Intel® Low Pin Count (LPC)

Interface Specification

August 2002

Revision 1.1

Document Number: 251289-001

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an ordering number and are referenced in this document, or other Intel literature, may be obtained from:

Intel Corporation

www.intel.com

or call 1-800-548-4725

Contents

Figures

Tables

Revision History

Rev. No.	Description	Rev. Date
-	• Revision 1.0 - Initial release. No Document Number Assigned	
- 001	• Revision 1.1 - Assigned Document Number 251289 - 001 • Added Sections 4.3 and 5.3 describing Firmware Memory Cycles including support for multi-byte read and write accesses. • Added electrical characteristics for LPCPD# and LSMI# signals in Section 10.2. Clarified electrical requirements for LRESET#, LCLK, and CLKRUN# in Section 10.3 • Added recommended pull-up resistor requirements in Section 10.4. • Added additional description to system reset requirements in Section 9. • Made clarifications and corrections.	August 2002

1 Introduction

This document contains a specification for a new low pin count bus interface, called LPC. The target audiences for this document are system and component designers.

1.1 Goals of the LPC Interface (I/F)

- Enable a system without an ISA or X-bus.
- Reduce the cost of traditional X-bus devices.
- Intended for use by devices down on a motherboard only (i.e. no connector).
- Meet the data transfer rate of X-bus, and exceed those data rates where appropriate.
- Perform the same cycle types as the X-bus: Memory, I/O, DMA, and Bus Master
- Support new Firmware Memory cycle type allowing separate boot BIOS firmware memory cycles and application memory cycles.
- Increase the memory space from 16MB on the X-bus to 4GB to allow BIOS sizes much greater than 1MB, and other memory devices outside of the traditional 16MB range.
- Synchronous design. Much of the challenge of an X-bus design is meeting the different, and in some cases conflicting, ISA timings. Make the timings synchronous to a reference well known to component designers, such as PCI.
- Software transparency: do not require special drivers or configuration for this interface. The motherboard BIOS should be able to configure all devices at boot.
- Support desktop and mobile implementations.
- Ability to support a variable number of wait-states.
- Ability to have I/O and memory cycles retried in SMM handler.
- Ability to support wake-up and other power state transitions.

1.2 Assumptions and Functionality Requirements

- Only the following class of devices may be connected to the LPC interface:
  - Super I/O (FDC, SP, PP, IR, KBC) => I/O slave, DMA, Bus Master (for IR, PP)
  - Audio, including AC'97 style design => I/O slave, DMA, Bus Master
  - Generic Application Memory, including BIOS => Memory Slave
  - BIOS Firmware Memory => Firmware Memory Slave
  - Embedded Controller => I/O slave, Bus Master
- Interrupts are communicated with the serial interrupt (SERIRQ) protocol.
- The LPC interface does not need to support high-speed buses (such as Cardbus, 1394, etc) downstream, nor does it need to support low-latency buses such as USB.

1.3 Terminology

Term	Description
Host	The portion of the interface that is connected either directly to the CPU or to upstream devices connected to the CPU. This is typically a system chip-set.
Peripheral	Devices downstream on LPC that were previously connected to the X-bus, such as Super I/O components, flash, and other embedded controllers.
'XXXXb'	Indicates the value of a signal in binary notation.

2 Signal Definition

Table 1 and Table 2 lists the 7 required and 6 optional signals used for the LPC interface. Many of the signals are the same as signals found on the PCI interface, and do not require any new pins on the host. Both hosts and peripherals must implement required signals. Optional signals may or may not be present on particular hosts or peripherals.

Table 1: LPC Required Signal List

Signal	Peripheral	Host	Description
LAD[3:0]	I/O	I/O	**Multiplexed Command, Address, and Data:** See Section 4.2.1 for details on the usage of these signals.
LFRAME#	I	O	**Frame:** Indicates start of a new cycle, termination of broken cycle.
LRESET#	I	I	**Reset:** Same as PCI Reset on the host. The host does not need this signal if it already has PCIRST# on its interface.
LCLK	I	I	**Clock:** Same 33MHz clock as PCI clock on the host. Same clock phase with typical PCI skew. The host does not need this signal if it already has PCICLK on its interface.

Table 2: LPC Optional Signal List

Signal	Peripheral	Host	Description
LDRQ#	O	I	**Encoded DMA/Bus Master Request:** Only needed by peripherals that need DMA or bus mastering. Requires an individual signal per peripheral. Peripherals may not share an LDRQ# signal.
SERIRQ	I/O	I/O	**Serialized IRQ:** Only needed by peripherals that need interrupt support. This signal is required for the host if it does not contain the ISA IRQ lines as inputs.
CLKRUN#	OD	I/OD	**Clock Run:** Same as PCI CLKRUN#. Only needed by peripherals that need DMA or bus mastering in a system that can stop the PCI bus (generally in mobile systems). This signal is optional for the host.
LPME#	OD	I/OD	**LPC Power Management Event:** Similar to PCI PME#. Used by peripherals to request wake-up from a low-power state.

Signal	Peripheral	Host	Description
LPCPD#	I	O	**Power Down:** Indicates that the peripheral should prepare for power to be removed from the LPC I/F devices. Actual power removal is system dependent. This signal is optional for the host.
LSMI#	OD	I	**SMI#:** Only needed if peripheral want to cause SMI# on an I/O instruction for retry. Otherwise can use SMI# via SERIRQ. This signal is optional for the host.

In comparison with existing ISA-based devices, the signal savings is substantial. The LPC interface will generally only require 6 new signals: LAD[0:3], LFRAME#, and LDRQ#. CLKRUN# is typically only implemented in mobile systems. LPCPD# is only needed for LPC devices that are partially powered during certain low-power states.

At a minimum, the following ISA/X-bus signals found on plug-n-play devices are no longer needed:

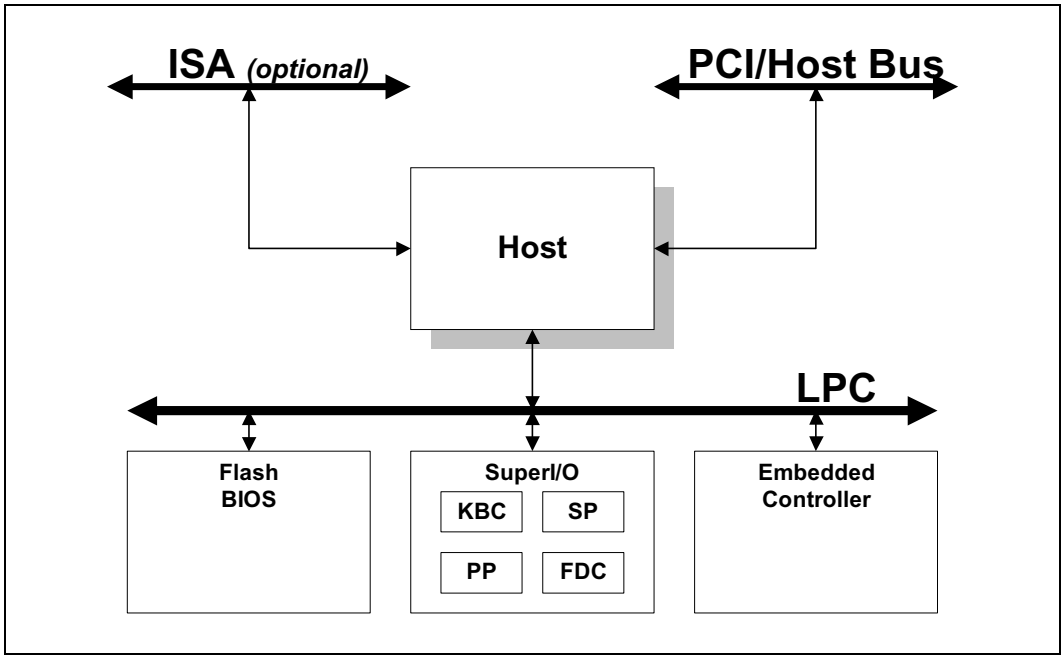
D[7:0], SA[15:0], DREQ[3:0], DACK#[3:0], TC, IOR#, IOW#, IOCHRDY	36 signals
Net Savings on Peripheral:	30

On 16 bit ISA based devices, the IOCS16# and MEMCS# signals are also saved. If the serial IRQ was not previously supported, and if 16 bit DMA channels are used by the peripheral, several more pins are saved. The net effect is that a fully functional Super I/O device that may have required a 160 pin package to implement can now fit in an 88 pin or less size package by using the LPC interface.

3 Block Diagram

Figure 1 contains a typical setup. In this setup, the LPC interface is connected through a host device to a PCI or Host Bus, such as that of Intel CPUs.

Figure 1: Typical Setup



This page is intentionally left blank.

4 Protocol Overview

4.1 Cycle Types

Table 3 shows the various types of cycles that are supported by the protocol.

Table 3: Cycle Types

Cycle Type	Sizes Supported	Comment
Memory Read	1 byte	Optional for both LPC hosts and peripherals.
Memory Write	1 byte	Optional for both LPC hosts and peripherals.
I/O Read	1 byte	Optional for peripherals.
I/O Write	1 byte	Optional for peripherals.
DMA Read	1, 2, 4 bytes	Optional for peripherals.
DMA Write	1, 2, 4 bytes	Optional for peripherals.
Bus Master Memory Read	1, 2, 4 bytes	Optional for both LPC hosts and peripherals, but strongly recommended for hosts.
Bus Master Memory Write	1, 2, 4 bytes	Optional for both LPC hosts and peripherals, but strongly recommended for hosts.
Bus Master I/O Read	1, 2, 4 bytes	Optional for both LPC hosts and peripherals.
Bus Master I/O Write	1, 2, 4 bytes	Optional for both LPC hosts and peripherals.
Firmware Memory Read	1, 2, 4, 128 bytes	Optional for both LPC hosts and peripherals.
Firmware Memory Write	1, 2, 4 bytes	Optional for both LPC hosts and peripherals.

The following rules are to be followed:

- Hosts and peripherals shall implement the cycle types as described in Table 3.
- Peripherals must not attempt Bus Master cycles that are not supported by the host. For example, if the host does not support Bus Master I/O cycles, the peripheral must not attempt those cycles.
- Peripherals must ignore cycles that they do not support.

4.2 Memory, I/O, and DMA Cycle Overview

This section describes the cycles for Memory, I/O, DMA and Bus Master cycles. Firmware Memory cycles are described in Section 4.3.

Data transfers on the LPC bus are serialized over a 4 bit bus. The general characteristics of this bus are:

- One control line, called LFRAME#, which is used by the host to start or stop transfers. No peripherals drive this signal.
- The LAD[3:0] bus, which communicates information serially. The information conveyed is cycle type, cycle direction, chip selection, address, data, and wait states.
- Side-band signals, optionally implemented, convey interrupts and power management features. These signals are the same as or similar to many signals found on current motherboard implementations.

The general flow of cycles is as follows:

1. A cycle is started by the host when it drives LFRAME# active and puts appropriate information on the LAD[3:0] signal lines.

2. The host drives information relative to the cycle, such as address, or DMA channel number, or bus master grant. For DMA and target cycles, the host also drives cycle type (memory or I/O), read/write direction, and size of the transfer.

3. The host optionally drives data, and turns the bus around to monitor the peripheral for completion of the cycle.

4. The peripheral indicates completion of the cycle by driving appropriate values on the LAD[3:0] signal lines, and potentially drives data.

5. The peripheral turns the bus around to the host, ending the cycle.

For bus master cycles, there are small changes to this protocol, as the bus master must drive control and address information to the host, and the host is responsible for ending the cycle, but in general, the flow is the same. Below is a more detailed look at the different signals in the transfer, and their operation at different points in the cycle.

4.2.1 LAD[3:0]

The LAD[3:0] signal lines communicate address, control, and data information over the LPC bus between a host and a peripheral. The information communicated is: start, stop (abort a cycle), transfer type (memory, I/O, DMA), transfer direction (read/write), address, data, wait states, DMA channel, and bus master grant.

Not all cycle types use the LAD bus in the same fashion. For example, DMA does not use addresses; instead it uses channel numbers. The following sections go into more detail about which fields are used, and in what order they are communicated.

4.2.1.1 START

This field indicates the start or stop of a transaction. All state peripheral machines will go to a state which monitors LAD[3:0] for this field when LFRAME# is asserted. The START field is valid on the last clock that LFRAME# is asserted. While LFRAME# is asserted, this field may take on many values, so peripherals should not make any assumptions on the validity of this field until LFRAME# is de-asserted.

This field is used to indicate a device number for bus masters, or "start/stop" indication for non-bus master cycles. The defined encodings are:

Bits[3:0]	Definition
0000	Start of cycle for a target. Used for Memory, I/O, and DMA cycles.
0001	Reserved
0010	Grant for bus master 0.
0011	Grant for bus master 1.
0100 - 1100	Reserved
1101	Start of Cycle for Firmware Memory Read cycle. See Section 4.3 for additional details.
1110	Start of Cycle for Firmware Memory Write cycle. See Section 4.3 for additional details.
1111	Stop/Abort: End of a cycle for a target. See Section 4.2.1.13 for a description of aborted cycles.

All encodings marked as "Reserved" are reserved for future use. Peripherals should make no assumption on the data being transmitted by the host if it sees a reserved field. Therefore, if the peripheral sees an encoding of this type, it must ignore the cycle and not monitor the bus until the next time that LFRAME# goes active.

4.2.1.2 Cycle Type / Direction (CYCTYPE + DIR)

This field is driven by the host, and is used to communicate cycle type (memory, I/O, DMA) and direction (read/write) for cycles. This field is driven by the host when it is performing DMA or target accesses, and by the peripheral on bus master accesses. Bit 0 of this field is reserved and must be ignored by the peripheral and driven to 0 by the host for host based accesses. For bus master accesses, it must be ignored by the host and driven to 0 by the peripheral. Valid values are:

Bits[3:2]	Bit[1]	Definition
00	0	I/O Read
00	1	I/O Write
01	0	Memory Read
01	1	Memory Write
10	0	DMA Read
10	1	DMA Write
11	x	**Reserved**: Neither the peripheral nor the host is allowed to drive this signal type. If this value is observed by the peripheral, the cycle must be ignored. If this signal is driven by the peripheral on bus master accesses, the host will terminate the transfer by driving LFRAME# active.

4.2.1.3 SIZE

This field is one clock. It is driven by the host on DMA transfers, and driven by the peripheral on bus master memory transfers, to determine how many bytes are to be transferred. Bits[3:2] are reserved, and must be driven to '00b' by the driver, and must be ignored by the target. The remaining bits are encoded as follows:

Bits[1:0]	Size
00	8 bits (1 byte)
01	16 bits (2 bytes)
10	**Reserved**: Must not be driven by the host or peripheral. If this value is observed by the peripheral, the cycle must be ignored. If this value is observed by the host on bus master transfers, it may abort the transfer.
11	32 bits (4 bytes)

4.2.1.4 Turn-Around (TAR)

This field is two clocks wide, and is driven by the host when it is turning control over to the peripheral, (for example, to read data), and is driven by the peripheral when it is turning control back over to the host.

On the first clock of this two clock wide field, the host or peripheral drives the LAD[3:0] lines to '1111b'. On the second clock of this field, the host or peripheral tri-states the LAD[3:0] lines. Since these lines have weak pull-ups on them, they will remain at a logical high state.

4.2.1.5 ADDR

This field is either 4 clocks wide for I/O cycles, or 8 clocks wide for memory cycles. It is driven by the host on target accesses, and driven by the peripheral on bus master accesses. This field is not driven on DMA cycles. When this field is driven, it is driven out with the most significant nibble first. For example, on memory transfers, the first clock of this field contains Address[31:28], and the last clock of this field contains Address[3:0].

4.2.1.6 CHANNEL

This is a one clock wide field that is driven by the host on DMA cycles to indicate to the peripherals which DMA channel has been granted. Bits[2:0] contain the DMA channel number, and Bit[3] contains the encoded ISA based TC (terminal count) line. This field is not driven on target or bus master transfers.

4.2.1.7 DATA

This field is 2 clocks wide, representing one data byte. The host drives it on target, DMA, and bus master cycles when data is flowing to the peripheral, and by the peripheral when data is flowing to the host. When data is driven, it is driven with the least significant nibble first. For example, on the first clock, Data[3:0] is driven, and on the second clock, Data[7:4] is driven.

4.2.1.8 SYNC

This field is used to add wait states. It can be several clocks in length. On target or DMA cycles, this field is driven by the peripheral. For bus master cycles, the host drives this field. Valid values for this field are:

Bits[3:0]	Indication
0000	**Ready:** SYNC achieved with no error. For DMA transfers, this also indicates DMA request de-assertion and no more transfers desired for that channel.
0001 - 0100	Reserved
0101	**Short Wait:** Peripheral indicating normal wait states. See below for additional information.
0110	**Long Wait:** Peripheral indicating abnormally long wait states. See below for additional information.
0111 - 1000	Reserved
1001	**Ready More (DMA Only):** SYNC achieved with no error and more DMA transfers desired to continue after this transfer. This value is valid only on DMA transfers and is not allowed to be returned for target accesses by the peripheral or bus master accesses by the host.
1010	**Error:** Sync achieved with error. This is generally used to replace the SERR# or IOCHK# signal on the PCI/ISA bus. It indicates that the data is to be transferred, but there is a serious error in this transfer. For DMA transfers, this also indicates DMA request de-assertion and no more transfers desired for that channel.
1011 - 1111	Reserved

If a peripheral needs to assert wait states, it does so by driving '0101b' (short SYNC) or '0110b' (long SYNC) on LAD[3:0] until it is ready. When ready, it may choose to drive '0000b' (Ready), '1010b' (Error), or, in the case of DMA transfers, '1001b' (Ready More).

On any particular cycle, if the host or peripheral chooses to insert wait states, it must choose one type of SYNC wait value ('0101b' or '0110b'), and must not change it until it asserts one of the ready SYNC values.

The short SYNC is used for normal wait-states. This is where the cycle will complete within a few clocks.

The long SYNC is used where the number of wait-states is large. This would typically be used for Enhanced Parallel Port (EPP) cycles, where the number of wait-states could be quite large (>1 microsecond). By distinguishing this from short SYNC, the host may issue separate time-out values to abort the cycle. See Section 4.2.1.9 below on time-out values.

4.2.1.9 SYNC Time-out

There are several potential error cases that can occur on the LPC I/F.

1. The host starts a cycle (Memory, I/O, DMA), but no device ever drives a defined (non-reserved) SYNC cycle. If the host observes 3 consecutive clocks without a defined SYNC, it can conclude that no peripheral will respond and can abort the cycle.

2. The host drives a cycle (Memory, I/O, DMA), a device drives a valid SYNC to insert wait states (LAD[3:0] = '0101b' or '0110b'), but never completes the cycle. This could occur if the peripheral locks up for some reason. Peripherals must be designed to prevent this case. However, since it is impossible to completely prevent this, the host will take the following policy:

- If the Sync pattern is '0101b', then the maximum number of SYNC clocks is assumed to be 8. If the host sees that more than 8 clocks of this SYNC value have been driven, it may abort the cycle.
- If the Sync pattern is '0110b', then no maximum number of SYNC clocks is assumed. The peripheral must have protection mechanisms to complete the cycle.

When the host is driving SYNC, it may have to insert a very large number of wait-states, depending on host latency variance. The peripherals should not assume any particular time-out.

4.2.1.10 SYNC Error Indication

The SYNC protocol allows the peripheral to report an error via the LAD[3:0] = '1010b' encoding. The intent of this encoding is to give peripherals a method of communicating errors to aid higher layers with more robust error recovery.

If the host was reading data from a peripheral, data will still be transferred in the next two nibbles. This data may be invalid, but it must be transferred by the peripheral. If the host was writing data to the peripheral, the data had already been transferred.

In the case of multiple byte DMA cycles, an error SYNC terminates the cycle. Therefore, if the host is transferring 4 bytes from a device, and if the device returns the error SYNC in the first byte, the other three bytes will not be transferred.

After receiving an error SYNC, the host has several options. If it has an ISA bus, it may choose to assert IOCHK#. If it has a PCI bus, it may choose to assert SERR#. Other options are to route the error condition to SCI or SMI#. A final option is to take no action to alert higher levels.

4.2.1.11 LFRAME#

LFRAME# is used by the host to indicate the start of cycles and the termination of cycles due to an abort or time-out condition. This signal is to be used by peripherals to determine when to monitor the bus for a cycle. The following sections describe when the host will drive this signal active and what action is to be taken by peripherals when this signal is seen as active.

This signal is used as a general notification that the LAD[3:0] lines contain information relative to the start or stop of a cycle, and that peripherals must monitor the bus to determine whether the cycle is intended for them. The benefit to peripherals of LFRAME# is it allows them to enter lower power states internally. There is no need for peripherals to monitor the bus if they are not being addressed, so the peripherals not being addressed can de-couple their state machines from the bus and internally gate their clocks.

When peripherals sample LFRAME# active, they are to immediately stop driving the LAD[3:0] signal lines on the next clock and monitor the bus for new cycle information.

4.2.1.12 Start of Cycle

The various cycles all start in a common way:

1. The host asserts LFRAME# for one or more clocks and drives a START value on LAD[3:0]. Upon observing LFRAME# active, all peripherals stop driving the LAD[3:0] signals, even if in the middle of a transfer.

At the beginning of a cycle, the host is permitted to keep LFRAME# active for more than one consecutive clock and even change the START value. The peripheral must always use the last START value when LFRAME# was active. For example, if LFRAME# is active for two consecutive clocks, the peripheral should ignore the value during the first clock and only use the second.

2. Peripherals monitor LAD[3:0] when LFRAME# is active for the appropriate START value. If a peripheral recognizes the START value, it should attempt to decode the rest of the cycle. If a peripheral does not recognize a particular START value, it may ignore the rest of the cycle until LFRAME# goes active again.

3. After the final START value is driven, and when the host is ready to begin the cycle, the host de-asserts LFRAME#. The peripheral must use the START value driven when LFRAME# is de-asserted.

Figure 2 shows the typical timing for LFRAME#. Figure 3 shows the timing where LFRAME# is active for more than one consecutive clock.

Figure 2: Typical Timing for LFRAME#

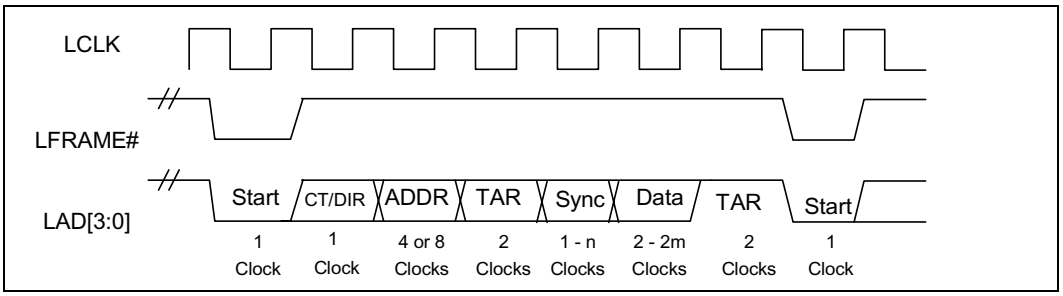
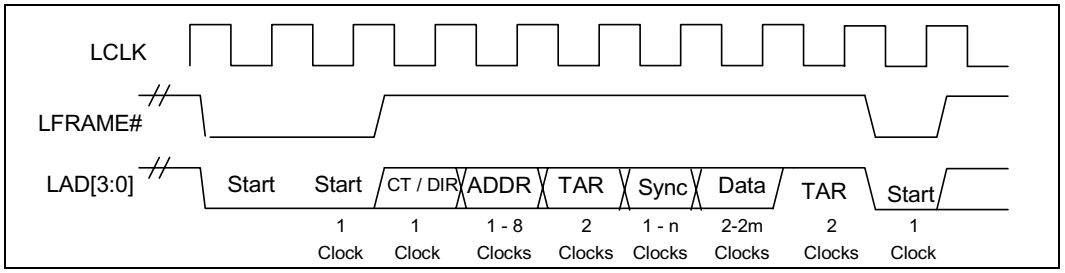


Figure 3: Extended Timing for LFRAME#



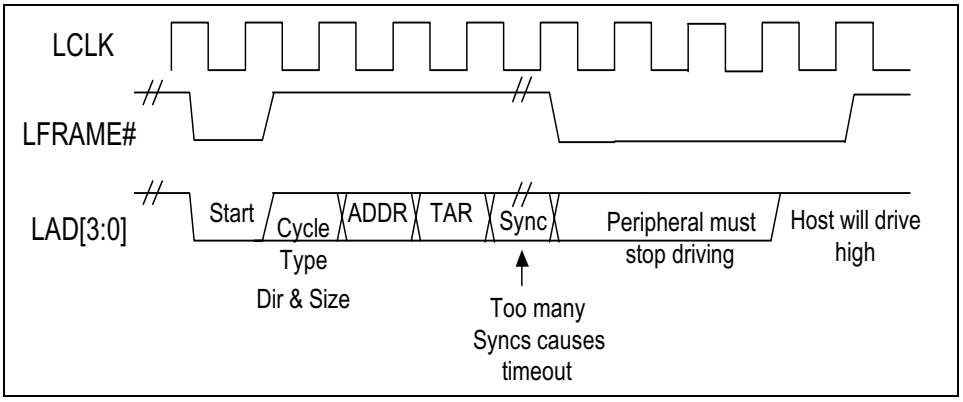
4.2.1.13 Abort Mechanism

Figure 4 shows the mechanism where LFRAME# is used to "kick" a peripheral off the LPC I/F. The host can cause an abort on the LPC I/F by driving LFRAME# active with a START value of '1111b'. Since LFRAME# is normally only active at the start of a cycle, if a device is in the process of running a cycle when LFRAME# goes active, it must assume the host is going to abort the cycle and stop driving the LAD[3:0] signals. To ensure that the abort will be seen, the host must keep LFRAME# active for at least four consecutive clocks and drive LAD[3:0] to '1111b' no later than the 4th clock after LFRAME# goes active. The host must drive LFRAME# inactive (high) for at least 1 clock after an abort.

An abort will typically occur on SYNC time-outs (when a peripheral is driving SYNC longer than allowed), on cycles where there is no response (which will occur if the host is programmed incorrectly), or if a device drives a reserved SYNC value. The abort is used to end the current cycle and to make sure that all devices are ready for the next cycle. An abort may also be used to cause a "soft reset" of the LPC I/F. This soft reset could be initiated via a bit or register in the host. It is recommended that aborts not be used when "long" SYNC cycles are being added.

The abort does not affect any LDRQ# sequences.

Figure 4: Abort Mechanism



On target I/O, target memory, and DMA cycles, if the host signals an abort before the peripheral has asserted the 'ready' or 'error' SYNC, the cycle is terminated. No data is to be transferred to the host on I/O and Memory reads/DMA writes, and the data written to the peripheral on I/O and Memory writes/DMA reads is to be ignored.

On bus master cycles, if the host signals an abort at any time, the cycle will be terminated. No data will be returned from the host on bus master reads, and any data transferred to the host on bus master writes will be ignored.

4.3 Firmware Memory Cycle Overview

This section describes a memory cycle type intended for system BIOS firmware. A slightly different chip select and addressing mechanism is used for this cycle type.

4.3.1 Field Definitions

4.3.1.1 START

This one clock field indicates the start of a cycle. It is valid on the last clock that LFRAME# is sampled low. The two start fields that are used for the cycle are shown in the table below. If the start field that is sampled is not one of these values, then the cycle attempted is not a Firmware Memory Cycle. It may be a valid memory cycle that the Firmware component may wish to decode, i.e. it may be of the standard memory cycle variety.

AD[3:0]	Indication
1101	Firmware Memory Read
1110	Firmware Memory Write

4.3.1.2 IDSEL (Device Select)

This one clock field is used to indicate which of multiple Firmware components is being selected. The four bits transmitted over AD[3:0] during this clock are compared with values strapped onto pins on the Firmware component. If there is a match, the Firmware component will continue to decode the cycle to determine which bytes are requested on a read or which bytes to update on a write. If there isn't a match, the Firmware component may discard the rest of the cycle and go into a standby power state.

4.3.1.3 MADDR (Memory Address)

This is a 7 clock field that gives a 28 bit memory address. This allows for up to 256MB per memory device, for a total of a 4GB addressable space. The address is transferred with the most significant nibble first.

4.3.1.4 MSIZE (Memory Size)

The encodings for this field are shown in Table 4 below. Single byte transfers are required for all devices that support Firmware Memory cycles. Multi-byte transfers are optional and both the host and the device require a mechanism to inform software of available support (i.e. capabilities register bit). Note that the definition of this mechanism is beyond the scope of this specification. For multi-byte writes, FLASH memory devices may require an increased programming voltage (Vpp) to perform a 4 byte memory write. The 1 or 2 byte memory write should be used if a higher

programming voltage is not available, but required by the peripheral to support the 4 byte memory write.

Table 4: Firmware Memory Size Field

Bits	Direction	Size of Transfer
0000	R / W	1 byte
0001	R / W	2 bytes (optional). Accesses must be aligned to a WORD boundary
0010	R / W	4 bytes (optional). Accesses must be aligned to a DWORD boundary
0011		**Reserved**: If this value is observed by the peripheral, the cycle must be ignored.
0100	R	16 bytes (optional). Accesses must be aligned to a 16 byte boundary.
0101-0110		**Reserved**: If this value is observed by the peripheral, the cycle must be ignored.
0111	R	128 bytes (optional). Accesses must be aligned to a 128 byte boundary.
1000-1111		**Reserved**: If this value is observed by the peripheral, the cycle must be ignored.

4.3.1.5 TAR

The TAR protocol is the same as described in Section 4.2.1.4.

4.3.1.6 SYNC

The SYNC fields are the same as described in Section 4.2.1.8.

4.3.1.7 DATA

This field is (2 * N) clocks wide, representing the transfer of "N" data bytes as determined by MSIZE field (see Table 4). The host drives it on firmware memory cycles when data is flowing to the peripheral (write cycle), and by the peripheral when data is flowing to the host (read cycle). Each byte of data is driven in little endian format, with the least significant nibble first. This means that for each byte, on the first clock, Data[3:0] is driven, and on the second clock, Data[7:4] is driven. It also means that the address of each subsequent data byte is incremented sequentially through the transfer.

4.3.1.8 Protocol

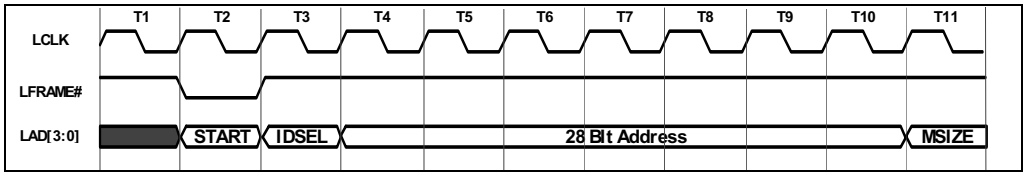
The Firmware Memory cycles use a sequence of events that begin with a START field (LFRAME# active with appropriate AD[3:0] combination) and end with the data transfer. The sections below describe the cycles in detail.

4.3.1.9 Preamble

The initiation of the Firmware Memory cycles is shown in Figure 5. The Firmware Memory transaction begins with LFRAME# going low and a START field driven on AD[3:0]. For Firmware Memory Read cycles, the START field must be '1101b'; for Firmware Memory Write cycles, the

START field must be ‘1110b’. Following the START field is the IDSEL field. This field acts like a chip select in that it indicates which device should respond to the current transaction. The next seven clocks are the 28-bit address from where to begin reading in the selected device. Next, the MSIZE value indicates the number of bytes to be transferred.

Figure 5: Firmware Memory Cycle Preamble



4.3.1.10 Firmware Memory Read Cycle

For read cycles, after the pre-amble described in Section 4.3.1.9, the host drives a TAR field to give ownership of the bus to the Firmware component. After the second clock of the TAR phase the target device assumes the bus and begins driving SYNC values. When the target device is ready, it drives the data, with the least significant nibble first, until all data is transferred and then follows with a TAR cycle to give control back to the host.

Figure 6: Firmware Memory Cycle Single Byte Read

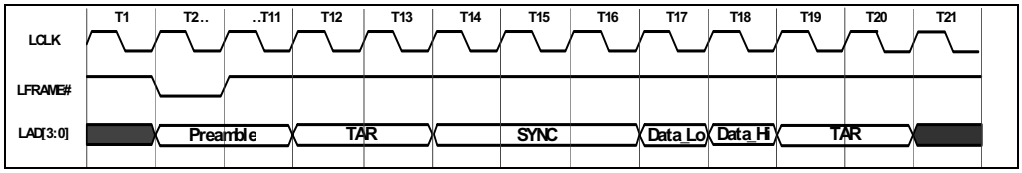


Figure 6 shows a device transferring 1 byte of data and requiring 3 SYNC clocks to access data. Since the access time can begin once the address phase has been completed, the two clocks of the TAR phase can be considered as part of the access time of the part. For example, a device with a 120ns access time could assert ‘0101b’ for clocks 1 and 2 of the SYNC phase and ‘0000b’ for the last clock of the SYNC phase. This would be equivalent to 5 clocks worth of access time if the device started that access at the conclusion of the Preamble phase. Once SYNC is achieved, the device then returns the data in two clocks and gives ownership of the bus back to the host with a TAR phase. For multi-byte reads, the additional data bytes are transferred in a sequential manner immediately following the first byte (i.e. between times T18 and T19 in Figure 6) and then followed by the TAR phase.

4.3.1.11 Firmware Memory Write Cycles

All devices that support Firmware Memory Write cycles must support single byte writes. Further support of multi-byte Firmware Memory Write cycles is optional. Firmware Memory Write cycles use the same preamble as Firmware Memory Read cycles described in Section 4.3.1.9.

Figure 7: Firmware Memory Cycle Single Byte Write

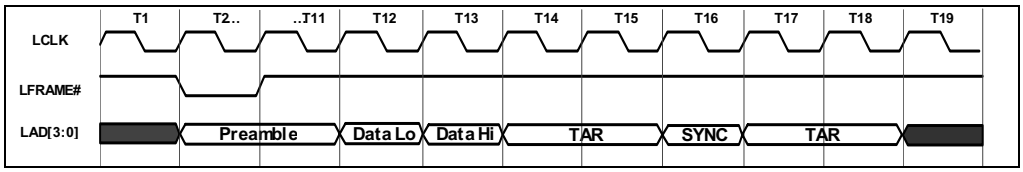


Figure 7 shows a Firmware Memory Write cycle where a single byte is transferred. The master asserts an MSIZE value of 0. After the address has been transferred, the 2 clock data phase begins. Following the data phase, bus ownership is transferred to the Firmware component with a TAR cycle. Following the TAR phase, the device must assert a SYNC value of '0000b' (ready) or '1010b' (error) indicating whether the data has been received. Refer to Section 4.3.1.12 for a description of possible errors. Bus ownership is then given back to the master with another TAR phase. For multi-byte writes, the additional data bytes are transferred in a sequential manner immediately following the first byte (i.e. between times T13 and T14 in Figure 7) and then followed by the first TAR phase.

Firmware Memory Writes only allow one clock for the SYNC phase. The TAR + SYNC + TAR phases at the end of Firmware memory write cycles must be exactly 5 clocks.

4.3.1.12 Error Reporting

There is no error reporting over the Firmware interface for Firmware Memory cycles. If an error occurs, such as an address out of range or an unsupported memory size, the cycle will continue from the host unabated. This is because these errors are the result of 'illegal' programming, and there is no efficient error reporting method that can be done to counter the programming error.

Therefore, the Firmware component must not report the error conditions over the LPC interface. It must only report wait states and the 'ready' condition. It may choose to log the error internally to be debugged, but it must not signal an error through the LPC interface itself.

5 Target Protocol

5.1 Memory Cycles

In these cases, memory read or write cycles are intended for memory-mapped devices. The Sync time will depend on the speed of the device.

The ADDR field is a full 32 bits, and transmitted with the most significant nibble first. Although a full 32 bits of addressing are supported, a memory device will typically support much less than this.

Table 5: Target Memory Cycle Field Definitions

Field	# Clocks	Comment	
START	1	**Start of Cycle:** '0000b' to indicate a start of a cycle.	
CYCTYPE + DIR	1	**Cycle Type:** Indicates the type of cycle. Bits 3:2 must be '01b' for memory cycle. Bit 1 indicates the direction of the transfer: '0b' for read, '1b' for write. Bit 0 is reserved.	
ADDR	8	**Address Phase for Memory Cycle:** This is the 32-bit memory address. It is transferred with the most significant nibble first.	
TAR	2	**Turn-Around Time:** The last component driving LAD[3:0] will drive it to "1111b" during the first clock, and tri-state it during the second clock.	
SYNC	N	**Sync:** Allows peripheral or host to synchronize (add wait-states). Generally, the peripheral or host drives 0101 or 0110 until no more wait-states are needed. At that point it will drive 0000. All other combinations are reserved. If the host sees a reserved combination, it is allowed to abort the transfer.	
		Bits	**Indication**
		0000	Sync Achieved with no error.
		0101	Indicates that Sync not Achieved yet, but the part is driving the bus.
		0110	Indicates that Sync not Achieved yet, but the part is driving the bus, and expects a long Sync.
		1010	Special Case: peripheral indicating errors. See Section 4.2.1.10 for details.
Data	2 (1 byte)	**Data Phase:** The data byte is transferred least significant nibble first (D[3:0] on LAD[3:0], then D[7:4] on LAD[3:0]).	

Table 6:Host Initiated Memory Read

Memory Read	Driven By	Clocks
START	Host	1
CYCTYPE + DIR	Host	1
ADDR	Host	8
TAR	Host	2
SYNC	Peripheral	5
DATA	Peripheral	2
TAR	Peripheral	2
Total Clocks	**21**	
Access Time (us)	**0.63**	
Bandwidth(*MB/x*)	**1.59**	

In the above example, a SYNC value of 5 clocks was chosen for reading the first byte. This value is based on known access times to memory components such as EPROMs, with typical access times of 120ns.

Table 7: Host Initiated Memory Write

Memory Read	Driven By	Clocks
START	Host	1
CYCTYPE + DIR	Host	1
ADDR	Host	8
DATA	Host	2
TAR	Host	5
SYNC	Peripheral	2
TAR	Peripheral	2
Total Clocks	**17**	
Access Time (us)	**0.51**	
Bandwidth(*MB/x*)	**1.96**	

In the above example, it is assumed that no wait states are necessary on the peripheral, because the memory device posts the data. Therefore, all that is necessary is the ready sync indicator.

5.2 I/O Cycles

In these cases, I/O read or write cycles are intended for the peripheral. These will generally be used for register or FIFO accesses, and will generally have minimal Sync times. The minimum number of wait-states between bytes is 1. Enhanced Parallel Port cycles will depend on the speed of the

external device, and may have much longer Sync times. Data transfers are assumed to be exactly 1 byte. The host is responsible for breaking up larger data transfers into 8 bit cycles.

Table 8: Target I/O Cycle Field Definitions

Field	# Clocks	Comment	
START	1	**Start of Cycle:** '0000b' to indicate a start of a cycle.	
CYCTYPE + DIR	1	**Cycle Type:** Indicates the type of cycle. Bits 3:2 must be '00b' for I/O cycle. Bit 1 indicates the direction of the transfer: '0b' for read, '1b' for write. Bit 0 is reserved and must be ignored by the peripheral.	
ADDR	4	**Address Phase for I/O Cycle:** 16 bit I/O address transferred with the most significant nibble first.	
TAR	2	**Turn-Around Time:** The last component driving LAD[3:0] will drive it to '1111b' during the first clock, and tri-state it during the second clock.	
Sync	N	**Synchronize:** Allows peripheral add wait-states. The peripheral drives '0101b' or '0110b' until no more wait-states are needed. At that point it will drive '0000b'. All other combinations are reserved.	
		Bits	**Indication**
		0000	Sync Achieved with no error.
		0101	Indicates that Sync not Achieved yet, but the part is driving the bus.
		0110	Indicates that Sync not achieved yet, but the part is driving the bus, and expects a long Sync (probably due to EPP cycle).
		1010	Special Case: peripheral indicating errors. See Section 4.2.1.10for details.
Data	2 (1 byte)	**Data Phase:** The data byte is transferred with the least significant nibble first: (Data[3:0] on LAD[3:0], then Data[7:4] on LAD[3:0]).	

Table 9: Host Initiated I/O Read Cycles

Memory Read	Driven By	Clocks
START	Host	1
CYCTYPE + DIR	Host	1
ADDR	Host	4
TAR	Host	2
SYNC	Peripheral	1
DATA	Peripheral	2
TAR	Peripheral	2
Total Clocks	**13**	
Access Time (us)	**0.39**	
Bandwidth(*MB/x*)	**2.56**	

In the above example, it is assumed that the peripheral has no wait states necessary to deliver read data, because it is most likely being pulled from an internal FIFO.

Table 10: Host Initiated I/O Write Cycles

Memory Read	Driven By	Clocks
START	Host	1
CYCTYPE + DIR	Host	1
ADDR	Host	4
DATA	Host	2
TAR	Host	2
SYNC	Peripheral	1
TAR	Peripheral	2
Total Clocks	**13**	
Access Time (us)	**0.39**	
Bandwidth(*MB/x*)	**2.56**	

In the above example, it is assumed that no wait states are necessary on the peripheral. Therefore, all that is necessary is the ready sync indicator.

5.3 Firmware Memory Cycles

Firmware memory read or write cycles are intended for PC system boot firmware although they can be used for any memory cycle. The Sync time will depend on the speed of the device.

The ADDR field is 28bits, and transmitted with the most significant nibble first. Although a full 28 bits of addressing are supported, a memory device may support much less than this.

Table 11: Target Firmware Memory Cycle Field Definitions

Field	# Clocks	Comment	
START	1	**Start of Cycle:** '1101b' to indicate a start of a firmware memory read cycle or '1110b' to indicate start of a firmware memory write cycle.	
IDSEL	1	**ID Select:** Selects the targeted firmware component based on device pin straps.	
ADDR	7	**Address Phase for Firmware Memory Cycle:** This is the 28-bit memory address. It is transferred with the most significant nibble first. The device selected by IDSEL field uses this to address its internal memory array.	
MSIZE	1	**Memory Size:** Indicates the number of bytes (N) to be transferred. See Table 4 above.	
TAR	2	**Turn-Around Time:** The last component driving LAD[3:0] will drive it to "1111b" during the first clock, and tri-state it during the second clock.	
SYNC	N	**Sync:** Allows peripheral or host to synchronize (add wait-states). Generally, the peripheral or host drives 0101 or 0110 until no more wait-states are needed. At that point it will drive 0000. All other combinations reserved. If the host sees a reserved combination, it is allowed to abort the transfer.	
		Bits	**Indication**
		0000	Sync Achieved with no error.
		0101	Indicates that Sync not Achieved yet, but the part is driving the bus
		0110	Indicates that Sync not Achieved yet, but the part is driving the bus, and expects a long Sync.
Data	2 * N (N bytes)	**Data Phase:** The data byte is transferred least significant nibble first (D[3:0] on LAD[3:0], then D[7:4] on LAD[3:0]), and so on.	

Table 12: Host Initiated Firmware Memory Read

Single Byte Memory Read	Driven By	Clocks
START	Host	1
IDSEL	Host	1
ADDR	Host	7
MSIZE	Host	1
TAR	Host	2
SYNC	Peripheral	3
DATA	Peripheral	2
TAR	Peripheral	2
Total Clocks	**19**	
Access Time (us)	**0.57**	
Bandwidth(*MB/x*)	**1.75**	

128 Byte Memory Read	Driven By	Clocks
START	Host	1
IDSEL	Host	1
ADDR	Host	7
MSIZE	Host	1
TAR	Host	2
SYNC	Peripheral	3
DATA	Peripheral	256
TAR	Peripheral	2
Total Clocks	**273**	
Access Time (us)		
Bandwidth(*MB/x*)		

In the above example, a SYNC value of 3 clocks was chosen for reading the first byte.

Table 13: Host Initiated Firmware Memory Write

Single Byte Memory Write	Driven By	Clocks
START	Host	1
IDSEL	Host	1
ADDR	Host	7
MSIZE	Host	1
DATA	Host	2
TAR	Host	2
SYNC	Peripheral	1
TAR	Peripheral	2
Total Clocks	**17**	
Access Time (us)	**0.52**	
Bandwidth(*MB/x*)	**1.94**	

4 Byte Memory Write	Driven By	Clocks
START	Host	1
IDSEL	Host	1
ADDR	Host	7
MSIZE	Host	1
DATA	Host	8
TAR	Host	2
SYNC	Peripheral	1
TAR	Peripheral	2
Total Clocks	**23**	
Access Time (us)	**0.70**	
Bandwidth(*MB/x*)	**5.74**	

In the above examples, it is assumed that no wait states are necessary on the peripheral because the memory device posts the data. Therefore, all that is necessary is the ready sync indicator.

6 Direct Memory Access (DMA) Protocol

6.1 Introduction

DMA on LPC is handled through the use of the LDRQ# lines from peripherals and special encodings on LAD[3:0] from the host. Single, Demand, Verify, and Increment mode are supported on the LPC interface. Block, decrement, and cascade modes are not supported. Channel's 0 - 3 are 8 bit channels. Channel's 5 - 7 are 16 bit channels. Channel 4 is reserved as a generic bus master request (see Section 7).

A new 32 bit mode of transfer is supported on the LPC interface, which can be used to transfer multiple bytes from an 8 or 16 bit channel per request for faster throughput.

6.2 Asserting DMA Requests

Peripherals that need DMA service encode their requested channel number on the LDRQ# signal. To simplify the protocol, each peripheral on the LPC I/F has its own dedicated LDRQ# signal (they may not be shared between two separate peripherals). The host will have at least two LDRQ# inputs, allowing at least two devices to support DMA or bus mastering. A single device that supports both DMA and bus mastering only needs one LDRQ# signal.

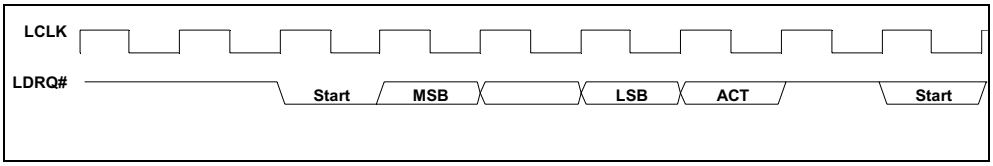
LDRQ# is synchronous with LCLK. As shown in Figure 8, the peripheral uses the following serial encoding sequence:

- Starts the sequence by asserting LDRQ# low (start bit). LDRQ# is high during idle conditions.
- The next 3 bits contain the encoded DMA channel number (MSB first).
- The next bit (ACT) indicates whether the request for the indicated DMA channel is active or inactive. The ACT bit will be a 1(high) to indicate if it is active and 0 (low) if it is inactive. The case where ACT is low will be rare, and is only used to indicate that a previous request for that channel is being abandoned. See Section 6.3 for reasons for abandoning DMA requests.
- After the active/inactive indication, the LDRQ# signal must go high for at least 1 clock. After that one clock, LDRQ# signal can be brought low to the next encoding sequence.

If another DMA channel also needs to request a transfer, another sequence can be sent on the LDRQ# signal. For example, if an encoded request is sent for channel 2, and then channel 3 needs a transfer before the cycle for channel 2 is run on the interface, the peripheral can send the encoded

request for channel 3. This allows multiple DMA agents behind an I/O device to request use of the LPC interface, and the I/O device does not need to self-arbitrate before sending the message.

Figure 8: DMA Request Assertion through LDRQ#



6.3 Abandoning DMA Requests

DMA Requests can be de-asserted in two fashions: on error conditions by sending an LDRQ# message with the 'ACT' bit set to '0', or normally through a SYNC field during the DMA transfer. This section describes boundary conditions where the DMA request needs to be removed prior to a data transfer. See Section 6.4.3 to see how DMA requests are terminated through a DMA transfer.

There may be some special cases where the peripheral desires to abandon a DMA transfer. The most likely case of this occurring is due to a floppy disk controller that has overrun or underrun its FIFO, or software stopping a device prematurely.

In these cases, the peripheral wishes to stop further DMA activity. It may do so by sending an LDRQ# message with the ACT bit as '0'. However, since the host received the DMA request, there is no way to determine that the cycle hasn't been granted and will shortly run on LPC. Therefore, peripherals must take into account that a DMA cycle may still occur. The peripheral can choose not to respond to this cycle, in which case the host will abort it, or it can choose to complete the cycle normally with any random data.

This method of DMA de-assertion should be prevented whenever possible, to limit boundary conditions both on the host and the peripheral.

6.4 DMA Transfers

Arbitration for DMA channels is performed through the 8237 DMA Controller within the host. Once the host has won arbitration on behalf of a DMA channel, it asserts LFRAME# on the LPC bus and begins the DMA transfer. The general flow for a basic DMA transfer is as follows:

1. Host starts transfer by asserting '0000b' on LAD[3:0] with LFRAME# asserted.
2. Host asserts 'cycle type' of DMA, direction based on DMA transfer direction.
3. Host asserts channel number and, if applicable, terminal count.
4. Host indicates the size of the transfer: 8, 16, or 32 bits.
5. If a DMA read:
   - The host drives the first 8 bits of data and turns the bus around.
   - The peripheral acknowledges the data with a valid SYNC.
   - If a 16 bit transfer, the process is repeated for the next 8 bits.
   - If a 32 bit transfer, the process is repeated two more times for each 8 bits of data.

6. If a DMA write:
   - The host turns the bus around and waits for data.
   - The peripheral indicates data ready through SYNC and transfers the first byte.
   - If a 16 bit transfer, the peripheral indicates data ready and transfers the next byte.
   - If a 32 bit transfer, the peripheral repeats the SYNC + data transfer two more times.
7. The peripheral turns around the bus.

The fields used in the DMA transfer described above are shown in Table 14:

Table 14: DMA Field Definitions

<table>
<tr><th>Field</th><th># Clocks</th><th colspan="4">Comment</th></tr>
<tr><td>START</td><td>1</td><td colspan="4">Start of Cycle: ‘0000b’ for DMA. The start condition is the last clock of LFRAME# active.</td></tr>
<tr><td>CYCTYPE
+
DIR</td><td>1</td><td colspan="4">Cycle Type: Bits [3:2] are ‘10b’ to indicate DMA. Bit 2 indicates the transfer direction: ‘0’ for DMA read (host to peripheral), and ‘1’ for DMA write (peripheral to host). Bit 0 is reserved and must be ignored by the peripheral.</td></tr>
<tr><td>SIZE</td><td>1</td><td colspan="4">Size of Transfer: Bits [3:2] are reserved and must be ignored by the peripheral.</td></tr>
<tr><td colspan="2" rowspan="6"></td><td>LAD[1:0]</td><td colspan="3">Transfer size</td></tr>
<tr><td>00</td><td colspan="3">8 bit</td></tr>
<tr><td>01</td><td colspan="3">16 bit</td></tr>
<tr><td>10</td><td colspan="3">Reserved</td></tr>
<tr><td>11</td><td colspan="3">32 bit</td></tr>
<tr><td colspan="4">32 bit transfers are optional. If the host drives a size field of ‘10b’, the peripheral must ignore the transfer.</td></tr>
<tr><td>TAR</td><td>2</td><td colspan="4">Turn-Around: The last component driving LAD[3:0] will drive it high during the first clock, and tri-state during the second clock.</td></tr>
<tr><td>CHANNEL</td><td>1</td><td colspan="4">Channel #: Used only for DMA cycles to indicate channel # being granted. The LAD[2:0] bits indicate the channel number being granted, and LAD[3] indicates the TC bit. See Section 6.4.1 for a description of terminal count. The encoding on LAD[2:0] for channel number is as follows:</td></tr>
<tr><td colspan="2" rowspan="6"></td><td>Bits</td><td>Indication</td><td>Bits</td><td>Indication</td></tr>
<tr><td>000</td><td>Channel 0</td><td>100</td><td>Reserved</td></tr>
<tr><td>001</td><td>Channel 1</td><td>101</td><td>Channel 5</td></tr>
<tr><td>010</td><td>Channel 2</td><td>110</td><td>Channel 6</td></tr>
<tr><td>011</td><td>Channel 3</td><td>111</td><td>Channel 7</td></tr>
<tr><td colspan="4">If a peripheral sees an encoding for a DMA channel that it did not request, or an encoding for channel 4, it must ignore the cycle.</td></tr>
</table>

Table 14: DMA Field Definitions, *continued*

Field	# Clocks	Comment	
SYNC	1 - N	**Sync:** Allows peripheral or host to synchronize (add wait-states). Generally, the peripheral or host drives '0101b' or '0110b' until no more wait-states are needed. At that point it will drive '0000b'. All other combinations are reserved.	
		Bits	**Indication**
		0000	Sync achieved with no error. Also indicates no more transfers desired for that channel, and DMA request is de-asserted.
		0101	Part indicating wait states.
		0110	Part indicating wait states, and many wait states will be added.
		1010	Sync achieved with error. Also indicates no more transfers desired for that channel, and DMA request is de-asserted.
		1001	Sync achieved with no error and more DMA transfers desired to continue after this transfer.
		If the host sees a SYNC value that is not one of the above combinations, it is allowed to abort the transfer.	
DATA	1, 2, 4 bytes	**Data Phase:** Data bytes transferred with the least significant nibble first (D[3:0] on LAD[3:0], then D[7:4] on LAD[3:0]). For transfers greater than 1 byte, the bytes are sent with least significant byte first.	

6.4.1 Terminal Count

Terminal count is communicated through LAD[3] on the same clock that DMA channel is communicated on LAD[2:0]. This field is the CHANNEL field. Terminal count indicates the last byte of transfer, based upon the size of the transfer.

For example, on an 8 bit transfer size (SIZE field is '00b'), if the TC bit is set, then this is the last byte. On a 16 bit transfer (SIZE field is '01b'), if the TC bit is set, then the second byte is the last byte. Finally, on a 32 bit transfer (SIZE field is '11b'), if the TC bit is set, then the fourth byte is the last byte. The peripheral, therefore, must internalize the TC bit when the CHANNEL field is communicated, and only signal TC when the last byte of that transfer size has been transferred.

6.4.2 Verify Mode

Verify mode is supported on the LPC interface. A verify transfer to the peripheral is similar to a DMA write, where the peripheral is transferring data to main memory. The indication from the host is the same as a DMA write, so the peripheral will be driving data onto the LPC interface. However, the host will not transfer this data into main memory.

6.4.3 DMA Request De-Assertion

An end of transfer message is communicated to the host through a special SYNC field transmitted by the peripheral. An LPC device must not attempt to signal the end of a transfer by de-asserting LDRQ#. If a DMA transfer is several bytes, such as a transfer from a demand mode device, the host needs to know when to de-assert the DMA request based on the data currently being transferred.

The DMA agent uses a SYNC encoding on each byte of data being transferred, which indicates to the host whether this is the last byte of transfer or if more bytes are requested. To indicate the last byte of transfer, the peripheral uses a SYNC value of '0000b' (Ready), or '1010b' (Error). These encodings tell the host that this is the last piece of data transferred on a DMA read (host to peripheral), or the byte that follows is the last piece of data transferred on a DMA write (peripheral to host).

When the host sees one of these two encodings, it ends the DMA transfer after this byte and de-asserts the DMA request to the 8237 controller. Therefore, if the host indicated a 16 bit transfer, the peripheral can end the transfer after one byte by indicating a SYNC value of '0000b' or '1010b'. The host will not attempt to transfer the second byte, and will de-assert the DMA request internally. This also holds true for any byte in a 32 bit transfer. This allows the peripheral, therefore, to terminate a DMA burst.

If the peripheral indicates a '0000b' or '1010b' SYNC pattern on the last byte of the indicated size, then the host will only de-assert the DMA request to the 8237 controller since it does not need to end the transfer.

If the peripheral wishes to keep the DMA request active, then it uses a SYNC value of '1001b' (ready plus more data). This tells the 8237 controller that more data bytes are requested after the current byte has been transferred, so the host will keep the DMA request active to the 8237 controller. Therefore, on an 8 bit transfer size, if the peripheral indicates a SYNC value of '1001b' to the host, the data will be transferred and the DMA request will remain active to the 8237 controller. At a later time, the host will then come back with another START ⇒ CYCTYPE ⇒ CHANNEL ⇒ SIZE etc. combination to initiate another transfer to the peripheral.

The peripheral must not assume that the next START indication from the host is another grant to the peripheral if it had indicated a SYNC value of '1001b'. On a single mode DMA device, the 8237 controller will re-arbitrate after every transfer. Only demand mode DMA devices can be assured to receive the next START indication from the host.

Note: Indicating a '0000b' or '1010b' encoding on the SYNC field of an odd byte of a 16 bit channel (first byte of a 16 bit transfer, third byte of a 32 bit transfer) is an error condition. The host will stop the transfer on the LPC bus as indicated, fill the upper byte with random data on DMA writes (peripheral to memory), and indicate to the 8237 controller that the DMA transfer occurred, incrementing the controller's address and decrementing its byte count.

6.4.4 SYNC field / LDRQ# Rules

Since DMA transfers on LPC are requested through an LDRQ# assertion message (described in Section 6.2), and are ended through a SYNC field during the DMA transfer, the peripheral must obey the following rule when initiating back-to-back transfers from a DMA channel.

The peripheral must not assert another message for 8 LCLKs after a de-assertion is indicated through the SYNC field. This is needed to allow the 8237 DMA Controller, which typically runs off a much slower internal clock, to see a message de-asserted before it is re-asserted so that it can arbitrate to the next DMA agent.

6.4.5 Performance Analysis

The performance of DMA on the LPC interface is shown in the following examples:

Table 15: DMA Read Cycle (Host to Peripheral)

DMA Read	Driven By	8 Bit	16 Bit	32 Bit
START	Host	1	1	1
CYCTYPE + DIR	Host	1	1	1
CHANNEL	Host	1	1	1
SIZE	Host	1	1	1
DATA	Host	2	4	8
TAR	Host	2	4	8
SYNC	Peripheral	1	2	4
TAR	Peripheral	2	4	8
Total Clocks		**11**	**18**	**32**
Access Time (us)		**0.33**	**0.54**	**0.96**
Bandwidth(*MB/x*)		**3.03**	**3.70**	**4.17**

In the above example, it is assumed that when the host transfers data to the peripheral, it is ready to take it instantly and does not have to add wait states. There is a SYNC ⇒ TAR for every byte. This figure does not show additional latency on the host to fetch the data from main memory. Therefore, the effective bandwidth will be less than what is shown, by a factor that is determined by overall system latency.

Table 16: DMA Write Cycle (Peripheral to Host)

DMA Read	Driven By	8 Bit	16 Bit	32 Bit
START	Host	1	1	1
CYCTYPE + DIR	Host	1	1	1
CHANNEL	Host	1	1	1
SIZE	Host	1	1	1
TAR	Host	2	2	2
SYNC	Peripheral	1	2	4
DATA	Peripheral	2	4	8
TAR	Peripheral	2	2	2
Total Clocks		**11**	**14**	**20**
Access Time (us)		**0.33**	**0.42**	**0.60**
Bandwidth(*MB/x*)		**3.03**	**4.76**	**6.67**

In the above example, it is assumed that the peripheral has all data necessary to perform the transfer, and therefore does not need to add wait states. In addition, DMA writes assume that the host has enough room to take the data requested, or else it would not have asserted the transfer to begin width. For example, if it only had room to take a 16 bit transfer, it would not ask for a 32 bit transfer.

Therefore, the peripheral does not need to turn around the bus between each byte and see whether the host can accept more transfers. This increases the overall bandwidth significantly on 16 and 32 bit transfers. Once again, this example does not take into account the overall latency required to place the data from the DMA transfer into main memory, so effective bandwidth will be less than what is shown.

6.5 Other Notes on 16 and 32 Bit DMA

Under default operation, the host will only perform 8 bit transfers on 8 bit channels and 16 bit transfers on 16 bit channels. In order to enable 16 and 32 bit transfers on 8 bit channels, and 32 bit transfers on 16 bit channels, the peripheral must communicate to system BIOS that larger transfer sizes are allowed. If the host has this capability, the BIOS will program the host to attempt larger transfer sizes.

The method by which this communication between host and peripheral through system BIOS is performed is beyond the scope of this specification. Since the host and peripheral are motherboard devices, no “plug-n-play” registry is required.

The peripheral must not assume that the host will be able to perform transfer sizes that are larger than the size allowed for the DMA channel, and be willing to accept a SIZE field that is smaller than what it may currently have buffered.

To that end, it is recommended that future devices that may appear on the LPC bus, which require higher bandwidth than 8 bit or 16 bit DMA allow, can do so with a bus mastering interface and not rely on the 8237 controller.

If the host can perform larger transfer sizes, it will attempt to do so in all cases except the following:

- For 8-bit channels (0 - 3), if the buffer starts on other than a 32-bit boundary, the host will perform 8-bit transfers (1, 2, or 3 times) until it reaches a 32-bit boundary. At that point it will start attempting 32-bit transfers. This is to keep its internal buffer aligned to 32 bit memory.
- For 16-bit channels (5 - 7), if the buffer starts on other than a 32-bit boundary, the host will perform a 16-bit transfer, then continue with 32-bit transfers.
- For 8-bit channels (0 - 3), if one byte remains prior to terminal count, the host will not attempt a 16 bit transfer. Instead it will perform an 8 bit transfer.
- For 8-bit channels (0 - 3), if three bytes remain prior to terminal count, the host will not attempt to perform a 32-bit transfer. Instead, it will perform three 8-bit transfers. The last of these 8-bit transfers will have TC set.
- For 16-bit channels (5 - 7), if only two bytes remain prior to terminal count being reached, the host will not attempt to perform a 32-bit transfer. Instead, it will perform one 16-bit transfer with TC set.

If the peripheral performs one of the larger transfer sizes, it must not request a DMA transfer until it can transfer the number of bytes specified in its maximum size field. The one exception to this rule is when the peripheral has reached the end of its data transfer, in which case it can terminate the DMA transfer through a SYNC value of ‘0000b’.

7 Bus Master Protocol

7.1 Introduction

The LPC interface does not support cascaded ISA master mode access on DMA channels. Bus master accesses are performed through a reserved encoding on the LDRQ# signal line of '100b'. By using this reserved encoding, a peripheral is able to create any number of bus masters, and is not limited to the number of DMA channels in the system.

7.2 Cycle Formats and Timing

Bus master START fields are associated with either Bus Master 0 ('0010b') or Bus Master 1 ('0011b'). At this time, there are only two bus masters supported on the LPC interface. After sending this field, the host performs a TAR to transfer control to the peripheral.

Table 17: Bus Master Cycle Field Definitions

Field	# Clocks	Comment	
START	1	**Start of Cycle:** '0010b' or '0011b' to indicate a start of a cycle for one of the two supported bus masters.	
CYCTYPE + DIR	1	**Cycle Type:** Indicates the type of cycle. Bits (3:2) must be '01b' for memory cycle or '00b' for I/O cycle. Bit 1 indicates the direction of the transfer: '0b' for read, '1b' for write. Bit 0 is reserved and must be ignored.	
SIZE	1	**Size of Transfer:** Indicates the size of the transfer. Bits (3:2) are reserved and must be ignored by the host.	
		LAD[1:0]	**Transfer size**
		00	8 bit
		01	16 bit
		10	*Reserved*
		11	32 bit
TAR	2	**Turn-Around Time:** The last component driving LAD[3:0] will drive it to '1111b' during the first clock, and tri-state it during the second clock.	

Table 17: Bus Master Cycle Field Definitions, *continued*

Field	# Clocks	Comment	
ADDR	4 or 8	**Address Phase for I/O or Memory Cycle:** This is either the 16-bit I/O address or the 32-bit memory address. It is transferred with the most significant nibble first. Note that the address must be aligned to the size of the data transfer. Example:	
		Transfer Size	**Address[1:0]**
		8 bits	XX (all combinations allowed)
		16 bits	x0 (bit 0 must be 0)
		32 bits	00 (bits 1 and 0 must be 0
		The examples above state that a 16-bit transfer must be 16-bit aligned, and a 32-bit transfer must be 32 bit aligned.	
SYNC	N	**Sync:** Allows host to add wait-states. The host drives '0110b' until no more wait-states are needed. At that point it will drive '0000b'. All other combinations reserved.	
		Bits	**Indication**
		0000	Sync Achieved with no error.
		0110	Indicates that Sync not Achieved yet, but the host is driving the bus, and expects a long Sync.
		1010	Special Case: peripheral indicating errors. See Section 4.2.1.10 for details.
DATA	1, 2, or 4 bytes	**Data Phase:** Data bytes transferred with the least significant nibble first (D[3:0] on LAD[3:0], then D[7:4] on LAD[3:0]). For transfers greater than 1 byte, the bytes are sent with the least significant byte first.	

Table 18: Peripheral Initiated Memory Read Cycle

Memory Read Cycle	Driven By	8 Bit	16 Bit	32 Bit
START	Host	1	1	1
TAR	Host	2	2	2
CYCTYPE + DIR	Peripheral	1	1	1
ADDR	Peripheral	8	8	8
SIZE	Peripheral	1	1	1
TAR	Peripheral	2	2	2
SYNC	Host	6	6	6
DATA	Host	2	4	8
TAR	Host	2	2	2
Total Clocks		**25**	**27**	**31**
Access Time (us)		**0.75**	**0.81**	**0.93**
Bandwidth(*MB/x*)		**1.33**	**2.47**	**4.30**

In the example above, 6 clocks were chosen as the wait states initiated by the host. This assumption is based upon a reasonable amount of time to communicate the address up to a PCI bus and receive a read response. This can be much longer in a heavily loaded system, and it most certainly will not be shorter.

Table 19: Peripheral Initiated Memory Write Cycle

Memory Write Cycle	Driven By	8 Bit	16 Bit	32 Bit
START	Host	1	1	1
TAR	Host	2	2	2
CYCTYPE + DIR	Peripheral	1	1	1
ADDR	Peripheral	8	8	8
SIZE	Peripheral	1	1	1
DATA	Peripheral	2	4	8
TAR	Peripheral	2	2	2
SYNC	Host	6	6	6
TAR	Host	2	2	2
Total Clocks		**25**	**27**	**31**
Access Time (us)		**0.75**	**0.81**	**0.93**
Bandwidth(*MB/x*)		**1.33**	**2.47**	**4.30**

Table 20: Peripheral Initiated I/O Read Cycle

I/O Read Cycle	Driven By	8 Bit	16 Bit	32 Bit
START	Host	1	1	1
TAR	Host	2	2	2
CYCTYPE + DIR	Peripheral	1	1	1
ADDR	Peripheral	4	4	4
SIZE	Peripheral	1	1	1
TAR	Peripheral	2	2	2
SYNC	Host	6	6	6
DATA	Host	2	4	8
TAR	Host	2	2	2
Total Clocks		**21**	**23**	**27**
Access Time (us)		**0.63**	**0.69**	**0.81**
Bandwidth(*MB/x*)		**1.59**	**2.90**	**4.94**

Table 21: Peripheral Initiated I/O Write Cycle

I/O Write Cycle	Driven By	8 Bit	16 Bit	32 Bit
START	Host	1	1	1
TAR	Host	2	2	2
CYCTYPE + DIR	Peripheral	1	1	1
ADDR	Peripheral	4	4	4
SIZE	Peripheral	1	1	1
DATA	Peripheral	2	4	8
TAR	Peripheral	2	2	2
SYNC	Host	6	6	6
TAR	Host	2	2	2
Total Clocks		**21**	**23**	**27**
Access Time (us)		**0.63**	**0.69**	**0.81**
Bandwidth(*MB/x*)		**1.59**	**2.90**	**4.94**

7.3 Request Assertion Rules

When asserting bus master requests, peripherals use the LDRQ# line and send an encoding of '100b'. After sending this message, the LDRQ# line must not assert another LDRQ# message of '100b' until the peripheral is granted with a bus master request. It may still send messages for DMA channels.

Once a bus master channel is granted, it may send another LDRQ# message. It may send another message one LCLK after driving the 'CYCTYPE + DIR' field.

Bus master requests are never lost. If a peripheral is granted, and it has no data to transfer, it must perform a read cycle that is supported by the host to ensure that it does not corrupt data. For example, if the host only supports bus master memory cycles, it must perform a memory read.

This page is intentionally left blank.

8 Power Management

8.1 CLKRUN# Protocol

In some low-power states, the LCLK may be stopped. To restart it, the peripheral should assert the PCI CLKRUN# signal. This will be required to drive the LDRQ# signal active. The protocol is defined in further detail in the PCI 2.3 specification.

Once a peripheral has started an LDRQ# sequence, the host must not de-assert CLKRUN# causing the clock to stop.

8.2 LPCPD# Protocol

The general timings to enter and exit the power down state are shown in Figure 9. Note that these timings only apply to entry and exit of low-power states. See Section 9 below for details on system reset characteristics.

Prior to going to a low-power state, the system will assert the LPCPD# signal. It will be asserted at least 30 microseconds prior to the LCLK# signal stopping low and the other host LPC I/F output signals being tri-stated or driven low.

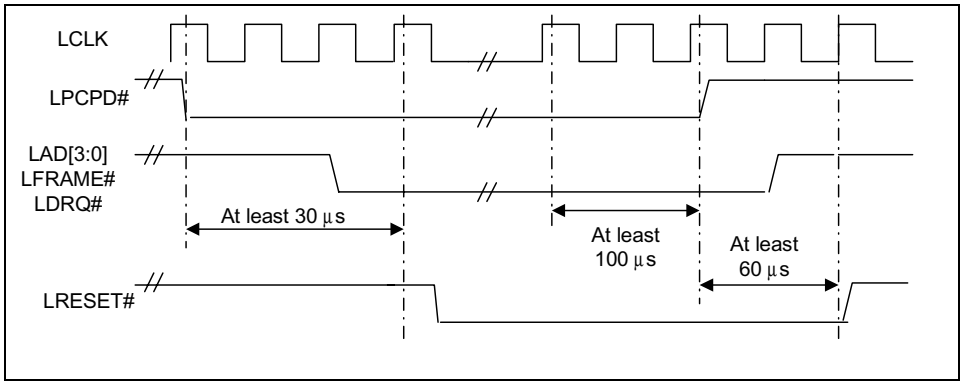
Upon recognizing LPCPD# asserted, there will be no further transactions on the LPC interface. The peripherals should drive the LDRQ# signal low or tri-state, and do so until LPCPD# is de-asserted. This is to prevent the peripherals from driving the LPC interface signals high into a potentially powered-down host.

Upon recognizing LPCPD# de-asserted, the peripheral should drive its LDRQ# signal high.

After LPCPD# is de-asserted, the LPC interface may be reset dependent upon the characteristics of system reset signal connected to LRESET#. Note that if some logic within the peripheral was used to wake the system from the low-power state, that logic should not be reset by LRESET#. Instead, if some type of reset is needed for the waking logic, it should be reset through software mechanisms.

The peripheral must asynchronously recognize LPCPD# going active (it may not meet setup times to LCLK). However, it can sample with LCLK, since it will be running for at least 30 microseconds (after LPCPD# goes low). The peripheral must asynchronously recognize LPCPD# going inactive. However, it can sample with LCLK, since it will be running for more than 30 microseconds prior to LPCPD# going high. From the time LPCPD# is recognized active until LPCPD# is recognized inactive, the peripheral should ignore LCLK.

Figure 9: Timing for Entering and Exiting the Power Down State



(Note this diagram assumes LRESET# is asserted which is system specific and not required.)

8.3 LPME# Usage

LPME# should not be connected to the PCI bus PME# signal in typical PCI based systems. This is because LPC devices do not typically implement the PCI Power Management Configuration Space which is required of PCI devices.

8.4 Lower Voltages

In the future, the LPC I/F may move to using lower voltages for the various signals. This is undefined at this time.

9 Reset Policy

The behavior of the LPC interface and timings of LRESET# are different for general system reset than for entry or exit from a low power state.

The host and peripherals must obey the following rules:

1. When LRESET# de-asserts, the clock is assumed to be running. The exact number of clocks will be the same as for the PCI specification.
2. When LRESET# is asserted, the host will behave as follows:
   - LFRAME# will be driven high
   - LAD[3:0] will be tri-stated
   - LDRQ[n]# is ignored
3. When LRESET# is asserted, peripherals will behave as follows:
   - LFRAME# can be ignored
   - LAD[3:0] will be tri-stated
   - LDRQ[n]# will be driven high

LRESET# assertion for general system reset may occur at any time and may be asynchronous to LCLK. There is no guaranteed LPCPD# to LRESET# timing relationship when entering a system reset condition.

This page is intentionally left blank.

10 Electrical Specification

10.1 LAD[3:0] / LFRAME# / LDRQ# / SERIRQ / LPME#

The AC and DC specifications for these signals are identical to those defined in Section 4.2.2 of the "PCI Local Bus Specification, Rev 2.3". That section contains the specifications for the 3.3V PCI signaling environment.

LAD[3:0] must go high during the TAR phase. The last device driving the LAD[3:0] is responsible to drive the signals high during the first clock of the TAR phase. During the 2nd clock, LAD[3:0] is floated and maintained high by weak pull-up resistors (see Section 10.4 below).

LDRQ# is an output from the peripherals and an input on the host. A separate weak pull-up resistor (see Section 10.4 below) or direct connect to Vcc will be required on each unconnected LDRQ# to make sure they do not accidentally go low when no devices are connected.

10.2 LPCPD# / LSMI#

The LPCPD# signal is driven by the host as a 3.3V output. Its electrical characteristics are given in Table 22 below.

Table 22: LPCPD# Electrical Characteristics

Parameter	Device	Min	Max	Units
V_{OL}	Host		0.4	V
V_{OH}	Host	2.5		V
V_{IL}	Device	-0.5	0.8	V
V_{IH}	Device	2	Vcc* + 0.5	V
* System Dependent Vcc specification				

The LSMI# signal is driven by the device to host. Its electrical characteristics are given in Table 23 below:

Table 23: LSMI# Electrical Characteristics

Parameter	Device	Min	Max	Units
V_{IL}	Host	-0.5	0.8	V
V_{IH}	Host	1.5	Vcc* + 0.5	V
V_{OL}	Device		0.4	V
V_{OH}	Device	2.0		V
* System Dependent Vcc specification				

10.3 LRESET# / LCLK / CLKRUN#

All the other LPC I/F signals are connected to other PCI signals that are system dependent. The peripherals on the LPC I/F should be designed to handle the same signaling levels (such as 5V or 3.3V) as would otherwise be supported in the system.

For example, if the system uses 5V signaling for the PCI Reset and Clock signals, then the peripheral on the LPC I/F should be 5V tolerant. If the system uses 3.3V signaling for the PCI Reset and Clock, then the peripheral on the LPC I/F must be able to recognize the 3.3V signaling levels.

10.4 Signal Pull-Up Requirements

The LAD[3:0] signals require pull-up resistors to maintain their state during the turnaround (TAR) periods of a cycle.

The LDRQ[1:0] signals require pull-ups if they are not connected to a LPC peripheral device. This will keep them in the inactive state.

See Table 24 below for recommended pull-up values. Some host devices will incorporate these pull-ups internally. Other signals may or may not require pull-up resistors, dependent on the specific system implementation.

Table 24: Recommended Pull-Up Values

Signal Name	Pull-Up
LAD[3:0]	15k - 100k ohm
LDRQ[1:0]#	15k - 100k ohm

11 Host / Peripheral Configuration

11.1 Plug and Play

Peripherals on the LPC I/F are assumed to be configured using motherboard or ACPI configuration techniques rather than PCI or ISA plug-n-play protocols. One method is to have a fixed configuration space, such as an index and data register, that can be used to program various parameters for the peripheral. To avoid conflicts, the peripheral should use I/O locations typically reserved for motherboard devices (such as 0000h-00FFh).

11.2 Host Decode Ranges

In order to allow the I/O and memory mapped cycles to go from the PCI bus to the LPC I/F, decoders will be needed in the host. Table 25 shows decode ranges that can be built into the host if necessary. The requirements for host implementations of these ranges are system specific.

Table 25: Legacy Host Decode Ranges

Device	# Ranges	I/O Ranges
Parallel Port	1 of 3 ranges	378h-37Fh (+ 778h-77Fh for ECP) 278h-27Fh (+ 678h-67Fh for ECP) See note 1 below 3BCh-3BFh (+ 7BCh-7BFh for ECP)
Serial Ports	2 of 8 ranges	3F8h-3FFh, 2F8h-2FFh, 220h-227h, 228h-22Fh, 238h-23Fh, 2E8h-2EFh, 338h-33Fh, 3E8h-3EFh
Audio	1 of 4 ranges	SoundBlaster* compatible: 220h-233h, 240-253h, 260h-273h, 280h-293h
MIDI	1 of 4 ranges	300h-301h, 310h-311h, 320h-321h, 330h-331h
MSS	1 of 4 ranges	530h-537h, 604h-60Bh, E80h-E87, F40h-F47h
FDC	1 of 2 ranges	3F0h-3F7h or 370h-377h
Game Ports	2 1 -byte ranges	Each mapped to any one byte in the 200h-20Fh range.
Wide Generic	16-bit base address register. 512 bytes wide	Can be mapped anywhere in lower 64kB. AC'97 and other configuration registers are expected to be mapped to this range. It is wide enough to allow many unforeseen devices to be supported.
KBC	60h and 64h	
ACPI Embedded Controller	62h and 66h	

Device	# Ranges	I/O Ranges
Ad-Lib	388h - 389h	
Super I/O Configuration	2Eh - 2Fh	
Alternate Super I/O Configuration	4E - 4Fh	

1. 279h is Read-Only. Writes to 279h are forwarded to ISA for PnP.

11.3 Bus Master START Fields

Bus Masters have a unique START field indicating their grant. In this specification, two bus masters are defined, with start fields of ‘0010b’ and ‘0011b’. The peripheral must be programmed by BIOS so that it knows which START field is its start field on a bus master request.

12 Bandwidth Calculations

12.1 Introduction

The following bandwidth calculations assume 0 latency on other parts of the system, such as the PCI bus or main memory. In real systems, the latencies will likely be much greater than 0. The performance calculations only indicate the theoretical performance that could be achieved through optimized system design. These performance calculations should not be used in peripheral design, such as for calculating FIFO depths.

12.2 System Performance Requirements

The following performance requirements are based on the peripheral using as much bandwidth as it possibly can, and using 8237 controller compatible cycles. The actual bandwidth will typically be much lower, and future devices and hosts will be able to use the faster 32-bit DMA. Furthermore, the usage will tend to be in bursts. For example, the FDC and parallel port can use a great deal of bandwidth, but typically only when using the floppy or printing. Likewise, the IR interface can use 0.5 Mbytes/s, but only while actually transferring data.

Because of the FIFO nature of the peripherals, some behaviors can be noted:

- The parallel port will typically perform a DMA transfer of 8 data bytes to/from its FIFO. If this is done as four consecutive 16-bit transfers, then this will take an average of 1.44 microseconds. It then takes approximately 4 microseconds to drain the FIFO. Thus, the Parallel Port could use 36% of the bus.
- The IR link will typically perform a DMA transfer of 8 data bytes to/from the FIFO. If this is done as four consecutive 16-bit transfers, then this will take an average of 1.44 microseconds. It then takes approximately 16 microseconds to drain the FIFO. Thus, the IR port could use 9% of the bus.
- The FDC will typically perform a DMA transfer of 8 data bytes to/from the FIFO. If this is done as eight consecutive 8-bit transfers, then this will take an average of 1.92 microseconds. It then takes approximately 32 microseconds to drain the FIFO. Thus, the FDC port could use 6% of the bus.
- The Audio will typically perform a DMA transfer of 8 data bytes to/from the FIFO. If this is done as four consecutive 16-bit transfers, then this will take an average of 1.44 microseconds. It then takes approximately 20.8 microseconds to drain the FIFO. Thus, the Audio could use 7% of the bus.
- A 16550 Serial Port will typically need to do I/O accesses of 8 data bytes to/from the FIFO. If this is done as 8 consecutive 8-bit transfers, then this will take an average of 3.15

microseconds. It then takes approximately 278 microseconds to drain the FIFO. Thus, the Serial Port could use 1.1% of the bus.

Table 26: IO Performance

Function	Max Kbits/sec	Max Mbytes/sec	Transfer	Transfer Time	Transfer Rep Rate	Estimated % Usage	Comments
PP	16,000	2.0000	16-bit DMA	1.82 µs	4 µs	45.5	May need to go faster in future
IR	4,000	0.5000	16-bit DMA	1.82 µs	16 µs	11.4	
FDC	2,000	0.2500	8-bit DMA	2.42 µs	32 µs	7.6	
Audio	3,072	0.3840	16-bit DMA	1.82 µs	20.8 µs	8.7	Full Duplex 16-bit Stereo at 48K samples
SP	230	0.0288	2 x 32-bit I/O	3.15 µs	278 µs	1.1	Full Duplex
Total:						**74.3**	

12.3 Conclusion

From a latency standpoint, the performance may not exactly fit. For example, if all ports are active, the maximum latency could be much greater than 4 microseconds for the Parallel port. However, this is already a limitation in present systems, and hardware and software must be able to handle this.

Long numbers of wait-state may also cause the performance to be lower than the worst-case requirements. However:

- The LPC interface is faster than ISA for all cycle types, so any scenario that would have performance problems for the LPC interface would have these problems exaggerated on ISA.
- The worst-case performance requirements do not seem likely to occur, and since this would already be a problem on today's systems, the software must already have appropriate recovery mechanisms in place.