

```

{***** R A W C O O K *****)
{
{
Task      :   provide two functions to switch
              a character device driver to the RAW-
              or the COOKED mode
}
}
{*****}
{
Author      :   MICHAEL TISCHER
Developed on :   08/16/87
Last update  :   02/17/92
}
{*****}

program RAWCOOKP;

Uses Crt, Dos;                                { Add CRT and DOS units }

const STANDARDIN = 0;                          { Handle 0 refers to standard input }
      STANDARDOUT = 1;                        { Handle 1 refers to standard output }

var Keys : char;                              { Needed for this demo only }

{*****}
{
* GETMODE: Reads device driver attribute.
* Input  : Handle passed (must refer to addressed device)
* Output : Device attribute
}
{*****}

function GetMode(Handle : integer) : integer;

var Regs : Registers;                        { Register variable for interrupt call }

begin
  Regs.ah := $44;                          { Function number for IOCTL: Get Mode }
  Regs.bx := Handle;
  MsDos( Regs );                          { Call DOS interrupt 21H }
  GetMode := Regs.dx;                      { Pass device attribute }
end;

{*****}
{
* SETRAW : Changes a character driver into RAW mode.
* Input  : Handle passed (must refer to addressed device)
* Output : None
}
{*****}

procedure SetRaw(Handle : integer);

var Regs : Registers;                        { Register variable for interrupt call }

begin
  Regs.ax := $4401;                        { Function number for IOCTL: Set Mode }
  Regs.bx := Handle;
  Regs.dx := GetMode(Handle) and 255 or 32; { new device attribute }
  MsDos( Regs );                          { Call DOS interrupt 21H }
end;

{*****}
{
* SETCOOKED : Changes a character driver into COOKED mode.
* Input      : Handle passed (must refer to addressed device)
* Output     : None
}
{*****}

procedure SetCooked(Handle : integer);

var Regs : Registers;                        { Register variable for interrupt call }

begin
  Regs.ax := $4401;                        { Function number for IOCTL: Set Mode }
  Regs.bx := Handle;
  Regs.dx := GetMode(Handle) and 223;      { New device attribute }
  MsDos( Regs );                          { Call DOS interrupt 21H }
end;

{*****}
{
* TESTOUTPUT : Displays a test string 1000 times on the standard
              output device.
* Input      : None
* Output     : None
}
{*****}

procedure TestOutput;

var Regs : Registers;                        { Register variable for interrupt call }
    LoopCnt : integer;                      { Loop variable }
    Test : string[9];                      { The test string for output }

begin

```

```

Test := 'Test....';
Regs.bx := STANDARDOUT;      { Output on the standard output device }
Regs.cx := 9;                { Number of characters }
Regs.ds := Seg(Test);        { Segment address of the text }
Regs.dx := Ofs(Test)+1;      { Offset address of the text }
for LoopCnt := 1 to 1000 do
begin
  Regs.ah := $40;             { Write function number for handle }
  MsDos( Regs );              { Call DOS interrupt 21H }
end;
writeln;
end;

{*****}
{*                MAIN PROGRAM                *}
{*****}

begin
  ClrScr;                     { Clear screen }
  writeln('RAWCOOK (c) 1987, 1992 by Michael Tischer'#13#10);
  writeln('Console driver is now in RAW mode. Control keys such as <Ctrl><C>');
  writeln('are not recognized during output. Press a key to display a text on');
  writeln('the screen, and try stopping the display by pressing <Ctrl><C>');
  Keys := ReadKey;            { wait for key }
  SetRaw(STANDARDIN);         { Console driver in RAW mode }
  TestOutput;                 { Output test string 1000 times }
  while KeyPressed do
  Keys := ReadKey;            { Empty keyboard buffer }
  writeln('The Console driver is now in COOKED mode. Control keys such as');
  writeln('<CTRL><C> are recognized during output');
  writeln('Press a key to start, then press <Ctrl><C> to stop the display');
  Keys := ReadKey;            { Wait for key }
  SetCooked(STANDARDIN);
  TestOutput;                 { Output test string 1000 times }
end.

```