เอกสารอ้างอิง

1. Hydra II Reference Guide, JDS Microprocessing, 1986.
2. Dimitri Bertsekas and Robert Gallager, Data Networks, pp. 1-103, Prentice-Hall, New Jersey, 1987.
3. Dixon R. Doll, Data Communication, pp. 259-274, Wiley, New York, 1978.
4. Macintosh User Education , Apple Computer Inc. ,California, 1985.
5. VM/System Product CMS Primer., 4th ed. International Business Machines Corporation, 1986.
6. W. David Schwaderer, "CRC Calculation," PC Tech Journal, pp. 118-133, April 1985.
7. LightspeedC User's Manual, 2nd ed., Think Technology Inc., Massachusetts, 1986.
9. กุณฑิรา สมุทรกลิน, "การพัฒนาการจัดเก็บแฟ้มข้อมูลของโปรแกรมสำเร็จรูปแมคดราฟท์," วิทยานิพนธ์มหาบัณฑิต, ภาควิชาวิศวกรรมคอมพิวเตอร์ บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย, 2531.
9. อาจิน จีรชีพพัฒนา, "การพัฒนาโปรแกรมสำเร็จรูปสำหรับสร้างภาพให้สอดคล้องกับคำบรรยายจากเทปบันทึกเสียง," วิทยานิพนธ์มหาบัณฑิต, ภาควิชาวิศวกรรมคอมพิวเตอร์ บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย, 2530.

ภาคผนวก ก

ขั้วต่อสายสัญญาณพอร์ทอนุกรมของแมคอินทอช

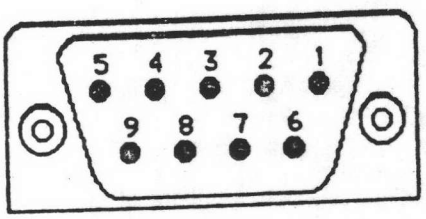
Modem and Printer Ports

Pin No.	Signal Name	Signal Description
1	HSKo	Handshake output. Connected to SCC Data Terminal Ready.
2	KSKi	Handshake input. Connected to SCC Clear to Send and Transmit/Receive Clock.
3	TxD-	Transmit Data (inverted). Connected to SCC transmit Data. Tri-stated when Request To Send is deasserted.
4	SG	Signal Ground. Connected to logic and chassis ground.
5	RxD-	Receive Data (inverted). Connected to SCC Receive Data.
6	TxD+	Transmit Data. Connected to SCC Transmit Data. Tri-stated when Request To Send is deasserted.
7	GPi	General-Purpose input. Connected to SCC Data Carrier Detect.
8	RxD+	Receive Data. Connected to the SCC Receive Data.

Connector Type: Mini DIN-8

Modem port only: Connected to Receive/Transmit clock if VIA1 SYNC signal is high.

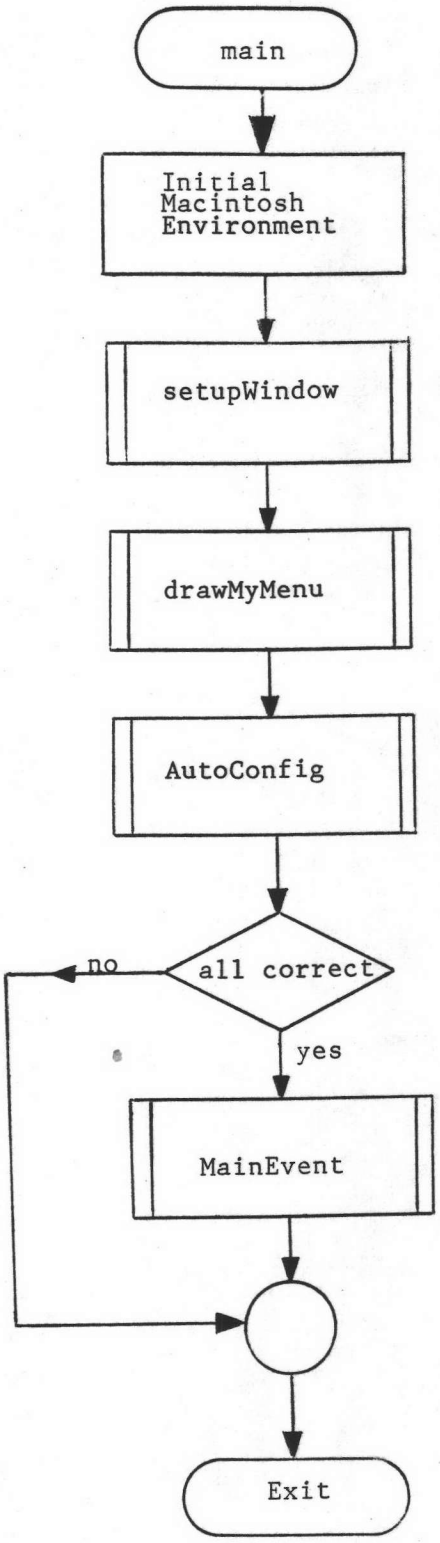
To connect DE-9 cables to the Mini DIN-8 port, use adapter cable 590-0341.



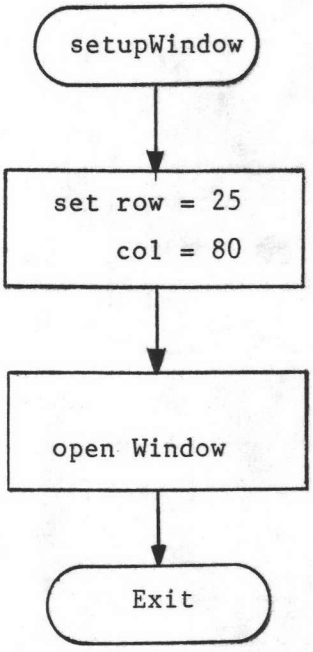
1	Ground
2	+5 volts
3	Ground
4	Transmit data +
5	Transmit data -
6	+12 volts
7	Handshake/external clock
8	Receive data +
9	Receive data -

DB-9 pinout for SCC Output jacks

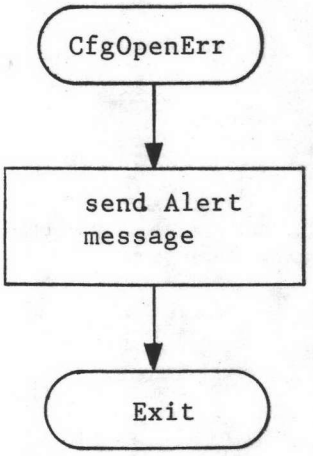
ภาคผนวก ข
ผังงานของโปรแกรมรับส่งข้อมูล



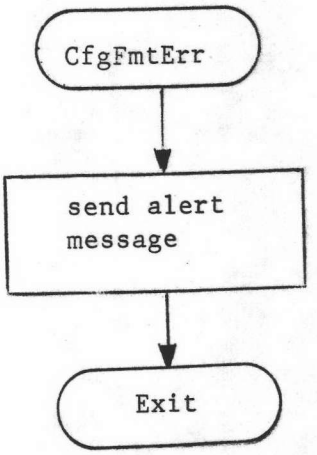
รูปที่ ข.1 ผังงานของฟังก์ชั่น main



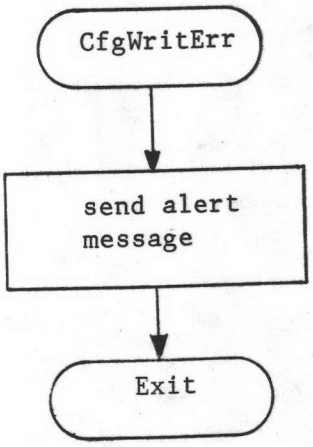
รูปที่ ข.2 ผังงานของฟังก์ชั่น setupWindow



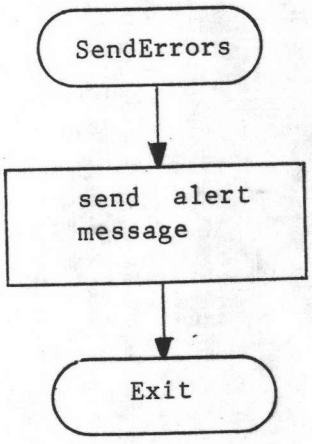
รูปที่ ข.3 ผังงานของฟังก์ชั่น CfgOpenErr



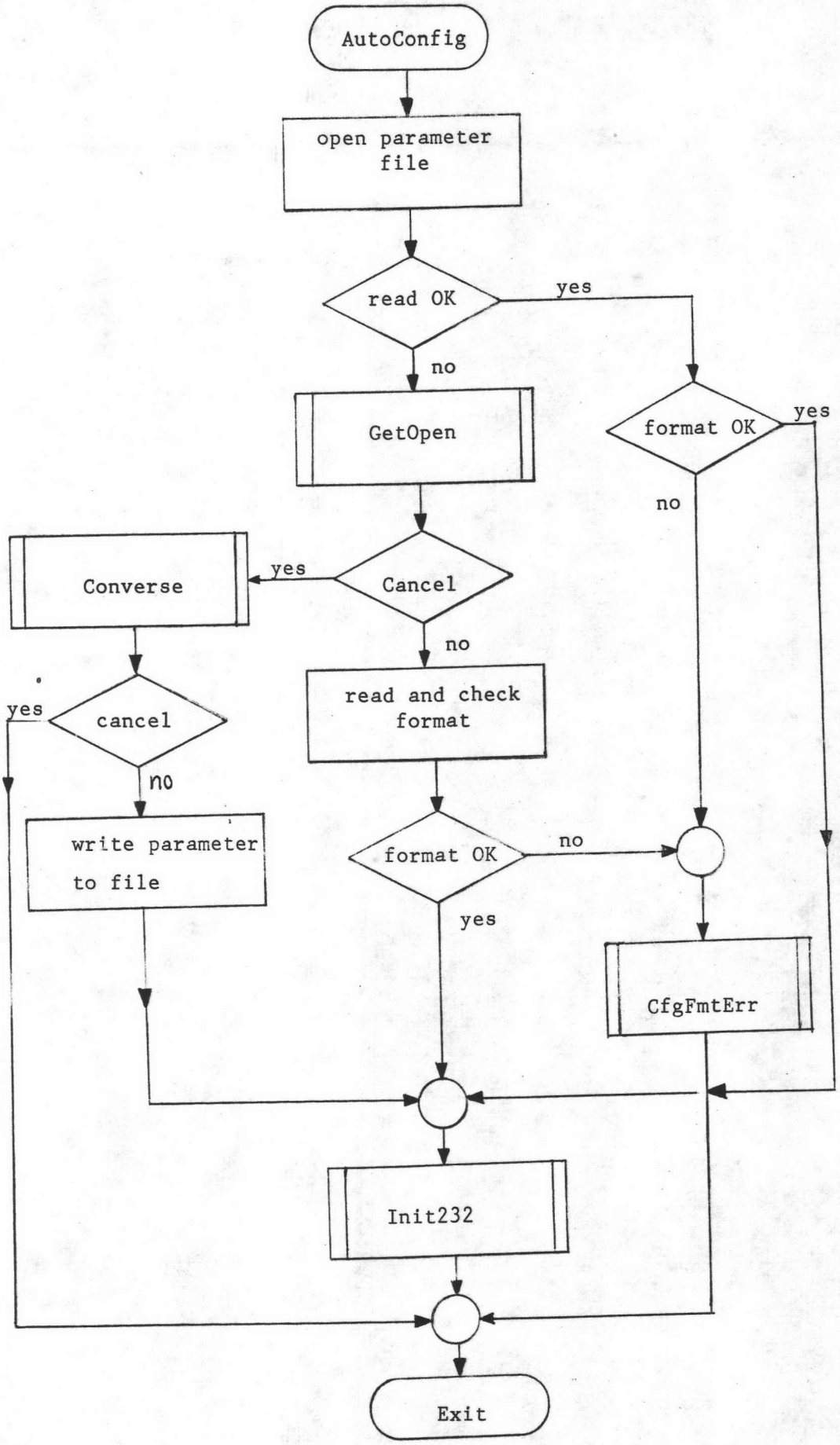
รูปที่ ข.4 ผังงานของฟังก์ชั่น CfgFmtErr



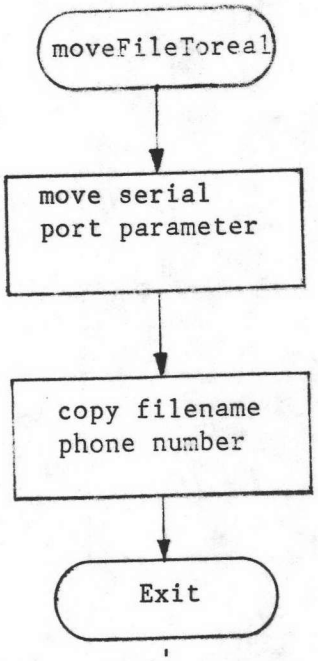
รูปที่ ข.5 ผังงานของฟังก์ชั่น CfgWritErr



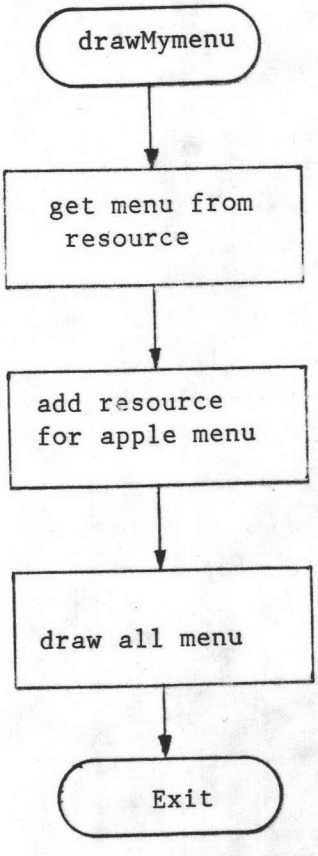
รูปที่ ข.6 ผังงานของฟังก์ชั่น SendErrors



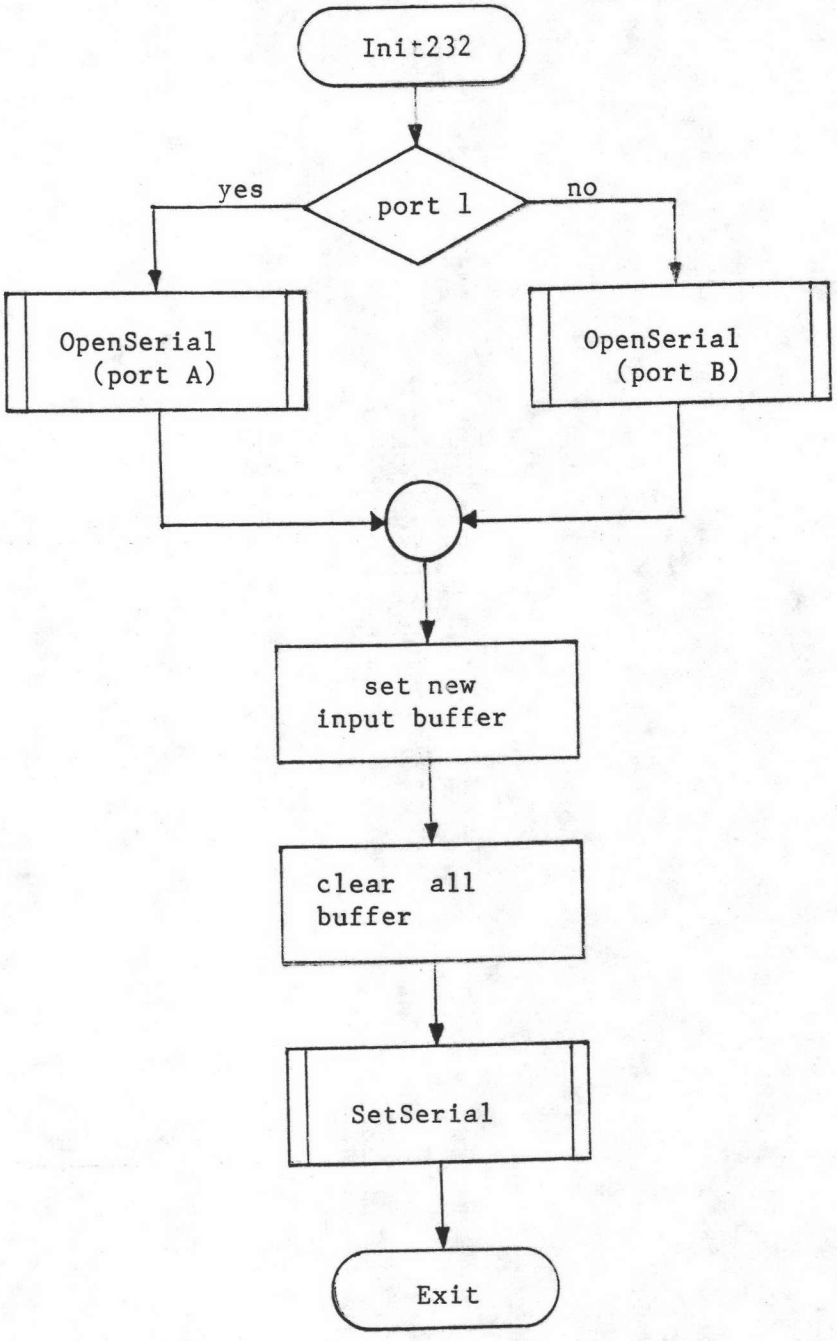
รูปที่ ข.7 ผังงานของฟังก์ชั่น AutoConfig



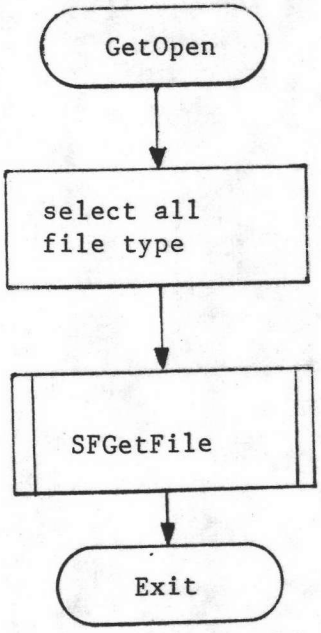
รูปที่ ข.8 ผังงานของฟังก์ชัน moveFileToReal



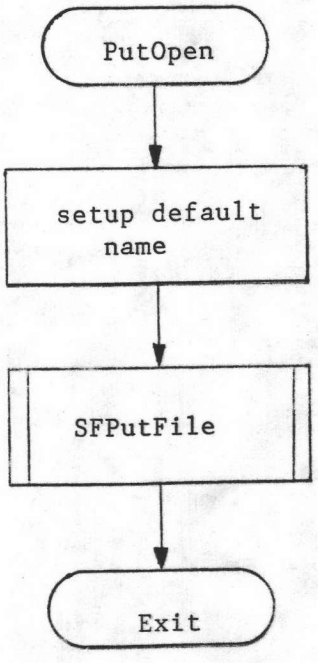
รูปที่ ข.9 ผังงานของฟังก์ชัน drawMyMenu



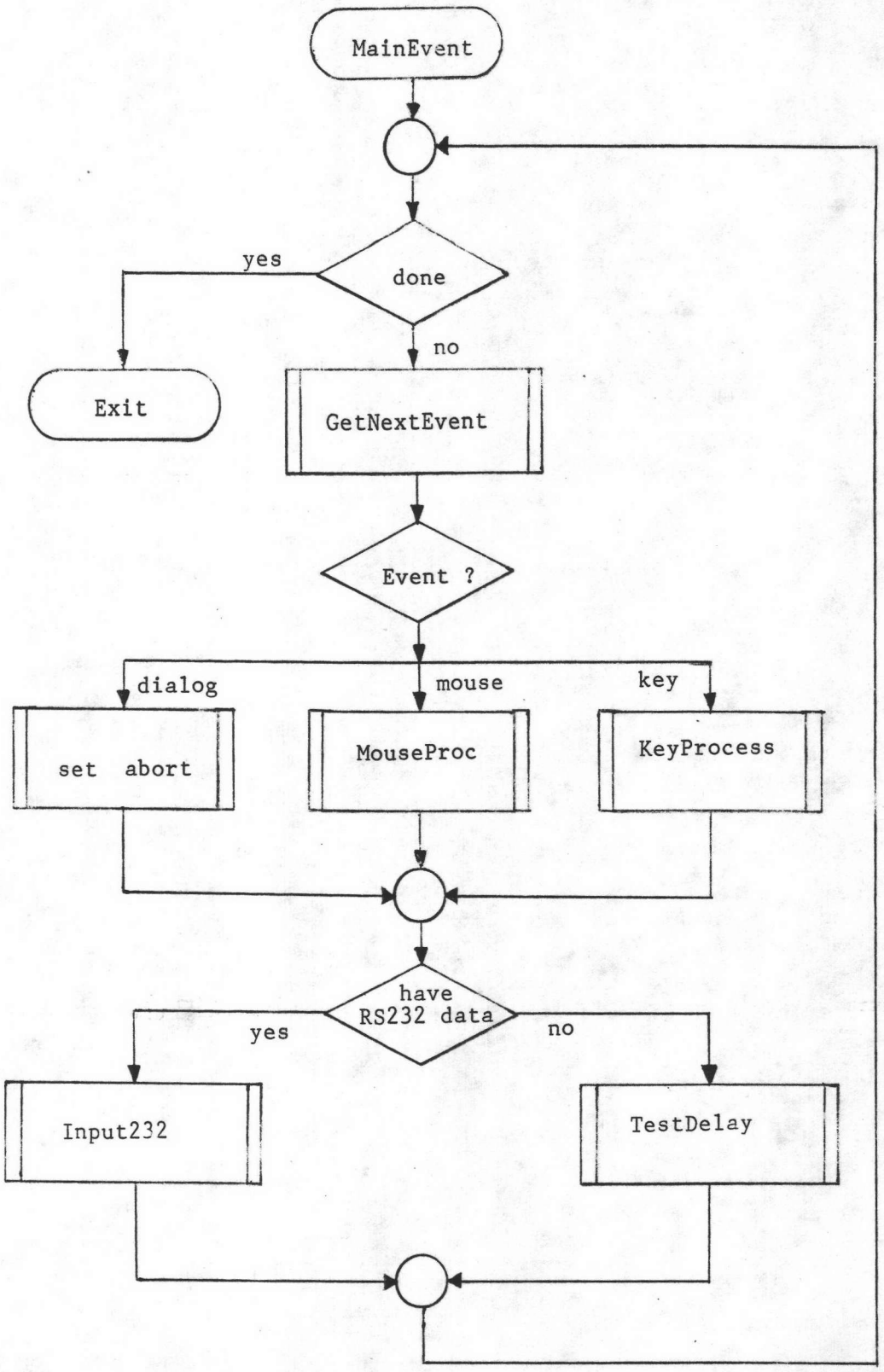
รูปที่ ข.10 ผังงานของฟังก์ชั่น Init232



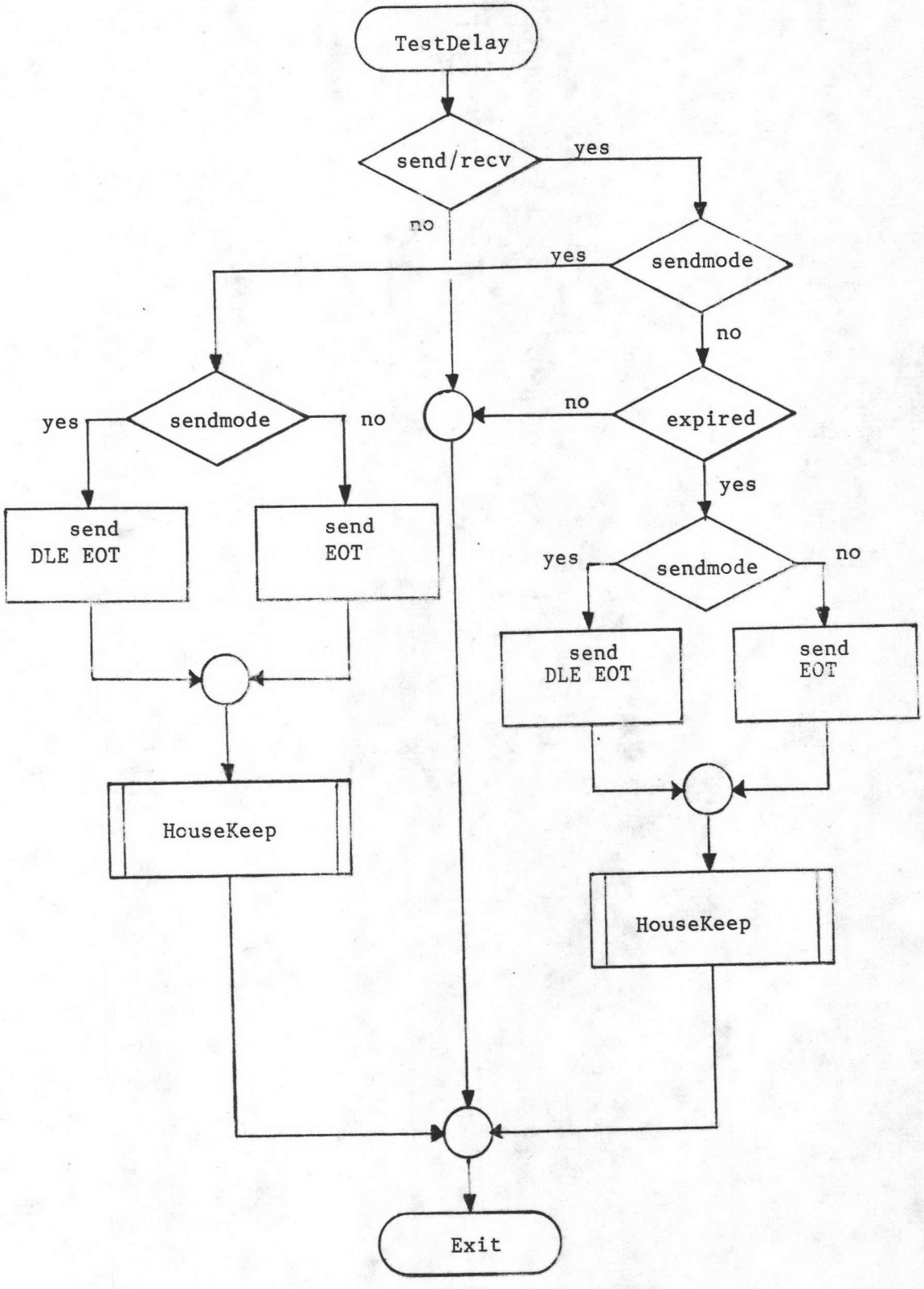
รูปที่ ข.11 ผังงานของฟังก์ชั่น GetOpen



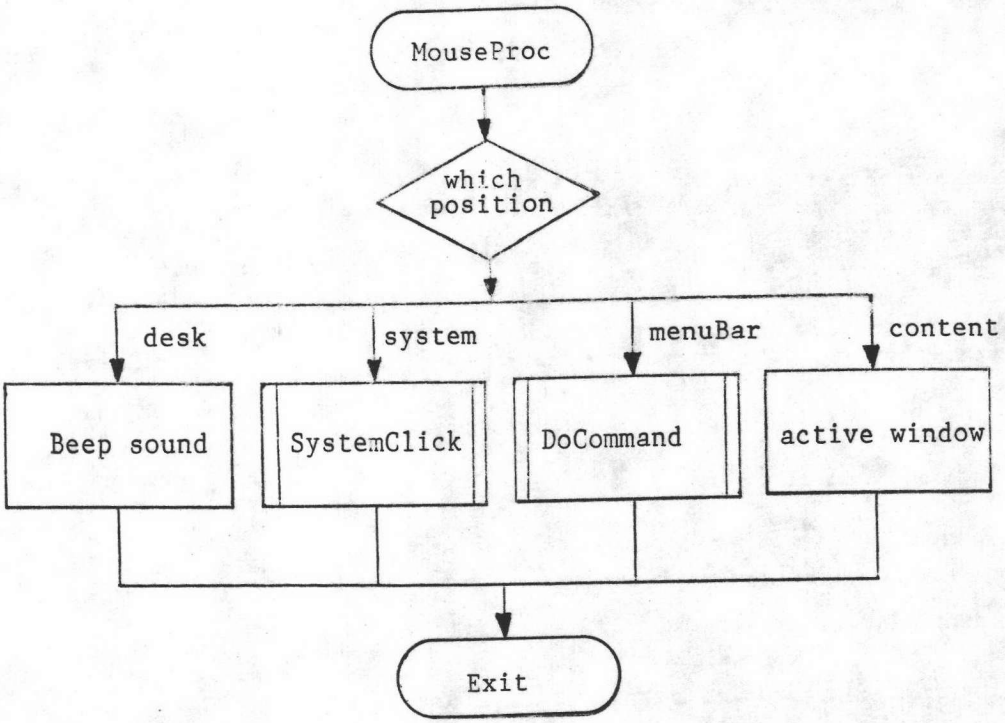
รูปที่ ข.12 ผังงานของฟังก์ชั่น PutOpen



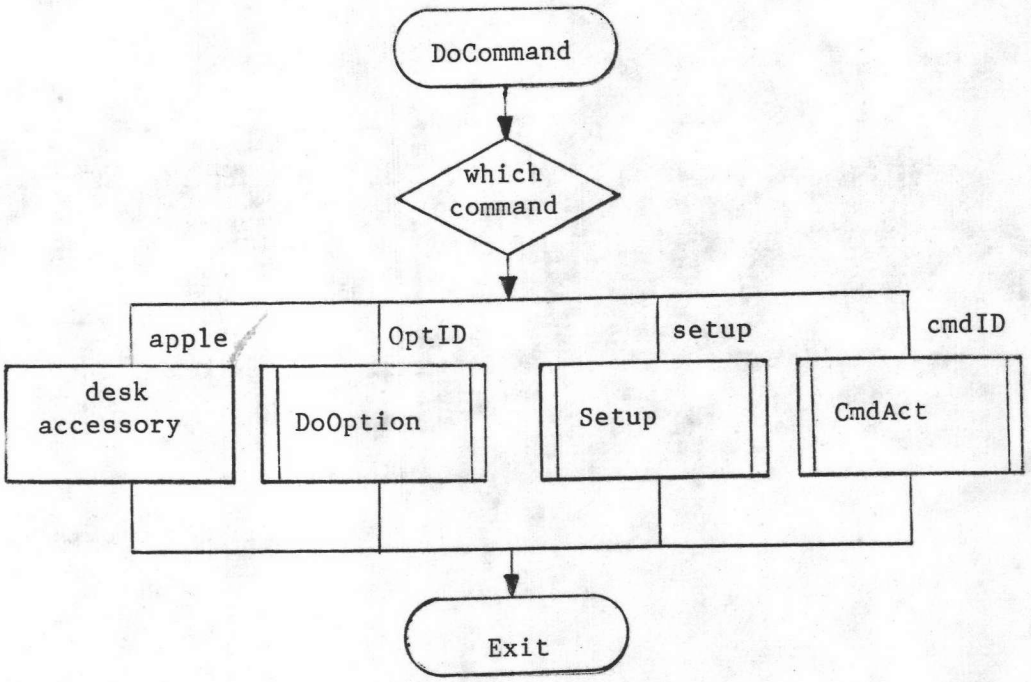
รูปที่ ข.13 ผังงานของฟังก์ชั่น MainEvent



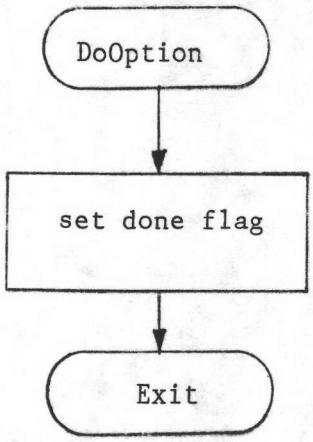
รูปที่ ข.14 ผังงานของฟังก์ชั่น TestDelay



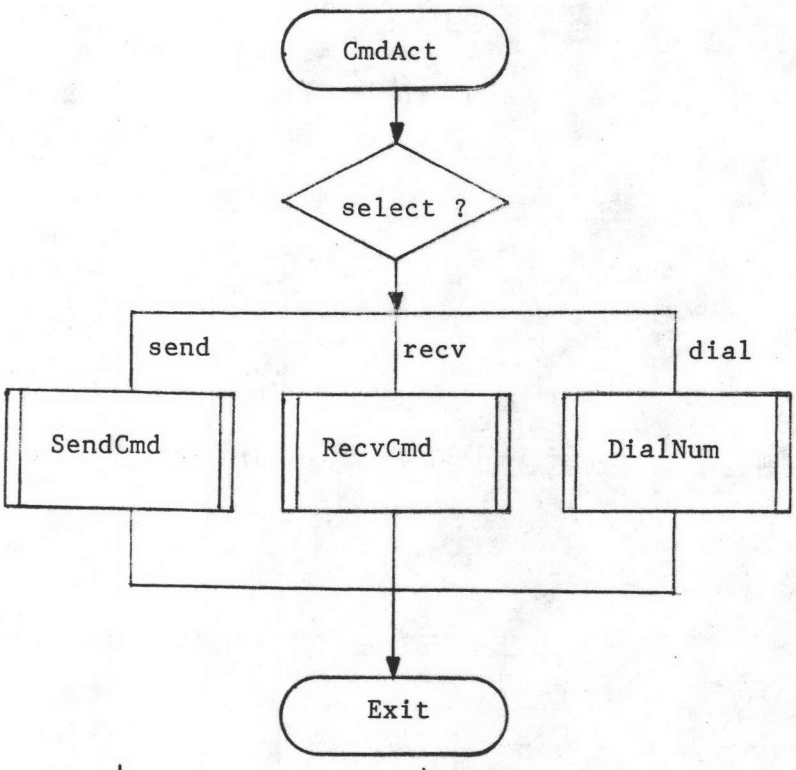
รูปที่ ข.15 ผังงานของฟังก์ชั่น MouseProc



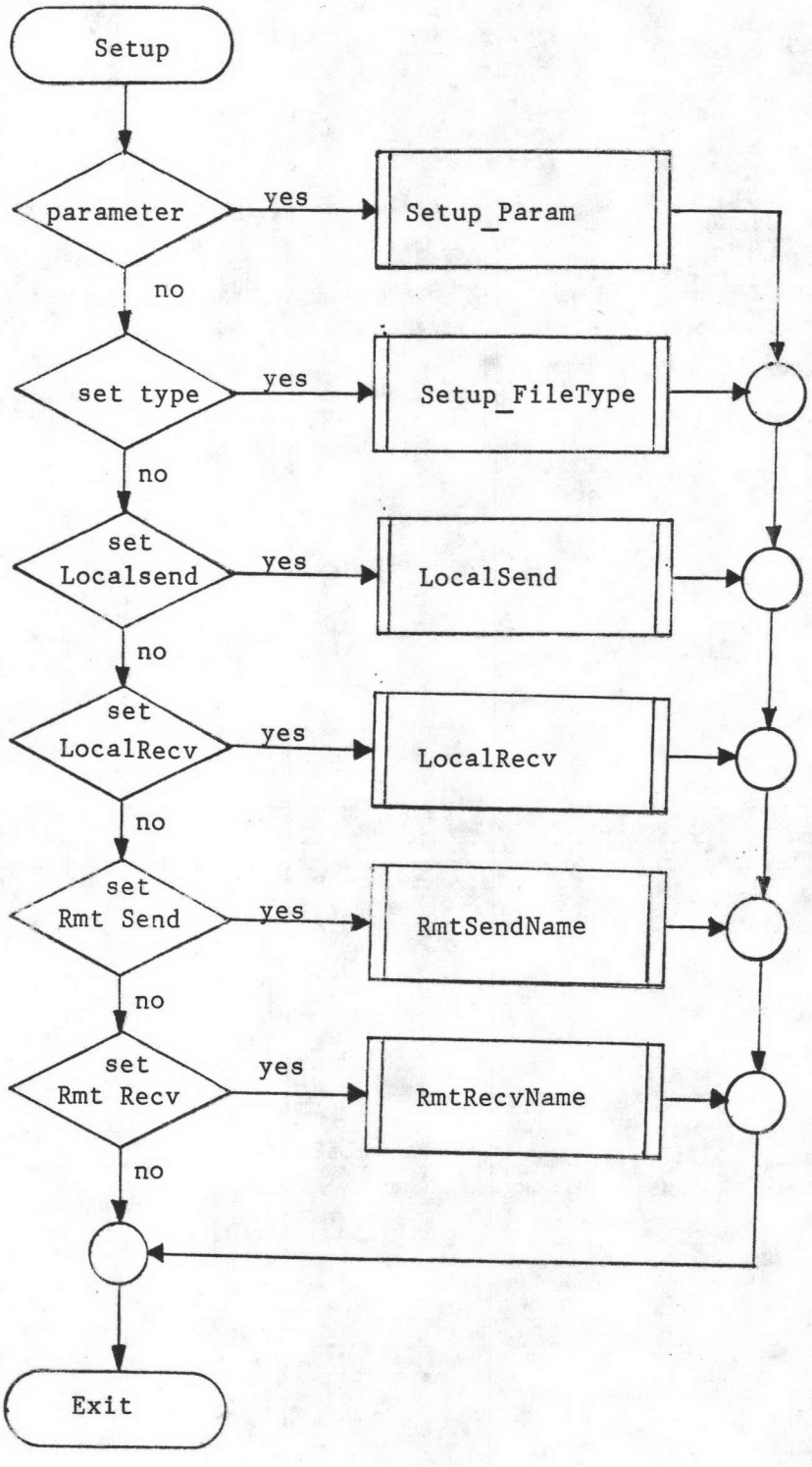
รูปที่ ข.16 ผังงานของฟังก์ชั่น DoCommand



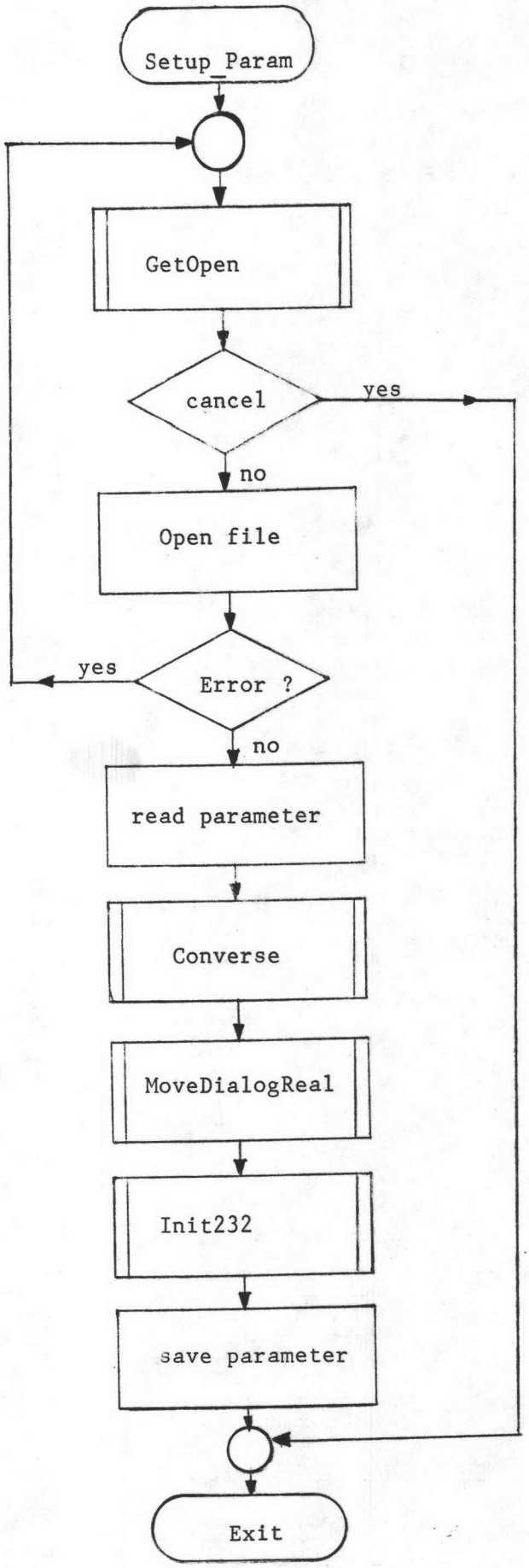
รูปที่ ข.17 ผังงานของฟังก์ชั่น DoOption



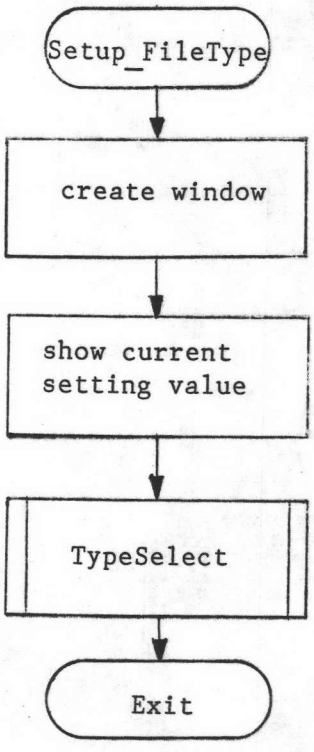
รูปที่ ข.18 ผังงานของฟังก์ชั่น CmdAct



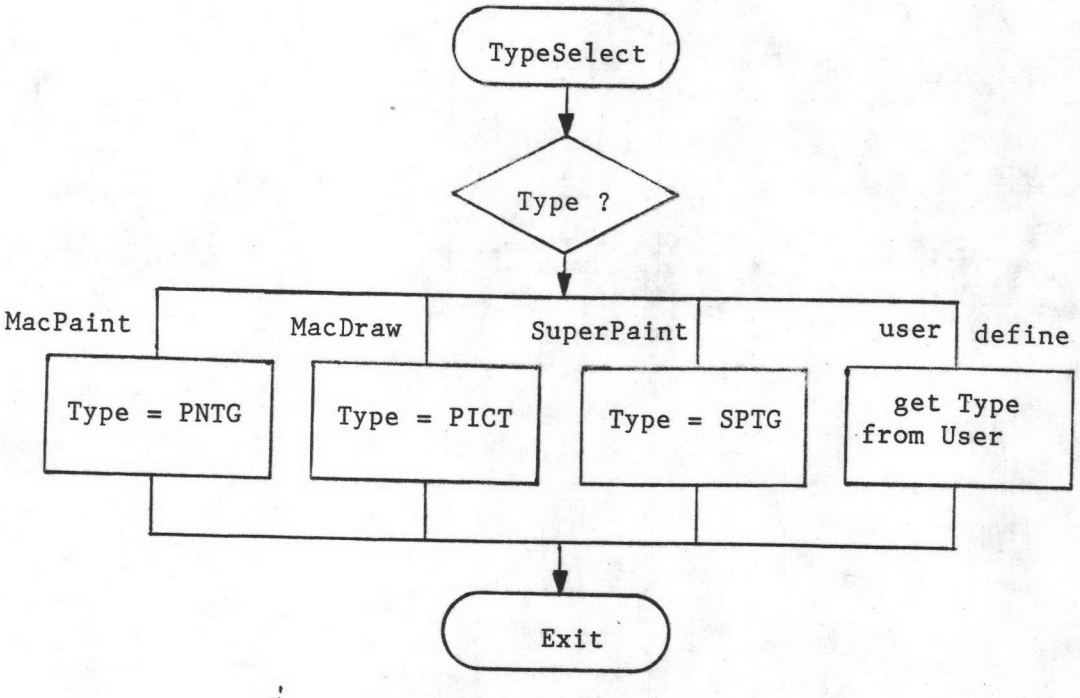
รูปที่ ข.19 ผังงานของฟังก์ชั่น Setup



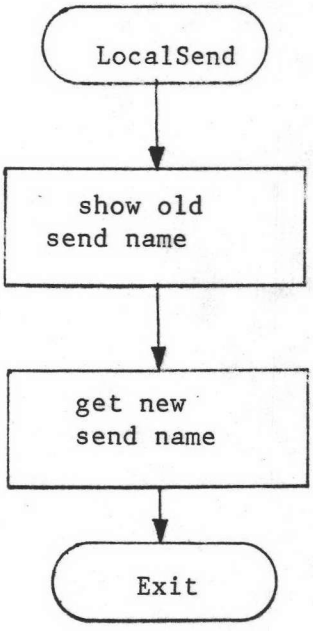
รูปที่ ข.20 ผังงานของฟังก์ชั่น Setup_Param



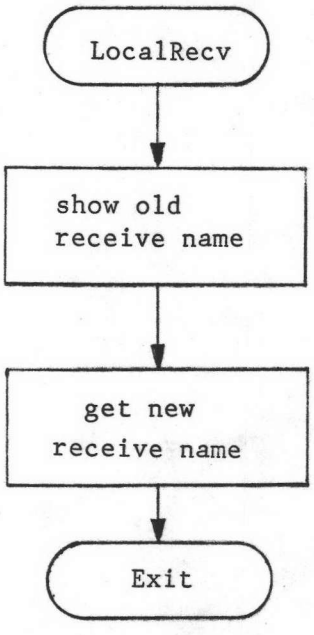
รูปที่ ข.21 ผังงานของฟังก์ชั่น Setup_FileType



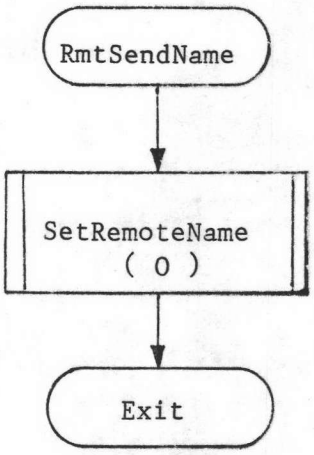
รูปที่ ข.22 ผังงานของฟังก์ชั่น TypeSelect



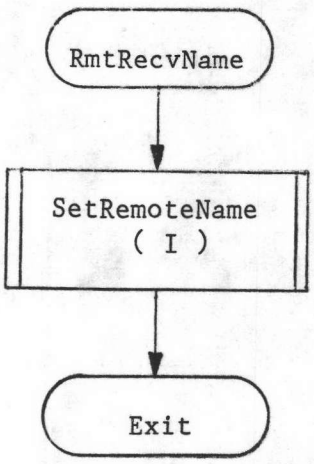
รูปที่ ข.23 ผังการทำงานของฟังก์ชั่น LocalSend



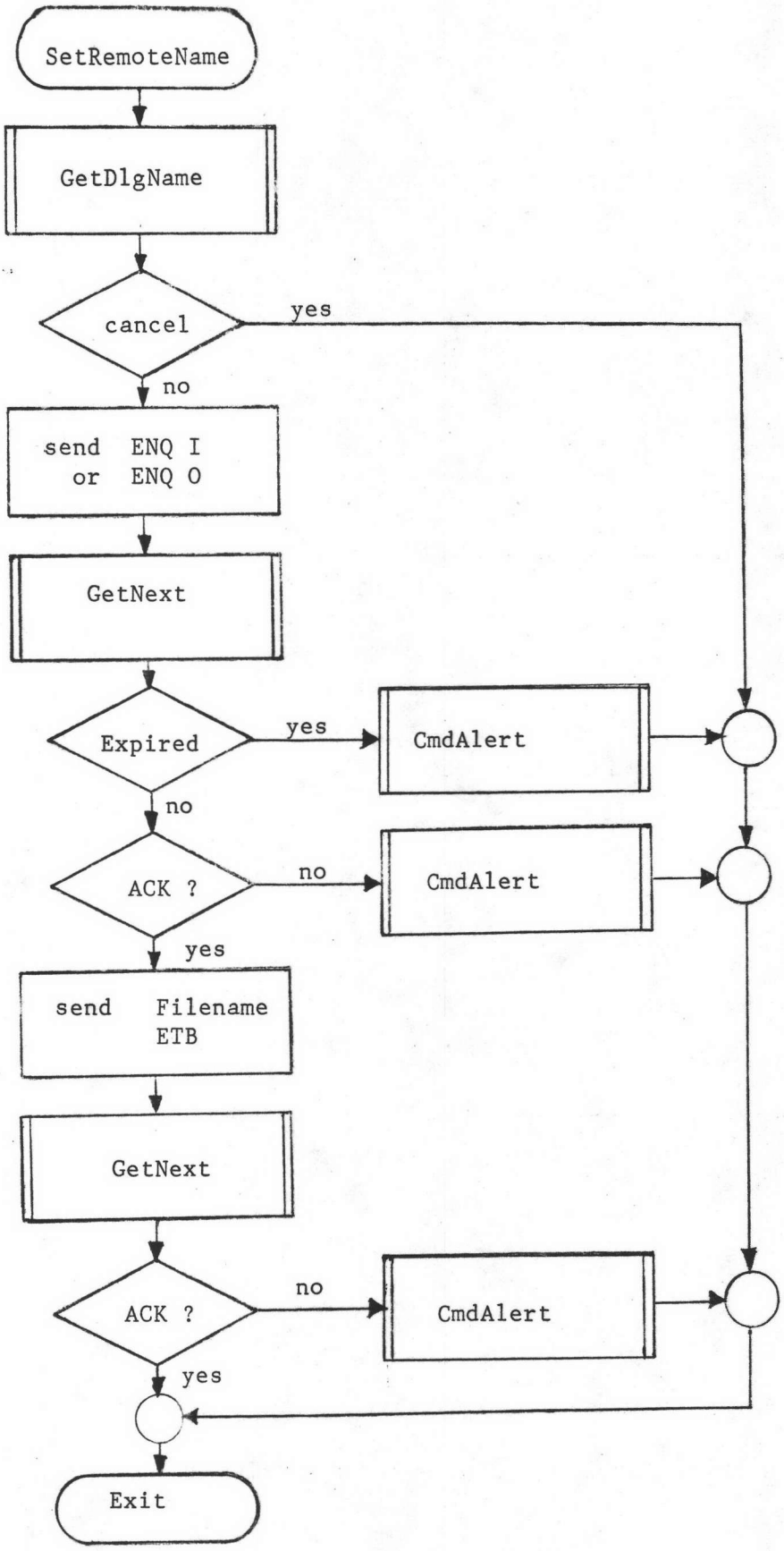
รูปที่ ข.24 ผังการทำงานของฟังก์ชั่น LocalRecv



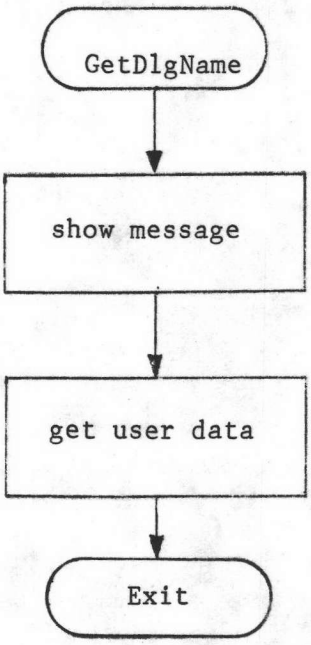
รูปที่ ข.25 ผังการทำงานของฟังก์ชั่น RmtSendName



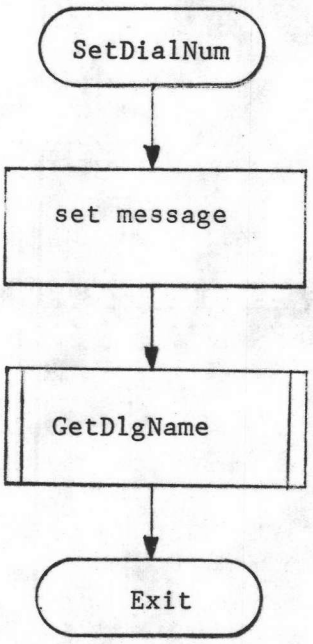
รูปที่ ข.26 ผังการทำงานของฟังก์ชั่น RmtRecvName



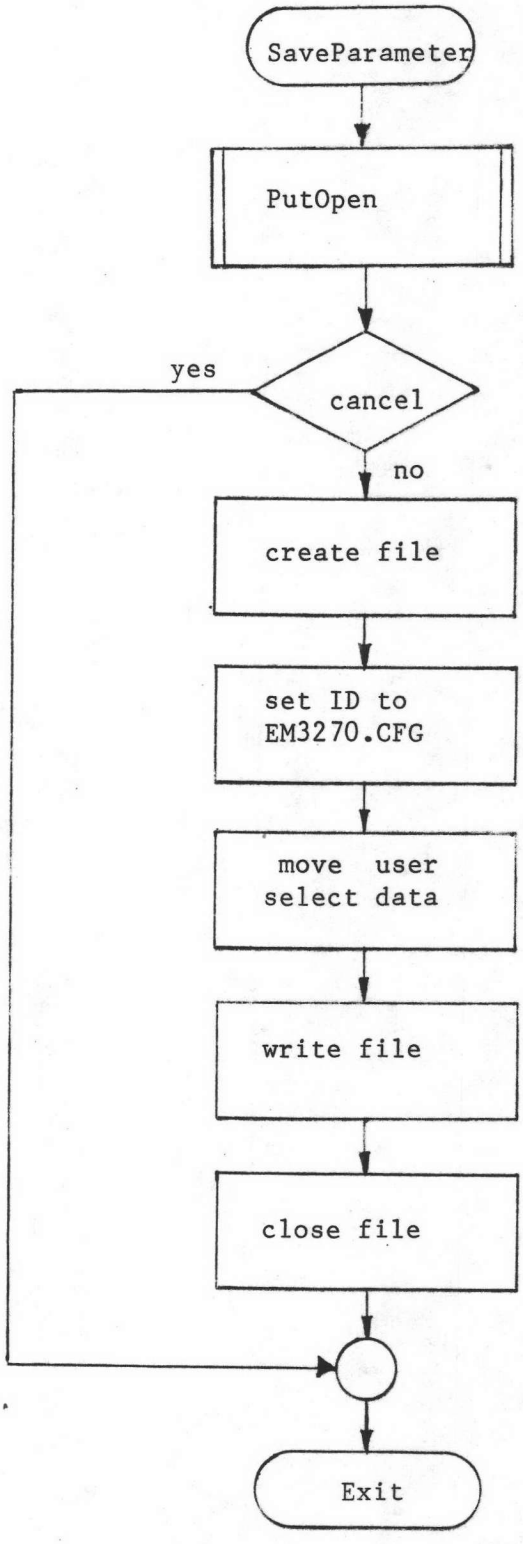
รูปที่ ข.27 ผังการทำงานของฟังก์ชั่น SetRemoteName



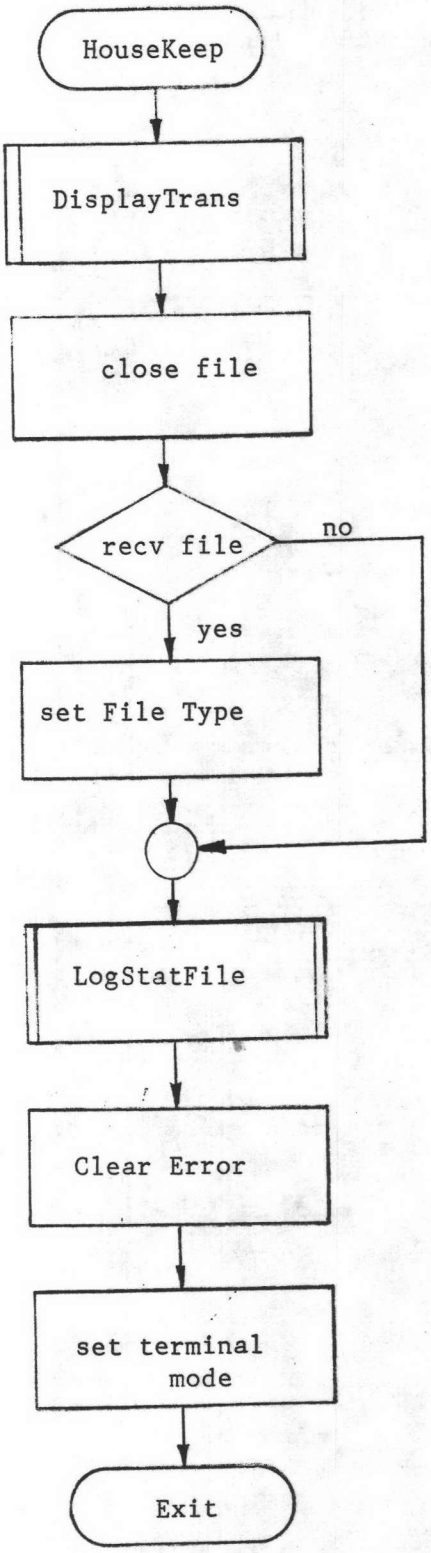
รูปที่ ข.28 ผังการทำงานของฟังก์ชั่น GetDlgName



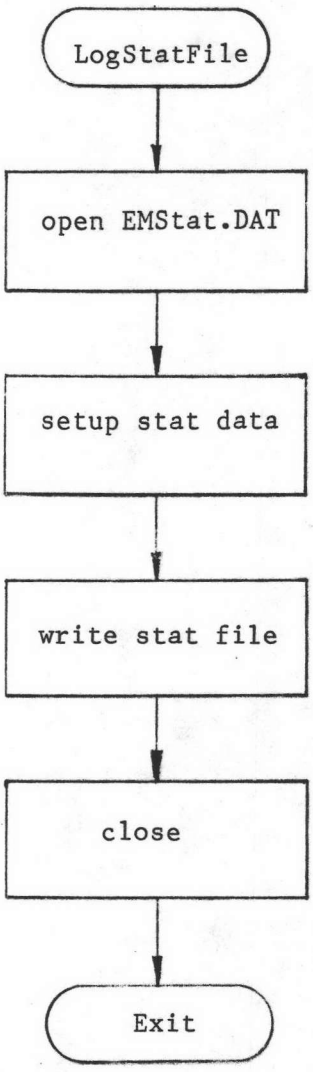
รูปที่ ข.29 ผังการทำงานของฟังก์ชั่น SetDialNum



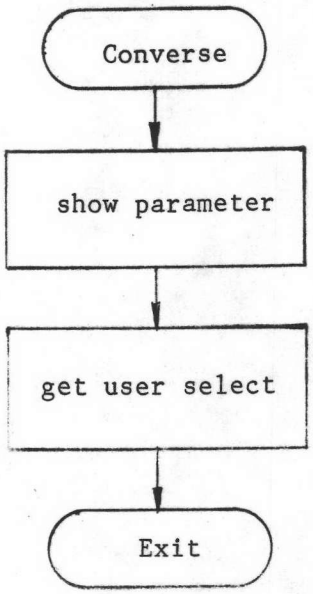
รูปที่ ข.30 ผังการทำงานของฟังก์ชั่น SaveParameter



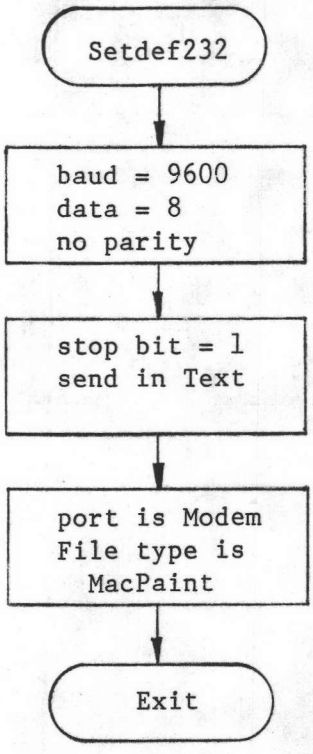
รูปที่ ข.31 ผังการทำงานของฟังก์ชั่น HouseKeep



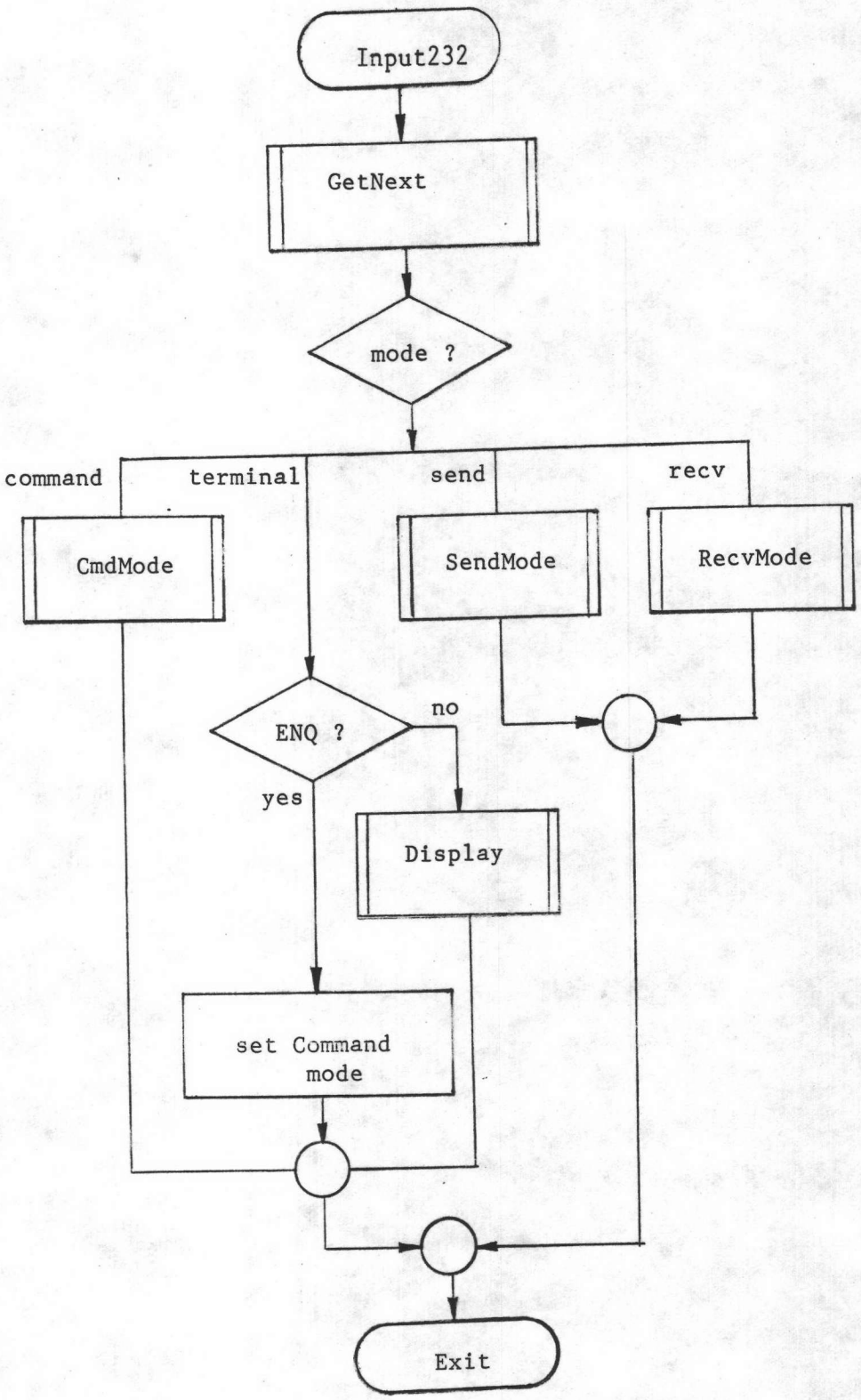
รูปที่ ข.32 ผังการทำงานของฟังก์ชั่น LogStatFile



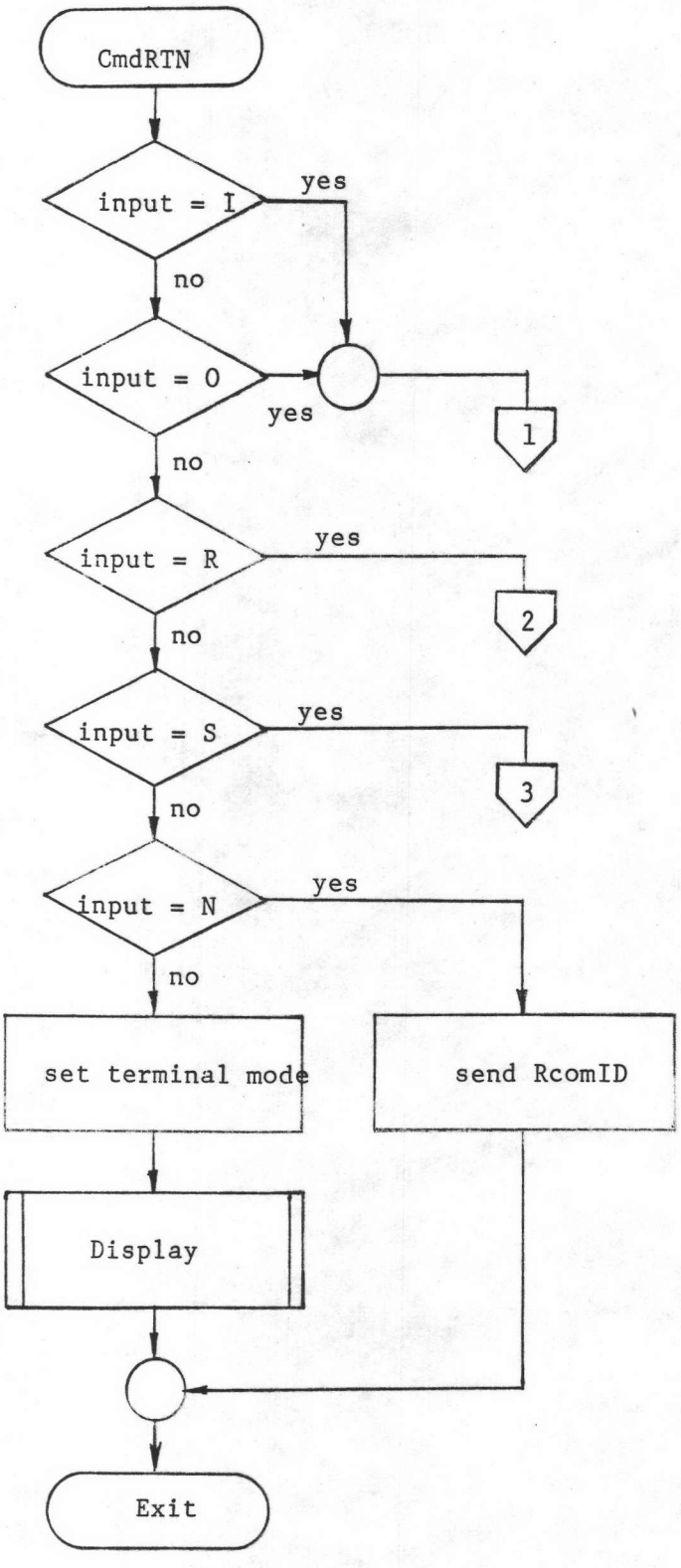
รูปที่ ข.33 ผังการทำงานของฟังก์ชั่น Converse



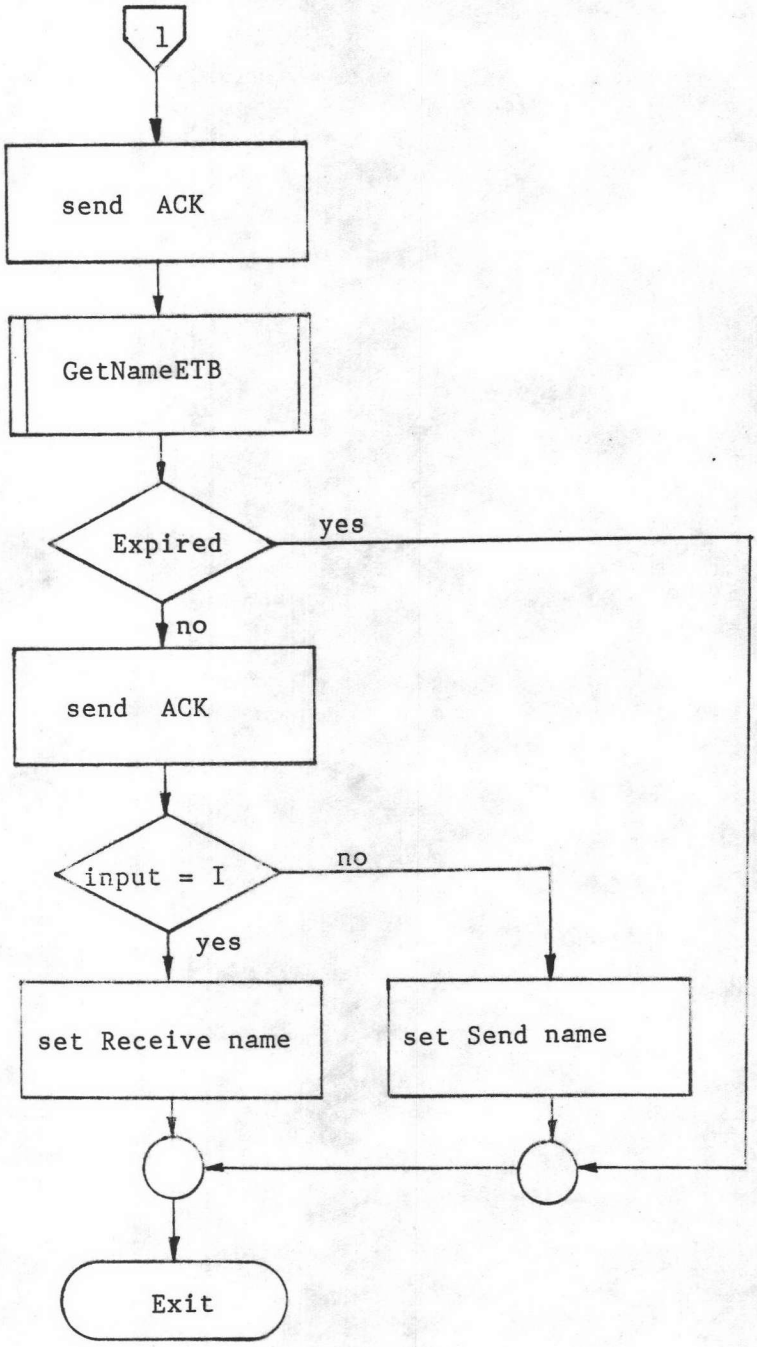
รูปที่ ข.34 ผังการทำงานของฟังก์ชั่น Setdef232



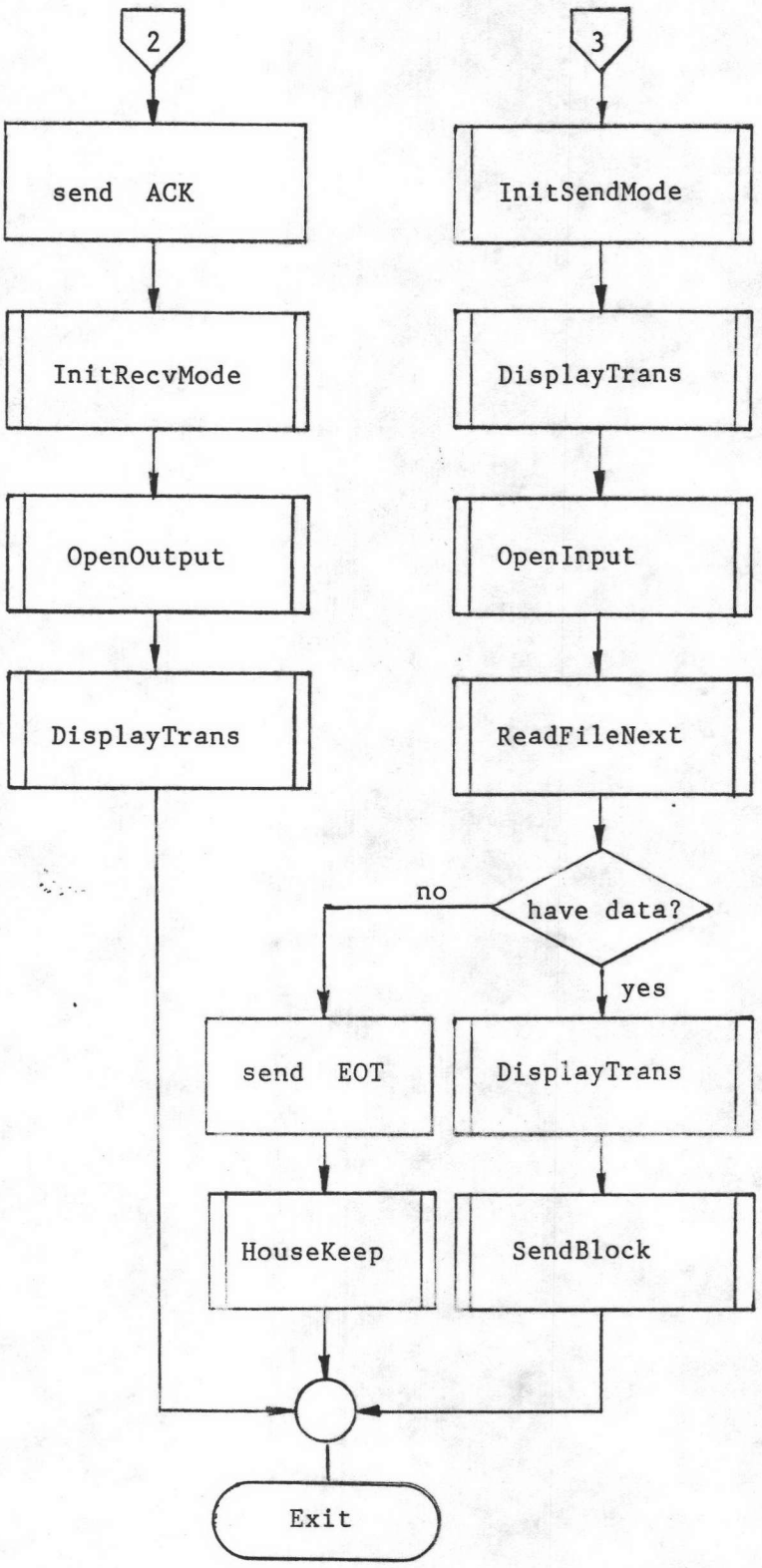
รูปที่ ข.35 ผังงานของฟังก์ชั่น Input232



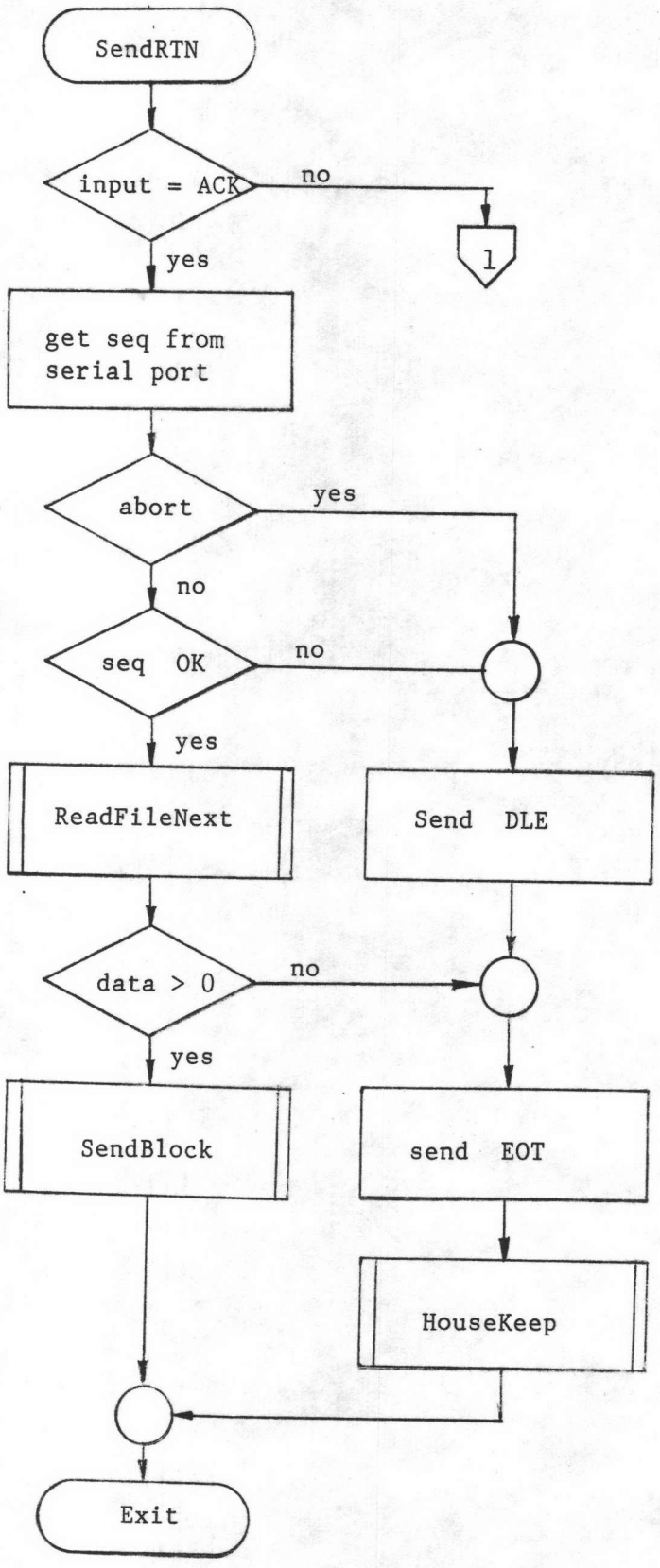
รูปที่ ข.36 ผังงานของฟังก์ชั่น CmdRTN



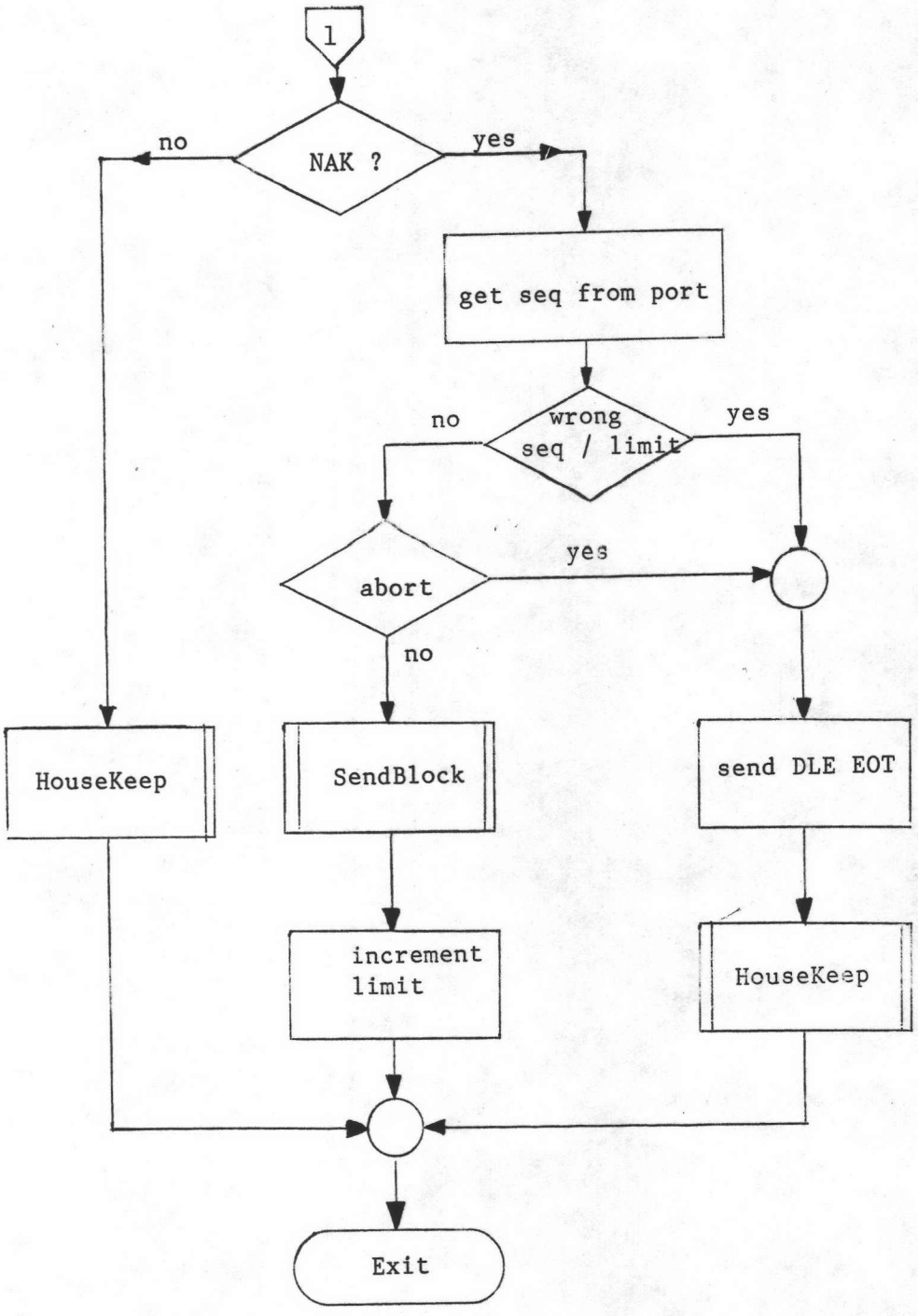
รูปที่ ข.36 ผังงานของฟังก์ชั่น CmdRTN (ต่อ)



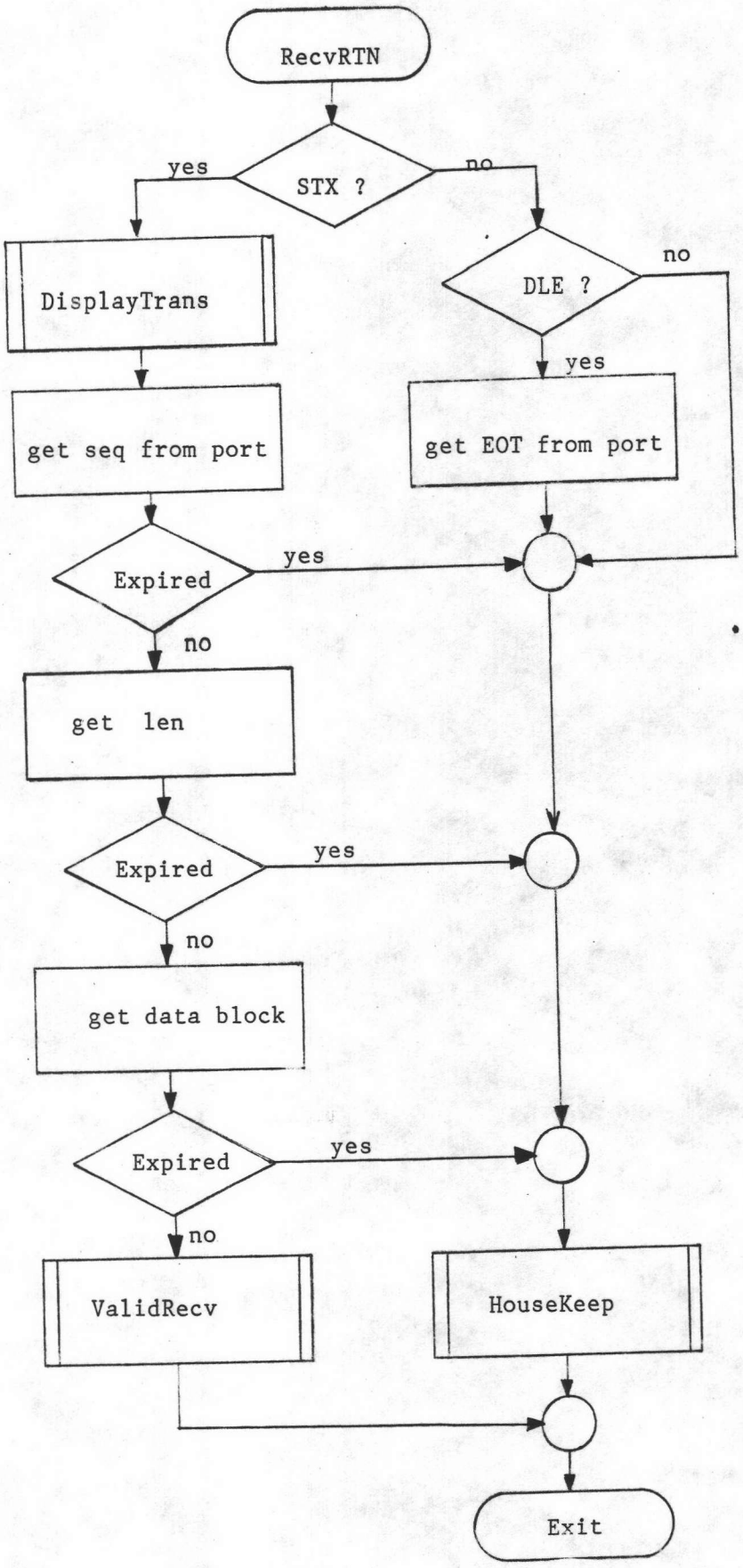
รูปที่ ข.36 ผังงานของฟังก์ชั่น CmdRTN (ต่อ)



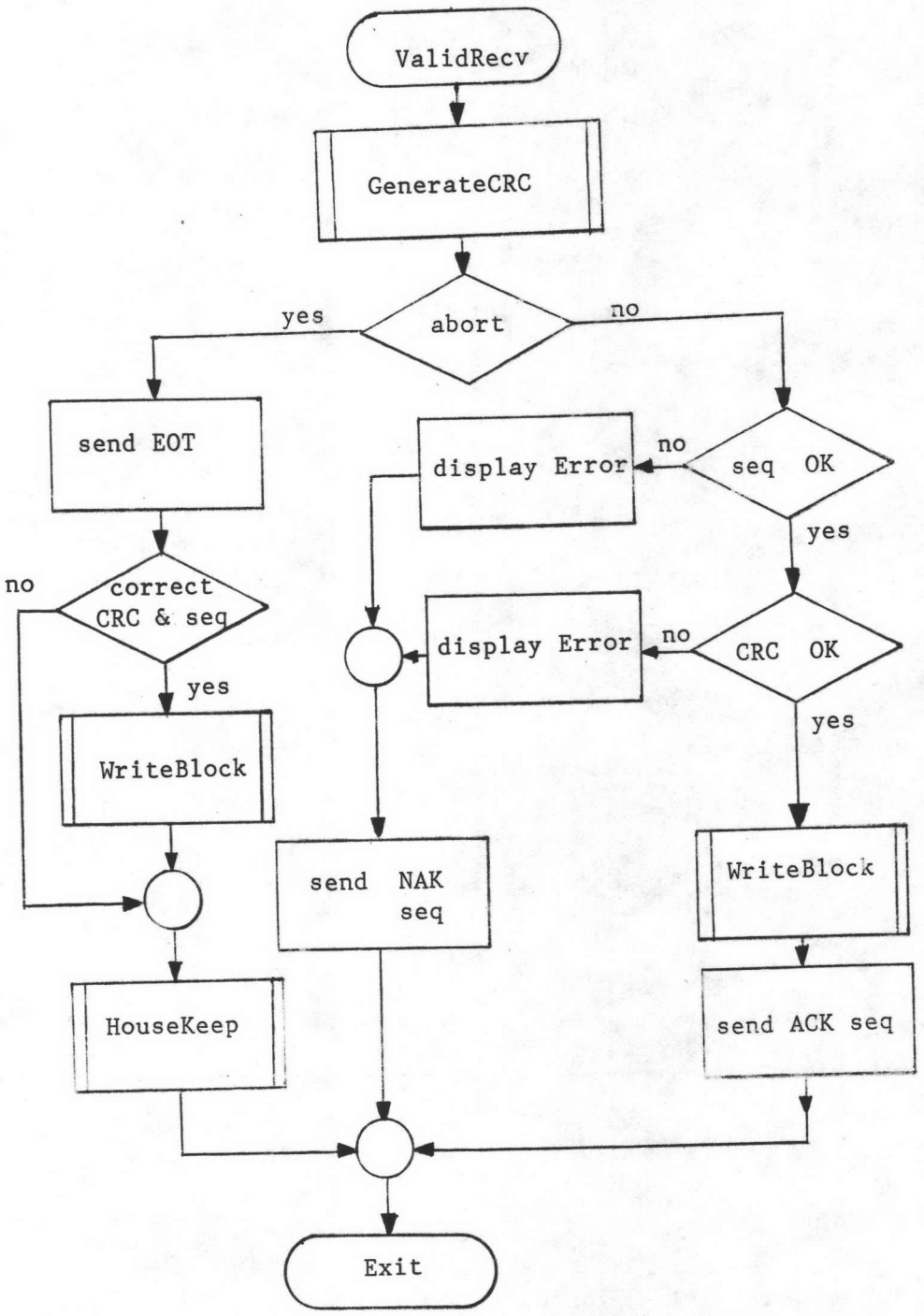
รูปที่ ข.37 ผังงานของฟังก์ชั่น SendRTN



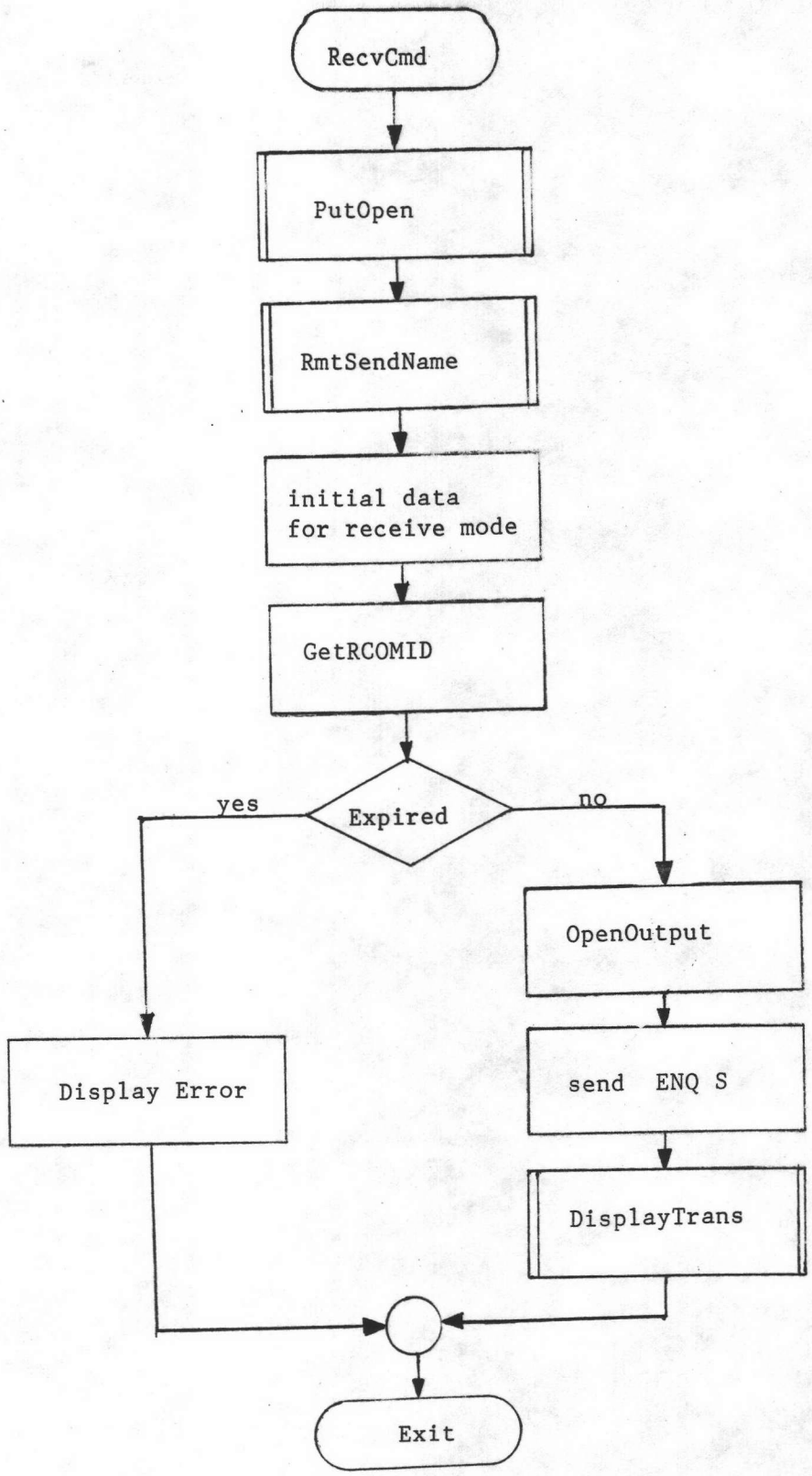
รูปที่ ข.37 ผังงานของฟังก์ชั่น SendRTN (ต่อ)



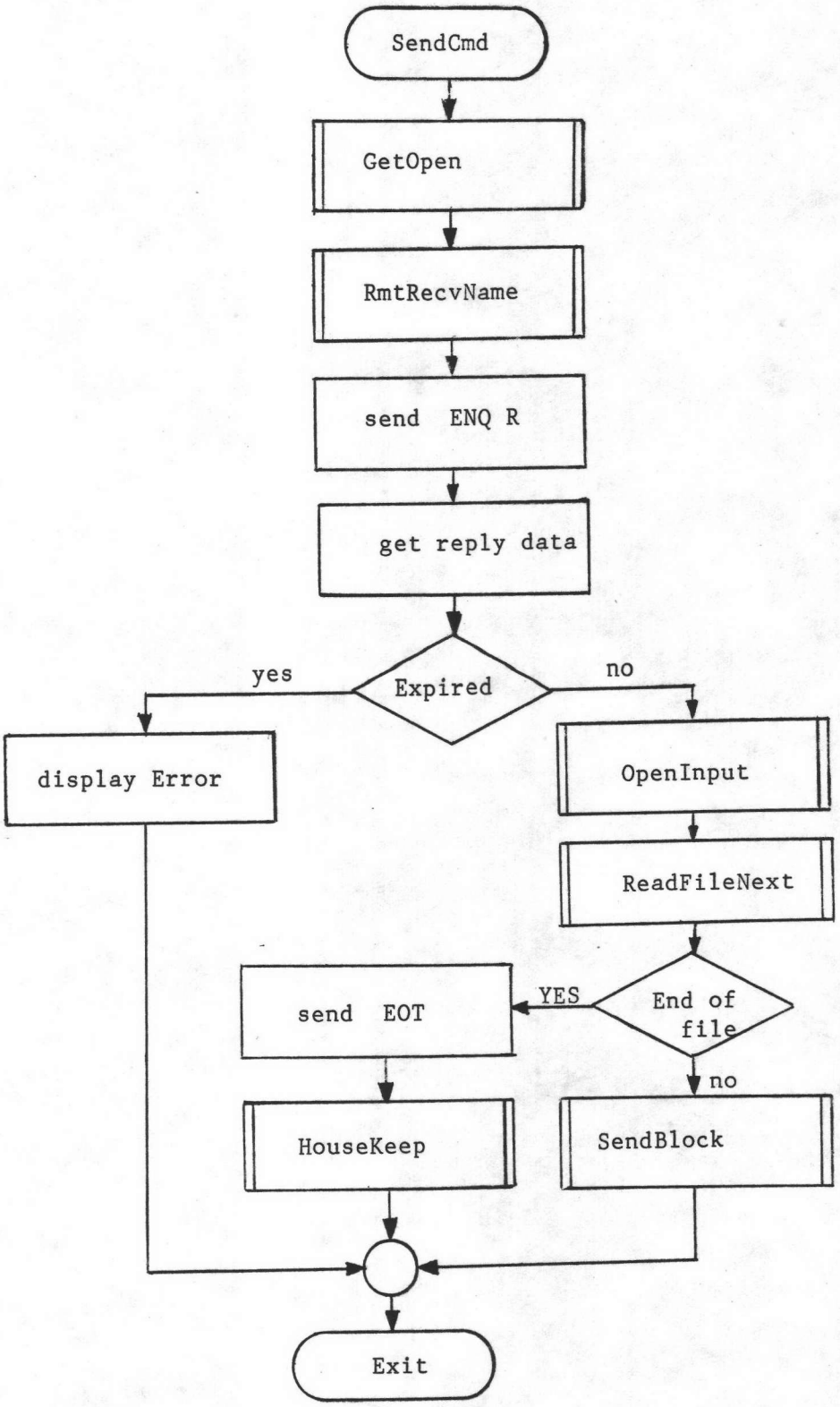
รูปที่ ข.38 ผังงานของฟังก์ชั่น RecvRTN



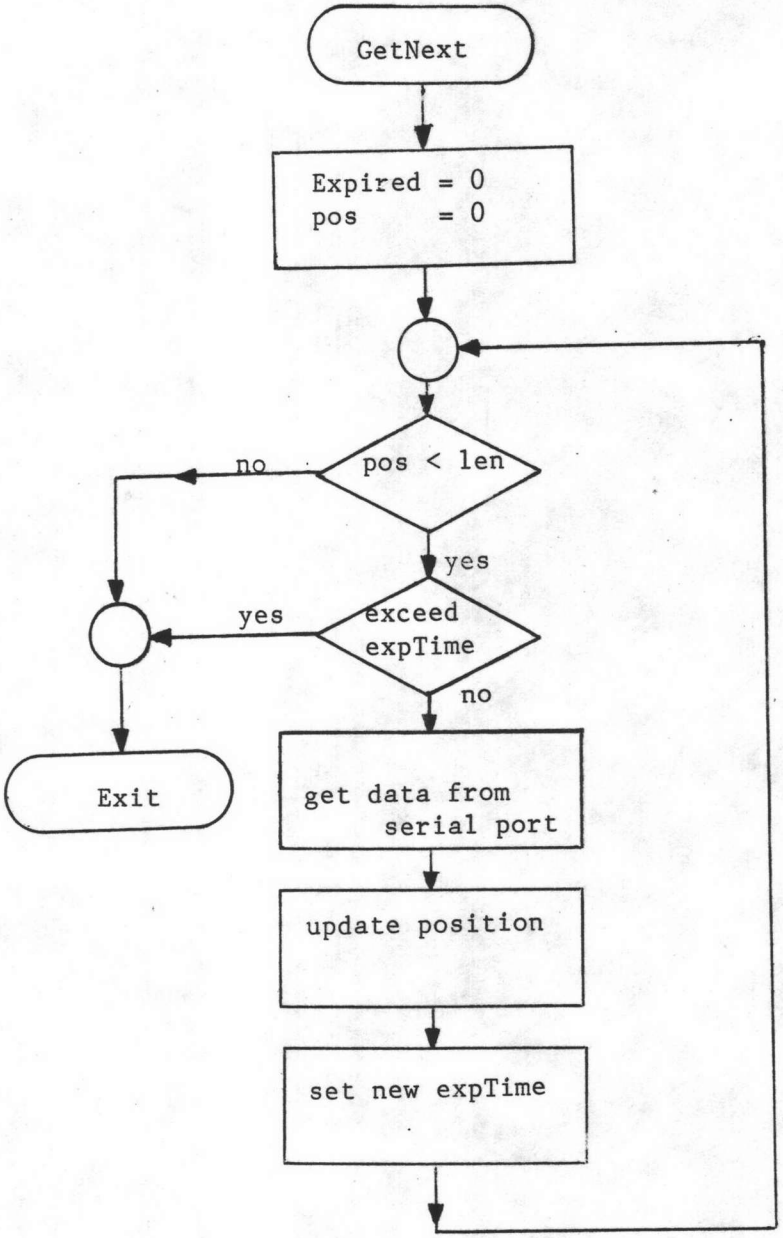
รูปที่ ข.39 ผังงานของฟังก์ชั่น ValidRecv



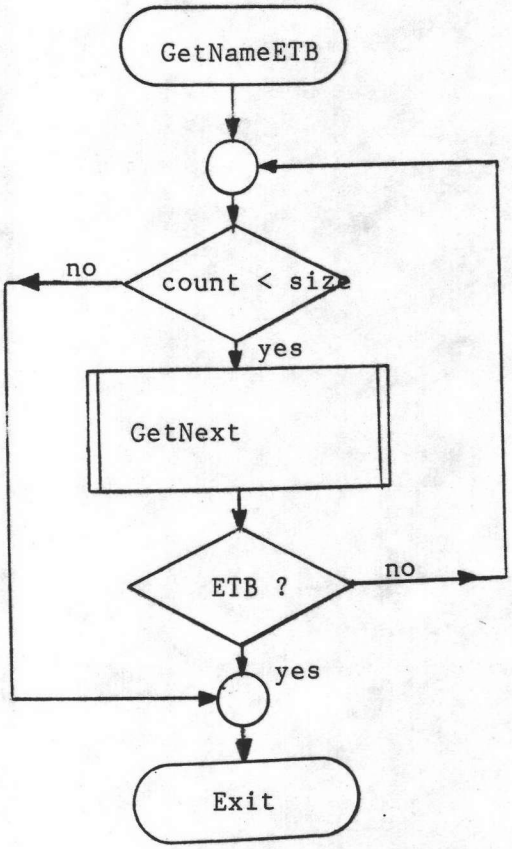
รูปที่ ข.40 ผังงานของฟังก์ชั่น RecvCmd



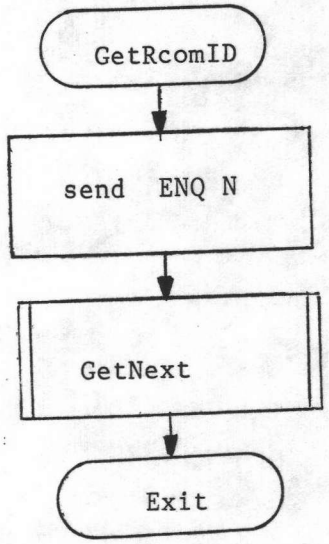
รูปที่ ข.41 ผังงานของฟังก์ชั่น SendCmd



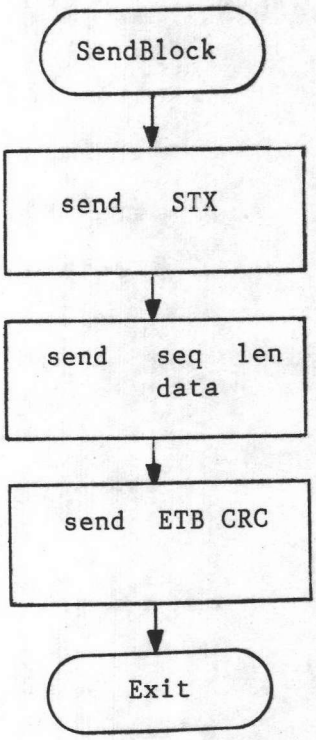
รูปที่ ข.42 ผังงานของฟังก์ชั่น GetNext



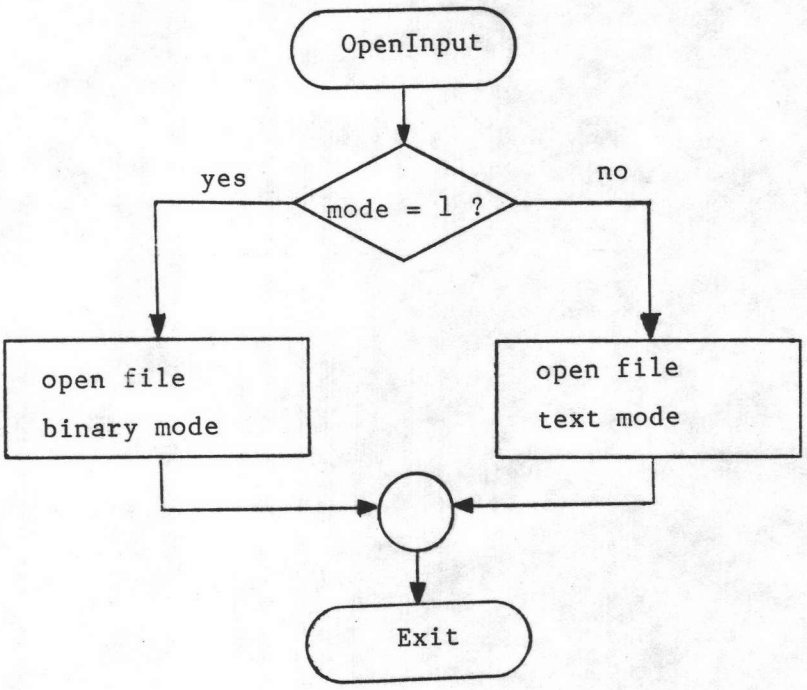
รูปที่ ข.43 ผังงานของฟังก์ชั่น GetNameETB



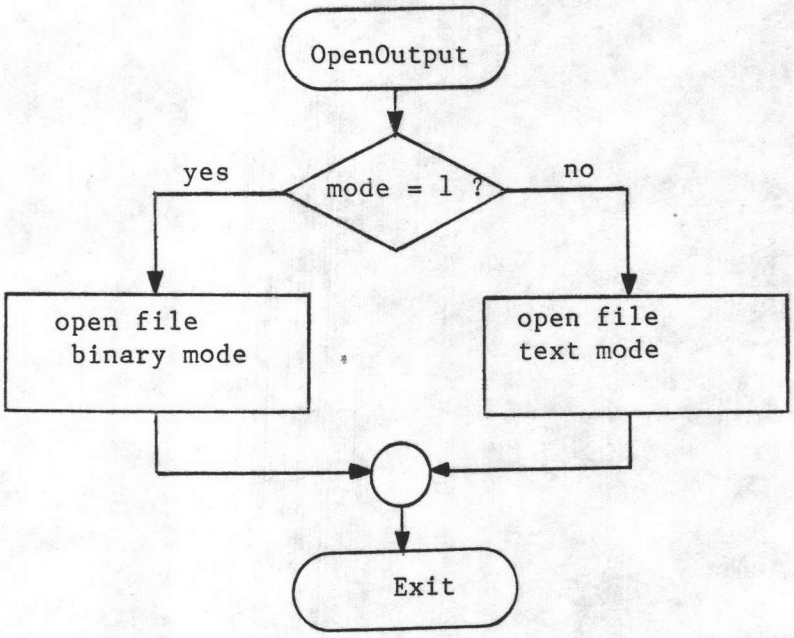
รูปที่ ข.44 ผังงานของฟังก์ชั่น GetRCOMID



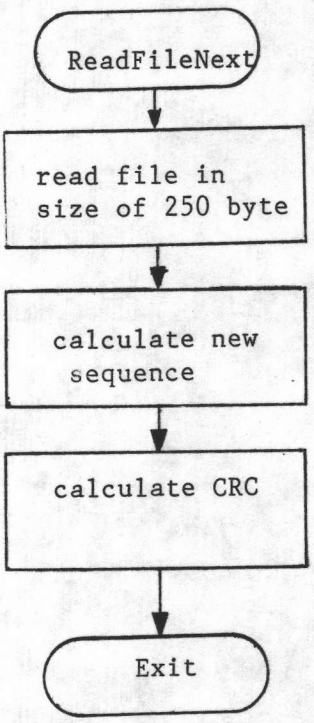
รูปที่ ข.45 ผังงานของฟังก์ชั่น SendBlock



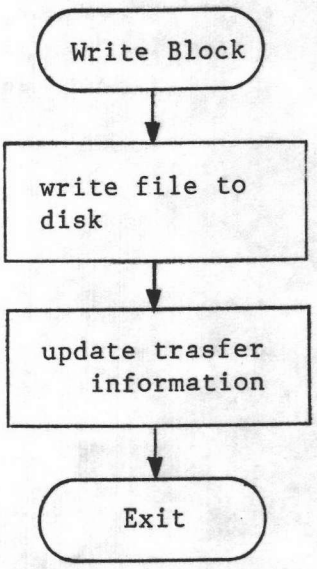
รูปที่ ข. 46 ผังงานของฟังก์ชั่น OpenInput



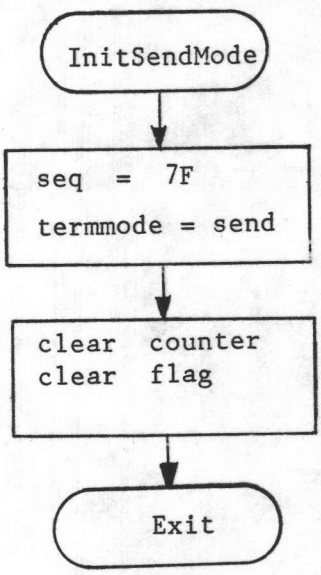
รูปที่ ข.47 ผังงานของฟังก์ชั่น OpenOutput



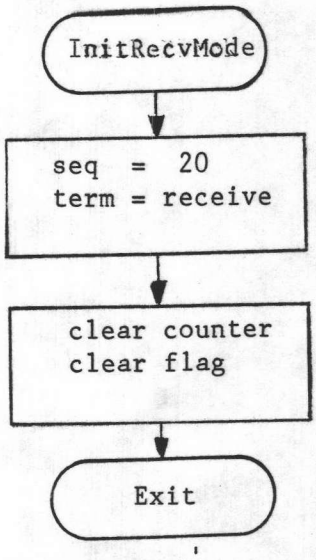
รูปที่ ข.48 ผังงานของฟังก์ชั่น ReadFileNext



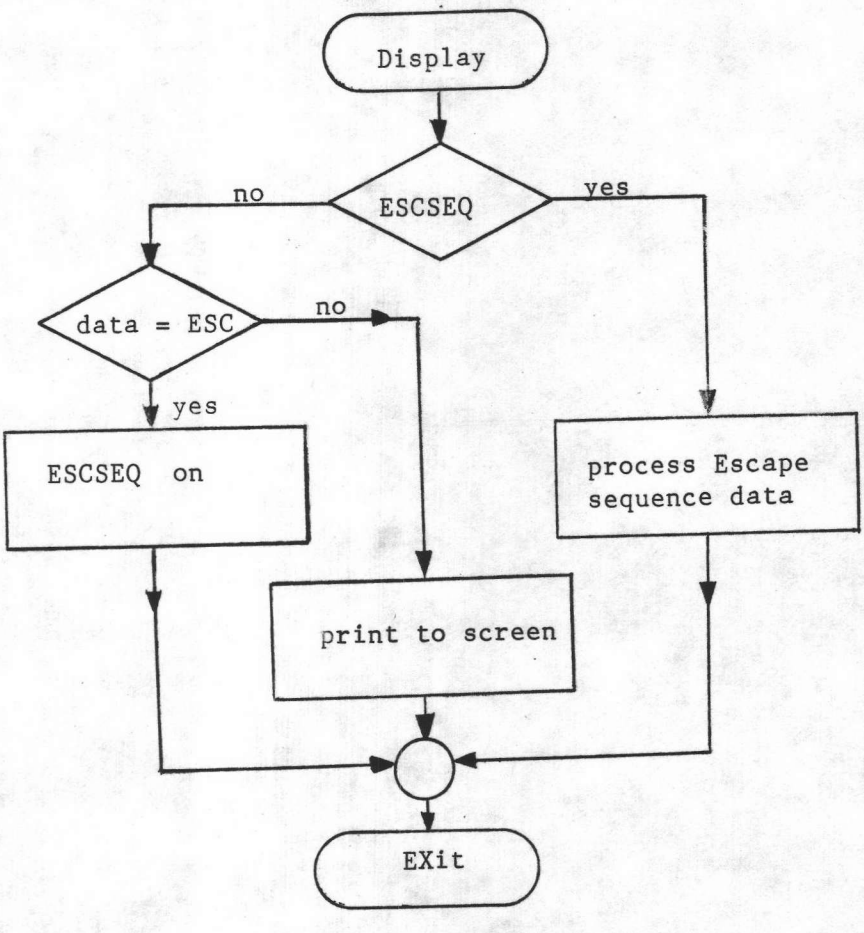
รูปที่ ข.49 ผังงานของฟังก์ชั่น WriteBlock



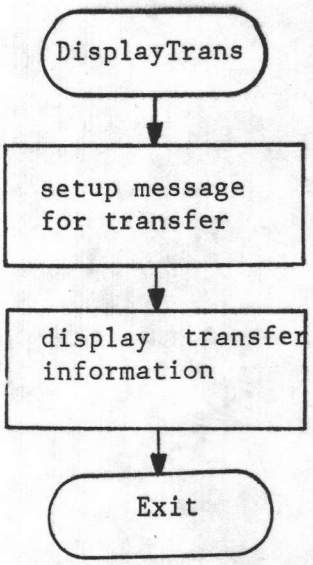
รูปที่ ข.50 ผังงานของฟังก์ชั่น InitSendMode



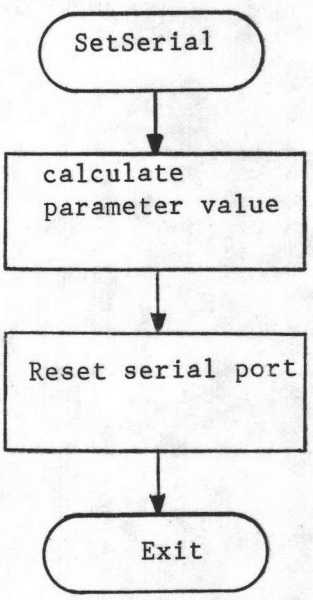
รูปที่ ข.51 ผังงานของฟังก์ชั่น InitRecvMode



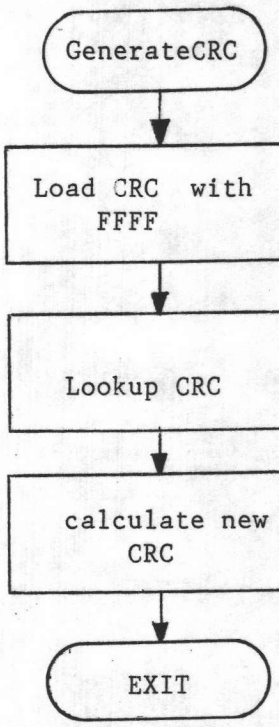
รูปที่ ข. 52 ผังงานของฟังก์ชั่น Display



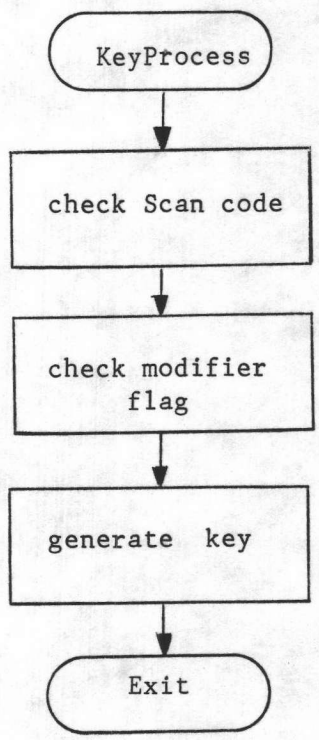
รูปที่ ข.53 ผังงานของฟังก์ชั่น DisplayTrans



รูปที่ ข.54 ผังงานของฟังก์ชั่น SetSerial



รูปที่ ข. 55 ผังงานของฟังก์ชั่น Generate CRC



รูปที่ ข. 56 ผังงานของฟังก์ชั่น KeyProcess

ภาคผนวก ค
โปรแกรมรับส่งข้อมูล

```
/*-------------------------------------------------
 *  File name : Globaldef.inc
 *  Function  : Define Global Variable for Main Routines
 *-------------------------------------------------*/

int     Abort;              /* Abort File Transfer */
int     Done;               /* Done  Terminate Program */

MenuHandle myMHdl[4];       /* Menu Handle for Menu BAR */

unsigned  char  SEQ;        /* SEQ   for  Send/ RECV  */
unsigned  int   RecvCRC;    /*  CRC received  */
unsigned  int   CRC;        /* Calculated CRC */

int     retry;              /* number of retry occurs */
long    ErrRec;             /* number of Error Records */
long    TransfByte;         /* number of transfer byte */

WindowPtr whichWindow;      /* Dummy Window Pointer */


char    RS232InputBuff[1024]; /* Input Driver buffer at least 257 byte*/
struct {
         unsigned char  Seq;
         unsigned char  Len;
         unsigned char  myBuff[256];
       } myMsg;

char myRCOMID[] = "334990002"; /* my Rcom Identifier number */
char hisRCOMID[10];

int  fileRef;               /* file no Opened */
int  ByteCount;             /* Byte Read From File */

int  termMode;              /* Terminal Operation Mode */
long ExpTime;               /* Expired Time */
int  Expired;               /* Expired Flag */
int  ErrCode;               /* Last Error Code of File Transfer */

int  baud, data;            /* Baud rate and Data Bits */
int  parity, stop;          /* Parity and Stop Bits */
int  mode, port;            /* Transfer Mode and Port */
int  bin;                   /* binary file type */
char FileType[5];           /* File Type for Creator */
char GetName[40];           /* Name of Remote File Received */
char RecvName[40];          /* Name of Actual Receive File */
char SendName[40];          /* Name of Actual Send File */
char OpenParm[40];          /* Name of Parameter file Open */
char DialName[40];          /* Dial Number */

char AIn[6]  = "\p.AIn";    /* Modem Port Input */
char AOut[6] = "\p.AOut";   /* Modem Port Output */
char BIn[6]  = "\p.BIn";    /* Print Port Input */
char BOut[6] = "\p.BOut";   /* Print Port Output */
int  refIN,refOUT;          /* Actual Reference Input/Output Port */
```

```
Point   SFGwhere = { 90, 82 };  /* Location of Get Package */
Point   SFPwhere = {106,104 };  /* Location of Put Package */
SFReply reply;                  /* Reply Message from File */

char TypeDlg[10];           /* Name of File Type */
char sendDlg[40];           /* Name of Send File from Dialog */
char recvDlg[40];           /* Name of Recv File from Dialog */
char dialDlg[40];           /* Name of Dial Phone number     */
int  baudDlg, dataDlg, parityDlg, stopDlg, modeDlg, portDlg, binDlg;

char  StatName[40] = "EMStat.Dat";   /* name of Status File */
char  ParmDef[] = "EM3270.CFG";   /* Name of Default parameter file EM3270.CFG */
char    myCFG[] = "EM3270.CFG";   /* Constant expect in IdCfg */
struct {
       char  IdCfg[11];
       int   fbaud, fdata, fparity, fstop, fmode, fport, fbin;
       char  fType[5];
       char  fsend[40];
       char  frecv[40];
       char  fdial[40];
       } parmrec;           /* Parameter Record */

int   EscSeq;               /* Escape sequence pointer */
char  EscArray[5];          /* Escape Array data  */

char STXstr[]   = "\pSTX";      /* start of text string */
char EOTstr[]   = "\pEOT";      /* End of Text string   */
char ENQstr[]   = "\pENQ";      /* Enquire string */
char ACKstr[]   = "\pACK";      /* Acknowledge string */
char BELstr[]   = "\pBEL";      /* BELL string   */
char DLstr[]    = "\pDLE";      /* DLE */
char DLEstr[]   = "\pDLE EOT"; /* DLE EOT string */
char NAKstr[]   = "\pNAK";      /* NAK string */
char SEQstr[]   = "\pSEQ NAK"; /* Sequence Error */
char CRCstr[]   = "\pCRC NAK"; /* Crc Error */
char WAITstr[]  = "\pWait";     /* Wait string */
char ABORTstr[] = "\pABORT";   /* Abort By operator */
char BLANKstr[] = "\p ";        /* Blank status */

char Sendstr[] = "Sending..  "; /* sending file message */
char Recvstr[] = "Receiving..  "; /* receiving file */

int  TransferDlg;    /* flag for check Transfer dialog init */
DialogPtr FileDlgPtr; /* transfer dialog pointer */

DialogPtr myDialogPtr; /* general dialog pointer */
int       dialogType;  /* general dialog type */
Handle    dialogHDL;  /* general dialog handle */
Rect      dialogRect; /* general dialog rectangle */

int       itemhit;
PenState  myPS;
Rect      dummyRect;
```

```
GrafPtr   savePort;
```

```
/*-----------------------------------------------
 * File name : ExternDef.inc
 * Function  : Define External Variable for Routines
 *             which use global variable
 *-----------------------------------------------*/

#include "DialogMgr.h"

extern int      Abort;              /* Abort File Transfer */
extern int      Done;               /* Done  Terminate Program */

extern MenuHandle myMHdl[4];        /* Menu Handle for Menu BAR */

extern unsigned  char  SEQ;         /* SEQ   for  Send/ RECV  */
extern unsigned  int   RecvCRC;     /*  CRC received  */
extern unsigned  int   CRC;         /* Calculated CRC */

extern int      retry;              /* number of retry occurs */
extern long     ErrRec;             /* number of Error Records */
extern long     TransfByte;         /* number of transfer byte */

extern char     RS232InputBuff[256];    /* Buffer for Input Driver */
extern struct {
          unsigned char  Seq;
          unsigned char  Len;
          unsigned char  myBuff[256];
        } myMsg;

extern char myRCOMID[];             /* my RCOM ID */
extern char hisRCOMID[] ;           /* his RcomID */
extern int  fileRef;                /* file no Opened */
extern int  ByteCount;              /* Byte Read From File */

extern int  termMode;               /* Terminal Operation Mode */
extern long ExpTime;                /* Expired Time */
extern int  Expired;                /* Expired Flag */
extern int  ErrCode;                /* Last Error Code of File Transfer */


extern int  baud, data;             /* Baud rate and Data Bits */
extern int  parity, stop;           /* Parity and Stop Bits */
extern int  mode, port;             /* Transfer Mode and Port */
extern int  bin;                    /* File type select */
extern char FileType[5];            /* File type */
extern char GetName[40];            /* Name of Remote File Received */
extern char RecvName[40];           /* Name of Actual Receive File */
extern char SendName[40];           /* Name of Actual Send File */
extern char StatName[40];           /* Status File Name */
extern char OpenParm[40];           /* Name of Parameter file Open */
extern char DialName[40];           /* Dial Number */

extern int  refIN,refOUT;           /* reference for Input/Output Port */


extern Point   SFGwhere ;  /* Location of Get Package */
```

```
extern Point   SFPwhere ;  /* Location of Put Package */
extern SFReply reply;                  /* Reply Message from File */

extern char TypeDlg[10];           /* File type from Dialog */
extern char sendDlg[40];           /* Name of Send File from Dialog */
extern char recvDlg[40];           /* Name of Recv File from Dialog */
extern char dialDlg[40];           /* Dial Number from Dialog */
extern int  baudDlg, dataDlg, parityDlg, stopDlg, modeDlg, portDlg, binDlg;

extern char  ParmDef[];   /* Name of Default parameter file EM3270.CFG */
extern char   myCFG[];    /* Constant expect in IdCfg */
extern struct {
       char  IdCfg[11];
       int   fbaud, fdata, fparity, fstop, fmode, fport, fbin;
       char  fType[5];
       char  fsend[40];
       char  frecv[40];
       char  fdial[40];
       } parmrec;          /* Parameter Record */

extern WindowPtr whichWindow;      /* Dummy Window Pointer */

extern int   EscSeq;        /* Escape Sequence Pointer */
extern char  EscArray[];    /* Escape Array data  */


extern char STXstr[];     /* start of text string */
extern char EOTstr[];     /* End of Text string   */
extern char ENQstr[];     /* Enquire string */
extern char ACKstr[];     /* Acknowledge string */
extern char BELstr[];     /* BELL string   */
extern char DLstr[];      /* DEL */
extern char DLEstr[];     /* DLE EOT string */
extern char NAKstr[];     /* NAK string */
extern char SEQstr[];     /* Sequence Error */
extern char CRCstr[];     /* Crc Error */
extern char WAITstr[];    /* Wait string */
extern char ABORTstr[];   /* Abort by Operator */
extern char BLANKstr[];   /* Blank status */

extern char Sendstr[];    /* sending file message */
extern char Recvstr[] ;   /* receiving file */

extern int  TransferDlg;     /* flag for check Transfer dialog init */

extern DialogPtr  FileDlgPtr;  /* dialog for file transfer */
extern DialogPtr myDialogPtr;
extern int       dialogType;
extern Handle    dialogHDL;
extern Rect      dialogRect;

extern int itemhit;
extern PenState  myPS;
extern Rect      dummyRect;
extern GrafPtr   savePort;
```

```
/*-------------------------------------------------
 * File name : defined.inc
 * Function  : define constant value for window,
 *             menu,  and communication data
 *-------------------------------------------------*/

#define  ErrorAlert  131    /* resource id for error alert */

#define  AppleMenu   0       /* Menu bar List Array */
#define  OptMenu     1       /* Menu Bar List Array */
#define  SetupMenu   2       /* Menu Bar List Array */
#define  CmdMenu     3       /* Menu Bar List Array */

#define  AppleID     127     /* Menu ID  of Apple   */
#define  OptID       128     /* Menu ID  of Option  */
#define  SetupID     129     /* Menu ID  of Setup   */
#define  CmdID       130     /* menu ID  of Command */

#define  quit        1       /* Quit  Item      */

#define  param       1       /* Parameter item  */
#define  setFileType 2       /* Setup File Type */
#define  setLocSend  4       /* Local send name */
#define  setLocRecv  5       /* Local recv name */
#define  setRmtSend  7       /* Remote send name */
#define  setRmtRecv  8       /* Remote Recv name */
#define  dialNum     10       /* Dial Number */

#define  sendfile    1       /* start send file */
#define  recvfile    2       /* start recv file */
#define  dialing     4       /* dialing phone */

#define  normal       1      /* Terminal Mode   */
#define  CmdMode      2      /* Command Mode    */
#define  SendMode     3      /* Send File Mode  */
#define  RecvMode     4      /* Recv File MOde  */


#define  DelayTime   40      /* delay time is 40 second */
#define  RetryLimit  31      /* retry on send 31 times  */

#define  SOH      0x01       /* Start Of Header  */
#define  STX      0x02       /* Start Of Text    */
#define  EOT      0x04       /* End   of Text    */
#define  ENQ      0x05       /* Enquiry          */
#define  ACK      0x06       /* Acknowledge      */
#define  BEL      0x07       /* Bell             */
#define  DLE      0x10       /* Data Link Escape */
#define  NAK      0x15       /* Negative Acknowledge */
#define  ETB      0x17       /* End Of Text Block */


#define  initSeq  0x20       /* START Sequence No */
#define  MaxSeq   0x7F       /* Max Sequence No   */
#define  maxsize  0xFA       /* Max Size of Block */
```

```
/*-----------------------------------------------------
 * File name : MainPgm.c
 * Function  : startup and initialize serial port,
 *             and calling MainEvent handle event
 *-----------------------------------------------------*/

#include  "stdio.h"
#include  "DialogMgr.h"
#include  "FileMgr.h"
#include  "StdFilePkg.h"
#include  "EventMgr.h"
#include  "MenuMgr.h"

#include  "unix.h"
#include  "fopenw.h"

#include  "defined.inc"
#include  "Globaldef.inc"


StdWindowOptions myopt;
FILE    *myWindow;
Point   Corner;

main()
{

   InitGraf(&thePort);                  /* initialize QucikDraw Port */
   InitFonts();                                 /* Initialize Font Manager */
   InitWindows();                       /* Initialize Window Manager */
   TEInit();                                    /* Initialize TextEdit  */
   InitDialogs(0L);                     /* Initialize Dialog Manager */
   InitMenus();                                 /* Initialize Menu   Manager */
   PenNormal();                                 /* Set Pen Symbol */
   InitCursor();                        /* Set Cursor  */
   Stdio_MacInit(true);                 /* tell Standard IO no need to Init */
   Click_On(0);                                 /* no Click Window when exit */
   OpenResFile("\pHydra.Rsrc"); /* Open resource file */
   setupWindow();                       /* setup my Window */
   gotoxy(0,20);                        /* goto  row 21 col 1 */
   EscSeq = 0;                                  /* no Escape Sequence */
   FlushEvents(everyEvent,0);           /* Flush all Event */
   drawMymenu();                        /* Draw Menu Bar   */
   refIN = 0;                                   /* no serial input port */
   refOUT = 0;                                  /* no serial output port */
   if (AutoConfig()) {                  /* Auto Config Ok ? */
      HiliteMenu(0);                /* then not highlight menu bar */
      GenerateTable();                  /* Generate CRC Table */
      FlushEvents(everyEvent,0);/* Flush all Pending Event */
      termMode = normal;                /* start with terminal Emulator mode */
      TransferDlg = 0;                  /* no transfer Dialog */
      Abort = 0;                        /* Set Not Abort */
      MainEvent();                      /* do  Main Event Loop procedure */
      SerSetBuf(refIN,0L,0);            /* reset to local driver buffer */
    }
```

```
    fclose(myWindow);              /* close my window when exit */
}

int  setupWindow()              /* Create Screen */
{

   myopt.maxrow = 25;               /* maximum row is 25 */
   myopt.maxcol = 80;               /* maximum col is 80 */
   myopt.cursor_visible = true; /* cursor is visible */
   myopt.echo_state = true;     /* echo   is true    */
   myopt._tab_width = 8;            /* tab width is 8    */
   myopt.no_pull_front = false; /* pull window to front */
   myopt.no_grow = false;           /* grow box */
   myopt.no_drag = true;            /* no drag permited */
   myopt.no_goaway = true;          /* no go away box */
   myopt.no_scroll = true;          /* no scroll */
   myopt.no_wrap = false;           /* with wrap */

   Corner.v = 40;  Corner.h = 6;/* location is 40,60 */
   myWindow = fopenw("3270 EMULATOR", Corner, &myopt);
   setwindow(myWindow);         /* set my window active window */
}

int  CfgOpenErr(f)
Str255 f;
/* This Function has been designed to show Error
   when we cannot open parameter file .          */
{
   CtoPstr(f);
   ParamText("\pCannot Open Parameter File",f,"\p","\p");
   Alert( ErrorAlert, 0L);
   PtoCstr(f);
}

int  CfgFmtErr(f)
Str255 f;
/* This Function has been designed to show Error
   when there is invalid data format in parameter file. */
{
   CtoPstr(f);
   ParamText("\pInvalid Data Format in Parameter file",f,"\p","\p");
   Alert( ErrorAlert, 0L);
   PtoCstr(f);
}

int  CfgWritErr(f)
Str255 f;
/* This Function has been designed to show Error
   when there is error while writing to parameter file  */
{
   CtoPstr(f);
   ParamText("\pError Writing in Parameter file",f,"\p","\p");
   Alert( ErrorAlert, 0L);
   PtoCstr(f);
}
```

```
int SendError(s)
Str255 s;
{
   ParamText(s,"\p","\p","\p");
   Alert( ErrorAlert, 0L);
}

int  AutoConfig()
/* Auto Configuration for setup serial port and parameter */
{
   SFTypeList  myTypes;
   int ConfNo;
   unsigned int count;
   unsigned int avail;

   HiliteMenu(SetupID);       /* Menu ID  of Setup   */
   strcpy(OpenParm,ParmDef); /* get default Name to Open Config file */
   if ( (ConfNo = open(OpenParm,O_BINARY)) == EOF) {

   GetAgain:
      if ( GetOpen(OpenParm) ) {
         if ( (ConfNo = open(OpenParm,O_BINARY)) == EOF) {
            CfgOpenErr(reply.fName);  /* Alert File Not Found */
            goto GetAgain;
         }
         else {
            count = sizeof(parmrec);
            avail = read(ConfNo, &parmrec, count );
            if ( avail == EOF  ) {
              CfgOpenErr(reply.fName);  /* Alert File Read Error */
              close(ConfNo);
              return(0);          /* termiante job */
            }
            if ( (avail != count) || (strcmp(myCFG,&parmrec.IdCfg) != 0)) {
               CfgFmtErr(reply.fName);  /* invalid Config file format */
               close(ConfNo);
               return(0);
            }
            moveFileToReal();
            close(ConfNo);
         }
      }
      else {
         if (! Converse(2) )  /* converse by default parm */
            return(0);
         MoveDialogReal();
         MoveRealFile();
         SaveParameter();     /* save to file */
      }
   }
   else {
      count = sizeof(parmrec);
      avail = read(ConfNo, &parmrec, count );
      if ( avail == EOF  ) {
```

```
            CfgOpenErr(reply.fName);  /* Alert File Read Error */
            close(ConfNo);
            return(0);          /* termiante job */
         }
         if ( (avail != count) || (strcmp(myCFG,&parmrec.IdCfg) != 0)) {
            CfgFmtErr(reply.fName);  /* invalid Config file format */
            close(ConfNo);
            return(0);
         }
         moveFileToReal();
         close(ConfNo);
      }
      if (Init232())
        return(1);
      else
        return(0);

}


int  moveFileToReal()
/*   move parameter from file to real parameter */
{
     baud = parmrec.fbaud;              /* get baud rate */
     data = parmrec.fdata;              /* get data bit */
     parity = parmrec.fparity;          /* get parity bit */
     stop = parmrec.fstop;              /* get stop bit */
     mode = parmrec.fmode;              /* get Send mode */
     port = parmrec.fport;              /* get Send Port */
     bin  = parmrec.fbin;               /* binary file select */
     strcpy(FileType,parmrec.fType);    /* get Type of file */
     strcpy(SendName,parmrec.fsend);    /* get Send Name */
     strcpy(RecvName,parmrec.frecv);    /* get Receive Name */
     strcpy(DialName,parmrec.fdial);    /* get Dial Character */
}

/*  Initialize Serial Port depending on value of
     parameter in real memory
     return 1 if Success  else zero        */
int  Init232()
{

  if (refIN != 0 )
     SerSetBuf(refIN,RS232InputBuff,0);
  if (port == 1 ) {
     if (OpenSerial(AIn,&refIN) != noErr)
        return(0);

     if (OpenSerial(AOut,&refOUT) != noErr)
        return(0);
  }
  else {
     if (OpenSerial(BIn,&refIN) != noErr)
        return(0);
```

```
      if (OpenSerial(BOut,&refOUT) != noErr)
         return(0);
   }

   /*  set buffer size to input buffer size */
   if (SerSetBuf(refIN,RS232InputBuff,sizeof(RS232InputBuff)) != noErr)
      return(0);

   if (KillIO(refIN) != noErr)
       return(0);

   if (KillIO(refOUT) != noErr)
       return(0);

   if (SetSerial(refIN,stop,parity,data,baud) != noErr)
       return(0);
   if (SetSerial(refOUT,stop,parity,data,baud) != noErr)
       return(0);
   else
       return(1);

}


int  drawMymenu()     /* Draw menu bar */
{
    int i;

    myMHdl[AppleMenu] = GetMenu(AppleID); /* read Apple menu */
    AddResMenu(myMHdl[AppleMenu],'DRVR'); /* add desk accessory name */
    myMHdl[OptMenu] = GetMenu(OptID);     /* read option menmu */
    myMHdl[SetupMenu] = GetMenu(SetupID); /* read setup menu */
    myMHdl[CmdMenu] = GetMenu(CmdID);     /* read command menu */
    for ( i= AppleMenu; ( i <= CmdMenu ) ; i++ )
        InsertMenu(myMHdl[i],0);
    DrawMenuBar();
}


int PstrCopy(d,s)     /*  Copy String of pascal type */
register char *d, *s;
{
register int  len;
   len = *d++ = *s++;
   for (;len > 0 ; len--)  *d++ = *s++;
}

GetOpen(ReplyName)    /*  Dialog for Getfile */
char     ReplyName[];
{
   SFReply    reply;
   SFTypeList myType;

   myType[0] = 'TEXT';
   SFGetFile(SFGwhere,"\p",0L,-1,myType,0L,&reply);
```

```
    if (reply.good) {
        PstrCopy(ReplyName,reply.fName);
        PtoCstr(ReplyName);
        SetVol("\p",reply.vRefNum);
    }
    return(reply.good);
}

PutOpen(Text,OpenName)      /* Dialog for Write file */
char  Text[], OpenName[];
{
    SFReply    reply;

    CtoPstr(Text);
    CtoPstr(OpenName);
    SFPutFile(SFPwhere,Text,OpenName,0L,&reply);
    PtoCstr(Text);
    if (reply.good) {
        PstrCopy(OpenName,reply.fName);
        PtoCstr(OpenName);
        SetVol("\p",reply.vRefNum);
    }
    else
        PtoCstr(OpenName);
    return(reply.good);
}
```

```
/*-------------------------------------------------
 *  File name : MainEvent.c
 *  Function  : Perform periodic task of Event and
 *              data from serial port
 *-------------------------------------------------*/

#include  "stdio.h"
#include  "EventMgr.h"
#include  "WindowMgr.h"
#include  "MenuMgr.h"
#include  "FileMgr.h"
#include  "STDfilePkg.h"
#include  "ToolBoxUtil.h"
#include  "OSUtil.h"
#include  "Unix.h"

#include  "defined.inc"
#include  "ExternDef.inc"

extern char MacPaint[];
extern char MacDraw[];
extern char SuperPaint[];

#define   BlockLimit  5    /* retry 5  times  */
#define   BlockDelay  80   /* delay 80 second */
int       BlockRetry;

OSType  MacPaintCreat  = 'MPNT';   /* Creator for MacPaint */
OSType  MacDrawCreat   = 'MDRW';   /* Creator for MacDraw  */
OSType  SuperCreat     = 'SPNT';   /* Creator for SuperPaint */
OSType  UserCreat      = '????';   /* Creator for User defined */

extern  FILE *myWindow;
EventRecord   myEvent;      /* my event record  */

int MainEvent()
/*  Looking for any pending Event and Data from serial port */
{
   DialogPtr DlgPtr;
   int     itemHit;
   long  anyRecv;

   while ( !Done ) {
      SystemTask();
      if (GetNextEvent(everyEvent, &myEvent) ) {
        if (! StdEvent(&myEvent)) {
          if ( IsDialogEvent(&myEvent) ) {
             if ( DialogSelect(&myEvent,&DlgPtr,&itemHit) ) {
                 Abort = 1;
             }
          }
          else {
             switch (myEvent.what) {
               case mouseDown : MouseProc();  /* user press mouse */
                                break;
```

```
                case keyDown   :
                case autoKey   : KeyProcess(); /* user press keyboard */
                                 break;
                      default  : break;
                }
             }
          }
       }
       SerGetBuf(refIN, &anyRecv);
       if ( anyRecv > 0L )  {   /* if any data receive */
            Input232();         /* get and process it  */
       }
       else
          TestDelay();
    }
}

int TestDelay()  /*  test for abort and delay time */
{
   if  ( termMode == SendMode || termMode == RecvMode ) {
       if ( Abort ) {
          if  ( termMode == SendMode ) {
               ErrCode = 204;  /* abort by operator in send mode */
               SendChar(DLE);
               SendChar(EOT);
          }
          else {
               ErrCode = 304;  /* abort by operator in recv mode */
               SendChar(EOT);
          }
          HouseKeep(&ABORTstr); /* perform house keeping */
       }
       else  {
          if  ( TickCount() > ExpTime ) {
              if ( BlockRetry < BlockLimit ) {
                  if  (termMode == SendMode ) {
                     SendChar(ENQ);
                     SysBeep(2);
                     DisplayTrans(&ENQstr,100,TransfByte,ErrRec);
                     BlockRetry++;
                     ExpTime = ExpTime + 60 * BlockDelay;
                     sleep(2);
                     DisplayTrans(&WAITstr,100,TransfByte,ErrRec);
                  }
              }
              else {
                if  ( termMode == SendMode ) {
                   ErrCode = 210;  /* abort by operator in send mode */
                   SendChar(DLE);
                   SendChar(EOT);
                }
                else {
                   ErrCode = 310;  /* abort by operator in recv mode */
                   SendChar(EOT);
                }
```

```
                HouseKeep(&ABORTstr); /* perform house keeping */
              }
            }
          }
        }

}

int MouseProc(myEvent)  /* process mouse event */
EventRecord myEvent;
{

        switch (FindWindow(myEvent.where,&whichWindow)) {
        case inDesk     : SysBeep(10);
                          break;
        case inSysWindow : SystemClick(&myEvent,whichWindow);
                          break;
        case inMenuBar : return( DoCommand(MenuSelect(myEvent.where)));
                          break;
        case inContent : if (whichWindow != FrontWindow())
                            SelectWindow(whichWindow);
                          break;
        }
        return;
}


int  DoCommand( mResult )  /* process selected command */
long mResult;
{
    int  theItem, temp;
    GrafPtr savePort;
    Str255  accessory;

    theItem = LoWord( mResult );
    switch (HiWord(mResult)) {
    case  AppleID:  GetItem(myMHdl[AppleMenu],theItem,accessory);
                    GetPort(&savePort);
                    OpenDeskAcc(accessory);
                    SetPort(savePort);
                    break;
    case  OptID  :  DoOption(theItem);
                    break;
    case  SetupID:  Setup(theItem);
                    break;
    case  CmdID  :  CmdAct(theItem);
    }
    HiliteMenu(0);
    return(1);
}

DoOption(theItem)  /* process option command */
int   theItem;
{
     switch (theItem) {
```

```
        case  quit  : Done = 1;  /* terminate application */
                      break;
        }
}

Setup(theItem)   /* setup any select item */
int theItem;
{
     switch (theItem) {
     case  param      : Setup_Param();  /* set parameter file and 232 */
                        break;
     case  setFileType: Setup_FileType(); /* set binary filetype */
                        break;
     case  setLocSend : LocalSend();    /* set local send filename */
                        break;
     case  setLocRecv : LocalRecv();    /* set local receive filename */
                        break;
     case  setRmtSend : RmtSendName();  /* set remote send filename */
                        break;
     case  setRmtRecv : RmtRecvName();  /* set remote receive filename */
                        break;
     case  dialNum    : SetDialNum();   /* set initial char for modem */
     }
}

CmdAct(theItem)   /* Process Command selected */
int theItem;
{
     switch (theItem) {
     case  sendfile : SendCmd();  /* Send File */
                      break;
     case  recvfile : RecvCmd();  /* receive file */
                      break;
     case  dialing  : DialNum();  /* sending initial char for modem */
                      break;
     }
}


int  Setup_FileType()  /* setup file type for binary file */
{
  DialogPtr TypeDlgPtr;
  int     dialogType;
  Handle  dialogHDL;
  Rect    dialogRect, dummyRect;
  PenState Ps;
  GrafPtr  savePort;
  int      item;

   TypeDlgPtr = GetNewDialog(133,0L,-1);
   strcpy(TypeDlg,FileType);
   CtoPstr(TypeDlg);
   binDlg = bin;

   GetDItem(TypeDlgPtr,1,&dialogType,&dialogHDL,&dialogRect);
```

```
    GetPenState(&Ps);
    dummyRect = dialogRect;
    GetPort(&savePort);
    SetPort(&((WindowPeek)TypeDlgPtr)->port);
    PenSize(3,3);
    InsetRect(&dummyRect,-4,-4);
    FrameRoundRect(&dummyRect,16,16);
    SetPort(savePort);
    SetPenState(&Ps);

    GetDItem(TypeDlgPtr,4,&dialogType,&dialogHDL,&dialogRect);
    SetIText(dialogHDL,TypeDlg);     /* set up File type */

    item = binDlg + 4;
    GetDItem(TypeDlgPtr,item,&dialogType,&dialogHDL,&dialogRect);
    SetCtlValue(dialogHDL,1);
    if ( binDlg == 4 )
       SelIText(TypeDlgPtr,4,0,5);   /* select text if user defined */

    do {
        ModalDialog(0L,&itemhit);
        TypeSelect(TypeDlgPtr,itemhit);
       }
    while (itemhit != 1 && itemhit != 2 );

    if  ( itemhit == 1 ) {
        bin = binDlg;
        PtoCstr(TypeDlg);
        strncpy(FileType,TypeDlg,4);
        FileType[4] = '\0';
    }
   DisposDialog(TypeDlgPtr);
}

int  TypeSelect(TypeDlgPtr,item) /* process the Selected type */
DialogPtr  TypeDlgPtr;
int        item;
{
int     dialogType;
Handle  dialogHDL;
Rect    dialogRect;
int     i;

  switch ( item ) {
   case   1 :
   case   2 :GetDItem(TypeDlgPtr,item,&dialogType,&dialogHDL,&dialogRect);
             SetCtlValue(dialogHDL,1);
             break;
   case   4 : if ( binDlg != 4 ) {
                SysBeep(2);
                GetDItem(TypeDlgPtr,item,&dialogType,&dialogHDL,
                        &dialogRect);
                SetIText(dialogHDL,TypeDlg);
              }
              else {
```

```
                GetDItem(TypeDlgPtr,item,&dialogType,&dialogHDL,
                        &dialogRect);
                GetIText(dialogHDL,&TypeDlg);
              }
              break;
    case   5 :
    case   6 :
    case   7 :
    case   8 : GetDItem(TypeDlgPtr,binDlg+4,&dialogType,
                        &dialogHDL,&dialogRect);
              SetCtlValue(dialogHDL,0);
              binDlg = item - 4;
              GetDItem(TypeDlgPtr,item,&dialogType,
                        &dialogHDL,&dialogRect);
              SetCtlValue(dialogHDL,1);
              switch ( item ) {
               case 5 :  /* select MacPaint */
                 PstrCopy(TypeDlg,MacPaint);
                 GetDItem(TypeDlgPtr,4,&dialogType,&dialogHDL,&dialogRect);
                 SetIText(dialogHDL,TypeDlg);
                 break;
               case 6 :  /* select MacDraw */
                 PstrCopy(TypeDlg,MacDraw);
                 GetDItem(TypeDlgPtr,4,&dialogType,&dialogHDL,&dialogRect);
                 SetIText(dialogHDL,TypeDlg);
                 break;
               case 7 :  /* select SuperPaint */
                 PstrCopy(TypeDlg,SuperPaint);
                 GetDItem(TypeDlgPtr,4,&dialogType,&dialogHDL,&dialogRect);
                 SetIText(dialogHDL,TypeDlg);
                 break;
               case 8 :  /* user define type */
                 SelIText(TypeDlgPtr,4,0,4);
             }

  }
}


int  Setup_Param()  /* Setup parameter file and rs422 */
{
   int  ConfNo;
   unsigned count,i;

    do {
       if ( GetOpen(OpenParm) ) {
         if ( (ConfNo = open(OpenParm,0_RDWR+0_BINARY)) == EOF ) {
            CfgOpenErr(&reply.fName[0]);
         }
       }
       else  {
           return(0);
       }
    }
    while ( ConfNo == EOF );
```

```
    count = sizeof(parmrec);
    i = read(ConfNo,&parmrec,count);
    close(ConfNo);    /* close config file */
    if ((i != sizeof(parmrec)) || (strcmp(myCFG,&parmrec.IdCfg) != 0)) {
       CtoPstr(OpenParm);
       CfgFmtErr(OpenParm);
       PtoCstr(OpenParm);
       return(0);
    }
    if ( ! Converse(1)) {
       return(0);
    }
    MoveDialogReal(); /* move to real parameter */
    Init232();        /* Initial RS232 */
    MoveRealFile();   /* move to File parameter */
    SaveParameter();  /* save parameter */

}

int LocalSend()   /* set local send filename */
{
    char anyTempStr[60];

    strcpy(anyTempStr,SendName);
    if (GetDlgName("Local Send File Name",anyTempStr)) {
       strcpy(SendName,anyTempStr);
    }
}

int LocalRecv()  /* set local receive filename */
{
    char anyTempStr[60];

    strcpy(anyTempStr,RecvName);
    if (GetDlgName("Local Receive File Name",anyTempStr)) {
       strcpy(RecvName,anyTempStr);
    }
}


int RmtSendName()  /* set remote send filename */
{
    return(SetRemoteName("Remote Send File Name",'O'));
}


int RmtRecvName() /* set remote received filename */
{
    return(SetRemoteName("Remote Receive File Name",'I'));
}

int SetRemoteName(dispMsg,IOMode) /* setup remote name */
char  dispMsg[];
char  IOMode;
{
```

```
    long SendLen;
    char RecvChar;
    char anyTempStr[60];        /* string result from dialog */
    DialogPtr  MsgDlgPtr;

    anyTempStr[0] = '\0';

    if (GetDlgName(dispMsg,anyTempStr)) {
       if ( ( SendLen = strlen(anyTempStr)) != 0 ) {
          SendChar(ENQ);
          SendChar(IOMode);
          /* set dialog to show message */
          MsgDlgPtr = GetNewDialog(132,0L,-1);
          RemoteMsg(MsgDlgPtr,103,1,anyTempStr);
          GetNext(&RecvChar,1);
          if ( Expired ) {
             DisposDialog(MsgDlgPtr);
             CmdAlert(102);
             return(0);
          }
          if ( RecvChar == ACK ) {
             FSWrite(refOUT, &SendLen, anyTempStr);
             SendChar(ETB);
          }
          else {
             DisposDialog(MsgDlgPtr);
             CmdAlert(104);
             return(0);
          }
          GetNext(&RecvChar,1);
          if ( Expired ) {
             DisposDialog(MsgDlgPtr);
             CmdAlert(102);
             return(0);
          }
          if ( RecvChar != ACK ) {
             DisposDialog(MsgDlgPtr);
             CmdAlert(104);
             return(0);
          }
          else {
             ErrCode = 0;
             DisposDialog(MsgDlgPtr);
          }
       }
    }
    return(1);
}


int RemoteMsg(MsgDlgPtr,msgNo,item,fileName)
/* show message while sending file name */
int msgNo, item;
DialogPtr MsgDlgPtr;
```

```
char fileName[];
{
   Handle    dialogHdl;
   Rect      dialogRect;
   int       dialogType;
   char      *TextPtr;

   Search(msgNo,&TextPtr);
   GetDItem(MsgDlgPtr,1,&dialogType,&dialogHdl,&dialogRect);
   SetIText(dialogHdl,TextPtr);
   GetDItem(MsgDlgPtr,2,&dialogType,&dialogHdl,&dialogRect);
   CtoPstr(fileName);
   SetIText(dialogHdl,fileName);
   PtoCstr(fileName);
}

int GetDlgName(anystr1,anystr2)
/*  get name from user keyboard */
char anystr1[], anystr2[];
{
  Rect     dummyRect;
  PenState Ps;
  GrafPtr  savePort;

   myDialogPtr = GetNewDialog(130,0L,-1);
   GetDItem(myDialogPtr,1,&dialogType,&dialogHDL,&dialogRect);
   GetPenState(&Ps);
   dummyRect = dialogRect;
   GetPort(&savePort);
   SetPort(&((WindowPeek)myDialogPtr)->port);
   PenSize(3,3);
   InsetRect(&dummyRect,-4,-4);
   FrameRoundRect(&dummyRect,16,16);
   SetPort(savePort);
   SetPenState(&Ps);
   GetDItem(myDialogPtr,3,&dialogType,&dialogHDL,&dialogRect);
   CtoPstr(anystr1);
   SetIText(dialogHDL,anystr1);
   PtoCstr(anystr1);
   GetDItem(myDialogPtr,4,&dialogType,&dialogHDL,&dialogRect);
   CtoPstr(anystr2);
   SetIText(dialogHDL,anystr2);
   PtoCstr(anystr2);
   SelIText(myDialogPtr,4,0,100);
   do {
       ModalDialog(0L,&itemhit);
       if (itemhit == 4 ) {
          GetDItem(myDialogPtr,4,&dialogType,&dialogHDL,&dialogRect);
          GetIText(dialogHDL,anystr2);
          PtoCstr(anystr2);
       }
   }
   while (itemhit != 1 && itemhit != 2 );
   CloseDialog(myDialogPtr);
   if  ( itemhit == 1 )
```

```
      return(1);
   else
      return(0);
}


int  SetDialNum()  /* setting User defined character for modem */
{
    char anyTempStr[60];

    strcpy(anyTempStr,DialName);
    if (GetDlgName("Auto Dial Characters",anyTempStr)) {
       strcpy(DialName,anyTempStr);
       return(1);
    }
    else
      return(0);
}

int  DialNum()  /* check dial character and sending to serial port */
{
   DialogPtr DialDlgPtr;
   char  TempStr[10];
   int   i;
   long  SendLen;
   if (SetDialNum()) {
      for (i = 0; i < 5 ; i++ ) {
         TempStr[i] = toupper(DialName[i]);
      }
      if ( strncmp(TempStr,"#NONE",5) != 0 ) {
         SysBeep(5);
         DialDlgPtr = GetNewDialog(132,0L,-1);
         DialMsg(DialDlgPtr);
         SendLen = strlen(DialName);
         FSWrite(refOUT, &SendLen,DialName);
         DisposDialog(DialDlgPtr);
      }
   }
}

int DialMsg(DialDlgPtr)
/* show message  while sending dial character */
DialogPtr DialDlgPtr;
{
   Handle    dialogHdl;
   Rect      dialogRect;
   int       dialogType;
   char      *TextPtr;

   Search(105,&TextPtr);
   GetDItem(DialDlgPtr,1,&dialogType,&dialogHdl,&dialogRect);
   SetIText(dialogHdl,TextPtr);
   GetDItem(DialDlgPtr,2,&dialogType,&dialogHdl,&dialogRect);
   CtoPstr(DialName);
   SetIText(dialogHdl,DialName);
```

```
    PtoCstr(DialName);
}


int  MoveDialogReal()  /* user defined parameter to real parameter */
{
    baud = baudDlg;
    data = dataDlg;
    parity = parityDlg;
    stop = stopDlg;
    mode = modeDlg;
    port = portDlg;
    bin  = binDlg;
    PtoCstr(TypeDlg);
    strncpy(FileType,TypeDlg,4);
    FileType[4] = '\0';
    CtoPstr(TypeDlg);
    PstrCopy(SendName,sendDlg);
    PtoCstr(SendName);
    PstrCopy(RecvName,recvDlg);
    PtoCstr(RecvName);
    PstrCopy(DialName,dialDlg);
    PtoCstr(DialName);
}


int MoveRealFile()  /* move real parameter to write file */
{
    parmrec.fbaud = baud;
    parmrec.fdata = data;
    parmrec.fparity = parity;
    parmrec.fstop = stop;
    parmrec.fmode = mode;
    parmrec.fport = port;
    parmrec.fbin  = bin;
    strcpy(parmrec.fType,FileType);
    strcpy(parmrec.fsend,SendName);
    strcpy(parmrec.frecv,RecvName);
    strcpy(parmrec.fdial,DialName);
}

int  SaveParameter()
/* Inquire user and write parameter to config file */
{
    int  fileNo;

    if ( PutOpen("Save Parameter As :",OpenParm) ) {
        if ( (fileNo = creat(OpenParm,O_RDWR+O_BINARY)) == EOF ) {
            CfgOpenErr(&reply.fName);
            return(0);                  /* not complete */
        }
        strcpy(parmrec.IdCfg,myCFG);
        if (write(fileNo,&parmrec.IdCfg,sizeof(parmrec)) == EOF ) {
            CfgWritErr(reply.fName);
        }
```

```
        close(fileNo);
    }
}


int  HouseKeep(anyPtr)
char *anyPtr;
{
/*                      House Keeping Routine
     for   perform any additional requirement when end file transfer */

char  VolName[40];
int   VolRef;
FInfo FileInfo;
int   rs;
char  temp[5];

   SysBeep(5);
   DisplayTrans(anyPtr,ErrCode,TransfByte,ErrRec);
   close(fileRef);
   if ( termMode == RecvMode) {
      if ( mode == 1 ) {  /* Binary Mode */
         GetVol(VolName,&VolRef);
         CtoPstr(RecvName);
         GetFInfo(RecvName,VolRef,&FileInfo);
         strncpy(&FileInfo.fdType,FileType,4);
         switch ( bin ) {
           case 1 : FileInfo.fdCreator = MacPaintCreat;
                    break;
           case 2 : FileInfo.fdCreator = MacDrawCreat;
                    break;
           case 3 : FileInfo.fdCreator = SuperCreat;
                    break;
           case 4 : FileInfo.fdCreator = UserCreat;
         }
         SetFInfo(RecvName,VolRef,&FileInfo);
         PtoCstr(RecvName);
         FlushVol(VolName,VolRef);
      }
   }

   MouseWait(2);       /* wait for 2 sec before clear transfer dialog */
   DisposDialog(FileDlgPtr);
   LogStatFile();
   TransferDlg = 0;
   Abort = 0;
   ErrCode = 0;
   retry = 0;
   termMode = normal;  /* force to normal Mode */
   EscSeq = 0;         /* No Escape Sequence   */
   ErrRec = 0;         /* No Error Record  */
   Expired = 0;        /* No Expired */
}

int MouseWait(delay) /* wait if user press mouse when end of transfer */
```

```
int delay;
{
long  time;
    time = TickCount() + delay * 60;    /* time when expired */
    while ( TickCount() < time ) {
      while ( Button() );   /* wait for mouse down */
    }
}

int LogStatFile()   /* logging status of file transmission */
{
  int  vref;
  char StatLine[100];
  char NumStr[22];    /* string for Convert Numeric  and date */
  long myTime;
  DateTimeRec myDT;

  vref = open(StatName,O_WRONLY | O_APPEND);
  if ( vref != EOF ) {
     if ( termMode == SendMode ) {
        strcpy(StatLine,Sendstr);
        strcat(StatLine,SendName);
     }
     else {
        strcpy(StatLine,Recvstr);
        strcat(StatLine,RecvName);
     }

     GetDateTime(&myTime);
     Secs2Date(myTime,&myDT);
     sprintf(NumStr," %02d/%02d/%02d  %02d:%02d:%02d",myDT.year-1900,
            myDT.month, myDT.day, myDT.hour, myDT.minute, myDT.second);
     strcat(StatLine,NumStr);

     NumToString(TransfByte,NumStr);
     PtoCstr(NumStr);
     strcat(StatLine," Byte Transfer = ");
     strcat(StatLine,NumStr);
     NumToString(ErrRec,NumStr);
     PtoCstr(NumStr);
     strcat(StatLine," NAKS = ");
     strcat(StatLine,NumStr);
     NumToString(ErrCode,NumStr);
     PtoCstr(NumStr);
     strcat(StatLine," Error Code = ");
     strcat(StatLine,NumStr);
     strcat(StatLine,"\n");
     write(vref,StatLine,strlen(StatLine));
     close(vref);
  }
}

int  CmdAlert(anyErr)
int  anyErr;
{
```

```
/*              Command End Routine
      for  display Any desired Message  transfer */
char *myMsg;

   Search(anyErr,&myMsg);
   ParamText(myMsg,"\p","\p","\p");
   Alert( ErrorAlert, 0L);

   ErrCode = 0; retry = 0;
   termMode = normal;  /* force to normal Mode */
   EscSeq = 0;         /* No Escape Sequence   */
   ErrRec = 0;         /* No Error Record  */
   Expired = 0;        /* No Expired */
}
```

```
/*-------------------------------------------------
 *  File name : myDialog.c
 *  Function  : perform conversation with user while
 *              setting configuration of system.
 *-------------------------------------------------*/

#include  "stdio.h"
#include  "EventMgr.h"
#include  "WindowMgr.h"
#include  "MenuMgr.h"
#include  "StdFilePkg.h"
#include  "ToolBoxUtil.h"

#include  "defined.inc"
#include  "ExternDef.inc"


char MacPaint[]   = "\pPNTG";  /* File Type of MacPaint */
char MacDraw[]    = "\pPICT";  /* File Type of MacDraw */
char SuperPaint[] = "\pSPTG";  /* File Type of SuperPaint */

static char senddef[] = "\pEMSend";
static char recvdef[] = "\pEMReceive";
static char dialdef[] = "\p#NONE";

int  Converse(any)
/*  Conversation with user for setting up config parameter */
int  any;              /* 1 = from file , 2 = from default */
{
   myDialogPtr = GetNewDialog(128,0L,-1);
   if ( any == 1 )
       setFileDlg();   /*  setup dialog data */
   else
       setdef232();    /* use default parameter */
   ShowParm(myDialogPtr);
   do {
       ModalDialog(0L,&itemhit);
       ParmSelect(myDialogPtr,itemhit);
      }
   while (itemhit != 1 && itemhit != 2 );
   DisposDialog(myDialogPtr);
   if (itemhit == 1 )
      return(1);
   else
      return(0);
}


int setFileDlg()  /* move config data from file to dialog */
{
    baudDlg = parmrec.fbaud;
    dataDlg = parmrec.fdata;
    parityDlg = parmrec.fparity;
    stopDlg = parmrec.fstop;
    modeDlg = parmrec.fmode;
```

```
    portDlg = parmrec.fport;
    binDlg  = parmrec.fbin;
    strcpy(TypeDlg,parmrec.fType);
    strcpy(sendDlg,parmrec.fsend);
    strcpy(recvDlg,parmrec.frecv);
    strcpy(dialDlg,parmrec.fdial);
    CtoPstr(TypeDlg);
    CtoPstr(sendDlg);
    CtoPstr(recvDlg);
    CtoPstr(dialDlg);
}

int setdef232()   /* set default value for serial port */
{

    baudDlg = 9600;                 /* 9600 BPS  */
    dataDlg = 8;                    /* 8 data bit*/
    parityDlg = 0;                  /* no parity */
    stopDlg = 1;                    /* 1 stop bit */
    modeDlg = 2;                    /* TEXT  mode*/
    portDlg = 1;                    /* modem port */
    binDlg  = 1;                    /* MAC PAINT */
    PstrCopy(TypeDlg,MacPaint);     /* PNTG      */
    PstrCopy(sendDlg,senddef);
    PstrCopy(recvDlg,recvdef);
    PstrCopy(dialDlg,dialdef);
}


int ParmSelect(anyDialogPtr,anyitem)
/* show selected config data */
DialogPtr anyDialogPtr;
int       anyitem;
{
  int  i;
  int       DlogType;
  Handle    DlogHDL;
  Rect      DlogRect;

    if ( anyitem == 1 || anyitem == 2 ) {
      GetDItem(anyDialogPtr,anyitem,&DlogType,&DlogHDL,&DlogRect);
      SetCtlValue(DlogHDL,1);
      return(0);
    }
    if ( anyitem > 3 && anyitem < 11 ) { /* found baud rate */
       switch ( anyitem ) {   /* set corresponding baud rate */
       case 4 : baudDlg = 150;
                break;
       case 5 : baudDlg = 300;
                break;
       case 6 : baudDlg = 600;
                break;
       case 7 : baudDlg = 1200;
                break;
       case 8 : baudDlg = 2400;
```

```
            break;
   case 9 : baudDlg = 4800;
            break;
   case 10: baudDlg = 9600;
            break;
   };
   for ( i = 4; i ( 11 ; i++) {
      GetDItem(anyDialogPtr,i,&DlogType,&DlogHDL,&DlogRect);
      if ( i == anyitem )
         SetCtlValue(DlogHDL,1);
      else
         SetCtlValue(DlogHDL,0);
    };
    return(0);
 }
if ( anyitem ) 11 && anyitem ( 14 ) { /* found data bits */
   if  ( anyitem == 12 )
       dataDlg = 7;
   else
       dataDlg = 8;
   for ( i = 12; i ( 14 ; i++) {
      GetDItem(anyDialogPtr,i,&DlogType,&DlogHDL,&DlogRect);
      if ( i == anyitem )
         SetCtlValue(DlogHDL,1);
      else
         SetCtlValue(DlogHDL,0);
    };
    return(0);
 }
if ( anyitem ) 14 && anyitem ( 18 ) { /* found parity bits */
   if ( anyitem == 15 )
      parityDlg = 0;              /* none parity */
   else if (anyitem == 16 )
            parityDlg = 1;        /* odd parity */
        else
            parityDlg = 2;        /* even parity */
   for ( i = 15; i ( 18 ; i++) {
      GetDItem(anyDialogPtr,i,&DlogType,&DlogHDL,&DlogRect);
      if ( i == anyitem )
         SetCtlValue(DlogHDL,1);
      else
         SetCtlValue(DlogHDL,0);
  * };
    return(0);
 }
if ( anyitem ) 18 && anyitem ( 22 ) { /* found stop bits */
   if ( anyitem == 19 )
      stopDlg = 1;                /* 1 stop bit  */
   else if (anyitem == 20 )
            stopDlg = 15;         /* 1.5 stop bit */
        else
            stopDlg = 2;         /* 2 stop bit */
   for ( i = 19; i ( 22 ; i++) {
      GetDItem(anyDialogPtr,i,&DlogType,&DlogHDL,&DlogRect);
      if ( i == anyitem )
```

```
            SetCtlValue(DlogHDL,1);
         else
            SetCtlValue(DlogHDL,0);
      };
      return(0);
   }
   if ( anyitem > 22 && anyitem < 25 ) { /* found Send Mode */
      if ( anyitem == 23 )
         modeDlg = 1;          /* send Binary Mode */
      else
         modeDlg = 2;          /* send Text Mode */
      for ( i = 23; i < 25 ; i++) {
         GetDItem(anyDialogPtr,i,&DlogType,&DlogHDL,&DlogRect);
         if ( i == anyitem )
            SetCtlValue(DlogHDL,1);
         else
            SetCtlValue(DlogHDL,0);
      };
      return(0);
   }
   if ( anyitem > 25 && anyitem < 28 ) { /* found Send Port */
      if ( anyitem == 26 )
         portDlg = 1;          /* send Port A */
      else
         portDlg = 2;          /* send Port B */
      for ( i = 26; i < 28 ; i++) {
         GetDItem(anyDialogPtr,i,&DlogType,&DlogHDL,&DlogRect);
         if ( i == anyitem )
            SetCtlValue(DlogHDL,1);
         else
            SetCtlValue(DlogHDL,0);
      };
      return(0);
   }
   if ( anyitem > 33 && anyitem < 40 ) { /* Set Binary Type */
      if ( anyitem > 35 )
        for ( i = 36; i < 40 ; i++) {
          GetDItem(anyDialogPtr,i,&DlogType,&DlogHDL,&DlogRect);
          if ( i == anyitem )
             SetCtlValue(DlogHDL,1);
          else
             SetCtlValue(DlogHDL,0);
        };
      switch ( anyitem ) {
      case 36 : binDlg = 1;   /* select MacPaint */
                PstrCopy(TypeDlg,MacPaint);
                GetDItem(anyDialogPtr,35,&DlogType,&DlogHDL,&DlogRect);
                SetIText(DlogHDL,TypeDlg);
                break;
      case 37 : binDlg = 2;   /* select MacDraw */
                PstrCopy(TypeDlg,MacDraw);
                GetDItem(anyDialogPtr,35,&DlogType,&DlogHDL,&DlogRect);
                SetIText(DlogHDL,TypeDlg);
                break;
      case 38 : binDlg = 3;   /* select SuperPaint */
```

```
                PstrCopy(TypeDlg,SuperPaint);
                GetDItem(anyDialogPtr,35,&DlogType,&DlogHDL,&DlogRect);
                SetIText(DlogHDL,TypeDlg);
                break;
        case 39 : binDlg = 4;   /* user define type */
                SelIText(anyDialogPtr,35,0,4);
                break;
        case 35 : /* user defined type */
                if  ( binDlg != 4 ) {
                  SysBeep(3);
                  GetDItem(anyDialogPtr,35,&DlogType,&DlogHDL,&DlogRect);
                  SetIText(DlogHDL,TypeDlg);
                }
                else {
                  GetDItem(anyDialogPtr,35,&DlogType,&DlogHDL,&DlogRect);
                  GetIText(DlogHDL,TypeDlg);
                }
        }
        return(0);
    }
    switch ( anyitem) {
    case  29 : {     .
                GetDItem(anyDialogPtr,29,&DlogType,&DlogHDL,&DlogRect);
                GetIText(DlogHDL,sendDlg);
               };
    case  31 : {
                GetDItem(anyDialogPtr,31,&DlogType,&DlogHDL,&DlogRect);
                GetIText(DlogHDL,recvDlg);
               };
    case  33 : {
                GetDItem(anyDialogPtr,33,&DlogType,&DlogHDL,&DlogRect);
                GetIText(DlogHDL,dialDlg);
               };
    };
}

int  ShowParm(anyDialogPtr)  /* show current user parameter  */
DialogPtr anyDialogPtr;
{
  int     dialogType;
  Handle  dialogHDL;
  Rect    dialogRect, dummyRect;
  PenState Ps;
  GrafPtr  savePort;
  int      item;

    GetDItem(anyDialogPtr,1,&dialogType,&dialogHDL,&dialogRect);
    GetPenState(&Ps);
    dummyRect = dialogRect;
    GetPort(&savePort);
    SetPort(&((WindowPeek)anyDialogPtr)->port);
    PenSize(3,3);
    InsetRect(&dummyRect,-4,-4);
    FrameRoundRect(&dummyRect,16,16);
    SetPort(savePort);
```

```
SetPenState(&Ps);

switch (baudDlg) {
  case 150  : item = 4;
              break;
  case 300  : item = 5;
              break;
  case 600  : item = 6;
              break;
  case 1200 : item = 7;
              break;
  case 2400 : item = 8;
              break;
  case 4800 : item = 9;
              break;
  case 9600 : item = 10;
              break;
};
GetDItem(anyDialogPtr,item,&dialogType,&dialogHDL,&dialogRect);
SetCtlValue(dialogHDL,1);

if ( dataDlg == 7 )
   item = 12;
else
   item = 13;
GetDItem(anyDialogPtr,item,&dialogType,&dialogHDL,&dialogRect);
SetCtlValue(dialogHDL,1);

if ( parityDlg == 0 )
   item = 15;
else  if (parityDlg == 1 )
         item = 16;
      else
         item = 17;
GetDItem(anyDialogPtr,item,&dialogType,&dialogHDL,&dialogRect);
SetCtlValue(dialogHDL,1);

if  (stopDlg == 1 )
    item = 19;                    /* 1 stop bit */
else  if (stopDlg == 15)
         item = 20;               /* 1.5 stop bit */
      else
         item = 21;               /* 2 stop bit */
GetDItem(anyDialogPtr,item,&dialogType,&dialogHDL,&dialogRect);
SetCtlValue(dialogHDL,1);

if (modeDlg == 1 )
    item = 23;
else
    item = 24;
GetDItem(anyDialogPtr,item,&dialogType,&dialogHDL,&dialogRect);
SetCtlValue(dialogHDL,1);

if (portDlg == 1 )
    item = 26;
```

```
    else
        item = 27;
    GetDItem(anyDialogPtr,item,&dialogType,&dialogHDL,&dialogRect);
    SetCtlValue(dialogHDL,1);

    switch ( binDlg ) {
      case  1 : item = 36;
                break;
      case  2 : item = 37;
                break;
      case  3 : item = 38;
                break;
      case  4 : item = 39;
    }
    GetDItem(anyDialogPtr,item,&dialogType,&dialogHDL,&dialogRect);
    SetCtlValue(dialogHDL,1);
    GetDItem(anyDialogPtr,35,&dialogType,&dialogHDL,&dialogRect);
    SetIText(dialogHDL,TypeDlg);  /* set File Type */


    GetDItem(anyDialogPtr,29,&dialogType,&dialogHDL,&dialogRect);
    SetIText(dialogHDL,sendDlg);  /* set send name */

    GetDItem(anyDialogPtr,31,&dialogType,&dialogHDL,&dialogRect);
    SetIText(dialogHDL,recvDlg);  /* set recv name */

    GetDItem(anyDialogPtr,33,&dialogType,&dialogHDL,&dialogRect);
    SetIText(dialogHDL,dialDlg);  /* set dial number */

    SelIText(anyDialogPtr,29,0,100); /* select Sendfile Name */
}
```

```
/*************************************************************
 *  File name : Input232.C
 *  Function  : process data from serial port depending on status
 *              such as display on console, or File transfer
 *              protocol
 *-----------------------------------------------------------*/

#include  "stdio.h"
#include  "EventMgr.h"
#include  "QuickDraw.h"
#include  "FontMgr.h"
#include  "WindowMgr.h"
#include  "MenuMgr.h"
#include  "Unix.h"
#include  "STDfilePkg.h"
#include  "ToolBoxUtil.h"

#include  "defined.inc"
#include  "ExternDef.inc"

extern FILE *myWindow;
extern int  BlockRetry;

int   Input232()
/* process input from serial port depending on mode of terminal */
{
unsigned char inpchar;

    GetNext(&inpchar,1);
    switch (termMode) {
       case   normal : if ( inpchar == ENQ ) {
                          termMode = CmdMode;
                          TimeStamp();
                       }
                       else {
                          Display(inpchar);
                       }
                       break;
       case  CmdMode : CmdRTN(inpchar);
                         break;
       case  SendMode: SendRTN(inpchar);
                         break;
       case  RecvMode: RecvRTN(inpchar);
                         break;
   }
}

int  TimeStamp()
{
     ExpTime = TickCount() + DelayTime * 60;
     BlockRetry = 0;
}

int  SendChar(anydata)
unsigned char anydata;
```

```
{
   long  Length = 1L;
   FSWrite( refOUT, &Length,&anydata );
}

int  CmdRTN(inpchar)
unsigned char inpchar;
{
long  Length;
     switch (inpchar) {
     case  'N' : SendRCOMID();
                 termMode = normal;
                 break;
     case  'I' :
     case  'O' : SendChar(ACK);
                 GetNameETB();
                 if (Expired) {
                   ErrCode = 102;
                 }
                 else {
                   SendChar(ACK);
                   if (inpchar == 'I')
                      strcpy(RecvName,GetName);
                   else
                      strcpy(SendName,GetName);
                 }
                 termMode = normal;
                 break;
      case  'R' : SendChar(ACK);
                 InitRecvMode();
                 DisplayTrans(ACKstr,100,TransfByte,ErrRec);
                 OpenOutput();
                 if ( fileRef == EOF ) {
                    ErrCode = 301;
                    HouseKeep(&BLANKstr);
                 }
                 else
                    DisplayTrans(&WAITstr,100,TransfByte,ErrRec);
                 TimeStamp();
                 break;
      case  'S' : InitSendMode();
                 DisplayTrans(&BLANKstr,100,TransfByte,ErrRec);
                 OpenInput();
                 if (fileRef == EOF ) {
                    ErrCode = 201;
                    HouseKeep(&WAITstr);
                 }
                 else {
                    ReadFileNext() ; /* read block from file */
                    if ( ByteCount == EOF ) {
                       ErrCode = 202;
                       HouseKeep(&WAITstr);
                    }
                    else {
                       if ( ByteCount > 0 ) {
```

```
                    DisplayTrans(&STXstr,100,TransfByte,ErrRec);
                    SendBlock();    /* send to Port */
                  }
                  else {
                    SendChar(EOT);  /* Reach EOF */
                    ErrCode = 0;
                    HouseKeep(&EOTstr);
                  }
                }
              }
              break;
    default   : termMode = normal;
              Display(inpchar);
    }
}


int  SendRTN(inpchar)
unsigned char inpchar;
{
unsigned char mySEQ;

    switch  (inpchar) {
    case ACK : DisplayTrans(&ACKstr,100,TransfByte,ErrRec);
               GetNext(&mySEQ,1);
               if ( ! Expired ) {
                  if ( Abort ) {
                    ErrCode = 204;
                    SendChar(DLE);
                    SendChar(EOT);
                    HouseKeep(&ABORTstr);
                    return;
                  }
                  if  (SEQ != mySEQ ) {
                    ErrRec++;
                    if ( retry++ < RetryLimit )  {
                      ErrCode = 212;
                      DisplayTrans(&ACKstr,ErrCode,TransfByte,ErrRec);
                      SendBlock();
                    }
                    else  {
                      ErrCode = 213;
                      SendChar(DLE);
                      SendChar(EOT);
                      HouseKeep(&DLEstr);
                    }
                    break;
                  }
                  else  {
                    retry = 0;              /* clear retry error */
                    ErrCode = 0;            /* and error code */
                    ReadFileNext();         /* get next data block */
                    if ( ByteCount == 0 ) {
                      SendChar(EOT);        /* normal End of File */
                      HouseKeep(&EOTstr);
```

```
                }
                else  {
                   if ( ByteCount == EOF) {
                      ErrCode = 202;   /* disk read error */
                      SendChar(DLE);
                      SendChar(EOT);
                      HouseKeep(&DLEstr);
                   }
                   else {
                     SendBlock();     /* any data to send */
                   }
                }
             }
          }
          else  {
              ErrCode = 210;
              HouseKeep(&WAITstr);
          }
          break;

case NAK : SysBeep(5);
          DisplayTrans(&NAKstr,100,TransfByte,ErrRec);
          GetNext(&mySEQ,1);
          ErrRec++;
          if ( ! Expired ) {
             if ( Abort ) {
                ErrCode = 204;
                SendChar(DLE);
                SendChar(EOT);
                HouseKeep(&ABORTstr);
                return;
             }
             if  (SEQ != mySEQ )
                 ErrCode = 212;  /* NAK data */
             else
                 ErrCode = 100;  /* Dummy Space for NAK */

             if ( retry++ < RetryLimit ) {
                 DisplayTrans(&SEQstr,ErrCode,TransfByte,ErrRec);
                 SendBlock();
             }
             else  {
                 ErrCode = 213;
                 SendChar(DLE);
                 SendChar(EOT);
                 HouseKeep(&DLEstr);
             }
          }
          else  {
              ErrCode = 310;
              HouseKeep(&WAITstr);
          }
          break;
case EOT : ErrCode = 211;   /* Abort by Reciever station */
          HouseKeep(&EOTstr);
```

```
                break;
    case BEL : SysBeep(5);
                DisplayTrans(&BELstr,205,TransfByte,ErrRec);
                if ( Abort ) {
                   ErrCode = 204;
                   SendChar(DLE);
                   SendChar(EOT);
                   HouseKeep(&ABORTstr);
                   return;
                }
                sleep(3);
                SendChar(ENQ);
                DisplayTrans(&ENQstr,100,TransfByte,ErrRec);
                sleep(2);
                DisplayTrans(&WAITstr,100,TransfByte,ErrRec);
                TimeStamp();
                break;
    default  : ErrCode = 310;
                HouseKeep(&WAITstr);
    }
}

int   RecvRTN(inpchar)
unsigned char inpchar;
{
long  Lenght;
char  nextChar;

     switch ( inpchar ) {
     case EOT : ErrCode = 0;
                HouseKeep(&EOTstr);
                break;
     case DLE : SysBeep(5);
                DisplayTrans(&DLstr,100,TransfByte,ErrRec);
                GetNext(&nextChar,1);
                if ( ! Expired ) {
                   if ( nextChar == EOT ) {
                        ErrCode = 311;
                        Abort = 1;
                        HouseKeep(&DLEstr);
                   }
                   else {
                        ErrCode = 310;
                        HouseKeep(&WAITstr);
                   }
                }
                else {
                   ErrCode = 310;
                   HouseKeep(&WAITstr);
                }
                break;
      case STX : DisplayTrans(&STXstr,100,TransfByte,ErrRec);
                GetNext(&myMsg.Seq,1);
                if ( Expired ) goto recvEXP;
                GetNext(&myMsg.Len,1);
```

```
                if ( Expired ) goto recvEXP;
                GetNext(&myMsg.myBuff,myMsg.Len);
                if ( Expired ) goto recvEXP;
                GetNext(&myMsg.myBuff[myMsg.Len],1); /* ETB */
                if ( Expired ) goto recvEXP;
                GetNext(&RecvCRC,2);
                if ( Expired ) goto recvEXP;
                ValidRecv();
                break;
        default :
        recvEXP :  ErrCode = 310;
                HouseKeep(&WAITstr);
                break;
        }
        return;
}

int  ValidRecv()
{
unsigned int myCRC;


    myCRC = GenerateCRC(myMsg.Len + 3,&myMsg.Seq);
    if ( Abort ) {
        ErrCode = 304;
        SendChar(EOT);
        if ( RecvCRC == myCRC && myMsg.Seq == SEQ )
           writeBlock();
        HouseKeep(&ABORTstr);
        return;
    }

    if ( myMsg.Seq != SEQ ) {
         retry++; ErrRec++;
         if  ( retry ) RetryLimit ) {
              ErrCode = 313;  /* Reach retry Limit */
              SendChar(EOT);
              HouseKeep(&EOTstr);
         }
         else {
              ErrCode = 312;  /* incorrect seq of block */
              SendChar(NAK);
              SendChar(SEQ);
              SysBeep(5);
              DisplayTrans(&SEQstr,ErrCode,TransfByte,ErrRec);
              TimeStamp();
         }
         return;
     }
     if  ( RecvCRC != myCRC ) {
          retry++;  ErrRec++;
          if  ( retry ) RetryLimit ) {
               ErrCode = 313;
               SendChar(EOT);
               HouseKeep(&EOTstr);
```

```
        }
        else {
            ErrCode = 314;  /* incorrect CRC of block */
            SendChar(NAK);
            SendChar(SEQ);
            SysBeep(5);
            DisplayTrans(&CRCstr,ErrCode,TransfByte,ErrRec);
            TimeStamp();
        }
        return;
    }
    retry = 0;
    writeBlock();
    if  (ByteCount == EOF ) {  /* there was error in write */
        ErrCode = 302;         /* writing Error */
        SendChar(EOT);         /* abort Operation */
        HouseKeep(&ABORTstr);  /* perform House keeping */
    }
    else {                     /* write successfull */
      SendChar(ACK);           /* acknowledge       */
      SendChar(SEQ);
      DisplayTrans(&ACKstr,100, TransfByte, ErrRec);
      if ( SEQ == MaxSeq )
         SEQ = initSeq;  /* wrap sequence when reach highest */
      else
         SEQ++;
      return;
    }
}

int  SendCmd()
{
unsigned char NextChar;
unsigned int  myCRC;


     if ( GetOpen(SendName) ) {
        if ( RmtRecvName() ) {
           InitSendMode();
           SendChar(ENQ);
           SendChar('R');       /* force another station to receive */
           DisplayTrans(&ENQstr,100,TransfByte,ErrRec);
           GetNext(&NextChar,1);     /* wait for acknowledge */
           if  ( Expired || NextChar != ACK) { /* if Expire or not ACK */
              DisposDialog(FileDlgPtr);  /* clear previous window */
              ErrCode = 102;             /* No response from remote PC */
              CmdAlert(ErrCode);         /* show error Message */
              return;
           }
           OpenInput();
           if (fileRef == EOF ) {       /* cannot open file */
              ErrCode = 201;            /* set error code */
              HouseKeep(&BLANKstr);     /* perform house keeping */
              return;
           }
```

```
            ReadFileNext();                 /* get data block */
            if ( ByteCount > 0 ) {
               DisplayTrans(&STXstr,100,TransfByte,ErrRec);
               SendBlock();
            }
            else {
               if  ( ByteCount == EOF ) {
                   ErrCode = 202;           /* cannot read data */
                   SendChar(DLE);           /* abort operation */
                   SendChar(EOT);
                   HouseKeep(&ABORTstr);
               }
               else {
                   ErrCode = 0;             /* normal end of File */
                   SendChar(EOT);           /* send End of data */
                   HouseKeep(&EOTstr);
               }
            }
         }
      }
   }

int  RecvCmd()
{
     if ( PutOpen("Save Received File As :",RecvName) ) {
        if ( RmtSendName() ) {
           InitRecvMode();
           GetRCOMID();
           if ( Expired ) {
              DisposDialog(FileDlgPtr);     /* clear previous window */
              ErrCode = 102;                /* no response */
              CmdAlert(ErrCode);            /* show Error message */
              return;
           }
           OpenOutput();
           if  (fileRef == EOF )  {         /* cannot open file */
              ErrCode = 301;                /* set error code */
              HouseKeep(&BLANKstr);         /* perform house keeping */
              return;
           }
           SendChar(ENQ);                   /* initiate to send */
           SendChar('S');
           DisplayTrans(&ENQstr,100,TransfByte,ErrRec); /* show status */
           TimeStamp();                     /* time stamp for expired */
           return;
        }
     }
}

int  SendRCOMID()   /* send RCOM ID Version and serial */
{
     long Length;

     Length = 9;
     FSWrite(refOUT,&Length,myRCOMID);
```

```
}

int  GetNext(anybuff,len)  /* get data from serial port */
char anybuff[];
int  len;
{
     long   getByte;
     long   recvByte;
     int    pos;

     Expired = 0;
     pos = 0;
     TimeStamp();
     while ( pos < len) {
       if  (TickCount() > ExpTime ) {
           Expired = 1;
           break;
       }
       SerGetBuf(refIN,&recvByte);
       if  ( recvByte > 0 ) {
         getByte = len - pos;  /* calculate remaining to get data */
         FSRead(refIN,&getByte,&anybuff[pos]);
         pos += getByte;
         TimeStamp();
       }
     }
     return(pos);
}

int  GetNameETB()      /* get filename until End of Text data */
{
     int  count;
     unsigned char namechar;

     count = 0;
     while  (count < sizeof(GetName)-1 ) {
       GetNext(&namechar,1);
       if ( ! Expired )
         if  ( namechar != ETB )
           GetName[count++] = namechar;
         else
           break;              /* caused by End of Text Block */
       else
          break;               /* caused by Time out */
     }
     GetName[count] = '\0';    /* make to C string */
     return(count);
}

int  GetRCOMID()       /* get RCOM ID */
{
     SendChar(ENQ);            /* Enquiry */
     SendChar('N');            /* RCOM ID */
     DisplayTrans(&ENQstr,100,TransfByte,ErrRec);
     GetNext(hisRCOMID,9);     /* ID Len = 9 */
```

```
    hisRCOMID[9] = '\0';    /* make to C string */
}

int  SendBlock()     /* send frame of packet */
{
    long  Length;

    SendChar(STX);          /* Send Start of TEXT */
    Length = myMsg.Len + 3; /* seq + len + ETB */
    FSWrite(refOUT,&Length,&myMsg.Seq); /* send data Block */
    Length = 2L;
    FSWrite(refOUT,&Length,&CRC); /* Plus Calculate CRC */
    DisplayTrans(&WAITstr,100,TransfByte,ErrRec);
    TimeStamp();            /* Time stamp for Expire Time */
}

int OpenInput()    /* open input file by mode */
{
   int  myMode;
   if  (mode == 1 )
        myMode = O_RDONLY + O_BINARY;  /* open file in binary format */
   else
        myMode = O_RDONLY + O_TEXT;    /* open file in text format */
   fileRef = open(SendName,myMode);    /* open Send file */
}

int OpenOutput()   /* open output file by mode */
{
   int  myMode;

   if  (mode == 1 )
        myMode = O_RDWR + O_BINARY;    /* open file in binary format */
   else
        myMode = O_RDWR + O_TEXT;      /* open file in text format */
   fileRef = creat(RecvName,myMode);   /* open Receive file */
}

int ReadFileNext()   /* read next data block */
{
     ByteCount = read(fileRef,myMsg.myBuff,maxsize);
     if  (ByteCount == EOF )
        myMsg.Len = 0;          /* if read error set length to zero */
     else {
        if  ( ByteCount ) 0 ) {
            if ( SEQ == MaxSeq )
                SEQ = initSeq;  /* wrap sequence when reach highest */
            else
                SEQ++;          /* increment sequence */
            myMsg.Seq = SEQ;    /* setup sequence */
            myMsg.Len = ByteCount;      /* setup length */
            myMsg.myBuff[ByteCount] = ETB;  /* setup ETB */
            CRC = GenerateCRC(ByteCount+3,&myMsg.Seq); /* cal CRC */
            TransfByte += ByteCount;   /* Update Read data byte */
        }
        else
```

```
            myMsg.Len = 0;      /* normal endof data */
        }
}

int writeBlock()   /* write block of data to file */
{
     ByteCount = write(fileRef,myMsg.myBuff,myMsg.Len);
     if ( ByteCount != EOF ) {
       TransfByte += ByteCount;
     }
}

int InitSendMode()  /* initial terminal to send mode */
{
    SEQ = MaxSeq;
    termMode = SendMode;
    ErrCode = 0;
    ErrRec  = 0;
    TransfByte = 0;
    TransferDlg = 0;
    Abort = 0;
}

int InitRecvMode()  /* initial terminal to receive mode */
{
    SEQ = initSeq;
    termMode = RecvMode;
    ErrCode = 0;
    ErrRec  = 0;
    TransfByte = 0;
    TransferDlg = 0;
    Abort = 0;
}

int Display(anychar)   /* display any data to screen */
unsigned char anychar;
{
   int  row,col,temp;

       row = getypos();  col = getxpos();

       switch ( EscSeq ) {
       case 0  : switch (anychar) {
                 case 0x09 : if  ( (temp = col / 8 * 8 + 8) > 79 )
                                 gotoxy(col,row);
                             else
                                 gotoxy(temp,row);
                             break;
                 case 0x1B : EscArray[++EscSeq] = anychar;
                             break;
                 case 0x07 :
                 case 0x08 :
                 case 0x0A :
                 case 0x0C :
                 case 0x0D : fprintf(myWindow,"%c",anychar);
```

```
                        break;
            default   : if ( anychar > 0x19 ) {
                           fprintf(myWindow,"%c",anychar);
                           if (row == 24 && col == 79)
                              gotoxy(79,24);
                        }
                        break;
           }
           break;
case 1   : switch ( anychar ) {
            case 0x4A : EscSeq = 0;  /* Clear to End of Screen */
                        CLREOS();
                        break;
            case 0x41 : if (row > 0) {   /* up arrow */
                          gotoxy(col,row-1);
                          EscSeq = 0;
                        }
                        break;
            case 0x42 : if (row < 24) { /* down arrow */
                          gotoxy(col,row+1);
                          EscSeq = 0;
                        }
                        break;
            case 0x43 : if ( col < 79 ) { /* right arrow */
                          gotoxy(col+1,row);
                          EscSeq = 0;
                        }
                        break;
            case 0x44 : if ( col > 0 )  { /* left arrow */
                          gotoxy(col-1,row);
                          EscSeq = 0;
                        }
                        break;
            case 0x53 :
            case 0x59 : EscArray[++EscSeq] = anychar;
                        break;
            case 0x1B : break;  /* do nothing for Esc */
            default   : fprintf(myWindow,"%c",anychar);
                        EscSeq = 0;
                        break;
           }
           break;
case 2 : if ( EscArray[2] == 0x53 ) {  /* set attribute */
             SetAttrib(anychar);
             EscSeq = 0;
           }
           else {
             EscArray[++EscSeq] = anychar;
           }
           break;
case 3 : gotoxy(anychar-0x20,EscArray[3]-0x20);
           EscSeq = 0;
           break;
}
}
```

```
int  CLREOS()  /* clear from current position to end of screen */
{
int   row,col;
register int   clrbyte;

      row = getypos();
      col = getxpos();
      clrbyte = 80 - col;          /* cal first row */
      clrbyte += (24-row)*80;      /* plus any other row */
      for ( ; clrbyte ) 0; clrbyte--)
          fprintf(myWindow,' ');
      gotoxy(col,row);
}



int SetAttrib(anychar)   /* set up attribute of screen */
unsigned anychar;
{
GrafPtr myGraf;
Style  myStyle;
int    myMode;

    myGraf = (GrafPtr)(myWindow->filebuf);
    myStyle = 0;
    myMode = srcOr;
    myGraf->txMode = myMode;
    myGraf->txFace = myStyle;
}
```

```
/*-------------------------------------------
 *  File name : TransfDialog.c
 *  Function  : Display dialog while sending or
 *              receiving data file.
 *-------------------------------------------*/

#include "stdio.h"
#include "Unix.h"
#include "QuickDraw.h"
#include "DialogMgr.h"
#include "MenuMgr.h"
#include "StdFilePkg.h"

#include "Defined.inc"
#include "Externdef.inc"

#define   XFRDlg   129

extern    DialogPtr FileDlgPtr;

extern    FILE *myWindow;

static char *ErrTable[] = {
            "100    ",
            "101 Other Program is not RCOM",
            "102 No Response from Remote computer",
            "103 Sending Remote File Name",
            "104 Could not send Remote file name",
            "105 Sending Dial Characters",
            "201 Could not Open Send file",
            "202 Error while Reading from disk",
            "204 Session was Abort by Operator",
            "205 Disk of receiving computer is full",
            "210 Time out waiting in Send mode",
            "211 Received Abort from Receiving computer",
            "212 Wrong Sequence Acknowledgement",
            "213 Retries exceed in Send mode",
            "301 Could not Open Receive file",
            "302 Error while writing to disk",
            "304 Session was Abort by Operator",
            "310 Time out while in Receive mode",
            "311 Received Abort from Sending Computer",
            "312 Received wrong Sequence Block",
            "313 Reach NAK retry Limit",
            "314 Unmatch CRC in Block Received" };

char  SuccessMsg[] = "\pFile Transfer Complete";
char  DefaultMsg[] = "\pCannot found Message      ";
char  ResultMsg[50];
char  RetryStr[]   = "\PRetries";
char  ErrRecStr[]  = "\pErrors";
char  MODE[]       = "Mode";
char  TEXT[]       = "Text ";
char  BINARY[]     = "Binary ";
```

```
DisplayTrans(stat,Errno,Amount,retry)
/* Display Transfer dialog data */
char *stat;
int  Errno;
long Amount, retry;
{
   char TempStr[40], *TextPtr;
   int  dialogType;
   Handle dialogHdl;
   Rect dialogRect;
   PenState Ps;
   GrafPtr  savePort;
   Rect dummyRect;

   if ( !( (termMode == SendMode) || (termMode == RecvMode ) ))
      return(0);    /* if not transfer not display */


   if ( TransferDlg != 1 ) {
      FileDlgPtr = GetNewDialog(XFRDlg,0L,-1);

      GetDItem(FileDlgPtr,1,&dialogType,&dialogHdl,&dialogRect);
      GetPenState(&Ps);
      dummyRect = dialogRect;
      GetPort(&savePort);
      SetPort(&((WindowPeek)FileDlgPtr)->port);
      PenSize(3,3);
      InsetRect(&dummyRect,-4,-4);
      FrameRoundRect(&dummyRect,16,16);
      SetPort(savePort);
      SetPenState(&Ps);

      TransferDlg = 1;
      if ( termMode == SendMode ) {
           strcpy(TempStr,Sendstr);
           strcat(TempStr,SendName);
      }
      else {
           strcpy(TempStr,Recvstr);
           strcat(TempStr,RecvName);
      }
      CtoPstr(TempStr);
      GetDItem(FileDlgPtr,2,&dialogType,&dialogHdl,&dialogRect);
      SetIText(dialogHdl,TempStr);

      if ( mode == 1 )
         strcpy(TempStr,BINARY); /* binary File Transfer */
      else
         strcpy(TempStr,TEXT);   /* Text   File Transfer */
      strcat(TempStr,MODE);
      CtoPstr(TempStr);

      GetDItem(FileDlgPtr,3,&dialogType,&dialogHdl,&dialogRect);
      SetIText(dialogHdl,"\pTransfer ");
```

```
    GetDItem(FileDlgPtr,5,&dialogType,&dialogHdl,&dialogRect);
    SetIText(dialogHdl,TempStr);

    GetDItem(FileDlgPtr,8,&dialogType,&dialogHdl,&dialogRect);
    if ( termMode == SendMode )
       SetIText(dialogHdl,RetryStr);
    else
       SetIText(dialogHdl,ErrRecStr);
  }

    NumToString(Amount,TempStr);
    GetDItem(FileDlgPtr,4,&dialogType,&dialogHdl,&dialogRect);
    SetIText(dialogHdl,TempStr);

    GetDItem(FileDlgPtr,6,&dialogType,&dialogHdl,&dialogRect);
    SetIText(dialogHdl,stat);

    Search(Errno,&TextPtr);
    GetDItem(FileDlgPtr,7,&dialogType,&dialogHdl,&dialogRect);
    SetIText(dialogHdl,TextPtr);

    NumToString(ErrRec,TempStr);
    GetDItem(FileDlgPtr,9,&dialogType,&dialogHdl,&dialogRect);
    SetIText(dialogHdl,TempStr);
}


char *copyMess(anyPtr)   /* Copy message */
char *anyPtr;
{
    strcpy(ResultMsg,anyPtr+3);
    CtoPstr(ResultMsg);
    return(ResultMsg);
}

Search(ErrNo,TextPtr)    /* Search message Code by binary search */
int  ErrNo;
char **TextPtr;
{
    int  first, last, mid;
    int  cpResult;
    long temp;
    char anyStr[10];

    if ( ErrNo == 0 ) {
       *TextPtr = SuccessMsg;
       return;
    }

    temp = ErrNo;
    NumToString(temp,anyStr);
    PtoCstr(anyStr);

    first = 0;
```

```
    last = sizeof(ErrTable) / sizeof(char *)  -1 ;
    do  {
       mid = ( first + last )/2;
       cpResult = strncmp( ErrTable[mid],anyStr, 3 ) ;
       if ( cpResult == 0 ) {
          *TextPtr = copyMess(ErrTable[mid]);
          return;
       }
       if ( cpResult < 0 )
          first = mid +1;
       else
          last = mid - 1;
    }
    while ( first <= last );
    strncat(&DefaultMsg[sizeof(DefaultMsg)-4],anyStr,3);
    *TextPtr = DefaultMsg;
}
```

```
/*-------------------------------------------
 * File name : Serial.c
 * Function  : Setting serial port and generate
 *             CRC of message.
 *-------------------------------------------*/

#include "SerialDvr.h"

static unsigned crc_table[256];


SetSerial(refno, stop, parity, data, baud)
/*  set up serial port */
int  refno, stop, parity, data, baud;
{
   int  serconfg;

   switch ( stop ) {
   case  1: serconfg = stop10;
            break;
   case  2: serconfg = stop20;
            break;
   case 15: serconfg = stop15;
            break;
   default : serconfg = stop10;
   };

   switch ( parity ) {
   case  0: serconfg += noParity;
            break;
   case  1: serconfg += oddParity;
            break;
   case  2: serconfg += evenParity;
   };

   switch ( data ) {
   case  5: serconfg += data5;
            break;
   case  6: serconfg += data6;
            break;
   case  7: serconfg += data7;
            break;
   case  8: serconfg += data8;
   };

   switch ( baud ) {
   case  150: serconfg = baud300 * 2;
              break;
   case  300: serconfg += baud300;
              break;
   case  600: serconfg += baud600;
              break;
   case 1200: serconfg += baud1200;
              break;
   case 1800: serconfg += baud1800;
```

```
            break;
   case 2400: serconfg += baud2400;
            break;
   case 3600: serconfg += baud3600;
            break;
   case 4800: serconfg += baud4800;
            break;
   case 7200: serconfg += baud7200;
            break;
   case 9600: serconfg += baud9600;
   };
   return(SerReset(refno, serconfg));
 }

OpenSerial(DrvName,refno)  /* open serial port */
int   *refno;
char  *DrvName;
{
    return( OpenDriver(DrvName,refno) );
}


void GenerateTable()
/*   Generate CRC table for lookup process */
{
  union { int i;
        struct {
                 unsigned    :8; /* unused        */
                 unsigned i8 :1; /* high order bit */
                 unsigned i7 :1;
                 unsigned i6 :1;
                 unsigned i5 :1;
                 unsigned i4 :1;
                 unsigned i3 :1;
                 unsigned i2 :1;
                 unsigned i1 :1; /* low order bit */
                } Bit;
      } iUn;

 union {  unsigned int Entry;
          struct {
                 unsigned b16:1; /* high order bit */
                 unsigned b15:1;
                 unsigned b14:1;
                 unsigned b13:1;
                 unsigned b12:1;
                 unsigned b11:1;
                 unsigned b10:1;
                 unsigned b9 :1;
                 unsigned b8 :1;
                 unsigned b7 :1;
                 unsigned b6 :1;
                 unsigned b5 :1;
                 unsigned b4 :1;
                 unsigned b3 :1;
```

```
                unsigned b2 :1;
                unsigned b1 :1; /* low order bit */
               } EntryBit;
        } EntryUn;


  for (iUn.i = 0; iUn.i < 256 ; iUn.i++ ) {

      EntryUn.Entry = 0; /* all bits are zeroed out now */

      EntryUn.EntryBit.b16 = (iUn.Bit.i4 ^ iUn.Bit.i8);
      EntryUn.EntryBit.b15 = (iUn.Bit.i3 ^ iUn.Bit.i7);
      EntryUn.EntryBit.b14 = (iUn.Bit.i2 ^ iUn.Bit.i6);
      EntryUn.EntryBit.b13 = (iUn.Bit.i1 ^ iUn.Bit.i5);
      EntryUn.EntryBit.b9  =  iUn.Bit.i8;
      EntryUn.EntryBit.b8  = (iUn.Bit.i7 ^ iUn.Bit.i8);
      EntryUn.EntryBit.b7  = (iUn.Bit.i6 ^ iUn.Bit.i7);
      EntryUn.EntryBit.b6  = (iUn.Bit.i5 ^ iUn.Bit.i6);
      EntryUn.EntryBit.b5  = (EntryUn.EntryBit.b16 ^ iUn.Bit.i5);
      EntryUn.EntryBit.b4  = (EntryUn.EntryBit.b15 ^ EntryUn.EntryBit.b16);
      EntryUn.EntryBit.b3  = (EntryUn.EntryBit.b14 ^ EntryUn.EntryBit.b15);
      EntryUn.EntryBit.b2  = (EntryUn.EntryBit.b13 ^ EntryUn.EntryBit.b14);
      EntryUn.EntryBit.b1  =  EntryUn.EntryBit.b13;

      crc_table[iUn.i] = EntryUn.Entry;
   }
}

unsigned GenerateCRC(Length, TextPtr)
unsigned Length;
register unsigned char *TextPtr;
/*  Generate CRC of Message */

{
  register int  i;
  register unsigned crc;
  int index;

  crc = 0xffff;
  for ( i = 0; i < Length; i++, TextPtr++) {
     index = (crc >> 8) ^ *TextPtr;
     crc  =  (crc << 8) ^ crc_table[index];
  }
  return(crc);
}
```

```
/*-------------------------------------------------------------
 *  File name : UserKBD.c
 *  Function  : Test for user press key by any combination of
 *              Command key, Option key and other key
 *              Process key as PF, PA, cursor movement or any
 *              key of 3270 then send result to HYDRA II
 *-------------------------------------------------------------*/

#include  "stdio.h"
#include  "EventMgr.h"
#include  "WindowMgr.h"
#include  "MenuMgr.h"
#include  "STDfilePkg.h"
#include  "ToolBoxUtil.h"

#include  "defined.inc"
#include  "ExternDef.inc"

extern  FILE *myWindow;
extern  EventRecord  myEvent;


int  KeyProcess()  /* Process User keyboard Event */
{
   if (myEvent.modifiers & cmdKey)
       CmdKeyProc();    /* if press Command key + any */
   else {
      if (myEvent.modifiers & optionKey)
         OptionProc();  /* if press Option key + any */
      else
         NormalProc();  /* if none of above */
   }
   return(1);
}


int CmdKeyProc()   /* process special key meaning except PF key */
{
   unsigned int ScanCode;

   ScanCode = (myEvent.message & keyCodeMask) )) 8;
   switch  ( ScanCode ) {
   case 35 :  ESCchar(0X1B);          /* PA1 */
              break;
   case 33 :  ESCchar(0X56);          /* PA2 */
              break;
   case 30 :  ESCchar(0X57);          /* PA3 */
              break;
   case 17 :  ESCchar(0X45);          /* Test */
              break;
   case  8 :  ESCchar(0X4A);          /* Clear */
              break;
   case 15 :  ESCchar(0X30);          /* Reset */
              break;
   case 02 :  ESCchar(0X7F);          /* Del */
```

```
                break;
    case 34 :   ESCchar(0X50);          /* Ins */
                break;
    case 03 :   ESCchar(0X49);          /* EOF */
                break;
    case 14 :   SendChar(0X05);         /* Erase Input */
                break;
    case 04 :   ESCchar(0X5A);          /* Home */
                break;
    case 31 :   ESCchar(0X58);          /* Home 2 */
                break;
    case 16 :   SendChar(0x19);         /* Eneter Diag Mode */
                break;
    case 00 :   ESCchar(0X44);          /* Left */
                break;
    case 01 :   ESCchar(0X43);          /* Right */
                break;
    case 13 :   ESCchar(0X41);          /* Up */
                break;
    case 06 :   ESCchar(0X42);          /* Down */
                break;
    }
    return(1) ;
}


int OptionProc()    /* Process PF Key */
{
    unsigned int ScanCode;
    unsigned int shift;

    ScanCode = (myEvent.message & keyCodeMask) )) 8;
    shift    = myEvent.modifiers & shiftKey;

    switch ( ScanCode ) {
    case 18 :
    case 71 : if (!shift)
                 ESCchar(0X61);  /*  PF1   */
              else
                 ESCchar(0X33);  /*  PF13  */
              break;
    case 19 :
    case 81 : if (!shift)
                 ESCchar(0X62);  /*  PF2   */
              else
                 ESCchar(0X34);  /*  PF14  */
              break;
    case 20 :
    case 75 : if (!shift)
                 ESCchar(0X63);  /*  PF3   */
              else
                 ESCchar(0X35);  /*  PF15  */
              break;
    case 21 :
    case 89 : if (!shift)
```

```
                ESCchar(0X64);  /*  PF4   */
            else
                ESCchar(0X36);  /*  PF16  */
            break;
    case 23 :
    case 91 : if (!shift)
                ESCchar(0X65);  /*  PF5   */
            else
                ESCchar(0X37);  /*  PF17  */
            break;
    case 22 :
    case 92 : if (!shift)
                ESCchar(0X66);  /*  PF6   */
            else
                ESCchar(0X38);  /*  PF18  */
            break;
    case 26 :
    case 86 : if (!shift)
                ESCchar(0X67);  /*  PF7   */
            else
                ESCchar(0X39);  /*  PF19  */
            break;
    case 28 :
    case 87 : if (!shift)
                ESCchar(0X68);  /*  PF8   */
            else
                ESCchar(0X3A);  /*  PF20  */
            break;
    case 25 :
    case 88 : if (!shift)
                ESCchar(0X69);  /*  PF9   */
            else
                SendChar(0X89); /*  PF21  */
            break;
    case 29 :
    case 83 : if (!shift)
                ESCchar(0X6A);  /*  PF10  */
            else
                SendChar(0X8A); /*  PF22  */
            break;
    case 27 :
    case 84 : if (!shift)
                ESCchar(0X31);  /*  PF11  */
            else
                SendChar(0X8B); /*  PF23  */
            break;
    case 24 :
    case 85 : if (!shift)
                ESCchar(0X32);  /*  PF12  */
            else
                SendChar(0X8C); /*  PF24  */
            break;
    }
    return(1);
}
```

```
int NormalProc()  /* User press only single key  */
{
   unsigned int ScanCode;
   unsigned int shift;
   unsigned char thechar;

   thechar  = myEvent.message & charCodeMask;
   ScanCode = (myEvent.message & keyCodeMask) >> 8;
   shift    = myEvent.modifiers & shiftKey;

   switch (ScanCode) {
   case 52 :
   case 76 : SendChar(0x0d);  /* Enter */
             break;
   case 36 : SendChar(0x0E);  /* NewLine */
             break;
   case 71 : ESCchar(0x4A);   /* Clear */
             break;
   case 51 : ESCchar(0x7F);   /* Delete */
             break;
   case 48 : if (!shift)
                SendChar(0x09); /* Tab */
             else
                ESCchar(0x47); /* BackTab */
             break;
   case 123: ESCchar(0x44);   /* left arrow */
              break;
   case 124: ESCchar(0x43);   /* right arrow */
              break;
   case 125: ESCchar(0x42);   /* Down arrow */
              break;
   case 126: ESCchar(0x41);   /* up arrow */
              break;
   default : SendChar(thechar);
   }
   return(1);
}

int ESCchar(chseq)     /* send Esc plus any character */
unsigned char chseq;
{
   char sendBuff[2];
   long sendByte = 2L;

   sendBuff[0] = 0x1B; /* ESC */
   sendBuff[1] = chseq;

   FSWrite( refOUT, &sendByte, &sendBuff);
}
```

```
*  File name : Hydra.R
*  Function  : Resource input for Resource Compiler
*              to produce Hydra.rsrc resource file.
*              This resource is used by 3270 Emulation program.
*  Written by  Wiwat Kasempannarai

Hydra.Rsrc              ;; output file
????????                ;; file type and creator


Type MENU
  ,127                  ;; apple id
  \14                   ;; apple symbol


  ,128                  ;; menu id
 Option                 ;; first menu title
  Quit                  ;; first menu item


  ,129                  ;; menu id
 Setup                  ;; second menu title
Parameter File          ;; parameter setup
Type for Binary File    ;; set up file type
(-                      ;; line break
Local Send File         ;; Set local send file
Local Receive File      ;; Set local receive file
(-                      ;; line break
Remote Send  File       ;; set remote Send Filename
Remote Receive File     ;; set remote Receive Filename
(-                      ;; line break
Phone Number            ;; Phone Number


  ,130                  ;; menu id
 Command                ;; second menu title
Sending File            ;; Send file to remote receive
Receiving File          ;; Receive file from remote sender
(-                      ;; line break
Dialing                 ;; dial to remote


Type WIND
  ,128                  ;; Logo window
 File Transfer          ;; Title for file transfer
 60 100 200 412         ;; the Window  Rectangle
 Visible GoAway
 0                      ;; Window Definition ID
 0                      ;; RefCon


Type DITL
  ,128                  ;; Resource ID
  39                    ;; number of items in the item list
```

```
 button                  ;; enable Button
90 410 110 470           ;; rectangle
OK                       ;; Text in button


 button                  ;; enable Button
145 410 165 470          ;; rectangle
Cancel                   ;; Text in button


 staticText              ;; static text item
10 10 30 90              ;; Rectangle of Text
Baud Rate                ;; the text


 radioButton             ;; radio button enable
30 10 50 70              ;; Rectangle of Text
150                      ;; The text


 radioButton             ;; radio button enable
50 10 70 70              ;; Rectangle of Text
300                      ;; The text


 radioButton             ;; radio button enable
70 10 90 70              ;; Rectangle of Text
600                      ;; The text


 radioButton             ;; radio button enable
90 10 110 70             ;; Rectangle of Text
1200                     ;; The text


 radioButton             ;; radio button enable
110 10 130 70            ;; Rectangle of Text
2400                     ;; The text


 radioButton             ;; radio button enable
130 10 150 70            ;; Rectangle of Text
4800                     ;; The text


 radioButton             ;; radio button enable
150 10 170 70            ;; Rectangle of Text
9600                     ;; The text


 staticText              ;; static text item
190 10 210 90            ;; Rectangle of Text
Data Bits                ;; the text
```

```
 radioButton            ;; radio button enable
210 10 230 80           ;; Rectangle of Text
7 Bits                  ;; The text


 radioButton            ;; radio button enable
230 10 250 80           ;; Rectangle of Text
8 Bits                  ;; The text


 staticText             ;; static text item
10 105 30 175           ;; Rectangle of Text
Parity                  ;; the text


 radioButton            ;; radio button enable
30 105 50 175           ;; Rectangle of Text
NONE                    ;; The text


 radioButton            ;; radio button enable
50 105 70 175           ;; Rectangle of Text
Odd                     ;; The text


 radioButton            ;; radio button enable
70 105 90 175           ;; Rectangle of Text
Even                    ;; The text


 staticText             ;; static text item
90 105 110 175          ;; Rectangle of Text
Stop Bits               ;; the text


 radioButton            ;; radio button enable
110 105 130 175         ;; Rectangle of Text
1 Bits                  ;; The text


 radioButton            ;; radio button enable
130 105 150 175           ;; Rectangle of Text
1.5 Bits                ;; The text


 radioButton            ;; radio button enable
150 105 170 175           ;; Rectangle of Text
2 Bits                  ;; The text


 staticText             ;; static text item
190 105 210 175         ;; Rectangle of Text
Mode                    ;; the text
```

```
 radioButton            ;; radio button enable
210 105 230 175         ;; Rectangle of Text
Binary                  ;; The text


 radioButton            ;; radio button enable
230 105 250 175         ;; Rectangle of Text
Text                    ;; The text


 staticText             ;; static text item
190 192 210 242          ;; Rectangle of Text
Port                    ;; the text


 radioButton            ;; radio button enable
210 192 230 262         ;; Rectangle of Text
Modem                   ;; The text


 radioButton            ;; radio button enable
230 192 250 262         ;; Rectangle of Text
Printer                 ;; The text


 staticText             ;; static text item
10 192 30 280           ;; Rectangle of Text
Send From :             ;; the text


 editText               ;; edit text
35 195 51 395           ;; rectangle


 staticText             ;; static text item
65 192 85 280           ;; Rectangle of Text
Receive As :            ;; the text


 editText               ;; edit text
90 195 106 395          ;; rectangle


 staticText             ;; static text item
120 192 140 251         ;; Rectangle of Text
Phone                   ;; the text


 editText               ;; edit text
145 195 161 395         ;; rectangle


 staticText             ;; static text item
190 280 210 407         ;; Rectangle of Text
```

```
Type of Binary File   ;; the text


 editText             ;; static text item
191 420 208 455       ;; Rectangle of Text


 radioButton          ;; radio button enable
210 280 230 363       ;; Rectangle of Text
MacPaint              ;; The text


 radioButton          ;; radio button enable
230 280 250 363       ;; Rectangle of Text
MacDraw               ;; The text


 radioButton          ;; radio button enable
210 370 230 476       ;; Rectangle of Text
SuperPaint            ;; The text


 radioButton          ;; radio button enable
230 370 250 476       ;; Rectangle of Text
User Defined          ;; The text


Type DLOG
  ,128                ;; resource id of Dialog template
  Setup Parameter     ;; Title of dialog
  60 15 320 500       ;; the Rectange (Top, Left, Bottom, Right)
  Visible  NoGoAway   ;; may be Invisible, and may be GoAway
  0                   ;; Dialog definition ID
  0                   ;; RefCon
  128                 ;; Resource ID for dialog item list


Type DLOG
  ,129                ;; resource id of Dialog template
  File Transfer       ;; Title of dialog
  60 55 240 460       ;; the Rectange (Top, Left, Bottom, Right)
  Visible  NoGoAway   ;; may be Invisible, and may be GoAway
  0                   ;; Dialog definition ID
  0                   ;; RefCon
  129                 ;; Resource ID for dialog item list


Type DITL
  ,129                ;; Resource ID
  9                   ;; number of items in the item list


  button              ;; enable Button
 133 238 153 299      ;; rectangle
 Stop                 ;; Text in button
```

```
 staticText             ;; static text item
15 20 35 383            ;; Rectangle of Text send / recv


 staticText             ;; static text item
55 20 75 80             ;; Rectangle of Text status


 staticText             ;; static text item
55 100 75 214           ;; Rectangle of Text error message


 staticText             ;; static text item
55 234 75 328           ;; Rectangle of Text transfer Byte
                        ;; Text in rectangle


 staticText             ;; static text item
94 19 114 79            ;; Rectangle of Text number of bytes


 staticText             ;; static text item
94 99 114 386           ;; Rectangle of Text


 staticText             ;; static text item
134 20 154 80           ;; Rectangle of Text


 staticText             ;; static text item
134 100 153 213         ;; Rectangle of Text


Type DLOG
  ,130                  ;; resource id of Dialog template
  Setup                 ;; Title of dialog
  60 85 220 430         ;; the Rectange (Top, Left, Bottom, Right)
  Visible  NoGoAway     ;; may be Invisible, and may be GoAway
  0                     ;; Dialog definition ID
  0                     ;; RefCon
  130                   ;; Resource ID for dialog item list


Type DITL
  ,130                  ;; Resource ID
  4                     ;; number of items in the item list


  button                ;; enable Button
 109 29 132 90          ;; rectangle
 OK                     ;; Text in button
```

```
 button                ;; enable Button
109 107 132 168        ;; rectangle
Cancel                 ;; Text in button


 staticText            ;; static text item
20 27 40 318           ;; Rectangle of Text send / recv


 editText              ;; static text item
60 30 76 315           ;; Rectangle of Text status


Type ALRT
 ,131                  ;; resource id of Alert template
 66 80 210 420         ;; the Rectange (Top, Left, Bottom, Right)
 131                   ;; Resource ID for dialog item list
 7777                  ;; Stage word in hex

Type  DITL
 ,131                  ;; Resource ID
 3                     ;; number of items in the item list


 button                ;; enable Button
104 16 128  85         ;; rectangle
OK                     ;; Text in button


 staticText            ;; static text item
22 16 48 321           ;; Rectangle of Text send / recv
^0                     ;; Static text


 staticText            ;; static text item
59 16 85 321           ;; Rectangle of Text status
^1                     ;; Static text


Type DITL
 ,132                  ;; Resource Id for dialog item list
 2                     ;; Number of item in the item list


staticText             ;; Static text item
26 22 50 297           ;; Rectangle of text


editText               ;; edit text
65 22 81 295           ;; rectangle


Type DLOG
 ,132                  ;; resource id of Dialog template
Setup Remote File Name ;; Title of dialog
```

```
  74 100 194 420      ;; the Rectange (Top, Left, Bottom, Right)
  Visible  NoGoAway   ;; may be Invisible, and may be GoAway
  0                   ;; Dialog definition ID
  0                   ;; RefCon
  132                 ;; Resource ID for dialog item list


Type DITL
  ,133                ;; Resource Id for dialog item list
  8                   ;; Number of item in the item list


  button              ;; enable Button
  106 63 129 124      ;; rectangle
  OK                  ;; Text in button


  button              ;; enable Button
  106 178 129 239     ;; rectangle
  Cancel              ;; Text in button


  staticText          ;; static text item
  15 40 35 175        ;; Rectangle of Text
Type of Binary File   ;; the text


  editText            ;; static text item
  17 205 34 240       ;; Rectangle of Text


  radioButton         ;; radio button enable
  50 40 70 123        ;; Rectangle of Text
  MacPaint            ;; The text


  radioButton         ;; radio button enable
  70 40 90 123        ;; Rectangle of Text
  MacDraw             ;; The text


  radioButton         ;; radio button enable
  50 155 70 261       ;; Rectangle of Text
  SuperPaint          ;; The text


  radioButton         ;; radio button enable
  70 155 90 261       ;; Rectangle of Text
  User Defined        ;; The text


Type DLOG
  ,133                ;; resource id of Dialog template
  Setup FileType      ;; Title of dialog
  85 105 247 405      ;; the Rectange (Top, Left, Bottom, Right)
```

```
Visible  NoGoAway    ;; may be Invisible, and may be GoAway
0                    ;; Dialog definition ID
0                    ;; RefCon
133                  ;; Resource ID for dialog item list
```

```
*  File name: Combine.r
*  Function : Combine Hydra resource file with 3270 Emulator
*             to get only one Application file

!EM3270appl
????????


include hydra.rsrc
```

ประวัติผู้เขียน

นายวิวัฒน์ เกษมพรรณราย เกิดเมื่อวันที่ 21 ตุลาคม 2504 ที่กรุงเทพ สำเร็จการศึกษาปริญญาตรีจากจุฬาลงกรณ์มหาวิทยาลัย คณะวิทยาศาสตร์ ภาควิชาเคมีเทคนิค สาขาเคมีวิศวกรรม เมื่อปีการศึกษา 2525 เข้าศึกษาต่อในระดับปริญญาโท สาขาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2527