```
/***********************************************************************


                          SPOOL
                          TCOM.LST
                          05/04/82
                          15:54:00



***********************************************************************/
```

```
SERIES-III 8086/8087/8088 MACRO ASSEMBLER V1.1 ASSEMBLY OF MODULE TCOM
OBJECT MODULE PLACED IN :F1:TCOM.OBJ
ASSEMBLER INVOKED BY:  ASM86.86 :F1:TCOM.A86 PRINT(:F1:TCOM.LST) DEBUG


LOC  OBJ                  LINE     SOURCE

                              +1  $TITLE('iLNA TCL Common Routines                    11/15/81 14:10')
                                          NAME    tcom
                              +1  $include (:f1:cpyrt.dca)
                        =1        ;
                        =1        ;                 /* Intel Corporation Proprietary Information.
                        =1        ;                 This listing is supplied under the terms of a
                        =1        ;                 license agrement with Intel Corporaton and
                        =1        ;                 may not be copied nor disclosed except in
                        =1        ;                 accordance with the terms of that agreement. */
                        =1        ;
                                  ;
                                  ;                         George D Marshall x7-5117
                                  ;
                                  ;         This is a collection of routines common to all TCL code,
                                  ;         crunched into assembler for code size reduction.
                                  ;         The first several routines are the 64k-modulo arithmetic used in
                                  ;         TCL sequence number operations; the remainder are various
                                  ;         small common routines that lent themselves to being re-implemented
                                  ;         in assembler for code reduction and spped increases.
                                  ;
                                  ;DECLARE
                                  ;    cur$max$cdbs         BYTE    EXTERNAL,
                                  ;    lcid$vector(max$cdbs$lit) WORD  EXTERNAL;   /* list of allocated CIDs */

                                  DGROUP  GROUP   DATA
----                              DATA    SEGMENT PUBLIC  'DATA'
                                          EXTRN   curmaxcdbs:BYTE,lcidvector:WORD
                                          EXTRN   rp_:WORD         ; in RP
----                              DATA    ENDS

                                  CGROUP  GROUP   CODE
----                              CODE    SEGMENT PUBLIC  'CODE'
                                          ASSUME  CS:CGROUP,DS:DGROUP

                                          PUBLIC  gtmod64k,gemod64k,maxmod64k,min,max
                                          PUBLIC  chksumcalc,search_lcidvector,DLSOURCE_EQ_HOST,stky_incr

                                  ;                                                    /********************/
                                  ;                                                    /**     gt$mod64k    */
                                  ;                                                    /********************/
                                  ;gt$mod64k: PROCEDURE(n,m) BYTE PUBLIC;
                                                                  ; Routine to return TRUE if n > m,
                                                                  ; modulo 64K. Test is conservative- if
                                                                  ; (n-m)=8000H, then result is false */
0000                              GTMOD64K:

                                  ;DECLARE (n,m) WORD;
                                  ;IF ((n - m) <> 0)   AND   ((n-m) < 8000H) THEN RETURN(true);
0000 59                                   POP     cx       ; get return address
0001 5B                                   POP     bx       ; m
```

```
LOC  OBJ            SOURCE

0002 58                     POP     ax      ; n
0003 2BC3                   SUB     ax,bx   ; ax <- n-m
0005 7408                   JZ      gt2     ; return false if n-m=0 (ax already has zero)
0007 7804                   JS      gt1     ; return false if n-m >= 8000 (i.e., sign bit on)
0009 B0FF                   MOV     AL,0FFH ; return true
000B FFE1                   JMP     cx      ; return to caller
                    ;ELSE RETURN(false);
000D B000           gt1:    MOV     AL,0H
000F FFE1           gt2:    JMP     cx      ; return to caller
                    ;END gt$mod64k;

                    ;                                                     /*******************/
                    ;                                                     /**    ge$mod64k    */
                    ;                                                     /*******************/
                    ;ge$mod64k: PROCEDURE(n,m) BYTE PUBLIC;
0011                GEMOD64K:
                                                   ; /* Routine to return TRUE if n >= m,
                                                   ; modulo 64K. Test is conservative- if
                                                   ; (n-m)=8000H, then result is false */
                    ;DECLARE (n,m) WORD;
                    ;IF (n-m) < 8000H THEN RETURN(true);
                    ;ELSE RETURN(false);
0011 59                     POP     cx      ; get return address
0012 5B                     POP     bx      ; m
0013 58                     POP     ax      ; n
0014 2BC3                   SUB     ax,bx   ; ax <- n-m
0016 7804                   JS      ge1     ; return false if n-m >= 8000H (i.e., sign bit is on)
0018 B0FF                   MOV     AL,0FFH ; set "true" return value
001A FFE1                   JMP     cx      ; return to caller
001C B000           ge1:    MOV     AL,0H   ; set "false" return value
001E FFE1                   JMP     cx      ; return to caller
                    ;END ge$mod64k;

                    ;                                                     /*******************/
                    ;                                                     /**    max$mod64k   */
                    ;                                                     /*******************/
                    ;max$mod64k: PROCEDURE(n,m) WORD PUBLIC;
                                                   ; Routine to return the higher of n and m,
                                                   ; modulo 64K. */
0020                MAXMOD64K:
                    ;DECLARE (n,m) WORD;
                    ;IF (n-m) < 8000H THEN RETURN(n);  ELSE RETURN(m);
0020 59                     POP     cx      ; return address
0021 5B                     POP     bx      ; m
0022 58                     POP     ax      ; n
0023 3BC3                   CMP     ax,bx   ; set flags for n-m
0025 7902                   JNS     maxm1   ; return n if (n-m) < 8000 (sign bit off)
0027 8BC3                   MOV     ax,bx   ; set return value = m
0029 FFE1           maxm1:  JMP     cx      ; return to caller

                    ;END max$mod64k;


                    ;                                                     /***********/
                    ;                                                     /***  min  **/
```

```
LOC  OBJ          SOURCE

                  ;                                                   /************/
                  ;min: PROCEDURE(n,m) WORD PUBLIC;
                                                ; routine to return minimum of two
                                                ; unsigned word values (not modulo).
002B              MIN:
                  ;DECLARE (n,m) WORD;
                  ;IF n < m THEN RETURN(n);
002B 59                   POP     cx      ; return address
002C 5B                   POP     bx      ; m
002D 58                   POP     ax      ; n
002E 3BC3                 CMP     ax,bx   ; set flags for n-m
0030 7202                 JB      min1    ; jump if n is less - its in ax for return
                  ;          ELSE RETURN(m);
0032 8BC3                 MOV     ax,bx   ; set return value to m
                  ;END min;
0034 FFE1         min1:   JMP     cx      ; return to caller


                  ;                                                   /****************/
                  ;                                                   /****   max   ****/
                  ;                                                   /****************/
                  ;max: PROCEDURE(n,m) WORD PUBLIC;
                                                ; routine to return maximum of two
                                                ; unsigned word values (not modulo).
0036              MAX:
                  ;DECLARE (n,m) WORD;
                  ;IF n > m THEN RETURN (n);
0036 59                   POP     cx      ; return address
0037 5B                   POP     bx      ; m
0038 58                   POP     ax      ; n
0039 3BC3                 CMP     ax,bx   ; set flags for n-m
003B 7702                 JA      max1    ; jump if n is greater - its in ax for return
                  ;          ELSE RETURN(m);
003D 8BC3                 MOV     ax,bx   ; set return value to m
                  ;END max;
003F FFE1         max1:   JMP     cx      ; return to caller

                  ;                                                   /********************/
                  ;                                                   /*** chk$sum$calc ***/
                  ;                                                   /********************/
                  ;chk$sum$calc: PROCEDURE(seg$o) WORD PUBLIC;
0041              CHKSUMCALC:
                                                ; This routine calculates the header
                                                ; checksum on the send or receive segment
                                                ; indicated by the offset supplied.  For
                                                ; send segments, all header fields must have
                                                ; been set prior to calling this routine.
                                                ; Note: checksum does not cover Data Link
                                                ; header, since there can be several DLLs,
                                                ; and a Network header is not yet defined. */
                  ;DECLARE
                  ;    chksum       WORD,
                  ;    seg$o        WORD,
                  ;    max$o        WORD,             /* max offset in seg to checksum */
                  ;    curr$wd$o    WORD,             /* ptr to current seg header wd */
```

```
LOC  OBJ                  SOURCE

                          ;     curr$wd BASED curr$wd$o  WORD,  /* the current header word being added */
                          ;                                     /* to the checksum */
                          ;     seg          BASED seg$o
                          ;$IF f7
                          ;$NOLIST INCLUDE (:f1:TCLSEG.INC)
                          ;$ELSE
                          ;$ENDIF
                          ;     ;

----                      segbuf  STRUC            ; segment buffer structure - must track TCLSEG.INC
0000                      kaosptr DD      ?        ; kaos field
0004                      buflen  DW      ?        ;  "      "
0006                      desth0  DW      ?        ; DLL field: destination host id - first word
0008                      desth1  DW      ?
000A                      desth2  DW      ?
000C                      srch0   DW      ?        ; DLL field: source host ID
000E                      srch1   DW      ?
0010                      srch2   DW      ?
0012                      dltype  DW      ?        ; DLL field: type field
0014                      tclversion DW   ?        ; first field of TCL header
----                      segbuf  ENDS

  0009                    tclheaderwds    EQU     9        ; number of 16-bit words in tcl header,
                                                           ; not counting the checksum.  This must
                                                           ; match the segment definition in file
                                                           ; TCLSEG.INC.
                                                           ; NOTE: TCLGBL.INC has tclheaderlen in bytes,
                                                           ;       including the checksum field.

                          ;curr$wd$o = .seg.tcl$version;        /* set base to first byte of TCL header */
0041 5A                           POP     dx       ; return address - NOTE cx not used as above due to LOOP
0042 5B                           POP     bx       ; segment buffer offset
0043 8D5F14                       LEA     bx,[bx].tclversion       ; bx holds currwdo
                          ;                                     /* max offset to sum is up to but not */
                          ;max$o = curr$wd$o + (tcl$header$len - 2);   /* including chksum wd */
0046 B90900                       MOV     cx,tclheaderwds ; NOTE: asm version exploits loop instruction,
                                                           ; so we set up an index count instead of an
                                                           ; offset limit. CX holds this.
                          ;chksum = 0;
0049 33C0                         XOR     ax,ax            ; set to zero - AX holds cummulative checksum
                          ;DO forever;                          /* Note: loop has been arranged to make */
                          ;     chksum = chksum + curr$wd;      /* this version of compiler generate efficient*/
004B 0307                 chk1:   ADD     ax,[bx]          ; add current word to checksum

                          ;     curr$wd$o = curr$wd$o + 2;      /* code-- don't change it without checking */
004D 83C302                       ADD     bx,2             ; bump offset to next word by two bytes

                          ;     IF curr$wd$o >= max$o THEN RETURN(chksum);  /* the resultant code. */
0050 E2F9                         LOOP    chk1             ; decr cx, jmp if not done
0052 FFE2                         JMP     dx               ; return to caller
                          ;END;

                          ;END chk$sum$calc;

                          ;                                         /***************************/
```

```
LOC  OBJ               SOURCE

                       ;                                                  /**** search_lcidvector *****/
                       ;                                                  /***************************/
                       ;search_lcidvector: PROCEDURE(find$target) WORD PUBLIC;
                                                               ; This routine accepts an alleged
                                                               ; connection ID, and searches the local
                                                               ; connection ID vector (lcidvector) to try
                                                               ; to find a match for it, returning OFFFFH
                                                               ; if no match, or the index into the vector
                                                               ; if it is found.
                                                               ; (This is really a customized FINDW; a code-
                                                               ; saver routine to avoid having the FINDW
                                                               ; code expanded in-line multiple places in
                                                               ; the code, and to eliminate having to push
                                                               ; common parameters on stack)
0054                   search_lcidvector:
                       ;DECLARE
                       ;    find$target WORD;
                       ;RETURN( FINDW(@lcidvector, find$target, max$cdbs) );

0054 5B                        POP     bx       ; return address
0055 BF0000        E           MOV     DI,OFFSET(LCIDVECTOR)
0058 58                        POP     ax       ; findtarget is in ax
0059 8A0E0000      E           MOV     CL,CURMAXCDBS
005D B500                      MOV     CH,0H
005F 1E                        PUSH    DS        ; 1
0060 07                        POP     ES        ; 1
0061 FC                        CLD
0062 8BD1                      MOV     DX,CX
0064 E306                      JCXZ    $+8H
0066 F2                        REPNZ   SCASW
0067 AF
0068 75F8                      JNZ     $-6H
006A 2BD1                      SUB     DX,CX
006C 4A                        DEC     DX
006D 8BC2                      MOV     AX,DX
006F FFE3                      JMP     bx        ; return to caller
                       ;END search_lcidvector;

                       ;                                                       /************************/
                       ;                                                       /**  dlsource_eq_host   **/
                       ;                                                       /************************/
                       ; dlsource_eq_host: PROCEDURE(host$p) BYTE PUBLIC;
0071                   DLSOURCE_EQ_HOST:
                       ; DECLARE
                       ;      host$p  POINTER;
                                                               ; code-saver routine to compare host ids
                                                               ; for the RP process
                                                               ; Logic has been re-done to elimate
                                                               ; the test for OFFFF required by PL/M's
                                                               ; CMPW builtin.
                       ; IF CMPW( @rp_.dl$source0, host$p, 3) = OFFFFH   /* order of operands affects code s
                       ize */
0071 5A                        POP     dx        ; return address
                                                 ; STATEMENT # 617
0072 BE0000        E           MOV     SI,OFFSET(RP_)
```

```
LOC  OBJ                  SOURCE

0075 5F                           POP     di       ; get offset part of hostp
0076 07                           POP     es       ; get base part of hostp
0077 B90300                       MOV     CX,3H    ; set up count register for compare
007A FC                           CLD
007B F3                           REPZ    CMPSW
007C A7
007D 7504                         JNZ     dls1     ; jump if no match
                          ;      THEN RETURN(true);
                                                   ; STATEMENT # 618
007F B0FF                         MOV     AL,0FFH  ; return value = true
0081 FFE2                         JMP     dx       ; return to caller
                          ; ELSE RETURN(false);
                                                   ; STATEMENT # 619
0083 B000                 dls1:   MOV     AL,0H    ; return value = false
0085 FFE2                         JMP     dx;      return to caller
                          ; END dlsource_eq_host;

                          ;                                                   /*******************/
                          ;                                                   /**    stky_incr   */
                          ;                                                   /*******************/
                          ;
                                                           ; Routine to do a "sticky increment"
                                                           ; of a word based on the pointer
                                                           ; argument, and return the result.

                          ;stky_incr: PROCEDURE(wd$p) PUBLIC;
0087                      STKY_INCR:
0087 59                           POP     cx       ; return address
0088 5B                           POP     bx       ; offset of argument
0089 07                           POP     es       ; base of argument
                          ;DECLARE
                          ;    wd$p    POINTER,
                          ;    wd  BASED wd$p  WORD;
                          ;IF wd <> 0FFFFH THEN wd = wd + 1;
008A 26813FFFFF                   CMP     ES:[BX],0FFFFH
008F 7403                         JZ      stky1
0091 26FF07                       INC     WORD PTR ES:[BX]
                          ;END stky_incr;
0094 FFE1                 stky1:  JMP     cx       ; return to caller
                          ;

----                      CODE    ENDS
                                  END

ASSEMBLY COMPLETE, NO ERRORS FOUND
```

```
/************************************************************************

                        SPOOL
                        GETCHK.LST
                        05/04/82
                        15:54:46


************************************************************************/
```

```
SERIES-III PL/M-86 V2.0 COMPILATION OF MODULE GETCHK
OBJECT MODULE PLACED IN :F1:GETCHK.OBJ
COMPILER INVOKED BY:  PLM86.86 :F1:GETCHK.P86 OPTIMIZE(3) XREF SET(F1) DEBUG


              $TITLE('iLNA TCL  Get and Check Address from mipform      11/23 14:00')
              $COMPACT DEBUG NOCOND
*** WARNING 10 IN 1 (LINE 2): RESPECIFIED PRIMARY CONTROL, IGNORED

              $IF f7
              $ELSE
              $INCLUDE (:F1:copyrt.dcp)

         =            /* Intel Corporation Proprietary Information.
         =            This listing is supplied under the terms of a
         =            license agrement with Intel Corporaton and
         =            may not be copied nor disclosed except in
         =            accordance with the terms of that agreement. */

              $ENDIF



                                            /* routine to convert a "mipform" of
                                            address from host software into a plm
                                            pointer, and check that a legal address
                                            was obtained.  Used by TCL and NML */
   1          getchk: DO;

              $IF f7
              $ELSE
              $INCLUDE (:F1:MIP.DCP)
   2   1   =  cq$mipsend: PROCEDURE(socket, msg$p) BYTE EXTERNAL;
   3   2   =      DECLARE socket  WORD,
           =              msg$p   POINTER;
   4   2   =      END cq$mip$send;

   5   1   =  cq$mipconnect: PROCEDURE(portid, mbx$o) BYTE EXTERNAL;
   6   2   =      DECLARE portid  BYTE,
           =              mbx$o   WORD;
   7   2   =      END cq$mipconnect;

   8   1   =  cq$mip$register: PROCEDURE(procedure$o) BYTE EXTERNAL;
   9   2   =      DECLARE procedure$o WORD;
  10   2   =      END cq$mip$register;

  11   1   =  cq$mip$get$address: PROCEDURE(mip_form) POINTER EXTERNAL;
  12   2   =      DECLARE mip_form    POINTER;
  13   2   =      END cq$mip$get$address;

  14   1   =  cq$mip$get$mip$form: PROCEDURE(ptr) POINTER EXTERNAL;
  15   2   =      DECLARE ptr POINTER;
  16   2   =      END cq$mip$get$mip$form;
              $ENDIF
              $RESTORE
  17   1        get$chk$address: PROCEDURE(mipform$p, ptr$o) BYTE PUBLIC;
  18   2        DECLARE
                    mipform$p    POINTER,              /* the address to be converted, in MIP FORM */
                    ptr$o        WORD,                 /* offset of ptr to store converted value */
                    target$ptr BASED ptr$o  POINTER;/* the pointer we will write into */

  19   2        target$ptr = cq$mip$get$address(mipform$p);
  20   2        IF SELECTOR$OF(target$ptr) = 0FFFFH THEN RETURN(0); /* return false if bad address */
  22   2        ELSE RETURN(0FFH);                     /* return true if ok */
  23   2        END get$chk$address;

  24   1        END get$chk;
```

```
                     CROSS-REFERENCE LISTING

   DEFN  ADDR   SIZE  NAME, ATTRIBUTES, AND REFERENCES
  ----- ------ -----  --------------------------------


      5  0000H        CQMIPCONNECT . . .    PROCEDURE BYTE EXTERNAL(1) STACK=0000H
     11  0000H        CQMIPGETADDRESS. .    PROCEDURE POINTER EXTERNAL(3) STACK=0000H       19
     14  0000H        CQMIPGETMIPFORM. .    PROCEDURE POINTER EXTERNAL(4) STACK=0000H
      8  0000H        CQMIPREGISTER. . .    PROCEDURE BYTE EXTERNAL(2) STACK=0000H
      2  0000H        CQMIPSEND. . . . .    PROCEDURE BYTE EXTERNAL(0) STACK=0000H
         0000H        GETCHK . . . . . .    PROCEDURE STACK=0000H
     17  0000H    36  GETCHKADDRESS. . .    PROCEDURE BYTE PUBLIC STACK=000EH
      6  0000H     2  MBXO . . . . . . .    WORD IN PROC (CQMIPCONNECT) PARAMETER           6
     18  0006H     4  MIPFORMP . . . . .    POINTER IN PROC (GETCHKADDRESS) PARAMETER AUTOMATIC         18   19
     12  0000H     4  MIP_FORM . . . . .    POINTER IN PROC (CQMIPGETADDRESS) PARAMETER           12
      3  0000H     4  MSGP . . . . . . .    POINTER IN PROC (CQMIPSEND) PARAMETER           3
      6  0000H     1  PORTID . . . . . .    BYTE IN PROC (CQMIPCONNECT) PARAMETER           6
      9  0000H     2  PROCEDUREO . . . .    WORD IN PROC (CQMIPREGISTER) PARAMETER          9
     15  0000H     4  PTR. . . . . . . .    POINTER IN PROC (CQMIPGETMIPFORM) PARAMETER           15
     18  0004H     2  PTRO . . . . . . .    WORD IN PROC (GETCHKADDRESS) PARAMETER AUTOMATIC      18   20
                      SELECTOROF . . . .    BUILTIN          20
      3  0000H     2  SOCKET . . . . . .    WORD IN PROC (CQMIPSEND) PARAMETER          3
     18  0000H     4  TARGETPTR. . . . .    POINTER BASED(PTRO) IN PROC (GETCHKADDRESS)           19*  20


MODULE INFORMATION:

     CODE AREA SIZE     = 0024H     36D
     CONSTANT AREA SIZE = 0000H      0D
     VARIABLE AREA SIZE = 0000H      0D
     MAXIMUM STACK SIZE = 000EH     14D
     64 LINES READ
     1 PROGRAM WARNING
     0 PROGRAM ERRORS

END OF PL/M-86 COMPILATION
```