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	ABSTRACT NUMBER
4052A/4054A Assembler	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 lines followed by blank line

          GLOBAL=0A5D   ; absolute address
LBL:      .BYTE 0A      ; leading zero required to make constant instead of label
MINUS_ONE: .WORD -1     ; underscores allowed

          INS           ; implied addressing
          CLR A         ; can also be written CLRA without intervening space
          LDA A GLOBAL  ; extended addressing
NEXT:     LDAX minus_one,I    ; upper/lower case the same
          FPSM 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
000A:  32                               pul a
000B:  4A                               dec a
000C:  27 03                            beq     next
000F:  7E 0007R                         jmp     top
0012:
0012:  B6 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
```

```
0011:
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: 8C 04 04 A0 00 00 00 00 00    10
DBC3: 11                    SEMITG
```

TITLE

4052A/4054A Assembler

ABSTRACT NUMBER

TEKniques Vol. 6 No. 4
Program 1

```
DBC4:  08 DF 66 00 00     Rel #
DBC9:  08 DF C4 00 00     Code #
DBCE:  17 C0 C9 AB 6E     "EXEC"  CALLTG
DBD3:  18 93 5A   EOLTG
DBD6:  03 40 94   GOSTG
DBD9:  18 93 5A   EOLTG
DBDC:  04 40 7E DF E6 0C 04 01 80 00 00 00 00 00 00 0C 04 04 A0 00 00 00 00
00   FORTG
DBF3:  18 93 5A   EOLTG
DBF6:  22 40 73   DOTG
DBF9:  18 5A 37   EOLTG
DBFC:  19    EOSTG
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

4052A/4054A Assembler

ABSTRACT NUMBER

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

TITLE

4052A/4054A Assembler

ABSTRACT NUMBER

TEKniques Vol. 6 No. 4
Program 1

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 bit 15 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
.	.	C	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 "+".

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

4052A/4054A Assembler

ABSTRACT NUMBER
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	

TITLE

4052A/4054A Assembler

ABSTRACT NUMBER

TEKniques Vol. 6 No. 4 T1
Program 1

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

Cperation:

[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

4052A/4054A Assembler

ABSTRACT NUMBER

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

4052A/4054A Assembler

ABSTRACT NUMBER

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	ABSTRACT NUMBER
4052A/4054A Assembler	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

4052A/4054A Assembler

ABSTRACT NUMBER

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	ABSTRACT NUMBER
4052A/4054A Assembler	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	ABSTRACT NUMBER
4052A/4054A Assembler	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	ABSTRACT NUMBER
4052A/4054A Assembler	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

4052A/4054A Assembler

ABSTRACT NUMBER

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

4052A/4054A Assembler

ABSTRACT NUMBER

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).

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

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.

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
401B	Bfralc	; Allocate I/O buffers using A.PRIM
401E	Kbgin	; ?
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
404B	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	Eoloff	; Release EOI control line on GPIB
4078	Eolon	; Assert EOI control line on GPIB
407B	Fixi	; Convert a floating point number to an integer
407E	Fixnum	; 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

4052A/4054A Assembler

ABSTRACT NUMBER

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 RO 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 OPTBL info about a BASIC token
40DE    Initmt  ; Switch shared PIAs to magtape
40E1    Inpcti  ; 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    Kbdgot  ; Fetch a key from the type-ahead queue
4108    Keyin   ; Input key
410B    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
4136    Modt88  ; Special entry to modify magtape header
4138    Mtafin  ; Locate and open a new magtape file
```

TITLE

4052A/4054A Assembler

ABSTRACT NUMBER

TEKniques Vol. 6 No. 4 T1
Program 1

```
413E    Mtdswp  ; 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    Mtrdfr  ; Controlled read of a magtape buffer
4165    Mtread  ; Read a record from the magtape
4168    Mtsset  ; Set the magtape format status register
4168    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    Nfrd    ; 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    Prlstg  ; Print a string
4198    Prval   ; 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	ABSTRACT NUMBER
4052A/4054A Assembler	TEKniques Vol. 6 No. 4 T1 Program 1

```
4100    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
41FB    Testcs  ; Conditional upcase depending on SET CASE/NOCASE
41FE    Tmims   ; Wait for no tape motion
4201    Transl  ; Translator mainline
4204    Truth   ; 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    Wrival  ; 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 PRINT32,24:
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 object
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    Vis18x  ; Set object's visibility
```

TITLE

4052A/4054A Assembler

ABSTRACT NUMBER

TEKniques Vol. 6 No. 4 T1
Program 1

```
42F0    Vipcmd  ; wait until NOT(VIP ! SHAKE) then data and cmd
4300    V54ect  ; Basic MOVE or DRAW
4303    SYMNCH  ; SYMTAB for n-char variable
4306    SYMLBL  ; SYMTAB for SUB names
4309    Intext  ; 814 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 Rompacks 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

TECO

ABSTRACT NUMBER
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¤EW10¤¤
```

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.

pg 98 is blank