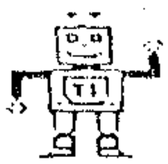


COMPUTER CONTROLLED ROBOT

By Ken Gladyszewski

CONTROL A ROBOT OR ANYTHING USING 'PIO'

Some years ago while visiting my local Radio Shack, I discovered a robot called 'The Mobile Armatron' and immediately decided that controlling this robot with my computer would make a great project. So did Radio-Electronics Magazine in an article which appeared in their May 1987 issue using a Commodore 64. This simple robot consists of five motors which cause different motions when energized by a handheld remote attached to the robot via a two foot, seven conductor ribbon cable. The remote consists of a number of pushbuttons which when pushed connect the wires of the ribbon cable in various combinations to energize one of the motors in a given direction. After reading and understanding the article, I concluded the TI parallel port is similar to the Commodore user port and could be used in a like manner to control the robot. These ports both have 8 data bit outputs which can be used to drive relays whose contacts then act just as the pushbuttons did in the remote. My design for the interface uses SPDT relays instead of the SPST relays used in the article and only 6 instead of 7 of the 8 available data bits. In addition to less hardware, the contacts are interconnected so that each motor acts as a generator when power is removed, causing the motion to stop quickly instead of coasting. The remaining bits can be used for other purposes, such as speed control, by switching out resistors to vary the voltage to the motor.

Before attempting to write the program and design the interface, I did some investigation into the workings of the parallel port and learned the following:

The parallel port on the TI RS232 card is bi-directional, meaning the port may be used to input as well as output information. A printer and the robot use the output ability solely and can be used with a parallel interface that can only output data, such as the Axiom parallel. The connector on the TI card is a 16 pin IDC type having 2 rows of 8 pins, per Fig. 1. The data bits are connected to a pair of cross wired 74LS373, 8 bit transparent latch IC's. These chips are capable of sinking 18 MA (input mode) and sourcing 6 1/2 MA (output mode). LED's and relays (even low power reed type) require more current than this and must be driven by a transistor.

When the computer outputs a byte, it causes the 'handshake out' line to go high when the 'data bit' lines have all changed. The data lines are latched, i.e., they remain in their last state until changed by new information. The peripheral device must cause the 'handshake in' line to go low when it has read the data lines. The computer is then allowed to change the 'data bit' lines to output another byte.

This information was used to design the universal interface shown in Fig. 2. The design is modular using a handful of inexpensive generic parts which are easily obtainable. All relay contacts are wired to a 25 pin D-Sub connector (like the one used with the RS232 serial port). Interconnections between the contacts required to use the interface with the robot are made in the mating interface cable which attaches between the interface box and the robot ribbon cable. Using this approach, one interface may be used for many different projects, each with it's own custom cable. Almost anything can be controlled using this interface and some imagination! These relays could energize bigger relays if final devices require a large amount of power. The program in Fig. 3 shows how keys 0-9 on the computer keyboard may be used to energize one or more relays at a time.

The actual program for the robot includes unlimited speech and the ability to store motion commands to disk. The commands may be input to the computer interchangeably through the keyboard or the handheld remote attached to the joystick port. The program is in this newsletter. It consists of 120 lines of Extended Basic with no Assembly routines, except for the text-to-speech routine. The program could be re-written to run in Basic on a console, using the Terminal Emulator cartridge, a cassette recorder and a stand alone parallel port.

If you have a Mobile Armatron and would like to computerize it now, or have other uses for this information and can't wait - - - write to me!

Ken Gladyszewski
6440 St. Rte 86
Concord, OH 44077

Editor's note:

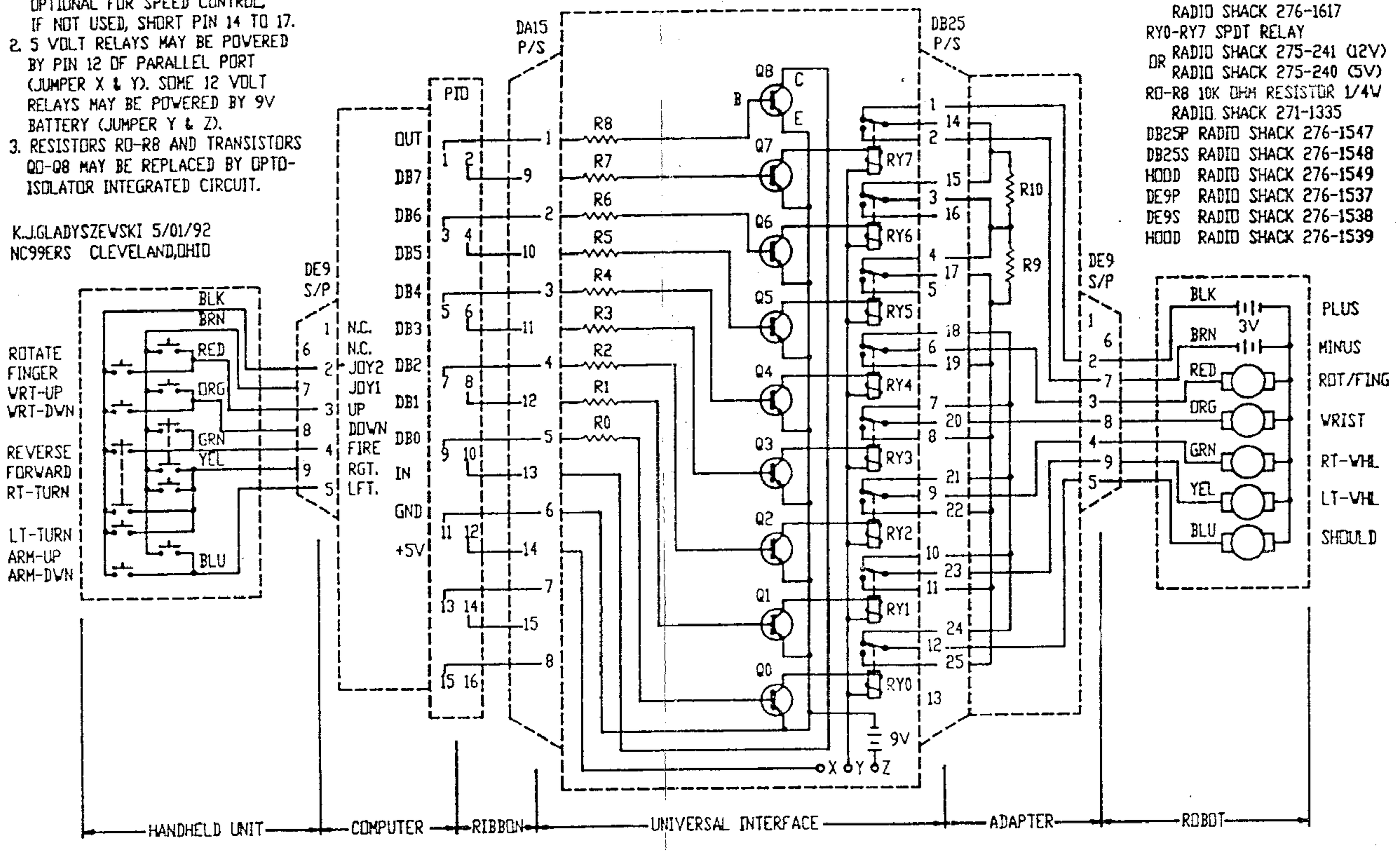
It must be very hard putting all this effort into letting others know what amazing things our TI can do when there is very little feedback from the readers. Ken told me this project has spawned many more ideas that cannot possibly be looked into without the help of other hardware hackers, as time is always an enemy which can't be overlooked! Please write Ken if you would like to get involved in this project or any other that Ken has written about in the past. Also, write Ken, just to let him know if you like his ideas or not. This way, you may see more projects printed in this Newsletter! Thanks, Harry

SCHEMATIC FOR COMPUTER CONTROLLED ROBOT

NOTES:
1. R6, R7, R9, R10, Q6, Q7, RY6, RY7, OPTIONAL FOR SPEED CONTROL. IF NOT USED, SHORT PIN 14 TO 17.
2. 5 VOLT RELAYS MAY BE POWERED BY PIN 12 OF PARALLEL PORT (JUMPER X & Y). SOME 12 VOLT RELAYS MAY BE POWERED BY 9V BATTERY (JUMPER Y & Z).
3. RESISTORS R0-R8 AND TRANSISTORS Q0-Q8 MAY BE REPLACED BY OPTO-ISOLATOR INTEGRATED CIRCUIT.

K.J.GLADYSZEWSKI 5/01/92
NC99ERS CLEVELAND,OHIO

PARTS LIST:
Q0-Q8 GENERAL PURPOSE NPN SWITCHING TRANSISTOR RADIO SHACK 276-1617
RY0-RY7 SPDT RELAY
OR RADIO SHACK 275-241 (12V)
RADIO SHACK 275-240 (5V)
R0-R8 10K OHM RESISTOR 1/4W RADIO SHACK 271-1335
DB25P RADIO SHACK 276-1547
DB25S RADIO SHACK 276-1548
HOOD RADIO SHACK 276-1549
DE9P RADIO SHACK 276-1537
DE9S RADIO SHACK 276-1538
HOOD RADIO SHACK 276-1539

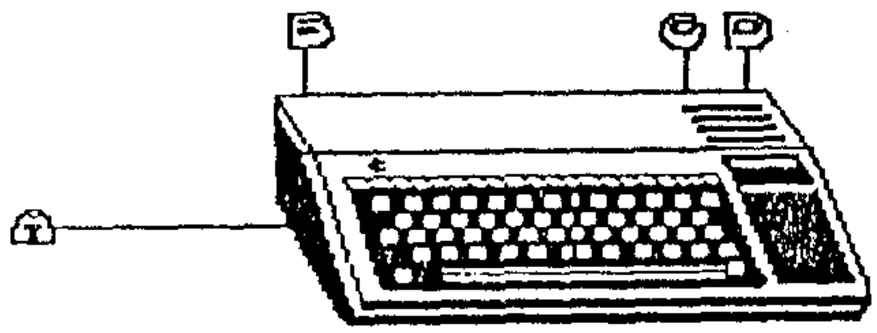


```
1 !SAVE DSK4.ROBOT-PIOS
100 !*******************
110 !** CAUTION !!    **
120 !** DO NOT PUT    **
130 !** BATTERY IN    **
140 !** ROBOT UNTIL   **
150 !** PROGRAM HAS   **
160 !** INITIALIZED   **
170 !** ALL RELAYS    **
180 !** ENERGIZED     **
190 !** AT POWER UP   **
200 !*******************
280 DIM A(12),B(250,1),C$(25
0):: OPEN #1:"PIO",FIXED 1 :
: PRINT #1:CHR$(0):: CLOSE #
1
290 CALL INIT :: CALL LOAD("
DSK1.SPEAK","DSK1.XLAT","DSK
1.SETUP"):: CALL LINK("SETUP
","DSK1.DATABASE")!MUST HAVE
 TEXT-TO-SPEECH FILES ******
300 RESTORE 310 :: FOR I=0 T
O 12 :: READ Z :: A(I)=Z ::
NEXT I
310 DATA 0,6,134,2,132,1,128
,8,135,16,143,131,129
320 SS$=" 1234567890+-*
330 PRINT RC$;"    M A I N
 M E N U"
340 PRINT :: PRINT TAB(4);"(
CTRL) T : TEACH(LEARN)" :: P
RINT :: PRINT TAB(4);"(CTRL)
 P : PERFORM(DO)"
350 PRINT :: PRINT TAB(4);"(
CTRL) S : SAVE" :: PRINT ::
PRINT TAB(4);"(CTRL> R : RET
RIVE"
360 PRINT :: PRINT TAB(4);"(
CTRL) Q : QUIT(END)" :: PRIN
T :: PRINT TAB(4);"(CTRL) U
: UNDO(RETRACE)"
370 PRINT :: PRINT "    WHIC
H... "
380 CALL KEY(5,SR,STATUS)::
IF STATUS=0 THEN 380
390 SR=SR-143 !DETERMINE WHI
CH MENU ENTRY *************
400 IF SR<1 OR SR>6 THEN 380
410 ON SR GOSUB 980,1410,128
0,1190,430,930
420 GOTO 300 !REDEFINE CORRE
CT RELAYS FOR NORMAL OPERATI
ON ************************
430 GOSUB 1420 :: J=B(0,0)::
 X=0 ! TEACH LEARN MODE ***
440 OPEN #1:"PIO",FIXED 1 !O
NLY ONE BYTE IS SENT TO PARA
LLEL PORT AT A TIME *******
450 CALL KEY(5,KEY,STATUS)::
 IF STATUS=0 THEN 550 !EXAMI
NE KEYBOARD FOR COMMAND ELSE
 JOYSTICK ******************
460 IF KEY=13 THEN LINPUT "P
HRASE- ":C$(0):: IF C$(0)=""
 THEN 480 :: IF ASC(SEG$(C$(
0),1,1))=60 THEN P=60 ELSE P
=50 !AT BEGINNING OF PHRASE
MEANS USE LOW PITCH *******
470 E$=C$(0):: IF KEY=13 THE
N CALL LINK("XLAT",C$(0),D$)
!CR CAUSES SPEECH ********
480 IF KEY=13 THEN CALL LINK
("SPEAK",D$,P,128):: GOTO 45
0
490 IF KEY=141 THEN CLOSE #1
 :: I=0 :: RETURN !BACK TO M
ENU ************************
500 I=POS(SS$,CHR$(KEY),1)!D
ETERMINE WHICH MOTION KEY WA
S PRESSED ******************
510 IF I>0 THEN I=I-1 ELSE C
ALL SOUND(100,220,0):: GOTO
450
520 PRINT #1:CHR$(A(I))!SEND
 DATA TO PIO PORT *********
530 CALL KEY(5,KEY1,STATUS):
: IF KEY=KEY1 THEN GOSUB 890
 ELSE 900 !LOOK FOR RELEASE
OF KEY ********************
540 GOTO 530
550 I=0 ! EXAMINE HANDHELD*
560 CALL JOYST(1,V,W):: IF V
=0 AND W=0 THEN 610
570 IF W=4 THEN I=9 :: GOTO
660
580 IF W=-4 THEN I=7 :: GOTO
 660
590 IF V=-4 THEN I=5 :: GOTO
 660
600 IF V=4 THEN CALL KEY(1,T
,R):: IF T=18 THEN I=1 :: GO
TO 660 :: ELSE IF T=-1 THEN
I=3 :: GOTO 660
610 CALL JOYST(2,Z,Y):: IF Z
=0 AND Y=0 THEN 450
620 IF Y=4 THEN I=10 :: GOTO
 660
630 IF Y=-4 THEN I=1 :: GOTO
 660
640 IF Z=-4 THEN I=6 :: GOTO
 660
650 IF X=4 THEN CALL KEY(2,K
,S):: IF K=18 THEN I=2 :: GO
TO 660 :: ELSE IF K=-1 THEN
I=4 :: GOTO 660
660 PRINT #1:CHR$(A(I)):: IF
 I=0 THEN 450 !OUTPUT COMMAN
D TO PIO ******************
670 ON I GOSUB 690,710,730,7
50,770,790,810,830,850,870
680 RETURN !LOOK FOR RELEASE
 OF BUTTON WHICH STARTED MOT
ION ************************
690 CALL JOYST(1,V,W):: IF V
=4 THEN GOSUB 890 ELSE 900
700 GOTO 690
710 CALL JOYST(2,Z,Y):: IF Z
=4 THEN GOSUB 890 ELSE 900
720 GOTO 710
730 CALL JOYST(1,V,W):: IF V
=4 THEN GOSUB 890 ELSE 900
740 GOTO 730
750 CALL JOYST(2,Z,Y):: IF Z
=4 THEN GOSUB 890 ELSE 900
760 GOTO 750
770 CALL JOYST(1,V,W):: IF V
=-4 THEN GOSUB 890 ELSE 900
780 GOTO 770
790 CALL JOYST(2,Z,Y):: IF X
=-4 GOSUB 890 ELSE 900
800 GOTO 790
810 CALL JOYST(1,V,W):: IF W
=-4 THEN GOSUB 890 ELSE 900
820 GOTO 810
830 CALL JOYST(2,Z,Y):: IF Y
=-4 THEN GOSUB 890 ELSE 900
840 GOTO 830
850 CALL JOYST(1,V,W):: IF W
=4 THEN GOSUB 890 ELSE 900
860 GOTO 850
870 CALL JOYST(2,Z,Y):: IF Y
=4 THEN GOSUB 890 ELSE 900
880 GOTO 870
890 X=X+1 :: RETURN ! IF SAM
E MOTION THEN INCREMENT TIME
 ****************************
900 PRINT #1:CHR$(0)!STOP AL
L MOTION BETWEEN COMMANDS**
910 J=J+1 :: B(J,0)=I :: B(J
,1)=X :: X=0 :: B(0,0)=J ::
C$(J)=E$ :: E$="" :: C$(0)="
" :: D$=""
920 PRINT B(0,0);B(J,0);B(J,
1):: GOTO 450
930 !UNDO RETRACE MODE******
940 RESTORE 950 :: FOR I=0 T
O 12 :: READ Z :: A(I)=Z ::
NEXT I
950 DATA 0,134,6,131,129,128
,1,135,8,16,143,2,132
951 ! DEFINE OPPOSITE MOTION
S ****************************
960 L=B(0,0):: M=1 :: N=-1 :
: GOTO 1010
970 PRINT :: PRINT "  UNDO(R
ETRACE) PROCEDURE" :: PRINT
:: PRINT
980 ! PERFORM(DO)MODE ******
990 L=1 :: M=B(0,0):: N=1
1000 PRINT :: PRINT "   PERF
ORM(DO) PROCEDURE" :: PRINT
:: PRINT
1010 IF B(0,0)=0 THEN PRINT
"NO PROCEDURE IN MEMORY." ::
 GOTO 1160
1020 PRINT "PRESS ANY KEY FO
R MOTION."
1030 CALL KEY(5,KEY,STATUS):
: IF STATUS=0 THEN 1030
1040 PRINT "EXECUTION IN PRO
GRESS" :: PRINT
1050 OPEN #1:"PIO",FIXED 1 :
: FOR I=L TO M STEP N
1060 IF C$(I)="" THEN 1100
1070 IF ASC(SEG$(C$(I),1,1))
=60 THEN P=60 ELSE P=50
1080 PRINT C$(I)
1090 CALL LINK("XLAT",C$(I),
B$):: CALL LINK("SPEAK",B$,P
,128)
1100 PRINT I;B(I,0);B(I,1)::
 PRINT :: I=0
1110 PRINT #1:CHR$(A(B(I,0))
)
1120 CALL KEY(5,KEY,STATUS):
: IF X<B(I,1)THEN GOSUB 1140
 ELSE 1150
1130 GOTO 1120
1140 X=X+1 :: RETURN
1150 PRINT #1:CHR$(0):: FOR
K=1 TO 1 :: NEXT K :: NEXT I
 :: CLOSE #1 :: PRINT "PROCE
DURE DONE"
1160 PRINT "PRESS ANY KEY FO
R MAIN MENU"
1170 CALL KEY(5,KEY,STATUS):
: IF STATUS=0 THEN 1170
1180 RETURN
1190 ! SAVE PROCEDURE *****
1200 IF B(0,0)=0 THEN PRINT
"NO PROCEDURE IN MEMORY." ::
 GOTO 1220
1210 INPUT "ENTER FILE NAME
TO SAVE ":F$ :: GOTO 1250
1220 PRINT "ABORT. PRESS ANY
 KEY..."
1230 CALL KEY(5,K,S):: IF S=
0 THEN 1230
1240 RETURN
1250 OPEN #2:F$ :: PRINT ::
PRINT "SAVING PROCEDURE. WAI
T."
1260 FOR I=0 TO B(0,0):: PRI
NT #2:B(I,0):B(I,1):C$(I)::
NEXT I :: CLOSE #2
1270 PRINT :: PRINT "PROCEDU
RE SAVED. PRESS ANY KEY." ::
 GOTO 1230
1280 ! RETRIEVE ***********
1290 PRINT "RETRIEVE PROCEDU
RE"
1300 IF B(0,0)=0 THEN 1330
1310 PRINT "PROCEDURE IN MEM
ORY. CONTINUE (Y/N)?" :: GOS
UB 1570
1320 IF CHR$(K)="N" THEN PRI
NT "ABORT." :: GOTO 1350
1330 INPUT "ENTER FILE NAME
TO RETRIEVE ":F$ :: GOTO 137
0
1340 PRINT "PRESS ANY KEY"
1350 CALL KEY(5,K,S):: IF K=
0 THEN 1350
1360 RETURN
1370 PRINT "RETRIEVING PROCE
DURE. WAIT" :: OPEN #2:F$
1380 INPUT #2:B(0,0),B(0,1),
C$(0)
1390 FOR I=1 TO B(0,0):: INP
UT #2:B(I,0),B(I,1),C$(1)::
NEXT I :: CLOSE #2
1400 PRINT "RETRIEVAL COMPLE
TE. " :: GOTO 1340
1410 END
1420 ! TEACH LEARN SCREEN**
1430 PRINT
1440 PRINT " ARMATRON ROBOT
TEACH MODE   " :: PRINT :: P
RINT
1450 PRINT "PRESS KEY TO DO
FUNCTION    PRESS ANY OTHER
KEY TO STOP"
1460 PRINT "PRESS (CTRL) M F
OR MAIN MENU" :: PRINT
1470 PRINT "   KEY FUCTION"
:: PRINT "   --- -------"
1480 PRINT "    1 = FORWARD"
 :: PRINT "    2 = BACKWARD"
1490 PRINT "    3 = RIGHT FO
RWARD TURN" :: PRINT "    4
= LEFT FORWARD TURN"
1500 PRINT "    5 = ARM UP"
:: PRINT "    6 = ARM DOWN"
1510 PRINT "    7 = WRIST UP
" :: PRINT "    8 = WRIST DO
WN"
1520 PRINT "    9 = HAND ROT
ATE" :: PRINT "    0 = FINGE
RS MOVE IN/OUT"
1530 PRINT "    + = RIGHT RE
VERSE TURN" :: PRINT "    -
= LEFT REVERSE TURN"
1540 PRINT "    SB= PAUSE"
1550 PRINT "    CR= SPEECH"
1560 RETURN
1570 CALL KEY(5,K,S):: IF S=
0 THEN 1570
1580 RETURN
```

1030 OPEN #1: "PIO, CR"

440 OPEN #1: "PIO", FIXED 1

PIN ASSIGNMENTS



PIN ASSIGNMENTS APPEAR AS THEY WOULD IF YOU LOOKED DIRECTLY INTO THE PLUGS.

FORT USER GROUP

FEBRUARY, 1987

A

I JOYSTICK PINS

1) N.C.
2) STICK B GROUND
3) UP
4) FIRE BUTTON
5) LEFT
6) N.C.
7) STICK A GROUND
8) DOWN
9) RIGHT

use female plug.

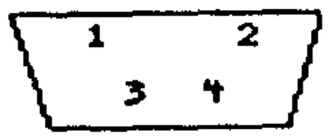
ATARI/COMMODORE JOYSTICK

1) UP
2) DOWN
3) LEFT
4) RIGHT
5) N.C.
6) FIRE BTN
7) N.C.
8) GROUND
9) N.C.

use male plug.

C

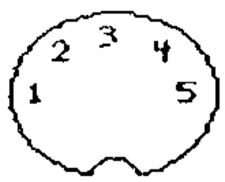
POWER SUPPLY PLUG



1) NOT USED
2) 8 VOLTS AC
3) 16 VOLTS AC
4) GROUND

B

MODULATOR PLUG

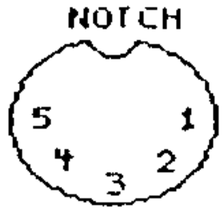


NOTCH

1) RED
2) TRANSPARENT
3) WIRE WRAP (not a ground)
4) BLACK (ground)
5) WHITE or YELLOW

VIDEO/SOUND PLUG

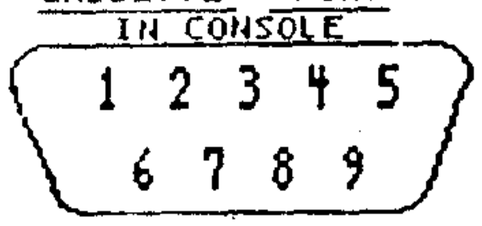
NOTCH



1) +12 volt
2) VIDEO OUT
3) SHIELD
4) GROUND
5) SOUND OUT

D

CASSETTE PORT IN CONSOLE



CASSETTE CABLE BELOW

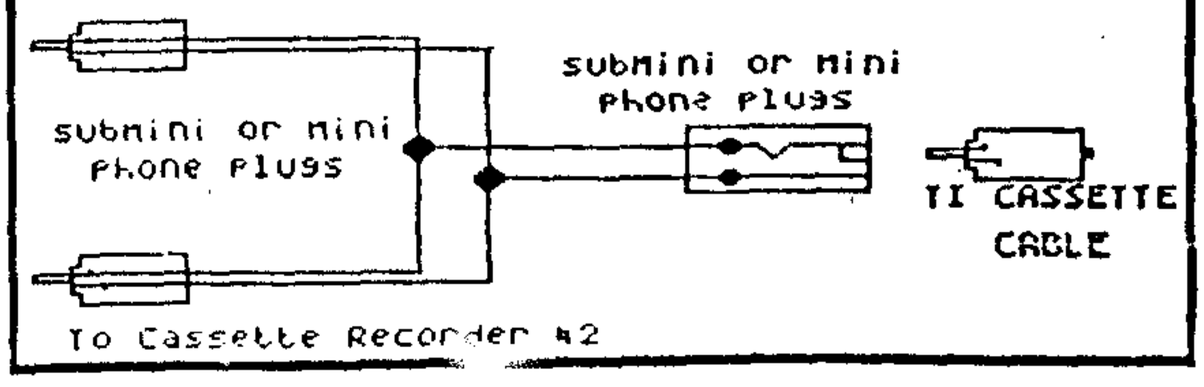
2) NEGATIVE REMOTE CS1 — sleeve
1) POSITIVE REMOTE CS1 — tip
3) NEGATIVE MIC 1 & 2 — sleeve
5) POSITIVE MIC 1 & 2 — tip
7) NEGATIVE REMOTE CS2 — sleeve
6) POSITIVE REMOTE CS2 — tip
9) NEGATIVE SPEAKER CS1 — sleeve
8) POSITIVE SPEAKER CS1 — tip
4) NOT USED

CS1 CS2

✕ SMALL REMOTE PLUG 3/

✕✕ PHONE PLUG 1/8"

Use Y splitter to record to two tapes



MODEMS, Part 2
by Al Kinney

In last month's article, we began by reviewing the structure of data as it applies to telecommunications and modems. This month, then, we'll add some more supporting information.

In addition to understanding the data you want to transmit, you should understand the basic terms that relate to data transfer. This section will explain all those technical words that pop up when the subject is data communications - words like *parity*, *asynchronous*, *bps*, *baud*, *full duplex*, *half duplex*, *XON/XOFF* and *flow control*.

What is Parallel or Serial Transmission?

Now that you understand that data is made of bits and bytes, it will be easy for you to understand how the data is sent from one computer to another. Basically, the computer sends out one kind of electrical signal for a "1" bit and a different signal for a "0" bit.

To understand parallel transmission, think about a large number of people trying to get into a movie. If eight entrances are available at the same time, eight people can get into the movie at the same time. If only one turnstile is available, only one person can enter at a time. When eight data lines move data at the same time, this is said to be parallel operation, while if only a single line is used, it is said to be serial operation.

There are, of course, advantages and disadvantages to using one over the other, but for modem use, it's only practical to use serial, since parallel requires 8 wires to transmit and receive the data, while serial only requires a single wire. Another major advantage of serial transmission is the ability to use standard telephone lines, which isn't possible using parallel connections.

Data Speed - Baud and bps

Terminology which relates to data speed is frequently misunderstood, even by some so-called "experts." Serial data speed is expressed as the number of bits transmitted per second (*bps*). This is often referred to as the *baud* rate, although baud and bps are not necessarily the same. When data speed is given in *bps*, the actual number of bits transmitted per second is specified. When data speed is given in baud, however, this is not true. You already know that data is sent as electrical signals and that the electrical signal for a "1" bit is different from the signal for a "0" bit. A *baud* is a unit of signaling speed that measures the number of electrical signals sent on the line during one second, much like a rain gauge measures inches of rainfall per hour. The baud is named for Frenchman J.M.E. Baudot, who developed the codes used for telegraph systems.

MODEMS, Part 2

If only one bit is transmitted per signal, the bit rate (bps) is the same as the baud rate. Otherwise the two are not equivalent. Typically, more than one bit is sent in one signal, and more than one signal is usually sent in one second. For example, a modem which sends data at 2400 baud can send data at 2400 bps if one bit is sent per signal, at 4800 bps if two bits are sent per signal, or 9600 bps if 4 bits are sent per signal. Special codes can be used to send more than one bit in a signal. The following data speeds are commonly used in data communications: 300, 1200, 2400, 4800, 9600 and 19200 bits per second. Be careful not to confuse baud and bps: *baud* is gradually being replaced by the term *bits per second* because bps has the same meaning all the time, whereas baud does not.

Remember two things about data rates:

- Two pieces of equipment must transmit at the same data rate in order to communicate.

- A common mistake in data communications is to use too high a data rate. For instance, while the port of the TI-99/4A will support 9600 bps operation I haven't found a comm program, which will manage a rate that high, with any consistency.

Synchronous or Asynchronous Transmission

Anything that is *synchronous* is done regularly, and in telecommunications, according to a timing signal. Data bytes are send down the line, one right after another, with no delays in between. Here is a picture of what data looks like when it is being transmitted synchronously:

>••• BYTE 3 BYTE 2 BYTE 1 •••>

If a communications line is *asynchronous*, a delay may occur between bytes, but a delay will *not necessarily* occur between every pair of bytes. Here is what data looks like when it is being transmitted asynchronously:

>•• BYTE 4 BYTE 3 •• BYTE 2 •• BYTE 1 ••>

Asynchronous communications is simpler and less expensive than synchronous, because synchronous communications *requires special hardware which is often quite costly.* Asynchronous communications is most often used by micro and minicomputers, and we'll only consider asynchronous communications in these articles.

Next month, we'll continue discussing Communications Parameters...DUPLEX, Full or Half?

Speech (Part 3)

SPEECH - THE CALL SPGET STATEMENT

(Reprinted from the Amarillo 99/4A Users Group newsletter, Feb 1983)

author unknown

The field of the CALL SAY statement is formatted with a series of "word" and "direct" string expressions, each separated by a comma. Further, the string expressions must be in a specific position in the field of the CALL SAY statement. Word strings must occup the odd positions and direct strings must occupy the even positions. To illustrate, here is an example of the use of word strings:

```
10 A$="HELLO" )
20 B$="HOW" ) word strings
30 C$="ARE YOU" )
40 CALL SAY(A$,"",B$,""C$)
```

Note the placement of the word strings in the odd positions of the CALL SAY field.
Direct strings are defined by the CALL SPGET statement. When the CALL SPGET is used, the speech code for the word is read from the ROM in the Speech Synthesizer and stored in active memory as a string variable with the name as specified in the statement. For example:

```
10 A$="HELLO"
20 CALL SPGET("HOW",B$)  )direct
30 CALL SPGET("YOU",C$)  )string
40 CALL SAY(A$,B$,"ARE",C$)
```

The speech code for "HOW" and "YOU" are stored in active memory and called B$ and C$ respectively.

The two previous examples show that the field of the CALL SAY statement must alternate between word-strings and direct-strings. The format is:

CALL SAY(Word string[,Direct string,Word string...])

The CALL SPGET statement actually calls the code pattern for a word resident in the Speech Synthesizer and assigns it to a string variable. You can then apply this string variable in a variety of ways including using it with a CALL SAY in the same program, storing the speech data on a storage device, or viewing the actual speech data. If the word or phrase specified in the CALL SPGET is not found in the Speech Synthesizer resident vocabulary, the code pattern for UHOH is stored in the string variable.

If the word called does exist, the speech data is stored in the string variable and is preceded by three bytes (ASCII characters) of control information following. The maximum number of bytes of speech data is 252 and the total length of a direct string cannot exceed 255 bytes.

The usefullness of the CALL SYGET statement is that speech data can be put in the form of a string variable that can be added to other speech data, shortened, and/or stored on cassette or disk.

Following is a program to store speech data on cassette tape:

```
10 OPEN #1:"CS1",INTERNAL,OUTPUT,FIXED
192
20 CALL SPGET("RED",B1$)
30 CALL SPGET("GREEN",B2$)
40 PRINT #1:B1$
50 PRINT #1:B2$
60 CLOSE #1
70 END
```

and following is a program to run the speech data from tape:

```
100 OPEN #1:"CS1",INTERNAL,INPUT,FIXED
192
110 INPUT #1:B1$
120 INPUT #1:B2$
130 CALL SCREEN(7)
140 CALL SAY("THIS IS",B1$)
150 FOR I=1 TO 500 :: NEXT I
160 CALL SCREEN(13)
170 CALL SAY("THIS IS",B2$)
180 GOTO 180
```

Following is a program to display speech data on the screen:

```
100 REM HEX DUMP OF SPEECH DATA
110 CALL CLEAR
120 INPUT "TYPE WORD: ":WORD$
130 CALL SAY(WORD$)
140 CALL SPGET(WORD$,R$)
150 HEX$="0123456789ABCDEF"
160 L=LEN(R$)
170 PRINT "LENGTH=";"BYTES":
180 FOR I= 1 TO L
190 DEC=ASC(SEG$(R$,I,1))
200 HIGH=INT(DEC/16)
210 LOW=DEC-16*HIGH
220 HIGH=HIGH+1
230 LOW=LOW+1
240 PRINT SEG$(HEX$,HIGH,1);
250 PRINT SEG$(HEX,LOW,1);
260 IF I/10(INT(I/10) THEN 280
270 PRINT
280 NEXT I
290 PRINT : :
300 GOTO 120
```

Finally, in appendix M of the XBASIC manual is a description of how to add some suffixes onto speech words. This is a good description of how SPGET is used and how the speech data can be taylored and combined with other speech data.

The majority of the information presented in this discussion on speech was derived from the SPEECH EDITOR CM MANUAL.>>>>>>>END<<<<<<<

News Release

Page Pro Page Composer Released

Asgard Software is pleased to announce *Page Pro Page Composer*, a new graphics utility for owners of *Page Pro 99*.

Page Composer is designed to allow you to prepare and print multi-page documents with *Page Pro 99*. With *Page Composer*, for the first time it's easy to generate newsletters and reports. *Page Composer* also allows you to create pages that you can create with no other utility for the TI-99/4A or the Geneve, and does things that no other program can do, including:

- Create documents containing up to 999 pages
- Pages can be oriented either landscape (sideways) or portrait (normal)
- Pages can be a variety of different printer resolutions - the 60 dots/inch mode of *Page Pro 99*, 80 dots/inch and 120 dots/inch - allowing you to create 60, 80 and 120 column pages
- You can place up to 30 pictures of any size on each page, and because of the increased printer resolution, you can use even the largest Macintosh and PC pictures
- Pictures can be "opaque" or "transparent" - you can merge pictures together
- When printing your document, you can print any range of pages and up to 999 copies

Using the powerful features of this program, you can easily create half-page "booklet-style" newsletters on landscape pages - ready to be copied. In fact, you can easily put two full-size *Page Pro 99* pages on a single *Page Composer* page! The possibilities are endless.

Not only is *Page Composer* powerful, it's also easy-to-use. It features a unique "graphical user interface" where you select program functions by selecting a picture that represents the function, move your screen window around a page by selecting arrows and "scroll bars", enter information by selecting "buttons" in pop-up windows, and so on. In fact, the program is designed so that the only time you have to touch the keyboard is when you enter a disk or filename, or a page number!

Page Composer requires a disk system, 32K and at least one DS/SD disk drive. Two disk drives are highly recommend. An Epson or compatible printer is also required. The program is compatible with the HFDC, the Horizon RAM-disks, the Myarc Geneve 9640, and the TIM, Dijit and Mechatronics 80-column cards. It works with a joystick, an *Asgard Mouse* or a *Myarc Mouse*.

Suggested retail price $14.95. Send all orders (plus $3.00 for S&H) or inquiries to:

Asgard Software • P.O. Box 10306 • Rockville, MD 20859-0306

NEXT MEETING TUESDAY, JANUARY 12, 1993 HAPPY NEW YEAR!!!

MUNCH OFFICERS AND NUMBERS (all in 508 area unless noted)

President	W.C. Wyman		865/1213
Vice President	Bruce Willard		852/3250
Secretary	Jim Cox		
Treasurer	Jim Cox		869-2704
Acting Editor	Jim Cox		
Adv.Prog. Chair	Dan Rogers		248-5502
Library	OPEN		
Disk Librarian	Lou Holmes	617	965/3584
Tape Librarian	Walter Nowak	413	436/7675
NEW-AGE/99	Jack Sughrue		476/7630

MUNCH DUES

NEW MEMBERSHIP	$25.00
RENEWAL MEMBERSHIP	$15.00
NEWSLETTER ONLY SUBSCRIPTION	$13.00

DECEMBER MEETING. There were 15 members present at this meeting, Bruce, Lou and Sal were some old friends who were able to come this month. We played the game Backenstein, Lou helped out with some graphics demos and Corson also pitched in. Jack was ill, we hope he is better. Stan won the raffle.

JANUARY MEETING. This month I hope Jack will get a chance to demo something new for us. I believe he told me he had something planned for this month, but I forgot what it was; what's new! I am sure the room will be warm and the meeting interesting.

RAFFLE. Every month we have a raffle to help defer the rental cost of our meeting hall. A typical raffle will have game and utility programs T-Shirts, books, bumper stickers, blank discs and all sorts of odds and ends for the T.I.

LIBRARY NOTICE. Please return any items borrowed from our library. If you can not come to a meeting or give these items to someone who will be at the meeting.

REPRINTS. Reprints are permitted as long as credit is given to M.U.N.C.H.

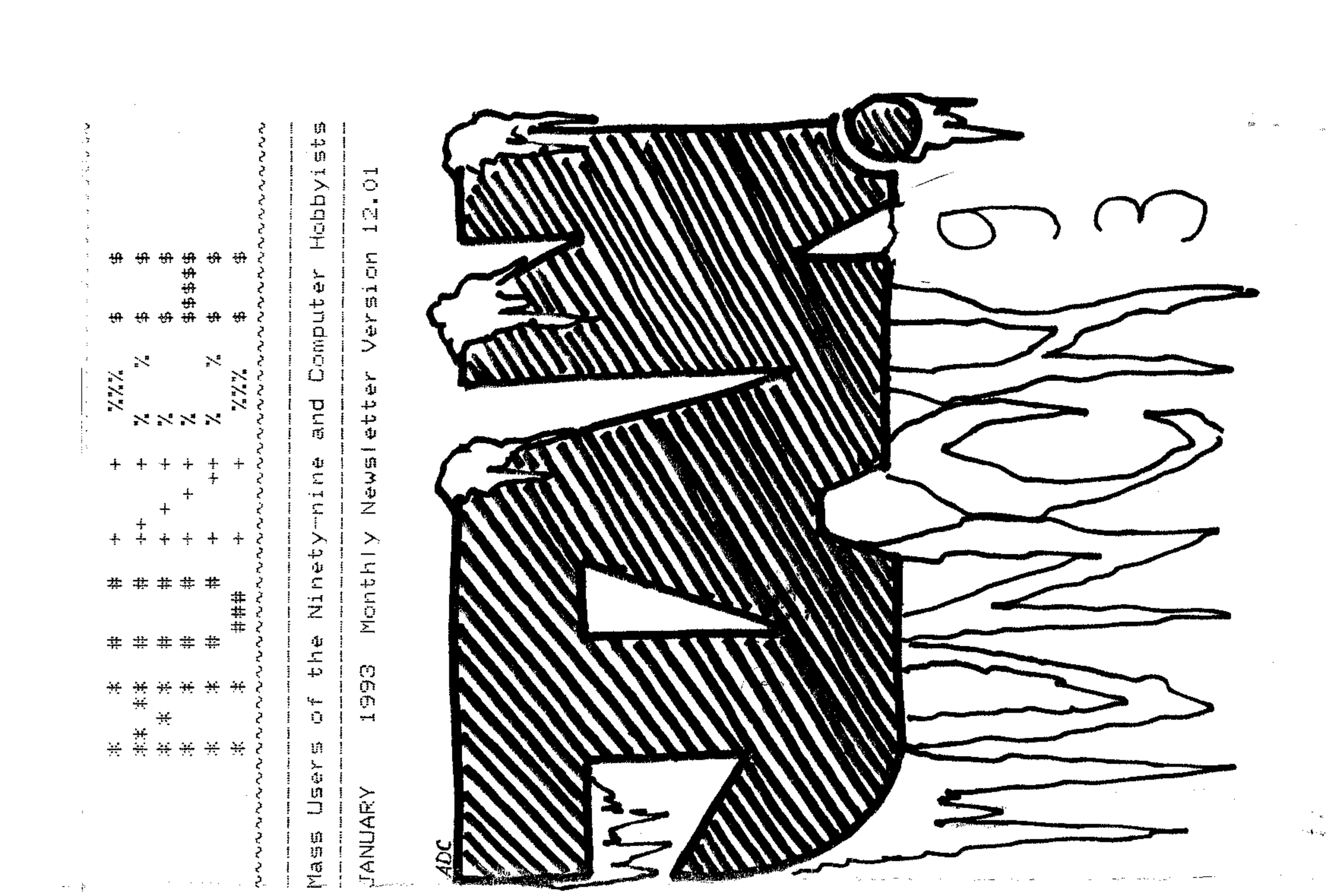
ARTICLES. I am always looking for articles for this newsletter, anything which interests you will probably interest other members of the TI community, so please share your ideas and opinions with all of us.

DISK LIBRARY. The disk library will be at the meetings from now on. We have copies of all disks in the library and they are available to members for just $1.50 each for single discs, $2.00 flippies, $3.00 double discs and $4.00 double flippie.

FOR SALE. The group has a TI Count Business Software package available for sale. If interested contact Jim Cox at the above numer or the club address. We also have blank disks for sale. The price is $6.00 for a package of 25 disks. Bruce still has his sister-in-law's system for sale, call him for details.

DISK OF THE MONTH. I am not sure if I will have a disk for this month because I am having problems with my disk drives.
Fitness, Touch Typing Tutor and Tunnels of Doom.

THE MUNCH VIDEO is ready, members can purchase it for $5.00, plus $3.00 postage for mail orders.



NEW ADDRESS:

M.U.N.C.H.
C/O J.W. COX
905 EDGEBROOK DR.
BOYLSTON, MA. 01505

FIRST CLASS

Next Meeting JANUARY 12th.

POSTMASTER: Forwarding and Address Correction Requested.