Inter-Office Memorandum

To 1976 Alto Users

Date

January 28,

From

Dan Ingalls & Diana Merry

Location

Palo Alto

Subject

Bit Boundary Block Transfer (Bit BLT)

Organization

n LRG

XEROX

THE "Bit BLT" package is a group of subroutines that implements various operations on display windows. Among their several uses are displaying characters, scrolling, and "painting" with brushes. The operations which can be performed using Bit BLT are not necessarily confined to uses with the Alto display, but the description below focuses on simple cases with the display in mind. More complicated possibilities are left to the reader to invent as necessity and curiousity dictate. The binary files required are available as BitBLTB.Br, BitBLTA.Br, BitBLT.Mb in <ALTO>BitBLT.Dm The sources, BitBLTB.Bcpl, BitBLTA.Asm, BitBLT.Mu are available on <ALTOSOURCE>.

Definitions

A bit map is a region of memory that describes a collection of scan-lines which have a base core address (bca) and bit map width (bmw), the later being a word value. Our discussion will assume that scan-lines run horizontally from left to right (bits are addressed from x = 0 to x = bmw*16-1). Scan-lines appear consecutively in a bit map. Thus the point (0,0) is at the upper left of the display bit map -y = 0 is the first scan-line, the next y = 1, etc. The core address of the first word of scan-line 0 is bca. The number of scan-lines is not relevant, for the purposes of Bit BLT. (Note that these conventions are similar to those for the Alto display bit map; if bca and bmw are both even, the bit map may be displayed 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 Leftx ("x offset")
Block's Topy ("y offset")
Block's Width
Block's Height

Thus the block is defined by:

Leftx $\leq x \leq$ Leftx + Width-1 Topy $\leq y \leq$ Topy + Height - 1.

It is left to the caller to insure that ranges of x and y indeed fall within the bit map.

The Bcpl routine contains a structure declaration for managing this descriptor:

structure BITRECTANGLE:

E bca bmw Leftx Topy Width Height

A block is sometimes used to designate a sequence of bits in memory, such as a 16 wide by 14 high region containing the bit pattern of a font character. In this case, bea points 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(lv destination, function, lv source, gray).

The destination is a pointer to a BITRECTANGLE structure.

The function is encoded as operation + source-type. The operation codes (2 low-order bits) are:

0	replacefunction:	Destination ← Source
1	paintfunction:	Destination ← Source ior Destination
2	invertfunction:	Destination ← Source xor Destination
3	erasefunction:	Destination ← not Source and Destination

These names and values are declares as manifests in the Bcpl routine.

The source-types (next 2 bits) are:

0	blocksource: A block	of a <i>bit map</i>
4	compblocksource:	The complement of a block of a bit map
8	brushsource: A block	as a brush with a gray
		A brush emits gray where the brush
		is 1, and a copy of the destination
		(transparency) where the brush is 0
12	constantsource.	Agrav

A gray is a one-word item of four 4-bit bytes that defines a 4-by-4 repeating bit pattern to be used as a source. A solid constant source is specified with a gray whose four fields are equal. (Note: The pattern always repeats so as to start at x and y positions in the destination that are even multiples of 4. Thus several adjoining blocks filled with the same gray pattern will not show a ragged halftone phase seam between them.)

There are several manifests declared defining a collection of useful values as constant sources:

black	=		-1
darkestgray	=		#76575
darkergray	=	٠.	#165727

darkgray	=	#125125
gray	=	#55132
lightgray	=	#36074
lightergray	=	#12050
lightestgray =	#101202	
white	=	0 -
texture1	=	#27164
texture2	=	#154033
texture3	=	#31714
texture4	=	#63
texture5	=	#177714

The source depends on the source type:

0,4	A pointer to a block descriptor
8	A pointer to a block descriptor
	and a <i>gray</i>
12	A gray

For source types 0 and 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. The check for whether or not the microcode is present is performed every time the lowerlevel routine (FillBits) is called.

Timing Details

The microcode has roughly the following speed characteristics:

```
Horizontally, along one raster line (so to speak)

Store constant

Move block (store)

Move block (OR)

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 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.

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 the microcode modification cited above will permit reentrant operation.

;//FILE MAINTENANCE ---- BBSCAN.SR ;//April 20, 1976

://Notes and Code --- BitBlt and Scan Conversion -- D. MERRY ://The formal format for the table associated with BitBlt 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 BitBlt and ;//the "strike" font format. The table passed to BitBlt will look like this:

;//the "str	ike" font format. The table passed to Bitt	Bit 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.
;// ;// ;//	WIDTH SUBROUTINE	Pointer to subroutine which when called will return width of character passed in ACO 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 (TEMP1)	Only necessary if going to have "gray" characters
;//30 ;//	GRAY COUNT (TEMP2)	Ditto
;//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:

.TITL STRIKESCAN

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

```
;//.BEXTZ
;//.BEXT
                                                                               -- DECLARED IN SMALL.SYMS
                          SETSCAN, PUTCHARS, DISPAD, DISPWD, DOJST
                                                                                                         ;//--DECLARED IN SMDISP.SYMS
.SREL
SETSCAN:
PUTCHARS:
DISPAD:
DISPWD:
                          SETSCANC
PUTCHARSC
                                                    0
EXCEPT: EXCEPTC
CLIP: CLIPC
STRIK ESCAN: STRIK ESCANC
ALSCAN: ALSCANC
DOJST: DOJSTC
                                                                               ://LOCAL SRELS
.NREL
C7:
C13:
.DSPAD:
.DSPWD:
                                                     13
DISPAD
DISPWD
BITBLT
                                                                               60400
://OFFSETS DEFINED IN SMDISP.SYMS
                                                                                                         ://OFFSETS INTO BBSTABLE
//GRAY
//DBASE
                                                     =
                                                                               1234567
                                                     =
://DRAST
                                                     =
;//DRAST
;//DESTX
;//DESTY
;//WIDTH
;//HEIGHT
;//SBASE
;//SRAST
;//SRCX
;//SRCY
                                                     =
                                                                               10
11
12
13
                                                     =
                                                     =
                                                     =
  //GRAY1
//GRAY2
                                                                                14
15
16
17
                                                     =
;//GRAY4
;//SAV1
;//SAV2
;//SAV3
;//BBFONT =
;//CHAR
;//CHANGE =
;//WIDTHSUBRE
                                                                               20
                                                                               21
22
                                                     =
23
                                                                               24
                                                     =
25
                                                                               26
27
                                                     =
 //HEIGHTSUBR
;//SCANSUBR =
;//SAVGRAY =
;//GRAYCNT =
;//TEMP1
                                                     30
31
32
                                                                              31 ://SHARED WITH 'GRAY' LOCATIONS --
32 ://USED MOSTLY FOR SAVING RETURNS
://IN SMALLTALK UTILITY SUBROUTINES
://USED WITH SMALLTALK JUSTICATION
34 ://FOR SMALLTALK WINDOW CLIPPING
35
36
36
**//SMALLTALK**
 //TEMP2
;//
://TRLCHR =
://CROSSLEFT
://CROSSRIGHT
://RIGHTMARGIN
                                                     33
                                                     =
                                                     =
                                                     <del>-</del>37
                                                                               36
;//SMALLTALK SWITCH -- SO PUTCHARS
;//CAN BE USED BOTH FOR MEASURING
;//AND SCANNING
40 ;//NEEDED ONLY IF .AL FONTS
;//MIGIT BE USED
41 ;//LASTVISIBLE CHARACTER FOR CLIPPING
 //MEASURE =
//
//TEMP3
//
//LASTVISIBLE
                                                     =
//;//FUNC
;//;//GRAY
;//FONT
                                                                               ;//OFFSETS INTO SCANTABLE
                                                     0
                                                     1
                                                                               2
3
4
5
                                                     =
//DX
;//DY
                                                     =
                                                     =
//STRINGTAB
                                                     =
                                                                               ://OFFSETS INTO FONT
//FORMAT =
//MAXWIDTH
//ASCENT =
//DESCENT =
                                                     0
 //XOFFSET
                                                     4
;//MIN
                                                                               5
```

```
;//MAX
://NSEGS
://SEGWIDTH
                                                                                                                                                                               6
7
                                                                                                                     =
                                                                                                                                                                               10
     //PINCHTOP =
     //PINCHBOT =
                                                                                                                       12
   ;//CHARPTRS
                                                                                                                                                                               13
    //BBSTABLEA:
                                                                                                                                                                             ;//ALLOCATED IN DSPGLBS.SR
                                                                                                                     .BLK 41
;//SETSCAN SETS UP AS MUCH OF BITBLT'S TABLE AS POSSIBLE.
;//IT WILL BE CALLED WHENEVER THERE IS A FONT CHANGE, A FUNCTION CHANGE,
;//OR A "GRAY" CHANGE. IT MUST BE CALLED ONCE BY THE USER IN ORDER
;// TO GET THE BBSTABLE ONTO AN EVEN WORD BOUNDARY. THEREAFTER,
;//PUTCHARS WILL CHECK THE CHANGE LOCATION IN THE BBSTABLE TO
;//DETERMINE IF IT NEED BE CALLED. THE CALL EXPECTS:
;// AC0 -- POINTER TO BLOCK OF 31 (OCTAL) WORDS FOR BITBLT TABLE
;// AC1 -- POINTER TO SCANTABLE:
;// GRAY
;// GRAY
;// GRAY
;// GRAY
;// (DX)
;// (DY)
;// (STRINGTAB)
  (STRINGTAB)

(STRINGTAB)

(Magnetic According 
  ;//FOR A STRIKE FONT, THE ROUTINE WILL ESTABLISH:
;// FUNCTION
;// GRAY
;// DESTINATION CORE BASE
                                                            DESTINATION CORE BASE
DESTINATION RASTER
                                                             HEIGHT
                                                           SOURCE CORE BASE
SOURCE RASTER
SOURCE Y
BBFONT
   ;//
;//
SCAN CONVERSION SUBROUTINE
  .GETWDSTRK:
.GETHTSTRK:
.STRIKESCAN:
.GETWDAL:
                                                                                                                      GETWDSTRK
                                                                                                                      GETHTSTRK
STRIKESCAN
                                                            GETWDAL
    .GETHTAL:
                                                            GETHTAL
                                                                                                                       ALSCAN
    .ALSCAN:
  JMASK:
                                                            7417
  C15:
NIBMASK:
                                                            17
   DOGRAY:
                                                                                                                                                                               ;//BBSTABLE PASSED IN AC2
;//MOSTLY A COPY OF DAN INGALLS CODE
;//HERE BELOW
                                                           STA 3,SAV3,2
                                                           LDA 3,DESTY,2
LDA 1,SRCY,2
SGE 3,1
JMP NOINVERT
                                                           LDA 3,JMASK
AND 0,3
SUBS 0,3
ADDS 3.0
NEG 1,1
                                                                                                                                                                               ;//INVERT NIBBLE ORDER
;//A B C D BECOMES
;// A D C B
   NOINVERT:
                                                            LDA 3,HEIGHT,2
ADD 3,1
LDA 3,C3
AND 3,1
                                                                                                                                                                              ;//SRCY - HEIGHT MOD 4
```

```
ADDZL 1,1
                                                                       ://* 4
                                                                      '/IS HOW MUCH TO ROTATE GRAY
'//SAVE IT IN BBSTABLE TEMP
'//SET UP COUNTER FOR FILLING SCRATCH WORDS
                        CYCLE 0
                        STA 0,SAVGRAY,2
                       LDA 1,C4
STA 1,GRAYCNT,2
MOV 2,3
                                                                       ;//GRAY PASSED IN ACO
;//GET BBSTABLE INTO AC3 SO COUNT DOWN IN
;//GRAYLOOP WILL WORK
GRAYLOOP:
                       LDA 0.SAVGRAY,2
CYCLE 4
STA 0.SAVGRAY,2
LDA 1.NIBMASK
AND 1.0
MOV 0.1
CYCLE 4
                                                                       ://GET HIGH ORDER NIBBLE INTO LOW ORDER
                                                                       ://SAVE FOR NEXT PASS
                                                                      ;//MASK OFF LOW ORDER NIBBLE
://SAVE IN ACI
;//MOVE NIBBLE A NIBBLE TO LEFT
;//NOW ACO + ACI MAKE A BYTE
;//MAKE HIGH ORDER BYTE
                        ADD 1,0
MOVS 0,1
                                                                       ://NOW AC0 + AC1 MAKE A WORD
://SAVE IT
                        ADD 1,0
                       STA 0,GRAY1,3
INC 3,3
DSZ GRAYCNT,2
JMP GRAYLOOP
                                                                       //BUMP PTR
                        JMP @SAV3,2
SETSCANC:
                       STA 3,1,2
MOV 0,3
                        SKEVĚŇ #3,3 SZC
                                                                       ://INSURE ON EVEN WORD BOUNDARY
                       SKEVEN #3,3 3
INC 3,3
STA 2,SAV2,3
MOV 1,3
LDA 0,FUNC,3
STA 0,FUNC,2
LDA 1,C7
SGT 0,1
                                                                      ;//SAVE AC2 -- MOSTLY FOR BCPL
;//GET BBSTABLE IN AC2
;//GET SCANTABLE IN AC3
;//SET UP FUNCTION
;//IF FUNCTION >7 DO GRAY SHUFFLE
                       JMP NOGRAY
LDA 0,GRAY,3
STA 0,GRAY,2
                                                                       ://SAVE PTR TO SCANTABLE WHILE DOING GRAY FIX ;//SAV1 HAPPENS TO BE AVAILABLE
                        STA 3, SAV1, 2
                        JSR DOGRAY
                        LDA 3,SAV1,2
                                                                       ;//RESOTRE SCANTABLE INTO AC3
NOGRAY:
                                                                      ://BEGINNING AD OF DISPLAY -- MOST LIKELY
://A GLOBAL IN SREL OR ZREL
://THAT'S THE CORE BASE FOR BB'S DESTINATION
://LIKEWISE DISPLAY WIDTH IS RASTER FOR
://BB'S DESTINATION
                        LDA 0,@.DSPAD
                       STA 0,DBASE,2
LDA 0,@.DSPWD
STA 0,DRAST,2
LDA 3,FONT,3
STA 3,BBFONT,2
MKZERO 0,0
STA 0,SRCY,2
STA 0,CHANGE,2
LDA 0,FORMAT,3
MOVZL #0,0 SNC
JMP SETAL
LDA 0,ASCENT,3
LDA 1,DESCENT,3
ADD 0,1
STA 1,HEIGHT,2
LDA 0,MIN,3
                                                                       ;//BBSTABLE NEEDS ITS OWN PTR TO THE FONT
                                                                       ://SOURCE Y IS ZERO
                                                                       ;//TURN OFF CHANGE SWITCH
;//HIGH ORDER BIT OF FIRST WORD OF FONT TELLS
;//US IF STRIKE OR .AL FORMAT. 1=STRIKE. 0=.AL
                                                                       ;//IF STRIKE COMPUTE HEIGHT OF FONT
                                                                      ;//SOURCE BASE ADDRESS FIRST ;//WORD OF GLYPHS
                        LDA 1,MAX,3
                       INC 1,1
INC 1,1
INC 1,1
                                                                       ;//MAX+2 - MIN + TOP OF XTABLE GET TO GLYPHS
                       SUB 0,1
LDA 0,C13
ADD 1,0
ADD 3,0
STA 0,SBASE,2
LDA 0,SEGWIDTH,3
LDA 1,C15
ADD 1,0
MOVZR 0,0
MOVZR 0,0
MOVZR 0,0
MOVZR 0,0
STA 0,SRAST.2
                        SUB 0,1
                                                                       ://(SEGMENT WIDTH + 15) / 16 = SOURCE RASTER
                        STA 0,SRAST,2
                       LDA 0,@.GETWDSTRK
STA 0,WIDTHSUBR,2
                                                                       ://SET UP WIDTH GETTING ROUTINE
                        LDA 0,@.GETHTSTRK
                                                                       ://SET UP HEIGHT GETTING ROUTINE
                        STA 0, HEIGHTSUBR, 2
```

```
LDA 0,@.STRIKESCAN
SETRTN:
                                                                        ;//PASS BACK PTR TO SCAN CONVERTING SUBR IN
                       STA 0,SCANSUBR,2
                                                                        //BBSTABLE
                       MOV 2,3
LDA 2,SAV2,2
                                                                       ://RESTORE AC2 -- AC3 CONTAINS
://PTR TO BBSTABLE
://WHICH CALLER WILL WANT TO SQUIRREL AWAY
                        JMP @1.2
SETAL:
                       INC 3,3
INC 3,3
STA 3,BBFONT,2
MKONE 0,0
STA 0,SRAST,2
MKZERO 0,0
STA 0,SRCX,2
LDA 0,@,GETWDAL
STA 0,WIDTHSUBR,2
LDA 0,@,GETHTAL
STA 0,IIEIGIITSUBR,2
LDA 0,@,ALSCAN
JMP SETRTN ://PASS
                                                                       ://MAKE BBFONT POINT AT FIRST TABLE ENTRY
                                                                       ;//SET UP WIDTH GETTING ROUTINE
                                                                        ://SET UP HEIGHT GETTING ROUTINE
                       JMP SETRTN ://PASS BACK PTR TO SCAN CONVERTING SUBR IN ://BBSTABLE
PUTCHARSC:
                       STA 3,1,2
MOV 0,3
LDA 0,CHANGE,3
SGZ 0,0
JMP NOCHANGE
                                                                        ://CHECK IF FONT, FUNCTION, OR GRAY CHANGED ;//SINCE LAST USE OF THIS TABLE
                                                                        ;//IF CHANGE SAVE RETURN PTR FOR PUTCHAR
;//IN BBSTABLE AND CALL
;//SETSCAN WITH PTR
;//TO BBSTABLE IN ACO -- PTR IN SCANTABLE
                       LDA 0,1,2
STA 0,SAV3,3
                       MOV 3,0
JSR SETSCANC
LDA 0,SAV3,3
                                                                        ://STILL IN ACI
://RESTORE RETURN PTR --
://BOMBED BY SETSCAN
                       STA 0,1,2
NOCHANGE:
                       STA 2, SAV2, 3
                                                                        ://SAVE PTR TO AC2 IN BBSTABLE --
                                                                       ;//SAVE FIR TO ACZ IN BBSTABLE SI
://FOR BCPL
://BBSTABLE IN AC2
://SCANTABLE IN AC3
://SET UP DESTINATION X OF FIRST
://CHAR IN STRING
                        MOV 3,2
MOV 1,3
LDA 1,DX,3
                       STA 1,DESTX,2
LDA 1,DY,3
STA 1,DESTY,2
MKZERO 1,1
                                                                       ;//AND TOP Y
                                                                       ://WIDTH GOES INTO SCAN
://CONVERSION AS ZERO
                       STA 1,WIDTH,2
LDA 1,STRINGTAB,3
LDA 0,SCANSUBR,2
                                                                        ://PTR TO STRING TABLE IN AC1 FOR SMF
://SCAN CONVERSION SUBR CALLED BY SMF
;//SMF (SUBSTRECTOR-MAP-FETCH) IS AN ENTRY IN A STRING PACKAGE WRITTEN BY ://LARRY TESLER -- ;//GOING IN:
                        ACO = PTR TO SUBROUTINE (MAP FUNCTION) TO BE CALLED WITH EACH CHARACTER
AC1 = PTR TO STRING TABLE:
                                                                       STRING POINTER
FIRST CHARACTER (BYTE PTR)
LAST CHARACTERT(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 ACO AND THE CURRENT CHARACTER (BYTE PTR) IN ACO
://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
                                                                       ;//NOSKP - SUBSTRG EXHAUSTED
;//FOR MEASURING----
;//AC0 = CHAR CODE
;//AC1 = BYTE PTR
                       ŠKIP
```

```
://SKP RTN FROM STRIKESCAN MEANS STOP PICKING
://WHEN SUBSTRING EXHAUSTED ACO + NIL
://AND NSPC GETS NIL TO TURN OFF JUSTIFICATION
                      JMP RETURN
                      LDA 3, DSPGLBS
                      NIL 0.0
                     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
STRIK ESCANC:
                     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 PIR 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
                                                                  WITHEIR 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
LDA 1,CHARPTRS,3
STA 1,SRCX,2
                                                                 ;//SET UP SOURCE X
                                                                 ;//SET UP WIDTH
                    LDA 0,CHARPTRS+1,3
SUB 1,0
STA 0,WIDTH,2
SNZ 0,0
ISR EXCEPTC
ISR CLIPC
JMP NOSCAN
                                                                 ://NEXTX - THISX
                     MOV 2,0
ZER 1,1
                                                                 ://BBSTABLE SENT IN ACO FOR NOW*******
://AND AC1 IS FLAG********
                      BITBLT
                      MOV 0,2
                                                                 ;//PUT TABLE BACK IN 2 CAUSE SMF BOMBS 3
NOSCAN:
                      JMP @SAV3,2
CLIPC:
                                                                 ://SUBR FOR WINDOW CLIPPING
://SAVE RETURN -- BBSTABLE COMES IN AC2
                      MOV 3,1
                     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 -- ARGHH!!
                      NOP
SPCIT:
                      LDA 1,WIDTH,2
                                                                  ://CHECK DESTX VISIBLE
                      LDA 0,DESTX,2
                                                                 ;//CROSSLEFT SET UP BEFORE PUTCHARS CALLED ;//IF DESTX GE CROSSLEFT THEN CHECK ;//CROSSRIGHT
                     LDA 3, CROSSLEFT, 2
SLT 0, 3
JMP CHKRT1
                                                                 ;//CKOSSRIGHT
;//OTHERWISE ADD IN WIDTH
;//GET IN BOTH ACO AND ACI
;//IF STILL NOT CROSSLEFT
;//THEN NOSKIP RTN AVOIDS BITBLT CALL
;//IF SPANNING CROSSLEFT PRUNE SRCX
                     ADD 0,1
MOV 1,0
SGT 1,3
                      JMP ŘTN
                      SUB 3.1
                                                                  //TO GET PARTIAL CHARACTER
                      LDA 3,WIDTH,2
STA 1,WIDTH,2
SUB 1,3
                                                                 ;//DIFF BETWEEN CROSSLEFT AND RIGHTX IS WIDTH ;//WIDTH - DIFF TO BE ADDED TO SRCX AND DESTX
```

```
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
                                                      ;//CHECK HERE IS INITIAL DESTX OFF RIGHT
                                                      ;//NOSKIP RTN AVOIDS BITBLT CALL
;//ADD IN WIDTH FOR NEXT CHECK
                  ADD 1,0
CHKRT2:
                  LDA 3,RIGHTMARGIN,2
LDA 1,CROSSRIGHT,2
SNNIL 1,1
                                                      ;//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
                  JMP CHKEDGE
SLT 1.3
JMP CHKEDGE
SGT 0.1
JMP CHKCHAR
                                                       ://IF WITHIN CROSSRIGHT SCAN IN
                  NIL 3,3
                                                       ://OTHERWISE SET CROSSRIGHT TO NIL AND
                  STA 3,CROSSRIGHT,2
                                                       //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
STA 1,WIDTH,2
PRUNE:
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
CHK EDGE:
                  MOV 3,1
LDA 3,CROSSRIGHT,2
SNNIL 3,3
                                                      ://IF ALREADY CROSSED RIGHT DON'T SCAN
                  JMP RTN
                  SGT 0,1
                                                      ://IF SPANNING RIGHT MARGIN THEN PRUNE
                  JMP CHKCHAR
                                                      ;//NIL TRAIL CHAR COUNT AS FLAG FOR STOPPING
;//AT MEASURING TIME
;//AND GO PRUNE
                  NIL 3,3
STA 3,TRLCHR,2
JMP PRUNE
SKPRTN:
                                                       ;//ACO PRESERVED HERE FOR MSE FINDING ROUTINES
                  LDA 1,MEASURE,2
SZE 1,1
JMP MRTN
                                                       //FIRST SEE IF WE'RE MEASURING -- DIFFERENT RTN
                  LDA 3,TEMP1,2
JMP 1,3
                                                       ;//SKPRTN MEAN SCAN IT IN
RTN:
                                                       ://AC0 PRESERVED HERE FOR MSE FINDING ROUTINES
                  LDA 1,MEASURE,2
SZE 1,1
JMP MRTN
                                                       ://FIRST SEE IF WE'RE MEASURING -- DIFFERENT RTN
                  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
                                                      ://BBSTABLE COMES IN AC2
                                                      ;//FOR SMALLTALK
                  LDA 1,TAB
SNE 0,1
JMP DOTAB
                  LDA 1,CR
SNE 0,1
                  JMP DOCR
                  ZER 0,0
```

LDA 1,SRCX,2

```
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
SZE 0,0
JMP MSPACE
LDA 0,THISLINE,3
SGZ 0,0
JMP NOJST
                                                    ://SEE IF IN MEASURE MODE
                                                    ://IF ACTUALLY SCAN CONVERTING THEN ://SEE IF JUSIFYING
                                                    ://ENTRY USED BY SMALLTALK FNDMS ROUTINE ;//IF SO, SEE IF WE'VE ;//COUNTED DOWN COUNTERS
DOJSTC:
                 DSZ CNT1,3
                 JMP JUSTIT
                 LDA 1.CNT2,3
SGZ 1.1 ;//CNT2 ALREADY ZERO OR COUNTED DOWN?
JMP LASTIST ;//IF SO ONE MORE TIME AND STOP
                 STA 1,CNT1,3
ISZ LEAD,3
ISZ NWID,3
MKZERO 0,0
STA 0,CNT2,3
JMP JUSTIT
                                                    ://UPDATE LEAD SO TABS WORK RIGHT ://NWID+1 NEW JUST SPACE
                                                    ://ZERO CNT2
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:
TAB:
                                  15
11
SPACE:
                                   40
.TABWD:
                 TABWD
                                                    ://IN DSPUTILS.SR
DOTAB:
                 JSRII .TABWD
JMP UPDATEWIDTH
                                                    ://RETURNS TABWIDTH IN ACO
                 STA 0,JSTCR,3 ://SAVE FOR CHECKING IN JUSTIFICATION LDA 1,SAV1,2;//GET CURRENT BYTE PTR INTO ACI LDA 3,SAV3,2 JMP 1,3 ://POP OUT OF SAVE IN THE
DOCR:
MSPACE:
                 LDA 0,TRLCHR,2
SGZ 0,0
JMP MULSPC
                                                    ://SPACE EXCEPTION CODE WHEN MEASURING ;//FIX MULTIPLE SPACES IF NECESSARY
                 ZER 0,0
STA 0,TRLCHR,2
                 STA 0,NSPC2,3
MULSPC:
                 LDA 1,SAV1,2
STA 1,LSTSP,3
ISZ NSPC,3
ISZ NSPC2,3
                                                    ;//SAVE PTR TO THIS SPACE ;//BUMP SPACE COUNTERS
                 JMP NOJST
                                                    ://IF NOT GO GET MORE CHARS
MRTN:
                 LDA 0,TRLCHR,2
SNIL 0,0
JMP @TEMP1,2
LDA 3,DSPGLBS
LDA 0,CHAR,2
                                                    ://NIL TRLCHR TELLS US PAST RIGHTMARGIN
                                                                     ://IF NOT GET MORE CHARS
                                                    ;//GET LAST CHAR INTO ACO FOR PSTRG
                 LDA 1,LSTSP,3
SNNIL 1,1
JMP NOSPC
                                                    ://NIL LSTSP MEANS NO SPACES IN LINE
```

```
;//MAKE LAST DEXTX
                       LDA 1,SPCX,3
                                                                       //CORRECT FOR JUSTIFICATION
                       STA 1,DESTX,2
LDA 1,LSTSP,3
INC 1,1
                                                                       ://SEND BYTE PTR BACK IN ACI AND CHAR CODE IN ://ACO -- EXPECTED IN PSTRG.SR
MOUT:
                       LDA 3,SAV3,2
JMP 1,3
                                                                       ://SKPRTN OUT OF SMF IN PUTCHARS
NOSPC:
                                                                      ://NO SPACES IN LINE MEANS
://SAV1 HAS CORRECT
://BYTE PTR FOR PSTRG.SR
                       LDA 1,SAV1,2
                       JMP MOUT
LBYTEMSK: RBYTEMSK:
                       177400
                       377
C20:
C257:
.CLIP:
                                               20
                                               401
                       CLIP
ALSCANC:
                       STA 3,SAV3,2 ://BBSTABLE COMES IN AC2 STA 1,SAV1,2 ://SAVE SMFS BYTE PTR
EXTENTION:
                        STA 0,CHAR,2
                                                                       ://SAVE CHAR CODE FOR EXCEPTION CHECKING
                       JIA O,CHAR,2
LDA 1,WIDTH,2
LDA 3,DESTX,2
ADD 1,3
STA 3,DESTX,2
LDA 3,BBFONT,2
ADD 0,3
LDA 0,0,3
ADD 0,3
LDA 0,0,3
STA 0,TEMP2,2
LDA 1,C20
MOVZR 0,0 SNC
MOV 1,0
STA 0,WIDTH,2
LDA 0,1,3
LDA 1,LBYTEMSK
ANDS 0,1
LDA 0,DESTY,2
STA 0,TEMP3,2
ADD 1,0
STA 0,DESTY,2
STA 0,TEMP3,2
ADD 1,0
STA 0,DESTY,2
LDA 0,1,3
LDA 1,RBYTEMSK
AND 0,1
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
                                                                       ://UPDATE X
://NOW GET FONT INTO AC2
                                                                       ://GET WIDTH WORD
://SAVE WIDTH WORD -- IN CASE OF EXTENTION
                                                                       ://SAVE DESTY FOR HIGHER ROUTINES
                                                                       ;//GET HEIGHT/DISPLACEMENT WORD BACK
                                                                       ://MASK OFF HEIGHT
                                                                       ://ADDR OF WIDTH - HEIGHT SHOULD BE SBASE
                       LDA 1,CR
SNE 0,1
EXCEPTION:
                       JSRII EXCEPT
JSRII 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,SRCX,2
LDA 0,TEMP2,2
MOVZR 0,0 SNC
JMP EXTENTION
JMP @SAV3 2;//AND GO FOR ANOTHER CHARACTER
                        JMP @SAV3,2;//AND GO FOR ANOTHER CHARACTER
.EXCEPT:
                                               EXCEPT
.END
```