

[illegible]

```
CCCCCCCCC  VV      VV      TTTTTTTTTT  TTTTTTTTTT  IIIIIII  MM      MM  EEEEEEEEEEE
CCCCCCCCC  VV      VV      TTTTTTTTTT  TTTTTTTTTT  IIIIIII  MM      MM  EEEEEEEEEEE
CC          VV      VV      TT          TT          II      MM      MM  EE
CC          VV      VV      TT          TT          II      MM      MM  EE
CC          VV      VV      TT          TT          II      MM      MM  EE
CC          VV      VV      TT          TT          II      MM      MM  EE
CC          VV      VV      TT          TT          II      MM      MM  EE
CC          VV      VV      TT          TT          II      MM      MM  EE
CC          VV      VV      TT          TT          II      MM      MM  EE
CC          VV      VV      TT          TT          II      MM      MM  EE
CC          VV      VV      TT          TT          II      MM      MM  EE
CCCCCCCCC  VV      VV      TT          TT          IIIIIII  MM      MM  EEEEEEEEEEE
CCCCCCCCC  VV      VV      TT          TT          IIIIIII  MM      MM  EEEEEEEEEEE

LL          IIIIIII  SSSSSSSSS
LL          IIIIIII  SSSSSSSSS
LL          II      SS
LL          II      SS
LL          II      SS
LL          II      SS
LL          II      SSSSSSS
LL          II      SSSSSSS
LL          II      SS
LL          II      SS
LL          II      SS
LLLLLLLLLLL  IIIIIII  SSSSSSSSS
LLLLLLLLLLL  IIIIIII  SSSSSSSSS
```


(2) 58
(4) 344
(5) 547
(6) 594
(7) 620
(8) 655
(9) 735

DECLARATIONS
LIB\$CVT_TIME
RETURN_STRING - RETURN DATE/TIME STRING IF REQUESTED
LIB\$CVT_ETIME - CONVERT ABSOLUTE TIME
LIB\$CVT_DTIME - CONVERT DELTA TIME
PARSE_INPUT_STRING
CNV_MNEM_DAY

```
0000 1 .TITLE CVTTIME - CONVERT CLI TIME TO BINARY
0000 2 .IDENT 'V04-000'
0000 3
0000 4
0000 5 *****
0000 6
0000 7 *
0000 8 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 9 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 10 * ALL RIGHTS RESERVED.
0000 11 *
0000 12 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 13 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 14 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 15 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 16 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 17 * TRANSFERRED.
0000 18 *
0000 19 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 20 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 21 * CORPORATION.
0000 22 *
0000 23 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 24 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 25 *
0000 26 *****
0000 27
0000 28
0000 29 ++
0000 30 FACILITY: GENERAL LIBRARY
0000 31
0000 32 ABSTRACT:
0000 33
0000 34 THESE ROUTINES ARE CALLED TO CONVERT AN ASCII STRING TO BINARY ABSO-
0000 35 LUTE OR DELTA-TIME FORMAT
0000 36
0000 37 ENVIRONMENT: USER MODE
0000 38
0000 39 AUTHOR: C. MONIA , CREATION DATE: 12-SEP-1977
0000 40
0000 41 MODIFIED BY:
0000 42
0000 43 V03-005 PCG0001 Peter George 23-Jun-1983
0000 44 Do not accept keywords as delta time strings.
0000 45 Remove leading blank from absolute time string.
0000 46
0000 47 V03-004 BLS0210 Benn Schreiber 3-Mar-1983
0000 48 Change third arg to be string descriptor, add 3 more
0000 49 args. Work around absolute addressing caused by reference
0000 50 to DAY TABLE. Get length of return value when calling
0000 51 SASCTIM from LIB$CVT_TIME.
0000 52
0000 53 V03-003 LMP49245 L. Mark Pilant, 20-Sep-1982 16:15
0000 54 Correct offset used to calculate "YESTERDAY".
0000 55
0000 56 --
```



```
0000 58      .SBTTL  DECLARATIONS
0000 59      :
0000 60      : INCLUDE FILES:
0000 61      :
0000 62      :
0000 63      $CLMSGDEF      ; DEFINE CLI STATUS CODES
0000 64      $$$DEF        ; DEFINE SYSTEM STATUS CODES
0000 65      :
0000 66      :
0000 67      :
0000 68      :
0000 69      : MACROS:
0000 70      :
0000 71      : DEFINE STATE
0000 72      :
0000 73      :
0000 74      .MACRO  STATE  NAME,FILL
0000 75      .SAVE
0000 76      .PSECT  CNVCLI_STATE,NOWRT,RD
0000 77      .IF NB  NAME
0000 78  NAME:
0000 79      .ENDC
0000 80      .REPT  NTOKNS
0000 81      .IF NB  FILL
0000 82      .BYTE  ^A/FILL/
0000 83      .IFF
0000 84      .BYTE  0
0000 85      .ENDC
0000 86      .ENDR
0000 87      .REPT  NTOKNS
0000 88      .WORD  ILLCHR-CNV_RELOC
0000 89      .WORD  0
0000 90      .ENDR
0000 91      .RESTORE
0000 92      .ENDM
0000 93      :
0000 94      :
0000 95      : DEFINE TOKEN LIST ENTRY
0000 96      :
0000 97      :
0000 98      .MACRO  TOKEN  TOKN,NXTSTATE,STRNGADR,FILL
0000 99      .SAVE
0000 100     .PSECT  _CNVCLI_STATE,NOWRT,RD
0000 101     $$$1=.
0000 102     .BLKL  -NTOKNS
0000 103     $$$2=-NTOKNS
0000 104     .IF IDN <->,<TKN>
0000 105     .BLKL  1
0000 106     $$$2=$$$2+1
0000 107     .ENDC
0000 108     .IF IDN <:,>,<TKN>
0000 109     .BLKL  2
0000 110     $$$2=$$$2+2
0000 111     .ENDC
0000 112     .IF IDN < >,<TKN>
0000 113     .BLKL  3
0000 114     $$$2=$$$2+3
```

```
0000 115 .ENDC
0000 116 .IF NB STRNGADR
0000 117 .WORD STRNGADR-CNV_RELOC
0000 118 .IFF
0000 119 .WORD ENDSECS-CNV_RELOC
0000 120 .ENDC
0000 121 .IF NB NXTSTATE
0000 122 .WORD NXTSTATE-CNV_RELOC
0000 123 .IFF
0000 124 .WORD 0
0000 125 .ENDC
0000 126 .=$$$2
0000 127 .IF NB <FILL>
0000 128 .IF IDN <RESCAN>,<FILL>
0000 129 .BYTE RESC
0000 130 .IFF
0000 131 .IF IDN <NULL>,<FILL>
0000 132 .BYTE 0
0000 133 .IFF
0000 134 .BYTE ^A/FILL/
0000 135 .ENDC
0000 136 .ENDC
0000 137 .ENDC
0000 138 .=$$$1
0000 139 .RESTORE
0000 140 .ENDM
0000 141
0000 142 :
0000 143 : DEFINE KEYWORD ENTRIES
0000 144 :
0000 145
0000 146 .MACRO KEY_ENTRY KEYWORD
0000 147
0000 148 FIRST_FLAG$$=128
0000 149 .IRPC CHAR$$,<KEYWORD>
0000 150 .BYTE ^A/CHAR$$/!FIRST_FLAG$$
0000 151 FIRST_FLAG$$=0
0000 152 .ENDR
0000 153 .ENDM
0000 154
0000 155 :
0000 156 : EQUATED SYMBOLS:
0000 157 :
0000 158
00000000 0000 159 EOS=0 : TERMINATION SYMBOL
00000019 0000 160 MAXLGH=25 : MAX STRING LENGTH
FFFFFFFFE 0000 161 RESC=-2 : RESCAN FLAG
0000003C 0000 162 TS_LEN=60 : LIB$CVTTIME TEMPORARY STRING LENGTH
0000 163
0000 164 :
0000 165 : DEFINE TOKEN DISPATCH TABLE OFFSETS
0000 166 :
0000 167
0000 168 .PSECT $AB$$,ABS
0000 169
00000002 0000 170 TSTRING:.BLKW 1 : OFFSET TO TERMINAL STRING
00000004 0002 171 NEXTSTATE:.BLKW 1 : OFFSET TO NEXT STATE
```



```
0004 172 :  
0004 173 : OFFSETS FROM FP USED BY LIB$CVT_TIME  
0004 174 : (NB: This is not the only use of the stack)  
0004 175 :  
0004 176 : $OFFSET 0,NEGATIVE,<-  
0004 177 : <TEMPSTRING,TS_LEN>,- : LIB$CVTTIME Temporary String  
0004 178 : <TEMPDESC,8>,- : Descriptor for TEMPSTRING  
0004 179 : <SCRATCHSIZE,0>- : Size of Scratch Area  
0004 180 :  
FFC4 TEMPSTRING:  
FFBC TEMPDESC:  
FFBC SCRATCHSIZE:  
0004 181 :  
0004 182 : OWN STORAGE:  
0004 183 :  
00000000 184 : .PSECT _CNVCLI_CODE NOWRT, LONG  
00000000 185 :  
00000000 186 CNV_RELOC:.BLKL 0 : DEFINE RELOCATION BASE  
00000000 187 :  
00000000 188 :  
00000000 189 : TOKEN LIST  
00000000 190 :  
00000000 191 :  
00 00000000 192 $$$2= :  
00 2D 3A 20 0000 193 TOKENS: .ASCII / :-/<EOS> : ORDER IS FIXED  
00000004 0004 194 NTOKNS=.-$$$2 :  
0004 195 :  
0004 196 :  
0004 197 : TOKEN SUBSTITUTION STRINGS  
0004 198 :  
0004 199 : ADJACENCY REQUIRED  
0004 200 :  
0004 201 : COLON  
0004 202 :  
0004 203 :  
2D 0004 204 COLDAY: .ASCII /-/  
2D 0005 205 COLMON: .ASCII /-/  
00 20 0006 206 COLYR: .ASCII / /  
0008 207 COLHR: :  
00 3A 0008 208 COLMIN: .ASCII /:/  
00 20 30 000A 209 COLD: .ASCII /0 /  
000D 210 :  
000D 211 :  
000D 212 : END OF STRING  
000D 213 :  
000D 214 :  
0000000D 000D 215 $$$3= :  
2D 000D 216 ENDDAY: .ASCII /-/  
2D 000E 217 ENDMON: .ASCII /-/  
000F 218 ENDD: :  
30 30 20 000F 219 ENDYR: .ASCII / 0/  
30 30 3A 0011 220 ENDDR: .ASCII /:00/  
30 30 3A 0014 221 ENDMIN: .ASCII /:00.00/  
00 001A 222 ENDSECS:.BYTE 0  
00 FF 001B 223 ILLCHR: .BYTE -1,0  
00000010 001D 224 FILCNT=.-$$$3 :  
0000002C 001D 225 TBUF=<<FILCNT+MAXLGH>+3>8-4 :  
: DD:  
: DD-MMM:  
: DD-MMM-YYYY:  
: [DD-MMM-YYYY]:HH:  
: [DD-MMM-YYYY]:HH:MM:  
: <BLANK>!<->[HH:MM:SS.SS]  
: DD<EOS>  
: DD-MMM<EOS>  
: DD<BLANK><EOS>  
: DD-MMM-YYYY<EOS>  
: [DD-MMM-YYYY]:HH<EOS>  
: [DD-MMM-YYYY]:HH:MM<EOS>  
: [DD-MMM-YYYY]:HH:MM:SS.SS<EOS>  
: ILLEGAL CHARACTER TERMINATOR  
: FILL COUNT  
: SIZE OF BUFFER ON STACK
```

```
001D 226
001D 227 :
001D 228 : BUFFER LENGTHS FOR ASCII TIME CONVERSION CALLS
001D 229 :
001D 230
0000000B 001D 231 SHORT_TIME_LEN=11 : LENGTH OF BUFFER FOR TIME WITHOUT HOURS
0000000B 001D 232 MIDNIGHT_LEN=8 : LENGTH OF MIDNIGHT ASCII TIME STRING
0000002C 001D 233 FULL_TIME_LEN=TBUF : FULL TIME BUFFER LENGTH
001D 234
001D 235 :
001D 236 : MINUS SIGN
001D 237 :
001D 238
001D 239 MINDAY: : DD-[MM-YYYY:HH:MM:SS.SS]<EOS>
00 2D 001D 240 MINMON: .ASCIZ /-/ : DD-MM-[YYYY:HH:MM:SS.SS]<EOS>
001F 241
001F 242 :
001F 243 : DEFINE STATE TABLES FOR TIME CONVERSION
001F 244 :
001F 245 : INITIAL STATE
001F 246 :
001F 247 : PARSE DAY OF MONTH
001F 248 :
001F 249 :
001F 250 STATE DAY : PARSE DAY OF MONTH
001F 251 TOKEN EOS,HOURS,COLDAY,RESCAN : ASSUME TIME ON END OF STRING
001F 252 TOKEN <:,>,HOURS,COLDAY,RESCAN : ASSUME TIME ON COLON
001F 253 TOKEN -,MONTH,MINDAY : PARSE MONTH OF YEAR
001F 254 TOKEN < >,HOURS,COLDAY,RESCAN : ASSUME TIME ON SPACE
001F 255
001F 256 :
001F 257 : PARSE MONTH OF YEAR
001F 258 :
001F 259 :
001F 260 STATE MONTH :
001F 261 TOKEN EOS,,ENDMON : TERMINATE ON END OF STRING
001F 262 TOKEN < >,HOURS,COLMON : TERMINATE MONTH, CONVERT TIME
001F 263 TOKEN <:,>,HOURS,COLMON : TERMINATE MONTH, CONVERT TIME
001F 264 TOKEN -,YEAR,MINMON : PARSE YEAR
001F 265
001F 266 :
001F 267 : PARSE YEAR
001F 268 :
001F 269 :
001F 270 STATE YEAR :
001F 271 TOKEN EOS,,ENDYR : TERMINATE ON END OF STRING
001F 272 TOKEN <:,>,HOURS,COLYR : CONVERT TIME, TERMINATE DATE
001F 273 TOKEN < >,HOURS,COLYR : CONVERT TIME, TERMINATE DATE
001F 274
001F 275 :
001F 276 : PARSE HOURS
001F 277 :
001F 278 :
001F 279 STATE HOURS,<0> :
001F 280 TOKEN EOS,,ENDHR : TERMINATE ON END OF STRING
001F 281 TOKEN <:,>,MINUTES,COLHR : MOVE TOKEN
001F 282 TOKEN < >,HOURS,,NULL : IGNORE SPACES
```



```
001F 283
001F 284 :
001F 285 : PARSE MINUTES
001F 286 :
001F 287 :
001F 288 STATE MINUTES,<0> :
001F 289 TOKEN EOS,,ENDMIN : TERMINATE ON END OF STRING
001F 290 TOKEN <:,>,SECONDS,COLMIN : MOVE PARSED STRING
001F 291 TOKEN < >,MINUTES,,NULL : IGNORE SPACES
001F 292 :
001F 293 :
001F 294 : PARSE SECONDS
001F 295 :
001F 296 :
001F 297 STATE SECONDS,<0> :
001F 298 TOKEN EOS,,ENDSECS : TERMINATE ON END OF STRING
001F 299 TOKEN < >,SECONDS,,NULL : IGNORE SPACES
001F 300 :
001F 301 :
001F 302 : PARSE DELTA-TIME FORMAT
001F 303 :
001F 304 : PARSE DELTA-DAY
001F 305 :
001F 306 :
001F 307 STATE DELTA DAY,<0> :
001F 308 TOKEN EOS,HOURS,COLD,RESCAN : RESCAN ON END OF STRING
001F 309 TOKEN <:,>,HOURS,COLD,RESCAN : RESCAN ON COLON
001F 310 TOKEN < >,HOURS,COLD,RESCAN : RESCAN ON SPACE
001F 311 TOKEN <->,HOURS,COLYR : MOVE PARSED STRING
```

```
001F 313 :  
001F 314 : DEFINE MNEMONIC KEYWORDS THAT CAN REPRESENT TIMES  
001F 315 :  
001F 316 :  
001F 317 DAY_TABLE:  
001F 318 KEY_ENTRY YEST ; 'YESTERDAY'  
0023 319 KEY_ENTRY TODA ; 'TODAY'  
0027 320 KEY_ENTRY TOMO ; 'TOMORROW'  
0000000C 002B 321 DAY_TBL_LEN=-DAY_TABLE ; LENGTH OF KEYWORD TABLE  
002B 322 :  
002B 323 :  
002B 324 : TIME FACTORS. EACH KEYWORD HAS AN ASSOCIATED TIME FACTOR, WHICH, WHEN ADDED  
002B 325 : TO THE CURRENT TIME VALUE, CREATES A 64-BIT TIME VALUE THAT IS EQUIVALENT TO  
002B 326 : THE KEYWORD STRING.  
002B 327 :  
002B 328 : YESTERDAY IS ONE NANOSECOND BEFORE ZERO TIME TODAY.  
002B 329 : TODAY IS ZERO TIME TODAY.  
002B 330 : TOMORROW IS ZERO TIME TODAY PLUS ONE DAY'S WORTH OF NANOSECONDS.  
002B 331 :  
002B 332 : NOTE THAT THE FIGURES IN THIS TABLE ARE 'MAGIC' NUMBERS.  
002B 333 :  
002B 334 :  
002B 335 TIME_TABLE:  
D5964000 002B 336 .LONG ^XD5964000 ; 'YESTERDAY'  
FFFFFFF36 002F 337 .LONG ^XFFFFFFF36  
00000000 0033 338 .LONG 0 ; 'TODAY'  
00000000 0037 339 .LONG 0  
2A69C000 003B 340 .LONG ^X2A69C000 ; 'TOMORROW'  
000000C9 003F 341 .LONG ^XC9  
0043 342
```



```
0043 344 .SBTTL LIB$CVT_TIME
0043 345 :++
0043 346 : FUNCTIONAL DESCRIPTION:
0043 347 :
0043 348 : THESE ROUTINES ARE CALLED TO CONVERT AN ASCII STRING INTO A FORM
0043 349 : THAT IS ACCEPTABLE TO THE 'GET TIME' SYSTEM SERVICES.
0043 350 :
0043 351 : ENTRY POINT LIB$CVT_TIME IS CALLED TO CONVERT A STRING TO ABSOLUTE
0043 352 : BINARY TIME.
0043 353 :
0043 354 : ENTRY POINT LIB$CVT_DTIME IS CALLED TO CONVERT A STRING TO DELTA
0043 355 : TIME
0043 356 :
0043 357 : ABSOLUTE TIME SYNTAX:
0043 358 :
0043 359 : THE FULLY EXPANDED TIME STRING, AS SEEN BY THE SYSTEM, HAS THE FORM:
0043 360 :
0043 361 : DD-MMM-YYYY<BLANK>HH:MM:SS.SS
0043 362 :
0043 363 : WHERE ANY FIELD MAY BE EMPTY.
0043 364 :
0043 365 : THE TIME CONVERSION ROUTINES CONVERT ABBREVIATED INPUT SUPPLIED BY
0043 366 : THE CALLER INTO THE ABOVE FORMAT, THEN INVOKE THE SYSTEM SERVICE TO
0043 367 : CONVERT THE STRING TO BINARY.
0043 368 :
0043 369 : THE FOLLOWING FORMS ARE ACCEPTABLE:
0043 370 :
0043 371 : DD-MMM-YYYY:HH:MM:SS.SS - FULL
0043 372 :
0043 373 : DD-MMM-YYYY<BLANK>HH:MM:SS.SS - FULL
0043 374 :
0043 375 : DD-MMM:HH:MM - TRUNCATED FIELDS
0043 376 :
0043 377 : DD--YYYY:HH::SS - NULL FIELDS
0043 378 :
0043 379 : HH:[MM:SS.SS] - TIME ONLY
0043 380 :
0043 381 : DELTA TIME SYNTAX
0043 382 :
0043 383 : THE FULLY EXPANDED DELTA TIME STRING AS SEEN BY THE SYSTEM IS:
0043 384 :
0043 385 : DD<BLANK>HH:MM:SS.S
0043 386 :
0043 387 : WHERE ANY FIELD MAY BE EMPTY.
0043 388 :
0043 389 : THE FOLLOWING FORMS ARE ACCEPTABLE:
0043 390 :
0043 391 : DD-HH:MM:SS.SS - FULL
0043 392 :
0043 393 : DD-HH: - TRUNCATED FIELDS
0043 394 :
0043 395 : DD-:MM - NULL FIELDS
0043 396 :
0043 397 : HH[:MM:SS.SS] - TIME ONLY
0043 398 :
0043 399 :
0043 400 : CALLING SEQUENCE:
```

```
0043 401 :  
0043 402 : CALL LIBSCVT_TIME (DESCR,TIME,[OUTDESC],[OUTLEN],[USEDLEN],[FLAGS])  
0043 403 : CALL LIBSCVT_ETIME (DESCR,TIME,[OUTDESC],[OUTLEN],[USEDLEN],[FLAGS])  
0043 404 : CALL LIBSCVT_DTIME (DESCR,TIME,[OUTDESC],[OUTLEN],[USEDLEN],[FLAGS])  
0043 405 :  
0043 406 : INPUT PARAMETERS:  
0043 407 :  
0043 408 : DESCR = ADDRESS OF INPUT STRING DESCRIPTOR  
0043 409 :  
0043 410 : TIME = ADDRESS OF QUADWORD TO RECEIVE CONVERTED TIME  
0043 411 :  
0043 412 : OUTDESC = [OPTIONAL] ADDRESS OF DESCRIPTOR FOR CONVERTED TIME  
0043 413 : ASCII STRING  
0043 414 :  
0043 415 : OUTLEN = [OPTIONAL] ADDRESS OF WORD TO RECEIVE OUTPUT STRING LENGTH  
0043 416 :  
0043 417 : USEDLEN = [OPTIONAL] ADDRESS OF WORD TO RECEIVE LENGTH OF STRING USED  
0043 418 :  
0043 419 : FLAGS = [OPTIONAL] NYI  
0043 420 :  
0043 421 : IMPLICIT INPUTS:  
0043 422 :  
0043 423 : NONE  
0043 424 :  
0043 425 : OUTPUT PARAMETERS:  
0043 426 :  
0043 427 : RO=CLIS_NORMAL  
0043 428 :  
0043 429 :  
0043 430 : TIME = ABSOLUTE BINARY TIME IN 64-BIT FORMAT  
0043 431 :  
0043 432 :  
0043 433 : RO=CLIS_IVVALU  
0043 434 :  
0043 435 : INVALID TIME FORMAT DETECTED  
0043 436 :  
0043 437 : IMPLICIT OUTPUTS:  
0043 438 :  
0043 439 : NONE  
0043 440 :  
0043 441 : COMPLETION CODES:  
0043 442 :  
0043 443 : RO LBS = SUCCESS  
0043 444 : RO LBC = SYNTAX ERROR  
0043 445 :  
0043 446 : SIDE EFFECTS:  
0043 447 :  
0043 448 : NONE  
0043 449 :  
0043 450 : --  
0043 451 :  
0043 452 : CONVERT FULL TIME STRING  
0043 453 :  
0043 454 : .ALIGN LONG  
0044 455 : .ENTRY LIBSCVT_TIME,^M<R2,R3,R4,R5,R6,R7,R8,R9>  
0046 456 : MOVAL SCRATCHSIZE(SP),SP ; RESERVE SCRATCH STORAGE  
004A 457 : MOVC5 #0,(SP),#0,#-SCRATCHSIZE,(SP) ; ZERO THE SCRATCH STORAGE
```

6E 0044 8F 00 5E BC AE 03FC DE 2C 004A

57	C4	AD	9E	0052	458	MOVAB	TEMPSTRING(FP),R7	: ADDR TO HOLD RETURNED STRING
BC	AD	3C	D0	0056	459	MOVL	#TS_LEN,TEMPDESC(FP)	: SETUP DESCRIPTOR
CO	AD	57	D0	005A	460	MOVL	R7,TEMPDESC+4(FP)	: ...
52	04	BC	7D	005E	461	MOVQ	@4(AP),R2	: GET STRING DESCRIPTOR
	52	52	3C	0062	462	MOVZWL	R2,R2	: Truncate length to a word
		52	D5	0065	463			
		09	13	0067	464	TSTL	R2	: CHARACTERS LEFT TO EXAMINE?
63		20	91	0069	465	BEQL	10\$	: NO, THEN SKIP
		04	12	006C	466	CMPB	#^A/ /,(R3)	: DELETE LEADING BLANK
		52	D7	006E	467	BNEQ	10\$	: BRANCH IF NOT BLANK
		53	D6	0070	468	DECL	R2	: DECREMENT LENGTH
				0072	469	INCL	R3	: INCREMENT ADDRESS
63	52	2B	3A	0072	471	10\$: LOCC	#^A/+/ ,R2,(R3)	: LOOK FOR PLUS SIGN
	56	50	D0	0076	472	MOVL	R0,R6	: SAVE BYTES REMAINING
		43	12	0079	473	BNEQ	40\$	: BR IF "+" FOUND
63	52	3A	3A	007B	474	LOCC	#^A/: / ,R2,(R3)	: LOOK FOR COLON
	54	51	D0	007F	475	MOVL	R1,R4	: SAVE ADDR IF FOUND
				0082	476			
63	52	2D	3A	0082	477	LOCC	#^A/- / ,R2,(R3)	: LOOK FOR DASH
		36	13	0086	478	BEQL	40\$	: BR IF ABS TIME ONLY
	55	51	D0	0088	479	MOVL	R1,R5	: SAVE LOC. OF FIRST DASH
	54	51	D1	008B	480	CMPL	R1,R4	: "-" RIGHT OF ""
		2E	1A	008E	481	BGTRU	40\$	: IF YES - THEN DELIMITER
				0090	482			
		50	D7	0090	483	DECL	R0	: ADJUST COUNT BEYOND ""
		51	D6	0092	484	INCL	R1	: ADJUST ADDR BEYOND ""
61	50	2D	3A	0094	485	LOCC	#^A/- / ,R0,(R1)	: LOOK FOR 2ND DASH
		0F	13	0098	486	BEQL	20\$	: IF EQL - THEN NOT FOUND
	54	51	D1	009A	487	CMPL	R1,R4	: "-" RIGHT OF ""
		1F	1A	009D	488	BGTRU	40\$	: IF YES - THEN DELIMITER
				009F	489			
		50	D7	009F	490	DECL	R0	: ADJUST COUNT BEYOND 2ND ""
		51	D6	00A1	491	INCL	R1	: ADJUST ADDR BEYOND 2ND ""
61	50	2D	3A	00A3	492	LOCC	#^A/- / ,R0,(R1)	: LOOK FOR THIRD DASH
		15	12	00A7	493	BNEQ	40\$	: IF FOUND - THEN DELIMITER
				00A9	494			
		55	D6	00A9	495	20\$: INCL	R5	: INCREMENT BEYOND ""
	3A	65	91	00AB	496	CMPB	(R5),#^A/: /	: ...
		0A	13	00AE	497	BEQL	30\$	
	30	65	91	00B0	498	CMPB	(R5),#^A/0/	: CAN NEXT CHAR BE A DIGIT?
		09	19	00B3	499	BLSS	40\$	: IF LESS THAN - THEN NO
	39	65	91	00B5	500	CMPB	(R5),#^A/9/	: CAN CHAR BE A DIGIT?
		04	14	00B8	501	BGTR	40\$	: IF GT - THEN NO
51	55	01	C3	00BA	502	30\$: SUBL3	#1,R5,R1	: BREAK STRING AT DIGIT OR DASH
				00BE	503			
				00BE	504	40\$:		
		7E	7C	00BE	505	CLRQ	-(SP)	: QUAD WORD TIME BUFFER
	7E	53	D0	00C0	506	MOVL	R3, -(SP)	: GET ADDR. OF ABS TIME STRING
	51	53	C3	00C3	507	SUBL3	R3,R1, -(SP)	: LENGTH OF ABS
		0C	12	00C7	508	BNEQ	60\$	: BR IF ABS TIME PRESENT
				00C9	509	\$GETTIM	S	: GET CURRENT TIME
		0E	11	00D3	510	BRB	-80\$	: DON'T CONVERT ABS TIME
		08	AE	7F	00D5	60\$: PUSHAQ	8(SP)	: ADDR. OF BUFFER
	04	AE	7F	00D8	512	PUSHAQ	4(SP)	: ADDR. OF DESCRIPTOR
0185'CF		02	FB	00DB	513	CALLS	#2, W^LIB\$CVT_ETIME	: CONVERT ABS TIME
	58	50	E9	00E0	514	BLBC	R0,180\$	: IF LBC - THEN ERROR

```
08 BC 08 AE 7D 00E3 515 80$: MOVQ 8(SP),a8(AP) ; MOVE TIME TO USER BUFFER
      00E8 516
      6E D6 00E8 517 INCL (SP) ; INCLUDE DELIMITER IN COUNT
04 AE 6E C0 00EA 518 ADDL (SP),4(SP) ; POINT TO DELTA TIME
6E 52 6E C3 00EE 519 SUBL3 (SP),R2,(SP) ; CALC LENGTH OF DELTA TIME
      2A 15 00F2 520 BLEQ 140$ ; EXIT IF NO DELTA TIME
      00F4 521
      08 AE 7F 00F4 522 PUSHAQ 8(SP) ; ADDR OF BUFFER
      04 AE 7F 00F7 523 PUSHAQ 4(SP) ; ADDR OF DESCRIPTOR
01B0'CF 02 FB 00FA 524 CALLS #2,W^LIB$CVT_DTIME ; CONVERT DELTA TIME
      39 50 E9 00FF 525 BLBC R0,180$ ; IF LBC - THEN ERROR CONVERTING
      0102 526
      51 08 AC D0 0102 527 MOVL 8(AP),R1 ; GET ADDR OF USER BUFFER
      56 D5 0106 528 TSTL R6 ; DELTA TIME PLUS OR MINUS?
      0B 13 0108 529 BEQL 120$ ; BR IF MINUS
      010A 530 ;
      010A 531 ; delta time is negative so arithmetic is inverted
      010A 532 ;
81 61 08 AE C3 010A 533 SUBL3 8(SP),(R1),(R1)+ ; SUBTRACT DELTA
      61 0C AE D9 010F 534 SBWC 12(SP),(R1) ; FROM ABS TIME
      09 11 0113 535 BRB 140$ ; RETURN
81 61 08 AE C1 0115 536 120$: ADDL3 8(SP),(R1),(R1)+ ; ADD DELTA
      61 0C AE D8 011A 537 ADWC 12(SP),(R1) ; TO ABS TIME
      03 6C 91 011E 538 140$: CMPB (AP),#3 ; DO WE NEED TO RETURN THE DATE/TIME?
      18 1F 0121 539 BLSSU 180$ ; IF LSSU NO, DON'T BOTHER
      0123 540 SASCTIM_S TIMADR=a8(AP),- ; YES, CONVERT BINARY TIME
      0123 541 TIMBUF=TEMPDESC(FP),-
      0123 542 TIMLEN=TEMPDESC(FP)
      58 BC AD 7D 0135 543 MOVQ TEMPDESC(FP),R8 ; LOAD DESCRIPTOR
      01 10 0139 544 BSBB RETURN_STRING ; RETURN STRING TO USER
      04 013B 545 180$: RET
```



```
013C 547 .SBTTL RETURN_STRING - RETURN DATE/TIME STRING IF REQUESTED
013C 548 :++
013C 549 : FUNCTIONAL DESCRIPTION:
013C 550 :
013C 551 : THIS ROUTINE RETURNS THE DATE/TIME STRING (AND STRING) LENGTH
013C 552 : TO THE CALLER IF THE OPTIONAL ARGUMENTS ARE PRESENT
013C 553 :
013C 554 : INPUTS:
013C 555 :
013C 556 : R8 LENGTH OF DATE/TIME STRING
013C 557 : R9 ADDRESS OF DATE/TIME STRING
013C 558 :
013C 559 : 12(AP) IF PRESENT, ADDRESS OF OUTPUT STRING DESCRIPTOR
013C 560 : 16(AP) IF PRESENT, ADDRESS OF WORD TO STORE OUTPUT STRING LENGTH
013C 561 :
013C 562 :--
013C 563 .WEAK DCL$SCOPY_DXDX ;DON'T COMPLAIN IF NOT FOUND
013C 564
013C 565 RETURN_STRING:
50 01 D0 013C 566 MOVL #1,R0 ;PRESET SUCCESS
03 6C 91 013F 567 CMPB (AP),#3 ;THIRD ARGUMENT PRESENT?
04 40 1F 0142 568 BLSSU 20$ ;BRANCH IF NOT
OC AC D5 0144 569 TSTL 12(AP) ;YES, BUT WAS IT DEFAULTED?
3B 13 0147 570 BEQL 20$ ;BRANCH IF SO
58 D5 0149 571 TSTL R8 ;NON-ZERO STRING?
09 13 014B 572 BEQL 5$ ;IF EQL ZERO STRING
20 69 91 014D 573 CMPB (R9),#^A/ / ;IS LEADING CHAR A SPACE?
04 12 0150 574 BNEQ 5$ ;IF NEQ NO
58 D7 0152 575 DECL R8 ;YES, STRIP IT OUT
59 D6 0154 576 INCL R9 ; AND DON'T COPY IT
59 DD 0156 577 5$: PUSHL R9 ;FORM DESCRIPTOR ON STACK
58 DD 0158 578 PUSHL R8 ;
OC AC DD 015A 579 PUSHL 12(AP) ;STACK OUTPUT DESCRIPTOR ADDRESS
04 AE 9F 015D 580 PUSHAB 4(SP) ;STACK INPUT DESCRIPTOR ADDRESS
50 00000000'GF 9E 0160 581 MOVAB G^DCL$SCOPY_DXDX,R0 ;IS DCL STRING COPY PRESENT?
07 12 0167 582 BNEQ 10$ ;IF NEQ YES--GO USE IT
50 00000000'GF 9E 0169 583 MOVAB G^LIB$SCOPY_DXDX,R0 ;NO--USE LIB$SCOPY
60 04 FB 0170 584 10$: CALLS #4,(R0) ;COPY STRING, CLEAR DESCRIPTOR
0173 585 ; FROM STACK
OE 50 E9 0173 586 BLBC R0,20$ ;BRANCH IF SOME SORT OF COPY ERROR
04 6C 91 0176 587 CMPB (AP),#4 ;WAS 4TH ARGUMENT PRESENT?
09 1F 0179 588 BLSSU 20$ ;IF LSSU NO
10 AC D5 017B 589 TSTL 16(AP) ;YES, DEFAULTED?
04 13 017E 590 BEQL 20$ ;IF EQL YES
10 BC 58 B0 0180 591 MOVW R8,@16(AP) ;NO, STORE OUTPUT STRING LENGTH
05 0184 592 20$: RSB
```

```
0185 594 .SBTTL LIBSCVT_ETIME - CONVERT ABSOLUTE TIME
0185 595
0185 596 :
0185 597 : CONVERT ABSOLUTE TIME
0185 598 :
0185 599
0185 600 .ENABL LSB
0185 601
0185 602
0185 603 .ENTRY LIBSCVT_ETIME, ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
5B 00000000 EF 9E 0187 604 MOVAB DAY,R11 : GET ADDRESS OF STATE TABLE
52 00038088 8F D0 018E 605 MOVL #CLIS_IVVALU,R2 : ASSUME STRING TOO LONG
56 04 BC 7D 0195 606 MOVQ @4(APT),R6 : GET STRING DESCRIPTOR
56 56 3C 0199 607 MOVZWL R6,R6 : Truncate length to a word
56 19 B1 019C 608 CMPW #MAXLGH,R6 : STRING TOO LONG?
5E 5E 1F 019F 609 BLSSU 20$ : IF LSSU YES
5E 2C C2 01A1 610 SUBL #TBUF,SP : ALLOCATE BUFFER ON STACK
59 58 D4 01A4 611 CLRL R8 : CLEAR OUTPUT STRING COUNT
00E4 30 01A6 612 MOVL SP,R9 : POINT TO BUFFER
01A9 613 BSBW CNV_MNEM_DAY : SEE IF THE INPUT STRING IS A WORD
01AC 614 : LIKE 'YESTERDAY', 'TODAY', OR
01AC 615 : 'TOMORROW'
25 50 E9 01AC 616 BLBC R0,10$ : IF NOT, CHECK OUT DELTA TIMES
04 01AF 617 RET
01B0 618 :
```



```
                                01B0 620      .SBTTL LIB$CVT_DTIME - CONVERT DELTA TIME
                                01B0 621
                                01B0 622      :
                                01B0 623      : CONVERT DELTA TIME
                                01B0 624      :
                                01B0 625
5B      00000078'EF 0FFC 01B0 626      .ENTRY LIB$CVT_DTIME, ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
52      00038088 8F 9E 01B2 627      MOVAB DELTA_DAY,R11      : GET ADDRESS OF STATE TABLE
      56 04 BC 7D 01B9 628      MOVL #CLI$_IVVALU,R2      : ASSUME STRING TOO LONG
      56 56 3C 01C0 629      MOVQ @4(AP),R6      : GET STRING DESCRIPTOR
      56 19 B1 01C4 630      MOVZWL R6,R6      : Truncate length to a word
      33 1F 01C7 631      CMPW #MAXLGH,R6      : STRING TOO LONG?
      5E 2C 01CA 632      BLSSU 20$      : IF LSSU YES
      58 D4 01CC 633      SUBL #TBUF,SP      : ALLOCATE BUFFER ON STACK
      59 5E D0 01CF 634      CLRL R8      : CLEAR OUTPUT STRING COUNT
      2D 10 01D1 635      MOVL SP,R9      : POINT TO BUFFER
      FF63 30 01D4 636 10$:      BSBB CNV PARSTRNG      : NORMAL TIME SPECIFIED. PARSE STRING
      26 50 E9 01D6 637      BSBW RETURN_STRING      : RETURN STRING TO USER IF REQUESTED
52      00030001 8F D0 01D9 638      BLBC R0,30$      : BRANCH IF SOME SORT OF ERROR
      7E 58 7D 01DC 639      MOVL #CLI$_NORMAL,R2      : ASSUME SUCCESS
      50 5E D0 01E3 640      MOVQ R8,-(SP)      : PUSH DESCRIPTOR
      01E6 641      MOVL SP,R0      : SAVE DESCRIPTOR ADDRESS
      01E9 642      $BINTIM,S -      : CONVERT TIME TO BINARY
      01E9 643      TIMBUF=(R0),-      : ADDRESS OF TIME BUFFER
      01E9 644      TIMADR=@8(AP)      : ADDRESS OF QUADWORD RESULT
52      00038088 8F E8 01F5 645      BLBS R0,20$      : IF LBS, SUCCESS
      50 52 D0 01F8 646      MOVL #CLI$_IVVALU,R2      : SET INVALID VALUE
      01FF 647      20$:      :
      01FF 648      :
      50 52 D0 01FF 649      MOVL R2,R0      : SET STATUS
      0202 650 30$:      :
      04 0202 651      RET      :
      0203 652      :
      0203 653      .DSABL LSB      :
```



```
0203 655 .SBTTL PARSE INPUT STRING
0203 656 :+
0203 657 : CNV_PARSTRNG - PARSE INPUT STRING
0203 658 :
0203 659 : THIS ROUTINE IS CALLED TO PARSE THE INPUT STRING.
0203 660 :
0203 661 : INPUTS:
0203 662 :
0203 663 : R6,R7 = INPUT STRING DESCRIPTOR
0203 664 : R8,R9 = OUTPUT STRING DESCRIPTOR
0203 665 : R11 = ADDRESS OF STATE TABLES
0203 666 :
0203 667 : OUTPUTS:
0203 668 :
0203 669 : R8,R9 = STRING DESCRIPTOR FOR PARSED STRING
0203 670 :
0203 671 : REGISTERS MODIFIED:
0203 672 :
0203 673 : R0 - R11
0203 674 :-
0203 675 :
0203 676 CNV_PARSTRNG:
53 52 D4 0203 677 CLRL R2
57 57 D0 0205 678 MOVL R7,R3
0208 679 10$:
54 00 90 0208 680 MOVB #EOS,R4
56 56 B5 020B 681 TSTW R6
05 13 020D 682 BEQL 20$
54 87 90 020F 683 MOVB (R7)+,R4
56 56 B7 0212 684 DECW R6
0214 685 20$:
FDE6 CF 04 54 3A 0214 686 LOCC R4,#NTOKNS,TOKENS
04 12 021A 687 BNEQ 30$
52 B6 021C 688 INCW R2
E8 11 021E 689 BRB 10$
0220 690 30$:
5A 70 9E 0220 691 MOVAB -(R0),R10
51 6B4A 9E 0223 692 MOVAB (R11)(R10),R1
FE 8F 61 91 0227 693 CMPB (R1),#RESC
11 12 022B 694 BNEQ 35$
56 52 A0 022D 695 ADDW R2,R6
57 52 C2 0230 696 SUBL R2,R7
00 54 91 0233 697 CMPB R4,#EOS
32 13 0236 698 BEQL 45$
57 D7 0238 699 DECL R7
56 D6 023A 700 INCL R6
2C 11 023C 701 BRB 45$
023E 702 35$:
7E 52 B0 023E 703 MOVW R2,-(SP)
09 12 0241 704 BNEQ 40$
53 51 D0 0243 705 MOVL R1,R3
63 95 0246 706 TSTB (R3)
02 13 0248 707 BEQL 40$
6E B6 024A 708 INCW (SP)
024C 709 40$:
50 8E 3C 024C 710 MOVZWL (SP)+,R0
19 13 024F 711 BEQL 45$

: CLEAR LENGTH OF PARSED STRING
: POINT TO START OF STRING
:
: ASSUME AT END OF STRING
: AT END OF STRING?
: IF EQL YES
: GET CURRENT CHARACTER
: DECREMENT BYTES REMAINING
:
: FIND TOKEN
: IF NEQ HAVE TOKEN
: INCREMENT LENGTH OF PARSED STRING
: GET NEXT CHARACTER
:
: REMOVE BIAS, SAVE INDEX
: POINT TO FILL BYTE
: RESCAN REQUIRED?
: IF NEQ NO
: BACK OUT STRING
:
: END OF STRING?
: IF EQL YES
: BACK OUT TOKEN
:
:
: SAVE LENGTH OF PARSED STRING
: IF NEQ STRING NOT NULL
: POINT TO FILL BYTE
: NULL STRING?
: IF EQL YES
: SET COUNT TO 1
:
: Pick up length
: Br if nothing
```


51	83	9A	0251	712	41\$:	MOVZBL	(R3)+,R1	:	Pick up next byte
61	8F	51	91	0254	713	CMPB	R1,#^A'a'	:	In lower case range?
		09	1F	0258	714	BLSSU	42\$	:	Br if not
7A	8F	51	91	025A	715	CMPB	R1,#^A'z'	:	Test high end
		03	1A	025E	716	BGTRU	42\$	:	Br if not
51	20	8A	0260	717		BICB2	^X20,R1	:	Convert to upper case
8849	51	90	0263	718	42\$:	MOVB	R1,(R8)+[R9]	:	Store in output buffer, bump count
E7	50	F5	0267	719		SOBGTR	R0,41\$	:	Loop for count
			026A	720	45\$:			:	
5B	04	AB4A	DE	026A	721	MOVAL	NTOKNS(R11)[R10],R11	:	POINT TO TOKEN LIST ENTRY
	50	6B	32	026F	722	CVTL	TSTRING(R11),R0	:	GET OFFSET TO STRING
50	FD89	CF40	9E	0272	723	MOVAB	CNV_RELOC[R0],R0	:	COMPUTE STRING ADDRESS
				0278	724			:	
8849	80	90	0278	725	50\$:	MOVB	(R0)+,(R8)+[R9]	:	SUBSTITUTE STRING FOR TOKEN
	FA	12	027C	726		BNEQ	50\$	:	GO AGAIN
	58	B7	027E	727		DECB	R8	:	STRIP EOS
5B	02	AB	32	0280	728	CVTL	NEXTSTATE(R11),R11	:	POINT TO NEXT STATE
	09	13	0284	729		BEQL	60\$	:	IF EQL, TERMINATE SCAN
5B	FD75	CF4B	9E	0286	730	MOVAB	CNV_RELOC[R11],R11	:	COMPUTE ADDRESS OF NEXT STATE
	FF74	31	0286	731		BRW	CNV_PARSTRNG	:	CONTINUE PARSE
			028F	732	60\$:			:	
		05	028F	733		RSB		:	

```
0290 735 .SBTTL CNV_MNEM_DAY
0290 736
0290 737 :++
0290 738 : FUNCTIONAL DESCRIPTION:
0290 739 :
0290 740 : THIS ROUTINE CONVERTS AN ASCII STRING REPRESENTING 'YESTERDAY', 'TODAY',
0290 741 : OR 'TOMORROW' INTO A FORM THAT IS ACCEPTABLE TO THE 'GET TIME' SYSTEM SERVICES.
0290 742 :
0290 743 : STRINGS ACCEPTED ARE THE FOLLOWING:
0290 744 :
0290 745 :     YESTERDAY
0290 746 :     TODAY
0290 747 :     TOMORROW
0290 748 :
0290 749 : ONLY THE FIRST FOUR CHARACTERS OF A STRING ARE READ. UNIQUE ABBREVIATIONS ARE
0290 750 : ACCEPTED.
0290 751 :
0290 752 : INPUTS:
0290 753 :
0290 754 :     R6,R7 - INPUT STRING DESCRIPTOR
0290 755 :     R8,R9 - OUTPUT STRING DESCRIPTOR
0290 756 :     8(AP) - ADDRESS IN WHICH TO PLACE ABSOLUTE BINARY TIME IN 64-BIT FORMAT
0290 757 :
0290 758 : OUTPUTS:
0290 759 :
0290 760 :     @8(AP) - 64-BIT ABSOLUTE BINARY TIME
0290 761 :     @12(AP) - OUTPUT STRING DESCRIPTOR...STRING COPIED
0290 762 :
0290 763 :     R0= LBS 1 - MATCH FOUND, CONVERSION SUCCEEDED.
0290 764 :     R0= LBS 0 - MATCH NOT FOUND OR CONVERSION FAILED.
0290 765 :
0290 766 :     R0=CLIS_ABKEYW - AMBIGUOUS KEYWORD
0290 767 :
0290 768 :     R0 MAY CONTAIN ERROR STATUS CODES FROM 'GET TIME' SYSTEM SERVICES.
0290 769 :
0290 770 : REGISTERS MODIFIED
0290 771 :
0290 772 :     R0-R5, R10
0290 773 :
0290 774 : --
0290 775 :
0290 776 :
0290 777 CNV_MNEM DAY:
55 04 D0 0290 778 MOVL #4,R5 : DEFAULT LENGTH OF SUBSTRING
0290 779 : PATTERN TO MATCH.
55 56 91 0293 780 CMPB R6,R5 : IS THE INPUT STRING SHORTER?
0296 781 BGTRU 10$ : IF NOT, USE DEFAULT LENGTH.
55 56 D0 0298 782 MOVL R6,R5 : IF SO, USE INPUT STRING LENGTH.
0298 783 :
7E 67 D0 029B 784 10$: MOVL (R7),-(SP) : COPY FIRST FOUR LETTERS OF ASCII
5A 5E D0 029E 785 MOVL SP,R10 : STRING ONTO STACK. LOAD STACK ADDRESS
02A1 786 : INTO TEMPORARY REGISTER.
6A 80 8F 88 02A1 787 BISB #128,(R10) : SET THE HIGH BIT IN THE BYTE
02A5 788 : THAT HOLDS THE FIRST CHARACTER
02A5 789 : OF THE INPUT STRING.
FD6C CF 6A 20202020 8F CA 02A5 790 BICL #^X20202020,(R10) : Convert lower to upper case.
OC 6A 55 39 02AC 791 MATCHC R5,(R10),#DAY_TBL_LEN,DAY_TABLE
```



```
02B3 792
02B3 793
02B3 794
05 13 02B3 795
50 D4 02B5 796
009A 31 02B7 797
02BA 798
54 53 D0 02BA 799 20$: MOVL R3,R4
02BD 800
02BD 801
63 52 6A 55 39 02BD 802
0A 12 02C2 803
50 00038010 8F D0 02C4 804
0086 31 02CB 805
02CE 806
54 D7 02CE 807 30$: DECL R4
02D0 808
02D0 809
50 FD4B CF 9E 02D0 810
54 50 C2 02D5 811
02D8 812
02D8 813
54 04 C6 02D8 814
58 0B D0 02DB 815
02DE 816
7E 58 7D 02DE 817
02E1 818
02E1 819
02E1 820
02E1 821
53 5E D0 02E1 822
02E4 823
02E4 824
02E4 825
02E4 826
02E4 827
02F3 828
02F3 829
0B A9 302E303A 303A3020 4E 50 E9 02F3 830
8F 7D 02F6 831
0302 832
0302 833
63 08 C0 0302 834
0305 835
0305 836
52 08 AC D0 0305 837
0309 838
0309 839
0309 840
0309 841
0309 842
0314 843
2D 50 E9 0314 844
54 02 C4 0317 845
031A 846
82 FDOC CF44 C0 031A 847
0320 848
```

```
BEQL 20$
CLRL R0
BRW 80$
MOVL R3,R4
MATCHC R5,(R10),R2,(R3)
BNEQ 30$
MOVL #CLIS_ABKEYW,R0
BRW 80$
DECL R4
MOVAB DAY_TABLE,R0
SUBL2 R0,R4
DIVL2 #4,R4
MOVL #SHORT_TIME_LEN,R8
MOVQ R8,-(SP)
MOVL SP,R3
$ASCTIM_S -
TIMBUF=(R3)
BLBC R0,40$
MOVQ #^A/ 0:0:0.0/,SHORT_TIME_LEN(R9)
ADDL2 #MIDNIGHT_LEN,(R3)
MOVL 8(AP),R2
$BINTIM_S -
TIMBUF=(R3),-
TIMADR=(R2)
BLBC R0,40$
MULL2 #2,R4
ADDL2 TIME_TABLE[R4],(R2)+
```

: LOOK FOR A SUBSTRING MATCH BETWEEN
: THE INPUT STRING AND THE TABLE
: OF KEYWORDS.
: IF MATCH IS FOUND, CONTINUE.
: OTHERWISE, SET ERROR STATUS CODE
: AND BRANCH TO RETURN CODE.

: SAVE POSITION OF INPUT STRING
: BYTE THAT FOLLOWS LAST MATCHING
: CHARACTER.
: TEST FOR AMBIGUOUS KEYWORD.
: NO MATCH MEANS NO AMBIGUITY.
: MATCH MEANS KEYWORD IS AMBIGUOUS.
: RETURN WITH AN ERROR STATUS CODE.

: DECREMENT POINTER SO THAT IT
: POINTS TO THE LAST CHARACTER
: IN THE SUBSTRING MATCH.
: GET ADDRESS OF DAY TABLE
: SUBTRACT THE ADDRESS OF THE
: BEGINNING OF THE KEYWORD TABLE
: FROM THIS CHARACTER'S ADDRESS.
: COMPUTE INDEX OF THIS KEYWORD.
: SETUP STRING DESCRIPTOR FOR A
: BUFFER TO HOLD THE CURRENT TIME.
: ALLOCATE THE STRING DESCRIPTOR ON
: THE STACK. SAY THAT THE LENGTH
: OF THE BUFFER IS QUITE SHORT SO
: THAT HOURS/MINUTES/SECONDS ARE
: NOT RETURNED.
: LOAD ADDRESS OF THE STRING
: DESCRIPTOR INTO A REGISTER SO
: THAT IT CAN BE USED AS A SYSTEM
: SERVICE CALL ARGUMENT.
: ASK SYSTEM FOR AN ASCII STRING
: REPRESENTING THE CURRENT DAYTIME.
: SPECIFY THE STRING DESCRIPTOR
: JUST MADE ON THE STACK.
: BRANCH ON ERROR RETURN.

: ADD MIDNIGHT TIME SPECIFICATION TO
: TODAY'S DATE.
: NOW EXTEND THE BUFFER LENGTH
: REPRESENTED IN THE STRING
: DESCRIPTOR TO NEW STRING LENGTH.
: GET THE ADDRESS INTO WHICH THE
: 64-BIT TIME VALUE IS TO BE RETURNED.
: ASK SYSTEM FOR THE BINARY TIME
: EQUIVALENT OF THE ASCII TIME STRING.
: SPECIFY AN INPUT BUFFER AND
: AN OUTPUT ADDRESS INTO WHICH THE
: VALUE CAN BE WRITTEN.
: BRANCH ON ERROR.
: CONVERT THE INDEX INTO THE TIME
: TABLE INTO A LONGWORD INDEX.
: ADD THE TIME FACTOR FOR "YESTERDAY",
: "TODAY", OR "TOMORROW" TO THE

```
62 FDOA CF44 D8 0320 849 ADWC TIME_TABLE+4[R4],(R2) ; 64-BIT BINARY TIME VALUE.
    03 6C 91 0326 850 CMPB (AP),#3 ; 3RD ARGUMENT PRESENT?
    63 23 1F 0329 851 BLSSU 60$ ; BRANCH IF NOT
    2C D0 032B 852 MOVL #FULL_TIME_LEN,(R3) ; ALLOW THE OUTPUT ASCII STRING TO
    032E 853 ; BE THE FULL SIZE OF THE OUTPUT BUFFER.
    032E 854 ; THE ADJUSTED BINARY TIME VALUE BACK
    032E 855 $ASCTIM_S - ; TO AN ASCII STRING ONCE AGAIN.
    032E 856 TIMBUF=(R3),- ; SPECIFY THE STRING DESCRIPTOR FOR
    032E 857 - ; THE OUTPUT BUFFER
    032E 858 TIMADR=@8(AP),- ; AND THE ADDRESS OF THE TIME VALUE.
    032E 859 - ; OUTPUT BUFFER.
    032E 860
    032E 861 MOVQ (R3),R8 ; GET DESCRIPTOR OF OUTPUT STRING
    58 63 7D 033E 862 BLBS R0,50$ ; BRANCH ON SUCCESS.
    04 50 E8 0341 863
    0344 864
    0344 865
    58 D4 0344 866 40$: CLRL R8 ; ERROR HANDLING
    09 11 0346 867 ; RESET LENGTH FIELD OF OUTPUT STRING
    0348 868 ; DESCRIPTOR BACK TO ZERO.
    FDF1 30 0348 869 ; BRANCH TO RETURN CODE.
    03 50 E9 034B 870 50$: BSBW RETURN_STRING ; RETURN STRING TO CALLER
    034E 871 ; BRANCH IF COPY ERROR
    50 01 D0 034E 872 60$: MOVL #1,R0 ; MOVE A SUCCESS INDICATOR INTO R0.
    0351 873
    5E 08 C0 0351 874 70$: ADDL2 #8,SP ; RESTORE THE STACK TO ITS INITIAL
    0354 875 ; STATE.
    0354 876
    5E 04 C0 0354 877 80$: ADDL2 #4,SP ; MORE STACK RESTORATION.
    0357 878 ; RETURN TO CALLER.
    0358 879
    0358 880
    0358 881 .END
```


CVTTIME
Symbol table

- CONVERT CLI TIME TO BINARY

H 12

15-SEP-1984 23:39:39 VAX/VMS Macro V04-00
4-SEP-1984 23:15:28 [CLIUTL.SRC]CVTTIME.MAR;1

Page 20
(9)

\$\$\$1	= 0000008C	R	03
\$\$\$2	= 00000079	R	03
\$\$\$3	= 0000000D	R	02
CLIS_ABKEYW	= 00038010		
CLIS_IVVALU	= 00038088		
CLIS_NORMAL	= 00030001		
CNV_MNEM_DAY	00000290	R	02
CNV_PARSTRNG	00000203	R	02
CNV_RELOC	00000000	R	02
COLD	0000000A	R	02
COLDAY	00000004	R	02
COLHR	00000008	R	02
COLMIN	00000008	R	02
COLMON	00000005	R	02
COLYR	00000006	R	02
DAY	00000000	R	03
DAY_TABLE	0000001F	R	02
DAY_TBL_LEN	= 0000000C		
DCL\$SCOPY_DXDX	*****W	GX	02
DELTA_DAY	00000078	R	03
DIR...	= FFFFFFFF		
ENDD	0000000F	R	02
ENDDAY	0000000D	R	02
ENDHR	00000011	R	02
ENDMIN	00000014	R	02
ENDMON	0000000E	R	02
ENDSECS	0000001A	R	02
ENDYR	0000000F	R	02
EOS	= 00000000		
FILCNT	= 00000010		
FIRST_FLAG\$	= 00000000		
FULL_TIME_LEN	= 0000002C		
HOURS	0000003C	R	03
ILLCHR	0000001B	R	02
LIB\$CVT_ETIME	00000185	RG	02
LIB\$CVT_DTIME	00000180	RG	02
LIB\$CVT_TIME	00000044	RG	02
LIB\$SCOPY_DXDX	*****	X	02
MAXLGH	= 00000019		
MIDNIGHT_LEN	= 00000008		
MINDAY	0000001D	R	02
MINMON	0000001D	R	02
MINUTES	00000050	R	03
MONTH	00000014	R	03
NEXTSTATE	00000002		
NTOKNS	= 00000004		
RESC	= FFFFFFFE		
RETURN_STRING	0000013C	R	02
SCRATCHSIZE	FFFFFFFBC		
SECONDS	00000064	R	03
SHORT_TIME_LEN	= 0000000B		
SY\$ASCTIM	*****	GX	02
SY\$BINTIM	*****	GX	02
SY\$GETTIM	*****	GX	02
TBUF	= 0000002C		
TEMPDESC	FFFFFFFBC		
TEMPSTRING	FFFFFFFC4		

TIME TABLE
TOKENS
TSTRING
TS_LEN
YEAR

0000002B	R	02
00000000	R	02
00000000		
= 0000003C		
00000028	R	03

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	FFFFFFC4 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
_CNVCLI_CODE	00000358 (856.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC LONG
_CNVCLI_STATE	0000008C (140.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	18	00:00:00.07	00:00:01.05
Command processing	112	00:00:00.89	00:00:04.29
Pass 1	277	00:00:09.38	00:00:28.77
Symbol table sort	0	00:00:00.95	00:00:01.91
Pass 2	157	00:00:02.65	00:00:07.23
Symbol table output	8	00:00:00.07	00:00:00.07
Psect synopsis output	3	00:00:00.02	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	577	00:00:14.03	00:00:43.37

The working set limit was 1500 pages.
52624 bytes (103 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 619 non-local and 33 local symbols.
881 source lines were read in Pass 1, producing 27 object records in Pass 2.
19 pages of virtual memory were used to define 17 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
_\$255\$DUA28:[CLIUTL.OBJ]CLIUTL.MLB;1	2
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	0
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	9
TOTALS (all libraries)	11

700 GETS were required to define 11 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:CVTTIME/OBJ=OBJ\$:CVTTIME MSRC\$:CVTTIME/UPDATE=(ENH\$:CVTTIME)+EXECMLS/LIB+LIB\$:CLIUTL/LIB

0049

AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY