High-Performance

PCM Simulator

Technical Manual

January 2005

U0700101
Glenn Toennes

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

1 — Introduction

1.1 General

The Lumistar Multimode PCM Simulator has features to perform static simulations with on-the-fly data updates, archive file playback, and uplink applications.

1.2 Specifications

Table 1–1. Simulator Specifications	
Logic-Level Outputs	TTL & RS-422 NRZ-L and PCM Data, 0-degree clock & minor frame strobe. Slave Clock out for sharing asynchronous embedded formats with a slave simulator.
Baseband Output	2Vp-p looking into 50Ω. On special order, programmable output level with pre-mod filters available **in lieu of** RF Output Option.
External Clock Input	Single-ended TTL or RS-422
Logic Level Data Input	TTL Slave Data/External Data Input.
Output Data Rate	64 bps to 20 Mbps (NRZ codes) 64 bps to 10 Mbps (all other codes)
Data Rate Stability	We use the best crystal oscillator $3 can buy.
PCM Codes	NRZ-L/M/S; BiΦ-L/M/S; DM-M/S; M^2, RNRZ-L-11/15
Convolutional Encoder	k=7 Rate 1/2, 1/3, normal or differential.
Word Length	Variable from 3 to 16 bits per word on a word-by-word basis
CRC Generator	CRC16/CCITT forward/reverse
Minor Frame Length	2 to 65,535 words per minor frame
Major Frame Length	Up to 65,535 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 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 256μs to 15 seconds in steps of 256μs. 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.
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.

Table 1–2. IRIG Time Generator Specifications	
Output Formats	IRIG B, IRIG A, IRIG G including control functions and binary time-of-day.
Carrier Rate	1/2x, 1x, 2x ± 100ppm
Preset	Preset sets generator to a time frame boundary.
TTL Outputs	TTL-Compatible DC Level. TTL-Compatible 1 PPS pulse 1 ms wide.
AM Carrier Output	Time carrier output levels 1V/3.3V p-p. Capable of driving <100 Ω loads.

Table 1–3. Optional RF Output Specifications	
Frequency	(Choose one) L-Band 1435.5-1539.5MHz S-Band 2200.0-2399.5MHz ?
Tuning Resolution	500kHz
Frequency Stability	3 ppm
Modulation Type	FM
Modulation Source	Simulator output or external baseband input.
External Input	50 Ω Z_{in}, 2Vp-p for nominal deviation.
Pre-Mod Filtering	5-pole Bessel, eight standard cutoffs of 250kHz, 500kHz, 1, 3, 6, 9, 12, 15MHz. Others substituted by special order.
Deviation	200kHz to 7MHz
Output Power	+10dBm to approximately –60dBm in 5dB steps.

Table 1–2. Mechanical Specifications	
Form Factors	Short “Desktop” PCI (2.2 M33, D32)
Power Dissipation	Tbd

Table 1–5. Environmental Specifications	
Temperature (Operating)	0 to 50 ^{o}C
Temperature (Non-Operating)	-25 to +70 ^{o}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

2.1.1 PCI Cards

(Refers to both PCI and CompactPCI form factors.) The simulator 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 simulator, and accesses to the first byte will return an ASCII identifier string.

The amount of memory space taken up by the card depends on the memory addressing mode recognized by the 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 maps 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.

2.1.2 VME Cards

VME boards operate at a VME address and occupy address space according to address switch settings. The VME card may be in VME Standard or Extended Space, and may be in a "paged" or "flat" addressing mode similar to the PCI form factors. A VME card set for flat mode occupies 512Kbytes of address space. A card set for page mode occupies 64Kbytes. VME addressing is controlled by three DIPswitches S1, S3, and S4. They must be set to the desired space, address, and mode as shown in Table 2–1.

You must also choose a VME interrupt level and set S2 accordingly. S2 Settings other than those shown in Table 2–2 are likely to result in spurious interrupt crashes.

2.2 Physical Installation

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

Table 2–1. VME Address Switch Settings

Switch	On	Off
S1-1	A24 = 0	A24 = 1
S1-2	A25 = 0	A25 = 1
S1-3	A26 = 0	A26 = 1
S1-4	A27 = 0	A27 = 1
S1-5	A28 = 0	A28 = 1
S1-6	A29 = 0	A29 = 1
S1-7	A30 = 0	A30 = 1
S1-8	A31 = 0	A31 = 1
S3-1	A16 = 0	A16 = 1
S3-2	A17 = 0	A17 = 1
S3-3	A18 = 0	A18 = 1
S3-4	A19 = 0	A19 = 1
S3-5	A20 = 0	A20 = 1
S3-6	A21 = 0	A21 = 1
S3-7	A22 = 0	A22 = 1
S3-8	A23 = 0	A23 = 1
S4-1..3	Flat Mode	Page Mode
S4-4..7	Page Mode	Flat Mode
S4-8	VME Extended	VME Standard

Table 2–2. VME Interrupt Switch Settings

Switch	IRQ 1	IRQ 2	IRQ 3	IRQ 4	IRQ 5	IRQ 6	IRQ 7
S2-1	On	Off	Off	Off	Off	Off	Off
S2-2	Off	On	Off	Off	Off	Off	Off
S2-3	Off	Off	On	Off	Off	Off	Off
S2-4	Off	Off	Off	On	Off	Off	Off
S2-5	Off	Off	Off	Off	On	Off	Off
S2-6	Off	Off	Off	Off	Off	On	Off
S2-7	Off	Off	Off	Off	Off	Off	On
S2-8	Off	On	Off	On	Off	On	Off
S2-9	On	Off	Off	On	On	Off	Off
S2-10	On	On	On	Off	Off	Off	Off

2.3 Indicators

Three board ID LED indicators are provided. These are connected to a static register and are intended for use by device drivers in environments where multiple cards are present to identify which board is assigned to which data stream. On desktop PCI and VME cards these indicators are chip LEDs on the board surface.

2.4 Interface

The simulator uses a 44-position female high density subminiature “DB” type connector J1 for I/O. This connector has three rows of pins. The first row of pins is primarily for use by the simulator. The second row is ground pins. A few pins from the third row are used. Pin assignments are shown in Table 2–3. This pin assignment comes with some diffidence, experience advances the premise that any pin assignment will be somehow inadequate.

The pinout is limited by the number of physical I/O pins that can be crammed onto the plate and, face it, the number of coaxial cables that can feasibly terminate into a "B-"size connector backshell. There are possibilities for other I/O’s than the default assignments. Hence the simulator has a patch array E1.

Simulators are normally shipped with pin-pairs strapped together as shown (Default) in Table 2–4. You can reshape the pinout as you will by pulling shunts, and wire-wrapping E1 pins to establish the desired pinout.

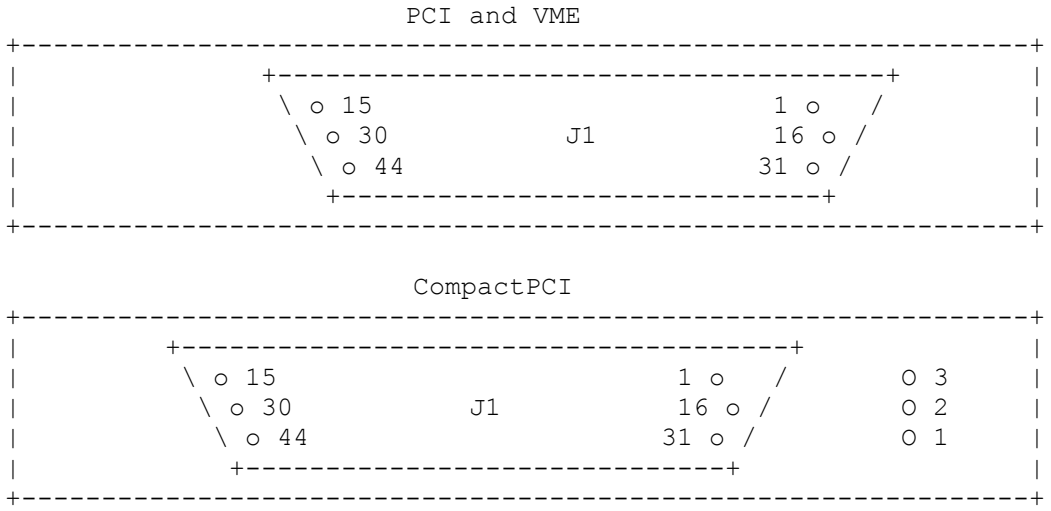

Figure 2–1. Front Plates.

2.6 Notes on Signal Names

The <–> symbol in the “Default” column in Table 2–2 shows where shunts are normally installed when built.

The simulator baseband output is a bipolar signal. It is a simulator PCM data representation and swings about ±2 volts peak-to-peak open-circuit, and about ±1 volt peak-to-peak looking into 75 ohms. The IRIG Time carrier input and output are nominally 1 volt peak to peak low level and 3.3 volts peak-to-peak high level. Logic-type inputs are compatible with TTL levels and terminated into about 130 ohms to 3 VDC. Differential signals are indicated by a trailing

polarity + or − character. Differential outputs are capable of driving RS-422 or TTL-compatible inputs. The differential auxiliary external clock input is pulled slightly asymmetrically so it may be left undriven. It may be driven by

Table 2–3. J1 I/O Connector Pinout

Pin	Signal (Nominal)	Pin	Signal	Pin	Signal (Nominal)
1	E1-2 (Ext Clock In)	16	Ground	31	Not Used
2	E1-4 (PCM Out+)	17	Ground	32	E1-34 (PCM Out−)
3	E1-6 (RUN Status Out)	18	Ground	33	Not Used
4	E1-8 (Slave Clock Out)	19	Ground	34	Not Used
5	E1-10 (Aux Data In)	20	Ground	35	Not Used
6	E1-12 (Clock Out+)	21	Ground	36	Ext Mod In
7	E1-14 (NRZ-L Out+)	22	Ground	37	Not Used
8	E1-16 (Frame Strobe+)	23	Ground	38	E1-36 (Frame Strobe−)
9	E1-18 (Baseband Out)	24	Ground	39	Not Used
10	E1-20 (Clock Out−)	25	Ground	40	Not Used
11	E1-22 (NRZ-L Out−)	26	Ground	41	Not Used
12	E1-24 (Aux Clock In+)	27	Ground	42	Not Used
13	E1-26 (Aux Clock In−)	28	Ground	43	E1-38 (DC Level Out)
14	E1-28 (Sym Clock Out+)	29	Ground	44	E1-40 (Sym Clock Out−)
15	E1-30 (IRIG Time Out)	30	E1-32 (1PPS Out)		

Table 2–4. E1 Patch Header Pinout

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
1	Ext Clock In	2	to J1-1	21	NRZ-L Out−	22	to J1-11
3	PCM Out+	4	to J1-2	23	Aux Clock In+	24	to J1-12
5	RUN Status Out	6	to J1-3	25	Aux Clock In−	26	to J1-13
7	Slave Clock Out	8	to J1-4	27	Sym Clock Out+	28	to J1-14
9	Slave/Aux Data In	10	to J1-5	29	IRIG Time Out	30	to J1-15
11	Clock Out+	12	to J1-6	31	1PPS Out	32	to J1-30
13	NRZ-L Out+	14	to J1-7	33	PCM Out−	34	to J1-32
15	Frame Strobe+	16	to J1-8	35	Frame Strobe−	36	to J1-38
17	Baseband Out	18	to J1-9	37	DC Level Out	38	to J1-43
19	Clock Out−	20	to J1-10	39	Sym Clock Out−	40	to J1-44

a single-ended TTL signal by leaving the "–" end unconnected, albeit at some loss of noise immunity.

The IRIG Time DC Level Out signal is a demodulated representation of the IRIG Time carrier output.

The Frame Strobe is a pulse whose trailing edge coincides with the simulator minor frame boundary. This pulse can be programmed to be high for one bit or for the period of the last word.

Ext Mod In is a direct input to the RF option modulator. The pin has a 75Ω nominal input impedance and expects a signal level of 2V p-p to generate indicated FM deviation.

The Slave signals are for use when this simulator is the master of two simulators are interconnected to create asynchronous embedded formats. Typically Slave Clock Out is connected to an external clock input of a slave simulator and Slave Data In is connected to that simulator's NRZ-L output.

The Slave Data In is also used as a TTL-level external data that can be selected to appear at the PCM and baseband outputs oe as a logic-level modulation source for the RF option.

3 — Programming Information

3.1 General

This chapter is targeted to authors of device drivers, API's, and telemetry applications who will need to know what all the bits do.

The simulator is controlled by an array of eight-bit registers, each identified by a register number. This was done to ease moving the implementation across different form factors and addressing schemes.

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.

On non-PC architectures you may run into Big/Little-Endian issues. Consider yourself to have been warned.

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 PCI9080 chips, set machine registers:

```
AX = 0xB102
CX = 0x9080
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 0x0700B00B, the handle points to a Lumistar simulator 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 – PLX9080 Runtime Registers Memory Address.
Register 0x14 – PLX9080 Runtime Registers I/O Address.
Register 0x18 – 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 PLX9080 runtime registers may be accessed via memory or I/O operations at your convenience. Skip out when you've read them all.

Microsoft operating environments are notorious for erasing the configuration registers of hardware they don't fancy. If the locating procedure places the buffer memory address at zero, that is what happened. The outwardly bizarre machinations described elsewhere for software installation are purposefully designed to circumvent this problem.

7. Increment the index value and try again.

The simulator may be configured to place its memory in protected memory space (“flat mode”) or in real space (“page mode.”) If the simulator is in flat mode the memory occupies 128 Kbytes of contiguous address space and the 3 lsbs of the BANKSWITCH field are ignored. If the simulator is in page mode the memory occupies 16 Kbytes of address space and high-order on-board address bits are supplied by the BANKSWITCH field. If the memory address returned from Configuration register 0x18 is zero, the simulator is in cripple mode. Direct access is not available and only mailbox accesses are allowed.

The simulator occupies 128 bytes of I/O space but only uses the lower part of that space.

3.3 VME Addressing

In page mode, registers appear at base address + 1 + 2 * register number. Memory appears starting at base address + 0x4000.

In flat mode, registers appear at base address + 0x8001 + 2 * register number. The card does not complete VME transactions below this address. Memory appears starting at base address + 0x20000.

VME cards DO NOT support unaligned VME transactiuons with memory.

3.4 Register Summaries

PCI simulators registers appear at the I/O address obtained by adding the register number to the I/O register address. Register bit assignments are summarized in the following tables and discussed in detail later on in this chapter. 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.

If the FM option is installed, additional registers are defined in Tables 3–9 and 3–10.

All register numbers (#) are hexadecimal.

Human exposure to software engineers reveals they refer to these summary tables **very often**. They are deliberately an *aide-mémoire* and many bit names have been shortened for typographical purposes (i. e., to fit page width) and have different (longer) mnemonics elsewhere in this narrative. Such exposure further indicates that slavishly naming bits and later building register values by folding together bit names equated elsewhere (e. g., in a distant .h file) can cross the bound between information-hiding and obfuscation and turn code into alphabet soup. This is completely language-independent. I wish I had an alternate suggestion.

Table 3–1. PCM Simulator Write Register Summary									
Register	**#**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Mainboard ID	00	–	–	–	–	–	LED3	LED2	LED1
Command	10	MREQ	Mread	IACK	IENB	RUN	UACK	PAGE	–
Bankswitch	11	BANKSWITCH				REGS	Bank4	ATTR	A16
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 3–2. PCM 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	Bank4	ATTR	A16
External Status (tbd)	12	–	–	–	–	–	–	XTS1	XTS0
Not defined	13	–	–	–	–	–	–	–	–
Memory Mailbox (Bs = 0xxx)	14	Defined Same As Write							
	15								

3.5 Board Identification

The Mainboard ID register controls three chip LEDs along the top edge. These LEDs are intended for use during system-building. Device drivers or software entities of that ilk are expected to assign a unique non-zero ID to each board and light the lights. This has two purposes:

To show the driver has located and connected to the board.

If multiple like boards are present, to show which is which.

The Identifier register returns a canned ASCII string. Successive reads from this register return iterations of "LS70SIM" bounded by nuls. PCI theory goes that a board should have been identified by this point, but that does not apply to other environments.

3.6 The 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.

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 lsbs 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 decoms insert 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.***

3.6.1 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 3–3. 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 3–4) 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.)

The Mainboard ID register controls the state of three board ID LEDs to allow identification of multiple simulators in the same backplane. The Identifier register returns a null-bound ASCII device identifier string. These registers are active only if the simulator is the primary device on the card.

Table 3–3. Simulator Command Register

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 andW 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 3–4. 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 *again* at the end of a minor frame.
13	FASTMODE	Causes simulator FIFO memory to run in an approximate helf-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 *bit*.

3.6.2 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 3–6) 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 3–5. 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 3–7.
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.

3.6.3 The Clock Generator

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

Below there are notions of "***writing a bit***" to the NCO. Herein, to write a 1, write 0x80, then 0x84 to the Low Exchange register. To write a 0, write 0x00, then 0x04.

Start with the desired output bit rate.

If the bitwise logical AND of 0x0C and the value (chosen according to the output code) written to the Encoder Control register is not zero, multiply the 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 20,000,000.

Or maybe 25,000,000. Or more. A prototype looked solid at 32Mbps but I'm not sure I believe myself about that.

If the result is 262,144 or greater, the DIV field in the Mode register (Table 3–4) 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 2^32 / 1.8*10^8, or 23.8609242. Truncate to an integer.

Write 0x08 to the simulator Bankswitch register and 0x02 to the Low Address register.

Write 0x04 to the low exchange register. Then write 0x02. Then write 0.

Repeat 32 times:
Write a bit equal to the LSB of the rate field. Then shift the rate field one bit to the right, discarding the LSB.
(End repeat)

Set the rate field to 0x00000001.

Repeat eight times:
Write a bit equal to the LSB of the rate field. Then shift the rate field one bit to the right, discarding the LSB.
(End repeat)

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

Table 3–6. 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. Choose one: 00 – Off 01 – RNRZ11 10 – RNRZ15
6	QUIET	Shuts off the PCM output. TTL-level PCM output held low, bipolar PCM output to halfway between 0 and 1. NRZ output not affected.
7		Meaningless.
8	RATE	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.

3.6.4 Communicating With Simulator Memory

The simulator is provided with 128K 32-bit memory words. This memory is dual-ported. When the memory is 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 BANK4 bit.

Table 3–7. Simulator Bankswitch Register

Bit	Mnemonic	Description
0	A16	Provides a 17th highest-order address bit for mailbox memory accesses. This bit is anologous to the PAGE bit.
1	ATTR	Specifies whether mailbox memory accesses connect the exchange register to the data (0) or attribute (1) portion of simulator memory.
2	BANK4	1 = direct-mapped access of attribute memory. 0 = data memory.
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. Bit 7 is anologous to the PAGE bit.

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 3–7.) When read, the bankswitch register returns it current value.

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

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.

3.6.6 Attributes and Data

Each word location has associated data and attributes. The attribute fields are accessed by setting the BANK4 bit for direct-mapped access or the ATTR bit for mailbox accesses. The memory word is formatted per Table 3–8. 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.

Table 3–8. Simulator Word Attributes

Bit	Mnemonic	Description
0	A16	Has meaning only if the IA bit is set. The highest-order address bit for indirect addresses.
1..3	WAVENUM	If IA and WAVE are both one, selects the wave counter used. 0 selects the minor frame counter. 1..7 select the asynchronous wave counters. If PRNS and SLAVE are set, values of 1..7 select PRN pattern generators 1..7.
4	SKIP/FSP	If NARCH = 0, 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 dymanics.
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 *that* 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.

3.6.7 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 ÷ *f* rounded to the nearest two's complement integer.

3.6.8 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 3–9. To access the control register, set REGS, and set the low address register to 16 + generator number.

Table 3–9. PRN Generator Control Register

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

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.

I briefly entertained the notion of making these universal, i. e., allowing the individual taps to be selected rather than choosing a length. That is conceptually simple but quickly grows the design out of an otherwise well-sized FPGA.

3.7 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.

Table 3–10. IRIG Generator Write Register Summary									
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 3–11. IRIG Generator Read Register Summary									
Register	#	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Not Defined	16	–	–	–	–	–	–	–	–
Register 17:	**Adr**								
BCD Frac Seconds	00	$10^{th's}$ Seconds				$100^{th's}$ Seconds			
BCD Seconds	01	DHold	10's Seconds			1's Seconds			
BCD Minutes	02	0	10's Minutes			1's Minutes			
BCD Hours	03	0	0	1's Hours		1's Hours			
BCD Days	04	10's Days				1's Days			
BCD Days	05	Indeterminate						100's Days	

3.7.1 Loading Time

Setting the IRIG generator up formally is a three step process. First, write the generator Control register (Table 3–12) 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 3–10.) 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.

The act of loading the IRIG Generator introduces a discontinuity into its output. This should not be allowed to become a habit.

3.7.2 Reading Generator Time

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 3–11.

Hint: To avoid rollover problems reading time, read all the time registers iteratively until the seconds are unchanged twice in a row. The readout will then be correct (unless you're so slow it takes you more than a second to read the time.)

Table 3–12. 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. Choose one: 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.

3.7.3 Time Generator Data Hold

Table 3–10 shows two registers to load a Data Hold value in seconds and fractional seconds. These 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.

3.8 FM Output Option

The simulator design allows an optional small FM transmitter (approx 100mw output.) If this option is present, additional registers are added to the simulator I/O map as shown in Tables 3–13 and 3–14. The Low Data and High Data registers are merely holding registers. Their contents have no meaning until the RF Command Register is written. None of these registers should be changed when this register is non-zero.

If the FM option is not to be used, write 0x08 to the RF Control Register.

Table 3–13. FM Option Write Register Summary

Register	#	7	6	5	4	3	2	1	0
RF Control	02	XDAT	RFEN	VCO	Xmod	CW	PMF		
Not used	03	–	–	–	–	–	–	–	–
RF Command	04	EEPROMCMD		LO	–	DAC	DEV	Addr[8..9]	
RF EEPROM Address	05	EEPROM Addr [7..0]							
Low RF Data	06	Data [7..0]							
High RF Data	07	Data [15..8]							

Table 3–14. FM Option Read Register Summary

Register	#	7	6	5	4	3	2	1	0
RF Control	02	XDAT	RFEN	VCO	Xmod	CW	PMF		
Not used	03	–	–	–	–	–	–	–	–
RF Status	04	–	EEOP	LO	–	DAC	–	–	–
Not used	05	–	–	–	–	–	–	–	–
Low RF Data Return	06	EEPROM Data [7..0]							
High RF Data Return	07	EEPROM Data [15..8]							

3.8.1 EEPROM Access

A 1Kx16 on-board EEPROM is included with the FM option. Constants associated with setting the output up are stored in the EEPROM. To read from an EEPROM address, ensure theRF Status register = 0. Write the 8 lsbs of the address to the EEPROM Address Register. Write 0x40 + address msbs to the RF Command register. When RF Status = 0 again, read the EEPROM data from the Data Return registers and concatenate the bytes together. EEPROM addresses are assigned as shown in Table 3–15.

Table 3–15. FM Option EEPROM Map	
Address	**Description**
00	Validation. Decimal 74 if FM option is actually present
01	Fc Low in MHz
02	Fc High in MHz
03	Implementation version. Initial release = decimal 100.
04	Reference Divider
05	Programmable Divide Factor
06	Reserved
07	Post-Attenuator offset in dB. Non-zero if fixed attenuation inserted into output.
08	Pre-Mod Filter 0 cutoff in kHz.
09	Pre-Mod Filter 1 cutoff in kHz.
09..15	More Pre-Mod Filter cutoffs 2..7 in kHz. 0 if filter not present. 65535 if unfiltered.
16	Attenuation D/A setting for +10dBm
17	Attenuation D/A setting for +5dBm
18..31	More Attenuation D/A settings for for 0, –5, –10dBm, etc. –1 = invalid.
32,33	0 kHz, 0 counts Dummy Deviation Lookup first entry
34,35	n kHz, n counts Deviation Lookup entry
36..	More Deviation Lookup entries
n, n+1	9000 kHz, 16363 counts Dummy Deviation Lookup last entry
n+2..1023	Reserved

3.8.2 Pre-Mod Filtering

PCM data sent over the airwaves is normally filtered to save bandwidth. The normal process is to set the pre-mod filter f1 (-3dB) cutoff point at about 0.7 of the symbol rate (i. e., bit rate for NRZ codes, twice bit rate for Bi-Phase and Miller codes.) Some folks have experimented with lower cutoffs but they tend to make the data ugly and many bit synchronizers take umbrage to it.

The FM option is normally provided with cutoffs of 250kHz, 500kHz, 1, 3, 6, 9, 12, and 15MHz. These are 5-pole Butterworth filters and can be changed on special order, so the cutoffs actually present should be read from EEPROM addresses 08..15. You would usually choose the smallest cutoff greater or equal to the simulator clock rate (See Step 4 in ¶3.6.3.)

The PMF value number selected needs to be written to the RF Control register PMF field. Setting CW outputs an unmodulated carrier.

3.8.3 Setting the RF Frequency

The transmitter is controlled by a frequency synthesizer. When a frequency is programmed, the transmitter output slews from wherever it is to the target frequency. This can cause much interference if the transmitter is actually connected to an antenna. To reduce the havoc, write 0 to RFEN and 1 to VCOEN in the RF control register before attempting to tune.

Two numbers need to be written to the synthesizer. The first number is called a "reference divider" value. It is a constant you can get by reading EEPROM address 4. Verify RF status = 0, cleave this value into bytes, writing the lsbs and msbs to the Low Data and High Data registers respectively. Then write 0x20 to the RF Command register. When LO clears, you can write the "programmable divider" value.

Choose a programmable divider value by dividing the desired frequency (expressed in kHz) by the programmable divide factor in EEPROM address 5. This is normally 250, so, e. g., to select 2200.5MHz, 2200500/250 = 8802. Round the result to the nearest **even** integer. Verify RF status = 0, cleave into bytes and write to the Data registers. Then write 0x20 to the RF Command register again.

3.8.4 RF Output Enable

When the VCOEN bit is changes from 0 to 1, the transmitter slews to the next frequency tuned to. About eight seconds of warmup are needed before setting RFEN. If the transmitter is to be tuned to a new frequency F_{delta}MHz away from the present frequency after RFEN has already been set, RFEN should be cleared and not set again until at least $F_{delta}/50$ seconds later.

3.8.5 Level Controls

The transmitter deviation and RF output power are set by two fourteen-bit digital-to-analog converters. Each has valid settings in the range [0..16383]. These devices are controlled by first writing a data value to the Data registers. Then, write a selector value of 0x0C (FM Deviation) or 0x08 (Output Power) to the RF Command register.

RF power output level values can be read from EEPROM (see Table 3–15) provided we get around to actually calibrating them.

Deviation calibration data is stored in pairs of EEPROM locations (see Table 3–15) The even-numbered location holds a calibrated deviation in kHz. The succeeding odd-numbered data value is the setting for that deviation. Calibration points are generated starting with a minimum deviation of 150kHz, in steps ascending by about 10% each. To set a specified deviation, search the frequency values in the table and find the smallest entry that is larger than the desired deviation. The table has dummy entries at the beginning and end to allow interpolation between settings *if you're that fussy*.

3.8.6 External Data Input

Setting the XMOD bit disconnects the simulator output from the RF circuitry. In its place, the RF output is modulated by the EXTMOD input (J1-36.) This input expects an externally supplied 2V p-p signal for indicated deviation.

Setting XDAT causes a similar function except the source of external modulation is a TTL signal coming in at J1-5. Setting XDAT also causes this signal to appear at the simulator PCM and baseand outputs. The NRZL output is unaffected.

When the *changement du jour* to add this feature came in, I asked whether this signal was a logic level or not and was told, "Yes." Grrrrrr.

3.9 DMA (PCI Only)

The PLX PCI9080 includes two DMA controllers that permit rapid data movement. The implication is to use them typically to move data from an archive file into on-board memory. If you plan to write for the DMA controller you may have need for the PLX9080 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 PCI9080 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 have one continuous physical buffer in system memory holding the output data. 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 PCI9080 allows the chaining table to be stored either in local (i. e., on-board) memory or PCI (i. e., elsewhere in the system) memory. If you can reserve enough space you can place your chaining tables in on-board memory.

3.9.1 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. Local memory appears as flat memory starting at address zero whether the board is in page or flat 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 0 to move data from system memory to the on-board 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.

3.9.2 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 PCI9080 descriptor register. This is not recommended for two reasons. First, because it creates the complication of a special case, and also because the PCI 9080 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 PCI9080 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.

3.9.3 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. Only bits 0..4 are defined.

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.