MTU - 130

UTILITY PROGRAMS

FEBRUARY, 1982 REV. A

MTU-130 UTILITY PROGRAMS

TABLE OF CONTENTS

UTILITY NAME: UPLOAD

PURPOSE: To transmit a text file to an external device through the serial port.

SYNTAX: UPLOAD <filename>

DESCRIPTION: UPLOAD is useful for communicating a CODOS standard text file to another system through the RS-232 interface on the MTU-130. The filename argument is the name of the CODOS text file that will be transmitted. If the filename is omitted, then the text to be transmitted is read from channel 5 which should have been preassigned to the file or device from which data is desired.

When UPLOAD is started, it will print: UPLOAD READY. START 4800 BAUD RECEIVER on the console device (display usually). It will then wait until it receives an ASCII DC3 (cntl/S or $13). At that point it begins transmitting the text file or data from channel 5 until end of file is reached. If during transmission a DC3 (cntl/S or $13) is received, transmission pauses until a DC1 is received. This is called "X-ON X-OFF protocol" and allows the receiving device to suspend transmission until it can "catch up". When end of file is reached, an ASCII ETX (cntl/C or $03) is sent and UPLOAD returns to CODOS with channel 5 freed.

EXAMPLES: UPLOAD MYTEXT.T

will transmit the contents of the file called MYTEXT.T located on the current default disk (usually drive 0).

UPLOAD

will transmit data read from whatever channel 5 is assigned to until end of file is reached.

NOTES: 1. If an ETX (cntl/C or $03) is encountered in the file, it is skipped and not transmitted.

2. Whenever a CR (cntl/M or $0D) is transmitted, an LF (cntl/J or $0A) is inserted immediately thereafter.

3. Up to two characters may be transmitted following receipt of DC1 before transmission is suspended.

4. Transmission format is 4800 baud, 8 data bits, 1 stop bit, no parity. See Patches section if a different rate or data format is desired.

5. A full duplex RS-232 communication line is required for the X-ON X-OFF protocol to work.

ERRORS:

1. FILE NOT FOUND - Specified file name was not found or channel 5 was not assigned.

PATCHES:

1. Baud rate - The least significant 4 bits of location B438 contains a code for the baud rate. Use the following table for a different baud rate:

HEX	BAUD	HEX	BAUD	HEX	BAUD	HEX	BAUD
0	-	4	134.5	8	1200	C	4800
1	50	5	150	9	1800	D	7200
2	75	6	300	A	2400	E	9600
3	110	7	600	B	3600	F	19200

2. When changing the baud rate, be careful to leave the most significant 4 bits alone (left hex digit should be 1) unless the data format needs to be changed. For more information, refer to the Programming section of the Monomeg manual in the Hardware Documentation portion of the MTU-130 manual.

3. Beginning of file character - Location B47D contains the ASCII code of the character that must be received before transmission starts.

4. End-of-file character - Location B4B6 contains the ASCII code of the character that is sent to signal end of file.

5. Stop transmission character - Location B4D3 contains the ASCII code of the character that suspends transmission when received.

6. Start transmission character - Location B4E3 contains the ASCII code of the character that restarts transmission after having been suspended.

For more extensive modification of UPLOAD, the assembler source code may be found in a file called UPLOAD.A.

UTILITY NAME: DOWNLOAD

PURPOSE: To receive a text file from an external device through the serial port.

SYNTAX: DOWNLOAD ⟨filename⟩

DESCRIPTION: Download is useful for receiving a text file from another system through the RS-232 interface on the MTU-130 and writing it on a CODOS file or other MTU-130 device. The filename argument is the name of the CODOS text file that will be created to store the text received. If the filename is omitted, then the text received is sent over channel 6 which should have been preassigned to the file or device which should get the data.

When DOWNLOAD is started, it will print: "DOWNLOAD READY. START 4800 BAUD TRANSMITTER." on the console device (display usually). It will then wait indefinitely until it receives the first character. That character and all succeeding ones are then saved in a large memory buffer. When the buffer becomes nearly full, a DC3 (cntl/S or $13) is sent out which is a signal that the transmitting system should suspend transmission temporarily. Anything received during the next second is put into the buffer too and then the buffer is written to the file or channel 6. After the buffer has been emptied, a DC1 (cntl/Q or $11) is sent which should cause the transmitting system to resume data transmission. This is called "XON-XOFF protocol" and allows the DOWNLOAD program to empty its buffer to disk without missing any transmitted characters.

End-of-file is assumed when an ETX (cntl/C or $03) character is received or when transmission is interrupted for more than 3 seconds without a DC3 having been sent to interrupt it. In either case, the partially full buffer is written to the file or device, channel 6 is freed, and DOWNLOAD returns to CODOS.

EXAMPLES: DOWNLOAD MYTEXT.T

will receive data from the serial interface and write it into a CODOS file named MYTEXT.T located on the current default disk (usually drive 0).

DOWNLOAD

will receive data from the serial interface and write it to whatever channel 6 is assigned to.

NOTES:

1. If receipt of an ETX (cntl/C or $03) triggers the end of file sequence, it is not stored in the file or sent to channnel 6.

2. Any LF (cntl/J or $0A) or NUL (cntl/@ or $00) characters received are discarded and are not stored in the file or sent to channel 6.

3. Reception format is 4800 baud, 8 data bits, 1 stop bit, no parity. See Patches section if a different rate or data format is desired.

4. A full duplex RS-232 communication line is required for the X-ON X-OFF protocol to work.

ERRORS:

1. FILE ALREADY EXISTS - The specified file name is already on the disk. Either delete the conflicting file or rerun DOWNLOAD with a different filename.

2. FILE UNSPECIFIED - No filename was specified and channel 6 had not been previously assigned to anything. Either assign channel 6 to a device or file or specify a filename (see example 1) when DOWNLOAD is run again.

PATCHES:

1. Baud rate - The least significant 4 bits of location B43C contains a code for the baud rate. Use the following table for a different baud rate:

HEX	BAUD	HEX	BAUD	HEX	BAUD	HEX	BAUD
0	-	4	134.5	8	1200	C	4800
1	50	5	150	9	1800	D	7200
2	75	6	300	A	2400	E	9600
3	110	7	600	B	3600	F	19200

2. When changing the baud rate, be careful to leave the most significant 4 bits alone (left hex digit should be 1) unless the data format needs to be changed. For more information, refer to the Programming section of the Monomeg manual in the Hardware Documentation portion of the MTU-130 manual.

3. End-of-file character - Location B4B0 contains the ASCII code of the character (normally $03) that will trigger end-of-file when received.

4. If end-of-file triggered by the 3 second timeout is not desired, change location B4AE to FB. When this is done, receipt of an ETX (or a substitute) character is the only way end-of-file can be recognized.

5. Stop transmission character - Location B4CA contains the ASCII code of the character that is sent when DOWNLOAD wants the transmitting system to pause.

6. Start transmission character - Location B4A0 contains the ASCII code of the character that is sent when DOWNLOAD wants the transmitting system to resume.

For more extensive modification of DOWNLOAD, the assembler source code may be found in a file called DOWNLOAD.A.

UTILITY NAME: SERLDR

PURPOSE: To receive a hex load file from the serial port and load it into memory.

SYNTAX: SERLDR [=<loadaddress>]

DESCRIPTION: SERLDR is used to load an object file created on another system into the memory of an MTU-130. The serial port is used to receive the ASCII hex load file in standard MOS Technology format (see next page for a description of MOS Technology hex load file format). If no argument is supplied, the code will be loaded into the addresses specified in the code itself. If an argument is supplied (it must be 1 to 4 hex digits preceeded by an =), then its value is subtracted from the load address specified by the first object record received to establish a load offset value. Thereafter, this offset is added to the load address of each object record received. The effect is to relocate the entire file received to the specified address.

When SERLDR is started, it will print: "SERLDR READY. START 4800 BAUD TRANSMITTER." on the console device (display usually). It will then wait indefinitely until the hex formatted data is received. When an end record is received, it will print: "SUCCESSFUL LOAD" and return to CODOS. After loading into memory, the data would typically be SAVEd onto a CODOS loadable file.

EXAMPLES: SERLDR

will receive MOS Technology standard hex object code from the serial interface and store it into memory at the addresses specified in the code itself.

```
SERLDR =1000
```

will receive MOS Technology standard hex object code from the serial interface and store it into memory starting at $1000. For example, if received code contained two blocks, one intended to load at $200-$765 and the other at $A00-$BFF, they will actually be loaded into memory at $1000-$1565 and $1800-$19FF.

NOTES:

1. SERLDR occupies addresses $00A5-$00AF and $B400-$B579 over and above those normally occupied by the operating system. Therefore no part of the data should be allowed to load into these addresses.

2. All characters except a semicolon are ignored before the first record or between records. There must be no extraneous characters within a record however.

3. The program may be stopped at any time by typing a cntl/C. It will respond by printing: "LOAD ABORTED" and then return to CODOS.

4. If an alternate load address is specified, it must not have a bank specification (bank 0 is assumed).

5. Each data byte is stored into memory as it is received before the checksum is verified. Therefore, a high error rate in the received data could cause the address field to become garbled and subsequent storing of the record in an unwanted area of memory.

6. Reception format is 4800 baud, 8 data bits, 1 stop bit, no parity. See Patches section if a different rate or data format is desired.

ERRORS:

1. ILLEGAL ARGUMENT - Either the alternate load address was not preceeded by an = or the argument was an illegal hex load address or a bank specification was included.

2. CHECKSUM ERROR - The checksum computed from the data in the record did not agree with the checksum field of the record. This can be caused by electrical noise on the serial line or an incorrect record format.

PATCHES:

1. Baud rate - The least significant 4 bits of location B415 contains a code for the baud rate. Use the following table for a different baud rate:

HEX	BAUD	HEX	BAUD	HEX	BAUD	HEX	BAUD
0	-	4	134.5	8	1200	C	4800
1	50	5	150	9	1800	D	7200
2	75	6	300	A	2400	E	9600
3	110	7	600	B	3600	F	19200

2. When changing the baud rate, be careful to leave the most significant 4 bits alone (left hex digit should be 1) unless the data format needs to be changed. For more information, refer to the Programming section of the Monomeg manual in the Hardware Documentation portion of the MTU-130 manual.

For more extensive modification of SERLDR, the assembler source code may be found in a file called SERLDR.A.

MOS TECHNOLOGY HEX LOAD FILE FORMAT

This format was developed as a method of sending arbitrary binary data over a communications channel designed to handle only text characters. It also has the advantage of being reasonably human readable. In this format, each byte is represented by two hexadecimal characters, the first representing the most significant 4 bits of the byte and the second representing the least significant 4 bits.

The overall data file is divided into records. Each record starts with a semicolon character and ends with two checksum bytes. Any character (such as CR and LF) may appear between records but only hex digits (0-9 and A-F) are allowed in the record itself. The detailed record format is given below:

BYTE #	IN CHKSUM	DESCRIPTION
0	YES	Count of data bytes in the record (referred to as N). An end record is signified by zero in this field.
1 & 2	YES	Load address of data bytes, least significant byte first.
3 - (N+2)	YES	N data bytes to be stored sequentially starting at the load address.
(N+3)&(N+4)	NO	16 bit checksum, is the unsigned sum of all bytes included in the checksum calculation. Least significant byte first

UTILITY NAME: SERDMP

PURPOSE: To dump memory as a hex load file over the serial port.

SYNTAX: SERDMP ⟨fromaddr⟩ [=⟨loadaddr⟩] ⟨toaddr⟩ . . .

DESCRIPTION: SERDMP is used to dump the contents of a section of MTU-100 memory through the serial port as an ASCII coded hex load file. This allows the MTU-130 to send arbitrary binary data to another system using only standard ASCII printable characters. Since the data is sent using standard MOS Technology object file format, most 6502 based computers will be able to load the data into their own memories.

Two arguments are required. The first argument specifies the first address in the block to be dumped while the second argument specifies the last address to dump. When there is no relocation argument, the address field of the code records sent specifies loading in the destination system at the same addresses the data was dumped from in the MTU-130. A relocation argument, if present, must be preceeded by an = and must be located between the fromaddr and toaddr arguments. The relocation argument will specify an alternate load address for the destination system. Thus a middle argument of =1000 would cause the data dumped from the MTU-130 to be loaded starting at $1000 in the destination system regardless of where it originally was in the MTU-130. Several independent blocks of memory may be dumped by giving several sets of arguments (two or three arguments per set).

When SERDMP is started, it will print: "SERIAL HEX DUMP AT 4800 BAUD." on the console device (display usually) and then immediately start sending data. It returns to CODOS when the contents of all of the specified addresses have been dumped.

EXAMPLES: SERDMP 1000 17FF

will format the 2048 bytes of data between $1000 and $17FF inclusive into an MOS Technology standard hex object code file and send it out over the serial port. The receiving system will load this data into its memory from $1000 to $17FF.

SERDMP 800 =2000 BFF

will format the 1024 bytes of data between $0800 and $0BFF inclusive into an MOS Technology standard hex object code file and send it out over the serial port. The receiving system will load this data into its memory from $2000 through $23FF.

SERDMP 2000 2145 3000 43FF 5000 =0 50FF

Will dump three blocks of data from MTU-130 memory; the first from $2000 through $2145, the second from $3000 through $43FF, and the third from 5000 through $50FF. This data will be loaded into the destination system's memory at $2000 through $2145, $3000 through $43FF, and at $0000 through $00FF.

NOTES:

1. SERDMP occupies addresses $00A3-$00AF and $B400-$B58F over and above those normally occupied by the operating system. Therefore these parts of memory will be changed when using SERDMP.

2. All code records sent have 16 bytes of data except possibly the last record in a block.

3. A CR (cntl/M or $0D) followed by an LF (cntl/J or $0A) is sent before each record.

4. A bank specification is not allowed on any of the arguments. Bank 0 is assumed.

5. An ETX (cntl/C or $03) character is sent after the checksum of the last record.

6. Transmission format is 4800 baud, 8 data bits, 1 stop bit, no parity. See the Patches section if a different rate or data format is desired.

7. See the SERLDR untility writeup for details on MOS Technology standard object code format.

ERRORS:

1. ILLEGAL OR MISSING ARGUMENT - One or more required arguments are missing or the = sign was omitted from a relocation specification or a relocation specification is misplaced or a non-hex character is encountered.

PATCHES:

1. Baud rate - The least significant 4 bits of location B415 contain a code for the baud rate. Use the following table for a different baud rate:

HEX	BAUD	HEX	BAUD	HEX	BAUD	HEX	BAUD
0	-	4	134.5	8	1200	C	4800
1	50	5	150	9	1800	D	7200
2	75	6	300	A	2400	E	9600
3	110	7	600	B	3600	F	19200

2. When changing the baud rate, be careful to leave the most significant 4 bits alone (left hex digit should be 1) unless the data format needs to be changed. For more information, refer to the Programming section of the Monomeg manual in the Hardware Documentation portion of the MTU-130 manual.

3. End-of-file character - Location B551 contains the ASCII code of the character (normally a $03) sent after the last record.

For more extensive modification of SERDMP, the assembler source code may be found in a file called SERDMP.A.

UTILITY NAME: SEND

PURPOSE: To send a short message out over the serial port

SYNTAX: SEND "⟨messagetext⟩"

DESCRIPTION: SEND is used to send one line messages to another system over the serial interface. This is particularly useful if the other system has a "remote console" feature because then the user can use SEND to give commands to the other system either directly from the MTU-130 console or from a CODOS "job" file.

When SEND is started, it scans the remainder of the command line for a " character. All characters following the " are transmitted until another " is seen. When the terminating " is seen, a CR (cntl/M or $0D) is sent and the utility returns to CODOS. If one wishes to send a " as part of the message, then two "s in a row can be used.

EXAMPLES: SEND "DELETE FILE Experiment_35"

will send the text DELETE FILE Experiment_35 followed by a carriage return through the serial port to the remote system. This would presumably cause that system to perform a delete file operation.

SEND "PRINT ""NOW IS THE TIME"""

will send the text PRINT "NOW IS THE TIME" followed by a carriage return through the serial port to the remote system.

NOTES:
1. Transmission format is 4800 baud, 8 data bits, 1 stop bit, no parity. See Patches section if a different rate or data format is desired.

ERRORS:
1. ILLEGAL OR MISSING ARGUMENT - Beginning or ending " missing or user failed to enter "" when a " embedded in the text was desired.

PATCHES:
1. Baud rate - The least significant 4 bits of location B416 contains a code for the baud rate. Use the following table for a different baud rate:

HEX	BAUD	HEX	BAUD	HEX	BAUD	HEX	BAUD
0	-	4	134.5	8	1200	C	4800
1	50	5	150	9	1800	D	7200
2	75	6	300	A	2400	E	9600
3	110	7	600	B	3600	F	19200

2. When changing the baud rate, be careful to leave the most significant 4 bits alone (left hex digit should be 1) unless the data format needs to be changed. For more information, refer to the Programming section of the Monomeg manual in the Hardware Documentation portion of the MTU-100 manual.

UTILITY NAME: RECEIVE

PURPOSE: To wait until a specified character is received from the serial interface.

SYNTAX: RECEIVE { "<singlecharacter>" / <hexnumber> }

DESCRIPTION: RECEIVE is used to wait until a remote system transmits a specific character. This is particularly useful if the other system has a "remote console" feature because then CODOS can be instructed to wait until that system sends its characteristic prompt character before continuing. RECEIVE would typically be used in conjunction with the SEND utility to allow a CODOS "job" file to control a remote system.

When started, RECEIVE will wait indefinitely for the specified character or byte value to be received and then return to CODOS. The wait may be terminated at any time by typing a cntl/C.

EXAMPLES: RECEIVE "-"

will wait until the - character is received from the serial interface and then returns to CODOS. This might be the prompt character of a remote system.

RECEIVE 0D

will wait until the byte value $0D (an ASCII carriage return) is received from the serial interface and then returns to CODOS.

NOTES:

1. Only one character may be specified.

2. The most significant bit of characters received is zeroed before being tested since ASCII is only a 7 bit character code.

3. Reception format is 4800 baud, 8 data bits, 1 stop bit, no parity. See the Patches section if a different rate or data format is desired.

ERRORS:

1. ILLEGAL OR MISSING ARGUMENT - Beginning or ending " is missing or there is more than one character between the beginning " and ending ".

PATCHES:

1. Baud rate - The least significant 4 bits of location B4DE contains a code for the baud rate. Use the following table for a different baud rate:

HEX	BAUD	HEX	BAUD	HEX	BAUD	HEX	BAUD
0	-	4	134.5	8	1200	C	4800
1	50	5	150	9	1800	D	7200
2	75	6	300	A	2400	E	9600
3	110	7	600	B	3600	F	19200

2. When changing the baud rate, be careful to leave the most significant 4 bits alone (left hex digit should be 1) unless the data format needs to be changed. For more information, refer to the Programming section of the Monomeg manual in the Hardware Documentation portion of the MTU-130 manual.

3. To look at all 8 bits of received characters, change location BFD0 to $FF.

UTILITY NAME: WAIT

PURPOSE: To wait a specified number of seconds before continuing.

SYNTAX: WAIT ⟨timedelay⟩

DESCRIPTION: WAIT is used to insert a delay into the execution of a CODOS "job" file. This function is very useful when preparing automated demonstrations where a delay is desired so that the results of the previous command may be viewed and understood. After the delay, the job file would proceed to the next command. WAIT can also be useful in automated remote operation of a system which does not signal its readiness for a new operation with a predictable prompt.

When started, WAIT will wait for the specified number of seconds and then return to CODOS. The wait may be terminated at any time by typing a cntl/C.

EXAMPLES: WAIT 10

will wait until 10 (decimal) seconds have elapsed and then return to CODOS.

WAIT 65535

will wait for 65535 seconds before returning to CODOS. Since this is over 18 hours, it is effectively a pause until the operator types cntl/C.

NOTES:

1. The argument is a decimal value and must be between 0 and 65535. A value of 0 will give a zero wait time.

2. Timing is performed with a timed loop and therefore is affected if the system is responding to interrupts during the wait period. Even without interrupts, the timing accuracy has not been optimized so errors of up to 1% can be expected.

ERRORS:

1. ILLEGAL ARGUMENT - Delay argument is missing or it is an invalid decimal number.

2. ARITHMETIC OVERFLOW - This CODOS error message indicates that the argument value is greater than 65535.

UTILITY NAME: BACKUP

PURPOSE: To make an exact duplicate of a CODOS diskette quickly.

SPECIAL HARDWARE: At least two disk drives are required.

SYNTAX: BACKUP

DESCRIPTION: BACKUP is designed to make the task of making duplicates of important diskettes faster and easier than the method offered by the standard CODOS command set. With a single BACKUP command, a single-sided duplicate diskette can be formatted, receive a copy of the data on the source diskette, and verified in approximately one minute. In addition this time remains constant regardless of the number of files on the source diskette. Double sided diskettes require about 2 minutes to duplicate their full megabyte of data.

BACKUP uses prompts on the console device to get information from the user and tell him what to do. These are generally self explanatory. To perform a backup, simply type the command name BACKUP and then a carriage return. The program will be loaded and will print the message:

```
CODOS DISK BACKUP PROGRAM
PLEASE INSERT MASTER DISKETTE INTO DRIVE 0 AND DUPLICATE DISKETTE INTO DRIVE 1.
DUPLICATE DISKETTE NEED NOT BE FORMATTED.
TYPE ANY CHARACTER WHEN READY.
```

At this point the operator should remove the system disk from drive 0 (unless of course it is the one to be backed up) and insert the disk to be copied. The disk to receive the copy should be inserted into drive 1. Since BACKUP does its own formatting, the copy diskette can be fresh from a new box. BE SURE YOU GET THE DISKS IN THE RIGHT DRIVES! The disk in drive 1 will be unconditionally written over just as surely as if the CODOS FORMAT command had been executed! Note that both diskettes must be single sided or both double sided; BACKUP cannot translate between the two formats (although the standard CODOS FORMAT and COPYF utilities can if necessary). When you are sure the right disks are in the right drives, hit the return key. You can always use the INT key to abort the program if you don't really want to go through with the copy.

The master disk on drive 0 will now be duplicated onto drive 1 a full track at a time. Because the two disks are not rotating in perfect synchronism, it is normal for the time between tracks to vary somewhat. Assuming no disk errors are encountered, BACKUP will print the following after about 1 minute:

```
BACKUP SUCCESSFUL!
TYPE Y IF YOU WISH TO MAKE ANOTHER BACKUP -
```

If you want to make additional copies of the same master disk or wish to backup other masters, type a Y and skip to the next paragraph. If you are finished making backups, simply type return which will cause the following message to appear:

```
REMOVE MASTER DISK FROM DRIVE 0 AND INSERT CODOS SYSTEM DISK.
TYPE ANY CHARACTER WHEN READY.
```

Do as the instructions say and then type return. The standard CODOS prompt should appear to indicate that the system is ready for another command.

If you had responded with a Y earlier, BACKUP will now print the following:

```
REMOVE DUPLICATE DISKETTE FROM DRIVE 1 AND INSERT ANOTHER BLANK DISK.
TYPE ANY CHARACTER WHEN READY.
```

Take the first copy disk out of drive 1 and put in another to receive the next copy. If necessary, change the master disk in drive 0 as well. When ready, type return. BACKUP will now loop back to the copy operation and reprompt when done as before.

NOTES:

1. BACKUP is designed to work efficiently with the CODOS standard sector skew of 2. If a disk with the "data acquisition skew" of 3 is backed up, operation will be very slow although still accurate. In addition, the copy disk will have a skew of 2 regardless of the skew of the source disk. See the FORMAT command in the CODOS manual for an explanation of skew.

2. BACKUP makes an exact copy of the master diskette. Thus if the master has been fragmented due to extensive deletions, the copy diskette will be fragmented as well. The COPYF utility must be used instead if one wishes to consolidate active index entries and data storage area on the copy disk.

3. BACKUP does not recognize the bad sector flag codes used by CODOS. Therefore only error free diskettes may be copied with BACKUP.

4. BACKUP uses the last two of CODOS's pool buffers therefore one should not have more than 4 files open when BACKUP is run.

5. If BACKUP is aborted in the middle of a copy operation for any reason (such as an unreadable disk), DO NOT attempt to use the partial copy in drive 1 since all of its tracks will not have been formatted.

6. In the MTU-130 system, software protection is intended to be implemented by matching the serial number on the protected software with the hardware serial number in the system rather than by copy protecting the program disk.

ERRORS:

1. *** DISKETTE IN DRIVE 1 MUST BE SINGLE SIDED SINCE MASTER IS SINGLE SIDED *** Remove the disk from drive 1, insert a single sided disk, and type a Y to continue.

2. *** DISKETTE IN DRIVE 1 MUST BE DOUBLE SIDED SINCE MASTER IS DOUBLE SIDED *** Remove the disk from drive 1, insert a double sided disk, and type a Y to continue.

3. *** SEEK OR RECALIBRATE ERROR, DISK DRIVE 0 PROBABLY NOT READY *** - Make sure a disk is inserted into drive 0 and the door is properly shut, then type Y to continue.

4. *** SEEK OR RECALIBRATE ERROR, DISK DRIVE 1 PROBABLY NOT READY *** - Make sure a disk is inserted into drive 1 and the door is properly shut, then type Y to continue.

5. *** CANNOT FORMAT DISK IN DRIVE 1, IS PROBABLY WRITE PROTECTED *** - Remove the disk in drive 1 and make sure a write allow sticker is covering the write protect slot. (note this is OPPOSITE the write protect convention of 5" diskettes.) Reinsert the disk and type a Y to continue.

6. *** PERMANENT READ ERROR ON MASTER DISK *** - A sector on the master disk is unreadable after 10 retries. In some cases removing the disk and re-inserting it in the drive alters the alignment enough to make the disk readable. Typing a Y will retry reading the track up to 10 more times. Typing anything else will abort the program. If the sector simply cannot be read, the CODOS COPYF utility should be used to recover as much data as possible from the master disk. This message will also occur if any bad sectors on the master disk had been flagged by the CODOS FORMAT command when it was created.

7. *** WRITE RETRY ON DRIVE 1 TRACK xx *** - This is a warning message to indicate that a write operation on the copy diskette was unsuccessful. Writing on the indicated track will be retried up to 3 times before error 8 is generated. While this error is not serious, it does serve to keep the user informed of the quality of the disks being used for backup.

8. *** PERMANENT WRITE ERROR ON DISK IN DRIVE 1 *** - This message indicates that the copy diskette is defective and cannot be used by BACKUP. The disk may still be usable by CODOS however since the FORMAT command will bypass bad sectors. Remove the defective disk from drive 1, insert another blank disk, and type Y to try the backup again. Typing anything else will abort the program.

9. *** PERMANENT VERIFY ERROR ON DISK IN DRIVE 1 *** - This message indicates that the copy diskette is defective and cannot be used by BACKUP. The disk may still be usable by CODOS however since the FORMAT command will bypass bad sectors. Remove the defective disk from drive 1, insert another blank disk, and type Y to try the backup again. Typing anything else will abort the program.

UTILITY NAME: DISKETTE

PURPOSE: To directly read or write any sector on a CODOS formatted diskette.

SYNTAX: DISKETTE {READ / WRITE} <memaddr> <drive> <track> <sector> [<sectorcount>]

DESCRIPTION: The DISKETTE utility is used to read or write any individual sector or group of sectors on a CODOS formatted diskette directly by track and sector number. This capability is useful for direct on-disk patching (such as the operating system itself), for deciphering non-CODOS formats, some types of debugging, and for analyzing or restoring an unreadable diskette. The detailed error diagnostics provided also make it useful in diagnosing disk drive problems.

The syntax of the command line is shown above. If there are no arguments (i.e., just the command DISKETTE), a short printout describing how to use DISKETTE will be produced. When the first argument is the word READ, data will be read from the diskette and placed into memory. If it is write, data will be written onto the diskette from memory. The second argument is the hex address of the beginning of the "buffer" area in memory that will receive/supply the data to be read/written. Data transfers are always to or from sequential memory locations. The third argument is the drive number either 0, 1, 2, or 3. The fourth argument is the track number which is normally specified in hex but may be specified in decimal if a decimal point preceeds the number. The fifth argument is the sector number which may also be either hex or decimal. The last argument is optional. If it is absent, only one sector will be read or written. If present, it specifies the number of consecutive sectors to read or write. The sectors are read/written in consecutive order. The program will automatically go to the next track when all of the sectors on the specified track have been read or written. If a double sided diskette is to be read or written, it is simply regarded as having 52 sectors (instead of 26) per track.

EXAMPLES: DISKETTE READ 1000 1 20 0 .26

will read the diskette in drive 1 into memory starting at $1000. Reading on the diskette will start at hex track 20 (decimal track 32) and sector 0. All 26 (decimal) sectors on that track will be read into memory, a total of 6.5K.

DISKETTE WRITE 3B23 0 5 16

will write onto the diskette in drive 0 (the CODOS system disk) from memory starting at 3B23. Only one sector will be written and it will be sector $16 on track 5. Presumably there would be a good reason for doing this since this position would likely be in the middle of a CODOS system program.

DISKETTE

will print a brief summary of the command syntax for DISKETTE on the console device to serve as a memory aid when the manual is not readily available.

NOTES:

1. CODOS disk format is double density on all tracks, 26 sectors of 256 bytes each per track. Double sided disks have 52 sectors per track; 0-25 on side 0 and 26-51 on side 1. Sector and track skew have no effect except on speed of data transfer.

2. The memory address argument must not specify a bank number. All data transfers are to or from bank 0.

3. The sector numbering sequence used by CODOS starts with zero. The formats used by some IBM systems start sector numbering at 1 and thus would give an error if you attempt to read sector zero.

4. DISKETTE uses memory locations 0000-000A and B000-BC5F therefore they must not be overwritten by a READ operation.

5. DISKETTE uses the last CODOS pool buffer therefore one should not have more than 5 files open when DISKETTE is run.

6. Be sure you know what you are doing before using the WRITE argument on a CODOS disk that may be used later. Side-effects of alteration of the block assignment table or other file management data may not show up for months!

7. DISKETTE can handle an alternate disk format of 1024 byte sectors and 8 sectors per track. To specify the alternate format, use the keywords READ8 and WRITE8 instead of READ and WRITE as the first argument.

ERRORS:

1. MISSING ARGUMENT OR INVALID DELIMITER - A typing error has made the command line undecipherable. Carefully check the syntax requirements on the previous page.

2. ARGUMENT OUT OF RANGE - The argument to the left of the pointer printed along with this message is too large. Track numbers must be less than $4D (.77), sector numbers must be less than $34 (.52), and the number of sectors must be $A0 (.160) or less which is 40K bytes of data.

3. SELECTED DISK IS NOT TWO SIDED - A sector number greater than 25 was specified and the diskette in the specified drive is single sided.

4. SELECTED DRIVE NOT READY - No diskette in specified drive or door not closed.

5. SEEK ERROR ON SELECTED DRIVE - Disk drive is not ready or a drive failure.

6. RECALIBRATE ERROR ON SELECTED DRIVE - Disk drive is not ready or a drive failure.

7. RAN OFF END OF DISK - While reading or writing multiple sectors, the sector count was not exhausted when the last sector of the last track was processed.

8. DISK ERROR XXXXXX TRACK YY SECTOR ZZ WILL RETRY - A disk error was detected at the reported track and sector. The 6 digit error number is the contents of the first 3 status registers in the disk controller chip and can be deciphered by referring to the uPD765 data sheet in the Disk Controller manual. Up to 10 retries will be performed before the error is declared permanent.

9. PERMANENT DISK ERROR AT TRACK YY SECTOR ZZ, ERROR ANALYSIS FOLLOWS: - After 10 retries a read or write error on the reported track and sector cannot be corrected. The error analysis printed is an interpretation of the disk controller status register contents. The messages in the analysis should be self-explanatory. For further interpretation, refer to the uPD765 data sheet in the Disk Controller Manual.

UTILITY NAME: BROWSE

PURPOSE: To permit convenient scanning of CODOS text files without using an editor.

SYNTAX: BROWSE <filename>[:<drive>]

DESCRIPTION: The BROWSE utility is intended to be used for examining text files at the console which are too large to be conveniently TYPEd. By using BROWSE, the user can quickly display any portion of the file at random and search for certain words or phrases much like an infinite scrolling text editor. Unlike an editor however, BROWSE cannot alter the file and thus is "safer" when one wishes to only scan the file.

When started, BROWSE will scan through the entire file and produce a line-by-line index in memory. This index is used by the BROWSE program to randomly access specified lines in the file. BROWSE builds the index at about 10K bytes per second so large files may require a few seconds to index. Once BROWSE has built its index, it will display " LINE 1 " followed the first 20 lines of the file, followed by the prompt:

BROWSE

If you enter a carriage return, BROWSE will erase the screen and display the next 20 lines of the file. To display 20 lines starting at a different line number, type in the desired line number followed by a carriage return. For example:

BROWSE 2500 (underlining indicates characters typed by the user)

will cause BROWSE to display lines 2500 to 2519. if you enter a signed number (either + or -) then BROWSE will backup (if -) or move forward (if +) the specified number of lines relative to the first line currently displayed. For example, if the last display started at 100, then entering "-10" will display beginning at line 90. Every carriage return or formfeed ($0C) in the file counts as one line. Form feeds are displayed as a carriage return.

You can also search for a text string, beginning with the top line displayed. For example:

BROWSE 'ERROR 34'

will search for "ERROR 34". You can optionally specify an ending line number for the search after the string, for example:

BROWSE 'ASL A' 200

searches for "ASL A" up to line 200. You may not use the signed number form for the limit, however.

To permanently exit BROWSE, enter any command starting with "Q" (QUIT). You may temporarily exit BROWSE by using CNTRL-C and re-enter it with a GO 4003 command, provided you have not disturbed any memory above $4000. There is no provision for saving the index for future reuse.

ERRORS:
1. ? - This is the response to any command that BROWSE does not understand.

UTILITY NAME: VMDUMPMX80 (MX-70/80 printer)

PURPOSE: To print an exact image of the MTU-130 screen on an Epson MX-70 or MX-80 (with "Graftrax" ROM) printer.

SYNTAX: VMDUMPMX80 [NL] [NS]

DESCRIPTION: This program will print an exact dot-for-dot image of the current MTU-100 screen contents on an Epson MX-80 printer. Before entering the command, make sure the MX-80 is powered up and ready and that the paper is positioned as desired. When the program is started, the current screen contents will be printed and then the program will return to CODOS. Approximately 2 minutes will be required to print the screen independent of what is being displayed for the MX-70 If you wish to stop the printing before it is done, hold cntl/C down until the end of a print line. This will insure that the program stops correctly and leaves the printer in the proper state for subsequent text printing.

By just giving the command name with no arguments, the entire screen is printed including the function key legends if any. In addition, the paper will be fed 1.94 inches after printing to provide spacing before whatever will be printed next. If the argument NL (No Legends) is included, then only the top 240 scan lines will be printed. If the argument NS (No Spacing) is included, then there will be no spacing after printing.

EXAMPLES: VMDUMPMX80

will print the entire 256 x 480 screen area and then feed the paper 1.94 inches.

VMDUMPMX80 NL

will print the top 240 x 480 screen area and then feed the paper 2.17 inches.

VMDUMPMX80 NS

will immediately print the entire screen area without any spacing.

VMDUMPMX80 NL NS

will immediately print the top 240 x 480 screen area without any spacing.

NOTES:

1. Since any given MTU-130 installation is unlikely to have more than one kind of printer, you may wish to delete the other versions of VMDUMP and rename VMDUMPMX80 to VMDUMP.

2. Using the INT or RESET key to stop the printout will likely leave the printer in graphics mode and thus interfere with subsequent use as a character printer. To clear this condition, turn the MX-80 off then on again.

3. Final image size on paper will be 3.5" high and 8" wide. When using standard print spacing (no NS argument), two images will fit on an 11 high page with spacing such that top and bottom margins on the paper are preserved even if a large number of images are printed.

4. The MX-80 dot spacing aspect ratio (dX/dY) is 1.2 instead of 1.0 therefore circles and other aspect ratio sensitive images will appear to be slightly elongated horizontally.

5. Use of the NS argument will suppress all vertical spacing between images. This can be useful in concatenating up to 3 images vertically per 11" page for an effective matrix size of 480 (X) by 768 (Y).

6. VMDUMPMX80 may be called as a subroutine. The entry point is B403 and all registers are altered. In addition CODOS User Registers U0 - U4 are altered. Cntl/C will halt printing and return to the caller.

7. VMDUMPMX80 occupies memory locations B400-B563

8. The MX-80 does not provide any exceptional condition feedback to the MTU-130 therefore you must be careful to avoid exhausting the paper supply and possibly damaging the ribbon or print head.

ERRORS:

1. ILLEGAL ARGUMENT - something other than NS or NL or blanks was seen on the command line.

UTILITY NAME: VMDUMPIDS440 (Integral Data Systems Paper Tiger model 440)

PURPOSE: To print an exact image of the MTU-130 screen on the Paper Tiger printer.

SYNTAX: VMDUMPMIDS440 [NL] [NS]

DESCRIPTION: This program will print an exact dot-for-dot image of the current MTU-130 screen contents on an Integral Data Systems Paper Tiger printer. Before entering the command, make sure the Paper Tiger is powered up and ready and that the paper is positioned as desired. When the program is started, the current screen contents will be printed and then the program will return to CODOS. Approximately 1 minute will be required to print the screen independent of what is being displayed. If you wish to stop the printing before it is done, hold cntl/C down until the program stops. This will insure that the program stops correctly and leaves the printer in the proper state for subsequent text printing.

By just giving the command name with no arguments, the entire screen is printed including the function key legends if any. In addition, the paper will be fed 1.94 inches after printing to provide spacing before whatever will be printed next. If the argument NL (No Legends) is included, then only the top 240 scan lines will be printed. If the argument NS (No Spacing) is included, then there will be no spacing after printing.

EXAMPLES: VMDUMPIDS440

will print the entire 256 x 480 screen area and then feed the paper 1.94 inches.

VMDUMPIDS440 NL

will print the top 240 x 480 screen area and then feed the paper 2.17 inches.

VMDUMPIDS440 NS

will immediately print the entire screen area without any spacing.

VMDUMPIDS440 NL NS

will immediately print the top 240 x 480 screen area without any spacing.

NOTES:

1. To work properly, the IDS-440 must be jumpered for 12 characters per inch.

2. Since any given MTU-130 installation is unlikely to have more than one kind of printer, you may wish to delete the other versions of VMDUMP and rename VMDUMPIDS440 to VMDUMP.

3. Using the INT or RESET key to stop the printout will likely leave the printer in graphics mode and thus interfere with subsequent use as a character printer. To clear this condition, turn the printer off then on again.

4. Final image size on paper will be 3.6" high and 7.48" wide. When using standard print spacing (no NS argument), two images will fit on an 11 high page with spacing such that top and bottom margins on the paper are preserved even if a large number of images are printed.

5. The IDS-440's dot spacing aspect ratio (dX/dY) is 1.11 (1.16 with 50Hz power) instead of 1.0 therefore circles and other aspect ratio sensitive images will appear to be slightly elongated horizontally.

6. Because of the Paper Tiger's mechanical construction, there will be a slight space (2 dot rows) between images when printing the entire screen with the NS argument. If absolutely no spacing is required, specify both NS and NL.

7. Use of the NS and NL arguments together will suppress all vertical spacing between images. This can be useful in concatenating up to 3 images vertically per 11" page for an effective matrix size of 480 (X) by 720 (Y).

6. VMDUMPIDS440 may be called as a subroutine. The entry point is B403 and all registers are altered. In addition CODOS User Registers U0 - U4 are altered. Cntl/C will halt printing and return to the caller.

7. VMDUMPIDS440 occupies memory locations B400-B544.

8. The recommended interface between the IDS440 and the MTU-130 does not provide any exceptional condition feedback to the MTU-130 therefore you must be careful to avoid exhausting the paper supply and possibly damaging the ribbon or print head.

ERRORS:

1. ILLEGAL ARGUMENT - something other than NS or NL or blanks was seen on the command line.

UTILITY NAME: TERM

PURPOSE: To simulate the operaton of a communications terminal with paper tape.

SYNTAX: TERM baudrate ECHO

DESCRIPTION: This program will allow the MTU-130 to simulate a teletype-like communications terminal with user controllable floppy disk storage of incoming and outgoing data. Overall operation closely resembles that of a teletype with paper tape but with the advantages of fast and quiet operation and reusability of the storage medium.

The first argument on the command line is the transmission and reception baud rate and must be present. Allowable values are 50, 75, 110, 135, 150, 300, 600, 1200, 1800, 2400, 3600, 4800, 7200, 9600, or 19200. Any other values will be flagged as an error. If the word ECHO appears as the second argument, then all characters transmitted will also be echoed to the receiver locally. If there is no second argument, the received data consists of only characters received from the communications line.

After the command line is typed in, the TERM program will be loaded, the arguments interpreted, the screen cleared, and the keyboard and display activated. All of the standard ASCII keys (those in the main key array except INSERT and DELETE) will transmit their standard ASCII code when pressed. Special MTU-130 keys, such as the cursors and function keys, will not transmit anything when pressed. The BRK key will send a line break which is interpreted by most time-sharing computers as an interrupt. All characters received are displayed as they are received when the display is on.

Several function keys are active when TERM is running and their legends are always displayed on the screen. Use of these keys allows "real time" control of the source of transmitted data and disposition of received data. The keys are as follows:

RDR FILE - TERM will ask for a file or device name to supply data as a simulated paper tape reader. The file must already exist. If an RDR file had been previously defined, it will be closed and the specified one opened. The dialog with TERM is not transmitted.

PCH FILE - TERM will ask for a file or device name to receive data as a simulated paper tape punch. If the file already exists, then any new data will be appended on the end of the specified file. If a PCH file had been previously defined, it will be closed and the specified one opened or created. The dialog with TERM is not transmitted.

CON FILE - TERM will ask for a file or device name to receive a record of all data received as a simulated printer. If the file already exists, then any new data will be appended on the end of the specified file. If a CON file had been previously defined, it will be closed and the specified one opened or created. The dialog with TERM is not transmitted.

RDR ON - When this legend is displayed, pressing the corresponding function key will cause transmitted data to be read from the previously defined RDR file. Reading will continue until end of file or receipt of an X-OFF (DC-3) from the communications line or the user presses the RDR OFF function key. If reading is suspended by receipt of an X-OFF, it will be restarted by receipt of an X-ON (DC-1) or by pressing RDR OFF then RDR ON function keys.

RDR OFF - When this legend is displayed, pressing the corresponding function key re-selects the keyboard as the source of transmitted data.

PCH ON - When this legend is displayed, pressing the corresponding function key will cause data received from the communications line to be written into the PCH file defined earlier.

PCH OFF - When this legend is displayed, pressing the corresponding function key will halt writing into the PCH file.

CON ON - When this legend is displayed, pressing the corresponding function key will cause data received from the communications line to be written into the CON file defined earlier.

CON OFF - When this legend is displayed, pressing the corresponding function key will halt writing into the CON file.

DISP ON - When this legend is displayed, pressing the corresponding function key will cause data received from the communications line to be displayed on the MTU-130 screen.

DISP OFF - When this legend is displayed, pressing the corresponding function key will halt the display of received data.

EXIT - Closes any active files and returns to CODOS

NOTES:

1. TERM uses teletype conventions for carriage return (CR) and line feed (LF) which means that transmitted data will start a new line with the sequence: CR-LF-NUL and received data is expected to use the same sequence for a new line. CODOS convention is used however in the RDR, PCH, and CON files meaning that an LF-NUL is transmitted after each CR read from the RDR file and all LFs immediately following CR's and all NULs are stripped when written into the punch or CON file. See the Patches section if this action is not suitable.

2. TERM uses interrupts and a large memory buffer to overcome display speed limitations at the higher baud rates. For average text with the display enabled, sustained reception at up to 2400 baud is possible and bursts of up to 32,000 characters can be handled at the higher rates. With the display disabled and either the PCH or CON files enabled, sustained reception up to 9600 baud is possible. There are no speed limitations when transmitting data.

3. Transmission format is 8 data bits, no parity, 1 stop bit for all rates except 110 where two stop bits are used and 135 where 7 data bits are used. The most significant bit of each byte transmitted is set to zero; see the patches section if this is not suitable.

4. Interface with the standard RS-232 modem control signals is minimal. In fact, successful operation is possible with only transmitted data, received data, and signal ground connected. If modem control signals are connected, TERM will set RTS and DTR true when it is executed and it will require CTS to be true (and noise-free). The status of the DSR and DCD signals is ignored but they must be noise-free and should not change during operation.

5. Any characters received with a framing error are replaced by "#".

6. Remember that received data is stored into a buffer before being displayed or written to a device or file. Therefore the display may fall behind the data actually being received if the baud rate is high (typically higher than 1200 baud) or the lines are short (less than 30 characters average).

7. Merely defining a RDR, PCH, or CON file will not turn the file on. The function key to turn the file on must be pressed after the file is defined.

8. When enabled, the contents of the RDR file are sent without forcing the most significant bit to a zero.

9. When the display is turned off, received data continues to be processed from the buffer. Therefore, if neither a PCH nor a CON file is active, received data will be lost. If there is a backup of received data in the buffer, it will be processed (or flushed) much more rapidly when the display is off.

10. TERM uses interrupts therefore the Addressing ROM change and wiring change for early model systems must be installed or TERM will crash when characters are received.

PATCHES:

1. A control byte is available for controlling the stripping of linefeeds and nulls from received data. Bit 7 at location $116B enables stripping of linefeeds when it is a one. Bit 6 enables stripping of nulls when it is a one.

2. A control byte is available for controlling the insertion of linefeeds and nulls into transmitted data. Bit 7 at location $116C enables the insertion of linefeeds after carriage returns if it is a one. Bit 6 enables the insertion of a single null after a CR-LF sequence if it is a one.

3. A 15 word table of 6551 Control register and Command register values corresponding to the 15 legal baud rates in ascending sequence is located at $118B. The left byte of each word is the Control register value and the right byte is the Command register value.

4. Location $09F2 may be changed to a JSR to a user routine for processing received data fetched from the buffer. The received character will be in A and X and Y should be preserved.

5. Location $090A may be changed to a JSR to a user routine for processing data immediately before it is transmitted. The character to be sent will be in A and X and Y should be preserved.

6. Page zero locations between $0032 and $00AF are not used. Additionally, locations between $125F and $13FF and between $A800 and $BDFF are not used.

UTILITY NAME: STRIP

PURPOSE: To strip extraneous linefeed and null characters from a text file.

SYNTAX: STRIP [<sourcefile> <destfile>]

DESCRIPTION: STRIP is useful for deleting extraneous linefeed and null characters from a text stream in order to make the resulting file compatible with MTU-130 editors and language processors. This may be necessary if the text originated in another system because CODOS uses the ASCII CR (carriage return) character as a "new line" character whereas text prepared according to "teletype" conventions will use the sequence "CR-LF" to start a new line. MTU-130 programs would interpret the "CR-LF" sequence as a new line followed by a null line, i.e., double spacing. Additionally, teletype oriented text may have one or more null characters ($00) following the CR-LF which are there to allow the teletype time to return its carriage. STRIP removes these also to save space and to prevent problems with editors and language processors.

If file names are specified, both must be specified. The source file name is first and the destination file name is second. One or more blanks must separate the names from each other and from the STRIP command name. Note that the destination file must not already exist on the disk. Standard drive notation (: drive) may be used on the file names. If file names are not specified, then the source text is read from channel 5 and written to channel 6. If this option is used, both channels 5 and 6 must have been previously assigned by the operator using the CODOS ASSIGN command. This feature allows STRIP to process text to or from devices as well as files.

EXAMPLES:

```
STRIP TELEXFILE.T GOODFILE.T
```

will read text from the file TELEXFILE.T, process it, and write the results on a new file called GOODFILE.T.

```
STRIP
```

will read text from channel 5, process it, and write the results on channel 6. Channel 5 must have been previously assigned to an input device or file and channel 6 assigned to an output device or file.

NOTES:

1. If the source is from a device, cntl/Z ($1A) is interpreted as end-fo-file.

2. If the destination is to a device, a cntl/Z is sent when end-of-file is encountered in the source.

3. You may append the STRIPped text onto the end of an existing file by assigning channel 6 to the file, issuing an "ENDOF 6" command, and executing STRIP without file name arguments (see the TYPE command description in the CODOS manual).

4. The stripping rules implemented by this program are as follows:
   A. Linefeeds immediately following carriage returns are deleted.
   B. Remaining linefeeds are converted into carriage returns.
   C. All nulls are deleted.
   D. All other characters including control and extended characters are passed unaltered.

ERRORS:

1. SOURCE FILE NOT FOUND - The file name specified for the source file does not exist on the specified disk drive.

2. DESTINATION FILE NOT SPECIFIED - Only the source file name was specified. If file names are used, both must be specified.

3. DESTINATION FILE ALREADY EXISTS - The file name specified for the destination file already exists on the specified disk drive. The destination file must be a new file when file name arguments are used.

4. CHANNEL 5 NOT ASSIGNED - Channel 5 had not been assigned to the source file or device prior to using STRIP.

5. CHANNEL 6 NOT ASSIGNED - Channel 6 had not been assigned to the destination file or device prior to using STRIP.

UTILITY NAME: CPUID

PURPOSE: To print out the hardware encoded CPU board identification numbers.

SYNTAX: CPUID

DESCRIPTION: CPUID is used to print out the identification numbers encoded into the hardware of your particular CPU board. These numbers are important because some very large software packages available for the MTU-130 will be keyed to run only on your particular computer. By having Group and Vendor numbers in addition to the unique User number it is possible to "master key" software as well so that it can be freely exchanged among a controlled group of MTU-130 users. This anti-piracy technique allows you to make as many copies of the protected program as you like and allows the program itself to use the standard CODOS operating system for all of its operations.

EXAMPLES:

```
CPUID
```

will read the hardware encoded identification numbers on the MTU-130 CPU board and print them on the console in the following format:

```
VENDOR NUMBER - VVVVV

GROUP NUMBER - GGGGG

USER NUMBER - UUUUU
```

where VVVVV is the 5 digit Vendor number, GGGGG is the 5 digit Group number, and UUUUU is the 5 digit user number of your particular CPU. These are hexadecimal numbers so the letters A-F may also appear in them. Other MTU-130 owners may have the same Vendor and Group numbers as you do but every system has a different User number.

ADDENDUM

MTU is constantly striving to improve and supplement the software supplied with the MTU-130 computer. For this reason there are several new files on the Distribution disk which are not described in the manual, but are described below:

UTILITY NAME: MEMORYTEST

PURPOSE: To test all 80K of standard MTU-130 memory.
SYNTAX: MEMORYTEST

DESCRIPTION: This program tests all memory in the standard MTU-130. When started, it will display a sign on message and then begin testing memory. No activity will be visible for several minutes, until the display memory is tested. At this point the display will be filled with a random pattern of dots which changes periodically. At the completion of one pass of the memory test, the display will be restored and a message issued indicating that pass 1 was completed. The test will then repeat indefinitely, displaying a message after each pass. If errors are detected, the program will display the error locations at the end of the pass and abort. A maximum of 20 errors will be displayed. Since this program tests all of memory including that part used by the operating system, it can only be terminated by re-booting the system (MOD-RESET). All MTU-130 computers have thorough memory tests and burn-ins before leaving the factory. Although modern memory chips are exceedingly reliable, you should probably run the memory test periodically, especially if you have noticed any strange system behavior. You may wish to leave it running overnight to thoroughly test the system.

UTILITY NAME: JOINFILES

PURPOSE: To concatenate two or more disk files.
SYNTAX: JOINFILES filename

DESCRIPTION: This program can be used to combine any number of CODOS files in a single file. It is particularly useful for combining several machine language object files. Any types of files may be joined. This program begins by prompting for the name of a file to be concatentated, if it was not specified on the command line. This file will be the new composite file. If it already exists, any concatentated files will be appended to its end. If it does not exist, a new file will be created. The program then asks for the name of the file to be appended. The program will make a copy of this file at the end of the first file. The prompt for another file is then repeated indefinitely, as long as you continue to enter file names. To terminate the program, just enter a carriage return instead of a file name.

FILE NAME: MTU130INDEX.T

PURPOSE: To provide a modifiable, on-line copy of the index to the MTU-130 System Manual.

DESCRIPTION: This is not a program but a text file which you may examine or modify with the Editor or other programs. It contains a copy of the master index for the MTU-130 System Manual. It is included on disk so that, if you wish, you may add your own index entries or modify it as you see fit. It also can serve as the data base for a user-written on-line "HELP" program which could display a manual page reference based on a keyword or phrase entered from the console. This type of program would make an interesting project to build familiarity with the powerful commands of the BASIC CIL library, which allows easy access to data files such as this.