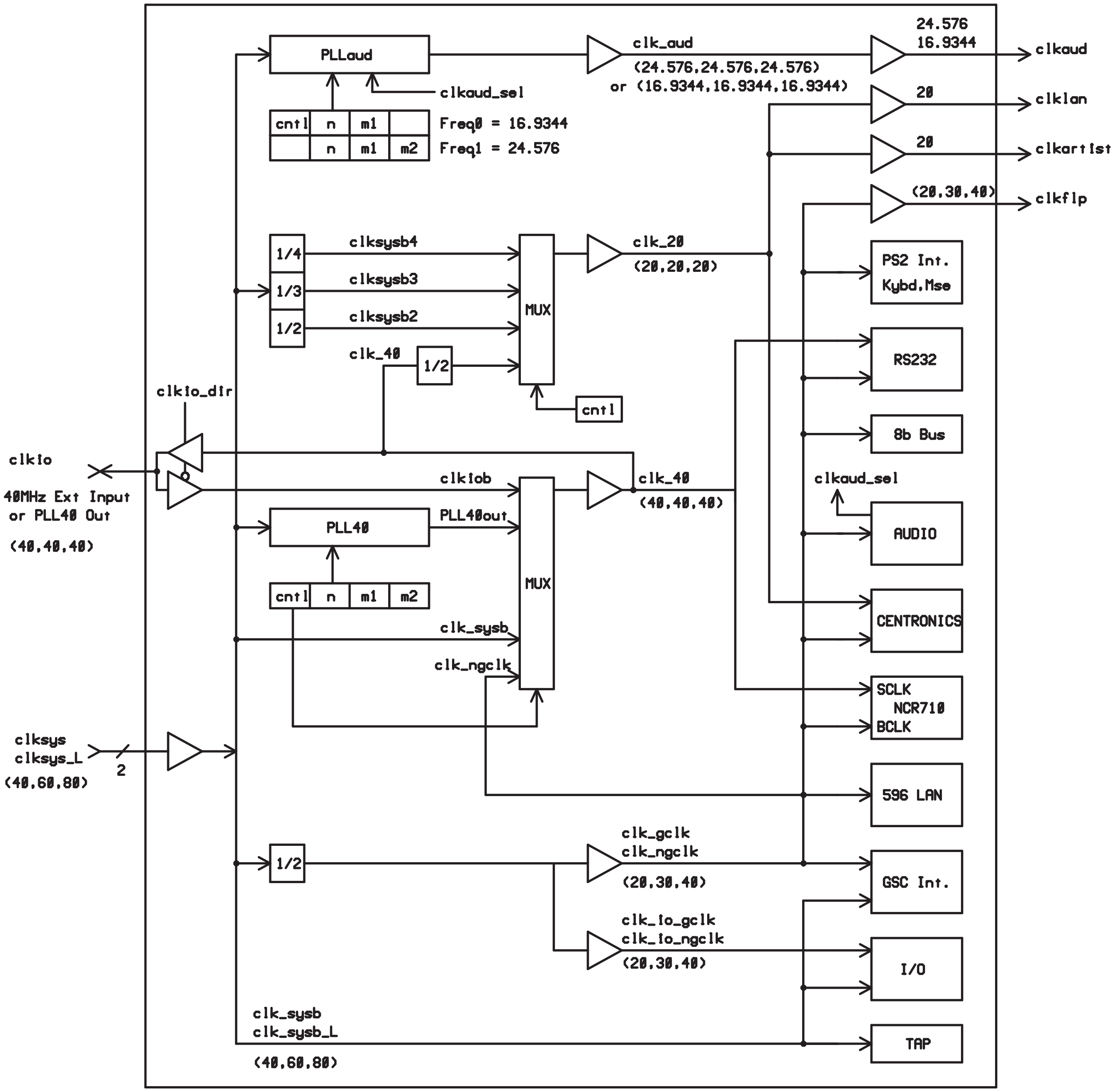
17 CLOCKS

Introduction

The LASI Clock Block generates clocks for: the GSC Interface, all the built–in I/O modules, and the reference clocks for the external LAN Transceiver, Artist graphics interface and CD Audio Interface. LASI uses Phase Lock Loops (PLLs) to generate or synthesize the required clock frequencies from a single input reference (the GECKO system clock reference, clksys). This on–chip clock generation reduces system cost by reducing the number of crystals or crystal oscillators in the system.

A block diagram of the LASI clock system is shown in **Figure 7**.

Figure 7. LASI Clock Diagram



Note: (##,##,##) indicate frequency set for clksys = 40,60,80 MHz respectively.

Overview

LASI generates 6 internal and 5 external reference clocks, as shown in **Table 24**.

Table 24. LASI Internal and External Clocks

Clock Name	Target Frequency	Use
clk_sysb, clk_sysb_L	clksys	Internal Processor clock.
clk_gclk, clk_ngclk	clksys ÷ 2	Internal GSC clock.
clk_40	40 MHz via one of the following: clk_40 PLL clksys clksys ÷ 2 clkio pad	Internal NCR 710 SCLK and RS–232.
clk_20	20 MHz via one of the following: clk_40 ÷ 2 clksys ÷ 2 clksys ÷ 3 clksys ÷ 4	Internal Centronics, External clklan, clkartist.
clklan	20 MHz	External LAN Transceiver Chip.
clkartist	20 MHz	External Artist Graphics Chip.
clkflp	clksys ÷ 2	Not used in Gecko.
clkaud	Freq0 = 16.9344 MHz or Freq1 = 24.576 MHz selectable–on–the–fly, PLL generated	External Audio Chip.
clkio	40 MHz	Optional output from clk_40.

The inputs are : clksys 712 system clock in the 40–80 MHz range
clkio Optional 40 MHz fixed clock when clk_40 PLL is disabled.

clkio is an optional 40 MHz input reference in the event that the clk_40 PLL jitter is greater than can be tolerated, but at a higher processor board cost. When the clk_40 PLL is disabled, clkio is the reference input for clk_40. When the clk_40 PLL is enabled, clkio is an output and drives the clk_40 PLL output off–chip.

Registers

Register addresses are offsets from the LASI Clock Configuration Registers base address.

Address (HEX)	Bits: 31–24	Bits: 23–16	Bits: 15–8	Bits: 7–0
000	clk_40_cntl	clk_40_n	clk_40_m1	clk_40_m2
004	clk_20_cntl			
008	clk_aud_cntl	clk_aud0_n	clk_aud0_m1	
00C		clk_aud1_n	clk_aud1_m1	clk_aud1_m2

Each of the above registers are Read/Write and word–accessible only.

The PLL coefficient registers for clk_40 and clkaud are defined as:

n = PLL Multiplier – 1
m1 = PLL Pre–Divisor – 1
m2 = PLL Post–Divisor – 1
div = PLL Internal Divisor, used at power up

such that $\text{Freq. Out} = \text{clksys} * \dfrac{(n+1)}{((m1+1)*(m2+1))}$

17.1 Register: clk_40

Bit(s)	Name	Definition	Pwr On Default
clk_40[31]	PLL40 reset_L	0 = PLL reset 1 = PLL run	1’b1
clk_40[30:28]	PLL40 div	PLL40 Internal Divisor	3’d3
clk_40[27:26]	Select clk_40 source:	00 = PLL40 output 01 = clksys 10 = clksys ÷ 2 11 = clkio.	2’b11
clk_40[25]	PLL40 enable:	0 = PLL bypassed and PLL40=clksys/(m2+1) 1 = PLL enabled.	1’b0
clk_40[24]	PLL40 sync enable:	0 = PLL free run 1 = PLL synchronous operation.	1’b0
clk_40[23:16]	PLL40 n	PLL40 Multplier – 1	8’d3
clk_40[15:8]	PLL40 m1	PLL40 Pre–Divisor – 1	8’d7
clk_40[3:0]	PLL40 m2	PLL40 Post–Divisor – 1	4’d0

17.1.1 Register: clk_20

Bit(s)	Name	Definition	Pwr On Default
clk_20[25:24]	Select clk_20 source:	00 = clk_40 ÷ 2 01 = clksys ÷ 2 10 = clksys ÷ 3 11 = clksys ÷ 4.	2'b00

17.1.2 Register: clk_aud0 = 16.9344 MHz Target Frequency

Bit(s)	Name	Definition	Pwr On Default
clk_aud0[31]	PLLaud reset_L	0 = PLL reset 1 = PLL run	1'b1
clk_aud0[30:28]	PLLaud div	PLLaud Internal Divisor	3'd3
clk_aud0[25]	PLLaud enable:	0 = PLL bypassed and PLLaud=clksys/(m2+1) 1 = PLL enabled.	1'b0
clk_aud0[24]	PLLaud sync enable	0 = PLL free run 1 = PLL synchronous operation.	1'b0
clk_aud0[23:16]	PLLaud0 n	PLLaud0 Multiplier – 1	8'd46
clk_aud0[15:8]	PLLaud0 m1	PLLaud0 Pre–Divisor – 1	8'd36

The clk_aud0[31:24] bits are the control bits for PLLaud.

The clk_aud1[3:0] bits are the m2 PLLaud Post–Divisor for PLLaud.

There are two sets of audio n and m1 PLL coefficients (PLLaud0 n/m1 and PLLaud1 n/m1) which are selected by the clkaud_sel signal from the audio interface:

clkaud_sel = 0: Use PLLaud0 n/m1
clkaud_sel = 1: Use PLLaud1 n/m1.

17.1.3 Register: clk_aud1 = 24.576 MHz Target Frequency

Bit(s)	Name	Definition	Pwr On Default
clk_aud1[23:16]	PLLaud1 n	PLLaud1 Multiplier – 1	8'd46
clk_aud1[15:8]	PLLaud1 m1	PLLaud1 Pre–Divisor – 1	8'd50
clk_aud1[3:0]	PLLaud1 m2	PLLaud1 Post–Divisor – 1	4'd5

PLL Coefficients

Table 25 below shows the PLL Coefficients required for clksys frequencies being considered and the the frequency error for cases where the exact frequency can not be generated.

Table 25. LASI PLL Coefficients

		Clock Frequencies				
		40	48	60	64	75
Target Freq.						
clk_40=40	n	7	9	5	4	7
	m1	7	11	8	7	14
	m2	0	0	0	0	0
	div	3	3	3	3	3
	%error	0	0	0	0	0
aud0=16.9344	n	46	11	34	49	6
	m1	110	16	123	188	30
	div	2	3	3	2	3
	%error	0.0150	0.0400	.0064	–0.0188	.0064
aud1=24.576	n	50	42	33	47	38
	m1	82	41	82	124	118
	m2	0	1	0	0	0
	div	3	2	3	3	3
	%error	0.0094	–0.0186	0.0094	0	0.0156

18 APPENDIX A: 712 I/O MEMORY MAP

The Lasi chip always responds to a contiguous 2M byte address space. This address space is relocatable at power–up to one of four locations. This relocation allows multiple Lasi chips to be used in the same system. Table 26 shows the address offset used for each I/O device in the 712. Lasi will respond to accesses in the unassigned areas of the address space, but the results of such accesses are undefined.

Address Offset		Block Size (bytes)	I/O Device
starting	ending		
00 0000	**0F FFFF**	1M	**External 8–bit Bus**
	00 0000		Lasi 2 Primary I/O Config Reg
	80 0000		Lasi 2 Secondary I/O Config Reg
10 0000	**10 0FFF**	4K	**Interrupt Registers**
	10 0000		Request Register
	10 0004		Mask Register
	10 0008		Pending Register
	10 000C		Control Register
	10 0010		Interrupt Address Register
10 1000	**10 1FFF**	4K	**Parallel Port DMA Registers**
	10 1000		Current Address Register
	10 1001		Current Count Register
	10 1008		Status Register
	10 100A		Write Single Mask Bit
	10 100B		Mode Register
	10 100C		Clear Byte Pointer
	10 100D		Master Clear
	10 100E		Clear Mask Register
	10 100F		Mask Register
	10 1010		Fifo Limit Register (not used)

Address Offset		Block Size (bytes)	I/O Device
starting	ending		
	10 1087		Current Address Low Page Register
	10 1401		High Current Count Register
	10 140A		Interrupt Pending Register
	10 1487		Current Address High Page Register
10 2000	**10 2FFF**	4K	**Parallel Interface Registers**
	10 2000		Parallel Slave Reset
	10 2800		ParData Register
	10 2801		ParStatus Register
	10 2802		ParDevCtl Register
	10 2803		Undefined
	10 2804		ModeCtl Register
	10 2805		IECtlStat Register
	10 2806		TDC0 Register
	10 2807		TDC1 Register
10 3000	**10 3FFF**	4K	**Parallel Port DMA Reset Registers**
10 4000	**10 4FFF**	4K	**Audio**
	10 4000		Audio ID Register
	10 4004		Audio Reset
	10 4008		Control Register
	10 400C		Gain Control Register
	10 4010		Playback Next Address Register
	10 4014		Playback Current Address Register
	10 4018		Recording Next Address Register
	10 401C		Recording Current Address Register
	10 4020		DMA Status Register
	10 4024		Over Range Register
	10 4028		PIO Register
	10 403C		DIAG Register
	10 4040		Receiver Buffer Register
	10 4040		Transmitter Holding Register

Address Offset		Block Size (bytes)	I/O Device
starting	ending		
	10 4040		Divisor Latch Reg LSB (DLAB=1)
	10 4041		Interrupt Enable Register
	10 4041		Divsior Latch Reg MSB (DLAB=1)
	10 4042		Interrupt Ident Register
	10 4042		Fifo Control Register
	10 4043		Line Control Register
	10 4044		Modem Control Register
	10 4045		Line Status Register
	10 4046		Modem Status Register
10 4047	10 405F		Undefined
	10 4060		Receiver Buffer Register
	10 4060		Transmitter Holding Register
	10 4060		Divisor Latch Reg LSB (DLAB=1)
	10 4061		Interrupt Enable Register
	10 4061		Divsior Latch Reg MSB (DLAB=1)
	10 4062		Interrupt Ident Register
	10 4062		Fifo Control Register
	10 4063		Line Control Register
	10 4064		Modem Control Register
	10 4065		Line Status Register
	10 4066		Modem Status Register
10 4067	10 4FFF		Undefined
10 5000	**10 5FFF**	4K	**RS232**
	10 5000		RS232 Reset
10 5001	10 57FF		Undefined
	10 5800		Receiver Buffer Register
	10 5800		Transmitter Holding Register
	10 5800		Divisor Latch Reg LSB (DLAB=1)
	10 5801		Interrupt Enable Register
	10 5801		Divsior Latch Reg MSB (DLAB=1)

Address Offset		Block Size (bytes)	I/O Device
starting	ending		
	10 5802		Interrupt Ident Register
	10 5802		Fifo Control Register
	10 5803		Line Control Register
	10 5804		Modem Control Register
	10 5805		Line Status Register
	10 5806		Modem Status Register
	10 5807		Scratch Register
10 5808	10 5FFF		Undefined
10 6000	**10 6FFF**	4K	**SCSI**
	10 6000		SCSI Reset
10 6004	10 60FF		Reserved
	10 6100		SIEN, SDID, SCNTL1,0
	10 6104		SOCL, SODL, SXFER, SCID
	10 6108		SBCL, SBDL, SIDL, SFBR
	10 610C		SSTAT2, SSTAT1, SSTAT0, DSTAT
	10 6110		DSA
	10 6114		CTEST3, 2, 1, 0
	10 6118		CTEST7, 6, 5, 4
	10 611C		TEMP
	10 6120		LCRC, CTEST8, ISTAT, DFIFO
	10 6124		DCMD, DBC
	10 6128		DNAD
	10 612C		DSP
	10 6130		DSPS
	10 6134		SCRATCH
	10 6138		DCNTL, DWT, DIEN, DMODE
10 613C	10 61FE		ADDER
	10 61FF		Reserved
10 7000	**10 7FFF**	4K	**LAN**
	10 7000		LAN Reset

Address Offset		Block Size (bytes)	I/O Device
starting	ending		
	10 7004		CPU PORT_L Access
	10 7008		Channel Attention (no data)
10 8000	**10 8FFF**	4K	**PS/2 Keyboard/Mouse Controller**
	10 8000		Reset
	10 8000		Keyboard ID
	10 8004		Data
	10 8008		Control
10 9000	**10 9FFF**	4K	**Real Time Clock**
	10 9000		Data
10 A000	**10 AFFF**	4K	**Floppy Disk Controller**
	10 A000		FDC Reset
	10 A004		FDC Data Register
	10 A008		FDC Master Status Register
	10 A014		FIFO Data Register
	10 A018		FIFO Control Register
	10 A01C		FIFO Status Register
	10 A020		FDC Control Register
	10 A040		FDC Operations Register
10 B000	**10 BFFF**	4K	**Clock Configuration Registers**
	10 B000		40MHz Clock Control Register
	10 B004		20MHz Clock Control Register
	10 B008		Audio Clock Control Register 0
	10 B00C		Audio Clock Control Register 1
10 C000	**10 CFFF**	4K	**Misc. Lasi Registers (reset, version, error)**
	10 C000		Power Control Register
	10 C004		Error Logging Register
	10 C008		Version Control Register
	10 C00C		I/O Reset Register
	10 C010		Arbitration Mask Register
10 D000	**10 FFFF**	972K	**Unassigned**

Table 26. Address offsets for712 I/O

The SPACE[1:0] bits are used by Lasi to determine the base address for the 2M byte address block. The value for the SPACE bits is set using resistors to pull bits[1:0] of the extrenal 8–bit bus either high or low during reset. These bits are then latched into Lasi on the rising edge of the RESETL signal as the SPACE[1:0] bits.

Table 27 shows the Lasi base address as a function of the SPACE bits.

SPACE[1:0]	Base Address
00	F000 0000
01	F040 0000
10	FF80 0000
11	FFC0 0000

Table 27. Lasi Base Addresses

The primary Lasi in a 712 system will always have SPACE[1:0]=00. If there is a second Lasi chip in 712, the SPACE[1:0] bits will be set to 01. The 10 and 11 values of SPACE[1:0] will not be used in the 712 product, however they are used in some servers, and in later B and C class workstations.

19 APPENDIX B: 712 SYSTEM MEMORY MAP

Address	Description	Notes
0000 0000 EFFF FFFF	Memory Space	
F000 0000 F00F FFFF	LASI ROM Space	
F010 0000 F01F FFFF	LASI I/O Space	
F020 0000 F020 7FFF	WAX I/O Space	Also used by Token RIng card in the 712.
F040 0000 F05F FFFF	Second LASI	
F100 0000 F3FF FFFF	Reserved for Future I/O	PDC will look here for IODC.
F400 0000 F5FF FFFF	Second ARTIST	
F800 0000 F9FF FFFF	ARTIST	
FC00 0000 FFBF FFFF	WAX EISA Space	Also used by token ring card in the 712.
FFFB 0000 FFFB AFFF	Unused Central Bus Modules	
FFFB B000 FFFB BFFF	Reserved for 712 PDC	Accesses to this page must always cause bus errors.
FFFB C000 FFFB DFFF	Reserved for CPU	This is where Venom used to be.
FFFB E000 FFFB EFFF	CPU	
FFFB F000 FFFB FFFF	MIOC	(PA7100LC Memory Control)
FFFC 0000 FFFF FFFF	Local/Global Broadcast	(READYL driven by PA7100LC)