The Challenger Line...

OSI tems

tems

1P Series

TC

OSI-tems, Volume II, Number 7. August 1, 1980

New York OSI Users Group

Table of Contents

OSI Users Group

Meets the first Thursday of every month at
Polk's/Aristocraft
314 Fifth Avenuue
New York, NY 10001
212-279-9034

David Gillet	President
Warren Modell	Vice President
Daniel Schwartz	Treasurer
Thomas Cheng	Secretary

Dues : $10.00 Standard Membership
$ 5.00 Student Membership

OSI-tems Staff

Larry Thaler
Thomas Cheng
Salomon Lederman
Michael Bassman
Terry Terrance

This month's Editor Thomas Cheng

Editorial

Congratulations. In this issue of OSI-tems, I believe that we've finally got the message through to a large number of members - contribute!

Finally, we have (I hope) a magazine which will satisfy (or at least whet the appetite) of both the C1 and C2/C4 user, with articles concerning both machines. A balance has been struck - at long last members are working on both hardware and software. Enough back patting. We want more!

We want every member to contribute. Get those articles in!

Letters:

Peter Schreiber
1609 Washington Avenue
Seaford, NY 11783

July Issue, Page 10 "Print At"

The program does not allow the user to have a message using BASIC tokens. Making the change:

Location	Current	New
022A	29	22

and using the format:

```
X=USR(X)"Message"
```

the program will recognize each character of the message and not BASIC tokens.

Corrections to our last issue:

Ugo V. Re
Levittown, NY

The article and programs for a Serial Interface Test (OSI-tems Vol. II, Number 6, page 30) is for a C2/C4 system with the serial port at $FC00.

A C1P has it's serial port at $F000. The following corretions must be made in the two programs to have them work on a C1P.

Change FC to F0 at the following addresses:

022F, 0237, 023A, 0266, 026F, 0284, 028A

General

Wow - we really packed them in at the last meeting. Lots of new faces. I hope we see you again and again and again!!

We're new at this and the meeting are improving. Perhaps you can offer suggestionF. Drop a note to our President David Gillett.

I'm suggesting making up an agenda and having a short meeting of perhaps 15 minutes a couple of days prior to the meeting. The Panel should consist of the officers and the Editor of the month.

Mr. Polk's suggestion of "a speaker" at each meeting was excellent. Most of our members are certainly well qualified and knowledgeable. We could also invite individuals from other user groups - not necessarily OSI. Perhaps we could invite computer educators, suppliers, publishers and writers, a stripper, or two.

Of course, we would have to rearrange some of the cartons in the basement so that we culd put out more chairs and make it look more like a meeting room. Some of the Early Bird Members could do this and the Late Ones can put the chairs back. How's that For Fair!!

One other item. If a page(s) from the bound copy of OSI-tems falls out, it's because the binder was a 15 page cover and was used for 20 pages of material.

If OSI-tems can be kept at 15 pages, I can supply the covers free as I have about 100 of them. If you expect it to be larger than 15 pages, I can get the larger sizes for about 40 cents apiece in quantities of 25. If you prefer to bind a year at a time, the prices for the larger binders are under a dollar, maybe 75 cents. I'll check and let you know. They also come in packages of 25.

Warren Modell
3133 Rochambeau Avenue
Bronx, NY 10467
014-3773

Poking 9791 for a C1 or 9761 for a C4 disk system will allow you to control the amount of screen scroll at the top - This would create a 'protected' area of screen memory. As always, the numbers are stored in 6502 format lo byte, hi byte (D1,D2 hex).

contributed by Hal Pollenz

Poking the ACIA flag 61440,3 then 61440,16 will allow a SAVE at 4800 baud. Unfortunately, we can't use this for tapes. However, it is handy for a fast peripheral. This ACIA is only for C1's - someone will have to look it up for C4's.

I Upped My Memory - Up Yours

by William J. Balaban

One of the most inevitable laws of compuuter programming states: "Any given program will expand to fill all available memory." By using the often ignored trick of 'piggybacking memory', any Superboard II or C1P can have it's memory literally doubled (16K). However, if you are still paying back the money you borrowed to buy the computer in the first place (next time, beware of any loan operation where the district manager is named Big Louie), expansion can be done in 1K increments. Also, should an expansion board with more memory be added later, the piggybacked memory can be relocated to a new area in memory (another 8K block) by moving one wire. It could be also be located so as to leave an 8K gap between it and existing memory; it would be noncontinuous memory, good for machine language programs since they would not get wiped out by BASIC (Oops, did I forget to use warm start again?). This added memory can be had for the price of however many 2114 memory chips you want (2 for every 1K), the cost of an extra 74LS138 decoder chip and the investment of a few nights of tedious labor (or wild excitement, if your sense of humor is as demented as mine is). Should you actually have the guts to make the attempt, read this article several times (if it does nothing else, at least I'll feel good in knowing that it is so well read) in order that you may understand everything that it entails.

The basic concept is simple. Since the +5 volt line, the ground line, the data lines and the address lines are all tied to each other respectively, from chip to chip, soldering extra memory chips to existing memory (pin for pin) is perfectly feasible. The only pin I have not mentioned is the chip enable (CE). This pin is the only means of giving the memory chip a unique location on the memory map, so that one will be discernible from another by the computer. OSI uses a 74LS138 decoder chip to select whatever memory chip pair the computer needs, while deselecting all others. By not soldering the new chips to the CE pin, a wire can be run from it to a new '138' decoder which will be set up to select and deselect it at the proper times. Soldering the upper CE pin to the lower CE pin will cause both chips to cover the same memory location; while this is an example of adding piggyback memory, it is not really anything to boast about.

When selecting memory chips, it is important to consider both speed and power (or is that what must be considered when buying a sports car?). The new 2114 chips must have an access time of 55 nanoseconds or less, as specified by OSI. They must also be the low power version in order to prevent overloading your power supply, as well as to minimize the heat build up.

The handling of the memory chips is a potential hazard point. The chips should come packed in an antistatic container or with their leads stuck into conductive foam or aluminum foil (any distributor who wraps them in old newspapers should be avoided). They should remain this way until needed. When working with them, static damage can be prevented by keeping the pins at the same potential: this can be accomplished by keeping wire or solder wrapped around the top edges of the pins and removing this only after the chip has been safely installed.

Once you have thrown caution to the winds and have bought the memory, you can either take the dealer's word that they will work or you can test them yourself. Your computer can do a quickie test by performing the BASIC memory scan. Simply remove the highest 1K of memory (U38 and U52) and replace them with the new chips (observe static precautions and be sure that the machine is unplugged, now and during all other work). Apply power and cold start BASIC; if the available memory goes down - you have a bad chip, if it remains the same - consider these to be good, if it goes up - apply for a patent! If you think that you have a dud chip, check to see if all the pins have gone into the socket; one may have slid outside or may have been bent underneath.

If by some miracle, all memory has tested good, you can proceed to the installation part of this project. The piggyback soldering can be done with the lower chips on the board (if you have the dexterity of a microsurgeon) or the lower chips may be stuck onto a wad of aluminum, so that the work may be done on any flat, uncluttered surface. All soldering should be done with a low wattage (10W to 30W) iron. Gently bend the CE pin of the upper chip to a 60 or 90 degree angle from the other pins. Next, try placing the upper chip onto the lower one (Note: a dot usually appears in the lower left hand corner to indicate the location of pin one or there will be a notch on one of the narrow ends to indicated the left side. Be certain to align them properly, both here and when installing them in the computer.) so that the pins of the upper slide over the pins of the lower and hold both chips snugly together. Several of the pins may require a slight bending to achieve this. Once accomplished (all upper pins, except 8, now in contact with the lowers), heat one of the pin pairs while feeding solder into the joint. Use only enough to achieve a firm bond; if you cannot see the chips through the solder, you have overdone it a bit. Proceed pin by pin around the chip, waiting 10 to 30 second per pin to allow the chips to cool. When finished, examine all bonds; make sure each one is firm, clean and not shorting to an adjacent pin.

In keeping with OSI's layout, install the first pair of double chips into sockets U31 and U45, the next pair will go into U32 and U46 and so on down the line (see OSI's board layout). By coming this far without resorting to a diet consisting solely of gin and tonic, you have done extremely well.

Next, install the 74LS138 into one of the open spots in the kludge area on the computer board. Orient it as the chips around it are; the notched or dotted end will point toward the keyboard and pin 1 will be on the lower right (the previous orientation description referred to a lengthwise view of the chip, don't be confused with this). Solder it into place.

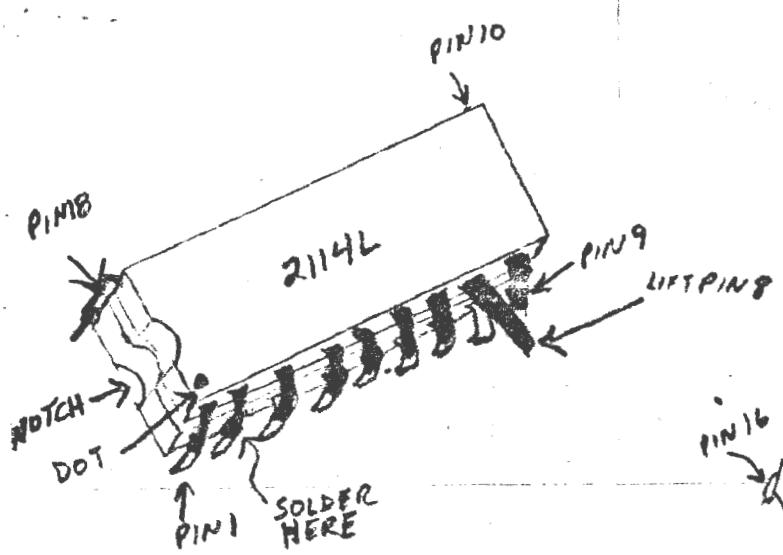
Wiring should be done with a stiff, thin, single stranded, insulated wire. Wiring from the '138' to the memory is best routed on top of the board. Precut the wire to the length required and bare both ends. Insert one end into the convenient hole next to the appropriate pin of the '138' and solder it. Loop the other end around the CE Pin of the nearest chip of the memory chip pair you are working with. Solder it. Precut and bare the ends of another wire, run it from this CE pin to that of the other chip of the pair and solder it. All other wiring is best done below the board. Insert the end of the precut wire into the appropriate hole from below and solder it. The other end will go to a fully soldered-in chip, so make a small loop or hook in this end (just large enough to slip over the tip of the pin as it protrudes through the board). Slip the loop over the pin and solder it in place. You can hold them in place with a few drops of glue.

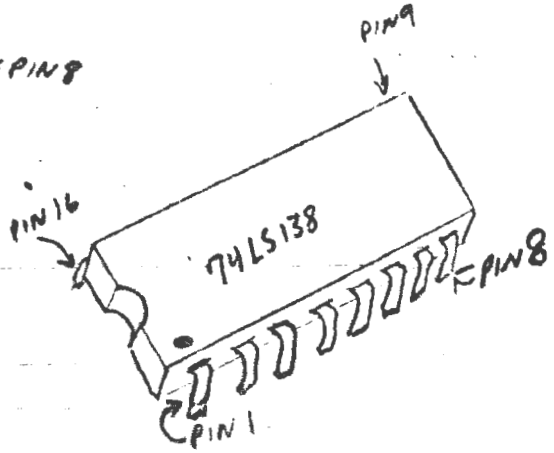
Using the suggestions for wiring above, wire pins 1,2,3,8 and 16 of the new '138' to pins 1,2,3,8 and 16 respectively of U20. Pin 5 of the new '138' goes into pin 5 of U22. Pin 4 of the '138' goes to pin 14 of U23. Next, pins 7,9,10,11,12,13,14,15 of the new '138' are wired to the new memory CE lines, as mentioned above (remember to add the wire between the CE lines of each chip pair); pin 7 of the first pair (U31 and U45), 9 to the second pair and so on. Wire pin 6 of the '138' to pin 16 of the '138'. The memory will be in the second 8K block. To move it to another 8K block, pin 4 of the '138' need only be rewired (see table on schematic) to another output pin of U23.

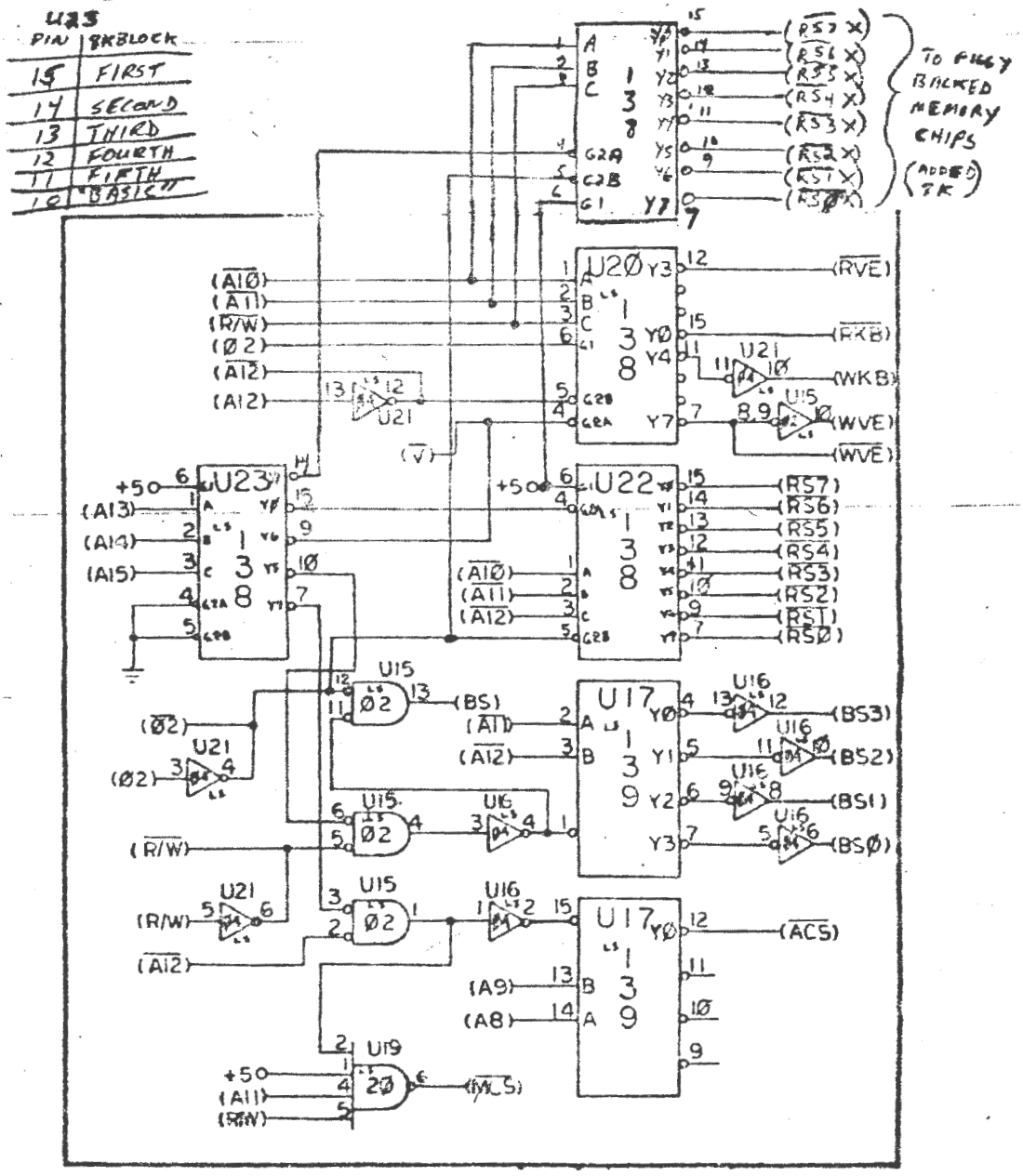
You are ready for a trial run. Prior to powering up the system:

1) Check to see that all wiring is correct.
2) See that there are no solder blobs creating shorts.
3) See that all chips are installed in the proper direction.
4) Swallow 4 strong drinks in quick succession; keep a fifth on stand by.

Power up the system. Do a BASIC cold start. Check that the usable memory equals the old memory plus the new memory minus the memory stolen by BASIC. If the memory figure is wrong or if BASIC does not cold start (if no cold start, page 1 is probably disabled), go back and check your work completely; the most trivial of mistakes can be the most elusive and the most frustrating.


PIN10
PIN8
2114L
PIN9
LIFT PIN 8
NOTCH
DOT
PIN1
SOLDER HERE


PIN9
PIN16
74LS138
PIN8
PIN1


U23
PIN | 8K BLOCK
15 | FIRST
14 | SECOND
13 | THIRD
12 | FOURTH
11 | FIFTH
10 | "BASIC"
TO FULLY BACKED MEMORY CHIPS (ADDED 7K)
U20
U21
U22
U15
U16
U17
U19
(A10)
(A11)
(R/W)
(Ø2)
(A12)
(A13)
(A14)
(A15)
(A9)
(A8)
(RVE)
(RKB)
(WKB)
(WVE)
(RS7)
(RS6)
(RS5)
(RS4)
(RS3)
(RS2)
(RS1)
(RS0)
(BS)
(BS3)
(BS2)
(BS1)
(BS0)
(ACS)

Waiting on BASIC

One feature of the Ohio Scientific Editor/Assembler which never fails to irritate me is the non-existance of a normal save routine. By this, I mean a machine language routine written from the Assembler/Editor is normally inaccessible from BASIC. If the routine lies in an area normally taken by the ASSEMBLER, the routine would have to be (shudder) transferred by hand.

One day, I had actually begun to perform such an action when it struck me that it could be performed by a virtual idiot - so I have written the following BASIC program, which takes in assembled output (from tape) in string format, then POKEs the translated values into the correct memory locations. (For further information as to the tape format, see page 4 of the OSI EXTENDED MONITER manual).

When the program is executed, it will wait for an ASSEMBLER format input (from tape), which it will store in the string array I$. The program itself will stop and execute the next section of itself if 1) the tape recorder is turned off, or 2) it receives more than 20 garbage characters after an input line.

I have done this by using a little known property of the C1P, the ability to control the cassette port by POK(E)ing and PEEKing it. Coupled with the WAIT function, the cassette port is easily controlled from BASIC.

The WAIT command has three arguments. For example, WAIT A, B, D would exclusive or (EOR) the contents of memory location A with memory location C, then AND the resultant with B. This series of operations will continue until a non-zero result is obtained. If C is omitted, it will be assumed to be zero - and EOR operation which will accomplish nothing. All of this is useless until you know where the cassette port is. On the C1P, it is memory location 61441 (decimal), and the location 61440 (decimal) is the ready flag for it, i.e., location 61440 will "tell" you when the cassette port is ready to receive (or send) a character.

My program works in the following way: it will moniter the cassette port until it receives an ASCII 59 (a ";"), which precedes a line of ASSEMBLER output. Upon receiving the semicolon, it will INPUT a line from the cassette, then go back to monitering the port for another semicolon. As I said earlier, if more than 20 extraneous characters reach the port after the end of a line, the program will then jump to line 50, where it will begin to POKE numbers into memory.

Lines 120-140 contains a hexadecimal to decimal routine, which takes in the hex number in B$, then returns the decimal value in A. This routine is called from lines 60-110, which breaks each string into portions for the hex conversion routine to work on.

If the location of the machine language routine will interfere with string storage, then either reserve memory when cold-starting BASIC, or poking an appropriate value in locations 133,134. The values in these locations will determine the highest memory location which BASIC will use. The formula

(PEEK (133) + (PEEK (134) * 256)

equals the top of memory for BASIC.

Hopefully, this program will make things easier for any machine language/BASIC programmer. Now, let BASIC wait on you !

Thomas Cheng
26 Madison Street Apt. 4I
New York, New York 10038

```
OK
LIST

 5 REM THOMAS CHENG
 7 REM FORMAT CHANGER
 8 REM TAKES IN ASSEMBLER OR  MONITER FORMAT, POKES INTO MEMORY
 10 DIMI$(20):FL=61440:PO=FL+1:LOAD
 20 WAITFL,1:NUM=PEEK(PO):X=X+1:IFX>20ANDX1THEN50
 30 IFNUM<>59THEN20
 40 INPUTI$(S):X=0:X1=-1:S=S+1:GOTO20
 50 POKE515,0:FORK=1TO32:PRINT:NEXT:PRINT"Please wait."
 60 FORT=0TOS-1:IFLEFT$(I$(T),2)<>"18"ANDT=0THEN110
 70 B$=MID$(I$(T),3,4):GOSUB120:PLACE=A
 80 FORTH=7TOLEN(I$(T))-4STEP2:B$=MID$(I$(T),TH,2):GOSUB120
 90 POKEPL,A:PRINTPL;A:PL=PL+1
 100 NEXTTH
 110 NEXTT:END
 120 A=0:A$="0123456789ABCDEF":FORK=1TOLEN(B$):FORL=1TO15
 130 IFMID$(B$,K,1)=MID$(A$,L,1)THENA=A+(16↑(LEN(B$)-K)*(L-1))
 140 NEXTL:NEXTK:RETURN
OK
```

Texas Instruments Model 733 'Silent Writer' Series Terminal

By Terry Terrance

Why, you might ask, is this guy reviewing a terminal that, first, costs a couple of grand new; and second, is a discontinued model. Well, it is precisely because this terminal is discontinued that it can be found for much less that its' new price and therefore interests us. In the unending search for printers that will work with OSI machines the TI 733 could be a 'gem', provided that you can get one for the right price.

Let's look at the 733 from a hardware point of view. The TI 733 comes in three configurations: RO (receive only), KSR (keyboard send receive), ASR(?). The RO configuration has only a thermal printer. The KSR has a thermal printer and keyboard. The ASR has printer, keyboard and two cassette tape drives. Al of these units feature modular construction so they are upgradable from RO to KSR to ASR. The RO and KSR models can have a built in modem. This feature is not available on the ASR models as they seem to be intended to be directly 'hardwired' to a computer.

All models 733 have RS 232 ASCII interfaces. Do not consider TI models 732 as they have Baudot coded interfaces: unless you are willing and able to write a software driver to handle the code conversions.

The following detailed description of the 733 comes from examination of the unit that I have, a 733 KSR. I assume that it shares many common features with the other 733's because of the modular construction.

The 733 is 21.2" wide, by 19.5" deep, by 7.2" high and weighs about 45 lbs. It has the same general contours as a C1, C2-4P, or C4 machine, but is considerably larger. My unit is light and dark gray but I have seen some units in beige and brown. The fiberglass cover is hinged at the rear to the metal base plate. If you lift at the front, the cover swings up to give easy access to the interior. Inside, there is a card cage/power supply unit that appears to be one module. A fan sits between the card cage and the power supply and first draws air over the cards then out over the power supply. The power supply is truly massive, and could probably power your whole system. The power supply always seems to run cool. It seems likely that other voltages (other that +5 volts) are available here, although I have not poked around to find out.

The 80 column printer is apparently another module. It friction feeds thermal paper from a 3 1/2" roll. Print quality seems to be the standard 5x7 dot matrix, except the 'dots' are actually square. Contrast for the printhead can be adjusted by means of a screwdriver turned trimpot accessible through a marked hole in the card cage cover. Lower case is available, except that the letters are not lower case in the usual sense. Lower case on the 733 is just small capitals. This looks strange at first, but, after a while, I've come to prefer it to true lower case without descenders, as most dot-matrix mechanisms use. For some reason, I cannot produce lower case from the keyboard. It is possible

that the previous owner internally strapped the machine to do this. Lower case sent from the computer works fine.

Because of the large numbers of these machines around, paper is no problem. It is available in white with black or blue print from various suppliers. Cost is between $3.50 and $6.50 a roll, depending on supplier. Some suppliers offer overnight delivery on thispaper. The 733 will also use a similar paper for HP terminals. I do not expect paper supply to be a problem for any reasonable time in the future.

The keyboard is another unit. The keyboard appears to be a heavy-duty unit made by Micro Switch. The keys do not have real tactile feedback but have more 'feel' than the standard OSI keyboard.

There are three edge connectors on the back apron. The smallest of these mates with a cable that came with my unit, and is the RS-232 interface. The function of the two larger ones is unknown to me; although I suspect that one might be used with an interface board to 'hardwire' the 733 directly to a TI computer.

Mounted on cards in the card cage and accessible through the card cage cover are several switches. One is marked Baud Rate and cryptically calibrated Lo, Med, and Hi. Another is marked full or half duplex. A third is marked for odd, mark, or even parity. The meager documentation that I had, actually little more than a few pages Xeroxed from a TI catalog, did no go into the function or operation of these switches. It did, however, give the description and pin-out of the RS-232 signals the 733 used and output.

This brings us around to interfacing. I brought the 733 to George Brown's when I went to pick up my C4 after he had populated the RS-232 port. We plugged in the TI, executed a SAVE, hit a few keys: and nothing happened. I half expected this. I must have tried every one of the 18 possible positions of the interior switches with no luck.

Next we thought that the 733 might need a negative swing in the input signal. To check this hypothesis, we looked at the signals that the 733 was putting out. We did find a negative on one of the pins. Not wanting to modify the computer to supply a negative voltage, I studied the documentation. It seemed that the TI is meant to be used with a modem either internal or external. Pins 6 and 8, Data Set Ready and Data Carrier Detect respectively, were expected to be pulled high by a modem in order to receive data. So, we fed through the ground, signal and RTS (request to send) lines; pulled pins 6 and 8 high by using the signal output by the 733 on pin 20 (always high according to the documentation), and lo and behold, we were able to print gibberish.

Blocked but undaunted, I tried different settings on the internal switches again. Finally, with the speed set to hi, full duplex and regardless of the parity switches position, the 733 printed. My current set up for the 733 makes use of a short connecting cable I made up where I pull the signal from pin 20 on the 733

and apply it to pins 6 and 8. In both the C1 and the C4 there is a resistor that can be installed for what OSI calls a 'special printer' which will pull up pins 6 and 8. So one has two ways to go in this respect.

Will the 733 input into an OSI computer? I don't know. Since I bought the unit only for its printer I have not tried this option. However, some other pins on the 733 would have to switched for output and your OSI would have to have its input switched also. Alll of this switching would definitely limit the usefullness of the TI as an alternate input to your computer.

Two slots in my 733's card cage are available but not filled, from what I can understand from the documentation these may be for use with the ASR option. I do not know much about these tape drives, except that they operate at 1200 baud and so, therefore, cannot be used with the standard modem equipped 733 (modems only operate at 300 baud). (Note - certain modems can operate up to 9600 baud over normal phone lines. Ed.)

One other internal switch is inside the 733, it selects either single or double line feed. The double line feed is nice when you want uncluttered listings for documentation or annotation.

The bottom line of all of this is -- how much does it cost? In mid-1979 I paid $435 for my used 733 KSR. I got it from Computer Applications, Box 203 446 Newbridge Avenue, East Meadow, NY 11554 who are essentially a basement operation dealing in small lots of 'big computer' peripherals and other surplus. Recently they have made the NY Amateur Computer Flea Market and you might see them there. They do not have a business address per se, so it might be best to write to them. If they don't have any more 733's they might have something else. One other source might be The Computer Room, 2522 Butler Street, Dallas, Texas 75235.

If you can find a 733 at a reasonable price and can live with a thermal printer you should consider it. It's advantages are: heavy duty electronics and mechanics, good supply paper, repairs and support from Texas Instruments, quiet operation and above all, it works with OSI. Don't overlook the advantages of a 733 KSR with or without built-in modem. With one of these you don't need a dumb terminal program, you have a real dumb terminal. And, if the price of my unit is indicative of the going price of these units, the RO unit should sell for under $400, which would make it the cheapest printer around.

```
OK
THISIS AN EXAMPLE OF MIXED UPPER AND LOWER CASE PRINTED ON A TI 733

1234567890:-  ABCDEFGHIJKLMNOPQRSTUVWXYZ  ABCDEFGHIJKLMNOPQRSTUVWXYZ

!"#$%&' ()*+=<>?+/.,
```

Understanding BASIC

Part I - The Routine at 00BC

Michael Bassman
39-65 52nd Street
Woodside, NY 11377

There is a very important subroutine on page zero, at location (hex) 00BC. This routine can be used to customize BASIC by adding commands and suchlike. The function of this routine is to get and additional character of the BASIC program. Thus, when you change this routine for your purposes, you must assume that it will be called each time any single character of your BASIC program is interpreted. First, I will give a dissassembly of the routine, then an explanation.

Location	Machine Code	Mnemonics
00BC	E6 C3	INC $00C3
00BE	D0 02	BNE $00C2
00C0	E6 C4	INC $00C4
00C2	AD 50 0B	LDA $0B50
00C5	C9 3A	CMP #$3A
00C7	B0 0A	BCS $00D3
00C9	C9 20	CMP #$20
00CB	F0 EF	BEQ $00BC
00CD	38	SEC
00CE	E9 30	SBC #$30
00D0	38	SEC
00D1	E9 D0	SBC #$D0
00D3	60	RTS

Address Range	Explanation
00BC-00C1	This portion increments the address of the character that is obtained by the instruction at 00C2. First, the low byte of the address is incremented, then if the carry is set (the number incremented was 255), the high byte of instruction is incremented.
00C2-00C2	This load instruction is the heart of the BC routine. It simply gets the next character of the BASIC program. If you are executing a command from immediate mode, the address at 00C3-00C4 will point at some place within the input buffer, which begins at $0013. If you want this routine to call up the character that has already been processed rather than the next character, begin execution of the routine at 00C2, rather than 00BC.
00C5-00C8	Here, we check to see if the character just added is not an ASCII numeral. The ASCII value of numbers ranges up to $39, which is why we compare with $3A. If the character is not a numeral, execution jumps to 00D3.

00C9-00CC — ASCII numbers are not the only important characters with values less than $3A. The space has a value of $20. Here, we check to see if the character is a space. If the character is a space, then we go back to the beginning of the routine, which increments the pointer address and fetches another character. The reason we do this is that spaces are meaningless, and if you put one in, it is ignored during execution.

00CD-00D2 — Previously, we have checked to see that the ASCII value is no greater than $39. In this piece of code, we check to see that the value is no less that $30. Microsoft has done this in a strange and lengthy, albeit working, way. As a matter of fact, they never even use a compare instruction. Instead, they use a series of subtractions which will clear the carry if the number is within the proper range (that being $30-$39). Since this is such an unusual piece of code, I will try to explain how it works. Two subtractions are done. First, $30 is subtracted, the $D0.
$30 + $D0 = $100, so the value is not actually changed. But if the value it less than $30, the first subtraction rather than the second will clear the carry. (Remember, in subtraction, the carry is cleared on overflow). Thus, if you have an invalid value (under $30), the carry will be set rather than cleared when this portion of the BC subroutine ends. This provides an easy means for a jump to an error subroutine. (Through a BCS instruction.)

00D3-00D3 — Return to wherever this subroutine was called from.

Using this information, you can begin to understand how a customization of BASIC could be accomplished. Next month, I will show you how to make your own commands through the BC routine.

Machine Language Call
by Michael Cohen

Here's one for disk users. Running this program will add a CALL statement by overlaying the code for NULL. If you still need a null function, you can do this by POKEing the value into 21. If you want, it can also disable CALL and restore NULL.

The syntax for the statement is simply CALL expression, where expression is any formula which evaluates to a valid address. The machine language routine must end with a RTS and cannot return any values.

```
9 PRINT:PRINT:PRINT:PRINT:PRINT
10 PRINTTAB(7);"- CALL -":PRINT:PRINT
20 INPUT "ENABLE OR DISABLE";A$
30 A$=LEFT$(A$,1)
40 IF A$="E" THEN 100
50 IF A$="D" THEN 200
60 DATA 32,24,22,208,250,232,234,10,176
70 DATA 32,185,12,32,114,22,108,25,0
80 GOTO 20
100 FOR M=0 TO 8:READ N:NEXT M
110 FOR M=0 TO 8:READ N:POKE 2157+M,N:NEXT M
115 POKE 709,ASC ("C")
116 POKE 710,ASC ("A")
117 PRINT
120 PRINT"CALL IS NOW ENABLED."
130 PRINT:PRINT"THE SYNTAX IS:"
140 PRINT"  CALL AD, WHERE AD IS  ANY VALID EXPRESSION."
150 PRINT"THE NULL STATEMENT IS NO LONGER AVAILABLE."
160 END
200 FOR M=0 TO 8:READ N:POKE 2157+M,N:NEXT M
210 POKE 709,ASC ("N")
220 POKE 710,ASC ("U")
230 PRINT:PRINT"THE CALL STATEMENT HAS  BEEN DISABLED AND THE"
240 PRINT"NULL STATEMENT RESTORED."
250 END
```

By Michael Cohen

Here is a listing of some magazine articles which some club members may find interesting. They all appeared in the last two years and the issues should still be available.

MICRO 1979-80

Article	Author	Issue	Page
A Close Look at the Superboard	Hoyt	Apr 79	15
Structured Basic Editor	Abrahamson	Jul 79	7
OSI Memory Test	Taylor	Jul 79	29
Tokens	Morris	Aug 79	20
Tiny Pilot	Urtis	Sep 79	41
Superboard II Peripherals	Hoyt	Oct 79	43
Useful ROM Subroutines	Murphy	Nov 79	9
C1P Graphics Part I	Taylor	Dec 79	61
Basic Compiler	Beach	Jan 80	9
Great Superboard Speed Up	J.R. Swindell	Feb 80	31
C1P Graphics Part III	Taylor	Feb 80	47
Polling OSI's Keyboard	E. Carlson	Mar 80	17
Challenger Cassette Techniques	R. Lary	Mar 80	25

Kilobaud 1979-80

Article	Author	Issue	Page
String Conversion Techniques	R. Roth	May 78	94
Swords and Sorcery	B. Turrie	Aug 78	54
Metric Conversion Program	M. Ferguson	Sep 78	46
Universal Number Converter	E. Beymer	Nov 78	67
Scratched Diskette?	M. Miller	Dec 78	106
A Block Structured Language	L. Fish	Feb 79	24
Dots 1	Pittman	Feb 79	84
The OSI Model 500	Ruckdeschel	Mar 79	130
Dots 2	Pittman	Apr 79	34
Superboard II	Chamberlain	Jul 79	66
1802 Pilot	Petty	Jul 79	78
Visit to OSI	Badgett	Aug 79	118
Tiny Text Editor for the 1802	Petty	Dec 79	116
Reverse Video on a 540	Lary	Dec 79	128
OSI Basic Renumber	Aughey	Jan 80	74
C1P MF Review	Curley	Jan 80	140
Microchess Modifications	McCormack	Feb 80	68

Creative Computing 1979-80

Article	Author	Issue	Page
OSI Challenger IIP Review	Shapiro	May 78	42
Star Wars Game	Ronayne	Sep 78	134
OSI Superboard II Review	Heuer	Jan 79	120
Educational Uses of the C1P	Kuska	Jul 79	40
Adventure Games	Adams	Aug 79	90
A 6502 Disassembler in Basic	Scarpelli	Sep 79	124
Adventure	Moser	Nov 79	108
Quick Printer II	Blank	Nov 79	32
A Program that Learns	Levinsky	Jan 80	90

Byte 1979-80

Article	Author	Issue	Page
Sound Off	S. Ciarcia	Jul 79	34
Quest	R. Chaffee	Jul 79	176
Othello	P. Maggs	Nov 79	66
6502 Indirect Addressing	K. Skier	Jan 80	118

Source Update
By Michael Bassman

XPLO Compiler/Interpreter

Finally, a new high level language for OSI computers. Since there is both a compiler and an interpreter, you can use the interpreter for easy debugging, then compile the program for machine language speed. A cassette version is available! This is the official description.

"XPLO is a block structured, high level compiler language for Ohio Scientific computers. This new programming language includes a self-contained editora and run-time interpreter. The editor alllows easy source code creation and editing, and the interpreter makes XPLO programs transportable to any computer that has the interpreter written for it. Also, the block structure allows the creation of easy to understand, self documenting code."

Pegasus Software
P.O. Box 10014
Honolulu, HI 96816

OSI Stringy Floppy

An alternative to the expense of a mini-floppy and an expansion board. There is both an OSI and RS-232C version. Will cost about $300. More details are forthcoming.

Exatron
181 Commercial Street
Sunnyvale, CA 94086
800-538-8559

New Software and other miscellaneous items

These people are advertising an 'Asteroids' type game.

Dare Data and Design
P.O. Box 8433
Baltimore, MD 21234

Numeric Keypad for the OSI

BKM Micro Systems Corp.
3809 Old Colleg Road
Bryan, Texas 77801

New Software Listings and Hardware Plans

A new firm, run by Arthur Mittendorf, is selling software listings for the Challenger 1P. He is also selling a kit for true high resolution on the C1P. This would enable you to get 192x192 on a 24x24 screen, and 256x256 on a 32x32 screen. Normal characters could also be displayed simultaneously with hi-res display. Also, he features a $15 simple music generator kit. His software listings are dirt cheap.

Mittendorf Engineerng
905 Villa Nueva Drive
Litchfield Park, AZ 85340

FORTH and other software for OSI

Technical Products Co., is now selling a version of the language FORTH for any OSI machine. This will even run on a 8K C1P. Send $1.00 for their full line catalog.

Technical Products Co.
Box 12983 University Station
Gainesville, FL 32604

Single Disk Drive

by Ugo V. Re

If you have a single disk drive on your system you know how hard it is to make a duplicate of a disk.

You can use "LOAD" and "PUT" or "CALL" and "SAVE" to transfer programs or tracks, however this method is very time consuming when you want to copy a full disk.

The AADVARK JOURNAL, JUNE 1980 issue had a program that you can use to copy tracks 1 to 14.

The following program written for a C2/C4/C8 system with a minumum of 20K RAM, will copy all tracks, except track zero from one disk to another, using the one drive.

The program should work on a C1P MF with 20K RAM if the following changes are made.

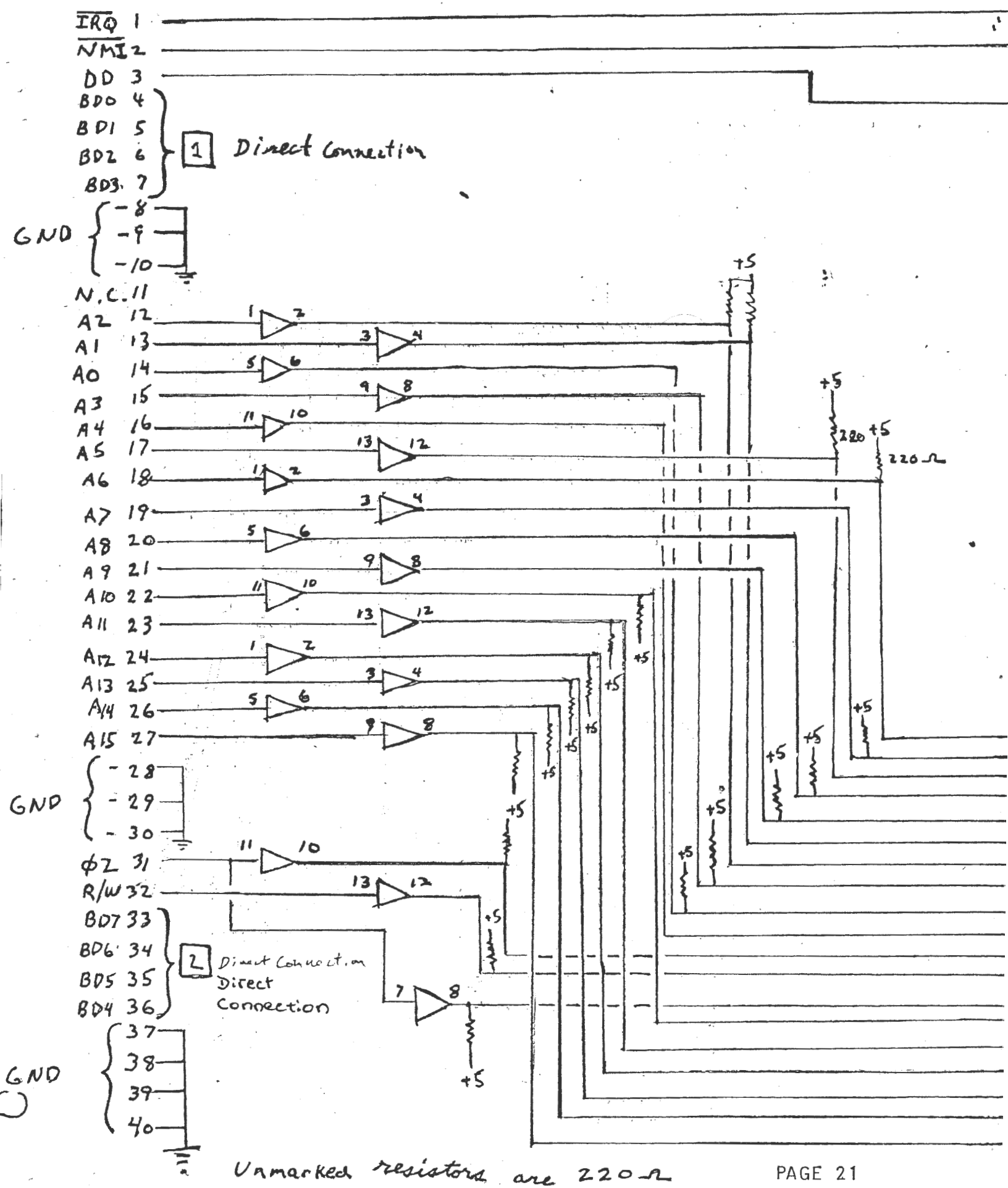
```
450 NUM = VAL(CHR$(PEEK(54178)))
470 PAG$(PAG+1) = CHR$(PEEK(54181-PAG*32))
530 POKE 57088,253 : KEY = PEEK(57088) : IF KEY <> 239 THEN 530
560 POKE 57088,253 : KEY = PEEK(57088)= : IF KEY <> 239 THEN 560
770 POKE 57088,253 : KEY = PEEK(57088) : IF KEY <> 239 THEN 770
850 POKE 57088,253 : KEY = PEEK(57088) : IF KEY <> 239 THEN 850
```

```
100 REM DISK COPIER
110 REM DISABLE CONTROL  C
120 REM LIMIT UPPER WORK SPACE
130 POKE 2073,96: POKE 132,240: POKE 133,66
140 FOR CL=1 TO 30: PRINT: NEXT CL
200 PRINT"     DISK  COPIER": PRINT: PRINT: PRINT
210 PRINT"  THIS PROGRAM WILL COPY ALL TRACKS EXCEPT"
220 PRINT"  TRACK ZERO.": PRINT: PRINT
230 PRINT"  FOLLOW ALL INSTRUCTIONS AS YOU COPY EACH"
240 PRINT"  TRACK FROM A SOURCE DISK TO A NEW DISK."
250 PRINT: PRINT
260 INPUT"  HAVE YOU INITIALIZED THE NEW DISK";A$
270 PRINT: PRINT: IF LEFT$(A$,1)<>"Y" THEN 800
300 INPUT"   COPY FROM TRACK";T1: PRINT
310 IF T1<1 OR T1>39 OR T1<>INT(T1) THEN 300
320 INPUT"   TO TRACK";T2: PRINT
330 IF T2<1 OR T2>39 OR T2<>INT(T2) THEN 320
340 IF T1>T2 THEN 300: PRINT: PRINT
360 PRINT"   LOAD THE SOURCE DISK IN THE DRIVE"
370 PRINT: INPUT"   ARE YOU READY TO COPY";A$
380 IF LEFT$(A$,1)<>"Y" THEN RUN 300
400 FOR TRK=T1+100 TO T2+100
410 TRK$=RIGHT$(STR$(TRK),2)
420 DISK!"DIR "+TRK$
*   450 NUM=VAL(CHR$(PEEK(55106)))
460 FOR PAG=0 TO NUM
*   470 PAG$(PAG+1)=CHR$(PEEK(55109-PAG*64))
480 NEXT PAG
500 FOR SEC=1 TO NUM: SEC$=RIGHT$(STR$(SEC),1)
510 DISK!"CA 4300="+TRK$+","+SEC$: PRINT
520 PRINT"   LOAD NEW DISK AND PRESS SPACE BAR"
*   530 POKE 57088,2: KEY=PEEK(57088): IF KEY<>16 THEN 530
540 DISK!"SA "+TRK$+","+SEC$+"=4300/"+PAG$(SEC)
550 PRINT"   LOAD SORCE DISK AND PRESS SPACE BAR"
*   560 POKE 57088,2: KEY=PEEK(57088): IF KEY<>16 THEN 560
570 NEXT SEC: NEXT TRK
600 PRINT: PRINT"   ALL TRACKS EXCEPT ZERO COPIED": PRINT
610 PRINT"   TO COPY TRACK ZERO DO THE FOLLOWING:": PRINT
620 PRINT"   WHEN  A*  APPEARS TYPE  CA 0200=13,1  RETURN"
630 PRINT"   WHEN  A*  PEARREARS TYPE  GO 0200  RETURN"
640 PRINT"   WHEN  ?  APPEARS TYPE  2  RETURN"
650 PRINT"   WHEN  ?  REAPPEARS TYPE  R4200  RETURN"
660 PRINT"   LOAD NEW DISK THEN TYPE  W4200/2200,8  RETURN"
700 PRINT:PRINT"   TO TEST THE NEW DISK DEPRESS THE"
710 PRINT"   BREAK KEY THEN THE  D  KEY.  THE NEW DISK"
720 PRINT"   SHOULD BOOT UP AND YOU CAN RUN A PROGRAM."
730 PRINT: PRINT
740 PRINT"   WHEN YOU ARE READY TO COPY TRACK ZERO"
750 PRINT"   LOAD THE  OS65D  DISK THEN DEPRESS SPACE BAR"
*   770 POKE 57088,2: KEY=PEEK(57088): IF KEY<>16 THEN 770
780 EXIT
800 PRINT"  TO INITIALIZE A NEW DISK PLACE IT IN THE"
810 PRINT"  DRIVE AND DEPRESS THE SPACE BAR.": PRINT
820 PRINT"  WHEN THE  OK  APPEARS PLACE THE SOURCE"
830 PRINT"  DISK INTO THE DRIVE, TYPE RUN 300 THEN RETURN"
840 PRINT: PRINT
*   850 POKE 57088,2: KEY=PEEK(57088): IF KEY<>16 THEN 850
860 FOR HOLD=1 TO 1000: NEXT
870 DISK!"INIT"
```

Ugo V. Re
Levittown,
New York

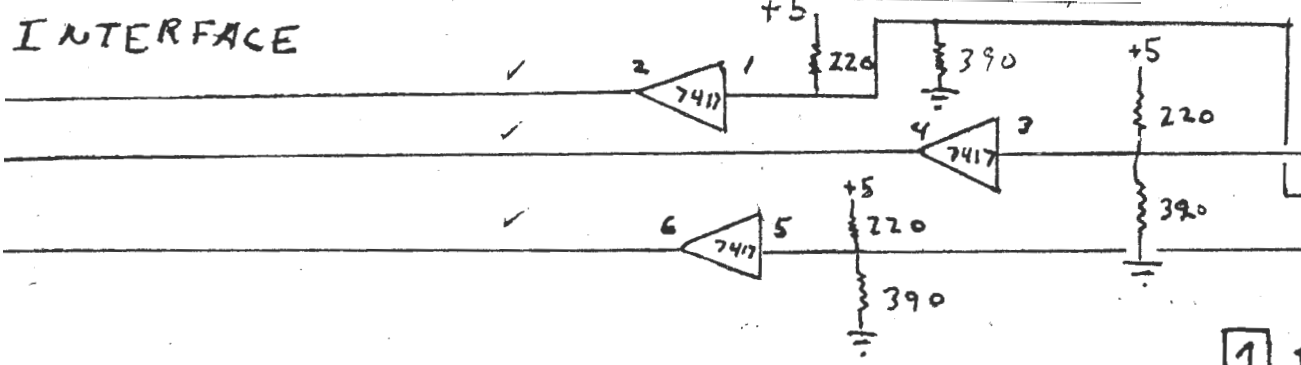
*Change fo
ClP

J1 Pin OSI 600 BOARD TO OSI. BUS



Unmarked resistors are 220Ω

INTERFACE

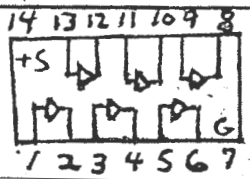


	Bus Line
B1	$\overline{\text{Wait}}$
B2	$\overline{\text{NMI}}$
B3	$\overline{\text{IRQ}}$
B4	Data Direct
B5	D0
B6	D1
B7	D2
B8	D3
B9	D4
B10	D5
B11	D6
B12	D7
B13	D8
B14	D9
B15	D10
B16	D11
B17	$\overline{\text{RESET}}$
B18	UNDEF
B19	A19
B20	A18
B21	A16
B22	A17
B23	+12V
B24	-9V
B25	+5V
B26	+5V
B27	GND
B28	GND
B29	A6
B30	A7
B31	A5
B32	A8
B33	A9
B34	A1
B35	A2
B36	A3
B37	A4
B38	A0
B39	$\phi 2$
B40	R/W
B41	VMA
B42	VMA·$\phi 2$
B43	A10
B44	A11
B45	A12
B46	A13
B47	A14
B48	A15

1

2

+5 — 220Ω — B41



7417 OPEN-COLLECTOR DRIVER