1771-01 Application Notes

INTRODUCTION

The FD1771-01 Floppy Disk Formatter/Controller is a MOS/LSI device designed to ease the task of interfacing the 8″ or 5¼ (mini-floppy) disk drive to a host processor. It is ideally suited for a wide range of microprocessors, providing an 8-bit bi-directional interface to the CPU for all control and data transfers. Requiring standard + 12, ±5V power supplies, the 1771 is available in ceramic or plastic 40 pin dual-in-line packages.

The 1771 has been designed to be compatible with the IBM 3740 standard. This single-density Frequency Modulated (FM) recording technique, records a clock bit between a data bit serially on each track. Figure 1 illustrates how a HEX "D2" is recorded. Note that when the data bit to be written is zero, no pulse or flux transition is recorded. For the 8″ drive, there are 77 tracks, with 26 sectors on each track. Each sector contains 128 bytes of data. Although there is no "standard" format for the mini-floppy, most manufacturers utilize either 35 or 40 tracks per side, wtih 16 sectors of 128 bytes each per track. Both the 8″ and 5¼″ formats must be soft-sectored, i.e., there are no physical holes to denote sector locations. The hard-sectored disk has been losing popularity, mainly due to the fact that the sector lengths cannot be increased.

Being soft-sector compatible, the 1771 must know where each sector begins on the track. This is performed by using Address Marks. These bytes are recorded on the disk with certain clock pulses missing, and are unique from all other data and gap bytes recorded on the track. Six distinct Address Marks can be used:

Description	Data	Clock Pattern
Index Address Mark	FC	D7
ID Address Mark	FE	C7
Data Address Mark	FB	C7
User defined	FA	C7
User Defined	F9	C7
Deleted Address Mark	F8	C7

The two "User Defined" Address Marks are unique to the 1771, and do not appear in the IBM 3740 standard. These Address Marks can be used to define the type of data i.e., "object" or "text" data, alternate sector data, or any other purpose the user chooses.

PROCESSOR INTERFACE

The 1771 contains five internal registers that can be accessed via the 8-bit DAL lines by the CPU. These registers are used to control the movement of the head, read and write sectors, and perform all other functions at the drive. Regardless of the operation performed, it must be initiated through one or more of these registers. They are selected by a proper binary code on the A0, A1 lines in conjunction with the $\overline{RE}$ and $\overline{WE}$ lines when the device is selected. The registers and their addresses are:

$\overline{CS}$	A_1	A_0	$\overline{RE} = 0$	$\overline{WE} = 0$
0	0	0	STATUS REG	COMMAND REG
0	0	1	TRACK REG	TRACK REG
0	1	0	SECTOR REG	SECTOR REG
0	1	1	DATA REG	DATA REG
1	X	X	Deselected	Deselected

Command Register: This is a write-only register used to send all commands to the 1771.

Status Register: This is a read-only register that must be read at the completion of every command to determine whether execution was successful. It may also be used to monitor command execution, and to sense when data is required by the drive for read or write operations.

Track Register: This R/W register holds the current position of the R/W head.

Sector Register: This R/W register holds the desired sector number for read and write commands.

Data Register: This R/W register contains the data to be read or written to a particular sector.

INTERRUPTS

There are two INTERRUPT lines for CPU use. These are the DRQ (Data Request) and INTRQ (Interrupt Request). These are active high, open drain outputs and require a pull-up resistor of 10K or greater to +5V. Both of these signals also appear in the status register as the Busy (INTRQ) and the data request (DRQ) bits. The user has the option of utilizing these hardware lines for system interrupts, or through

software by polling the status register. The choice is dependent upon the particular microprocessor and support hardware of the system.

INTRQ: This line is used to signify the completion of any command. It is reset low when a new command is loaded into the command register, or when the status register is read.

DRQ: This line is active high whenever the data register requires servicing. During a read command, it signifies that the data register contains a byte of data from the disk and may be read by the CPU. During a write command, it signifies that the data register is empty and may be loaded with the next byte to be written on the disk. The DRQ line is reset whenever the data register is read or written to. It is also reset when a new command is loaded into the command register, providing the new command is not a Forced Interrupt, and the 1771 is not busy (Busy Bit = 0).

WRITE SECTOR

With the use of the WRITE SECTOR command, the CPU can access any desired sector(s) in a track. Prior to loading this command, the R/W head of the drive must be positioned over the specific track. This can be first accomplished with the use of any of the Type I commands. Once positioned, the CPU must load the desired sector number into the sector register, then issue the command. The head will load, and the 1771 will begin searching for the correct ID field. If the correct sector and track is not found within 2 revolutions of the disk, the RECORD-NOT-FOUND bit will be set in the status register, and the command will be terminated. Once found, the 1771 will issue a DRQ in request of the first data byte to be written. Once the data register is loaded, the 1771 will issue a DRQ for each byte to be recorded, until the entire sector is written. For the 8" drive, the user must load the data register 24 microseconds after a DRQ is generated. Failure to meet this time will cause the lost data bit to be set, and a byte of zeros substituted and written on the disk.

READ SECTOR

The READ SECTOR command functions in much the same way as the WRITE SECTOR command. The sector register must again be loaded with the desired sector number, before the read command can be loaded. After the ID field has been found, the 1771 will begin generating DRQ's, with the data register being loaded with each byte of the sector field. For the 8" drive, the user must read the data register at least 26 microseconds after the DRQ is generated. Failure to meet this time will cause the lost data bit to be set in the status register, while the next assembled byte will overwrite the contents of the data register.

Both the Read and Write sector commands also contain an "m" flag for accessing multiple sectors. The sector register is incremented internally after each sector is read or written to. Eventually the sector register will exceed the physical number of sectors on the track. The user can either issue the Forced Interrupt command after the last sector, or wait for the 1771 to interrupt out. In the latter case, the RECORD-NOT-FOUND status bit will be set.

FLOPPY DISK INTERFACE

For the most part, the actual Floppy Disk Interface will consist mainly of Buffer/Drivers. Most drives manufactured today require an open collector TTL interface, with appropriate resistor terminal networks. Figure 2 shows the interface of the 1771 to a Shugart SA400 Drive. Aside from the data seperator, the interface consists mainly of 7438's and 7414 TTL gates. A 9602 one-shot is used for the desired head load delay. In this illustration, the 6800 microprocessor is used via a 6820 Peripheral Interface Adapter to control all functions of the 1771. Similarly, other parallel port devices (such as the 8255 for 8080 systems) can be used for the interface, or the 1771 may simply be tied directly to the systems data bus and control lines, providing TTL loading factors are observed.

DATA SEPERATION

The internal DATA SEPERATOR of the 1771 can be used by tying the $\overline{XTDS}$ line high, and supplying the combined clock and data pulses on the FD data line. In order to maintain an error rate better than 1 in 10^8, and external data seperator is recommended.

Since the 1771 system clock is at 2 MHz, this allows for a 500 ns resolution. The internal data window will move 500 ns with respect to the incoming data bit. On the inner tracks of the drive, the bit shift is more severe and may occasionally cause a data or clock bit to fall outside of this data window. Since the 1771 will perform up to 5 retries, this error rate may be acceptable for some applications.

When the $\overline{XTDS}$ line is forced low, the 1771 will accept seperated clock and data on the FDCLOCK and FDDATA lines. Figure 3 illustrates the timing of these signals. The actual FDCLOCK and FDDATA lines may be reversed; the 1771 will determine which line is clock and which is data when an Address Mark is detected. This feature greatly simplifies the design of the data seperator.

Figure 4 illustrates the Phase-Lock Loop method for data seperation. The circuit operates at 8 MHz, or 32 times the frequency of a received bit cell. The MC4024 VCO is used to supply the nominal clock frequency. The first 74LS161 counter provides a divide by 16 frequency and a carry to one side of the MC4044 phase detector. The other input of the MC4044 is tied to another 74LS161 counter which is affected by the incoming data stream. The output of

the phase detector is a signal proportional to the differences of the incoming pulses. This is then fed through a low pass filter, and to the input of the MC4024 to adjust the output frequency. Figures 5 thru 8 illustrate other types of data seperators.

These employ the "Counter Seperator" techniques and are quite different from the Phase-Lock-Loop method. With the addition of "One-Shot" delay element or an input clock, most of the complexity of the PPL circuit can be eliminated.
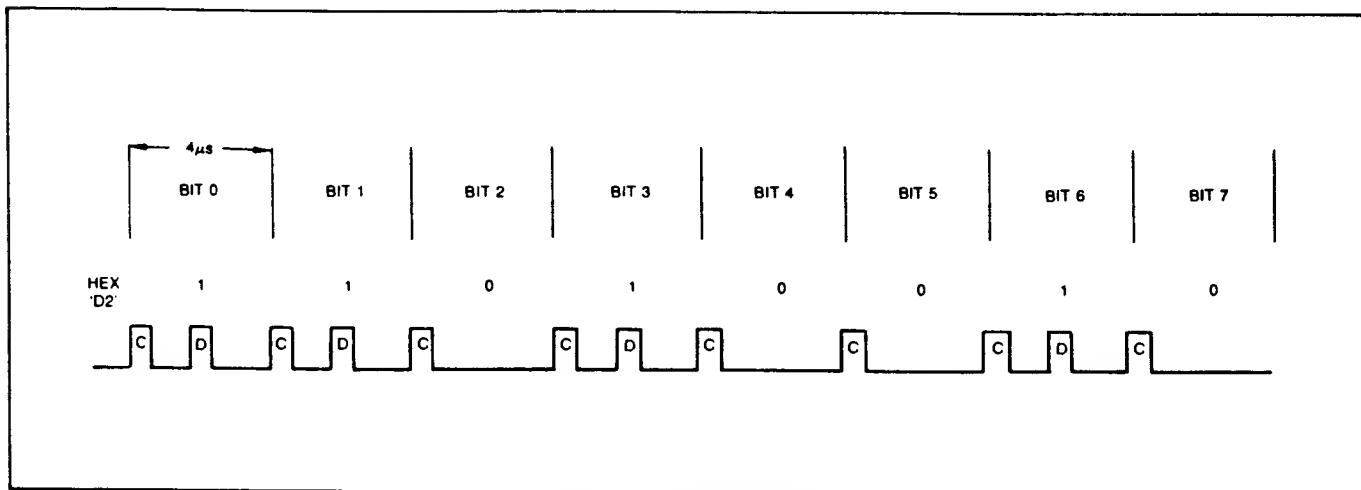

FIGURE 1. FM RECORDING.
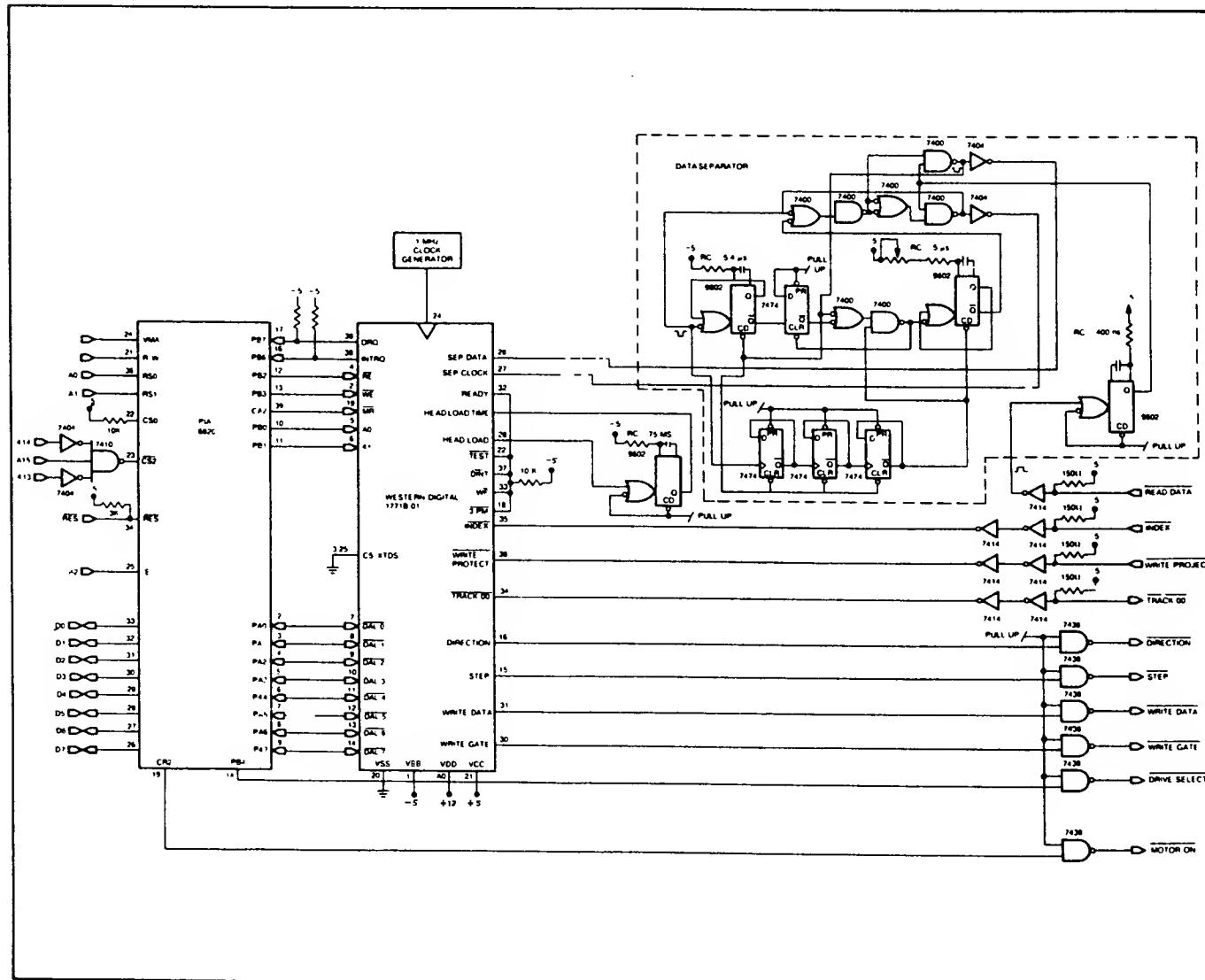
FIGURE 2. 1771 TO SHUGART SA400 DRIVE
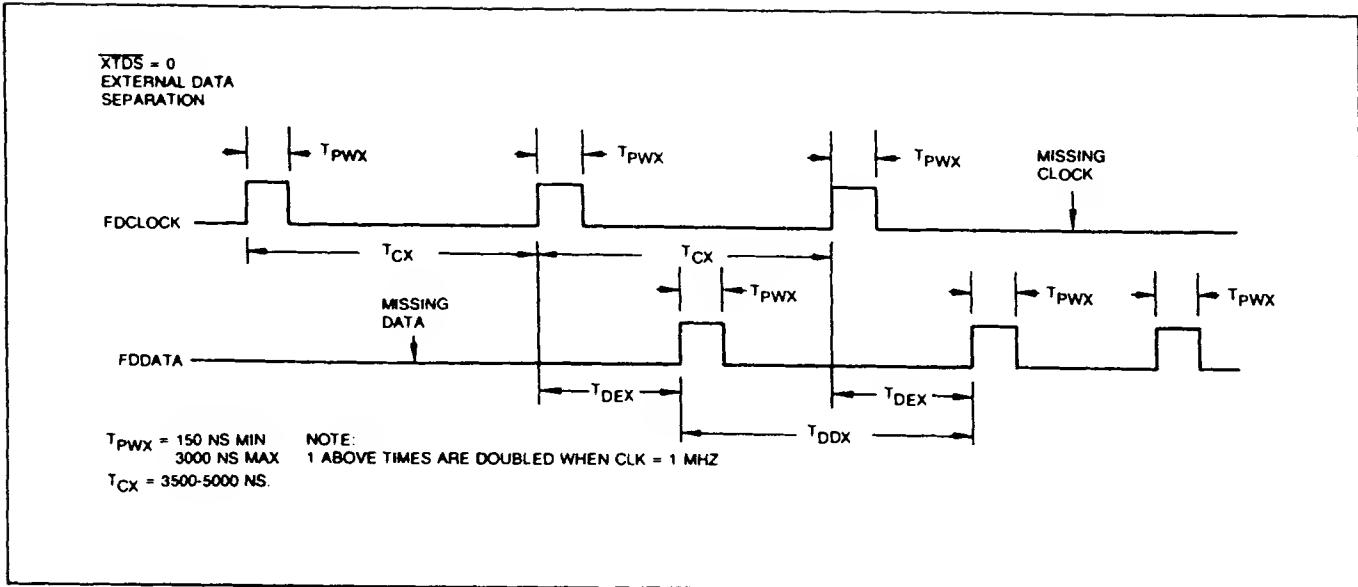
FIGURE 3. EXTERNAL DATA SEPERATOR TIMING.
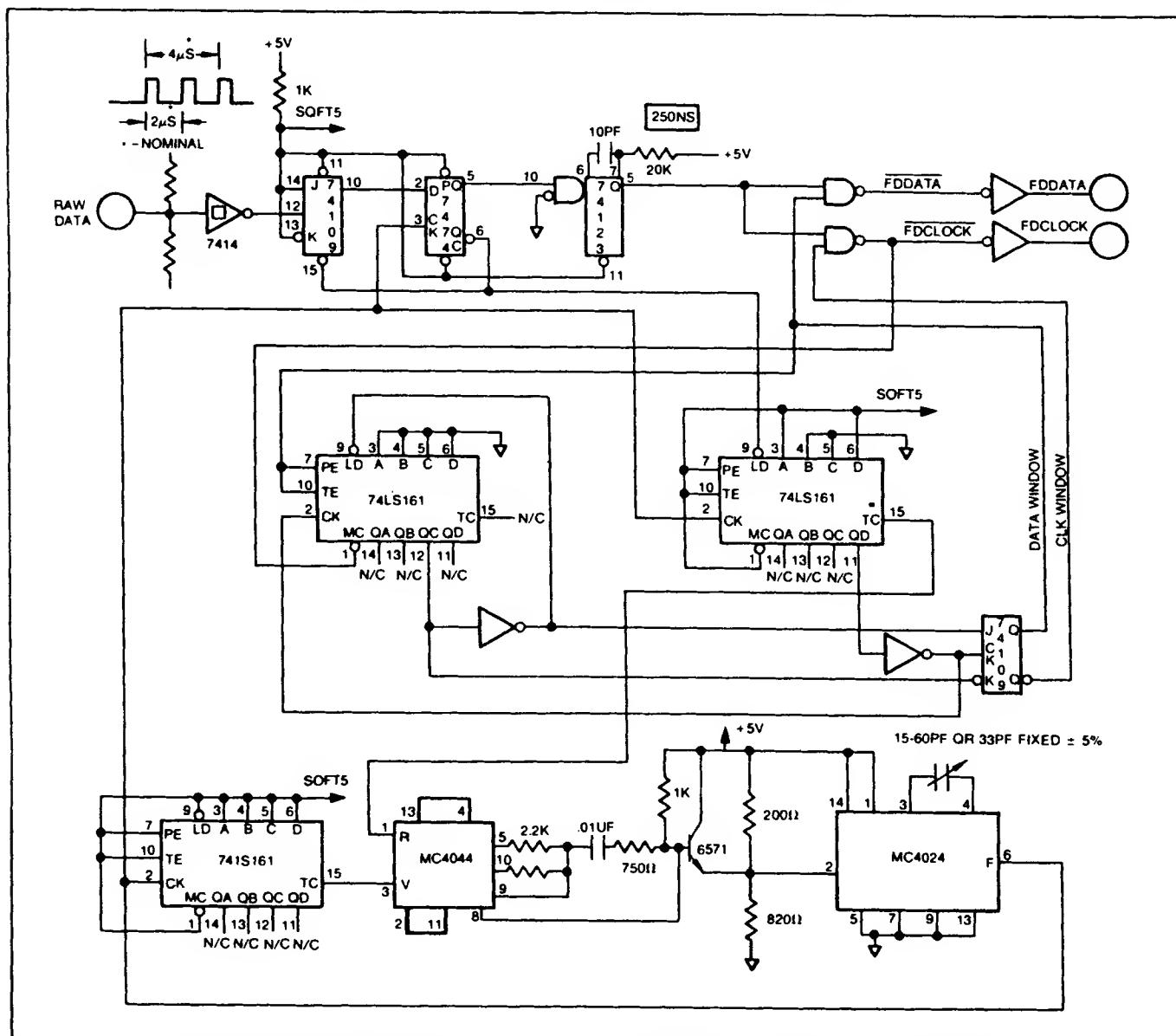
FIGURE 4. CIRCUIT PROVIDED COURTESY OF MOTOROLA AND ICOM CORPS.

4
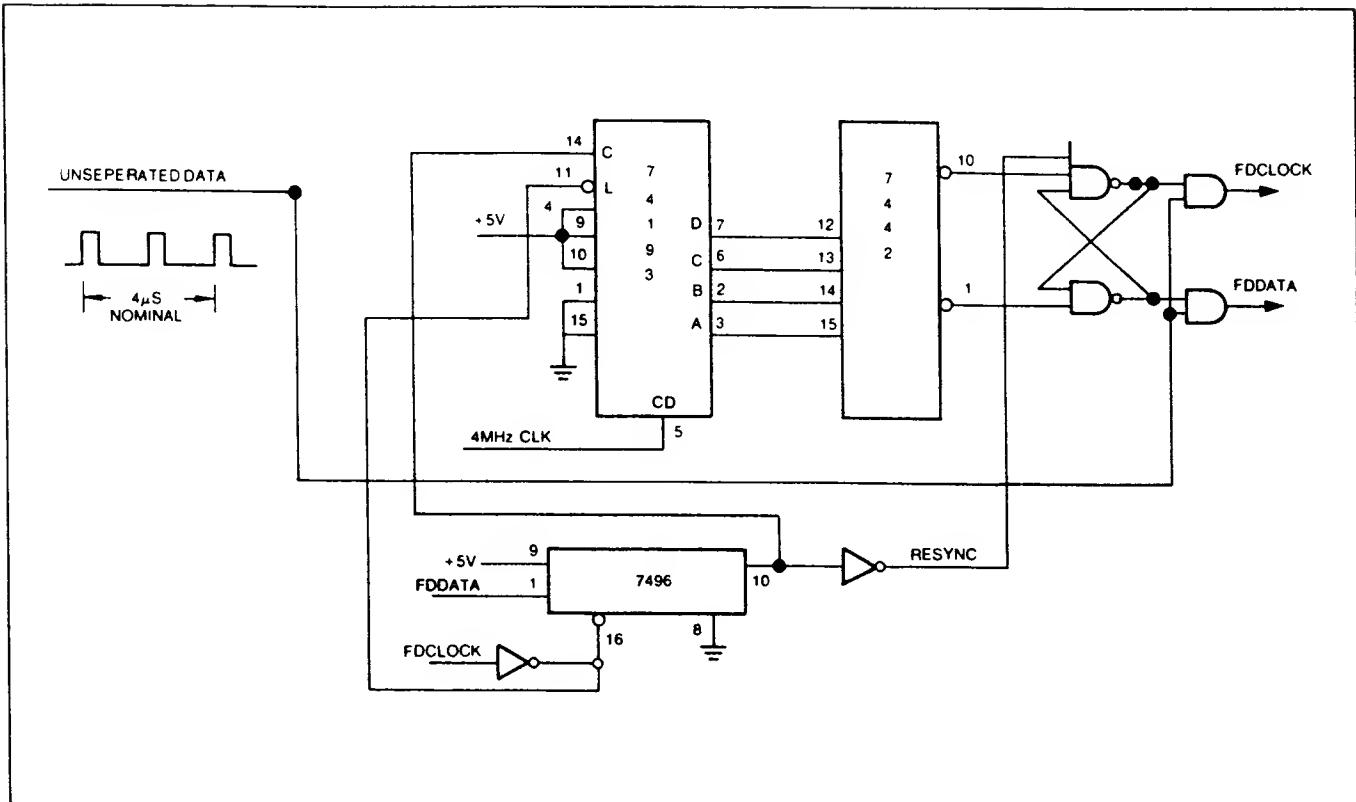
FIGURE 5. CIRCUIT PROVIDED COURTESY OF PROCESSOR APPLICATIONS LTD.
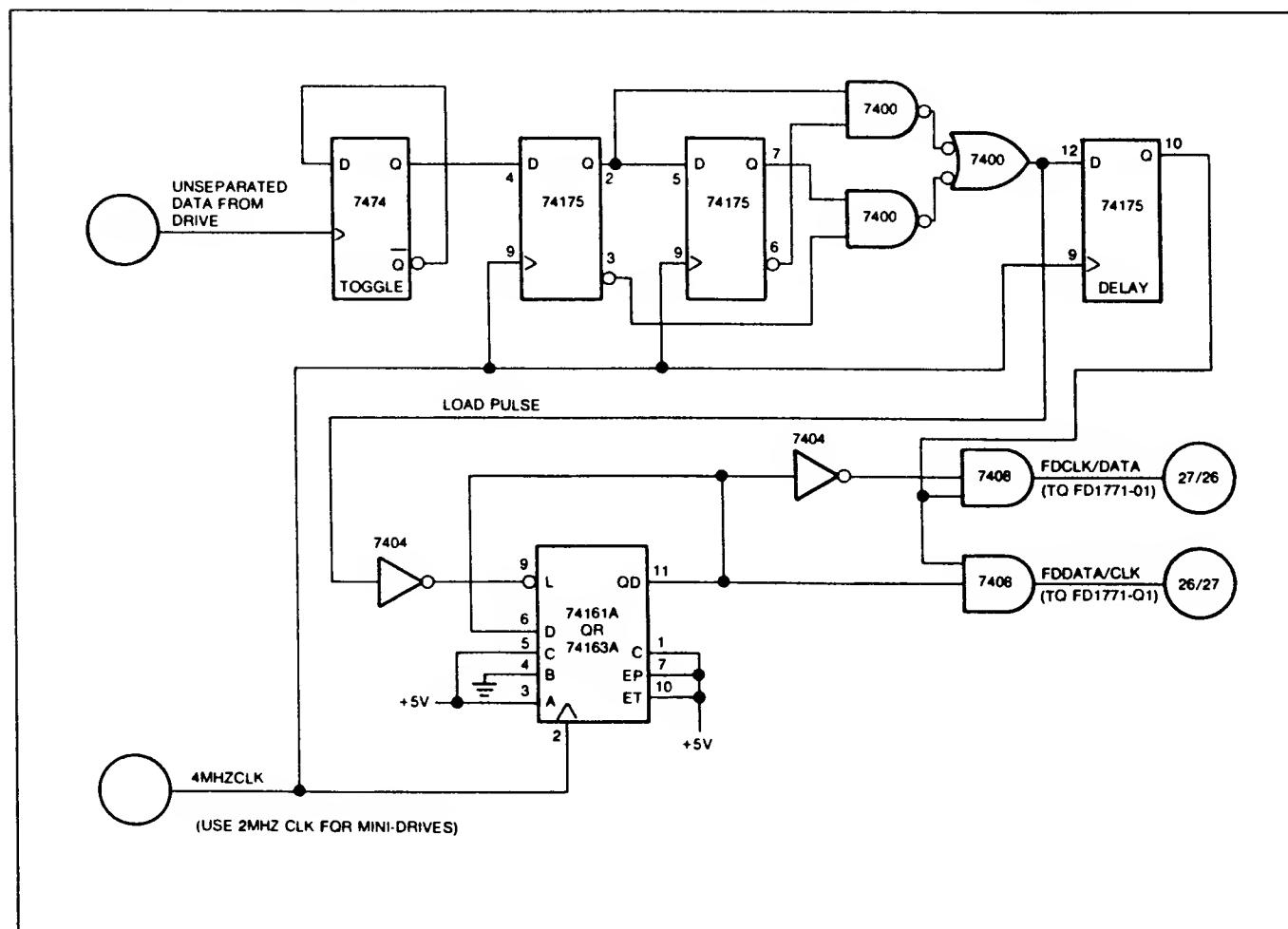
FIGURE 6.
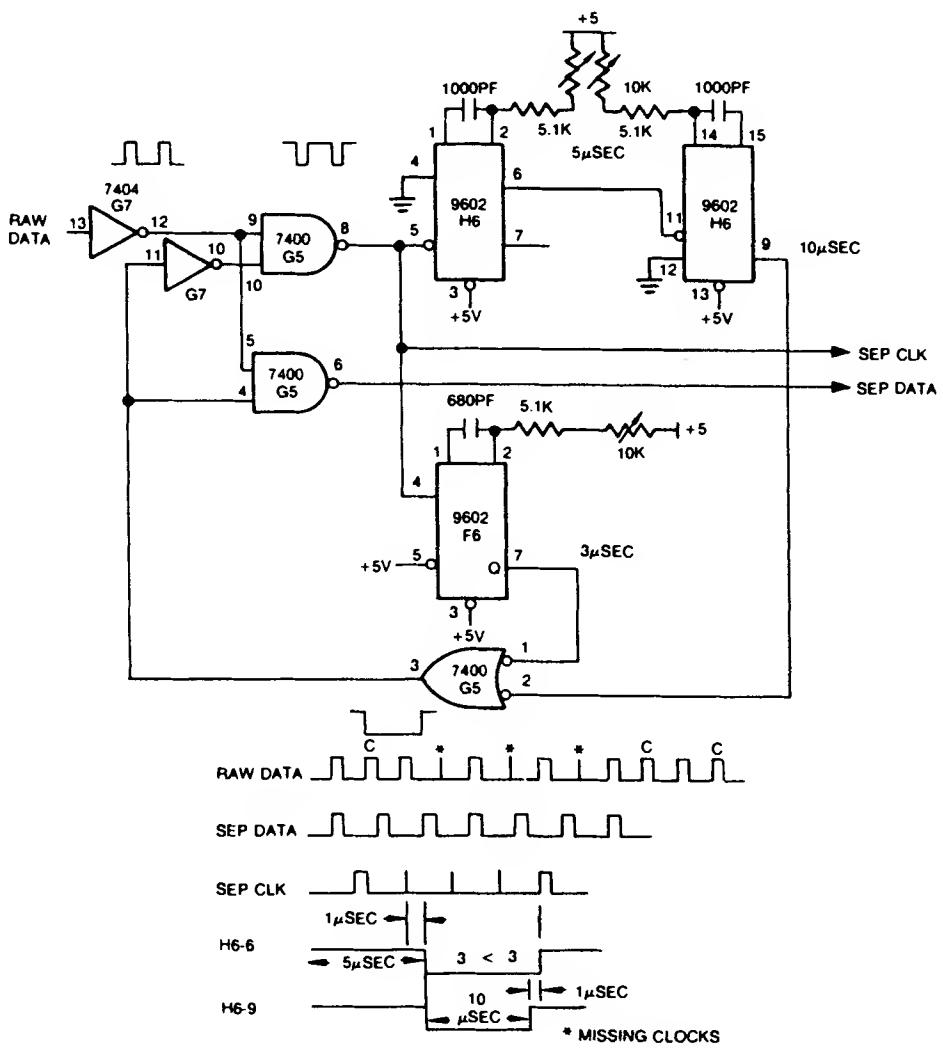
**FIGURE 7. CIRCUIT PROVIDED COURTESY OF ACUTEST CORP.**
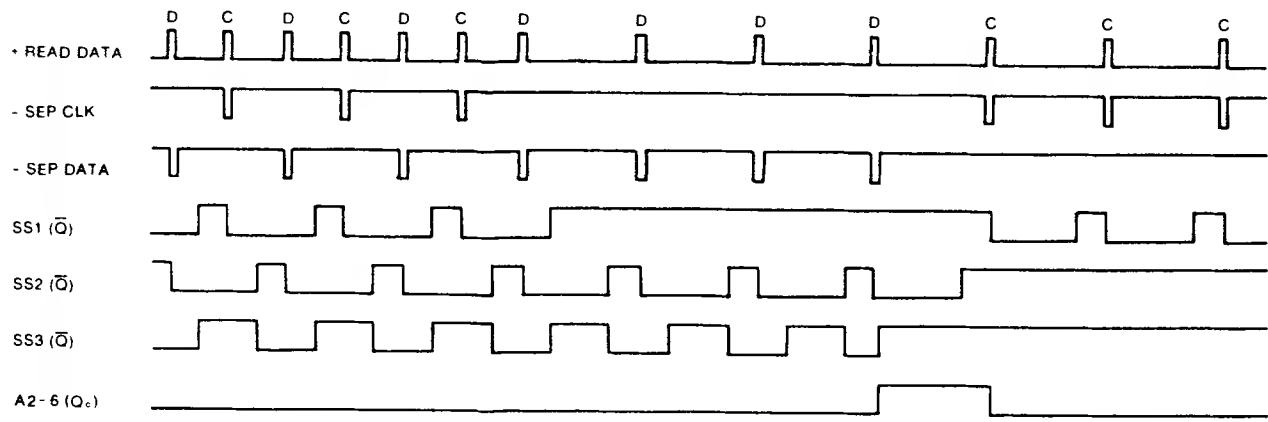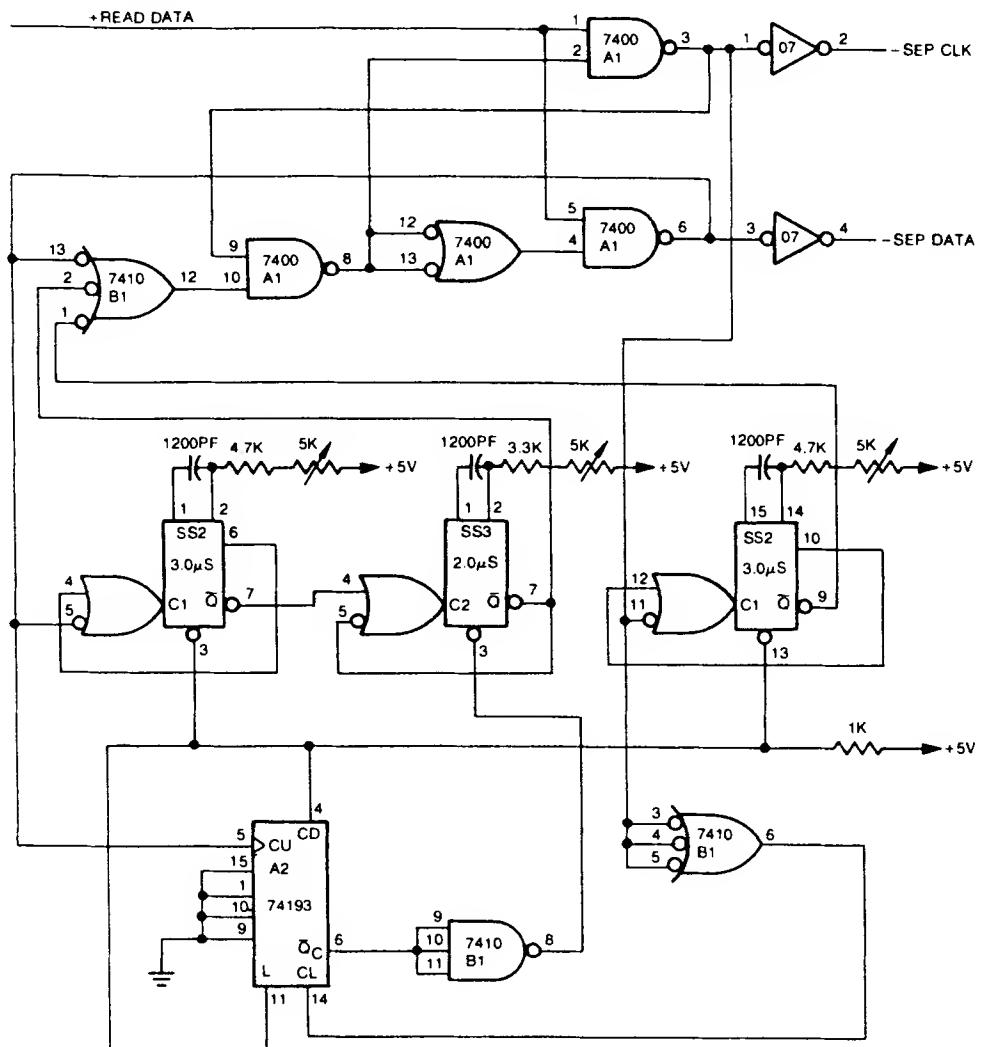
FIGURE 8. CIRCUIT PROVIDED COURTESY OF SHUGART ASSOCIATES.

7

All diagrams within this applications note are shown for illustrative purposes and may not necessarily reflect the total logic for implimentation.