Micro Technology Unlimited

MTU-130 NEWS

Volume I Issue 1 May, 1982

FROM THE PRESIDENT

As valued customers, I welcome you to the elite club of MTU-130 owners! Many of you have been buying from MTU for years, feeding our growth and our dreams. As you know, our past products have been above the industry average. Economic design is as important as the system performance features and we feel our expertise is apparent in both categories. Our tradition has been and will continue to be that of providing you with the very best within viable economic restraints.

Three years ago, in early 1979, we began the conceptualization and design of the MTU-130. Our Visible Memory concept of presenting information to the user, and our super fast and flexible disk controller system were the beginning. Now our direction is unveiled as these two concepts are seen as major performance features of the MTU-130.

It is a major departure for a perpherial manufacturer to design the complete system, but we felt it was time to give the market a computer which would not be obsolete next year. Our goal was to develop a fast, universal human interface system which is totally software programmable. Based on our 11 years in computer architectural development, and by planning and thinking out the design in advance, we have removed many problems that have plagued other systems in the past. You now are discovering what we have been working toward and dreaming about for so long.

During a demonstration last week we were asked to prove our flexibility by entering text from right to left. Enter the following and find out how we scored:

SET CC46 F0 CE and SET CD7F .80

This is not the proper way to operate in a production environment, but it does point out that our programmability equates to flexibility in solving real world problems, which is how computers are justified (in addition to being fun)!

Our priorities now are the generation of expansion peripherals and application software. Peripherals are our history and we will continue our good heritage in this area. For the needed applications, we are working hard internally and looking outside. Since each of you already owns an MTU-130 system, and understand its capabilities, you are prime candidates to help. Any worthwhile programs, ideas or specifications you have are wanted, please write to me to discuss what we can do with them. We also need your comments as to what you would like to see us provide.

Again I welcome and sincerely thank you for participating in our dream.

DAVID B. COX

NEW PRODUCTS

THE DATAMOVER-256

Perhaps you have seen in MTU advertisements and sales literature references to the DATAMOVER-256 and are wondering what it is. Essentially, the DATAMOVER-256 combines 256K bytes of RAM with a Motorola 68000 CPU on one small board that fits neatly in one of the MTU-130's empty bus slots. The DATAMOVER-256 is simply the most powerful add-on CPU board available ANYWHERE for ANY personal computer. The DATAMOVER-256's $999 price makes an MTU-130 plus DATAMOVER-256 the least expensive, complete 68000 system available. The DATAMOVER-256 is available now.

Memory Function

The DATAMOVER-256 looks like a 256K byte memory board to the MTU-130. The 256K is accessible as two GROUPS of 128K each. At any one time, one group is accessible in Banks 2 and 3 of the MTU-130's address space. As far as the MTU-130's 6502 processor is concerned, it is plain, unadulterated, zero wait state memory. An expansion socket on-board the DATAMOVER-256 allows even MORE memory to be added on a "piggyback" board. The additional memory is accessible as up to 6 more Groups of 128K each for a potential total of 1 MILLION bytes! An "Enable Register" on the DATAMOVER-256 allows the 6502 program to select which 128K group is to be accessible to the 6502. The 68000, with its 24 bit address bus, always has direct access to all of the DATAMOVER-256's memory. While the memory appears byte-oriented to the 6502, it is 16 bit word-oriented in the eyes of the 68000.

68000 Processor Function

Of course the most exciting DATAMOVER-256 feature is the super powerful 16 bit MC68000 microprocessor. The DATAMOVER-256 operates the 68000 at 8MHz and virtually zero wait states which means that it runs at 98.4% of maximum potential speed. If you havn't looked at the 68000 instruction set or read about its superior performance in benchmark testing, you've got a treat in store. It is particularly effective in math programming with its 9uS 16x16 multiply and and 32/16 divide instructions. It actually performs software floating point faster than "math processors" such as the AM9511 and AM9512.

Interprocessor Communication

Two methods are provided for transferring information between the 6502 and the 68000. Obviously they can exchange vast amounts of data through the shared memory. Since shared data is simultaneously in the memory address space of both processors, there is no need to physically move the data between them as in most other 68000 add-on schemes. Also, the 6502 can interrupt the 68000 on two levels and the 68000 can in turn interrupt the 6502.

For more information, contact MTU.

FORTH LANGUAGE

An outstanding implementation of the FORTH language has been developed by Saturn Software for the MTU-130. It is a full implementation of the FORTH-79 standard language, with many enhancements including graphics and CODOS interfaces. If you've heard about FORTH and want to learn it, this is the way to go! MTU-FORTH is available from MTU for $149.

MTU-130 NEWS

Volume I Issue 1 May, 1982

MTU-130 NEWS is a quarterly publication for MTU-130 users. Please address all correspondence to:

Newsletter Editor
Micro Technology Unlimited
P.O. Box 12106
2806 Hillsborough Street
Raleigh, N.C. 27605 U.S.A.

INPUTS

USER INPUT INFORMATION

We welcome input from all of our users whether that input be programs, hints, music, pictures or any other item which you think may be useful and/or interesting to other MTU-130 users.

We would prefer that your input be provided on single sided diskettes--see below for information on their return. Programs should be accompanied by a separate text file containing a one paragraph abstract of the program suitable for publishing in the next newsletter as well as specific operating instructions for your program. Articles or other text should be written using the MTU Editor with 43 columns maximum width. We use two spaces before the first word in the paragraph and a line between paragraphs.

All programs submitted to MTU will be considered in the public domain and the user submitting the program is requested to include a statement regarding the original author if other than the submitter as well as any known previous publication history.

For those submitting material on diskette we will make a copy of that material and a copy of the current user group diskette for return to you in the same packing in which we received it.

HOW TO GET YOUR USER DISKETTE

All of the programs listed below can be running on your system for only $15! You will be sent a copy of the current user diskette on receipt of your $15. In order to keep costs to a minimum we will only be able to fill prepaid orders or orders which charge the purchase to Visa or MasterCard.

USER DISKETTE CONTENTS

ANNOTATE.C - submitted by Hal Chamberlin
A BASIC program for adding headlines, captions, etc. to a screen image file. It prints letters or other shapes on the screen with independently selectable horizontal magnification, vertical magnification, horizontal thickness, and vertical thickness. The user may create the character shape database by using the VBFONTENTER program.

ANNOTATE.T - submitted by Hal Chamberlin
Description and operating instructions for ANNOTATE.

APPLEREAD.C - submitted by Wayne Monteath
Reads APPLE format cassette tapes input through the MTU-130 cassette input port.

APPLEWRITE.A - submitted by Wayne Monteath
Describes APPLEWRITE.C.

APPLEWRITE.C - submitted by Wayne Monteath
Writes APPLE format cassette tapes connected to the MTU-130 cassette output port.

ARCH.B - submitted by Dr. J.W. Froelich
Plots Archimedes spiral using user-defined parameters.

B51.B - submitted by Wayne Monteath
A bomber game.

BIRDSHOT.B - submitted by Wayne Monteath
A game with a bird which flaps its way across the top of the screen and a gun which the user may elevate and shoot. Score is kept on the screen.

BIRDSET.Z - submitted by Wayne Monteath
The special font used in the birdshot game.

BREAKER.C - submitted by Dr. J.W. Froelich
A rewritten version of breakforth from BYTE August 1980. The program was written in MTU-FORTH (available from Saturn Software). The program does not require the user to have FORTH to run since it was sent to us in the turnkey mode which copies only that part of FORTH needed to run this specific program.

CBMMTUMOD.E - submitted by Dr. J.W. Froelich
Converts tokenized form of PET BASIC programs to MTU BASIC. Some I/O must be converted by hand.

CONVERT.B - submitted by Dr. J.W. Froelich
Converts decimal to hex, octal or binary. Also hex and octal to decimal.

DISK_SPACE.A - submitted by Bruce Carbrey
Source listing of subroutine to determine the amount of free space on the disk while running a program.

DISK_SPACE.L - submitted by Bruce Carbrey
A subroutine which can be used in another program to determine the amount of unused disk space on a designated drive of an MTU-130 with CODOS 2.0.

FLIST.B - submitted by Dr. J.W. Froelich
Formatted listing program.

FLIST.L - submitted by Dr. J.W. Froelich
A listing of FLIST.B.

FLISTDOC.L - submitted by Dr. J.W. Froelich
Describes FLIST.B which is a formatted print program which lists ASCII files from disk to an output device. Program adds page number, date and file name to the top of each printed page with option page widths, number of lines per page, variable spacing, and stopping at the end of each page.

IMAGE1.G - submitted by Dr. J.W. Froelich
A screen image of a 3-D figure.

MDUMP.C - submitted by Keith Sproul
A screen editor to modify memory directly. Allows the user to specify an area of memory to be examined on the screen and altered. The cursor keys allow easy selection of the locations to be altered. Turns numeric pad into a HEX pad to allow very easy entry of hexadecimal data. Allows selection of bank and 128 Kbyte of DATAMOVER-256 board to be changed.

MDUMP.T - submitted by Keith Sproul
Description and operating manual for the MDUMP program.

POOL.C - submitted by Bruce Carbrey
Animated pocket billiards game. Moves all 16 balls on the screen in real time! Ball motion is accurately computed by actually solving the equations of motion including rolling friction and inelastic momentum transfer. Machine language.

POOL.T - submitted by Bruce Carbrey
Description of the Pool program.

PIRATE_ADV.B - submitted by Wayne Monteath
Pirate adventure game.

RANDOM.B - submitted by Dr. J.W. Froelich
Demonstrates the speed of graphics under basic by drawing random figure.

READ_APPLE.A - submitted by Wayne Monteath

READ_APPLE.C - submitted by Wayne Monteath

READAPPLEDOC.T - submitted by Wayne Monteath

REALESTATE.B - submitted by Dr. J.W. Froelich
Computes future balance, future value, present balance, present value, payment amount, interest rate, number of payments, purchase price at a given yield, yield at a given purchase price, balloon payment, amortization schedule, and yield of uneven cash flows.

RENUMBER.E - submitted by Wayne Monteath
A BASIC program to renumber your BASIC programs.

RENUMBERDOC.T - submitted by Wayne Monteath
Documentation for renumber program.

SLIDE_SHOW.C - submitted by Wayne Monteath
A program to read an APPLE format cassette tape containing hi-res picture images and convert the image so it may be displayed on the MTU-130 video monitor.

SLIDE_SHOW.A - submitted by Wayne Monteath
Source listing for above.

SLD_SHW_HDR.D - submitted by Wayne Monteath
Special title needed by SLIDE_SHOW.C.

SLIDESHOWDOC.T -submitted by Wayne Monteath
Documentation for above.

SORT_STRING.R - submitted by W.S. Smith, Jr
A quicksort program in BASIC using the CIL library. Entry requires a file with fixed length records assigned to a channel, the record length, the offset of the sorted element from the record start, the channel number for storing sorted pointers and a flag to determine if sorting by last name is desired.

STARTREK.B - submitted by Dr. J.W. Froelich
Startrek game from BYTE March 1977.

STARTREK1.B - submitted by Wayne Monteath
Another version of Startrek.

SURVIVAL.B - submitted by Wayne Monteath
An adventure game which requires the player to use his wits to escape from the moon following a crash of his space craft.

TAPE_ANLYSR.A - submitted by Wayne Monteath
Source file for tape analyzer program below.

TAPE_ANLYSR.C - submitted by Wayne Monteath
Program described below.

TAPE_ANLYSRDOC.T - by Wayne Monteath
Description of program to analyse input from the MTU-130 cassette input port. Draws horizontal lines which are inversely proportional to the frequency of the input.

VBFONTENTER.B - submitted by Hal Chamberlin
A BASIC program for creating shape or character font files usable by ANNOTATE or user written programs. The light pen is used with an expanded grid to specify the shapes. Shapes are stored in vector form rather than dot-matrix form and may be easily scaled and rotated.

VBFONTENTER.T - submitted by Hal Chamberlin
Description of program to enter special font characters.

NOTES FROM THE FACTORY.

LIGHT PEN BOO-BOO

It seems that Murphy paid MTU a visit when the master Distribution 1.2 and 1.3 disks were being put together and made us leave out the last 6 bytes of the light pen subroutine! The effect of the omission is an error of as much as 8 pixels in the light pen X coordinate in certain areas of the screen. This really shows up only when you attempt to draw with the pen in which case lines may become unusually jagged or points may refuse to be set in some areas. The pen may even seem to work better on some days than on others!

The error is in the GRAPHDRIVER.Z file which is likely to be on every disk you own which has an operating system. To correct this error (and it is strongly recommended that you do correct it), use the following procedure:

1. Insert the disk to be corrected in drive 0 and OPEN it.

2. Enter: GET GRAPHDRIVER.Z

3. Enter: SET C5A2 4 8 10 20 40 80

4. Enter:

RE!GRAPHDRIVER.Z=327 324 356 C000 C5A7

5. CLOSE drive 0

Repeat steps 1-5 for each disk to be corrected. The light pen should now perform much better.

SOME CORRECTIONS TO THE BASIC MANUAL

Here are a couple of corrections to the MTU BASIC Reference Manual. First, the Standard Keyword List in Appendix A is in error. The token values given for the right column are all 1 higher than they should be. The "+" should have a token value of 173 (AD hex), and "GO" should have a value of 206 (CE hex).

Also, the discussion for the standard INPUT statement fails to describe how the colon (:) is treated when encountered as part of the input. If the colon is part of a string, and the string is enclosed within quotes, then the colon is treated

as a normal character. In all other cases, it is treated as if it was a carriage return instead. For example, if the statement:

```
INPUT NM$,D
```

is executed, then the following responses will give the described results.

```
MYFILE.T:1,10
```

will cause NM$ to become "MYFILE.T" and the INPUT statement will ask for more data.

```
"MYFILE.T:1",10
```

will cause NM$ to become "MYFILE.T:1" and D will be set to 10. If a comma is part of a string that is enclosed within quotes, then the comma will also be treated as a normal character, not a terminator.

A BUG IN TERM

A bug has been found in the TERM terminal emulator program that is supplied on the distribution disk included with all MTU-130s. It will show up ONLY if you are using the Data Set Ready (DSR) modem control signal on the MTU-130 serial port. If DSR is driven low by an external device, TERM will display all received data as a string of #####, i.e., it will think there was a framing error on every received character. A simple patch is all that is required to correct this problem. To patch TERM, enter the following commands (disk with TERM assumed to be in drive 0):

```
GET TERM
SET 8D7=58
SET A69=58
RESAVE TERM 700 125E
```

This should be done to every copy of TERM you have made. The bug is due to testing the wrong bit in the 6551 serial I/O chip for a received character framing error. The two patches will change the program so the correct bit is tested.

UPGRADING THE MTU-130 DEMO DISK

Each MTU-130 computer is shipped with a Distribution diskette and a Demonstration diskette. Because some last minute improvements were made to the operating system prior to release, some MTU-130s were shipped with a slightly different version of the CODOS operating system on the Demo disk than on the Distribution disk. The differences between the systems are minor and will normally not cause any difficulties.

If you have followed the directions in the MTU-130 manual and prepared your "working disks" from the Distribution Disk (or copies of the Distribution Disk), your working disks will have the correct version of the operating system. If, however, you generated your working disk from the Demo Disk, then your working disks will have the older operating system. You can test to see which system is present on any disk by using the CODOS command:

```
GETLOC CODOS.Z
```

The results indicate which system you have:

```
CODOS.Z:0=E600 E54A FDE5  OLD
 operating system

CODOS.Z:0=E600 E54A FDEE   New correct
 operating system
```

If you discover any disks with the old operating system, you should upgrade them to the correct operating system. To do this, you will need one new or scratch disk, the disk to be upgraded, and the Distribution disk or a disk with the correct operating system on it. Boot up the Distribution disk (or a working disk with the correct operating system). Put the new scratch disk in drive one and FORMAT the disk, replying "YES" to the "WANT TO COPY DRIVE 0 SYSTEM?" prompt. You now have a new disk with the correct system. Boot it up in drive 0, put the disk with the wrong operating system in drive 1 and OPEN it. Now type:

```
COPYF *.*:1
```

which will copy all remaining files from the old disk to the new disk. The disk in drive 0 is now updated with the correct operating system and all files from the old disk. Repeat the procedure for any other disks (you can use the old disk as your new scratch disk now if you want).

If you don't upgrade the old operating system, you will probably not have any problems. However, the old operating system differs from the correct operating system in the following respects:

1. SVC #21 does not return the A register correctly as described in the manual. This may affect the Editor or other programs assigning files.

2. A DATE command in a job file will terminate the job file instead of continuing with the next command.

3. The operating system will not respond correctly to an end-of-file signal from user-defined input devices.

4. The FORMAT command will produce a slightly different skew and interleave on the disk.

5. When using SVC #3 for single character input from the keyboard, the echo flag at $E788 will have the opposite effect.

UPGRADED EDITOR NOW AVAILABLE

MTU-130 Computers are now being shipped with Distribution Disk version 1.3. The primary change in this new version compared with version 1.1 shipped with earlier MTU-130s is the inclusion of EDIT 1.1, an upgraded screen editor. All known bugs in the early version (which were minor) have been eliminated. In addition, several new features have been incorporated, including: (1) ability to limit search-and-replace operations to specified columns as well as lines; (2) improved tabstops; (3) automatic return to text mode after command completion; (4) ability to terminate searches with the BREAK key; (5) ability to embed codes for printers with extended character sets.

Additional screen dump utilities (SPRINT and VMDUMP) have also been added for the NEC-8023 and IDS Prism printers. All EPSON printers are also supported.

MTU-130 owners wishing to upgrade may purchase the new Distribution Disk 1.3 and manual addendum for a media and handling charge of $15, prepaid.

SOME NOTES ON PRINTER SOFTWARE

MTU has committed to providing printer software for each printer that passes through our hands. So far, this includes the MX70, MX80, MX100, NEC-8023A, IDS-440, IDS-PRISM, and ANACOM-150. The support consists of making sure that the SYSGENPRINTR utility program can generate an appropriate PRINTDRIVER.Z file to allow for ASCII output to the printer. In addition, VMDUMP and SPRINT programs are provided to perform dot graphics screen prints. The names of the files on disk will have appropriate letters appended to "VMDUMP" and "SPRINT" to indicate which printer they are used with.

The VMDUMP program is a utility, to be executed as a CODOS command. You may leave the VMDUMP utility named as is, or rename it to just "VMDUMP.C". The SPRINT program is used by the IGL Library's SPRINT command to perform the actual screen print. To use it, you must rename the appropriate version of the SPRINT program to "SPRINT.Z".

Unfortunately, there has been some inconsistency in the printer software that was shipped on the Distribution Disks. Originally (i.e. Fall 1981) the printer software was written to assume that the printer would do an automatic line-feed in response to a carriage return. Then in January, it was decided that the software should send the line-feed, and not have the printer do it automatically. This has the advantage of allowing a program to just send a carriage return so it can reprint the line, perhaps for underlining. Unfortunately, the Distribution Disk 1.2 did not get the updated VMDUMP and SPRINT software. Actually, this only affected VMDUMPMX80.C and SPRINTMX80.C files,

which still expected the printer to do its own line-feed. The new Distribution Disk 1.3 has these files updated, plus the new printer software for the NEC-8023A and IDS-PRISM printers. For those who received the Distribution Disk 1.2, you may obtain all of the latest printer software by purchasing Distribution Disk 1.3.

OUTPUTS

HIDDEN GOLD IN BACKUP

by Hal Chamberlin, MTU

One of the most useful MTU-130 utilities is the BACKUP program which quickly makes an identical copy of any CODOS diskette. Due to a documentation mixup however it can actually do MORE than advertised!

The manual (page U8-1) states that the copy disk created by BACKUP always uses the standard sector and track skew. In reality, BACKUP will accept skew arguments just like FORMAT does and the copy disk will be formatted as specified. Thus BACKUP can also be used to convert a CODOS disk to a different track and sector skew. Remember, BACKUP does not care what the skews of the source disk are, but it will run more slowly if the source disk was formatted with a sector skew other than 2.

Y DEVICE OUTPUTS TO CONSOLE AND PRINTER

by Bruce D. Carbrey, MTU

The CODOS operating system allows great flexibility in redirecting input and output to various devices. I-O channels can be assigned to any device or file, and multiple channels can be assigned to the same device or file at the same time. Sometimes however, you may want to output over a single channel and have the output appear on two devices at the same time. In particular, it is often desirable to output to the console and the printer at the same time. CODOS does not directly support assignment of a single channel to multiple devices, but it can easily be accomplished by defining a new device which outputs to both devices. Here is a simple way to define a new output device which prints on the Console and the printer at the same time. We will call this new device the "Y" device. A plumber knows that a "Y"-shaped pipe will split the flow, which is why we call it "Y"!

To add the Y device, use the EDITor to add this line to your STARTUP.J file:

```
SET D27D=20 21 E6; Y DEVICE DRIVER.
```

Now run the SYSGENDEVICE program and add the Y device, specifying the output driver address as D27D. That's all there is to it! After re-booting the system, you can assign channels to Y at will. To output to the Y device from BASIC, type:

```
ASSIGN 6 Y
```

before entering BASIC. Once in BASIC, type:

```
OUTCHAN 6
```

All subsequent PRINT statements will print on the console and the printer, until an error or an OUTCHAN 2 command restores output to the C device.

How does the Y device work? The standard printer driver entry point is at $D280. The patch to the STARTUP.J file inserts a JSR to the console output routine (at $E621). Therefore the Y device driver simply calls the console output driver and then falls through into the printer output driver, with the character to be output still in the A register. The result is output to both devices.

TAMING THE MTU-130'S SERIAL PORT

by Hal Chamberlin, MTU

The Synertek 6551 Asynchronous Communication Interface Adapter IC used in the MTU-130 is a marvel of modern integrated circuit technology. In a single 28 pin package that costs less than a good dinner is a full serial transmitter function, receiver function, baud rate

generator, and control logic. The control logic is even smart enough to perform a number of functions automatically without program intervention. Unfortunately, some of these automatic functions can get in the way when trying to utilize the serial port in your own programming. Whether you call them features, side effects, gotchas, or bugs, they are etched in silicon and have to be understood to properly utilize the serial port.

The Clear-to-Send Signal

Perhaps most annoying when it causes trouble is the clear-to-send (CTS) input signal. Although called a "modem control signal", it is often used by serial printers to indicate when they are able to accept data. Its intended interface function is to inhibit the transmitter when it is EIA-low (-12 volts) and enable the transmitter when it is EIA-high (+12 volts). Unfortunately its action is unnecessarily abrupt. When the 6551 sees that CTS is low, it INSTANTLY stops the transmitter, even in mid-character, and forces it to send a marking (low) level. When CTS goes back high, the transmitter is allowed to send the next character, if any. Thus if CTS is arbitrarily applied to stop transmission, the character being transmitted at the time will be garbled. Even a brief noise pulse on CTS will be recognized and cause garbled characters to be sent.

Fortunately this problem is usually easy to overcome in the typical serial printer or other device interface. Perhaps the easiest solution is to use one of the other input signals (Data Set Ready, DSR; or Data Carrier Detect, DCD) for the printer's handshake signal since they do not affect the transmitter. They are not, however, totally problem free either (see below). If CTS must be used, the secret is to give the printer or other device sufficient time after sending it a character to respond on the CTS line before sending it another character. If this is done, CTS will not be changing states while a character is being transmitted. You can test this out by programming a variable delay after each character is sent. Then starting at about 12/(baud rate) seconds, increase the delay until the data is transmitted accurately. If the necessary delay is such that performance deteriorates, try a higher baud rate. Most matrix printers will lower CTS only at the end of a text line, i.e., only after receiving a carriage return, and thus the delay may only be necessary after sending the carriage return character.

As mentioned earlier, noise pickup on the CTS line can also interfere with the transmitter. Such noise is likely to be a problem only when CTS is not connected to anything and therefore is allowed to "float high". The noise can be particularly severe when CTS connects to the user's cable at the MTU-130 end but not at the device end. In this case it can pick up all kinds of noise through crosstalk in the cable, usually from the transmitter itself. Therefore the first rule is: if CTS is not used, it should not be connected to anything, even at the 130 end of the interface cable. Occasionally there may be enough crosstalk in the INTERNAL cable that connects the serial port jack to the CPU board to cause problems. The simplest fix is to connect a .1uF disk capacitor between CTS (pin 5) and ground (pin 7) on the 130 end of your serial cable. You may also want to filter DSR and DCD (pins 6 and 8) as well if you have been getting spurious IRQ interrupts while using the serial port (see below). A permanent fix on-board the Monomeg CPU would be to connect these 3 signals to 10K pullup resistors connected to +12 volts. It would then take quite a bit of noise indeed to cause problems.

Data-Set-Ready and Data-Carrier-Detect

These are two additional input signals are normally used to indicate the status of a modem. Unlike clear-to-send, these two signals have no direct effect on the transmitter. Data-carrier-detect (DCD) must however be high (+12 volts) or floating for the receiver to operate. Both of these signals are connected to status bits in the status register so the interface program can directly determine their state. Additionally, when either of these signals CHANGE from high-to-low or low-to-high, an interrupt may be generated. The catch is that the only way to DISABLE these interrupts is to disable the whole receiver; there is no modem control interrupt enable bit in the command

register. Since its tough to receive data with the receiver disabled, your serial interface program may need to be able to handle these interrupts.

If your interface program does polled I/O (i.e., sits in a loop waiting for something to happen) and doesn't use interrupts for anything else (CODOS doesn't use interrupts at all), the simplest thing to do is to set the 6502 IRQ disable at the beginning of your program with an SEI instruction. Before your program exits though, it should reset the serial interface by WRITING to the serial status register. If your interface program does use interrupts, it should be able to recognize interrupts from these two signals and take the appropriate action or just ignore them altogether.

Like clear-to-send, the DSR and DCD signals are sensitive to even very narrow noise spikes and therefore can be a source of spurious interrupts even when not used. Unless actually being used, avoid connecting them to your interface cable. If necessary, install filter capacitors on the cable connector or pullup resistors on the CPU board. Lastly, keep the receiver disabled when not being used to prevent interrupts when the serial cable is plugged or unplugged. If a serial interface interrupt does occur and interferes with other programs, simply press the RESET key or enter: SET BFC9 0.

Using Serial Port Interrupts

This last serial interface difficulty is completely obvious and well documented (Monomeg manual, page 39) but is so unusual that it is apt to be overlooked. Interrupts, ALL INTERRUPTS, from the serial interface are cleared merely by READING the serial status register! There are 4 possible sources of interrupt: receiver data register full, transmitter data register empty, DSR changed, and DCD changed. If two or more of them occur simultaneously or nearly so, a single read of the status register clears them all. Therefore your interrupt service routine must read this register ONCE ONLY and then service each condition that could have caused the interrupt before returning from interrupt. Furthermore, if you plan to mix polled serial I/O with interrupt I/O, the polling loop must be able to recognize conditions that would have caused an interrupt (reading the status register for the poll has already cleared it) and then handle them. Failure to account for this "feature" can cause lost characters if the receiver is run on interrupts or lockups if the transmitter is run on interrupts.

Conclusion

This covers all of the known difficulties in using the 6551 serial I/O chip. If any reader has experienced a problem not covered, we would like to hear about it, particularly if a neat solution has been found. In the meantime everyone should hold their breath until Synertek comes out with a 6551"A" with on-chip fixes for at least some of these--features.

SOME COMMON QUESTIONS ABOUT BASIC

by Larry Isaacs, MTU

There have been a few questions that many MTU-130 owners have asked. Since they are questions that most everybody will ask sooner or later, I will try to provide some answers for these questions.

QUESTION 1.

Is there some way I can test the keyboard and return immediately, even if no key is pressed?

This question arises from the fact that the GET command always waits for a key to be entered before allowing the BASIC program to continue. Sometimes it is useful to be able to test the keyboard and not have it wait for a key. This would allow the BASIC program to do other things while periodically testing the keyboard. Fortunately there is a simple way to accomplish this.

The way to do this is to reprogram the KEY variable. The operation of the KEY variable is not tied directly to the function keys. Instead, it is tied to the values in a couple of memory locations. The first location, called FFKEY, is found at 2054 (806 hex). FFKEY contains the ASCII

value of the first key recognized by the KEY variable. If this key is pressed, the KEY variable will return a value of 1. The second location, called LFKEY, is found at 2055 (807 hex). LFKEY contains the last key recognized by the KEY variable. If this key is pressed, the KEY variable will return a value equal to the number of keys contained in the range from FFKEY to LFKEY.

The following BASIC subroutine shows how to compute the value which can be returned by the KEY variable in response to a pressed key. In this subroutine, KEYVAL contains the value of the pressed key. KEYVAL is 0 if no key is pressed on the keyboard.

```
IF KEYVAL=0 THEN KEY=0:RETURN
IF KEYVAL<FFKEY THEN KEY=0:RETURN
IF KEYVAL>LFKEY THEN KEY=0:RETURN
KEY=KEYVAL-FFKEY+1:RETURN
```

In the last statement of the subroutine, one is added to KEY so the sequence will start with 1 instead of 0. Normally, FFKEY is set to 128 (f1 key), and LFKEY is set to 135 (f8 key). This causes the KEY variable to return a value from 1 to 8 in response to the function keys. By setting FFKEY to 1 and LFKEY to 255, then the KEY variable will return values for the entire keyboard. Also, the value returned will match the ASCII value of the key. There is one exception, however. In order to guarantee proper operation of Control-S, BASIC is made to ignore entry of this key value.

QUESTION 2.

How can I detect if a key is down, even if it has already been detected and is still down?

In order to achieve proper keyboard operation, the keyboard scan routines will wait for a previous key to be released before they will report that any new key has been pressed. This means that pressing a key will only enter the key value once. If you continue to hold that key down, the keyboard scan routines will continue to report that no key is down, or more accurately, no new key is down.

Sometimes it is desirable to be able to determine if a key is down, even if that key has already been detected. To do this, you simply clear the memory location that indicates which key, if any, was down last time the keyboard was scanned. This location, called LSTKEY, is found at 525 (20D hex). If you execute the statement:

```
POKE 525,0
```

then the next time the keyboard is scanned, a key that was previously down can be detected again.

Also, it may be important to note that the value contained in LSTKEY is not the ASCII value for the key. Instead this value is related to the key's position on the keyboard.

QUESTION 3.

Sometimes when I am printing to the screen, I get a carriage return where it is not supposed to occur. Where does this carriage return coming from, and how can I eliminate it?

BASIC keeps a count of characters which are output through BASIC's output channel. This count is the value returned by the POS function. If this count reaches the maximum line length, normally set to 192, then BASIC will output a carriage return. This extraneous carriage return usually occurs when executing a sequence of INPUT statements with fairly long prompt strings. The carriage you enter at the end of your data does not get passed to BASIC's output routines, so the character count does not get reset. Therefore the character count from the input prompts accumulate until the count exceeds the maximum. The result is an extraneous carriage return in the middle of one of the prompts.

To prevent the carriage return from occurring, you must either output a character which resets the count, or reset the count with a POKE statement. There are three characters which will reset the count to zero when output. They are carriage return (CHR$(13), form-feed (CHR$(12), and cancel (CHR$(24), Control-X) which clears the current line. If you wish to POKE the count directly, you can find it at location

21 (15 hex). Also, the TCURSOR command in the IGL Library will set the count to the column value you specify in the command.

QUESTION 4.

CODOS has a RESAVE command, why doesn't BASIC have a similar command?

Actually, there is a provision already in the BASIC interpreter to allow for overwriting an existing file. A RESAVE command was not implemented in BASIC because there is only one unused command token left. It was decided it would be better to put a RESAVE command into a library so that the remaining command token could be used for something more important that might turn up later.

Fortunately all is not lost. What is provided is a memory location that indicates if it is alright to overwrite an existing file. If set to 128, then overwriting is allowed. If set to zero, then SAVEing or LISTing to an existing file results in an error. Yes, the overwriting applies to the LIST command also. This memory location is found at 2066 (812 hex). So, if you want to SAVE or LIST a program to an existing file, just execute:

```
POKE 2066,128
```

first. Once you have output the program, it would be safest to set the location back to its normal state by:

```
POKE 2066,0
```

KEYBOARD ECHO

Echo
echo
....

by Bruce D. Carbrey, MTU

If you've tired to use the keyboard echo flag described in the CODOS manual, you may have run into some problems. The keyboard echo flag is used to determine whether or not pressing a key causes the character to display on the CRT immediately. The problem is, we built in so much flexibility in the system that there are really three keyboard echo flags, each of which can be altered independently of the others! Here's a summary of all three that should help:

1. HFDPLX, address $E788, normally 0. Used only for single-key input from console (by SVC #3 or by direct subroutine call of $E61D). Normally does not echo. Set to $80 to echo (you will need to UNPROTECT first).

2. KBECHO, address $020F, normally 0. This is the echo flag at the lowest level keyboard driver, GETKEY (entry point $0306), as described on page 8-2 of the CODOS manual. Since practically all system routines call this routine, setting it to $80 will usually cause an extra echo (doubled characters).

3. NOLEKO, address $0239, normally 0. This echo flag is used by the input-line and edit-line routines (entry points at $031E and $0321 as described on page 8-3), plus any system line input including SVC #5. It is toggled on and off by CNTRL-E. Set this flag to $80 to disable keyboard echo.

ONE BYTE CHANGE CREATES NEW UTILITY

by Bruce D. Carbrey, MTU

Have you ever wanted to use the COPYF Utility program to copy all files except one particular file or files with one particular file extension? Here's a quick and easy way to create a new Utility program which will copy all files that DO NOT match a specified pattern:

```
GET COPYF
SET B4B7=90
SAVE COPYNOT B400 B698
```

You now have a new Utility! For example, to copyf all files except files with a ".A" extension, use:

```
COPYNOT *.A
```

One note of caution. In order for COPYNOT to work, the filename argument must contain a wildcard. Otherwise, it will simply attempt to copy the specified file, just like COPYF. Also note that the patch given above is valid for CODOS 2.0.

IN THE QUEUE

New and exciting things are under development at MTU for MTU-130 owners! Several projects are nearing completion and will soon be released as products. Here are some highlights.

WORD PROCESSOR SOFTWARE

We think you'll agree this one was worth waiting for. It handles proportional printing and proportional spacing for a "typeset" appearance (see the note below on the preparation of this newsletter). Text is justified on-screen with automatic word-wrap at all times, including during insertions. Documents can have variable margins, type pitch, line spacing, tabstops, and justification within the same document. A simple method is provided for generating subscripts, superscripts, underlining, wide characters, bold characters, etc. An interrupt-driven keyboard routine allows type-ahead with no loss of keystrokes regardless of the complexity of screen or disk activity. You can even include graphics images in the file!

"C" PROGRAMMING LANGUAGE

This is not a "tiny" or "small" implementation of the C language, but a complete compiler with all features except the bit field data type. Yes, it does support floating point arithmetic, long integers, pointers, structures, macros and all the other goodies in the original Bell System PDP-11 UNIX version. The compiler generates compact P-code which can be linked to separately compiled routines in a library. Execution time is many times faster than BASIC.

DIGITAL & ANALOG I-O BOARD

Though your MTU-130 comes with more I-O as standard equipment than many computers can have when fully expanded, we've been brewing up an add-on board to meet the specialized needs for laboratory, process-control, and other I-O intensive applications. Included on the board are two additional parallel ports, two additional RS-232 serial ports with both synchronous and asynchronous capability, an IEEE-488 port capable of talker/listener/controller functions, eight 12-bit Analog to digital channels, two 12-bit Digital to Analog channels, and a clock/calendar with battery backup and automatic battery recharge circuitry.

ABOUT THIS NEWSLETTER

This newsletter was prepared using a preliminary version of the MTU-130 word processor and was printed on an NEC-8023 dot matrix printer in a single pass, without any paste-up or other manual operations. Although inexpensive, the NEC printer has four character sizes, bit-mapped graphics, Greek and math fonts, and proportional print capabilities. MTU can supply you with this truly remarkable printer, color-coordinated with the MTU-130, for $645,complete with software driver and cable.

Trademarks: UNIX is a trademark of Bell Telephone Laboratories.


MTU
Micro Technology Unlimited
2806 Hillsborough Street
P.O. Box 12106
Raleigh, NC 27605, U.S.A.