北海道大学工学部研究報告
第141号（昭和63年）

Bulletin of the Faculty of Engineering,
Hokkaido University. No. 141 (1988)

An Implementation of Personal Computer Network Using a Public Telephone Line Network System

Ban GUO, Yoshinao AOKI and Peikai ZENG

(Received March 31, 1988)

Abstract

This paper describes the design and implementation of a local network between the same level of personal computers through a universal modem equipment linked to public telephone lines. The software of the Personal Computer Communication System (PCCS) is written in MS-C for the efficiency and convenience of running different computers. Now, the test system is run on a general purpose personal computer of PC -9801.

The system provides almost all of the commands provided in typical MS-DOS operating systems and allows users to use them for operating local computers and the remote computers linked through public telephone lines. The program facilitates the users to render access to remote hardware and software resources between the same level personal computers, instead of having access to a high level host computer. Using the system, the user can easily control their application programs within remote machines and obtain results in the same way as using neighboring ones. Certainly, it also provides those functions supported in commercial communication systems such as autodialing, data or file transfer and mail messages.

Also, the design of automatic main power on-off circuit for a computer system, the procedure of display data transfer by X parameter control and the control of modem equipment on telephone line are discussed.

1. Introduction

Due to the growing demand for efficient utilization and resource sharings of computers distributed over a large geographic area, computer communication becomes increasingly important. For large computers, there are certain types of Local Area Network which are special computer networks with high rate data transfer and diameter of a few kilometers. Most often they are used in office automation. Currently two types are most popular: carrier sense and ring networks. However, it is necessary for these host computers to be connected to the network through physical data link controllers linked by special purpose cables. Thus hundreds of users can share the resources of the host computer, by running their application programs and access results interactively through remote terminals far from the same computer.

With the recent and rapid developments of hardware and software of personal computers, many communication programs are available which enable a personal computer to

Department of Information Engineering

play the role of operation terminal or a data transmitting terminal. They are called emulator programs. Some of them are using special purpose cables and some use public telephone lines. But their data flows are limited to the transfers between lower level computers and host computer or databese manipulator. Recently, although some application programs can be used to connect personal computers of the same level through public telephone lines, the supported functions are only table looking, text file transfer and network parameter managements. The major problem for these communication programs is that they do not support the remote operations and program control on the computers at the other end of the network.

While the cost of personal computers are being reduced, their performances are also approaching perfection. We can find them everywhere, in places such universities, small laboratories and even ordinary households. If a personal computer user can connect his machines with a same level one far away but on available through public telephone lines, and if he can control remote machine operations just as a face-to-face desk top computer, the spare time resources of computer hardware and software located over a large geographic area can be fully utilized by simply making a telephone call. This is our goal of developing a Personal Computer Communication System.

Our network program provides users with an operating environment such as the usual MS-DOS systems, so they can use any typical system command and execute application programs. Also, a programming environment can be easily created on local computer, regardless of whether the physical machine is connected to others through communication lines. Interactive operations can be executed on both local and remote personal computers immediately. The two set of commands exactly the same on typical MS-DOS can be used on the system, which are distinguished only by a communication mark after. The commands with the mark are considered as remote commands to be sent to and executed by remote computers connected on other end of the telephone line, those without the mark are considered as local commands to be executed on local computer. This job is completed by a command analysis processor. The other major module of the system is a data transfer processor which manipulates the data on terminal computers to be transferred and those on the line. The transfer data include operating commands sent to remote computers, execution commands or running application programs on a remote computer, resulting in messages of remotely executed programs, text files and execution programs.

In addition to normal purposes for typical network software, we also design a small circuit which switches off or on the power supply of computer itself. The circuit is controlled by signals from both modem interface and computer. With the circuit, we can shut down the main power of a computer system and leave only the modem's power on when we leave. It operates when being rung up by other computers to turn on the power supply and automatically enter the network system to connect the telephone line. It can also turn off the power when we close the line. For the case of executing programs that require a long time, the circuit can maintains the power on even when the telephone line has been closed.

For the purpose, a signal is sent out from the computer system to control the small power switching unit.

2. The program main routine

The PCCS program can be conceptually divided into three parts: command processor, data transferring manipulator and interface controller. Every part includes many sub-routines of performing sub-functions. They are working around a main routine. The main routine is illustrated in figure 1. The programming implementation is shown in appendix 1.

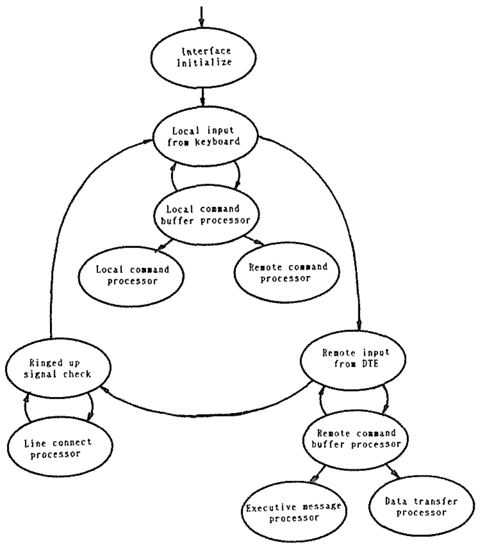

Figure 1. The diagram of main routine.

When entering the system, it is necessary to initialize the DTE interface of personal computer and the communication modem. The initialization module implements the functions of interface error test, DTE state set and check, modem working environment set, modem answer message check and setting flag registers according to the initial state of the interface connecting loops.

The main loop consists of three procedures: local keyboard input process, remote line input process and called up process. The first procedure controls character input from local keyboard, that continuously checks KB status and manipulates the local command buffer. If the input is a break character, the system returns the control to normal MS-DOS system. If it is an end mark of communication, the system sends a close message to the other end side of the telephone line and closes down communication line. If not either, the buffered string is a command which may be either local or remote. We simply separate them by a distinguishing mark. And we process them by using two command processors separately, these are referred to as local command processor and remote command processor.

The second procedure controls the remote input from DTE interface, say from the remote computer system. This includes a DTE check routine and a remote command buffer manipulator. It connects to two processors: data block remote transfer processor and executive message transmitter. The remote transfer processor controls the data flow of both text file copy and executive program copy. The message transmitter implements the loading and sending of a system command message and application program executive message including either error message or result data.

The third one is a controller for local modem through DTE interface when it is signalled (rung up) by remote computers. This procedure consists of a ring signal check routine and a line connection routine. The ring from telephone line acts as an interrupt signal that leads the system to stop the current execution to process the line connection. Within the loop, we incorporate a state display routine of indicating current states of no modem, ringing up or rung up, telephone line opened and data tranferred respectively in the lower middle location of the local computer display. This helps the user to understand his operations and system state at all times to avoid operational mistakes.

3. Remote command processor

It can be said that a typical MS-DOS system command set is well known to users of personal computers. Our system allows the use of these commands on both local and remote computers. We distinguish them with a communication mark, that is to say, a command line with a mark is decided to be executed by a remote machine, while a command line without a mark is considered as local command. Naturally, we perform error checks of these command lines in the analysis, transfer and the receipt processes. For example, it is known the command DIR for listing file informations in a directory. In this case, if we use DIR only, it means to see the directory currently logged in on local computer, if we use DIR !, it means to see that logged in on remote computer. The others of MS-DOS command set can be utilized in the same way.

Now, we describe the procedure for remote command, the programming is shown in appendix 2. In fact, this includes two basic process: one is for sending commands to a remote terminal connected to the line, another is for receiving commands from a remote terminal. The remote command lines for transfer are inputs from local keyboards. First they are checked according to the rule provided for remote commands. When the telephone line is not opened yet, we should call the subroutines for checking and opening of the line. If the opening fails, all of typed remote commands from that time are ignored, resulting in the system accepting only local commands. If successful, the system can send this command to a remote terminal. But these commands should be one of the three types: one term command such as DIR or CL for application program, multiple term commands such as "TYPE ABC. DOC" etc. and data set transfer command such as "COPY ABC. DOC ! ABC1. DOC" etc. The formal two types are sent to remote terminals simply and processed by receipt programs. As shown in the appendix program, in the case of data set transfer command, the system reads the communication mark and decides that the term with a mark is a file name located in remote terminal DOS system. Then we send it with a flag to instruct remote terminal to open the file and transfer them to our terminal. The term without a communication mark is considered as a file name which should be created in the DOS system at the local terminal. For the flexibility of transfer any types of DOS files, the system does not use any ending mark for file receipt. A delay detecting routine is used for deciding file transfer end. The data set transfer command can transfer all kinds of data blocks in DOS system such as text files, resulting data output by user program and executive programs. To distinguish with normal communication software capable of transferring only text files, we call it data set transfer processor.

The receipt processor for remote commands takes over the command line from the system main routine and cuts out the communication mark. If its first term is a data copy flag, it submits the job to data set transfer routine. If it is not so, it creates a temporary file for storing executive resultant messages and then transfers this file through DTE to terminal from which the command is passed. So the user who executes his program or command on remote machines can obtain resultant messages as he submits the same job to the local terminal. Also, the temporary file transfer allows break off by use of an X control parameter. The data transfer and temporary file transfer procedures will be discussed in the next section.

4. Data transferring manipulator

As mentioned above, the system provides a remote copy command for transferring of data block which is similar to an ordinary local file copy command. For example, the command line of "COPY A ! B" copies disk file A on remote computer to local computer as disk file B. While the command "COPY A B !" means transfer of local file A to remote file B. When we use the command to copy remote file to local, the file name to read is sent to remote terminal and starts a remote data file transfer routine which reads and sends its contents till the end of that file. And then it starts a file receipt routine to load data coming from DTE interface of a local computer. For the case of transferring data file to a remote terminal, we have almost the same procedure. For correct receipt of data block including any types of DOS files described in the last section, the file receipt routine does not use any end mark for ending file loading from DTE. Here, we develop a detect module for DTE status. During data loading, we always check its status before inputing a code, when there is no flag set indicating data ready after a finite time of status asking, we assume that it has come to the end of transfer data block. This solves the problems of falling into a dead loop of waiting data from DTE when the file read error at any side for correct receipt of any type of files.

Another batch data transfer is referred to as the access of a temporary file. Because the file consists of only resulting data or executive messages, it is text file. In this case, the system attaches an end mark to indicate the end transfer file. When a remote system finishes a specified operation, it collects executive messages to create a temporary file. And then the file is sent to a local computer through the connected line. The loading routine on local computer receives the code to display them on local terminal resulting in interactive operation environment. In the process of receiving and displaying of the file, an X parameter control routine is designed. When we press Cntl-F key during display of the file on local terminal, the routine sends a pause code to the remote computer to suspend the file output process and meanwhile suspends the local display. When the key is pressed again, the transfer and display procedure is resumed. The data for transfer can be the full range of the characters allowed on a typical personal computer terminals which includes ASCII and Japanese characters.

5. Interface controller

The modem is connected to the DTE interface, for instance RS232C series connecter, of the personal computer. Computer system controls the modem and trasfers data and commands through the interface. Thus both the DTE and the modem serve as an interface of the computer system. In the program, we include the interface controller of manipulating DTE transmitter and modem itself.

The controller of DTE interface completes the functions of defining data transfer rate, setting working parameters, allocating input data buffer and initializing working status of a series I/O chip connecting to system DTE, When the jobs are performed successfully, it will set a true flag for the DTE. When it fails the flag is false.

The modem controller consists of modules for initializing modem, testing modem return code, receiving answer messages from the modem, opening and closing the telephone line, displaying modem working message and communication state marks. The programming of this module is shown in appendix 3. Every command with the communication mark can open the telephone line. It will be closed when input command is only the mark. The module of initializing modem defines the working style by sending modem commnds, setting parameters to its registers and monitoring its working condition at all times. To avoid falling into a dead loop of waiting modem answer message when difficulties arise, the module for receipt is composed of a finite loop which detects the DTE status. The module for displaying communication state sets a mark on the lower middle of the local screen. When no modem is connected or it encounters some difficulties for either DTE or modem, no mark is displayed, otherwise,a skyblue block is displayed. When making a call or being rung up. (called) , the mark become winks on or off, until the party is on the line. When data are transferred on the connected line between terminal computers, the mark will become yellow with blinking. Thus the user knows his communication line at all times by monitoring the state mark.

6. Description for modem equipment and power switch

The modem equipment used in our communication system is EPSON personal medem SR-120AT. It is designed for personal data communication and can satisfy portable and economical demands. The modem can perform high data transfer in full duplex and automatically select the transfer rate in 300bps or 1200bps from the carrier caught on telephone line. It provides two communication ways corresponding to Bell and CCITT standard. Furthermore, with the support of internal Network Control Unit, it can accomplish auto-dialing, auto-answering, auto-deciding of pulse dial and tone dial, self diagnosing and so on, intelligently.

SR-120AT can be directly controlled by the software way. A set of commands and a set of registers are installed in the modem. The commands include dialing, answer, standard operation and register operation. By setting and reading the register set, the user can fully apply the flexibility in communication. It returns answer messages according to its current state changes through DTE to computer connected to it. Its acting states are divided into local command state and on-line state. When in local command state, the user can send control commands to it, when in on line state, all the data from DTE will be transferred to the other end of the telephone line.

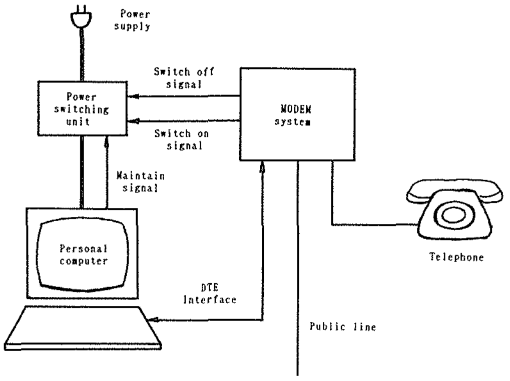

Figure2. Connection of modem and power switching unit.

The connection to personal computer and public line is shown in figure 2 including our designed power switching circuit. The switching unit acts when the modem is rung up to turn on the main power of computer system. When we finish our jobs on the remote machine, the main power will be turned off as the public line is closed. At times, when we wish to execute a big program which requires lengthly hours on a remote machine and can not maintainig the public line opening for economical reasons. Thus we add an auxiliary signal from DTE in the computer to control the power switching unit together with signals from the modem. Those are power maintainance signals, power on signal and power off signal. By this unit, we can fully controll the remote computer only through a telephone line.

7. Conclusion

In this paper we have described the problem of building networks between the same level of personal computers through public telephone lines. The designed software system and its usages are shown in detail. The system provides users with the same operation environment as in typical MS-DOS system. They can connect their computers to any of remote computers linked to public telephone including ones in local area even in other countries. Through the communication system, users can submit jobs to remote machine for execution and read the results and messages such as operating on his desk-top computer. The data files trasferred between terminals can be executive programs or text files or any type of files allowed in a normal DOS system. It will become very important and useful among personal computer users intending to utilize the hardware and software resources on the same level computers installed far from themselves. The system provides an environment of controlling remote computers when no one is beside them. It allows for the potentiality of one user to control and operate several computers located in different areas at the same time.

The research of incorporating the functions of auto-operation on set time and auto-data change remain open for innovation work in the future.

8. Acknowledgement

Here, we would like to express our sincere appreciation to the Telecommunications Advancement Foundation for their continuous support during the research and the experiments.

References

1) Seiko Co. Ltd. "Personal Modem SR-120AT User's Manual", 9M1SR12AT, Y34099100101, (1986).
2) D. Novak and Y. Aoki, "Local Area Networks-A Structured Implementation Example", Bulletin of the Faculty of Engineering, Hokkaido University, No. 131, pp. 11-20, (1986).
3) K. A. Hwang and W. C. Lee, "Message Control Program: A Remote Access Facility in Uni-Net", Proceedings of ICS'86, vol. 2, pp. 624-628, (1986).

Appendix 1. The programming of main routine

```
main ( )                                              /*  Program main routine  */
{
int       ri, rc, ki, kc, loop0=1, loopl, i, msg;
char      cmd [CMDLEN] , rmd [CMDLEN] , rmd0 [CMDLEN] , * rmdl;
     cflg=0; comf=0;
     if (rs232init( )) printf ("¥nRS232C initial error ¥!¥n");
     if ((cflg & 0x04)= =0x04)
          if (modeminit( )) printf ("¥nMODEM0 initial error ¥!¥n");
     while (loop0) {
          loop=1;
```

```
stdco (logdsk ( )+0x41);                                             /* Logged disk  */
comflag ( );
ki=0; ri=0;
while (loop1) {
      if (stdst ( )) {                                                /* keyboard st */
            if ((kc=stdci ( ))==BRK) {
                  if (!(cflg & 0x03)) loop0=0;                       /* Loop break   */
                  break; }
            else {if (kc==BKS) {
                              if (ki>0) {stdco (kc); stdco (0x20);
                                                           stdco (kc); ki--; }
                        }
                  elese if (isprint (kc)) stdco (kc);
                  if (kc==EDC) {
                        cmd [ki]='¥0';
                        if (cmd [0]==MRK) {                          /* End of comm. */
                              telclos ( );
                              cmd [0]='¥0'; }
                        if (strlen (cmd)) {
                              stdco (0x0d); stdco (0x0a);
                              if (strchr (cmd, MRK))
                                    rmtcmd (cmd);                    /* Remote cmd */
                              else
                                    loccmd (cmd);                    /* Local cmnd */
                              }
                        break; }
                  else if ((kc !=BKS) && isprint (kc)) cmd [ki++]=kc;
                  }
            }
      if (rstat( ) && ((cflg & 0x07)==0x07)) {
            if ((rc=rsinn ( ))==EDC) {
                  rmd [ri]='¥0';
                  if (rmd [1]=='#') {wait (CLOCK, 80);
                                                      cflg &=0xfc; break; }
                  strcpy (rmd0, rmd);
                  if ((rmdl=strsep (rmd, '    '))=='¥0') {
                        strcat (rmd, ">running. tmp¥0");
                        trncmd (rmd); }
                  elese {
                        if (!strcmp (rmdl, "copy")) {
                              rmtcpy (rmd); cflg ¦ =0x01; comflag ( ); }
                        else {
                              strcat (rmd0, ">running. tmp¥0");
                            trncmd (rmd0); }
                        }
                  break;
                  }
            else if (rc !=MRK) rmd [ri++]=mc;
            }
      if ((cflg & 0x07)==ox04)
            if (!lodmsg (&msg))
                  if (msg=='2') {
                        printf ("¥nBeing rung up by others ¥!¥n");
                        cflg=(cflg & 0xfc) ¦ 0x01;
                        comflag ( ); wait (CLOCK, 50);
                        for (i=0; i<RNG; i++) {
```

```
                                        if (!lodmsg (&msg))
                                              if (!promsg (msg)) {
                                                    cflg=(cflg & 0xf8) ¦ 0x07 ;
                                                    comflag (  ) ;
                                                    break ; }
                                        }
                                  if ((cflg & 0x07) !=0x07) {
                                        printf ("¥nConnecting failed ¥!¥n") ;
                                        cflg &=0xfc ; comflag (  ) ; }
                                  }
                  }
            }
      cflg=0 ; comflag (  ) ;
      printf ("¥nGame over ¥! Please keyin CCOM for replay. ¥n") ;
}
<Welcome to our new house !>
```

Appendix 2. The programming of remote command processor

```
rmtcmd (cmd0)                                          /・* Remote command processor */
char          * cmd0 ;
 {
char          * cmmd, * cmdl, * cmd2, * cmd3 ;
      cmmd=strdup (cmd0) ; cmdl=strchr (cmmd, MRK) ; cmdl++ ;
      if (! (cflg & 0x04) ¦ ¦ strchr (cmdl, MRK) ¦ ¦ (cflg & 0x08)) return ;
      if (!(cflg & 0x03)) if (telopen (  )) {
                                          cflg &=0xfe ; comflag (  ) ;
                                          printf ("¥nTelephone line open error ¥!¥n") ;
                                          return ; }
      if ((cmd2=strsep (cmmd, '     '))=='¥0') {
            sndcmd (cmmd) ; rmtdsp (  ) ; return ; }
      if (!strcmp (strlwr (cmd2)̄ "copy")) {
            if (cmmd [strlen (cmmd)-1]==MRK) {
                  cmdl=strdup ("copy!¥0") ;
                  strcat (cmdl, " aux ¥0") ;
                  if ((cmd3=strsep (cmmd, '     '))== '¥0') return ;
                  if ((cmd2=strsep (cmmd, '     ')) != '¥0') return ;
                  cmmd [strlen (cmmd)-1]= '¥0' ;
                  strcat (cmdl, cmmd) ; strcat (cmdl, "¥0") ;
                  sndcmd (cmdl) ; if (sndfil (cmd3)) endfil (0) ;           /* send file */
                  cflg ¦ =0x01 ; comflag (  ) ; }
            else {
                  cmdl=strdup ("copy! ¥0) ;
                  if ((cmd2=strsep (cmmd, '     '))== '¥0') return ;
                  if ((cmd3=strsep (cmmd, '     ')) != '¥0') return ;
                  if ((cmd3=strchr (cmd2, MRK))) cmd3 [0]='¥0' ; else return ;
                  strcat (cmdl, cmd2) ; stracat (cmdl, " aux¥0") ;
                  sndcmd (cmdl) ;
                  wait (CLOCK, 18) ; if (lodfil (cmmd)) endfil (1) ;         /* load file */
                  cflg ¦ =0x01 ; comflag (  ) ; }
            }
      else { sndcmd (cmd0) ; rmtdsp (  ) ; }
}
 sndcmd (cmmd)                                         /* Send command via RS232C-0   */
 char          * cmmd ;
  {
 int           i=0 ;
```

```
        while (cmmd [i] != '¥0') rsout (cmmd [i++] );
        rsout (EDC);
}
rmtcpy (cm0)                                                    /* Remote file copy routine   */
char        * cm0;
 {
char        * cm, * cm1;
        cm=strdup (cm0);
        if ((cm1=strsep (cm, '    '))== '¥0) {
                if (strcmp (cm, "aux")) endfil (0); return ;}
        else {
                if (strcmp (cm, "aux")) { if (lodfil (cm)) endfil (1); }
                else { if (sndfil (cm1)) endfil (0); }
                }
}
trncmd (cd)                                                     /* Transfer command process   */
char        * cd;
 {
int         i=0;
        while (cd [i] != '¥0') if (!isprint (cd [i++] )) return;
        if (system (cd)== -1) perror ( );
        sndtmp ( ); rsout (EDF); cflg | =0x01; comflag ( );
}
<Welcome to our new house !>
```

Appendix 3. The programming of modem controller

```
modeminit ( )                                                   /* Epson modem initialize */
 {
int         msg;
        if (modemtest ( )) { cflg | = 0x08; return (1); }
        wait (CLOCK, 8); sndcmd ("ATZ");                        /* Select answer code   */
        if (lodmsg (&msg) | | (msg != '0')) return (1);
        wait (CLOCK, 8); sndcmd ("ATS18=0");                    / * Disable prompt mark  /
        if (lodmsg (&msg) | | (msg != '0')) return (1);
        sndcmd ("ATS0=2");                                      /* Set ring couter    */
        if (lodmsg (&msg) | | (msg != '0')) return (1);
        sndcmd ("ATS7=55")                                      /* Wait time of carrier */
        if (lodmsg (&msg) | | (msg != '0')) return (1);
        cflg | = 0x04; return (0);
}
lodmsg (mssg)                                                   /* Read message from modem   */
int         * mssg;
 {
int         i, mc= 'a';
        for (i=0; i<DELAY; i++) {
                if (rstat ( )) {                                /* Read modem message   */
                        * mssg=rsinn ( );
                        if (isdigit (mc) && (* mssg==0x0d)) {
                                * mssg=mc;
                                return (0); }
                        mc= * mssg; }
                if (stdst ( )) {                                /* Anykey for dial stop */
                        rsout (0x0d); break; }
                }
        * mssg=0x39;                                            /* Dummy return number   */
```

```
      return (1);
}
telopen ( )                                              /* Open telephone line of com.  */
 {
int           msg;
char           * strd, * oel;
       tel=getelno ( ); if (tel [0]==0) return (1);
       strd=strdup ("ATD"); strcat (strd, tel); strcat (strd, "¥0");
       sndcmd (strd);
       cflg=(cflg & 0xfc) ¦ 0x01; comflag ( );
       wait (CLOCK, 50);
       if (lodmsg (&msg)) { rsout (0x0d); return (1); }
       if (msg== '7') {
              printf ("¥nRedialing ¥!¥n");
              if (lodmsg (&msg)) return (1); }
       switch            (msg) {
              case       '1':
                     printf ("¥nConneco to 300 bps line ¥!¥n");
                     break;
              case       '2':
                     errmsg (2); return (1);
                     break;
              case       '3':
                     errmsg (3); return (1);
                     break;
              case       '5':
                     printf ("¥nConnect to 1200 bps line ¥!¥n");
                     break;
              case       '6':
                     errmsg (6); return (1);
                     break;
              case       '8':
                     errmsg (8); return (1);
                     break;
              default:
                     errmsg (4); return (1);
                     break;
              }
       cflg=(cflg & 0xfc) ¦ 0x03; comflag ( );
       return (0);
}
telclos ( )                                              /* Close telephone line of com. */
 {
       if ((cflg & 0x07) !=0x07) return (1);
       sndcmd ("###"); wait (CLOCK, 20):
       rsout (ESP); rsout (ESP); rsout (ESP)                         /* Escape code   */
       wait (CLOCK, 20); modeminit ( );
       cflg &=0xfc; comflag ( );
       return (0);
}
<Welcome to our new house !>
```