PART NO.
062-6704-00

TEKNIQUES
VOL. 6 NO. 4 T1

APPLICATIONS LIBRARY
APPLICATIONS LIBRARY
APPLICATIONS LIBRARY
APPLICATIONS LIBRARY
APPLICATIONS LIBRARY
APPLICATIONS LIBRARY
APPLICATIONS LIBRARY
APPLICATIONS LIBRARY
APPLICATIONS LIBRARY
APPLICATIONS LIBRARY
APPLICATIONS LIBRARY
APPLICATIONS LIBRARY

Tektronix®
COMMITTED TO EXCELLENCE

Tektronix®
COMMITTED TO EXCELLENCE

TEKNIQUES
VOL. 6 NO. 4 T1
062-6704-01

DOCUMENTATION

Applications Library
Group 451
Tektronix, Inc.
P.O. Box 500
Beaverton, Oregon 97007

10/82

Prepared by the 4050 Series Applications Library. The 4050 Series Applications Library is maintained as a service for our customers by the Information Display Division of Tektronix, Inc., Group 451, P.O. Box 500, Beaverton, Oregon 97077 U.S.A.

Tektronix®
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE	PART NUMBER
TEKNIQUES VOL. 6 No. 4 T1	062-6704-01
ORIGINAL DATE: November, 1982	REVISION DATE:

ABSTRACT

TEKniques Vol. 6 No. 4 T1 tape consists of 17 programs: two CAD, four Education/Research, three Graphing, two Interfacing, one Mechanical Engineering, two Programming Aids, one Recordkeeping, one Text Processing, and one Utility.

Four of the program must be transferred to their own dedicated tapes. Complete instructions for accomplishing the transfers are included in the documentation.

The individual abstracts describe the programs. Read the documentation before running!

Program #	Title	File #	Documentation Page #
-	Directory	1	
1	Assembler	2	1
2	TECO	3	91
3	PCB Layout	4-5	99
4	Pipe Construction	6	105
5	Moment of Inertia	7	109
6	FFT of 2048 Real Numbers	8	121
7	IFT of 1024 Complex Numbers	9	121
8	Newton Integration and Plot	10	123
9	Bauer-Reinsch Inversion	11	133
10	4050-4010 Utilities	12-13	137

TITLE	NUMBER
TEKniques Vol. 6 No. 4 T1	062-6704-01

Program #	Title		File #	Documentation Page #
11	Telex Tape		14	147
12	Data Chart		15	155
13	SDBAR		16-17	159
14	4050/468 Utility III	(transfer)	18	165
15	Inventory Control	(transfer)	19-22	177
16	Flowcharter II	(transfer)	23-27	183
17	Friendly Graphing	(transfer)	28-30	193

TITLE	PART NUMBER
TEKniques Vol. 6 No. 4 T1	062-6704-01

TRANSFERRING FILES TO A NEW TAPE

PLOT 50 General Utilities Vol. 1 (TEKTRONIX Part #4050A08) contains a program to transfer any type of 4050 files (program/data/text) quickly and easily along with the header names; however, it requires a 4924 Tape Drive, as does TAPEDUPE program contained on TEKniques Vol. 5 No. 4 T1 tape.

Transferring ASCII or BINARY PROGRAMS without a transfer program

Step 1. Do a TLIST of the MASTER program tape.

Step 2. Record which files go with which program (they are all named) and the size of each file.

Step 3. MARK your new tape to accept the respective files for that program, e.g.,

```
FIND 0
MARK 1,20000
FIND 2
MARK 1,4000
etc.
```

Step 4. Insert the MASTER tape.

```
FIND a file
OLD for ASCII or CALL "BOLD" for BINARY
```

Step 5. Insert the new tape

```
FIND the file to receive the file in memory
SAVE for ASCII or CALL "BSAVE" for BINARY
```

REPEAT Steps 4 and 5 until all files comprising that program are transferred to the new tape. Note: This procedure will not retain the file header names.

Transferring ASCII or BINARY DATA to a new tape

The 4051R06 Editor ROM could be used to transfer ASCII DATA files.

4050 Applications Library program "Binary Data File Duplicator" will transfer BINARY DATA files without any peripheral.

4050 Applications Library program "Tape Duplication" will transfer ASCII or BINARY DATA or PROGRAM files, but requires a 4924 Tape Drive.

Both of these programs are contained on the 4050 Applications Library UTILITIES T1 tape (TEKTRONIX Part # 062-5974-01), and UTILITIES D1 disk (TEKTRONIX Part #062-5975-01).

TITLE	PART NUMBER
TEKniques Vol. 6 No. 4 T1	062-6704-01

Tektronix®
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 Program 1
ORIGINAL DATE: November, 1982 / REVISION DATE:	EQUIPMENT AND OPTIONS REQUIRED: 4052/54 A-Series, 64K
AUTHOR: Ed Post — Tektronix, Inc. Wilsonville, OR	PERIPHERALS: Optional-4907 File Manager, Extended Memory

ABSTRACT

Files: 1 ASCII Program

Statements: 591

A CALL "EXEC" routine has been added to the 4052A and 4054A Series Graphics Systems to allow users to execute programs written in 6800 machine code. Extensions allow access to the enhanced A-Series instruction set.

This program is an assembler written in 4052A/54A extended BASIC which will read an assembly language program from a tape, disk, or extended memory file, assemble it (generating relocation information as well), then store the object code in another file for later execution.

Any of the editors available in the Applications Library, or the 4052R06 Editor ROM pack, can be used to create an assembly program in a file.

This assembler program will prompt for the input file of the assembly program and the output file on which to store the object code. A listing will also be displayed on the screen with any syntax errors listed below the erring line of code. A symbol table is produced after the completed listing, showing all absolute and relative labels generated.

This is not meant to be a production assembler. It's missing several features commonly available in assemblers such as expressions, ASCII constants, decimal and octal modes.

It assembles about two lines of code a second. It does, however, document the command format for "EXEC", give some idea of the format of the extended opcodes designed into the 4052A/54A bit-slice processor, and really work.

Users who experiment with "EXEC", however, will undoubtedly crash the firmware regularly until they figure out what they are doing.

NO SUPPORT BY TEKTRONIX IS IMPLIED OR WILL BE PROVIDED.

Included in the 4052A/54A Assembler documentation is the complete description of all new instructions in the "A" instruction set, and a listing of "entry points" to system firmware routines. The user will also need the M6800 Programming Reference Manual published by Motorola, Inc.

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 Program 1

OVERVIEW OF CALL "EXEC"

The first character in the command CALL "EXEC" is control-E. This prevents A-Series users from trying to use 4051 machine code programs since these programs will require modification to run on the A-Series.

The system assumes the CALL "EXEC" command string is in binary (not hex) and executes it directly inside the string variable. The new CHR function of the A-Series allows creation of all 256 ASCII characters.

Absolute addressing (for JMP and JSR commands) is permitted. An optional relocation parameter allows the user to specify words of the execute string subject to relocation. "EXEC" will determine the absolute location of the string and relocate these specified words as necessary.

The ROM space from which the 4052A/4054A fetches its system instructions is separate from the space assigned for data storage. When the CALL "EXEC" command string is executed, however, the "EXEC" instructions are placed in the data storage area and the instruction fetch space set to this data space. This makes it impossible to call code in the system ROMs using JSR. The solution is to use the SWI (software interrupt) instruction. The SWI handler in the system ROM essentially converts the sequence <SWI> <16 bit address> into a "JSR-to-ROM" instruction: the next two bytes in the instruction stream are taken as an address in ROM to JSR to. After calling the ROM routine, the system will return to the data space containing the "EXEC" string of instructions and continue executing them.

The address of a set of instructions that return control to the BASIC operating system is left on the stack by "EXEC", so the user routine can use the normal RTS instruction to terminate execution of CALL "EXEC".

Users of CALL "EXEC" feature should be cautioned: NO SUPPORT BY TEKTRONIX IS IMPLIED OR WILL BE PROVIDED.

Tektronix will not attempt to fix firmware problems caused by programs that use the CALL "EXEC" feature.

It is a common experience of first time users of CALL "EXEC" to accidentally execute a microcode-test instruction (00) causing the system to initialize itself.

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 Program 1

COMMAND FORMAT

CALL "EXEC",EX$[,REL$][;parm1,parm2,...] {first character ctrl-E}

semantics:

EX$ Contains a binary string which will be JSRed to by the system. Due to the fact that this is a binary string, using a string constant for EX$ or REL$ would not be useful and is therefore not allowed.

REL$ If present, contains relocation information in binary form. The bytes of this string are combined two at a time to form words, any odd final byte being ignored. The first word contains the assumed starting address of EX$. If this address agrees with the actual location of EX$, no relocation is performed. Otherwise, successive words of REL$ are the relative locations of words inside EX$ that must be relocated. This is accomplished by adding the value (location_of_EX$ - word_0_of_REL$) to these words in EX$. When done, the new location of EX$ is placed in word zero of REL$.

parm1... Each parameter after the semicolon is placed on the stack in standard form: expressions are evaluated and a VALTG item placed, variables are represented by a PNTSTG or PNTNTG item, literal strings as PLOSTG items, and array elements as PAETG items. If the user supplies zero parameters, there will still be a SEMITG item placed on the stack to denote the end of user parameters -- this makes it easier for the user's parameter parser to avoid mistaking EX$ and REL$ for parameters.

Stack organization at entry will look this:

```
SP --->   return addr  (2 bytes)
          parmn

          .
          .
          parm2
          parm1
          SEMITG
          REL$
          EX$
          CALLTG
```

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 Program 1

Possible errors (all signalled by "PARAMETER ERROR" message):

1) call not in form:
```
                    CALL "EXEC",EX$
                or: CALL "EXEC",EX$,REL$
                or: CALL "EXEC",EX$;PARM1,PARM2...
                or: CALL "EXEC",EX$,REL$;PARM1,PARM2....
```

2) EX$ or REL$ are not string variables, or are undefined

3) EX$ is zero length. (At least a RTS instruction should be present)

4) A word outside EX$ requested for relocation by REL$

EXAMPLE: CALL "EXEC",EX$,REL$;ARG1,ARG2

```
0000:                               PUTCHR=4063
0000:   4F                          CLR A
0001:   3F 4063             LOOP:  SWI PUTCHR
0004:   4C                          INC A
0005:   27 03                       BEQ     DONE
0007:   7E 0001                     JMP     LOOP
000A:   39                   DONE:  RTS
```

To represent this code, EX$ and REL$ would have these values:

EX$ = 4F 3F 40 63 4C 27 03 7E 00 01 39 (HEX)

which would be coded as:

EX$="O?@cL'C~@A9" (binary representation)

and

REL$ = 00 00 00 08 (HEX)

coded as:

REL$="@@@H"

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 Program 1

This REL$ asks the system to relocate the address in the JMP instruction relative to the start of EX$. (The first two bytes in REL$ are the assumed location of the start of the code. If they match the actual location of the code, no relocation is done.) Let's assume that the address of the first byte of data in EX$ is at #H5227. After the relocation operation, the new valus of EX$ and REL$ would be:

EX$ = 4F 3F 40 63 4C 27 03 7E 52 28 39 (HEX)

REL$ = 52 27 00 08 (HEX)

After this, the string can be executed and the JMP instruction will go to the proper location in memory. The new value in the start of REL$ ensures that the next call to "EXEC" will again work even if the string moves to a new location between invocations. Let's say it moves to #H5000. The new value of location 0008 in the string is (5228-5227)+5000 or 5001. So:

EX$ = 4F 3F 40 63 4C 27 03 7E 50 01 39 (HEX)

REL$ = 50 00 00 08 (HEX)

ASSEMBLER PROGRAM

A small assembler has been written to show users how to effectively use the "EXEC" feature of the 4052A/4054A. It has the capability of assembling code written for the standard Motorola 6800 processor as well as code using the extended opcodes designed into the 4052A/4054A processor instruction set.

TITLE

4052A/4054A Assembler

ABSTRACT NUMBER
TEKniques Vol. 6 No. 4
Program 1

ASSEMBLY LANGUAGE FORMAT

Lines of code allowed in the assembler follow one of these formats:

1) Blank line.

2) Comment line. The first non-blank non-tab character is a semicolon.

3) Absolute address declaration.

   <LABEL> = *<CONSTANT>* [; *<COMMENT>*]

4) .BYTE declaration.

   [*<LABEL>* :] *.BYTE <LABEL:CONSTANT>* [; *<COMMENT>*]

5) .WORD declaration.

   [*<LABEL>* :] *.WORD <LABEL:CONSTANT>* [; *<COMMENT>*]

6) Code line. An optional LABEL: followed by an opcode, optionally followed by a parameter and an addressing mode.

   [*<LABEL>* :] *<OPCODE>* [*<A:B:X:S:G>*] [*<LABEL:CONSTANT>*] [,*D:,X:,I*]
   [; *<COMMENT>*]

<LABEL> is an arbitrary sized list of alpha, numeric, period, or underscore characters, starting with an alpha.

<CONSTANT> is an arbitrary sized list of numeric and [A..F] starting with a numeric or a minus sign. These are always interpreted in hexacecimal.

<COMMENT> is anything not containing a carriage return.

<LABEL:CONSTANT> is either a *<CONSTANT>* or a label that appears exactly once either as a [*<LABEL>* :] in some line, or as a *<LABEL>* = *<CONSTANT>*.

Example lines

```
                        ; comment line, followed by blank line

            GLOBAL=0A4D ; absolute address
ER:         .BYTE 0A    ; leading zero required to make constant instead of label
MINUS_ONE:  .WORD -1    ; underscores allowed

            IND         ; implied addressing
            CLR A       ; can also be written CLRA without intervening space
            LDA A GLOBAL ; extended addressing
HERE:       LDAX minus_one,I   ; upper/lower case the same
            EPSH 0123456789ABCDEF,I  ; an eight byte quantity
            RTS         ; every program should have one of these
```

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 Program 1

CAVEAT

This is not meant to be a production assembler. It's missing several features commonly available in assemblers--expressions, ASCII constants, decimal and octal modes. It assembles about two lines of code a second. It does have a few virtues, however:

1) it documents the command format for "EXEC";

2) it gives some idea of the format of the extended opcodes designed into the 4052A/54A bit-slice processor;

3) it really does work.

However, if you experiment with "EXEC", you will undoubtedly crash the firmware regularly until you figure out what you are doing.

Example: the sample listing contains a call to a HEXOUT routine used by the SYSERR code. It doesn't appear in the SYSJMP table and is, therefore, subject to relocation in future system releases. In fact, the location in the sample listing is probably not accurate. (BACKUP, DSPCHR, and PSTK should be safe.)

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 Program 1

OPERATING INSTRUCTIONS

1) Using any of the editors available in the Applications Library, or the 4052/4 Editor ROM pack, create your assembly program on a tape, disk, or extended memory file.

2) RUN the assembler. You will be prompted for "input file". Reply with the file number or name of the file created in step 1.

3) You will be prompted for "output file". Reply with the file number or name of a file into which you wish the binary EX$ and REL$ to be written. A file number of 0 can be supplied, in which case the code will be executed immediately instead of written to tape or disk.

4) Wait for pass 1 to finish. This takes about 1/4 second per line of assembly code.

5) Pass two prints a listing to the screen, with any syntax errors listed below the erring line of code. A symbol table is generated after the completed listing, showing all absolute and relative labels generated.

```
RUN
input file: 5
input file is tape file #5
output file: 0
output file is null file
PASS 1
***EOF***
PASS 2
0000:                          ; file 5 -- a real EXEC test program
0000:                          ;
0000:  4018                            bell=4018
0000:  4063                            dspchr=4063
0000:  0020                            space=20          ; ascii code
0000:
0000:  BD 0004R                        jsr     doit
0003:  39                              rts
0004:  B6 001DR                doit:   lda a   times
0007:  36                      top:    psh a
0008:  3F 4018                         swi     bell
000B:  32                              pul a
000C:  4A                              dec a
000D:  27 03                           beq     next
000F:  7E 0007R                        jmp     top
0012:
0012:  86 20                   next:   lda a   space,i
0014:  36                      loop:   psh a
0015:  3F 4063                         swi     dspchr
0018:  32                              pul a
0019:  4C                              inc a
001A:  2E F8                           bgt     loop
001C:
001C:  39                              rts
001D:
001D:
001D:  10                      times:  .byte 10
```

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 Program 1

```
001E:
001E:
***EOF***
symbols---
bell=  4018        dspchr=  4063        space=  0020        doit:  0004
top:  0007         next:  0012          loop:  0014         times:  001D

error count: 0
done
executing code...
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑_`abcdefg
hijklmnopqrstuvwxyz{|}~↓█done
```

```
run
input file: 9
output file: 10
PASS 1
***EOF***
PASS 2
0000:                                ;
0000:                                ; file 9 -- stack dumper
0000:                                ;
0000:                                ; description: this program lists the item
s on the stack in hex from
0000:                                ;     top to bottom, terminating at the EO
STG item.
0000:                                ;
0000:
0000:  0000                                R0=0
0000:  0001                                R0_PLUS_1=1
0000:  0014                                R10=14
0000:  0019                                EOSTG=19
0000:  0049                                PSTK=49
0000:
0000:  72E0                                HEXOUT=72E0      ; SYSTEM ROUTINE T
O PRINT HEX BYTE
0000:  4063                                DSPCHR=4063      ; PRINT SINGLE ASC
II CHAR
0000:  4012                                BACKUP=4012      ; SCAN DOWN STACK
ITEMS
0000:
0000:  62 49                               PSHRET PSTK      ; STORE RETURN ADD
RESS AWAY
0002:  9F 00                               STS R0,D         ; STORE STACK POIN
TER FOR BACKUP
0004:
0004:  96 00                         LOOP: LDA A R0,D
0006:  3F 72E0                             SWI HEXOUT
```

```
0009:  96 01                    LDA A R0_PLUS_1,D
000B:  3F 72E0                  SWI HEXOUT
000E:  86 3A                    LDA A 3A,I        ; COLON CHARACTER
0010:  3F 4063                  SWI DSPCHR
0013:  86 20                    LDA A 20,I
0015:  3F 4063                  SWI DSPCHR
0018:
0018:  DE 00                    LDX R0,D
001A:  A6 01                    LDA A 1,X
001C:  81 19                    CMP A EOSTG,I
001E:  27 1F                    BEQ DONE
0020:  DF 14                    STX R10,D
0022:  3F 4012                  SWI BACKUP
0025:
0025:  DE 14                L2: LDX R10,D
0027:  A6 01                    LDA A 1,X
0029:  3F 72E0                  SWI HEXOUT
002C:  86 20                    LDA A 20,I        ; SPACE CHARACTER
002E:  3F 4063                  SWI DSPCHR
0031:  DE 14                    LDX R10,D
0033:  A3 14                    INXSTX R10
0035:  9C 00                    CPX R0,D
0037:  26 EC                    BNE L2
0039:
0039:  BD 0056R                 JSR CRLF
003C:  7E 0004R                 JMP LOOP
003F:
003F:  3F 72E0              DONE: SWI HEXOUT
0042:  BD 0056R                 JSR CRLF
0045:  86 45                    LDA A 45,I
0047:  3F 4063                  SWI DSPCHR
004A:  86 4E                    LDA A 4E,I
004C:  3F 4063                  SWI DSPCHR
004F:  86 44                    LDA A 44,I
0051:  3F 4063                  SWI DSPCHR
```

```
0054:
0054:  65 49                              RTRN PSTK
0056:
0056:  86 0D                    CRLF:     LDA A 0D,I
0058:  3F 4063                            SWI DSPCHR
005B:  39                                 RTS
***EOF***
symbols---
R0=  0000          R0_PLUS_1=  0001   R10=  0014          EOSTG=  0019
PSTK=  0049        HEXOUT=  72E0      DSPCHR=  4063      BACKUP=  4012
LOOP:  0004        L2:  0025          DONE:  003F        CRLF:  0056

error count: 0
writing code to file 10
call "Exec",code$,rel$
DBE9: 11                    SEMITG
DBEA: 08 DF 66 00 00        rel$
DBEF: 08 DF C4 00 00        code$
DBF4: 17 C0 E9 AB 6E        "EXEC" CALLTG
DBF9: 18 5A 37              EOLTG
DBFC: 19                    EOSTG
END

list 50000,60000
50000 DO
50010    FOR I=1 TO 10
50020       GOSUB 50030
50030       CALL "EXEC",Code$,Rel$;10,"HI",A$
50040       END

run 50000
DBB0: 08 DF F3 00 00        A$
DBB5: 01 40 A2 40 BD        "HI"
DBBA: 0C 04 04 A0 00 00 00 00 00    10
DBC3: 11                    SEMITG
```

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 Program 1

```
DBC4: 08 DF 66 00 00          Rel 0
DBC9: 08 DF C4 00 00          Code 0
DBCE: 17 C0 C9 AB 6E          "EXEC"  CALL 6
DBD3: 18 93 5A                EOL 6
DBD6: 03 40 94                GOSUB 6
DBD9: 18 93 5A                EOL 6
DBDC: 04 40 7E DF E6 0C 04 01 80 00 00 00 00 00 0C 04 04 A0 00 00 00 00 00
00    FORK 6
DBF3: 18 93 5A                EOL 6
DBF6: 22 40 73                DO 6
DBF9: 18 5A 37                EXIT 6
DBFC: 19                      EOS 6
END
```

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 Program 1

EXTOPS

Extended Operations for the 4052/4054 and 4050A

This document contains complete descriptions of all new instructions in the 4052/54 instruction set. The document is arranged alphabetically (within Chapters) based on the instruction mneumonics.

The operation of each instruction is shown, followed by a brief description of what the instruction does, the effects (if any) on the condition codes, the particulars of the instruction (addressing modes, opcode, and operand field syntax), and any notes.

The only NOP instructions in this instruction set are opcodes 01 and 02. All other unused opcodes are treated as SWI instructions by the micromachine.

The 6800 DAA instruction is not implemented in the 4052/54. The opcode is treated as an illegal opcode.

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 Program 1

The following nomenclature is used in the subsequent definitions.

(a) Operators

()	= contents of register
[]	= contents of memory
<--	= is transferred to
^	= "is pulled from stack"
v	= "is pushed into stack"

(b) Registers in the MPU

ACCA	= Accumulator A
ACCG	= A extended to 16 bits (A is low order 8) available in 4050GX only
ACCB	= Accumulator B
ACCx	= Accumulator ACCA or ACCB
CC	= Condition codes register
X	= Index register, 16 bits
XH	= Index register, higher order 8 bits
XL	= Index register, lower order 8 bits
PC	= Program counter, 16 bits
PCH	= Program counter, higher order 8 bits
PCL	= Program counter, lower order 8 bits
SP	= Stack pointer, 16 bits
SPH	= Stack pointer, higher order 8 bits
SPL	= Stack pointer, lower order 8 bits

(c) Memory and Addressing

EA	= Effective address of operand
M	= A memory location (one byte)
M +1	= The byte of memory at 0001 plus the address of the memory location indicated by "M".
Rel	= Relative address (i.e. the two's complement number stored in the second byte of machine code corresponding to a branch instruction).

(d) Bits 0 thru 7 of the Condition Codes Register

C	= Carry/Borrow	bit0
V	= Two's complement overflow indicator	bit1
Z	= Zero indicator	bit2
N	= Negative indicator	bit3
I	= Interrupt mask	bit4
H	= Half carry	bit5
D	= Data space indicator (1 --> A)	bit6
F	= Fetch space indicator (1 --> B)	bit7

Detailed definitions of the new executable instructions of the source language are provided on the following pages.

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 Program 1

INDEX

CHAPTER 1 NEW OPERATIONS FOR THE 4052/4054

ADAX Add A to Index Register 20
ADXI Add to Index Register Immediate 20
ASPI Add to Stack Pointer Immediate. 21
CBUG Clear Debug Interrupt Vectors 21
CPCH Call Code in PATCH Space. 22
CPX Compare Index Register. 23
FADD Floating Point Add. 24
FDIV Floating Point Divide 25
FDUP Duplicate Floating Point. 26
FMUL Floating Point Multiply 27
FNRM Normalize Floating Point. 28
FPSH Push Floating Point 29
FPUL Pull Floating Point 30
FSUB Floating Point Subtract 31
FSWP Swap Floating Point 32
JMPAX Jump Double-Indexed 32
JMPIN Jump Indirect 33
LDAX Load A Register Double-Indexed. 33
LDBX Load B Register Double-Indexed. 34
LDXX Load X Register Double-Indexed. 34
MOVLR Block Move Low to High. 35
MOVRL Block Move High to Low. 36
NEG Negate (2's complement) 36
PCH Jump to Code in PATCH Space 37
PSHRET Push Return Address on Special Stack. 38
PSHX Push X on the Stack 39
PULX Pull X from the Stack 39
RTRN Return Via the Special Stack. 40
SBUG Set Debug Interrupt Vectors 41
SDA Set Data Space to A 41
SDB Set Data Space to B 42
SFA Set FETCH Space to A. 42
STAX Store B Register Double-Indexed 43
STRK Compute Stroke. 44
TAP A --> CC Not Including Space Bits 45
TAPX A --> CC Including Space Bits 45
TEST Microcode Restart 46
TPA CC --> A Not Including Space Bits 46
TPAX CC --> A Including Space Bits 47
VECT Compute Vector. 48
WADGX Add G to Index Extended 49
WADX Add Memory to Index 50
General Utility MACROS 51

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 Program 1

CHAPTER 2 NEW OPERATIONS FOR 4050GX

ADDG Add to G Accumulator. 53
BUFIN Read a buffer from the GPIB 54
BUFOUT Write a buffer to the GPIB 57
CLRGH Clear High Byte of G 59
CMPGX Compare G and X 60
CMPSYM Compare Name in a Symbol Table Record 61
DEVIN Read a buffer from an I/O device 62
DEVOUT Write a buffer to an I/O device 63
FIXRND Round a float to an integer 64
IFLOAT Convert an integer to a float 65
INXSTX Increment Index Register and Store It 66
LDAG Load G Accumulator. 67
LDAGX Load G Accumulator Double-Indexed 67
PSHG Push G on the Stack 68
PULG Pull G From the Stack 68
SEABNK Search for a CALL name in a ROM bank. 69
STAG Store G Accumulator 70
STAGX Store G Accumulator Double-Indexed. 70
SUBG Subtract From G Accumulator 71
TGX Transfer G to the Index Register. 72
TMULT Multiply a 6 byte integer by 10 73
TXG Transfer the Index Register to G. 73

CHAPTER 3 EXTENSIONS TO 6800 INSTRUCTIONS FOR 4050GX

ABA Add Accumulator B to A. 75
ADD Add to Accumulator (A or B) 75
ASL Arithmetic Shift Left (A or B). 76
CLR Clear Accumulator (A or B). 76
COM Complement Accumulator (A or B) 77
DEC Subtract One From Accumulator (A or B). 77
INC Add One to Accumulator (A or B) 78
LDA Load Accumulator (A or B) 79
PUL Pull Accumulator From Stack (A or B). 80
RTI Return from Interrupt 81
SBA Subtract Accumulator B From A 82
SUB Subtract From Accumulator (A or B). 82
TAB Transfer Accumulator A to B 83
TBA Transfer Accumulator B to A 83

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 Program 1

CHAPTER 1

NEW OPERATIONS FOR THE 4052/4054

ADAX — Add A to Index Register

Operation:

A+X --> X

Description:

The unsigned value in A (eight assumed bits of 0) is added to the index register.

Condition Codes:

H	I	N	Z	V	C	
.	.	1	0	x	x	New value in X is negative
.	.	0	1	x	x	New value in X is zero
.	.	x	x	1	x	Two's complement overflow in addition
.	.	x	x	x	1	Carry out of bit15 in addition

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	CC	

ADXI — Add to Index Register Immediate

Operation:

X+M --> X

Description:

The signed two's complement operand is added to the value currently in the index register.

Condition Codes:

Not affected.

Particulars:

Addressing Modes	Op	Operand Field Syntax
Immediate	14	<8-bit signed number>

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 Program 1

ASPI — Add to Stack Pointer Immediate

Operation:

Sp+M --> Sp

Description:

The signed two's complement operand is added to the value currently in the stack pointer.

Condition Codes:

Not affected

Particulars:

Addressing Modes	Op	Operand Field Syntax
Immediate	15	<8-bit signed number>

CBUG — Clear debug interrupt vectors

Operation:

Interrupt vectors located in A-space from FEF0 to FEFF.

Description:

This is the complement of the SBUG instruction. The interrupt vectors are restored to their normal area in A-space.

Condition Codes:

Not affected.

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	DD	

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 Program 1

CPCH — Call code in PATCH space

Operation:

```
(Pc) + 2 --> Pc
V (PCL)
(Sp)-1 --> Sp
V (PCH)
(Sp)-1 --> Sp
Rel*4+4400 --> Pc
```

Description:

The second byte of the instruction specifies the offset (times 4) into the patch area of code to be executed instead of the original code. The return address following the patch is pushed on the stack before the patch code is executed.

Condition Codes:

Not affected.

Particulars:

Addressing Modes	Op	Operand Field Syntax
Immediate	F3	<8-bit offset>

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 Program 1

CPX — Compare Index Register

Operation:

X-[M,M+1]

Description:

The current contents of X are compared to the 16-bit operand. A two's complement subtract is used to set the CC register, and the resulting CC reflects a valid 16-bit compare. This is a standard 6800 instruction which has been extended to be more useful, by correctly setting the C bit in the CC register (like the 6800 does for the CMP instruction).

Condition Codes:

H	I	N	Z	V	C	
.	.	1	0	.	.	X arithmetically < M
.	.	0	1	.	.	X = M
.	.	0	0	.	.	X arithmetically > M
.	.	.	.	1	.	Two's complement overflow in compare
.	.	x	0	.	1	X logically < M
.	.	x	0	.	0	X logically > M

Particulars:

Addressing Modes	Op	Operand Field Syntax
Immediate	8C	<16-bit value>,I
Immediate	8C	#<16-bit value>
Direct	9C	<8-bit address>,D
Indexed	AC	<8-bit offset>,X
Extended	BC	<16-bit address>

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 Program 1

FADD — Floating Point Add

Operation:

Sp+9 --> Sp
[M[Sp+2..Sp+9]] + [M[Sp-7..Sp]] --> M[Sp+2..Sp+9]

Description:

A floating point add is performed on the two top floating point numbers on the stack. The result is left on the stack.

Condition Codes:

H	I	N	Z	V	C	
.	.	1	0	x	x	Result is negative
.	.	0	1	x	0	Result is zero
.	.	0	0	x	x	Result is positive
.	.	0	1	1	0	Underflow - zero result
.	.	x	0	1	0	Overflow - plus/minus infinity result

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	52	

Programming Note:

Acc A crashed.
Acc B is the number of shifts used to normalize the result.
To do "6.0 + 7.0", first push 6.0, then 7.0, and then do the "+".

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 Program 1

FDIV — Floating Point Divide

Operation:

Sp+9 --> Sp
[M[Sp+2..Sp+9]] / [M[Sp-7..Sp]] --> M[Sp+2..Sp+9]

Description:

A floating point divide is performed on the two top floating point numbers on the stack. The result is left on the stack.

Condition Codes:

H	I	N	Z	V	C	
.	.	1	0	x	x	Result is negative
.	.	0	1	x	0	Result is zero
.	.	0	0	x	x	Result is positive
.	.	0	1	1	0	Underflow - zero result
.	.	x	0	1	0	Overflow - plus/minus infinity result
.	.	x	0	1	1	Division by zero (causes interrupt)

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	5E	

Programming Note:

Acc A crashed.
Acc B is the number of shifts used to normalize the result.
To do "6.0 / 7.0", first push 6.0, then 7.0, and then do the "/".

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 Program 1

FDUP — Duplicate Floating Point

Operation:

[M[Sp..Sp+9]] --> M[Sp-9..Sp]
Sp-9 --> Sp

Description:

The floating point number on the top of the stack is replicated on the stack.

Condition Codes:

Not affected

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	4E	

Programming Note:

To save microcode time, the byte below that pointed to by the stack pointer is duplicated also, making a total of 10 bytes duplicated.

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 Program 1

FMUL — Floating Point Multiply

Operation:

Sp+9 --> Sp
[M[Sp+2..Sp+9]] * [M[Sp-7..Sp]] --> M[Sp+2..Sp+9]

Description:

The floating point multiply is performed on the two top floating point numbers on the stack. The result is left on the stack.

Condition Codes:

H	I	N	Z	V	C	
.	.	1	0	x	x	Result is negative
.	.	0	1	x	0	Result is zero
.	.	0	0	x	x	Result is positive
.	.	0	1	1	0	Underflow - zero result
.	.	x	0	1	0	Overflow - plus/minus infinity result

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	5B	

Programming Note:

Acc A crashed.
Acc B is the number of shifts used to normalize the result.
To do "6.0 * 7.0", first push 6.0, then 7.0, and then do the "*".

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 Program 1

FNRM — Normalize Floating Point

Operation:

Normalize {[M[Sp+2..Sp+9]]} --> M[Sp+2..Sp+9]

Description:

The floating point number on the top of the stack is normalized.

Condition Codes:

H	I	N	Z	V	C	
.	.	1	0	0	0	Result is negative
.	.	0	1	x	0	Result is zero
.	.	0	0	0	0	Result is positive
.	.	0	1	1	0	Underflow. Result zero, Floating Fault Interrupt

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	61	

Programming Note:

Acc A crashed.
Acc B is the number of shifts used to normalize the result.

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

FPSH — Push Floating Point

Operation:

```
V M[EA+7]
V M[EA+6]
V M[EA+5]
V M[EA+4]
V M[EA+3]
V M[EA+2]
V M[EA+1]
V M[EA+0]
V Valtg
```

Description:

The floating point number specified by the operand is pushed on the stack, along with the floating point tag.

Condition Codes:

Not affected.

Particulars:

Addressing Modes	Op	Operand Field Syntax
Direct	3A	<8-bit address>,D
Indexed	3C	<8-bit offset>,X
Extended	3D	<16-bit address>
Immediate	41	#^H<16-digit hex val>
		<16-digit hex val>,I

Programming Note:

A deferred fetch-space change will occur after an FPSH immediate instruction.

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

FPUL — Pull Floating Point

Operation:

Sp+9 --> Sp
[M[Sp-7..Sp]] --> M..M+7

Description:

The floating point number specified by the operand is pulled from the stack. The floating point tag is discarded.

Condition Codes:

Not affected.

Particulars:

Addressing Modes	Op	Operand Field Syntax
Direct	42	<8-bit address>,D
Indexed	45	<8-bit offset>,X
Extended	4B	<16-bit address>

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

FSUB — Floating Point Subtract

Operation:

Sp+9 --> Sp
[M[Sp+2..Sp+9]] - [M[Sp-7..Sp]] --> M[Sp+2..Sp+9]

Description:

A floating point subtract is performed on the two top floating point numbers on the stack. The result is left on the stack.

Condition Codes:

H	I	N	Z	V	C	
.	.	1	0	x	x	Result is negative
.	.	0	1	x	0	Result is zero
.	.	0	0	x	x	Result is positive
.	.	0	1	1	0	Underflow - zero result
.	.	x	0	1	0	Overflow - plus/minus infinity result

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	55	

Programming Note:

Acc A crashed.
Acc B is the number of shifts used to normalize the result.
To do "6.0 - 7.0", first push 6.0, then 7.0, and then do the "-".

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

FSWP — Swap Floating Point

Operation:

M[Sp+2..Sp+9] <--> M[Sp+11..Sp+18]

Description:

The top two floating point numbers on the stack are interchanged.

Condition Codes:

Not affected.

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	51	

Programming Note:

Only the eight bytes of the actual values are swapped. The tags are not!

JMPAX — Jump Double-Indexed

Operation:

A+X --> Pc

Description:

The next instruction to be executed is to be found at X+A.

Condition Codes:

Not affected.

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	1F	

JMPIN — Jump Indirect

Operation:

[M,M+1] --> Pc

Description:

The next instruction to be executed is to be found at the 16-bit address pointed to by the 16-bit operand address.

Condition Codes:

Not affected.

Particulars:

Addressing Modes	Op	Operand Field Syntax
Extended	38	<16-bit address>

LDAX — Load A Register Double-Indexed

Operation:

[X+A] --> A

Description:

The byte at X+A is loaded into A.

Condition Codes:

H	I	N	Z	V	C	
.	.	1	0	0	.	New value in A is negative
.	.	0	1	0	.	New value in A is zero
.	.	0	0	0	.	New value in A is positive

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	1C	

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

LDBX — Load B Register Double-Indexed

Operation:

[X+A] --> B

Description:

The byte at X+A is loaded into B.

Condition Codes:

H	I	N	Z	V	C	
.	.	1	0	0	.	New value in B is negative
.	.	0	1	0	.	New value in B is zero
.	.	0	0	0	.	New value in B is positive

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	1D	

LDXX — Load X Register Double-Indexed

Operation:

[X+A,X+A+1] --> X

Description:

The 16-bit value at X+A is loaded into X.

Condition Codes:

H	I	N	Z	V	C	
.	.	1	0	0	.	New value in X is negative
.	.	0	1	0	.	New value in X is zero
.	.	0	0	0	.	New value in X is positive

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	1A	

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

MOVLR Block Move Low to High

Operation:

M[Sp+1],M[Sp+2] source address (lowest)
M[Sp+3],M[Sp+4] destination address (lowest)
M[Sp+5],M[Sp+6] byte count (may be zero)

Data moved

Sp+6 --> Sp

Description:

A block of data in memory is moved, incrementing the pointers, until the byte count is zero.

Condition Codes:

Crashed.

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	E3	

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

MOVRL — Block Move High to Low

Operation:

M[Sp+1],M[Sp+2] source address (highest)
M[Sp+3],M[Sp+4] destination address (highest)
M[Sp+5],M[Sp+6] byte count (may be zero)

Data moved

Sp+6 --> Sp

Description:

A block of data in memory is moved, decrementing the pointers, until the byte count is zero.

Condition Codes:

Crashed.

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	EC	

NEG — Negate (2's complement)

Operation:

0 - 8-bit value --> 8-bit value

Description:

See the 6800 manual.

Condition Codes:

////////////// WARNING \\\\\\\\\\\\\
The 4052/54 set the Carry bit exactly opposite from how it is set in the 6800 and 4052A/54A. See 6800 manual.

Particulars:

See 6800 manual.

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

PCH — Jump to code in PATCH space

Operation:

Rel*4+4400 --> Pc

Description:

This instruction forces a JUMP to code in the patch space. The code begins at the second byte times 4 plus 4400 hex.

Condition Codes:

Not affected.

Particulars:

Addressing Modes	Op	Operand Field Syntax
Immediate	FD	<8-bit offset>

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

PSHRET — Push return address on special stack

Operation:

```
M[Sp+1],M[Sp+2] --> M[Psp],M[Psp+1]
Sp+2 --> Sp
Psp+2 --> Psp
Sp+1 --> X
```

Description:

The return address on the regular stack is transferred to the pseudo stack referenced by the specified page zero pointer.

Condition Codes:

Not affected

Particulars:

Addressing Modes	Op	Operand Field Syntax
Direct	62	<8-bit address>,D

Programming Notes:

The macro in EXTOPS.MAC uses the page 0 variable XEQSP. The macro Pshps can be used to specify some other page 0 variable.

Notice the implicit TSX at the end of the instruction!!!

PSHX — Push X on the stack

Operation:

Sp-2 --> Sp
X --> M[Sp+1],M[Sp+2]

Description:

The index register is pushed on the stack.

Condition Codes:

Not affected

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	6B	

PULX — Pull X from the stack

Operation:

M[Sp+1],M[Sp+2] --> X
Sp+2 --> Sp

Description:

The index register is pulled from the stack.

Condition Codes:

Not affected.

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	75	

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

RTRN — Return via the special stack

Operation:

```
Psp-2 --> Psp
A --> B
M[Psp],M[Psp+1] --> Pc
```

Description:

Fetch the return address from the pseudo stack with stack pointer on page zero.

Condition Codes:

See the TAB instruction.

Particulars:

Addressing Modes	Op	Operand Field Syntax
Direct	65	<8-bit address>,D

Programming Notes:

The macro in EXTOPS.MAC uses the page 0 variable XEQSP. The macro Rtrps can be used to specify some other page 0 variable.

An implicit TAB instruction is done!!!

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

SBUG — Set debug interrupt vectors

Operation:

Swap in debug interrupt vectors

Description:

Subsequent interrupts will be serviced via vectors in B-space from locations 2 to F in the bank with address 20. This supports the diagnostic ROMpack.

Condition Codes:

Not affected.

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	DC	

SDA — Set Data Space to A

Operation:

CC ! D --> CC

Description:

Subsequent memory accesses for data will access A space.

Condition Codes:

D set

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	18	

TITLE: 4052A/4054A Assembler

ABSTRACT NUMBER: TEKniques Vol. 6 No. 4 T1 Program 1

SDB — Set Data Space to B

Operation:

CC & NOT D --> CC

Description:

Subsequent memory accesses for data will access B space.

Condition Codes:

D reset

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	21	

SFA — Set FETCH Space to A

Operation:

CC & NOT F --> CC

Description:

Instructions subsequent to the next JSR, RTS, BSR, JMP, BRA, relative branch, RTRN, JMPAX, RTI, or FPSH immediate instruction will come from DATA space.

Condition Codes:

F reset

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	03	

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

STAX — Store B Register Double-Indexed

Operation:

B --> [X+A]

Description:

The byte at X+A is loaded from B.

Condition Codes:

H	I	N	Z	V	C	
.	.	1	0	0	.	New value at X+A is negative
.	.	0	1	0	.	New value at X+A is zero
.	.	0	0	0	.	New value at X+A is positive

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	1E	

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

STRK — Compute Stroke

Operation:

```
X+4 --> X
Let: BaseX[15:0] = [[X-4],[X-3]]
     BaseY[15:0] = [[X-2],[X-1]]
     CurrX[15:0] = [[X],[X+1]]
     CurrY[15:0] = [[X+2[,[X+3]]
     DesiredX[15:0] = [[X+4],[X+5]]
     DesiredY[15:0] = [[X+6],[X+7]]
     StrokeX = [A[6:4]]
     StrokeY = [A[3:0]]
     Scale = CASE [B]-1 OF:
          9;   B=1
          8;   B=2
          5.5; B=3
          5;   B=4

BaseX+Scale*StrokeX --> DesiredX
BaseY+Scale*StrokeY --> DesiredY
IF ([A[7]]=1) THEN continue as
     in VECT instruction
```

Description:

Given a stroke from the character stroke table in A and a scale code in B, this instruction computes the desired X and desired Y position for the stroke. If the stroke has the negative bit set, the vector-drawing information needed by the display interface of the 4054 is pushed onto the stack as in VECT.

Condition Codes:

Set to state of A register.

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	71	

TITLE

4052A/4054A Assembler

ABSTRACT NUMBER

TEKniques Vol. 6 No. 4 T1 Program 1

TAP — A --> CC Not including space bits

Operation:

low 6-bits of A --> CC

Description:

Set the CC register to the contents of the A register. The 2 high bits of A are ignored and the 2 high bits of CC are left unchanged.

Condition Codes:

Set to contents of A.

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	06	

TAPX — A --> CC including space bits

Operation:

A --> CC

Description:

Set the CC register to the contents of the A register. All bits are moved.

Condition Codes:

Set to contents of A.

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	12	

Programming Note:

If the F-bit changes, the instruction space change will be deferred as in the SFA instruction.

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

TEST — Microcode restart

Operation:

uCode restart

Description:

This instruction performs a microcode restart without disturbing the hardware.

Condition Codes:

Not affected.

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	00	

TPA — CC --> A Not including space bits

Operation:

low 6-bits of CC --> A
11 --> 2 high bits of A

Description:

Set the A register to the contents of the CC register, except for the 2 high bits.

Condition Codes:

Not affected.

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	07	

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

TPAX CC --> A including space bits

Operation:

CC --> A

Description:

Set the A register to the contents of the CC register. All bits are moved.

Condition Codes:

Not affected.

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	13	

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

VECT — Compute Vector

Operation:

```
Let:    CurrX[15:0]=[[X],[X+1]]
        CurrY[15:0]=[[X+2],[X+3]]
        DesiredX[15:0]=[[X+4],[X+5]]
        DesiredY[15:0]=[[X+6],[X+7]]
        dX[15:0]=DesiredX-CurrX
        dY[15:0]=DesiredY-CurrY
        Direction[7:3]=0
        Direction[2]=Sign(dY)
        Direction[1]=Sign(dX)
        Direction[0]=if (dX>dY) then 1 else 0
        NdX=Abs(dX)*2^(12-Floor(Log2(Max(Abs(dX),Abs(dY)))))
        NdY=Abs(dY)*2^(12-Floor(Log2(Max(Abs(dX),Abs(dY)))))
Push(Max(Abs(dX),Abs(dY)))      ;Vector length parameter
Push(Direction)                 ;Vector direction parameter
Push(NdX)                       ;Normalized X
Push(NdY)                       ;Normalized Y
```

Description:

This instruction pushes onto the stack the vector-drawing information needed by the display interface of the 4054.

Condition Codes:

Set to the state of the A register.

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	72	

Programming Note:

Upon entry to this instruction:

X+6 ->	Desired Y Position
X+4 ->	Desired X Position
X+2 ->	Current Y Position
X+0 ->	Current X Position

Upon exit:

SP+A ->	Vector Length(+A,+B)

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

SP+9 ->	Direction (1 byte)
SP+7 ->	X Rate
SP+5 ->	Y Rate
SP+3 ->	Normalized X
SP+1 ->	Normalized Y
Sp+0 ->	

WADGX — Add G to index extended

Operation:

G + X --> X

Description:

The G accumulator is added to the index register.

Condition Codes:

H	I	N	Z	V	C	
.	.	1	0	x	x	Result is negative
.	.	0	1	x	x	Result is zero
.	.	x	x	1	x	Overflow
.	.	x	x	x	1	Carry out of bit15

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	CD	

Programming Note:

Cannot be used with old 4052/4 (ie. non GX) because interrupts (maskable & nonmaskable) will screw up the high byte of G.

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

WADX — Add memory to index

Operation:

[M[Pc+1,Pc+2]] + X --> X

Description:

The sixteen-bit value in memory is added to the index register.

Condition Codes:

H	I	N	Z	V	C	
.	.	1	0	x	x	Result is negative
.	.	0	1	x	x	Result is zero
.	.	x	x	1	x	Overflow
.	.	x	x	x	1	Carry out of bit15

Particulars:

Addressing Modes	Op	Operand Field Syntax
Extended	ED	<16-bit address>

TITLE

4052A/4054A Assembler

ABSTRACT NUMBER

TEKniques Vol. 6 No. 4 T1
Program 1

General Utility MACROS

There are about a dozen generally used macros here.

Documentation aids:

```
ENTRY  <--> declares a name to be a global
            entry point, and makes it a
            subtitle for that module.
LOCAL  <--> declares a name as a local
            branch point, and makes it a
            subtitle for that module.
HEADER <--> puts copyright info in.  Should
            be used at the front of each
            module.
```

Transfer of control:

```
JMPX     <--> LDXX and then JMP 0,X
CALL Foo <--> JSR Foo
JUMP Foo <--> JMP Foo
```

Stack related:

```
PLRA <--> pull 16 bits from stack and
          store in location DREXTA+1
PLRB <--> pull 16 bits from stack and
          store in location DREXTB+1
```

A vs B space related:

Each is 3 instructions: SDB, an opcode, SDA

```
CMPF <--> opcode is CMP (reg. A or B)
LDAF <--> opcode is LDA (reg. A or B)
LDXF <--> opcode is LDX
TSTF <--> opcode is TST (memory)
```

Unsigned branching tests:

```
BLTU <--> BCS
BLEU <--> BLS
BGTU <--> BHI
BGEU <--> BCC
If T1 - T2 was formed (via SUB, CMP, or SBC)
then  BLTU  branches if:   T1 < T2.
```

For use after BIT tests:

```
BON  <--> BNE
BOFF <--> BEQ
```

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

CHAPTER 2

NEW OPERATIONS FOR 4040A

TITLE

4052A/4054A Assembler

ABSTRACT NUMBER

TEKniques Vol. 6 No. 4 T1
Program 1

ADDG — Add to G Accumulator

Operation:

G+[M,M+1] --> G

Description:

The 16-bit value at M is added to G.

Condition Codes:

H	I	N	Z	V	C	
.	.	1	0	x	x	New value in G is negative (bit 15 is 1)
.	.	0	1	x	x	New value in G is zero
.	.	x	x	1	x	Two's complement overflow
.	.	x	x	x	1	Carry out of bit 15

Particulars:

Addressing Modes	Op	Operand Field Syntax
Direct	87	<8-bit address>,D
Indexed	8F	<8-bit offset>,X
Extended	FC03	<16-bit address>
Immediate	FC02	<16-bit value>,I

Programming Note:

The Immediate and Extended modes of ADDG and SUBG are not speed efficient, since they are implemented as escape codes, and therefore take 4 bytes. In fact, 1 ADDG (extended) is 50% slower than 2 ADD A (extended)! These two modes are included to "complete" the instruction set, and should be used with caution (due to the speed penalty). The rest of the new instructions are as fast or faster (many are twice as fast) than comparable 6800 instruction pairs. But CMPGX is half as fast as CPX (by itself).

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

BUFIN — Read a buffer from the GPIB

On entry:

```
G   -- maximum address for the buffer
B   -- flags indicating:
       80H bit indicates Ascii read operation
       40H bit indicates Secret read operation
          other bits - don't care
X -- address for start of buffer
dmyWs -- byte at ^h00FC contains 9914 status

The following parameters are necessary only for
ascii mode transfers:

M[Sp+1] -- etxchr, end of file character
M[Sp+2] -- eolchr, end of line character
M[Sp+3] -- nulchr, character to be ignored
           negative nulchr indicates ignore none
```

On exit:

```
X   -- returns the address of one past the last valid
       byte in the I/O buffer. (even if timed out)
B   -- 80H bit unchanged
       40H bit unchanged
       20H bit - timed out:  about 1 mS. passed without
                 a byte transfered.
       10H bit - EOI detected  terminated transfer
       08H bit - ETX character terminated transfer
       04H bit - EOL character terminated transfer
       02H bit - zero
       01H bit - zero
Sp  -- unchanged
G   -- unchanged
dmyWs  -- byte at ^HFC is restored with new status OR'ed i
```

Description:

BUFIN reads bytes from the GPIB bus and transfers them into a buffer in memory. If the transfer is in ascii mode then end of line, end of file and null characters are handled properly. The top bit of ascii data is stripped off.

At entry, we should have been initialized as a listene with no holdoff in effect, and a transfer in progress (ie. we check for the BI flag BEFORE we read the byte) We also expect to be in hdfa mode (holdoff on all data

The buffer should be at least one byte long.

At exit with a timeout condition, a transfer is in progress and the bus must be cleared by firmware.

At normal exit, the last transfer has completed and a holdoff is in effect which must be cleared by firmware.

TITLE

4052A/4054A Assembler

ABSTRACT NUMBER

TEKniques Vol. 6 No. 4 T1
Program 1

Condition Codes:

H	I	N	Z	V	C	
.	.	0	0	0	0	Terminated with buffer full
.	.	0	0	0	1	Terminated on EOI, EOL or ETX
.	.	0	1	0	0	Timeout occurred before normal termination (1 mS. passed between successive bytes)

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	C706	

Algorithm:

```
PROCEDURE BufIn
BEGIN
  {reset n,z,v,c}
  b := b AND ^HC0                    ; mask off all but ascii & secret
  stat := MEM[^H00FC]                ; dmyWs
  IF (b AND ^H80
  THEN BEGIN                         ; ascii
    etx := MEM[SP+1]
    eol := MEM[SP+2]
    nul := MEM[SP+3]
  END
  LOOP
    time := {1 mS. number}
    LOOP
      IF time = 0                    ; timed out
      THEN BEGIN
        B := B OR ^H20
        {set z bit}                  ; BEQ will pass on exit
        GOTO end
      END
      temp := intSt0 AND ^H38        ; 9914 interrupt Status 0
      stat := stat OR temp           ; form combined status
    WHILE NOT (stat AND ^H20)        ; check Byte In (BI) bit
      time := time-1
    REPEAT
    IF (stat AND ^H08)
    THEN BEGIN
      {set c bit}                    ; check for EOI with this byte
      b := b OR ^H10
    END
    data := ti9914.dataIn
    stat := stat AND ^H00D7          ; clear BI and End bits
    IF (b AND ^H80)
    THEN BEGIN                       ; ASCII transfer requested
```

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

```
      IF (data = etx)
      THEN BEGIN                          ; check for ETX character
        {set c bit}
        b := b OR ^H08
        GOTO end
      END
      data := data AND ^H7F               ; strip off top bit
      IF (data <> nul)
      THEN BEGIN
        IF (data = eol)                   ; not null - check for EOL
        THEN BEGIN
          {set c bit}                     ; eat EOL character and exit
          b:=b OR ^H04
          GOTO end
        END
        ELSE BEGIN                        ; not nul or eol
          MEM[x] := data
          x := x+1
        END
      END
    END
    ELSE BEGIN                            ; Binary transfer
      MEM[x] := data
      x := x+1
    END
  WHILE NOT (b AND ^H10)                  ; was EOI found?
  WHILE x <= g
    9914.auxCmd := rhdf                   ; release holdoff - we want mor
  REPEAT
end:  MEM[^H00FC] := stat                 ; restore status byte
END
```

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

BUFOUT — Write a buffer to the GPIB

On entry:

```
G   -- ending address for the buffer
B   -- 80H bit indicates EOI to be sent with last byte
       other bits ignored
X   -- address for start of buffer
dmyWs -- byte at ^H00FC containing 9914 status
```

On exit:

```
X -- buffer address of last byte put on bus. (after a
     timeout, transfer of this byte was not completed)
B -- 80H bit - unchanged
     40H bit - zero
     20H bit - timed out.  a byte took longer than 1 mS

     10H bit - zero
     08H bit - zero
     04H bit - zero
     02H bit - zero
     01H bit - zero
G -- unchanged
dmyWs -- byte at ^H00FC contains updated 9914 status
```

Description:

BUFOUT writes bytes to the GPIB bus from a buffer in memory. Bus timeouts of approximately 1 mS. cause the instruction to terminate to facilitate timeouts to be trapped by BASIC. The instruction expects the buffer to be at least one byte long.

When first called, BUFOUT expects the bus to be initialized as a talker, with the BO bit cleared.

Condition Codes:

H	I	N	Z	V	C	
.	.	0	0	0	0	Transfer completed without errors
.	.	0	1	0	0	Transfer of a byte timed out after about 1 mS.

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	C707	

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

Algorithm:

```
      PROCEDURE BufOut
      BEGIN
        {reset n,z,v,c bits}
        b := b AND ^H80
        stat := MEM[^H00FC]                          ; get existing status
        LOOP
          IF (x=g AND b<0) THEN ti9914.auxCmd := feoi
          ti9914.dataOut := MEM[x]
          time:= {magic 1 mS. number}
          LOOP
            IF time=0
            THEN BEGIN
              b:=b OR ^H20
              {set z bit}
              GOTO end
            END
            temp := intSt0 AND ^H38                  ; get new status
            stat := stat OR temp                     ; form composite status
          WHILE (stat AND ^H40) = 0                  ; test for xfer complete
            time := time-1
          REPEAT
          stat := stat AND ^HEF                      ; clear BO flag only
        WHILE x <> g
          x : x+1
        REPEAT
end:    MEM[^H00FC] := stat
      END
```

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

CLRGH — Clear High Byte of G

Operation:

0 --> high byte of G

Description:

Clears the high byte of G and sets condition codes based on the entire (new) 16-bit value in G. The low byte of G is unchanged.

Condition Codes:

H	I	N	Z	V	C	
.	.	1	0	0	.	Value in G is negative (can not happen)
.	.	0	1	0	.	Value in G is zero
.	.	0	0	0	.	Value in G is positive

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	C702	

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

CMPGX — Compare G and X (i.e. G-X)

Operation:

G-X

Description:

The index register is subtracted from the G accumulator, and the condition codes are set accordingly.

Condition Codes:

H	I	N	Z	V	C	
.	.	1	0	x	x	G < X
.	.	0	1	x	x	G = X
.	.	x	x	1	x	Two's complement overflow
.	.	x	x	x	1	Carry out of bit 15

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	FC06	

Programming Note:

WARNING: CMPGX is half as fast as CPX (all forms), but is comparable to the pair: STAG / CPX

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

CMPSYM — Compare the Name in a Symbol Table Record with the Name in a String

Operation:

G -- pointer to start of the string
B -- length of the name in the string
X -- pointer to symbol table record
The name in the record is compared against the name in the string, and the condition codes are set accordingly. The name in the record is found by:
  if high bit of [X+3] = 0 then the name is in [X+2,X+3]
  otherwise, the low 5 bits of [X+3] are the length of the name, and the characters of the name are at [X+13,X+14,...]
G, B, and X are unchanged by CMPSYM.

Description:

The name in the string is compared against the name in the record. Note that B must be >=2 if the name is in [X+2,X+3]! (i.e. the user of CMPSYM must pad the string name with a blank if the name is only 1 character long)
Doing:

```
CMPSYM
BGT
```

will branch if the string name is greater than the record name. Similarly for:
BLT, BEQ, BGT, BLE, BGE, BNE.

Condition Codes:

H	I	N	Z	V	C	
.	.	1	0	0	.	string name < record name
.	.	0	1	0	.	names are equal
.	.	0	0	0	.	string name > record name

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	FC07	

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

DEVIN — Read a buffer from an I/O device

On entry:

G -- Byte count - number of bytes to be transfered to the I/O device. (>= 1 byte)

X -- address for start of buffer

M[Sp+1, Sp+2] -- Address of I/O Device.

On exit:

X -- returns the address of the last buffer entry
Sp -- Sp+2
G -- Zero

Description:

DEVIN transfers bytes from an I/O device in data space to a buffer in memory.

Condition Codes:

H	I	N	Z	V	C
.	.	x	x	0	0

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	C709	

NOTE: Interupts are not serviced until this uCode instruction completes the data transfer.

Algorithm:

```
PROCEDURE devin
BEGIN
  ioadr := MEM16[SP+1]
  LOOP
    temp := MEM[ioadr]
    MEM[X] := temp
    G := G-1
  WHILE G <> 0
    X := X+1
  REPEAT
  SP := SP+2
END
```

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

DEVOUT — Write a buffer to an I/O device

On entry:

G -- Byte count - number of bytes to be transfered to the I/O device. (>= 1 byte)

X -- address for start of buffer

M[Sp+1, Sp+2] -- Address of I/O Device.

On exit:

X -- returns the address of the last buffer entry
Sp -- Sp+2
G -- Zero

Description:

DEVOUT transfers bytes from a buffer in memory to an I/O device in data space.

Condition Codes:

H	I	N	Z	V	C
.	.	x	x	0	0

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	C70A	

NOTE: Interupts are not serviced until this uCode instruction completes the data transfer.

Algorithm:

```
PROCEDURE devout
BEGIN
  ioadr := MEM6[SP+1]
  LOOP
    temp ::= MEM[X]
    MEM[ioadr] := temp
    G := G-1
  WHILE G <> 0
    X := X+1
  REPEAT
  SP := SP+2
END
```

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

FIXRND — Round FP to integer

On entry:

```
[X]   -- tag for floating point
[X+1] -- high byte of exponent
[X+2] -- low  byte of exponent
[X+3] -- Most significant byte of mantissa
   .
   .
   .
[X+8] -- Least significant byte of mantissa
```

On exit:

```
A      -- 0    if no error
          #H60 if absolute value overflows 2^16-1
          #H5F if integer is negative
X      -- unchanged
[X]    -- unchanged
[X+3]  -- High byte of integer
[X+4]  -- Low  byte of integer
```

Description:

FIXRND rounds a standard floating point number pointed to by the index register, and converts it into an unsigned integer pointed to by the index registe

For negative values, the absolute value is returned.

If the absolute value overflows, then the integer returned is #H0FFFF.

An error code is returned in register A.

Condition Codes:

H	I	N	Z	V	C	
.	.	0	1	0	0	No error occured
.	.	0	0	0	0	Error occurred

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	C704	

Programming Note:

A JSR FIX1 can be replaced with:

```
   FIXRND
   BEQ   1$
   STA A ERRCD,D
1$: ...   ; the line after JSR FIX1
```

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

IFLOAT — Convert integer to FP

On entry:
```
M[Sp+1] -- tag for integer
M[Sp+2] -- high byte of integer
M[Sp+3] -- low  byte of integer
```

On exit:
```
A       -- crashed
B       -- number of shifts used to normalize result
Sp      -- Sp - 6
M[Sp+1] -- tag for floating point
M[Sp+2] -- high byte of exponent
M[Sp+3] -- low  byte of exponent
M[Sp+4] -- most significant byte of mantissa
    .
    .
    .
M[Sp+9] -- least significant byte of mantissa
```

Description: IFLOAT converts an unsigned integer on the stack into a floating point number on the stack.

Condition Codes:

H	I	N	Z	V	C	
.	.	0	1	0	0	Result is zero
.	.	0	0	0	0	Result is positive
.	.	0	1	1	0	Underflow. Result zero, Floating Fault Interrupt

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	C703	

Programming Note:

A JSR FLOAT can be replaced with:
```
   IFLOAT
```

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

INXSTX — Increment Index Register And Store It

Operation:

X+1 --> X
X --> [M,M+1]
where M must be a "direct" address

Description:

This one instruction executes the same as the pair:

```
   INX                      <--> INXSTX t
   STX t, D
```

Condition Codes:

H	I	N	Z	V	C	
.	.	1	0	0	.	new value of bit 15 in X is 1
.	.	0	1	0	.	New value of X is zero
.	.	0	0	0	.	New value of X is positive (i.e. non-zero and bit 15 = 0)

Particulars:

Addressing Modes	Op	Operand Field Syntax
Direct	A3	<8-bit address>

TITLE

4052A/4054A Assembler

ABSTRACT NUMBER

TEKniques Vol. 6 No. 4 T1
Program 1

LDAG — Load G Accumulator

Operation:

[M,M+1] --> G

Description:

The 16-bit value at M is loaded into G.

Condition Codes:

H	I	N	Z	V	C	
.	.	1	0	0	.	New value in G is negative (bit 15 is 1)
.	.	0	1	0	.	New value in G is zero
.	.	0	0	0	.	New value in G is positive

Particulars:

Addressing Modes	Op	Operand Field Syntax
Direct	04	<8-bit address>,D
Indexed	05	<8-bit offset>,X
Extended	B3	<16-bit address>
Immediate	D3	<16-bit value>,I

LDAGX — Load G Accumulator Double-Indexed

Operation:

[X+B,X+B+1] --> G

Description:

The 16-bit value at X+B is loaded into the G accumulator. (Only the low byte of B is used!)

Condition Codes:

H	I	N	Z	V	C	
.	.	1	0	0	.	New value in G is negative (bit 15 is 1)
.	.	0	1	0	.	New value in G is zero
.	.	0	0	0	.	New value in G is positive

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	FC08	

TITLE: 4052A/4054A Assembler

ABSTRACT NUMBER: TEKniques Vol. 6 No. 4 T1 Program 1

PSHG — Push G on the Stack

Operation:

Sp-2 --> Sp
G --> M[Sp+1],M[Sp+2]

Description:

The 16-bit G accumulator is pushed on the stack.

Condition Codes:

Not affected

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	FC00	

PULG — Pull G From the Stack

Operation:

M[Sp+1],M[Sp+2] --> G
Sp+2 --> Sp

Description:

The 16-bit G accumulator is pulled from the stack.

Condition Codes:

Not affected.

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	FC01	

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

SEABNK — Search for a CALL name in a ROM bank

Operation:

Absolute addresses 2,3,4,5,6,7 in the current Data-space contain the desired CALL name.
X (input) -- points at a table (see below) in B-space which is searched for the CALL name.
X (output)-- if the name is found then X has the address for the CALL routine (from the table). If the name is not found then X is undefined.

Description:

The table in B-space pointed at by X has the format:

```
Filler: 13. bytes at the front of the table
Item1 = Record
   Name: 6 bytes  { 1st byte is <> 0     }
   Addr: 2 bytes  { addr of this routine }
End  { Item1 }
Item2 = ...
   .  .  .        { each Item is 8 bytes }
   .  .  .
ItemN = ...
Terminator: byte = 0
```

SEABNK searches for an Item in the table whose Name is the same as in 2..7. The Name and 2..7 both contain only uppercase letters, so a strict numeric comparison for equality (comparing two bytes at a time) is used.

Condition Codes:

H	I	N	Z	V	C	
.	.	.	1	.	.	the name was not found
.	.	.	0	.	.	the name was found

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	C708	

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

STAG — Store G Accumulator

Operation:

G --> [M,M+1]

Description:

The 16-bit value from G is put into M.

Condition Codes:

H	I	N	Z	V	C	
.	.	1	0	0	.	Value in G is negative (bit 15 is 1)
.	.	0	1	0	.	Value in G is zero
.	.	0	0	0	.	Value in G is positive

Particulars:

Addressing Modes	Op	Operand Field Syntax
Direct	7B	<8-bit address>,D
Indexed	83	<8-bit offset>,X
Extended	C3	<16-bit address>

STAGX — Store G Accumulator Double-Indexed

Operation:

G --> [X+B,X+B+1]

Description:

The 16-bit value from G is stored at X+B.
(Only the low byte of B is used!)

Condition Codes:

H	I	N	Z	V	C	
.	.	1	0	0	.	Value in G is negative (bit 15 is 1)
.	.	0	1	0	.	Value in G is zero
.	.	0	0	0	.	Value in G is positive

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	FC09	

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

SUBG — Subtract From G Accumulator

Operation:

G - [M,M+1] --> G

Description:

The 16-bit value at M is subtracted from G.

Condition Codes:

H	I	N	Z	V	C	
.	.	1	0	x	x	New value in G is negative (bit 15 is 1)
.	.	0	1	x	x	New value in G is zero
.	.	x	x	1	x	Two's complement overflow
.	.	x	x	x	1	Carry out of bit 15

Particulars:

Addressing Modes	Op	Operand Field Syntax
Direct	93	<8-bit address>, D
Indexed	9D	<8-bit offset>,X
Extended	FC05	<16-bit address>
Immediate	FC04	<16-bit value>,I

Programming Note:

The Immediate and Extended modes of ADDG and SUBG are not speed efficient, since they are implemented as escape codes, and therefore take 4 bytes. In fact, 1 ADDG (extended) is 50% slower than 2 ADD A (extended)! These two modes are included to "complete" the instruction set, and should be used with caution (due to the speed penalty). The rest of the new instructions are as fast or faster (many are twice as fast) than comparable 6800 instruction pairs. But CMPGX is half as fast as CPX (by itself).

TGX — Transfer G to the Index Register

Operation:

G --> X

Description:

The 16-bit G accumulator is transferred to the index register.

Condition Codes:

H	I	N	Z	V	C	
.	.	1	0	0	.	Value in G is negative (bit 15 is 1)
.	.	0	1	0	.	Value in G is zero
.	.	0	0	0	.	Value in G is positive

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	C700	

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

TMULT — Mult a 6 byte integer by 10

On entry:

A -- bias to be added to result
MEM[X5] -- most significant byte of integer
.
.
MEM[X0] -- least significant byte of integer

X5 corresponds to A space address 00CE, X4 is 00CF and so on through X0 which is 00D3.

On Exit:

B -- "carry" digit (0..10)
MEM[X5] -- most significant byte of result
.
.
MEM[X0] -- least significant byte of result.

Description:

TMULT multiplies a 6 byte unsigned integer located at fixed system addresses X5 throuth X0 by a constant decimal 10, and then add the bias A.

Condition Codes:

crashed

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	C705	

TXG — Transfer the Index Register to G

Operation:

X --> G

Description:

The 16-bit G accumulator is loaded from the index register.

Condition Codes:

H	I	N	Z	V	C	
.	.	1	0	0	.	Value in G is negative (bit 15 is 1)
.	.	0	1	0	.	Value in G is zero
.	.	0	0	0	.	Value in G is positive

Particulars:

Addressing Modes	Op	Operand Field Syntax
Inherent	C701	

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

CHAPTER 3

EXTENSIONS TO 6800 INSTRUCTIONS FOR 4050A

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

ABA — Add Accumulator B to A

Operation:

(16-bit B) + (16-bit A) --> (16-bit A)

Description:

The 16-bit B is added to the 16-bit "A". The condition codes are based on the low byte of A only (same as in normal 6800, which is why only low byte B is used).

Condition Codes:

See 6800 manual.

Particulars:

See 6800 manual.

ADD — Add to Accumulator (A or B)

Operation:

low ACCX + [M] --> low ACCX
high ACCX + carry --> high ACCX, normally
But in indexed addressing mode:
Trash bits --> high ACCX

Description:

Add an 8-bit value to the 16-bit "A" or "B". Condition codes are based on the low byte of A (or B) only! (same as in normal 6800)

Condition Codes:

See 6800 manual.

Particulars:

See 6800 manual.

Programming Note:

The high byte of A (or B) is trashed by:

```
ADD  ,X
LDA  ,X
SUB  ,X
PUL
```

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

ASL — Arithmetic Shift Left (A or B)

Operation:

(16-bit ACCX)*2 --> 16-bit ACCX

Description:

The 16 bits in "A" or B are shifted left one bit. The low bit is zeroed. The Carry bit and other condition codes are based on the low byte of A (or B) only! (same as in normal 6800)

Condition Codes:

See 6800 manual.

Particulars:

See 6800 manual.

CLR — Clear Accumulator (A or B)

Operation:

0 --> ACCX low byte
0 --> ACCX high byte

Description:

Clears the entire 16 bits of "A" (i.e. G) or B.

Condition Codes:

See 6800 manual.

Particulars:

See 6800 manual.

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

COM — Complement Accumulator (A or B)

Operation:

"flip" all bits in ACCX

Description:

Form a 1's complement of A or B.
Condition codes are based on the low byte of A (or B) only! (same as normal 6800)

Condition Codes:

See 6800 manual.

Particulars:

See 6800 manual.

DEC — Subtract One From Accumulator (A or B)

Operation:

16-bit ACCX -1 --> 16-bit ACCX

Description:

Subtract one from the 16-bit "A" or "B".
Condition codes are based on the low byte of A (or B) only! (same as normal 6800)

Condition Codes:

See 6800 manual.

Particulars:

See 6800 manual.

Programming Note:

DEC(0) --> FFFF

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

INC — Add One to Accumulator (A or B)

Operation:

low ACCX + 1 --> low ACCX
high ACCX + carry --> high ACCX

Description:

Add one to the 16-bit "A" or "B". Condition codes are based on the low byte of A (or B) only! (same as normal 6800)

Condition Codes:

See 6800 manual.

Particulars:

See 6800 manual.

Programming Note:

INX(FFFF) --> 0

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

LDA — Load Accumulator (A or B)

Operation:

[M] --> ACCX low byte
0 --> ACCX high byte, normally
But in indexed addressing mode:
Trash bits --> ACCX high byte

Description:

Operates the same as the normal 6800 instruction, and the high byte of A (or B) is zeroed. Condition codes are based on low byte of A (or B) only! (same as in normal 6800)

Condition Codes:

See 6800 manual.

Particulars:

See 6800 manual.

Programming Note:

The high byte of A (or B) is trashed by:

```
ADD  ,X
LDA  ,X
SUB  ,X
PUL
```

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

PUL — Pull Accumulator From Stack (A or B)

Operation:

```
M[Sp+1] --> ACCX low byte
Sp+1    --> Sp
Trash bits --> ACCX high byte
```

Description:

Operates the same as the normal 6800 instruction, and the high byte of A (of B) is TRASHED! Condition codes are based on low byte of A (or B) only! (same as in normal 6800)

Condition Codes:

See 6800 manual.

Particulars:

See 6800 manual.

Programming Note:

The high byte of A (or B) is trashed by:

```
    ADD  ,X
    LDA  ,X
    SUB  ,X
    PUL
```

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

RTI — Return from Interrupt

Operation:

pop 11 bytes off the stack to restore:
- CC (1st popped)
- Low B
- Low A
- High X
- Low X
- High PC
- Low PC
- High B
- Low B (popped & ignored by RTI)
- High A (G register)
- Low A (popped & ignored by RTI)

Description:

RTI pops 11 bytes (6800 popped only 7) to restore the hardware registers to the state they were before an interrupt occurred (or SWI [ODT only] or WAI [not used in 4052]). When the interrupt occurred the CC was pushed onto the stack and then the D and F bits in CC were set to 1 (1 --> Fetch B and Data A).

Condition Codes:

See 6800 manual.

Particulars:

See 6800 manual.

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

SBA — Subtract Accumulator B From A

Operation:

(16-bit A) - (16-bit B) --> (16-bit A)

Description:

The 16-bit B is subtracted from the 16-bit "A". Condition codes are based on the low byte of A only! (same as in normal 6800)

Condition Codes:

See 6800 manual.

Particulars:

See 6800 manual.

SUB — Subtract From Accumulator (A or B)

Operation:

(16-bit ACCX) - [M] --> (16-bit ACCX), normally
But in indexed addressing mode:
Trash bits --> high ACCX

Description:

Subtract an 8-bit value from the 16-bit "A" or "B". Condition codes are based on the low byte of A (or B) only! (same as in normal 6800)

Condition Codes:

See 6800 manual.

Particulars:

See 6800 manual.

Programming Note:

The high byte of A (or B) is trashed by:

```
ADD  ,X
LDA  ,X
SUB  ,X
PUL
```

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

TAB — Transfer Accumulator A to B

Operation:

16-bit A --> 16-bit B

Description:

16-bit A (i.e. G) is transferred to the 16-bit B, but the condition codes are set based on low byte of B only! (same as in normal 6800)

Condition Codes:

See 6800 manual.

Particulars:

See 6800 manual.

TBA — Transfer Accumulator B to A

Operation:

16-bit B --> 16-bit A

Description:

16-bit B is transferred to the 16-bit A (i.e. G), but the condition codes are set based on low byte of A only! (same as in normal 6800)

Condition Codes:

See 6800 manual.

Particulars:

See 6800 manual.

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

4052A/4054A JUMP TABLE

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

ADDRESS (HEX)	NAME	DESCRIPTION
4000	Adrdev	; Address a device
4003	Afpitt	; Convert ASCII to floating point number
4006	Appoid	; Do the I/O portion of APPEND
4009	Ascfpn	; Convert string of ASCII chars into FPN
400C	Atnoff	; Clear ATN on GPIB
400F	Atnon	; Assert ATN on GPIB
4012	Backup	; Back up one stack entry using R0
4015	Bakars	; Back up one record on magtape
4018	Belcal	; Ring the bell
4018	Bfralc	; Allocate I/O buffers using A.PRIM
401E	Kbqin	; ?
4021	Cidle	; Display cursor while waiting for keyboard I/O
4024	Clrarg	; Clear argument entry point to TYPARG
4027	Clrbnk	; Set the bank switch to zero
402A	Crlf	; Send CR to output buffer
402D	Crlflf	; Send two CR's to output buffer
4030	Crtrst	; Restore shared PIAs to CRT
4033	Ctlchr	; Display control character on screen
4036	Datin	; Read from a DATA statement
4039	Defpnt	; Default PRINT formatter
403C	Delay	; Delay number of MS in IX register
403F	Delet	; Delete program lines
4042	Delyls	; wait some time
4045	Dely3s	; Another wait routine
4048	Dimstr	; Dimension a string variable
4048	Disble	; No BREAK BREAK allowed
404E	Diskci	; Call disk interface ROMpack
4051	Dltall	; Perform the basic DELETE ALL function
4054	Drbusy	; wait for display to be ready
4057	Dsdraw	; Send GDU numbers to display vectors (DRAW)
405A	Dshome	; Home the display cursor
405D	Dsmove	; Send GDU numbers to display vectors (MOVE)
4060	Dspage	; Send FF (page) command to display
4063	Dspchr	; Send char to display
4066	Dspcpy	; Make copy of display
4069	Dspout	; Output to display
406C	Edtclr	; Clear the line buffer
406F	Edtcls	; Close the line buffer
4072	Enable	; Allow BREAK BREAK
4075	Eoioff	; Release EOI control line on GPIB
4078	Eoion	; Assert EOI control line on GPIB
407B	Flxl	; Convert a floating point number to an integer
407E	Flxnum	; Fix I/O list entries for internal use
4081	Float	; Convert an integer to a floating point number
4084	Fmtpnt	; Formatted PRINT handler
4087	Fpaitt	; Convert FPN to ASCII (internal buffers)
408A	Fpcmp	; Compare FP numbers on stack
408D	Fpnasc	; Reduce FPN to fraction for later ASCII conversion
4090	Fpneg	; Negate FPN on stack
4093	Fulscn	; Scan blinking F while waiting for PAGE key
4096	Fuzzle	; Do a fuzzle compare on stack
4099	Gencur	; Generate a blinking cursor
409C	Getchr	; Get next character from input buffer

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

```
409F    Getkey  ; Get key interrupt
40A2    Getlar  ; Locate next larger basic line
40A5    Getln   ; Convert FPN to address of given line
40A8    Getlna  ; Convert KO to address of given line
40AB    Getsma  ; Locate next smaller BASIC line
40AE    Ginset  ; Set up pointers for GIN input
40B1    Grfar.  ; Perform clipping on vectors
40B4    Grfini  ; Reset graphic system default values
40B7    Halta   ; Halt and take DREXTA
40BA    Haltr   ; Halt and return
40BD    Idle    ; Main basic idle loop entry point
40C0    Ieceoi  ; Service EOI hardware interrupts
40C3    Iecifc  ; Assert IFC on GPIB
40C6    Iecin   ; Input a character from the GPIB
40C9    Iecoff  ; Get off the GPIB
40CC    Iecout  ; Output a character to the GPIB
40CF    Iecrd   ; Read from the GPIB
40D2    Iecsnd  ; Send to the GPIB
40D5    Inagte  ; Convert integer to ASCII
40D8    Inaitt  ; Convert ASCII to a floating point number
40DB    Info    ; Return OPTEL info about a BASIC token
40DE    Initmt  ; Switch shared PIAs to magtape
40E1    Inpctl  ; Input controller
40E4    Inpstg  ; Input a string
40E7    Inpval  ; Input a value
40EA    Intacp  ; Set GPIB hardware to LISTEN state
40ED    Intcn   ; Float and stack an integer
40F0    Intmla  ; Integer multiply with accumulator
40F3    Intmlc  ; Integer multiply with no accumulate
40F6    Intsrc  ; Initialize GPIB hardware to source state
40F9    Iocinr  ; Clean up stack after I/O and return to original caller
40FC    Iodark  ; Turn off the I/O light on the front panel
40FF    Iolite  ; Turn on  the I/O light on the front panel
4102    Ioscan  ; I/O list scanner
4105    Kbqout  ; Fetch a key from the type-ahead queue
4108    Keyin   ; Input key
4108    Lex     ; Convert input lines from ASCII to postfix internal form
410E    Lincn   ; Float a line number and stack it
4111    Litcn   ; Push pointer to literal
4114    Litrel  ; Literal relation - string compare
4117    Lnevl   ; Evaluate one BASIC line
411A    Loctg   ; Locate a tagged stack entry
411D    Loctgr  ; Locate a tag in a range
4120    Marks   ; Mark new magtape files
4123    Matmat  ; Apply a scaler operator over two arrays
4126    Matmov  ; Assign one matrix to another
4129    Matscl  ; Apply a scaler to an array and a scaler
412C    Matsiz  ; Calculate control constants for a matrix
412F    Maxans  ; Push largest FP number on stack
4132    Mkfile  ; Mark a magtape file
4135    Modmth  ; Modify magtape header
4138    Modt88  ; Special entry to modify magtape header
413B    Mtafin  ; Locate and open a new magtape file
```

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

```
413E    Mtbswp  ; Swap magtape buffer pointers
4141    Mtclos  ; Close a magtape file
4144    Mtfind  ; Find the file specified on the stack
4147    Mtkill  ; Kill a file on the magtape
414A    Mtmark  ; Mark a file on the magtape
414D    Mtnull  ; Null the magtape buffer
4150    Mtpadr  ; Address the magtape
4153    Mtpin   ; Get a logical buffer from the magtape
4156    Mtpinr  ; Magtape input request validity interpreter
4159    Mtpinw  ; Magtape output request validity interpreter
415C    Mtpout  ; Send a buffer to the magtape
415F    Mtprwd  ; Controlled rewind of the magtape
4162    Mtrbfr  ; Controlled read of a magtape buffer
4165    Mtread  ; Read a record from the magtape
4168    Mtsset  ; Set the magtape format status register
416B    Mtwrit  ; Write a record to the magtape
416E    Mtwrt   ; Ditto
4171    Newbfr  ; Get a new input buffer
4174    Newtap  ; Swap in magtape hardware, find self on new tape
4177    Nfr8    ; Position magtape back in file gap
417A    Niodev  ; Set no I/O device error message
417D    Oldone  ; Special entry point for OLD control
4180    Outbfr  ; Send a full output buffer to the current I/O device
4183    Outctl  ; Output controller
4186    Pchar   ; Put a char to display
4189    Pgmevl  ; Program evaluator
418C    Pgmlis  ; Insert a new line into the current program
418F    Plotvl  ; Convert user space to screen space
4192    Popes   ; Pop evaluator status
4195    Pristg  ; Print a string
4198    Prival  ; Print a value
419B    Prterr  ; Print an error message
419E    Puptap  ; Initialize magtape hardware
41A1    Pushes  ; Push evaluator status
41A4    Pushx   ; Push index register and Acc A on stack
41A7    Pushxt  ; Push index register and give it a special tag
41AA    Putbyt  ; Put a character in the output buffer
41AD    Rbytvl  ; Get a single byte from the GPIB in RBYTE format
41B0    Rdfilh  ; Read and validate file header record
41B3    Readrs  ; Read a physical magtape record
41B6    Reastg  ; Read a string
41B9    Reaval  ; Read a value
41BC    Renum   ; Renumber a BASIC program
41BF    Restz   ; Reset the DATA statement pointers
41C2    Rewind  ; Rewind the tape
41C5    Rndata  ; Round FPN and convert to ASCII
41C8    Scimat  ; Apply a scaler operator over a scaler and an array
41CB    Serchs  ; Fast search on the magtape
41CE    Setarg  ; Routine to prime TYPARG calls
41D1    Skips   ; Skip a magtape record
41D4    Sndbfr  ; Send a buffer to a device
41D7    Sngmat  ; Apply a monadic scaler operator over one array
41DA    Spolon  ; Send Serial Poll Enable
```

TITLE

4052A/4054A Assembler

ABSTRACT NUMBER

TEKniques Vol. 6 No. 4 T1
Program 1

```
41DD    Squisn  ; Compress user memory
41E0    Srqoff  ; Reset SRQ pending bit
41E3    Srqrdy  ; Process SRQ level requests
41E6    Stkbld  ; Build special stack entries for I/O system
41E9    Storit  ; Update magtape control register
41EC    Stpmts  ; Stop motor
41EF    Symtab  ; Symbol table handler
41F2    Syserr  ; Flame out for FATAL FATAL error
41F5    Tape    ; Process MARK, FIND commands
41F8    Tapfil  ; Similar to Tapf2
41F8    Testcs  ; Conditional upcase depending on SET CASE/NOCASE
41FE    Tnims   ; Wait for no tape motion
4201    Transl  ; Translator mainline
4204    Trutn   ; Determine if FPN closer to 0 or 1
4207    Tsteof  ; Test for End of File conditions
420A    Tstint  ; Test for raised ON conditions and pending user keys
420D    Tst8nf  ; Alternate action based on file gap size
4210    Tutss   ; Wait for motor up to speed
4213    Typarg  ; Determine type of argument
4216    Typext  ; Special entry point in MTCTL
4219    Typin   ; Line editor for basic
421C    Typres  ; Test for valid result areas for operands
421F    Unadr   ; Release a device
4222    Uncomp  ; Convert postfix internal lines to ASCII
4225    Upcase  ; Convert lower case to upper case
4228    Vector  ; Send a vector to the screen
422B    Wbytvl  ; Send one byte from the WBYTE list
422E    Wrlstg  ; Write a string
4231    Wrlval  ; Write a value
4234    Wrrs    ; Write a physical record
4237    Xfrctl  ; Attach an I/O driver given A.PRIM and a table pointer
423A    Zans    ; Put a floating point 0 on stack

; Vectors into COMM IF modules

423D    Ain     ; Input a char from channel A
4240    Aout    ; Output a char into channel A
4243    Cmrpia  ; ?
4246    Cmspia  ; ?
4249    Commem  ; ?
424C    Mvbytf  ; ?
424F    Mvbyte  ; ?
4252    Prtmsf  ; ?
4255    Prtmsg  ; ?
4258    Resreg  ; ?
425B    Rsbank  ; ?
425E    Savreg  ; Save registers
4261    Tbadd   ; Two byte add
4264    Tbaddi  ; Two byte add immediate
4267    Tbcmp   ; Two byte compare
426A    Tbcmpi  ; Two byte compare immediate
426D    Tbsub   ; Two byte subtract
```

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

```
; Vectors into the 4054/refresh option modules

4270    Appn8x  ; Append to an object
4273    Blnk8x  ; Blink an object
4276    Bri8x   ; Set intensity and focus
4279    B54elc  ; Ring the bell
427C    Clea8x  ; Re-initialize
427F    Clos8x  ; Close an open object
4282    Cro     ; POINTER command
4285    C54rtr  ; Restore display hardware
4288    Drag8x  ; Drag an object
428B    Draw8x  ; Refresh draw
428E    Dsinit  ; Initialize the display
4291    Dspage  ; Page the screen
4294    D54nco  ; Hardcopy 4054 screen load
4297    D54rbu  ; wait for display not busy
429A    D54sho  ; Home the cursor
429D    Fixv    ; Lindsay's funny fixer
42A0    Fix8x   ; Fix an object
42A3    Fndobj  ; Pointer locate
42A6    Hndshk  ; Handshake with refresh
42A9    Kill8x  ; RDElete an object
42AC    Loca8x  ; Return object's setpoint
42AF    Mem8x   ; Return refresh memory left
42B2    Oedini  ; Initialize display for Opt. 1 and editor
42B5    Oedqui  ; Clean up after Option 1 or Editor
42B8    Opc8x   ; Open an object
42BB    Open8x  ; Open an object
42BE    Pick8x  ; Set the Pointer object
42C1    Poin8x  ; Refresh point
42C4    Post8x  ; Post object in X
42C7    Prat24  ; Do PRI@32,24:
42CA    Pwru54  ; Power up 4054
42CD    P54cha  ; Print a character on the screen
42D0    Rebild  ; Rebuild the cursor object
42D3    Rep18x  ; Replace an object with another
42D6    Revxfm  ; Reverse transform X from 63/64 to 4096 space
42D9    Rfidle  ; Refresh replacement for CIDLE
42DC    Rfshnd  ; Force END to refresh
42DF    Rfshrs  ; Restore refresh context
42E2    Rfshsv  ; Save refresh context
42E5    Setmsk  ; Set storage dash mask
42E8    Setp8x  ; Setpoint an ooject
42EB    Shacmd  ; Handshake command with 4054 display
42EE    Spac8x  ; Return refresh memory used
42F1    Unpo8x  ; Unpost object in X
42F4    Unsyob  ; Unpost all system objects
42F7    Vctwt   ; Draw a vector after setup is done
42FA    Visi8x  ; Set object's visibility
```

TITLE	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

```
42FD    Vipcmd  ; Wait until NOT(VIP ! SHAKE) then data and cmd
4300    V54ect  ; Basic MOVE or DRAW
4303    SYMNLN  ; SYMTAB for n-char variable
4306    SYMLBL  ; SYMTAB for SUB names
4309    Intext  ; R14 entry into Intsrv
430C    CUTBAK  ; safely cuts the stack back (re. BASIC SUBs)
430F    Pnproc  ;  Entry to Page - Hardcopy  Processor
4312    Kbntcp  ;  Entry to Kbint so Rombacks can intercept
                                              Keyboard Interrupts
```

Tektronix
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE	ABSTRACT NUMBER
TECO	TEKniques Vol. 6 No. 4 T1 Program 2
ORIGINAL DATE: November, 1982 / REVISION DATE:	EQUIPMENT AND OPTIONS REQUIRED: 64K "A" Series 4050
AUTHOR: Ed Post, Tektronix, Inc. Wilsonville, OR	PERIPHERALS

ABSTRACT

Files: 1 ASCII Program

Statements: 667

For those of you Real Programmers that think TECO is the only REAL text editor, there now exists one that runs on the 4052A and 4054A. This TECO implements most of the commands available in common versions running on DEC time sharing systems, and is capable of editing files on tape, disk or extended memory. Numeric and string "Q" registers are available, and Q registers can be run as Macros. The combination of TECO, the 4050A assembler, and the extended memory option makes creation and testing of assembly language programs convenient.

TITLE	ABSTRACT NUMBER
TECO	TEKniques Vol. 6 No. 4 T1 Program 2

INTRODUCTION

TECO is a popular character-oriented editor, available on most DEC computers, with variants running on almost every other computer system. TECO is at once powerful and cryptic; compact and dangerous. It includes facilities for string searches, iteration, macros, mathematical expressions.

TECO commands are all similar in structure. They are almost exclusively single letters (usually the first character of the operation being performed) optionally preceded by one or two numeric arguments, optionally followed by a string argument terminated with an escape character. Each numeric argument can be defined as an arbitrarily long sequence of numbers or variables separated by operators. The string argument can be arbitrarily long and extend over many lines of text.

Command strings are lists of commands, terminated by a double escape. The commands in a command string are executed one at a time after the terminating double escape is typed. If any errors are detected during execution, execution of the command string is terminated and an error message generated. If the user wishes to abort execution of a command string after the double escape has been typed, he or she need only strike any key on the keyboard and the operation will terminate.

PRINTING CHARACTERS

Characters are displayed on the screen in such a way that they are always uniquely recogniziable.

* All printing characters (space through tilde) are displayed as is.

* Control characters (with certain exceptions) are displayed as up-arrow followed by the corresponding uppercase alpha character.

* Tab (control-I) moves to the next 4050 tab stop (every 20 characters).

* Carriage return causes a newline action.

* Escape is printed as less-than overstrike greater-than: ≪

* Rubout echos as H overstrike I overstrike X: ▩ -- a blot to rub out the previous character.

* High-parity characters appear as their corresponding low-parity characters, in boldface.

TITLE	ABSTRACT NUMBER
TECO	TEKniques Vol. 6 No. 4 T1 Program 2

COMMAND ENTRY

TECO prompts for command entry with a '*'. The user can then type a command string in, terminating with a double escape. At any time during entry of a command, the following options apply:

1) The user can type rubout. The last character in the command string is deleted. The rubout character is echoed as a blot that covers the character being deleted.

2) The user can type '↑G ' (control-G space). The current command string is echoed back to the screen. This is useful after several rubouts to see exactly what you have left.

3) The user can type '↑G↑G' (two control-G's). The current command string is deleted, and a new '*' prompt printed.

4) As the very first two characters after the prompt, the user can type *q$$ (splat, Q-register, double escape). The previous command string will be stored in the named Q-register for later execution (more on macros later). In particular, if the last command string terminated with an error, or was deleted with a ↑G↑G, it might be reasonable to store this entry as a macro, modify it, then re-execute.

TITLE	ABSTRACT NUMBER
TECO	TEKniques Vol. 6 No. 4 T1 Program 2

COMMANDS

↑A Print all characters between this and the next control-A in the command string to the screen.

. The current location of the cursor, between Ø and Z. Ø means the cursor is in front of the first character in the buffer.

:Gq Print the contents of Q-register q to the screen.

< Repeat the commands between this and the following '>' forever.

n< Repeat the commands between this and the following '>' for n iterations. (n>0).

n= Print the decimal value of n on the screen.

> Close a loop started by '<'.

C Character: Move one character forwards.

nC Move n characters forwards (backwards if n is negative).

D Delete the next character.

nD Delete the next n characters (previous n characters if negative).

ERfile⎋ Edit Read: Concatenate the contents of the named file to the end of the buffer. If 'file' is a purely numeric value, the file is taken to be the tape file by that number, otherwise, it's the file by that name on the currently active 4907 or E-disk unit.

EWfile⎋ Edit Write: Write the contents of the buffer out to the named file.

ECfile⎋ Edit Create: Create (on disk) the file specified, ASCII mode.

EX EXit: Terminate execution. (Execution can be continued after this command or after a 4050 error code using function key #2.)

FRtext⎋ Replace: The n characters immediately in front of the cursor are deleted, replaced by the text argument--n being the length of the most recent search argument. The cursor is placed immediately after the last character.

FStext1⎋text2⎋ The next instance of text1 in the buffer is searched for. If found, it is replaced by text2.

nFStext1⎋text2⎋ The n'th successive (previous, if negative) instance of text1 is replaced by text2.

TITLE	ABSTRACT NUMBER
TECO	TEKniques Vol. 6 No. 4 T1 Program 2

Gq The contents of Q-register q are copied into the buffer at the current cursor position. The cursor is left immediately after the last character inserted.

H Equivalent to 'Ø,Z'. For commands that take two numeric arguments delimiting a part of the buffer, H delimits the entire buffer: HK deletes the entire buffer; HT types the entire buffer.

Itext* Insert: the text argument is inserted into the buffer. The cursor is left immediately after the last character inserted.

J Jump: Move to the beginning of the buffer. (Short for ØJ).

nJ Jump: Move to after the n'th character in the buffer. ZJ moves to the end of the buffer, because Z holds the current size of the buffer.

K Kill: Delete characters up to and including the next return character in the buffer. This deletes a line of text.

nK Delete the next n lines of text. A negative value causes the n lines preceding the cursor to be deleted.

ØK Delete characters between the cursor and the closest return character to the left of the cursor. ØKK deletes the entire current line, no matter where the cursor is on the line.

m,nK Delete all characters after the n'th, and before the m+1'th.

HK Delete everything. Same as Ø,ZK.

L Line: Move the cursor immediately after the next return character. The effect is to move to the start of the next line.

nL Move to the start of the n'th line following the one on which the cursor is currently positioned. A negative n causes the cursor to be positioned on the n'th line previous to the current one.

ØL Move to the start of the current line.

Mq Macro: Execute Q-register q as a command string. The current command string is interrupted, and Q-register m is executed as a command string. When complete, processing returns to the previous command string. Macros may be nested to an arbitrary (but finite) degree.

Qq Returns the value of numeric Q-register q.

R Reverse: Move one character to the left. Same as -C. All combinations of arguments work similarly.

TITLE	ABSTRACT NUMBER
TECO	TEKniques Vol. 6 No. 4 T1 Program 2

Stext⌘ Search: The text argument is searched for in the buffer starting at the current cursor position. If found, the cursor is positioned immediately after the text. If not, an error; the cursor is positioned at the beginning of the buffer.

S⌘ with no argument, the text searched for is the same as the last time a search was performed.

nStext⌘ Search for the n'th occurrence of the text in the buffer. If n is negative, the search proceeds in reverse.

T Type: print the contents of the buffer through the next return character. This types the current line.

nT Type the next n lines.

ØT Type the beginning of the current line, up to the cursor position.

m,nT Type the characters between the m'th and the n'th.

nUq Store the numeric argument in numeric Q-register q.

V View: Type the current line in context. Roughly equivalent to ØT↑A↓↑AT, where ↑A↓↑A types an arrow at the cursor position.

nV Same, but more like 1-nT↑A↓↑Ant.

Xq Store the next line in Q-register q. The current contents of the Q-register are lost. The buffer is not changed, nor is the cursor position.

nXq Store the next n lines in Q-register q.

ØXq Corresponds to ØT.

m,nXq Corresponds to m,nT.

Z Returns the current size of the buffer, in characters.

TITLE	ABSTRACT NUMBER
TECO	TEKniques Vol. 6 No. 4 T1 Program 2

EXAMPLES

```
JSthis⎋ØLIStart⎋V⎋⎋
```

Jump to the start of the buffer;
Find the first instance of 'this' in the buffer;
Insert "Start" at the beginning of the line;
View the changed line.

```
HKERmyfile⎋J<SChapter⎋LI----------
⎋>⎋⎋
```

Read myfile into an empty buffer;
Find all occurrences of "Chapter" in the file;
Add a line of dashes under each of these lines.
(Process terminates with an error when no more lines are found.)

```
ERfile1⎋ERfile2⎋EW1O⎋⎋
```

Concatenate the contents of disk files "file1" and "file2";
Write the results to tape file 10.

```
ZJERjunk⎋.,ZT.,ZK⎋⎋
```

Read the file "junk" into the buffer;
Print it;
Then delete it.

TITLE	ABSTRACT NUMBER
TECO	TEKniques Vol. 6 No. 4 T1 Program 2

Tektronix
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE	ABSTRACT NUMBER
Printed Circuit Board Layout	TEKniques Vol. 6 No. 4 T1 Program 3
ORIGINAL DATE: March, 1982 / REVISION DATE:	EQUIPMENT AND OPTIONS REQUIRED: 4054, Opt. 30, 64K
AUTHOR: Franz Reiter, Rohde & Schwarz-Tektronix Austria	PERIPHERALS: 4663 Plotter

ABSTRACT

Files: Binary Program File

1 Data File (example)

Statements: 900

Design the printed circuit board on the screen. Define symbols of up to 50 solder-tags for quick duplication. All lines and solder-tags are in the standard grid (DIP size, 2.54mm) or in half grid. Design two layers at once, one drawn in dashed lines on the screen or with the second pen on the plotter. Redraw just Layer 1 or Layer 2 or just all solder-tags or the whole drawing. Full zooming of any board section, no restriction of board size. Delete any line, solder-tag or symbol for correction. The standard line is 0.3mm broad, any other value selectable. The standard solder-tag has a diameter of 1.5mm, any other value selectable.

Plot the drawing on foil in any desired scaling. Generate your copper board now with an ultraviolet-sensative lacquer or make a printing-foil of it. Store the drawing on tape. Retrieve the drawing, delete and add as you need, make a new plot and store again.

TITLE	ABSTRACT NUMBER
Printed Circuit Board Layout	TEKniques Vol. 6 No. 4 T1 Program 3

PRELIMINARY OPERATING INSTRUCTIONS

The Printed Circuit Board Layout program is provided in Binary Format.

If the program file is transferred to another tape and is to be autoloaded, a short ASCII program will have to occupy file one which calls the program from file 2.

The program will store the drawing away on tape and recall it. However, the storage file will have to be pre-marked, the size depending on your drawing.

Statements 30002 and 30006 contain an error check so the autoload and program file assumed to occupy files 1 and 2 aren't overwritten.

This program occupies file 4 of the TEKniques program tape. A data file is on file 5.

TITLE	ABSTRACT NUMBER
Printed Circuit Board Layout	TEKniques Vol. 6 No. 4 T1 Program 3

DATA STORAGE

Four line types are possible: horizontal, vertical, 45 degree and -45 degree. They are stored in Arrays V1 (horizontal), V2 (vertical), V3 ($+45^{o}$), and V4(-45^{o}).

For any of these lines three numbers are stored. X_1 and Y_1 are the coordinates of the starting point and X_2 and Y_2 are the coordinates of the ending point. In Array V2 are stored: X_1, Y_1, Y_2. In Array V1, V3, and V4 are stored: X_1, Y_1 and X_2. The corresponding counters are S1, S2, S3 and S2. (They contain the actual number of lines, so the array contains three times as many numbers.)

If the first X-value is negative, the vector belongs to the second layer. If the first X-value is not standard broadness, the broadness is added as broadness times 0.001.

S5 is the number of symbols in the drawing. In Array V5 are the starting coordinates of these symbols. In B$ (it has S5-characters) is the type of the symbol descripted. An "L" is a single solder-tag, a "G" is a solder-tag with non-standard size. A number from 1 to 9 represents the corresponding symbol. A$ contains the symbol-description. 12 characters for each pair of relative coordinates of a solder tag, 50 solder tags for one symbol, nine symbols. This makes 600 characters for one symbol.

If the X-coordinate in V5 is negative, the corresponding symbol is to be turned 90 degrees.

The diameter of the tag (if nonstandard!) is added to the X-value as diameter times 0.001.

S6 is the max Y-coordinate (for WIN).
T$ is the Textstring. T1 and T2 are the text starting coordinates.

TITLE	ABSTRACT NUMBER
Printed Circuit Board Layout	TEKniques Vol. 6 No. 4 T1 Program 3

Description of PCB--Board-layout Program
of IDD-Ost of RST/Vienna

Hardware: 4054 with Option 24 and 30 and 4663 or 4662

Load Program with Autoload-Key.
Wait for the following Main-Menu.

1) Create a New Drawing
2) Plot the previously created Drawing
3) Store the Drawing on tape
4) Read a Drawing from tape
5) Purge the Drawing (not the symbols!)
6) End

Press the corresponding Key (for ex.the "1", "CRT")

1) Create a New Drawing

At first you see written declaration and System asks:
Maximum board size in millimeters? Enter for example 100, CR.
Now the working grid comes up. The distance between two points is one "E", this is 2.54mm. System puts any line or soldertag into a grid point or exactly between it. Actual the grid distance is half of one "e" - This is 1.27 mm (with "H" depressed you can draw lines on a quarter grid - 0,63mm distance).

The crosshair curser is enabled, the following commands are possible:

M.... Move. Fixes a new starting position for a line. The last point always is the default point. The starting point is the center of the crossed axis.

D.... Draw. Draws a line (or a Symbol for fixes the position of a solder-tag for the symbol definition). The line will be from the starting point (center of crossed axis) to the center of the crosshair cursor.

L.... Draws a solder-tag at the center of the crosshair cursor. (hole-diameter 0,8mm)

A.... Alternate Layer (Moves and Draws with dashed lines, on the plotter with the second Pen). Back to the standard Layer with A again.

H.... You now can enter any quarter-point in the grid (0,63mm) Return to normal grid with "H" again.

Z....Zoom.Enter the Koordinates of the lower left point (X-Value, CR,Y-Value,CR) and the upper Y-Koordinate of the desired View.

TITLE	ABSTRACT NUMBER
Printed Circuit Board Layout	TEKniques Vol. 6 No. 4 T1 Program 3

B.... Enter the desired Broadness of the following lines, Return. This value even will be the diameter of solder-tags drawn with "G".

G.... Draws a solder-tag with hole-diameter "B".

S.... Starts the definition of a symbol.(A symbol consists of max.50 solder-tags of 0,8mm) System asks: Symbol-Nr.? Enter a number from 1 to 9 (You can have nine different symbols at one time), Return.
With the thumbweels you now can move a single solder-tag. Any time you press "D" you store one more solder-tag at the current position in the symbol definition.
This first drawing of the symbol-at the definition - becomes not stored in the picture! Return ends the symbol definition.

1,2,3 - or any number up to 9 calls the corresponding symbol.
"+" will turn the symbol a 90 degrees,
"-" will it back. "D" stores this symbol.

I.... ("Irrtum") removes the last entered line or symbol.

E.... "Erase" removes the line or symbol at the current crosshair-center. (Symbols can only in their origin-signed by a cross- be erased).

N.... "New" redraws the whole drawing.

T.... "Text"-starting at the current crosshair-center.
(Only one field with Max.72 characters.Change with "T" again- Text - Return - "N".)
Stop entering Text with return

F.... Function select.
1... Redraw whole drawing.
2... Redraw first layer only.
3... Redraw second layer only.
4... Redraw solder-tags only.

Return brings you back to the main menu.

TITLE	ABSTRACT NUMBER
Printed Circuit Board Layout	TEKniques Vol. 6 No. 4 T1 Program 3

2) Plot the picture

System asks: Proportion - transformation 1 to?
For 1 Unit = 2,54mm enter 1 (no transformation)
Return
System asks: Coodinates of lower left point?
(necessary for correct positioning when enlarging the drawing; system puts the drawing to the lower left by the value of the entered Coordinates)
Enter X-Value, Return, Y-Value, Return.
(for instance O, CR, O, CR-no displacement)

System asks: Both Layers or.....
Enter 1 for both Layers. (The second Layer will be drawn with the second pen).

3 and 4

Enter the file number.

5) Purge the Drawing

Purges the drawing from memory, but not symbols which have been created.

Tektronix®
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE	ABSTRACT NUMBER
Pipe Construction	TEKniques Vol. 6 No. 4 T1 Program 4
ORIGINAL DATE: May, 1978 / REVISION DATE:	EQUIPMENT AND OPTIONS REQUIRED: 16K
AUTHOR	PERIPHERALS: Optional - 4662/3 Plotter

ABSTRACT

Files: 1 ASCII Program

Statements: 290

The pipe construction program illustrates the use of graphics in a mechanical engineering or construction area.

The program calculates and makes a scale drawing of two pipes connected together at any angle between 0 to 45^{0}, giving inside and outside dimensions, wall thickness and bell diameter. Required inputs for this program are pipe diameter, pipe lengths and bend angle. Measurements returned are in standard form, i.e., wall thickness is the standard size for the pipe diameter.

This program will draw on the screen or the plotter.

This particular program is used in a pipe prefabrication plant and gives the designer a graphic representation of the final product as well as supplying him with all measurements in standard pipe sizes.

TITLE	ABSTRACT NUMBER
Pipe Construction	TEKniques Vol. 6 No. 4 T1 Program 4

OPERATING INSTRUCTIONS

Load the program through the tape directory, or FIND 6 and OLD and RUN.

The program will ask for the following values:

1. Pipe Diameter. Enter a diameter from 18 to 54 inches.

   Standard pipes come in 3" increments; the program will round your input to the closest multiple of 3.

2. Left Leg Length. Enter a value for the left section of pipe. (I.e., 24,6 for a length of 24',6".

3. Right Leg Length. Same as above.

4. Enter bend angle - 0^{o} to 45^{o}.

Once a pipe has been constructed, use the User-Definable Keys to draw the picture on the CRT, the plotter, or to change the dimensions.

OUTPUT

1. Pipe Diameter. Diameter returned is value entered rounded to the closest increment of 3".

2. Bend Angle. Same as entered.

3. Scale Factor. For scale drawing use.

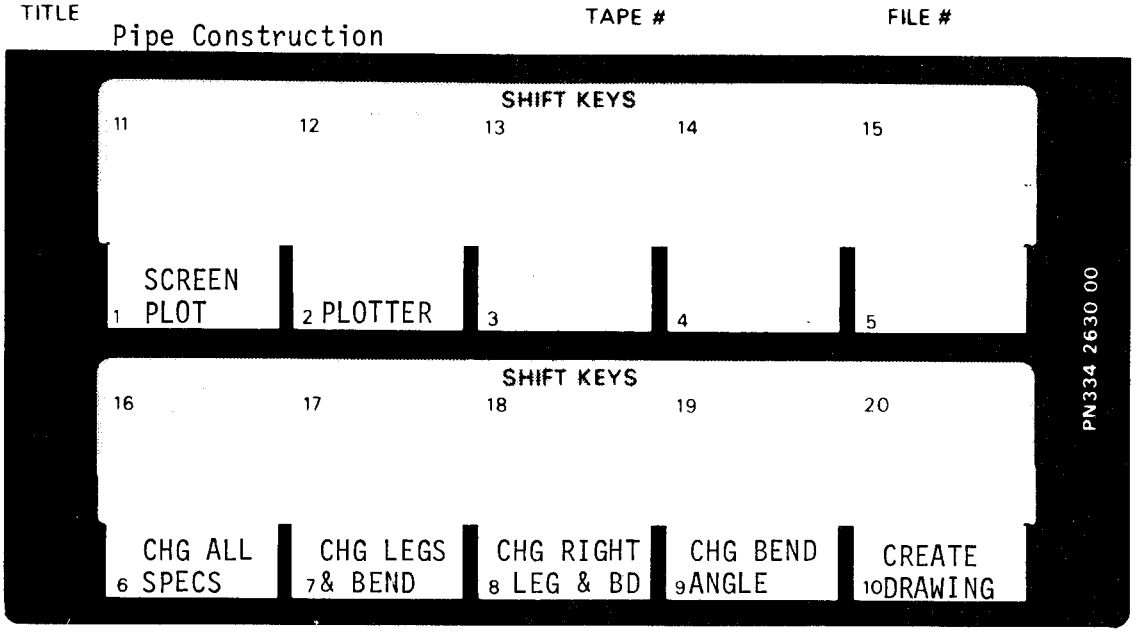
4. Left and Right Leg Length. Same as entered

5. Bell Diameter. Calculated Bell Diameter for the size of pipe.

6. Wall diameter. Standard wll diameter for pipe size.

7. Final product measurements.

TITLE	ABSTRACT NUMBER
Pipe Construction	TEKniques Vol. 6 No. 4 T1 Program 4



1. Draws existing pipe design on screen.
2. Draws existing pipe design on plotter.
6. Prompts for new dimensions.
7. Prompts for leg and bend dimensions.
8. Prompts for right leg and bend dimensions.
9. Prompts for bend angle.
10. Begin program anew.

PIPE DIA= 21'' BEND ANGLE= 33 DEGREES SCALE= 2.62ft. to 1 in.
LEFT LEG LENGTH= 20' 6.00''
RIGHT LEG LENGTH= 10' 4.00''
BELL DIA= 2' 6.50'' WALL THICK= 2.38''

A= 20' 2.89'' B= 20' 9.11''
C= 10' 0.89'' D= 10' 7.11''
E= 30' 2.39''

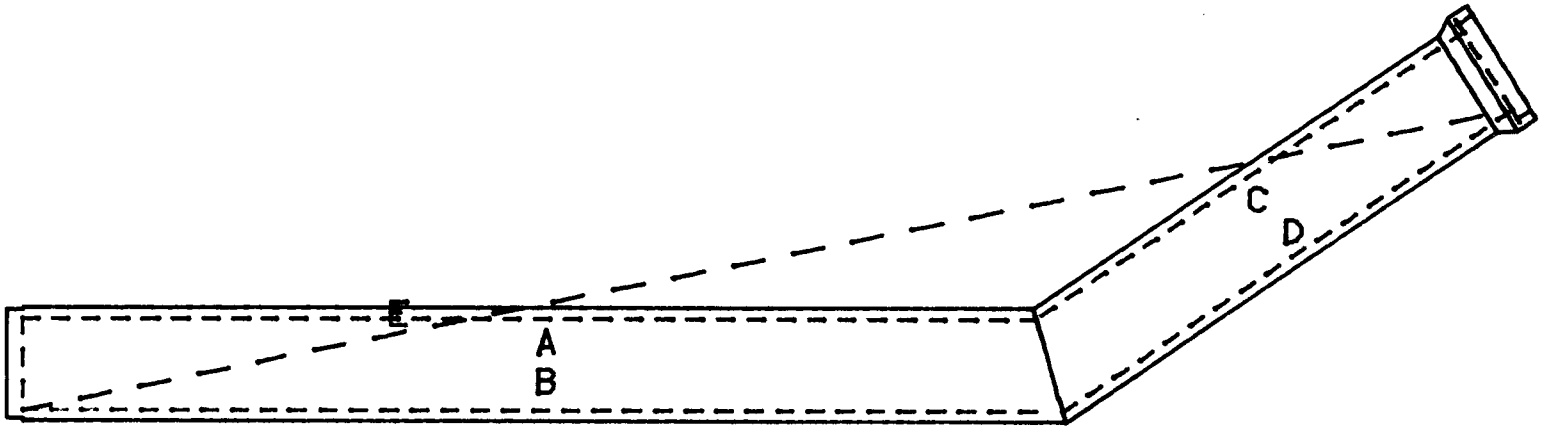


Fig. 1

TITLE
Pipe Construction

ABSTRACT NUMBER
TEKniques Vol. 6 No. 4 T1
Program 4

PAGE NUMBER 108

7/80

Tektronix
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE		ABSTRACT NUMBER
Moment-of-Intertia Curve Plots		TEKniques Vol. 6 No. 4 T1 Program 5
ORIGINAL DATE March, 1981	REVISION DATE	EQUIPMENT AND OPTIONS REQUIRED 32K
AUTHOR Khiem Ho George Tzitzikalakis	Columbia University Biomechanics New York, NY	PERIPHERALS Optional-4662 Plotter

ABSTRACT

Files: 1

Statements:

This program plots Moment-of-Inertia curves for different cross-sections, given the necessary parameters. It also prints out discrete values for different points. The plot can draw at most two independent variables of the cross-section at a time.

TITLE

PLOTTING MOMENT-OFINERTIA CURVES

ABSTRACT NO:

TEKniques Vol. 6 No. 4 T1

Program 5

I) DESCRITION

This program plots Moment-of-Inertia curves for 6 types of cross-sections. The formula used are as follows:

1. Square

$$I = \frac{X^4}{12}$$

where X is the length of one side of the square.

2. Rectangle

$$I = \frac{BH^3}{12}$$

where B is the width of the rectangular cross-section;
H is the height of "

3. Solid Circle

$$I = \frac{(PI)R^4}{4}$$

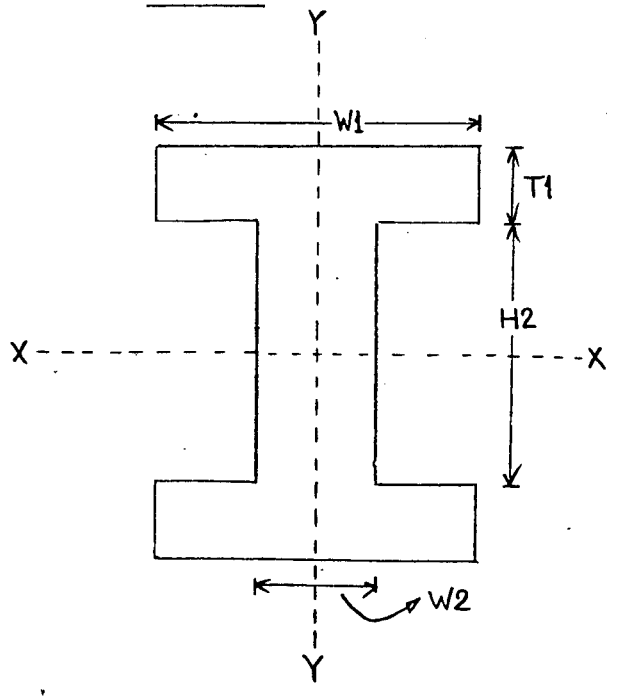
where PI is 3.14159
R is the radius of the solid circular cross-section

4. Hollow Circle

$$I = \frac{(PI)(Ro^4 - Ri^4)}{4}$$

where Ro is the outer radius,
Ri is the inner radius.

5. I-Beam



T1: Flange's Thickness
W1: Flange's Width
W2: Web's Thickness
H2: Web's Width

TITLE	ABSTRACT NO:
PLOTTING MOMENT-OF-INERTIA CURVES	TEKniques Vol. 6 No. 4 T1 Program 5

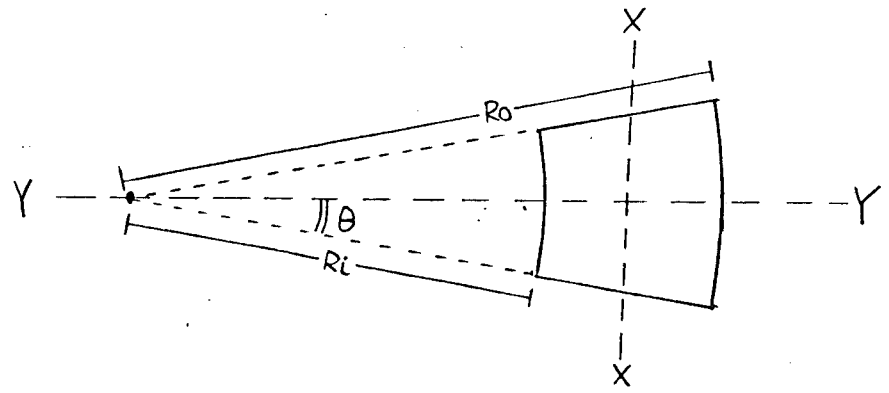
i) About the YY-axis

$$I = 2\frac{(T1 \times W1^3)}{12} + \frac{H2 \times W2^3}{12}$$

ii) About the XX-axis

$$I = 2\left[\frac{W1 \times T1^3}{12} + \left[T1 \times W1\left(\frac{T1}{2} + \frac{H2}{2}\right)^2\right]\right] + \frac{W2 \times H2^3}{12}$$

6. Hollow Circular Sector



i) About the YY-axis

$$I = \frac{(Ro^4 - Ri^4)}{4}(\theta - \sin\theta\cos\theta)$$

11) About the XX-axis

$$I = \frac{(Ro^4 - Ri^4)}{4}(\theta + \sin\theta\cos\theta) - \frac{4(Ro^3 - Ri^3)^2(\sin\theta)^2}{9\theta(Ro^2 - Ri^2)}$$

Ref: Formulas for stress and strain,
By Raymond J. Roark.
Fourth edition
McGraw-Hill Book Company

TITLE

PLOTTING MOMENT-OF-INERTIA CURVES

ABSTRACT NO:

TEKniques Vol. 6 No. 4 T1

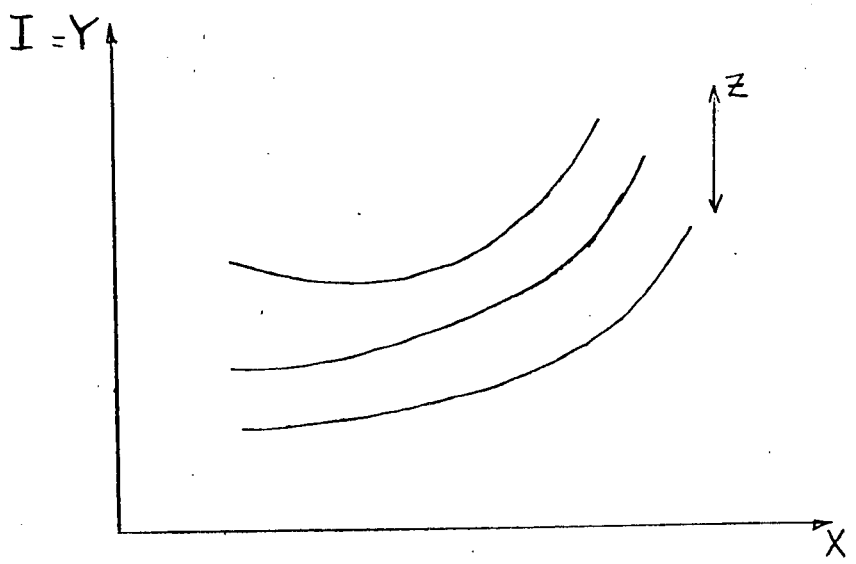
Program 5

For plots with two independent variables, the user has a choice to select which variables will be on the X-axis, and which will be on the Z-axis (see example *).

For plot with more than two independent variables, the user has a choice to select which variable(s) will be constant(s) for the plot, so that there are only two independent variables at a time.

Discrete values of the plot are given if the user selects to have them. An X-value must be entered, and the program prints out the Moment-of-Inertia values for different Z-values.

Example *



TITLE

PLOTTING MOMENT-OF-INERTIA CURVES

ABSTRACT NO:

TEKniques Vol. 6 No. 4 T1

Program 5

II) METHOD

At the user's selection of the type of cross-section, the program will define the appropriate function for the Moment of Inertia (called I). From the typical parameters given by the user, the corresponding I is calculated.

To plot the Moment of Inertia, the program will get from the user the selection of first and second independent variables, and the appropriate constants. It then defines the appropriate function, generates points and plot the function either on the screen or on the plotter. Next, again at the user's selection, the program will use that function to calculate I's from a point on the X-axis supplied by the user.

TITLE
PLOTTING MOMENT-OF-INERTIA CURVES

ABSTRACT NO:

TEKniques Vol. 6 No. 4 T1

Program 5

III) FLOW-CHART

100 - 500	Initialization and Main program to ask for which kind of cross-section.
600 - 900	Subroutine for Square.
1000 - 1410	Rectangle.
1500 - 1800	Solid Circle.
1900 - 2300	Hollow Circle.
2400 - 4220	I-Beam.
4300 - 4940	Hollow Circular Sector.
4950 - 5080	Ask for plot.
5100 - 5210	Y-axis data.
5220 - 5310	Ask for type of output.
5320 - 5450	Plot with one independent variable.
5460 - 5690	Draw axes.
5700 - 5790	Draw plot with 2 independent variables.
5800 - 5930	Calculate discrete I's.
5940 - 6040	Output preference for discrete I's
6050 - 6170	Print out discrete I's
6180 - 6490	Ask for discrete calculation.
6500 - 6780	Data for two-variable plot.
6800 - 6880	Type of axis of cross-section.

IV) OPERATING INSTRUCTION

1. LOAD program from tape.
2. Execute ("RUN") it.
3. Select type of cross-section.
4. Enter necessary values as asked for program to calculate one typical point of I, so that we can have an idea of the range of I.
5. Enter YES to question "plotting or not?" to plot curves.
6. Enter the appropriate values as asked.
7. Chose output device.
8. Print out.
9. Select whether to calculate discrete values of I.
10. Enter an X-value.
11. Select output device.
12. Print out.
13. Step 10 repeats.

TITLE
PLOTTING MOMENT-OFINERTIA CURVES

ABSTRACT NO:
TEKniques Vol. 6 No. 4 T1
Program 5

V) DATA TAPE STRUCTURE

There is no data file for this program.

VI) INTERNAL DATA STORAGE

Variable	Used to store	Type
X	X-points	Array (50)
Y	Y-points	Array (50)
S	Selection of type of plot	Simple
R1	Minimum of X-axis	"
R2	Maximum of Y-axis	"
FNA	Function for I	function
S1	Type of variation in a plot	simple
S3	I-beam: selection for which are constants	"
S4	I-beam: which kind of variation	"
S5	Selection of axes of cross-section	"
T, T1	Hollow-circular: angle	"
Y1	Minimum Y-axis	"
Y2	Maximum Y-axis	"
O9	Output device	"
M	Discrete points Y's	array
M1	Discrete points Z's	array
Y4	X-value for discrete	simple

TITLE	ABSTRACT NUMBER
Moment of Inertia Curves	TEKniques Vol. 6 No. 4 T1 Program 5

There are 6 types of section in this program:

1. Square
2. Rectangle
3. Solid circle
4. Hollow Circle
5. I-Beam
6. Hollow Circular Sector
   (eg. A-O Plate section)

Please select one and enter the appropriate number. S= 2
This is to find the MOMENT OF INERTIA of a Rectangle.
**NOTE: the principal central Axis 1 is parallel to WIDTH

Please enter the WIDTH and the HEIGHT respectively.
5
7

MOMENT OF INERTIA is 142.916666667

Do you want to generate plots for different values?
Please enter Y for Yes, or N for No. y

TITLE	ABSTRACT NUMBER
Moment of Inertia Curves	TEKniques Vol. 6 No. 4 T1 Program 5

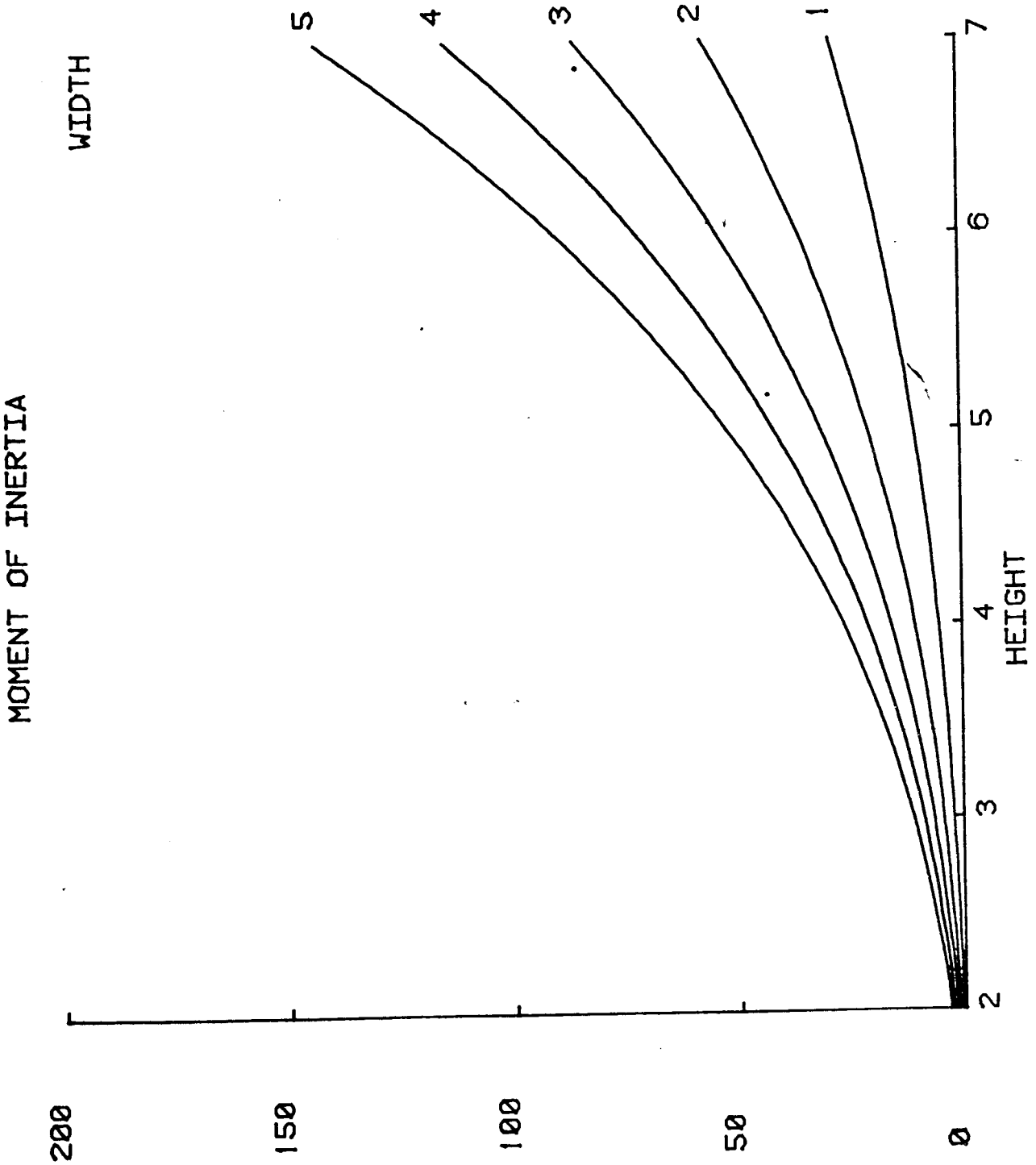
```
This Moment of Inertia depends on two variables:
WIDTH and HEIGHT.
Please enter 1 if you want to see variation of WIDTH over
              a range of HEIGHT.
             2 if you want to see variation of HEIGHT
             over a range of WIDTH.
 1
Please enter the range of variation of WIDTH
(MIN and MAX respectively)
 1
 5
Please enter the Interval of each variation. V3= 1
Please enter the range of HEIGHT
(MIN and MAX respectively)


 2
 7

Please enter Y-MIN and Y-MAX respectively
 0
 200
Please enter value for Y-axis interval. Y3= 50
Please enter value for X-axis interval. X3= 1


Do you want SCREEN(=32) or PLOTTER(=1) printout?
 1
```

TITLE	ABSTRACT NUMBER
Moment of Inertia Curves	TEKniques Vol. 6 No. 4 T1 Program 5



TITLE	ABSTRACT NUMBER
Moment of Inertia Curves	TEKniques Vol. 6 No. 4 T1 Program 5

Do you want to try a value of X and get the Moments
of Inertia for different values of the second variable?
Please answer Y or N. Y

Please enter the value of HEIGHT
3

TITLE

Moment of Inertia Curves

ABSTRACT NUMBER

TEKniques Vol. 6 No. 4 T1
Program 5

RESULTS FOR DISCRETE MOMENTS OF INERTIA

HEIGHT	WIDTH	MOMENT OF INERTIA
3.0		
	1.0	2.3
	2.0	4.5
	3.0	6.8
	4.0	9.0
	5.0	11.3

Tektronix
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE	ABSTRACT NUMBER
FFT in 2048 Numbers - IFT in 1024 Complex	TEKniques Vol. 6 No. 4 T1 Program 6 and Program 7
ORIGINAL DATE: February, 1982 / REVISION DATE:	EQUIPMENT AND OPTIONS REQUIRED 56K
AUTHOR: Piere Thore — L.A.G.A.S, avenue du Doyen France	PERIPHERALS 4052R08 FFT ROM

ABSTRACT

Files: 2 ASCII Program

Statements: 38 and 77

The FFT program performs FFT of an array of 2048 real numbers. It provides the result in the same G array, under the same format as after a ROM pack computing. The program turns with ROM pack n°2 "FFT".

The computation last 30 s'.

The IFT program performs the reverse operation with the same I/O formats.

TITLE	ABSTRACT NUMBER
FFT in 2048 Numbers - IFT in 1024 Complex	TEKniques Vol. 6 No. 4 T1 Program 6 and Program 7

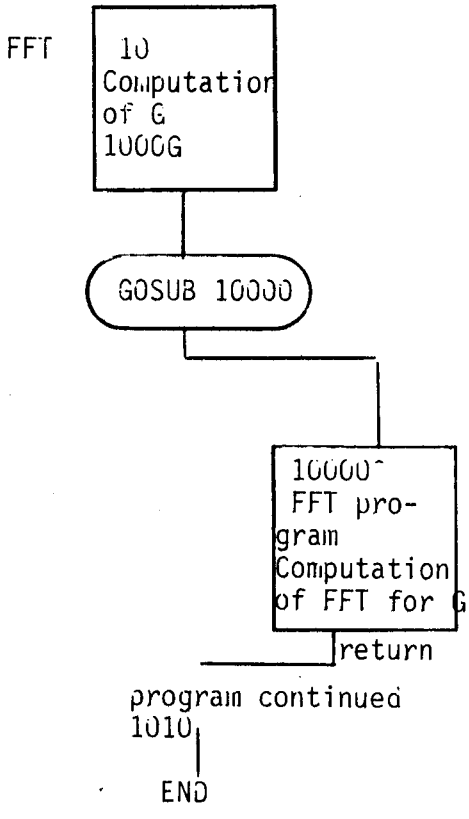
METHODS

An FFT on 2048 numbers may be reduced to 2 FFT on 1024 numbers.

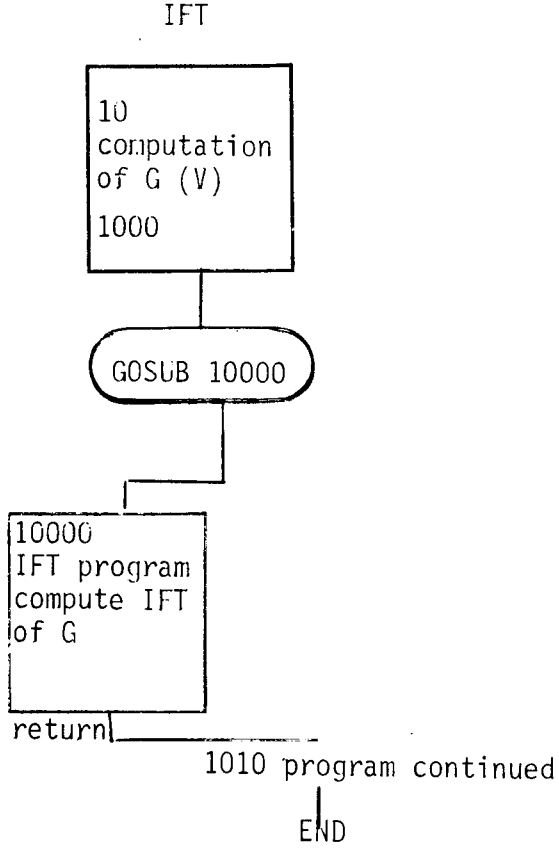
The result of an IFT may be achieved through an FFT.

These programs should be considered subprograms of a master program that must compute the G array then lead to the computation of FFT through a GOSUB.

Sample program



IFT



Note : after the performing of FFT, G is similar to FFT throught ROM PACK N°2
G(1) = value of the nul frequency
(2) = of NICKWIST
(3) = real part of frequency n° 1
(4) = " imaginary " "

G array should appear as if it had been computed through the ROM PACK

idem

refer to the FFT ROM PACK instruction manual

Tektronix®
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE	ABSTRACT NUMBER
NEWTON INTEGRATION AND PLOT	TEKniques Vol. 6 No. 4 T1 Program 8
ORIGINAL DATE / REVISION DATE	EQUIPMENT AND OPTIONS REQUIRED: 4051 (16K Memory)
AUTHOR: J. E. Jobaris, U. S. Postal Service (415) 876-9364 Engineering Systems, WE422, San Bruno, CA 94099	PERIPHERALS: Optional: Hardcopy Unit (4631)

ABSTRACT

This program uses the Newton (sometimes called the Newton-Rhapson) method to solve an equation of two variables for which no direct or easy solution is available. The Newton method iterates the following equation:

$$X_{n+1} = X_n - F(X_n)/F'(X_{n+1}) \qquad n = 0, 1, \ldots$$

until the term $F(X_n)/F'(X_{n+1})$ has no effect on the last decimal place of accuracy as specified by the user.

An optional plot can be produced within the range of the independent variable as input by the user. The plotting subroutines were taken taken from a 4051 Datagraphing program. The plot of the function can be produced with or without the root of the equation at the request of the user. The plot with the solution uses dashed lines whose intersection represents a root. Because the Newton method calculates only one root, the intersection of the horizontal dashed line and the funtion plot indicates other solution points.

The equation, its first derivative, and the value of the equation at which a solution is required are entered at specified lines in the program.

TITLE	ABSTRACT NUMBER
NEWTON INTEGRATION AND PLOT	TEKniques Vol. 6 No. 4 T1 Program 8

1. DESCRIPTION

The Newton method has been used by the author in various application programs. These programs are used for engineering and economics problems which require a numerical technique to solve complex mathematical equations. The Newton method is exceptionally fast despite its relative simplicity. It is well-suited as a subroutine for multiple calls if one variable is changed such as in 'what if' simulations.

Its one shortcoming is that only one root is found, the root found being dependent on the initial starting value of the iteration. For this reason, a 4051 Datagraphing program was modified and the Newton method incorporated to enable us to visualize the plot of an equation and choose the starting iterative value in the area of practical value to us. The second example illustrates this point.

In this program (Newton Integration and Plot), the starting value is taken as the midpoint of the range of the variable to be found. The range (minimum and maximum values) is required by the modified 4051 Datagraphing program.

2. DATA STRUCTURE

Data files are not used in this program.

TITLE	ABSTRACT NUMBER
NEWTON INTEGRATION AND PLOT	TEKniques Vol. 6 No. 4 T1 Program 8

3. INTERNAL DATA STORAGE

VARIABLE	USED TO STORE	TYPE
Y	Function of X (dependent variable)	Simple
X	Independent variable	Simple
Y1	Function of X (Y) to be solved minus the value of Y for which X is sought	Simple
Y2	First derivative of Y (or Y1)	Simple
Q0	Value of Y (Function of X) for which X is sought	Simple
A	Value of X for Y + Q0	Simple
Y3	Trigger for plot only, root only, or both	Simple
A1	Minimum value of X	Simple
A2	Maximum value of X	Simple
D	Decimal places of accuracy for root	Simple
A$	Used to display answer	String
L	Length of A$	Simple
R5	Print location for axis labels	Simple
R6	Print location for axis labels	Simple

The remaining variables are from the function plot portion of a 4051 Datagraphing program. These variables are used for the graphic display (plot) portion of the program and are not understood by this author.

TITLE	ABSTRACT NUMBER
NEWTON INTEGRATION AND PLOT	TEKniques Vol. 6 No. 4 T1 Program 8

5. OPERATING INSTRUCTIONS

A. An overlay is not used

B. To load this ASCII program, type

FIND N (where N = tape file number)

OLD

C. Program Execution:

Change line 100 to the equation (function of X) which is to be solved. This equation must be expressed in terms of Y. For instance, if we wanted to solve the equation Y = X - 3/2 COS(X) for X at Y = -1, line 100 should read:

```
100  Y = X - 3/2 * COS(X)
```

Change line 110 to the Y value for which X is sought. This value of Y must be expressed as Q0, and for the example above, line 110 would read:

```
110 Q0 = -1
```

Change line 120 to the equation which represents the first derivative of Y as given in line 100. The derivative must be expressed as Y2. In this example, line 120 would read:

```
120 Y2 = 1 + 3/2 * SIN(X)
```

After these changes are made, type RUN.

Answer the interactive inquiry about the range of the X values; i.e., minimum and maximum X values. Even if a plot is not required, the input values of X are necessary for a starting iteration value for the Newton method.

TITLE	ABSTRACT NUMBER
NEWTON INTEGRATION AND PLOT	TEKniques Vol. 6 No. 4 T1 Program 8

Answer 1, 2, or 3 in response to the question for a plot only, for a root only, or for both, respectively.

If a root is not necessary, the screen will clear and the function will be plotted between the range of the minimum and maximum values of X. If a root is necessary (2 or 3 in response to question in H), the program will ask for the number of decimal places of accuracy required and calculate a root. After the root is calculated, the root will be printed on the screen or displayed with the plot depending on the answer of item H.

The solution of this equation at Y = -1 for three decimal places of accuracy is 0.388. Attachment 1 shows the operator entered responses to the program questions and the resulting solution and plot.

Another example is to find the roots of the equation $Y = X^3 + X^2 - 5X - 7$ for Y = -5. In this example, line 100 reads:

```
100  Y = X^3 + X^2 - 5X - 7
```

Line 110 would read:

```
110  Q0 = -5
```

And line 120, the first derivative of Y, is entered as

```
120  Y2 = 3 * X^2 + 2 * X - 5
```

If the range of X where chosen as -5 to +5, a solution of -0.382 would be found. However, the plot would indicate two other solutions, see attachment 2. To find the exact value of the other roots, the range of X would have to be changed. For an input range of 0 to 4, the solution is 2, see attachment 2.

TITLE

NEWTON INTEGRATION AND PLOT

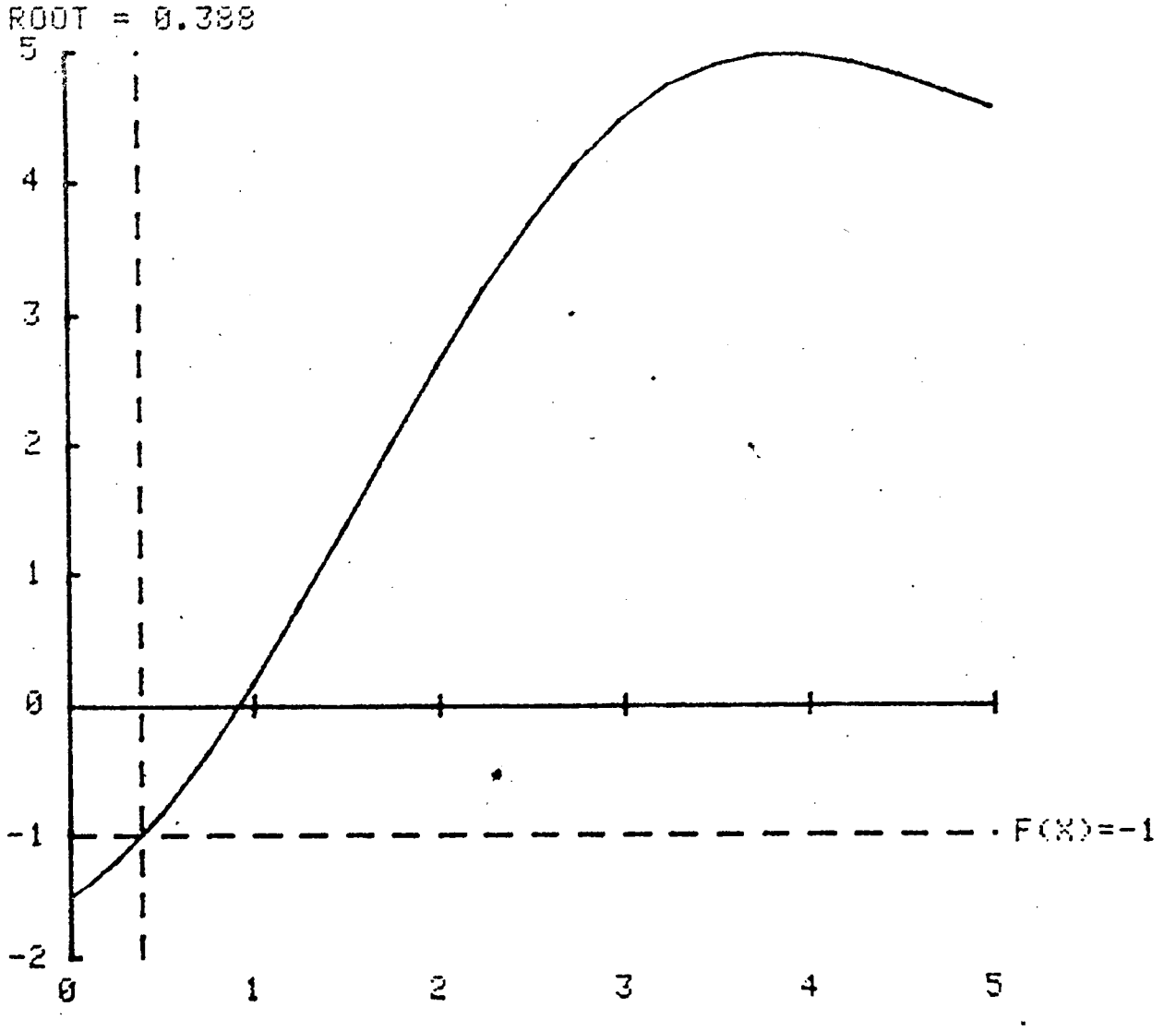
ABSTRACT NUMBER

TEKniques Vol. 6 No. 4 T1
Program 8

Visual observation of the plot would give approximate values for all the solution points within the original range. Exact answers for the solutions can be determined by running the program again and selecting the range such that the value of X as approximately determined by visual observation is tightly bracketed by the range of the input values. In other words, only the root which is within the range of the minimum and maximum values would be chosen. The range should also be chosen such that the approximate value of X determined by visual observation is at the approximate midpoint of the input range.

TITLE	ABSTRACT NUMBER
Newton Integration and Plot	TEKniques Vol. 6 No. 4 T1 Program 8

```
PLEASE ENTER X DATA
                    MINIMUM 0
                    MAXIMUM 5
ENTER 1 FOR PLOT ONLY, 2 FOR ROOT ONLY, 3 FOR BOTH 3
ENTER NO. OF DECIMAL PLACES OF ACCURACY DESIRED 3
```



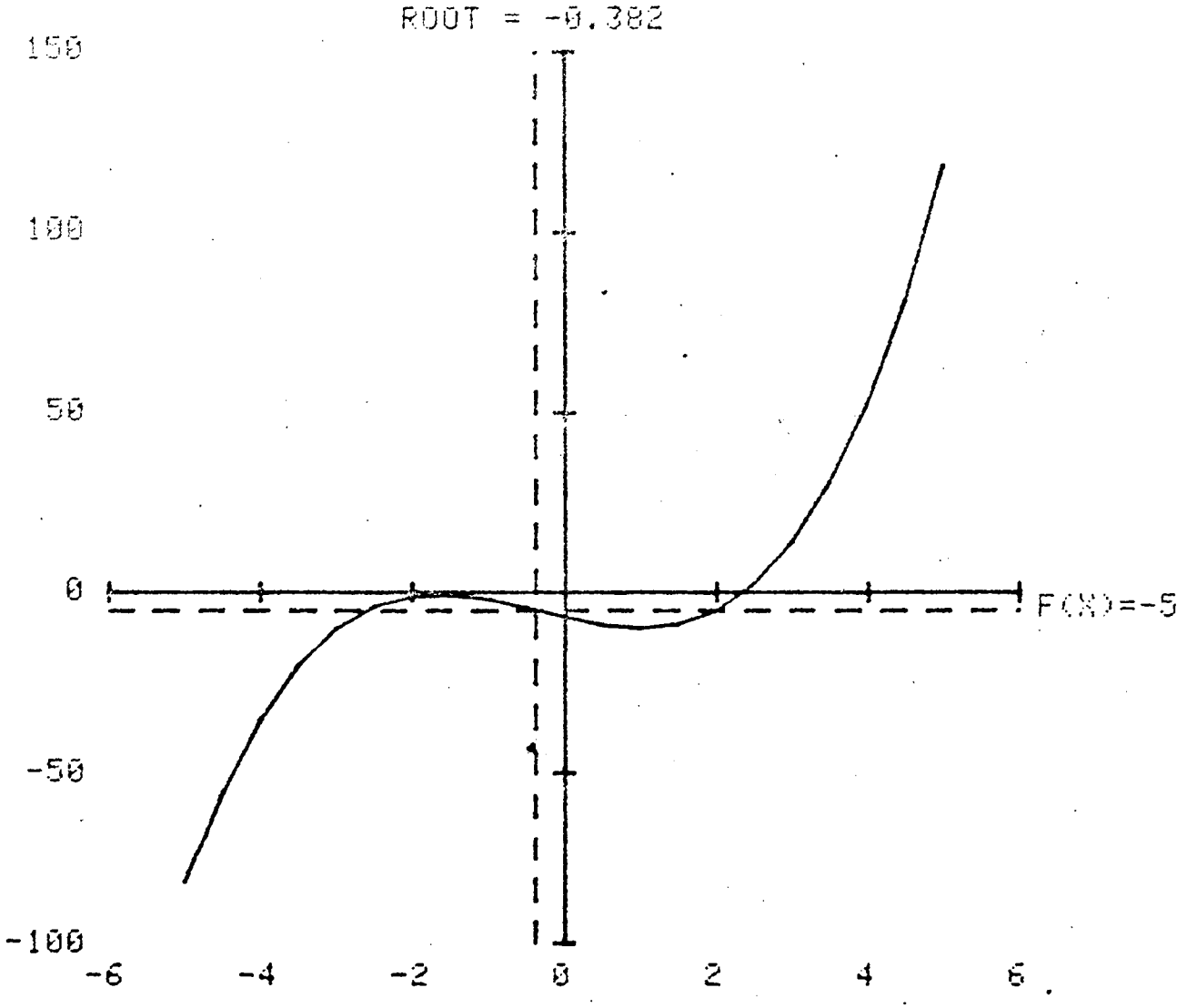
TITLE

Newton Integration and Plot

ABSTRACT NUMBER

TEKniques Vol. 6 No. 4 T1
Program 8

```
PLEASE ENTER X DATA
                    MINIMUM -5
                    MAXIMUM 5
ENTER 1 FOR PLOT ONLY, 2 FOR ROOT ONLY, 3 FOR BOTH 3
ENTER NO. OF DECIMAL PLACES OF ACCURACY DESIRED 3
```



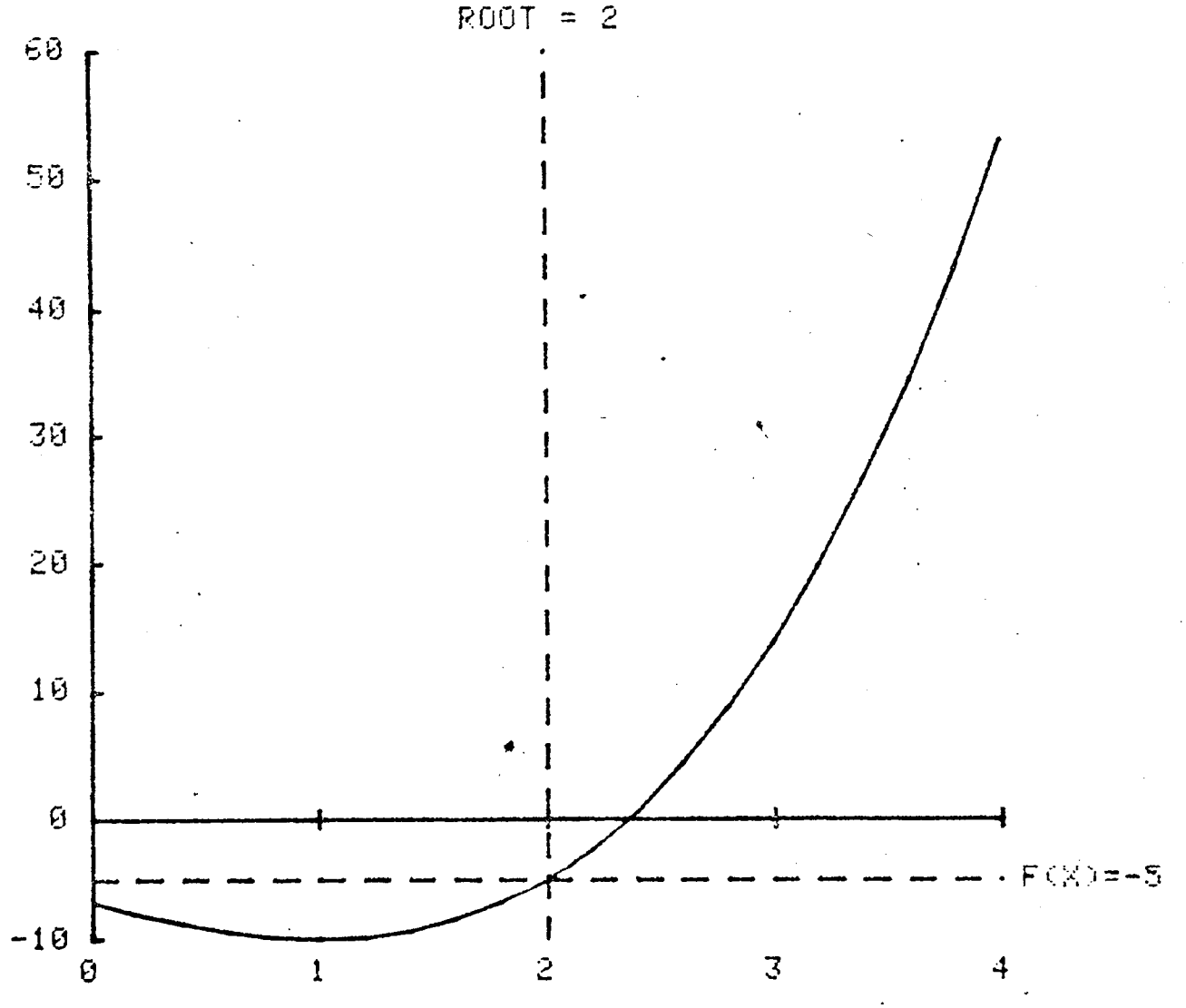
TITLE

Newton Integration and Plot

ABSTRACT NUMBER

TEKniques Vol. 6 No. 4 T1
Program 8

```
PLEASE ENTER X DATA
                    MINIMUM 0
                    MAXIMUM 4
ENTER 1 FOR PLOT ONLY, 2 FOR ROOT ONLY, 3 FOR BOTH 3
ENTER NO. OF DECIMAL PLACES OF ACCURACY DESIRED 3
```

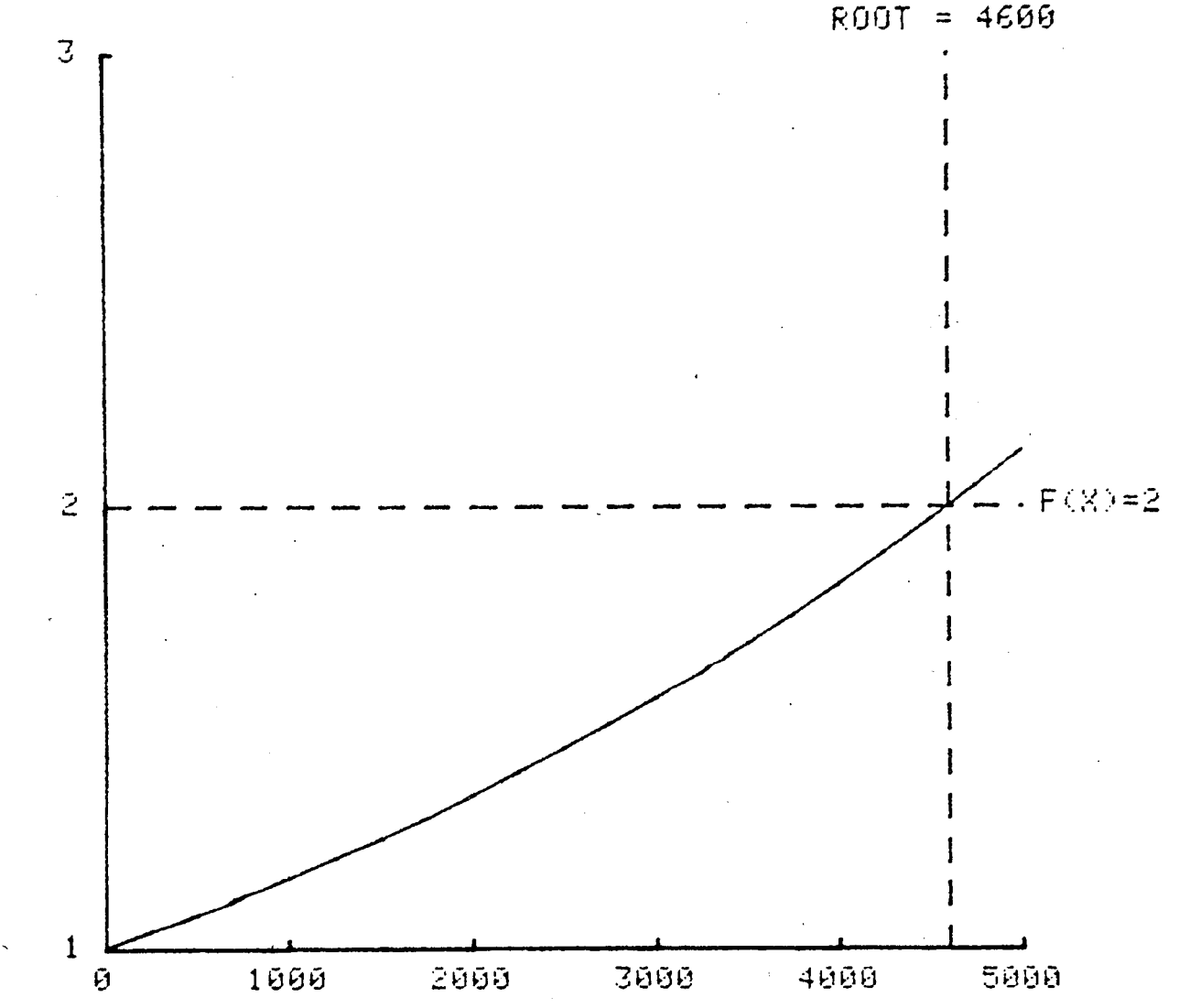


TITLE	ABSTRACT NUMBER
Newton Integration and Plot	TEKniques Vol. 6 No. 4 T1 Program 8

The following run is the solution of the number of days (365 days per year) it would take a deposit to double for a 5.5% interest rate compounded daily.

```
.IST 1,150
 GO TO 5000
100 Y=(1+0.055/365)↑X
105 REM     Y1 & Y2 ARE FOR NEWTON INTEGRATION SOLUTION TO DETERMINE
106 REM     THE VALUE OF X FOR A GIVEN VALUE OF 00
107 REM     00 IS THE SOLUTION PT. I.E. THE F(X) VALUE
110 00=2
120 Y1=Y-00
130 Y2=Y*LOG(1+0.055/365)
```

```
PLEASE ENTER X DATA
                    MINIMUM 0
                    MAXIMUM 5000
ENTER 1 FOR PLOT ONLY, 2 FOR ROOT ONLY, 3 FOR BOTH 3
ENTER NO. OF DECIMAL PLACES OF ACCURACY DESIRED 0
```



Tektronix®
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE	ABSTRACT NUMBER
Bauer-Reinsch inversion of a positive definite symmetric matrix.	TEKniques Vol. 6 No. 4 T1 Program 9
ORIGINAL DATE / REVISION DATE	EQUIPMENT AND OPTIONS REQUIRED: 4051 with 16K
AUTHOR: Roger P. Denlinger, U.S.G.S. MS 404, Denv. Fed. Ctr, Lakewood, CO 80225	PERIPHERALS

ABSTRACT

A modification of the Gauss Jordan algorithm solves the linear problem Ax=b, where A is a positive definite symmetric matrix. The working storage needed is only the matrix itself, as the inverse is overwritten on the original matrix. Starting with Ax=b after k rows have been reduced in Gauss-Jordan fashion;

$$\begin{pmatrix} I_k & X \\ 0 & Z \end{pmatrix} x = \begin{pmatrix} W & 0 \\ Y & I_{n-k} \end{pmatrix} b \qquad \text{with} \quad X=-Y^T.$$

A sequential reordering of the rows and columns of A is done so that the arithmetic is always performed with k=1. This renumeration relabels (j+1) as j for j=1,2,...,(n-1) and relabels 1 as n. Letting $p=a_{11}$ for the k-1 step, the Gauss Jordan process is for the k^{th} step;

$$a_{nn} = 1/p$$

$$a_{i-1,n} = a_{i1}/p$$

$$a_{n,j-1} = -a_{1j}/p$$

$$a_{i-1,j-1} = a_{ij} - a_{i1}a_{1j}/p$$

for i,j=2,...,n

A working vector is used to prevent a_{1j}/p for the k-1 step from being over-written by $a_{i-1,j-1}$. The matrix of coefficients is in the form, because of the renumbering, $\begin{pmatrix} Z & Y \\ X & W \end{pmatrix}$ where $Z=Z^T$ and $Y=-X^T$. After n renumerations the array elements are in the correct order.

TITLE	ABSTRACT NUMBER
Bauer Reinsch inversion	TEKniques Vol. 6 No. 4 T1 Program 9

2. DATA STRUCTURE

Program set up for keyboard input here, but is designed to be used either as a subroutine or for input from tape or disc with very little modification.

3. INTERNAL DATA STORAGE

a) Table of variables;

Variable	Used to store...	Type
n	number of matrix rows or columns	simple
A	matrix of coefficients, and inverse	array(n*(n-1)/2)
X	temporary storage vector	array(n*(n-1)/2)
q	index	simple
m	index	simple
t	index	simple
s	temporary storage of a_{11}	simple

b) A_{ij} = a(j*(j-1)/2 + i), same for X.

4. METHODS

The method used in the algorithm exactly follows the modification of the Gauss Jordan elimination for n steps which is listed in the abstract.

5. OPERATING INSTRUCTIONS

Program presented here is interactive, with keyboard input of the matrix of coefficients. (it may be modified for use as a subroutine). The two inputs needed are the number of columns in the array A_{ij} and the coefficients of A_{ij} entered column by column.

A_{ij} is entered such that (j*(j-1)/2 + i) is the order of entry. The matrix A_{ij} is assumed to be square and symmetric, and is overwritten by its own inverse during program execution.

TITLE	ABSTRACT NUMBER
Bauer Reinsch inversion	TEKniques Vol. 6 No. 4 T1 Program 9

6. REFERENCES

Wilkinson, J.H., and Reinsch, C. (eds.) 1971;
Linear Algebra, Handbook for Automatic Computation, vol. 2.
Berlin: Springer Verlag. pg. 45.

TITLE	ABSTRACT NUMBER
Bauer Reinsch Inversion	TEKniques Vol. 6 No. 4 T1 Program 9

EXAMPLES:

1) matrix A:

1	-1	-1	-1	-1
	2	0	0	0
		3	1	1
			4	2
				5

inverse of A:

86	43	22	12	8
	22	11	6	4
		6	3	2
			2	1
				1

inverse of inverse retrieves A by typing RUN 320.
if matrix A is computationally indefinite, routine
will fail and print warning message.

Tektronix®
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE	ABSTRACT NUMBER
4050-4010 Utilities	TEKniques Vol. 6 No. 4 T1 Program 10
ORIGINAL DATE: August, 1980 / REVISION DATE:	EQUIPMENT AND OPTIONS REQUIRED: 16K
AUTHOR: Ed Sawicki, Tektronix, Inc. Wilsonville, OR	PERIPHERALS: 4010 Graphics System

ABSTRACT

Files: 2 ASCII Program

Statements: 300

This is a package of routines for driving 4010 family terminals from a 4050 Graphics System. The routines are written in 4050 BASIC and are organized as callable subroutines.

Most of the routines perform conversion from 4010 style data to decimal data (or vice-versa) which can be manipulated easily by a user-written mainline program. Each routine is well documented with a banner preceding the actual code. The banner lists subroutine entry and exit requirements as well as temporary (scratch) variables used.

Since all variables in 4050 BASIC are global, it is the user's responsibility to ensure that no conflict between mainline and subroutine variables occurs.

TITLE	ABSTRACT NUMBER
4050-4010 Utilities	TEKniques Vol. 6 No. 4 T1 Program 10

```
90 REM Sample Interactive graphics program

START   INIT
        PAGE
        PRINT @40:"[L"
        CALL "WAIT",1
LOOP    PRINT @40:"[Z";
        GOSUB  GETGIN
        IF E9=1 THEN  LOOP
        GOSUB  GINDEC10
        IF K$="m" OR K$="M" THEN DARK
        IF K$="d" OR K$="D" THEN LIGHT
        GO TO  LOOP
LIGHT   PRINT @40:"]";
        GOSUB  DECGRAF10
        PRINT @40:X$;
        E$=X$
        GO TO  LOOP
DARK    GOSUB  DECGRAF10
        PRINT @40:"]";E$;X$;
        E$=X$
        GO TO  LOOP
```

TITLE

4050-4010 Utilities

ABSTRACT NO:

TEKniques Vol. 6 No. 4 T1

Program 10

```
10000 REM         4010 FAMILY UTILITIES in 4050 BASIC
10010 REM             Ed Sawicki - Tektronix Inc.


10020 REM     +-------------------------------------------------+
10030 REM     |                  TABDEC10                       |
10040 REM     |  Convert tablet data to decimal ordinates       |
10060 REM     |  Enter with 4 bytes of tablet data in I$        |
10070 REM     |           ( HIY,LOY,HIX,LOX )                   |
10080 REM     |  Exit with decimal ordinates in X & Y           |
10090 REM     |                ( 1024 X 1024 )                  |
10100 REM     |  Uses X$ as a temporary variable                |
10110 REM     +-------------------------------------------------+

10120 REM  Compute Y ordinate
10130 X$=SEG(I$,1,1)
10140 Y=(ASC(X$)-32)*32
10150 X$=SEG(I$,2,1)
10160 Y=ASC(X$)-32+Y

10170 REM  Compute X ordinate
10180 X$=SEG(I$,3,1)
10190 X=(ASC(X$)-32)*32
10200 X$=SEG(I$,4,1)
10210 X=ASC(X$)-32+X

10220 RETURN
```

TITLE

4050-4010 Utilities

ABSTRACT NO:

TEKniques Vol. 6 No. 4 T1

Program 10

```
10230 REM     +-----------------------------------------------+
10240 REM     |                  TABDEC12                     |
10250 REM     |   Convert tablet data to decimal ordinates    |
10270 REM     |   Enter with 5 bytes of tablet data in I$     |
10280 REM     |          ( HIY,XLOY,LOY,HIX,LOX )             |
10290 REM     |   Exit with decimal ordinates in X & Y        |
10300 REM     |                ( 4096 X 4096 )                |
10310 REM     |   Uses X$ and T for temporary variables       |
10320 REM     +-----------------------------------------------+

10330 X$=SEG(I$,1,1)
10340 Y=(ASC(X$)-32)*128
10350 X$=SEG(I$,3,1)
10360 Y=(ASC(X$)-32)*4+Y

10370 X$=SEG(I$,4,1)
10380 X=(ASC(X$)-32)*128
10390 X$=SEG(I$,5,1)
10400 X=(ASC(X$)-32)*4+X

10410 X$=SEG(I$,2,1)
10420 T=ASC(X$)
10430 IF T/2=INT(T/2) THEN 10460
10440 X=X+1
10450 T=T-1

10460 IF T/4=INT(T/4) THEN 10490
10470 X=X+2
10480 T=T-2

10490 IF T/8=INT(T/8) THEN 10520
10500 Y=Y+1
10510 T=T-4

10520 IF T=32 THEN 10540
10530 Y=Y+2

10540 RETURN
```

TITLE

4050-4010 Utilities

ABSTRACT NO:

TEKniques Vol. 6 No. 4 T1

Program 10

```
10550 REM     +-------------------------------------------------+
10560 REM     |                  DECGRAF12                      |
10570 REM     |  Convert X,Y to 401X graph mode data - 12 Bit   |
10590 REM     |  Enter with decimal ordinates in X & Y          |
10600 REM     |                ( 4096 X 4096 )                  |
10610 REM     |  Exit with graph mode data in X$                |
10620 REM     |          ( HIY,XLOY,LOY,HIX,LOX )               |
10630 REM     |  Uses Y$ as a temporary variable                |
10640 REM     +-------------------------------------------------+

10650 X$=CHR(INT(Y/128)+32)
10660 Y$=CHR((Y-INT(Y/4)*4)*4+(X-INT(X/4)*4)+96)
10670 X$=X$&Y$

10680 Y$=CHR(INT(Y/4-INT(Y/128)*32)+96)
10690 X$=X$&Y$

10700 Y$=CHR(INT(X/128)+32)
10710 X$=X$&Y$

10720 Y$=CHR(INT(X/4-INT(X/128)*32)+64)
10730 X$=X$&Y$
10740 RETURN


10750 REM     +-------------------------------------------------+
10760 REM     |                  DECGRAF10                      |
10770 REM     |   Convert X,Y to 401X graph mode data - 10 Bit  |
10790 REM     |  Enter with decimal ordinates in X & Y          |
10800 REM     |                ( 1024 X 1024 )                  |
10810 REM     |  Exit with graph mode data in X$                |
10820 REM     |          ( HIY,LOY,HIX,LOX )                    |
10830 REM     |  Uses Y$ as a temporary variable                |
10840 REM     +-------------------------------------------------+

10850 X$=CHR(INT(Y/32)+32)
10860 Y$=CHR(INT(Y-INT(Y/32)*32)+96)
10870 X$=X$&Y$

10880 Y$=CHR(INT(X/32)+32)
10890 X$=X$&Y$

10900 Y$=CHR(INT(X-INT(X/32)*32)+64)
10910 X$=X$&Y$
10920 RETURN
```

TITLE

4050-4010 Utilities

ABSTRACT NO:

TEKniques Vol. 6 No. 4 T1

Program 10

```
10930 REM     +-----------------------------------------------+
10940 REM     |                  OPTGRAF10                    |
10950 REM     |   Convert X,Y to graph mode data - 10 Bit     |
10960 REM     |          with SHORTENED ADDRESSING            |
10970 REM     |  Enter with decimal ordinates in X & Y        |
10980 REM     |               ( 1024 X 1024 )                 |
10990 REM     |  Exit with graph mode data in X$              |
11000 REM     |    Z$ is `old' address                        |
11010 REM     |  Uses Y$,V$,W$ and S9 as temporary variables  |
11020 REM     +-----------------------------------------------+
11030 X$=""
11040 S9=0

11050 REM       --- Do High Y ---
11060 W$=CHR(INT(Y/32)+32)
11070 V$=SEG(Z$,1,1)
11080 IF W$=V$ THEN 11110
11090 X$=W$

11100 REM       --- Now Low Y ---
11110 Y$=CHR(INT(Y-INT(Y/32)*32)+96)
11120 V$=SEG(Z$,2,1)
11130 W$=W$&Y$
11140 IF Y$=V$ THEN 11180
11150 S9=1
11160 X$=X$&Y$

11170 REM       --- Then High X ---
11180 Y$=CHR(INT(X/32)+32)
11190 W$=W$&Y$
11200 V$=SEG(Z$,3,1)
11210 IF Y$=V$ THEN 11270
11220 X$=X$&Y$
11230 IF S9=1 THEN 11270
11240 Y$=CHR(INT(Y-INT(Y/32)*32)+96)
11250 X$=REP(Y$,LEN(X$),0)

11260 REM       --- Finally Low X ---
11270 Y$=CHR(INT(X-INT(X/32)*32)+64)
11280 W$=W$&Y$
11290 X$=X$&Y$
11300 IF W$<>Z$ THEN 11320
11310 X$=""
11320 Z$=W$
11330 RETURN
```

TITLE

4050-4010 Utilities

ABSTRACT NO:

TEKniques Vol. 6 No. 4 T1

Program 10

```
11340 REM     +-------------------------------------------------+
11350 REM     |                   OPTGRAF12                     |
11360 REM     |   Convert X,Y to graph mode data - 12 Bit       |
11370 REM     |          with SHORTENED ADDRESSING              |
11380 REM     |  Enter with decimal ordinates in X & Y          |
11390 REM     |              ( 4096 X 4096 )                    |
11400 REM     |  Exit with graph mode data in X$                |
11410 REM     |    Z$ is `old' address                          |
11420 REM     |  Uses Y$,V$,W$ and S9 as temporary variables    |
11430 REM     +-------------------------------------------------+

11440 X$=""
11450 S8=0
11460 S9=0

11470 REM       --- Do High Y ---
11480 W$=CHR(INT(Y/128)+32)
11490 V$=SEG(Z$,1,1)
11500 IF W$=V$ THEN 11530
11510 X$=W$

11520 REM       --- And XLOY ---
11530 Y$=CHR((Y-INT(Y/4)*4)*4+(X-INT(X/4)*4)+96)
11540 W$=W$&Y$
11550 V$=SEG(Z$,2,1)
11560 IF Y$=V$ THEN 11600
11570 S8=1
11580 X$=X$&Y$

11590 REM       --- Now Low Y ---
11600 Y$=CHR(INT(Y/4-INT(Y/128)*32)+96)
11610 W$=W$&Y$
11620 V$=SEG(Z$,3,1)
11630 IF Y$=V$ AND S8=0 THEN 11700
11640 S9=1
11650 X$=X$&Y$
11660 IF S8=1 THEN 11700
11670 Y$=CHR((Y-INT(Y/4)*4)*4+(X-INT(X/4)*4)+96)
11680 X$=REP(Y$,LEN(X$),0)
```

(continued next page)

TITLE

4050-4010 Utilities

ABSTRACT NO:

TEKniques Vol. 6 No. 4 T1

Program 10

(OPTGRAF12 continued)

```
11690 REM       --- Then High X ---
11700 Y$=CHR(INT(X/128)+32)
11710 W$=W$&Y$
11720 V$=SEG(Z$,4,1)
11730 IF Y$=V$ THEN 11790
11740 X$=X$&Y$
11750 IF S9=1 THEN 11790
11760 Y$=CHR(INT(Y/4-INT(Y/128)*32)+96)
11770 X$=REP(Y$,LEN(X$),0)

11780 REM       --- Finally Low X ---
11790 Y$=CHR(INT(X/4-INT(X/128)*32)+64)
11800 W$=W$&Y$
11810 X$=X$&Y$
11820 IF W$<>Z$ THEN 11840
11830 X$=""
11840 Z$=W$
11850 RETURN
```

TITLE

4050-4010 Utilities

ABSTRACT NO:

TEKniques Vol. 6 No. 4 T1

Program 10

```
11860 REM     +-------------------------------------------------+
11870 REM     |                  GINGRAF10                      |
11880 REM     |       Convert GIN data to GRAPH data  - 10 Bit  |
11890 REM     |                                                 |
11900 REM     |     Enter with GIN data in I$                   |
11910 REM     |     Exit with graph data in Y$                  |
11920 REM     |        ( HIY,LOY,HIX,LOX )                      |
11930 REM     |     Uses X$ as a temporary variable             |
11940 REM     +-------------------------------------------------+

11950 X$=SEG(I$,4,1)
11960 X$=CHR(ASC(X$)+64)
11970 Y$=REP(X$,2,1)
11980 X$=SEG(I$,2,1)
11990 X$=CHR(ASC(X$)+32)
12000 Y$=REP(X$,4,1)
12010 X$=SEG(I$,1,1)
12020 Y$=REP(X$,3,1)
12030 X$=SEG(I$,3,1)
12040 Y$=REP(X$,1,1)

12050 RETURN


12060 REM     +-------------------------------------------------+
12070 REM     |                  GINDEC10                       |
12080 REM     |     Convert GIN data to decimal ordinates       |
12100 REM     |     Enter with GIN data in I$                   |
12110 REM     |     Exit with decimal ordinates in X & Y        |
12120 REM     |          ( 1024 X 1024 )                        |
12130 REM     |                                                 |
12140 REM     +-------------------------------------------------+

12150 REM  Compute Y ordinate
12160 X$=SEG(I$,3,1)
12170 Y=(ASC(X$)-32)*32
12180 X$=SEG(I$,4,1)
12190 Y=ASC(X$)-32+Y

12200 REM  Compute X ordinate
12210 X$=SEG(I$,1,1)
12220 X=(ASC(X$)-32)*32
12230 X$=SEG(I$,2,1)
12240 X=ASC(X$)-32+X

12250 RETURN
```

TITLE

4050-4010 Utilities

ABSTRACT NO:

TEKniques Vol. 6 No. 4 T1

Program 10

```
12260 REM     +-------------------------------------------------+
12270 REM     |                    GETGIN                       |
12280 REM     |  Get GIN data and check if legal                |
12290 REM     |                                                 |
12300 REM     |   Returns with keyboard character in K$ and     |
12310 REM     |    GIN data in I$. E9 is returned as zero if    |
12320 REM     |    data is OK and one if error.                 |
12330 REM     |   Uses I9 and X$ as temporary variables         |
12340 REM     +-------------------------------------------------+

12360 E9=0
12370 INPUT @40:I$
12380 IF LEN(I$)=0 THEN 12430

12390 K$=SEG(I$,1,1)
12400 I$=REP("",1,1)
12410 IF ASC(K$)=4 AND LEN(I$)=5 THEN 12390
12420 IF LEN(I$)=4 THEN 12450

12430 E9=1
12440 GO TO 12500

12450 FOR I9=1 TO 4
12460 X$=SEG(I$,I9,1)
12470 IF ASC(X$)<64 AND ASC(X$)>31 THEN 12490
12480 E9=1
12490 NEXT I9

12500 RETURN
```

Tektronix®
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE	ABSTRACT NUMBER
TELEX	TEKniques Vol. 6 No. 4 T1 Program 11
ORIGINAL DATE: April, 1982 / REVISION DATE:	EQUIPMENT AND OPTIONS REQUIRED: 16K
AUTHOR: Ruud Borstel — Tektronix, Inc. Amstelveen, The Netherlands	PERIPHERALS: Opt. 1 Data Comm I/F

ABSTRACT

RS-232 Paper Tape Punch
Optional-4641/3 Printer

Files: 1 ASCII Program

Statements: 301

The telex programme offers the possibility to create telex code papertapes. As source of data it will use files from a cartridge in the internal tape-unit. These files (containing ASCII data) can be produced by other programmes (e.g. MATRIX package). This makes it possible to put (computer generated) data on a telex without human interference.

The Software

The programme consists of a main programme with 8 subroutines. The main programme will guide the user through the different steps which are necessary for the creation of a telex.

TITLE	ABSTRACT NUMBER
TELEX	TEKniques Vol. 6 No. 4 T1 Program 11

To start-up programme:

1. Load and RUN the programme

2. Specify display device

```
Telex punch program
===================

Telex to be displayed on
                    (S) SCREEN
                    (P) PRINTER
Enter your selection (S or P, default = S) :
```

3. Next a telex form will appear on the screen

```
TELEX
=====

TO: ____________________________     FROM: ____________________________
COUNTRY: _______________________     CC: _____________________________
REF: ___________________________     ____________________________________
SUBJ: __________________________     ____________________________________
                                     ____________________________________
=========================================================================
```

TITLE	ABSTRACT NUMBER
TELEX	TEKniques Vol. 6 No. 4 T1 Program 11

4. Fill in the form

```
TELEX
=====

TO: JOHN_SMITH_____________________     FROM: PAUL_JOHNSON_*_____________
COUNTRY: U.K._____________________      CC: AN_OTHER_*____________________
REF: YOUR_TELEX___________________          ------------------------------
SUBJ: TEST_TRANSMISSION___________          ------------------------------
                                            ------------------------------
================================================================================
MESSAGE :

|DEAR JOHN,                                                          |
|THIS IS A TEST OF THE NEW TELEX TAPE CREATION                       |
|PROGRAM. THE FOLLOWING INFO COMES FROM THE INTERNAL                 |
|TAPE UNIT :                                                         |
|                                                                    |

MESSAGE FROM TAPE FILE # 1
```

A "*" at the end of the "FROM" and the "CC" fields will automatically be replaced by "-teurbv" (the code for Tek Europe). Before specifying the tape file number the user can enter from the keyboard an additional message. Approx. 1000 characters can be entered. The maximum number of characters per line (51) is indicated on the screen. The message can be terminated by entering an empty line.

TITLE	ABSTRACT NUMBER
TELEX	TEKniques Vol. 6 No. 4 T1 Program 11

5. Immediately after specification of the tape file number the programme will start the creation of the papertape. At the same time the telex will be displayed on the selected display device

```
   TO: JOHN SMITH - U.K.
   FROM: PAUL JOHNSON - TEURBV
   CC: AN OTHER - TEURBV
   REF: YOUR TELEX
   SUBJ: TEST TRANSMISSION
   DEAR JOHN,
   THIS IS A TEST OF THE NEW TELEX TAPE CREATION
   PROGRAM. THE FOLLOWING INFO COMES FROM THE INTERNAL
   TAPE UNIT :
   123  456  789  142  547  238  453  765  234  255
   123  456  789  142  547  238  453  765  234  255
   123  456  789  142  547  238  453  765  234  255
   123  456  789  142  547  238  453  765  234  255
   123  456  789  142  547  238  453  765  234  255
   123  456  789  142  547  238  453  765  234  255
   123  456  789  142  547  238  453  765  234  255
   REGARDS,
   PAUL

READY
```

DISPLAY OF TELEX ON THE SCREEN.

TITLE	ABSTRACT NUMBER
TELEX	TEKniques Vol. 6 No. 4 T1 Program 1 1

When the line printer is selected, the telex will be displayed twice. Once in the original form and once in the "telex" form. The difference will be the non transmittables (ASCII characters which do not exist in the telex character set)

```
:   TO: JOHN SMITH - U.K.                                    :
:   FROM: PAUL JOHNSON - TEURBV                              :
:   CC: AN OTHER - TEURBV                                    :
:   REF: YOUR TELEX                                          :
:   SUBJ: TEST TRANSMISSION                                  :
:   DEAR JOHN,                                               :
:   THIS IS A TEST OF THE NEW TELEX TAPE CREATION            :
:   PROGRAM. THE FOLLOWING INFO COMES FROM THE INTERNAL      :
:   TAPE UNIT :                                              :
:   123  456  789  142  547  238  453  765  234  255        :
:   123  456  789  142  547  238  453  765  234  255        :
:   123  456  789  142  547  238  453  765  234  255        :
:   123  456  789  142  547  238  453  765  234  255        :
:   123  456  789  142  547  238  453  765  234  255        :
:   123  456  789  142  547  238  453  765  234  255        :
:   123  456  789  142  547  238  453  765  234  255        :
:   REGARDS,                                                 :
:   PAUL                                                     :
```

a) TELEX as created on 405x.

```
:   TO: JOHN SMITH - U.K.                                    :
:   FROM: PAUL JOHNSON - TEURBV                              :
:   CC: AN OTHER - TEURBV                                    :
:   REF: YOUR TELEX                                          :
:   SUBJ: TEST TRANSMISSION                                  :
:   DEAR JOHN,                                               :
:   THIS IS A TEST OF THE NEW TELEX TAPE CREATION            :
:   PROGRAM. THE FOLLOWING INFO COMES FROM THE INTERNAL      :
:   TAPE UNIT :                                              :
:   123  456  789  142  547  238  453  765  234  255        :
:   123  456  789  142  547  238  453  765  234  255        :
:   123  456  789  142  547  238  453  765  234  255        :
:   123  456  789  142  547  238  453  765  234  255        :
:   123  456  789  142  547  238  453  765  234  255        :
:   123  456  789  142  547  238  453  765  234  255        :
:   123  456  789  142  547  238  453  765  234  255        :
:   REGARDS,                                                 :
:   PAUL                                                     :
```

b) TELEX as sent to TAPE.

DISPLAY OF TELEX ON THE LINE-PRINTER.

TITLE	ABSTRACT NUMBER
TELEX	TEKniques Vol. 6 No. 4 T1 Program 11

Description of the Subroutines

3000 EXTRACT CONTROL CHARACTERS

This routine removes control characters from a string before it is sent to the display device to prevent unwanted cursor movements (e.g. form feed)

4000 ALPHA CURSOR POSITIONING

Form fill-out is possible through this routine which does alpha cursor positioning by sending a sequence of line feeds and spaces (starting from HOME position)

5000 PUNCH LEADER/TRAILER

To begin and end the papertape a sequence of 30 NULL-characters is sent to the papertape puncher (device D1)

6000 CONSTRUCT A TELEX RECORD

This routine takes the ASCII string A$, converts it to telex code (T$) and sends it to device D1 (papertape puncher). Carriage return (ASCII code 13) and line feed (ASCII code 10) are converted and punched also. String T$ is converted back to ASCII and then sent to the display device D2 to inform the user of what is actually put on the telex.

10000 INIT CONVERSION TABLE (ASCII to TELEX)

In this routine the conversion table from ASCII characters to TELEX characters is created. For each ASCII character a TELEX code is defined. A positive number means a TELEX character from the "letter table" and a negative number defines a character in the "figure table".

11000 CONVERSION ROUTINE (ASCII to TELEX)

This routine converts the ASCII string A$ into a TELEX string T$. The ASCII characters are converted one by one into their TELEX equivalents. Each time a TELEX character has a different sign than the previous TELEX character, the routine inserts the appropriate shift character.

TITLE	ABSTRACT NUMBER
TELEX	TEKniques Vol. 6 No. 4 T1 Program 11

12000 INIT CONVERSION TABLE (TELEX to ASCII)

In this routine the conversion table from TELEX characters to ASCII characters is created. For each TELEX character an ASCII code is defined. The table consists of two parts: 1. the "letters table" - the first 32 numbers
2. the "figures table" - the second 32 numbers

When a location in the table is 0, it means that this TELEX code should not be translated into ASCII.

13000 CONVERSION ROUTINE (TELEX to ASCII)

This routine converts the TELEX string T$ into an ASCII string A$. The TELEX characters are converted one by one into their ASCII equivalents. Each time a shift character is found the routine switches to the appropriate part of the conversion table.

THE CONVERSION TABLE.

The Telex Code is a five-bit code that allows for 32 unique bit patterns. One of these patterns means 'letters shift', and another pattern means 'figures shift'. The remaining 30 bit patterns represent one set of characters when they follow a 'letters shift', and another set of characters when they follow a 'figures shift'.

TITLE	ABSTRACT NUMBER
TELEX	TEKniques Vol. 6 No. 4 T1 Program 11

The table below shows which character is assigned to which Telex Code. For convenience, both the Octal and Hexadecimal values of each Telex Code are given.

TELEX CODE Hex	Octal	CHARACTER Letters	Figures
18	30	A	-
13	23	B	?
0E	16	C	:
12	22	D	
10	20	E	3
16	26	F	$
0B	13	G	&
05	05	H	#
0C	14	I	8
1A	32	J	BELL
1E	36	K	(
09	11	L	)
07	07	M	.
06	06	N	,
03	03	O	9
0D	15	P	0

TELEX CODE Hex	Octal	CHARACTER Letters	Figures
1D	35	Q	1
0A	12	R	4
14	24	S	'
01	01	T	5
1C	34	U	7
0F	17	V	=
19	31	W	2
17	27	X	/
15	25	Y	6
11	21	Z	+
04	04	SPACE	
02	02	CARRIAGE RETURN	
08	10	LINE FEED	
00	00	NULL	
1B	33	FIGURES SHIFT	
1F	37	LETTERS SHIFT	

Any character not defined in the above conversion table is punched as a SPACE (Octal 04). Lower case alphabetic characters are converted to upper case.

To illustrate the way the conversion works, the line:

SEND £800 PACKS

will be punched (in octal) as:

37	24	20	06	22	04	04	33	14	15	15	04	37	15	30	16	36	24
ls	S	E	N	D	sp	sp	fs	8	0	0	sp	ls	P	A	C	K	S

ls = LETTERS SHIFT
fs = FIGURES SHIFT
sp = SPACE

Tektronix
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE	ABSTRACT NUMBER
DATA CHART PROGRAM	TEKniques Vol. 6 No. 4 T1 Program 12
ORIGINAL DATE: June 1982 / REVISION DATE:	EQUIPMENT AND OPTIONS REQUIRED: 32K (OPTION 22)
AUTHOR: Hayward Hulick AC 717-782-6745, New Cumberland Army Depot, New Cumberland, PA 17070	PERIPHERALS: 4662 Plotter required, 4907 File Manager optional

ABSTRACT

1. DESCRIPTION.

a. A menu driven program which can plot a graph and corresponding data for up to four lines of up to 15 periods each. The graph can use any of eight plotting styles (4 line, 4 bars, not plotted). The scale, scale interval, and x axis headings are user specified. Chart files may be stored on a mounted disk, marked tape or both. The chart is plotted on the 4662 Plotter and allows the user to change pen colors for each line.

2. DATA TAPE STRUCTURE.

a. Binary, sequential files are used to store charts.

b. Files used are determined by the user (but must be pre-MARKED to at least 1500 bytes if tape is used.) Files are automatically CREATED on the disk.

c. Data is stored sequentially as follows:

1. X-numeric array (4,15)
2. A$-string (290)
3. Y-numeric array (4,6)
4. L$-string (51)
5. P-number
6. F$-string (41)
7. L-number
8. M$-string (106)
9. SØ-number
10. S1-number
11. S2-number

3. INTERNAL DATA STORAGE

Variables used

Variable	Used to Store	Type
X	Data for chart	Array (4,15)
A$	Title String	String (290)
Y	Style, source, length of title,etc.	Array (4,6)
L$	Line heading strings	String (51)

TITLE
DATA CHART PROGRAM

ABSTRACT NUMBER
TEKniques Vol. 6 No. 4 T1
Program 12

Variable	Used to Store	Type
P	Number of periods	Simple
F$	Formal string for output of data	String (41)
L	Number of lines used	Simple
M$	Periods string	String (106)
SØ	Scale minimum	Simple
S1	Scale maximum	Simple
S2	Scale interval	Simple
B$	Misc	String
Z$	Misc	String
Y$	Misc	String
X$	File name or #	String
D	Device for output	Simple
I	Misc	Simple
J	Misc	Simple
K	Misc	Simple
M	Misc	Simple
T	Tape/Disk Indicator	Simple
V	Viewport maximum	Simple
X1	Misc	Simple
X2	Misc	Simple
X3	"	"
Y1	"	"
Y2	"	"
Y3	"	"
B	"	"
B1	"	"
B2	"	"
D1	"	"
D2	"	"
S7	"	"
S8	"	"
T2	"	"
S	"	"
Z	"	"
H	"	"

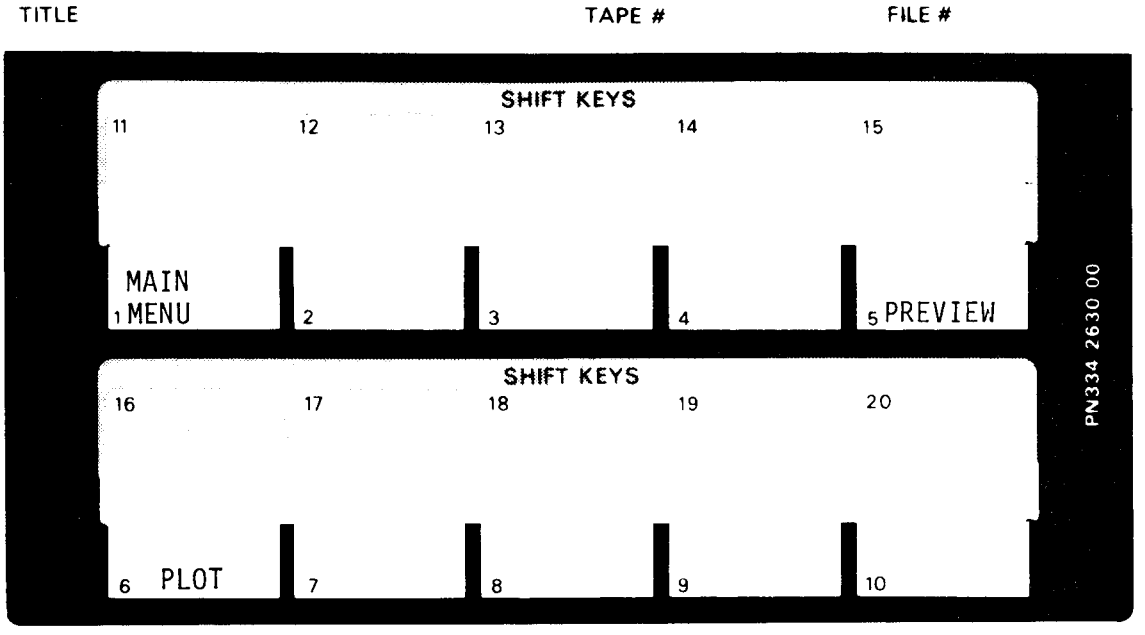
4. OPERATING INSTRUCTIONS.

a. OLD the program into the system and RUN. The main menu will appear. Make the selection to create a chart (5). When the program is RUN all variables are reset to specified values and blanks are inserted into all strings. To create a chart you change these items as desired. Future changes may be made by selecting LOOK/CHANGE (2). Each chart is retained in memory until replaced by a RECALL FILE selection, a RUN, or a CREATE NEW CHART selection.

To plot a chart, place paper with its longest side horizontal. SET the limits of the plotter to the corners of the paper allowing a slight margin.

Blanks may be entered for data by pressing RETURN without an entry.

TITLE	ABSTRACT NUMBER
Data Chart	TEKniquesVol. 6 No. 4 T1 Program 12



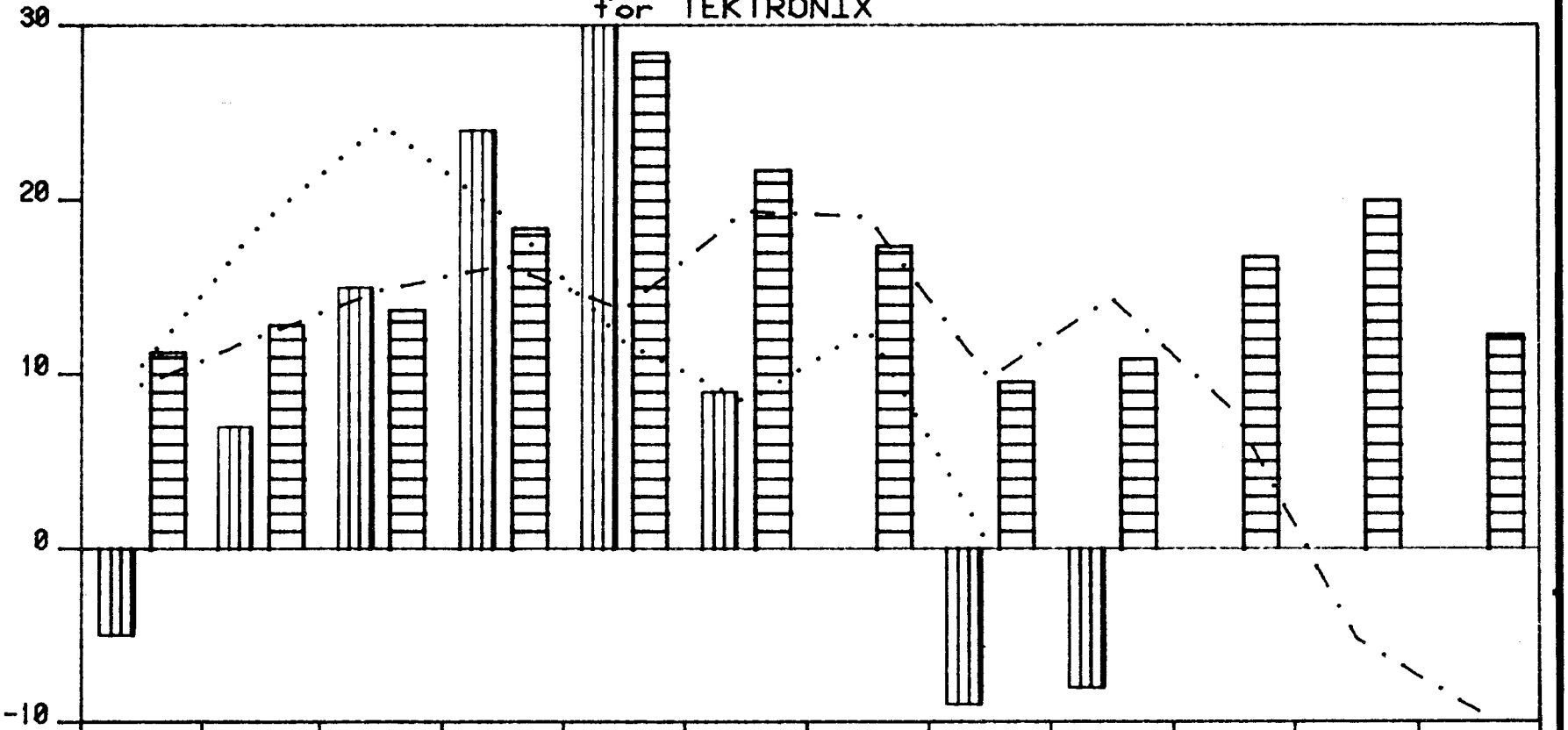
UDK #1 Returns you to main menu without changing any data. Use to recover from errors.

UDK #5 Previews the chart on the screen without the tabular data printed below.

UDK #6 Plots the chart on the plotter.

THIS IS AN EXAMPLE of the DATA CHART PROGRAM for TEKTRONIX

Scale Note



	OCT	NOV	DEC	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP
These	-5	7	15	24	35	9		-9	-8			
... are	10.5	18.6	24.3	19.1	11.8	8.4	12.6	0.0				
line	11.25	12.84	13.69	18.41	28.45	21.73	17.36	9.57	10.87	16.74	19.98	12.26
-- headings.	9.365	12.145	14.789	16.258	13.654	19.321	18.963	9.741	14.325	7.854	-5.219	-9.458

Tektronix®
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE	ABSTRACT NUMBER
SDBAR	TEKniques Vol. 6 No. 4 T1 Program 13
ORIGINAL DATE: June 1, 1982 / REVISION DATE:	EQUIPMENT AND OPTIONS REQUIRED 4052 Graphics System(8K min) 4663 Plotter
AUTHOR Leland C. Sudlow, Purdue University, West Lafayette, IN	PERIPHERALS

ABSTRACT

SDBAR is a short program which will draw Std. Dev. bars on multiple line graphs that have been previously generated from any source. The program requires the user to directly control the plotter via the joystick to mark the viewport for the plotter and the points which need Std. Dev. bars. The user then will input directions (which direction to draw the line and the length of the line for the Std. Dev. bar) for each point on each line.

Files: 2 ASCII Program

Statements: 194

TITLE	ABSTRACT NUMBER
SDBAR	TEKniques Vol. 6 No. 4 T1 Program 13

Operating instructions

Load SDBAR from tape or disc.
Make sure the graph you want to use is on the plotter.
Check the Parameter Entry Card to be sure that the system is configured correctly for your graph.

(1) Input the number of lines on the graph
(2) Move the pen arm via the joystick control so that the crosshairs are over the origin. Then move the crosshairs over the lower right corner of the graph, then move the crosshairs over the upper right corner of the graph.

Note*** the lower right and upper left corners refer to the ends of the X and Y axis respectively

(3) Input the highest and lowest values of the Y and X axis
(4) Move the crosshairs to the first point on the line. Indicate whether the plotter is to mark the Std. Dev. bar up, down, or in both directions from the current point on the line. Then input the Std. Dev. around the point. The plotter will then mark the Std. Dev. bar according to the user's instructions.
(5) Repeat step 4 until all the points in all the lines on the graph have been marked.

Listing

Attached

Example of output

Attached

*** Note ***

Type in, save, and run the attached program called "DUMMY.SDB". This program will generate a two lined, 15 point graph with which to practice SDBAR on. The maximum for both the X and Y axis is 100, and the minimum for both is 0. It would be best to practice using SDBAR before using this program on important graphs.

TITLE	ABSTRACT NUMBER
SDBAR	TEKniques Vol. 6 No. 4 T1 Program 13

Internal Data Storage

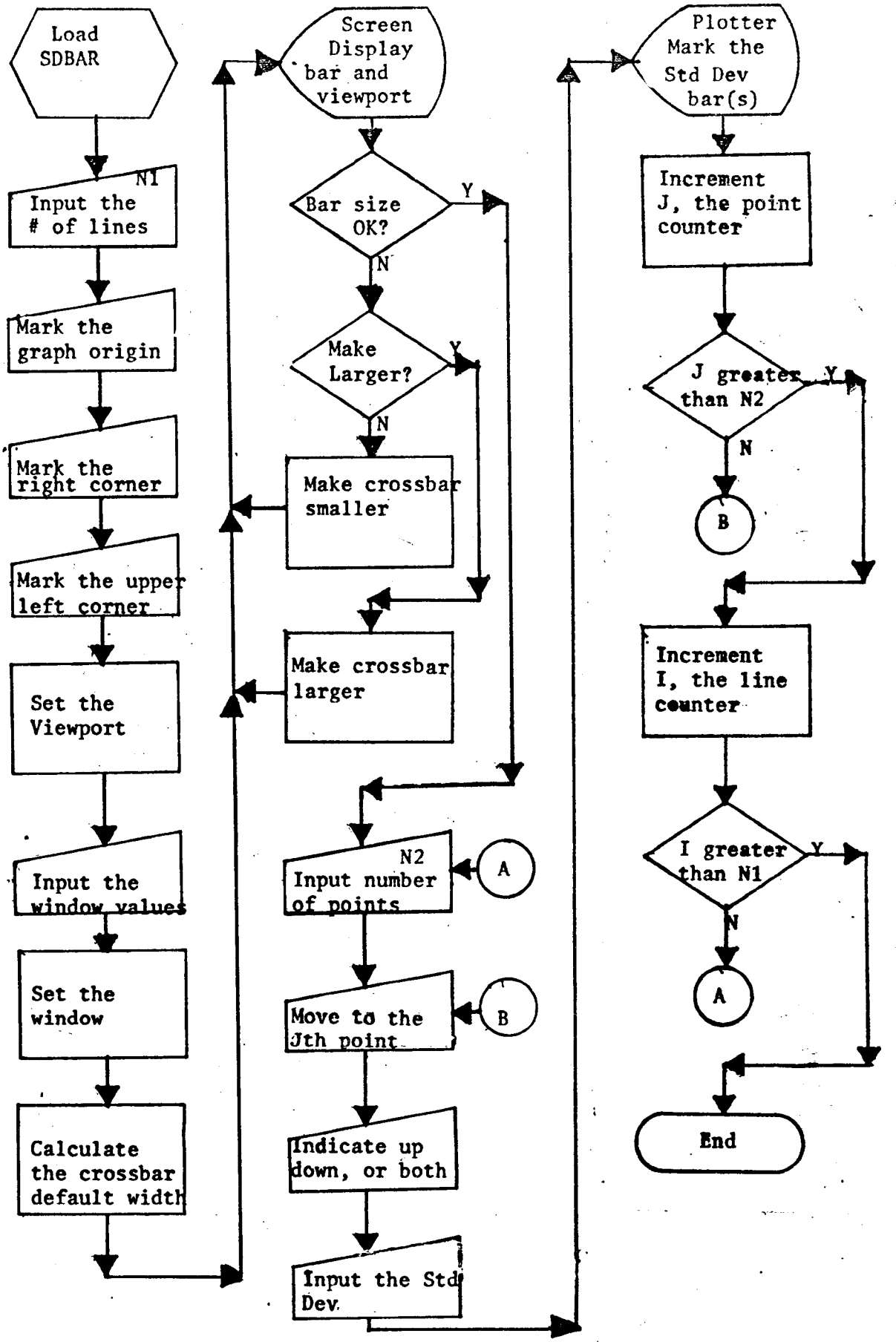
Variable	Used to store...	Type
B$	Keyboard input	String
C	Adjusted crossbar width	Simple
D	Address of plotter	Simple
G	Default width of crossbar	Simple
H1	Low value X-axis scale	Simple
H2	High value X-axis scale	Simple
I	Line Counter	Simple
J	Point Counter	Simple
N1	Number of lines on graph	Simple
N2	Number of points on line	Simple
Q1	Std. Dev. around each point	Simple
U1	X coordinate for origin	Simple
U2	High X viewport value	Simple
U9	Low X viewport value	Simple
V1	Y coordinate for origin	Simple
V2	High Y viewport value	Simple
V3	High value Y-axis scale	Simple
V4	Low value Y-axis scale	Simple
V9	Low Y viewport value	Simple
X1	Current pen position, X	Simple
Y1	Current pen position, Y	Simple
Z$	Keyboard input	String

TITLE

SDBAR

ABSTRACT NUMBER

TEKniques Vol. 6 No. 4 T1
Program 13

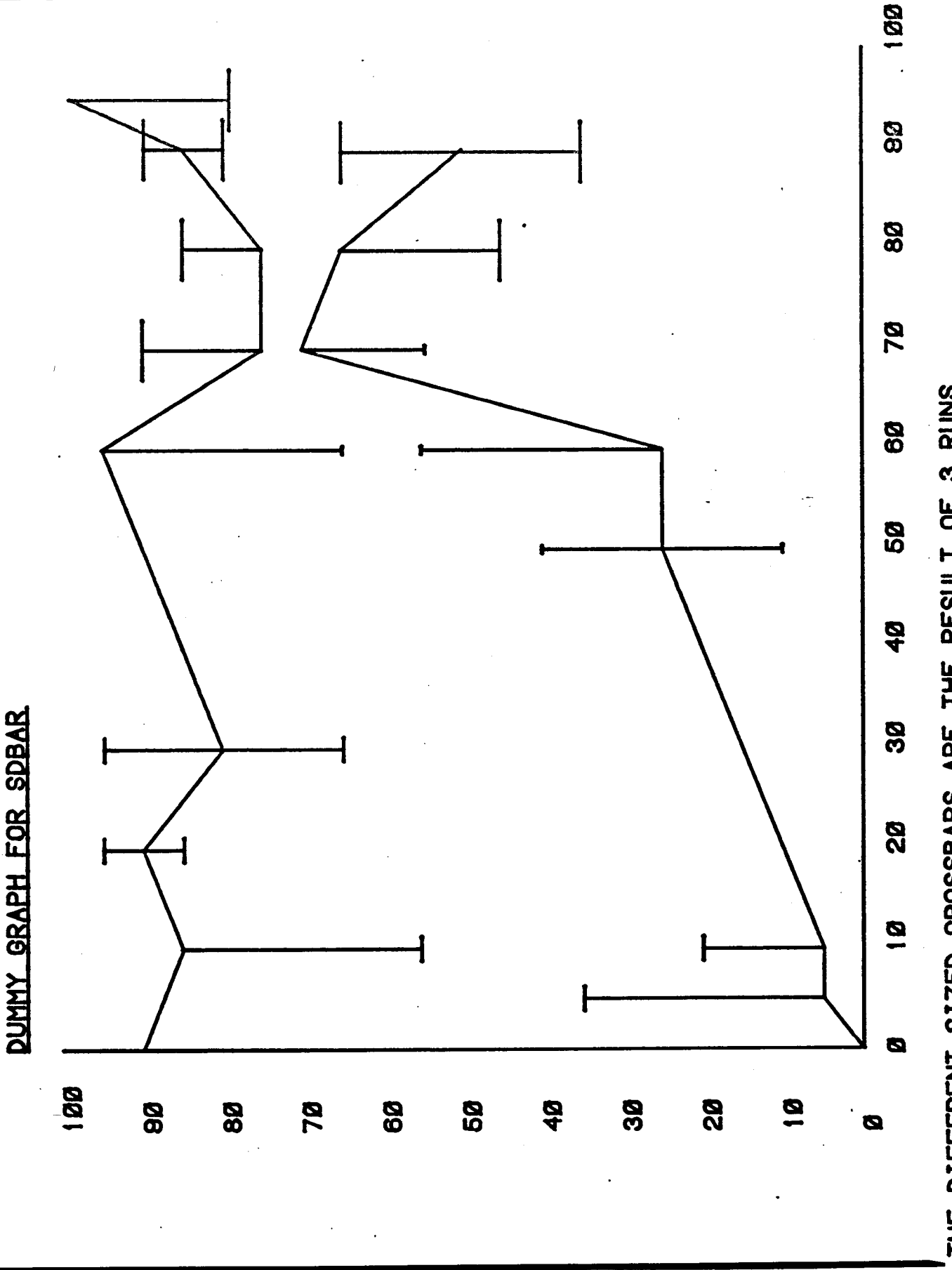


TITLE

SDBAR

ABSTRACT NUMBER

TEKniques Vol. 6 No. 4 T1
Program 13

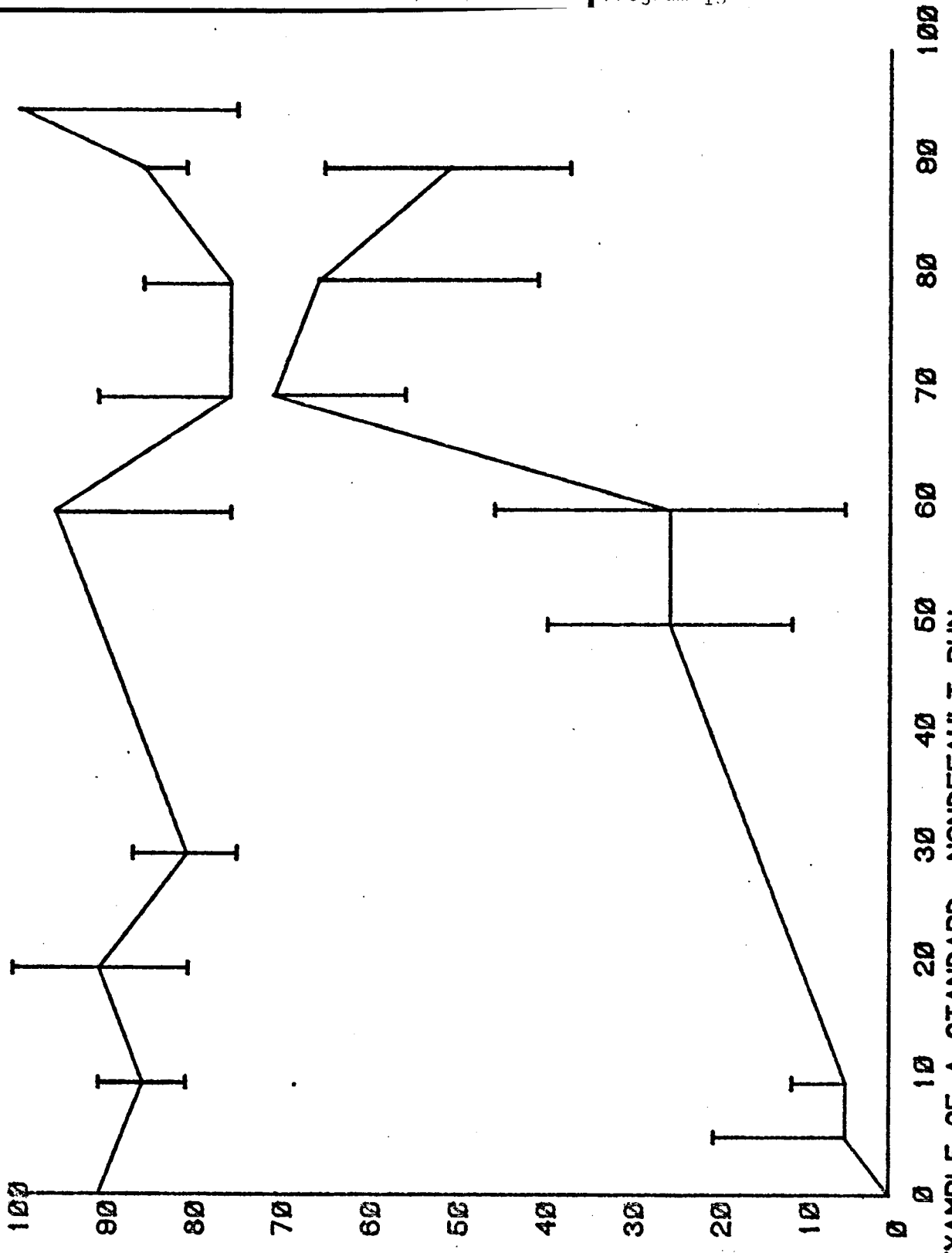


TITLE

SDBAR

ABSTRACT NUMBER

TEKniques Vol. 6 No. 4 T1
Program 13



Tektronix®
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE	ABSTRACT NUMBER
4050/468 Utility III	TEKniques Vol. 6 No. 4 T1 Program 14
ORIGINAL DATE: May, 1982 / REVISION DATE:	EQUIPMENT AND OPTIONS REQUIRED: 4052/4 64K Bytes
AUTHOR: Craig Bulmer, Tektronix, Inc. Chicago, IL	PERIPHERALS: 4052R07 and 4052R08

ABSTRACT

Tektronix 468 Digitizing Oscilloscope

Files: 1 AutoLoad
1 Binary Program
1 ASCII Program

Statements: 1257

This program is similar to 4050/468 Utility II (in TEKniques Vol. 6 No. 1 T1 tape) with additional features. The addition of these functions was at the cost of plotter support.

The program will take waveforms from the 468 Oscilloscope and display the waveforms on the 4050 screen; with printed header information of Channel 1, 2 and/or Add; Volts/Div; Time/Div; Trigger Point; Max Volts; Min Volts; Min/Max Pulse Parameters; Histogram Pulse Parameters; Integrate Waveform; Differentiate Waveform; FFT; and Waveform Analysis.

Added functions will multiply waveforms (channel 1 x channel 2 waveform stored in Add channel); Lissajous pattern (channel 1 vs. channel 2), and waveform cursors on 4052 display with analysis of data between cursors. (Cursors are moveable with constant readout of both cursors' voltage and time from start of sweep and delta time and voltage between cursors.)

TITLE	ABSTRACT NUMBER
4050/468 Utility III	TEKniques Vol. 6 No. 4 T1 Program 14

PRELIMINARY OPERATING INSTRUCTIONS

The 4050/468 Utility III tape must be set up in the following fashion:

File 1 - Autoload Program

File 2 - 4050/468 Utility III program in BINARY

File 3 - 4050/468 Utility III program in ASCII

The program maintains the data files and assumes the data files begin on file 4.

Therefore, mark the 4050/468 Utility III tape:

```
FIN 1, MARK 1,768
FIN 2, MARK 1,35328
FIN 3, MARK 1,32512
```

Put a small autoload program on file 1:

```
100 FIND 2
110 CALL "BOLD"
120 END
```

Transfer the 4050/468 Utility III program from TEKniques tape to the dedicated tape:

INSERT TEKniques VOL. 6 NO. 4 T1 tape:

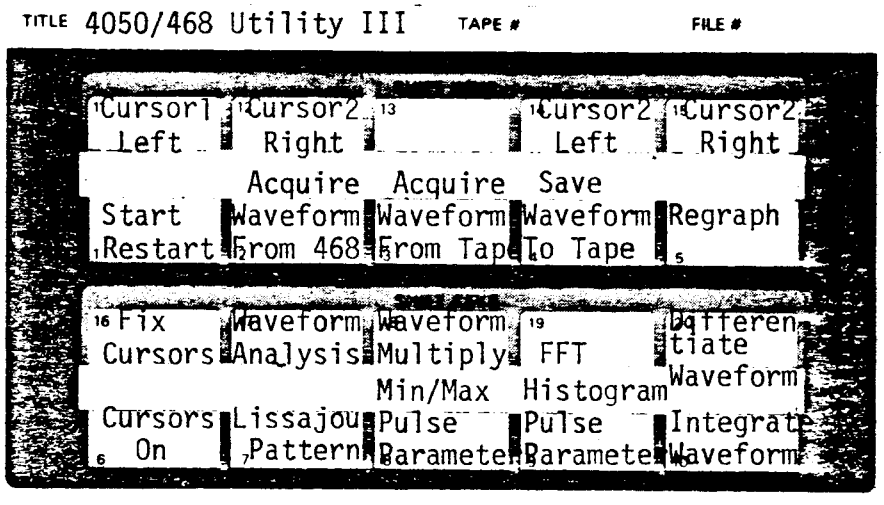
```
FIND 18
OLD
```

INSERT 4050/468 Utility III tape:

```
FIND 2
CALL "BSAVE"
FIND 3
SAVE
```

Your 4050/468 Utility III tape is now ready to use.

TITLE	ABSTRACT NUMBER
4050/468 Utility III	TEKniques Vol. 6 No. 4 T1 Program 14



FUNCTION KEYS:

FK1 (Start/Restart): Initialize system.

FK2 (Acquire Waveform from 468): Acquires all digitized waveforms display on 468 (up to 3 waveforms) except save reference and displays to screen of 4052 with scope grid and scope settings plus min. and max. voltage of each waveform.

FK3 (Acquire waveform from tape): Display previously saved waveforms in same format as function key 2. Find file by name.

FK4 (Save waveform to tape): Save all acquired waveforms to tape with a name (40 Characters).

FK5 (Regraph): Display current waveforms in same format as function key 2.

FK6 (Cursors on): Displays chosen waveform on 4052 screen with moveable cursors and with constant readout of time from start of sweep for both cursors and voltage readout of both cursors; also delta time and voltage between cursors.

TITLE

4050/468 Utility III

ABSTRACT NO:

TEKniques Vol. 6 No. 4 T1
Program 14

FUNCTION KEYS: (continued)

FK7 (Lissajous Pattern): Uncalibrated display of channel 1 vs. channel 2 amplitude data.

FK8 (Min/Max Pulse Parameters): Display of chosen waveform with voltage levels and time at 0%, 10%, 50%, 90%, and 100%. 0% is at minimum of waveform and 100% is at maximum of waveform. Negative time is indication that a negative going transition was encountered first and is a measure of fall time. If at least 3 50% crossing are on the display the time between the first and third crossing is displayed and the symmetry first to second 50% crossing vs. first to third 50% crossing is also displayed.

FK9 (Histogram Pulse Parameters): Same features as FK8 but 0% level is determined by greatest number of voltage values at lower level, and 100% level is determined by greatest number of voltage values at upper level. Truest measurement for pulses with overshoot and/or undershoot. Because of program size the system runs out of memory if this function is selected when 3 waveform (alternate sweep) are in memory; if this condition exists the option of deleting one of the waveforms is offered and then returns to perform analysis.

FK10 (Integrate Waveform): Displays chosen waveform and the waveform after integration is applied.

FK11 (Cursor 1 Left): Moves cursor 1 left for FK6 will not go past first location on screen.

FK12 (Cursor 1 Right): Moves cursor 1 right for FK6 will not go past cursor 2.

FK13 (No Function): No-Op.

FK14 (~~Cursor 2 Left~~): ~~Moves cursor~~ 2 left ~~for~~ FK6 ~~will not go past~~ cursor 1.

FK15 (Cursor 2 Right): Moves cursor 2 right for FK6 will not go past last location on screen.

FK16 (Fix Cursors): Ends FK6 routine fixes and marks cursor locations; prints out time from start of sweep for both cursors; voltage level for both cursors and voltage and time between cursors. The minimum and maximum voltage, the mean (AC & DC) voltage, and true RMS (AC & DC) voltage is displayed for the part of the waveform between cursors.

TITLE	ABSTRACT NO:
4050/468 Utility III	TEKniques Vol. 6 No. 4 T1 Program 14

FUNCTION KEYS: (continued)

FK17 (Waveform Analysis): Displays chosen waveform and prints mean (AC & DC), RMS (AC & DC), minimum, and maximum voltage of displayed waveform. If there are at least 3 50% crossings, based on min. and max. waveform values, prints out mean, RMS, period, and symmetry of waveform between first and third crossings and marks first 3 50% crossings.

FK18 (Waveform Multiply): Multiplies channel 1's waveform by channel 2's waveform storing results in add channels the wavefrom is then displayed and analyzed using waveform analysis FK17. If there is already data in add channels the program asks if you wish to continue this operation. For most accurate results use chopped mode operation or trigger both channels from the same source; also have on screen display of both channels approximately covering the same number of divisions (display size not voltage).

FK19 (FFT): First ask if you wish cosine window applied to chosen waveform. If yes, original waveform and waveform after cosine window is applied are displayed. After a pause the screen is erased and the amplitude and phase of the chosen waveform is displayed. The frequency range of the 2 displays is from DC to the Nyquist Frequency. Because of program size the system runs out of memory when 3 512 point waveforms are in memory. If this condition exists the option of deleting one of the waveforms is offered and then returns to perform analysis.

FK20 (Differentiate Waveform): Displays chosen waveform and the waveform after either 2-point or 3-point differentiation is applied.

All displays except Lissajous Pattern (FK7) have amplitude and time scaling annotated on their axis.

For most accurate results set ground references.

Hopefully this software is modular enough and commented enough that the package can be easily broken apart to meet application needs.

IMPORTANT VARIABLES:

Variables for bringing in waveforms from 468 and for waveform analysis:

TITLE	ABSTRACT NO:
4050/468 Utility III	TEKniques Vol. 6 No. 4 T1 Program 14

IMPORTANT VARIABLES: (continued)

A$ String Dim (175) contains waveform header.

B Number of points in waveform.

S1 Volts/Div.

T1 Time/Div.

R$ String horizontal units (i.e. MS).

S$ String vertical units (i.e. MV).

Y Array Dim (B) digitized values.

Y5 Vertical zero.

Y6 Vertical multiplier.

Channel 1 variables:

X$ String Dim (180) contains channel 1 waveform header.

B1 Number of points in channel 1 waveform.

Y1 Array Dim (B1) raw digitized values of channel 1.

Channel 2 variables:

Y$ String Dim (180) contains channel 2 waveform header.

B2 Number of points in channel 2 waveform.

Y2 Array Dim (B2) raw digitized values of channel 2.

Add Channel & Waveform Multiply variables:

Z$ String Dim (180) contains add channel or result of waveform multiply waveform header.

B3 Number of points in add channel or as a result from waveform multiply.

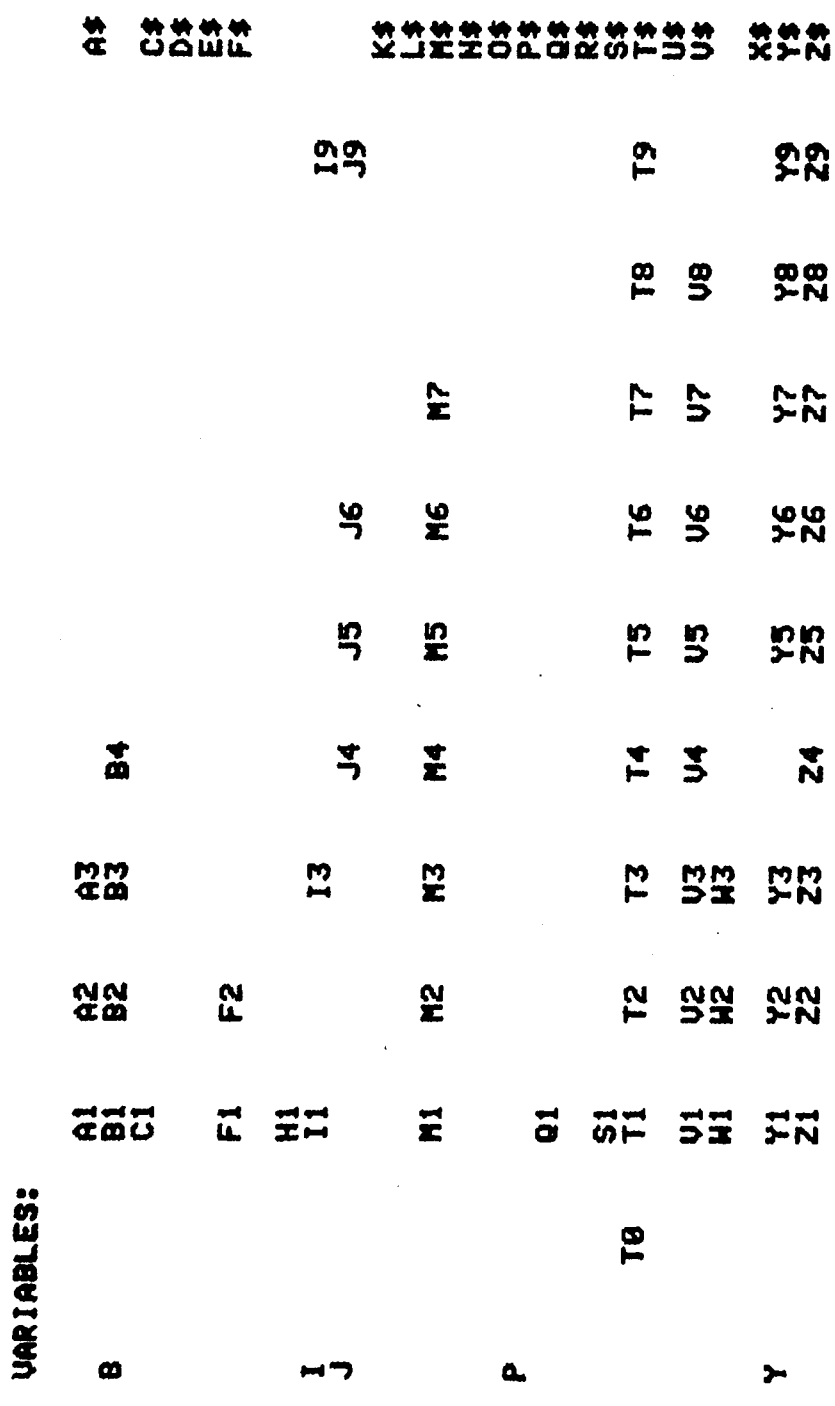
Y3 Array Dim (B3) raw digitized values of add channel or results of waveform multiply.

TITLE

4050/468 Utility III

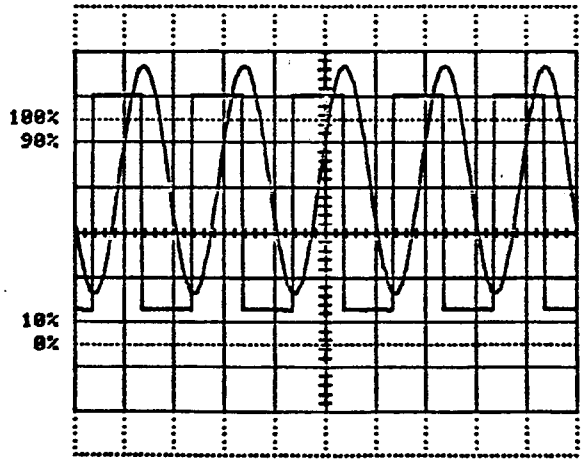
ABSTRACT NUMBER

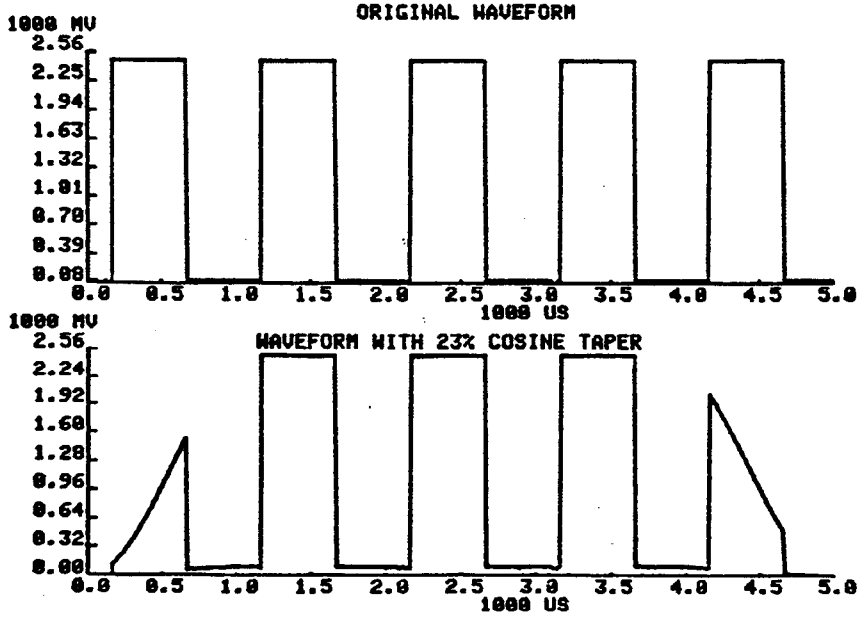
TEKniques Vol. 6 No. 4 T1
Program 14



TITLE	ABSTRACT NUMBER
4050/468 Utility III	TEKniques Vol. 6 No. 4 T1 Program 14

	CHANNEL_1 ACQUIRED	CHANNEL_2 ACQUIRED
COUPLING	DC	DC
TIME/DIV	500.00 US	500.00 US
VOLTS/DIV	500.00 MV	100.00 MV
TRIGGER PT.	640.00 US	640.00 US
MAX VOLTS	2.480 V	356.000 MV
MIN VOLTS	80.000 MV	-140.000 MV



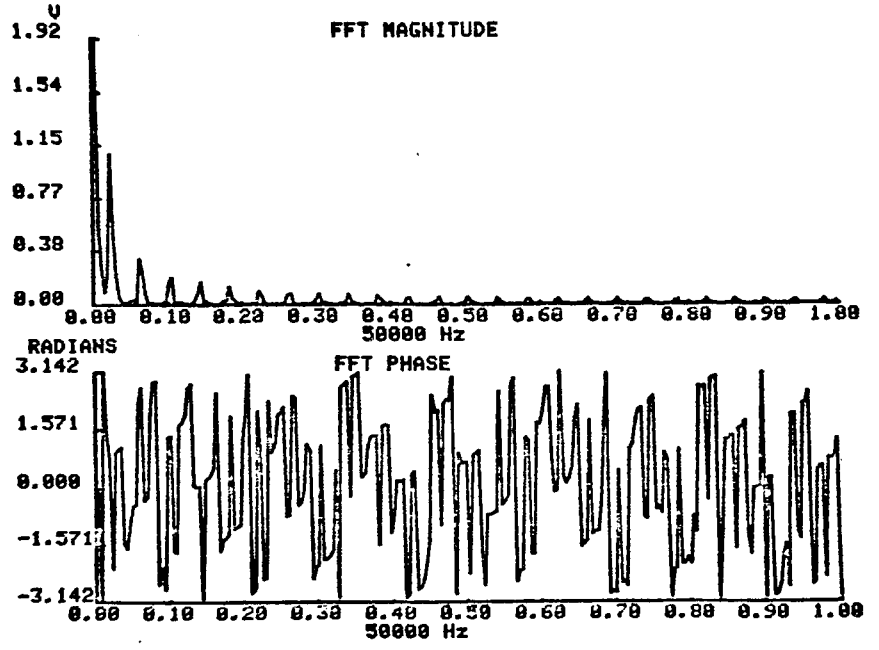


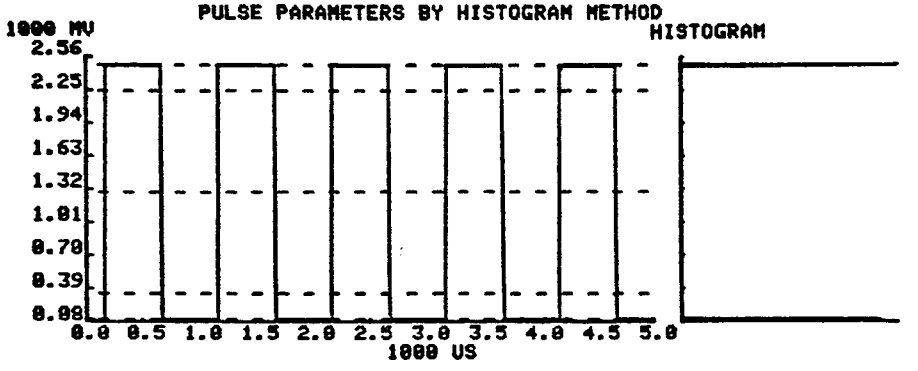
TITLE

4050/468 Utility III

ABSTRACT NUMBER

TEKniques Vol. 6 No. 4 T1
Program 14





W A V E F O R M P A R A M E T E R S

0% LEVEL:	99.41 MV	RISE TIME	7.943E+000 US
10% LEVEL:	337.71 MV	90% WIDTH	4.920E+002 US
50% LEVEL:	1290.88 MV	50% WIDTH	5.000E+002 US
90% LEVEL:	2244.05 MV	10% WIDTH	5.079E+002 US
100% LEVEL:	2482.35 MV	FALL TIME	8.010E+000 US
		PERIOD AT 50% WIDTH	1.000E+003 US
		SYMMETRY AT 50% WIDTH	50.00 %

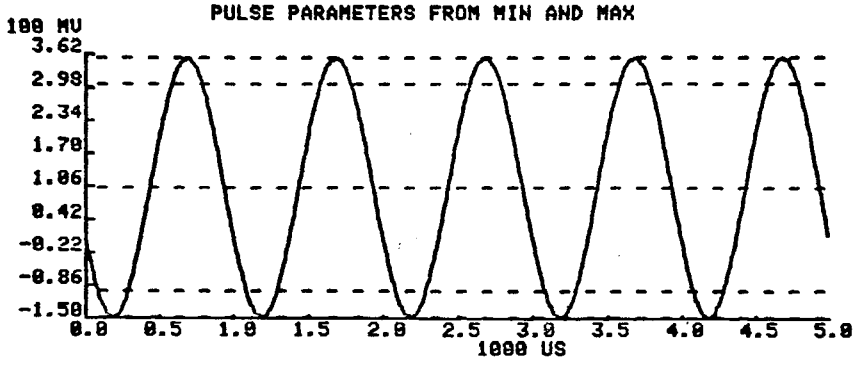
TITLE

4050/468 Utility III

ABSTRACT NUMBER

TEKniques Vol. 6 No. 4 T1
Program 14

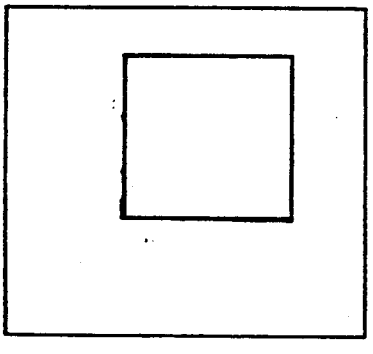
PULSE PARAMETERS FROM MIN AND MAX



W A V E F O R M P A R A M E T E R S

0% LEVEL:	-148.00 MV	RISE TIME	5.000E+002 US
10% LEVEL:	-97.60 MV	90% WIDTH	2.040E+002 US
50% LEVEL:	104.00 MV	50% WIDTH	5.000E+002 US
90% LEVEL:	305.60 MV	10% WIDTH	2.017E+002 US
100% LEVEL:	356.00 MV	FALL TIME	-5.023E+002 US
		PERIOD AT 50% WIDTH	1.000E+003 US
		SYMMETRY AT 50% WIDTH	50.00 %

CHANNEL 1 vs. CHANNEL 2 DISPLAY
AMPLITUDE UNCALIBRATED

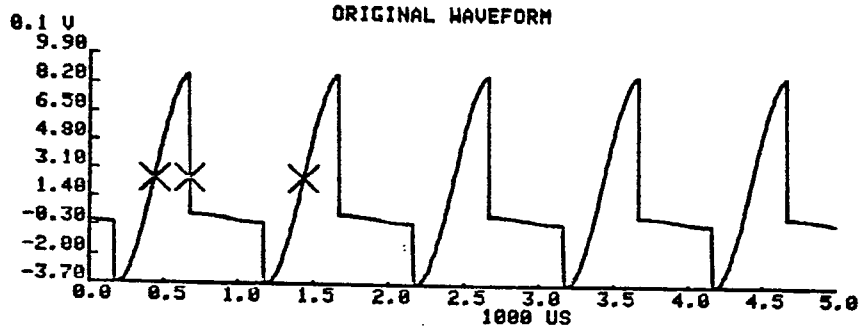


TITLE

4050/468 Utility III

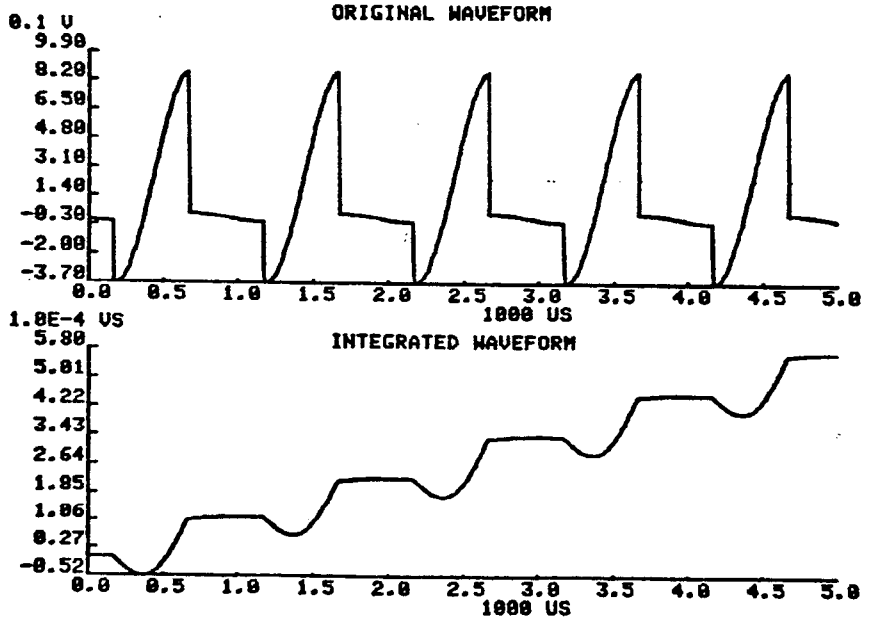
ABSTRACT NUMBER

TEKniques Vol. 6 No. 4 T1
Program 14

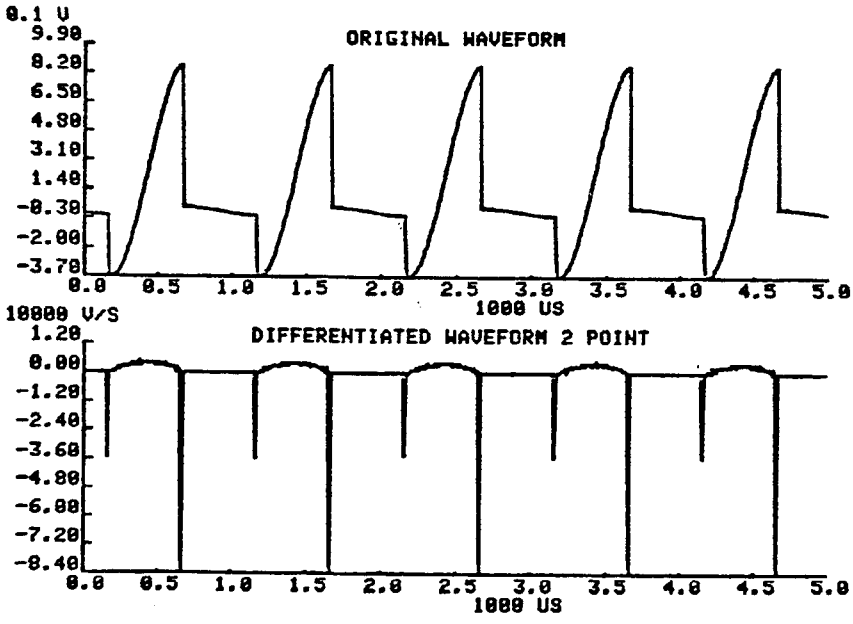


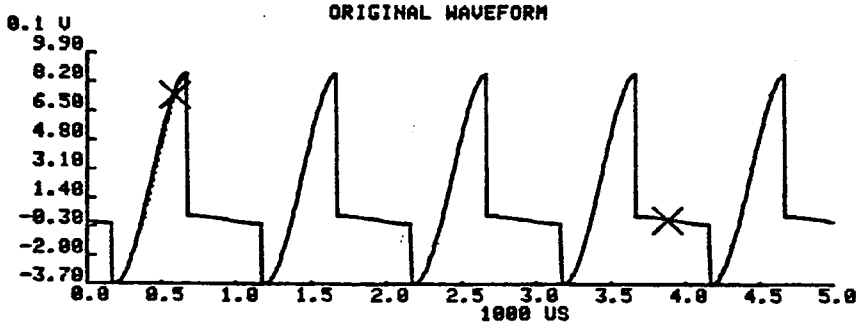
WAVEFORM PARAMETERS

MEAN OF WAVEFORM (AC+DC)	=	0.1141 V
RMS OF WAVEFORM (AC+DC)	=	0.3469 V
MAX OF WAVEFORM	=	0.8738 V
MIN OF WAVEFORM	=	-0.3670 V
MEAN OF ONE CYCLE (AC+DC)	=	0.1156 V
RMS OF ONE CYCLE (AC+DC)	=	0.3473 V
PERIOD OF ONE CYCLE	=	1000.0000 US
SYMMETRY AT 50% LEVEL	=	23.6151 %



TITLE	ABSTRACT NUMBER
4050/468 Utility III	TEKniques Vol. 6 No. 4 T1 Program 14





CURSOR PARAMETERS

CURSOR 1 VOLTAGE	CURSOR 1 TIME	CURSOR 2 VOLTAGE	CURSOR 2 TIME	DELTA VOLTAGE	DELTA TIME
0.75 V	590.00 US	0.02 V	3900.00 US	0.74 V	3310.00 US

VALUES BETWEEN CURSORS

MIN = -0.367 V MAX = 0.873 V

MEAN (AC+DC) = 0.1280 V RMS (AC+DC) = 0.3577 V

Tektronix
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE		ABSTRACT NUMBER
Inventory Control		TEKniques Vol. 6 No. 4 T1 Program 15
ORIGINAL DATE March, 1978	REVISION DATE June, 1979	EQUIPMENT AND OPTIONS REQUIRED 32K
AUTHOR Larry E. Davies	Tektronix, Inc. St. Louis, MO	PERIPHERALS 4641 Printer w/Opt. 10 I.F. Editor ROM Pack

ABSTRACT

Files: 4 ASCII Program

Statements: 319

This program was designed to handle approximately 465 11-digit (including hyphens, i.e., 670-4289-01) part numbers to control field service exchange assemblies.

Five functions are available: ADD, ALTER, LOCATE, LIST, SORT

The fields are stored and listed as follows:

PART #	MODEL-DESCRIPTION	LOCATION
(11 digits)	(20 digits)	(20 digits)

Each entry can be altered at any time with error-checking capabilities so that one's entry doesn't exceed the number of digits allowed for that field. Also, any part number can just be located if desired.

The entire inventory can be sorted by Part #, Model, or Location.

TITLE	ABSTRACT NUMBER
Inventory Control	TEKniques Vol. 6 No. 4 T1 Program 15

PRELIMINARY OPERATING INSTRUCTIONS

The four files comprising this program must be transferred to a dedicated tape.

Following the instructions on page iii, transfer the following files:

FROM TEKniques Vol. 6 No. 4 T1 tape			TO Inventory Control Tape	
File #	Type	Bytes	File #	Bytes
19	ASCII Prog	4864	1	4864
20	" "	1280	2	1280
21	" "	2048	3	2048
22	" "	1536	4	1536

Then MARK files 5, 6, 7 and 8 for 25088 bytes each.

This will enable you to catalog approximately 465 11-digit part numbers.

TITLE	ABSTRACT NUMBER
Inventory Control	TEKniques Vol. 6 No. 4 T1 Program 15

OPERATING INSTRUCTIONS

Auto-Load the tape and choose from the menu.

ADD

The program will prompt you for the part number, description and location. The data will be entered on file 5 at the end of any data already therein.

Enter STOP to discontinue the entry. You will be returned to the menu.

ALTER

Before running this, you should get a listing of your items (see LIST).

The program will prompt you for the part number of the item you wish to change. It if locates it in the random file, it will display it and ask how you wish to change it (or not change it). When complete, the random data is rewritten to the file with the change made.

LOCATE

The program will prompt you for the part number of the item you wish to locate. If it locates it in the random file, it will display its location (field #3) and prompt for another part number.

To terminate, enter STOP when prompted for the part number.

LIST

This routine will list any one of the four data files: random, by part number, by model (description) or by location.

SORT

This routine requires the 4052R06 Editor ROM. It will sort the random file by whichever field you wish, then write the sorted file out to the correct data file (see TAPE STRUCTURE).

TITLE

INVENTORY CONTROL

ABSTRACT NO:

TEKniques Vol. 6 No. 4 T1
Program 15

2. INTERNAL DATA STORAGE

Variable	Used to Store . . .	Type
S$	Office	String
M$	Date	String
A	Menu Item Selection	Simple
Z$	Main String for Storage of one completed data item	String
H$	Main String for storage of all data items	String
C$	Input string	String
Y	For-next loop	Simple
Q$	Part # to be located Part # to be altered	
Z	Numeric	
N	Numeric	
A$	Used to segment data string	
D$	Used to segment data string	
E$	Used to segment data string	
G$	(Return)	
T$	Spous (20)	
D	Field to change	

3. TAPE STRUCTURE

File 1: Menu and Alter and Locate Routines
File 2: Add Routine
File 3: List Routine
File 4: Sort Routine
File 5: Random Data storage of entered part numbers
File 6: Sorted by Part number data
File 7: Sorted by Model number data
File 8: Sorted by Location data

TITLE	ABSTRACT NUMBER
Inventory Control	TEKniques Vol. 6 No. 4 T1 Program 15

ST.LOUIS
**********EXCHANGE INVENTORY**********
8-21-79

PART#	MODEL-DESCRIPTION	LOCATION
000-0000-XX	4014-15 BISYNC#CV	ORDERED BY SER SUP
000-0000-XX	4014-15 BISYNC #CN	ORDERED BY SER SUP
021-0187-XX	4023 CURRENT LOOP	FIELD SERVICE
118-0151-XX	4956 LOGIC	FIELD SERVICE
118-0152-XX	4956 GPIB	FIELD SERVICE
118-0153-XX	4956 POWER SUPPLY	FFIELD SERVICE
118-0154-XX	4956 PULSER	FIELD SERVICE
118-0155-XX	4956 PREAMP	FIELD SERVICE
119-0304-02	4006-4010 KEYBOARD	FIELD SERVICE
119-0304-02	4010-4006 KEYBOARD	FIELD SERVICE
119-0304-02	4006-4010 KEYBOARD	BIN 21
119-0304-02	4010-4006 KEYBOARD	FIELD SERVICE KIT#2
119-0363-00	4023 MONITOR	R&R #4
119-0363-00	4023 MONITOR	FIELD SERVICE
119-0363-00	4023 MONITOR	FIELD SERVICE
119-0374-01	4023 KEYBOARD	FIELD SERVICE
119-0433-00	TEK 31 MAG TAPE	FIELD SERVICE
119-0433-00	TEK 31 MAG TAPE	FIELD SERVICE
119-0433-XX	TEK31 MAGTAPE(INST)	CALCULATOR
119-0436-00	TEK 31 PRINTER	BIN 22
119-0436-XX	TEK31 PRI FOR INST	CALCULATOR
119-0483-02	4014 KEYBOARD	FIELD SERVICE
119-0483-02	4014 KEYBOARD	R&R SM-035713
119-0483-02	4014 KEYBOARD	FIELD SERVICE
119-0483-03	4012 KEYBOARD	??????
119-0483-03	4012 KEYBOARD	FIELD SERVICE
119-0488-01	4013 KEYBOARD	FIELD SERVICE KIT#1
119-0623-00	4953-54 PREAMP	FIELD SERVICE
119-0623-00	4953-54 PREAMP	FIELD SERVICE
119-0624-00	4953-54 PULSER	FIELD SERVICE
119-0624-00	4953-54 PULSER	FIELD SERVICE
119-0707-01	4051 KEYBOARD	FIELD SERVICE KIT#1
119-0707-03	4051 KEYBOARD	FIELD SERVICE KIT#2
119-0708-00	4051 MAG TAPE	FIELD SERVICE
119-0708-00	4051 MAG TAPE	R&R SM=022477
119-0708-03	4051 MAG TAPE	FIELD SERVICE
119-0769-00	4081 CPU HI	FIELD SERVICE
119-0770-00	4081 CPU LO	FIELD SERVICE
119-0908-00	4025 MONITOR ASSY	CRT AREA
119-0908-00	4025 MONITOR ASSY	CRT AREA
119-0909-01	4025 KEYBOARD	FIELD SERVICE

TITLE

Inventory Control

ABSTRACT NUMBER

TEKniques Vol. 6 No. 4 T1
Program 15

```
672-0426-00      4632 VIDEO I/F          FIELD SERVICE KIT#1
672-0443-04      4632 MUX I/F            FIELD SERVICE KIT#1
672-0466-01      4632 VIDEO              FIELD SERVICE KIT#1
672-0488-01      4631 I/F                FIELD SERVICE KIT#1
672-0488-01      4631 I/F                FIELD SERVICE KIT#2
672-0488-01      4631 INTERROGATE        FIELD SERVICE
672-0489-01      4631 MUX I/F            FIELD SERVICE KIT#1
672-0489-01      4631 MUX ASSY           BIN 16
672-0489-01      4631 MUX                FIELD SERVICE
672-0503-02      4631-32 H.V.            R&R SM-035679
672-0503-04      4631-32 H.V.            FIELD SERVICE
672-0503-05      4631-32 H.V.            FIELD SERVICE
672-0503-05      4631-32 H.V.            R&R SM-035663
672-0503-XX      4631-32 H.V.            R&R SM-035684
672-0503-XX      4631 H.V.               FIELD SERVICE
672-0526-00      4006 P.S.               FIELD SERVICE
672-0537-06      4006 DISPLAY            FIELD SERVICE
672-0541-00      4662 P.S.               R&R SM-035693
672-0541-02      4662 P.S.               FIELD SERVICE
672-0546-02      4051 DISPLAY            FIELD SERVICE
672-0546-03      4051 DISPLAY            R&R SM-035676
672-0692-01      4633 H.V.               BIN 7
672-0739-00      4025 DISPLAY MEM        FIELD SERVICE
672-0740-00      4025 DISPLAY MEM        FIELD SERVICE
672-0744-00      4025 GRAPHICS MEM       FIELD SERVICE KIT#1
672-0745-00      4025 GRAPHIC MEN        FIELD SERVICE KIT#2

THIS LIST IS UPDATED WEEKLY. PLEASE CONSULT CURRENT LIST
```

Tektronix®
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE	ABSTRACT NUMBER
Flowcharter II	TEKniques Vol. 6 No. 4 T1 Program 16
ORIGINAL DATE: September 1979 / REVISION DATE:	EQUIPMENT AND OPTIONS REQUIRED: 4050 32K
AUTHOR Richard G. Meitzler 714-225-2726 USS Barb (SSN 596),FPO San Francisco 96601	PERIPHERALS: 4631 Hard Copy Unit

ABSTRACT

Programs: 5

Statements: 1168

This program will flowchart any 4050 BASIC program stored in ASCII.

The first pass of the program builds a branch table, a FOR...TO table and a NEXT table.

The second pass matches the FOR...NEXT statements in the two tables.

The third pass draws the actual flowchart using standard ANSI symbols.

The page number, starting and finishing line numbers are printed at the bottom of each page.

The user may store the program being analyzed and the results of the first two passes on tape.

Program limitations: 500 branches
200 FOR statements
200 NEXT statements

TITLE	ABSTRACT NUMBER
Flowcharter II	TEKniques Vol. 6 No. 4 T1 Program 16

PRELIMINARY OPERATING INSTRUCTIONS

The five files comprising the flowchart program must be transferred to a dedicated tape. 16 additional files must be premarked.

1. Mark the following files on the new Flowchart tape:

File #	Bytes Required	File #	Bytes Required
1	4096	10	10240
2	2560	11	8192
3	3072	12	30208
4	14848	13	10240
5	10240	14	8192
6	8192	15	30208
7	30208	16	10240
8	768	17	8192
9	768	18	30208
		19	10240
		20	8192
		21	30208

2. Using the Instructions on Page iii, transfer the following files from the TEKniques tape to the Flowcharter II tape:

FROM TEKniques Tape	TO Flowcharter II Tape	
File #23	File #1	
File #24	File #2	
File #25	File #3	
File #26	File #4	
File #27	File #8	(note: out of sequence!)

TITLE

Flowchart Program

ABSTRACT NO:

TEKniques Vol. 6 No. 4 T1
Program 16

2. TAPE STRUCTURE

File	Format	Contents	Space	
1	ASCII	Part I	4096	
2	ASCII	Part II	2560	
3	ASCII	Part III	3072	
4	ASCII	Part IV	14848	
5	BINARY	Work File	10240	Data-Length of 'FOR table (3,C3)/ Length of 'FOR' string/Length of Branch table(C2)/"FOR' table/'FOR' String/Branch table/Title/Indicator for Set Key.
6	BINARY	Work File	8192	Data-Number indicating if the data is from part II or III/Length of 'NEXT' table (2,C1) or (C1)/"NEXT' table.
7	ASCII	Program	30208	Program being analyzed.
8	ASCII	Part I-1	768	
9	BINARY	Data I-1	768	Two File numbers used with Part I-1.
10	BINARY	Storage	10240	Same as file 5.
11	BINARY	File	8192	Same as file 6.
12	ASCII	One	30208	Same as file 7.
13	BINARY	Storage	10240	Same as file 5.
14	BINARY	File	8192	Same as file 6.
15	ASCII	Two	30208	Same as file 7.
16	BINARY	Storage	10240	Same as file 5.
17	BINARY	File	8192	Same as file 6.
18	ASCII	Three	30208	Same as file 7.
19	BINARY	Storage	10240	Same as file 5.
20	BINARY	File	8192	Same as file 6.
21	ASCII	Four	30208	Same as file 7.

The program and data are all on the same tape and must be pre-marked in accordance with the forgoing table.

TITLE	ABSTRACT NO:
Flowchart Program	TEKniques Vol. 6 No. 4 T1 Program 16

1. DESCRIPTION:

When the program is started, it starts with Part I and prints the menu. Option (1) allows the operator to transfer the program to be flowcharted manually. When (1) is selected the program prints a message telling the operator to put the program in file 7, in ASCII, then ends. When option (2) is selected, the program prompts the operator to put in the tape that has the program to be analyzed and enter the file number. The program to analyzed must be in ASCII. The program then finds the file and reads the program into a string (P$), then prompts the operator to put in the flowchart tape and press "RETURN". The program in the string (P$) is put on file 7. The program then continues with Part II. When option (3) is selected the computer reads Part II into memory and runs it. Option (3) can also be selected by pressing the 'RETURN' key. When option (4) is selected the program asks the operator which set of storage files he wants the data transfered to, either 1,2,3,or 4. The program then finds the selected storage file , and if any data was previously stored in that file, then asks the operator if he wants to write over that data. If the operator says no, the program asks for another storage file. The program then transfers the two data files from the work files to the storage files selected, the using Part I-1, the program that is associated with the data files is transfered from the work file to the selected storage file. Part I is then read back into memory and run. When option (5) is selected the process is almost idential but a selected storage file is transfered to the work files. When option (6) is selected the program prompts the operator to input the file number of the file he wishes the name of. File Ø for the work file and files 1,2,3, or 4 for the storage files. The program the finds the file and prints the title. When option (7) is selected Part III of the program is read into memory and run. Part II must have been run previously. When Part II is run, the program asks the operator for the title. The program then goes to file 7 and reads one program line at a time for analysis. The program makes three tables of data. The branch table has the line numbers of all the lines branched TO in the program. The "FOR" table has the line numbers of the line after the "FOR" statements, the length of the "FOR" statements and the position in the "FOR" string that the "FOR" statement will be put into for future use. The "NEXT" table has all the line numbers of all the "NEXT" statements in the program. The branch table is then sorted numerically. The three tables, the "FOR" string and the title is recorded on data files 5&6. Part III is then read into memory and run. Part III matches all the "NEXT" statements in the "NEXT" table with the "FOR" statements from the "FOR" table.

TITLE	ABSTRACT NO:
Flowchart Program	TEKniques Vol. 6 No. 4 T1 Program 16

1. DESCRIPTION (Con't)

The modified "NEXT" table is the put in data file 6. Part IV is then read into memory and run. Part IV draws the actual flowchart using standard ANSI symbols. When a page isfilled, the page number and the starting and finishing line numbers are printed at the bottom of the page. The the program waits for the operator to press "RETURN", then it continues with the next page. When the program has been flowcharted, the operator presses "RETURN" and program looks through the program being analyzed and prints all DATA and IMAGE statements.

TITLE	ABSTRACT NO:
Flowchart Program	TEKniques Vol. 6 No. 4 T1 Program 16

3. INTERNAL DATA STORAGE

a. List in a table format all variables used by the program:

Variable	Used to Store . . .	Type
	PART I	
A	Scratch	Simple
B	Branch Table	Arry (C2)
B1	Length of F$	Simple
C1	Length of 'NEXT' table	Simple
C2	Length of BRANCH table	Simple
C3	Length of 'FOR' table	Simple
D	Number of starage file selected	Simple
F	FOR table	Array (3,C3)
F1	First Data file of the 'TO' storage file	Simple
F2	Second Data file of the 'TO' storage file	Simple
F3	Program file of the 'TO' storage file	Simple
F4	First Data file of the 'FROM' storage file	Simple
F5	Second Data file of the 'FROM' storage file	Simple
F6	Program file of the 'FROM' storage file	Simple
K	Data from file F4	Simple
N	"NEXT' table	Array (C1) or (2,C1)
T	Scratch	Simple
A$	Scratch	String
B$	Individual program line as read from the internal tape	Dim (74)
F$	"FOR' String	Dim (B1)
P$	String the Program is stored in	Dim (20000)
Q$	Scratch	String
T$	Title Data	String
	PART I-1	
F3	File number for Program 'TO'	Simple
F6	File number for program 'FROM'	Simple
B$	Scratch	String
C$	Individual program line as read from the internal tape.	Dim (74)
D$	String program is stored in .	Dim (2000)

TITLE	ABSTRACT NO:
Flowchart Program	TEKniques Vol. 6 No. 4 T1 Program #6

3. INTERNAL DATA STORAGE (Con't)

a. List in a table format all variables used by the program:

PART II

Variable	Used to Store . . .	Type
A	Scratch	Simple
A1	Scratch	Simple
B	Branch Table	Array (500)
C0	Max values for loops	Simple
C1	Counter for "NEXT" table (N)	Simple
C2	Counter for Branch Table (B)	Simple
C3	Counter for "FOR" table (F)	Simple
C4	Counter for Z	Simple
D	Scratch	Simple
F	"FOR" table	Array (3,200)
I	Loop	Simple
I1	Loop	Simple
K	Indicator for Set Key	Simple
N	"NEXT" table	Array (200)
Z	Sorted Branch Table	Array (C2)
A$	Scratch	String
B$	Scratch	String
F$	"FOR" String	Dim (3000)
L$	Individual program line as read from the internal tape.	Dim (74)
T$	Program Title	String

PART III

A	Position in program string of a line number	Simple
A1	Scratch	Simple
A2	Scratch	Simple
B	Position in program string of space after line number	Simple
B0	List of branches to follow	Array (75)
B1	Counter for B0	Simple
B5	Master list of all branches	Array (200)
B8	List of all followed	Array (75)
B9	Counter for B8	Simple
C	Scratch	Simple
C0	Counter for "FOR" table	Simple
C1	Length of array M	Simple

TITLE

Flowchart Program

ABSTRACT NO:

TEKniques Vol. 6 No. 4 T1
Program 16

3. INTERNAL DATA STORAGE (Con't)

PART III (Con't)

Variable	Used to Store . . .	Type
C2	Scratch	Simple
C3	Length of "FOR" table	Simple
C5	Counter for B5	Simple
D	Scratch	Simple
F	"FOR" table	Array (C3)
I	Loop	Simple
M	"NEXT" table (Temporary)	Array (C1)
N	"NEXT" table (Final)	Array (2,C1)
Q0	Scratch	Simple
Q1	Scratch	Simple
Q2	Scratch	Simple
X	Scratch	Simple
A$	Scratch	Simple
B$	Individual program line as read from internal tape	Dim(74)
C$	Scratch	String
D$	Scratch	String
E$	Scratch	String
F$	Scratch	String
P$	String program is stored in	Dim (2000)

PART IV

A	Scratch	Simple
A1	Scratch	Simple
A2	Scratch	Simple
A3	Scratch	Simple
B	Branch Table	Array (C2)
B0	Indication if it's a GOTO or GOSUB	Simple
C1	Length of "NEXT" table	Simple
C2	Length of Branch table	Simple
C3	Length of "FOR" table	Simple
C5	Counter for "NEXT" table	Simple
C6	Counter for Branch table	Simple
C7	Scratch	Simple
D	Length of "FOR" String	Simple
F	"FOR" table	Array (3,C3)
F1	Line number of First line on a page	Simple
F2	Line number of last line on a page	Simple
I	Loop	Simple

TITLE	ABSTRACT NO:
Flowchart Program	TEKniques Vol. 6 No. 4 T1 Program 16

3. INTERNAL DATA STORAGE (Con't)

PART IV (Con't)

Variable	Used to Store . . .	Type
I1	Scratch	Simple
K	Indicator for Set Key	Simple
L0	Scratch	Simple
L1	Long side of the Box	Simple
L2	Scratch	Simple
L3	Scratch	Simple
L8	Decimal equivalent for letter in flowchart continuation circle	Simple
N	'NEXT' table	Array (2,C1)
N0	Scratch	Simple
N1	Number of letter in flowchart continuation circle	Simple
P1	Page number	Simple
R	Radius of circle	Simple
S	A number between 1&6 to indicate symbol type	Simple
S1	Short side of Box	Simple
S2	Scratch	Simple
S3	Scratch	Simple
T	Start point of new box, 'Y' value.	Simple
T1	Scratch	Simple
T2	Scratch	Simple
X	Line number of line being analyzed	Simple
X0	Center line for drawing box, 'X' value	Simple
X1	Scratch	Simple
X3	Scratch	Simple
X5	Scratch	Simple
Y	Scratch	Simple
Y1	Scratch	Simple
A$	Scratch	Dim(74)
B$	Scratch	String
C$	Scratch	String
D$	Scratch	String
F$	"FOR" string	Dim(D)
G$	Scratch	String
H$	Scratch	String
J$	Contains ASCII character ''	String
L$	Line read from Program and line to be printed	Dim(74)
M$	Scratch	String
N$	Scratch	String
Q$	Scratch	Dim(74)

TITLE	ABSTRACT NO:
Flowchart Program	TEKniques Vol. 6 No. 4 T1 Program 16

5. OPERATING INSTRUCTIONS:

Insert the Flowchart Program tape and press autoload.
Refer to description for explanation of different menu items.
In the beginning of Part II the program asks for the title; enter a title and press "RETURN".
The rest of Part II is automatic.
Part III is all automatic.
In Part IV when the program has completed drawing a page the operator will see a flashing question mark in the upper left corner of the screen. Press "RETURN" to continue with the next page of the flowchart.

Tektronix
COMMITTED TO EXCELLENCE

DESKTOP COMPUTER APPLICATIONS LIBRARY PROGRAM

TITLE	ABSTRACT NUMBER
Friendly Graphing	TEKniques Vol. 6 No. 4 T1 Program 17
ORIGINAL DATE / REVISION DATE	EQUIPMENT AND OPTIONS REQUIRED
	4052/54 - 56K Memory
AUTHOR Paul Howard — Tektronix, Inc. Wilsonville, Or	PERIPHERALS Optional-4662/63 Plotter -4641/43 Printer

ABSTRACT

Files - 2 Program
1 ASCII Text

Statements - 1767

Requires dedicated tape.

Friendly Graphing allows a 4052/-4 user to easily create, modify, save, copy and draw graphs. Defaults, automatic tape marking and graph directory maintenance mean a user needs minimal 4052/54 experience.

Four graphs may be plotted on one page.

Graph Options

1 - 6 curves
comparative bar
stacked bar
solid line
dashed lined (6 types)
triline
phantom

52 events per graph

data from: keyboard
tape
running average
moving average
cumulative sum
least squares fit of a previous curve
difference of two previous curves
% of two previous curves
average of all previous curves
sum of all previous curves

TITLE	ABSTRACT NUMBER
Friendly Graphing	TEKniques Vol. 6 No. 4 T1 Program 17

Graph Parameters

Standard/staggered X-axis labels
Zoom on events and/or curves
Auto/Manual scaling
Grid on X, Y, both or none
Hidden Lines

Graph Labels

1 to 2 for Heading
Free one placed anywhere
X-Axis
Left and Right Y-Axes
Curves

Editing

Insert data at any point
Delete an event from all curves
Add a new curve in any sequence
Parameters
Labels

Utilities

Curves or Graphs may be saved and recalled from tape (up to 30 per tape)
Plot to screen or plotter. Option 31 (8-pen) plotter provided for.
Graph/Curve Directory automatically maintained.
Friendly Graphing files may be copied to another tape.

TITLE	ABSTRACT NUMBER
Friendly Graphing	TEKniques Vol. 6 No. 4 T1 Program 17

PRELIMINARY OPERATING INSTRUCTIONS

The three files comprising Friendly Graphing must be transferred from the TEKniques master program tape to a tape dedicated to Friendly Graphing.

This is a simple process. Using the instructions on page iii, transfer files 28, 29 and 30 on the TEKniques tape to files 1, 2 and 3 on the dedicated tape.

Or, do it manually by following these steps:

STEP 1: Insert the Friendly Graphing tape into your 4050 system.

MARK file 1 for 5120 bytes

MARK file 2 for 48128 bytes

MARK file 3 for 12800 bytes

STEP 2: Insert the TEKniques tape into your system.

FIND 28
OLD

STEP 3: Insert the Friendly Graphing tape into your system.

FIND 1
SAVE

STEP 4: Insert the TEKniques tape into your system.

FIND 29
CALL "BOLD"

STEP 5: Insert the Friendly Graphing tape into your system.

FIND 2
CALL "BSAVE"

STEP 6: KEY IN: DEL ALL

Write a small program:
Insert TEKniques tape into the system and RUN.

When prompted for the Friendly Graphing tape, insert it and press RETURN.

```
110  A$=CHR(13)
120  DIM L$(12289)
130  L$=""
140  FIN 30
150  FOR I = 1 to 357
160  INP@33:Z$
170  L$=L$&Z$
180  L$=L$&A$
190  NEXT I
200  PRI "INSERT FRIENDLY GRAPHING TAPE"
210  INPUT Z$
220  FIND 3
230  PRI@33:L$
240  CLOSE
250  END
```

Once these files have been transferred to your Friendly Graphing tape, you may transfer the programs to another tape by simply running file 1 on your Friendly Graphing tape and responding to the prompts.

TITLE	ABSTRACT NUMBER
Friendly Graphing	TEKniques Vol. 6 No. 4 T1 Program 17

TAPE STRUCTURE

The Friendly Graphing dedicated tape will contain programs on the first two files and ASCII data on the third.

The remaining files will be automatically marked. The fourth file will contain the graph/curve directory. The following files will be marked for 6400 as needed to contain graph or curve data.

EQUIPMENT REQUIRED

4052/54 Desktop Computer with Option 24 (56K Total M emory)

4631 Hard Copier (for normal graphs, one per page)

4662/63 Plotter with GPIB cable (up to four graphs/page). NOTE: This program is compatible with 4662 multi-pen Option 31 . The plotter address must be set to 1.

Optional: 4641/43 Line Printer and 4052 Option 10 interface installed in backpack slot #51 (right-hand slot).

TITLE	ABSTRACT NUMBER
Friendly Graphing	TEKniques Vol. 6 No. 4 T1 Program 17

OPERATING INSTRUCTIONS

Press Auto-Load.

Press the User-Definable Key of your choice.

KEY #1 - MENU for Auto Load Program

KEY #6 - Runs the FRIENDLY GRAPHING program.

KEY #10 - Lists the manual contained in file 3 to a line printer or the screen.

KEY #11 - Will interrupt the manual listing at any time.

KEY #20 - Will duplicate the three files comprising Friendly Graphing to another tape. Simply respond to its prompts.

FRIENDLY GRAPHING PROGRAM INSTRUCTIONS

When prompted for the date, be sure to enter the hyphens when you key in the date, i.e.

11-APR-82

Select the User-Definable Key to run the function of your choice.

TITLE	ABSTRACT NUMBER
Friendly Graphing	TEKniques Vol. 6 No. 4 T1 Program 17

FRIENDLY GRAPHG USER's MANUAL - NOTES

If a PROMPT(QUESTION) ends with a "#", then ONLY a number or a RETURN is valid.

If a RETURN is entered and NO RETURN prompt is indicated, the present function is not performed, and the user is returned to the menu.

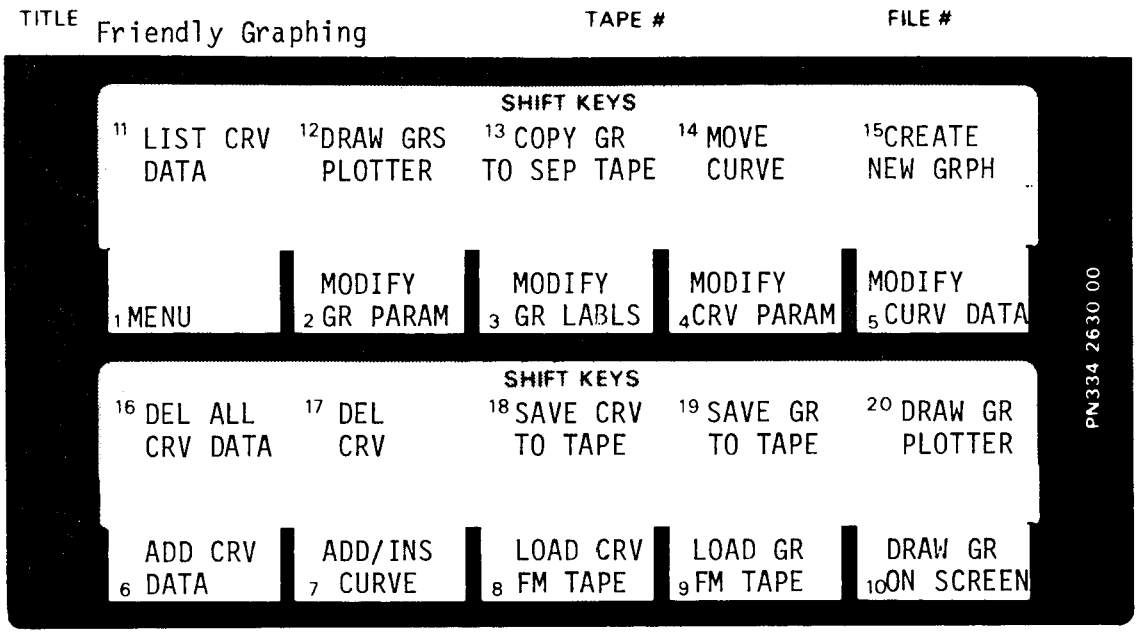
If the PROMPT ends with a "?", then ONLY a Y or N (or RETURN) is valid. Any key except Y (including RETURN) implies a NO.

The ZOOM feature allows the user to specify a portion of events and/or cruves to display on the screen or plotter.

The ZOOM feature doesn't alter the actual data, it only "zooms" in on the data of interest and thus does not display undesired data.

Normally, the ZOOM is set to the maximum # of events and curves defined. UDK #2 can set the ZOOM to any subset of events and/or curves desired. ZOOM is automatically reset by UDK #2, as well as any other time the graph data is altered.

TITLE	ABSTRACT NUMBER
Friendly Graphing	TEKniques Vol. 6 No. 4 T1 Program 17



UDK #	FUNCTION
1	List Friendly Graphing Menu
2	Set or Reset the following graph parameters: DISPLAY G# - Displays Graph number in the upper left corner. HIDDEN LINES - Lines which fall below the maximum value of the previous curves are not drawn. NOTE: The first bar style curve resets the hidden curve "horizon" to Ø. STAGGER EVEN LABELS - Staggers the X-Axis Event Labels. ZOOM EVENTS and/or CURVES - Draw only a portion of a graph using a subset of EVENTS and/or CURVES. AUTO or MANUAL SCALING - Auto scaling automatically computes the next larger Y-AXIS scale so that 5 Y-Axis labels are drawn using multiples of 1, 2, 2.5, 5.0, or 7.5. MANUAL scaling allows the user to set Y min and max, and to specify the number of Y-Axis tics. X, Y, X&Y or no GRID. - If grid selected and AUTO scaling is in effect, the grids will follow the X and Y tics. If MANUAL scaling is in effect, the user can specify grids per label or labels per grid.
3	MODIFY GRAPH LABELS - Alter Graph Titles, Curve Labels, or Event Labels.
4	MODIFY CURVE PARAMETERS - Alter a curve's name, line type, shading (if a bar curve), and data source(s).

TITLE	ABSTRACT NUMBER
Friendly Graphing	TEKniques Vol. 6 No. 4 T1 Program 17

UDK #	FUNCTION
5	MODIFY A CURVE's DATA - Alter any data event within a curve. NOTE: to jump between non-sequential data events (i.e., from event 10 to event 1), enter "J" followed by the event number desired (i.e., J1). To "null" a data event, enter "N".
6	ADD (insert data between existing) DATA to all curves - Add data events to all curves before the first, between the first and last, or after the last data event. Only 52 events maximum per curve are allowed.
7	ADD (insert a curve before) a CURVE - Add a new curve before, between, or after existing curves. Only 6 curves maximum per graph.
8	LOAD A CURVE FROM THE TAPE - Copy the data on a "CURVE" data file into a presently existing curve. A "CURVE" data file can only be created by first saving that curve onto the tape using UDK #18. NOTE: Once done, that curve's old data is gone!
9	LOAD A GRAPH FROM THE TAPE - Copy the graph data on a "GRAPH" tape file into memory. NOTE: Once done, any previous graph in memory is gone!
10	DRAW A GRAPH ON THE SCREEN - Display the graph in memory to the screen for hard copying. NOTE: Only one graph can be displayed on the screen at any time.
11	LIST ALL CURVE DATA ON THE SCREEN/PLOTTER @1:/PRINTER@51 . List the present ZOOMed range of graph data in table form on the screen, plotter, or printer.
12	DRAW 2, 3, or 4 GRAPHS on ONE PLOTTER PAGE - If 2 graphs are drawn, they will be side by side. The graphs are brought in from tape. If 3 graphs, the first will be in the upper left corner, the second in the lower left corner, and the third centered on the right.
13	COPY a GRAPH to another FRIENDLY GRAPHING TAPE - Load a graph from the present Friendly Graphing tape and save it onto another Friendly Graphing tape. Graphs cannot be copied onto non-Friendly Graphing tapes. Copying does not alter the original or duplicate graph.
14	MOVE A CURVE - Any curve can be moved after (to a higher curve number) any other curve, but no calculated curve (type 4 or higher) can be moved before its data source(s). This restriction assures that a curve which is calculated from a lower curve which is also calculated from a lower curve ends up with realistic data, since the curves are calculated from 1 to 6.

TITLE	ABSTRACT NUMBER
Friendly Graphing	TEKniques Vol. 6 No. 4 T1 Program 17

UDK # FUNCTION

15 CREATE A NEW GRAPH (& DELETE PRESENT GRAPH FROM MEMORY)

The program will prompt for graph and axes labels. All are optional.

The 1ST and 2ND graph labels are centered at the top. The FREE title can be positioned anywhere within the graph, where the upper left corner is 1,1 and other positions are located by the number of spaces and retruns from the upper left corner to that desired position.

The X-Axis, Left Y-Axis and Right Y-Axis titles are on the bottom, vertical left, and vertical right, respectively.

The number of data events are on the X-axis. A maximum of 52.

EVENT LABEL - Auto Sequence by Numbers
Weeks
Periods
Months

or

User Input

NUMBER OF CURVES - Maximum of 6. For each curve you will be prompted for title, line type (comparative, comparative bar, stacked bar, solid line, dashed line--6 patterns--triline--a small triangle with a dot in its center for data point plots--or phantom--usually used when only the results of a calculation based on that curve, displayed by another curve, is desired, and where the original curve would only clutter up the graph.

DATA SOURCE - KEYBOARD = user input.
CURVE tape file = will prompt for the file number.
CONSTANT = All of a curve's data is set to one value.
RUNNING AVERAGE = calculates each data point based on the sum of all previous data points divided by the number of previous data points.
MOVING AVERAGE = calculates similar to the RUNNING AVERAGE, except only the 'n' most recent data points are used for the calculation, where 'n' is the "window."
CUMULATIVE SUM = adds all of the previous data points together to get each calculated point
LEAST SQUARES FIT = fits a straight line through the "average" of the previous curve's data.
DIFFERENCE = different of two previous curves
PERCENTAGE = percentage of two previous curves
AVERAGE = of all previous curves
SUM = of all previous curves

A "previous curve" is any curve which has a lower curve number than the present curve i.e., curve 4 could be the PERCENTAGE of curves 2 and 3, but could not be the percentage of curves 2 and 5.

TITLE	ABSTRACT NUMBER
Friendly Graphing	TEKniques Vol. 6 No. 4 T1 Program 17

UDK #	FUNCTION
16	DELETE DATA FROM ALL CURVES - Remove (for good!) any data event from all curves.
17	DELETE A CURVE - Remove (for good!) any curve from the present graph.
18	SAVE A CURVE ONTO TAPE - Copy a curve's data onto a tape file. The program will prompt for the GR#. This number specifies which file following the Directory file that data will be saved into. The only way to copy a curve's data between graphs is to save that curve's data onto a "CURVE" data file, load (or create) the graph in which that data will be copied to, and then use UDK #8 to load that curve over one of the graph's existing curves. NOTE: When curve data is saved to a file, that file's old data is gone.
19	SAVE A GRAPH ONTO TAPE - Copy the graph currently in memory onto a tape file. The program will prompt for the GR# which is the file # following the Directory file onto which that data will be saved. Once done, any previous data on that file is gone! NOTE: The files for curve/graph data are all marked for 6400 bytes which will hold the largest graph possible. Therefore, old curve or graph files no longer needed may be used to store new data. If the file has already been marked, the program will not remark it.
20	DRAW A GRAPH ON THE PLOTTER @1- Draw the present graph in memory onto the plotter. To draw more than one graph on one plotter page, use UDK #12.

TITLE	ABSTRACT NUMBER
Friendly Graphing	TEKniques Vol. 6 No. 4 T1 Program 17

UDK # FUNCTION

15 CREATE A NEW GRAPH (& DELETE PRESENT GRAPH FROM MEMORY)

The program will prompt for graph and axes labels. All are optional.

The 1ST and 2ND graph labels are centered at the top. The FREE title can be positioned anywhere within the graph, where the upper left corner is 1,1 and other positions are located by the number of spaces and retruns from the upper left corner to that desired position.

The X-Axis, Left Y-Axis and Right Y-Axis titles are on the bottom, vertical left, and vertical right, respectively.

The number of data events are on the X-axis. A maximum of 52.

EVENT LABEL - Auto Sequence by Numbers
Weeks
Periods
Months

or

User Input

NUMBER OF CURVES - Maximum of 6. For each curve you will be prompted for title, line type (comparative, comparative bar, stacked bar, solid line, dashed line--6 patterns--triline--a small triangle with a dot in its center for data point plots--or phantom--usually used when only the results of a calculation based on that curve, displayed by another curve, is desired, and where the original curve would only clutter up the graph.

DATA SOURCE - KEYBOARD = user input.
CURVE tape file = will prompt for the file number.
CONSTANT = All of a curve's data is set to one value.
RUNNING AVERAGE = calculates each data point based on the sum of all previous data points divided by the number of previous data points.
MOVING AVERAGE = calculates similar to the RUNNING AVERAGE, except only the 'n' most recent data points are used for the calculation, where 'n' is the "window."
CUMULATIVE SUM = adds all of the previous data points together to get each calculated point
LEAST SQUARES FIT = fits a straight line through the "average" of the previous curve's data.
DIFFERENCE = different of two previous curves
PERCENTAGE = percentage of two previous curves
AVERAGE = of all previous curves
SUM = of all previous curves

A "previous curve" is any curve which has a lower curve number than the present curve i.e., curve 4 could be the PERCENTAGE of curves 2 and 3, but could not be the percentage of curves 2 and 5.

TITLE	ABSTRACT NUMBER
Friendly Graphing	TEKniques Vol. 6 No. 4 T1 Program 17

UDK #	FUNCTION
16	DELETE DATA FROM ALL CURVES - Remove (for good!) any data event from all curves.
17	DELETE A CURVE - Remove (for good!) any curve from the present graph.
18	SAVE A CURVE ONTO TAPE - Copy a curve's data onto a tape file. The program will prompt for the GR#. This number specifies which file following the Directory file that data will be saved into. The only way to copy a curve's data between graphs is to save that curve's data onto a "CURVE" data file, load (or create) the graph in which that data will be copied to, and then use UDK #8 to load that curve over one of the graph's existing curves. NOTE: When curve data is saved to a file, that file's old data is gone.
19	SAVE A GRAPH ONTO TAPE - Copy the graph currently in memory onto a tape file. The program will prompt for the GR# which is the file # following the Directory file onto which that data will be saved. Once done, any previous data on that file is gone! NOTE: The files for curve/graph data are all marked for 6400 bytes which will hold the largest graph possible. Therefore, old curve or graph files no longer needed may be used to store new data. If the file has already been marked, the program will not remark it.
20	DRAW A GRAPH ON THE PLOTTER @1- Draw the present graph in memory onto the plotter. To draw more than one graph on one plotter page, use UDK #12.