Tasmania Department of Mines—Report 1989/20

Control of the Teac tape replay unit

by R. J. Sedgman

Abstract

A Teac tape replay unit was installed to transfer SIE down-hole logger data from cassette tape to a standard 9-track computer tape for archive and data manipulation. A microcomputer was employed to allow more reliable control and less overhead for the minicomputer.

INTRODUCTION

To allow data from the SIE down-hole logger to be archived and distributed in a standard format the logger tapes must be transcribed onto standard 9-track computer tape. The method adopted to transfer data recorded on the SIE cassette tapes to a minicomputer was via the TEAC MT-2 tape transport system. Direct connection and control was not possible, due to the nature of the Teac tape drive and the minicomputer, so the control and transfer of data is handled via a microcomputer.

CHAPTER 1

Overview of the Teac tape replay system

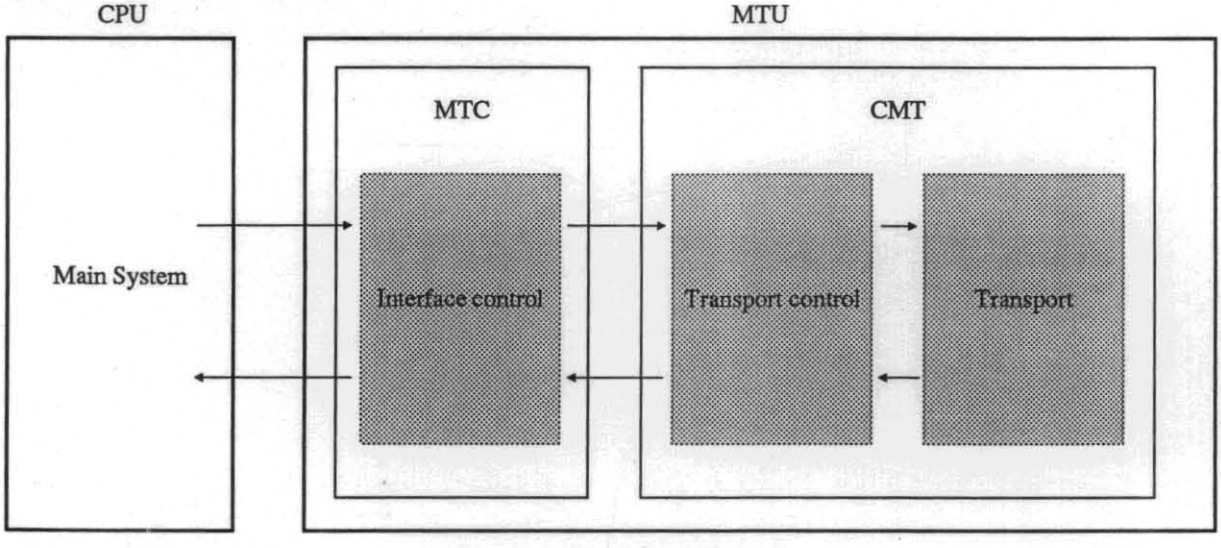
INTRODUCTION

The TEAC MT-2 cassette magnetic tape unit is designed to replay cassette tapes which comply with ISO, JIS, ECMA and ANSI standards. It is designed to allow interchangeability of cassette tapes with tape units with the above standards. It has a basic command set which allows control of the tape unit and interrogation of internal registers. Connection to the unit is via a 50-pin flat connector, which also supplies power from an external source.

Control of the Teac tape unit is via a microcomputer. This computer converts commands from a minicomputer to a 16 bit control line linked to the Teac tape unit via the 50-pin flat connector. A simple control word is given to the microcomputer from the minicomputer and it is converted to control signals for the Teac tape unit. The microcomputer then returns any relevant data to the minicomputer for manipulation.

The minicomputer is used to give simple commands to what it sees as the tape unit; it then waits for relevant data, followed by the current status of the tape unit, to act upon any errors which may have occured. All error handling is carried out by the minicomputer control program.

Figure 1.1. *Overview of system operation.*



CONTROL FLOW OF THE CMT

CHAPTER 2

The Teac Tape Replay Unit

INTRODUCTION

The Teac Tape Replay Unit is basically used to extract data from a cassette tape and transfer this data to a controlling computer. The details on how this system functions are given in as much simplistic detail as possible in the following chapters. A knowledge of low level language programming is essential to understand the operation of controlling the unit. It has been found that if these instructions are not adhered to by the letter, control of the unit is virtually impossible.

Some of the features of the unit have been deliberately left out, as they do not apply to this application. For details on the range of capabilities refer to the TEAC MODEL MT-2 Instruction Manual.

GENERAL DESCRIPTION

The MTU (magnetic tape unit) is divided into three parts according to their functions. These parts are transport, transport electronics, and interface control. In the following, the transport and transport electronics is referred to as the CMT, and the interface control is referred to as the MTC.

The CMT is controlled by a main system (called the CPU), in this case a minicomputer via a microcomputer. All error handling is performed by the minicomputer program and not by the microcomputer interface.

Logic levels shown in this section of the manual are "1" (ground, 0 volts) and "0" (+5 volts). An example is a clock pulse shown to go high then low when the resulting physical output control voltage would be 0 volts then +5 volts.

CONTROL METHOD

The MTC has eight registers to be accessed by the CPU. Each of these registers has particular significance for the control of the MTU by selecting different functions using these registers. There are 8 registers in the MTC accessible by the CPU; detailed descriptions are given later.

Register 0	.....Data buffer register (DBR)
Register 1	.....Word counter (WDC)
Register 2	.....Command register (CDR)
Register 3	.....Mode register 0 (MDR0)
Register 4	.....Cassette status register (CSR)
Register 5	.....Error status register (ESR)
Register 6	.....Interrupt status register (ISR)
Register 7	.....Mode register (MDR)

Figure 2.1

Control from the microcomputer is performed via a 16 Bit word, divided into two parts. The top 8 Bits are used for selection of the registers, clock, reset etc. and the bottom 8 Bits for data. This data may be information from the cassette itself, or used for writing/reading information to/from a selected register.

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset	Clock	Select	Read	Interrupt request	Register select 2	Register select 1	Register select 0

Figure 2.2. *Top 8 Bits*

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Data 7	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1	Data 0

Figure 2.3. *Bottom 8 Bits*

Input/Output signals.

(1) *Data 0–Data 7 Input/output signals.* Eight signal lines to transmit data between the microcomputer and MTU.

(2) *Select Input signal.* Used to allow writing/reading to/from the internal registers.

(3) *Register select 0-2 Input signal.* Signals to select the registers in the MTU. Any of the eight registers may be selected with combinations of these three signals.

(4) *Read Input signal.* A signal to determine the direction of data transfer for byte Data 0–Data 7.

(5) *Clock Input signal.* Used to determine the data transfer timing for all data transfers and input signals, except the reset command.

(6) *Reset Input signal.* A signal to reset all the registers in the controller of the MTU. 2 µsecond or longer signal. '1' level resets.

(7) *Interrupt request.* Output signal. A request from the MTU to the microcomputer signalling an Interrupt is required. Cleared after reading of the Interrupt Status Register.

Internal Registers

Table 2.1 shows the registers in the MTU which are accessible to the microcomputer. One of the eight registers is selected by the microcomputer through the three register select signals. The data transfer direction is determined by the read signal from the microcomputer control program.

RS2	RS1	RS0	REGISTER NOS.	REGISTERS	ABBR.	DIRECTION
0	0	0	R0	Data Buffer Register	DBR	Input/Output
0	0	1	R1	Word Counter	WDC	Input/Output
0	1	0	R2	Command Register	CDR	Input
0	1	1	R3	Mode Register 0	MDR0	Input
1	0	0	R4	Cassette Status Register	CSR	Output
1	0	1	R5	Error Status Register	ESR	Output
1	1	0	R6	Interrupt Status Register	ISR	Output
1	1	1	R7	Mode Register	MDR1	Input

(*Input* is defined as input to the Teac drive unit, *Output* is defined as output to the microcomputer)

Table 2.1. *Function of Internal Registers.*

Function of Internal Registers

(1) DBR (DATA BUFFER REGISTER) INPUT/OUTPUT, R0

				Signification
RS0	0	D0-b0		DATA0 (Least Significant Bit : LSB)
RS1	0	D1-b1		DATA1
RS2	0	D2-b2		DATA2
		D3-b3		DATA3
		D4-b4		DATA4
		D5-b5		DATA5
		D6-b6		DATA6
		D7-b7		DATA7 (Most Significant Bit : MSB)

Read data from the cassette tape is transferred through this register. The contents of this register can be read out to the microcomputer at any time. Data transfer is a byte at a time.

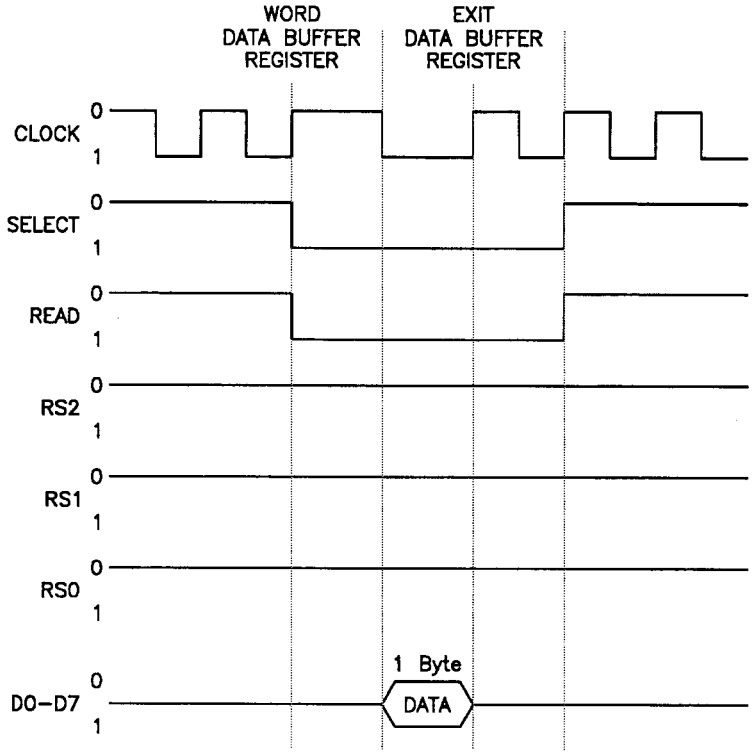


Figure 2.4

(2) WDC (WORD COUNTER) INPUT/OUTPUT, R1

				Signification
RS0	1	D0-b0		2^0
RS1	0	D1-b1		2^1
RS2	0	D2-b2		2^2
		D3-b3		2^3
		D4-b4		2^4
		D5-b5		2^5
		D6-b6		2^6
		D7-b7		2^7

WDC is a register effective only for the execution of control commands for data transfers. The number of Bytes to be transferred is determined by this register. The register is written every time before the input of the control command for the data transfer.

(a) The number of data values to be transferred is designated in Bytes, in this case 252 Bytes are required.

(b) The register is a counter and the contents are decremented by each data transfer request from the MTC to the microcomputer. Therefore it is easy to find out how much data has been transferred by reading the contents of this register.

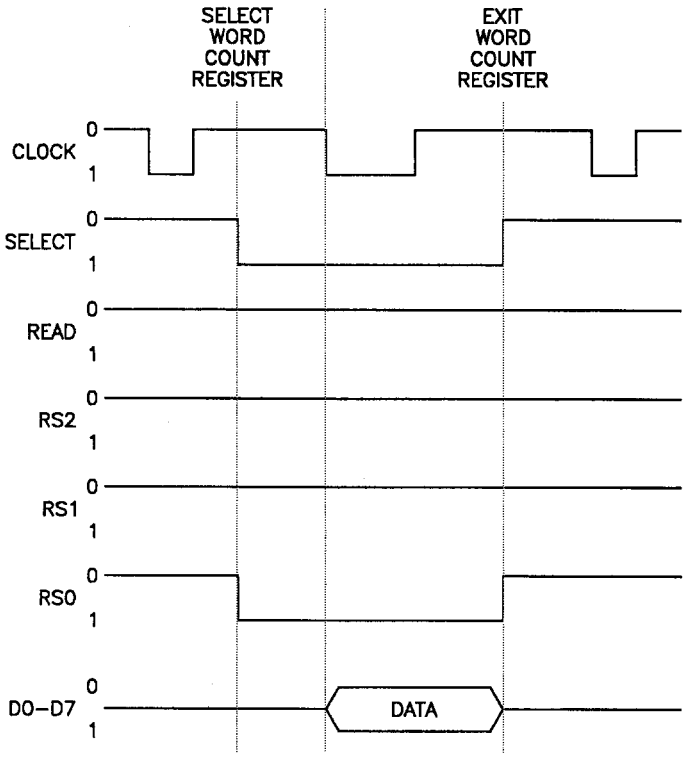


Figure 2.5

(c) Writing and reading into or from this register can be done anytime.

(3) CDR (COMMAND REGISTER) INPUT, R2

			Signification
RS0	0	D0-b0	CMD0 (Command Code 0)
RS1	1	D1-b1	CMD1 (Command Code 1)
RS2	0	D2-b2	CMD2 (Command Code 2)
		D3-b3	CMD3 (Command Code 3)
		D4-b4	
		D5-b5	
		D6-b6	IM0 (Interrupt mask flag 0)
		D7-b7	IM1 (Interrupt mask flag 1)

By writing a control command code to Bits CMD0 (b0) to CMD3 (b3), the MTU is instructed to start its operation. At the same time, write IM0(b6) and IM1(b7) to mask or not mask the interrupt during the execution of the control command.

Once a control command is written to this register, a new control command will not be accepted until the completion of the previous command.

(a) Command Codes (CMD0–CMD3)

Table 2.2 shows the control command codes; detailed descriptions of the control commands are given in the section Command Code Registers.

(b) Interrupt mask flag 0 (IM0)

This flag is set to 1 to disable the Interrupt request flag, as data transfers use the Data Available (DA) flag (explained later) instead.

(c) Interrupt mask flag 1 (IM1)

This flag is set to 0 to enable the control command end flag (explained later) to be used for MTU control.

No.	Command codes				Hexa-decimal notation	Control commands	Abbr	Data transfer
	CMD3	CMD2	CMD1	CMD0				
1	0	0	0	0	0	NO OPERATION	NOP	X
2	0	0	0	1	1	WRITE ONE BLOCK	WRT	O
3	0	0	1	0	2	WRITE TAPE MARK	WTM	X
4	0	0	1	1	3	ERASE	ERA	X
5	0	1	0	0	4	READ ONE BLOCK	RDL	O
6	0	1	0	1	5	READ ONE BLOCK	RDH	O
7	0	1	1	0	6	SKIP ONE BLOCK	SKP	X
8	0	1	1	1	7	REVERSE ONE BLOCK	REV	X
9	1	0	0	0	8	SET LOAD POINT	SLP	X
10	1	0	0	1	9	SET LOAD POINT WITH ERASE	SLE	X
11	1	0	1	0	A	REWIND START	REW	X
12	1	0	1	1	B	NO OPERATION	NOP	X
13	1	1	0	0	C	SEARCH TAPE MARK	STM	X
14	1	1	0	1	D	HIGH SPEED SEARCH	HSS	X
15	1	1	1	0	E	NO OPERATION	NOP	X
16	1	1	1	1	F	NO OPERATION	NOP	X

Note: The mark 'O' in the column of data transfer indicates that the command accompanies data transfer, and the mark 'X' indicates the command does not accompany data transfer. Also the commands No. 5 and No. 6 are the same command (READ ONE BLOCK). The command codes for them are only distinguished by the abbreviated command name.

Table 2.2 *Control commands and codes.*

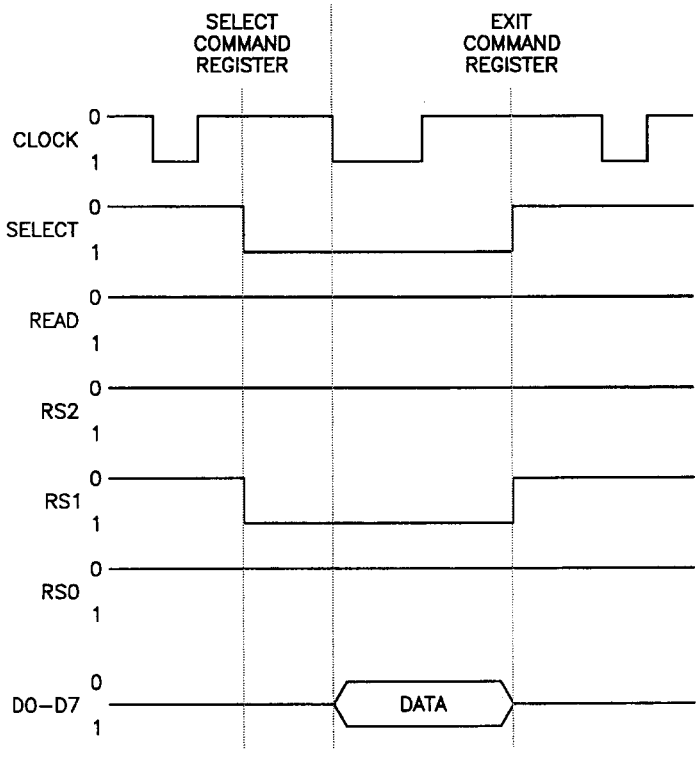


Figure 2.6

(4) MDR0 (MODE REGISTER 0) INPUT, R3

Not used (only needed for DMA transfers and software)

(5) MDR1 (MODE REGISTER 1) INPUT, R7

				Signification
RS0	1	D0-b0	0	'0'
RS1	1	D1-b1		CFRE (CONTROLLER FREE REQUEST (CFR) ENABLE)
RS2	1	D2-b2		CFRIM (CFR INTERRUPT MASK FLAG)
		D3-b3		
		D4-b4		
		D5-b5		
		D6-b6		
		D7-b7		

Bit b0 must be '0'

(a) CFRE (controller free request enable)

If this flag is set to '1', CFR (controller free) is set when the MTC becomes free. The condition of MTC free means that no control command is given and the MTU completely stops.

Used only for controlling the unloading of a cassette from the MTU.

(b) CFRIM (Controller free interrupt mask flag)

Not used.

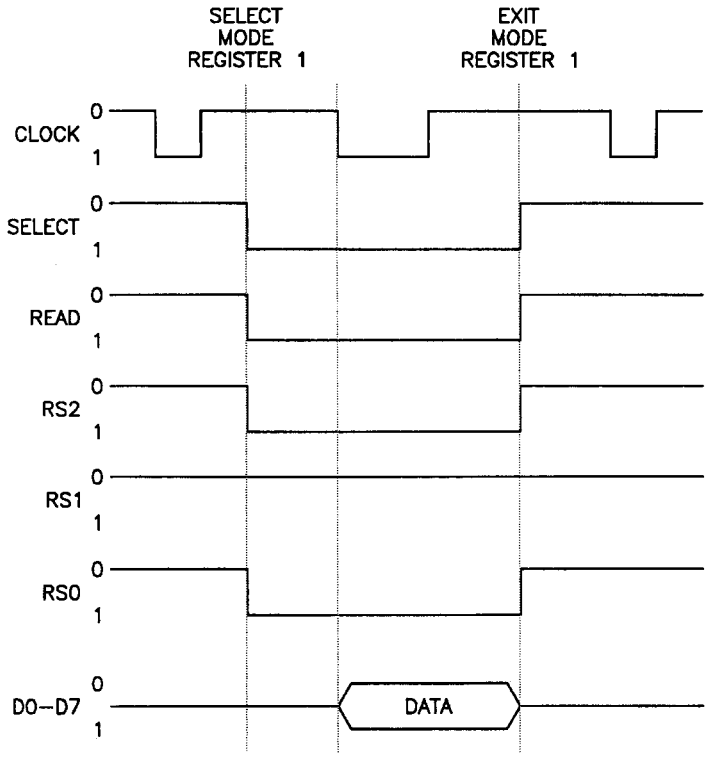


Figure 2.7

(6) CSR (CASSETTE STATUS REGISTER) OUTPUT, R4

				Signification
RS0	0	D0-b0		EOT (end of tape)
RS1	0	D1-b1		TM (tape mark detect)
RS2	1	D2-b2		FPT (file protect)
		D3-b3		CSD (cassette side)
		D4-b4		NRDY (not ready)
		D5-b5	0	'0'
		D6-b6	0	'0'
		D7-b7		ERR (error status)

This register should NEVER be written to, as some of the data bits may be reset.

(a) EOT (end of tape flag)

EOT must be looked at after every control command so as not to allow the MTU to run off the rails.

(b) TM (tape mark)

The flag is set when one of the following conditions are satisfied:

(i) Tape mark detected by *Read one Block* command
(ii) Tape mark detected by *Skip one Block* command
(iii) Tape mark detected by *Search tape Mark* command.

This flag is reset after the CSR is read.

(c) CSD (cassette side)

The flag shows which side of the cassette is being read, i.e. '0' Side A, '1' Side B.

(d) NRDY (not ready)

The flag is '1' when the cassette is not ready or '0' when all conditions are satisfied for normal operation.

(e) ERR (error status)

Shows if an error has occurred; if this is true, i.e. '1' then the ESR (error status register) should be read. This flag is reset when the ESR is read.

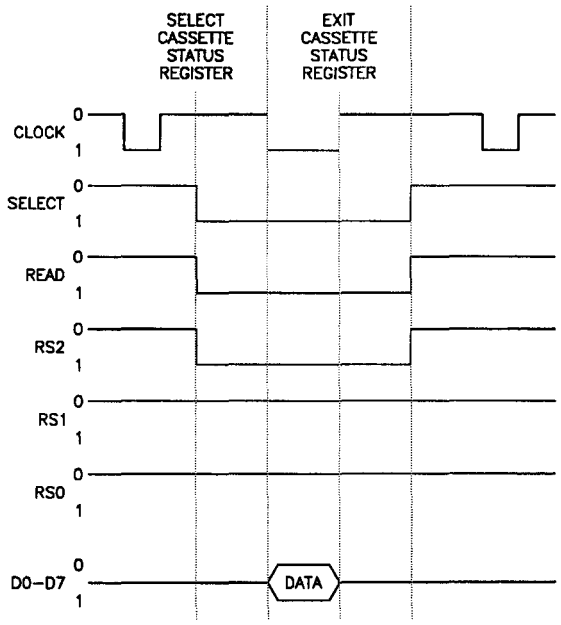


Figure 2.8

(7) ESR (ERROR STATUS REGISTER) OUTPUT, R5

				Signification
RS0	1	D0-b0		CRCER (CRC error)
RS1	0	D1-b1		RTIMER (read timing error)
RS2	1	D2-b2		WTIMER (write timing error)
		D3-b3		PPER (preamble/postamble error)
		D4-b4		WDCER (word count error)
		D5-b5		LIBG0 (long IBG 0)
		D6-b6		LIBG1 (long IBG 1)
		D7-b7		'0'

Various error-checks are performed in the MTU during the execution of a control command. If an error or errors are detected by the error checks, the causes of errors are indicated in this register.

When the contents of this register are read the errors are cleared.

This register should NEVER be written to, as the data will be lost.

All error handling and actions are handled by the high level control programs, so the only requirement is for the programmer to know what each error is and how to fix it.

(a) CRCER (Cyclic Redundancy Check Error)

CRC checking is only performed during the execution of the *Read One Block* command.

(b) RTIMER (read timing error)

Read timing is checked only during the execution of a *Read One Block* command.

If this error does occur then the microcomputer program is at fault, and the section on the microcomputer program should be read.

This error occurs when the MTC is giving out data faster than the microcomputer control program can read it. The maximum time allowed between the *Data Available* signal and *Data Read* is approximately 54.0 microseconds.

(c) WTIMER (write timing error)

Not used.

(d) PPER (preamble/postamble error)

Only checked during *Read One Block* command.

(e) WDCER (word count error)

Only checked during *Read One Block* command.

The MTC compares the number of data bytes from the cassette tape with the number previously written in the register WDC before the input of the *Read One Block* command, and if both numbers do not match, the flag is set to '1'.

If the number of data bytes actually read is greater than the number written in WDC, data transfer requests equal to the number written in the WDC are supplied to the microcomputer and the residual data ignored.

(f) LIBG0 (long inter block gap 0)

This check is performed during the execution of High Speed Search.

(g) LIBG1 (long inter block gap 1)

Checked during all commands sent to the control register.

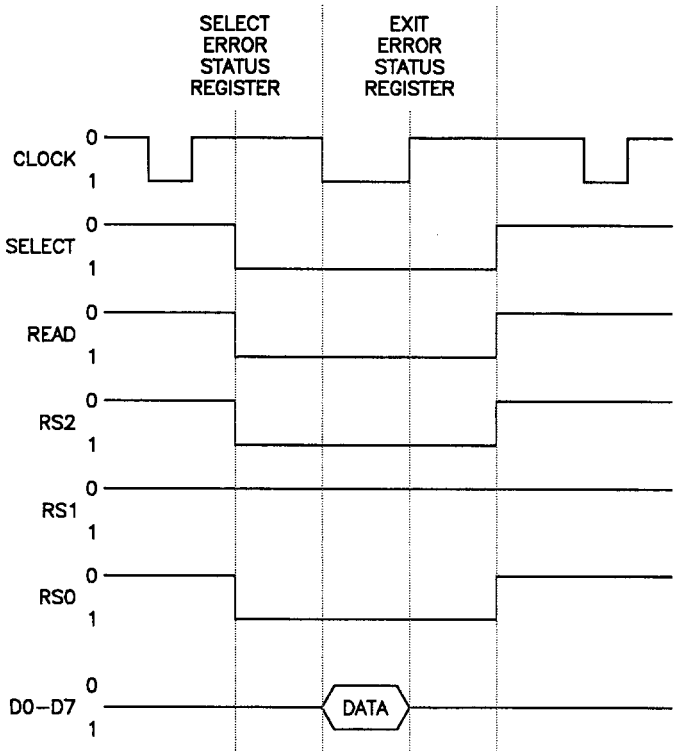


Figure 2.9

(8) ISR (INTERRUPT STATUS REGISTER) OUTPUT, R6

				Signification
RS0	0	D0-b0	0	
RS1	1	D1-b1	0	
RS2	1	D2-b2	0	
		D3-b3	0	
		D4-b4		CFR (controller free request)
		D5-b5		DA (data available)
		D6-b6		DBRE (data buffer register empty)
		D7-b7		CCE (control command end)

If a request to the microcomputer for service occurs in the MTC, a flag corresponding to the request is set in this register.

When the microcomputer reads the contents of the register, the contents are automatically cleared.

NEVER write to this register, as the contents may be cleared.

(a) CFR (controller free request)

If CFRE is '1', the CFR flag is set when the MTC is free or when it becomes free.

(b) DA (data available)

This flag is only effective during the execution of *Read One Block* command. This flag cannot be set during the execution of other control commands.

The MTC sets this flag to '1' to request the services of the microcomputer to transfer each byte to the microcomputer of read data (1 Byte) off the cassette tape from the register DBR.

(c) DBRE (data buffer register empty)

Only used for writing to tape.

(d) CCE (control command end)

This flag is set when all the basic operations of each control command are completed and the MTC is ready for receiving the next control command.

This flag is set on completion of each control command except for the NOP command.

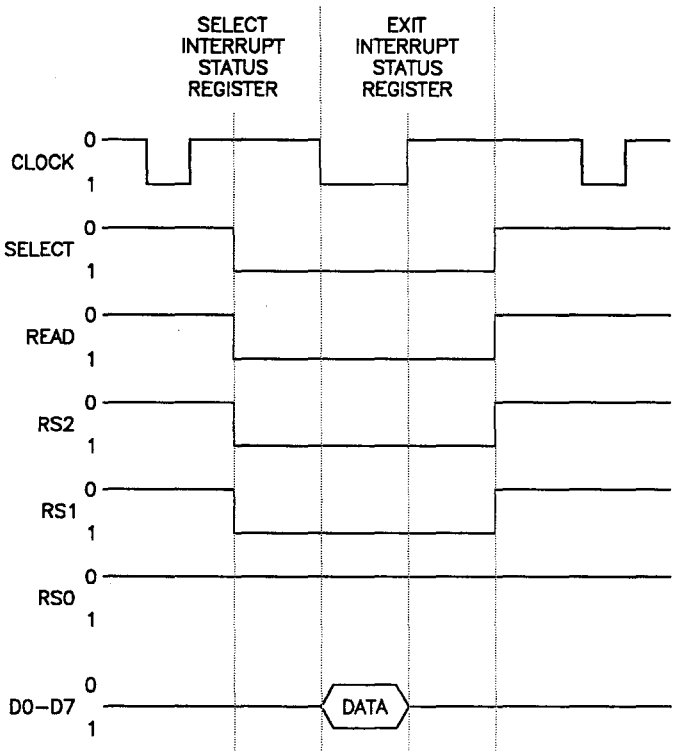


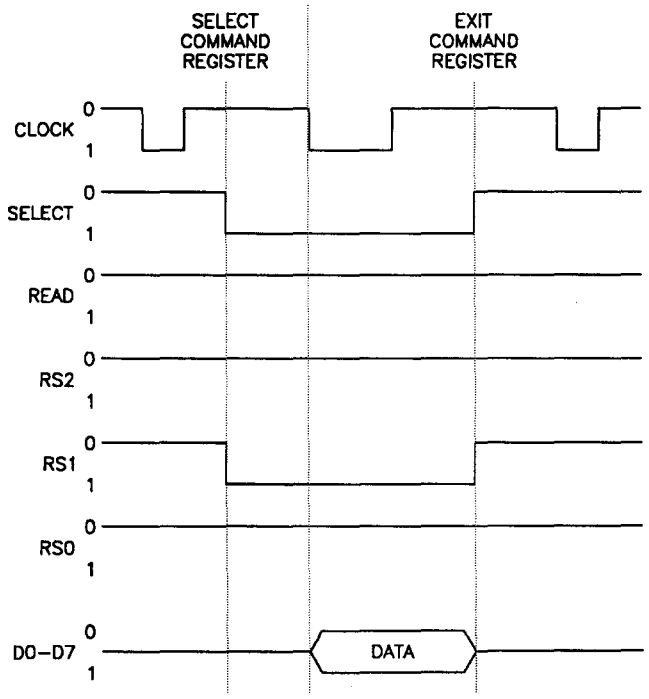
Figure 2.10

COMMAND CODE REGISTER (CDR)

Use of the command code register requires 8 data bits (Bottom 8 bits of the 16 bit word) along with Command Code Register instruction (Top 8 bits of the 16 bit word).

Command register diagram

The register has to be selected, with clock pulse high (Bit 14), then the instruction required (e.g. Set Load Point) along with a clock pulse low. To finish the instruction the clock pulse has to be sent high to exit from command mode. Then clock high then low, then clock high before next instruction (**ALWAYS FINISH WITH CLOCK HIGH**), e.g. set load point (Bit 14 = clock pulse)



The notation used in this section may seem confusing, so a brief explanation will be attempted.

The notation (ISR, b5) refers to the Interrupt Status Register, bit 5. EOT (b0), refers to the End of Tape marker or bit 0 of the register you are currently looking at.

Command codes:

(NOP)	No operation—does nothing
(WTM)	Write tape mark—not used in this application
(WRT)	Write one block—not used in this application
(ERA)	Erase—not used in this application
(RDL)	Read one block—reads one block of 252 bytes of data
(RDH)	Read one block—as above
(SKP)	Skip one block—looks for next tape mark in slow forward mode
(REV)	Reverse one block—looks for previous tape mark in slow reverse mode
(SLP)	Set load point—move the tape in slow forward from BOT (begin of tape) to the load point marker
(SLE)	Set load point with erase—not used in this application
(REW)	Rewind start—rewinds tape in fast reverse to clear leader
(STM)	Search tape mark—searches next tape mark in slow forward mode
(HSS)	High speed search—searches next tape mark in fast forward mode (not used)

Read one block (RDL, RDH)

(1) Command codes (CDR)

b7	b6	b5	b4	b3	b2	b1	b0
1,0	1,0			0	1	0	0

b7	b6	b5	b4	b3	b2	b1	b0
1,0	1,0			0	1	0	1

Figure 2.11

(2) Explanation of the command

Two commands (RDL and RDH) to read one block of data from the cassette tape are available.

Receiving this command, the MTU makes the tape run in slow forward mode and starts the read operation at the detection of a block. The read data are loaded in the register DBR in the form of bytes. Then the DA flag (ISR,b5) is set to "1". With each setting of the DA flag (ISR,b5), the contents of register WDC is decremented until it reaches "0". When the contents becomes "0", no data transfer request will occur even if the reading of data on the tape continues.

(3) Check for the contents of the register CSR

When the microcomputer detects the completion of the command (CCE="1"), read the contents of register CSR and check the following flags.

(a) EOT (b0)

If the tape passes over the EOT hole in the time period from the starting of the RDL or RDH operation to the reading out of the contents of CSR, this flag is set to "1".

(b) TM (b1)

The "1" state of this flag indicates that the block read was a tape mark. In this case, WDCER (ESR,b4) has occurred and the ERR (CSR,b7) flag is set to "1".

(c) ERR (b7)

(i) When the block read was a tape mark (TM="1"), this flag is set to "1" as explained in the above item (b). It is required for the CPU to ignore the ERR flag in this case. Then be sure to read out the register ESR to reset WDCER. If it is not done, WDCER will maintain "1" until the end of the next command.

(ii) If this flag is "0" without the detection of a tape mark, the command has been executed correctly.

(iii) If this flag is "1" without the detection of a tape mark (TM="0"), it indicates that an error or errors have occurred. In such event, read the register ESR to check the contents of the register ESR.

(4) Check for the contents of the register ESR

The following five flags are effective for RDL and RDH commands. Other flags are "0" at this time (see ESR register description).

(a) CRCER (b0)
(b) RTIMER (b1)
(c) PPER (b3)
(d) WDCER (b4)
(e) LIBG1 (b6)

(5) Precautions for controlling

(a) Remember to write the number of data bytes to be transferred into the register WDC. You may write the number any time before the first data transfer request (DA="1") from the MTU. However, it is usually written before input of a control command.

If an RDL or an RDH command is executed without the above designation, one data transfer request occurs whatever the contents of WDC may be, and WDCER (ESR,b4) is set to "0" at the end of the command.

(b) Table 2.3 shows the number written in the register WDC and the number of the data on the tape. The table shows the case when the WDC is written to 100 and a block of 97 bytes–103 bytes is read.

CASE	(A)	(B)	(C)	(D)	(E)	(F)	(G)
No. written in WDC	100	100	100	100	100	100	100
No. of actual read data	97	98	99	100	101	102	103
No. of data transfer request to micro	98	99	100	100	100	100	100
WDCER (ESR,b4)	"1"	"1"	"1"	"0"	"1"	"1"	"1"
Residual No. of WDC	2	1	0	0	0	0	0

Table 2.3. *Number in WDC and Number of read data*

(i) The cases (A)–(C) show that the number of data actually read are smaller than the number written in WDC. The case (D) shows that both numbers are equal. The cases (E)–(G) show that the number of data actually read are greater than the number written in WDC.

(ii) For the cases (A)–(C), the first one byte of the CRC is transferred to the microcomputer. Preamble and postamble are not transferred in any cases.

(iii) The residual number of WDC shows the contents of WDC at the completion of the command; the microcomputer can read the contents freely.

(d) Remember that only one data transfer request (DA="1") occurs when a tape mark is read.

(e) Recognition of a noise block.

Any blocks which are shorter than 15 bits are recognised as noise blocks and they are ignored.

(f) If a CRCER (ESR,b0) or a PPER (ESR,b3) occurs, read the block again after executing a REV command.

(g) If RTIMER (ESR,b1) occurs, read the block again after executing a REV command.

REVERSE ONE BLOCK (REV)

(1) Command code (CDR)

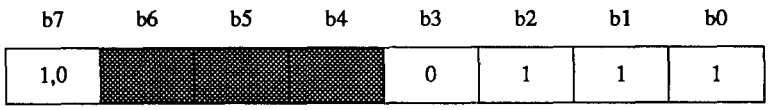


Figure 2.12

(2) Explanation of the command

A command to make the tape run one block in slow reverse mode. No data are transferred by this command. When the end of the reversed block is detected, CCE (ISR,b7) is set to "1" to inform the command end to the microcomputer.

Error checking for the blocks is not performed by the REV command.

(3) Check for the contents of the register CSR

When the microcomputer detects the end of the command (CCE="1"), read out the contents of the register CSR and check the following flags in the register.

(a) EOT (b0): If the tape passes over the EOT hole in the time period between the starting of the REV operation and the reading out of the contents of CSR, this flag will be reset.

(b) ERR (b7): If this flag is "0" it indicates that the command has been executed correctly. If it is "1" read the contents of the register ESR.

(4) Check for the contents of the register ESR

The REV command only checks for LIBG1 (b6). Other flags maintain "0".

(5) Precautions for controlling

(a) Recognition of noise blocks: Any blocks which are shorter than 15 bits are recognised as noise blocks and they are ignored.

(b) If a REV command is supplied before the first block on the tape (BOT hole side), the tape runs passing over the BOT hole by about 400 mm and stops. In such an event, LIBG1 (ESR,b6) will be set to "1". Then if a forward run command is supplied from that stopped point, EOT (CSR,b0) will not be set to "1" even if a hole is detected. However, LIBG1 might be detected.

SKIP ONE BLOCK (SKP)

(1) Command code (CDR)

b7	b6	b5	b4	b3	b2	b1	b0
1,0				0	1	1	0

Figure 2.13

(2) Explanation of the command

A command to make the tape run one block in slow forward mode. No data are transferred by the command. When the end of the skipped block is detected, CCE (ISR,b7) is set to "1" to inform the end of the command to the microcomputer.

By the SKP command, a tape mark is detected, while the error checks for the data block are not performed.

(3) Check for the contents of the register CSR

Read the contents of the register CSR and check the following flags in the register, when the microcomputer detects the end of the command (CCE="1").

(a) EOT (b0): If the tape passes over the EOT hole in the time period from the starting of the SKP operation to the reading out of the CDR contents, this flag indicates "1".

(b) TM (b1): When the skipped block was a tape mark, this flag is set to "1".

(c) ERR (b7): If this flag is "0", it indicates that the command has been executed correctly. If it is "1", confirm the contents of the register ESR.

(4) Check for the contents of the register ESR

SKP command only checks for LIBG1 (b6). Other flags maintain "0".

(a) Recognition of noise blocks: Any blocks which are shorter than 15 bits are recognised as noise blocks and they are ignored.

SET LOAD POINT (SLP)

(1) Command code (CDR)

b7	b6	b5	b4	b3	b2	b1	b0
1,0				1	0	0	0

Figure 2.14

(2) Explanation of the command

A command to make the tape run in slow forward mode from the BOT side clear leader to the load point (initial position). Start any read operations for the cassette tape after executing this command. If the cassette tape is not wound up to the BOT side clear leader, execute a REW command first to wind the tape to the BOT side clear leader, and then execute the SLP command.

Figure 2.15 shows the head and tape position after executing the SLP command.

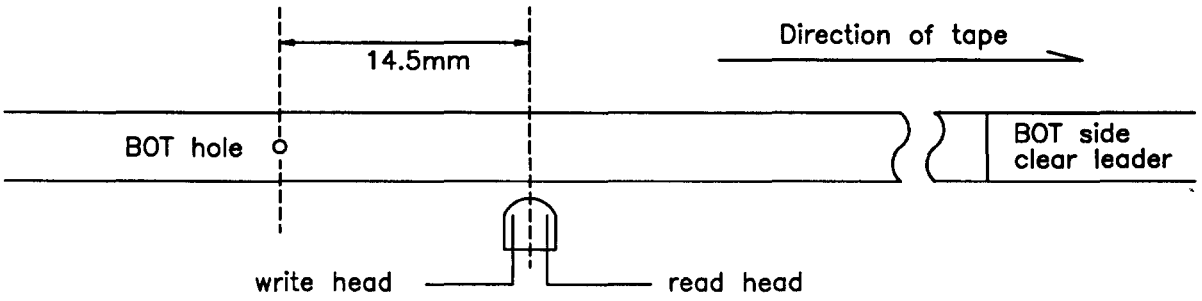


Figure 2.15

(3) Check for the contents of the register CSR

When the microcomputer detects the end of the command (CCE="1"), read out the contents of the register CSR and check for ERR (b7). EOT (b0) and TI (b1) flags maintain "0".

(a) ERR (b7): If this flag is "0", it indicates that the SLP command has been executed correctly. If it is "1", read out the register ESR.

(4) Check for the contents of the register ESR

The SLP command only checks for LIBG1 (b6). Other flags maintain "0".

(a) LIBG1 (b6): If the BOT hole is not detected after 22.0–24.5 inches (558.8–622.3 mm) run of the tape from the BOT side end of the clear leader, this flag is set to "1" to complete the command.

(5) Precautions for controlling

(a) If the SLP command is supplied when the tape is in magnetic tape area, the tape starts to run and if a hole is not detected during the 600 mm (approximately) run, LIBG1 (ESR,b6) is set to "1" to complete the command. Even if a hole is detected within the 600 mm run, there is no way to distinguish the EOT hole from the BOT hole.

(b) If the SLP command is supplied in the EOT side clear leader, magnetic tape area is not detected and the CCE(ISR,b7) will not be set.

(c) The MTU accepts the SLP command even if a cassette tape is not inserted. The execution of the SLP command will be started when a cassette is inserted and NRDY becomes "0".

Be sure to confirm that the cassette to be inserted is wound up to the BOT side clear leader.

REWIND START (REW)

(1) Command code (CDR)

b7	b6	b5	b4	b3	b2	b1	b0
1,0				1	0	1	0

Figure 2.16

(2) Explanation of the command

A command to rewind the tape at fast reverse mode (45ips) to the BOT side clear leader end. When the execution of this command starts, CCE (ISR,b7) is set to "1" immediately and the starting of the rewind operation is informed to the microcomputer. If a new command is supplied after CCE="1", the new command will be executed successively after the completion of the rewind operation.

When you rewind the tape to remove the cassette from the MTU, no other operation is needed after confirming rewinding start (CCE="1").

(3) Check for the contents of the register CSR

The REW command does not check errors. TM(b1) and ERR(b7) flags maintain "0". The EOT(b0) flag will be reset if an EOT hole is detected during the rewind operation.

For the above reason, you need not check the contents of the register CSR if it is not required for special purposes.

(4) Precautions for controlling

(a) The MTU accepts the REW command even if a cassette tape is not inserted. The execution of the REW command will be immediately started when a cassette is inserted and NRDY becomes "0". The CCE (ISR,b7) is set to "1" immediately after the starting of the command execution.

(b) For the detection of the rewinding operation end, CFR (ISR,b4) is available.

SEARCH TAPE MARK (STM)

(1) Command code (CDR)

b7	b6	b5	b4	b3	b2	b1	b0
1,0				1	1	0	0

Figure 2.17

(2) Explanation of the command

A command to make the tape run continuously at slow forward mode and to search for the tape mark. When the tape mark is detected, CCE(ISR,b7) is set to "1" and the end of the command is informed to the microcomputer.

Error checking and data transfers of the blocks, except for the tape mark, are not performed by the STM command.

(3) Check for the contents of the register CSR

If the microcomputer detects the command end (CCE="1"), read the contents of the register CSR and check the following flags in the register.

(a) EOT(b0): If the tape passes over the EOT hole in the time period from the starting of the STM operation to the reading out of the CSR contents, this flag is set to "1".

(b) TM (b1): When the tape mark is detected, this flag is set to "1". ERR(b7) is "0" at this time.

(c) ERR (b7): When this flag is "1", read the register ESR.

(4) Check for the contents of the register ESR

STM command only checks for LIBG1(b6). Other flags maintain "0".

(5) Precautions for controlling

(a) Recognition of a noise block: Any blocks which are shorter than 15 bits are recognised as noise blocks and they are ignored.

(b) Input a SKP command if you want to stop the STM operation after it is started. The MTU will execute the SKP command instead of the STM command and set the CCE (ISR,b7) to "1" to complete the command.

Never input a command except for the SKP. If it were given, troubles might occur.

(c) When an error occurs in the tape mark, the tape will be forwarded without a stop, as the block cannot be recognised as a tape mark.

HIGH SPEED SEARCH (HSS)

(1) Command code (CDR)

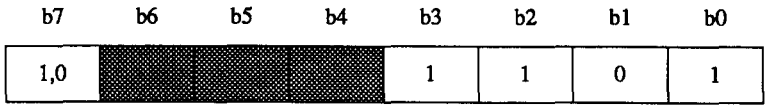


Figure 2.18

(2) Explanation of the command

A command to make the tape run continuously at fast forward mode (45ips) and to detect the tape mark. When the tape mark is detected, CCE (ISR,b7) is set to "1" and the end of the command is informed to the microcomputer.

(3) Check for the contents of the register CSR

If the microcomputer detects the end of the command (CCE="1"), read the contents of the register CSR and check the following flags in the register. Remember to ignore the TM(b1).

(a) EOT (b0): If the tape passes over the EOT hole in the time period from the starting of the HSS operation to the reading out of the CSR contents, this flag is set to "1".

(b) ERR (b7): If this flag is "0", it indicates that the command has been executed correctly. If it is "1", read the register ESR.

(4) Check for the contents of the register ESR

HSS command only checks LIBG0(ESR,b5). Other flags maintain "0".

(5) Precautions for controlling

(a) How to detect the tape mark: As the tape mark pattern is not checked by the HSS command, the tape mark is recognised by the method in Figure 2.19.

This method utilizes the differences in the block length, that the standardised tape mark in the various standards is 32 bits, minimum data block is 48 bits, and the maximum noise block recognised by the MTU at fast mode is 11 bits.

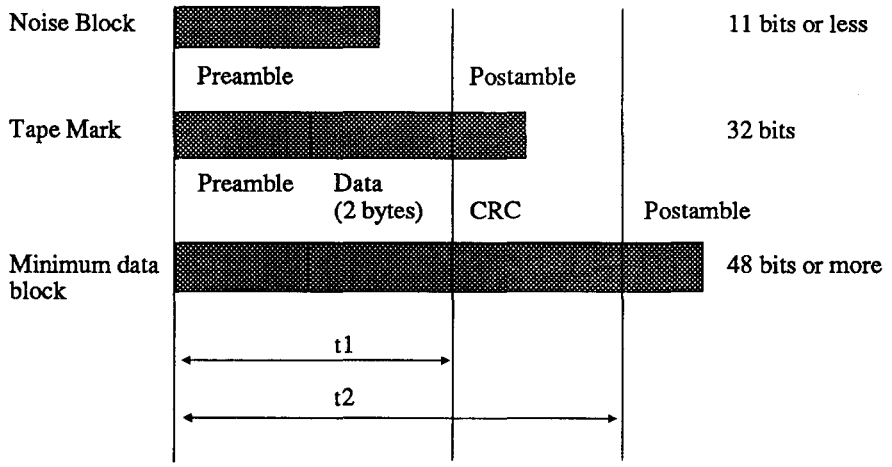


Figure 2.19

(b) Start/stop distances: As the HSS command is executed at fast mode, a longer tape distance is required for start/stop operations than at slow mode. Be sure to insert a longer IBG than usual.

The tape length required to recognise the tape mark from the difference in block length after starting the HSS operations is 2.4 inches (60.96 mm) maximum. The tape length travelled from the trailing edge of the tape mark detected by HSS command to the point where tape completely stops is 1.9 inches (48.26 mm) maximum.

STATUS INFORMATION

Status information is indicated by registers CSR and ESR. Some of the status information is always effective, and some is not always effective, indicating the result of the control command. The status is alway acted upon by the minicomputer, so this section is to be read in conjunction with Chapter 4 (Minicomputer Control Program).

(1) Status always effective

The status FPT(CSR,b2), CSD(CSR,b3) and NRDY(CSR,b4) are always effective. Each of these status indicates the cassette tape condition. These status are usually checked before the execution of a control command.

(2) Status indicating the result of a control command executed

Table 2.4 shows the status information to be checked for each control command executed. The status in the table which has "0" (except for the status with asterisks) indicates that the checking is not done for the command.

(a) EOT (CSR,b0): The flag becomes effective after executing a control command except for REW, SLP, and SLE. The "1" state is maintained as long as the cassette tape is in EOT state (see Table 2.4). The flag is reset by the input of RST signal or by executing SRST (soft reset).

(b) TM (CSR,b1): The flag becomes effective after executing a control command of WTM, RDL, RDH, or STM. It is reset when the register CSR is read out. The TM flag indicates either "0" or "1" after the execution of the HSS command. However, either state "0" or "1" should be ignored.

(c) ERR (CSR,b7): The flag becomes effective after executing a control command except for REW and ERA, and is reset after the register ESR is read out. When the ERR flag indicates "1", read the register ESR to confirm its contents and to clear the contents for the execution of the next command. If the contents of ESR is not read out, it will be maintained after the execution of the next control command.

(3) Notes for Table 2.4

(a) The table indicates the cases without operation errors and troubles of the MTU.

(b) The numbers with asterisk indicates that the checking is performed. However, the status are as indicated in the table when no troubles occur in the MTC.

(c) A "0" without asterisk indicates that the checking is not performed.

(d) The status information with double asterisks (**) are ineffective. They should be ignored by the CPU side of operations.

Control commands	Register CSR			Register ESR						
	ERR b7	TM b1	EOT b0	LIBG1 b6	LIBG0 b5	WDCER b4	PPER b3	WTIMER b2	RTIMER b1	CRCER b0
WRT	1,0	0	1,0	*0	0	0	1,0	1,0	0	1,0
WTM	*0	1,0	1,0	*0	0	0	0	0	0	0
ERA	0	0	1,0	0	0	0	0	0	0	0
RDL	1,0	1,0	1,0	1,0	0	1,0	1,0	0	1,0	1,0
RDH	1,0	1,0	1,0	1,0	0	1,0	1,0	0	1,0	1,0
SKP	1,0	1,0	1,0	1,0	0	0	0	0	0	0
REV	1,0	**0 or 1	1,0	1,0	0	0	0	0	0	0
SLP	1,0	0	0	1,0	0	0	0	0	0	0
SLE	1,0	0	0	1,0	0	0	0	0	0	0
REW	0	0	0	0	0	0	0	0	0	0
STM	1,0	1,0	1,0	1,0	0	0	0	0	0	0
HSS	1,0	**0 or 1	1,0	0	1,0	0	0	0	0	0

Table 2.4

CHAPTER 3

Microcomputer interface

INTRODUCTION

The microcomputer interface enables communication between a minicomputer and the Teac tape unit. The advantages of this type of control are as follows:

(a) less overhead for the minicomputer.

(b) allows a high level language control program to operate the Teac tape drive, without the operator needing to know how the Teac tape unit works.

(c) allows buffering of data and direct control of the tape unit, letting the minicomputer perform other tasks whilst data retrieval occurs.

INSTRUCTION SET

The microcomputer accepts the following instructions from the mini computer and translates them into Teac control codes.

Instruction Code (from minicomputer)	*(hexadecimal notation)*
NOP (no operation)	1
Write One Block (not used)	2
Write Tape Mark (not used)	4
Erase (not used)	8
Read One Block	F
Read One Block	10
Skip One Block	20
Reverse One Block	40
Rewind Set Load Point	80
Set Load Point With Erase (not used)	F0
Unload	100
NOP (no operation)	200
Search Tape Mark	400
High Speed Search	800
NOP (no operation)	F00
NOP (no operation)	1000

BASIC OPERATION

The basic operation of the microcomputer interface is as follows:

(1) On power up

(i) Reset all program parameters (i.e. pointers, interupts etc.).
(ii) Initialise and reset the Teac tape unit.
(iii) Wait for a request to perform an instruction from the minicomputer control program.

(2) On receiving a request to perform an instruction.

(i) Instruct the minicomputer that an instruction is ready to be received.
(ii) accept the instruction and convert it into control signals for the Teac tape unit.
(iii) Perform instruction.
(iv) Return to the minicomputer any errors or faults that may have occurred during operation so that any errors can be acted upon by the minicomputer.

PROGRAM STRUCTURE

The program is set up as a main program dealing with power up, reset, and interpreting the commands from the minicomputer. All commands are dealt with by subroutines designed to issue these instructions to the tape unit. These subroutines may then call up ancillary subroutines which may be shared by a number of command subroutines; these include error handling, clock pulses, and tape unit control information. The third level of subroutines is the control of the communication link between the microcomputer and the minicomputer; this includes handshaking and output of data.

DESCRIPTION OF CONTROL COMMAND SUBROUTINES

Read One Block (RDL or RDH)

The *Read one Block* command requires the most interaction between the Teac tape unit and the microcomputer. As data are transferred from the tape unit as 252 byte blocks, the microcomputer program has to be fast enough to read and store

the data one byte at a time without handshaking or buffering. The time restriction imposed by the this problem means that the program has to be as efficient as possible in terms of speed and not programming technique.

Upon receiving the command from the minicomputer to read one block of data the microcomputer enters the *Read one Block* subroutine which acts in the following way:

(i) Resets all memory pointers
(ii) Writes to the tape unit Word counter register the number of bytes per block of data to be retrieved (i.e. 252).
(iii) Clears the interrupt status register
(iv) Instructs the tape unit to read one block of data
(v) Read the interrupt status register allowing the checking of the data available flag (DA) to be checked as soon as the next subroutine is entered.
(vi) Jump to a subroutine that deals with the reading and storing of data from the tape unit; this subroutine will deal with the whole 252 bytes (1 block) of data (this is the subroutine that has to be the most efficient as far as the time restraints are concerned).
(vii) Give the 252 bytes (1 block) to the minicomputer. This can be done without any time restrictions being imposed by data transfer, as this operation is handshake controlled.
(viii) Give the minicomputer the contents of the error status register and the cassette status register, so that any errors can be dealt with.

The *Read one Block* command is broken up into basically 3 sections:

(1) Set up the tape unit to replay 1 block of data
(2) Read and store in memory 1 block of data
(3) Give this one block of data to the minicomputer

The main control subroutine is described in the flow chart (fig. 3.1).

Described in the following flow chart (fig. 3.2) is the subroutine which is the most critical section of the *Read one Block* command. The structure of this flow chart should be followed to the letter; any deviation will cause drastic errors and possible loss of data.

Skip One Block (SKP) (fig. 3.3)

This command moves the tape forward one block of data in slow forward mode. It does not return any data to the microcomputer, but it does return the contents of the Error and Cassette status registers to the minicomputer for error handling.

Reverse One Block (REV) (fig.3.4)

This command rewinds the tape in slow mode to the beginning of the previous block of data. No data is read, and the contents of the Error and Cassette status registers are returned to the minicomputer for error handling.

Rewind Set Load Point (REW) (fig 3.5)

This command fully rewinds the tape (to the clear leader) then moves it forward in slow mode until the load point marker is detected by the tape unit, then stops on that point. No data is read, and the contents of the Error and Cassette status registers are returned to the minicomputer for error handling.

Unload (fig. 3.6)

This command is similar to the REW command except for two major differences:

(i) The tape is rewound to the beginning of the tape (clear leader) and not moved forward to the Load point.
(ii) The contents of the Error and Cassette status registers are not returned to the minicomputer for error checking.

This command is issued only to remove a cassette tape from the Teac tape unit.

Search Tape Mark (STM) (fig 3.7)

This command searches for the next tape mark that appears on the tape in slow forward mode. No data is read, and the contents of the Error and Cassette status registers are returned to the minicomputer for error handling.

High Speed Search (HSS) (fig.3.8)

This command is similar to the STM command except that the search is carried out in fast forward mode. It has been found through experience that this command can tend to skip the tape marks and cause drastic errors, so it is recommended that this command not be used.

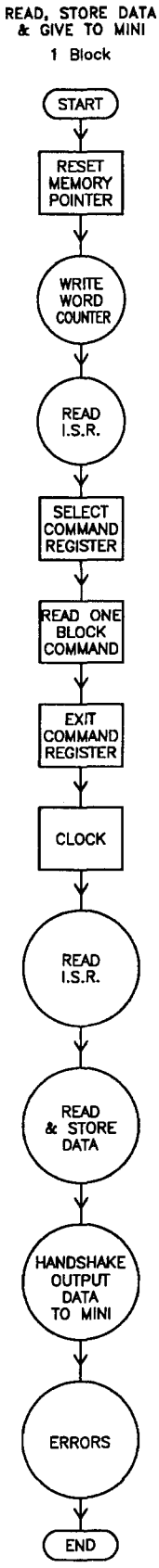


Figure 3.1

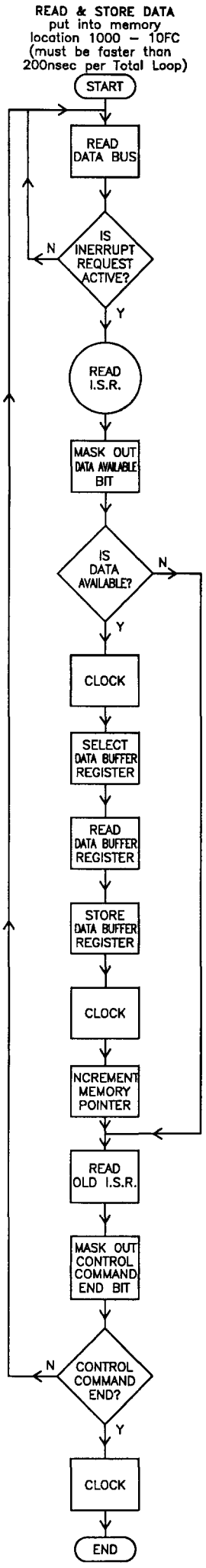


Figure 3.2

SKIP ONE BLOCK

Go foward one block
Don't read data

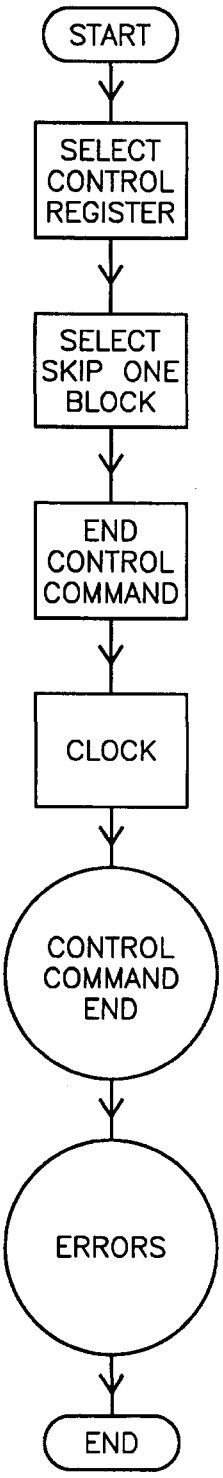


Figure 3.3

REVERSE ONE BLOCK

Reverse
Do not read data

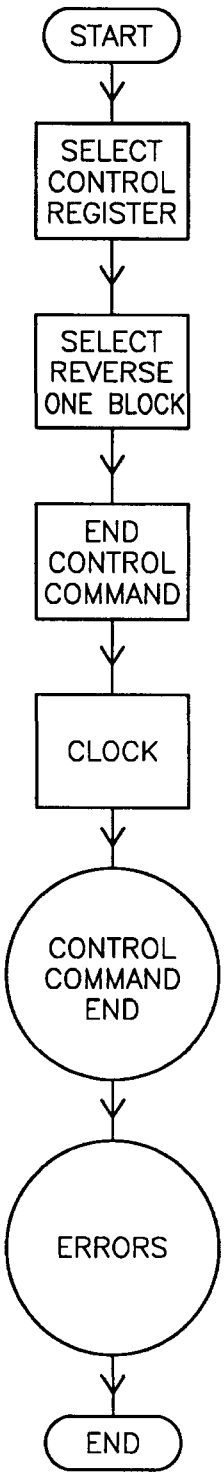


Figure 3.4

REWIND

Rewind
set load point

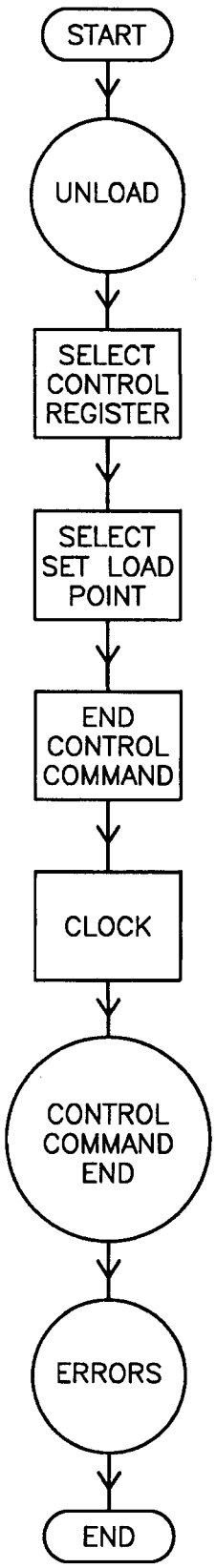


Figure 3.5

UNLOAD (TAPE FROM TEAC)

Rewind to clear leader

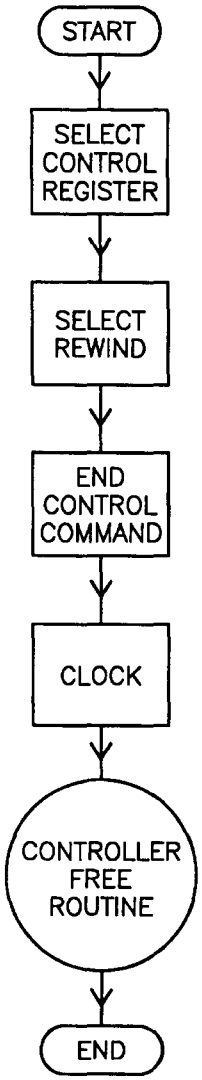


Figure 3.6

SEARCH TAPE MARK

Look for next tape mark
Slow Forward

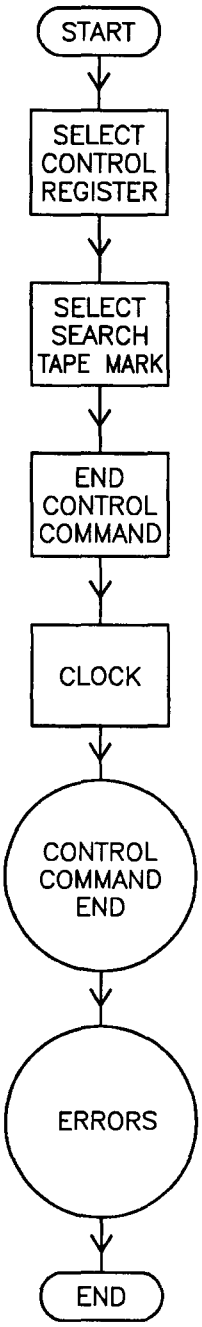


Figure 3.7

HIGH SPEED SEARCH

Look, for next tape mark,
Fast Foward

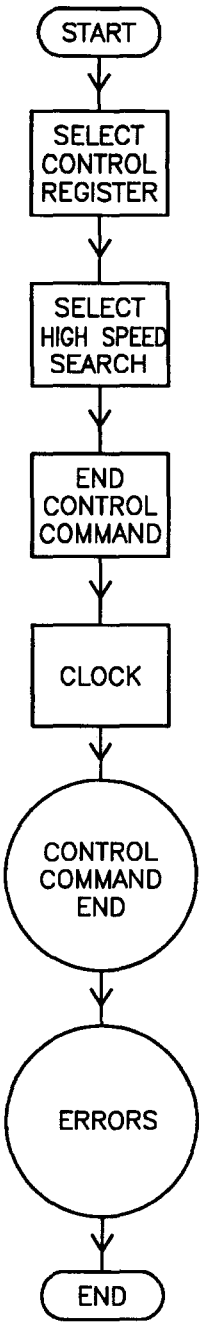


Figure 3.8

ANCILLARY COMMANDS

Error Status Register (ESR) (fig. 3.9)

Read the contents of the tape unit error status register and store the data into one of the microcomputers internal memory locations which will not be overwritten by any other subroutine.

Cassette Status Register (CSR) (fig. 3.10)

Read the contents of the tape unit cassette status register and store the data into one of the microcomputers internal memory locations which will not be overwritten by any other subroutine.

Interrupt Status Register (fig. 3.11)

Read the contents of the tape unit interrupt status register and store the data into one of the microcomputers internal memory locations which will not be overwritten by any other subroutines. The data from this register are used for control of the tape unit from the microcomputer and not for use by the minicomputer control program.

Write Word Counter (WDC) (fig. 3.12)

The write word counter subroutine is used to tell the tape unit the number of bytes of data to be read as one block. In this case the number is 252, which is determined by the way the tape has been formatted on recording; i.e. 252 bytes per block. Once data has been written to the word counter it is not read from or written to until the next block of data are to be read.

Controller Free Detect (CFRE) (fig. 3.13)

The Controller Free Detect is used to determine if the motor status of the tape unit is stop. This enables the use of the unload command, otherwise it would be impossible to detect when the tape is free to be removed from the unit, as the REW command by itself does not return a value to the minicomputer when it has completed the command.

Teac Reset (TRT) (fig. 3.14)

Used on power-up, this subroutine establishes communication to the tape unit and resets all hardware and software control to the initialising state. Only used on power-up, not during normal operation.

Control Command End (CCE) (fig. 3.15)

The Control Command End subroutine is one of the most important of the ancillary subroutines, as it detects when an operation such as reverse one block, skip one block etc. has been completed. If this flag is not checked for, it would be very easy for the tape unit to miss instructions from the microcomputer because it may be executing an operation whilst another instruction is passed onto it, and therefore miss the new instruction altogether. This CCE subroutine is to be used at the end of every control command subroutine (except for unload which uses CFRE flag) before returning for another control command.

Errors (ERRS) (fig. 3.16)

At the completion of all the command control subroutines (except unload), the results of the ESR and CSR are given to the minicomputer for error diagnostics. In this particular case the two registers (8 bits each) are combined into one word (16 bits) to be decoded by the minicomputer.

ERROR STATUS

Read Register & store location ØØØ2

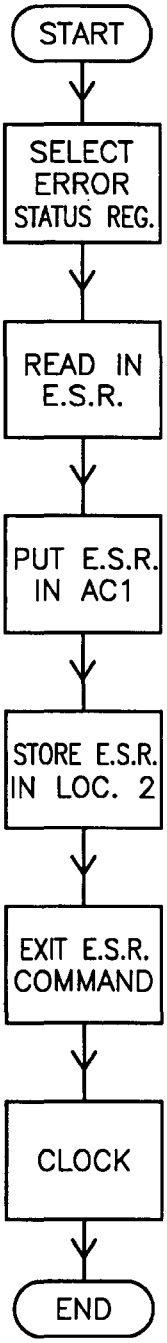


Figure 3.9

CASSETTE STATUS REGISTER

Read Register & store location ØØØ0

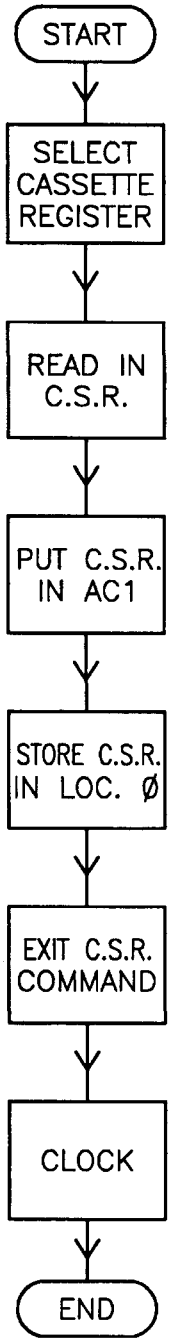


Figure 3.10

INTERRUPT STATUS REGISTER

Read Register & store location ØØØ1

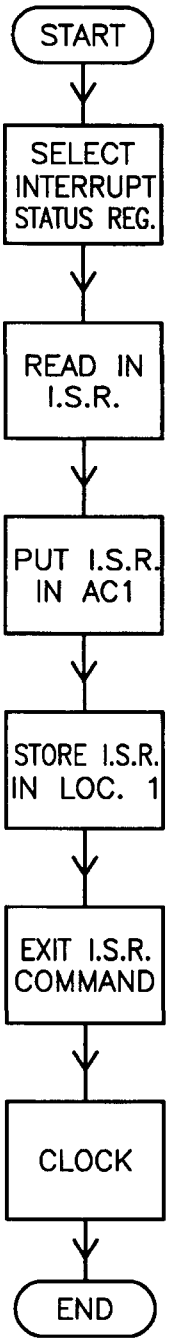


Figure 3.11

WRITE WORD COUNTER

252 Bytes per Block

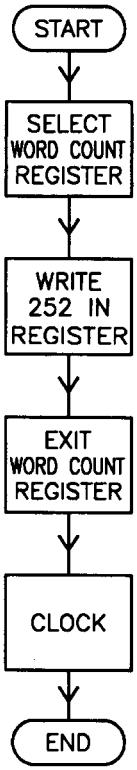


Figure 3.12

CONTROLLER FREE DETECT

used when motor has stopped on rewind

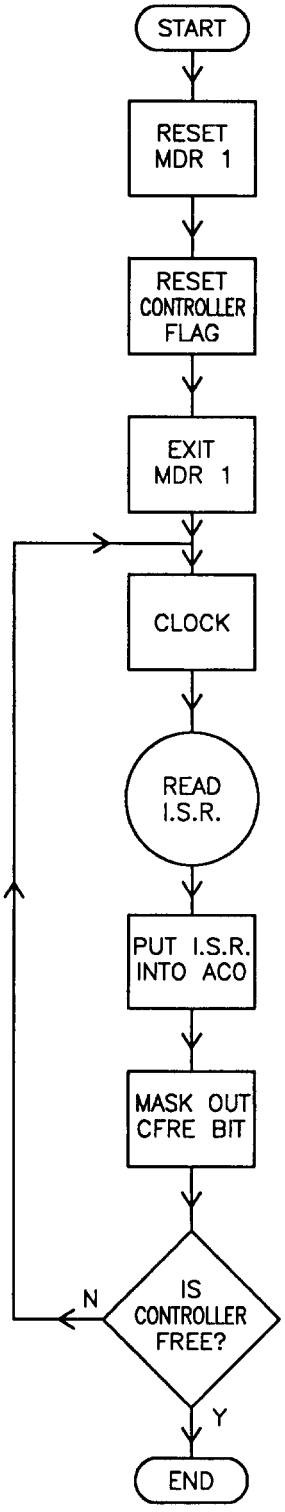


Figure 3.13

TEAC RESET

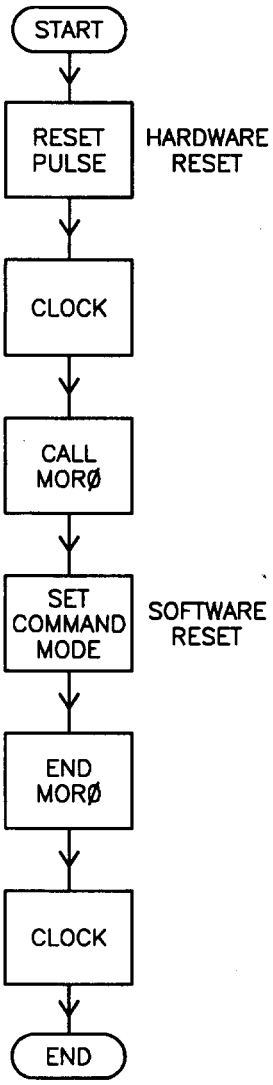


Figure 3.14

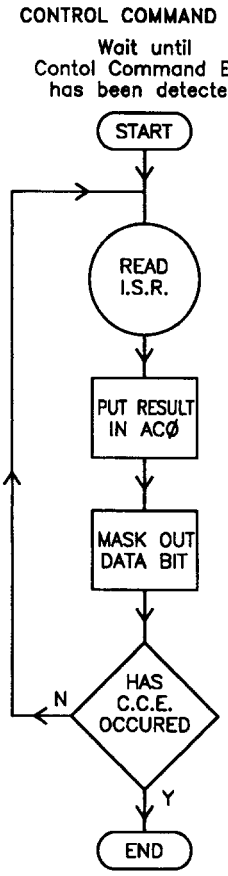


Figure 3.15

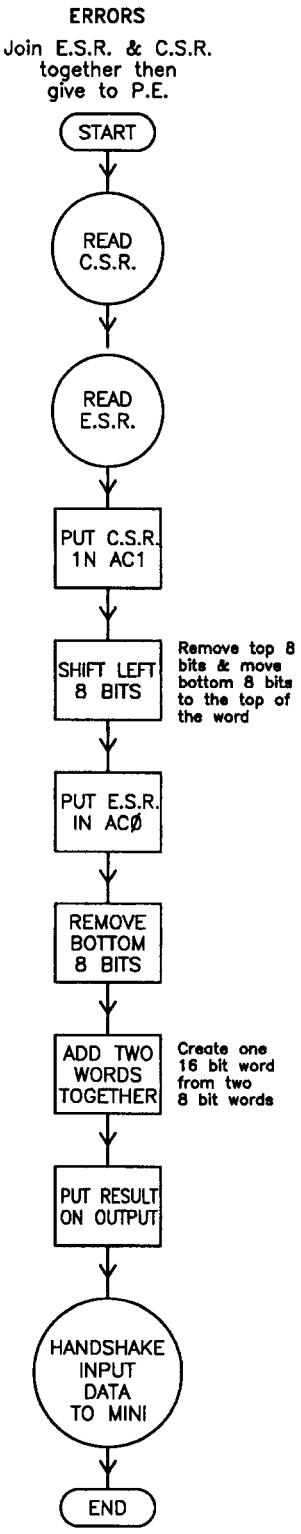


Figure 3.16

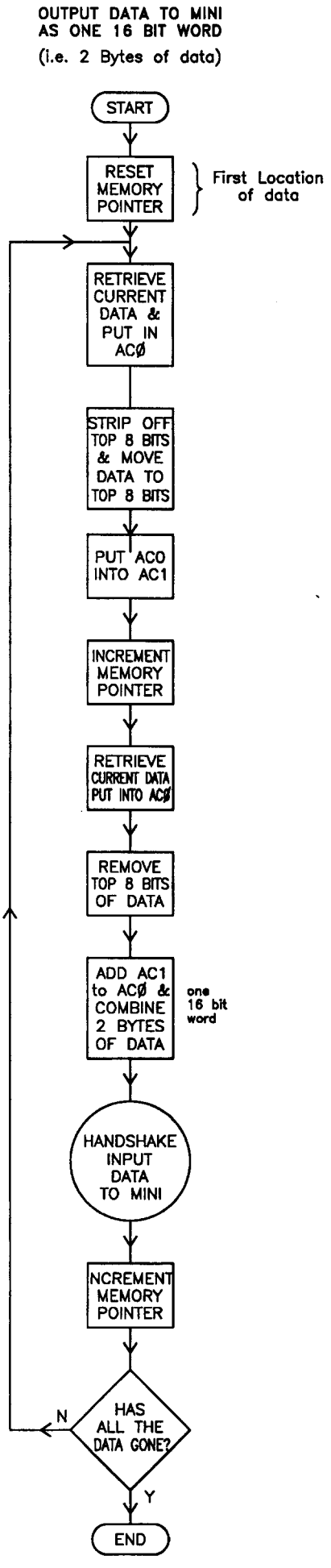


Figure 3.17

MINICOMPUTER COMMUNICATION CONTROL SUBROUTINES

Output Data To The Minicomputer (OUT) (fig. 3.17)

Data from the tape unit is output to the minicomputer from the microcomputer by control of this subroutine. Data from the tape unit are in the form of 8 bit words (1 byte), data transferred to the minicomputer from the microcomputer are in the form of 16 bit words (2 bytes). To make life easier for the data transfers, data from the tape unit are joined together to form 16 bit words (i.e. 2 consecutive bytes of data joined together) and passed on to the minicomputer. The top 8 bits of the word are the first piece of data, and the bottom 8 bits are the second piece of data.

Handshake Input To Minicomputer (HANDIN) (fig. 3.18)

This subroutine allows the synchronization between the minicomputer and microcomputer during data transfers from the microcomputer to the minicomputer.

Handshake Output From Minicomputer (HANDOUT) (fig. 3.19)

This subroutine allows the synchronization between the minicomputer and microcomputer during data transfers from the minicomputer to microcomputer.

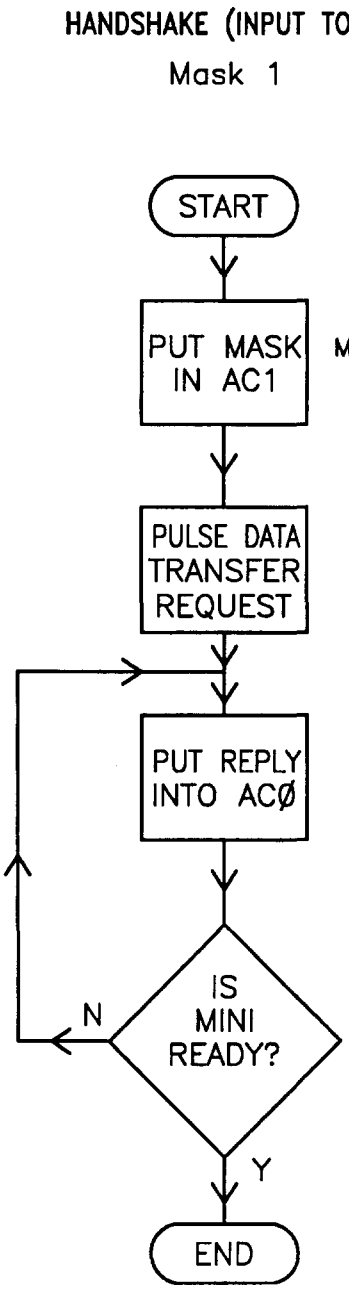


Figure 3.18

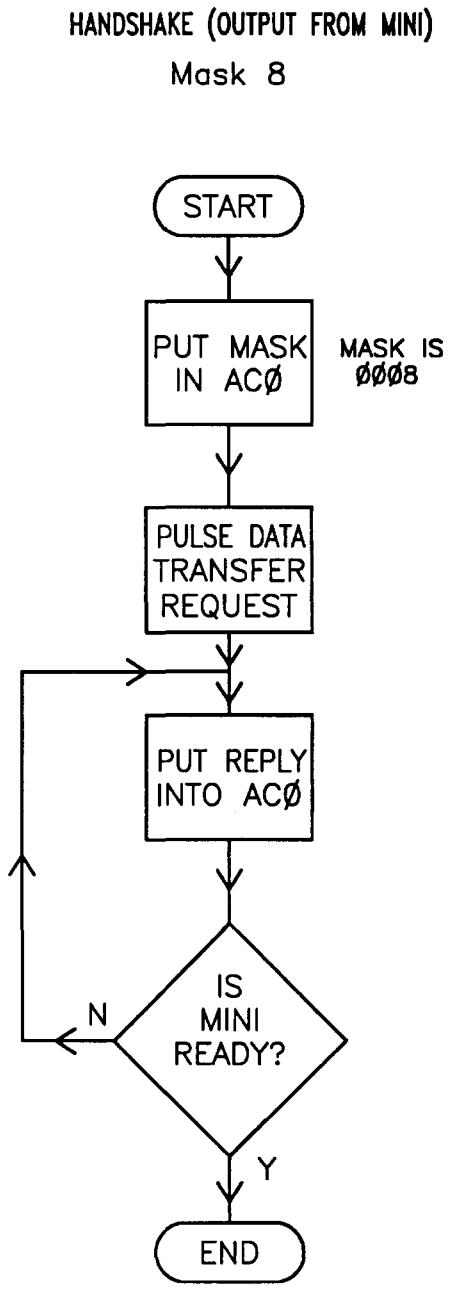


Figure 3.19

CHAPTER 4

Minicomputer control program

INTRODUCTION

The minicomputer program is designed to retrieve data from a cassette tape and archive it on to magnetic tape in compressed form for long term storage. Error messages are returned to the minicomputer from the tape unit (via the microcomputer) and are acted upon accordingly. A detailed description of the control and archiving technique are given in Richardson (1989).

REFERENCES

RICHARDSON, R. G. 1989. WIRELOG—Fortran programs for replaying , archiving and retrieving wireline logging data. *Unpubl. Rep. Dep. Mines Tasm.* 1989/04.

NATIONAL SEMICONDUCTOR CORPORATION. 1976. *Pace low cost development system (LCDS). Users Manual.* National Semiconductor Corporation : Santa Clara, California.

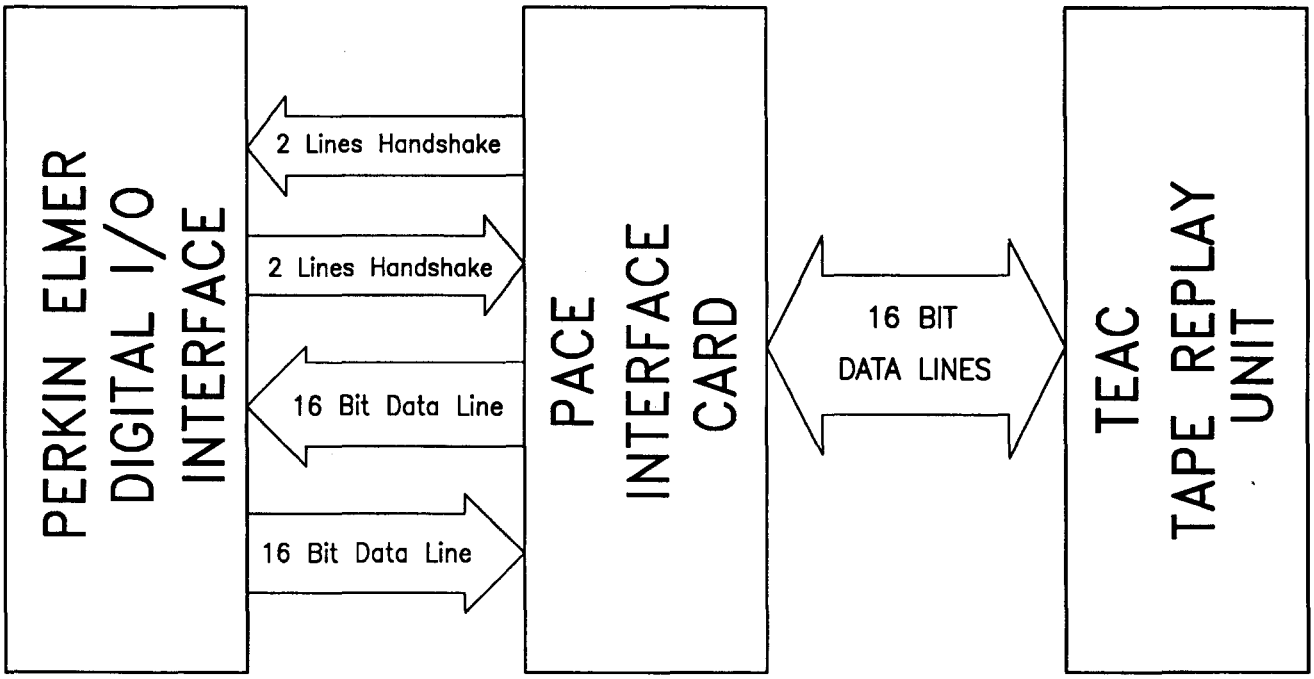
NATIONAL SEMICONDUCTOR CORPORATION. 1976. *Pace low cost development system (LCDS). Assembly language.* National Semiconductor Corporation : Santa Clara, California.

TEAC CORPORATION. *Teac Model MT-2 (-04) instruction manual.* Teac Corporation 10131035-3.

[18 September 1989]

APPENDIX 1

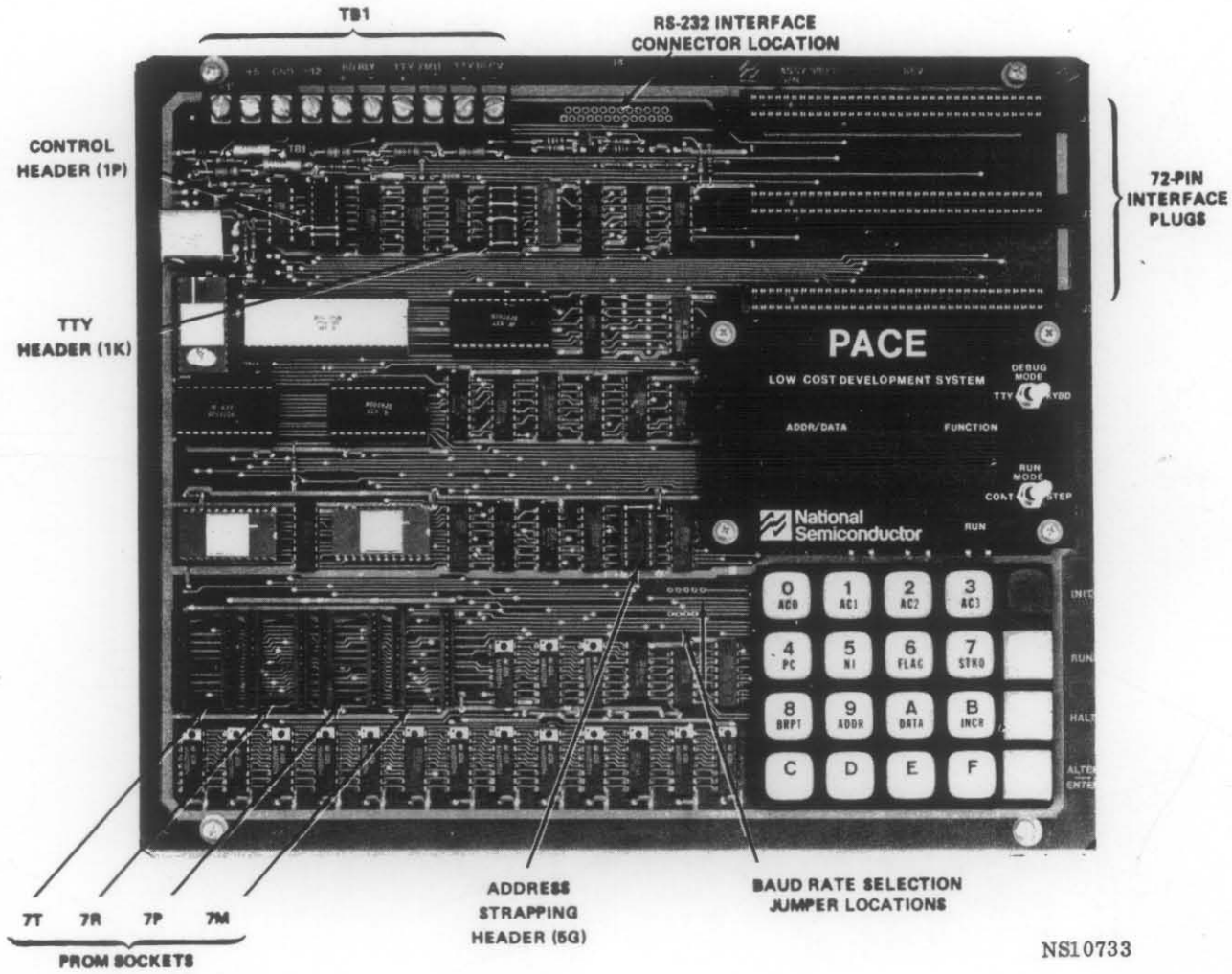
Components of the system as Version 1



APPENDIX 2

PACE minicomputer system

The PACE Low Cost Development System (LCDS) is a self-contained microcomputer which provides all the capabilities necessary for developing, testing, and debugging PACE hardware and software applications design. The LCDS is completely contained on a single printed circuit board, and only requires connection of a power supply to be fully operational. A PACE microprocessor and 1024 16 bit words of read/write memory provide a ready-to-use environment for user's applications programs. A further 4 K of battery-powered cmos RAM, which is manually write enabled/protected, and a further 1 K of TTL RAM were also installed. A RS-232c interface is provided, allowing direct connection to a VDU; this allows easy access to the resident debugging firmware. Programming of the LCDS is required in machine code language (hexadecimal notation); assembly code can be used to develop software but has to entered into the LCDS as machine code either via the VDU or keyboard on the LCDS.



Further information on the programming of the PACE LCDS can be found by reading the PACE LCDS Users Manual in conjunction with the PACE LCDS Assembly Language Programming Manual.

APPENDIX 3

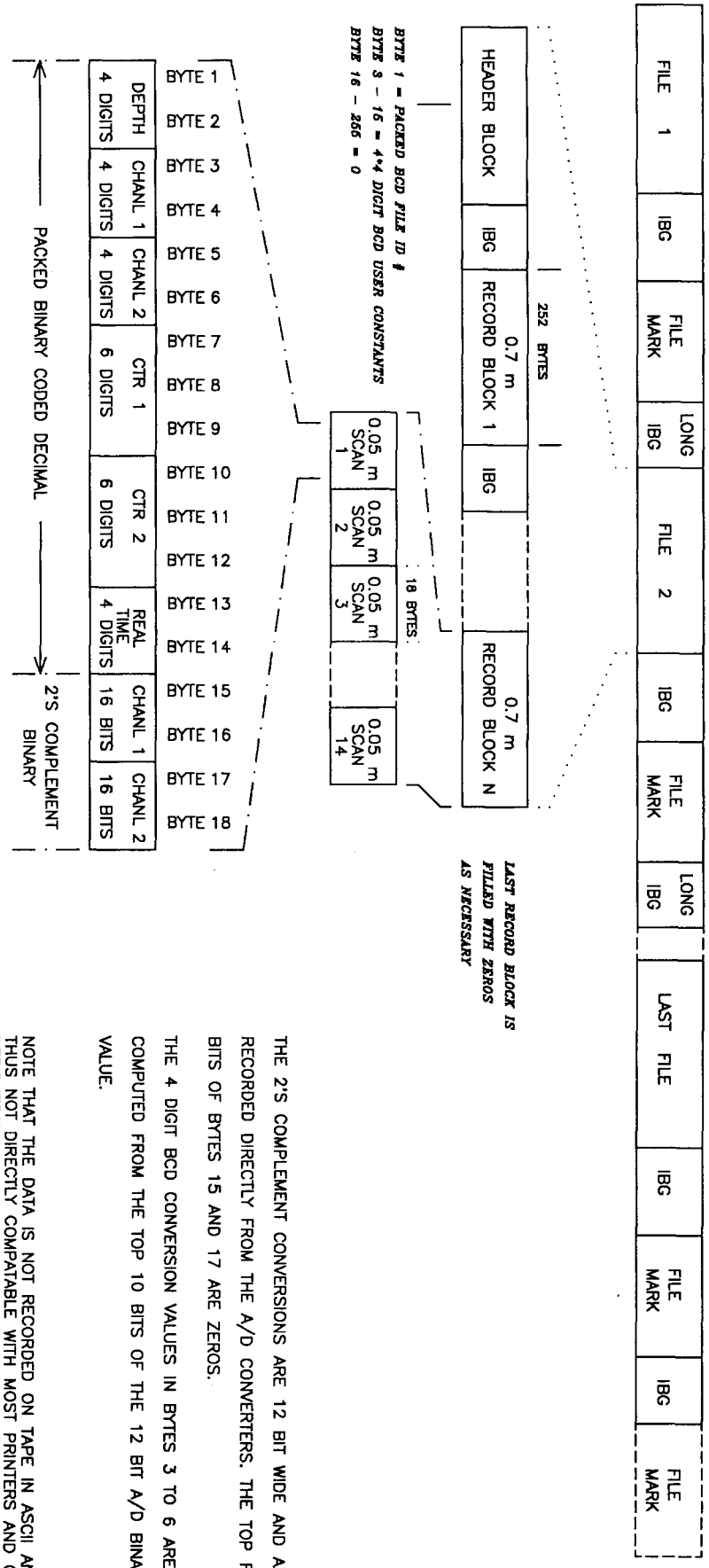
Trouble shooting and error handling

The only time that a serious error would occur is when the PACE LCDS and the interface board on the PERKIN-ELMER become out of sync (i.e. the handshaking has gone horribly wrong). To fix this problem, which is signified by giving a feed back response from the PERKIN-ELMER driver program of TIME-OUT, is to run the program on the PERKIN-ELMER called INIT. After running this program the PACE should be reinitialised and the driver program restarted.

Any other problems which occur are sure to be serious, and should be attended to by a qualified person.

APPENDIX 4

Tape format



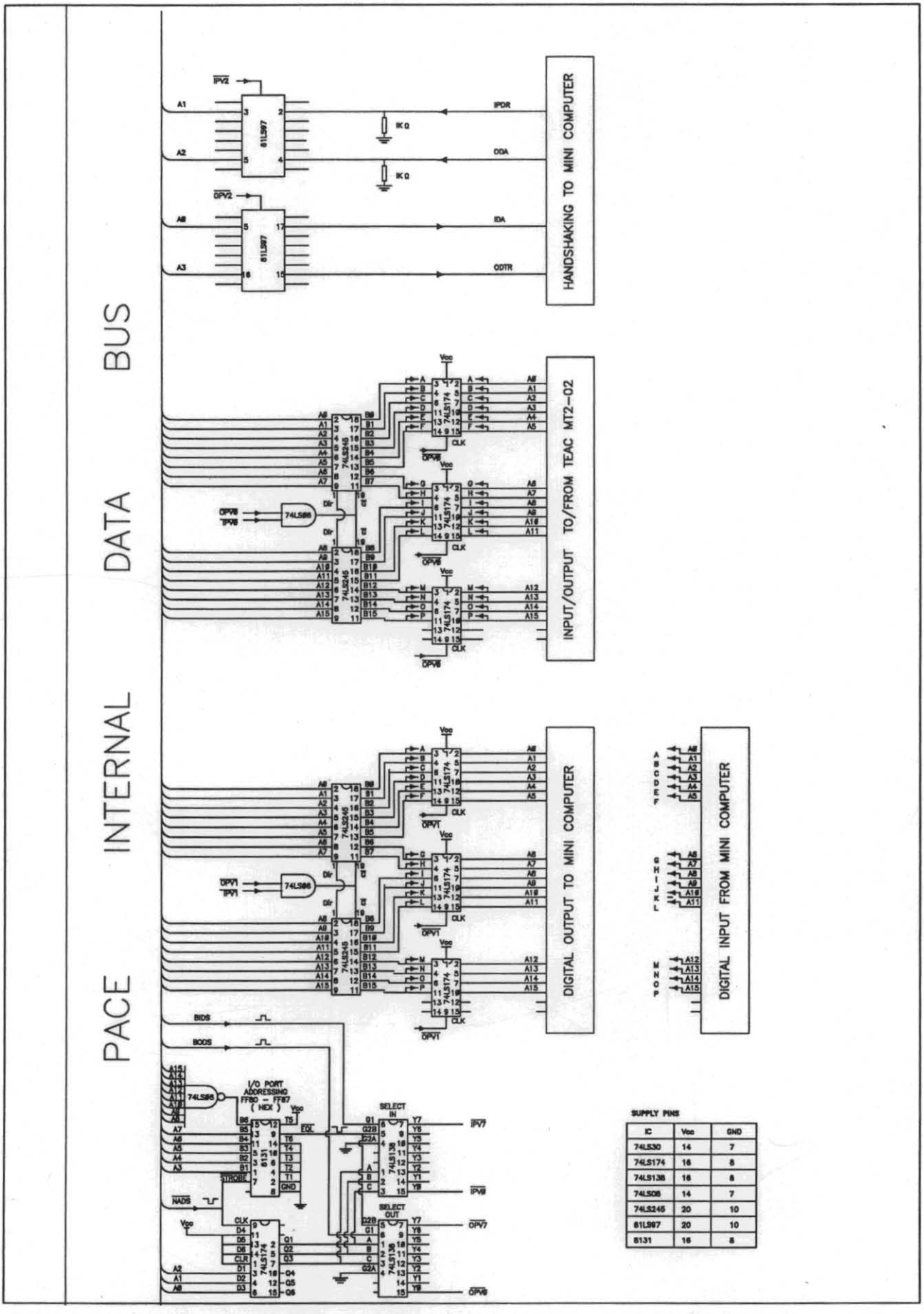
THE 2'S COMPLEMENT CONVERSIONS ARE 12 BIT WIDE AND ARE RECORDED DIRECTLY FROM THE A/D CONVERTERS. THE TOP FOUR BITS OF BYTES 15 AND 17 ARE ZEROS.

THE 4 DIGIT BCD CONVERSION VALUES IN BYTES 3 TO 6 ARE COMPUTED FROM THE TOP 10 BITS OF THE 12 BIT A/D BINARY VALUE.

NOTE THAT THE DATA IS NOT RECORDED ON TAPE IN ASCII AND IS THUS NOT DIRECTLY COMPATABLE WITH MOST PRINTERS AND OTHER PERIPHERALS WITHOUT REFORMATTING.

APPENDIX 5

Interface circuit diagrams



IC	Vcc	GND
74LS30	14	7
74LS174	16	8
74LS138	16	8
74LS08	14	7
74LS245	20	10
81LS97	20	10
8131	16	8

APPENDIX 6

PACE computer control program

The following program resides in the memory of the PACE LCDS. A full listing of the code, plus the assembly language, is given below. The program resides in the CMOS battery-powered memory, from location 0900 (hex) to 0FFF (hex). To start the program it is necessary to turn the power on to the unit and type the following on the VDU keyboard:

SS F00 <CR>

This will start the program running. To halt the program for any reason just press the init button on the front panel.

FORMAT OF COMPUTER PROGRAM

; — Denotes comment

: Memory location : Machine code : Label : Assembly code : Comment :
4 chars 4 chars 6 chars 8 chars

MAIN PROGRAM

```
;*************************************************************************
;*                                                                       *
;*   PROGRAM TO DRIVE TEAC TAPE UNIT AND TRANSFER DATA TO PERKIN ELMER.  *
;*                                                                       *
;*                     LISTING OF SUBROUTINES USED                       *
;*                                                                       *
;*                  (MAIN PROGRAM AT END OF LISTING)     (30/09/86)      *
;*************************************************************************
                ;
                ;  CLOCK SUBROUTINE
                ;  GIVE 1 HIGH - LOW - HIGH PULSE TO THE TEAC TAPE DRIVE
                ;
    0900 C106   CLOCK:  LD      0,DAT1        ;CLOCK HIGH
    0901 D080           ST      0,80          ;OUTPUT TO TEAC
    0902 C105           LD      0,DAT2        ;CLOCK LOW
    0903 D080           ST      0,80          ;OUTPUT TO TEAC
    0904 C102           LD      0,DAT1        ;CLOCK HIGH
    0905 D080           ST      0,80          ;OUTPUT TO TEAC
    0906 8000           RTS                   ;RETURN FROM SUBROUTINE
    0907 FFFF   DAT1:   FFFF                  ;DATA
    0908 BFFF   DAT2:   BFFF                  ;DATA
                ;
                ;  TEAC RESET SUBROUTINE
                ;  INITIALIZE TEAC FOR OPERATION AS A
                ;  COMMAND CONTROL AND NOT DMA CONTROL
                ;
    090A C10A   RESET:  LD      0,DAT1        ;RESET PULSE
    090B D080           ST      0,80          ;OUTPUT TO TEAC
    090C 9440           JSR     CLOCK         ;CLOCK PULSE
    090D C108           LD      0,DAT2        ;CALL MDR0
    090E D080           ST      0,80          ;OUTPUT TO TEAC
    090F C107           LD      0,DAT3        ;SET COMMAND MODE
    0910 D080           ST      0,80          ;OUTPUT TO TEAC
    0911 C106           LD      0,DAT4        ;END MDR0
    0912 D080           ST      0,80          ;OUTPUT TO TEAC
    0913 9440           JSR     CLOCK         ;CLOCK PULSE
    0914 8000           RTS                   ;RETURN FROM SUBROUTINE
    0915 7FFF   DAT1:   7FFF                  ;DATA
    0916 F4FF   DAT2:   F4FF                  ;DATA
    0917 B4ED   DAT3:   B4ED                  ;DATA
    0918 F4ED   DAT4:   F4ED                  ;DATA
                ;
                ;  UNLOAD SUBROUTINE
                ;  REWIND TAPE TO CLEAR LEADER FOR UNLOADING
                ;
    091A C108   UNLOAD: LD      0,DAT1        ;SELECT CONTROL REGISTER
    091B D080           ST      0,80          ;OUTPUT TO TEAC
    091C C107           LD      0,DAT2        ;SELECT REWIND
    091D D080           ST      0,80          ;OUTPUT TO TEAC
    091E C106           LD      0,DAT3        ;END CONTROL COMMAND
    091F D080           ST      0,80          ;OUTPUT TO TEAC
```

```
0920 9440           JSR     CLOCK          ;CLOCK PULSE
0921 9458           JSR     CFR            ;CONTROLLER FREE?
0922 8000           RTS
0923 F5FF   DAT1:   F5FF                   ;DATA
0924 B575   DAT2:   B575                   ;DATA
0925 F575   DAT3:   F575                   ;DATA
            ;
            ;  REWIND SUBROUTINE
            ;  REWIND TO START OF TAPE AND SET LOAD POINT
            ;
0926 9442   REWIND: JSR     UNLOAD         ;REWIND TO CLEAR LEADER
0927 C109           LD      0,DAT1         ;SELECT CONTROL REGISTER
0928 D080           ST      0,80           ;OUTPUT TO TEAC
0929 C108           LD      0,DAT2         ;SELECT SET LOAD POINT
092A D080           ST      0,80           ;OUTPUT TO TEAC
092B C107           LD      0,DAT3         ;END CONTROL COMMAND
092C D080           ST      0,80           ;OUTPUT TO TEAC
092D 9440           JSR     CLOCK          ;CLOCK PULSE
092E 9450           JSR     CCE            ;WAIT FOR CONTROL COMMAND END
092F 944B           JSR     ERRORS         ;TEAC ERROR STATUS =} P.E.
0930 8000           RTS                    ;RETURN FROM SUBROUTINE
0931 F5FF   DAT1:   F5FF                   ;DATA
0932 B577   DAT2:   B577                   ;DATA
0933 F577   DAT3:   F577                   ;DATA
            ;
            ;  WRITE WORD COUNTER
            ;  WRITE INTO WORD COUNTER THE NUMBER OF
            ;  BYTES TO BE READ PER BLOCK
            ;
0934 C107   WCOUNT: LD      0,DAT1         ;SELECT WORD COUNT REGISTER
0935 D080           ST      0,80           ;OUTPUT TO TEAC
0936 C106           LD      0,DAT2         ;WRITE 252 BYTES INTO REG.
0937 D080           ST      0,80           ;OUTPUT TO TEAC
0938 C105           LD      0,DAT3         ;EXIT WORD COUNT REGISTER
0939 D080           ST      0,80           ;OUTPUT TO TEAC
093A 9440           JSR     CLOCK          ;CLOCK PULSE
093B 8000           RTS
093C F6FF   DAT1:   F6FF                   ;DATA
093D B603   DAT2:   B603                   ;DATA  COMP(03) = FC = 252
093E F603   DAT3:   F603                   ;DATA
            ;
            ;  CASSETTE STATUS REGISTER
            ;  READ IN CSR AND STORE INTO LOCATION 0000
            ;
0940 C109   CSR:    LD      0,DAT1         ;SELECT CASSETTE STATUS REG.
0941 D080           ST      0,80           ;OUTPUT TO TEAC
0942 C108           LD      0,DAT2         ;READ IN CSR COMMAND
0943 D080           ST      0,80           ;OUTPUT TO TEAC
0944 C480           LD      1,80           ;PUT CSR STATUS IN AC1
0945 D400           ST      1,0            ;PUT AC1 INTO LOCATION 0000
0946 C103           LD      0,DAT1         ;EXIT CSR
0947 D080           ST      0,80           ;OUTPUT TO TEAC
0948 9440           JSR     CLOCK          ;PULSE CLOCK
0949 8000           RTS
094A E3FF   DAT1:   E3FF                   ;DATA
094B A3FF   DAT2:   A3FF                   ;DATA
            ;
            ;  INTERRUPT STATUS REGISTER
            ;  READ IN ISR INVERT DATA AND STORE IN LOCATION 0001
            ;
094E C10A   ISR:    LD      0,DAT1         ;SELECT INTERRUPT STATUS REG.
094F D080           ST      0,80           ;OUTPUT TO TEAC
0950 C109           LD      0,DAT2         ;READ IN ISR COMMAND
0951 D080           ST      0,80           ;OUTPUT TO TEAC
0952 C480           LD      1,80           ;PUT ISR INTO AC1
0953 7100           CAI     1,0            ;COMPLEMENT ISR
0954 D401           ST      1,1            ;PUT AC1 INTO LOCATION 0001
0955 C103           LD      0,DAT1         ;EXIT ISR COMMAND
0956 D080           ST      0,80           ;OUTPUT TO TEAC
0957 9440           JSR     CLOCK          ;PULSE CLOCK
0958 8000           RTS
0959 E1FF   DAT1:   E1FF                   ;DATA
095A A1FF   DAT2:   A1FF                   ;DATA
```

```
                ;
                ;  ERROR STATUS REGISTER
                ;  READ IN ESR AND STORE IN LOCATION 0002
                ;
095C C109   ESR:    LD      0,DAT1          ;SELECT ERROR STATUS REGISTER
095D D080           ST      0,80            ;OUTPUT TO TEAC
095E C108           LD      0,DAT2          ;READ IN ESR COMMAND
095F D080           ST      0,80            ;OUTPUT TO TEAC
0960 C480           LD      1,80            ;PUT ESR INTO AC1
0961 D402           ST      1,2             ;PUT AC1 INTO LOCATION 0002
0962 C103           LD      0,DAT1          ;EXIT ESR COMMAND
0963 D080           ST      0,80            ;OUTPUT TO TEAC
0964 9440           JSR     CLOCK           ;PULSE CLOCK
0965 8000           RTS
0966 E2FF   DAT1:   E2FF                    ;DATA
0967 A2FF   DAT2:   A2FF                    ;DATA
                ;
                ;  READ AND STORE ONE BLOCK
                ;  READ ONE BLOCK FROM THE TEAC AND
                ;  GIVE TO THE PERKIN ELMER
                ;
0970 C10F   RBLOCK: LD      0,DAT1          ;MEMORY POINTER=1000
0971 D006           ST      0,6             ;STORE POINTER IN 0006
0972 9444           JSR     WCOUNT          ;TELL TEAC TO GET 252 BYTES
0973 9446           JSR     ISR             ;READ ISR
0974 C10C           LD      0,DAT2          ;SELECT COMMAND REGISTER
0975 D080           ST      0,80            ;OUTPUT TO TEAC
0976 C10B           LD      0,DAT3          ;READ ONE BLOCK COMMAND
0977 D080           ST      0,80            ;OUTPUT TO TEAC
0978 C10A           LD      0,DAT4          ;EXIT COMMAND REGISTER
0979 D080           ST      0,80            ;OUTPUT TO TEAC
097A 9440           JSR     CLOCK           ;PULSE CLOCK
097B 9446           JSR     ISR             ;READ ISR
097C 9449           JSR     DSTORE          ;READ AND STORE DATA IN PACE
097D 944A           JSR     DOUT            ;OUTPUT DATA TO THE P.E.
097E 944B           JSR     ERRORS          ;OUTPUT TEAC ERROR STATUS PE
097F 8000           RTS
0980 1000   DAT1:   1000                    ;DATA
0981 F5FF   DAT2:   F5FF                    ;DATA
0982 B5FB   DAT3:   B57B                    ;DATA
0983 F5FB   DAT4:   F57B                    ;DATA
                ;
                ;  OUTPUT DATA TO PERKIN ELMER
                ;  OUTPUT TWO 8 BIT WORDS AS A SINGLE
                ;  16 BIT WORD TO THE PERKIN ELMER
                ;
09A9 C113   DOUT:   LD      0,DAT2          ;POINTER FOR FIRST MEM LOCN.
09AA D006           ST      0,6             ;TEMP STORAGE LOCN FOR POINT.
09AB 5C00                                   ;NOP
09AC A006   L2:     LD@     0,6             ;PUT DATA FROM 1000 ON IN AC0
09AD 2810           SHL     0,8,0           ;MOVE AC0 TO THE LEFT 8 BITS
09AE 5D00           COPY    0,1             ;COPY AC0 INTO AC1
09AF 8C06           ISZ     6               ;NEXT MEMORY LOCATION
09B0 A006           LD@     0,6             ;PUT NEXT BIT OF DATA IN AC0
09B1 A909           AND     0,MASK          ;REMOVE THE TOP 8 BITS
09B2 6840           RADD    1,0             ;ADD AC0 TO AC1 RESULT IN AC0
09B3 D081           ST      0,81            ;PUT DATA ON OUTPUT FOR P.E.
09B4 944C           JSR     HANDIN          ;HANDSHAKE INPUT TO P.E.
09B5 8C06           ISZ     6               ;NEXT MEMORY LOCATION
09B6 C006           LD      0,6             ;PUT POINTER INTO AC0
09B7 F104           SKNE    0,DAT1          ;HAS ALL THE DATA GONE TO P.E
09B8 1901           JMP     L1              ;DATA HAS BEEN TRANSFERED
09B9 19F2           JMP     L2              ;CONTINUE TO PASS DATA
09BA 8000   L1:     RTS
09BB 00FF   MASK:   00FF                    ;8 BIT MASK
09BC 10FC   DAT1:   10FC                    ;DATA. LAST MEMORY LOCATION
09BD 1000   DAT2:   1000                    ;DATA. FIRST MEMORY LOCATION
                ;
                ;  ERRORS
                ;  JOIN THE CSR AND ESR INTO ONE WORD
                ;  AND GIVE TO PERKIN ELMER.
                ;
09C0 9445   ERRORS: JSR     CSR             ;READ IN CSR
```

```
09C1 9447           JSR     ESR             ;READ IN ESR
09C2 C400           LD      1,0             ;PUT CSR IN AC1
09C3 2910           SHL     0,8,0           ;MOVE TO LEFT 8 BITS
09C4 C002           LD      0,0             ;PUT ESR IN AC0
09C5 A904           AND     0,MASK          ;REMOVE THE TOP 8 BITS
09C6 6840           RADD    1,0             ;ADD AC0 TO AC1 RESULT IN AC0
09C7 D081           ST      0,81            ;PUT DATA ON OUTPUT FOR P.E.
09C8 944C           JSR     HANDIN          ;HANDSHAKE INPUT TO P.E.
09C9 8000           RTS
09CA 00FF   MASK:   00FF                    ;8 BIT MASK

            ;
            ;  HANDSHAKE INPUT TO P.E.
            ;  HANDSHAKE BY PULSING INPUT DATA TRANSFER REQUEST ONCE
            ;  THEN WAIT FOR A REPLY PULSE FROM THE P.E.
            ;
09D0 5001   HANDIN: LI      0,1             ;PUT 1 INTO AC0
09D1 D083           ST      0,83            ;PULSE DATA TRANSFER REQUEST
09D2 C082   L1:     LD      0,82            ;PUT HANDSHAKE REPLY INTO AC0
09D3 4401           BOC     5,L2            ;EXIT IF DATA ACCEPTED
09D4 19FD           JMP     L1              ;WAIT FOR HANDSHAKE REPLY
09D5 8000   L2:     RTS
            ;
            ;  HANDSHAKE OUTPUT TO P.E.
            ;  HANDSHAKE BY PULSING OUTPUT DATA TRANSFER REQUEST ONCE
            ;  THEN WAIT FOR A REPLY PULSE FROM THE P.E.
            ;
09DA 5008   HANDOT: LI      0,8             ;PUT 8 INTO AC0
09DB D083           ST      0,83            ;PULSE DATA TRANSFER REQUEST
09DC C082   L1:     LD      0,82            ;PUT HANDSHAKE REPLY INTO AC0
09DD 4601           BOC     7,L2            ;EXIT IF DATA ACCEPTED
09DE 19FD           JMP     L1              ;WAIT FOR HANDSHAKE REPLY
09DF 8000   L2:     RTS
            ;
            ;  SKIP ONE BLOCK
            ;  GO FAST FORWARD ONE BLOCK DO NOT READ DATA
            ;
09E5 C109   SKIPBL: LD      0,DAT1          ;SELECT CONTROL REGISTER
09E6 D080           ST      0,80            ;OUTPUT TO TEAC
09E7 C108           LD      0,DAT2          ;SELECT SKIP ONE BLOCK
09E8 D080           ST      0,80            ;OUTPUT TO TEAC
09E9 C107           LD      0,DAT3          ;END CONTROL COMMAND
09EA D080           ST      0,80            ;OUTPUT TO TEAC
09EB 9440           JSR     CLOCK           ;CLOCK PULSE
09EC 9450           JSR     CCE             ;WAIT FOR EXECUTION COMPLETE
09ED 944B           JSR     ERRORS          ;TEAC ERROR STATUS =} P.E.
09EE 8000           RTS
09EF F5FF   DAT1:   F5FF                    ;DATA
09D0 B579   DAT2:   B579                    ;DATA
09D1 F579   DAT3:   F579                    ;DATA
            ;
            ;  REVERSE ONE BLOCK
            ;  GO FAST REVERSE ONE BLOCK DO NOT READ DATA
            ;
09F3 C109   REVBLK: LD      0,DAT1          ;SELECT CONTROL REGISTER
09F4 D080           ST      0,80            ;OUTPUT TO TEAC
09F5 C108           LD      0,DAT2          ;SELECT REVERSE ONE BLOCK
09F6 D080           ST      0,80            ;OUTPUT TO TEAC
09F7 C107           LD      0,DAT3          ;END CONTROL COMMAND
09F8 D080           ST      0,80            ;OUTPUT TO TEAC
09F9 9440           JSR     CLOCK           ;CLOCK PULSE
09FA 9450           JSR     CCE             ;WAIT FOR EXECUTION COMPLETE
09FB 944B           JSR     ERRORS          ;TEAC ERROR STATUS =} P.E.
09FC 8000           RTS
09FD F5FF   DAT1:   F5FF                    ;DATA
09FE B578   DAT2:   B578                    ;DATA
09FF F578   DAT3:   F578                    ;DATA
            ;
            ;  CONTROL COMMAND END
            ;  WAIT UNTIL CONTROL COMMAND END HAS BEEN DETECTED
            ;  BEFORE RETURNING TO THE CALL SUBROUTINE
            ;
0A05 9446   CCE:    JSR     ISR             ;READ IN ISR
```

```
0A06 C001           LD      0,1             ;PUT ISR INTO AC0
0A07 A904           AND     0,MASK          ;CHECK CONTROL COMMAND END
0A08 F103           SKNE    0,MASK          ;JMP 1 INSTR. IF NOT(CCE)
0A09 1901           JMP     L1              ;CONTROL COMMAND ENDED
0A0A 19FA           JMP     CCE             ;WAIT FOR CCE
0A0B 8000   L1:     RTS
0A0C 0080   MASK:   0080                    ;CCE MASK
            ;
            ;  SEARCH TAPE MARK
            ;  LOOK FOR NEXT TAPE MARK IN SLOW FORWARD
            ;  DO NOT READ DATA AND HALT AT THE END OF COMMAND
            ;
0A10 C109   SRCHTM: LD      0,DAT1          ;SELECT CONTROL REGISTER
0A11 D080           ST      0,80            ;OUTPUT TO TEAC
0A12 C108           LD      0,DAT2          ;SELECT SEARCH TAPE MARK
0A13 D080           ST      0,80            ;OUTPUT TO TEAC
0A14 C107           LD      0,DAT3          ;END CONTROL COMMAND
0A15 D080           ST      0,80            ;OUTPUT TO TEAC
0A16 9440           JSR     CLOCK           ;CLOCK PULSE
0A17 9450           JSR     CCE             ;WAIT FOR EXECUTION COMPLETE
0A18 944B           JSR     ERRORS          ;TEAC ERROR STATUS =} P.E.
0A19 8000           RTS
0A1A F5FF   DAT1:   F5FF                    ;DATA
0A1B B573   DAT2:   B573                    ;DATA
0A1C F573   DAT3:   F573                    ;DATA
            ;
            ;  HIGH SPEED SEARCH
            ;  LOOK FOR NEXT TAPE MARK IN FAST FORWARD
            ;  DO NOT READ DATA AND HALT AT THE END OF COMMAND
            ;
0A20 C109   HSSTM:  LD      0,DAT1          ;SELECT CONTROL REGISTER
0A21 D080           ST      0,80            ;OUTPUT TO TEAC
0A22 C108           LD      0,DAT2          ;SELECT HIGH SPEED SEARCH
0A23 D080           ST      0,80            ;OUTPUT TO TEAC
0A24 C107           LD      0,DAT3          ;END CONTROL COMMAND
0A25 D080           ST      0,80            ;OUTPUT TO TEAC
0A26 9440           JSR     CLOCK           ;CLOCK PULSE
0A27 9450           JSR     CCE             ;WAIT FOR EXECUTION COMPLETE
0A28 944B           JSR     ERRORS          ;TEAC ERROR STATUS =} P.E.
0A29 8000           RTS
0A2A F5FF   DAT1:   F5FF                    ;DATA
0A2B B572   DAT2:   B572                    ;DATA
0A2C F572   DAT3:   F572                    ;DATA
            ;  WRITE ONE BLOCK
            ;  (NOT USED AS YET)
            ;
0A30 944B   WONEBL: JSR     ERRORS          ;TEAC ERROR STATUS =} P.E.
0A31 8000           RTS
            ;
            ;  WRITE TAPE MARK
            ;  (NOT USED AS YET)
            ;
0A40 944B   WRITTM  JSR     ERRORS          ;TEAC ERROR STATUS =} P.E.
0A41 8000           RTS
            ;
            ;  ERASE
            ;  (NOT USED AS YET)
            ;
0A50 944B   ERASE   JSR     ERRORS          ;TEAC ERROR STATUS =} P.E.
0A51 8000           RTS
            ;
            ;  SET LOAD POINT WITH ERASE
            ;  (NOT USED AS YET)
            ;
0A60 944B   SELDPE  JSR     ERRORS          ;TEAC ERROR STATUS =} P.E.
0A61 8000           RTS
            ;
            ;  NO OPERATION
            ;  ONLY RETURNS THE ERROR STATUS
            ;
0A70 944B   NOP     JSR     ERRORS          ;TEAC ERROR STATUS =} P.E.
0A71 8000           RTS
```

```
                ;
                ;  CONTROLLER FREE DETECT
                ;  USED TO DETECT WHEN THE CONTROLLER IS FREE
                ;  ie:WHEN THE MOTOR HAS STOPPED ON A REWIND
                ;
0A75 C10E   CFRE:   LD      0,DAT1      ;SELECT MDR1 REGISTER
0A76 D080           ST      0,80        ;OUTPUT TO TEAC
0A77 C10D           LD      0,DAT2      ;RESET CONTROLLER FLAG
0A78 D080           ST      0,80        ;OUTPUT TO TEAC
0A79 C10C           LD      0,DAT3      ;EXIT MDR1
0A7A D080           ST      0,80        ;OUTPUT TO TEAC
0A7B 9440   L2:     JSR     CLOCK       ;PULSE CLOCK
0A7C 9446           JSR     ISR         ;READ IN ISR
0A7D C001           LD      0,1         ;PUT ISR INTO AC0
0A7E A904           AND     0,MASK      ;CHECK FOR CONTROLLER FREE
0A7F F103           SKNE    0,MASK      ;JUMP 1 INTR. IF NOT(CFRE)
0A80 1901           JMP     L1          ;CONTROLLER IS FREE
0A81 19F9           JMP     L2          ;WAIT UNTIL CONTROLLER FREE
0A82 8000   L1:     RTS
0A83 0010   MASK:   0010                ;MASK FOR CFRE
0A84 F0FF   DAT1:   F0FF                ;DATA
0A85 B009   DAT2:   B00D                ;DATA
0A86 F009   DAT3:   F00D                ;DATA
                ;
                ;  READ AND STORE DATA
                ;  READ DATA INTO THE PACE AND STORE
                ;  INTO MEMORY LOCATIONS 1000-10FC
                ;
B00 C529            LD      1,DAT1      ;SELECT DBR DATA
B01 C929            LD      2,DAT2      ;READ DBR DATA
B02 CD2F            LD      3,DAT9      ;CLOCK HIGH DATA
B03 C080    L1      LD      0,80
B04 7000            CAI     0,0         ;LOOK FOR INTERRUPT REQUEST
B05 B925            SKAZ    0,DAT2
B06 1901            JMP     L2          ;INTERRUPT REQUEST AVAILABLE
B07 19FB            JMP     L1          ;INTERRUPT REQUEST NOT AVAIL.
B08 C125    L2      LD      0,DAT5      ;SELECT INTERRUPT STATUS REG.
B09 D080            ST      0,80        ;OUTPUT TO TEAC
B0A C124            LD      0,DAT6      ;READ ISR COMMAND
B0B D080            ST      0,80        ;OUTPUT TO TEAC
B0C C080            LD      0,80        ;PUT ISR IN AC0
B0D 7000            CAI     0,0         ;COMPLEMENT ISR
B0E D001            ST      0,01        ;STORE IN LOCATION 01
B0F B91F            SKAZ    0,DAT6      ;CHECK IF DATA IS AVAILABLE
B10 1901            JMP     L3          ;DATA AVAILABLE
B11 1912            JMP     L4          ;DATA NOT AVAILABLE
B12 DC80    L3      ST      3,80        ;CLOCK HIGH
B13 C118            LD      0,DAT3      ;PUT CLOCK LOW IN AC0
B14 D080            ST      0,80        ;CLOCK LOW
B15 DC80            ST      3,80        ;CLOCK HIGH
B16 D080            ST      0,80        ;CLOCK LOW
B17 5C00            NOP
B18 DC80            ST      3,80        ;CLOCK LOW
B19 D480            ST      1,80        ;SELECT DBR
B1A D880            ST      2,80        ;READ DATA BUFFER REGISTER
B1B C080            LD      0,80        ;PUT DBR IN AC0
B1C B006            ST@     0,6         ;STORE DATA IN MEMORY
B1D DC80            ST      3,80        ;CLOCK HIGH
B1E C10D            LD      0,DAT3      ;PUT CLOCK LOW INTO AC0
B1F D080            ST      0,80        ;CLOCK LOW
B20 DC80            ST      3,80        ;CLOCK HIGH
B21 D080            ST      0,80        ;CLOCK LOW
B22 DC80            ST      3,80        ;CLOCK HIGH
B23 8C06            ISZ     06          ;INCREMENT MEMORY POINTER
B24 C001    L4      LD      0,01        ;PUT OLD ISR IN AC0
B25 B90B            SKAZ    0,DAT8      ;CHECK CONTROL COMMAND END
B26 1901            JMP     L5          ;CONTROL FINISHED
B27 19DB            JMP     L1          ;CONTROL NOT FINISHED
B28 9440    L5      JSR     CLOCK       ;CLOCK
B29 8000            RTS
B2A E7FF    DAT1
B2B A7FF    DAT2
B2C BFFF    DAT3
```

```
B2D 2000    DAT4
B2D E1FF    DAT5
B2E A1FF    DAT6
B2F 0020    DAT7
B30 0080    DAT8
B31 FFFF    DAT9
            ;
            ;
            ;
            ;
            ;
            ;
            ;
            ;  SUBROUTINE POINTERS
            ;  POINTERS WILL BE MOVED FROM READ ONLY MEMORY LOCATIONS
            ;  C00-C18  TO THE BASE PAGE LOCATIONS 40-58  FOR EASIER
            ;  ADDRESSING FROM THE MAIN PROGRAM
            ;
0C00 0900   CLOCK:  0900                ;CLOCK
0C01 090A   RESET:  090A                ;RESET AND INITIALIZE TEAC
0C02 091A   UNLOAD: 091A                ;REWIND TAPE TO CLEAR LEADER
0C03 0926   REWIND: 0926                ;REWIND AND SET LOAD POINT
0C04 0934   WCOUNT: 0934                ;WRITE TO WORD COUNTER
0C05 0940   CSR:    0940                ;CASSETTE STATUS REGISTER
0C06 094E   ISR:    094E                ;INTERRUPT STATUS REGISTER
0C07 095C   ESR:    095C                ;ERROR STATUS REGISTER
0C08 0970   RBLOCK: 0970                ;READ ONE BLOCK COMMAND
0C09 0B00   DSTORE: 098A                ;READ IN DATA AND STORE(PACE)
0C0A 09A9   DOUT:   09AA                ;OUTPUT 126,16 BIT WORDS(P.E)
0C0B 09C0   ERRORS: 09C0                ;OUTPUT CSR,ESR TO P.E.
0C0C 09D0   HANDIN: 09D0                ;HANDSHAKE INPUT TO P.E.
0C0D 09DA   HANDOT: 09DA                ;HANDSHAKE OUTPUT FROM P.E.
0C0E 09E5   SKIPBL: 09E5                ;SKIP ONE BLOCK
0C0F 09F3   REVBLK: 09F3                ;REVERSE ONE BLOCK
0C10 0A05   CCE:    0A05                ;CONTROL COMMAND END
0C11 0A10   SRCHTM: 0A10                ;SEARCH TAPE MARK
0C12 0A20   HSSTM:  0A20                ;HIGH SPEED SEARCH
0C13 0A30   WONEBL: 0A30                ;WRITE ONE BLOCK
0C14 0A40   WRITTM: 0A40                ;WRITE TAPE MARK
0C15 0A50   ERASE:  0A50                ;ERASE
0C16 0A60   SELDPE: 0A60                ;SET LOAD POINT WITH ERASE
0C17 0A70   NOP:    0A70                ;NO OPERATION
0C18 0A75   CFRE:   0A75                ;CONTROLLER FREE FLAG
            ;
            ;
            ;
            ;  START UP FOR PACE
            ;  MOVE THE POINTERS FOR THE SUBROUTINES
            ;  INTO THE BASE PAGE FOR EASY ADDRESSING
            ;
0EE0 5040   STARUP: LI      0,40        ;FIRST POINTER LOCATION
0EE1 D001           ST      0,1         ;PUT POINTER IN LOCN. 0001
0EE2 C10C           LD      0,SPOINT    ;POINT. START OF DATA TRANSFR
0EE3 D000           ST      0,0         ;PUT POINTER IN LOCN. 0000
0EE4 CC01   L1:     LD      3,1         ;PUT BASE PAGE POINTER IN AC3
0EE5 C800           LD      2,0         ;PUT PROT MEM POINTER IN AC2
0EE6 C200           LD      0,0(2)      ;PUT SUBROUTINE POINT. IN AC0
0EE7 D300           ST      0,0(3)      ;SUBROUTINE POINT.FROM 40=}
0EE8 8C00           ISZ     0           ;NEXT SUBROUTINE POINTER
0EE9 8C01           ISZ     1           ;NEXT BASE PAGE POINTER
0EEA C105           LD      0,FPOINT    ;LAST SUBROUTINE POINTER
0EEB F000           SKNE    0,0         ;IS CURRENT POINT{}LAST POINT
0EEC 1901           JMP     L2          ;TRANSFER HAS BEEN COMPLETED
0EED 19F6           JMP     L1          ;CONTINUE
0EEE 8000   L2:     RTS
0EEF 0C00   SPOINT: 0C00                ;START POINTER OF PROT MEM.
0EF0 0C30   FPOINT: 0C20                ;LAST POINTER OF PROT MEM.
            ;
            ;
            ;
            ;
```

```
;*************************************************************************
;*                                                                       *
;*                            MAIN PROGRAM                               *
;*                            ============                               *
;*                                                                       *
;*                  (WRITTEN 30/09/86.   R.SEDGMAN)                      *
;*                                                                       *
;*      THIS PROGRAM IS FOR TRANSFERRING DATA FROM A CASSETTE TO THE     *
;*             PERKIN ELMER COMPUTER VIA A TEAC TAPE DRIVE.              *
;*                                                                       *
;*    INSTRUCTIONS ARE SENT FROM THE P.E. TO THE PACE WHICH ACTS UPON    *
;*              THEM AND CONTROLS THE TEAC TAPE DRIVE.                   *
;*                                                                       *
;*     ALL INSTRUCTIONS TO AND FROM THE P.E. ARE DONE VIA HANDSHAKING    *
;*      THIS ALLOWS BOTH SYSTEMS TO BE IN SYNC WITHOUT TIEING UP TOO     *
;*              MUCH CPU TIME.                                           *
;*                                                                       *
;*      DATA FORMAT:      ___________________                  *         *
;*                       |1ST 8 BIT WORD | 2ND 8 BIT WORD |  (FROM TEAC)  *
;*                        ___________________                  *         *
;*                       |      1 * 16 BIT WORD              |  (TO P.E.)    *
;*                        ___________________                  *         *
;*                                                                       *
;*                                                                       *
;*   WHEN A COMMAND IS RECEIVED BY THE PACE IT ACTS UPON IT ACCORDINGLY.  *
;*   AT THE END OF EVERY COMMAND (EXCEPT UNLOAD) AN ERROR STATUS IS       *
;*   RETURNED TO THE P.E.                                                *
;*   EG:  THE COMMAND REWIND.                                            *
;*                              THE TEAC IS REWOUND TO CLEAR LEADER THEN *
;*   IS MOVED FORWARD TO THE LOAD POINT,IT THEN TELLS THE P.E THAT THE   *
;*   ERROR STATUS IS ON THE OUTPUT TO BE READ.                           *
;*   THE PACE WILL THEN WAIT FOR THE NEXT INSTRUCTION TO BE GIVEN,       *
;*   IF HOWEVER THE SEQUENCE OF SEND RECEIVE DATA IS NOT FOLLOWED THE    *
;*   SYSTEM WILL SIMPLY CRASH AND HAVE TO BE RESTARTED.                  *
;*                                                                       *
;*   TO START THE PROGRAM RUNNING SIMPLY SET THE PROGRAM COUNTER TO 0F00 *
;*   THIS WILL AUTOMATICALLY GET THE PROGRAM UNDER-WAY ,WHICH WILL BE    *
;*   WAITING FOR A COMMAND AT THIS POINT IN TIME.                        *
;*                                                                       *
;*                                                                       *
;*************************************************************************
                ;
                ;
                ;
0F00 15DF   MAIN:   JSR     STARUP          ;BOOT THE PACE
    0F01 9441           JSR     RESET           ;RESET THE TEAC
    0F02 944D   WAITIN: JSR     HANDOT          ;WAIT FOR INSTRUCTION FROM PE
    0F03 C081           LD      0,81            ;READ INSTRUCTION
    0F04 411F           BOC     1,L1            ;NOP
    0F05 2C02           SHRR    0,1,0           ;SHIFT AC0 RIGHT 1,NO LINK
    0F06 411F           BOC     1,L2            ;WRITE ONE BLOCK
    0F07 2C02           SHRR    0,1,0           ;SHIFT AC0 RIGHT 1,NO LINK
    0F08 411F           BOC     1,L3            ;WRITE TAPE MARK
    0F09 2C02           SHRR    0,1,0           ;SHIFT AC0 RIGHT 1,NO LINK
    0F0A 411F           BOC     1,L4            ;ERASE
    0F0B 2C02           SHRR    0,1,0           ;SHIFT AC0 RIGHT 1,NO LINK
    0F0C 411F           BOC     1,L5            ;READ ONE BLOCK
    0F0D 2C02           SHRR    0,1,0           ;SHIFT AC0 RIGHT 1,NO LINK
    0F0E 411D           BOC     1,L5            ;READ ONE BLOCK
    0F0F 2C02           SHRR    0,1,0           ;SHIFT AC0 RIGHT 1,NO LINK
    0F10 411D           BOC     1,L6            ;SKIP ONE BLOCK
    0F11 2C02           SHRR    0,1,0           ;SHIFT AC0 RIGHT 1,NO LINK
    0F12 411D           BOC     1,L7            ;REVERSE ONE BLOCK
    0F13 2C02           SHRR    0,1,0           ;SHIFT AC0 RIGHT 1,NO LINK
    0F14 411D           BOC     1,L8            ;REWIND,SET LOAD POINT
    0F15 2C02           SHRR    0,1,0           ;SHIFT AC0 RIGHT 1,NO LINK
    0F16 411D           BOC     1,L9            ;SET LOAD POINT WITH ERASE
    0F17 2C02           SHRR    0,1,0           ;SHIFT AC0 RIGHT 1,NO LINK
    0F18 411D           BOC     1,L10           ;UNLOAD
    0F19 2C02           SHRR    0,1,0           ;SHIFT AC0 RIGHT 1,NO LINK
    0F1A 4109           BOC     1,L1            ;NOP
    0F1B 2C02           SHRR    0,1,0           ;SHIFT AC0 RIGHT 1,NO LINK
    0F1C 411B           BOC     1,L11           ;SEARCH TAPE MARK
```

```
0F1D 2C02           SHRR    0,1,0       ;SHIFT AC0 RIGHT 1,NO LINK
0F1E 411B           BOC     1,L12       ;HIGH SPEED SEARCH
0F1F 2C02           SHRR    0,1,0       ;SHIFT AC0 RIGHT 1,NO LINK
0F20 4103           BOC     1,L1        ;NOP
0F21 2C02           SHRR    0,1,0       ;SHIFT AC0 RIGHT 1,NO LINK
0F22 4101           BOC     1,L1        ;NOP
0F23 1917           JMP     END         ;JUMP TO END (CONTINUE)
0F24 9457   L1:     JSR     NOP
0F25 1915           JMP     END         ;JUMP TO END (CONTINUE)
0F26 9453   L2:     JSR     WONEBL
0F27 1913           JMP     END         ;JUMP TO END (CONTINUE)
0F28 9454   L3:     JSR     WRITTM
0F29 1911           JMP     END         ;JUMP TO END (CONTINUE)
0F2A 9455   L4:     JSR     ERASE
0F2B 190F           JMP     END         ;JUMP TO END (CONTINUE)
0F2C 9448   L5:     JSR     RBLOCK
0F2D 190D           JMP     END         ;JUMP TO END (CONTINUE)
0F2E 944E   L6:     JSR     SKIPBL
0F2F 190B           JMP     END         ;JUMP TO END (CONTINUE)
0F30 944F   L7:     JSR     REVBLK
0F31 1909           JMP     END         ;JUMP TO END (CONTINUE)
0F32 9443   L8:     JSR     REWIND
0F33 1907           JMP     END         ;JUMP TO END (CONTINUE)
0F34 9456   L9:     JSR     SELDPE
0F35 1905           JMP     END         ;JUMP TO END (CONTINUE)
0F36 9442   L10:    JSR     UNLOAD
0F37 1903           JMP     END         ;JUMP TO END (CONTINUE)
0F38 9451   L11:    JSR     SRCHTM
0F39 1901           JMP     END         ;JUMP TO END (CONTINUE)
0F3A 9452   L12:    JSR     HSSTM
0F3B 19C6   END:    JMP     WAITIN      ;GO BACK AND WAIT FOR COMMAND
0F3C 0000           HALT                ;SHOULD NEVER GET HERE
```

APPENDIX 7

Downloading the PACE computer program from the Perkin Elmer

This program is used to dump the control program from the Perkin Elmer to the PACE LCDS, in case the program is corrupted or lost in the PACE memory.

The program has to be manually typed into the PACE LCDS, starting from memory location FA0. The write-protect switch on the CMOS memory board has to be in the *WRITE ENABLE* position.

To start this program type

SS FA0 <CR>

When the program has finished move the write enable switch to the *WRITE PROTECT* position. The main program is now ready to run.

MAIN PROGRAM

```
;*************************************************************************
;*                                                                       *
;*          PROGRAM TO TAKE A PROGRAM FROM THE PERKIN ELMER              *
;*          AND STORE IT MEMORY LOCATIONS 900 TO F40 (HEX).              *
;*                                                                       *
;*************************************************************************
                ;
                ;  MAIN PROGRAM
                ;
    0FA0 C10F           LD@     0,DAT2          ;FIRST MEMORY LOCATION
    0FA1 D006           ST      0,6             ;TEMP STORAGE LOCATION
    0FA2 5008   L2:     LI      0,8             ;PUT 8 INTO AC0 (MASK)
    0FA3 D083           ST      0,80            ;PULSE DATA TRANSFER REQUEST
    0FA4 C082   L3:     LD      0,82            ;GET HANDSHAKE REPLY
    0FA5 4601           BOC     7,L1            ;EXIT IF PE READY
    0FA6 19FD           JMP     L3              ;WAIT FOR HANDSHAKE REPLY
    0FA7 C081   L1:     LD      0,81            ;READ DATA
    0FA8 B006           ST@     0,6             ;STORE AWAY DATA
    0FA9 8C06           ISZ     6               ;NEXT MEMORY LOCATION
    0FAA C006           LD      0,6             ;PUT COUNTER INTO AC0
    0FAB F103           SKNE    0,DAT1,L5       ;HAVE WE FINISHED
    0FAC 1901           JMP     L4              ;YES
    0FAD 19F4   L5:     JMP     L2              ;NOT FINISHED
    0FAE 0000   L4:     HALT                    ;STOP
    0FAF 0F40   DAT1:   0F40                    ;DATA
    0FB0 0900   DAT2:   0900                    ;DATA
```