

Inter-Office Memorandum

To Alto Users

Date November 19, 1975

From Dan Ingalls

Location Palo Alto

Subject Bit BLT

Organization LRG

XEROX

Filed on: <INGALLS>BITBLT.BRAVO

This is a final description of the BIT BLT routines which grew out of discussions among Larry Tesler, Bob Sproull, Diana Merry and me. They are available as a BCPL driver with machine code inner loop and microcode which gets called if it is loaded. These routines assume that appropriate clipping or bounds checking has already been done on input. They are available as BITBLTB.BCPL, BITBLTA.ASM, BITBLT.MU in <INGALLS>BITBLT.DM.

Definitions

A *bit map* is a region of memory defined by *bca* and *bmw*, where *bca* is the *base core address* (starting location) and *bmw* is the *bit map width* in words; the number of scan lines is irrelevant for our purposes. (If both *bmw* and *bca* are even, then the bit map may be displayed on the screen using standard Alto facilities.)

A *block* is a rectangle within a bit map. It has four corners which need not fall on word boundaries. Any given block is described by a *block descriptor* whose contents are:

Bit map's base core address (*bca*)
Bit map's width in words (*bmw*)
Block's Left *x* ("x offset")
Block's Top *y* ("y offset")
Block's **Width**
Block's **Height**

A *block* is sometimes used to designate a sequence of bits in memory, such as the 16 wide 14 high region containing the bit pattern of a font character. In this case, *bca* point to the font character, *bmw* is 1, *x* and *y* are 0, *width* is 16, and *height* is 14.

Block Operations

From BCPL, one uses the call

FillBitMap (destination, function, source, gray) //in Caravan, or

FillBitMap (lv destination, function, lv source, gray) //in Alto OS.

The *destination* is a pointer to a *block descriptor*.

The *function* is encoded as *operation + source-type*. The *operation* codes are:

| | | | |
|---|----|----------|--|
| 0 | 0 | Replace: | Destination ← Source |
| 1 | 4 | Paint: | Destination ← Source ior Destination |
| 2 | 8 | Invert: | Destination ← Source xor Destination |
| 3 | 12 | Erase: | Destination ← not Source and Destination |

The *source-types* are:

| | | | |
|-----|---|--|-------|
| 0 | 0 | A block of a bit map | Block |
| 1 | 1 | The complement of a block of a bit map | Block |
| 2 | 2 | A block as a brush with a gray | Brush |
| | | A brush emits gray where the brush is 1, and a copy of the destination (transparency) where the brush is 0 | |
| 123 | 3 | A gray | Gray |

A gray is a one-word item specifying a 4-by-4 bit pattern to be used as a source. A solid constant source is specified with a gray whose four fields are equal.

The source depends on the source type:

| | | |
|---------|-----|--|
| 0-1-0,4 | 0,4 | A pointer to a block descriptor |
| 2 | 8 | A pointer to a block descriptor and a gray |
| 3 | 12 | A gray |

For source types 0-4, the source width and height are ignored and a simple transfer between equally-sized rectangles is performed.

The routine first considers the possibility of source-destination overlap and decides in which order to transfer words. It also generates masks and counts to be used in the transfer loops. Then a lower level routine is called which jumps into microcode if the RAM is loaded; otherwise it does the work in novacode.

Timing Details

The microcode has roughly the following speed characteristics:

| | |
|---|--|
| Horizontally, along one raster line (so to speak) | |
| Store constant | 15 cycles/word |
| Move block (store) | 36 cycles/word |
| Move block (OR) | 42 cycles/word |
| Vertical loop overhead (time to change raster lines) | |
| | 25-30 cycles, depending on source/dest alignment |
| Initial setup overhead (time to get going or resume from interrupt) | |
| | approx 150 cycles |

These are all in terms of Alto minor cycles and do include all memory wait time and do not include any degradation due to competing tasks, such as the display or disk.

Interim Details

The machine code is not presently reentrant, and consequently will crash if you try to use it in separate processes. We do not intend to alter this situation since, when the microcode is used, things behave reentrantly, and there is no problem. It is not hard to do, though, if someone has a good reason for it.

The microcode is not presently interruptable, and consequently large operations will cause significant delays in interrupt service (like 1/4 second to move most of the screen). This will be relieved shortly by allowing the microcode to save its state in the ACs and emerge to the nova emulator, later to resume where it left off.

We will probably provide an efficient entry for CONVERT-like operations; your suggestions in that direction are hereby solicited.

://FILE MAINTENANCE ---- BBSCAN.SR

://April 20, 1976

://Notes and Code --- BitBlit and Scan Conversion -- D. MERRY

://The formal format for the table associated with BitBlit looks as follows:

| | | |
|-------|-----------------------|--|
| ://0 | FUNCTION | First address must be on even word boundary |
| ://1 | "GRAY" | One-word constant defined by user |
| ://2 | DESTINATION CORE BASE | |
| ://3 | DESTINATION RASTER | Width in "Nova" words of destination rectangle |
| ://4 | DESTINATION X | |
| ://5 | DESTINATION Y | |
| ://6 | WIDTH | In bits |
| ://7 | HEIGHT | In bits |
| ://10 | SOURCE CORE BASE | |
| ://11 | SOURCE RASTER | |
| ://12 | SOURCE X | |
| ://13 | SOURCE Y | |
| ://14 | SCRATCH GRAY | Four locations for building gray words for microcode |
| ://15 | SCRATCH GRAY | |
| ://16 | SCRATCH GRAY | |
| ://17 | SCRATCH GRAY | |

://The format of the "strike" font is the simple case of Cypress Glyphs as described //in the FONT FORMAT memo of January 29, 1976. <MERRY>STRIKEFORMAT.BRAVO

| | | |
|-------|---------------|---|
| ://0 | FORMAT | If high order bit on, it's a "strike" format font, otherwise it's in .AL format. For the simple case only the high order bit can be on in the strike format. |
| ://1 | MAXIMUM WIDTH | Width of the widest character |
| ://2 | ASCENT | |
| ://3 | DESCENT | Ascent + Descent = Segheight (Fontheight) |
| ://4 | XOFFSET | Negative for kerned font, 0 normally. |
| ://5 | MIN | Smallest legal Ascii in this font. Characters less than Min not used for some sort of control by the user will be displayed as illegal character |
| ://6 | MAX | Largest legal Ascii in this font. Max + 1 will probably be the "Shazam" character which will be displayed whenever a character greater than Max is requested. |
| ://7 | NSEGS | Must be 1 in the simple case |
| ://10 | SEGMENT WIDTH | Total width of font in bits. This value + 15 and divided by 16 yields the raster value for BitBlit. (simple case) |

```

: //11 PINCH TOP 0 in simple case.
: //12 PINCH BOTTOM 0 in simple case.
: //13 CHARPOINTERS MIN thru MAX+2, indexed by Ascii.
: // Value is left x of selected glyph.
: // MAX + 1 is the "illegal" character
: // and MAX + 2 the right x of
: // "illegal" character.
: // SEGMENT ((Segwidth + 15)/16) * segheight

: //The following is code for scan conversion of characters using BitBlit and
: //the "strike" font format. The table passed to BitBlit will look like this:

: //0 FUNCTION May vary with each character -- set
: // by user. This location must be on
: // even word boundary.
: //1 "GRAY" Only relevant if "painting"
: // characters, i.e. Function > 7.
: //2 DESTINATION CORE BASE First word address of memory used
: // for Alto display
: //3 DESTINATION RASTER Width of Display in "Nova" words
: //4 DESTINATION X Must be set by user for every
: // "new line", updated by routine on
: // each character.
: //5 DESTINATION Y Set by user -- will typically
: // have "fontheight" added to it for
: // a new line
: //6 WIDTH Computed -- Ascii+1's x - Ascii's x.
: //7 HEIGHT Fontheight
: //10 SOURCE CORE BASE Pointer to the bits of the font --
: // created by some setup routine
: //11 SOURCE RASTER (Segmentpointer + 15) / 16
: //12 SOURCE X Value in location Charpointers + Ascii
: //13 SOURCE Y 0
: //14 SCRATCH GRAY Place for Scratch Grays in case
: // painting font.
: //15 SCRATCH GRAY
: //16 SCRATCH GRAY
: //17 SCRATCH GRAY
: //20 SAVE AC1 Place to save AC's.
: //21 SAVE AC2
: //22 SAVE AC3
: //23 BITBLT FONT Place to keep pointer to font whilst putting
: // out a string of characters.
: //24 CHARACTER Place to hold character code, to
: // facilitate exception checking.
: //25 CHANGE Zero means there has been no change
: // in the font, the function, or the "gray"
: // since the last time a string was scan
: // converted. > 0 means to set up BBSTABLE
: // according to information provided in
: // SCANTABLE passed in AC1.
: // Put RETURN
: // WIDTH SUBROUTINE Pointer to subroutine which when called
: // will return width of character passed in
: // AC0 -- expects pointer to BBSTABLE
: // in AC2

```

```

//      HEIGHT SUBROUTINE      Pointer to subroutine which when called
//                               will return height of font --
//                               expects pointer to BBSTABLE in AC2
//26     SCAN SUBROUTINE      Pointer to subroutine which will make call
//                               on BitBlt -- depends on the kind of font
//                               being used.
//27     SAVE GRAY            Only necessary if going to
//      (TEMP1)                have "gray" characters
//30     GRAY COUNT          Ditto
//      (TEMP2)
//31     TRAIL CHARS         Needed only for justification in Smalltalk.
//                               Signals whether there have been non-space
//                               characters since the last space -- helps to
//                               deal with multiple spaces.
//      CROSSLEFT           Used in Smalltalk for window clipping
//      CROSSRIGHT          Ditto
//      RIGHTMARGIN         Ditto
//      MEASURE             Smalltalk switch -- so that PUTCHARS code
//                               can be used both for measuring and scan
//                               converting
//      TEMP3              Needed only if .AL fonts expected
//      LASTVISIBLE         For returning last visible character
//                               when clipping occurs

```

```

//The format of the SCANTABLE passed in AC1 whenever there is a call for
//putting a string of characters is or setting up the BBSTABLE in preparation
//for a call for displaying a string of characters is as follows:

```

```

//0     FUNCTION            OR, STORE, etc.
//1     GRAY               Meaningful only if FUNCTION >7.
//2     FONT              Pointer to first word of font.
//3     DESTINATION X     Beginning X of first character of STRING.
//4     DESTINATION Y     "Top" Y of first character of STRING.
//5     STRING TABLE     Pointer to table with following format:
//                               STRING POINTER
//                               BEGINNING CHARACTER (BYTE)
//                               LAST CHARACTER

```

```

//The following code will probably eventually check to see if the font has
//been set up and call setup code if necessary. In any event it assumes
//the following contain appropriate values:

```

```

//FUNCTION                Usually "or" or "store"
//"GRAY"                 Usually not relevant
//DESTINATION CORE BASE   Usually upper left hand corner display word ad
//DESTINATION RASTER     "Display width"
//DESTINATION X          Set by user for first character in line --
//                          updated by this routine for subsequent characters
//                          in line
//DESTINATION Y          Set by user line by line
//HEIGHT                 Fontheight -- set once per font
//SOURCE CORE BASE       First word ad of font bits -- set once per font
//SOURCE RASTER          Width of font bits in words -- set once per font
//SOURCE Y              Always 0 in the simple case
//

```

```

//The Ascii value is received in AC0 and a pointer to the BitBlt table in AC1.

```

```

.TITL STRIKESCAN

```

```

.GETNOLIST "SMALL.OPS"
.GETNOLIST "SMALL.SYMS"
.GETNOLIST "SMDISP.SYMS"

```

```

;/.BEXTZ      .SMF                --DECLARED IN SMALL.SYMS
;/.BEXT       SETSCAN, PUTCHARS, DISPAD, DISPWD,DOJST
;              ;//--DECLARED IN SMDISP.SYMS

```

```

.SREL
SETSCAN:     SETSCANC
PUTCHARS:    PUTCHARSC
DISPAD:      0
DISPWD:      0

```

```

EXCEPT:    EXCEPTC           ;//LOCAL SRELS
CLIP:        CLIPC
STRIKESCAN:  STRIKESCANC
ALSCAN:      ALSCANC
DOJST:       DOJSTC

```

```
.NREL
```

```

C7:          7
C13:         13
.DSPAD:      DISPAD
.DSPWD:      DISPWD

```

```
BITBLT      =          60400
```

```
;//OFFSETS DEFINED IN SMDISP.SYMS
```

```

;//FUNC          =          0          ;//OFFSETS INTO BBSTABLE
;//GRAY          =          1
;//DBASE         =          2
;//DRAST         =          3
;//DESTX         =          4
;//DESTY         =          5
;//WIDTH         =          6
;//HEIGHT        =          7
;//SBASE         =         10
;//SRAST         =         11
;//SRCX          =         12
;//SRCY          =         13
;//GRAY1         =         14
;//GRAY2         =         15
;//GRAY3         =         16
;//GRAY4         =         17
;//SAV1          =         20
;//SAV2          =         21
;//SAV3          =         22
;//BBFONT =      =         23
;//CHAR          =         24
;//CHANGE =      =         25
;//WIDTHSUBR    =         26
;//HEIGHTSUBR   =         27
;//SCANSUBR =    =         30
;//SAVGRAY =     =         31
;//GRAYCNT =     =         32
;//TEMP1        =         31          ;//SHARED WITH 'GRAY' LOCATIONS --
;//TEMP2        =         32          ;//USED MOSTLY FOR SAVING RETURNS
;//              ;//IN SMALLTALK UTILITY SUBROUTINES
;//TRLCHR =      =         33          ;//USED WITH SMALLTALK JUSTIFICATION
;//CROSSLEFT    =         34          ;//FOR SMALLTALK WINDOW CLIPPING
;//CROSSRIGHT   =         35
;//RIGHTMARGIN  =         36
;//MEASURE =     =         37          ;//SMALLTALK SWITCH -- SO PUTCHARS
;//              ;//CAN BE USED BOTH FOR MEASURING
;//              ;//AND SCANNING
;//TEMP3        =         40          ;//NEEDED ONLY IF .AL FONTS
;//              ;//MIGHT BE USED
;//LASTVISIBLE  =         41          ;//LASTVISIBLE CHARACTER FOR CLIPPING
;//
;//;//FUNC =      =          0          ;//OFFSETS INTO SCANTABLE
;//;//GRAY =      =          1
;//FONT         =          2
;//DX           =          3
;//DY           =          4
;//STRINGTAB    =          5
;//
;//FORMAT =      =          0          ;//OFFSETS INTO FONT
;//MAXWIDTH     =          1
;//ASCENT       =          2
;//DESCENT      =          3
;//XOFFSET      =          4
;//MIN          =          5

```

```

SETRTN:   LDA 0,@.STRIKESCAN
          STA 0,SCANSUBR,2      ;//PASS BACK PTR TO SCAN CONVERTING SUBR IN
                                ;//BBSTABLE
          MOV 2,3
          LDA 2,SAV2,2          ;//RESTORE AC2 -- AC3 CONTAINS
                                ;//PTR TO BBSTABLE
                                ;//WHICH CALLER WILL WANT TO SQUIRREL AWAY
SETAL:    JMP @1,2
          INC 3,3                ;//MAKE BBFONT POINT AT FIRST TABLE ENTRY
          INC 3,3
          STA 3,BBFONT,2
          MKONE 0,0
          STA 0,SRASST,2
          MKZERO 0,0
          STA 0,SRGX,2
          LDA 0,@.GETWDAL        ;//SET UP WIDTH GETTING ROUTINE
          STA 0,WIDTHSUBR,2
          LDA 0,@.GETHTAL        ;//SET UP HEIGHT GETTING ROUTINE
          STA 0,HEIGHTSUBR,2
          LDA 0,@.ALSCAN
          JMP SETRTN ;//PASS BACK PTR TO SCAN CONVERTING SUBR IN
                                ;//BBSTABLE

PUTCHARSC:
STA 3,1,2 STA 3,PUTRTN,2
          MOV 0,3
          LDA 0,CHANGE,3        ;//CHECK IF FONT, FUNCTION, OR GRAY CHANGED
                                ;//SINCE LAST USE OF THIS TABLE
          SGZ 0,0
          JMP NOCHANGE
          LDA 0,1,2              ;//IF CHANGE SAVE RETURN PTR FOR PUTCHAR
          STA 0,SAV3,3           ;//IN BBSTABLE AND CALL
                                ;//SETSCAN WITH PTR
          MOV 3,0                ;//TO BBSTABLE IN AC0 -- PTR IN SCANTABLE
          JSR SETSCANC           ;//STILL IN AC1
          LDA 0,SAV3,3           ;//RESTORE RETURN PTR --
                                ;//BOMBED BY SETSCAN
          STA 0,1,2

NOCHANGE:
          STA 2,SAV2,3           ;//SAVE PTR TO AC2 IN BBSTABLE --
                                ;//FOR BCPL
          MOV 3,2                ;//BBSTABLE IN AC2
          MOV 1,3                ;//SCANTABLE IN AC3
          LDA 1,DX,3             ;//SET UP DESTINATION X OF FIRST
                                ;//CHAR IN STRING
          STA 1,DESTX,2
          LDA 1,DY,3             ;//AND TOP Y
          STA 1,DESTY,2
          MKZERO 1,1            ;//WIDTH GOES INTO SCAN
                                ;//CONVERSION AS ZERO
          STA 1,WIDTH,2
          LDA 1,STRINGTAB,3      ;//PTR TO STRING TABLE IN AC1 FOR SMF
          LDA 0,SCANSUBR,2      ;//SCAN CONVERSION SUBR CALLED BY SMF

;//SMF (SUBSTRECTOR-MAP-FETCH) IS AN ENTRY IN A STRING PACKAGE WRITTEN BY
;//LARRY TESLER --
;//GOING IN:
;//      AC0 = PTR TO SUBROUTINE (MAP FUNCTION) TO BE CALLED
;//              WITH EACH CHARACTER
;//      AC1 = PTR TO STRING TABLE:
;//              STRING POINTER
;//              FIRST CHARACTER (BYTE PTR)
;//              LAST CHARACTER(BYTE PTR)
;//      AC2 = TRANSPARENT
;//THE 1 FOLLOWING THE JSR MEANS WE'RE LOOKING AT A STRING AND
;//PROCEEDING FROM FIRST TO LAST
;//A -1 WOULD MEAN TO PROCEED FROM LAST TO FIRST
;//
;//      EACH TIME SMF CALLS THE DESIGNATED SUBROUTINE WITH AN
;//      ASCII VALUE IN AC0 AND THE CURRENT CHARACTER (BYTE PTR)
;//      IN AC1
;//
;//A NOSKIP RETURN MEANS THAT THE STRING HAS BEEN EXHAUSTED -- EITHER
;//LAST CHARACTER REACHED OR BEYOND LENGTH OF STRING. A SKIP RETURN
;//IS CAUSED BY A SKIP RETURN FROM THE USER'S SUBROUTINE. I ASSUME THERE
;//IS A SIMILAR PACKAGE FOR PUMPING STRINGS IN BCPL

          JSR @.SMF
          I
          SKIP                    ;//NOSKIP - SUBSTRG EXHAUSTED
                                ;//FOR MEASURING----
                                ;//AC0 = CHAR CODE
                                ;//AC1 = BYTE PTR

```

```

JMP RETURN ;//SKP RTN FROM STRIKESCAN MEANS STOP PICKING
LDA 3,DSPGLBS ;//WHEN SUBSTRING EXHAUSTED AC0 + NIL
NIL 0,0 ;//AND NSPC GETS NIL TO TURN OFF JUSTIFICATION
STA 0,NSPC,3
LDA 1,SAV1,2 ;//AND AC1 GETS BYTE PTR + 1 -- WHEN SUBSTRG
INC 1,1 ;//SMF DOES NOT RETURN BYTE PTR

RETURN:
LDA 2,SAV2,2
JMP @1,2

STRIKESCAN:
STA 3,SAV3,2 ;//BBSTABLE IN AC2
LDA 3,BBFONT,2 ;//CHECK FOR 'LEGAL' ASCII
STA 0,CHAR,2 ;//EXPEDIENT -- MAY NOT BE NEEDED
STA 1,SAV1,2 ;//DITTO -- BUT REQUIRED IN SMALLTALK
;//CURRENT BYTE PTR FROM SMF HERE

LDA 1,MIN,3
SGE 0,1
JSR EXCEPTC
LDA 1,MAX,3
SLE 0,1
INC 1,0 ;//MAX + 1 IS 'ILLEGAL' CHARACTER
STA 0,CHAR,2

;//EXCEPTION CODE, E.G. FOR CR,SPACE,ETC.
;//COULD BE PUT HERE -- FOR SMALLTALK
;//EXCEPTION CHARACTERS WILL BE KNOWN BY
;//THEIR ZERO WIDTH

LDA 1,MIN,3 ;//SUB MIN FROM CHAR CODE -- SO INDEXING
SUB 1,0 ;//CORRECT

LDA 3,WIDTH,2 ;//UPDATE DESTX--WIDTH 0 FIRST TIME--SET IN
;//PUTCHARS

LDA 1,DESTX,2
ADD 1,3
STA 3,DESTX,2
LDA 3,BBFONT,2
ADD 0,3 ;//SET UP SOURCE X
LDA 1,CHARPTRS,3
STA 1,SRCX,2 ;//SET UP WIDTH

LDA 0,CHARPTRS+1,3
SUB 1,0 ;//NEXTX - THISX
STA 0,WIDTH,2
SNZ 0,0
JSR EXCEPTC
JSR CLIPC
JMP NOSCAN

MOV 2,0 ;//BBSTABLE SENT IN AC0 FOR NOW*****
ZER 1,1 ;//AND AC1 IS FLAG*****
BITBLT
MOV 0,2 ;//PUT TABLE BACK IN 2 CAUSE SMF BOMBS 3

NOSCAN:
JMP @SAV3,2

CLIPC:
MOV 3,1 ;//SUBR FOR WINDOW CLIPPING
;//SAVE RETURN -- BBSTABLE COMES IN AC2

STA 1,TEMP1,2
LDA 0,CHAR,2
LDA 1,SPACE
SNE 0,1
JMP SPCIT
ISZ TRLCHR,2 ;//INDICATE NON-SPACE CHAR -- HELP DEAL
;//WITH MULTIPLE SPACES IN JUSTIFICATION
;//SOMETIMES USED AS NIL FLAG -- ARGIIII!

SPCIT:
NOP

LDA 1,WIDTH,2 ;//CHECK DESTX VISIBLE
LDA 0,DESTX,2
LDA 3,CROSSLEFT,2 ;//CROSSLEFT SET UP BEFORE PUTCHARS CALLED
SLT 0,3 ;//IF DESTX GE CROSSLEFT THEN CHECK
JMP CHKRT1 ;//CROSSRIGHT
ADD 0,1 ;//OTHERWISE ADD IN WIDTH
MOV 1,0 ;//GET IN BOTH AC0 AND AC1
SGT 1,3 ;//IF STILL NOT CROSSLEFT
JMP RTN ;//THEN NOSKIP RTN AVOIDS BITBLT CALL
SUB 3,1 ;//IF SPANNING CROSSLEFT PRUNE SRCX
;//TO GET PARTIAL CHARACTER

LDA 3,WIDTH,2
STA 1,WIDTH,2 ;//DIFF BETWEEN CROSSLEFT AND RIGHTX IS WIDTH
SUB 1,3 ;//WIDTH - DIFF TO BE ADDED TO SRCX AND DESTX

```

```

LDA 1,SRCX,2
ADD 3,1
STA 1,SRCX,2
LDA 1,DESTX,2
ADD 3,1
STA 1,DESTX,2
JMP CHKRT2
; //AND DESTX
; //AND CHECK RIGHT BOUNDARIES
CHKRT1: LDA 3,CROSSRIGHT,2
SLE 0,3
JMP RTN
ADD 1,0
; //CHECK HERE IS INITIAL DESTX OFF RIGHT
; //NOSKIP RTN AVOIDS BITBLT CALL
; //ADD IN WIDTH FOR NEXT CHECK
CHKRT2: LDA 3,RIGHTMARGIN,2
LDA 1,CROSSRIGHT,2
SNNIL 1,1
JMP CHKEDGE
SLT 1,3
JMP CHKEDGE
SGT 0,1
JMP CHKCHAR
NIL 3,3
STA 3,CROSSRIGHT,2
; //NOW CHECK FOR SPANNING CROSSRIGHT
; //IF CROSSRIGHT NIL-- WE'VE ALREADY CROSSED IT
; //SO CHECK RIGHTMARGIN -- SO MEASURING WORKS
; //IF CROSSRIGHT LESS THAN RIGHTMARGIN
; //IF EQUAL CHECK SPANNING OF RIGHTMARGIN
; //IF WITHIN CROSSRIGHT SCAN IN
; //OTHERWISE SET CROSSRIGHT TO NIL AND
; //PRUNE
PRUNE: LDA 3,SAV1,2 ; //PRUNED CHARACTER WILL BE LAST VISIBLE
STA 3,LASTVISIBLE,2
SUB 1,0
LDA 1,WIDTH,2
SUB 0,1
STA 1,WIDTH,2
; //MAINLY FOR SMALLTALK PURPOSES
; //IF SPANNING CROSSRIGHT OR RIGHT MARGIN
; //PRUNING WIDTH WILL GET PARTIAL CHAR
CHKCHAR: LDA 0,CHAR,2
LDA 1,SPACE
LDA 3,TAB
SEQ 0,1
SNE 0,3
JMP RTN
JMP SKPRTN
; //SKPRTN WILL CALL BITBLT
CHKEDGE: MOV 3,1
LDA 3,CROSSRIGHT,2
SNNIL 3,3
JMP RTN
SGT 0,1
JMP CHKCHAR
NIL 3,3
STA 3,TRLCHR,2
JMP PRUNE
; //IF ALREADY CROSSED RIGHT DON'T SCAN
; //IF SPANNING RIGHT MARGIN THEN PRUNE
; //NIL TRAIL CHAR COUNT AS FLAG FOR STOPPING
; //AT MEASURING TIME
; //AND GO PRUNE
SKPRTN: ; //AC0 PRESERVED HERE FOR MSE FINDING ROUTINES
; //*****
; //FIRST SEE IF WE'RE MEASURING -- DIFFERENT RTN
LDA 1,MEASURE,2
SZE 1,1
JMP MRTN
LDA 3,TEMP1,2
JMP 1,3
; //SKPRTN MEAN SCAN IT IN
RTN: ; //AC0 PRESERVED HERE FOR MSE FINDING ROUTINES
; //*****
; //FIRST SEE IF WE'RE MEASURING -- DIFFERENT RTN
LDA 1,MEASURE,2
SZE 1,1
JMP MRTN
JMP @TEMP1,2
; //NOSKP MEANS NO SCAN
; //EXCEPTION CODE -- HANDLES CR, TAB, AND SPACE
EXCEPTC: MOV 3,1
STA 1,TEMP1,2
LDA 3,DSPGLBS
LDA 0,CHAR,2
LDA 1,SPACE
SNE 0,1
JMP DOSPACE
LDA 1,TAB
SNE 0,1
JMP DOTAB
LDA 1,CR
SNE 0,1
JMP DOCR
ZER 0,0
; //BBSTABLE COMES IN AC2
; //FOR SMALLTALK

```

```

STA 0,WIDTH,2
LDA 0,C257                ;//RECALL SCAN SUBR
                          ;//WITH ILLEGAL CHAR
LDA 3,SCANSUBR,2
JMP 1,3                    ;//+1 TO KEEP SCANSUBR'S RETURN CORRECT

DOSPACE:
LDA 0,MEASURE,2          ;//SEE IF IN MEASURE MODE
SGZ 0,0
JMP MSPACE
LDA 0,THISLINE,3        ;//IF ACTUALLY SCAN CONVERTING THEN
SGZ 0,0                  ;//SEE IF JUSTIFYING
JMP NOJST

DOJSTC:
DSZ CNT1,3               ;//ENTRY USED BY SMALLTALK FNDMS ROUTINE
                          ;//IF SO, SEE IF WE'VE
                          ;//COUNTED DOWN COUNTERS
JMP JUSTIT
LDA 1,CNT2,3
SGZ 1,1                  ;//CNT2 ALREADY ZERO OR COUNTED DOWN?
JMP LASTJST ;//IF SO ONE MORE TIME AND STOP
STA 1,CNT1,3
ISZ LEAD,3               ;//UPDATE LEAD SO TABS WORK RIGHT
ISZ NWID,3                ;//NWID+1 NEW JUST SPACE
MKZERO 0,0
STA 0,CNT2,3             ;//ZERO CNT2
JMP JUSTIT

LASTJST:
STA 1,THISLINE,3        ;//SHUT OFF JUSTIFICATION FOR NEXT TIME
                          ;//AND UPDATEx
JUSTIT:
LDA 0,LEAD,3             ;//BUMP TAB DELTA -- LEAD SET IN LNOUT.SR
LDA 1,DELTA,3
ADD 0,1
STA 1,DELTA,3
LDA 0,NWID,3

UPDATEWIDTH:
STA 0,WIDTH,2
LDA 1,DESTX,2
ADD 0,1
STA 1,SPCX,3            ;//SAVE PROPER RIGHTX
                          ;//FOR JUSTIFICATION
JMP RTN

NOJST:
LDA 0,FSPACE,3
JMP UPDATEWIDTH

CR:                        15
TAB:                       11
SPACE:                      40

.TABWD:
DOTAB:
TABWD                        ;//IN DSPUTILS.SR
JSR .TABWD
JMP UPDATEWIDTH           ;//RETURNS TABWIDTH IN AC0

DOCR:
STA 0,JSTCR,3            ;//SAVE FOR CHECKING IN JUSTIFICATION
LDA 1,SAV1,2 ;//GET CURRENT BYTE PTR INTO AC1
LDA 3,SAV3,2
JMP 1,3                  ;//POP OUT OF SMF IN PUTCHARS

MSPACE:
LDA 0,TRLCHR,2          ;//SPACE EXCEPTION CODE WHEN MEASURING
SGZ 0,0                  ;//FIX MULTIPLE SPACES IF NECESSARY
JMP MULSPC
ZER 0,0
STA 0,TRLCHR,2
STA 0,NSPC2,3

MULSPC:
LDA 1,SAV1,2
STA 1,LSTSP,3           ;//SAVE PTR TO THIS SPACE
ISZ NSPC,3              ;//BUMP SPACE COUNTERS
ISZ NSPC2,3
JMP NOJST               ;//IF NOT GO GET MORE CHARS

MRTN:
LDA 0,TRLCHR,2          ;//NIL TRLCHR TELLS US PAST RIGHTMARGIN
SNIL 0,0
JMP @TEMP1,2            ;//IF NOT GET MORE CHARS
LDA 3,.DSPGLBS
LDA 0,CHAR,2           ;//GET LAST CHAR INTO AC0 FOR PSTRG
LDA 1,LSTSP,3
SNNIL 1,1              ;//NIL LSTSP MEANS NO SPACES IN LINE
JMP NOSPC

```

```

LDA 1,SPCX,3           ;//MAKE LAST DEXTX
                       ;//CORRECT FOR JUSTIFICATION
STA 1,DESTX,2
LDA 1,LSTSP,3
INC 1,1                ;//SEND BYTE PTR BACK IN AC1 AND CHAR CODE IN
                       ;//AC0 -- EXPECTED IN PSTRG.SR
MOUT:                  LDA 3,SAV3,2           ;//SKPRTN OUT OF SMF IN PUTCHARS
                       JMP 1,3
NOSPC:                 LDA 1,SAV1,2           ;//NO SPACES IN LINE MEANS
                       ;//SAV1 HAS CORRECT
                       JMP MOUT             ;//BYTE PTR FOR PSTRG.SR

LBYTEMK: 177400
RBYTEMK: 377
C20:      20
C257:     401
.CLIP:    CLIP

ALSCANC:              STA 3,SAV3,2           ;//BBSTABLE COMES IN AC2
                       STA 1,SAV1,2 ;//SAVE SMF'S BYTE PTR
EXTENTION:           STA 0,CHAR,2           ;//SAVE CHAR CODE FOR EXCEPTION CHECKING
                       LDA 1,WIDTH,2
                       LDA 3,DESTX,2
                       ADD 1,3
                       STA 3,DESTX,2       ;//UPDATE X
                       LDA 3,BBFONT,2     ;//NOW GET FONT INTO AC2
                       ADD 0,3
                       LDA 0,0,3
                       ADD 0,3
                       LDA 0,0,3         ;//GET WIDTH WORD
                       STA 0,TEMP2,2     ;//SAVE WIDTH WORD -- IN CASE OF EXTENTION
                       LDA 1,C20
                       MOVZR 0,0 SNC
                       MOV 1,0
                       STA 0,WIDTH,2
                       LDA 0,1,3
                       LDA 1,LBYTEMK
                       ANDS 0,1
                       LDA 0,DESTY,2
                       STA 0,TEMP3,2     ;//SAVE DESTY FOR HIGHER ROUTINES
                       ADD 1,0
                       STA 0,DESTY,2
                       LDA 0,1,3         ;//GET HEIGHT/DISPLACEMENT WORD BACK
                       LDA 1,RBYTEMK
                       AND 0,1           ;//MASK OFF HEIGHT
                       STA 1,HEIGHT,2
                       SUB 1,3
                       STA 3,SBASE,2
                       LDA 0,CHAR,2
                       LDA 1,SPACE
                       SNE 0,1
                       JMP EXCEPTION
                       LDA 1,TAB
                       SNE 0,1
                       JMP EXCEPTION
                       LDA 1,CR
                       SNE 0,1
EXCEPTION:           JSR II .EXCEPT
                       JSR II .CLIP
                       JMP NOSCANAL
                       MOV 2,0
                       ZER 1,1
                       BITBLT
                       MOV 0,2
NOSCANAL:           LDA 0,TEMP3,2         ;//FIX DESTY
                       STA 0,DESTY,2
                       MKZERO 0,0 ;//FIX SRCX -- LEFTSIDE CLIPPING MAY ALTER
                       STA 0,SRX,2
                       LDA 0,TEMP2,2
                       MOVZR 0,0 SNC
                       JMP EXTENTION
                       JMP @SAV3,2 ;//AND GO FOR ANOTHER CHARACTER

.EXCEPT:           EXCEPT
.END

```