

CCCCCCCCCCCC	00000000	8888888888	RRRRRRRRRR	TTTTTTTTTTTT	LLL
CCCCCCCCCCCC	00000000	8888888888	RRRRRRRRRR	TTTTTTTTTTTT	LLL
CCCCCCCCCCCC	00000000	8888888888	RRRRRRRRRR	TTTTTTTTTTTT	LLL
CCC	000	888	RRR	TTT	LLL
CCC	000	888	RRR	TTT	LLL
CCC	000	888	RRR	TTT	LLL
CCC	000	888	RRR	TTT	LLL
CCC	000	888	RRR	TTT	LLL
CCC	000	888	RRR	TTT	LLL
CCC	000	888	RRR	TTT	LLL
CCC	000	888	RRR	TTT	LLL
CCC	000	888	RRR	TTT	LLL
CCC	000	888	RRR	TTT	LLL
CCC	000	888	RRR	TTT	LLL
CCC	000	888	RRR	TTT	LLL
CCC	000	888	RRR	TTT	LLL
CCC	000	888	RRR	TTT	LLL
CCC	000	888	RRR	TTT	LLL
CCC	000	888	RRR	TTT	LLL
CCC	000	888	RRR	TTT	LLL
CCCCCCCCCCCC	00000000	8888888888	RRR	TTT	LLLLLLLLLLLLLLLL
CCCCCCCCCCCC	00000000	8888888888	RRR	TTT	LLLLLLLLLLLLLLLL
CCCCCCCCCCCC	00000000	8888888888	RRR	TTT	LLLLLLLLLLLLLLLL

```

CCCCCCCC 000000 BBBB8888 FFFFFFFFFE SSSSSSSS CCCCCCCC GGGGGGGG FFFFFFFFFE NN NN
CCCCCCCC 000000 000000 BBBB8888 FFFFFFFFFE SSSSSSSS CCCCCCCC GGGGGGGG FFFFFFFFFE NN NN
CC        00    00 BB      BB FFF          SS          CC        GG          FFF          NN NN
CC        00    00 BB      BB FFF          SS          CC        GG          FFF          NN NN
CC        00    00 BB      BB FFF          SS          CC        GG          FFF          NN NN
CC        00    00 BB      BB FFF          SS          CC        GG          FFF          NN NN
CC        00    00 BBBB8888 FFFFFFFFFE SSSSSS          CC        GG          FFF          NN NN
CC        00    00 BBBB8888 FFFFFFFFFE SSSSSS          CC        GG          FFF          NN NN
CC        00    00 BB      BB FFF          SS          CC        GG  GGGGGG FFF          NN NN
CC        00    00 BB      BB FFF          SS          CC        GG  GGGGGG FFF          NN NN
CC        00    00 BB      BB FFF          SS          CC        GG          FFF          NN NN
CC        00    00 BB      BB FFF          SS          CC        GG          FFF          NN NN
CC        00    00 BBBB8888 FFFFFFFFFE SSSSSSSS          CC        GGGGGG FFF          NN NN
CCCCCCCC 000000 BBBB8888 FFFFFFFFFE SSSSSSSS CCCCCCCC GGGGGG          FFFFFFFFFE NN NN
CCCCCCCC 000000 BBBB8888 FFFFFFFFFE SSSSSSSS CCCCCCCC GGGGGG          FFFFFFFFFE NN NN

```

```

LL        111111 SSSSSSSS
LL        111111 SSSSSSSS
LL        11          SS
LL        11          SS
LL        11          SS
LL        11          SS
LL        11          SSSSSS
LL        11          SSSSSS
LL        11          SS
LL        11          SS
LL        11          SS
LL        11          SS
LLLLLLLLLL 111111 SSSSSSSS
LLLLLLLLLL 111111 SSSSSSSS

```

```

1 0001 0 %TITLE 'COB$$$ESCAPE_GENERATOR - Escape sequence generator for screen mgmt'
2 0002 0 MODULE COB$$$ESCAPE_GENERATOR (
3 0003 0 IDENT = '1-003' ! File: COBESCGEN.B32 Edit: STAN1003
4 0004 0 ) =
5 0005 1 BEGIN
6 0006 1
7 0007 1 *****
8 0008 1 *
9 0009 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *
10 0010 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
11 0011 1 * ALL RIGHTS RESERVED. *
12 0012 1 *
13 0013 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
14 0014 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
15 0015 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
16 0016 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
17 0017 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
18 0018 1 * TRANSFERRED. *
19 0019 1 *
20 0020 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
21 0021 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
22 0022 1 * CORPORATION. *
23 0023 1 *
24 0024 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
25 0025 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
26 0026 1 *
27 0027 1 *
28 0028 1 *****
29 0029 1
30 0030 1
31 0031 1 ++
32 0032 1 FACILITY: General Utility Library
33 0033 1
34 0034 1 ABSTRACT:
35 0035 1
36 0036 1 This module contains routines which return a device-specific
37 0037 1 escape sequence to perform a specified function.
38 0038 1
39 0039 1 These are low level routines; the burden of validity checking
40 0040 1 is on the caller. For example, buffers are allocated by the caller,
41 0041 1 and these routines do not check for overflowing the buffers bounds.
42 0042 1 If the device is not a video terminal, no escape sequence will be
43 0043 1 generated, and the routine will return with a success status.
44 0044 1
45 0045 1 ENVIRONMENT: User mode, Shared library routines.
46 0046 1
47 0047 1 AUTHOR: P. Levesque, CREATION DATE: 7-Mar-1983
48 0048 1
49 0049 1 MODIFIED BY:
50 0050 1
51 0051 1 1-001 - Original. PLL 7-Mar-1983
52 0052 1 1-002 - Add COB$$$SET_ATTRIBUTES ONLY.
53 0053 1 Fix call to COB$$$SET_CURSOR_ABS_R4 in COB$$$SET_CURSOR_REL.
54 0054 1 Fix to COB$$$SET_CURSOR_REL. If we are at the 1st column and the
55 0055 1 previous character was a <CR>, then the terminal driver may give
56 0056 1 us a 'free' <LF> on our next operation if it is a read. To avoid
57 0057 1 the problem, just make sure <CR> is not the last thing in the
    
```

```

: 58      0058 1 !      output buffer.
: 59      0059 1 !      Rename module from SMG$ESCAPE_GENERATOR to COB$ESCAPE_GENERATOR.
: 60      0060 1 !      LGB 20-FEB-1984
: 61      0061 1 ! 1-003 - Removed informational errors. STAN 24-Jul-1984.
: 62      0062 1 ! --
: 63      0063 1 !
```

```

65      0064 1 %SBTTL 'Declarations'
66      0065 1
67      0066 1 SWITCHES:
68      0067 1
69      0068 1
70      0069 1
71      0070 1 LINKAGES:
72      0071 1
73      0072 1     NONE
74      0073 1
75      0074 1 INCLUDE FILES:
76      0075 1
77      0076 1 REQUIRE 'RTLIN:COBPROLOG';           ! Defines psects, macros, &
78      1593 1                                     ! terminal defs
79      1594 1 REQUIRE 'RTLIN:COBLNK';             ! Linkages
80      1669 1
81      1670 1 TABLE OF CONTENTS:
82      1671 1
83      1672 1
84      1673 1 FORWARD ROUTINE
85      1674 1
86      1675 1     COB$$DOWN_SCROLL_R2 : COB$$ESC_R2_LNK, ! Create downscroll sequence
87      1676 1     COB$$ERASE_LINE_R2 : COB$$ESC_R2_LNK, ! Create erase line sequence
88      1677 1     COB$$ERASE_PAGE_R2 : COB$$ESC_R2_LNK, ! Create erase page sequence
89      1678 1     COB$$ERASE_WHOLE_LINE_R2 : COB$$ESC_R2_LNK, ! Create erase whole line sequence
90      1679 1     COB$$ERASE_WHOLE_PAGE_R2 : COB$$ESC_R2_LNK, ! Create erase whole page sequence
91      1680 1     COB$$SET_ATTRIBUTES,                ! Create set attributes sequences w text
92      1681 1     COB$$SET_ATTRIBUTES_ONLY,          ! Create set attributes sequences w no text
93      1682 1     COB$$SET_CURSOR_ABS_R4 : COB$$ESC_R4_LNK, ! Create absolute set cursor sequence
94      1683 1     COB$$SET_CURSOR_REL,              ! Create relative set cursor sequence
95      1684 1     COB$$SETUP_TERM_TYPE,              ! Setup terminal type for COB$$ calls
96      1685 1     COB$$UP_SCROLL_R2 : COB$$ESC_R2_LNK; ! Create upscroll sequence
97      1686 1
98      1687 1
99      1688 1 MACROS:
100     1689 1
101     1690 1
102     1691 1
103     1692 1 EQUATED SYMBOLS:
104     1693 1
105     1694 1
106     1695 1
107     1696 1 FIELDS:
108     1697 1
109     1698 1     NONE
110     1699 1
111     1700 1 PSECTS:
112     1701 1
113     1702 1
114     1703 1 OWN STORAGE:
115     1704 1
116     1705 1     NONE
117     1706 1
118     1707 1
119     1708 1 EXTERNAL REFERENCES:
120     1709 1
121     1710 1

```

```
: 122      1711  1 EXTERNAL ROUTINE
: 123      1712  1
: 124      1713  1      LIB$FREE_EF,      ! free event flag number
: 125      1714  1      LIB$GET_EF;      ! get event flag number
: 126      1715  1
: 127      1716  1 !<BLF/PAGE>
```

```

129 1717 1 %SBTTL 'COB$$$DOWN_SCROLL_R2 - Create downscroll sequence'
130 1718 1 GLOBAL ROUTINE COB$$$DOWN_SCROLL_R2 (
131 1719 1     TERM_TYPE,
132 1720 1     BUFFER,
133 1721 1     CUR_SIZE
134 1722 1 ) : COB$$$ESC_R2_LNK =
135 1723 1
136 1724 1  +-+
137 1725 1  FUNCTIONAL DESCRIPTION:
138 1726 1  This routine generates the escape sequence for down scroll
139 1727 1  and appends the string to a given output buffer.
140 1728 1
141 1729 1  CALLING SEQUENCE:
142 1730 1
143 1731 1  ret_status.wlc.v = COB$$$DOWN_SCROLL_R2 (TERM_TYPE.rl.v, BUFFER.mt.r,
144 1732 1  CUR_SIZE.ml.r)
145 1733 1
146 1734 1  FORMAL PARAMETERS:
147 1735 1
148 1736 1  TERM_TYPE.rl.v      terminal type
149 1737 1  BUFFER.mt.r         addr of buffer
150 1738 1  CUR_SIZE.ml.r      # bytes currently in buffer
151 1739 1
152 1740 1  IMPLICIT INPUTS:
153 1741 1
154 1742 1  NONE
155 1743 1
156 1744 1  IMPLICIT OUTPUTS:
157 1745 1
158 1746 1  NONE
159 1747 1
160 1748 1  COMPLETION STATUS:
161 1749 1
162 1750 1
163 1751 1  SIDE EFFECTS:
164 1752 1
165 1753 1  NONE
166 1754 1  --
167 1755 1
168 1756 2  BEGIN
169 1757 2
170 1758 2  LOCAL
171 1759 2  FREE_ADDR;
172 1760 2
173 1761 2  BIND
174 1762 2  VT05_DOWN = UPLIT (BYTE (CR, VT05_CUP, NULL)),
175 1763 2  VT52_DOWN = UPLIT (BYTE (ESC, VT52_DWN)),
176 1764 2  VT100_DOWN = UPLIT (BYTE (ESC, VT100_DWN));
177 1765 2
178 1766 2  FREE_ADDR = .BUFFER + ..CUR_SIZE;
179 1767 2
180 1768 2  CASE .TERM_TYPE FROM UNKNOWN TO HARDCOPY OF
181 1769 2  SET
182 1770 2  [VT05]:
183 1771 3  BEGIN
184 1772 3  CH$MOVE (3, VT05_DOWN, .FREE_ADDR);
185 1773 3  .CUR_SIZE = ..CUR_SIZE + 3;
    
```

```

186 1774 2      END;
187 1775 2
188 1776 2      [VT52]:
189 1777 3      BEGIN
190 1778 3      CH$MOVE (2, VT52_DOWN, .FREE_ADDR);
191 1779 3      .CUR_SIZE = ..CUR_SIZE + 2;
192 1780 2      END;
193 1781 2
194 1782 2      [VT100]:
195 1783 3      BEGIN
196 1784 3      CH$MOVE (2, VT100_DOWN, .FREE_ADDR);
197 1785 3      .CUR_SIZE = ..CUR_SIZE + 2;
198 1786 2      END;
199 1787 2
200 1788 2      [HARDCOPY, UNKNOWN, VTFORIGN]:
201 1789 2      ;
202 1790 2
203 1791 2      [INRANGE, OTRANGE]:
204 1792 2      RETURN 0;
205 1793 2      ! should never get here
206 1794 2
207 1795 2      TES;
208 1796 2      RETURN (SS$_NORMAL);
209 1797 2
210 1798 1      END;
    
```

.TITLE COB\$\$\$ESCAPE\_GENERATOR COB\$\$\$ESCAPE\_GENERATOR - E  
 scape sequence generat

.IDENT \1-003\

.PSECT \_COB\$CODE,NOWRT, SHR, PIC,2

```

00 1A 0D 0000 P.AAA: .BYTE 13, 26, 0
0003 .BLKB 1
49 1B 0000 P.AAB: .BYTE 27, 73
0006 .BLKB 2
4D 1B 0000 P.AAC: .BYTE 27, 77
    
```

```

VT05_DOWN= P.AAA
VT52_DOWN= P.AAB
VT100_DOWN= P.AAC
.EXTRN LIB$FREE_EF, LIB$GET_EF
    
```

```

51 62 C0 0000 COB$$$DOWN_SCROLL_R2:
001F 05 00 50 CF 00003 ADDL2 (CUR_SIZE), FREE_ADDR : 1766
000E 0026 00007 1$: CASEL TERM_TYPE, #0, #5 : 1768
0026 0000F .WORD 6$-1$, -
3$-1$, -
4$-1$, -
6$-1$, -
6$-1$
61 18 00 DE 1C 11 00013 BRB 7$ : 1792
AF F0 00015 2$: INSV VT05_DOWN, #0, #24, (FREE_ADDR) : 1772
03 C0 0001B ADDL2 #3, (CUR_SIZE) : 1773
0D 11 0001E BRB 6$ : 1768
    
```



```

61      D7  AF  B0 00020 3$:  MOVW  VT52_DOWN, (FREE_ADDR)      : 1778
        04  11 00024      BRB      5$                               : 1779
61      D5  AF  B0 00026 4$:  MOVW  VT100_DOWN, (FREE_ADDR)     : 1784
62      02  C0 0002A 5$:  ADDL2  #2, (CUR_SIZE)           : 1785
50      01  D0 0002D 6$:  MOVL   #1, R0                          : 1796
        05  00030      RSB                               :
        50  D4 00031 7$:  CLRL   R0                            : 1798
        05  00033      RSB                               :
  
```

: Routine Size: 52 bytes, Routine Base: \_COB\$CODE + 000A

: 211 1799 1 !<BLF/PAGE>

```

213 1800 1 %SBTTL 'COB$ERASE LINE R2 - Create erase line sequence'
214 1801 1 GLOBAL ROUTINE COB$ERASE_LINE_R2 (
215 1802 1     TERM_TYPE,
216 1803 1     BUFFER,
217 1804 1     CUR_SIZE
218 1805 1 ) : COB$ESCAP_R2_LNK =
219 1806 1
220 1807 1 ++
221 1808 1 FUNCTIONAL DESCRIPTION:
222 1809 1     This routine generates the escape sequence for erasing a
223 1810 1     line from the current cursor position. The string is
224 1811 1     appended to the given output buffer.
225 1812 1
226 1813 1 CALLING SEQUENCE:
227 1814 1
228 1815 1     ret_status.wlc.v = COB$ERASE_LINE_R2 (TERM_TYPE.rl.v,
229 1816 1     BUFFER.mt.r, CUR_SIZE.ml.r)
230 1817 1
231 1818 1 FORMAL PARAMETERS:
232 1819 1
233 1820 1     TERM_TYPE.rl.v     terminal type
234 1821 1     BUFFER.mt.r       addr of buffer
235 1822 1     CUR_SIZE.ml.r     # bytes currently in buffer
236 1823 1                   updated to reflect erase seq added
237 1824 1
238 1825 1 IMPLICIT INPUTS:
239 1826 1
240 1827 1     NONE
241 1828 1
242 1829 1 IMPLICIT OUTPUTS:
243 1830 1
244 1831 1     NONE
245 1832 1
246 1833 1 COMPLETION STATUS:
247 1834 1
248 1835 1
249 1836 1 SIDE EFFECTS:
250 1837 1
251 1838 1     NONE
252 1839 1 --
253 1840 1
254 1841 2 BEGIN
255 1842 2
256 1843 2 LOCAL
257 1844 2     FREE_ADDR;                ! addr of next free byte in buffer
258 1845 2
259 1846 2 BIND
260 1847 2     VT05_LINE = UPLIT (BYTE (VT05_EOL, NULL, NULL)),
261 1848 2     VT52_LINE = UPLIT (BYTE (ESC, VT52_EOL)),
262 1849 2     VT100_LINE = UPLIT (BYTE (ESC, LB, VT100_EOL));
263 1850 2
264 1851 2 FREE_ADDR = .BUFFER + ..CUR_SIZE;
265 1852 2
266 1853 2 CASE .TERM_TYPE FROM UNKNOWN TO HARDCOPY OF
267 1854 2 SET
268 1855 2     [VT05]:
269 1856 2     BEGIN
    
```



61	18	00	D4	AF	F0	00026	4\$:	INSV	VT100	LINE, #0, #24, (FREE_ADDR)
		62		03	C0	0002C	5\$:	ADDL2	#3, (CUR_SIZE)	
		50		01	20	0002F	6\$:	MOVL	#1, R0	
					25	00032		RSB		
				50	24	00033	7\$:	CLRL	R0	
					05	00035		RSB		

: 1869  
 : 1870  
 : 1881  
 : 1883  
 :

: Routine Size: 54 bytes, Routine Base: \_COB\$CODE + 004B

: 297 1884 1 !<BLF/PAGE>

```

299 1885 1 %SBTTL 'COB$$$ERASE PAGE R2 - Create erase page sequence'
300 1886 1 GLOBAL ROUTINE COB$$$ERASE_PAGE_R2 (
301 1887 1     TERM_TYPE,
302 1888 1     BUFFER,
303 1889 1     CUR_SIZE
304 1890 1 ) : COB$$$ESC_R2_LNK =
305 1891 1 ++
306 1892 1 FUNCTIONAL DESCRIPTION:
307 1893 1
308 1894 1     This routine generates the escape sequence for erasing the
309 1895 1     page from the current cursor position to the end of the
310 1896 1     page. The sequence is appended into the output buffer.
311 1897 1
312 1898 1 CALLING SEQUENCE:
313 1899 1
314 1900 1     ret_status.wlc.v = COB$$$ERASE_PAGE_R2 (TERM_TYPE.rl.v,
315 1901 1     BUFFER.mt.r, CUR_SIZE.ml.r)
316 1902 1
317 1903 1 FORMAL PARAMETERS:
318 1904 1
319 1905 1     TERM_TYPE.rl.v     terminal type
320 1906 1     BUFFER.mt.r       addr of buffer
321 1907 1     CUR_SIZE.ml.r     # bytes currently in buffer
322 1908 1
323 1909 1 IMPLICIT INPUTS:
324 1910 1
325 1911 1     NONE
326 1912 1
327 1913 1 IMPLICIT OUTPUTS:
328 1914 1
329 1915 1     NONE
330 1916 1
331 1917 1 COMPLETION STATUS:
332 1918 1
333 1919 1
334 1920 1 SIDE EFFECTS:
335 1921 1
336 1922 1     NONE
337 1923 1 --
338 1924 1
339 1925 2 BEGIN
340 1926 2
341 1927 2 LOCAL
342 1928 2     FREE_ADDR;                ! addr of next free byte in buffer
343 1929 2
344 1930 2 BIND
345 1931 2     VT05_ERASE = UPLIT (BYTE (VT05_EOS, NULL, NULL)),
346 1932 2     VT52_ERASE = UPLIT (BYTE (ESC, VT52_EOS)),
347 1933 2     VT100_ERASE = UPLIT (BYTE (ESC, LB, VT100_EOS));
348 1934 2
349 1935 2     FREE_ADDR = .BUFFER + ..CUR_SIZE;
350 1936 2
351 1937 2 CASE .TERM_TYPE FROM UNKNOWN TO HARDCOPY OF
352 1938 2 SET
353 1939 2     [VT05]:
354 1940 3     BEGIN
355 1941 3     CH$MOVE (3, VT05_ERASE, .FREE_ADDR);
  
```

```

356 1942 3          .CUR_SIZE = ..CUR_SIZE + 3;
357 1943 3          END;
358 1944 3
359 1945 3          [VT52]:
360 1946 3          BEGIN
361 1947 3          CH$MOVE (2, VT52_ERASE, .FREE_ADDR);
362 1948 3          .CUR_SIZE = ..CUR_SIZE + 2;
363 1949 3          END;
364 1950 3
365 1951 3          [VT100]:
366 1952 3          BEGIN
367 1953 3          CH$MOVE (3, VT100_ERASE, .FREE_ADDR);
368 1954 3          .CUR_SIZE = ..CUR_SIZE + 3;
369 1955 3          END;
370 1956 3
371 1957 3          [HARDCOPY, UNKNOWN, VTFOREIGN]:
372 1958 3          ;
373 1959 3
374 1960 3          [INRANGE, OVRANGE]:
375 1961 3          RETURN 0;
376 1962 3          ! should never get here
377 1963 3
378 1964 3          TES;
379 1965 3          RETURN (SS$_NORMAL);
380 1966 3
381 1967 1          END;
! End of routine COB$ERASE_PAGE_R2
    
```

00	00	1F	00081	P.AAG:	.BLKB	3	
			00084		.BYTE	31, 0, 0	:
	4A	1B	00087	P.AAH:	.BLKB	1	:
			00088		.BYTE	27, 74	:
4A	5B	1B	0008A	P.AAI:	.BLKB	2	:
			0008C		.BYTE	27, 91, 74	:
				VT05_ERASE=		P.AAG	
				VT52_ERASE=		P.AAH	
				VT100_ERASE=		P.AAI	

		51	62	C0	00000	COB\$ERASE_PAGE_R2::	
						ADDL2 (CUR_SIZE), FREE_ADDR	: 1935
001F	05	00	50	CF	00003	CASEL TERM_TYPE, #0, #5	: 1937
	0016	000E	0028		00007	1\$: .WORD	
		0028	0028		0000F	6\$-1\$,-	
						2\$-1\$,-	
						3\$-1\$,-	
						4\$-1\$,-	
						6\$-1\$,-	
						6\$-1\$	
			1E	11	00013	BRB	: 1961
61	18	00	DD	AF	F0 00015	2\$: INSV	: 1941
				OF	11 0001B	BRB	: 1942
		61	D9	AF	B0 0001D	3\$: MOVW	: 1947
		62		02	C0 00021	ADDL2	: 1948
				09	11 00024	BRB	: 1937
61	18	00	D4	AF	F0 00026	4\$: INSV	: 1953
						VT100_ERASE, #0, #24, (FREE_ADDR)	

62	03	C0 0002C	5\$:	ADDL2	#3, (CUR_SIZE)	:	1954
50	01	D0 0002F	6\$:	MOVL	#1, R0	:	1965
		05 00032		RSB		:	
	50	D4 00033	7\$:	CLRL	R0	:	1967
		05 00035		RSB		:	

: Routine Size: 54 bytes, Routine Base: \_COB\$CODE + 008F

: 382 1968 1 !<BLF/PAGE>

```

384 1969 1 %SBTTL 'COB$ERASE WHOLE LINE R2 - Create erase whole line sequence'
385 1970 1 GLOBAL ROUTINE COB$ERASE_WHOLE_LINE_R2 (
386 1971 1     TERM_TYPE,
387 1972 1     BUFFER,
388 1973 1     CUR_SIZE
389 1974 1 ) : COB$ESC_R2_LNK =
390 1975 1 +-+
391 1976 1 FUNCTIONAL DESCRIPTION:
392 1977 1
393 1978 1     This routine generates the escape sequence to erase the entire
394 1979 1     line containing the current cursor position. The string is
395 1980 1     appended into the output buffer.
396 1981 1
397 1982 1     Notice that only VT100s have the ability to erase an entire
398 1983 1     line regardless of whether the cursor is at the beginning
399 1984 1     of that line. Most terminals can only erase from the cursor
400 1985 1     to the end of line.
401 1986 1
402 1987 1 CALLING SEQUENCE:
403 1988 1
404 1989 1     ret_status.wlc.v = COB$ERASE_WHOLE_LINE_R2 (TERM_TYPE.rl.v,
405 1990 1     BUFFER.mt.r,
406 1991 1     CUR_SIZE.ml.r)
407 1992 1
408 1993 1 FORMAL PARAMETERS.
409 1994 1
410 1995 1     TERM_TYPE.rl.v     terminal type
411 1996 1     BUFFER.mt.r        addr of buffer
412 1997 1     CUR_SIZE.ml.r      # bytes currently in buffer
413 1998 1
414 1999 1 IMPLICIT INPUTS:
415 2000 1
416 2001 1     NONE
417 2002 1
418 2003 1 IMPLICIT OUTPUTS:
419 2004 1
420 2005 1     NONE
421 2006 1
422 2007 1 COMPLETION STATUS:
423 2008 1
424 2009 1
425 2010 1 SIDE EFFECTS:
426 2011 1
427 2012 1     NONE
428 2013 1 --
429 2014 1
430 2015 2 BEGIN
431 2016 2
432 2017 2 LOCAL
433 2018 2     FREE_ADDR;           ! addr of next free byte in buffer
434 2019 2
435 2020 2 BIND
436 2021 2     VT05_LINE = UPLIT (BYTE (VT05_EOL, NULL, NULL)),
437 2022 2     VT52_LINE = UPLIT (BYTE (ESC, VT52_EOL)),
438 2023 2     VT100_WHOLE_LINE = UPLIT (BYTE (ESC, LB, TWO, VT100_EOL));
439 2024 2
440 2025 2     EE_ADDR = .BUFFER + ..CUR_SIZE;
  
```



```

441 2026 2
442 2027
443 2028 CASE .TERM_TYPE FROM UNKNOWN TO HARDCOPY OF
444 2029 SET
445 2030 [VT05]:
446 2031 BEGIN
447 2032 CHSMOVE (3, VT05_LINE, .FREE_ADDR);
448 2033 .CUR_SIZE = ..CUR_SIZE + 3;
449 2034 END;
450 2035 [VT52]:
451 2036 BEGIN
452 2037 CHSMOVE (2, VT52_LINE, .FREE_ADDR);
453 2038 .CUR_SIZE = ..CUR_SIZE + 2;
454 2039 END;
455 2040 [VT100]:
456 2041 BEGIN
457 2042 CHSMOVE (4, VT100_WHOLE_LINE, .FREE_ADDR);
458 2043 .CUR_SIZE = ..CUR_SIZE + 4;
459 2044 END;
460 2045
461 2046 [HARDCOPY, UNKNOWN, VTFOREIGN]:
462 2047 ;
463 2048
464 2049 [INRANGE, OTRANGE]:
465 2050 RETURN 0;
466 2051 ! should never get here
467 2052
468 2053 TES;
469 2054
470 2055 RETURN (SS$_NORMAL);
471 2056
472 2057 END;
! End of routine COB$ERASE_WHOLE_LINE_R2
    
```

00	00	1E	000C5	P.AAJ:	.BLKB	3	
			000C8		.BYTE	30, 0, 0	
			000CB	P.AAK:	.BLKB	1	
	4B	1B	000CC		.BYTE	27, 75	
			000CE	P.AAL:	.BLKB	2	
4B	32	5B	1B	000D0	.BYTE	27, 91, 50, 75	
				VT05_LINE=		P.AAJ	
				VT52_LINE=		P.AAK	
				VT100_WHOLE_LINE=		P.AAL	

0022	05	00	51	62	CO	00000	COB\$ERASE WHOLE LINE R2::		
	0019	000E	00	50	CF	00003	ADDL2 (CUR_SIZE), FREE_ADDR		: 2025
		0029	0029	0029		00007	CASEL TERM_TYPE, #0, #5		: 2027
		0029				0000F	1\$: .WORD	58-1\$, -	
								28-1\$, -	
								38-1\$, -	
								48-1\$, -	
								58-1\$, -	
								68-1\$, -	
				1F	11	00013	BRB	68	: 2051

61	18	00	DC	AF	F0	00015	2\$:	INSV	VT05_LINE, #0, #24, (FREE_ADDR)	:	2031
		62		03	C0	0001B		ADDL2	#3, (CUR_SIZE)	:	2032
				10	11	0001E		BRB	5\$	:	2027
		61	D5	AF	B0	00020	3\$:	MOVW	VT52_LINE, (FREE_ADDR)	:	2037
		62		02	C0	00024		ADDL2	#2, (CUR_SIZE)	:	2038
				07	11	00027		BRB	5\$	:	2027
		61	D0	AF	D0	00029	4\$:	MOVL	VT100_WHOLE_LINE, (FREE_ADDR)	:	2043
		62		04	C0	0002D		ADDL2	#4, (CUR_SIZE)	:	2044
		50		01	D0	00030	5\$:	MOVL	#1, R0	:	2055
					05	00033		RSB		:	
				50	D4	00034	6\$:	CLRL	R0	:	2057
					05	00036		RSB		:	

; Routine Size: 55 bytes, Routine Base: \_COB\$CODE + 00D4

; 473 2058 1 !<BLF/PAGE>

```

475 2059 1 %SBTTL 'COB$ERASE WHOLE PAGE R2 - Create erase whole page sequence'
476 2060 1 GLOBAL ROUTINE COB$ERASE_WHOLE_PAGE_R2 (
477 2061 1     TERM_TYPE,
478 2062 1     BUFFER,
479 2063 1     CUR_SIZE
480 2064 1 ) : COB$ESC_R2_LNK =
481 2065 1
482 2066 1  **
483 2067 1  FUNCTIONAL DESCRIPTION:
484 2068 1      This routine generates the escape sequence to erase the
485 2069 1      whole page regardless of cursor position. The string is appended
486 2070 1      into the output buffer.
487 2071 1
488 2072 1  CALLING SEQUENCE:
489 2073 1
490 2074 1      ret_status.wlc.v = COB$ERASE_WHOLE_PAGE_R2 (TERM_TYPE.rl.v,
491 2075 1      BUFFER.mt.r,
492 2076 1      CUR_SIZE.ml.r)
493 2077 1
494 2078 1  FORMAL PARAMETERS:
495 2079 1
496 2080 1      TERM_TYPE.rl.v      terminal type
497 2081 1      BUFFER.mt.r        addr of buffer
498 2082 1      CUR_SIZE.ml.r     # bytes currently in buffer
499 2083 1
500 2084 1  IMPLICIT INPUTS:
501 2085 1
502 2086 1      NONE
503 2087 1
504 2088 1  IMPLICIT OUTPUTS:
505 2089 1
506 2090 1      NONE
507 2091 1
508 2092 1  COMPLETION STATUS:
509 2093 1
510 2094 1
511 2095 1  SIDE EFFECTS:
512 2096 1
513 2097 1      NONE
514 2098 1  --
515 2099 1
516 2100 2  BEGIN
517 2101 2
518 2102 2  LOCAL
519 2103 2      FREE_ADDR;          ! addr of next free byte in buffer
520 2104 2
521 2105 2  LITERAL
522 2106 2      LINE1 = 32;         ! 1 + 31 bias
523 2107 2      COL1 = 32;         ! 1 + 31 bias
524 2108 2
525 2109 2  BIND
526 2110 2      VT05_ERASE = UPLIT (BYTE (VT05_EOS, NULL, NULL)),
527 2111 2      VT52_ERASE = UPLIT (BYTE (ESC, VT52_SC, LINE1, COL1,
528 2112 2      ESC, VT52_EOS)),
529 2113 2      VT100_ERASE_WHOLE = UPLIT (BYTE (ESC, LB, TWO, VT100_EOS));
530 2114 2
531 2115 2  FREE_ADDR = .BUFFER + ..CUR_SIZE;
    
```

```
532 2116 2
533 2117
534 2118
535 2119
536 2120
537 2121
538 2122
539 2123
540 2124
541 2125
542 2126
543 2127
544 2128
545 2129
546 2130
547 2131
548 2132
549 2133
550 2134
551 2135
552 2136
553 2137
554 2138
555 2139
556 2140
557 2141
558 2142
559 2143
560 2144
561 2145
562 2146
563 2147
564 2148
565 2149
566 2150
567 2151
568 2152 1

CASE .TERM_TYPE FROM UNKNOWN TO HARDCOPY OF
SET
  [VT100]:
  BEGIN
    CH$MOVE (4, VT100_ERASE_WHOLE, .FREE_ADDR);
    .CUR_SIZE = ..CUR_SIZE + 4;
  END;

  [VT52]:
  BEGIN
    +
    ! There is no sequence to erase the screen and leave the
    ! cursor where it was, so on a VT52 we have to settle for
    ! setting the cursor to 1,1 and erasing to the end of screen.
    -
    CH$MOVE (6, VT52_ERASE, .FREE_ADDR);
    .CUR_SIZE = ..CUR_SIZE + 6;
  END;

  [VT05]:
  BEGIN
    CH$MOVE (3, VT05_ERASE, .FREE_ADDR);
    .CUR_SIZE = ..CUR_SIZE + 3;
  END;

  [HARDCOPY, UNKNOWN, VTFOREIGN]:
  :

  [INRANGE, OUTRANGE]:
  RETURN 0; ! should never get here

TES:
RETURN (SS$_NORMAL);

END; ! End of routine COB$ERASE_WHOLE_PAGE_R2
```

					0010B		.BLKB	1		
		00	00	1F	0010C	P.AAM:	.BYTE	31, 0, 0	:	
					0010F		.BLKB	1	:	
	4A	1B	20	20	59	1B	00110	P.AAN:	.BYTE	27, 89, 32, 32, 27, 74
							00116		.BLKB	2
		4A	32	5B	1B	00118	P.AAO:	.BYTE	27, 91, 50, 74	:
							VT05_ERASE=		P.AAM	
							VT52_ERASE=		P.AAN	
							VT100_ERASE_WHOLE=		P.AAO	
			00F8	8F	BB	00000	COB\$ERASE WHOLE PAGE R2::			
							PUSHR	#^M<R3,R4,R5,R6,R7>	:	2060
		56		52	D0	00004	MOVL	R2, R6	:	
57		51		66	C1	00007	ADDL3	(CUR_SIZE), BUFFER, FREE_ADDR	:	2115
05		00		50	CF	0000B	CASEL	TERM_TYPE, #0, #5	:	2117

000E	0017	0021 002A	002A 002A	0000F 00017	1\$:	.WORD	5\$-1\$,- 4\$-1\$,- 3\$-1\$,- 2\$-1\$,- 5\$-1\$,- 5\$-1\$		
				21	11	0001B	BRB	6\$	2146
		67	DC	AF	D0	0001D	MOVL	VT100_ERASE_WHOLE, (FREE_ADDR)	2121
		66		04	C0	00021	ADDL2	#4, (CUR_SIZE)	2122
				13	11	00024	BRB	5\$	2117
	67	CA	AF	06	28	00026	MOV3	#6, VT52_ERASE, (FREE_ADDR)	2132
			66	06	C0	0002B	ADDL2	#6, (CUR_SIZE)	2133
				09	11	0002E	BRB	5\$	2117
67	18		BD	AF	F0	00030	INSV	VT05_ERASE, #0, #24, (FREE_ADDR)	2138
				03	C0	00036	ADDL2	#3, (CUR_SIZE)	2139
				01	D0	00039	MOVL	#1, R0	2150
				02	11	0003C	BRB	7\$	
				50	D4	0003E	CLRL	R0	2152
			00F8	8F	BA	00040	POPR	#^M<R3,R4,R5,R6,R7>	
				05	00044	RSB			

: Routine Size: 69 bytes, Routine Base: \_COB\$CODE + 011C

: 569 2153 1 !<BLF/PAGE>

```
571 2154 1 %SBTTL 'COB$$$SET_ATTRIBUTES - Create set attributes sequence'  
572 2155 1 GLOBAL ROUTINE COB$$$SET_ATTRIBUTES (  
573 2156 1     TERM_TYPE,  
574 2157 1     IN_TEXT,  
575 2158 1     IN_LEN,  
576 2159 1     FLAGS,  
577 2160 1     OUT_BUF,  
578 2161 1     OUT_LEN  
579 2162 1 ) =  
580 2163 1 +-  
581 2164 1 FUNCTIONAL DESCRIPTION:  
582 2165 1  
583 2166 1     This routine generates the escape sequence turning on  
584 2167 1     attributes such as bolding and blinking. The attribute  
585 2168 1     sequence is placed in the output buffer, the input text  
586 2169 1     is copied over, and then the sequence to turn off graphics  
587 2170 1     is appended.  
588 2171 1  
589 2172 1 CALLING SEQUENCE:  
590 2173 1  
591 2174 1     ret_status.wlc.v = COB$$$SET_ATTRIBUTES (TERM_TYPE.rl.v, IN_TEXT.rt.r,  
592 2175 1     IN_LEN.rl.v, FLAGS.rl.v,  
593 2176 1     OUT_BUF.mt.r, OUT_LEN.ml.r)  
594 2177 1  
595 2178 1 FORMAL PARAMETERS:  
596 2179 1  
597 2180 1     TERM_TYPE.rl.v     terminal type  
598 2181 1     IN_TEXT.rt.dx     descriptor of text which will have attr on  
599 2182 1     IN_LEN.rl.v     length of caller's text  
600 2183 1     FLAGS.rl.v     flags specifying which attributes to turn on  
601 2184 1     OUT_BUF.mt.r     addr of output buffer  
602 2185 1     OUT_LEN.ml.r     # bytes in output buffer, includes attributes,  
603 2186 1     caller's text, & turn off graphic rendition  
604 2187 1  
605 2188 1 IMPLICIT INPUTS:  
606 2189 1  
607 2190 1     NONE  
608 2191 1  
609 2192 1 IMPLICIT OUTPUTS:  
610 2193 1  
611 2194 1     NONE  
612 2195 1  
613 2196 1 COMPLETION STATUS:  
614 2197 1  
615 2198 1  
616 2199 1 SIDE EFFECTS:  
617 2200 1  
618 2201 1     NONE  
619 2202 1 --  
620 2203 1  
621 2204 2 BEGIN  
622 2205 2  
623 2206 2 LOCAL  
624 2207 2     FREE_ADDR;  
625 2208 2  
626 2209 2 MACRO  
627 2210 2     VT100_OFF = %STRING (%CHAR (ESC), %CHAR (LB), '0m')%;
```

```

628 2211 2
629 2212 2
630 2213 2
631 2214 2
632 2215 2
633 2216 2
634 2217 2
635 2218 2
636 2219 2
637 2220 2
638 2221 2
639 2222 2
640 2223 2
641 2224 2
642 2225 2
643 2226 2
644 2227 2
645 2228 2
646 2229 2
647 2230 2
648 2231 2
649 2232 2
650 2233 2
651 2234 2
652 2235 2
653 2236 2
654 2237 2
655 2238 2
656 2239 2
657 2240 2
658 2241 2
659 2242 2
660 2243 2
661 2244 2
662 2245 2
663 2246 2
664 2247 2
665 2248 2
666 2249 2
667 2250 2
668 2251 2
669 2252 2
670 2253 2
671 2254 2
672 2255 2
673 2256 2
674 2257 2
675 2258 2
676 2259 2
677 2260 2
678 2261 2
679 2262 2
680 2263 2
681 2264 2
682 2265 2
683 2266 2
684 2267 2

FREE_ADDR = .OUT_BUF + ..OUT_LEN;          ! init to first free byte

CASE .TERM_TYPE FROM UNKNOWN TO HARDCOPY OF
SET
[HARDCOPY, UNKNOWN, VT05, VT52, VTFOREIGN]:
BEGIN
+
! Renditions not supported on these devices. Just
! copy the text into the output buffer and return.
-
CH$MOVE (.IN_LEN, .IN_TEXT, .FREE_ADDR);
.OUT_LEN = ..OUT_LEN + .IN_LEN;
RETURN (SS$NORMAL);
END;

[INRANGE, OUTRANGE]:
RETURN 0;          ! error

[VT100]:
BEGIN
IF .FLAGS <0,4> EQL 0
THEN
BEGIN
! no attr requested
CH$MOVE (.IN_LEN, .IN_TEXT, .FREE_ADDR);
.OUT_LEN = ..OUT_LEN + .IN_LEN;
RETURN (SS$NORMAL);
END;
+
! For each attribute bit set in flags, copy
! the appropriate ASCII graphic rendition byte
! followed by a ';' into the output buffer.
! Note use of autoincrementing.
-
CH$WCHAR_A (ESC, FREE_ADDR);
CH$WCHAR_A (LB, FREE_ADDR);
INCR I FROM 0 TO 3
DO
BEGIN
! build attribute string
BIND
ATTRTABL = UPLIT (BYTE ('1754')) : VECTOR [4, BYTE];

IF .FLAGS <.I, 1>
THEN
BEGIN
CH$WCHAR_A (.ATTRTABL [.I], FREE_ADDR);
CH$WCHAR_A (%C';', FREE_ADDR);
.OUT_LEN = ..OUT_LEN + 2; ! keep updating length
END;
END;

+
! When we fall out of above loop we have deposited
! an extra ';' at the end of the buffer. Back up
! FREE_ADDR and write VT100_SGR on top of it.
-
FREE_ADDR = .FREE_ADDR - 1;
    
```

```

685 2268 3      CH$WCHAR_A (VT100_SGR, FREE_ADDR);
686 2269 2
687 2270 2      TES;
688 2271 2
689 2272 2
690 2273 2      !+
691 2274 2      ! If we get here, the appropriate graphic rendition string has
692 2275 2      ! been moved to the output buffer. Now copy the user's text over.
693 2276 2      FREE_ADDR = CH$MOVE (.IN_LEN, .IN_TEXT, .FREE_ADDR);
694 2277 2
695 2278 2      !+
696 2279 2      ! Append in sequence to turn off graphic rendition.
697 2280 2
698 2281 2      CH$MOVE (%CHARCOUNT (VT100_OFF), UPLIT (BYTE (VT100_OFF)), .FREE_ADDR);
699 2282 2
700 2283 2      !+
701 2284 2      ! Set the output length and exit.
702 2285 2
703 2286 2      .OUT_LEN = ..OUT_LEN + .IN_LEN + 6; ! add length of caller's text &
704 2287 2      ! turn on/off graphic rendition
705 2288 2      RETURN (SS$_NORMAL);
706 2289 2
707 2290 1      END;

```

! End of routine COB\$\$\$SET\_ATTRIBUTES

					00161				.BLKB	3	
	34	35	37	31	00164	P.AAP:	.ASCII	\1754\			
	6D	30	5B	1B	00168	P.AAQ:	.ASCII	<27>\[0m\			

					00FC	00000			.ENTRY	COB\$\$\$SET_ATTRIBUTES, Save R2,R3,R4,R5,R6,R7	2155
				18	AC	DO	00002		MOVL	OUT_LEN, R6	2212
	57	14	56		AC	C1	00006		ADDL3	(R6), OUT_BUF, FREE_ADDR	
	05		00	04	AC	CF	0000B		CASEL	TERM TYPE, #0, #5	2214
000E	0014		0014		0014		00010	1\$:	.WORD	3\$-1\$, -	
			0014		0014		00018			3\$-1\$, -	
										3\$-1\$, -	
										2\$-1\$, -	
										3\$-1\$, -	
										3\$-1\$	
					4D	11	0001C		BRB	8\$	2228
			0F	10	AC	93	0001E	2\$:	BITB	FLAGS, #15	2232
					0C	12	00022		BNEQ	4\$	
	67	08	BC		AC	28	00024	3\$:	MOV3	IN_LEN, @IN_TEXT, (FREE_ADDR)	2235
			66		OC	C0	0002A		ADDL2	IN_LEN, (R6)	2236
					37	11	0002E		BRB	7\$	2237
			87	5B1B	8F	B0	00030	4\$:	MOVW	#23323, (FREE_ADDR)+	2245
					50	D4	00035		CLRL	I	2247
	0B	10	AC		50	E1	00037	5\$:	BBC	I, FLAGS, 6\$	2253
			87		B8	AF40	90	0003C	MOV3	ATTRTABL[I], (FREE_ADDR)+	2256
			87		38	90	00041		MOV3	#59, (FREE_ADDR)+	2257
			66		02	C0	00044		ADDL2	#2, (R6)	2258
	EC		50		03	F3	00047	6\$:	AOBLEQ	#3, I, 5\$	2247
			77		6D	8F	90	0004B	MOV3	#109, -(FREE_ADDR)	2268



```

        67      08  BC      0C      57  D6 0004F      INCL  FREE_ADDR
        57      57      53  D0 00051      MOVCL IN_LEN, @IN_TEXT, (FREE_ADDR)
        67      9F  AF  D0 0005A      MOVL  R3, FREE_ADDR
        50      66      0C  AC  C1 0005E      MOVL  P.AAQ, (FREE_ADDR)
        66      06  A0  9E 00063      ADDL3 IN_LEN, (R6), R0
        50      01  D0 00067 7$:      MOVAB 6(R0), (R6)
        04 0006A      MOVL  #1, R0
        50      D4 0006B 8$:      RET
        04 0006D      CLRI  R0
    
```

: Routine Size: 110 bytes, Routine Base: \_COB\$CODE + 016C

: 708 2291 1 !<BLF/PAGE>

```

710 2292 1 %SBTTL 'COB$SET_ATTRIBUTES_ONLY - Create only set attributes sequence'
711 2293 1 GLOBAL ROUTINE COB$SET_ATTRIBUTES_ONLY (
712 2294 1     TERM TYPE,
713 2295 1     FLAGS,
714 2296 1     PREFIX BUF,
715 2297 1     P_PREFIX_LEN,
716 2298 1     SUFFIX BOF,
717 2299 1     P_SUFFIX_LEN
718 2300 1 ) =
719 2301 1 **
720 2302 1 FUNCTIONAL DESCRIPTION:
721 2303 1
722 2304 1     This routine generates the escape sequences turning on and off
723 2305 1     attributes such as bolding and blinking. These attribute
724 2306 1     sequences are placed in two buffers supplied by the caller.
725 2307 1     No input text is specified.
726 2308 1
727 2309 1 CALLING SEQUENCE:
728 2310 1
729 2311 1     ret_status.wlc.v = COB$SET_ATTRIBUTES (TERM TYPE.rl.v,
730 2312 1     FLAGS.rl.v,
731 2313 1     PREFIX BUF.mt.r,
732 2314 1     P_PREFIX_LEN.ml.r,
733 2315 1     SUFFIX BOF.mt.r,
734 2316 1     P_SUFFIX_LEN.ml.r)
735 2317 1
736 2318 1 FORMAL PARAMETERS:
737 2319 1
738 2320 1     TERM TYPE.rl.v     terminal type
739 2321 1     FLAGS.rl.v        flags specifying which attributes to turn on
740 2322 1     PREFIX BUF.mt.r   addr of output buffer to receive prefix string
741 2323 1     P_PREFIX_LEN.ml.r # bytes in already in prefix buffer
742 2324 1                     gets updated to include size of prefix
743 2325 1     SUFFIX BUF.mt.r   addr of output buffer to receive suffix string
744 2326 1     P_SUFFIX_LEN.ml.r # bytes in already in suffix buffer
745 2327 1                     gets updated to include size of suffix
746 2328 1
747 2329 1 IMPLICIT INPUTS:
748 2330 1
749 2331 1     NONE
750 2332 1
751 2333 1 IMPLICIT OUTPUTS:
752 2334 1
753 2335 1     NONE
754 2336 1
755 2337 1 COMPLETION STATUS:
756 2338 1
757 2339 1
758 2340 1 SIDE EFFECTS:
759 2341 1
760 2342 1     NONE
761 2343 1 --

```

```

763 2344 2 BEGIN
764 2345 2
765 2346 2 BIND
766 2347 2
767 2348 2 PREFIX_LEN = .P_PREFIX_LEN, ! holds length of prefix buffer
768 2349 2 SUFFIX_LEN = .P_SUFFIX_LEN; ! holds length of suffix buffer
769 2350 2
770 2351 2 LOCAL
771 2352 2
772 2353 2 BUFFER_PTR;
773 2354 2
774 2355 2 MACRO
775 2356 2
776 2357 2 VT100_OFF = %STRING (%CHAR (ESC), %CHAR (LB), '0m')%;
777 2358 2
778 2359 2 BUFFER_PTR = .PREFIX_BUF + .PREFIX_LEN; ! init to first free byte of prefix
779 2360 2
780 2361 2 CASE .TERM_TYPE FROM UNKNOWN TO HARDCOPY OF
781 2362 2 SET
782 2363 2 [HARDCOPY, UNKNOWN, VT05, VT52, VTFORIGN]:
783 2364 2 BEGIN
784 2365 2 !+
785 2366 2 ! Renditions not supported on these devices. Just return.
786 2367 2 !-
787 2368 2 RETURN SSS_NORMAL
788 2369 2 END;
789 2370 2
790 2371 2 [INRANGE, OVRANGE]:
791 2372 2 RETURN 0; ! error
792 2373 2
793 2374 2 [VT100]:
794 2375 2 BEGIN
795 2376 2 IF .FLAGS <0,4> EQL 0
796 2377 2 THEN
797 2378 2 RETURN (SSS_NORMAL); ! no attributes requested
798 2379 2
799 2380 2 !+
800 2381 2 ! for each attribute bit set in flags, copy
801 2382 2 ! the appropriate ASCII graphic rendition byte
802 2383 2 ! followed by a ':' into the output buffer.
803 2384 2 ! Note use of autoincrementing.
804 2385 2 !-
805 2386 2
806 2387 2 CH$WCHAR_A (ESC, BUFFER_PTR);
807 2388 2 CH$WCHAR_A (LB, BUFFER_PTR);
808 2389 2 PREFIX_LEN = .PREFIX_LEN + 2; ! Start with 2 chars: <ESC> '['
809 2390 2 INCR I FROM 0 TO 3
810 2391 2 DO
811 2392 2 BEGIN ! build prefix attribute string
812 2393 2 BIND
813 2394 2 ATTRTABL = UPLIT (BYTE ('1754')) : VECTOR [4, BYTE];
814 2395 2
815 2396 2 IF .FLAGS <.I, 1>
816 2397 2 THEN
817 2398 2 BEGIN
818 2399 2 CH$WCHAR_A (.ATTRTABL[I], BUFFER_PTR);
819 2400 2 CH$WCHAR_A (%C',', BUFFER_PTR);
    
```

```

820      2401      5          PREFIX_LEN = .PREFIX_LEN + 2; ! keep updating length
821      2402      4          END;
822      2403      3          END; ! build prefix attribute string
823      2404      2
824      2405      1
825      2406      0
826      2407      0
827      2408      0
828      2409      0
829      2410      0
830      2411      0
831      2412      0
832      2413      0
833      2414      0
834      2415      0
835      2416      0
836      2417      0
837      2418      0
838      2419      0
839      2420      0
840      2421      0
841      2422      0
842      2423      0
843      2424      0
844      2425      0
845      2426      0
846      2427      0
847      2428      0
848      2429      0
849      2430      0
850      2431      0
851      2432      0
852      2433      1
    
```

! End of routine COB\$\$\$SET\_ATTRIBUTES\_ONLY

```

          34 35 37 31 001DA .BLKB 2
          6D 30 5B 1B 001DC P.AAR: .ASCII \1754\
          001E0 P.AAS: .ASCII <27>\[0m\
    
```

ATTRTABL= P.AAR

000E	51	0C	52	10	AC	D0	00002	.ENTRY	COB\$\$\$SET_ATTRIBUTES_ONLY, Save R2	2293
	05		AC	04	62	C1	00006	MOVL	P PREFIX_LEN, R2	2348
	0046		00		AC	CF	0000B	ADDL3	(R2), PREFIX_BUF, BUFFER_PTR	2359
	0046		0046		0046		00010	CASEL	TERM TYPE, #0, #5	2361
			0046		0046		00018	.WORD	5\$-1\$, -	
									5\$-1\$, -	
									5\$-1\$, -	
									2\$-1\$, -	
									5\$-1\$, -	
									5\$-1\$, -	
									6\$	2372
		0F	08	3C	11	0001C		BRB	6\$	2376
				AC	93	0001E	2\$:	BITB	FLAGS, #15	
				32	13	00022		BEQL	5\$	
		81	5B1B	8F	B0	00024		MOVW	#23323, (BUFFER_PTR)+	2387

		62		02	C0	00029		ADDL2	#2, (R2)	:	2389
				50	D4	0002C		CLRL	I	:	2390
0B	08	AC		50	E1	0002E	3\$:	BBC	I, FLAGS, 4\$	:	2396
		81	C1	AF	40	90	00033	MOVB	ATTRTABL[I], (BUFFER_PTR)+	:	2399
		81		3B	90	00038		MOVB	#59, (BUFFER_PTR)+	:	2400
		62		02	C0	0003B		ADDL2	#2, (R2)	:	2401
EC		50		03	F3	0003E	4\$:	AOBLEQ	#3, I, 3\$	:	2390
		71	6D	BF	90	00042		MOVB	#109, -(BUFFER_PTR)	:	2411
				51	D6	00046		INCL	BUFFER_PTR	:	
51	14	AC	18	BC	C1	00048		ADDL3	@P_SUFFIX_LEN, SUFFIX_BUF, BUFFER_PTR	:	2420
		61	AB	AF	D0	0004E		MOVL	P.AAS, (BUFFER_PTR)	:	2423
	18	BC		04	C0	00052		ADDL2	#4, @P_SUFFIX_LEN	:	2429
		50		01	D0	00056	5\$:	MOVL	#1, R0	:	2431
					04	00059		RET		:	
				50	D4	0005A	6\$:	CLRL	R0	:	2433
				04	0005C			RET		:	

: Routine Size: 93 bytes. Routine Base: \_COB\$CODE + 01E4

: 853 2434 1 !<BLF/PAGE>

```

855 2435 1 %SBTTL 'COB$$$SET_CURSOR_ABS_R4 - Create absolute set cursor sequence'
856 2436 1 GLOBAL ROUTINE COB$$$SET_CURSOR_ABS_R4 (
857 2437 1     TERM_TYPE,
858 2438 1     LINE_NO,
859 2439 1     COL_NO,
860 2440 1     BUFFER,
861 2441 1     CUR_SIZE
862 2442 1 ) : COB$$$ESC_R4_LNK =
863 2443 1
864 2444 1 ++
865 2445 1 FUNCTIONAL DESCRIPTION:
866 2446 1     This routine generates the escape sequence for a set cursor
867 2447 1     position and appends the string to a given output buffer.
868 2448 1
869 2449 1 CALLING SEQUENCE:
870 2450 1
871 2451 1     ret_status.wlc.v = COB$$$SET_CURSOR_ABS_R4 (TERM_TYPE.rl.v, LINE_NO.rl.v,
872 2452 1     COL_NO.rl.v, BUFFER.mt.r,
873 2453 1     CUR_SIZE.ml.r)
874 2454 1
875 2455 1 FORMAL PARAMETERS:
876 2456 1
877 2457 1     TERM_TYPE.rl.v     terminal type
878 2458 1     LINE_NO.rl.v      line number
879 2459 1     COL_NO.rl.v      column number
880 2460 1     BUFFER.mt.r      addr of buffer
881 2461 1                     this buffer should be at least
882 2462 1                     20 bytes
883 2463 1     CUR_SIZE.ml.r    # bytes currently in buffer
884 2464 1
885 2465 1 IMPLICIT INPUTS:
886 2466 1
887 2467 1     NONE
888 2468 1
889 2469 1 IMPLICIT OUTPUTS:
890 2470 1
891 2471 1     NONE
892 2472 1
893 2473 1 COMPLETION STATUS:
894 2474 1
895 2475 1
896 2476 1 SIDE EFFECTS:
897 2477 1
898 2478 1     NONE
899 2479 1 --
900 2480 1
901 2481 2 BEGIN
902 2482 2
903 2483 2 LOCAL
904 2484 2     VT100CTL : VECTOR [1, 8] INITIAL (
905 2485 2     DSC$K_CLASS_S ^24 + DSC$K_DTYPE_T ^16 + 10,
906 2486 2     UPLIT( BYTE (ESC, LB, '!OL;!UL', VT100_SC )))
907 2487 2     ! dsc for cvt to vt100 sequence
908 2488 2     ! FAO control string
909 2489 2     FREE_ADDR : REF VECTOR [,BYTE]; ! addr of 1st free byte
910 2490 2
911 2491 2
    
```

```

912 2492 2 FREE_ADDR = .BUFFER + ..CUR_SIZE; ! addr of next free byte
913 2493
914 2494 CASE TERM_TYPE FROM UNKNOWN TO HARDCOPY OF
915 2495 SET
916 2496
917 2497 [HARDCOPY, UNKNOWN, VTFOREIGN]:
918 2498 ; ! do nothing
919 2499
920 2500 [VT05]:
921 2501 BEGIN
922 2502 .CUR_SIZE = ..CUR_SIZE + 3; ! update current size of buffer
923 2503 FREE_ADDR [0] = VT05_SC; ! put set cursor sequence into buffer
924 2504 FREE_ADDR [1] = CB + .LINE_NO;
925 2505 FREE_ADDR [2] = CB + .COL_NO;
926 2506 END;
927 2507
928 2508 [VT52]:
929 2509 BEGIN
930 2510 .CUR_SIZE = ..CUR_SIZE + 4; ! update current size of buffer
931 2511 FREE_ADDR [0] = ESC; ! put set cursor sequence into buffer
932 2512 FREE_ADDR [1] = VT52_SC;
933 2513 FREE_ADDR [2] = CB + .LINE_NO;
934 2514 FREE_ADDR [3] = CB + .COL_NO;
935 2515 END;
936 2516
937 2517 [VT100]:
938 2518 BEGIN
939 2519 LOCAL
940 2520 STATUS,
941 2521 CVT_ARGS : VECTOR [2],
942 2522 FAO_BUFFER : BLOCK [8, BYTE],
943 2523 FAO_LEN : WORD;
944 2524
945 2525 CVT_ARGS [0] = .LINE_NO;
946 2526 CVT_ARGS [1] = .COL_NO;
947 2527 FAO_BUFFER [DSC$B_DTYPE] = DSC$K_DTYPE_T;
948 2528 FAO_BUFFER [DSC$B_CLASS] = DSC$K_CLASS_S;
949 2529 FAO_BUFFER [DSC$W_LENGTH] = 20; ! arbitrary - sb large enough
950 2530 FAO_BUFFER [DSC$A_POINTER] = .FREE_ADDR;
951 2531
952 2532 !+
953 2533 ! Convert to ASCII characters and move to buffer.
954 2534 !-
955 2535 P STATUS = $FAOL (CTRSTR = VT100CTL, OUTLEN = FAO_LEN,
956 2536 OUTBUF = FAO_BUFFER, PRMLST = CVT_ARGS);
957 2537 IF NOT STATUS THEN RETURN (.STATUS);
958 2538 .CUR_SIZE = ..CUR_SIZE + .FAO_LEN; ! add length of appended string
959 2539
960 2540 END;
961 2541
962 2542 [INRANGE, OTRANGE]:
963 2543 RETURN 0; ! should never get here
964 2544
965 2545 TES;
966 2546
967 2547 RETURN 1;
968 2548
    
```

: 969 2549 1 END; ! End of routine COB\$SET\_CURSOR\_ABS\_R4

Label	Address	Op	Operand 1	Operand 2	Operand 3	Comment	Address
	4C 55 21 3B 4C 5B 1B					P.AAT: .BLKB 3	
						.BYTE 27, 91	
						.ASCII \!UL;!UL\	
						.BYTE 102	
						.EXTRN SYSS\$FAOL	
	5E 1C C2 0000					COB\$SET_CURSOR_ABS_R4::	
	14 AE 010E000A	8F DO 00003				SUBL2 #28, SP	2436
	18 AE E8	AF 9E 0000B				MOVL #17694730, VT100CTL	2481
	53	64 CO 00010				MOVAB P.AAT, VT100CTL+4	
0034	00	50 CF 00013				ADDL2 (CUR_SIZE), FREE_ADDR	2492
	000E	0060 00017	1\$:			CASEL TERM_TYPE, #0, #5	2494
	0060	0060 0001F				.WORD 5\$-1\$,-	
						2\$-1\$,-	
						3\$-1\$,-	
						4\$-1\$,-	
						5\$-1\$,-	
						5\$-1\$	
						6\$	2543
	64	03 CO 00025	2\$:			ADDL2 #3, (CUR_SIZE)	2502
	63	0E 90 00028				MOVB #14, (FREE_ADDR)	2503
01 A3	51	1F 81 0002B				ADDB3 #31, LINE_NO, 1(FREE_ADDR)	2504
02 A3	52	1F 81 00030				ADDB3 #31, COL_NO, 2(FREE_ADDR)	2505
		40 11 00035				BRB 5\$	2494
	64	04 CO 00037	3\$:			ADDL2 #4, (CUR_SIZE)	2510
	63	8F 80 0003A				MOVW #22811, (FREE_ADDR)	2511
02 A3	51	1F 81 0003F				ADDB3 #31, LINE_NO, 2(FREE_ADDR)	2513
03 A3	52	1F 81 00044				ADDB3 #31, COL_NO, 3(FREE_ADDR)	2514
		2C 11 00049				BRB 5\$	2494
	0C AE	51 7D 0004B	4\$:			MOVQ LINE_NO, CVT_ARGS	2525
	04 AE	8F DO 0004F				MOVL #17694740, FAO_BUFFER	2529
	08 AE	53 DO 00057				MOVL FREE_ADDR, FAO_BUFFER+4	2530
		0C AE	9F 0005B			PUSHAB CVT_ARGS	2536
		08 AE	9F 0005E			PUSHAB FAO_BUFFER	
		08 AE	9F 00061			PUSHAB FAO_LEN	
		20 AE	9F 00064			PUSHAB VT100CTL	
	00000000G	00	04 FB 00067			CALLS #4, SYSS\$FAOL	
		0D	50 E9 0006E			BLBC STATUS, 7\$	2537
		50	6E 3C 00071			MOVZWL FAO_LEN, R0	2538
		64	50 CO 00074			ADDL2 R0, (CUR_SIZE)	
		50	01 DO 00077	5\$:		MOVL #1, R0	2547
			02 11 0007A			BRB 7\$	
		50	D4 0007C	6\$:		CLRL R0	2549
	5E	1C	CO 0007E	7\$:		ADDL2 #28, SP	
			05 00081			RSB	

: Routine Size: 130 bytes, Routine Base: \_COB\$CODE + 024E

: 970 2550 1 !<BLF/PAGE>



```

: 972 2551 1 %SBTTL 'COB$$$SET_CURSOR_REL Create relative cursor position sequence'
: 973 2552 1 GLOBAL ROUTINE COB$$$SET_CURSOR_REL (
: 974 2553 1     TERM_TYPE,
: 975 2554 1     LINE_NO,
: 976 2555 1     COL_NO,
: 977 2556 1     LINE_PLUS,
: 978 2557 1     COL_PLUS,
: 979 2558 1     BUFFER,
: 980 2559 1     CUR_SIZE
: 981 2560 1 ) =
: 982 2561 1
: 983 2562 1 **
: 984 2563 1 FUNCTIONAL DESCRIPTION:
: 985 2564 1     This routine generates the escape sequence to position
: 986 2565 1     the cursor relative to the specified line and column, or
: 987 2566 1     relative to the current position if none is specified.
: 988 2567 1     The set cursor sequence is appended to the output string.
: 989 2568 1
: 990 2569 1     Notice that the ANSI sequences can become quite large.
: 991 2570 1     For instance, it is possible that 50 up arrows (2 bytes each)
: 992 2571 1     will be only a part of the resulting sequence. It is
: 993 2572 1     recommended that the output buffer be 512 bytes long.
: 994 2573 1
: 995 2574 1 CALLING SEQUENCE:
: 996 2575 1
: 997 2576 1     ret_status.wlc.v = COB$$$SET_CURSOR_REL (TERM_TYPE.rl.v, LINE_NO.rl.v,
: 998 2577 1     COL_NO.rl.v, LINE_PLUS.rl.v,
: 999 2578 1     COL_PLUS.rl.v, BUFFER.mt.r,
: 1000 2579 1     CUR_SIZE.ml.r)
: 1001 2580 1
: 1002 2581 1 FORMAL PARAMETERS:
: 1003 2582 1
: 1004 2583 1     TERM_TYPE.rl.v     terminal type
: 1005 2584 1     LINE_NO.rl.v      line number
: 1006 2585 1     COL_NO.rl.v       column number
: 1007 2586 1     LINE_PLUS.rl.v   offset from line number
: 1008 2587 1     COL_PLUS.rl.v   offset from column number
: 1009 2588 1     BUFFER.mt.r      addr of buffer
: 1010 2589 1     CUR_SIZE.ml.r   # bytes currently in buffer
: 1011 2590 1
: 1012 2591 1 IMPLICIT INPUTS:
: 1013 2592 1
: 1014 2593 1     NONE
: 1015 2594 1
: 1016 2595 1 IMPLICIT OUTPUTS:
: 1017 2596 1
: 1018 2597 1     NONE
: 1019 2598 1
: 1020 2599 1 COMPLETION STATUS:
: 1021 2600 1
: 1022 2601 1
: 1023 2602 1 SIDE EFFECTS:
: 1024 2603 1
: 1025 2604 1     NONE
: 1026 2605 1
: 1027 2606 1
: 1028 2607 1

```

The following table shows the cursor positioning used for every

1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1060  
1061  
1062  
1063  
1064

2608 1 combination of the LINE and COLUMN phrases on both ANSI devices  
2609 1 and VT100s. The arrows on the VT52 can only be moved one position at  
2610 1 a time. This may be slower, but at least the results will be the  
2611 1 same as far as cursor positioning goes on both types of terminals.

2612 1  
2613 1 'v' = down arrow  
2614 1  
2615 1 '^' = up arrow  
2616 1

2617 1 LINE a : LINE PLUS b : COLUMN c : COLUMN PLUS d : Cursor Pos. Used

LINE a	LINE PLUS b	COLUMN c	COLUMN PLUS d	Cursor Pos. Used
N	N	N	N	Current Rules
N	N	N	Y	d "-->"
N	N	Y	N	<CR> : c-1 "-->"
N	N	Y	Y	<CR> : (c-1)+d "-->"
N	Y	N	N	b <LF>
N	Y	N	Y	b <LF> : d "-->"
N	Y	Y	N	b <LF> : <CR> : c-1 "-->"
N	Y	Y	Y	b <LF> : <CR> : (c-1)+d "-->"
Y	N	N	N	Home : a-1 "v"
Y	N	N	Y	24 '^' : a-1 "v" ; d "-->"
Y	N	Y	N	Direct a,c
Y	N	Y	Y	Direct a,c+d
Y	Y	N	N	Home : a-1 "v" ; b "LF"
Y	Y	N	Y	24 '^' : a-1 "v" ; b <LF>
Y	Y	Y	N	d "-->"
Y	Y	Y	Y	Direct a,c ; b <LF>
Y	Y	Y	Y	Direct a,c+d ; b <LF>

2638 1  
2639 1  
2640 1 note: <lf> for all LINE PLUS to get scrolling  
2641 1 note: 24 up arrows used instead of home - this maintains the current  
2642 1 column position  
2643 1

```

1066 2644 1
1067 2645 2
1068 2646 2
1069 2647 2
1070 2648 2
1071 2649 2
1072 2650 2
1073 2651 2
1074 2652 2
1075 2653 2
1076 M 2654 2
1077 M 2655 2
1078 M 2656 2
1079 M 2657 2
1080 M 2658 2
1081 M 2659 2
1082 M 2660 2
1083 M 2661 2
1084 M 2662 2
1085 M 2663 2
1086 M 2664 2
1087 M 2665 2
1088 M 2666 2
1089 M 2667 2
1090 M 2668 2
1091 M 2669 2
1092 M 2670 2
1093 M 2671 2
1094 M 2672 2
1095 M 2673 2
1096 M 2674 2
1097 M 2675 2
1098 M 2676 2
1099 M 2677 2
1100 M 2678 2
1101 M 2679 2
1102 M 2680 2
1103 M 2681 2
1104 M 2682 2
1105 M 2683 2
1106 M 2684 2
1107 M 2685 2
1108 M 2686 2
1109 M 2687 2
1110 M 2688 2
1111 M 2689 2
1112 M 2690 2
1113 M 2691 2
1114 M 2692 2
1115 M 2693 2
1116 M 2694 2
1117 M 2695 2
1118 M 2696 2
1119 M 2697 2
1120 M 2698 2
1121 M 2699 2
1122 M 2700 2

```

BEGIN

```

+
The following macro will put the VT100 sequence for
multiple arrow movement into the buffer and update
the length and pointer. Sequences are of the form
ESC [ num arrow.

```

MACRO

\$APPEND\_VT100\_SEQ (NUM, CTR\_ARROW) =

BEGIN

LOCAL

CVT\_ARG,

FAO\_BUF : BLOCK [8, BYTE],

FAO\_LEN : WORD,

STATUS;

IF NUM NEQ 0

THEN

BEGIN

CVT\_ARG = NUM;

FAO\_BUF [DSC\$B\_DTYPE] = DSC\$K\_DTYPE\_T;

FAO\_BUF [DSC\$B\_CLASS] = DSC\$K\_CLASS\_S;

FAO\_BUF [DSC\$W\_LENGTH] = 15;

FAO\_BUF [DSC\$A\_POINTER] = .FREE\_ADDR;

! arbitrary - sb big enough

STATUS = \$FAOL (CTRSTR = CTR\_ARROW, OUTLEN FAO\_LEN,

OUTBUF = FAO\_BUF, PRMLST = CVT\_ARG);

IF NOT .STATUS THEN RETURN .STATUS;

.CUR\_SIZE = ..CUR\_SIZE + .FAO\_LEN;

FREE\_ADDR = .FREE\_ADDR + .FAO\_LEN;

END;

END

! end macro \$append\_vt100\_seq

```

+
This macro puts NUM arrows into the buffer.
The next free byte and buffer size are updated.

```

MACRO

\$APPEND\_N\_ARROWS (NUM, DIRECTION) =

BEGIN

INCR COUNTER FROM 1 TO NUM DO

BEGIN

FREE\_ADDR = CH\$MOVE (2, UPLIT (BYTE (ESC, DIRECTION)), .FREE\_ADDR);

.CUR\_SIZE = ..CUR\_SIZE + 2;

END;

END;

! end of macro append\_n\_arrows

MACRO

\$APPEND\_VT100\_HOME =

BEGIN

FREE\_ADDR = CH\$MOVE (3, UPLIT (BYTE (ESC, LB, f)),

.FREE\_ADDR);

```

: 1123      M 2701 2      .CUR_SIZE = ..CUR_SIZE + 3;
: 1124      M 2702 2      END
: 1125      M 2703 2      x:
: 1126      M 2704 2
: 1127      M 2705 2      MACRO
: 1128      M 2706 2      $APPEND_VT52_HOME =
: 1129      M 2707 2      BEGIN
: 1130      M 2708 2      FREE_ADDR = CH$MOVE (2, UPLIT (BYTE (ESC, H)), .FREE_ADDR);
: 1131      M 2709 2      .CUR_SIZE = ..CUR_SIZE + 2;
: 1132      M 2710 2      END;
: 1133      M 2711 2      x:
: 1134      M 2712 2
: 1135      M 2713 2      LOCAL
: 1136      M 2714 2      FREE_ADDR : REF VECTOR [,BYTE],
: 1137      M 2715 2      UP_CTL : VECTOR [1, 8] INITIAL (
: 1138      M 2716 2      DSC$K_CLASS_S ^ 24 + DSC$K_DTYPE_T ^ 16 + 6,
: 1139      M 2717 2      UPLIT (BYTE (ESC, LB, '!UL', A))),
: 1140      M 2718 2      DOWN_CTL : VECTOR [1, 8] INITIAL (
: 1141      M 2719 2      DSC$K_CLASS_S ^ 24 + DSC$K_DTYPE_T ^ 16 + 6,
: 1142      M 2720 2      UPLIT (BYTE (ESC, LB, '!UL', B))),
: 1143      M 2721 2      RIGHT_CTL : VECTOR [1, 8] INITIAL (
: 1144      M 2722 2      DSC$K_CLASS_S ^ 24 + DSC$K_DTYPE_T ^ 16 + 6,
: 1145      M 2723 2      UPLIT (BYTE (ESC, LB, '!UL', C))),
: 1146      M 2724 2
: 1147      M 2725 2      BIND
: 1148      M 2726 2      UP = A,           ! equate letters to directions
: 1149      M 2727 2      DOWN = B,
: 1150      M 2728 2      RIGHT = C;
: 1151      M 2729 2
: 1152      M 2730 2      LITERAL
: 1153      M 2731 2      K_MAX_RMS_SIZE = 255;

```

```

1155 2732 2 IF .TERM_TYPE NEQ VT100 AND
1156 2733 2 .TERM_TYPE NEQ VT52
1157 2734 2 THEN RETURN (SS$_NORMAL); ! don't do anything for other
1158 2735 2 ! terminal types
1159 2736 2
1160 2737 2 FREE_ADDR = .BUFFER + ..CUR_SIZE;
1161 2738 2
1162 2739 2 IF .LINE_NO NEQ 0 AND
1163 2740 2 .COL_NO NEQ 0
1164 2741 2 THEN ! direct cursor addressing
1165 2742 2 BEGIN
1166 2743 2 COB$$$SET_CURSOR_ABS_R4 (.TERM_TYPE, .LINE_NO,
1167 2744 2 .COL_NO + .COL_PLUS, .BUFFER,
1168 2745 2 .CUR_SIZE);
1169 2746 2 FREE_ADDR = .BUFFER + ..CUR_SIZE; ! update addr next free byte
1170 2747 2 END;
1171 2748 2
1172 2749 2 IF .LINE_NO NEQ 0 AND
1173 2750 2 .COL_NO EQL 0
1174 2751 2 THEN
1175 2752 2 BEGIN
1176 2753 2 IF .COL_PLUS EQL 0
1177 2754 2 THEN ! insert home sequence
1178 2755 2 BEGIN
1179 2756 2 IF .TERM_TYPE EQL VT100
1180 2757 2 THEN
1181 2758 2 $APPEND_VT100_HOME
1182 2759 2 ELSE
1183 2760 2 $APPEND_VT52_HOME;
1184 2761 2 END
1185 2762 2 ELSE
1186 2763 2 BEGIN ! insert a bunch of up arrows
1187 2764 2 MACRO
1188 2765 2 UP_ARROW = %STRING (%CHAR (ESC), %CHAR (A))%;
1189 2766 2 BIND
1190 2767 2 UP_24 = UPLIT (BYTE (REP 24 OF (UP_ARROW)));
1191 2768 2
1192 2769 2 IF .TERM_TYPE EQL VT100
1193 2770 2 THEN
1194 2771 2 $APPEND_VT100_SEQ (24, UP_CTL)
1195 2772 2 ELSE
1196 2773 2 BEGIN
1197 2774 2 FREE_ADDR = CH$MOVE (48, UP_24, .FREE_ADDR);
1198 2775 2 .CUR_SIZE = ..CUR_SIZE + 48;
1199 2776 2 END;
1200 2777 2 END;
1201 2778 2
1202 2779 2 !+
1203 2780 2 !- Insert line_no down arrows regardless of col_plus
1204 2781 2
1205 2782 2 IF .TERM_TYPE EQL VT100
1206 2783 2 THEN
1207 2784 2 $APPEND_VT100_SEQ (.LINE_NO - 1, DOWN_CTL)
1208 2785 2 ELSE
1209 2786 2 $APPEND_N_ARROWS (.LINE_NO - 1, DOWN);
1210 2787 2 END;
1211 2788 2 IF .LINE_NO EQL 0 AND
    
```

```

: 1212      2789  2      .COL_NO NEQ 0
: 1213      2790  2      THEN
: 1214      2791  2      BEGIN
: 1215      2792  2      FREE_ADDR [0] = CR;
: 1216      2793  2      FREE_ADDR = .FREE_ADDR + 1;
: 1217      2794  2      .CUR_SIZE = ..CUR_SIZE + 1;
: 1218      2795  2      END;
: 1219      2796  2
: 1220      2797  2      IF .LINE_PLUS NEQ 0
: 1221      2798  2      THEN
: 1222      2799  2      BEGIN
: 1223      2800  2      FREE_ADDR = CH$FILL (LF, .LINE_PLUS, .FREE_ADDR);
: 1224      2801  2      .CUR_SIZE = ..CUR_SIZE + .LINE_PLUS;
: 1225      2802  2      END;
: 1226      2803  2
: 1227      2804  2      IF (.COL_PLUS NEQ 0 OR .COL_NO NEQ 0) AND
: 1228      2805  2      (.LINE_NO EQL 0 OR .COL_NO EQL 0) ! didn't do direct cursor addr
: 1229      2806  2      THEN
: 1230      2807  2      BEGIN
: 1231      2808  2      LOCAL
: 1232      2809  2      COL;
: 1233      2810  2      COL = .COL_NO - 1;
: 1234      2811  2      IF .COL LSS 0
: 1235      2812  2      THEN
: 1236      2813  2      COL = 0;
: 1237      2814  2      IF .TERM_TYPE EQL VT100
: 1238      2815  2      THEN
: 1239      2816  2      $APPEND_VT100_SEQ (.COL + .COL_PLUS, RIGHT_CTL)
: 1240      2817  2      ELSE
: 1241      2818  2      $APPEND_N_ARROWS (.COL + .COL_PLUS, RIGHT);
: 1242      2819  2      END;
: 1243      2820  2
: 1244      2821  2      RETURN (SS$_NORMAL);
: 1245      2822  2
: 1246      2823  1      END;

```

4C	5B	1B	002D0	P.AAU:	.BYTE	27, 91
	55	21	002D2		.ASCII	\!UL\
		41	002D5		.BYTE	65
			002D6		.BLKB	2
4C	5B	1B	002D8	P.AAV:	.BYTE	27, 91
	55	21	002DA		.ASCII	\!UL\
		42	002DD		.BYTE	66
			002DE		.BLKB	2
4C	5B	1B	002E0	P.AAW:	.BYTE	27, 91
	55	21	002E2		.ASCII	\!UL\
		43	002E5		.BYTE	67
			002E6		.BLKB	2
66	5B	1B	002E8	P.AAX:	.BYTE	27, 91, 102
			002EB		.BLKB	1
	48	1B	002EC	P.AAY:	.BYTE	27, 72
			002EE		.BLKB	2
	41	1B	002F0	P.AAZ:	.ASCII	<27>\A\
	41	1B	002F2		.ASCII	<27>\A\
	41	1B	002F4		.ASCII	<27>\A\

.....

```

41 1B 002F6 .ASCII <27>\A\
41 1B 002F8 .ASCII <27>\A\
41 1B 002FA .ASCII <27>\A\
41 1B 002FC .ASCII <27>\A\
41 1B 002FE .ASCII <27>\A\
41 1B 00300 .ASCII <27>\A\
41 1B 00302 .ASCII <27>\A\
41 1B 00304 .ASCII <27>\A\
41 1B 00306 .ASCII <27>\A\
41 1B 00308 .ASCII <27>\A\
41 1B 0030A .ASCII <27>\A\
41 1B 0030C .ASCII <27>\A\
41 1B 0030E .ASCII <27>\A\
41 1B 00310 .ASCII <27>\A\
41 1B 00312 .ASCII <27>\A\
41 1B 00314 .ASCII <27>\A\
41 1B 00316 .ASCII <27>\A\
41 1B 00318 .ASCII <27>\A\
41 1B 0031A .ASCII <27>\A\
41 1B 0031C .ASCII <27>\A\
41 1B 0031E .ASCII <27>\A\
42 1B 00320 P.ABA: .BYTE 27, 66
      00322 .BLKB 2
43 1B 00324 P.ABB: .BYTE 27, 67

UP= 65
DOWN= 66
RIGHT= 67
UP_24= P.AAZ
    
```

```

OFFC 00000 .ENTRY COB$SET_CURSOR_REL, Save R2,R3,R4,R5,R6,- ; 2552
5B 00000000G 00 9E 00002 MOVAB R7,R8,R9,R10,R11
5A 9E AF 9E 00009 MOVAB SY$FAOL, R11
5E 38 C2 0000D SUBL2 P.AAU, R10
30 AE 010E0006 8F D0 00010 MOVL #56, SP
34 AE 010E0006 6A 9E 00018 MOVL #17694726, UP_CTL ; 2645
28 AE 010E0006 8F D0 0001C MOVAB P.AAU, UP_CTL+4
2C AE 08 AA 9E 00024 MOVL #17694726, DOWN_CTL
20 AE 010E0006 8F D0 00029 MOVAB P.AAV, DOWN_CTL+4
24 AE 10 AA 9E 00031 MOVL #17694726, RIGHT_CTL
59 04 AC D0 00036 MOVAB P.AAW, RIGHT_CTL+4
03 59 D1 0003A MOVL TERM_TYPE, R9 ; 2732
08 13 0003D CMPL R9, #3
02 59 D1 0003F BEQL 1$ ; 2733
03 13 00042 CMPL R9, #2
0181 31 00044 BEQL 1$
56 1C AC D0 00047 1$: MOVL 23$ ; 2737
55 18 AC 66 C1 0004B ADDL3 CUR_SIZE, R6
57 08 AC D0 00050 MOVL (R6), BUFFER, FREE_ADDR ; 2739
58 D4 00054 CLRL LINE_NO, R7
57 D5 00056 TSTL R8
22 13 00058 BEQL R7
58 D6 0005A INCL R7
0C AC D5 0005C TSTL COL_NO ; 2740
1B 13 0005F BEQL 2$
    
```

52	0C	AC	14	AC	C1	00061	ADDL3	COL_PLUS, COL_NO, R2	2744	
		54		56	DO	00067	MOVL	R6, R4	2743	
		53	18	AC	DO	0006A	MOVL	BUFFER, R3		
		51		57	DO	0006E	MOVL	R7, R1		
		50		59	DO	00071	MOVL	R9, R0		
				FEB1	30	00074	BSBW	COB\$\$SET_CURSOR_ABS R4		
55	18	AC		66	C1	00077	ADDL3	(R6), BUFFER, FREE_ADDR	2746	
		03		58	E8	0007C	2\$:	BLBS	R8, 4\$	2749
				00B9	31	0007F	3\$:	BRW	14\$	
			0C	AC	D5	00082	4\$:	TSTL	COL_NO	2750
				F8	12	00085		BNEQ	3\$	
			14	AC	D5	00087		TSTL	COL_PLUS	2753
				20	12	0008A		BNEQ	6\$	
				58	D4	0008C		CLRL	R8	2756
			03	59	D1	0008E		CMPL	R9, #3	
				10	12	00091		BNEQ	5\$	
				58	D6	00093		INCL	R8	
85	18	00	18	AA	F0	00095	INSV	P.AAX, #0, #24, (FREE_ADDR)+	2757	
		55		02	C0	0009B	ADDL2	#2, FREE_ADDR		
		66		03	C0	0009E	ADDL2	#3, (R6)		
				4D	11	000A1	BRB	8\$	2756	
		85	1C	AA	B0	000A3	5\$:	MOVW	P.AAY, (FREE_ADDR)+	2759
		66		02	C0	000A7	ADDL2	#2, (R6)		
				44	11	000AA	BRB	8\$	2753	
				58	D4	000AC	6\$:	CLRL	R8	2769
			03	59	D1	000AE		CMPL	R9, #3	
				32	12	000B1		BNEQ	7\$	
				58	D6	000B3		INCL	R8	
		6E		18	DO	000B5	MOVL	#24, CVT_ARG	2771	
	18	AE	010E000F	8F	DO	000B8	MOVL	#17694735, FAO_BUF		
	1C	AE		55	DO	000C0	MOVL	FREE_ADDR, FAO_BUF+4		
				5E	DD	000C4	PUSHL	SP		
			1C	AE	9F	000C6	PUSHAB	FAO_BUF		
			0C	AE	9F	000C9	PUSHAB	FAO_LEN		
			3C	AE	9F	000CC	PUSHAB	UP_CTL		
		6B		04	FB	000CF	CALLS	#4, SYSSFAOL		
		43		50	E9	000D2	BLBC	STATUS, 9\$		
		50	04	AE	3C	000D5	MOVZWL	FAO_LEN, R0		
		66		50	C0	000D9	ADDL2	R0, (R6)		
		50	04	AE	3C	000DC	MOVZWL	FAO_LEN, R0		
		55		50	C0	000E0	ADDL2	R0, FREE_ADDR		
				0B	11	000E3	BRB	8\$	2769	
65	20	AA		30	28	000E5	7\$:	MOV3	#48, UP 24, (FREE_ADDR)	2774
		55		53	DO	000EA	MOVL	R3, FREE_ADDR		
		66		30	C0	000ED	ADDL2	#48, (R6)	2775	
		39		58	E9	000F0	8\$:	BLBC	R8, 11\$	2781
		01		57	D1	000F3		CMPL	R7, #1	2783
				43	13	000F6		BEQL	14\$	
	08	AE	FF	A7	9E	000F8	MOVAB	-1(R7), CVT_ARG		
	18	AE	010E000F	8F	DO	000FD	MOVL	#17694735, FAO_BUF		
	1C	AE		55	DO	00105	MOVL	FREE_ADDR, FAO_BUF+4		
			08	AE	9F	00109	PUSHAB	CVT_ARG		
			1C	AE	9F	0010C	PUSHAB	FAO_BUF		
			14	AE	9F	0010F	PUSHAB	FAO_LEN		
			34	AE	9F	00112	PUSHAB	DOWN_CTL		
		6B		04	FB	00115	CALLS	#4, SYSSFAOL		
		01		50	E8	00118	9\$:	BLBS	STATUS, 10\$	





```

        F5          66          02 C0 001C'      ADDL2 #2, (R6)
                   51          50 F3 001C4 22$:  AOBLEQ R0, COUNTER, 21$
                   50          01 D0 001C8 23$:  MOVL #1, R0
                   04 001CB 24$:  RET
  
```

```

:
:
: 2821
: 2823
  
```

: Routine Size: 460 bytes, Routine Base: \_COB\$CODE + 0326

: 1247 2824 1 !<BLF/PAGE>

```

: 1249 2825 1 %SBTTL 'COB$$$SETUP TERM TYPE - Setup terminal type for COB$$ routines'
: 1250 2826 1 GLOBAL ROUTINE COB$$$SETUP_TERM_TYPE (
: 1251 2827 1     FILE_NAME,
: 1252 2828 1     NAME_LEN,
: 1253 2829 1     TERM_TYPE,
: 1254 2830 1     SEC_DEV_CHAR,
: 1255 2831 1     DEVICE_TYPE : REF VECTOR [,BYTE],
: 1256 2832 1     RES_NAME_LEN : REF VECTOR [,WORD],
: 1257 2833 1     RES_NAME_ADDR
: 1258 2834 1 ) =
: 1259 2835 1 **
: 1260 2836 1 : FUNCTIONAL DESCRIPTION:
: 1261 2837 1 :
: 1262 2838 1 :     This routine uses the specified file name to determine device
: 1263 2839 1 :     characteristics and assign a terminal type code which is understood
: 1264 2840 1 :     by other COB$$ routines. COB$$ routines use the terminal type to
: 1265 2841 1 :     determine the correct escape sequence for a given function (ex. set
: 1266 2842 1 :     cursor).
: 1267 2843 1 :
: 1268 2844 1 : CALLING SEQUENCE:
: 1269 2845 1 :
: 1270 2846 1 :     ret_status.wlc.v = COB$$$SETUP_TERM_TYPE (FILE_NAME.rt.r,
: 1271 2847 1 :     NAME_LEN.rl.v,
: 1272 2848 1 :     TERM_TYPE.wl.r
: 1273 2849 1 :     [,SEC_DEV_CHAR.wlu.r]
: 1274 2850 1 :     [,DEVICE_TYPE.wbu.r]
: 1275 2851 1 :     [,RES_NAME_LEN.wwu.r]
: 1276 2852 1 :     RES_NAME_ADDR.wt.r])
: 1277 2853 1 :
: 1278 2854 1 : FORMAL PARAMETERS:
: 1279 2855 1 :
: 1280 2856 1 :     FILE_NAME.rt.r      addr of file name text
: 1281 2857 1 :     NAME_LEN.rl.v      length of file name text
: 1282 2858 1 :     TERM_TYPE.wl.r     terminal type code, one of the following:
: 1283 2859 1 :                       unknown
: 1284 2860 1 :                       vt05
: 1285 2861 1 :                       vt52
: 1286 2862 1 :                       vt100
: 1287 2863 1 :                       vtforeign
: 1288 2864 1 :                       hardcopy
: 1289 2865 1 :
: 1290 2866 1 :     SEC_DEV_CHAR.wlu.r [Optional] If supplied, the address of
: 1291 2867 1 :     a longword to receive the secondary
: 1292 2868 1 :     device dependent bits. This is the
: 1293 2869 1 :     field that, e.g. tells whether a VT100
: 1294 2870 1 :     has AVO.
: 1295 2871 1 :
: 1296 2872 1 :     DEVICE_TYPE.wbu.r [Optional]. If present, address of byte
: 1297 2873 1 :     to receive hardware device type. These
: 1298 2874 1 :     are the DTS_type codes.
: 1299 2875 1 :
: 1300 2876 1 :     RES_NAME_LEN.wwu.r [Optional -- if provided, RES_NAME_ADDR
: 1301 2877 1 :     must be provided as well.] If present,
: 1302 2878 1 :     the address of a word to receive the
: 1303 2879 1 :     length of the resultant name string.
: 1304 2880 1 :
: 1305 2881 1 :     RES_NAME_ADDR.wt.r [Optional -- if provided, RES_NAME_LEN
    
```

L 13

```

1306 2882 1 |
1307 2883 1 |
1308 2884 1 |
1309 2885 1 |
1310 2886 1 |
1311 2887 1 |
1312 2888 1 |
1313 2889 1 |
1314 2890 1 |
1315 2891 1 | IMPLICIT INPUTS.
1316 2892 1 |     NONE
1317 2893 1 |
1318 2894 1 |
1319 2895 1 | IMPLICIT OUTPUTS:
1320 2896 1 |     NONE
1321 2897 1 |
1322 2898 1 |
1323 2899 1 | COMPLETION STATUS:
1324 2900 1 |
1325 2901 1 |
1326 2902 1 | SIDE EFFECTS:
1327 2903 1 |     NONE
1328 2904 1 |
1329 2905 1 |
1330 2906 1 |
1331 2907 2 | BEGIN
1332 2908 2 |
1333 2909 2 | BUILTIN
1334 2910 2 |     NULLPARAMETER;
1335 2911 2 |
1336 2912 2 | LOCAL
1337 2913 2 |     DEVNAM DSC : BLOCK [8, BYTE], | dsc for name
1338 2914 2 |     DVI_ITMLST : VECTOR [3*3 + 1] INITIAL | item list for $GETDVI
1339 2915 2 |         (DVI$_DEVTYPE ^ 16 + 4, 0, 0, | device type
1340 2916 2 |         DVI$_DEVDEPEND2 ^ 16 + 4, 0, 0, | device dependent bits
1341 2917 2 |         DVI$_DEVNAM ^ 16 + 64, 0, 0, | result name string
1342 2918 2 |         0), | terminator
1343 2919 2 |
1344 2920 2 |     DVI_EFN, | event flag for $GETDVI,
1345 2921 2 |     STATUS, | status retd by called routines
1346 2922 2 |     DEV_TYPE : VOLATILE, | storage for $GETDVI value
1347 2923 2 |     DEV_DEPEND2 : VOLATILE, | storage for $GETDVI value
1348 2924 2 |
1349 2925 2 |     DEV_DEVNAM : VECTOR [64, BYTE], | Buffer for result name
1350 2926 2 | | string
1351 2927 2 |
1352 2928 2 |     DEV_NAMLEN : VOLATILE WORD; | Length of returned
1353 2929 2 | | resultant name string
1354 2930 2 | BIND
1355 2931 2 |     DVI_TYPE = DVI_ITMLST + 4, | make it easy to reference
1356 2932 2 |     DVI_DEPEND2 = DVI_ITMLST + 16, | items retd by $GETDVI
1357 2933 2 |     DVI_DEVNAM = DVI_ITMLST + 28, |
1358 2934 2 |     DVI_NAMLEN = DVI_ITMLST + 32; |
1359 2935 2 |
1360 2936 2 | MAP
1361 2937 2 |     DEV_DEPEND2 : BLOCK [4, BYTE];
1362 2938 2 |
    
```

must be provided as well.] If present, the address of a buffer to receive the resultant name string. NOTE: This routine assumes that the supplied buffer is large enough to contain the resultant name string. It must be a minimum of 4 bytes long and should be at least 64 bytes long to guarantee that the name will fit.

```

1363 2939 2 DVI_TYPE = DEV_TYPE; ! fill in rest of itmlst
1364 2940 2 DVI_DEPEND2 = DEV_DEPEND2;
1365 2941 2 DVI_DEVNAM = DEV_DEVNAM;
1366 2942 2 DVI_NAMLEN = DEV_NAMLEN;
1367 2943 2
1368 2944 2 ! IF NOT (STATUS = LIB$GET_EF (DVI_EFN))
1369 2945 2 THEN RETURN (.STATUS); ! get unique event flag number
1370 2946 2
1371 2947 2 DEVNAM_DSC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
1372 2948 2 DEVNAM_DSC [DSC$B_CLASS] = DSC$K_CLASS_S;
1373 2949 2 DEVNAM_DSC [DSC$W_LENGTH] = .NAME_LEN;
1374 2950 2 DEVNAM_DSC [DSC$A_POINTER] = .FILE_NAME; ! dsc needed for $GETDVI
1375 2951 2
1376 P 2952 2 STATUS = $GETDVI (EFN = .DVI_EFN, DEVNAM = DEVNAM_DSC,
1377 2953 2 ITMLST = DVI_ITMLST);
1378 2954 2 IF NOT .STATUS THEN RETURN (.STATUS);
1379 2955 2
1380 2956 2 $WAITFR (EFN = .DVI_EFN); ! make $GETDVI synchronous
1381 2957 2
1382 2958 2 IF NOT (STATUS = LIB$FREE_EF (DVI_EFN))
1383 2959 2 THEN RETURN (.STATUS); ! free event flag
1384 2960 2
1385 2961 2 SELECTONE .DEV_TYPE OF
1386 2962 2 SET
1387 2963 2 [DTS_VT100]:
1388 2964 2 .TERM_TYPE = VT100;
1389 2965 2
1390 2966 2 [DTS_VT52, DTS_VT55]:
1391 2967 2 .TERM_TYPE = VT52;
1392 2968 2
1393 2969 2 [DTS_VT05]:
1394 2970 2 .TERM_TYPE = VT05;
1395 2971 2
1396 2972 2 [DTS_FT1 TO DTS_FT2]:
1397 2973 2 .TERM_TYPE = VTFOREIGN;
1398 2974 2
1399 2975 2 [DTS_LA36, DTS_LA120, DTS_LA34, DTS_LA38]:
1400 2976 2 .TERM_TYPE = HARDCOPY;
1401 2977 2
1402 2978 2 [OTHERWISE]:
1403 2979 2 IF .DEV_DEPEND2 [TT2$V_DECCRT] OR
1404 2980 2 .DEV_DEPEND2 [TT2$V_ANSICRT]
1405 2981 2 THEN
1406 2982 2 .TERM_TYPE = VT100 ! VT100 compatible (ANSI)
1407 2983 2 ELSE
1408 2984 2 .TERM_TYPE = UNKNOWN; ! really unknown
1409 2985 2
1410 2986 2 TES;
1411 2987 2
1412 2988 2 ! Return optional parameters if requested.
1413 2989 2
1414 2990 2 IF NOT NULLPARAMETER (4)
1415 2991 2 THEN
1416 2992 2 .SEC_DEV_CHAR = .DEV_DEPEND2;
1417 2993 2
1418 2994 2 IF NOT NULLPARAMETER (5)
1419 2995 2 THEN
    
```

```

: 1420      2996      2      DEVICE_TYPE [0] = .DEV_TYPE;
: 1421      2997      2
: 1422      2998      2
: 1423      2999      2
: 1424      3000      2
: 1425      3001      2
: 1426      3002      2
: 1427      3003      2
: 1428      3004      2
: 1429      3005      2
: 1430      3006      2
: 1431      3007      2
    IF NOT NULLPARAMETER (6) AND
    NOT NULLPARAMETER (7)
    THEN
    BEGIN
    CH$MOVE ( .DEV_NAMLEN, DEV_DEVNAM, .RES_NAME_ADDR);
    RES_NAME_LEN [0] = .DEV_NAMLEN;
    END;
    RETURN (.STATUS);
    END;
    ! End of routine COB$SETUP_TERM_TYPE
    
```

00000000	00000000	001C0004	00000000	00000000	00060004	004F2	P.ABC:	.BLKB	2	393220, 0, 0, 1835012, 0, 0, 2097216, 0, -	:
		00000000	00000000	00000000	00200040	004F4		.LONG		0, 0	:
						0050C		.EXTRN		SYSSGETDVI, SYSSWAITFR	
						007C		.ENTRY		COB\$SETUP_TERM_TYPE, Save R2,R3,R4,R5,R6	2826
	50	AE	CE	5E	80	AE	9E	MOVAB		-128(SP), SP	
			54	AF		28	28	MOVAB		#40, P.ABC, DVI_ITMLST	2918
			60	AE	4C	AE	9E	MOVAB		DEV_TYPE, DVI_TYPE	2939
			6C	AE	48	AE	9E	MOVAB		DEV_DEPEND2, DVI_DEPEND2	2940
			70	AE	08	AE	9E	MOVAB		DEV_DEVNAM, DVI_DEVNAM	2941
					06	AE	9E	MOVAB		DEV_NAMLEN, DVI_NAMLEN	2942
						5E	DD	PUSHL		SP	2944
		00000000G	00			01	FB	CALLS		#1, LIB\$GET_EF	
			56			50	D0	MOVL		R0, STATUS	
			41			56	E9	BLBC		STATUS, 1\$	
			7A	AE	010E	8F	B0	MOVW		#270, DEVNAM_DSC+2	2947
			78	AE	08	AC	B0	MOVW		NAME_LEN, DEVNAM_DSC	2949
			7C	AE	04	AC	D0	MOVL		FILE_NAME, DEVNAM_DSC+4	2950
						7E	7C	CLRQ		-(SP)	2953
						7E	7C	CLRQ		-(SP)	
					60	AE	9F	PUSHAB		DVI_ITMLST	
					F8	AD	9F	PUSHAB		DEVNAM_DSC	
						7E	D4	CLRL		-(SP)	
					1C	AE	DD	PUSHL		DVI_EFN	
		00000000G	00			08	FB	CALLS		#8, SYSSGETDVI	
			56			50	D0	MOVL		R0, STATUS	
			15			56	E9	BLBC		STATUS, 1\$	2954
						6E	DD	PUSHL		DVI_EFN	2956
		00000000G	00			01	FB	CALLS		#1, SYSSWAITFR	
			00			5E	DD	PUSHL		SP	2958
		00000000G	00			01	FB	CALLS		#1, LIB\$FREE_EF	
			56			50	D0	MOVL		R0, STATUS	
			03			56	E8	BLBS		STATUS, 2\$	
						009C	31	BRW		12\$	
			50		4C	AE	D0	MOVL		DEV_TYPE, R0	2961
		00000060	8F			50	D1	CMPL		R0, #96	2963
						48	13	BEQL		7\$	
			3F			50	D1	CMPL		R0, #63	2966
						0F	15	BLEQ		3\$	
		00000041	8F			50	D1	CMPL		R0, #65	

				06	14	0008F	BGTR	3\$			
	0C	BC		02	D0	00091	MOVL	#2,	@TERM_TYPE		2967
				3D	11	00095	BRB	9\$			
		01		50	D1	00097	3\$:	CMPL	R0, #1		2969
				06	12	0009A	BNEQ	4\$			
	0C	BC		01	D0	0009C	MOVL	#1,	@TERM_TYPE		2970
				32	11	000A0	BRB	9\$			
		10		50	D1	000A2	4\$:	CMPL	R0, #16		2972
				08	19	000A5	BLSS	5\$			
		11		50	D1	000A7	CMPL	R0, #17			
				06	14	000AA	BGTR	5\$			
	0C	BC		04	D0	000AC	MOVL	#4,	@TERM_TYPE		2973
				22	11	000B0	BRB	9\$			
		20		50	D1	000B2	5\$:	CMPL	R0, #32		2975
				08	19	000B5	BLSS	6\$			
		23		50	D1	000B7	CMPL	R0, #35			
				06	14	000BA	BGTR	6\$			
	0C	BC		05	D0	000BC	MOVL	#5,	@TERM_TYPE		2976
				12	11	000C0	BRB	9\$			
04		4B	AE	05	E0	000C2	6\$:	BBS	#5, DEV_DEPEND2+3, 7\$		2979
			4B	AE	E9	000C7	BLBC	DEV_DEPEND2+3, 8\$			2980
	0C	BC		03	D0	000CB	7\$:	MOVL	#3, @TERM_TYPE		2982
				03	11	000CF	BRB	9\$			
			0C	BC	D4	000D1	8\$:	CLRL	@TERM_TYPE		2984
		04		6C	91	000D4	9\$:	CMPB	(AP), #4		2990
				0A	1F	000D7	BLSSU	10\$			
			10	AC	D5	000D9	TSTL	16(AP)			
				05	13	000DC	BEQL	10\$			
	10	BC	48	AE	D0	000DE	10\$:	MOVL	DEV_DEPEND2, @SEC_DEV_CHAR		2992
		05		6C	91	000E3	CMPB	(AP), #5			2994
				0A	1F	000E6	BLSSU	11\$			
			14	AC	D5	000E8	TSTL	20(AP)			
				05	13	000EB	BEQL	11\$			
	14	BC	4C	AE	90	000ED	11\$:	MOVB	DEV_TYPE, @DEVICE_TYPE		2996
		06		6C	91	000F2	CMPB	(AP), #6			2998
				1B	1F	000F5	BLSSU	12\$			
			18	AC	D5	000F7	TSTL	24(AP)			
				16	13	000FA	BEQL	12\$			
		07		6C	91	000FC	CMPB	(AP), #7			2999
				11	1F	000FF	BLSSU	12\$			
			1C	AC	D5	00101	TSTL	28(AP)			
				0C	13	00104	BEQL	12\$			
	1C	BC	08	AE	28	00106	MOV3	DEV_NAMLEN, DEV_DEVNAM, @RES_NAME_ADDR			3002
			18	BC	06	AE	MOVW	DEV_NAMLEN, @RES_NAME_LEN			3003
				50	D0	00112	12\$:	MOVL	STATUS, R0		3006
				04	00	00115	RET				3007

; Routine Size: 278 bytes, Routine Base: \_COB\$CODE + 051C

; 1432 3008 1 !<BLF/PAGE>

```

1434 3009 1 %SBTTL 'COB$UP_SCROLL_R2 - Create up scroll sequence'
1435 3010 1 GLOBAL ROUTINE COB$UP_SCROLL_R2 (
1436 3011 1     TERM_TYPE,
1437 3012 1     BUFFER,
1438 3013 1     CUR_SIZE
1439 3014 1 ) : COB$ESC_R2_LNK =
1440 3015 1 --
1441 3016 1 FUNCTIONAL DESCRIPTION:
1442 3017 1
1443 3018 1     This routine generates the escape sequence for up scroll.
1444 3019 1     The string is appended into the buffer.
1445 3020 1
1446 3021 1 CALLING SEQUENCE:
1447 3022 1
1448 3023 1     ret_status.wlc.v = COB$UP_SCROLL_R2 (TERM_TYPE.rl.v, BUFFER.mt.r,
1449 3024 1     CUR_SIZE.ml.r)
1450 3025 1
1451 3026 1 FORMAL PARAMETERS:
1452 3027 1
1453 3028 1     TERM_TYPE.rl.v     terminal type
1454 3029 1     BUFFER.mt.r        addr of buffer
1455 3030 1     CUR_SIZE.ml.r      # bytes currently in buffer
1456 3031 1
1457 3032 1 IMPLICIT INPUTS:
1458 3033 1
1459 3034 1     NONE
1460 3035 1
1461 3036 1 IMPLICIT OUTPUTS:
1462 3037 1
1463 3038 1     NONE
1464 3039 1
1465 3040 1 COMPLETION STATUS:
1466 3041 1
1467 3042 1
1468 3043 1 SIDE EFFECTS:
1469 3044 1
1470 3045 1     NONE
1471 3046 1 --
1472 3047 1
1473 3048 2 BEGIN
1474 3049 2
1475 3050 2 LOCAL
1476 3051 2     FREE_ADDR : REF VECTOR [,BYTE];
1477 3052 2
1478 3053 2     FREE_ADDR = .BUFFER + ..CUR_SIZE;
1479 3054 2
1480 3055 2 CASE .TERM_TYPE FROM UNKNOWN TO HARDCOPY OF
1481 3056 2 SET
1482 3057 2     [VT05]:
1483 3058 2     BEGIN
1484 3059 2     FREE_ADDR [0] = LF;
1485 3060 2     FREE_ADDR [1] = NULL;
1486 3061 2     FREE_ADDR [2] = NULL;
1487 3062 2     FREE_ADDR [3] = NULL;
1488 3063 2     .CUR_SIZE = ..CUR_SIZE + 4;
1489 3064 2 END;
1490 3065 2
    
```



```

: 1491      3066 2      [VT52, VT100]:
: 1492      3067      BEGIN
: 1493      3068      FREE_ADDR [0] = LF;
: 1494      3069      .CUR_SIZE = ..CUR_SIZE + 1;
: 1495      3070      END;
: 1496      3071
: 1497      3072      [HARDCOPY, UNKNOWN, VTFORIGN]:
: 1498      3073      ;
: 1499      3074
: 1500      3075      [INRANGE, OTRANGE]:
: 1501      3076      RETURN 0;          ! should never get here
: 1502      3077
: 1503      3078      TES;
: 1504      3079
: 1505      3080      RETURN (SS$_NORMAL);
: 1506      3081
: 1507      3082      END;          ! End of routine COB$SUP_SCROLL_R2
    
```

		51	62	C0 00000	COB\$SUP_SCROLL_R2::		
					ADDL2 (CUR_SIZE), FREE_ADDR		3053
0016	05	00	50	CF 00003	CASEL TERM_TYPE, #0, #5		3055
	0016	000E	001B	00007	.WORD 4\$-1\$,-		
		001B	001B	0000F	2\$-1\$,-		
					3\$-1\$,-		
					3\$-1\$,-		
					4\$-1\$,-		
					4\$-1\$		
			11	11 00013	BRB 5\$		3076
		61	0A	D0 00015	2\$: MOVL #10, (FREE_ADDR)		3059
		62	04	C0 00018	ADDL2 #4, (CUR_SIZE)		3063
			05	11 0001B	BRB 4\$		3055
		61	0A	90 0001D	3\$: MOVB #10, (FREE_ADDR)		3068
			62	D6 00020	INCL (CUR_SIZE)		3069
		50	01	D0 00022	4\$: MOVL #1, R0		3080
				05 00025	RSB		
			50	D4 00026	5\$: CLRL R0		3082
				05 00028	RSB		

: Routine Size: 41 bytes, Routine Base: \_COB\$CODE + 0632

: 1508 3083 1 !<BLF/PAGE>

```

COB$$$ESCAPE_GEN COB$$$ESCAPE GENERATOR - Escape sequence generat 16-Sep-1984 00:06:34 VAX-11 Bliss-32 V4.0-742
1-003 COB$$$UP_SCROLL_R2 - Create up scroll sequence 14-Sep-1984 12:10:44 [COBRTL.SRC]COBESCGEN.B32;1 Page 48
: 1510 3084 1 END . End of module COB$$$ESCAPE_GENERATOR
: 1511 3085 1
: 1512 3086 0 ELUDOM

```

PSECT SUMMARY

```

: Name Bytes Attributes
: _COB$CODE 1627 NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)

```

Library Statistics

File	Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	30	0	581	00:00.7
_\$255\$DUA28:[COBRTL.OBJ]SMGLIB.L32;1	469	31	6	38	00:00.2

COMMAND QUALIFIERS

```

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS$:COBESCGEN/OBJ=OBJ$:COBESCGEN MSRC$:COBESCGEN/UPDATE=(ENH$:COBESCGEN
: )

```

```

: Size: 1396 code + 231 data bytes
: Run Time: 00:24.7
: Elapsed Time: 01:33.0
: Lines/CPU Min: 7484
: Lexemes/CPU-Min: 27092
: Memory Used: 234 pages
: Compilation Complete

```



0062 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

The image displays a grid of 120 terminal windows, arranged in 10 rows and 12 columns. Each window shows a different screen of a COBOL program. The programs are:

- COBDIVQ LIS
- COBFINDA LIS
- COBDEXCE LIS
- COBEXPI LIS
- COBDEEDIT LIS
- COBDISPLA LIS
- COBESCGEN LIS
- COBERROR LIS
- COBDHANDL LIS

The screens contain various data, including headers, tables, and control panels. The text is small and dense, typical of a terminal printout.