



PRELIMINARY

“V2” Platform

PCM Decommutator

or

High-Performance PCM Simulator

Technical Manual

***U520401
Glenn Toennes
February 2007***

***Lumistar, Inc
2701 Loker Avenue West Suite 230
Carlsbad, California 92008 USA***

Table of Contents

1 — Introduction

1.1 General	1-5
1.2 Lumistar Universal Daughterboard Family	1-5
1.3 Specifications	1-6
Table 1–1. PCM Decommutator Specifications.....	1-6
Table 1–2. Time Reader Specifications	1-6
Table 1–3. Mechanical Specifications	1-6
Table 1–4. General PCM Simulator Specifications	1-7
Table 1–5. LS50 PCM Simulator Specifications	1-7
Table 1–6. LS70 PCM Simulator Specifications	1-8
Table 1–7. Environmental Specifications	1-8

2 — Installation

2.1 Addressing	2-1
2.2 Physical Installation	2-1
2.3 Indicators	2-1
Figure 2–1. Front Plate.....	2-2
2.4 J1 Interface	2-2
Table 2–1. Switch SW1 Definition	2-3
Table 2–2. E1 Patch Definition	2-3
Table 2–3. J1 I/O Connector Pinout.....	2-4
Table 2–4. Switch SW2 Definition	2-4
2.5 LS50 Parallel Output	2-4
Table 2–5. Parallel Output Pinout	2-5
2.6 RF Control Output	2-5
Table 2–5. J3 RF Control Pinout.....	2-5

3 — Programming Information

3.1 General	3-1
3.2 Locating a PCI Device	3-1
3.3 Register Summaries	3-3
3.4 General Registers	3-3
Table 3–1. General Write Register Summary	3-3
Table 3–2. General Read Register Summary	3-3
3.4.1 Board ID Register	3-3
3.4.2 Identifier Register	3-3
3.5 LS50 Decommutator Registers	3-4
Table 3–3. LS50 Decom Write Register Summary.....	3-4
Table 3–4. LS50 Decom Read Register Summary	3-4
3.5.1 The Control Register	3-5
Table 3–5. Control Register	3-5
3.5.2 Selecting the Input Source	3-5

Table 3–6. Source Control Register	3-6
3.5.3 PCM Code Control.....	3-6
Table 3–7. Decommutator PCM Codes	3-6
3.5.4 The Frame Sync Pattern	3-7
Table 3–8. Polarity Control Register.....	3-8
3.5.5 The Decommutator Format Memory	3-8
Table 3–9. Decommutator Attribute Word	3-9
3.5.6 Major Frame Synchronization.....	3-9
3.5.6.1 SFID Correlation.....	3-10
3.5.6.2 FCC Correlation.....	3-11
3.5.6.3 URC Correlation	3-11
Table 3–10. Major Frame Sync Control Register	3-12
3.5.7 The Decommutator Output	3-12
Table 3–11. Frame Header	3-13
3.5.8 Status	3-13
Table 3–12. Buffer Control and Status Register	3-14
Table 3–13. Status Register.....	3-14
Table 3–14. Header Register	3-15
3.6 The IRIG Time Reader	3-15
Table 3–15. IRIG Reader Write Register Summary	3-15
Table 3–16. IRIG Reader Read Register Summary	3-16
3.6.1 Setting the Real Time Clock	3-16
Table 3–17. IRIG Reader Control Register.....	3-16
3.6.2 Reading Time	3-17
3.7 The LS50 PCM Simulator.....	3-17
Table 3–18. LS50 Simulator Write Register Summary	3-17
Table 3–19. LS50 Simulator Read Register Summary	3-18
3.7.1 Simulator Command Register and Mode Registers.....	3-18
Table 3–20. Simulator Command Register	3-19
Table 3–21. Simulator Mode Register	3-20
Table 3–22. Simulator Frame Start Register.....	3-20
3.7.2 Output Formatting	3-21
Table 3–23. Simulator Encoder Control Register	3-21
3.7.3 The Clock Generator.....	3-22
3.7.4 Communicating With Simulator Memory	3-22
Table 3–24. Simulator Bankswitch Register.....	3-23
3.7.5 The Simulator Memory Map.....	3-23
Table 3–25. Simulator Memory Map.....	3-24
3.7.6 Attributes and Data.....	3-24
Table 3–26. Simulator Word Attributes.....	3-25
Table 3–27. Simulator Frame Attributes.....	3-25
3.7.7 Baseband and RF Control	3-25
3.7.7.1 EEPROM Access	3-25
Table 3–28. RF EEPROM Map	3-26
3.7.7.2 Baseband Output Level	3-26
3.7.7.3 Pre-Mod Filtering.....	3-27

3.7.7.4 External Data Input	3-27
3.7.7.5 The Quasonix Transmitter	3-27
3.8 LS70 PCM Simulator	3-28
Table 3–29. LS70 Simulator Write Register Summary	3-29
Table 3–30. LS70 Simulator Read Register Summary	3-29
3.8.1 Simulator Command Register and Mode Registers	3-31
Table 3–31. Simulator Command Register	3-31
Table 3–32. Simulator Mode Register	3-32
3.8.2 Output Formatting	3-32
Table 3–33. Simulator Frame Stop Register	3-33
3.8.3 The Clock Generator	3-33
3.8.4 Communicating With Simulator Memory	3-33
Table 3–34. Simulator Bankswitch Register.....	3-33
Table 3–35. Simulator Word Attributes.....	3-34
3.8.5 Attributes and Data	3-35
3.8.6 Wave Counters	3-35
3.8.7 PRN Pattern Generators	3-35
Table 3–36. PRN Generator Control Register.....	3-36
3.9 The IRIG Time Generator	3-36
Table 3–37. IRIG Generator Write Register Summary.....	3-36
Table 3–38. IRIG Generator Read Register Summary	3-37
3.9.1 Setting Time	3-37
Table 3–39. IRIG Generator Control Register	3-37
3.9.2 Time Generator LS70 Data Hold	3-37
3.10 Interrupts	3-38
3.10.1 Polling	3-38
3.10.2 Using Interrupts	3-38
3.10.2.1 Connecting to the System	3-39
3.10.2.2 Preparing to be Interrupted	3-39
3.10.2.3 Being Interrupted	3-40
3.11 DMA	3-41
3.11.1 DMA Descriptors	3-42
3.11.2 DMA Channel Mode Register	3-42
3.11.3 DMA Channel Command Register	3-43
3.12 Bit Error Rate Measurement	3-44
Table 3–40. Pattern Registers.....	3-44
Table 3–41. Error Count High Register	3-45
3.13 Daughtercard Interface	3-45
Table 3–42. Daughtercard Write Register Summary	3-45
Table 3–43 Daughtercard Read Register Summary	3-45
Table 3–44. Daughtercard Control Register.....	3-45
Table 3–45. Daughtercard Status Register	3-46
3.13.1 Plug-and-Play	3-46
3.13.2 LS40 Bit Synchronizer	3-46
Table 3–46. LS40 Bit Synchronizer Input Source	3-47
3.13.3 LS38 70MHz Demodulator	3-47

Table 3-47. LS38 Command Packet..... 3-47
Table 3-48. LS38 Status Packet..... 3-48

1 — Introduction

1.1 General

This is a manual rewrite for a major revision of the existing Lumistar PCM Decommutator/PCM Simulator. Inputs from the field, and the shop, indicated some changes were wanted. Technological advances in the capability of Field-Programmable Gate Arrays allow Lumistar to release this new "V2" hardware platform. The platform allows placing the full PCM decommutator/simulator capability of our LS50 product, or two of them, or the full data stream creation capability of our LS70 High-Performance PCM simulator product, or two of them on a single reduced-length PCI card.

This platform is capable of hosting one daughtercard, which may be an LS40 PCM Bit Synchronizer, an LS38 70MHz FM/SOQPSK Demodulator, or one or two of several types of low-powered FM or SOQPSK RF Test Transmitters."

1.2 Lumistar Universal Daughterboard Family

1.3 Specifications

Table 1–1. PCM Decommutator Specifications	
Input Data Rate	<100.0 bps to >33 Mbps
Input Signals	PCM Data and Symbol-Rate clock (>15 Mbps: NRZ-L data & 0-degree clock)
Input Levels	Single-ended TTL & RS-422
Word Length	Variable from 3 to 16 bits per word on a word-by-word basis
CRC checker	CRC16/CCITT
Minor Frame Length	2 to 16,383 words per minor frame
Major Frame Length	Up to 1024 minor frames per major frame
Bit Order	MSB or LSB-first (word-by-word basis)
Frame Sync Pattern	Up to 64 bits (any pattern, including "don't care" bits (X) may be used)
Frame Sync Location	Beginning or end of the frame
Frame Sync Strategy	Adaptive mode (search-lock-verify) & burst mode (search-lock)
Sync Error Tolerance	0 to 15 bits (selectable)
Sync Slip Window	1 or 3 bits wide (selectable)
Data Polarity	Normal, inverted or automatic
Major Frame Sync	FCC (FAC), SFID or URC
URC Location	Any 32 bit window within the first minor frame not including the last bit in the minor frame
SFID Location	Any series of contiguous bits not including the last bit in the minor frame
System Output	Buffered output with status, time, & data. Buffer size up to 64K words.

Table 1–2. Time Reader Specifications	
Time Reader Input Format	IRIG A, B, or G
Input signal level	1 v p-p nominal
Data Outputs	Automatic time tags for PCM data blocks Time accessible in register space

Table 1–3. Mechanical Specifications	
Form Factors	7.55" long "Desktop" PCI (2.2 M33, D32)
Power Dissipation	<i>tbd</i>

Table 1–4. General PCM Simulator Specifications	
Outputs	PCM Data, symbol-rate clock & minor frame strobe. Slave Clock out for sharing asynchronous embedded formats with a slave simulator.
Output Levels (Logic)	Single-ended TTL & RS-422
Baseband Output	Software-controlled adjustable <200mv to 8V p-p. Standard pre-modulation filters: 5-pole 0.25, 0.5, 1, 3, 6, 8, 12, 15MHz
Output Data Rate	64 bps to 33 Mbps (NRZ codes) 64 bps to 15 Mbps (all codes) Internal or external clock
Data Rate Stability	We use the best crystal oscillator \$2 can buy.
PCM Codes	NRZ-L/M/S; Bi-Phase -L/M/S; DM-M/S; M ² , RNRZ-L-11/15, k=7 Convolutional Rate 1/2, 1/3
Word Length	Variable from 3 to 16 bits per word on a word-by-word basis
CRC Generator	CRC16/CCITT forward/reverse
Frame Sync Pattern	Up to 256 words (any series of 0s or 1s may be used)
Major Frame Sync	FCC (FAC), SFID
Master/Slave	TTL-level interfaces available to serve as a master or slave simulator for asynchronous embedded PCM format simulation.
PRN Output	Output may be pre-empted by a PRN pattern with forced error for link bit error rate (BER) tests.
Time Generator Output	IRIG A, B, or G

Table 1–5. LS50 PCM Simulator Specifications	
Minor Frame Length	2 to 16,384 words per minor frame
Major Frame Length	Up to 1024 minor frames per major frame
Bit Order	MSB or LSB-first
Frame Sync Pattern	Up to 256 words (any series of 0s or 1s may be used)
Major Frame Sync	FCC (FAC), SFID
Common Words	May be a single value or selected from a group of one minor frame. Data may be changed while operating.
Unique Words	Seven may be programmed in any mainframe, super-commutated, or subcommutated channel. Data may be changed while operating.
Waveform Words	Five may be programmed to appear in every frame at the same location. Data may be changed while operating.

Table 1–6. LS70 PCM Simulator Specifications	
Minor Frame Length	1 to 65,535 words per minor frame
Major Frame Length	Up to 131,071 words or 1024 minor frames per major frame
Bit Order	MSB or LSB-first, word-by-word,
Frame Sync Pattern	Unlimited. Normal or FAC.
Major Frame Sync	FCC, SFID
Common Words	Word by word. Data may be changed synchronously or asynchronously while operating.
Unique Words	Seven may be programmed in any mainframe, super-commutated, or subcommutated channel. Data may be changed while operating.
Waveforms	Eight waveform fundamental periods available, major frame rate (one point/frame up to 1024 points), and seven others (256 points) frame or supercommutated, adjustable from 256 μ s to 15 seconds in steps of 256 μ s. Multiple wave tables may be tied to each period, limited by available memory. Data may be changed while operating.
PRN Data	Seven PRN pattern generator outputs may be inserted into specified word locations. Each generator can produce an 11, 15, 17, 19, 21, 23, 25-bit forward or reverse PRN sequence.

Table 1–7. Environmental Specifications	
Temperature (Operating)	0 to 50 °C
Temperature (Non-Operating)	-25 to +70 °C
Humidity (Operating)	10% to 90% Non-Condensing
Humidity (Non-Op)	Packaging must prevent contact with moisture and contaminants
Special Handling	Standard ESD methods required

2 — Installation

2.1 Addressing

The board occupies both PCI I/O space and memory space. *No address switch is used; the address is determined by the system.* 128 bytes of I/O space are always occupied. The card will respond to any access in its I/O space. The first 64 bytes of that space are assigned to the main ("Ch 0") channel, and accesses to the first byte will return an ASCII identifier string. If a second channel ("Ch 1") is configured, it will respond with its own identifier string 64 bytes up from the base address of the card.

The amount of memory space taken up by the card is circumstantial. If Ch 1 is configured, then twice as much space will be occupied and the upper half of that space will access Ch 1. The other factor is the memory addressing mode recognized by the buffer memory. PCI cards are normally shipped in a "flat" addressing mode wherein the 128Kbyte buffer memory is mapped one-to-one into PCI memory space. The configuration can be changed to activate a bankswitch register and map the selected bank into 16Kbytes of MS-DOS real memory space. Your system may not allow you to use this mode, but we use it here for testing purposes. In either case, if Ch 1 is present it will map either 128K or 16K higher in system memory space.

2.2 Physical Installation

The board can be installed in any physical slot where it fits. Remove and discard the blanking plate from the chosen slot (Save the screw!) and carefully insert the card.

2.3 Indicators

Nine LED indicators are provided. Three chip LEDs are board ID indicators. These are connected to a static register and are for use by device drivers in environments where multiple cards are present. Software for these boards is expected to light one or more of these to signify the boards have been detected and to identify which board is assigned to which data stream. Two more rows of three indicators are visible through the faceplate and used as status indicators. These indicators are sketched out in Figure 2-1. For a decommutator, Indicator 4 is a minor frame lock indication, Indicator 5 is a major frame lock indication and Indicator 6 lights when the IRIG time reader detects a valid IRIG time carrier. For an LS70 simulator configuration,

Indicator 4 is RUN status, Indicator 5 is the SW signal (its meaning is a function of an RF transmitter) and Indicator 6 is the RF output enable.

If Ch 1 is configured, Indicators 7 through 9 are work in the same manner as indicators 4 through 6, reflecting the status of Ch 1. These indicators are shared with the daughtercard. Indicator 7 is a signal present indication. Indicator 8 is a bit synchronizer lock indication. For LS40 modules, indicator 9 lights if the estimated E_b/N_0 exceeds 5dB. If Ch 1 and the daughtercard interface are both in use, the indications are wire-ored together.

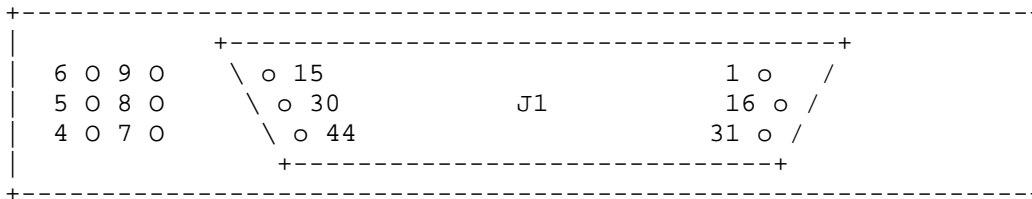


Figure 2–1. Front Plate.

2.4 J1 Interface

The board uses a 44-position female high density subminiature “D” type connector J1 for I/O. This connector has three rows of pins. Pin assignments are shown in Table 2–3. The existence of far more signals than there are pins unavoidably creates much pin-sharing complexity here. The first row of pins (1..15) are reserved for Ch 0. In the second row, pins 16 through 29 are grounds; pin 30 is a TTL-compatible 1PPS output from the IRIG time reader. The third row of pins are connected to the center row of patch array E1 adjacent to the connector. E1 (See Table 2–2) allows these pins to be dedicated to either a daughtercard module or to Ch 1 by installing a 2mm shunt patch.

An 8-position switch SW1 adjacent to the faceplate also affects the pinout. The actions of this switch are defined in Table 2–1. A four-position switch SW2 near the top board edge is applicable if, e. g., an LS40 Bit Synchronizer daughtercard is installed. It determines what termination, if any, is provided for the selected mezzanine input. (Unselected inputs are open.) The actions of this switch are defined in Table 2–4.

The LS38 module does not provide all the I/O capability of the LS40, so certain output signals are “manufactured” on-board. References to “LS38” in these tables apply only to boards that have been configured to host an LS38 module. Such boards also include an unreferenced SMA faceplate connector, used as the LS38 70MHz input. References to “LS40” apply only to boards hosting an LS40.

Table 2–1. Switch SW1 Definition	
SW1 Section	Definition
1	Off: J1-11 is Ch 0 Decom Slave Clock Out J1-13 is Ch 0 Decom Slave Data Out On: J1-11 is Ch 0 Sim RS422 Symbol Clock(-) J1-13 is Ch 0 Sim RS422 PCM Out(-)
2	Off: E1A-22 is Ch 1 Decom Slave Clock Out. E1A-26 is Ch 1 Decom Slave Data Out On: E1A-22 is Ch 1 Sim RS422 Symbol Clock(-) E1A-26 is Ch 1 Sim RS422 PCM Out(-) If used, E1A-22 is usually associated with J1-41 If used, E1A-26 is usually associated with J1-43
3	On: J1-36 is Ch 0 Sim External Baseband Input
4	On: J1-37 is Ch 1 Sim External Baseband Input
5	Off: J1-5 is Ch 0 Decom Aux Data In/ RS422 Status In(+) On: J1-5 is Ch 0 Sim Slave Clock Output
6	Off: E1A-10 is Ch 1 Decom Aux Data In/ RS422 Status In(+) On: E1A-10 is Ch 1 Sim Slave Clock Output If used, E1A-10 is usually associated with J1-35
7	Off: J1-15 is IRIG Time Reader Input On: J1-15 is also IRIG Time Generator Output
8	On: J1-15 is terminated into 100 ohms to ground.
LS70 configurations normally have sections 1, 2, 5, 6, 7 on, 8 off.	

Table 2–2. E1 Patch Definition					
E1A	Signal	E1A	Signal	E1B	Signal
26	See Table 2–1 SW1-2	25	J1-43	13	LS40 NRZ Out(-)
24	Ch 1 Sim PCM Out(+)	23	J1-42	12	LS40 Input 6
22	See Table 2–1 SW1-2	21	J1-41	11	LS40 Clock Out(-)
20	Ch 1 Sim Symbol Clock(+)	19	J1-40	10	LS40 Input 8
18	Ch 1 Sim Baseband Out	17	J1-39	9	LS40 Input 4
16	Ch 1 TTL Aux Clock In LS38 Tape Out	15	J1-38	8	LS40 Tape Out
14	Ch 1 Sim Slave Data In Ch 1 Sim TTL Ext Data In	13	J1-37	7	LS40 Input 3
12	Ch 1 Sim Ext Clock In Ch 1 RS422 Status In(-)	11	J1-36	6	LS40 Input 1
10	See Table 2–1 SW1-6	9	J1-35	5	LS40 Input 7
8	Ch 1 TTL Ext Sync In Ch 1 RS422 Data In(-)	7	J1-34	4	LS40 Input 5
6	Ch 1 TTL Ext Status In Ch 1 RS422 Data In(+)	5	J1-33	3	LS40 Lock Out LS38 Lock Out
4	Ch 1 TTL Data In Ch 1 RS422 Clock In(-)	3	J1-32	2	LS40 NRZ Out(+)
2	Ch 1 TTL Clock In Ch 1 RS422 Clock In(+)	1	J1-31	1	LS40 Clock Out(+)

Table 2-3. J1 I/O Connector Pinout			
Pin	Signal(s)	Pin	Signal(s)
1	Ch 0 TTL Clock In Ch 0 RS422 Clock In(+)	31	See Table 2-2 E1A-1 LS38 Clock Out(+)
2	Ch 0 TTL Data In Ch 0 RS422 Clock In(-)	32	See Table 2-2 E1A-3 LS38 NRZ Out(+)
3	Ch 0 TTL Ext Status In Ch 0 RS422 Data In(+)	33	See Table 2-2 E1A-5
4	Ch 0 TTL Ext Sync In Ch 0 RS422 Data In(-)	34	See Table 2-2 E1A-7
5	See Table 2-1 SW1-5	35	See Table 2-2 E1A-9
6	Ch 0 Sim External Clock In Ch 0 RS422 Status In(-)	36	See Table 2-2 E1A-11 See also Table 2-1 SW1-3
7	Ch 0 Sim Slave/Ext Data In	37	See Table 2-2 E1A-13 See also Table 2-1 SW1-4
8	Ch 0 TTL Aux Clock In	38	See Table 2-2 E1A-15
9	Ch 0 Sim Baseband Out (LS40 Input 2)	39	See Table 2-2 E1A-17
10	Ch 0 Sim Symbol Clock Out(+)	40	See Table 2-2 E1A-19
11	See Table 2-1 SW1-1	41	See Table 2-2 E1A-21 LS38 Clock Out(-)
12	Ch 0 Sim PCM Out(+)	42	See Table 2-2 E1A-23
13	See Table 2-1 SW1-1	43	See Table 2-2 E1A-25 LS38 NRZ Out(-)
14	Ch 0 Sim Scope Trigger Out.	44	Ch 1 Sim Scope Trigger Out.
15	IRIG Timecode. See Table 2-1 SW1-7, -8.	16-29	Ground.
		30	IRIG reader 1pps Out.

Table 2-4. Switch SW2 Definition	
Section	Definition
1	On: Selected input is terminated into 50 ohms to ground. Use for single-ended inputs only.
2	On: Selected input is terminated into 75 ohms to ground. Use for single-ended inputs only.
3	On: Selected inputs terminate into 120 ohms to each other. Use for differential inputs only.
4	Not used.

2.5 LS50 Parallel Output

The decommutator also provides a parallel output port. This output appears at connector J2. The pinout is shown in Table 2-5. If a Ch 1 decommutator is configured, it has its own parallel output on connector J7. Its pinout is identical to the J2 pinout.

Table 2–5. Parallel Output Pinout			
J2 (J7) Pin	Signal	J2 (J7) Pin	Signal
1	Ground	2	Ground
3	OD1	4	OD9
5	OD2	6	OD10
7	OD3	8	OD11
9	OD4	10	OD12
11	OD5	12	OD13
13	OD6	14	OD14
15	OD7	16	OD15
17	OD8	18	OD16
19	Ground	20	Ground
21	WdStb	22	Ground
23	FrmStb	24	Ground
25	MFSStb	26	Ground
27	Clock	28	Ground
29	1stBit	30	Ground
31	Lock	32	Ground
33	MFLock	34	Ground
35	Ground	36	Ground
37	Ground	38	Ground
39	Ground	40	Ground

2.6 RF Control Output

The V2 hardware platform provides for control of a proprietary external low-powered test transmitter using data from the PCM simulator. If Ch 1 is configured, the external module for it may be connected to J3. This is a 14-pin 53780-type Molex connector at the top board edge. The pinout is shown in Table 2–6. A module associated with the Ch 0 simulator is connected to a similar connector on a small paddleboard installed onto header J5. Transmit Control refers to RS232 signals intended to connect to a Quasonix transmitter.

Table 2–5. J3 RF Control Pinout			
Pin	Signal	Pin	Signal
1	PCM Data	8	Attenuator Control Serial Data
2	Ground	9	Attenuator Control Serial Clock
3	Symbol Clock	10	Ground
4	Ground	11	Attenuator Control Chip Select
5	Transmit Control Out	12	RF Mode Switch Control
6	Transmit Control In	13	Ground
7	Ground	14	RF Output Amplifier Control

3 — *Programming Information*

3.1 *General*

Authors of device drivers, API's, and telemetry applications will need to know what all the bits do. You are the audience. This chapter concludes with narratives meant as general guidance in converting a telemetry format definition into a download pattern for the board. Most of this setup can be done in any convenient manner. In those few cases where things are order-dependent we will note them.

3.2 *Locating a PCI Device*

PCI components do not have fixed address assignments. At system startup a power-on routine scans the computer for PCI interfaces and assigns system resources such as address space to them.

Each PCI component is assigned an array of sixty-four 32-bit registers in what is referred to as configuration space. This area is normally not accessible anywhere in system address space and must be accessed by special means that are system-dependent.

The following discussion applies to systems using *MS-DOS* or Microsoft *Windows 3/95/98* where PCI configuration space is accessed by BIOS calls. Other environments will have system-specific ways to get this information; you will have to consult your operating system documentation to find out how. To locate a receiver in your system...

1. Initialize an "*index*" value to zero. This index is allowed to grow as large as 255 by the PCI specification, but in practice you never get that far.

2. To locate PCI9056 chips, set machine registers:

```
AX = 0xB102
CX = 0x9056
DX = 0x10B5
SI = index
```

3. Issue a software interrupt 0x1A. If the system returns from interrupt with the carry flag set, any such devices are already located and no (more) exist. Skip out of your scanning routine. If the carry flag is clear, the BIOS call will have returned a "*handle*" in BX.

4. If the carry flag was clear, read the sub-identifier. Set registers:

AX = **0xB10A**
BX = *handle*
SI = **0x2C**

5. Issue another software interrupt 0x1A. The interrupt returns a value in ECX. If the value returned is 0x0500B00B (LS50) or 0x0700B00B (LS70) the handle points to us and other configuration registers may be accessed to obtain base addresses. Otherwise skip to step 7. Set registers as shown below.

Register numbers are:

Register 0x10 – PCI9056 Runtime Registers Memory Address.

Register 0x14 – PCI9056 Runtime Registers I/O Address.

Register 0x18 – Buffer Memory Address.

Register 0x1C – I/O Register Address.

Register 0x3C – (ISA-equivalent) IRQ Number.

AX = **0xB10A**
BX = *handle*
SI = *register number*

6. Issue yet another software interrupt 0x1A. The value returned in ECX is the register value. When reading the IRQ Number register, only the eight LSBs are important to you. They are the IRQ ("8259") number assigned to the PCI interrupt. If these bits are 0xFF, the system was unable to assign an interrupt for some reason. When reading addresses, logically AND the value returned in ECX with 0xFFFFFFF0. This yields the base address. If the LSB of ECX was a zero, the address is in memory space. If the bit was a one, the address is in I/O space. Reload AX, BX, and SI and repeat the call to obtain the necessary addresses. The PCI9056 runtime registers may be accessed via memory or I/O operations at your convenience.

Microsoft operating environments are notorious for erasing configuration registers of hardware they don't fancy. If the locating procedure places the buffer memory address at zero, that is what happened.

7. Increment the index value and try again.

The board may be configured to place the buffer memory in protected memory space ("flat mode") or in real space ("page mode.") In flat mode the buffer memory occupies 128 Kbytes of contiguous address space and the Bankswitch register is ignored. In page mode the buffer memory occupies 16 Kbytes of address space and three high-order on-board address bits are supplied by the Bankswitch register.

The board occupies 128 bytes of I/O space. Ch 0 uses the first 64 bytes. If Ch 1 is configured, it will return an identifier string from the second identifier register, 0x40 bytes away, and its memory will appear 128 Kbytes (flat) or 16 Kbytes (page) above the board base memory address.

3.3 Register Summaries

Registers appear at the I/O address obtained by adding the (hexadecimal) register number to the I/O register address. For this document, register bit assignments are summarized in tables associated with main functional divisions. In many cases read and write bit assignments for the same register are different. Also note there are several sets of indirect addresses associated with register accesses. Bits defined with a – dash are meaningless. Register assignments for Ch 1 start 0x40 bytes higher in I/O space.

3.4 General Registers

The Board ID and Identifier registers are basic to the board and not to any particular section.

Table 3–1. General Write Register Summary

Register	#	7	6	5	4	3	2	1	0
Board ID	20	–	–	–	–	–	LED3	LED2	LED1

Table 3–2. General Read Register Summary

Register	#	7	6	5	4	3	2	1	0
Ch 0 Identifier	00	0	"LS50" or "LS70SIM"						
Ch 1 Identifier	40	0	"LS50" or "LS70SIM" or "NOTHING"						

3.4.1 Board ID Register

This register setting has no effect on the operation of the board. It controls only the state of indicators 1..3. If there are multiple instances of the same PCI device, there is no way to tell which is which. You can use this register as you see fit.

3.4.2 Identifier Register

When read repeatedly, this register returns a null-bounded ASCII string. For Lumistar decommutators it returns the string "LS50" to identify the board. If Ch 1 is configured, it will return its own identifier at what would be register 0x40. Otherwise reads from register 0x40 will return "NOTHING"

3.5 LS50 Decommutator Registers

Table 3–3. LS50 Decom Write Register Summary

Register	#	7	6	5	4	3	2	1	0
Source Control	00	CkPol	SOURCE			Force	Rev	CRC	CCIT
FSP Write (RS=0)	01	–	–	–	–	–	–	Mask	!FSP
URC Write (RS=1)	01	–	–	–	–	–	–	Mask	URC
FSP Threshold (RS=0)	02	–	Threshold Value						
URC Threshold (RS=1)	02	–	–	Threshold Value					
Polarity Control	03	Polarity Xtol		FAC	Trail	FSP Tolerance			
Fmt Mem Lo (RUN=0)	04	LSBF	MASK	Sfwd	Lcwd	WL (Word Length-1)			
Mezzanine PCM Decoder (Run=1, RS=0)		PCM Output Code				PCM Input Code			
General PCM Decoder (Run=1, RS=1)	04	–	–	–	–	PCM Input Code			
Fmt Mem Hi (RUN=0)	05	–	–	–	–	Spare (tbd)		CRC	PASS
BERT Pattern (RUN=1)	05	–	–	–	–	REV	PATTERN		
Fmt Mem Addr Lo	06	Address [7..0] (LSB is RS bit)							
Fmt Mem Addr Hi	07	Address [15..8]							
Control (CFG0 = 0)	08	RUN	Wobl	Wind	RA	Burst	Gmod	VFL	2T15
Control (CFG0 = 1)	08	RUN	WINDOW		RA	Burst	Gmod	VFL	2T15
SfSync Position (RS=0)	09	SFW				SFB			
SfSync Control (RS=1)	09	Maj Fr Mode		Slsbf	SFUP	LastFr [9..8]		1stFr [9..8]	
First Frame (RS=0)	0A	First Minor Frame Number [7..0]							
Last Frame (RS=1)	0A	Last Minor Frame Number [7..0]							
Buffer Block Count	0B	Minor Frames/Block (MAJOR=0)							
Bankswitch (pagemode)	0C	–	–	–	–	–	PAGE		
Buffer Control	0D	IENB	AD13	Major	Frnch	NOEL	–	CLRS	CLRD

Table 3–4. LS50 Decom Read Register Summary

Register	#	7	6	5	4	3	2	1	0
Error Count Lo	01	Error Counter [7..0]							
Error Count Mid	02	Error Counter [15..8]							
Error Count Hi	03	OOS	Woos	Ecovf	–	Error Counter [19..16]			
Fmt Mem Lo (RUN=0)	04	LSBF	MASK	Sfwd	Lcwd	WL (Word Length-1)			
Clk Count Stat (RUN=1)	04	Update	Ovflo	–	–	–	–	–	–
Fmt Mem Hi (RUN=0)	05	–	–	–	–	Spare (tbd)		CRC	PASS
Clk Count Lo (RUN=1)	05	Clock Counter [7..0]							
Clk Count Mid (RUN=1)	06	Clock Counter [15..8]							
Clk Count Hi (RUN=1)	07	Clock Counter [23..16]							
Status	08	Intrpt	POL	Xstat	Dead	Mlok	Msrc	Lock	Srch
Header	09	SLIP	Lock	Mlok	Extpin	Crcerr	CFG2	CFG1	CFG0
Buffer Size Lo	0A	Buffer Size [7..0]							
Buffer Size Hi	0B	Buffer Size [15..8]							
Bankswitch	0C	1	1	1	CFG4	CFG3	PAGE		
Buffer Control	0D	IENB	AD13	Major	Frnch	NOEL	DMA	SIRQ	DIRQ

3.5.1 The Control Register

The Control register has mode bits that affect the decommutator.

Table 3–5. Control Register		
Bit	Mnemonic	Description
0	2T15	Selects the pattern length for the BER synchronizer. See ¶3.11.
1	VFL	Allows a new frame to start whenever a minor frame sync pattern is detected. Setting this bit is recommended only if frames vary in length and the longest expected frame is longer than the shortest expected time between sync patterns. If the time between patterns is longer than the longest frame, you should use BURST instead. It's okay to set BURST and VFL at once, though.
2	GMODE	Normally the decommutator output stops when it loses minor frame lock. If this bit is set, the decommutator will continue to block incoming bits into "frames" and output them. If it detects a sync pattern while in this state, it will abort the "frame" it is on and start a new one. To be meaningful, the FRNCH bit in the Buffer Control register must also be set.
3	BURST	Set this bit if the incoming data consists of fixed-length frames separated by zero or more fill bits. The data in the frames will be output and the fill bits discarded. Do not set GMODE or FRNCH along with BURST. Note: The CRC checker is reset at the start of each minor frame if BURST is set.
4	RA	For words less than 16 bits, the decommutator parallel output and buffer memory data is left-aligned with trailing zero fill to expedite number system conversions. Set this bit to yield right-aligned data with leading zero fill (certain daughtercards that use the decommutator parallel output may not function properly if RA is set.)
5	WINDOW (CFG0 = 0)	If set, the decommutator will set the SLIP status and slide over to align with an incoming frame that is one bit too short or one bit too long for the format definition.
6	WOBBLE (CFG0 = 0)	If the format definition has a major frame structure using SFID mode that is more than two minor frames long, set, WOBBLE to speed up major frame synchronization.
6..5	WINDOW (CFG0 = 1)	Allows the decommutator to set the SLIP status and slide over to align with an incoming frame that is too short or too long for the format definition. Values: 00 ("1-Bit") Frames must be the right length. 01 ("3-Bit") Frame length may be one bit off. 10 ("5-Bit") Frame length may be zero to two bits off. 11 ("7-Bit") Frame length may be zero to three bits off.
7	RUN	Set to run data and access the clock counter. Cleared to access the format memory.

3.5.2 Selecting the Input Source

The decommutator has five sets of data and clock inputs. The SRC field in the Source Control register determines the selection. In most system environments this is more a configuration than a format parameter.

Bit	Mnemonic	Description
0	CCITT	Set for formats including a CCITT CRC checkword.
1	CRCEN	Set for formats including a CRC-16 or CCITT CRC checkword.
2	REVCRC	Set for reversed CRC's.
3	FORCE	Set for pseudotelemetric applications where the data stream does not include frame sync patterns, rather the first bit of the frame is defined by a pulse on the FORCE input line. Meaningful for sources 000 and 001.
6..4	SRC	Clock/Data input source selected from following: 000 – Primary TTL Clock/Data Input 001 – RS-422 Clock/Data Input 010 – Mezzanine Clock/Data Input (from LS40 or LS38.) 011 – Tertiary (from embedded format master) clock/data input 100 – On-board simulator clock/data input 101 – Reserved 110 – Reserved 111 – On-board simulator clock/data input
7	CLKPOL	Set for 180-degree input clock.

3.5.3 PCM Code Control

The decommutator incorporates two PCM decoders and one PCM encoder into its input path. These code-changers are all controlled by four-bit values (Table 3–7) that are **not** the same values used to control the simulator output code. If a Bi-Phase, Miller, or RZ code is selected, the input clock is treated as a twice-rate clock.

The "Mezzanine" decoder is connected in series with the Mezzanine input. This decoder drives the decommutator input when the mezzanine source is selected, and also the "Mezzanine" encoder. If the board is configured to host an LS38, the mezzanine encoder output pre-empts the simulator baseband output for Ch 1. Further, this output is fixed to yield square-sided data with an amplitude of approximately 2V p-p unloaded.

The General PCM decoder is driven by the decommutator source select and affects any selected input source.

Value	PCM Code	Value	PCM Code
0000	NRZ-L	1000	M ²
0001	NRZ-M	1001	M ² -S
0010	NRZ-S	1010	Inverted NRZ-L
0011	Bi-Phase-L	1011	Inverted Bi-Phase-L
0100	Bi-Phase-M	1100	RZ
0101	Bi-Phase-S	1101	Inverted RZ
0110	DM-M	1110	RNRZ11
0111	DM-S	1111	RNRZ15

3.5.4 The Frame Sync Pattern

PCM formats generally consist of strings of bits divided into words. A known group of these words is called a minor frame, whose boundaries are located by a frame sync pattern at one end or the other. Sync patterns are themselves strings of bits, usually carefully chosen to be easily recognizable by hardware. These patterns are often documented as numbers (They really aren't – in reality all the binary digits have equal weight!) Different patterns are used depending on the sync budget and perspectives of the entities who designed the format, but certain strings are used more often than any others. Also, in most PCM formats all or most of the words are the same length, and the pattern is usually chosen to be a multiple of that length. Hence, you will probably see a number from one of 0xEB90 or 0xFE6B2840 (8- or 16-bit words,) 0xEDE20 (10-bit words,) or 0xFAF320 (8-, 12-, or 16-bit words) but the decommutator can be programmed to use any pattern so long as it can be contained in 64 consecutive binary digits. Sometimes, too, the pattern may include "don't care" digits that are not part of the pattern, or may be offset from the frame boundary. The ARINC 573 Flight Data Recorder format, for example, starts its sync pattern two bits after the actual frame boundary, and uses those first two bits as a SFID count. Because these numbers are chosen for robust detection, the user may allow a "tolerance," meaning that any one or more bits can be wrong and still have the pattern be recognized.

Substitution of "digits" for "bits" in places is deliberate. Each digit ends up with three possible values. Treat the pattern as a string.

The decommutator always presumes minor frames start with "Word 1." Word 1 may be defined as coinciding with the beginning of sync (leading sync,) or as starting immediately after the end of sync (trailing sync.)

The pattern actually written to the decommutator must be extended to exactly 64 digits in length. To extend the pattern for leading sync, enough "don't care" digits must be appended after the last sync bit to make exactly 64 digits. For trailing sync, "don't care" digits must be prefixed before the first sync digit to make 64 digits. The Decommutator Low Address register must be set to 0x00 to access the Frame Sync Pattern and Tolerance registers. Starting with the first bit, write all 64 digits to the Frame Sync Pattern Register in sequence, translating by:

Zero: 0x03
One: 0x02
Don't Care: 0x00

While sending the pattern out, count the number of digits that are not "don't care." Subtract the tolerance value (the result must be greater than zero or the

format definition is nonsense) and write the result to the Frame Sync Threshold register.

Some means of transmission have inherent ambiguities that may result in the data at the decommutator input being inverted. Hence the decommutator can be programmed to accept patterns of either data polarity and automatically inverting the data if it is not locked to the data and an upside-down pattern is detected. This is called "Automatic Polarity" and you probably want to default to it unless the format has Frame Alternating Complement (FAC) or some other such weirdness to add confusion. This value is among the fields in the Polarity Control register.

Table 3–8. Polarity Control Register		
Bit	Mnemonic	Description
0..3	TOLERANCE	Maximum number of errors allowed in a valid frame sync pattern.
4	TRAIL	Set for trailing sync. Also set when the FORCE input is used.
5	FAC	Set for FAC or Frame Code Complement (FCC) formats. Causes true and inverted frame sync patterns to be treated equally.
7..6	POLARITY	Data polarity control selected from the following: 00 – Inverted. 10 – Automatic. 11 – True.

3.5.5 The Decommutator Format Memory

The decommutator uses a memory-intensive approach to a number of format parameters. The format memory holds an attribute word for each word in the minor frame that holds the word length and a number of flags associated with that word. To access the format memory, the Control register RUN bit must be cleared. Then to access the attribute word for format word number k , cleave $(k-1)$ into bytes and write them to the low and high halves of the Format Memory Address register.

The LSB of the address register is also used as an indirect address bit where register numbers are overloaded (noted as encountered herein.) This function is independent of RUN. The rest of the address register is relevant only if RUN is clear.

Please do two discrete single-byte writes for immunity to Big/Little-Endian issues on non-PC architectures.

Once the address has been written you can access that location through the read/write Format Memory registers

Again, please do two discrete single-byte accesses whether you are reading or writing.

When setting up a format with n words per minor frame, you must load the first n locations of the memory. The attributes for word 1 are written to location zero, the attributes for word 2 go to location 1, ... for word n (with the LCWD bit set) to location $n-1$. Finally, another copy of the attributes for word 1 must be written to location n . Each attribute word is formatted as shown in Table 3-9.

Bit	Mnemonic	Description
0..3	WL	The word length in bits, less 1.
4	LCWD	Set to identify last word in the minor frame.
5	SFWD	For SFID and URC formats, set to identify the word during which major frame correlation is to take place.
6	MASK	Setting this bit causes the word to be suppressed, i. e., not to appear at the output.
7	LSBF	Set for LSB-first word assembly. Clear for MSB-first.
8	PASS	In the decommutator processing the outer format of a data stream with an embedded asynchronous format, set this bit to identify the words belonging to the embedded format.
9	CRC	Set to identify word where a CRC checkword begins.
11..15		Not used.

3.5.6 Major Frame Synchronization

Many formats define structures consisting of groups of consecutively numbered minor frames. Such a structure is called a major frame. The content of the minor frames differs from one to the next so one needs to know which is which. The decommutator has a ten-bit frame counter to identify consecutively numbered frames which appears in the frame header at the output. Such formats include ways to synchronize this counter to the larger structure.

The simplest technique, of course, is simply not to have a major frame structure; all frames are created equal. If there is no major frame structure, the SFWD bit is not set for any location. The major frame lock status has no meaning and should be ignored.

The most common technique is called SubFrame IDentification (SFID.) In this method a word (or part of a word) is reserved in a fixed location in the minor frame. That field has a count that increments (or decrements) from one frame to the next, starting at a known value and ending at some other known value and immediately restarting again.

More rarely encountered is Frame Code Complement (FCC.) In this method there is no defined count field in the data. The first frame in each major frame has its frame sync pattern inverted with respect to the others. This technique has the advantage that no overhead bits are needed for major frame

synchronization, with the corresponding disadvantages that the decommutator can correlate to the major frame structure only once per major frame, and a data polarity ambiguity is introduced by the inverted sync pattern.

Most rarely used is Unique Recycling Code (URC.) This method has a field within the minor frame like SFID, but instead of an incrementing count, the field has a known value that is alleged to appear only once per major frame. This technique manages to combine some of the disadvantages of both the other techniques. Based on experience, the probability of your encountering a URC format is much less than 1%.

Setting the decommutator to synchronize to a major frame includes loading several registers and (usually) setting the SFWD attribute bit (Table 3–9) in the proper format memory location.

Caveat: The major frame synchronizer may not work properly if a SFID or URC field ends on the minor frame boundary.

3.5.6.1 SFID Correlation

If the format contains a SFID count, the SFWD bit must be set in the format memory location that corresponds to the word where the count field ends (Usually the same word where it begins; the decommutator allows the count to cross a word boundary, but in practice this almost never happens.)

Write the eight LSBs of the SFID count start value to the First Frame register (Write 0x00 to the address register first!)

Write the eight LSBs of the SFID count ending value to the Last Frame register (Write 0x01 to the address register first!)

You need to calculate two values for the SFID Position register. The SFW is the length of the SFID count. This is one less than the number of bits needed contain the largest value the SFID count can legally have., e. g., if the count spans the range [0..63] the SFW value is 5. The SFB value locates the count field in the SFWD word. This value is calculated by one of the methods below.

Read carefully! Experience shows this is the area most prone to error in setup development.

If the SFID word is transmitted MSB-first, SFB is 15 less the number of bits separating the LSB of the SFID count and the LSB of the SFWD word, i. e., 15 in the usual case where the count is right-aligned.

If the SFID word is transmitted LSB-first, SFB is 15 less the number of bits separating the MSB of the SFID count and the MSB of the SFWD word.

Shift SFW four bits to the left, add SFB, and write the result to the SFID position register (Write 0x00 to the address register first!)

Calculate and write the Major Frame Sync Control register value as shown in Table 3–10 (Write 0x01 to the address register first!)

3.5.6.2 FCC Correlation

For FCC correlation, the SFWD bit is not set anywhere in the format memory. The starting and ending frame count values are set as for SFID mode. Set the Major Frame Sync Control register value as shown in Table 3–10 (Write 0x01 to the address register first!) You must also set the FAC bit in the Polarity Control register (Table 3–8.)

3.5.6.3 URC Correlation

A URC format will have a URC pattern value associated with it. Like a frame sync pattern, a URC pattern consists of a string of one, zero, and “don’t care” digits, and is loaded much the same way as a *trailing* frame sync pattern is loaded. Enough “don’t care” digits are prefixed onto the front to make at least 32 digits. The Decommutator Low Address register must be set to 0x01 to access the URC Sync Pattern and Tolerance registers. Starting with the first bit, write all 32 digits to the URC Sync Pattern Register in sequence, translating by:

Zero: 0x02
One: 0x03
Don't Care: 0x00

(Not quite same translation as for frame sync patterns!) While sending the pattern out, count the number of digits that are not “don’t care.” Subtract the tolerance value (the result must be greater than zero or the format definition is nonsense) and write the result to the URC Threshold register.

Set the SFWD format memory bit for the location where the URC pattern ends.

Set the first and last frame values as for SFID mode. You need to calculate and set an SFB value (SFW is meaningless) using the same sort of calculation as for SFID, except the bit reference is to the last (youngest) bit of the URC pattern, whether the word is LSB- or MSB-first, and set the Major Frame Sync Control register value as shown in Table 3–10 (Write 0x01 to the address register first!)

Bit	Mnemonic	Description
1..0		Bits [9..8] of the first frame value.
3..2		Bits [9..8] of the last frame value.
4	SFUP	Set if the frame count increments from one minor frame to the next. Clear if it decrements.
5	SLSBF	Set if a SFID count is present and is transmitted LSB-first.
7..6	SFMODE	Major frame synchronizer mode: 00 – SFID. 01 – FCC. 10 – URC.

3.5.7 The Decommutator Output

The decommutator output is a stream of words from the input data, with a header prefixed to the beginning of each minor frame. This data is grouped into “blocks” of one or minor frames each and written to on-board buffer memory. Two such memories are provided. Normally while the decommutator writes to one memory, the other is accessible for your use. When a block’s worth of data has been written, an interrupt is generated and the two memories are logically switched so the fresh data is now available. You can directly access that memory through the system bus, or you can use one of the PLX PCI9056 DMA controllers to move that data into your own buffers in system memory.

The header preceding each minor frame consists of four words of BCD timestamp and one word of decommutator status information, as shown in Table 3–11.

The decommutator has several registers associated with the buffer memory.

The Buffer Control and Status Register (Table 3–12) sets the operating mode of the buffer memory and manages interrupts. The fields in this register are in two groups. The five MSBs are control bits. Reads from the register return the value written. The LSBs are interrupt flags. Interrupts come from three different sources. When the decommutator system interrupts, at least one of these bits is set. These bits deliberately have the peculiar behavior that writing ones to them clears them. As part of your interrupt acknowledge ritual you must read this register and then immediately rewrite the value read back to it to release the interrupt.

When the memory is in page mode, only 16Kbytes of the memory are directly accessible. The 3 LSBs of the Bankswitch register select a page within the 128Kbyte memory. When the memory is in flat mode, this register is ignored.

Table 3–11. Frame Header																
Wd	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	10's Days				1's Days				0	0	Fly	Err	100's Days			
1	10's Minutes				1's Minutes				10's Hours				1's Hours			
2	100's ms				10's ms				10's Seconds				1's Seconds			
3	10's μ s				1's μ s				1's ms				100's μ s			
4	Slip	Lock	MLOCK	Ext	Crc	0	Minor Frame Number									
Mnemonics in the Frame Header Table																
FLYWHEEL		The IRIG time reader is in flywheel mode and the time carrier was lost for at least one cycle in the last time frame.														
ERROR		Seconds in the last IRIG time frame disagreed with the internal seconds count in the time reader.														
SLIP		WINDOW is non-zero in the decommutator and the preceding frame was too long or too short.														
LOCK		Decommutator minor frame Lock state.														
MLOCK		Decommutator major frame Lock state.														
EXTPIN		The instantaneous state of the Status input signal associated with Source 000 (regardless of what input source is selected).														
CRCERR		The most recent CRC check failed.														

The Buffer Block Count register is used to set the number of minor frames (1..256) are gathered in a data block. Set this register to the desired number of frames, less 1. The register value is ignored if the buffer control register MAJOR bit is set, however. In any case, **you** must ensure the defined data block fits into 64K words; if the buffer memory controller runs off the end of the memory it wraps around and starts to overwrite data at the beginning. The results are not pretty.

The Buffer Size register is primarily meant for VFL applications. At each buffer turn, it is set to the number of words in the present buffer.

3.5.8 Status

The decommutator returns status in several registers. The main status return is the Status register (Table 3–13.)

The instantaneous state of the status signals that will be written as the next frame status word in buffer memory can be read from the Header register (Table 3–14.) These bits can change asynchronously and this register is primarily for maintenance purposes.

The Bankswitch register can also be read. The three Bankswitch bits return the value written. The five MSBs are named CFG[7..3]. These bits are version and option information. For the V2 platform they are generally “canned.”

Table 3–12. Buffer Control and Status Register

Bit	Mnemonic	Description
0	DINT CLRINT	When read as a 1, the decommutator has generated an end-of-block interrupt since the last interrupt acknowledge. Writing a 1 to this bit clears it. Writing a zero has no effect.
1	SINT CLRSINT	When read as a 1, the simulator has generated an end-of-frame interrupt since the last interrupt acknowledge. Writing a 1 to this bit clears it. Writing a zero has no effect.
2	DMAINT	When read as a 1, the PCI9056 DMA controller has generated an end-of-transfer interrupt since the last interrupt acknowledge. This bit is read-only. You must clear the interrupt condition at the DMA controller. Note that this bit will be set only on the Ch 0 decommutator. If Ch 1 is using the other PCI9056 DMA channel, this bit will still be set in this register for the Ch 0.
3	NOEL	When set, data is blocked according to the block count, but if a major frame boundary occurs, the current buffer is terminated and a new one started. We suggest you set this bit only if the major frame length is a multiple of the block count, or the data blocks come out different lengths, some of which will have frames of stale data at the end.
4	FRNCH	Set to allow interrupts when the decommutator is not locked. You must set this bit to access the "frames" that can occur if GMODE is set in the decommutator control register. Do not set this if BURST is set unless VFL is also set.
5	MAJOR	When set, the block count is ignored and data blocks are aligned with major frame boundaries.
6	AD13	This bit is primarily for maintenance purposes. When set, the decommutator itself and the system bus are connected to the same memory. It is possible to use this bit to access a block that was partially filled when the test vehicle smashed into something hard and the decommutator lost lock.
7	IENB	Decommutator end-of-block interrupts are allowed only if set.

Table 3–13. Status Register

Bit	Mnemonic	Description
0..1		Minor Frame Correlator Status: 00 – Verify 01 – Search 10 – Lock
2..3		Major Frame Correlator Status: 00 – Verify 01 – Search 10 – Lock
4	DEAD	Set if the input clock stops or drops below a rate of (tbd) bps.
5	XSTAT	Returns the signal level of the Status input if one is associated with the selected input source. Otherwise meaningless.
6	POL	Set if the data at the decommutator input is being inverted, either automatically or under program control.
7	INTRPT	Set if a data block has ended, whether decommutator interrupts are enabled or not. Reading the Buffer Control register clears this bit.

Table 3–14. Header Register		
Bit	Mnemonic	Description
0	CFG0	Set for hardware revisions that recognize wider sync windows.
2..1		Meaningless.
3	CRCERR	Set if most recent CRC check failed.
4	EXTPIN	Similar to XSTAT except this bit always returns the state of the status line for Source 0, regardless of the selected input source.
5	SLOCK	Major Frame Lock state.
6	LOCK	Minor Frame Lock state.
7	SLIP	WINDOW is set in the decommutator and the preceding frame was too long or too short.

The decommutator monitors the incoming bit rate by counting clocks at the selected input and registering the count every second. If RUN is set, you can learn the incoming rate by polling UPD (bit 7 of the Clock Count Status register.) When it comes on, read the three Clock Count registers and concatenate their values.

Reading the MSBs of the Clock Count clears the UPD and OVF bits.

The OVF flag (bit 6 of the Clock Count Status register) can be treated as a 25th bit, allowing a range up to 33 MHz.

Editorial: If the decommutator is in minor frame lock, then the clock count value is not very interesting, because whatever it reads is probably the right value. Nevertheless, if the decommutator is *not* in lock, then the clock count may be cogent. So if you bother to display status, you may as well also display the count.

3.6 The IRIG Time Reader

An IRIG time reader is juxtaposed with the decommutator. This reader can either be synchronized with an IRIG time carrier, or be seeded with local time and used as a free-running clock. The time reader is primarily used to provide timestamps for incoming data, but you can also read the time directly back.

Table 3–15. IRIG Reader Write Register Summary									
Register	#	7	6	5	4	3	2	1	0
Control	18	–	–	Arrow		–	Flywhl	MODE	
RTC Setting	19	" ^AAdddhmms^W "							
Freeze Command	1A	–	–	–	–	–	–	–	–

Register	#	7	6	5	4	3	2	1	0
BCD Time Return at last Freeze Command (except BUSY)	18	10's μ s				1's μ s			
	19	1's ms				100's μ s			
	1A	100's ms				10's ms			
	1B	0	10's Seconds			1's Seconds			
	1C	0	1's Minutes			1's Minutes			
	1D	0	0	10's Hours		1's Hours			
	1E	10's Days				1's Days			
	1F	BUSY	0	FLY	ERR	100's Days			

Operating modes for the IRIG reader are set in the IRIG reader Control register (Table 3–17.)

3.6.1 Setting the Real Time Clock

The real time clock free runs at the rate controlled by the ARROW value. To set the time, you need to put the reader in Real Time Clock mode and convert date and time of year to an ASCII string:

`"^AAdddhmmss^W"`

where ^A is 0x01, ddd is a zero-extended day number [001..366], hh is a zero-extended hour number [00..23], etc., and ^W is 0x17. Write the characters of this string in sequence to the RTC Setting register. After each write, poll and wait for the BUSY flag (bit 7 of register 0x1F) to clear (it only takes a few hundred nanoseconds!) before continuing. While you are loading the time, the reader output is held, fractional seconds are cleared, and the respective time digits appear as you load them. When you send the last character the clock starts to run.

Bit	Mnemonic	Description
1..0	IRIGMODE	Reader mode and carrier select: 00 – Real time clock. Any incoming time carrier is ignored. 01 – IRIG B. 10 – IRIG A. 11 – IRIG G.
2	FLYWHEEL	Set to allow reader time to flywheel during time carrier dropouts. Must be cleared if the time carrier is not running at the selected rate.
3		Meaningless.
5..4	ARROW	Specifies the length of the arrow of time in RTC or carrier flywheel: 00 – Real time 01 – Time at half rate. 10 – Time at twice rate.
7..6		Meaningless.

3.6.2 Reading Time

The main purpose of the time reader is to provide timestamps for data. However, you can read time directly from the reader into the system without disturbing that operation. To read time, you must first capture it by writing (anything) to the Freeze Command register. You can then read BCD time of year in microseconds by reading registers as shown in Table 3–16.

3.7 The LS50 PCM Simulator

The PCM simulator can be used to generate a test data stream. This simulator outputs a primarily static data stream and is not intended for such purposes as archival playback or uplink command generation, albeit we have seen this sort of simulator misused successfully for such ilk.

Table 3–18. LS50 Simulator Write Register Summary

Register	#	7	6	5	4	3	2	1	0
Command	10	MREQ	Mread	IACK	IENB	RStrt	XCLK	PAGE	–
Bankswitch	11	0	0	0	0	REGS	MBF	PAGE	MB1
Low Address	12	Mailbox/Exchange Address [7..0]							
High Address	13	Mailbox/Exchange Address [15..8]							
Data Memory (Bs=00x1)	14	Word Attributes/Right-Aligned Data [7..0]							
	15	Word Attributes/Right-Aligned Data [15..8]							
Frame Attr Memory (Bs=01x0)	14	EOSF	UW6	UW5	UW4	UW3	UW2	UW1	UW0
Frame Start (Bs=1000, Address=0)	14	First Minor Frame Number [7..0]							
	15	1	FCC	FAC	SFUP	–	–	1stFr [9..8]	
Mode (Bs=1000, Address=1)	14	0	ClkDiv		CWS	LSBF	–	CRC	CCIT
	15	WIDE	Uplink	–	BCRC	Event	BERT	ERR	2T15
NCO Setup (Bs=1000, Address=2)	14	NCO Control Bits							
Encoder Control (Bs=1000, Address=3)	14	QUIET	Slave	RNRZ Control		PCM Code			
	15	–	–	–	DIFF	INV	Swap	1/3	RATE
External Register (Bs=1000, Address=4)	14	Reserved for future use							
BERT Pattern (Bs=1000, Address=17)	14	–	–	–	–	REV	PATTERN		
RS232 Data	21	(ASCII Character)							
RF Control	22	XDAT	RFEN	SW	Xmod	PMF			
RS232 Baud Rate	23	8 LSBs of –96 / (BaudRate / 1200)							
RF Command	24	EEPROMCMD		LO	–	DAC	DEV	Addr[9..8]	
RF EEPROM Address	25	EEPROM Addr [7..0]							
Low RF Data	26	EEPROM or DAC Data [7..0]							
High RF Data	27	EEPROM or DAC Data [15..8]							

When setting up a PCM simulator for a given format, the same issues of sync pattern and format “shape” arise as when setting up a decommutator. The decommutator need not provide data content; that is obviously a product of its

environment. A simulator needs to provide data content; it needs to output *something* for every position in the format.

A simulator also needs to provide something else a decommutator gets from its environment, a data rate clock. That is why, after describing the Command and Mode registers that control the simulator, the following paragraphs start at the end of things before jumping back to the beginning.

Table 3–19. LS50 Simulator Read Register Summary

Register	#	7	6	5	4	3	2	1	0
Command	10	MREQ	Mread	Intrpt	IENB	0	XCLK	PAGE	Pgif
Not defined	11	–	–	–	–	–	–	–	–
Not defined	12	–	–	–	–	–	–	–	–
Not defined	13	–	–	–	–	–	–	–	–
Memory Mailbox (Bs = 0xxx)	14	Defined Same As Write							
	15								
RS232 Data	21	(ASCII Character)							
RF Control	22	Defined Same As Write							
RS232 Status	23	–	–	–	–	–	–	XBE	RBF
RF Status	24	–	EEOP	LO	–	DAC	–	–	–
Not Used	25	–	–	–	–	–	–	–	–
Low EEPROM Return	26	EEPROM Data [7..0]							
High EEPROM Return	27	EEPROM Data [15..8]							

3.7.1 Simulator Command Register and Mode Registers

The simulator Command register has a variety of bits that want quick access, so this is the one simulator register that is directly accessed. Other operational registers are indirect addressed in an effort to fit the simulator into a limited amount of I/O space. The Command register is laid out as shown in Table 3–20. Note the register is read/write but some of the bits have subtly different but related meanings for write and read operations – purposely so to allow for using read-modify-write-type accesses sensible under a variety of conditions. The Mode register (Table 3–21) and Frame Start Register (Table 3–22) are used to set static operating modes for the simulator. These registers are indirect addressed, so access to them is slower, but their contents are not likely to change except when you perform a complete simulator setup.

To access the Mode register, write 0x08 to the simulator Bankswitch register (setting only the REGS bit), and write 0x01 to the Low Address register.

To access the Frame Start register, write 0x08 to the simulator Bankswitch register (setting only the REGS bit), and write 0x00 to the Low Address register.

Table 3–20. Simulator Command Register		
Bit	Mnemonic	Description
0	PAGE IN EFFECT	Has no meaning when written. When read, returns the state of the simulator internal PAGE flag. This flag is copied from the PAGE bit (bit 1 of this register) on each minor frame boundary and determines which page of the simulator memory is to be used during the next frame.
1	PAGE	The simulator memory is divided into two equivalent pages. This bit specifies which page to use. If UPLINK is not set, PAGE can be used to synchronously switch the simulator between two formats. If the bit is unchanged, the same page is used over and over again unless... If UPLINK is set, the format defined in page 0 is output repeatedly until PAGE is set. The <i>next</i> minor frame is in the format defined in page 1 and is output <i>once</i> and PAGE is cleared by the simulator. When read, returns the last value written.
2	XCLK	When set the simulator clock generator is ignored and the simulator clock is to be supplied from the simulator external clock input. Set this bit if external clocking is desired. If this simulator is the slave of a simulator-pair generating an asynchronous embedded format, the simulator external clock must be connected to the slave clock output of the master simulator and this bit must be set to run data. Conversely, this bit should be cleared during a simulator setup to ensure the simulator clock is being allowed to run. When read, returns the last value written.
3	RESTART	Writing a one clears the simulator word and frame counters and aborts from any simulator memory access in progress. Writing a zero has no effect. When read, always returns zero.
4	IENB	Setting this bit causes the simulator to generate a system interrupt each time the INTRPT bit is set. When read, returns the last value written.
5	IACK INTRPT	When read, returns the state of the simulator interrupt flag. This flag is set on every minor frame boundary, whether interrupts are enabled or not. Writing a one to this bit clears the interrupt flag. Writing a zero has no effect.
6	MREAD	Controls direction of transfer between the simulator memory and exchange registers for memory accesses. Set for reads, clear for writes. When read, returns the last value written.
7	MREQ	Setting this bit initiates a simulator memory access. Writing a zero has no effect. When read, returns a one if an access is in progress and not completed yet.

Table 3–21. Simulator Mode Register		
Bit	Mnemonic	Description
0	CCITT	When set, causes a CRC-CCITT checkword to be calculated. Otherwise CRC-16 is calculated. Has no meaning if CRCEN is clear or if no CRC location is specified in the simulator word attributes.
1	CRCEN	When set, causes a CRC checkword to be calculated. Has no meaning if no CRC location is specified in the simulator word attributes.
2	REVCRC	When set, causes a reversed CRC checkword to be calculated. Has no meaning if CRCEN is clear.
3	LSBF	When set, all simulator data is output LSB-first.
4	CWS	When cleared, simulator common output data is read from the common data area in simulator memory. This means all words not pre-empted by sync, unique, or waveform words. When set, the common data area is ignored and all common output words have the value of the simulator CWS memory location.
6..5	DIV	Selects a prescale ratio for the simulator clock: Choose one of: 00 – Divide by 1. 01 – Divide by 16. 10 – Divide by 256. 11 – Divide by 4096.
7	MREQ	Maintenance use only. Do not set this bit.
8	2T15	Specifies a 32,767-bit PRN pattern. See ¶3.12. This bit works but is redundant if CFG6 is set.
9	ERR	Forces one error every PRN pattern iteration. See ¶3.12.
10	BERT	Pre-empts simulator output with PRN pattern. Zero for normal operation. See ¶3.12.
11	EVENT	0-to-1 transition forces a single PRN pattern error. See ¶3.12.
12	BCRC	Normally the CRC generator is reset at the end of the checkword. Set this bit to cause the generator to be reset again at the end of the minor frame.
13	Meaningless	
14	UPLINK	When set, clears the PAGE bit if the PAGE IN EFFECT bit is set.
15	WIDE	Set this bit to cause the simulator frame strobe output to rise at the beginning of the last word in the minor frame. If not set, the frame strobe rises with the beginning of the last bit. The strobe always falls on the frame boundary.

Table 3–22. Simulator Frame Start Register		
Bit	Mnemonic	Description
9..0	1STFRAME	Frame number of the first minor frame in the major frame.
11..10		Meaningless.
12	SFUP	When set, causes minor frame numbers to increment in the course of the major frame.
13	FAC	When set, words with the FSP attribute are inverted during odd-numbered minor frames.
14	FCC	When set, words with the FSP attribute are inverted during the first minor frame of each major frame.
15	WDST	Always set this bit.

3.7.2 Output Formatting

Aside from a straight serial (NRZ-L) data stream and clock, the simulator has an additional output that is encoded by one of a set of standardized schemes used for telemetry transmission. At the output is a k=7 convolutional encoder, followed by a randomizer, followed by a PCM encoder. These encoders are set up by an Encoder Control register. This register is accessed by indirect addressing through the Exchange register. To write values to it, you must write 0x08 to the simulator Bankswitch register (setting only the REGS bit), and write 0x03 to the Low Address register. The upper and lower halves of the Encoder Control register (Table 3–23) can then be accessed by writing the upper and lower halves of the Exchange register.

One of the parameters associated with this register is the output PCM Code. There are a number of selections here. Each has an implied parameter called the Code Factor associated with it. This factor and others are used in the calculations to set up the simulator clock generator; to properly set the data rate the value written to this register must be known.

Table 3–23. Simulator Encoder Control Register

Bit	Mnemonic	Description
3..0	PCM CODE	PCM Output code. Choose one of the following: 0000 – NRZ-L 0001 – Inverted NRZ-L 0010 – NRZ-M 0011 – NRZ-S 0100 – RZ 0110 – Inverted RZ 1000 – Bi-Phase-L 1001 – Inverted Bi-Phase-L 1010 – Bi-Phase-M 1011 – Bi-Phase-S 1100 – DM-M 1101 – DM-S 1110 – M ² 1111 – M ² -S
5..4	RANDOMIZE	RNRZ Randomizer Control: 00 – Off 01 – RNRZ11 10 – RNRZ15
6	SLAVEN	When set, if Unique Word 6 is to appear at the output, it is preempted. The simulator slave output clock runs during this word and data from a slave simulator is inserted. This feature is for use in the master of a simulator-pair to create a simulated stream with an asynchronous embedded format. Meaningless for LS70 configuration.
7	QUIET	For maintenance use. Do not set this bit.
8	RATE	Enables the convolutional encoder. Causes output to be rate-1/2 encoded unless 1/3 is also set.
9	1/3	If RATE is set, causes output to be rate-1/3 encoded.
10	SWAP	Swaps the G1 and G2 symbol when set.
11	INVERT	Inverts the G1 symbol when set.
12	DIFF	Enables differential encoding when set.
15..13		Meaningless.

3.7.3 The Clock Generator

The simulator uses a Number Controlled Oscillator (NCO) to generate its output clock. Exercise the following algorithm to get the NCO operating.

If the logical AND of 0x0C and the value (chosen according to the output code) written to the Encoder Control register is not zero, multiply the desired output bit rate by 2. Otherwise multiply by 1.

If the RATE bit in the Encoder Control register is set, multiply the rate by 2, but if the 1/3 bit is also set, multiply the rate by 3.

Clamp the upper bound of the rate at 35,000,000. Neither the NCO nor the simulator are certified reliable beyond that point.

If the result is 262,144 or greater, the DIV field in the Mode register (Table 3–21) should be 00. Otherwise choose a DIV field and multiply the rate by the "by" factor to get larger than 262,144 if possible.

Multiply the rate by 35.791394. Truncate the result to an integer.

Write 0x08 to the simulator Bankswitch register (setting only the REGS bit), and write 0x02 to the Low Address register.

Write 4 to the low exchange register. Then write 2. Then write 0.

Repeat 32 times:

The value "x" is 0x80 if the LSB of the rate is 1, or 0x0 if it is zero. Write x to the low exchange register. Then write x+4. Then shift the rate one bit to the right, discarding the LSB.

(End repeat)

Set the rate value to 1. Then repeat 8 times:

The value "x" is 0x80 if the LSB of the rate is 1, or 0x0 if it is zero. Write x to the low exchange register. Then write x+4. Then shift the rate one bit to the right, discarding the LSB.

(End repeat)

Write 2 to the low exchange register. Then write 0.

3.7.4 Communicating With Simulator Memory

The simulator uses two separate memories during its operation. Each minor frame word location has an attribute word associated with, and a data value to be output. The attribute table and common value tables are each 16K (16-bit) words long. There are also tables of unique, sync, and waveform values.

These items are together in one 64K word memory. Additionally, there are two pages of this memory, making 128K words total.

Juxtaposed with that, there are two pages of simulator frame attributes in a separate memory. Each page is 1K (8-bit) words.

To access memory, the simulator clock must be running. If you will be performing a large number of accesses, write to the Command register clearing the XCLK bit and also the Mode register, setting the DIV field to 00. Set the simulator clock generator to some convenient rate.

Specify a word location by writing to the Low and High Address registers. Select which memory to access by writing to the Bankswitch register. Set **only one** of the MB1, MBF, or REGS bits.

Bit	Mnemonic	Description
0	MB1	Set to access data/word attribute memory.
1	MB0	Memory PAGE associated with memory accesses. Meaningless if neither MB1 nor MBF are set.
2	MBF	Set to access frame attribute memory.
3	REGS	Set to write indirectly-addressed simulator registers instead of memory.
7..4		Reserved. Do not set any of these bits.

If you are performing a memory write, write the data to the Low and High Exchange registers and then write to the Command register, clearing MREAD and setting MREQ. Repeatedly poll the Command register, waiting for MREQ to go away, which will take one to three clock times.

If you are performing a memory read, write to the Command register, setting MREAD and MREQ. Repeatedly poll the Command register, waiting for MREQ to go away, which will take one to three clock times. The returned data can then be read from the Exchange registers.

3.7.5 The Simulator Memory Map

The data/word attribute memory is mapped as shown in Table 3–25. Remember there are actually two such memories, selected by an additional PAGE address bit. When you access memory, use the MB0 bit in the Bankswitch register to select. When the simulator uses the memory operationally it will use its PAGE IN EFFECT bit. The frame attribute memory is strictly a lookup by minor frame number plus the PAGE bit.

Table 3–25. Simulator Memory Map	
Range	Definition
0x0000–0x3FFF	Common data values, lookup by word number mod 16,384 (CWS=0.)
0x4000–0x4006	Unique word values, lookup by unique word number.
0x4007	CWS data value (CWS=1.)
0x4008–0x43FF	Not used.
0x4400–0x47FF	Frame Sync Pattern data, lookup by word number modulo 1024.
0x4800–0x4BFF	SFID data, lookup by minor frame number.
0x4C00–0x4FFF	Waveform 1 data, lookup by minor frame number.
0x5000–0x53FF	Waveform 2 data, lookup by minor frame number.
0x5400–0x57FF	Waveform 3 data, lookup by minor frame number.
0x5800–0x5BFF	Waveform 4 data, lookup by minor frame number.
0x5C00–0x5FFF	Waveform 5 data, lookup by minor frame number.
0x6000–0x7FFF	Not used.
0x8000–0xBFFF	Word attributes, lookup by word number modulo 16,384.
0xC000–0xFFFF	Not used.

3.7.6 Attributes and Data

The simulator Frame Attribute Memory is loaded with frame attributes (believe it or not!) The Data and Word Attribute Memory holds both output data and word attributes.

Each minor frame word location has an associated attribute word. The attribute words are stored in data memory (See Table 3–25 for location.) The attribute word is formatted per Table 3–26.

Each minor frame in the major frame has an attribute word in the Frame Attribute memory, as shown in Table 3–27.

Data in the data areas is always right-aligned, regardless of word length or bit ordering. However, for LSB-first data, the frame sync words need to be bit-reversed to get the pattern to output properly. Also note the simulator allocates and integral number of words to the frame sync pattern, and an entire word to the SFID count, regardless of how many bits they actually use.

The simulator design provides support for formats using FCC or SFID major frame correlation. There is no provision for URC formats *per se*. To simulate a URC, you will need to pre-empt enough unique words to put the URC pattern in the first minor frame.

Table 3–26. Simulator Word Attributes		
Bit	Mnemonic	Description
5..0	WSPL n	If FSPL n ($n=0..5$) is set for the current frame, substitute contents of unique word n location for whatever other data would be output here.
6	WSPL6	If FSPL6 bit is set for the current frame, substitute the contents of unique word 6 location for whatever other data would be output here. ... BUT ... If SLAVEN is set in the Encoder Control register (Table 3–23), do not substitute the unique word. Instead, allow the slave clock output to run during this word and insert whatever appears at the slave data input.
7	CRCW	Output the CRC checkword, starting with the first bit of this word. The checkword output lasts for 16 bit periods, during which any other data otherwise defined for output is discarded.
11..8	WL	The word length in bits, less 1.
14..12	I	Data source for this word, unless overridden by CRC, slave, or unique word: 000 – Common data. 001 – Frame Sync Pattern data 010 – SFID data 011 – Waveform 1 data 100..111 – Waveform 2..5 data
15	EOF	Set to identify last word in minor frame.

Table 3–27. Simulator Frame Attributes		
Bit	Mnemonic	Description
6..0	FSPL n	If WSPL n set for the current word, substitute contents of unique word n location for whatever other data would be output here.
7	EOSF	Set to identify last minor frame in the major frame.

3.7.7 Baseband and RF Control

History has combined two functions that were more closely related in past versions of this hardware. The V2 hardware platform includes a serial EEPROM holding 1024 sixteen-bit words (Ch 1 includes another such EEPROM.) The EEPROM holds configuration data for the simulator (and RF output, if present.)

Multiple RF output options are provided for. At this writing the only implemented one uses a Quasonix low-powered RF transmitter module that provides FM and SOQPSK modulated signals. Controls for the baseband output and pre-modulation filters have no effect on this RF output.

3.7.7.1 EEPROM Access

A given EEPROM location is read by performing this sequence:

Wait for the RF status register to become zero (usually immediate.)

Write the location (8 LSBs) to the RF EEPROM Address Register.

Shift the location right 8 bits, add 0x40. Write to the RF Command Register. The RF Status Register EEOP bit comes on.

Wait (about 15 μ s) for the RF status register to become zero.

Concatenate the values of the Low and High EEPROM Return registers to get a 16-bit result.

The first EEPROM operation following system reset sometimes yields an improper result. Start by performing a dummy read, discarding the data.

The EEPROM data is arranged as shown in Table 3–28.

Table 3–28. RF EEPROM Map		
Address	Default	Description
0	75	74 => RF output present. Other values reserved.
1	0	2 = Quasonix Transmitter. Other values reserved.
2	0	Reserved.
3	102	Identifies this EEPROM map.
4	1	Number of RF Bands implemented.
5	2200	Band 0 Minimum RF Output Frequency (MHz)
6	2394	Band 0 Maximum RF Output Frequency (MHz)
7..10	0	Reserved for more band limits.
11	10241	Legacy Reserve
12	250	Legacy Reserve
13..14	0	Reserved
15	8	Number of pre-modulation filter selections available.
16	250	Pre-Mod Filter 0 cutoff in kHz. 65535 if square-sided.
17	500	Pre-Mod Filter 1 cutoff in kHz. 0 if filter not present.
18..31		etc. More Pre-Mod Filter selections.
32	10	Maximum RF output level in dBm.
33	-5	RF Attenuator step (dB)
34	19	Number of RF attenuator Steps available.
35		RF Attenuator D/A setting for listed maximum output level.
36		RF Attenuator D/A setting for listed maximum output + location 33.
37		RF Attenuator D/A setting for listed maximum output + 2 * location 33.
38..63		More RF Attenuator D/A settings
64,65	0, 0	Dummy Baseband Amplitude Lookup first entry
66,67		n (mv), n counts Baseband Amplitude Lookup entry
68..153		More Deviation Lookup entries
154..		Reserved

3.7.7.2 Baseband Output Level

The output level is adjustable from less than 200mv to approximately 8000mv p-p. This "volume control" is set by a D/A converter. To choose the setting, starting with the desired (unloaded) output level in millivolts and scan even-numbered EEPROM locations starting with location 64. When the EEPROM

data value is about to exceed the level setting, stop. These settings are sufficiently close together so the D/A value can be calculated by linear interpolation between the count values in the next-higher odd-numbered EEPROM locations. Example:

The desired level is 1000mv. Scanning the EEPROM yields:

Loc 104 = 919. $1000 - 919 = 81$.

Loc 105 = 6830.

Loc 106 = 1011. $1011 - 919 = 92$.

Loc 107 = 7214. $7214 - 3830 = 384$.

Value = $6830 + 384 \times 81 \div 92 = 7168$.

To set the level, split the value into bytes and write the eight LSBs to the Low RF Data Register, the MSBs to the High RF Data Register. Then write 0x0C to the RF Command Register. The DAC bit comes on in the RF status register and persists for about 10 μ s.

3.7.7.3 Pre-Mod Filtering

The baseband output passes through a pre-modulation filter on its way out. The effect is determined by the value of EEPROM location n+16, where n is the value of the PMF field. A value of 0 means the output is disabled. A value of 65535 means the data output is square-sided (i. e., unfiltered.) Any other value is the filter cutoff in kHz.

3.7.7.4 External Data Input

Setting the XMOD bit disconnects the simulator from the baseband output driver. Instead, the baseband output is driven from the external baseband input. To use this input, SW1-3 (SW1-4 for Ch 1) must be turned on and no shunt is permitted on E1A-11 (E1A-13 for Ch 1.)

To modulate the Quasonix transmitter with an external data source, connect the (TTL) signal to the simulator Slave Data Input and set the XDAT bit. This data is expected to be synchronous with the simulator clock.

3.7.7.5 The Quasonix Transmitter

The Quasonix transmitter is controlled by an RS232 link. An on-board UART provides the necessary communications path. Before attempting to use this path, initialize by writing 0xFE to the RS232 Baud Rate Register (57600 baud.) Then read the RS232 data register and discard the value.

In normal operation the Quasonix transmitter is controlled by sending a series of commands using this RS232 link. The transmitter responds to each

command with a response string ending with a '+' character. That results in the following general protocol to send a command:

Read the RS232 Data Register and discard.

Send a character by writing it to the RS232 Data Register. Wait (about 180µs) for the XBE bit to come back on in the RS232 Status Register. Repeat until all characters are sent.

Read the RS232 Status Register and wait for the RBF bit to come on. Read the RS232 Data Register. Repeat if the value returned is not '+' (0x2B.)

In the following, <sp> means 0x20 (ASCII space) and <cr> means 0x0D (carriage return.)

To set the RF output frequency send **FR<sp>nnnn.n<cr>** where nnnn.n is Fc in kHz.

To set the modulation type send **MO<sp>n<cr>** where n is 0 for SOQPSK and 1 for FM. Sending this command will "hang" the interface for approximately a second. This is normal.

To enable RF output send **RF<sp>1<cr>** and set the RFEN bit in the RF Control Register (SW should always be zero at this writing.)

To disable RF output send **RF<sp>0<cr>** and clear RFEN.

3.8 LS70 PCM Simulator

The PCM simulator architecture is a compromise between the requirements of generating a more-or-less static test data stream without processor intervention, but also allowing fairly efficient archival data playback, or uplink command generation. The simulator will need to be programmed using rather different philosophies for these different requirements.

Simulator operation is based on a repeating sequence of data output called a cycle. The cycle is defined by the contents of simulator memory starting at a given location, and continuing to be defined in consecutive locations thereafter until the end of cycle is identified by a bit in memory. Hence, operating the simulator consists of defining a cycle in memory, setting up registers, and starting it up. How to define the cycle is determined by the target use.

For static simulations a cycle wants to be either a minor frame or a major frame, if there is a major frame structure in the format and the minor frame is

short. If a cycle is defined as a major frame (which must be less than 128K words) a straightforward approach is to start the major frame definition at the beginning of page zero and define data values all the way through. You must supply a value for each word. The frame counter is not used.

Table 3–29. LS70 Simulator Write Register Summary									
Register	#	7	6	5	4	3	2	1	0
Command	10	MREQ	Mread	IACK	IENB	RUN	UACK	PAGE	–
Bankswitch	11	BANKSWITCH				REGS	–	ATTR	PAGE
Low Address	12	Mailbox/Exchange Address [7..0]							
High Address	13	Mailbox/Exchange Address [15..8]							
Data Memory (Bs=xx0x)	14	Data [7..0]							
	15	Data [15..8]							
Attribute Memory (Bs=xx1x)	14	IA	EOC	Wave	SKIP	WAVENUM		A16	
	15	Slave	LSBF	EOF	CRC	WL (Word Lgth-1)			
Frame Stop (Bs=1000, Address=0)	14	Last Minor Frame Count[7..0]							
	15	WDST	FCC	FAC	PRNS	Narch	FODD	LstFr [9..8]	
Mode (Bs=1000, Address=1)	14	XCLK	ClkDiv		XSEL	RA	Rev	CRC	CCIT
	15	WIDE	Uplink	FAST	BCRC	Event	BERT	ERR	2T15
NCO Control (Bs=1000, Address=2)	14	DATA	–	–	–	–	CLK	UPD	RST
	15	–	–	–	–	–	–	–	–
Encoder Control (Bs=1000, Address=3)	14	–	0	RNRZ Control		PCM Code			
	15	–	–	–	DIFF	INV	Swap	1/3	RATE
External Control (Bs=1000, Address=4)	14	Interrupt Vector (VME only)							
	15	(To be defined)							
Wave Step Period (Bs=1000, Addr=9..15)	14	–Period (μs) [7..0]							
	16	–Period (μs) [15..8]							
PRN Generator Control (Bs=1000, Addr=17..23)	14	–	–	–	–	REVP	PATTERN		
	15	–	–	–	–	–	–	–	–

Table 3–30. LS70 Simulator Read Register Summary									
Register	#	7	6	5	4	3	2	1	0
Identifier	00	0	"LS70SIM"						
Command	10	MREQ	Mread	Intrpt	IENB	RUN	UFLO	PAGE	Pgif
Bankswitch	11	BANKSWITCH				REGS	–	ATTR	PAGE
External Status (tbd)	12	–	–	–	–	–	0	XTS1	XTS0
Not defined	13	–	–	–	–	–	–	–	–
Memory Mailbox (Bs = 0xxx)	14	Defined Same As Write							
	15								

If the cycle is defined as a minor frame, words that cyclically change value or meaning from one frame to the next are defined as "wave 0" words. For each wave 0 word the IA and Wave attributes are set, and the data is fetched from memory located by concatenating the ten-bit frame counter value to bits 16..10 of the memory word. This would apply to the SFID count, if used, and anything else that varies cyclically with a period of one major frame. This could also be used to introduce URC patterns.

Seven other families of wave words, defined as "wave 1..7" are possible. For each family, data is fetched from memory located by concatenating an eight-bit wave counter value to bits 16..08 of the memory word. Each wave counter increments every n (1..65,535) microseconds, giving fundamental frequencies from about .015Hz to about 4kHz.

You may change data values for static simulation on-the-fly, but for more real-time usage it will probably be easier to declare a current value table approach. This can apply to either a minor or major frame-based scheme. Data items that might change are grouped into a table. Format locations associated with those items have their IA bits set and the location in the table stored in their data fields. Again, doing a minor-frame based simulation with subcommutated items will require the use of wave words.

For command uplink, where the data is sporadic bursts separated by idle fill, you need to make two format definitions. At the beginning of page 0 of memory you create a short (possibly a single word) cycle of checkerboard or whatever fill is defined. Commands are stored as data words to be output, starting at the beginning of page 1. Set the UPLINK bit in the mode register. When a command (or group of them) is loaded, set the PAGE bit in the command register. In this scheme, the lsb's of the command register acquire these meanings:

00 – Idle.

01 – A command is being output.

1x – A command is primed but has not started out yet.

For archival data playback, you also need to make two format definitions, in this case they are identical except being in different pages. Hopefully there will be some sort of repeating structure less than 64K words long in the archive. Presuming that, each page is set up to repeat that structure. Word lengths and data alignment need to reflect the conditions when the archive was recorded.

Most archive data structures have overhead that was not part of the original data stream. For instance, Lumistar decomm inserts a timestamp and status word at the start of each minor frame. Some archive formats have additional overhead. For each overhead word, insert a placeholder in simulator memory with the SKIP bit set. This permits these overhead words to be shoveled out along with the data. The simulator ignores them.

Hybrids of these schemes, or others yet unimagined, are not impossible. Research is left as an exercise for the experimenter. I suspect there will be free-lancing and creative misuse. The trouble with something this generalized is that *If you've seen one application, you've seen **one application**.*

3.8.1 Simulator Command Register and Mode Registers

The simulator Command register has a variety of bits that want quick access, so this is the one simulator register that is directly accessed. Other operational registers are indirect addressed in an effort to fit the simulator into a limited amount of I/O space. The Command register is laid out as shown in Table 3–31. Note the register is read/write but some of the bits have subtly different but related meanings for write and read operations – purposely so to allow for using read-modify-write-type accesses sensible under a variety of conditions. The Mode register (Table 3–32) is used to set static operating modes for the simulator. The register and others are indirect addressed, so access is slower, but their contents are not likely to change except when you perform a complete simulator setup.

To access these indirect registers, write 0x08 to the simulator Bankswitch register (setting only the REGS bit), and write their register number to the Low Address register. Then write the register values to the Exchange registers (0x14 and/or 0x15.)

Bit	Mnemonic	Description
0	PAGE IN EFFECT	Has no meaning when written. Returns the state of the simulator internal PAGE bit. This bit is copied from the Command register PAGE bit on each cycle boundary and serves as the MSB of the simulator memory start address for the cycle. The simulator output is defined starting at address 0xP000W of memory where P is the value of this bit and W is the value of WDST.
1	PAGE	Specifies the value of PAGE IN EFFECT starting with the next cycle boundary. If the bit is unchanged, the same page is used over and over. When read, returns the last value written unless the Mode register UPLINK bit is set.
2	UACK UFLOW	When read, returns the state of the simulator underflow flag. Hopefully this never happens. Writing one clears the flag. Writing zero has no effect.
3	RUN	Writing zero stops the generation of data. The simulator output clock is stopped. Writing one allows output to commence based on the definition starting at the address specified by the PAGE and WDST bits. When read, returns the last value written.
4	IENB	Causes the simulator to generate a system interrupt when INTRPT is set. When read, returns the last value written.
5	IACK INTRPT	When read, returns the state of the simulator interrupt flag. This flag is set on every cycle boundary, whether interrupts are enabled or not. Writing one to this bit clears the flag. Writing zero has no effect.
6	MREAD	Controls direction of transfer between the simulator memory and exchange registers for mailbox memory accesses. Set for reads, clear for writes. When read, returns the last value written.
7	MREQ	Writing one initiates a simulator mailbox memory access. Writing zero has no effect. When read, returns a one if an access is in progress.

Table 3–32. Simulator Mode Register		
Bit	Mnemonic	Description
0	CCITT	Causes a CRC-CCITT checkword to be calculated. Otherwise CRC-16 is calculated.
1	CRCEN	Causes a CRC checkword to be calculated. Has no meaning if no CRC location is specified in the simulator word attributes.
2	REVCRC	Causes a reversed CRC checkword to be calculated.
3	RA	Data values written to simulator memory are right-aligned.
4	XSEL	Selects which external clock input to use. Zero selects TTL-compatible clock input 0. One selects RS422 clock input 1.
6..5	DIV	Selects a prescale ratio for the simulator clock: Choose one of: 00 – Divide by 1. 01 – Divide by 16. 10 – Divide by 256. 11 – Divide by 4096.
7	XCLK	Use external datarate simulator clock selected by XSEL.
8	2T15	Specifies a 32,767-bit PRN pattern when BERT set. This bit is logically OR'd with bit 1 of PRN Generator #1 Control register (0x11)
9	ERR	Forces one error every PRN pattern iteration.
10	BERT	Pre-empts simulator output with the output of PRN pattern generator 1, and causes that generator to run continuously. Zero for normal operation.
11	EVENT	A 0-to-1 transition of this bit causes a single error event if BERT set.
12	BCRC	Normally the CRC generator is reset when the CRC checkword is output. Set this bit to reset the CRC generator <i>again</i> at the end of a minor frame.
13	FASTMODE	Causes simulator FIFO memory to run in an approximate half-full state. Clearing it causes the FIFO to run in an almost empty state. This bit should be set except for low-speed uplink data. When the bit is cleared, the simulator interrupt occurs when the last word in the cycle starts to go out. When set, the interrupt can occur as much as 128 words early.
14	UPLINK	Clears the Command register PAGE bit when PAGE IN EFFECT is set.
15	WIDE	Simulator frame strobe output high during the last word in the minor frame. Otherwise the strobe is high during the last <i>bit</i> .

3.8.2 Output Formatting

The simulator output is a PCM data stream and symbol-rate (e. g., twice data rate for Bi-Phase codes, etc.) clock. At the output is a k=7 convolutional encoder, followed by a randomizer, followed by a PCM encoder. These encoders are set up by an Encoder Control register. This register is accessed by indirect addressing through the Exchange register. To write values to it, you must write 0x08 to the simulator Bankswitch register (setting only the REGS bit), and write 0x03 to the Low Address register. The upper and lower halves of the Encoder Control register (Table 3–23) can then be accessed by writing the upper and lower halves of the Exchange register. One of the parameters associated with this register is the output PCM Code. There are a number of selections here. Each has an implied parameter called the Code

Factor associated with it. This factor and others are used in the calculations to set up the simulator clock generator; to properly set the data rate the value written to this register must be known.

The RF and baseband output controls of the LS50 simulator in ¶3.7.7 also apply verbatim to the LS70.

Table 3–33. Simulator Frame Stop Register

Bit	Mnemonic	Description
9..0	LASTFRAME	Terminal value for the frame counter. Set to one less than the number of frames per major frame.
10	FODD	Meaningless unless NARCH and FAC both set. Set to invert the minor frame sync pattern in alternate frames starting with the second frame in the major frame. Clear to start with the first frame.
11	NARCH	Clear for archive playback. Set for static simulation if FAC or FCC is needed.
12	PRNS	Changes interpretation of the SLAVE and WAVNUM attribute fields if set. See Table 3–35.
13	FAC	If NARCH = 1, causes words identified with the SKIP/FSP attribute to be inverted in alternate minor frames.
14	FCC	If NARCH = 1, causes words identified with the SKIP/FSP attribute to be inverted in the first minor frame of the major frame.
15	WDST	The LSB of the starting address of the cycle definition in memory.

3.8.3 The Clock Generator

The LS70 simulator clock generator is set to a particular data rate in exactly the same manner as the LS50 simulator; the same values are written the same way. See ¶3.7.3.

3.8.4 Communicating With Simulator Memory

The simulator is provided with 256K 32-bit memory words. When accessed through the mailbox, the data portion is the 16 LSBs and the attributes for that data are in the 16 MSBs. When the memory is accessed by direct-mapping, it appears to be organized differently. A 32-bit memory access references either two consecutive data values, or two consecutive attribute values, depending on the setting of the ATTR bit.

Table 3–34. Simulator Bankswitch Register

Bit	Mnemonic	Description
0	PAGE	Specifies memory accesses reference memory Page 1.
1	ATTR	Specifies memory accesses reference attribute memory.
2		Not defined.
3	REGS	Use exchange register for indirectly-addressed simulator registers instead of memory.
7..4	BANK	Address bits 17..14 for direct-mapped memory accesses. All four bits are used in page-mode. Only the MSB used in flat mode.

For mailbox accesses, specify a word location by writing to the Low and High Address registers. Select which memory to access in the two LSBs of the Bankswitch register (Table 3–34.) When read, the bankswitch register returns its current value.

From the viewpoint of the mailbox address register and addresses activated by the IA bit, memory is considered to be word-addressed!

Table 3–35. Simulator Word Attributes

Bit	Mnemonic	Description
0	A16	Has meaning only if the IA bit is set. Specifies a page for indirect accesses. Only the first 64K words of the page are accessible.
1..3	WAVENUM	If IA and WAVE are set, selects the wave counter used. 0 selects the minor frame counter. 1..7 select the asynchronous wave counters. If PRNS (Table 3–33) and SLAVE are set, values of 1..7 select PRN pattern generators 1..7.
4	SKIP/FSP	If NARCH = 0 (Table 3–33) overrides the attributes and causes any data associated with this word not to be output. For playback purposes, this can be used to ignore recurring overhead in the archive, e. g., timestamps. If NARCH = 1, identifies words containing the minor frame sync pattern but meaningless unless either FAC or FCC is set.
5	WAVE	Meaningless if IA is zero. If IA is one, causes the memory address of output data to be generated by concatenating the A16 bit, the MSBs of the data field, and the frame or wave counter value. Allows introducing SFID counts, waveforms, and other dynamics.
6	EOC	Set to identify the last word in the cycle. Sets the Interrupt event flag and causes the next output word to be determined by the setting of PAGE IN EFFECT.
7	IA	If zero, the data field associated with this attribute is output. If one, the A16 bit is concatenated with the data field (see WAVE) and interpreted as a memory address and <i>that</i> data is output instead.
11..8	WL	The word length in bits, less 1.
12	CRCW	Output the CRC checkword, starting with the first bit of this word. The checkword output lasts for 16 bit periods, during which any other data otherwise defined for output is discarded.
13	EOF	If the frame counter equals the LastFrame register value, clears it. Otherwise increments the frame counter. If SKIP is not set, causes a pulse at the frame strobe output.
14	LSBF	Causes the data to be output LSB-first.
15	SLAVE	IA data value in memory for this word is ignored. Allow the slave clock output to run during this word. If PRNS = 0, insert whatever appears at the slave data input. If PRNS = 1, run the PRN generator selected by WAVENUM during this word and insert its output.

If you are performing a memory write, write the data to the Low and High Exchange registers and then write to the Command register, clearing MREAD and setting MREQ. Poll the Command register, waiting for MREQ to go away, which will usually take no more than 200ns.

If you are performing a memory read, write to the Command register, setting MREAD and MREQ. Poll the Command register, waiting for MREQ to go away. The returned data can then be read from the Exchange registers.

3.8.5 Attributes and Data

Each word location has associated data and attributes. The attribute fields are accessed by setting the ATTR bit. The memory word is formatted per Table 3–35. For LSB-first data, the frame sync words need to be bit-reversed to get the pattern to output properly. If you have trouble wrapping your brain around the preceding sentence, force the LSBF bit to zero except for playback purposes. Also note the simulator allocates an integral number of words to the frame sync pattern, and an entire word to the SFID count, regardless of how many bits they actually use.

3.8.6 Wave Counters

Seven counters are used to select outputs of waveforms slower than the minor frame rate. These counters address sections of data memory, selected by A16 and the MSBs of the address field. Wave counter 0 is the ten-bit minor frame counter. It is incremented by the EOF bit and has a period of one major frame. Wave counters 1..7 are asynchronous to the format. Each has a period of 256 times the wave step period associated with the counter. To set a fundamental frequency of f Hz, set the Wave Step Period register to $-3906.25 \div f$ rounded to the nearest two's complement integer.

3.8.7 PRN Pattern Generators

Seven Pseudo-Random-Number (PRN) generators are provided. These are enabled by the PRNS bit in the Frame Stop Register. Setting this bit overrides the normal meaning of the SLAVE attribute field, making it a PRN field. When this attribute appears, the WAVENUM attribute is used to select one of the seven generators. During the period of the word, the generator runs, and its output appears in the data stream. At the end of the word the generator stops, and its value is held. Each PRN generator 1..7 has its own control register, as shown in Table 3–36. To access the control register, set REGS, and set the low address register to $16 + \text{generator number}$.

<p>It is not given that which direction is "reverse" is without discussion. It is further not given the proper tap selections are all themselves without discussion. For the 11- and 15-bit patterns, I selected taps consistent with the fraternity's definition of RNRZ codes, and taps for longer patterns in a likewise manner.</p>
--

Bit	Mnemonic	Description
0..2	PATTERN	Selects the PRN polynomial taps used by this generator. For generator 1 only, bit 1 is OR'd with Mode register bit 2T15. 0: 11-bit (taps 9, 11) 1: Checkerboard (101010...) 2: 15-bit (taps 14,15) 3: 17-bit (taps 14,17) 4: 19-bit (taps 13, 17, 18, 19) 5: 21-bit (taps 19, 21) 6: 23-bit (taps 18, 23) 7: 25-bit (taps 18, 25)
3	REVP	Generate PRN bits in reverse order.
4..15		Meaningless

3.9 The IRIG Time Generator

The IRIG Time Generator is physically part of the PCM Simulator but a distinct logical entity. It has its own setup registers, all accessed through a single I/O address, the adjacent address being an indirect address register. Hence, to access a register in the generator, write the register number to the address register (register 0x16 relative to the base I/O address) and then access the data through register 0x17.

Register	#	7	6	5	4	3	2	1	0
Indirect Address	16	–	–	–	–	Adr			
Register 17:	Adr								
BCD Seconds Preset	00	–	10's Seconds			1's Seconds			
BCD Minutes Preset	01	–	10's Minutes			1's Minutes			
BCD Hours Preset	02	–	–	10's Hours		1's Hours			
BCD Days Preset	03	10's Days				1's Days			
	04	–	–	–	–	–	–	100's Days	
Control Functions (by index number)	05	57	56	55	54	53	52	51	50
	06	66	65	64	63	62	61	60	58
	07	75	74	73	72	71	70	68	67
	08	–	–	–	–	–	78	77	76
Seconds from Midnight Preset (IRIG A, IRIG B)	09	Seconds [7..0]							
	0A	Seconds [15..8]							
Control	0B	SET	–	Arrow		MODE		HOLD	Sec16
Data Hold Frac Secs	0C	10 th s Seconds				100 th s Seconds			
BCD Data Hold Secs	0D	DHold	10's Seconds			1's Seconds			

Table 3–38. IRIG Generator Read Register Summary

Register	#	7	6	5	4	3	2	1	0
Not Defined	16	–	–	–	–	–	–	–	–
Register 17:	Adr								
BCD Frac Seconds	00	.1's Seconds				.01's Seconds			
BCD Seconds	01	0	10's Seconds			1's Seconds			
BCD Minutes	02	0	10's Minutes			1's Minutes			
BCD Hours	03	0	0	10's Hours		1's Hours			
BCD Days	04	10's Days				1's Days			
BCD Days	05	Indeterminate						100's Days	

3.9.1 Setting Time

Setting the IRIG generator up formally is a three step process. First, write the generator Control register (Table 3–31) setting the MODE and ARROW fields to get the time carrier running at the right frequency. Then write the start time into the preset registers (Table 3–29.) There isn't much call for the control functions, but if you have values, write them at this time. Finally, write the Control register again, this time with the PRESET bit set. This loads the time counters and resets the generator back to the beginning of the (first) time frame of that second.

You may also read the time of day back from the generator, but the data returned is unfrozen and may be subject to rollover errors. This path is mostly for maintenance purposes. The time is in BCD. See Table 3–38.

Table 3–39. IRIG Generator Control Register

Bit	Mnemonic	Description
0	SEC16	Binary time in seconds from midnight is 17 bits long. This is the MSB to go into effect on a PRESET.
1	HOLD	When set, stops time in seconds from incrementing.
3..2	MODE	Selects a time carrier: 0x – IRIG B. 10 – IRIG A. 11 – IRIG G.
5..4	ARROW	Specifies the length of the arrow of time, i. e., carrier frequency: 00 – Real time 01 – Time at half rate. 10 – Time at twice rate.
6		Meaningless.
7	PRESET	Resets the generator to the beginning of a time frame, clears fractional seconds, and places the time loaded into the Preset registers into effect.

3.9.2 Time Generator LS70 Data Hold

Table 3–37 shows two registers to load a Data Hold value in seconds and fractional seconds. These registers are applicable only for LS70 configurations and their values are meaningless unless you also set the Dhold flag. If this flag

is set (its state is returned when time is read) the simulator stops outputting fresh data (although its clock continues to run if RUN is set. This was designed deliberately with the following sequence of operations in mind. Bear in mind that any time reader connected will probably take several time frames to recover from the time-seeding activity.

1. Load the simulator with a format (or the beginning of archived data.) DO NOT set the simulator RUN flag.
2. Seed the time generator with time at least a second prior to a desired start time.
3. Set the Data Hold registers to the hundredth-of-second of the minute the data should start.
4. Now set simulator RUN. The data rate clock runs, but no data comes out.
5. When the programmed Data Hold time is reached sometime within the next minute, data starts. Simultaneously, the Dhold flag clears.

3.10 Interrupts

If your data rates are extremely low and your operational demands are not great, you may be able to avoid using interrupts, using the polling technique below. Unfortunately that doesn't seem to happen often.

3.10.1 Polling

The decommutator and simulator both have interrupt flags that latch set on a particular event, regardless of whether interrupts are actually enabled.

If you want to use polling to synchronize your program with the decommutator, wait for the Status register INTRPT bit to come on (Table 3–13) to indicate a buffer turnover. As soon as you see it, read from the Buffer Status register, discarding the value. This will turn off the flag. You can (and should) immediately move the data from buffer memory.

In the event you want to poll to synchronize your program with the operation of the simulator, wait for the Command register INTRPT bit to come on (Table 3–20 or 3–31) to indicate a minor frame boundary. As soon as you see it, turn off the flag by clearing bit 7 (MREQ) of the value read and writing it back. Bit 0 returns which page of simulator memory is in use at that moment.

3.10.2 Using Interrupts

Polling techniques will suffice only for the least-demanding applications. Usually you will have to engage interrupts and synchronize at interrupt level.

You will usually need to connect your driver or application to the PCI interrupt assigned to the board. This calls for careful setup and usage.

3.10.2.1 Connecting to the System

You need an interrupt handler in your driver or application. The usual universal rules of interrupt processing apply; you need to save the processor state, acknowledge the interrupt, expeditiously do what time-critical things you need, restore the processor state and get back out. In the iAP86 (PC) environment, the only state saved by the interrupt itself is the program counter and processor flag register. Any CPU registers you use must be saved for later restoration. Of course, the stackpointer needs to be left where you found it.

In PCI a further complication is caused by the fact that as a PCI device, the physical interrupt may be shared with some other device. This means you may get interrupts that are not yours, and you cannot simply restore registers and return from exception at the end. You need to interrogate the card to find out which interrupt has been assigned (§3.2) and connect your handler, usually by a system call passing the address of the handler. Before doing that, though, you may (as in MS-DOS) be required to first interrogate the by another such system call to determine who currently "owns" the interrupt. On exit your handler must restore CPU registers and end by transferring control (e. g., by a far jump) to *that* entity. Other environments may have different ways to accomplish this.

3.10.2.2 Preparing to be Interrupted

After you have connected your handler to the system, you must further prepare the system and the board for interrupts. For PC environments this means making sure the "8259" interrupt is unmasked for the selected IRQ, and experience indicates it is wise to issue a non-specific End-of-Interrupt at this time. Again for PC environments, this means writing 0x20 to I/O port 0x20 and, if the IRQ number is greater than 7, also writing 0x20 to I/O port 0xA0. Theoretically you should not need to do this, but the theory is contrary to experiment, and this seems to be harmless.

History of moving designs across architectures has left several levels of interrupt enable. The decommutator *per se* and simulator both have INTRPT status flags. You must clear out any pending interrupts. For the decommutator, read the Buffer Control/Status register and write the value read back to it. For the simulator, read the Command register and write the value read back to it.

If the DMA controller is to be used, you need to clear any pending DMA interrupt. This is done by reading the DMA Command/Status registers and writing 0x08 back if the value read had bit 4 (i. e., logical AND with 0x10) set. These are at offsets 0xA8 (Ch 0) and/or 0xA9 (Ch 1) in the PCI9056

Runtime Register space. (PCI9056 Runtime Registers appear in both memory and I/O space.)

You must enable the PCI9056 PCI interrupt setting the Interrupt Control/Status register at PCI9056 Runtime Register offset 0x69, with the logical OR of 0x09 with the value read from that register.

Finally you must set the IENB bit(s) for the decommutator and/or simulator in the Decommutator Control and/or Simulator Command registers. If data is running (for the simulator, data is always running if its clock is running) you will be interrupted eventually.

3.10.2.3 Being Interrupted

At interrupt time your handler will be called. Again, because of the shared nature of PCI interrupts, you must interrogate the card to find out if he is the one interrupting. Decommutator and simulator interrupts set the DINT and SINT bits in the Buffer Control/Status register. Immediately write back the value you read. This will clear these bits and release the PCI interrupt *if they were interrupting* (writing zero back to these bits has no effect.) If the bit in the value read is clear, skip over the operation it calls for and continue with the rest of your handler. In practice, this may mean skipping everything.

For the simulator interrupt you would ordinarily set a semaphore for some operation down at task level, and possibly rewrite the simulator Command register, toggling its PAGE bit.

For the decommutator you would initiate whatever sort of operation you have designed to move data from its buffer memory, by means of pick/choose, block move, or initiating a DMA operation.

If bit 2 of the Buffer Status is set, the interrupt was from the PCI9056 itself, usually a DMA end-of-operation. You cannot clear this bit in the Buffer Status by writing to it. There is the further complication in two-channel environments. The PCI9056 has two DMA channels, with the implication that one is reserved for each channel, but only one interrupt pin; both interrupts are connected to bit 2 of the Buffer Status for the Ch 0 decommutator – if the Ch 1 is also a decommutator, bit 2 of *its* Buffer Status is meaningless. Therefore you need to poll the two DMA Command/ Status registers. If bit 4 was set, the DMA interrupt for that channel is active and you need to write 0x08 back to clear it. Usually for a DMA interrupt you would post some sort of semaphore indicating data is available in system memory.

Finally, if you did anything, you should issue a non-specific End-of-Interrupt before leaving your handler. This is fail-safe because you are still at interrupt level. Any other handlers daisy-chained downstream from yours will still run.

3.11 DMA

The PLX PCI9056 includes two DMA controllers that permit rapid data movement. The implication is to use them to move incoming data from the buffer to system memory after the buffers toggle. By arbitrary convention, DMA Channel 1 is assigned to Ch 1, and Channel 0 is reserved for use by Ch 1 if configured. If you plan to write for the DMA controller you may have need for the PCI9056 data sheet. You can get this document from your local PLX Technologies, Inc. representative if one is handy. Failing that, you may be able to download a .pdf from <http://www.plxtech.com>. Failing that, contact us.

DMA operations require knowledge of physical addresses in system memory, which may or may not be the same as the logical addresses used by your application. You will need to make the necessary system calls to convert logical addresses to physical addresses. If your operating system does not provide this capability, you cannot use the DMA Controller to move data.

For all DMA applications the PCI9056 PCI Command Register (a 16-bit register in Runtime register space at offset 0x04) should be set to 0x07. Additionally, each DMA controller has three data items in registers. These registers are in the Runtime register space:

- A 32-bit DMA Mode register at offset 0x94 (0x80 for DMA Channel 0.)
- A 16-byte Descriptor at offset 0x98 (0x84 for Channel 0.)
- An 8-bit Command register at offset 0xA9 (0xA8 for Channel 0.)

DMA operations may be run as chained or unchained. Unchained DMA, using a single descriptor, can be used if you can always move the entire active part of the data buffer to one continuous physical buffer in system memory. The memory management used by some operating systems (e. g., Windows NT) does not always permit that because it breaks all user memory buffers into segments of some arbitrary size (4096 bytes for NT) or less. You need to set up chained DMA operations in these systems. A chained DMA operation needs multiple descriptors (collectively called a "chaining table" albeit the actual structure is that of a singly-linked list) in memory someplace.

The PCI9056 allows the chaining table to be stored either in local (i. e., on-board) memory or PCI (i. e., elsewhere in the system) memory. Some similar products from other companies used an earlier version of the PLX part that did not allow the chaining table to be in PCI memory. Those products had local memory reserved for the chaining table. Conversely, the board has no local memory where the chaining table can reliably be stored.

3.11.1 DMA Descriptors

A descriptor is a structure of four 32-bit items. When descriptors are stored in memory, each must start on a paragraph boundary. This means the physical address of the first byte of the descriptor must end in 0x0.

The first item of the descriptor is the PCI physical address of the target buffer in system memory. For unchained DMA this is the start address. For chained DMA, this is the starting physical address of the segment.

The next item of the descriptor is the local physical address of the source data. The active data buffer is at local addresses 0x00000..0x1FFFF and there is no local mapping in this area so the address in the (first) descriptor is normally always zero.

For educational purposes local address space 0x20000..0x23FFF also points into the buffer, but the high-order local address bits are supplied by the bankswitch register. The DMA controller has no knowledge or control over the bankswitch register so this is of little utility. **HOWEVER**, the other mapping (starting at local address 0) is *always in effect*, so the DMA controller can be used to pour out the entire buffer whether the PCI interface is in flat or page mode.

The third item in the descriptor is the transfer size in bytes. For an unchained DMA operation this is the active buffer size. For chained DMA this is the size of the current segment.

The fourth item is called a descriptor pointer. This item is split into two fields. Bits 04..31 have meaning only for chained DMA. They are the 28 MSBs of the physical address of the next PCI physical address field in the chaining table (why this is referred to as a pointer.) When this value is used as an address the four LSBs are understood to be zero regardless of their real value.

Bit 03 is a transfer direction bit and is always 1 to move data from buffers to system memory.

Bit 0 has meaning only for chained DMA. It is 1 to specify the next descriptor pointer field is a PCI physical address and must be 1 for all applications using chained DMA.

Bits 01..02 have meaning only for chained DMA. They must be 11 for the last descriptor in the chaining table, and 00 for all of its predecessors.

3.11.2 DMA Channel Mode Register

The DMA Mode Register specifies operating conditions for a DMA operation. Only a few values are meaningful here. The recommended basic value is 0x0143. Add 0x200 more to this value to specify a chained DMA.

Additionally, add yet 0x400 more if you want a second interrupt when the DMA operation completes.

When setting up a chained DMA operation, the first descriptor can be loaded directly into the PCI9056 descriptor register. This is not recommended for two reasons. First, because it creates the complication of a special case, and also because the earlier PCI9080 has a known bug that causes improper operation if physical PCI memory mapping could result in a mixture of chained and unchained DMA operations. If you will have *any* need for chained DMA, you should use chaining for *all* DMA operations. The descriptor register in the PCI9056 is loaded with PCI and local addresses that are meaningless, a byte count of all zeros, and the descriptor pointer set up to point to the first entry in the chaining table. We tried this and it appears to always work.

3.11.3 DMA Channel Command Register

The Command Register is used to start/stop DMA operations and monitor their progress. This register is a set of eight isolated bits:

Bit 0 is a channel enable bit. This bit should always be written as a one except in the unlikely event of wanting to pause or abort a DMA operation in progress, which you would ordinarily never do. When read, returns the bit value written.

Bit 1 is the DMA Start Command bit. Write a 1 after the descriptor(s) have been set up to start a DMA operation. This bit is write-only and writing a zero has no effect.

Bit 2 is the DMA Abort Command bit. Writing a 1 (with bit 0 cleared) terminates a DMA operation in progress. Ordinarily you would never do this, though. This bit is write-only and writing a zero has no effect.

Bit 3 is the DMA Interrupt Acknowledge bit. You must write a 1 in response to a DMA completion interrupt. This is the only way to clear the DMA bit of the Buffer Control register. This bit is write-only and writing a zero has no effect.

Bit 4 is read-only. It returns 1 whenever there is no DMA operation in progress. You can use this bit to monitor the progress of a DMA operation when running without using a DMA completion interrupt.

Bits 5..7 are undefined.

3.12 Bit Error Rate Measurement

The decommutator can perform simple Bit Error Rate (BER) measurements where a test loop can be driven from the simulator and monitored by the decommutator. The decommutator is equipped with a "sidelong" PseudoRandomNoise (PRN) pattern synchronizer. This constantly attempts to lock to the selected one of seven PRN patterns and count any errors detected. This runs all the time. The PRN synchronizer needs at least sixteen consecutive error-free bits to achieve lock, but thereafter can maintain lock unless the short-term BER exceeds 4×10^{-1} .

Every second, when the decommutator clock counter updates, the error accumulated error count is latched and can be read from the error count registers, and the error counter is cleared. The error count also includes several status bits as shown in Table 3–41. If the status bits do not show overflow or loss of sync, the BER can be calculated by dividing the clock count value into the error count value read on the same update.

The error count value is meaningless unless a well-formed PRN pattern is being received. The simulator has the capability of generating such a pattern, controlled by bits in the simulator Pattern register (Table 3–40.) For compatibility with older boards the 2T15 bit in the Mode register is logically OR'ed with bit 1 of the Pattern register. Normally the decommutator and simulator Pattern register values must match each other. 2T15 in the simulator Mode register must match the state of the 2T15 bit in the decommutator Control register. Setting the BERT bit replaces the normal simulator NRZL and PCM outputs with the PRN pattern, and also causes a one-bit pulse on the simulator Frame Strobe output with each iteration of the pattern. Setting the ERR bit causes one bit of each iteration of the pattern to be wrong. With this bit set, and no other link errors, the BER should be 1 divided by the pattern length, e. g., 4.885×10^{-4} for a 2047-bit pattern. Setting the EVENT bit, then clearing it, causes a single error to be introduced into the pattern. This event is asynchronous, i. e., the error is not a specific bit.

Table 3–40. Pattern Registers		
Bit	Mnemonic	Description
2..0	PATTERN	PRN pattern length, defined as: 000: $2^{11}-1$ (2047 bits) 100: $2^{19}-1$ (524,287 bits) 001: Reserved 101: $2^{21}-1$ (2,097,151 bits) 010: $2^{15}-1$ (32767 bits) 110: $2^{23}-1$ (8,388,607 bits) 011: $2^{17}-1$ (131,071 bits) 111: $2^{25}-1$ (33,554,431 bits)
3	REV	"Reverse." Moves inner tap selections away from the "big" end of the shift register toward the "little" end.
7..4		Meaningless.

Table 3–41. Error Count High Register		
Bit	Mnemonic	Description
3..0		Bits [19..16] of the error count.
4		Meaningless.
5	ECOVF	Set if the error counter overflowed during the last sample.
6	WOOS	Set if the PRN synchronizer lost lock during the last sample. If this bit is set the BER calculation may be grossly inaccurate.
7	OOS	This bit is not latched. It is set if the PRN synchronizer is presently out of lock.

3.13 Daughtercard Interface

The !PRES bit in the Daughtercard Status Register will return zero if a daughtercard is present. This makes the other daughtercard registers meaningful. The Data Register is used to write a series of setup bytes to the daughtercard. Write by this sequence:

Write the [next] setup byte to the Data Register.

Write the Control Register with the !STROBE bit cleared. BUSY sets. Immediately write the control register again with !STROBE set.

Wait until BUSY clears. Repeat if more to send.

Table 3–42. Daughtercard Write Register Summary									
Register	#	7	6	5	4	3	2	1	0
Daughtercard Data	0E	See LS40 Manual or ¶3.12.3							
Daughtercard Control	0F	–	–	Source3..2	!INIT	!STB	Source1..0		

Table 3–43 Daughtercard Read Register Summary									
Register	#	7	6	5	4	3	2	1	0
Daughtercard Data	0E	See LS40 Manual or ¶3.12.3							
Daughtercard Status	0F	1	READ	0	!Pres	1	BUSY	LOCK	SIG

Table 3–44. Daughtercard Control Register		
Bit	Mnemonic	Description
1..0	SOURCE	LS40 Bit Synchronizer Input source LSBs. See Table 3–46.
2	!STROBE	Clearing the bit, then setting again indicates you wrote fresh data to the Command Register. Set by system reset.
3	!INIT	Daughtercard Reset when cleared. For compatibility with other systems. This bit has no meaning for Lumistar Daughtercards. Set by system reset.
5..4	SOURCE	LS40 Bit Synchronizer Input source MSBs. See Table 3–46.
7..6		Not Used.

Table 3–45. Daughtercard Status Register		
Bit	Mnemonic	Description
0	SIG	Signal Present. Set to indicate the input signal amplitude is valid.
1	LOCK	Bit Synchronizer Lock status.
2	BUSY	Set in response to !STROBE. Persists until the daughtercard is ready to accept another character.
3	1	Always 1.
4	!PRESENT	Set if NO Daughtercard installed.
5	0	Always 0.
6	READ	Readback Operation in progress
7	1	Always 1.

Certain commands cause the daughtercard to return a string of response bytes, including all LS40 commands of the form 0xEn. If the daughtercard has a response, it will assert READ in the status register before it releases BUSY. To gather the string of response bytes:

Wait until BUSY clears. If READ is clear, skip out.

Read the Data Register and save the value.

Write the Control Register with the !STROBE bit cleared. BUSY sets. Immediately write the control register again with !STROBE set.

Go back and wait on BUSY again.

3.13.1 Plug-and-Play

If !PRESENT is zero, the board can be queried for exactly what daughtercard is installed. To do so, send data bytes 0xED and 0x0A in succession. The daughtercard should assert READ and return six bytes. The third byte returned may be interpreted as (other values tbd):

0x31 – 10MBPS LS40 Bit Synchronizer.

0x32 – 20MBPS LS40 Bit Synchronizer.

0x38 – LS38 70MHz Demodulator/Bit Synchronizer.

3.13.2 LS40 Bit Synchronizer

Setup and status responses for the LS40 are defined in the LS-040 Technical Manual. Additionally, the Input Source fields in the Daughtercard Control Register are defined in Table 3–38.

MSBs	LSBs	Description
00	00	J1-36
00	01	J1-9 PCM Simulator Baseband output
00	10	J1-37
00	11	J1-39
01	00	J1-34
01	01	J1-42
01	10	J1-35
01	11	J1-49
10	00	Differential J1-34 – J1-36
10	01	Differential J1-42 – J1-9 Simulator Output
10	10	Differential J1-35 – J1-37
10	11	Differential J1-40 – J1-39

3.13.3 LS38 70MHz Demodulator

These notes to apply until superceded by an LS38 Technical Manual.

The LS38 is set up by writing a fourteen-byte command packet (Table 3–47.)

Byte	7	6	5	4	3	2	1	0
1	0x10							
2	Bit Synchronizer Loop Width: 00000 = 2% 01010 = 0.1% 00001 = 1% 11000 = 0.05% 01000 = 0.5% 11001 = 0.02% 01001 = 0.2% 11010 = 0.01%					0	Bit Rate [25..24]	
3	Bit Rate [23..16]							
4	Bit Rate [15..8]							
5	Bit Rate [7..0]							
6	FM Modulation Index (0000)				0 = FM. 1 = SOQPSK			
7	Demodulator Loop Bandwidth: 0x00 = 100 Hz 0x03 = 1000 Hz 0x06 = 10000 Hz 0x01 = 250 Hz 0x04 = 2500 Hz 0x07 = 25000 Hz 0x02 = 500 Hz 0x05 = 5000 Hz 0x08 = 50000 Hz							
8	–	–	–	–	–	DERAND	DPOL	CLKPOL
9	–	–	–	–	–	–	–	–
10	–	–	–	–	–	–	–	–
11	–	–	–	–	–	–	–	–
12	–	–	–	–	–	–	–	–
13	–	–	–	–	–	–	–	–
14	0x0A							

Setting DERAND turns on an RNRZ15 derandomizer. Setting DPOL or CLKPOL inverts the output data or clock.

The LS38 can be caused to send a status response by sending data bytes 0x20 and 0x0A in succession. This will cause the LS38 to return a fifteen-byte status packet defined in Table 3–48.

Table 3–48. LS38 Status Packet								
Byte	7	6	5	4	3	2	1	0
1	0x20							
2	QUAL	LOCK	SIGNAL	–	–	–	Clock Count [25..24]	
3	Clock Count [23..16]							
4	Clock Count [15..8]							
5	Clock Count [7..0]							
6	FM Modulation Index = $0.02n + 0.56$				–	–	–	–
7	Signal Quality							
8	Saturation Count							
9	–	–	–	–	–	Positive	SigStr[9..8]	
10	SigStr [7..0]							
11	0							
12	0							
13	0							
14	0							
15	0							

QUAL, LOCK, SIGNAL reflect the state of the board status LED drivers.

Signal Quality is a dimensionless number in the range 0 (very poor) to 255 (very good.)

Saturation Count is the count of input signal overranges in the last (?)

SigStr is the estimated input level in tenths of dBm