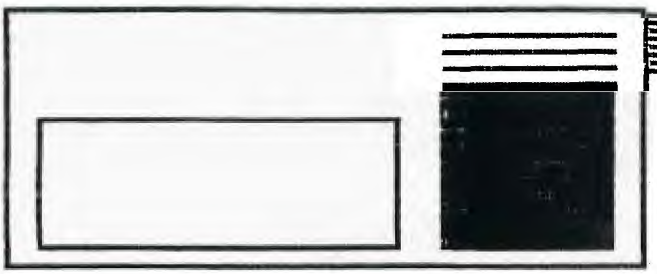
019
8709



THE DATA BUS

ISSUE :
VOL. 5
NO. 8
- - - -
S E P T
1 9 8 7

THE DELAWARE VALLEY USERS GROUP

DEDICATED TO THE TI AND COMPATIBLE HOME COMPUTER FAMILY

P.O. BOX 6240 STANTON BRANCH, WILMINGTON DE 19804

The Making Of The DATABUS
by Jim Folz

I spend a lot of time getting a newsletter put together. It could be argued that some of the things I do are not necessary but THEY WILL BE DONE before the newsletter goes out. Some of the contributors to the DATABUS ask how they can reduce this load. I am writing this so that 1) future contributors can set up their articles more completely, 2) other members can learn some things about TI Writer, and 3) users groups can get some understanding of what it takes to put out a newsletter.

At the September meeting, I will probably discuss TI Writer if there is interest. A newsletter editor becomes familiar with TI Writer out of necessity (a good way to learn). When I took on this job, I had just gotten my first printer and I knew NOTHING about TI Writer. I still have to laugh sometimes when I remember Jack Thorpe giving me the short course one Saturday afternoon. I am sure that he thought I knew LESS about TI Writer when we got done. Jack Thorpe was a newsletter editor, himself, and is an excellent source if you need help.

After Jack, Jack Shattuck became an editor. I am sure he could also show you a thing or two.

If anyone needs help with TI Writer, I suggest you contact me or one of the others.

HINTS

Please consider these points when putting together an article for the DATABUS.

* DATABUS columns are 48 characters wide by 77 lines long. In an effort to bring as much info to our members as possible, we settled on this column size assuming Near Letter Quality letters and 74 percent reduction.

* To prevent copy placement near the edge of the paper and to allow notes in the margins, the 48x77 column is placed in the "center" of the workspace. (Actually, the file started as 50x77 and I never bothered to change my standard Tab settings.) In the editor, you set up the Tabs by typing T and enter. I put an L at 14 and a R at 62. I don't use the others so I put periods everywhere else. This puts a 15 character margin on the left and a 17 character margin on the right. Using 62 allows you to put in the 48th character without autowrap. When done using arrows and windows to get around you exit Tabs by pressing enter.

* Start all files as follows:

```
.LM 14;RM 61;PL 85;FI;AD;LS 1;IN +0(c/r)
.TL 91:27,66,4(c/r)
.TL 93:27,66,5(c/r)
@Title(c/r)
[by Author, Users Group (if not DVUG)(c/r)
.SP;IN +5(c/r)
     Text
```

LM, RM - Left and Right Margins
PL - Number of lines to page plus top margin (8)
FI, AD - Fill and Adjust Commands
LS - Line Space (Single Spacing)
IN - Indent from Left Margin-0 for Title/Author
TL - Transliterates [as Near Letter Quality on and] as Near Letter Quality off

* Put a (c/r) at the end of each paragraph to prevent accidental Reformatting.

* Put a required space (^) after the period and before the capitalized letter of the next sentence if reducing the spacing there will allow you to place more characters on that line.

* Use .SP;IN +10(c/r) before a program listing and .SP;IN +5(c/r) after the listing. Since programs have 28 characters per line, it helps people to type in programs when the listings are provided in this form. Each line should have 28 characters in it and end in a (c/r).

* Since most people write the program first, listings are usually available in program form first and are converted into DIS/VAR 80 files later (usually by listing the program to a disk drive). Please provide the program in runnable form as well. This version will accompany the article into the Software Library and will eventually go onto TIBBS.

* Replace & and @ with double characters. Avoid using asterisks followed by numbers.

Believe me I know this is a lot of work. You might think that this is all there is to it. Things get really wild when you throw in a few assembly programs, tables and figures. If you have any questions, catch up with me at the meeting. Until then ...

DVUG EXECUTIVE COMMITTEE MEMBERS IN 1987

PRESIDENT .....TOM AUGUST
VICE PRESIDENT .....JIM DAVIS
SECRETARY .....TIM EVERS
TREASURER .....TOM KLEIN
SGT. AT ARMS .....JIM FOLZ
DELMARVA CHAPTER CHR .....CHARLES BOWER
SO.JERSEY CHAPTER CHR .....TONY DIFEBBO
SHORE CHAPTER CHR .....HARVEY ADAMS

NORMAL MEETING SCHEDULE

CHRISTIANA GROUP	4th Thursday	6:30-9:30
DELMARVA CHAPTER	2nd Monday	7:00-9:00
SOUTH JERSEY CHAPTER	3rd Monday	6:45-9:00
SHORE CHAPTER	1st Thursday	7:30-9:00

MEETING PLACES

CHRISTIANA GROUP: Delaware's Christiana Mall on Rte. 7, at I-95 Exit 4-S. We meet in the Community Room. Enter between J. C. Penney and Liberty Travel inside the Mall.

DELMARVA CHAPTER: Kent County Courthouse, Basement Conference Rm #2S, Green & State Sts, Dover, De. Use the Green St. side entrance.

SOUTH JERSEY CHAPTER: Deptford Municipal Bldg, Cooper Ave. and Delsea Drive, (Rtes. 534 & 47), in Gloucester County. Enter and park in rear of the building.

SHORE CHAPTER: Scullville Firehouse #1, County Rte. 559 (on left, between mile markers 4 and 3), in Atlantic County. Ignore Station #2 on right enroute.

DVUG BULLETIN BOARDS

(302)322-3999	Anytime
(302)674-1449	Anytime
(609)429-7792	Monday-Thursday 3:00 PM-7:00 AM
	Friday 3:00 PM-Monday 7:00 AM

For general information, you may contact

TOM KLEIN	Pa.	(215)494-1372
JIM FOLZ	Del.	(302)995-6848
BUTCH FISHER	N.J.	(609)783-8276
GUS LEWIS	N.J.	(609)927-5601

Delaware Valley Users Group membership includes: library and software privileges, monthly DATABUS newsletter, plus other special benefits. Annual membership rates are: Family or Individual $15; Student $10; Newsletter only (beyond 75 mi) $10.

TRANSMIT YOUR NEWSLETTER COPY TO: The Data Bus Editor --- Jim Folz, Telephone (302)995-6848, or use the DVUG mailing address shown on Page One. PLEASE SUBMIT NEWSLETTER ARTICLES FOR AN ISSUE BEFORE THE 2ND THURSDAY OF EACH MONTH.

DVUG ADVERTISING RATES FOR THE DATA BUS:

1/4 page - $ 5/issue, or $ 45/12 issues
1/2 page - $ 8/issue, or $ 75/12 issues
Full page - $15/issue, or $125/12 issues

FIVE LINER CONTEST
by Jim Folz

These entries have been received for the second five-liner competition. (Unfortunately, I was unable to get mine done in time.) The winner will be announced at the next meeting.

Loan Payment

```
100 CALL CLEAR :: PRINT "CAL
CULATE LOAN PAYMENT" :: INPU
T "Enter Principle $":P :: C
ALL CLEAR :: INPUT "Enter #
of payments per YEAR":N
200 CALL CLEAR :: INPUT "Ent
er Whole Number of YEARS":Y
:: CALL CLEAR :: INPUT "Ente
r Number of Payments  Beyo
nd Last Whole YEAR ":M
300 CALL CLEAR :: INPUT "Ent
er Interest RATE":I :: RATE=
I/N/100 :: R=P*RATE/(1-(RATE
+1)^-(N*Y+M)) :: CALL CLEAR
:: CALL SCREEN(14)
400 DISPLAY AT(7,1):"MONTHLY
 PAYMENT = $" :: DISPLAY AT(
7,20):USING "######.##":R ::
 PRINT "PRINCIPAL =",P :: PR
INT "ANNUAL RATE =",I
500 PRINT "# OF PAYMENTS",N*
Y+M :: PRINT " " :: INPUT "C
ALCULATE ANOTHER LOAN?(Y/N)
":ANS$ :: CALL SCREEN(8) ::
IF ANS$="Y" THEN 100 :: END
```

Screen Display

```
110 DEF R=(RND-.5)20 :: CAL
L CLEAR :: CALL SPRITE(#1,48
,5,192,1,#2,42,7,96,128,R,R)
:: J=7
120 CALL JOYST(1,X,Y):: GOSU
B 140 :: U=U+X :: V=V+Y :: C
ALL MOTION(#1,-V,U):: GOSUB
140 :: S=S+1 :: DISPLAY AT(2
4,2):S :: I=I+1 :: CALL DIST
ANCE(#1,#2,D):: CALL SOUND(-
10,SQR(D)+110,4):: GOSUB 140
125 J=J+1 :: CALL SCREEN(J):
: IF J=12 THEN J=7
130 GOSUB 140 :: IF I=10 THE
N I=0 :: CALL MOTION(#2,R,R)
:: GOTO 120 ELSE GOTO 120
140 CALL COINC(ALL,C):: IF C
 THEN STOP ELSE RETURN
```

For Sale: TI Console and Cassette Recorder - $50
David Porter
3111 Winterhaven Dr.
Newark, De. 19702
(302) 737-6852

CONTENTS OF THE SEPTEMBER ISSUE OF THE DATA BUS:

BASIC/XBASIC Programming Techniques
by Jack Shattuck Phone: (302) 764-8619

PRINTER PRATTLE: Setting, Changing Printer Codes

Several items prompt this month's topic for discussion. A recent visit to a friend included an exchange of files, after which he wanted to catalogue the disk to see its contents. Seeking to print the data on as small size paper as was possible, he wanted to use condensed print, with 88 lines per page (1/8" space between lines) but had momentarily forgotten the Escape codes.

He suggested I incorporate them into the printer set-up routine for DM1000, which he uses to catalog disks, so it wouldn't be forgotten in the future.

If you've been getting copies of DM1000 for running the program lately, you probably didn't get the documentation for its use. I last saw those DM-HELP files on Version 2.2 (which is in our DVUG Software Library, by the way). If you go to the File Utility section, you'll see some discussion by displaying the DV/80 file MISCUTIL accompanying Version 2.2. (Or either DISKUTIL or FILEUTIL references, if you prefer.)

Anyway, the procedure for setting a printer control code with DM1000 is to do <FCTN 3> from the main menu, identify your printer name, set a desired control (Escape) code, and save it under the name DSK1.TEST. You won't need punctuation, only the appropriate numbers.

I mention this latter item because the User Manual for the Gemini 10X/15X has a typo on Page 70, wherein it says to use ESC$;"A";n to set the line spacing. As shown in that Manual's sample program (page 63, line 350), it should have read ESC$: (note colon not semi-colon) then "A";n. It took us a while to figure that out.

Last month's DATA BUS had an example by Jim Folz of higher character sets, or perhaps hidden character sets, on the Epson printer. The Epson printer is the industry standard, and was chosen by TI as well as IBM as their choice in 1981. In the case of IBM, for graphics; for TI, to use as a replacement for the Thermal Printer ("TP").

In those days, selling price for the Epson MX-80 was between $750-800. Subsequent models - FX, RX, etc. have maintained popularity for the home computer as either additional features were added, or prices dropped. Star Micronics' Gemini 10 was the first to bring prices within reach at half the cost of an Epson and even more features while being "Epson compatible" - i.e., using the same printer codes. MX-80 and Gemini 10's still are in use today, and other Epson-compatibles, such as the Panasonic, have joined the TI/99/4A market as well.

Another powerful workhorse has been the NEC 8023A-C, and C.Itoh Prowriter, which some of the TI software authors place third behind the other top two in popularity. An inverted code system in the dot matrix configuration hindered program conversion in some cases, although seems to have caused no problem for Dave Rose (CSGD author), Great Lakes Software or Quality 99 Software, as they turn out wonderful graphic programs and the essential screen dump routines for that group of printer users. (Extended Software was an early Prowriter-compatible software source, as well.)

Among features favored by the NEC/Prowriter users were an adjustable tractor, or "pin" feed, a Proportional character set, and a reverse line feed, which were only available (if available at all) on Epson models for extra cost. The reverse feed was used for THE DATA BUS logo during Jan. 1985 - Aug. 1986, when DVUG newsletters were run from a NEC 8023A-C (otherwise on a Gemini 10).

With continued upgrade needs, older printer versions reappear on the market, joined by some other newer popular models such as an Okidata 92 and other Japanese imports and offshoots (Axiom, the Seikosha GP100A, for instance).

Among the Okidata advantages: a Near Letter Quality (Correspondence) mode, for Pica or Elite printing, ability to accept an additional down-loaded or programmed character set, and a faster print speed (when not in NLQ mode). Price was also an attractive feature.

(There is now a chip available for the NEC/Prowriter to provide Near Letter Quality. It is available from MicroAge Computers, near Corning, N.Y. at a cost of $35 or $45 (including s/h) for different chips. Write HOUSE OF HARDWARE, RD#1, Box 227, Burdett, NY 14818, or call Laurie or Leeann at 607-936-3053. They accept either check or credit card. NLQ mode replaces Proportional printing on those models, which is close to NLQ anyway, the more you compress it. The advantage probably would be its use in the Editor mode of TI Writer, or other occasions so you don't have to worry about a ragged right format such as the unpredictable Proportional printing offers.)

The newer users of older printer versions have had difficulty in understanding the awkward printer manuals for print mode configurations so here's a convenient set-up routine for two such popular non-Epson brands, the NEC/Prowriter and Okidata 92 (see listing on adjacent page to this article).

In the programs, choices selected are shown by the moving Cursor (character 30), called by a VCHAR statement. The Nec/Prowriter program was originally written in BASIC, as I've used it for four years. This XB version is slightly quicker at start-up, and the multi-statement lines are more convenient for this newsletter.

The Okidata program has more complications because the NLQ (Correspondence) mode won't work on condensed print. To save program lines, I've used a CALL GCHAR routine (Line 280) to find out what status print applies (i.e., what the cursor currently shows). CALL GCHAR is only available in XB, not BASIC.

A note on definitions, since some printers' manuals vary in their use of terms:

Condensed = compressed type (17 characters to the inch, or 136/line).

NLQ = correspondence mode on the Oki, as we discussed above.

Bold Face = emphasized type, printed by the printing of a second impression, 1/2 dot over to right. This is contrasted with a double strike, or enhanced type, created by a second line being printed 1/3 of a line lower. I've used only one version of double print, the former (Bold Face), in these programs. (Okidata can do both.)

Underlining is NOT underscoring. To obtain true underlining using TI Writer, you'll have to add a transliteration in the Format mode, or set your printer as needed (which is what is done by my program). UnderSCORING is a broken dash. You can see it by typing the _ <FCTN U> on your TI keyboard. Note it is really 0000000000000FF as a defined character, which is to say, it takes a separate line when printing.

P.S. I have Epson/Gemini versions too, but I assume those codes are easily available.

Program Listing

```
100 REM PRINTER PROGRAM IN
        XB FOR NEC/PROWRITER
        BY JACK SHATTUCK
110 DISPLAY AT(1,1)ERASE ALL
:"NEC/PROWRITER MODE SELECTO
R:----------------------------
--"
120 OPEN #1:"PIO",VARIABLE 1
36
130 DISPLAY AT(5,1):"<A> NOR
MAL (PICA)    80 COLS.<B> ELI
TE            96 COLS.<C> CON
DENSED      136 COLS.<D> PRO
PORTIONAL"
140 DISPLAY AT(10,8):"SPECIA
L MODES:":"<E> BOLD FACE
  <H> CLEAR<F> WIDE TYPE
  <I> CLEAR<G> UNDERLINE
  <J> CLEAR"
150 DISPLAY AT(15,1):"<K> 66
 Lines/Page  <L> 88/pg<M> PR
INTER TEST":"<N> QUIT "
160 DISPLAY AT(22,4):"Press
Down Alpha Lock.":"PRINTER O
N LINE? PRESS ENTER"
170 CALL KEY(0,K,S):: IF K<>
13 THEN 170
180 CALL HCHAR(22,4,32,59)::
DISPLAY AT(22,8):"CURRENT MO
DE IS:"        INDICATED ABO
VE" :: GOSUB 390 :: GOTO 230
190 CALL KEY(3,K,S):: IF (K<
65)+(K>78)THEN 190
200 ON K-64 GOTO 250,260,270
,280,290,310,330,300,320,340
,350,360,450,370
210 REM ..... "N","E","Q"
,"P",BF.,WT.,UN.,XBF,XWT,XUN
,66L,88L,PIO,QUIT
220 CALL VCHAR(11,25,30,3)
230 CALL VCHAR(5,6,30):: CAL
L VCHAR(11,25,32,3):: CALL V
CHAR(15,6,30):: PRINT #1:CHR
$(27)&"N" :: PRINT #1:CHR$(2
7)&CHR$(34);
240 PRINT #1:CHR$(15);:: PRI
NT #1:CHR$(27)&"Y";:: PRINT
#1:CHR$(27)&"A";:: GOTO 190
!LINES 220-240 FOR START-UP
250 CALL VCHAR(5,6,32,4):: C
ALL VCHAR(5,6,30):: PRINT #1
:CHR$(27)&"N";:: GOTO 190
260 CALL VCHAR(5,6,32,4):: C
ALL VCHAR(6,6,30):: PRINT #1
:CHR$(27)&"E";:: GOTO 190
270 CALL VCHAR(5,6,32,4):: C
ALL VCHAR(7,6,30):: PRINT #1
:CHR$(27)&"Q";:: GOTO 190
280 CALL VCHAR(5,6,32,4):: C
ALL VCHAR(8,6,30):: PRINT #1
:CHR$(27)&"P";:: GOTO 190
290 CALL VCHAR(11,6,30):: PR
INT #1:CHR$(27)&CHR$(33);::
GOTO 190
300 CALL VCHAR(11,6,32):: CA
LL VCHAR(11,25,30):: PRINT #
1:CHR$(27)&CHR$(34);:: CALL
VCHAR(11,25,32):: GOTO 190
310 CALL VCHAR(12,6,30):: PR
INT #1:CHR$(14);:: GOTO 190
320 CALL VCHAR(12,6,32):: CA
LL VCHAR(12,25,30):: PRINT #
1:CHR$(15);:: CALL VCHAR (12
,25,32):: GOTO 190
330 CALL VCHAR(13,6,30):: PR
INT #1:CHR$(27)&"X";:: GOTO
190
340 CALL VCHAR(13,6,32):: CA
LL VCHAR(13,25,30):: PRINT #
1:CHR$(27)&"Y";:: CALL VCHAR
(13,25,32):: GOTO 190
350 CALL VCHAR(15,6,30):: CA
LL VCHAR(15,25,32):: PRINT #
1:CHR$(27)&"A";:: GOTO 190
360 CALL VCHAR(15,25,30):: C
ALL VCHAR(15,6,32):: PRINT #
1:CHR$(27)&"8";:: GOTO 190
370 CALL HCHAR(17,6,30)
380 END
390 DATA MAKE YOUR CHOICE(S)
 -
400 READ M$
410 FOR I=1 TO LEN(M$)
420 CALL HCHAR(3,6+I,ASC(SEG
$(M$,I,1)))
430 NEXT I
440 RETURN
450 CALL VCHAR(16,6,30)
460 PRINT #1:"ABCDEFGHIJKLMN
OPQRSTUVWXYZ1234567890abcdef
ghijklmnopqrstuvwxyz!@#$%^&*
()+=~[]_?'""-/|{};\'<,>."
470 CALL VCHAR(16,6,32):: GO
TO 190

100 REM PRINTER PROGRAM IN
        XB FOR OKIDATA 92
        BY JACK SHATTUCK
110 DISPLAY AT(1,1)ERASE ALL
:"OKIDATA PRINT MODE SELECTO
R:----------------------------
--"
120 OPEN #1:"PIO",VARIABLE 1
36
130 DISPLAY AT(5,1):"<A> NOR
MAL (PICA)    80 COLS.<B> ELI
TE            96 COLS.<C> CON
DENSED      136 COLS.": :"
     SPECIAL MODES:"
140 DISPLAY AT(10,1):"<D> CO
RRESPONDENCE <H> CLEAR<E> BO
LD FACE      <I> CLEAR<F> WI
DE TYPE      <J> CLEAR<G> UN
DERLINE      <K> CLEAR"
150 DISPLAY AT(15,1):"<L> 66
 Lines/Page  <M> 88/pg<N> PR
INTER TEST":"<O> QUIT "
160 DISPLAY AT(22,4):"Press
Down Alpha Lock.":"PRINTER O
N LINE? PRESS ENTER"
170 CALL KEY(0,K,S):: IF K<>
13 THEN 170
180 CALL HCHAR(22,4,32,59)::
DISPLAY AT(22,8):"CURRENT MO
DE IS:" INDICATED ABOVE" ::
GOSUB 390 :: GOTO 230
190 CALL KEY(3,K,S):: IF (K<
65)+(K>79)THEN 190
200 ON K-64 GOTO 250,260,270
,280,290,310,330,490,300,320
,340,350,360,450,370
210 REM ..... "N","E","Q"
,NLQ,BF.,WT.,UN.,XLQ,XBF,XWT
,XUN,66L,88L,PIO,QUIT
220 CALL VCHAR(11,25,30,3)
230 CALL VCHAR(5,6,30)::CALL
 VCHAR(11,25,32,3):: CALL VC
HAR(15,6,30):: PRINT #1:CHR$
(24);:: GOTO 190
240 !LINES 220-230 FOR START
-UP
250 CALL VCHAR(5,6,32,4):: C
ALL VCHAR(5,6,30):: PRINT #1
:CHR$(30);:: GOTO 190
260 CALL VCHAR(5,6,32,4):: C
ALL VCHAR(6,6,30):: PRINT #1
:CHR$(28);:: GOTO 190
270 CALL VCHAR(5,6,32,6):: C
ALL VCHAR(7,6,30):: PRINT #1
:CHR$(27)&CHR$(48);:PRINT #
1:CHR$(29);:GOTO 190
280 CALL GCHAR(7,6,X)::IF X=
30 THEN 480::CALL VCHAR(10,6
,30)::PRINT #1:CHR$(27)&CHR$
(49);:GOTO 190
290 CALL VCHAR(11,6,30):: PR
INT #1:CHR$(27)&CHR$(84);:G
OTO 190
300 CALL VCHAR(11,6,32):: CA
LL VCHAR(11,25,30):: PRINT #
1:CHR$(27)&CHR$(73);:CALL V
CHAR(11,25,32):: GOTO 190
310 CALL VCHAR(12,6,30):: PR
INT #1:CHR$(31);:: GOTO 190
320 CALL VCHAR(12,6,32):: CA
LL VCHAR(12,25,30):: PRINT #
1:CHR$(30);:: CALL VCHAR(12,
25,32):: GOTO 190
330 CALL VCHAR(13,6,30)::PRI
NT #1:CHR$(27)&CHR$(67);::GO
TO 190
340 CALL VCHAR(13,6,32):: CA
LL VCHAR(13,25,30):: PRINT #
1:CHR$(27)&CHR$(68);::CALL V
CHAR(13,25,32):: GOTO 190
350 CALL VCHAR(15,6,30):: CA
LL VCHAR(15,25,32):: PRINT #
1:CHR$(27)&"6";:: GOTO 190
360 CALL VCHAR(15,25,30):: C
ALL VCHAR(15,6,32):: PRINT #
1:CHR$(27)&"8";:: GOTO 190
370 CALL HCHAR(17,6,30)
380 END
390 DATA MAKE YOUR CHOICE(S)
 -
400 READ M$
410 FOR I=1 TO LEN(M$)
420 CALL HCHAR(3,6+I,ASC(SEG
$(M$,I,1)))
430 NEXT I
440 RETURN
450 CALL VCHAR(16,6,30)
460 PRINT #1:"ABCDEFGHIJKLMN
OPQRSTUVWXYZ1234567890abcdef
ghijklmnopqrstuvwxyz!@#$%^&*
()+=~[]_?'""-/|{};\'<,>."
470 CALL VCHAR(16,6,32):: GO
TO 190
480 FOR Z=1 TO 35::DISPLAY A
T(10,5):"Can't when Condense
d"::NEXT Z::DISPLAY AT(10,5)
:"CORRESPONDENCE <H> CLEAR":
:GOTO 190
490 CALL VCHAR(10,6,32)::CAL
L VCHAR(10,25,30)::PRINT #1:
CHR$(27)&CHR$(48);::CALL VCH
AR(10,25,32)::GOTO 190
```

Bits 'n' bots - Part 3
by Jim Davis and Jim Folz

This part of the series on computer control of various devices deals with the stepper motor drive circuit and simple BASIC software to test it. Part 2 (DATABUS - 8/87) explained stepper motor function and Part 1 (DATABUS - 7/87) discussed the choice of the parallel port.

CIRCUIT:

The circuit was designed to use parts (except for the motor) which are readily available, i.e. available from Radio Shack. It was designed for a "unipolar" stepper motor.

MOSFET power transistors are used to switch current in the motor windings. Since the "on" resistance is low, no heatsink is required and construction is simplified. A diode is used to protect the MOSFET from the inductive "kick" that occurs when the current thru the motor winding is turned off. Best speed performance is achieved when the diode is attached to a zener diode whose voltage is equal to the motor supply voltage. A zener is not used here because the step rate is slow and a little money is saved.

The MOSFET used here needs 6 volts on the gate to turn "on", i.e. pass 3 amperes at a minimum voltage drop. The voltage from the PIO parallel port is "TTL" compatable, meaning its logic high state is guaranteed only to be more than 2.4 volts. Thus a level convertor is required. We could have used a high voltage open collector hex buffer (SN7407 TTL integrated circuit) to accomplish this function, but a simple transistor switch is cheap and uses parts that otherwise would be left over.

The level convertor is also used to protect the motor from normal misuse. When the computer is first turned on, the PIO port is in the INPUT mode and cannot supply current. This circuit turns "off" all MOSFETs in this condition so that the motor does not overheat. Also, if the cable to the computer is not plugged in, the circuit protects the motor. Unfortunately, the logic sense is inverted, so we need to pay extra attention to the software codes. The resistors were chosen for a 12 volt supply. Other voltages may be used with resistor values from the table.

Voltage	5	12	16	24
R1	10K	22K	33K	47K
R4	NONE	NONE	NONE	22K

Ordinarily, it is necessary to use a latch to hold the pattern for the motor phases. Fortunately, the TI RS232 module does that for us. Thus we can use data lines to directly drive our circuit.

Finally, a handshake circuit is needed. A simple transistor inverter which is powered from the RS232 module is sufficient. In the output mode, the handshake output is normally high. When the new data is ready on the data lines, the handshake output goes low. This condition remains (and the computer is unable to do any other processing) until the handshake input goes from low to high.

SOFTWARE

The first thing in the BASIC program is to initialize or establish the PIO port. This is done with an "open" statement. In addition, we wish to use some options. Often such options are selected by mechanical switch settings. TI uses "software" switches to "configure" the hardware. We do not want the normal carriage return-line feed, otherwise the last character outputted would always be the linefeed and that is what the latch would hold, not the motor phase pattern. This software switch (.CR) is described in the RS232 instruction manual.

A simple test program for checking (with a voltmeter) the handshake circuit and the individual data lines is:

```
100 OPEN #1:"PIO.CR"
110 INPUT X
120 PRINT #1:CHR$(X)
130 GOTO 110
```

So, if we type in a value for X of zero, pins 2 through 9 will have less than 0.4 volts relative to ground (pin 11). A value of 15 for X will give a voltage greater than 2.4.

DATA PIN	DATA VALUE	X=0	1	2	4	8	7	11	13	14	15
2	1	0	1	0	0	0	1	1	1	0	1
3	2	0	0	1	0	0	1	1	0	1	1
4	4	0	0	0	1	0	1	0	1	1	1
5	8	0	0	0	0	1	0	1	1	1	1
6	16	0	0	0	0	0	0	0	0	0	0
7	32	0	0	0	0	0	0	0	0	0	0
8	64	0	0	0	0	0	0	0	0	0	0
9	128	0	0	0	0	0	0	0	0	0	0

We need to change the motor codes from that described in the previous articles because of the logic sense inversion. The principles are as before and use codes from the table above. We will use string variables to hold the various motor phase codes.

CODE	DATA PATTERN	MOTOR PATTERN
A$=CHR$(14)	1 1 1 0	0 0 0 1
B$=CHR$(11)	1 0 1 1	0 1 0 0
C$=CHR$(13)	1 1 0 1	0 0 1 0
D$=CHR$(07)	0 1 1 1	1 0 0 0

To make the motor move forward one electrical cycle from phase pattern A, we print B$, C$, D$, A$. To make the motor move backward one electrical cycle, we print D$, C$, B$, A$. A simple BASIC program to step the motor forward is:

```
100 OPEN #1:"PIO.CR"
110 A$= CHR$(14)
111 B$= CHR$(11)
112 C$= CHR$(13)
113 D$= CHR$(07)
119 PRINT #1:A$
120 FOR I= 1 TO 100
130 PRINT #1:B$
140 PRINT #1:C$
150 PRINT #1:D$
160 PRINT #1:A$
170 NEXT I
180 PRINT #1:CHR$(0)
```

Motor speed is limited by the speed of the BASIC language interpreter. For the next article, we will work on stepping faster, how to ramp the speed of the motor (required for moving large masses) and perhaps a bipolar drive circuit (since most of the surplus motor seem to be 4 wire or bipolar).

CONTINUED ON PAGE 10

Error Check For XBASIC Program Entry

by Tom Freeman

Have you ever typed in a TI 99/4A version of a BASIC program from a magazine, and noticed that the other versions have little numbers at the end of the lines that you don't have? They were for error checking on your typing, to insure no mistakes. Have you ever, laboriously, typed in a long program and run it, only to find that it crashes, or doesn't work as it is supposed to, all because of a simple typing error that you can't find? So why doesn't TI have one? NOW YOU DO!!

This may be the most useful program that I have published for general use because almost everyone does BASIC programs at one time or another. It involves only one extra step for the programmer, and one for the user who is typing the published program in. It is really a rather simple method and depends on the manner in which TI stores BASIC programs. Please note, however, that it requires a memory expansion and disk drive, and works only in Extended Basic (although BASIC programs can be entered in XBASIC, saved, and then run in BASIC).

You may remember the format in which "MERGE" type programs are stored on disk. If you don't, see our article a couple of months back on the various formats in which programs are stored. The MERGE format is actually a duplicate of the way in which the actual program is stored in memory or on disk, the difference being that it is a display type file, with each record starting with two bytes for the line number, and then the actual program line. In memory, however, the program lines are stored contiguously and in seemingly random order (actually the order depends on the order in which they were entered). A separate line number table is stored below the program area and keeps track of the line numbers and pointers to where each line begins. Now each line consists of one byte "tokens" for all reserved words (see the list I published last month) with all strings, including the names of subprograms such as LOAD, SCREEN, etc., being spelled out directly.

When you enter any line in XBasic (either a command, or a program line with the line number coming first), it is first moved to the so-called "Edit Buffer" at address >8C0 in VDP. The BASIC bias is preserved. The purpose of this is that if you press FCTN 8 (REDO) then the whole line or lines can be retrieved. Next, everything is "crunched" by replacing each reserved word with its token, subtracting the BASIC bias from strings, computing their length etc., and placing the result in the "Crunch Buffer" at >820 in VDP. Once it is there, it can be transferred to the appropriate place in memory expansion. This is the area that is used when my program computes the "checksum" by merely adding the value of each byte! The number is never allowed to go over hex >FF - the high byte is ignored (thus, in decimal, no number over 255). The assumption is that it is extremely unlikely, probability approaching zero, that a small number of mistakes will result in a number that differs by exactly 256, or a multiple thereof. The one exception is if you transpose two characters - there's nothing I can do about that!

Now what does the programmer do? First, his program must be completely debugged as no changes can be made after the checksums are computed or they will, of course, differ. Next, he saves his program in merge format. Now the following program must be run on the result.

```
100 !CREATE CHECKSUMS FOR XB
ASIC PROGRAMS, BY TOM FREEMA
N, LA 99'ERS !250
110 !SHOULD BE USED TOGETHER
 WITH "CHECK" ASSEMBLY FILE
THAT WILL PRINT CHECKSUMS ON
 SCREEN !099
120 DISPLAY AT(2,1)ERASE ALL
:"CREATE CHECKSUMS FOR XBASI
C ERROR CHECKING": :"    by
Tom Freeman" !085
130 DISPLAY AT(10,1):"INPUT
MERGE FILE?":" DSK1." !007
140 DISPLAY AT(13,1):"OUTPUT
 MERGE FILE?":" DSK1." !108
150 ACCEPT AT(11,3)SIZE(-15)
BEEP:I$ :: OPEN #1:I$,VARIAB
LE 163,INPUT !192
160 ACCEPT AT(14,3)SIZE(-15)
BEEP:O$ :: OPEN #2:O$,VARIAB
LE 163,OUTPUT !053
170 DISPLAY AT(20,1):"ANALYZ
ING LINE":"CHECKSUM IS " !01
4
180 LINPUT #1:A$ :: IF LEN(A
$)=2 THEN CLOSE #1 :: PRINT
#2:CHR$(255)&CHR$(255):: CLO
SE #2 :: STOP !115
190 Z=ASC(A$)*256+ASC(SEG$(A
$,2,1)):: DISPLAY AT(20,15)B
EEP:Z !141
200 B$=SEG$(A$,3,163):: L=LE
N(B$):: IF L>157 THEN 230 !1
62
210 N=0 :: FOR X=1 TO L :: Y
=ASC(SEG$(B$,X,1)):: N=N+Y :
: NEXT X :: N=N AND 255 :: N
$=STR$(N):: N$=RPT$("0",3-LE
N(N$))&N$ !088
220 DISPLAY AT(21,13)BEEP:N$
 :: PRINT #2:SEG$(A$,1,L+1)&
CHR$(131)&N$&CHR$(0):: GOTO
180 !252
230 DISPLAY AT(22,1)BEEP:"WA
RNING!":" LINE";Z;"IS TOO LO
NG!":"PRESS ANY KEY TO CONTI
NUE" !123
240 CALL KEY(0,K,S):: IF S=0
 THEN 240 ELSE PRINT #2:A$ :
: GOTO 180 !232
```

Notice the "!" and 3 numbers at the end of each line? The program was run on itself! Here is what happens. Each record of the merge file is read in, the first two bytes ignored (we don't need the line number) and the rest are added up. Next, the identical record is printed to the output file with the addition of the token for ! (remark) and the 3 characters of the checksum. This will work even if the program line already contained a remark (as in lines 100-110). THE USER MUST BE WARNED NOT TO TYPE THESE 4 CHARACTERS since they were not computed into the checksum. At the end (it may take a little while with a long program but it only needs to be run once), the programmer types NEW and merges in the output file, then saves it in normal mode, or lists it to printer, or whatever. This is the form to be published.

Now what the user must do is once type in the source code attached to the end of this article, and assemble it (a CALL LOAD version is also supplied for those who don't have the Editor/Assembler). If the object code created was called "CHECK" then he must type the following upon entry into XBasic: CALL INIT ::

Continued On Next Page

CALL LOAD("DSKx.CHECK") :: CALL LINK("CURSOR"). This one line with a line number can be saved on disk and then RUN each time it is needed, rather than type the whole line. What the assembly routine at CURSOR does is some housekeeping such as moving the numbers 0-9 to character sets 13-14, changing the colors there, redefining the cursor, putting up the title screen etc., and then turning on the user defined interrupt. Now at every VDP interrupt (each 1/60 second), the routine at CHECK begins. The interrupt can be turned off with CALL LINK("OFF") and back on with CALL LINK("ON") at any time, and the shape of the cursor will tell you which mode you're in. Now EVERY TIME you enter a new program line (and for some reason also after FCTN 8 - REDO even if no changes are made) the checksum will appear at the bottom of the screen and one extra line scrolled up. HERE IS THE KEY - IT SHOULD CORRESPOND TO THE ONE PUBLISHED THAT YOU ARE ATTEMPTING TO COPY IN. Hence, no errors!!!

I think the source code is sufficiently commented to explain what is going on. I must add that I spent many hours with Miller Graphics "EXPLORER", by Doug Warren, finding out WHAT is going on when you enter a line in XBasic. The address range in GROM of >6AA0 to >6AD8 should be broad enough to cover the various versions of XBasic out there since they differ by a few bytes here and there (the actual range needed in my module was >6AAE to >6ACA. This area contains the loop where the first key press on entry of a new line is located. As soon as the first key is pressed then the GROM code moves on. I needed this area so as to reset the flag that indicates the checksum has been printed in order to avoid having it printed again and again! Notice the fairly cumbersome method of peeking at the GROM address, which must then be reset since just looking at it destroys it! I discovered that the line number entered is saved at <u>both</u> >8304 and >834A and only when it is at both is the crunch buffer finished being filled with the crunched line. If you are entering a direct command, >8304 is not used until much later, which is why I clear it at the beginning of each entry, so the routine won't get confused.

Finally, if all the criteria are met (>8304=>834A and KEY (>8375) contains the valid entry key: enter >0D, up arrow >0B, or down arrow >0A), then the meat of the program goes to work, computes the checksum and puts it on the screen after an extra scroll (XBasic does its own scroll after I'm finished). Please note that I use BLWP @XMLLNK with data SCROLL instead of adding the whole routine. This saves a lot of typing. However, for those of you who are interested, I am also providing the entire routine done by DISKASSEMBLER, so that you can place it in an E/A assembly file if you wish as this one exists in Bank 1 of XBasic's ROM at >6000->7FFF, and hence can't be used by E/A.

I'm hoping that everyone finds this program useful and that it is widely used. I'm only sorry I didn't write it three years ago! Finally, I would like to thank Doug Warren for writing "EXPLORER" without which I could not have done this since I needed to find out where XBasic does what! (I also must blame Doug for my bleary eyes!) And I especially would like to thank Craig Miller for his invaluable help and advice while I was writing the program. As Craig slowly leaves the TI community, we will all feel the loss.

```
* SOURCE CODE TO WRITE CHECKSUM FOR ENTERED XB
* LINE ON SCREEN
* BY TOM FREEMAN, LA 99ERS
* THIS IS PUBLIC DOMAIN, PLEASE DISTRIBUTE IT
* WIDELY!
        DEF  ON,OFF,CHECK,CURSOR
VMBR    EQU  >202C
VMBW    EQU  >2024
VSBR    EQU  >2028
VSBW    EQU  >2020
VWTR    EQU  >2030
XMLLNK  EQU  >2018
SCROLL  EQU  >0026          ADDRESS OF ROUTINE IN
*                           ROM INDEXED ON >6010
NSAVE   EQU  >8304          EQU >7ADA IN MY XB
*                           MODULE
LSAVE   EQU  >8342          ADDRESS WHERE LENGTH OF
*                           CRUNCHED LINE IS SAVED
FAC     EQU  >834A
GRMRA   EQU  >9802          GROM READ ADDRESS PORT
GRMWA   EQU  >9C02          GROM WRITE ADDRESS PORT
DONE    DATA 0
SAV11   DATA 0
SAVEGA  DATA 0
LOWAD   DATA >6AA0          /ADDRESS RANGE IN GROM
*                           WHERE FIRST KEY PRESS
HIAD    DATA >6AD8          ON COMMAND LINE IS
*                           \REQUESTED
ENTER   DATA >000A,>0B0D    ENTER KEY, UP AND DOWN
*                           ARROW
COUNT   DATA 0
CUR1    BSS  8
CUR2    DATA >007E,>4242,>4242,>7E00 HOLLOW
*                           CURSOR DATA
INVVID  DATA >1F1F          INVERSE VIDEO COLORS,
*                           THIS IS BLACK ON WHITE
TITLE1  TEXT '  XBASIC ERROR CHECKER  '
TITLE2  TEXT '    USING CHECKSUMS     '
TITLE3  TEXT 'BY TOM FREEMAN, LA 99ERS'
GETDEC  CI   R4,10          /IF NUMBER IS 10+ THEN
*                           NEED TO GET TO >41 "A"
        JLT  GO             \NOT >3A
        AI   R4,7
GO      AI   R4,>30         MAKE IT AN ASCII
*                           CHARACTER
        MOV  R4,R1
        AI   R1,>80         THIS IS BASIC BIAS OF
*                           >60 PLUS >50 TO GET TO
*                           ALTERNATE CHARACTER SET
*                           AT ASCII 128
        SWPB R1
        BLWP @VSBW          WRITE ON SCREEN
        RT
CURSOR  LI   R0,>03F0
        LI   R1,CUR1
        LI   R2,8
        BLWP @VMBR          SAVE ORIGINAL CURSOR
*                           PATTERN AT CUR1
        LI   R0,>480        /THE 80 BYTES FROM >480
*                           !TO >4CF ARE ASCII 48-
        LI   R1,LBUF        !57 ("0" TO "9").
*                           !TEMPORARILY STORED AT
        LI   R2,80          \LBUF
        BLWP @VMBR
        LI   R0,>700
        BLWP @VMBW          NOW PUT THEM AT >700
*                           AS ALTERNATE CHAR. SET
        BLWP @XMLLNK
        DATA SCROLL         SCROLL UP 1 LINE
        LI   R2,TITLE1
        LI   R3,>6060       ADD BASIC BIAS TO TITLE
*                           CHARACTERS
        LI   R4,36
        MOV  R2,R1
CR1     A    R3,*R2+
        DEC  R4
```

Continued On Next Page

```
       JNE  CR1
       LI   R0,>2E4
       LI   R2,24
       BLWP @VMBW          WRITE 1ST LINE
       BLWP @XMLLNK
       DATA SCROLL         SCROLL AGAIN
       LI   R0,>2E4
       LI   R1,TITLE2
       LI   R2,24
       BLWP @VMBW          WRITE 2ND LINE
       BLWP @XMLLNK
       DATA SCRDLL         SCRDLL AGAIN
       LI   R0,>2E4
       LI   R1,TITLE3
       LI   R2,24
       BLWP @VMBW          WRITE 3RD LINE
* CALL LINK("CURSDR") DOES THE SETUP AND
* CONTINUES ON TO "ON"
* CALL LINK("DN") STARTS HERE AND DOESN'T NEED
* THE SETUP
ON     LI   R0,>03F0
       LI   R1,CUR2
       LI   R2,8
       BLWP @VMBW          LOAD THE HOLLOW CURSOR
*                          INTO VDP
       LI   R0,CHECK       LOAD THE INTERRUPT
*                          ADDRESS INTO THE ISR
       MOV  R0,@>83C4      (INTERRUPT SERVICE
*                          ROUTINE) HOOK AT >83C4
       RT
OFF    LI   R0,>03F0
       LI   R1,CUR1
       LI   R2,8
       BLWP @VMBW          RELOAD THE ORIGINAL
*                          CURSOR
       CLR  @>83C4         CLEAR THE ISR HOOK
*                          (TURN OFF INTERRUPT)
       RT
CHECK  MOVB @GRMRA,@SAVEGA "PEEK" AT THE
*                          CURRENT GROM ADDRESS
*                          AND SAVE IT AT SAVEGA,
*                          MSB 1ST.  GROM ADDRESS
*                          IS NOW INDETERMINATE
       SWPB @SAVEGA
       MOVB @GRMRA,@SAVEGA
       SWPB @SAVEGA
       DEC  @SAVEGA        ADJUST FOR AUTO
*                          INCREMENT
       C    @SAVEGA,@LOWAD TEST FOR THE LOW
*                          END OF RANGE WHERE
*                          START OF COMMAND LINE
       JL   CHECK1         IS,JUMP OUT IF TOO LOW
       C    @SAVEGA,@HIAD  HIGH END OF RANGE
       JH   CHECK1         JUMP OUT IF TOO HIGH
       CLR  @DONE          RESET FLAG FROM
*                          PREVIOUS CHECKSUM
*                          ROUTINE
       CLR  @NSAVE         THIS CORRECTS FOR A
*                          MYSTERIOUS ERROR I
*                          FOUND!
CHECK1 MOVB @SAVEGA,@GRMWA RESET GROM ADDRESS
*                          THROUGH GRMWA PORT
       SWPB @SAVEGA
       MOVB @SAVEGA,@GRMWA
*
* NEXT 4 LINES SET THE "INVERSE VIDED" FOR
* CHECKSUMS - CAN BE DELETED
       LI   R0,>81C        RESET COLORS FOR
       LI   R1,INVVID      CHARACTER SETS 13-14 AT
       LI   R2,2           EVERY INTERRUPT(XB
       BLWP @VMBW          ALWAYS RESETS TO
*                          DEFAULT).  DELETE THESE
* 4 LINES IF YOU DON'T LIKE THE INVERSE VIDED
* EFFECT
*
* NEXT 9 LINES CHANGE SCREEN & CHAR COLORS
* WHILE IN CHECKSUM MODE AND CAN BE DELETED IF
* YOU DON'T LIKE THE EFFECT
       LI   R0,>80F        START OF COLOR TABLE
*                          FOR CHAR SET 0
       LI   R1,>F400       WHITE ON BLUE
       LI   R2,13          13 COLOR SETS
COL    BLWP @VSBW          WRITE A BYTE TO COLOR
*                          TABLE
       INC  R0             NEXT COLOR SET
       DEC  R2
       JNE  COL
       LI   R0,>0704       SCREEN COLOR 4(DARK
*                          BLUE)
       BLWP @VWTR
* END OF OPTIONAL LINES
*
       ABS  @DONE          /IF THE ROUTINE WAS
       JNE  RETURN         !ALREADY DONE GET
*                          \OUTTA HERE!
       LI   R1,3           CHECK FOR THE 3 VALID
CHECK2 CB   @ENTER(R1),@>B375  ENTRY KEYS AND
*                          LEAVE IF THERE AREN'T
* ANY.  NOTE USE OF INDEXING
       JEQ  C1             IF VALID KEY THEN GO ON
       DEC  R1             GO FOR MORE
       JNE  CHECK2
       RT
C1     MOV  @NSAVE,@NSAVE  WHEN >B304 CONTAINS
       JEQ  RETURN         A NON ZERO KEY AND IS
*                          WHAT IS IN >B34A THEN
* WE'RE READY TO GO!
       C    @NSAVE,@FAC
       JNE  RETURN
       SETO @DONE          INDICATE THE CHECKSUM
*                          IS ABOUT TO BE WRITTEN
       MOVB @LSAVE,R2      GET THE LENGTH BYTE OF
*                          CRUNCHED LINE
       SRL  R2,8           MOVE TO LSB
       LI   R0,>0820       CRUNCH BUFFER
       LI   R1,LBUF        WHERE WE WILL STORE IT
       BLWP @VMBR          MOVE IT
       CLR  @COUNT         COUNT WILL CONTAIN
*                          CHECKSUM, IN BINARY
C2     AB   *R1+,@COUNT+1  ADD EACH BYTE OF
       DEC  R2             CRUNCHED LINE TO IT, 1
       JNE  C2             BY 1 BECAUSE WE ARE
*                          ADDING BYTES,WHEN WE GO
* OVER >FF, THE CLOCK GOES BACK TO ZERO
D0     MOV  R11,@SAV11     SAVE THE RETURN
*                          ADDRESS
       BLWP @XMLLNK
       DATA SCROLL         SCROLL UP THE SCREEN
       LI   R0,>2E2        3RD COLUMN, BOTTOM ROW
*                          OF SCREEN
       MOV  @COUNT,R5      MOVE THE VALUE AT
*                          COUNT (WORD VALUE BUT
* LESS THAN 256) TO R5
       LI   R2,10          R2 AND R3 CONTAIN THE
       LI   R3,100         DIVISORS
       LI   R6,2           2 LOOPS FOR 100'S AND
*                          10'S PLACE
D1     CLR  R4             ASL DIVISION IS DONE
       DIV  R3,R4          THIS WAY.  VALUE OF 1ST
*                          R IS DIVIDED "INTO" 2ND
* R (E.G. R3 INTO R4).  THE 2ND REG IS ACTUALLY
* 2 CONTIGUOUS REGISTERS.  THE QUOTIENT IS
* PLACED IN THE FIRST AND THE REMAINDER IN THE
* 2ND.  ORIGINALLY THE FIRST MUST BE 0, OR THERE
* WILL BE AN "OVERFLOW" SO R4 NOW CONTAINS THE
* INTEGER QUOTIENT
       BL   @GETDEC        CONVERT IT TO ASCII
*                          AND PUT ON SCREEN
       INC  R0             NEXT SCREEN POSITION
       MOV  R2,R3          NEXT DIVISOR
       DEC  R6             ANY MORE TO DO?
       JNE  D1
       MOV  R5,R4          1'S PLACE IS THE
*                          REMAINDER FROM 2ND
```

Continued On Next Page

```
* DIVISION
       BL   @GETDEC        PUT THIS ONE ON SCREEN
*                          TOO
       MOV  @SAV11,R11     RESTORE RETURN ADDRESS
RETURN RT                  AND RETURN
*
* THIS IS END OF PROGRAM AND IS A CONVENIENT
* PLACE TO PUT THE BUFFER, WHICH HAS NO DATA TO
* START
LBUF   END

* SCROLL SUBROUTINE - FOR USE IN OTHER PROGRAMS
* WORKSPACE MUST BE >83E0
SCROLL LI   R12,>02E0
       LI   R10,>0020
       CLR  R9
       MOV  R11,R6
       BL   @AA
       LI   R5,>8C00
       LI   R4,>02E0
       LI   R1,>7F80
       LI   R2,>001C
       BL   @AF
       MOVB R1,*R5
       SWPB R1
AB     MOVB R1,*R5
       DEC  R2
       JNE  AB
       SWPB R1
       MOVB R1,*R5
       MOVB R1,*R5
       B    *R6
AA     CLR  R8
       MOVB @>83F5,*R15
       STWP R7
       MOVB R10,*R15
AD     MOVB @>8800,*R7+
       INC  R10
       INC  R8
       DEC  R12
       JEQ  AC
       CI   R8,>000C
       JLT  AD
AC     MOVB @>83F3,*R15
       ORI  R9,>4000
       MOVB R9,*R15
       STWP R7
AE     MOVB *R7+,@>8C00
       INC  R9
       DEC  R8
       JNE  AE
       MOV  R12,R12
       JNE  AA
       B    *R11
AF     MOVB @>83E9,*R15
       ORI  R4,>4000
       MOVB R4,*R15
       NOP
       MOVB R1,@>8C00
       B    *R11
```

```
100 CALL INIT :: CALL LOAD(9
460,0,0,0,0,0,0,106,160,106,
216,0,10,11,13,0,0) !180
110 CALL LOAD(9484,0,126,66,
66,66,66,126,0,31,31,32,32,8
8,66,65,83,73,67,32,69) !144
120 CALL LOAD(9504,82,82,79,
82,32,67,72,69,67,75,69,82,3
2,32,32,32,32,32,85,83,73,78
) !107
130 CALL LOAD(9526,71,32,67,
72,69,67,75,83,85,77,83,32,3
2,32,32,32,66,89,32,84,79,77
) !119
140 CALL LOAD(9548,32,70,82,
69,69,77,65,78,44,32,76,65,3
2,57,57,69,82,83,2,132,0,10)
!052
150 CALL LOAD(9570,17,2,2,36
,0,7,2,36,0,48,192,68,2,33,0
,176,6,193,4,32,32,32) !199
160 CALL LOAD(9592,4,91,2,0,
3,240,2,1,37,4,2,2,0,8,4,32,
32,44,2,0,4,128) !121
170 CALL LOAD(9614,2,1,39,22
,2,2,0,80,4,32,32,44,2,0,7,0
,4,32,32,36,4,32) !166
180 CALL LOAD(9636,32,24,0,3
8,2,2,37,22,2,3,96,96,2,4,0,
36,192,66,172,131,6,4) !204
190 CALL LOAD(9658,22,253,2,
0,2,228,2,2,0,24,4,32,32,36,
4,32,32,24,0,38,2,0) !067
200 CALL LOAD(9680,2,228,2,1
,37,46,2,2,0,24,4,32,32,36,4
,32,32,24,0,38,2,0) !020
210 CALL LOAD(9702,2,228,2,1
,37,70,2,2,0,24,4,32,32,36,2
,0,3,240,2,1,37,12) !006
220 CALL LOAD(9724,2,2,0,8,4
,32,32,36,2,0,38,36,200,0,13
1,196,4,91,2,0,3,240) !119
230 CALL LOAD(9746,2,1,37,4,
2,2,0,8,4,32,32,36,4,224,131
,196,4,91,216,32,152,2) !239
240 CALL LOAD(9768,36,248,6,
224,36,248,216,32,152,2,36,2
48,6,224,36,248,6,32,36,248,
136,32) !133
250 CALL LOAD(9790,36,248,36
,250,26,8,136,32,36,248,36,2
52,27,4,4,224,36,244,4,224,1
31,4) !013
260 CALL LOAD(9812,216,32,36
,248,156,2,6,224,36,248,216,
32,36,248,156,2,2,0,8,28,2,1
) !054
270 CALL LOAD(9834,37,20,2,2
,0,2,4,32,32,36,2,0,8,15,2,1
,244,0,2,2,0,13) !105
280 CALL LOAD(9856,4,32,32,3
2,5,128,6,2,22,251,2,0,7,4,4
,32,32,48,7,96,36,244) !204
290 CALL LOAD(9878,22,62,2,1
,0,3,152,33,36,254,131,117,1
9,3,6,1,22,250,4,91,200,32)
!180
300 CALL LOAD(9900,131,4,131
,4,19,49,136,32,131,4,131,74
,22,45,7,32,36,244,208,160,1
31,66) !038
310 CALL LOAD(9922,9,130,2,0
,8,32,2,1,39,22,4,32,32,44,4
,224,37,2,184,49,37,3) !195
320 CALL LOAD(9944,6,2,22,25
2,200,11,36,246,4,32,32,24,0
,38,2,0,2,226,193,96,37,2) !
138
330 CALL LOAD(9966,2,2,0,10,
2,3,0,100,2,6,0,2,4,196,61,3
,6,160,37,94,5,128) !027
340 CALL LOAD(9988,192,194,6
,6,22,248,193,5,6,160,37,94,
194,224,36,246,4,91) !104
350 CALL LOAD(16376,79,78,32
,32,32,32,37,244) !042
360 CALL LOAD(16368,79,70,70
,32,32,32,38,14) !240
370 CALL LOAD(16360,67,72,69
,67,75,32,38,36) !002
380 CALL LOAD(16352,67,85,82
,83,79,82,37,122) !053
390 CALL LOAD(8194,39,22,63,
224):: CALL LINK("CURSOR") !
143
```

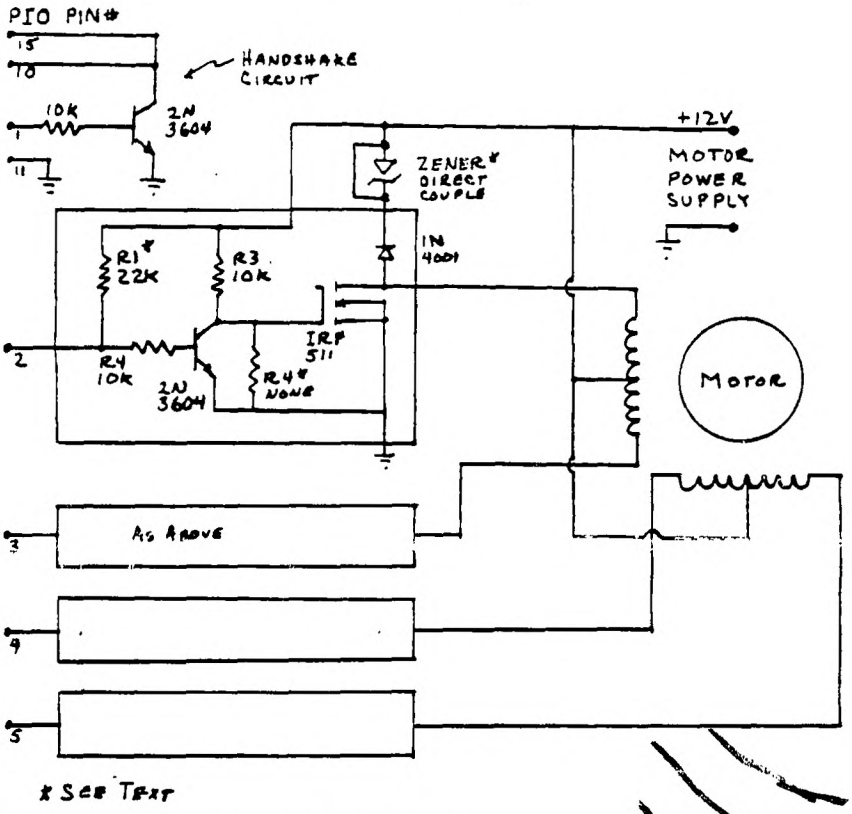
Drive Circuit Parts List

ITEM	RADIO SHACK #	#/PACK	#NEEDED
2N3604 NPN TRANSISTOR	276-1603	15	5(1 PACK)
IRF511 MOSFET TRANSISTOR	276-2027	EA	4
1N4001 DIODE (OR EQUIV)	276-1563	25	4(1 PACK)
10K 1/4 WATT RESISTOR	271-1335	5	9(2 PACK)
22K 1/4 WATT RESISTOR	271-1339	5	4(1 PACK)
PC BOARD	276-168	EA	1

AIRPAX L82701-P2 STEPPER MOTOR
(12 VOLT/36 OHM/48 STEP/REV)

The K82701-P2 (low inductance) or K83701-P2 (24 STEP/REV) will work too. Other motors may require additional resistance in series to limit the current, do not exceed the 3 amp rating of the MOSFET and check to limit the case temperature (add heat sink if required).

SCHEMATIC - STEPPER MOTOR DRIVE



* SEE TEXT

DELAWARE VALLEY USERS' GROUP
P. O. BOX 6240
STANTON BRANCH
WILMINGTON, DE 19804

Subscription expires with date on label

FIRST CLASS MAIL

TO:

0000 EDMONTON 99'ER ONLINE
P.O. BOX #11983
EDMONTON ALB. CANADA T5J 3L1