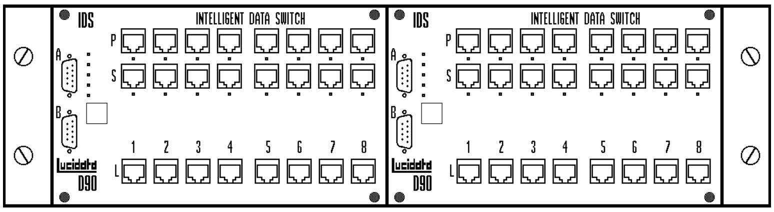
Lucidata D90-IDS
Intelligent Data Switch
User Guide



Lucidata House
Selwyn Close
Great Shelford
CAMBRIDGE CB2 5HA
England

tel: +44(0)1223 846100
fax: +44(0)1223 846200

Publication Details

All possible care has been taken in the preparation of this publication, but Lucidata accepts no liability for any inaccuracies that may be found.

Lucidata reserves the right to make changes without notice to both this publication and to the product which it describes.

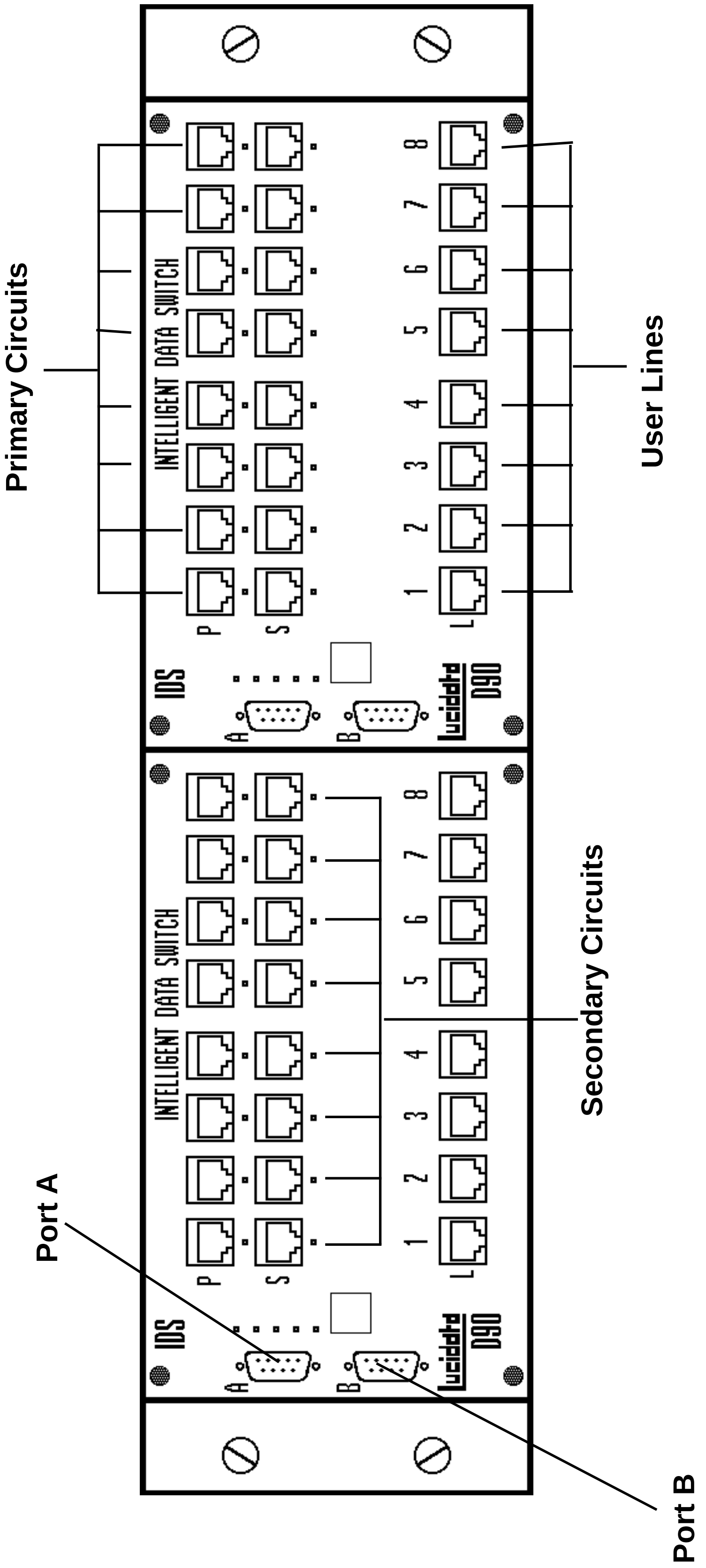
If you find any errors in this publication or would like to make suggestions for improvement, please write to the Company at the address below.

Lucidata House
Selwyn Close
Great Shelford
CAMBRIDGE
CB2 5HA
England
tel:(01223) 846100
int:+44 1223 846100
fax:(01223) 846200
email: docs@lucidata.com

Publication number D90/IDS Issue number 1 (07/94)

Revision Details	Date	Pages

Lucidata warrants that this product is free from defects in manufacture and that it meets the specifications outlined in this User Guide for a period of one year following purchase provided that the product has only been used in the manner and for the purpose described in this User Guide. This Warranty does not affect your statutory rights.


Primary Circuits
User Lines
Secondary Circuits
Port A
Port B
INTELLIGENT DATA SWITCH
INTELLIGENT DATA SWITCH
IDS
IDS
P
S
L
P
S
L
1
2
3
4
5
6
7
8
1
2
3
4
5
6
7
8
A
B
A
B
Lucidata D90
Lucidata D90

Introduction

The D90 Intelligent Data Switch (IDS) enables user Lines connected to it to be switched between a Primary circuit and a Secondary circuit. A typical example would be where the Primary circuit is from the main system, and the Secondary circuit is from a backup system. The user Lines can be switched individually, in groups , or all at once under the control of a Simple Command Language (SiCL). The switching commands can either be entered manually from a terminal connected to Port A of the master bank, or generated by a higher level network management system. Used this way the Intelligent Data Switch can be configured to act as a remotely operated patch panel.

A physical switch (panic button) is also provided to switch all connected lines in one simple manual operation.

The Intelligent Data Switch (IDS) is part of the D90 range of modular datacommunication products and can communicate with other modules via the D90 bus.

One IDS module can switch eight, four-pair Lines to either a Primary or Secondary circuit. The user lines (L)are connected to the bottom row of 8 RJ45 sockets. The Primary service (P) is connected to the top row of sockets and the Secondary service (S) to the middle row. Each set of eight Lines is called a Bank and a single D90 rack can hold a maximum of two Banks.

Each IDS module has two standard 9 pin female D-type connectors. These are used to daisy-chain any number of racks together via crossover cables from Port B on one IDS module to Port A on the next rack.

The very first IDS module in the chain is considered to be the Master module and it can be controlled by a Simple Command Language (SiCL) either via Port A or over channel 1 of the D90BUS.

Setup

All the IDS modules can be controlled from the Master Module which is the first module. Up to 128 racks (each containing 2 IDS modules) can be daisy-chained together by connecting Port B of the left-hand module to Port A of the left-hand module in the next rack. The pin connections are described later in this document.

Communication with the Master Module is via Port A. This can be connected to a terminal, or a to a PC running a terminal emulation. The pin connections for Port A are given in the *Control Port* section of this manual. Port A of the Master Module can also be connected to the physical switch (panic button). If a VT100 or Wyse50 terminal or emulation is used, a simple formatted display may be obtained by first selecting the terminal type. This is done by typing CTRL/P and entering the digit indicated on the resulting menu to select the desired terminal.

A power lead should be connected to each rack and an appropriate Mains supply (220-240 volt 50Hz).

Initialisation

On power-up the IDS automatically determines how many IDS modules are daisy-chained together, identifies them, and initialises itself.
If there is a configuration stored in Battery-Backed RAM then this will automatically be invoked. Otherwise all lines will be set to the Primary circuit.

Program Control

One way of simply changing the configuration is to use the program Dataswitch which is supplied as the executable file IDS.EXE. This is written in Borland Turbo Pascal and a listing is given in the Appendix.

Configuration File

The program loads a Configuration File which can be created by any simple text editor such as MS-DOS's EDIT. An example of a Configuration File is given in the Appendix.

Each line is uniquely identified by the Bank number and Line number within that Bank. The user can also give a unique identifier (up to 8 characters) to each cable or plug. He can also choose a name of up to 24 characters for the Primary and Secondary services and an abbreviation for that name of up to 8 characters.
By running IDS.EXE, the configuration can be inspected, altered and saved as a new Configuration File by using the simple keystrokes indicated at the bottom of the screen.

The displayed configuration will not be implemented until the “Use Current” Command is invoked - ie Function Key F5 is pressed.

Print Configuration

The displayed configuration can be printed on a suitable printer via the LPT1: port on a PC. The printout forms a useful record for the Network Manager and his engineers. An example of the format is given in the Appendix.

Note: *The supplied program contains the printer control sequences for a Hewlett-Packard Deskjet printer and may need to be changed for other printers.*

For users wishing to write their own control program, or to generate the command strings in some other way, a description of the Simple Command Language SiCL follows.

The following paragraphs describe the Simple Command Language (SiCL) that can be used to activate the switches in the IDS modules.The language has been kept simple so that it can be easily incorporated into the user’s own Network Management System.

SiCL is a true command language in that commands are only defined in the direction Master module to Subordinate module. All other data flow are responses.

Lexical Tokens These are as follows:

Commands **? P S N**

Symbols *** ‘ , - B L 0 2 3 4 5 6 7 8 9**

Delimiter and Editor **CR BS** (The ASCII characters with value 13 and 8)

Language Syntax

```
COMMAND ::=      ‘?’!’P’<IDENTIFIER>!’S’<IDENTIFIER>!’N’<NUMBER>CR

IDENTIFIER ::=   ‘*’!’B’<BANKLIST>!’B’<BANKLIST>’L’<LINELIST>

BANKLIST ::=     NUMBER<‘,’BANKLIST>!NUMBER’-’NUMBER<‘,’BANKLIST>

LINELIST ::=     DIGIT<‘,’LINELIST>!DIGIT’-’DIGIT<‘,’LINELIST>!'*'

DIGIT ::=  1..8

NUMBER  ::=      1..255 (Release 1.0)
```

A command is not actioned until a CR terminator is received so editing of a command line can be performed using the destructive BS prior to sending the CR.

SiCL has been kept spartan so that it is easy to remember and so that command strings can be easily generated by a Network Management program for example.

Language Semantics The semantics of the language follows:

Initialise The Next Bank command, ‘N’, informs the module that it is the Nth bank in the chain and that it should note the fact and use the value in future messages.

Status The Enquiry command ‘?’, causes the module to report the connection state of each of its Lines in the following format

Bnnnnllllllll CR
where nnnn is the Bank number and llllllll are the connections of the lines one to eight.

Examples: B1PPPPSPSS or B123SSSSPSSS
It is the only command that invokes a response from the IDS module.

The switch commands, ‘P’ or ‘S’, instruct the module to switch the identified lines to either the Primary or Secondary circuit. The identification is heirarchical, thus minimizing the number of characters needing to be sent. Thus to cause all lines in the system to be switched to their Secondary circuits requires only the sequence:

```
S* CR
```

This is equivalent to the longer

```
SB1-(maximum bank number) CR
```

Whole banks of lines can be switched by specifying a list of banks

```
SB1,3,4,8-20 CR PB2,5,6,7 CR
```

To switch just a single line, a command such as

```
SB99L3 CR
```

will just switch line 3 on bank 99 to the secondary circuit.
To switch a selection of lines in a single bank

```
SB99L1,3,6-8 CR
```

Weight & Dimensions

Height of subrack	132.5mm (3U)
Width of subrack	482.6mm (19")
Depth of subrack	240mm
Weight of subrack and power supply	7kg(approx)

Electrical Requirements

Frequency	47-440Hz single phase
Voltage	180-264/90-132V (factory selectable)
Internal Power Supply	20 watt switched mode

Operating environment

Temperature	0-50^{o}C
Humidity	0-90% non-condensing

External connectors

Power	IEC mains plug
24 x RJ45	
2 x 9-pin D-type	

External Indicators

5 x LEDs	On Port A
8 x LEDs	On Primary Sockets
8 x LEDs	On Secondary Sockets

Configuration

At power up	Automatic - all lines switched to Primary or to values in Battery backed RAM (if present)

Data Circuits

Designed to support data rates in excess of 100 Mbps on 100ohm UTP cabling and plugs wired to T568A, T568B, 10BASE-T, Token Ring or TP-PMD formats.

Control Circuits

Data rate of 9600 bps, 8 data bits, no parity, 1 stop bit.
Supports dumb terminal or genuine VT100 or WY50.

Control Port A

The table below shows the pin connections to Port A when it is connected to a VDU or IBM PC COM port.

PIN NO.		
1	CD	Carrier Detect asserted high by IDS when port enabled
2	RXD	Received Data - IDS transmits data on this pin
3	TXD	Transmitted Data - IDS receives data on this pin
4	DTR	Data Terminal Ready - enables IDS transmitter
5	SG	Signal Ground
6	DSR	Held high by IDS as long as unit powered up
7	RTS	Request To Send - sensed by IDS
8	CTS	Clear To Send - internally connected to pin 7
9	RI	Ring Indicator - not used

Both pin 4 and pin 7 must be held high for the control Port A to function.

Panic Mode

If pin 7 on Port A is lowered it will force an immediate switch over of all lines to their Secondary circuits. If this feature is not required, pins 4 and 7 of Port A on Bank 1 should be connected to pin 6 and only a simple three wire connection (pins 2,3 and 5) made to the terminal or PC. This will prevent unwanted switching when the PC is turned on or off.

Port B

This port can be connected to Port A *in the next rack.* If there are two IDS modules in a rack, they are internally connected via the D90 rack BUS. The pin connections for Port B are the same as those given for Port A.

LED Indicators

There are five LEDs located to the right of Port A and one red LED by each of the eight Primary and eight Secondary RJ45 sockets. The LEDs by the sockets simply indicate to which circuit the corresponding Line is connected. The other five have the following meaning.

Starting from the top, the yellow LED will toggle every 2.5 seconds for as long as the module is functioning.

The second LED is red and if illuminated indicates that there are two modules in the rack and they have established a relationship.
The third LED is red and when illuminated indicates that pins 4 and 7 on Port A is high.

The fourth LED is red and when illuminated indicates that pins 4 and 7 on Port B is high.

The fifth green LED flickers if the module is addressed on the D90BUS.

Appendix

Example Configuration File

Listing of Program Dataswitch

Example of Printout from Program Dataswitch

Configuration File

The following is an example of a Configuration File. This file is the input data for the Pascal program called Dataswitch that follows. After reading in the Configuration File, the Dataswitch program enables the lines to be switched and stored as another Configuration File. In this example there are 4 banks. The first and third bank have all their lines set to the Primary Service and the second and fourth banks have all their lines switched to the Secondary Service. In practice it is not necessary to include the service number at the end of each cable line as they can be set by the Dataswich program.

```
* ALL COMMENT LINES START WITH AN ASTERIX AND ARE IGNORED
* SERVICES AND CABLE DATA MUST BE IMMEDIATELY PRECEDED BY A *SERVICE
* AND A *CABLE LINE AS SHOWN BELOW WITH NO SPACE AFTER THE ASTERIX
*  CABLE IDs MAY HAVE A MAXIMUM OF 8 CHARACTERS
*SERVICES: [NO.],[FULLNAME(24 chars max)],[SHORTNAME(8 chars max)]
1,PRIMARY,P
2,SECONDARY,S
*CABLE DATA:[BANK],[LINE],[LINECABLEID],[PRIMARYCABLEID],[SECONDARYCABLEID],[SERVICE NO.]
1,1,L0001,P0001,S0001,1
1,2,L0002,P0002,S0002,1
1,3,L0003,P0003,S0003,1
1,4,L0004,P0004,S0004,1
1,5,L0005,P0005,S0005,1
1,6,L0006,P0006,S0006,1
1,7,L0007,P0007,S0007,1
1,8,L0008,P0008,S0008,1
2,1,L0009,P0009,S0009,2
2,2,L0010,P0010,S0010,2
2,3,L0011,P0011,S0011,2
2,4,L0012,P0012,S0012,2
2,5,L0013,P0013,S0013,2
2,6,L0014,P0014,S0014,2
2,7,L0015,P0015,S0015,2
2,8,L0016,P0016,S0016,2
3,1,L0017,P0017,S0017,1
3,2,L0018,P0018,S0018,1
3,3,L0019,P0019,S0019,1
3,4,L0020,P0020,S0020,1
3,5,L0021,P0021,S0021,1
3,6,L0022,P0022,S0022,1
3,7,L0023,P0023,S0023,1
3,8,L0024,P0024,S0024,1
4,1,L25,P25,S25,2
4,2,L26,P26,S26,2
4,3,L27,P27,S27,2
4,4,L28,P28,S28,2
4,5,L29,P29,S29,2
4,6,L30,P30,S30,2
4,7,L31,P31,S31,2
4,8,L32,P32,S32,2
```

```
(********************************************************************)
(*                                                                  *)
(*  THE INTELLIGENT DATA SWITCH CONTROL PROGRAM                     *)
(*                                                                  *)
(*  written by Eileen Bennee for LUCIDATA                           *)
(*  Version 1.0 14-07-94                                            *)
(*                                                                  *)
(********************************************************************)
program dataswitch;

uses crt,printer;

const
  nlines=512;           (* total number of lines (16 x No. of racks) *)
  linespp=16;                (* actual data lines displayed per page *)
  sp=#32;
  esc=#27;
  bell=#7;
  ymin=3;                       (* top and bottom positions on page   *)
                                (* of data lines ymax=ymin+linespp-1  *)
type
   cable = record
             bank:integer;
             line:integer;
             cl:string[8];
             cp:string[8];
             cs:string[8];
             serviceno:byte;
           end;

    service = record
                no:integer;
                fullname:string[24];
                idname:string[8];
              end;

var  i,l,dum,firstline,lastline,firstrecord,nopages,linesonlastpage:integer;
    ch:char;
    servicedata,cabledata:boolean;
    nrec,nservices,page,ybar,ymax,serno:integer;
    dfname:string;
    s:string;
    datafile:text;
    cport:text;
    c:array[1..nlines] of cable;
    ser:array[1..8] of service;
```

```
(*********************************************************************)
(*  THE FOLLOWING PROCEDURES ARE NOT MACHINE INDEPENDENT             *)
(*********************************************************************)

procedure ejectpage;                        (* eject page on printer *)
begin
  write(lst,esc,'&l0H');                           (* for HP Deskjet *)
end;
procedure resetprinter;
begin
  write(lst,esc,'E');                              (* for HP Deskjet *)
end;
procedure highlight;                                       (* for PC *)
begin
  textcolor(red);
end;
procedure normal;                                          (* for PC *)
begin
  textcolor(white);
end;


(*********************************************************************)
(*  INITIALISE                                                       *)
(*********************************************************************)

procedure clearcablearray;
var i:integer;
begin
  for i:=1 to nlines do
  begin
    c[i].bank:=0;
    c[i].line:=0;
    c[i].cl:='';
    c[i].cp:='';
    c[i].cs:='';
    c[i].serviceno:=0;
  end;
end;
```

```
procedure init;
var ok:boolean;
begin
  clearcablearray;
  ok:=false;
  textcolor(white);
  textbackground(black);
  clrscr;
  gotoxy(28,5);
  write(‘WELCOME TO THE LUCIDATA’);
  gotoxy(28,7);
  write(‘INTELLIGENT DATA SWITCH’);
  gotoxy(32,9);
  write(‘CONTROL PROGRAM’);
  gotoxy(30,15);
  write(‘Version 1.0 14-07-94’);
  gotoxy(28,17);
  write(‘copyright LUCIDATA 1994’);
  gotoxy(25,20);
  write(‘HIT ANY CHARACTER TO CONTINUE’);
  ch:=readkey;

  repeat
    {$I-}
    assign(cport,’AUX’);
    rewrite(cport);
    writeln(cport,’N1');
    {$I+}
    if ioresult <> 0 then
    begin
      clrscr;
      gotoxy(20,5);
      write(‘CHECK THAT THE COM1: PORT IS CONNECTED ‘);
      gotoxy(23,7);
      write(‘AND THE IDS UNIT IS POWERED UP !’);
      gotoxy(20,20);
      write(‘HIT RETURN TO CONTINUE    [ESC] TO QUIT’ ,bell);
      ch:=readkey;
      if ch=esc then halt;
    end else ok:=true;
  until ok;

end;
```

```
procedure prs(s:string;f:integer); (* prints string in fieldwidth f *)
var i,l:integer;                   (* left justified                *)
begin
  l:=length(s);
  if l<=f then
  begin
    for i:=1 to l do write(lst,s[i]);
    for i:=1 to f-l do write(lst,sp);
  end else for i:=1 to f do write(lst,s[i]);
end;

procedure writes(s:string;f:integer);            (* displays string *)
var i,l:integer;                                 (* in fieldwidth f *)
begin
  l:=length(s);
  if l<=f then
  begin
    for i:=1 to l do write(s[i]);
    for i:=1 to f-l do write(sp);
  end else for i:=1 to f do write(s[i]);
end;
(********************************************************************)
(*  SWITCHING PROCEDURES                                            *)
(********************************************************************)

procedure  linechange(row:integer);
var i:integer;
begin
   i:=(page-1)*linespp+(row-2);

  (* new code will be required here for multiple  *)
  (* switching and abbreviated forms of switching *)
  (* commands for very large numbers of lines     *)

  if c[i].serviceno=1 then
  begin
     writeln(cport,'SB',c[i].bank,'L',i);
     c[i].serviceno:=2
  end  else
  begin
     writeln(cport,'PB',c[i].bank,'L',i);
    if c[i].serviceno=2 then c[i].serviceno:=1;
  end

end;
```

```
procedure bankchange(row:integer);           (* F2 key - switch bank *)
var b,s,j,i:integer;

begin
   i:=(page-1)*linespp+(row-2);
  b:=c[i].bank;
  s:=c[i].serviceno;
  if c[i].serviceno=1 then
  begin
     writeln(cport,'SB',c[i].bank,'L*');
    for j:=1 to nrec do if c[j].bank = b then c[j].serviceno:=2;
  end else
  begin
     writeln(cport,'PB',c[i].bank,'L*');
    for j:=1 to nrec do if c[j].bank = b then c[j].serviceno:=1;
  end;
end;

procedure allprimary;         (* F3 key - set all to Primary Service *)
var i:integer;
begin
  for i:=1 to nrec do c[i].serviceno:=1;
  writeln(cport,'P*');
end;

procedure allsecondary;     (* F4 key - set all to Secondary Service *)
var i:integer;
begin
  for i:=1 to nrec do c[i].serviceno:=2;
  writeln(cport,'S*');
end;

procedure doconfig;              (* F5 key use current configuration *)
var i:integer;
begin
  for i:=1 to nrec do
  begin
     if c[i].serviceno=1 then writeln(cport,'PB',c[i].bank,'L',c[i].line)
                         else writeln(cport,'SB',c[i].bank,'L',c[i].line);
  end;
end;

procedure saveconfig;           (* F6 key save current configuration *)
var ch:char;
    filename:string;
    datafile:text;
```

```
procedure writedata;
var i:integer;
begin
  rewrite(datafile);
  writeln(datafile,’*SERVICES’);
  for i:=1 to nservices do
       writeln(datafile,ser[i].no,’,’,ser[i].fullname,’,’,ser[i].idname);
  writeln(datafile,’*CABLES’);
  for i:=1 to nrec do
  begin
     write(datafile,c[i].bank,’,’,c[i].line,’,’);
     writeln(datafile,c[i].cl,’,’,c[i].cp,’,’,c[i].cs,’,’,c[i].serviceno);
  end;
end;

begin
  clrscr;
  gotoxy(20,2);
  write(‘SAVE CURRENT CONFIGURATION’);
  gotoxy(20,4);
  write(‘ENTER FILE NAME - ‘);
  readln(filename);
  {$I-}
  assign(datafile,filename);
  reset(datafile);
  {$I+}
  if ioresult <> 0 then
  begin                   (* file does NOT exist so we can open one *)
    rewrite(datafile);
    writedata;
    gotoxy(20,6);
    write(‘CONFIGURATION SAVED ON ‘,filename);
    ch:=readkey;
  end else
  begin
    gotoxy(20,6);
    write(bell,bell,’WARNING FILE ‘,filename,’ ALREADY EXISTS’);
    gotoxy(20,7);
    writeln(‘DO YOU WANT TO OVERWRITE ? y/n’);
    ch:=readkey;
    if ch=’y’ then begin
                     rewrite(datafile);
                     writedata;
                     gotoxy(20,6);
                     write(‘CONFIGURATION SAVED ON ‘,filename);
                     ch:=readkey;
                   end;
  end;
  close(datafile);
end;
```

```
(*********************************************************************)
(*  DISPLAY PROCEDURES                                               *)
(*********************************************************************)

procedure mainhead;                (* display heading top and bottom *)
begin
  clrscr;
  textcolor(white);
  textbackground(black);
  gotoxy(1,1);
  write(‘BANK’);
  gotoxy(10,1); write(‘LINE’);
  gotoxy(20,1); write(‘SERVICE    NO. OF LINES = ‘,nrec);
  gotoxy(1,21);
  write(‘SWITCH - [F1] Line [F2] Bank    [F3] All ‘);
  writes(ser[1].idname,8);
  write(‘ [F4] All ‘);
  writes(ser[2].idname,8);
  gotoxy(1,22);
  write(‘CONFIG - [F5] Use Current   [F6] Save Current’);
  write(‘  [F7] Print Current  [F8] Load New ‘);
  gotoxy(1,23);
  write(‘FILE   - ‘,dfname);

  gotoxy(1,24);
  write(‘USE ARROWS ‘,chr(24),chr(25));
  write(‘ TO SELECT LINE  Page Up/Page Down TO SELECT PAGE’);
  write(‘     [ESC] TO QUIT’);

end;

procedure bar(row:integer);  (* positions cursor bar at row position*)
var i:integer;
begin
  i:=(page-1)*linespp+(row-2);
  if (i>0) and (i<=nrec) then
  begin
    gotoxy(1,row);
    highlight;
    gotoxy(1,row);
    write(c[i].bank:4,sp:4,c[i].line:4);
    gotoxy(20,row);
    writes(ser[c[i].serviceno].fullname,24);
    normal;
  end;
end;
```

```
(*********************************************************************)
(*  PRINTOUTDATA                                                     *)
(*********************************************************************)

procedure printoutdata;
const maxlinesperpage=60;
var i,j,k:integer;
procedure printheader;
begin
   writeln(lst);
   writeln(lst);
   write(lst,sp:24);
   write(lst,'– USER CABLE IDENTIFICATION –',sp:5,'CONNECTED');
   writeln(lst);
   write(lst,'    BANK    LINE',sp:8,'LINE        PRIMARY');
   writeln(lst,'     SECONDARY     SERVICE');
   writeln(lst);
end;
begin
  resetprinter;
  printheader;
  for i:=1 to nrec do
  begin
     write(lst,sp:4,c[i].bank:4,sp:4,c[i].line:4);
    write(lst,sp:8);
    prs(c[i].cl,8);
    write(lst,sp:4);
    prs(c[i].cp,8);
    write(lst,sp:4);
    prs(c[i].cs,8);
    write(lst,sp:6);
     prs(ser[c[i].serviceno].idname,14);
    writeln(lst);
    if i mod maxlinesperpage = 0 then
    begin
      ejectpage;
      printheader;
    end;
  end;
  ejectpage;
end;
```

```
(*********************************************************************)
(*  READINDATA                                                       *)
(*********************************************************************)

procedure readindata;
var i,j,k,l,m,n,nr,sr:integer;
    error:integer;
    p:array[1..10] of byte;
    a:array[1..10] of string;
    endcomma,ok:boolean;
    l1,l2,ncommas:integer;

procedure getvalues;
var iclear:integer;                  (* decode strings separated by *)
begin
  for iclear:=1 to 10 do a[iclear]:='';
  endcomma:=false;                  (* commas into array a (max 10) *)
  l:=length(s);
  if s[length(s)]=',' then endcomma:=true;
  i:=0;
  repeat                           (* get position of all the commas *)
    i:=i+1;
    p[i]:=pos(',',s);
    s[p[i]]:=' ';
  until p[i]=0;
  p[i]:=l+1;
  ncommas:=i;
  i:=0;
  l1:=1;
  repeat
    i:=i+1;
    l2:=p[i]-l1;
    a[i]:=copy(s,l1,l2);
    l1:=p[i]+1;
  until i=ncommas;
  if endcomma then n:=i-1 else n:=i;
end;

begin                                                (* readindata *)
  cabledata:=false;
  servicedata:=false;
  ok:=false;
  repeat
  clrscr;
  gotoxy(25,8);
  write('ENTER NAME OF DATA FILE - ');
  readln(dfname);
  gotoxy(25,12);
  clreol;
  {$I-}
  assign(datafile,dfname);
  reset(datafile);
  {$I+}
  if ioresult <> 0 then
  begin
    gotoxy(29,12);
    write('FILE ',dfname,' NOT FOUND !',bell);
    gotoxy(18,20);
    write('HIT RETURN TO CONTINUE  [ESC] TO QUIT');
    ch:=readkey;
    if ch=esc then halt;
  end else ok:=true;
  until ok;
  nr:=0;
  sr:=0;
  while not eof(datafile) do
  begin
    readln(datafile,s);
    if ioresult<>0 then write('IORESULT NON ZERO');
    if s[1]='*' then                                (* comment line *)
    begin
      cabledata:=false;
      servicedata:=false;
      if (s[2]='S') or (s[2]='s') then servicedata:=true;
      if (s[2]='C') or (s[2]='c') then cabledata:=true;
    end else
    begin                                             (* data line *)
      getvalues;
      if cabledata then
      begin
        nr:=nr+1;
        val(a[1],c[nr].bank,error);
        if error <>0 then writeln('INPUT ERROR ',error);
        val(a[2],c[nr].line,error);
        if error <>0 then writeln('INPUT ERROR ',error);
        c[nr].cl:=a[3];
        c[nr].cp:=a[4];
        c[nr].cs:=a[5];
        if length(a[6])<>0 then
        begin
          val(a[6],c[nr].serviceno,error);
          if error <>0 then writeln('INPUT ERROR ',error);
        end else c[nr].serviceno:=1;
      end else
      if servicedata then
      begin
        sr:=sr+1;
        val(a[1],serno,error);
        if error <>0 then writeln('INPUT ERROR ',error);
        ser[sr].no:=serno;
        ser[sr].fullname:=a[2];
        ser[sr].idname:=a[3];
end else
      begin           (* type of data not flagged with comment line *)
        write('TYPE OF DATA UNKNOWN!');
        exit;
      end;
    end;
  end;
  nrec:=nr;
  nservices:=sr;
end;


(********************************************************************)
(*   MAIN PROGRAM                                                   *)
(********************************************************************)
begin
  init;
  readindata;
  firstrecord:=0;
  firstline:=ymin;
  lastline:=ymax;
  nopages:=nrec div linespp+1;
  linesonlastpage:=nrec-(nopages-1)*linespp;
  page:=1;
  mainhead;
  l:=ymin-1;
  ybar:=firstline;
  repeat
    ymax:=ymin+linespp-1;
    mainhead;
    l:=ymin-1;
    i:=firstrecord;
    repeat
      i:=i+1;
      l:=l+1;
      gotoxy(1,l);
      write(c[i].bank:4,sp:4,c[i].line:4);
      gotoxy(20,l);
      writes(ser[c[i].serviceno].fullname,24);
      bar(ybar);
    until (i mod linespp=0) or (i=nrec);
    if page=nopages then ymax:=ymin+linesonlastpage-1;
    ch:=readkey;


    if ch=#0 then                                   (* function key *)
    begin
      ch:=readkey;
      if ch=#80 then
      begin                                           (* down arrow *)
        if ybar = ymax then ybar:=ymin else ybar:=ybar+1;
      end;
      if ch=#81 then
      begin                           (* pagedown - go to next page *)
        ybar:=ymin;
        firstrecord:=firstrecord+linespp;
        if i>=nrec then
        begin                              (* go back to first page *)
          firstrecord:=0;
          page:=1;
        end else
        begin
          firstline:=ymin;
          page:=page+1;
        end;
      end;

      if ch=#72 then                                     (* up arrow *)
      begin
        ybar:=ybar-1;
        if ybar<ymin then ybar:=ymax;
      end;
      if ch=#73 then                                      (* page up *)
      begin
        if page>1 then
        begin
          page:=page-1;
          firstrecord:=firstrecord-linespp;
          ybar:=ymin;
        end else
        begin                                   (* back to first page *)
          page:=1;
          ybar:=ymin;
          firstrecord:=0;
        end;
      end;
      if ch=#59 then linechange(ybar);
      if ch=#60 then bankchange(ybar);
      if ch=#61 then allprimary;
      if ch=#62 then allsecondary;
      if ch=#63 then doconfig;
      if ch=#64 then saveconfig;
      if ch=#65 then printoutdata;
      if ch=#66 then begin
                       close(datafile);
                       readindata;
                     end;
    end;                              (* end of function key choices *)


  until ch=esc;
  if ch=esc then exit;
end.
```

```
                           --- USER CABLE IDENTIFICATION ---     CONNECTED
    BANK     LINE          LINE         PRIMARY      SECONDARY     SERVICE

       1        1          L0001        P0001        S0001         P
       1        2          L0002        P0002        S0002         P
       1        3          L0003        P0003        S0003         P
       1        4          L0004        P0004        S0004         P
       1        5          L0005        P0005        S0005         P
       1        6          L0006        P0006        S0006         P
       1        7          L0007        P0007        S0007         P
       1        8          L0008        P0008        S0008         P
       2        1          L0009        P0009        S0009         P
       2        2          L0010        P0010        S0010         P
       2        3          L0011        P0011        S0011         P
       2        4          L0012        P0012        S0012         P
       2        5          L0013        P0013        S0013         P
       2        6          L0014        P0014        S0014         P
       2        7          L0015        P0015        S0015         P
       2        8          L0016        P0016        S0016         P
       3        1          L0017        P0017        S0017         P
       3        2          L0018        P0018        S0018         P
       3        3          L0019        P0019        S0019         P
       3        4          L0020        P0020        S0020         P
       3        5          L0021        P0021        S0021         P
       3        6          L0022        P0022        S0022         P
       3        7          L0023        P0023        S0023         P
       3        8          L0024        P0024        S0024         P
       4        1          L25          P25          S25           P
       4        2          L26          P26          S26           P
       4        3          L27          P27          S27           P
       4        4          L28          P28          S28           P
       4        5          L29          P29          S29           P
       4        6          L30          P30          S30           P
       4        7          L31          P31          S31           P
       4        8          L32          P32          S32           P
       5        1          L33          P33          S33           P
       5        2          L34          P34          S34           P
       5        3          L35          P35          S35           P
       5        4          L36          P36          S36           P
       5        5          L37          P37          S37           P
       5        6          L38          P38          S38           P
       5        7          L39          P39          S39           P
```