TEC TALK

This page is for TEC owners. Through this, we can conduct a forum on the uses and future of the TEC. As we cannot reply to every letter sent in, we will attempt to answer letters of common interest through this page.

When writing in, put your letter on a seperate page if you are ordering kits etc. This helps us file things in some sort of reasonable order.

SENDING IN PROGRAMS VIA TAPE

We are looking forward to readers sending programs to us. There aren't any in this issue due mainly to the shortage of space. Hopefully, issue 16 will contain several pages or more.

A big factor in deciding weather or not we publish a "reader send-in" is the way the program is sent to us.

If you do have something to send, here is what we want you to do.

Provide us with a copy of the program Save it on tape with a crystal speed of half 3.58MHz. Put your name and address on the tape so we can send it back, if requested.

We also need documentation on the program. Write what it does and where it runs in memory and include any notes you may have generated. The first thing we will do is disassemble it and load it into our IBM clone. Here we can format it for publishing.

For the sake of our disassembler, please, if you can, put tables at the end of the program code and write down where the tables are located. This way we can use our HEX dump routine and tack the tables on at the end of the code.

JMON UP-GRADES

JMON has been designed to be upgraded without losing software compatibility.

Some likely changes are the removal of the low speed tape save (unless there is a storm of protest). This will decrease the software overhead in the tone routine and even-up the period measurement. The result will be an increase in the tolerance of different TEC frequencies and different tape speeds. This should make it possible to freely interchange half 3.58MHz and half 4.MHz tape software as well as allowing poorer quality tape players to be used.

The single stepper, which has no effect on the MONitor at all, may be shifted to a more specialized ROM to increase the stepper's abilities.

The keyboard and LCD RST's will not be changed, so any routine you write using these will run on future up-grades. The same cannot be said if you directly call into JMON. So don't do it!

ISSUE 15 CONTENT

Missing from the TEC section are two usual features. The reader send-ins and tutorial section.

The reason for this is mainly due to lack of room. Already the TEC section is the largest ever and the material left over will be a good start for issue 16.

A different direction is planned for issue 16. The basic lay-out will be two MAJOR add-ons and the rest of the article will be filled with programs (mine and yours, so send them in). There are a couple of reader send-ins that we intend to publish, so if you have sent something in already, we haven't just tossed it in the bin!

JIM's PACKAGE

This package is centered around JMON. The main feature is a complete line-by-line disassembly of the JMON ROM. I hope that, with careful study, you will be able to look at any instruction and understand its role.

My programming style is very optimized. Generally my programs are short and to the point. This does make them a little difficult to read but at the same time by studying and learning my ideas, your own programming abilities will be improved.

If the role of every instruction escapes you, you will still learn important concepts and a better way to do some things.

I wish I had a Jim's package when I was just starting out!

The package will be 20 pages long as this is the limit of our collating photocopier. If there is enough room, some other notes and programs will be included.

It is a pity that such a listing was not available for the earlier MONitors.

Because Jim's package contains every byte in JMON, you can actually burn your own JMON ROM.

Keep in mind that this means typing in 2k worth of program and one mistake will ruin the whole MONitor.

If you feel up to it then go to it. We don't mind you doing this ONCE for YOUR OWN USE.

This offer does not apply to schools or commercial buyers.

If you don't wish to try to type out JMON, I present you with this offer:

Purchase and pay for JMON and Jim's package together, and save $3.50.

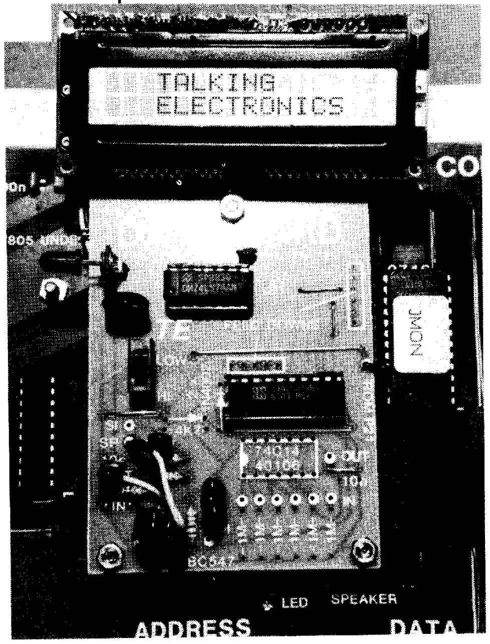
This means the total price for both is $27.50 instead of $31

ISSUE 16 and the TEC

Most of issue 16's TEC pages have been allocated and about half of these are already finished.

If you have some thing for us, don't waste time if you want the chance to see it in print.

- Jim



Completed DAT board displaying "TALKING ELECTRONICS." See the DAT board article starting on P. 47, and LCD article on P.52. Read the whole TEC article before starting anything!

THE TAPE SYSTEM

This discussion covers all the areas needed to use the tape save and its various options.

TEC CONSIDERATIONS

The tape software works on any type of TEC, the only consideration is the various different clock speeds.

The following description generally applies to TEC's with a crystal oscillator that is fitted with a colour burst (3.58MHz) crystal and divide-by-two stage.

If you are still using a 4049 based oscillator, the tape system will work ok, but it will be very important to note the TEC clock speed when saving as the TEC must be set to the same speed when re-loading. Another problem can be the drift in frequency over a temperature range and the different oscillator frequencies between TEC's.

When saving a tape, the best idea is to wind the clock up to full speed, and then turn back the speed control pot one quarter of a turn. This will allow you compensate for speed drift if ever required.

The tape also works very reliably with a 4MHz crystal and divide by two stage, however a tape written using a 3.58MHz oscillator cannot be loaded by a TEC that uses a 4MHz oscillator, and vice versa.

If you are sending programs into TE on tape, they must be recorded with the 3.58MHz crystal. (divided by two).

The tape system has been extensively tested and found to be very reliable under a wide range of conditions. We don't expect you to have any trouble in getting it to work reliably for yourself.

LET'S BEGIN

To start with, you need a JMON monitor ROM as the tape software is inside this ROM.

Secondly, you will need a cassette recorder with both "mic" and "ear" sockets. Any audio cassette player of reasonable quality should be ok, provided it has the two sockets mentioned above.

We have tested more than six types, and found them to be quite suitable.

Thirdly, you will need to have constructed the cassette interface on the LCD interface board and have made up the two connecting cables, with 3.5mm plugs on each end. Finally you will need a new C60 or C90 cassette of the better quality types, such as TDK or Sony. We found the cheap tapes from the junk shops or supermarkets to be unreliable. (Some of them didn't work AT ALL, so don't take the chance).

Now connect the "mic" on the tape recorder to the "tape out" from the TEC and "ear" socket to the "tape-in" on the TEC. (It's a good idea to mark the cables between the recorder and the TEC to prevent incorrectly connecting the leads).

Insert a tape and we are ready to learn how to operate the system.

HOW TO OPERATE THE TAPE SYSTEM.

We will start by saving a few bytes at 0900. Enter at 0900 the following: 01 02 03 04 05 06 07 08 09 0A.

OK. Now connect up the tape recorder as described above and call up the tape software by pushing shift and zero at the same time or Address, "+","0" consecutively.

The TEC display will now show "SAVE-H" and this is the heading for SAVE at HIGH SPEED. Now select this by hitting "GO"

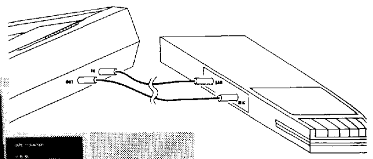
The display will now have a random two-byte value in the address display and "-F" in the data display. The "-F" in the data display is for the file number, while the address number is just junk from the RAM. You can enter a file number by pressing the data keys. Enter anything you want. The numbers you enter will shift across in the same fashion as when entering an address on the MONitor. Then when you have entered a file number, press the "+" key. The data display will now show "-S". This is where you enter the start of the block you wish to output. Enter 0900, and then press "+". The data displays now show "-E." This is where you enter the address of the last byte of the block to be saved.

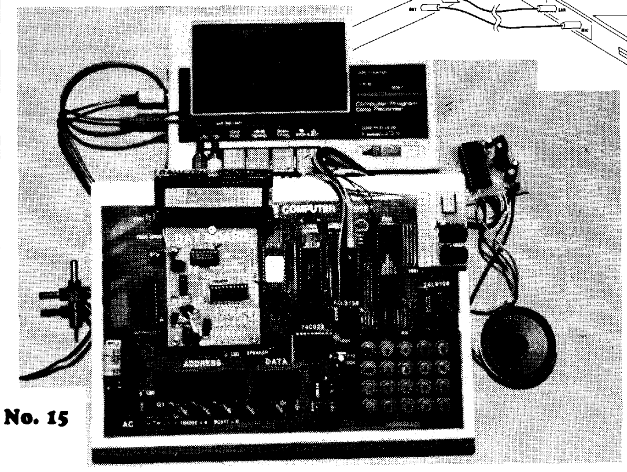
Enter 090A and press "+". The next data display is "-G". This is the OPTIONAL AUTO-GO address, this is always set to FFFF by the software as this is its NON-ACTIVE state, i.e. NO AUTO-GO upon a re-load. ANY other value entered here will result in an automatic execution upon a SUCCESSFUL LOAD AT THE ADDRESS ENTERED HERE.

We don't require auto execution, so leave this at FFFF.

Now press play and record on the tape recorder and wait for the clear plastic leader to pass if at the start of a tape. Incidentally, it is not a good idea to remove this leader as has been advised in another magazine, as it protects the tape from stretching and possibly breaking, when rewound.

When the tape is right, press GO. The display will blank and a continuous tone will be heard from the speaker. After a few seconds the file information will be outputted and then a period of high frequency tone. This "middle sync" tone is to cover the time that the filename is displayed when re-loading.





After the high tone, the code will be outputted and also a digit will appear on the TEC LED display. This is the number of COMPLETE pages to be saved. In this case it will be zero.

A point to raise here is that if you ever accidentally enter a start address that is HIGHER than the end address, when GO is pressed, the software will detect this and display "Err-In". In this case, Push "+" or "-" to go back to the perimeter handler where you can correct the error.

When the code has been saved, a short end tone will be heard and then the menu will re-appear with "-END-S", meaning end of save.

Once the code has been saved, rewind the tape.

To re-load the tape press the "+" key and you will see "SAVE-L" on the display, then "TEST-BL", "TEST-CS", then you will come to "LOAD-T" (for load tape). Note that there is no "TEST-H" or "TEST-L" for low and high speeds as the test and load routine will load either speed automatically.

Press GO. The data display shows "-F" for file number. This will be as you left it when you saved. When loading or testing from tape, the file number here determines which file will be subject to the selected operation. If you enter FFFF here, the next file found will be used, regardless of its file number.

For now, we will leave it as it is.

Next push "+". The data display will show "-S", meaning Start address. This is always set to FFFF by the software. The start address allows you to optionally load a file or test a file at an address different to the one on the tape, (which is the address from which it was saved). To demonstrate its operation and to make it a more convincing trial, we will enter 0A00. The file will now be loaded at 0A00. If you press the "+" key again, you will be back at the file name. (This last point demonstrates the programmable number of "windows" feature of the perimeter handler. It was set up for 2 "windows" by a short routine entered from the Menu driver before passing control to the Perimeter handler, remember that there was 4 "windows" when you saved the file).

Now press GO.

The display will blank. Now start the tape playing. The sound from the tape will be echoed on the TEC speaker. Soon the leader will be heard and it should sound as crisp as when it was saved. If not, experiment with the volume. The interface allows for a wide variation of volumes but 3/4 volume is a good place to start.

After the leader has passed, the file name is loaded and should appear on the display. If it was not correctly loaded, "FAIL-Ld" will appear. In this case experiment with the volume and retry. After a few seconds the file name disappears and the number of complete pages to load are displayed on the middle digit. The code is now being loaded.

The code is loaded very quickly and hopefully a "PASS-Ld" will appear. If not, re-try with a different volume setting. After you have successfully loaded, hit reset and ADdress 0A00 and 01, 02 etc. will be found.

If you are unable to get a successful load after many attempts, then skip ahead to the trouble shooting section.

Now we have a successful load, we will experiment with the TEST BLOCK function.

Change a byte in the 0A00 block. Now call up the tape software (Shift-0, or ADdress, "+" ,0), select "TEST-BL", and LEAVE FFFF at the optional start. ("-S") Then rewind the tape and play it back like you did when loading.

At the end of the test, the display comes back with "PASS-TB". Now do this again, but this time enter 0A00 at the optional start and FFFF for the file. This will demonstrate the load/test next file feature.

Because 0A00 has been entered in the optional start "window", the test will be between tape and the code at 0A00.

Rewind the tape and press "GO" on the TEC, then play the tape.

Because a byte has been changed, the test this time will fail and the display will show "FAIL-TB."

Use the test-block feature whenever you wish to compare a tape file with a memory block or test that a save operation was successful.

If ever revising software on a tape of which you do not have a copy in memory, use the test checksum (TEST-CS) to ensure that the file is good. By use of the "LOAD NEXT FILE" feature (FFFF in the file number window) you can go through a tape completely, checking each file.

THE "AUTO-GO"

To use the Auto-GO feature, you must enter the required GO address WHEN YOU SAVE THE FILE. The go address is entered under the "-G" data display.

Experiment with the following:

```
0900: 21 10 09 11 00 08 01 06
0908: 00 ED B0 CD 36 08 18 FB
0910: 6F EA C6 EB E3 EB.
```

Save this as described above, but this time enter 0900 under the "-S" heading, 0921 under "-E", and 0900 under "-G".

Now re-load it and if the load is successful, the program will start automatically and an appropriate display message will appear.

USING THE TAPE SYSTEM.

The primary use for your tape save system is as a mass storage device for your files.

Files may be saved and loaded as described previously, the important addition here is good paper-work habits. It is very important to keep a log of your files or you will quickly forget what you have, where it is located, and you will end up writing over your files!

Your log system must include identifying each cassette and the side of the tape, the files on the cassette in the correct order, how many of each file, the date and any notes on the file. If your recorder has a tape counter facility, it makes good practice to record the readings from this, so that files may be quickly found anywhere on the tape.

Also a great aid is to log approximately the location of each file e.g. half-way, 30 seconds from rewind from end etc.

Apply the above idea to the start of vacant area on the tape also.

Another very good way to use the tape system is as a "RUNNING LOG", where a whole side of a cassette is used to save a developing program, stage-by-stage. If you crash your program, you can re-load it back from tape. A good idea here is to use the high byte of the file number as the program identification and the low byte as the progressive count or version number on the tape.

When you have a final version, then save that on a permanent cassette. The "RUNNING LOG" cassette can then be used over and over.

Once again, paper work is very important. Make sure you document any differences between successive files. This may help later in de-bugging. Also, always include the date and time as this will give a chronological order to your work.

If you are wondering how many times you should save a file, and at what speed, the answer really depends on the reliability of your system.

The major factors in reliability are your tape player and cassette quality and how well you constructed your interface. If any of these are borderline, the system may work but you may have a higher than normal failure rate. Our tests show reliability at better than 98% on saves of 2K blocks. Different cassettes and players were used over many months and rarely did a fault creep in.

You can test your system out by saving the monitor 10 times on each speed and then perform a BLOCK TEST. You should get at the very least, 17 out of 20 passes. If not, some trouble-shooting may be required. If you get 19 or 20 you could probably get away with high speed saves and not have to worry about checking them on your running log. For permanent storage, a good system is a high speed save, then two low speed saves and check each afterwards.

The low speed save should be more reliable than the high speed save as the low speed save will tolerate the occasional hiccup. However, this extra reliability does not cover all possible causes of failure, e.g. problems related to frequency or bandwidth restrictions of your tape player as the period is not changed only the ratio of pulses.

Finally, a file that is absolutely necessary to be retrieved from tape must be stored on two tapes. This provides a double back-up facility against accidental erasure or damage.

"OH NO!" IT DOESN'T WORK.

If your tape system fails to work correctly, then check the interface board or better still, have a friend check it.

Eliminate any problem and re-try.

If problems still exist, test the cassette player with a normal pre-taped audio tape. The music should sound normal and not flutter. If it flutters, the tape player is due for a service or replacement, or if battery operated, the batteries may be flat.

Various sections may be eliminated by listening to the tape signal. If the signal saved on the tape sounds ok when played back on the player, but is not heard on the TEC, check the input section of the interface board and also the "E" output of the player with a pair of Walkman-type headphones.

It is possible that the volume output is not high enough to be amplified on the interface board.

This is very unlikely though on ordinary tape players but we found this to be the case with our VZ-200 data cassette player.

If no signal is getting to the TEC and everything else seems to be ok, test the input buffer by setting the tape software to load and taking the input high and low with a jumper lead.

The LED on the speaker should echo the inverse of the input. If not, shift the jumper to the collector lead of the input transistor and repeat the process.

If the speaker LED now toggles, the input transistor is faulty.

If not, investigate the latch chip. Make sure all the pins are well soldered and the feed-throughs are connecting properly.

If the tape signal is heard, in the TEC speaker, but the file number is not recognised, loaded correctly, or the tape fails to load the data blocks consistently, try a better quality cassette tape. If problems persist, try a different player as the signal may be distorted or not have enough amplitude.

If you still can't get it to go, a repair service is available for $9.00 plus $2.50 postage.

RUNNING OLD PROGRAMS WITH JMON

Most old programs will run with JMON without to much alteration.

For most, they will only need to be relocated from 0800 to 0900 and that's all. Those that use old MON-1 routines such as the running letter program or the tune player can't, of course, run with JMON.

Of those which use the keyboard, most can be easily altered but some require a complete overhaul.

These ones listed below cannot run on JMON, or require more extensive changes that those presented here:

SPIROID ALIENS, HALILOVIC'S PIANO, BIG BEN CHIMES, WINNERS CALL, YOU'RE DEAD FUNERAL DIRGE, TOCCATA, THE STRIPPER, ADDING AUTO REPEAT, AUTO RETURN AND STOP, AUTO MOVEMENT AND HALT, THE ROMMED PRINTER SOFTWARE and SPACE INVADERS SHOOTING.

Those not mentioned should run ok if re-located to 0900 and the mods listed below are done, if required. These mods apply only to routines which use the keyboard.

TEC KEYBOARD THEORY

Basically the keyboard usage of earlier routines is broken up into two types: Those which halt and wait for a input via the interrupt, and those which intialize the input buffer (the interrupt vector register or I register) and read it "on the fly."

The first type may again be broken into two groups. Those which explicitly read the value from the input buffer, with a LD A,I instruction (ED 57), and those which assume the input to be inside the Accumulator after the interrupt has occurred. (Remember the earlier MON-1 series did not save the accumulator during the NMI routine but instead returned with the input key value inside the accumulator. A disastrous state of affairs)!!

JMON

The specially-provided routines in JMON will work for all the above types, the only difference being the way you alter the program in question. Here's how to alter the routines:

To up-date any type which uses a HALT instruction (76H) as part of the keyboard input section, change the HALT instruction (76H) to a RST 08 (CF).

The RST 08 routine SIMULATES the halt instruction by first looping until NO key is pressed then looping until a key IS pressed.

After a key press is received the input value is masked to remove unwanted bits and stored in the input buffer, (the interrupt vector register), in identical fashion to the old interrupt routine.

Now if the halt instruction is immediately followed by a LOAD A,I (ED 57), you may leave it as it is or remove it as it is not required any more as the input value is returned in A.

If a program doesn't have a HALT instruction but uses the keyboard, then look for the LOAD A,I instruction (ED 57). Change this to a RST 20 (E7) and place a NOP over the unused byte. Notice that this IS NOT the same RST instruction as above.

Be careful not to mistake the LOAD A,I (ED 57) with a LOAD I,A (ED 47) otherwise you program may get upset and go on strike.

Programs which have neither a HALT or LD A,I instruction cannot be altered by any of the above methods because they enter a continuous loop and require the interrupt to force an input value into the accumulator. A classic example of this is the "space invaders shooting" on page 14, issue 14. This above loop is located at 0821. (while you're looking at this, grab a pen and change the byte at 0812 from 02 to 01, at least it will run correctly with MON-1)! All the above types are among those listed as not being suitable for modification via these methods.

FINALLY

If you find a program which doesn't work (we haven't tried them all) or something else interesting, please write and let us know.

USING THE KEYBOARD IN YOUR PROGRAMS

The new keyboard set-up is no more difficult to use now than before. In actual fact it is easier and requires less bytes than before thanks to the use of the RST instructions.

Four RST's are provided to handle the keyboard in different ways. The first RST we shall look at is RST 08 (CFH). This RST is a "loop until a NEW key press is detected" routine. If you refer to the section on running old programs, you will see that this RST is used to simulate/replace the HALT instruction. (You know how to use it Already!)

An important feature of this RST is that it ignores any current key PRESSED, that is if a key is being pressed when this RST is performed, it will not be recognized. This mimics the NMI which only recognized a key press once. (This is why the auto-repeat feature could not be done with the keyboard hooked up to the NMI).

When this RST detects a valid key press, it inputs the value from the key encoder and masks the unwanted bits and stores the input in the interrupt vector register (as did the MON-1 series). The input value is also returned in the accumulator. The shift key can not be read from this (or any other MONitor keyboard routine) as the shift input bit (bit 5) is masked off.

Here is an example of its use:

```
0900  CF      RST 08
0901  FE 12   CP 12
0903  20 04   JR NZ,0909
0905  3E EA   LD A,EA
0907  18 06   JR 090F
0909  FE 01   CP 01
090B  20 F3   JR NZ,0900
090D  3E 28   LD A,28
090F  D3 02   OUT (02),A
0911  3E 01   LD A,01
0913  D3 01   OUT (01),A
0915  18 E9   JR 0900
```

The first thing you should notice when you enter and run the above, is that the "go" key is not detected when the routine is first started, even though it is being pressed. This is because the first part of the RST loops until the key being pressed is released. The RST then loops until a new key press is detected. When the RST returns, the input value is both in the interrupt vector register and the accumulator. The rest of the routine tests for either a 01 or "GO" key and outputs to the display.

Use this RST when ever you want the TEC to go "dead" and wait for a key press.

The second RST is RST 10 (D7). This is similar to the first RST but has one very important difference. The difference is that this RST DOES NOT wait for a key being pressed to be released before returning. While this is not as likely to be used as much as the first RST, it does have some good uses. Any program which requires some action to take place while there is a key pressed, but do nothing when there is not, may make good use of this RST. Some possible uses include random number generation on the time the key is held down; count while a key is pressed; turn on a relay while a key is pressed etc. As you can see, this RST simulates momentary action switches.

This RST exits with the input stored in the same fashion as the above RST.

The third RST is RST 18 (DF). This is a LED scan loop and keyboard reader. The scan routine will scan the 6 TEC LED displays once with the display codes addressed by the address at 082A. (0800 is stored here by JMON. You can leave it as it is, just store what ever you want at 0800 before using this RST). After the scanning routine is done, the keyboard routine is called. The keyboard routine is actually called from RST 20. What happens is this. After the scan has been called from the RST 18, the program continues on at 0020, which is the start address for RST 20. So the RST 18 is the same as RST 20 EXCEPT THAT RST 18 CALLS FASTSCAN. Therefor the description below applies to BOTH RST 18 AND RST 20.

This keyboard routine is very intelligent and is able to detect several different conditions.

One important feature is that it "remembers" if it has already detected the one key press and it ignores it if it has. This provides us with a "ONE AND ONLY ONE" key recognition for each key press. Each key press is "recognized" on the first detection.

The key is checked for a "FIRST KEY PRESS" by the use of a flag byte. When the routine is entered AND NO KEY IS PRESSED, this flag byte is CLEARED. When a key is detected, the flag byte is checked. If zero, the key is accepted as a "FIRST KEY PRESS." The flag byte is then set to stop further "validating" of the same key press. The input value is then masked and returned in the Accumulator (only).

If the flag byte IS NOT CLEAR, then the key is not recognized as "valid."

Careful consideration was giving to the interaction of the MONitor and user routines so that the "GO" command from the MONitor WILL NOT BE TREATED AS THE FIRST KEY PRESS of a user routine. (This was achieved by using the same flag byte for both JMON and any user routine).

HOW TO INTERPRET THIS RST

If a key is recognized as a "FIRST KEY PRESS" then the ZERO FLAG will be set to its active state (a logic 1) and the MASKED KEY INPUT will be returned in the accumulator.

If the key is NOT valid then the ZERO FLAG will be clear AND the accumulator WILL HAVE ALL ITS BITS SET (FF).

(FOR ADVANCED PROGRAMMERS)

In addition to the zero flag being conditionally set, the RST 20 (E7) also sets the carry conditionally, according to the following conditions:

If there is a key pressed then the carry will be SET REGARDLESS of whether it is a "first key pressed" or NOT. If NO key is pressed then the carry is cleared.

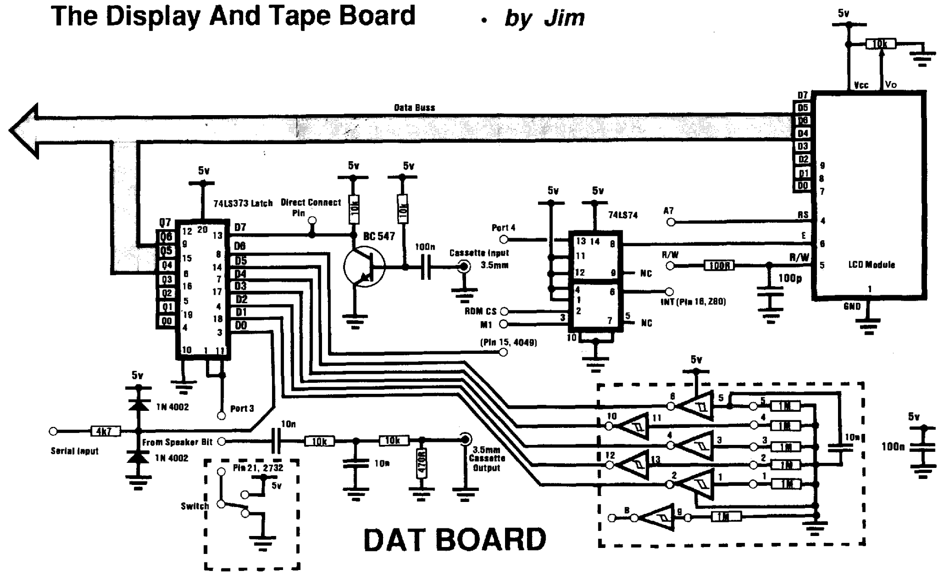
This allows you to interpret the keyboard the way you want, while still giving you the convenience of using the RST to do some of the work.

Jim's section cont P 47.

THE DAT BOARD

Parts $50.80
PC board $4.55
Total $55.35
Parts without LCD: $11.80

The Display And Tape Board • *by Jim*



This board will change the way you program for ever. The DAT BOARD is perhaps the most vital addition to the TEC ever. Not just a part time "add on," but rather a permanent addition to your TEC.
Once you start using it, we think you'll agree.
The name "DAT" is an acronym for Display And Tape. While others brawl over "their" DAT, (have you seen one?), we have quietly slipped in the back door with our version.

The DAT BOARD provides these functions:
* 16x2 LCD display.
* Cassette tape I/O interface.
* Single stepper module.
* 5 Buffered and latched input bits.
* 1 Inverter for general use.
* Diode clipped input line. (For RS232 input)
* MON select switch.

PORT 3

Port 3 addresses an input latch. Below is a break-down of the bits on port 3.

BIT#	
0 -	Serial in
1 -	input 1
2 -	input 4
3 -	input 2
4 -	input 5
5 -	input 3

The above are the inputs from the 74C14.

6 -	key pressed signal.
7 -	Tape input.

CONNECTION

Up until now, TEC add-on's have been connected via the expansion port. We wished to avoid this as there are too many devices cluttering up this area already. The search was on for a better place to put our new board. We decided upon the blank area left of the eprom, because it is common to all TEC's and has up until now not been used by anything else.

But there's nothing to connect to there! I hear you say. Well not quite, Simply solder a cut-up I.C. socket onto the links and you have an (almost) instant data buss socket. The DAT BOARD has a set of feed downs that push into the sockets and serve the dual purposes of connection and fixation.

PARTS LIST

1 - 100R
1 - 470R
4 - 10k
1 - 10k mini trimpot

1 - 100p ceramic
2 - 10n greencaps
3 - 100n greencaps

1 - BC 547
1 - 74LS373
1 - 74LS74

1 - 5mm LED (for trimpot handle)
2 - 3.5mm sockets
1 - 20 pin IC socket
2 - 14 pin sockets (one to cut-up)
1 - 20cm 12 way ribbon cable
1 - 50cm figure-8 shielded cable
1 - 1.2 metres hook-up wire
4 - 3.5mm mono plugs
1 - 100cm tinned copper wire
1 - Female matrix connector
3 - 32mm x 2.5mm bolts
9 - 2.5mm nuts
1 - 16 character x 2 line LCD*
1 - DAT PC Board
*Don't Forget: The LCD display can be bought separately.

The feed downs are simply lengths of stiff wire soldered to the underside of the P.C. that extend about 1 to 1.5 cm down to push into the I.C. sockets.
The fixing of the DAT BOARD is also aided by three "stand offs," in the form of three bolts with nuts to tighten against the board. These may extend through the TEC board if you want as there is no track work underneath.

CONSTRUCTING THE DAT BOARD

Originally, the kit of parts for the DAT BOARD was going to be supplied in two sections. We have changed our minds since, but have decided to present these construction notes unchanged.
The first thing to do, is to fit ALL the links, regardless of what section you are constructing.
If you have already built the TAPE and keyboard section and/or are now constructing the LCD/SINGLE STEPPER interfaces then skip ahead to the respective notes. Once you have built the LCD section skip back to the notes on inserting the feed downs, stand-offs and control buss leads.

THE TAPE AND LATCH SECTION

Most the components for the TAPE SECTION are fitted on the bottom left corner of the board. The exceptions being a 100n greencap, that goes on the middle left of the board, the latch chip and its socket. Fit these in the order you prefer and then solder a short piece of tinned copper wire in the hole marked "SP."
This is where the female matrix connector will slide on. If you are wondering why we recommend a piece of tinned wire instead of a male matrix pin, the reason is that the force needed to push a female over a male matrix pin is far to great to be healthy for the TEC or DAT PCBs. (The keyboard is destructive enough). The tinned wire can be tinned again to give just the right fitting diameter, if required.
After fitting all the components, cut the length of hook-up wire into 4 equal sections. Strip and tin each end of all the lengths. Solder two pieces to the ground strip next to the tape in and tape out pads on the DAT BOARD. The other ends of these wires solder to the top tags of the 3.5mm sockets.
Solder the two remaining wires to the tape in and tape out pads. The other ends are soldered to the DIAGONALLY OPPOSITE tags on the 3.5mm sockets. Keep track of which socket the wires are joined to, and mark them accordingly.
Drill two holes large enough for the 3.5mm sockets in the back or side of the RETEX case and fit the sockets in place.
Strip the ends of the shielded cable and twist the shield into one strand. Remove the covers of the 3.5mm plugs and slide them onto the figure 8 cables, so they are back to back. Solder the shields to the larger tags on the plugs. The middle conductor is soldered to the smaller tags. Do this for each of the four ends.
Solder a 5cm piece of hook-up wire on the 1K resistor which connects the output latch to the speaker transistor. The wire is soldered on the LATCH SIDE of the resistor. The other end of the wire is soldered to the female matrix connector. This matrix connector slides over the pin marked SP on the DAT BOARD.
Now you are ready to insert the feed downs.

INSERTING THE FEED DOWNS

The feed down are made of stiff tinned wire of about 2cm length. The easiest way to solder these is to solder a continuous length in each hole, and then trim it down afterwards. Do this for all the feed downs and try to get them straight as possible.
The feed downs plug in to a cut-up IC socket soldered across the links near the EPROM. The socket is soldered where the links form a straight line as they disappear into the TEC PCB. (See diagram). If you want, you may make the feed downs longer, remove the links, and permanently solder the DAT BOARD in place. Of course, you will need to put jumpers beneath the board to replace the missing links. This arrangement will provide a far more reliable circuit connection. Make sure you have finished the board COMPLETELY before you do this, as you will not be able to solder underneath the board afterwards.

CONNECTION OF THE CONTROL LINES

There are 10 control lines that are soldered to the bottom of the TEC board. A 20 cm 12 way ribbon cable is used to make all the connections. The ribbon cable is soldered to a row of pads on the DAT BOARD about 2.5cm below the top edge. The ribbon cable is soldered to the BOTTOM SIDE of the DAT BOARD and then drops down between the TEC board and the RETEX case (if you have one).
All the connections to the DAT BOARD are printed on the solder side of the board while the connections to the TEC are made as per the wiring diagram.
The two 3.5 mm sockets for the tape in/out are mounted in either the back or side of the RETEX CASE. If you do not have a case, then the sockets can be connected with short pieces of wire and left "floating." We do not recommend that you drill holes in either the TEC or DAT boards for the sockets. This is to save the expensive TEC board from the excessive force involved in plugging and unplugging the leads. The best idea is to hold the sockets when inserting the leads.

THE STAND-OFFs

In addition to the feed downs, three bolts act as stand-offs. The head of these bolts sits on the TEC board or, if you wish, you may drill into the board and feed the bolts up through the board. If you have the original TEC-1 board with the 8212 latch chips, the top bolt will not be able to be feed through the board as there is track work associated with the (now aborted) on-board tape interface and battery backed RAM.
If you have drilled the holes, then feed the bolts up from the bottom of the TEC and lock each in place with a nut. A second nut is screwed down to about 1cm off the TEC board on each bolt. This sets the height of the DAT BOARD. The DAT BOARD is then placed over the two bolts and a third nut is tightened onto the DAT BOARD.
If you to not wish to drill into your TEC, which is quite understandable, then place a nut on each bolt and wind it down to about 1cm from the head. Poke the bolts through the the hole in the DAT BOARD and tighten down the second nut.
Next, insert the board and note how high it is off the TEC. Ideally it should be 1.5 to 2cm off the board. Trim the feed downs until you are happy with the height. Adjust the stand-offs until they all sit neatly on the TEC board. Finally, a blob of blu-tack can be used to secure the top stand-off on to the board. This will help keep the DAT BOARD square on the TEC.

TESTING THE LATCH/TAPE INTERFACE

The latch is easily tested by running up JMON. If the keyboard works then the latch is obviously working. You can test each bit of the latch by taking the remaining inputs to ground. These pins are connected to pins 2,4,6,8 and 12 on the 74C14 socket and also pin 3 of the latch chip itself. Make sure that you don't have the 74C14 fitted as this may damage the chip.

The following program will echo the latch on the LED display:

```
0900 3E 3F D3 02 DB 03 E6 3F
0908 D3 01 C3 00 09
```

To test the tape, refer to the pages on using the tape system that show how to use and trouble shoot the tape interface.

THE SINGLE STEPPER/LCD INTERFACES

If you are constructing this section before the tape/latch section, you will need to make a modification to the TEC. The mod is to add a 4k7 resistor between pin 15 of the 4049 and pin 10 of the Z80. The purpose of this mod is to route the DATA AVAILABLE SIGNAL to the DATA BUSS. Without this, JMON is unable to read the keyboard. (This mod is described numerous times throughout this issue). The LCD interface consists of just four components. They are a D flip flop, a 100p cap, a 100R resistor and a 10k trimpot. The D flip flop, (that was spare) is configured to act as an INVERTER!! This design saved us from having to use another chip.

The single stepper interface simply uses one half of a dual D flip flop!

CONSTRUCTION NOTES

These 2 interfaces are simple to construct. Just take care with the orientation of the 74LS74 chip. If you have a spare LED on hand then you can solder it onto the trimpot to use as a knob (one is provided in the kit).

FITTING THE LCD

Place the LCD FACE DOWN on the work bench and feed a 5cm length of tinned copper wire into each hole on the LCD. Solder the wires in place and then, starting at one end, trim the wires to form a ramp. This helps you to insert the 14 wires one-at-a-time into the DAT BOARD. The DAT BOARD edge connector is placed at the top of the DAT BOARD and the LCD overhangs the board like a verandah.

Insert the LCD into the DAT BOARD as best you can. A second person with a pair of tweezers could help tremendously in getting each wire down its hole.

After you have fitted the wires into their holes, position the LCD to the height you want. This should be about 1cm to 1.5cm, and carefully solder it in place.

TESTING THE LCD

After you have finished construction and wired the DAT BOARD to the TEC as shown in the wiring diagram, you're ready to go. Fit the board in place and turn the 10k trimpot clockwise when looking at it from the left. Turn it as far as it goes, then turn it back just slightly. This sets the contrast level and if it is not approximately at the position described above, nothing will appear on the LCD.

If you have JMON then fit it into the EPROM socket and power up the TEC. All things being equal, the display will show the following:

```
0900>xx xx xx xx
Data  xx xx xx xx
```

If not, the most likely cause is that one of the data lines is not getting to the display. The easiest way to check this is to type in the following:

```
0900 3E 55 D3 04 C7
```

AFTER you have entered this, connect a jumper between port 4 and the wait line of the Z80. When you have done this, hit go.

The TEC should go "dead." Now, with a logic probe, test the edge connector of the LCD. Starting from the right, the logic levels should be: H, L, H, L, H, L, H AND L.

If not, then check all the connections and retry until right. If the connections are right, but there is nothing on the display, check the voltage on pin three of the LCD. This voltage should be in the range of 0.5v to 1v. Adjust the trimpot until you measure this voltage.

Still no luck? Turn off the TEC, hold reset down and turn the TEC back on while still holding down the reset. The top row of the LCD should be dark and the bottom line should be light. If not then there maybe no power getting to the LCD, the contrast voltage may be incorrect (but you have already checked this), or the display has been damaged, they are all tested before they leave TE).

If the top line is dark when power is applied but the display does not respond when reset is released, then put your logic probe on pin 6 of the LCD. Hold down the "+" key and watch the logic probe. Pin three should pulse HIGH each time the TEC beeps. If not then check that you have the wire going to port 4 in the correct place. Check the track work around the 74LS74 chip and the chip itself.

If pin 6 seems ok, then check that the 100p cap is fitted as this is VERY IMPORTANT. Pin 5, the r/w line, should always be pulsing. Check this with the logic probe.

The only other line left to test is the register select (RS). This line is address 7, and the easiest way to check this is with a continuity tester. If the LCD clears when power is applied, but nothing appears on the LCD, then it is odds-on that the cause is address 7 not being wired correctly.

TE REPAIR SERVICE

Still can't get it going? Check it all through again, keep in mind that the most likely cause is a mistake in your wiring. As a VERY last resort (after ringing us) send it in and we'll see what we can do.

Our repair fee is $9.00, plus $2.50 for post and handling. This includes replacement of all parts except the LCD (that was tested before leaving us). Before you send it in, remove the control buss wires (the ribbon cable) from the DAT BOARD. Pack it up securely and send it down. If you want the tape section tested leave the 3.5mm sockets connected.

TESTING THE SINGLE STEPPER

This is easy. With JMON fitted, enter this at 0900:

```
0900: 00 00 00 00 00 C3 00 09
```

Now, press shift 2. The single stepper will show 0900 PC. Press any data key and the single stepper will cycle automatically. The occasional clicking you (may) hear is a result of the interaction of the interrupt response cycle and the decoding of the 74LS138 decoder chip.

If the single stepper doesn't work, then check your wiring as it is doubtful that the 74LS74 chip is faulty.

WHAT THE LCD INTERFACE DOES

The LCD is designed to directly interface to microprocessors. Unfortunately there are two main types of microprocessor buss timing and the LCD is designed for the wrong type (as far as we are concerned). In order to get the LCD to interface to the Z80, a little bit of juggling with the timing is needed.

The first problem is the the LCD requires an active HIGH Enable signal. This has been achieved by inverting the PORT 4 I/O select line. This inverting is done by the spare D flip flop on the DAT BOARD. By looking at the TRUTH TABLE for the 74LS74, I found that it was possible to configure it as an inverter if I used the CLR pin as the DATA input!

To cut a long story short, the idea worked. Eureka!

The next problem is the LCD requires R/W to be stable on the falling edge of the E signal. If you look at the Z80 timing, you will see that the R/W line and the IORQ change state simultaneously. By the time that IORQ has gated port 4 and the port 4 signal has been inverted, the R/W line will actually change (slightly) before the E line on the LCD!

To overcome this problem, a simple RC network has been placed on the R/W line. This RC delay holds the R/W line stable while the E line goes low. The time we are talking about here is just a fraction of a microsecond, but that is all it takes for the chips in the LCD to accept or reject the in-coming signals.

Another problem is that the LCD requires 2 ports to communicate with the Z80. It also wants to decode the second port itself. This is a common requirement of many peripheral devices, and the solution provided here is also useful for all these.

To give the LCD its second port, and let it decode it for itself, address line 7 has been presented to the LCD. This means that the second port is decoded (by the LCD) on port 84.

DISPLAY CONTRAST

The LCD requires an external voltage to set the contrast level.

The contrast of LCDs varies with temperature and viewing angle. To allow for this, the LCD has an external contrast control. The contrast is controlled by adjusting the voltage on this pin. This is the function of the 10k trimpot, that is wired as a voltage divider.

OPTIONS

Several optional extras can be added to the DAT BOARD. Below is a description of each:

MON SELECT SWITCH

When you add the DAT BOARD, there may not be enough room between the board and the EPROM to fit your MON select switch. If this is the case, provision has been made to fit the switch to the DAT BOARD. Simply install the dotted link and move your switch to the dotted switch position on the DAT BOARD. Run a wire between the pin marked 'ROM P21' and pin 21 of the EPROM.

SERIAL INPUT

The SERIAL INPUT (SI)

This input is for a serial signal, or a RS232 level signal from a printer or RS232 device. This input clips the signal, which can be +/-15V to +/-25V, to safe logic levels.

This signal winds up as D0 on the 74LS373.

THE 74C14

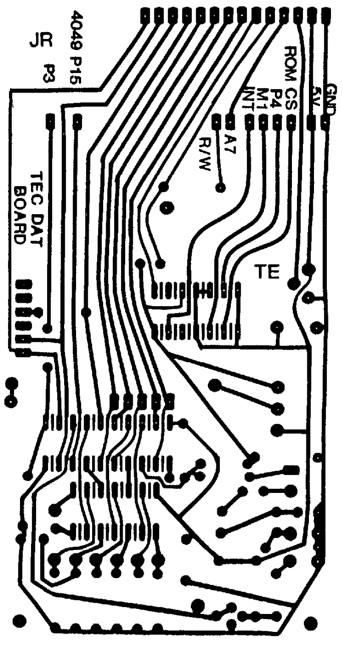
This has been added to increase the versatility of the DAT BOARD. Some possibilities for it include a touch sensitive qwerty key pad, an external time reference, a thermistor controlled oscillator for temperature measurement or just buffered inputs. Nothing permanent has been planned for it, it is mainly for experimentation. We are open to your ideas!

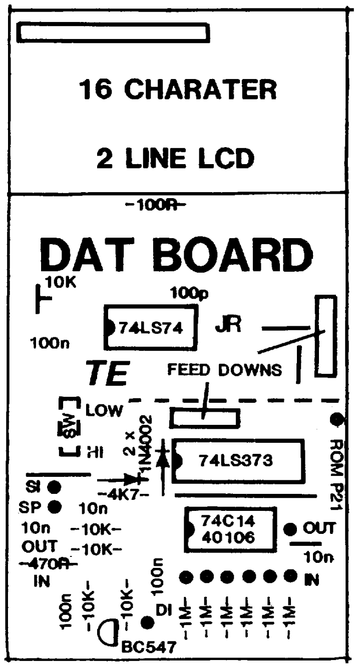
THE DIRECT CONNECT PIN

This is located between the transistor and the 6 x 1M resistors. The purpose of this pin is to allow direct connection between two TECs. One TEC can down load to another through the tape software or a serial communication program. (I have a 9600 Baud routine that also talks to IBM's and compatibles).

THE UNUSED INVERTER

The input for the unused inverter is the right most matrix pin on the bottom right-hand side of the DAT BOARD. The output is the matrix pin directly above it.



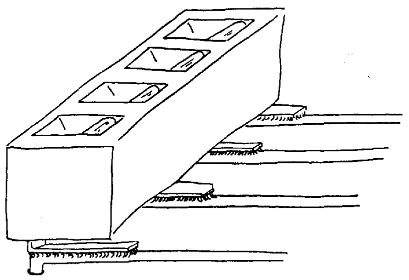


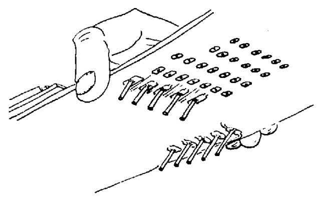
HOW THE TAPE CIRCUIT WORKS

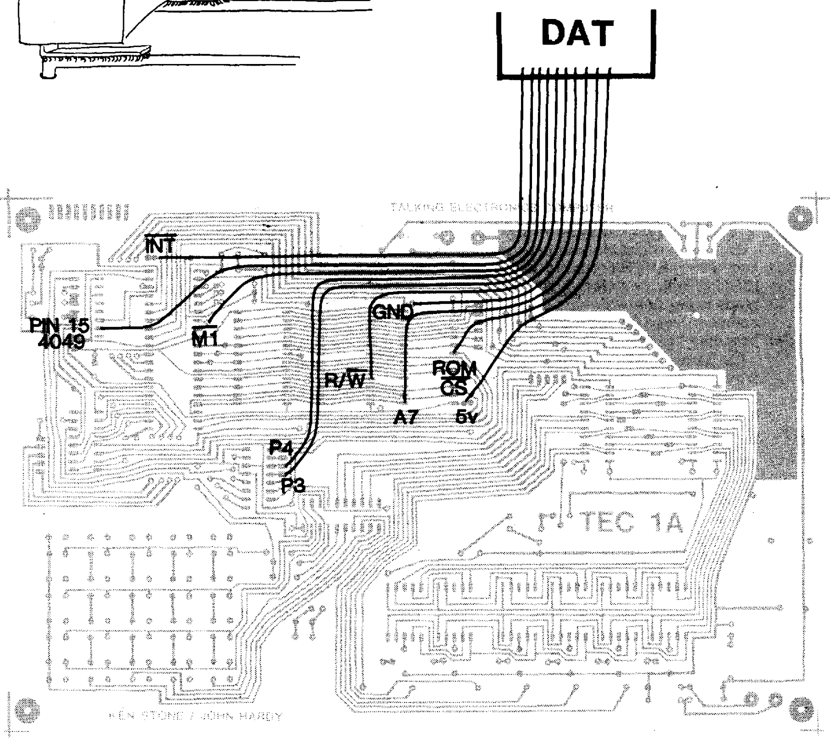
There's not much to describe about the tape circuit as all the hard work is done by software. The output section consists basically of an AC coupled LOW PASS filter with some attenuation on the end to prevent the digital level voltage from over driving the cassette players input.
The input section is just a simple AC coupled common emitter transistor amplifier with the base heavily biased on. The bias on the transistor is important as this ensures that the software is able to read a steady logic 0 when no (AC) input is present.

HOW THE SINGLE STEPPER INTERFACE WORKS

The single stepper INTERFACE works by interrupting the Z80 after each instruction. The interrupts are generated from a D flip flop on the DAT BOARD. Each time the Z80 fetches the first byte of an instruction a special signal called M1 is generated. This M1 is used to clock the ROM CS line into the D flip flop. The Q-bar output of the flip flop is connected to the INTerrupt pin. This means that an interrupt will be requested on every instruction fetch unless the instruction was fetched from the MONitor ROM.
It is important to prevent interrupts while executing in the MONitor ROM. If we don't, then an interrupt will occur just after it is re-enabled, at the end of the stepper routine. Immediately following the EI (enable interrupt), is a RETurn. If an interrupt occurs on this RETurn, then the stepper routine is re-invoked and each time this RETurn is reached, the program loops back to the stepper routine forever!! (If it wasn't for this problem we would not require any external hardware at all). ●







TOP RIGHT
Bottom side of the DAT board with the feed-downs fitted.

TOP LEFT
Diagram showing how the cut-up IC socket is mounted on the links.

LEFT
Wiring diagram showing where the "flying leads" from the underside of the DAT board are connected to the TEC.
Note that the diagram DOES NOT show the wires leaving the DAT board in the correct order, only the correct places on the TEC board. Use the labels on the underside of the DAT board for the correct DAT wiring positions.

THE LIQUID CRYSTAL DISPLAY

by Jim.

INSIDE THE DISPLAY

The display has three internal registers though which all communication is done. These are the registers:

THE DATA REGISTER

The data register is a read or write register to which all DISPLAY DATA (in ASCII format) and BIT MAPPED PROGRAMMABLE CHARACTERS are sent. This DATA register acts as a TEMPORARY BUFFER between the internal DISPLAY RAM or CHARACTER GENERATOR RAM (both described below) and the host computer (our TEC).

Characters may also be read from this register.

Internal operations transfer data between this register and the internal RAM (or between RAM and this register). This register is located on port 84H.

THE INSTRUCTION (or CONTROL) REGISTER

The instruction register receives all instruction bytes. ALL bytes sent to this register will be interpreted as CONTROL by the LCD. This is a WRITE ONLY register and is decoded on port 04.

THE ADDRESS COUNTER/BUSY FLAG

Bit 7 of this register is used as the busy flag. After EVERY operation it goes HIGH to indicate that the display is not ready to perform ANY type of additional operation yet. As soon as the display becomes "on line" again, it will go LOW.

The lower 7 bits are the current address of the internal cursor. All read or write operations occur between the data register and the address held in this register.

This register is READ ONLY. (The ADDRESS COUNTER is set or altered by instructions sent to the INSTRUCTION REGISTER and then transferred into THE ADDRESS REGISTER by an INTERNAL operation). This register is located on port 04 with the control register. Internal decoding gates the R/W line to select between each register.

As well as the registers, the display contains both RAM and ROM. Below is a description of the internal memory inside the LCD.

THE DISPLAY RAM

All the display information sent to the DATA REGISTER is transferred into the DISPLAY RAM by an internal operation. This RAM can hold 80 bytes of display information. While the LCD may only display 32 characters at a time, the extra bytes allow for the display to be shifted or can serve as general purpose storage RAM. An unusual feature of the display RAM is that the address from the last location on the top line (27H) to the address on the bottom line (40H) IS NOT CONSECUTIVE.

THE CHARACTER GENERATOR ROM

This ROM contains 192 different 5x7 dot matrix characters. These include full upper and lower case Alphabet characters, numbers, maths symbols, Greek and Japanese characters.

All of the most used characters are here. Any type of character we need that is not there, can be made up on the CHARACTER GENERATOR RAM.

THE CHARACTER GENERATOR RAM

The CHARACTER GENERATOR RAM allows us to define up to 8 different characters of our choice. The format of each is a 5x8 dot matrix with the cursor making up the 8th row. Any or all can be displayed together on different parts of the LCD and also may appear in several places at once. We can use this to make games characters.

GETTING SOMETHING ON THE LCD

Using the LCD is easy because it contains its own "intelligent" chips which do all the hard work for us. From JMON, putting anything on the LCD is VERY easy because the LCD has been set-up by JMON.

JMON sets the LCD to shift the cursor right after each entry. You cannot see the cursor as it has been turned off by the software in JMON.

To aid with the experiments below, put FF at 0821 (the LCD will stop changing after the first F) and AA at 08FF. These disable the LCD from the MONitor (the FF at 0821) and stop the MONitor rebooting its variables on a reset (the AA at 08FF). The MONitor will reset to 0A00 to remind you that the variables have not be re- booted on reset. (Unless a key was held down while reset was pushed, in which case you must again put FF at 0821 and AA at 08FF).

Ok, lets start by putting the letter L on the screen.

Firstly we must clear the screen and send the cursor home. This may be done by one instruction - 01. We output this to the control register on port 04. Before we can output to the LCD we must wait until it is ready. Because this is required to be done frequently, the RST 30 instruction has been used to do this for us. The RST 30 reads the LCD busy flag and loops until it goes LOW.

Ok lets type this in:

```
0A00  F7      RST 30
0A01  3E 01   LD A,01
0A03  D3 04   OUT (04),A
0A05  76      HALT
```

Reset, Go

The display will go blank and the (invisible) cursor will return to home (top left-hand corner). The 01 instruction sets all the display RAM locations to 20H (space). The 01 instruction doesn't affect any previous mode setting or display options (discussed below).

Now enter this with the RST over the HALT at 0A05:

```
0A05  F7      RST 30
0A06  3E 4C   LD A,4C "(L)"
0A08  D3 84   OUT (84),A
0A0A  76      HALT
```

Reset, Go

The letter L appears in the top left corner.

Ok, now as before, put this in with the RST over the HALT:

```
0A0A  F7      RST 30
0A0B  3E 43   LD A,43 "(C)"
0A0D  D3 84   OUT (84),A
0A0F  F7      RST 30
0A10  3E 44   LD A,44 "(D)"
0A12  D3 84   OUT (84),A
0A14  76      HALT
```

Reset, Go

The above section outputs two more bytes to the DATA REGISTER.

Until now we have just been using a simple method to output data. This has shown us the basic way to talk to the LCD. Now that we have come this far and learned the basics, we'll advance to something more useful.

The code below will output a word onto the bottom line of the LCD. The display DATA will be held in a table at 0B00.

```
0A14  F7        RST 30
0A15  3E C0     LD A,C0
0A17  D3 04     OUT (04),A
0A19  01 84 06  LD BC,0684
0A1C  21 00 0B  LD HL,0B00
0A1F  F7        RST 30
0A20  ED A3     OUTI
0A22  20 FB     JRNZ 0A1F
0A24  76        HALT

0B00 4D 41 53 54 45 52
```

To set the cursor to the bottom line we output 80 to the instruction register (bit 7 sets the cursor address entry) + 40 (which is the actual address of bottom left display) = C0.

The OUTI instruction is new to our repertoire. It's operation is to output the byte addressed by HL to the port addressed by C. HL is then incremented and B is decremented. If B becomes ZERO the ZERO FLAG is set and the operation is complete. This instruction can output up to 256 bytes at a time.

Because we need to check the busy flag we loop back to the RST 30 until all the bytes have been done. If we didn't need to check the busy flag we could have used the OTIR instruction which automatically repeats itself until B=0.

All the above is done with the cursor switched off. For the next section we want to have the cursor on. To switch on the cursor output 0E to the instruction register on port 04.

```
0A00  F7        RST 30
0A01  3E 0E     LD A,0E
0A03  D3 04     OUT (04),A
0A05  76        HALT
0A06  C7        RST 00
```

Now let's see what does what on the display.

Using the above routine, output the bytes below one at a time, to port 04 and HALT between each. (leave what's on the display there).

Check the function of each on the table of controls.

```
18 1C 1C 1C 02 14
14 10 0C 0F 08 0C
```

Good luck!!

SETTING THE ENTRY MODE

The display may be configured to perform several different functions UPON EACH DATA BYTE ENTRY. They are:

1 INCREMENT CURSOR ADDRESS after storing inputted data byte (06H). This is our normal mode.

2 DECREMENT CURSOR ADDRESS after storing input (04).

3 SHIFT THE DISPLAY RIGHT after entry (05).

4 SHIFT THE DISPLAY LEFT after entry (07).

Each mode is selected by outputting the byte shown to port 04.

Once the entry mode is set it IS ONLY CHANGED BY ANOTHER ENTRY MODE SET COMMAND. None of the other control bytes will alter the entry mode.

The shift on entry feature (05,07) has been found to be difficult to use and even appears to contain design bugs.

You may experiment with it but we won't be using it in these notes.

The CURSOR DECREMENT may come in handy sometimes but it's more likely to be useful to processors which move blocks of data around in a more limited way to the Z80.

RUNNING WORDS ON THE LCD

Running words along the LCD is also simple because the LCD'S intelligent chips do most the work for us again. Our job is to enter the words we want to scroll (up to 16 characters per line for this routine) and send shift commands each time we want a shift.

The routine below is entered in 3 sections. Each section is a logical progression and increases the programs abilities. You can look at the instructions in each section and compare it to what the section does. This way you can learn how to put blocks together to use the display any way you want. Before entering the code below put FF at 0821 and AA at 08FF as described before.

Enter this and INCLUDE the NOPS and the table at 0B00 then run it:

```
0A00  3E 01     LD A,01
0A02  D3 04     OUT (04),A
0A04  F7        RST 30
0A05  3E 06     LD A,06
0A07  D3 04     OUT (04),A
0A09  F7        RST 30
0A0A  3E 0C     LD A,0C
0A0C  D3 04     OUT (04),A
0A0E  F7        RST 30
0A0F  00        NOP
0A10  00        NOP
0A11  00        NOP
0A12  00        NOP
0A13  00        NOP
0A14  01 84 10  LD BC,1084
0A17  21 00 0B  LD HL,0B00
0A1A  F7        RST 30
0A1C  ED A3     OUTI
0A1D  20 FB     JRNZ 0A1A
0A1F  F7        RST 30
0A20  3E C0     LD A,C0
0A22  D3 04     OUT (04),A
0A24  F7        RST 30
0A25  21 30 0B  LD HL,0B30
0A28  06 10     LD B,10
0A2A  F7        RST 30
0A2B  ED A3     OUTI
0A2D  20 FB     JRNZ 0A2A
0A2F  76        HALT

0B00: 54 41 4C 4B 49 4E 47 20
0B08: 20 20 20 20 20 20 20 20
        (TALKING)
0B30: 45 4C 45 43 54 52 4F 4E
0B38: 49 43 53 20 20 20 20 20
        (ELECTRONICS)
```

This will put "TALKING" on the top line and "ELECTRONICS" on the bottom line of the LCD and stop. Study the above section and see if you can work out the role of each instruction.

Now we'll add the shift section. Enter this with the first "NOP" over the last "HALT" and run it:

```
0A2F  00        NOP
0A30  00        NOP
0A31  3E 18     LD A,18
0A33  D3 04     OUT (04),A
0A35  01 00 60  LD BC,6000
0A38  0B        DEC BC
0A39  78        LD A,B
0A3A  B1        OR C
0A3B  20 FB     JRNZ 0A38
0A3D  18 F2     JR 0A31
```

The above code loads the shift instruction (18H) into the accumulator and outputs it to the control register on port 04.

As you can see it shifts the display, but this method is not very good if we want to shift only a few characters as we must wait for them to be shifted through the entire display RAM before they re-appear. To overcome this we can count the number of shifts and reset the display with a 02 command, as soon as all the letters have been shifted outside the display. The 02 instruction resets the display from shift WITHOUT CHANGING the contents of the DISPLAY RAM, CHARACTER GENERATOR RAM, or the CONTROL MODE. Because we would like the words to shift across the entire display and re-appear as soon as they have all gone, we must load the words just outside the screen to the right. The following additions make the words start shifting into the display from right-to-left.

Ok, Now enter the following, AT THE ADDRESSES SHOWN:

```
0A0F  3E 90     LD A,90
0A11  D3 04     OUT (04),A
0A13  F7        RST 30
-----------------------
0A22  3E D0     LD A,D0
```

The above instructions set the DISPLAY RAM ADDRESSES to the RAM locations just right of the screen. The address of the top line is 90 and the address of the bottom line is D0. (Actually these are the addresses + 80H, the SET ADDRESS instruction).

```
0A2F  16 1B       LD D, 1B
```

(The D register is our shift counter).

```
0A3D  00          NOP
0A3E  00          NOP
0A3F  15          DEC D
0A40  20 EF       JRNZ 0A31
0A42  3E 02       LD A,02
0A44  D3 04       OUT (04),A
0A46  F7          RST 30
0A47  18 E6       JR 0A2F
```

The last group makes up the shift counter and resets the display when the counter reaches Zero. When the 02 command is received by the LCD the display is returned to its NORMAL position. This means that the inputted data is returned to WHERE IT WAS ENTERED (just right of the screen). Now, when the next shift command is received, the letters start to shift left back on to the screen.

QUESTION:
Why don't we need to wait for the BUSY flag to go low after the shift instruction?

If you wish to change the number of characters to be shifted, you may do so by putting your new characters at 0B00 for the top line +and at 0B30 for the bottom line. Unused locations should have 20 (space) inserted until 16 locations are filled. (From 0B00 to 0B10 and from 0B30 to 0B40). The value of the loop counter loaded into D at 0A2F should also be changed. The value of the loop counter is best set to 10H + the number of letters occurring in the longest line.
e.g. For the the example above:
ELECTRONICS = 11 (0BH) Letters.

So add 0BH + 10H = 1BH.
So 1BH is loaded into D at 0A2F.
To understand the above formula better, try 1C and 1A and see the result.

FINAL NOTES
The slow response of the LCD detracts from the effectiveness of the shifting a little but by experimenting with the delay at 0A35 you should be able to get a good compromise between speed and display clarity.

The above shifting method is just one of dozens of ways we could have used. A more complex program could shift information across and out one end and load new information in the other to create a running information display.

Use the blocks in this program and the others to make up your own display routines. If you come up with something interesting, write in. We would love to see what you've come up with.

DESIGNING YOUR OWN CHARACTERS

You can have up to eight different characters stored in a character-generator RAM. Each character is displayed on the screen when it is addressed in the display RAM. The addresses are between 0-7. The user-defined characters are made up of an 8x8 matrix (only 5 columns are displayed, the cursor makes up the 8th row.)

To set up a character, 8 bytes are outputted to the character- generator RAM. The first byte makes up the top row (only the 5 lower bits are displayed). The second byte makes up the second row etc.

Before sending the 8x8 character (actually a 5x8 character), the entry mode must be set (if not already) to address-increment with no display shift (06) and a set character RAM address operation must be done.

The control byte for this is 40 + the address of the first byte of each character-matrix. E.g: 40, 48, 50 for characters 1, 2, 3 etc.

Once a character is set up, it is displayed by placing its address in the DISPLAY DATA RAM. Before doing this the DISPLAY RAM must be selected via 80 + address.

OK, let's put our own character on the LCD.

```
0A00  F7          RST 30
0A01  3E 01       LD A,01
0A03  D3 04       OUT (04),A
0A05  F7          RST 30
0A06  3E 40       LD A,40
0A08  D3 04       OUT (04),A
0A0A  01 84 08    LD BC,0884
0A0D  21 00 0B    LD HL,0B00
0A10  F7          RST 30
0A11  ED A3       OUTI
0A13  20 FB       JRNZ 0A10
0A15  F7          RST 30
0A16  3E 80       LD A,80
0A18  D3 04       OUT (04),A
0A1A  F7          RST 30
0A1B  3E,00       LD A,00
0A1D  D3 84       OUT (84),A
0A1F  76          HALT

0B00:

11, 0A, 04, 11, 0A, 04, 11, 0A
```

Experiment with the values in the table and see how it all goes together. By increasing the value loaded into B, to 10(hex) (at 0A0C) a second character may be programmed at the same time. The table for the second character will start at 0B08. This will be displayed when a 01 is written into the DATA DISPLAY REGISTER. Experiment and see if you can get 8 characters appearing in several places at once on the display.

MYSTERY EFFECT

The routine below produces a very interesting effect. It uses the PROGRAMMABLE CHARACTER GENERATOR to produce 8 different characters some of which are displayed several times. We won't tell you the effect, we'll let you type it in and see for yourself. You won't be disappointed!

The program consolidates much of what we have learned about "driving" the LCD. If you experiment further and add a shift to it then it will be a complete revision of what we have covered in these pages.

Now that you know how to use the LCD, start writing some programs that use it. If you come up with something interesting don't hesitate to send it in to TE. We would be very interested in some simple animation or an adventure game or anything that others would be interested in seeing. Go to it!

```
0A00  F7          RST 30
0A01  3E 01       LD A,01
0A03  D3 04       OUT (04),A
0A05  F7          RST 30
0A07  3E 06       LD A,06
0A08  D3 04       OUT 04
0A0A  21 00 0B    LD HL,0B00
0A0D  01 84 10    LD BC,1084
0A10  F7          RST 30
0A11  ED A3       OUTI
0A13  20 FB       JRNZ 0A10
0A15  F7          RST 30
0A16  3E 40       LD A,40
0A18  D3 04       OUT (04),A
0A1A  21 20 0B    LD HL,0B20
0A1D  06 40       LD B,40
0A1F  F7          RST 30
0A20  ED A3       OUTI
0A22  20 FB       JRNZ 0A1F
0A24  F7          RST 30
0A25  3E C0       LD A,C0
0A27  D3 04       OUT (04),A
0A29  21 10 0B    LD HL,0B10
0A2C  06 10       LD B,10
0A3E  F7          RST 30
0A3F  ED A3       OUTI
0A31  20 FB       JRNZ 0A3E
0A33  76          HALT

0B00: 20 4D 49 52 52 4F 52 20
0B08: 49 4D 41 47 45 21 20 20
0B10: 20 00 01 02 02 03 02 20
0B18: 01 00 04 05 06 07 20 20
0B20: 00 11 11 11 15 15 1B 11
0B28: 00 0E 04 04 04 04 04 0E
0B30: 00 11 12 14 1E 11 11 1E
0B38: 00 0E 11 11 11 11 11 0E
0B40: 00 11 11 1F 11 11 11 0E
0B48: 00 0F 11 11 17 10 11 0E
0B50: 00 1F 10 10 1E 10 10 1F
0B58: 00 04 00 00 04 04 04 04
```

CONCLUSION

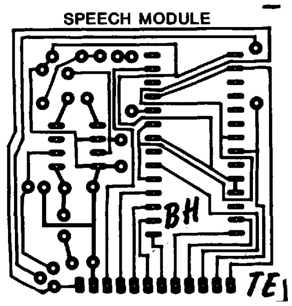
This concludes this issues instalment on the LCD.
Study the previous notes carefully and get to know the LCD fully. There is enough information here for you to write routines using the LCD and we would like to see some ideas sent to us for issue 16.
The LCD will be supported further in issue 16 and if all goes well, we will have a cheap, full alpha-numeric keyboard with supporting software. I am working towards the stage were you can anotate your routines and send the text and the routine in on tape. We can then load them into our desk top publisher.
Don't forget if you have any good ideas or questions about the TEC, send them in to "TEC TALK."

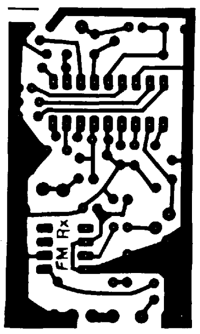
Below is the table of LCD control bytes. Use these in conjunction with the previous notes

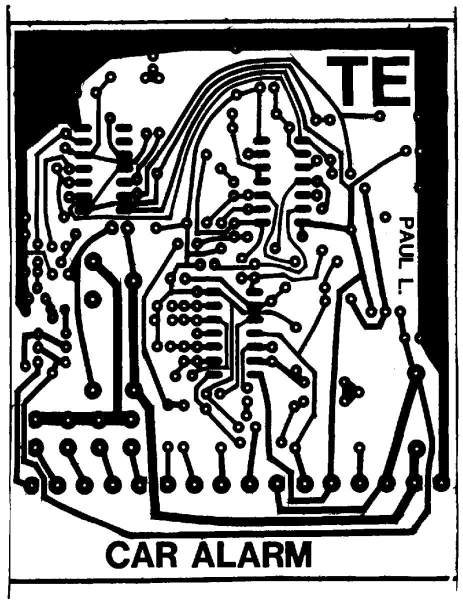
Instruction	Code										Function	Execution time
	RS	$R/\overline{W}$	DB_7	DB_6	DB_5	DB_4	DB_3	DB_2	DB_1	DB_0		
(1) Display clear	0	0	0	0	0	0	0	0	0	1	Clears all display and returns cursor to home position (address 0)	1.64 ms
(2) Cursor Home	0	0	0	0	0	0	0	0	1	*	Returns cursor to home position. Shifted display returns to home position and DD RAM contents do not change.	1.64 ms
(3) Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets direction of cursor movement and whether display will be shifted when data is written or read	40 µs
(4) Display ON / OFF control	0	0	0	0	0	0	1	D	C	B	Turns ON/OFF total display (D) and cursor (C), and makes cursor position column start blinking (B)	40 µs
(5) Cursor/Display Shift	0	0	0	0	0	1	S/C	R/L	*	*	Moves cursor and shifts display without changing DD RAM contents	40 µs
(6) Function Set	0	0	0	0	1	DL	1	*	*	*	Sets interface data length (DL)	40 µs
(7) CG RAM Address Set	0	0	0	1	A_{CG}						Sets CG RAM address to start transmitting or receiving CG RAM data	40 µs
(8) DD RAM Address Set	0	0	1	A_{DD}							Sets DD RAM address to start transmitting or receiving DD RAM data	40 µs
(9) BF/Address Read	0	1	BF	AC							Reads BF indicating module in internal operation and AC contents (used for both CG RAM and DD RAM)	0 µs
(10) Data Write to CG RAM or DD RAM	1	0	Write Data								Writes data into DD RAM or CG RAM	40 µs
(11) Data Read from CG RAM or DD RAM	1	1	Read Data								Reads data from DD RAM or CG RAM	40 µs

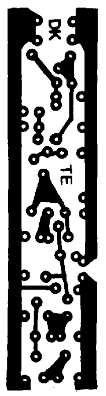
* : Invalid bit	I/D = 1 : Increment	C = 1 : Cursor ON	R/L = 1 : Right shift
A_{CG} : CG RAM address	I/D = 0 : Decrement	C = 0 : Cursor OFF	R/L = 0 : Left shift
A_{DD} : DD RAM address			
	S = 1 : Display shift	B = 1 : Blink ON	DL = 1 : 8 bits
	S = 0 : No display shift	B = 0 : Blink OFF	DL = 0 : 4 bits
	D = 1 : Display ON	S/C = 1 : Display shift	BF = 1 : Internal operation in progress
	D = 0 : Display OFF	S/C = 0 : Cursor movement	BF = 0 : Instruction can be accepted

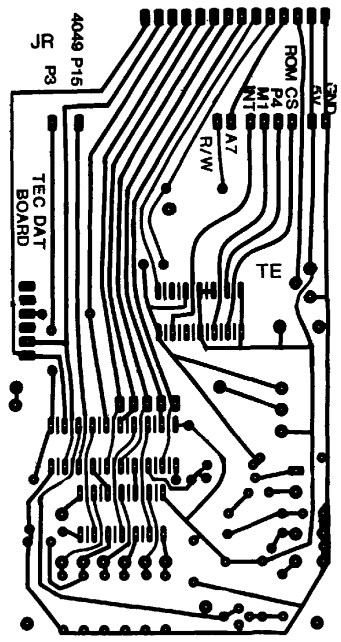
PC ARTWORK FOR THIS ISSUE

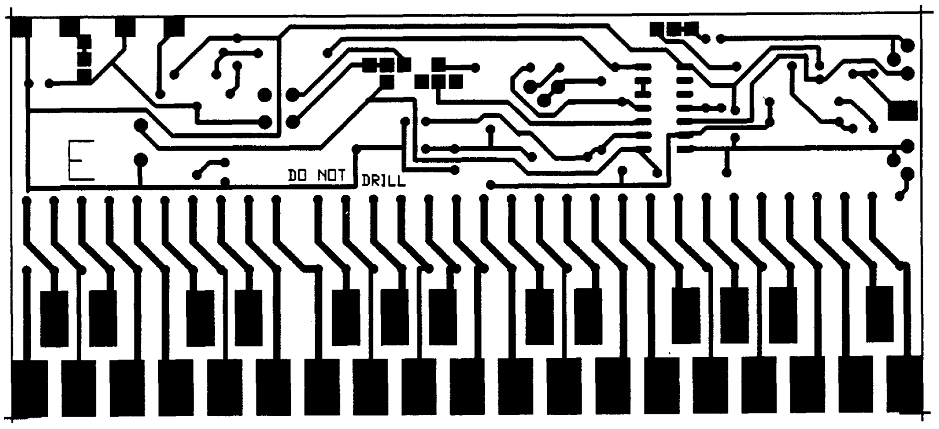






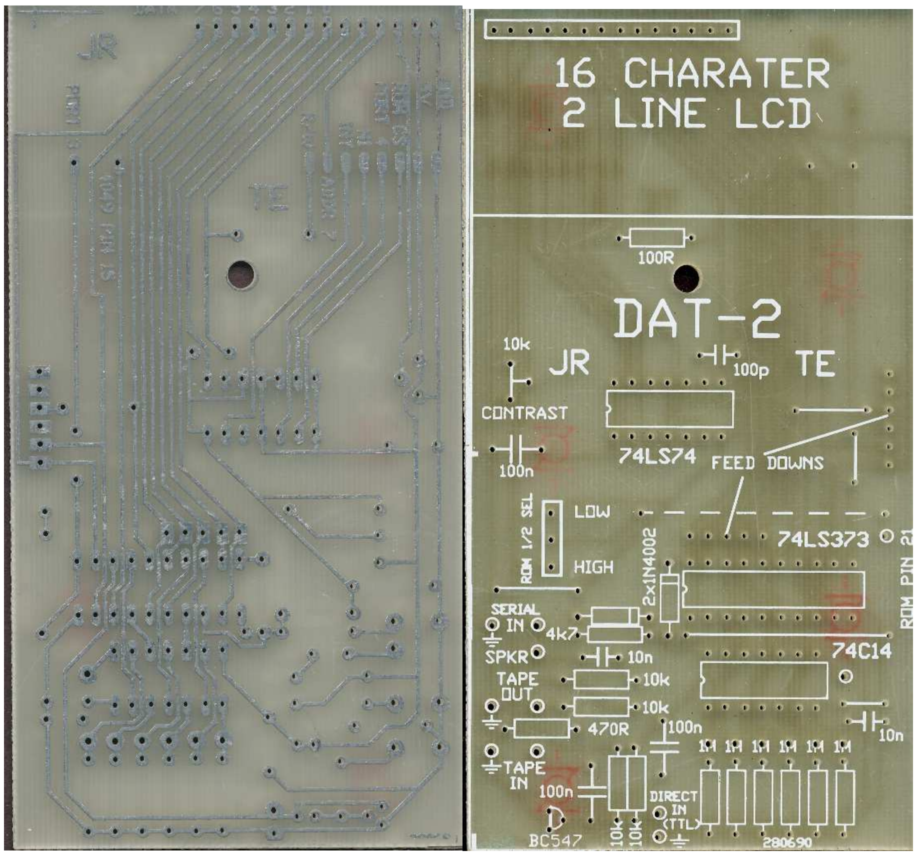






ASCII DISPLAY TABLE FOR LCD

Upper bit 4 bit / Lower bit 4 bit	0000	0010	0011	0100	0101	0110	0111	1010	1011	1100	1101	1110	1111
× × × ×0000	CG RAM (1)		0	@	P	`	p		ー	タ	ミ	α	p
× × × ×0001	(2)	!	1	A	Q	a	q	。	ア	チ	ム	ä	q
× × × ×0010	(3)	"	2	B	R	b	r	「	イ	ツ	メ	β	θ
× × × ×0011	(4)	#	3	C	S	c	s	」	ウ	テ	モ	ε	∞
× × × ×0100	(5)	$	4	D	T	d	t	、	エ	ト	ヤ	μ	Ω
× × × ×0101	(6)	%	5	E	U	e	u	・	オ	ナ	ユ	σ	ü
× × × ×0110	(7)	&	6	F	V	f	v	ヲ	カ	ニ	ヨ	ρ	Σ
× × × ×0111	(8)	'	7	G	W	g	w	ァ	キ	ヌ	ラ	g	π
× × × ×1000	(1)	(	8	H	X	h	x	ィ	ク	ネ	リ	√	x̄
× × × ×1001	(2)	)	9	I	Y	i	y	ゥ	ケ	ノ	ル	⁻¹	y
× × × ×1010	(3)	*	:	J	Z	j	z	ェ	コ	ハ	レ	j	千
× × × ×1011	(4)	+	;	K	[	k	{	ォ	サ	ヒ	ロ	ˣ	万
× × × ×1100	(5)	,	<	L	¥	l	|	ャ	シ	フ	ワ	¢	円
× × × ×1101	(6)	-	=	M	]	m	}	ュ	ス	ヘ	ン	£	÷
× × × ×1110	(7)	.	>	N	^	n	→	ョ	セ	ホ	゛	ñ	
× × × ×1111	(8)	/	?	O	_	o	←	ッ	ソ	マ	゜	ö	█


16 CHARATER
2 LINE LCD
100R
DAT-2
10k
JR
100p
TE
CONTRAST
100n
74LS74
FEED DOWNS
LOW
ROM 1/2 SEL
HIGH
2x1N4002
74LS373
ROM PIN 21
SERIAL IN
4k7
SPKR
10n
74C14
TAPE OUT
10k
10k
470R
100n
10n
1M 1M 1M 1M 1M 1M
TAPE IN
100n
DIRECT IN (TTL)
BC547
10k
10k
280690