PT600 Portable Terminal

Programming Reference Guide

Document number: 3506000120

Table of Contents

Chapter 1 BIOS and System Functions

The PT600 operating system supports DOS/BIOS function calls that a programmer can access when developing an application for the portable.

1.1 Interrupt Vector Assignment for I/Os

Vector	BIOS Function
09 H	Display Font
0F H	Kermit Server
10 H	LCD Control
21 H	System Functions
22 H	Power Manager
31 H	Beeper Control
33 H	RS232 Control

1.2 Display Font Functions (INT 09H)

00 Select Large Font

Entry Parameter: AH = 0/1 ; select 8*16-dot character font (4 lines * 16
; columns display)

Return Value: None

Example:

```
void TL_font(int status)
{
  regs.h.ah = (unsigned char)status;
  int86(0x09,&regs,&regs);
}
```

01 Select Small Font

Entry Parameter: AH = 1 ; select 6*8-dot character font (8 lines * 20 columns display)

Return Value: None

Example:

```
void TL_font(int status)
```

```
{
   regs.h.ah = (unsigned char)status;
   int86(0x09,&regs,&regs);
}
```

02 Set Font Type

3 Entry Parameter: AH= 2

4 AL= 0/1 ; set to large/small font

5 Return Value: None

Example:

```
void TL_font(int status)
{
  regs.h.ah = (unsigned char)status;
  regs.h.al = 2;
  int86(0x09,&regs,&regs);
}
```

03 Get Font Type

Entry Parameter: AH= 3

Return Value: AL= 0/1 ; large/small font

Example

```
int TL_get_font_type()
{
  regs.h.ah = 3;
  int86(0x09,&regs,&regs);
  return(regs.h.al);
}
```

04 Set User-Defined Font for All Characters

Entry Parameter: AH= 4

AL= 0/1 ; large/small font

DS:DX ; pointer to the buffer with font data

; (for large font: buffer size = 16*256 =4096 bytes

; for small font: buffer size = 6*256 =1536 bytes)

Return Value: None

Example

```
void TL_change_all_ASCII_font(int type,unsigned char *str)
{
  regs.h.ah=4;
  regs.h.al=(unsigned char)type;
  segregs.ds = FP_SEG(str);
  regs.x.dx = FP_OFF(str);
  int86x(0x9,&regs,&regs,&segregs);
}
```

05 Get Font Data for All Characters

Entry Parameter: AH= 5

AL= 0/1 ; large/small font

DS:DX ; pointer to the buffer
; (for large font: buffer size = 16*256 =4096 bytes
; for small font: buffer size = 6*256 =1536 bytes)

Return Value: Font data in the buffer

Example:

```
void TL_get_all_ASCII_font(int type,unsigned char *str)
{
  regs.h.ah=5;
  regs.h.al=(unsigned char)type;
  segregs.ds = FP_SEG(str);
  regs.x.dx = FP_OFF(str);
  int86x(0x9,&regs,&regs,&segregs);
}
```

06 Set User-Defined Font for One Character

Entry Parameter: AH= 6

AL= 0/1 ; large/small font
CL =0 - 255 ; character
DS:DX ; pointer to the buffer with font data
; (for large font: buffer size = 16 bytes
; for small font: buffer size = 6 bytes)

6 Return Value: None

Example:

```
void TL_change_one_ASCII_font(int type,int ascii_code,unsigned char
*str)
```

```
{
  regs.h.ah=6;
  regs.h.al=(unsigned char)type;
  regs.h.cl=(unsigned char)ascii_code;
  segregs.ds = FP_SEG(str);
  regs.x.dx = FP_OFF(str);
  int86x(0x9,&regs,&regs,&segregs);
}
```

07 Get Font Data for One Characters

Entry Parameter: AH= 7
AL= 0/1 ; large/small font
CL =0 - 255 ; character
DS:DX ; pointer to the buffer
; (for large font: buffer size = 16 bytes
; for small font: buffer size = 6 bytes)

Return Value: Font data in the buffer

Example:

```
void TL_get_one_ASCII_font(int type,int ascii_code,unsigned char *str)
{
  regs.h.ah=7;
  regs.h.al=(unsigned char)type;
  regs.h.cl=(unsigned char)ascii_code;
  segregs.ds = FP_SEG(str);
  regs.x.dx = FP_OFF(str);
  int86x(0x9,&regs,&regs,&segregs);
}
```

1.3 Kermit Server Function (INT 0FH)

Entry Parameter: None
Return Value: None

When INT 0FH is called in user application, PT600 will enter Kermit server. Pressing [CMD] key then [Exit] key to leave it and return to user application.

Example :

```
void TC_kermit_mode()
{
  int86(0x0F,&regs,&regs);
```

```
}
```

1.4 LCD Control Functions (INT 10H)

00 Clear LCD Screen

Entry Parameter: AH = 0
Return Value: None

Example:

```
void TL_clrscr()
{
  regs.h.ah= 0;
  int86(0x10,&regs,&regs);
}
```

01 Enable/Disable Screen Scroll

Entry Parameter: AH = 1
AL = 0/1 ; disable/enable screen scroll
Return Value: None

Example:

```
void TL_scroll(int status)
{
  regs.h.ah = 1;
  regs.h.al = (unsigned char)status;
        int86(0x10,&regs,&regs);
}
```

02 Set Cursor Position

Entry Parameter: AH = 2
DH = Row
DL = Column

	PT600	
	6x8	**8x16**
Row	0~7	0~3
Column	0~15	0~12

Return Value: None

Example :

```
void TL_gotoxy(int x,int y)
{
```

```
  regs.h.ah = 2;
  regs.h.dh = (unsigned char)y;
  regs.h.dl = (unsigned char)x;
  int86(0x10,&regs,&regs);
}
```

03 Get Cursor Position

Entry Parameter: AH = 3 ;
Return Value: DH = Row
DL = Column

Example :

```
void TL_getxy(int *x,int *y)
{
  regs.h.ah = 3;
  int86(0x10,&regs,&regs);
  *y = regs.h.dh;
  *x = regs.h.dl;
}
```

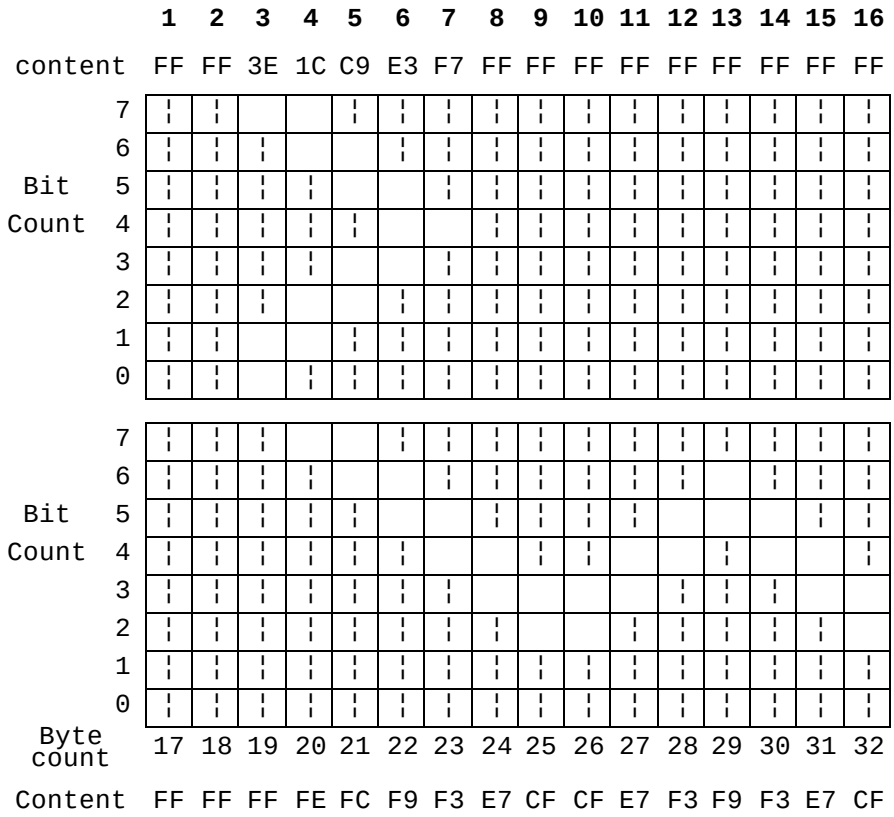
04 Display 16x16 Bitmap

Entry Parameter: AH = 4
DH = Row
DL = Column
DS:BX ; pointer to bitmap (32-bytes pattern data)
Return Value: None
Note: This function is available only in large font.

Example :

```
void TL_display_16x16_location(int x,int y,unsigned char *str)
{
    regs.h.ah=4;
    regs.h.dh=(unsigned char)y;
    regs.h.dl=(unsigned char)x;
    segregs.ds = FP_SEG(str);
    regs.x.bx = FP_OFF(str);
    int86x(0x10,&regs,&regs,&segregs);
}
```

Sample C program to display 16x16 graphic pattern.




```
#include <stdio.h>
#include <stdlib.h>
#include <dos.h>
#include <conio.h>

unsigned char logo[32]={0xFF,0xFF,0x3E,0x1C,0xC9,0xE3,0xF7,0xFF,
                        0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
                        0xFF,0xFF,0xFF,0xFE,0xFC,0xF9,0xF3,0xE7,
                                0xCF,0xCF,0xE7,0xF3,0xF9,0xF3,0xE7,0xCF};

void main()
{
  void far *buff;
  union REGS regs;
  struct SREGS sregs;

  regs.h.ah=0;
  int86(0x10, &regs, &regs);    /* clear screen */
```

```
regs.h.ah=0;
int86(0x9, &regs, &regs);      /* set large font */

regs.h.ah=5;
regs.h.al=0;
int86(0x10, &regs, &regs);    /* set cursor off */

buff=logo;
regs.h.ah=4;
regs.h.dh=1;       /* Row    */
regs.h.dl=2;       /* Column */
regs.x.bx=FP_OFF(buff);
sregs.ds=FP_SEG(buff);
int86x(0x10, &regs, &regs, &sregs);
getch();
}
```

05 Set Cursor On/Off

Entry Parameter: AH = 5
AL = 0/1 ; set cursor OFF/ON
Return Value: None

Example :

```
void TL_cursor_type(int status)
{
  regs.h.ah = 5;
  regs.h.al = (unsigned char)status;
  int86(0x10,&regs,&regs);
}
```

06 Enable/Disable Power-on Logo Display

Entry Parameter: AH = 6
AL = 0/1 ; disable/enable
Return Value: None

Example

```
void TL_cursor_type(int status)
{
  regs.h.ah = 5;
  regs.h.al = (unsigned char)status;
  int86(0x10,&regs,&regs);
}
```

0A Display Character

Entry Parameter: AH = 0AH
AL = 0 - 255 ; character to display
Return Value: None

Example

```
regs.h.ah = 0x0A;
regs.h.al = cc;
int86(0x10,&regs,&regs);
```

4F Display 16*16 Bitmap at Current Cursor Position

Entry Parameter: AH = 4FH
DS:BX ; pointer to bitmap (32-bytes pattern data)
Return Value: None
Note: This function is available only in large font.

Example :

```
void TL_display_16x16(unsigned char *str)
{
    regs.h.ah=0x4F;
    segregs.ds = FP_SEG(str);
    regs.x.bx = FP_OFF(str);
    int86x(0x10,&regs,&regs,&segregs);
}
```

1.5 System Functions (INT 21H)

00 Terminate Program

Entry Parameter: AH = 0
Return Value: None

Example:

```
void TS_exit_program()
{
    regs.h.ah= 0;
   int86(0x21,&regs,&regs);
}
```

01 Read Keypad (wait if no key) and Write to LCD

Entry Parameter: AH = 1
Return Value: AL = 0 – 255 ; ASCII character

Example:

```
unsigned char TS_stdin()
{
  regs.h.ah= 1;
  int86(0x21,&regs,&regs);
  return(regs.h.al);
}
```

02 Write LCD

Entry Parameter: AH = 2
DL = 0 – 255 ; ASCII character
Return Value: None

Example:

```
void TS_stdout(unsigned char ch)
{
  regs.h.ah= 2;
  regs.h.dl= ch;
  int86(0x21,&regs,&regs);
  return;
}
```

03 Read RS232 (wait if no character)

Entry Parameter: AH = 3
Return Value: AL = 0 – 255 ; ASCII character
Note: Only for NONE communication protocol

Example:

```
unsigned char TS_stdaux_in()
{
  regs.h.ah= 3;
  int86(0x21,&regs,&regs);
  return(regs.h.al);
}
```

04 Write RS232

Entry Parameter: AH = 4
DL = 0 – 255 ; ASCII character

Return Value: None
Note: Only for NONE communication protocol

Example:

```
void TS_stdaux_out(unsigned char ch)
{
  regs.h.ah= 4;
  regs.h.dl= ch;
  int86(0x21,&regs,&regs);
  return;
}
```

06 Direct Console I/O

Entry Parameter: AH = 6
DL = 0 – 254 ; write ASCII character in DL to LCD
255 ; read ASCII character from keypad
Return Value: 1) When read (DL <> 255):
Character in AL if ZERO flag is cleared
No input if ZERO flag is set
2) When write (DL= 255):
None

Example:

```
unsigned char TS_stdin_out(unsigned char ch)
{
  regs.h.ah= 6;
  regs.h.dl= ch;
  int86(0x21,&regs,&regs);
  if (ch == 0xFF)
  {
    if ((regs.x.cflag & 0x40) == 0) return(regs.h.al);
    else return(0);
  }
  return(0);
}
```

07 Read Keypad (wait if no key)

Entry Parameter: AH = 7
Return Value: AL = 0 – 255 ; ASCII character

Example:

```
unsigned char TS_stdin_noecho()
{
  regs.h.ah= 7;
  int86(0x21,&regs,&regs);
  return(regs.h.al);
}
```

08 Read Keypad (wait if no key)

Entry Parameter: AH = 8
Return Value: AL = 0 – 255 ; ASCII character

09 Write Character String to LCD

Entry Parameter: AH = 9

DS:DX ; pointer to string buffer

Return Value: None

Example:

```
void TS_stdout_string(unsigned char *str)
{
  segregs.ds = FP_SEG(str);
  regs.x.dx = FP_OFF(str);
  regs.h.ah= 9;
  int86x(0x21,&regs,&regs,&segregs);
  return;
}
```

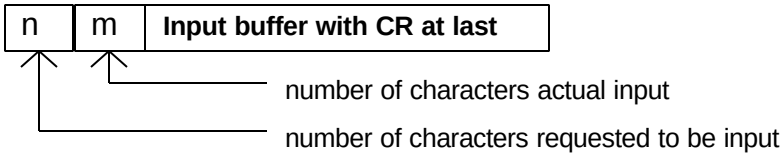
0A Buffered Keypad Input

Entry parameter: AH = 0AH

DS:DX ; pointer to data buffer

Return value: DS:DX ; pointer to data buffer

Data format in buffer:



Example:

```
void TS_stdin_string(unsigned char *str)
{
  segregs.ds = FP_SEG(str);
  regs.x.dx = FP_OFF(str);
  regs.h.ah= 0x0a;
  int86x(0x21,&regs,&regs,&segregs);
  return;
}
```

0B Check Keypad Status

Entry Parameter: AH = 0BH

Returned Value: AL = 0 ; no keys were pressed

0FFH ; key was pressed and input character is Ready

Example:

```
unsigned char TS_kbhit()
{
```

```
    regs.h.ah= 0x0b;
    int86(0x21,&regs,&regs);
    return(regs.h.al);
  }
```

1A LCD Backlight ON/OFF Control

Entry Parameter: AH = 1AH
BH = 0
AL = 0/1 ; set LCD backlight OFF/ON

Return Value: None

Example:

```
  void TL_backlight(int status)
  {
    regs.h.ah= 0x1A;
    regs.h.bh= 0;
    regs.h.al= (unsigned char)status;
    int86(0x21,&regs,&regs);
  }
```

1A Buzzer ON/OFF Control

Entry Parameter: AH = 1AH
BH = 1
AL = 0/1 ; set Buzzer OFF/ON

Return Value: None

Example:

```
  void TD_buzzer(int status)
  {
    regs.h.ah= 0x1A;
    regs.h.bh= 1;
    regs.h.al= (unsigned char)status;
    int86(0x21,&regs,&regs);
  }
```

1A Beeper Volume

Entry parameter: AH = 1AH
BH = 3
AL = 0/1/2 ; set LOW/MEDIUM/HIGH beeper volume

Return Value: None

=2 high

Example:

```
void TD_beeper_vol(int status)
{
  regs.h.ah= 0x1A;
  regs.h.bh= 3;
  regs.h.al= (unsigned char)status;
  int86(0x21,&regs,&regs);
}
```

1A Enable/Disable RS232 Port

Entry Parameter: AH = 1AH
BH = 4
AL = 0/1 ; disable/enable
Return Value: None

Example:

```
void TD_serial(int status)
{
  regs.h.ah= 0x1A;
  regs.h.bh= 4;
  regs.h.al= (unsigned char)status;
  int86(0x21,&regs,&regs);
}
```

1A Enable/Disable Key or Key Function

Entry Parameter: AH = 1AH
BH = 5
AL = 0 ; All keys
1 ; Supervisor mode
2 ; Cold start
3 ; Warm start
4 ; User mode (press [CMD] for 2 seconds)
5 ; ALPHA key
BL = 0 ; Disable key or key function
1 ; Enable key or key function
Return Value: None

Example:

```
void TD_keylock(int type,int status)
{
  regs.h.ah= 0x1A;
```

```
    regs.h.bh= 5;
    regs.h.al= (unsigned char)type;
    regs.h.bl= (unsigned char)status;
    int86(0x21,&regs,&regs);
}
```

1A Set Keypad Language

Entry Parameter: AH = 1AH

BH = 7

AL = 0 ; English

1 ; Sweden / Finland

2 ; Danish

3 ; Spanish

4 ; French

5 ; German

6 ; Italian

Return Value: None

Note: The functions defines input characters in ALPHA mode for key [0]:

Language	**Mode**	**Hex Code**	**Display**
0.English	ALPHA-1	5B	@
	ALPHA-2	5C	?
	ALPHA-3	5D	&
1.Sweden/Finland	ALPHA-1	8F	Å
	ALPHA-2	8E	Ä
	ALPHA-3	99	Ö
2.Danish	ALPHA-1	92	Æ
	ALPHA-2	9D	Ø
	ALPHA-3	8F	Å
3.Spanish	ALPHA-1	AD	¡
	ALPHA-2	A5	Ñ
	ALPHA-3	A8	¿
4.French	ALPHA-1	F8	º
	ALPHA-2	87	Ç
	ALPHA-3	26	&
5.German	ALPHA-1	8E	Ä
	ALPHA-2	99	Ö
	ALPHA-3	9A	Ü
6.Italian	ALPHA-1	F9	•
	ALPHA-2	5C	\
	ALPHA-3	82	É

'¥' (9DH) in ASCII is replaced with 'Ø' when Danish is selected.

Example:

```
void TD_key_language(int status)
{
  regs.h.ah= 0x1A;
  regs.h.bh= 7;
  regs.h.al= (unsigned char)status;
  int86(0x21,&regs,&regs);
}
```

1A Check Main Battery Status

Entry Parameter: AH = 1AH
BH = 8
Return Value: AL = 0/1 ; Normal / Battery low

Example:

```
int TS_battery()
{
  regs.h.ah= 0x1A;
  regs.h.bh= 8;
  int86(0x21,&regs,&regs);
  return(regs.h.al);
}
```

1A Check Backup Battery Status

Entry Parameter: AH = 1AH
BH = 9
Return Value: AL = 0/1 ; Normal / Battery Low

Example:

```
int TS_lithium_battery()
{
  regs.h.ah= 0x1A;
  regs.h.bh= 9;
  int86(0x21,&regs,&regs);
  return(regs.h.al);
}
```

1A Set Good-read LED

Entry Parameter: AH = 1AH

BH = 0AH

AL = 0 ; Set the Good-read LED (green light) OFF

1 ; Set the Good-read LED (green light) ON

2 ; Set the Good-read LED controlled by system

Return Value: None

Note: If the function is called by AL=0 or AL=1, the system will not control Good-read LED ON/OFF when a bar code label is decoded successfully.

Example:

```
void TS_bar_good_read(int status)
{
  regs.h.ah= 0x1A;
  regs.h.bh= 0x0A;
  regs.h.al= (unsigned char)status;
  int86(0x21,&regs,&regs);
}
```

1A Set Laser Scanner Trigger Mode

Entry Parameter: AH = 1AH
BH = 0BH
AL = 0/1 ; Normal / Flash mode

Return Value: None

Example:

```
void TD_flash_trigger(int status)
{
  regs.h.ah= 0x1A;
  regs.h.bh= 0x0B;
  regs.h.al= (unsigned char)status;
  int86(0x21,&regs,&regs);
}
```

1A Enable/Disable Double Verification When Read Bar Code Label

Entry Parameter: AH = 1AH
BH = 0CH
AL = 0/1 ; Disable/Enable double verification

Return Value: None

Example:

```
void TD_double_verify_bar(int type)
{
  regs.h.ah = 0x1A;
  regs.h.bh = 0x0C;
  regs.h.al = (unsigned char)type;
  int86(0x21,&regs,&regs);
}
```

1A Check Laser Scanner

Entry Parameter: AH = 1AH

BH = 0DH
Return Value: AL = 0 ; Has no built-in or clip on laser scanner
1 ; Has built-in or clip on laser scanner

Example:

```
int TD_check_scanner()
{
  regs.h.ah = 0x1A;
  regs.h.bh = 0x0D;
  int86(0x21,&regs,&regs);
  return(regs.h.al);
}
```

1A Support Aim Mode for SE-1200 family laser engines

a) Manual Setting
USER MENU/ 8. SET/ 2. SCANNER /LASER AIM :
Selection: Disable /Enable (Default = Disable)

b) Fuction Call (INT 0x21/AH=0x1A/BH=0x0F)
Call: AL = 0/1: disable/enable
Return: none

1A Support Keypad Input in Lower/Upper Case

a) Manual setting
USER MENU/ 8. SET/ 4. KEYPAD / ALPHA CHARACTER
Selection: Default ; as defined by keymap
Lower ; always in lower case
Upper ; always in upper case

b) Fuction Call (INT 0x21/AH=0x1A/BH=0x11)
Call: AL = 0/1/2: ; default /lower/upper
Return: none

1B Get Scanner Port Status

Entry Parameter: AH = 1BH
BH = 5
Return values: AL = 0 ; Scanner port is disabled
1 ; Scanner port is enabled

Example:

```
int TD_get_bar_type()
{
  regs.h.ah = 0x1B;
  regs.h.bh = 5;
  int86(0x21,&regs,&regs);
  return(regs.h.al);
}
```

1B Get MULTIPOINT Address

Entry Parameter: AH = 1BH
BH = 6

Returned Value: AL = Address ; ASCII character ‘A’ - ’Y’ or ’0’ - ’6’

Example:

```
char TC_get_address()
{
  regs.h.ah = 0x1B;
  regs.h.bh = 6;
  int86(0x21,&regs,&regs);
  return((char)regs.h.al);
}
```

1C Set Communication Parameters

Entry Parameter: AH = 1CH
BH = 1
AL bits 7-4: 0001xxxx baud 150
0010xxxx baud 300
0011xxxx baud 600
0100xxxx baud 1200
0101xxxx baud 2400
0110xxxx baud 4800
0111xxxx baud 9600

1000xxxx baud 19200
1001xxxx baud 38400
1010xxxx baud 57600
bits 3-2: xxxx00xx none parity
xxxx01xx odd parity
xxxx11xx even parity
bit 1: xxxxxx0x one stop bit
xxxxxx1x two stop bits
bit 0: xxxxxxx0 7 data bits
xxxxxxx1 8 data bits

Return Value: None

Example:

```
int TC_232_parameter(long baud,int parity,int stop,int data)
{
  unsigned char cc=0;
  unsigned int i_baud;

  i_baud = (int)(baud / 10L);
  switch (i_baud)
  {
   case   11 : cc=0x00; break;
   case   15 : cc=0x10; break;
   case   30 : cc=0x20; break;
   case   60 : cc=0x30; break;
   case  120 : cc=0x40; break;
   case  240 : cc=0x50; break;
   case  480 : cc=0x60; break;
   case 1920 : cc=0x80; break;
   case 3840 : cc=0x90; break;
   case 5760 : cc=0xA0; break;
   default: cc=0x70;  break;
  }
  switch (parity)
```

```
    {
      case 0 : break;
      case 1 : cc=cc|0x04; break;
      case 2 : cc=cc|0x0c; break;
    }
    switch (stop)
    {
      case 1 : break;
      case 2 : cc=cc|0x02; break;
    }
    switch (data)
    {
      case 7 : break;
      case 8 : cc=cc|0x01; break;
    }
    regs.h.ah = 0x1C;
    regs.h.bh = 1;
    regs.h.al  = cc;
    int86(0x21,&regs,&regs);
  }
```

1C Software Control Flow

Entry Parameter: AH = 1CH
BH = 2
AL = 0 ; Enable XON/XOFF control flow
1 ; Disable XON/XOFF control flow
Return Value: None
Note: Only for NONE communication protocol

Example:

```
void TC_flow_ctrl(int status)
{
  if (status == 0)              // Set flow control to none
  {
        regs.h.ah = 0x1C;
      regs.h.bh = 2;
      regs.h.al  = 1;
      int86(0x21,&regs,&regs);

        regs.h.ah = 0x1C;
      regs.h.bh = 3;
      regs.h.al  = 1;
      int86(0x21,&regs,&regs);
```

```
    }
    else if (status == 1)
    {
            regs.h.ah = 0x1C;
        regs.h.bh = 2;
        regs.h.al  = 0;
        int86(0x21,&regs,&regs);
    }
    else
    {
            regs.h.ah = 0x1C;
        regs.h.bh = 2;
        regs.h.al  = 1;
        int86(0x21,&regs,&regs);
  }
```

1C Hardware Control Flow

Entry parameters: AH = 1CH
BH = 3
AL = 0 ; Enable CTS/RTS control flow
1 ; Disable CTS/RTS control flow
Return Value: None
Note: Only for NONE communication protocol

1C Set Communication Protocol

Entry Parameter: AH = 1CH
BH = 4
AL = 2 ; Set to MULTIPOINT protocol
3 ; Set to NONE protocol
Return Value: None

Example:

```
  void TC_protocol(int status)
  {
      regs.h.ah = 0x1C;
      regs.h.bh = 4;
      regs.h.al = (unsigned char)status;
      int86(0x21,&regs,&regs);
  }
```

1C Get MULTIPOINT Address

Entry Parameter: AH = 1CH
BH = 5
Returned Value: AL = Address ; ASCII character 'A' - 'Y' or '0' - '6'

Example:

```
char TC_get_address()
{
  regs.h.ah = 0x1C;
  regs.h.bh = 5;
  int86(0x21,&regs,&regs);
  return((char)regs.h.al);
}
```

1C Set MULTIPOINT Address

Entry Parameter: AH = 1CH
BH = 6
AL = Address ; ASCII character 'A' - 'Y' or '0' - '6'
Return Value: AL = 0 ; if successful
1 ; if failed

Example:

```
int TC_set_address(char status)
{
   regs.h.ah = 0x1C;
   regs.h.bh = 6;
   regs.h.al = status;
   int86(0x21,&regs,&regs);
   return((char)regs.h.al);
}
```

1C Set File-Uploading Message ON/OFF

Entry Parameter: AH = 1CH
BH = 0AH
AL = 0 ; Not display message while uploading file
AL = 1 ; Display message while uploading file
Return Value: None

Example:

```
void TD_upload_message(int status)
{
```

```
    regs.h.ah= 0x1C;
    regs.h.bh= 0x0A;
    regs.h.al= (unsigned char)status;
    int86(0x21,&regs,&regs);
  }
```

1D Set Terminal ID

Entry Parameter: AH = 1DH
BH = 0
DS:DX ; pointer to buffer of 8 characters ID string

Return Value: None

Example:

```
  void TT_id(unsigned char *str)
  {
    segregs.ds = FP_SEG(str);
    regs.x.dx = FP_OFF(str);
    regs.h.ah=0x1D;
    regs.h.bh=0;
    int86x(0x21,&regs,&regs,&segregs);
    return(regs.h.al);
  }
```

1D Set ONLINE/LOCAL in Terminal Mode

Entry Parameter: AH = 1DH
BH = 1
AL = 0/1 ; ONLINE/LOCAL

Return Value: None

Example:

```
  void TT_online_local(int status)
  {
      regs.h.ah=0x1D;
      regs.h.bh=1;
      regs.h.al=(unsigned char)status;
      int86x(0x21,&regs,&regs,&segregs);
  }
```

1D Set ECHO ON/OFF in Terminal Mode

Entry Parameter: AH = 1DH
BH = 2
AL = 0/1 ; ECHO ON/OFF
Return Value: None

Example:

```
void TT_echo(int status)
{
    regs.h.ah=0x1D;
    regs.h.bh=2;
    regs.h.al=(unsigned char)status;
    int86x(0x21,&regs,&regs,&segregs);
}
```

1D Set AUTOLF ON/OFF in Terminal Mode

Entry Parameter: AH = 1DH
BH = 3
AL = 0/1 ; AUTOLF ON/OFF
Return Value: None

Example:

```
void TT_auto_LF(int status)
{
  regs.h.ah= 0x1D;
  regs.h.al= (unsigned char)status;
  regs.h.bh= 3;
  int86(0x21,&regs,&regs);
}
```

1D Set CHARACTER/BLOCK in Terminal Mode

Entry Parameter: AH = 1DH
BH = 4
AL = 0 ; CHARACTER mode
1 ; BLOCK mode
DX = 0 ; block = LINE
1 ; block = PAGE
2 ; block = LINE or PAGE
Return Value: None

Example:

```
void TT_mode(int status,int status1)
{
  regs.h.ah= 0x1D;
  regs.h.bh= 4;
```

```
  regs.h.al= (unsigned char)status;
  regs.x.dx = (unsigned char)status1;
  int86(0x21,&regs,&regs);
}
```

1D Define LINE Character in Terminal Mode

Entry parameters: AH = 1DH
BH = 5
AL = 0 – 255 ; ASCII character
Return Value: None

Example:

```
void TT_line_terminal(unsigned char status)
{
  regs.h.ah= 0x1D;
  regs.h.bh= 5;
  regs.h.al = status;
  int86(0x21,&regs,&regs);
}
```

1D Define Page Character in Terminal Mode

Entry Parameter: AH = 1DH
BH = 6
AL = 0 – 255 ; ASCII character
Return Value: None

Example:

```
void TT_page_terminal(unsigned char status)
{
  regs.h.ah= 0x1D;
  regs.h.bh= 6;
  regs.h.al= status;
  int86(0x21,&regs,&regs);
}
```

1E Set User-defined Key-map

Entry Parameter: AH = 1EH
AL = 0
DS:DX ; pointer to Key-map buffer with 5*24
; characters corresponding to keypad in
; 5 input modes
Return Value: None

Example:

```
void TD_key_map(unsigned char *str)
{
  segregs.ds = FP_SEG(str);
  regs.x.dx = FP_OFF(str);
  regs.h.ah=0x1E;
  regs.h.al=0;
  int86x(0x21,&regs,&regs,&segregs);
}
```

Note: Default system key-map:

Non-ALPHA Mode ALPHA-1 Mode ALPHA-2 Mode ALPHA-3

byte Seq.	key name	hex code	byte Seq.	key name	hex code	byte Seq.	key name	hex code	byte Seq.	key name	hex code	byte Seq.	key name	hex code
0	CMD	…	24	CMD	…	48	CMD	...	72	CMD	…	96	CMD	...
1	F1	86	25	F1	86	49	F1	86	73	F1	86	97	F5	8A
2	ALPHA	…	26	ALPHA	…	50	ALPHA	…	74	ALPHA	...	98	ALPHA	...
3	7	37	27	A	41	51	B	42	75	C	43	99	?	…
4	4	34	28	J	4A	52	K	4B	76	L	4C	100	:	3A
5	1	31	29	S	53	53	T	54	77	U	55	101	-	2D
6	CLR	08	30	CLR	08	54	CLR	08	78	CLR	08	102	CLR	08
7	…	…	31	...	...	55	…	…	79	…	…	103	...	...
8	F4	89	32	F4	89	56	F4	89	80	F4	89	104	F8	8D
9	F2	87	33	F2	87	57	F2	87	81	F2	87	105	F6	8B
10		11	34		11	58		11	82		11	106		13
11	8	38	35	D	44	59	E	45	83	F	46	107	?	…
12	5	35	36	M	4D	60	N	4E	84	Q	4F	108	=	3D
13	2	32	37	V	56	61	W	57	85	X	58	109	+	2B
14	0	30	38	[	5B	62	\	5C	86	]	5D	110	SP	20
15	…	…	39	…	…	63	…	…	87	…	…	111	…	…
16	.	2E	40	.	2E	64	.	2E	88	.	2E	112	EXIT	84
17	F3	88	41	F3	88	65	F3	88	89	F3	88	113	F7	8C
18		10	42		10	66		10	90		10	114		12
19	9	39	43	G	47	67	H	48	91	I	49	115	?	…
20	6	36	44	P	50	68	Q	51	92	R	52	116	/	2F
21	3	33	45	Y	59	69	Z	5A	93	#	23	117	*	2A
22	ENTER	0D	46	ENTER	0D	70	ENTER	0D	94	ENTER	0D	118	ENTER	0D
23	…	…	47	…	…	71	…	…	95	…	…	119	…	…

Mode CMD Mode

? : LCD Backlight
? : LCD Contrast
? : Beeper Volume

Keyname and Hexcode with ... means DON'T CARE.

1E Get System Key-map

Entry Parameter: AH = 1EH
AL = 1
DS:DX ; pointer to 120 bytes buffer
Return Value: Key-map in buffer

Example:

```
void TD_get_key_map(unsigned char *str)
{
  segregs.ds = FP_SEG(str);
  regs.x.dx = FP_OFF(str);
```

```
    regs.h.ah=0x1E;
    regs.h.al=1;
    int86x(0x21,&regs,&regs,&segregs);
    return(regs.h.al);
  }
```

1F Enable/Disable Decoding of a Barcode Symbology

Entry Parameter: AH = 1FH

BH = 1

AL = 1 ; Enable decoding of barcode symbology

0 ; Disable decoding of barcode symbology

BL = 0 ; All supported bar code symbologies

1 ; Code 39

2 ; I 2 of 5

3 ; Codabar

4 ; EAN/UPC

5 ; Code 128

6 ; EAN 128

Return Value: None

Example:

```
void TD_set_decode_status(int type,int status)
{
  regs.h.ah= 0x1F;
  regs.h.al= (unsigned char)status;
  regs.h.bh= 1;
  regs.h.bl= (unsigned char)type;
  int86(0x21,&regs,&regs);
}
```

1F Get Decoding Status of a Barcode Symbology

Entry parameter: AH = 1FH

BH = 2

BL = 1 ; Code 39

2 ; I 2 of 5

3 ; Codabar

4 ; EAN/UPC

5 ; Code 128

6 ; EAN 128

Return Value: AL = 0/1 ; Disable/Enable

Example:

```
int TD_get_decode_bar(int type)
{
  regs.h.ah = 0x1F;
  regs.h.bh = 2;
  regs.h.bl = (unsigned char)type;
  int86(0x21,&regs,&regs);
  return(regs.h.al);
}
```

1F Code 39 Settings

Entry parameter: AH = 1FH

BH = 3

BL = 1

AL bit 0= 0/1 ; disable/enable Code 39 decoding

1= 0/1 ; disable/enable Check Digit verification

2= 0/1 ; no-send/send Check Digit

3= 0/1 ; no-send/send Start/Stop characters

4= 0/1 ; Full ASCII OFF/ON

Return Value: None

Example:

```
void TD_decoder_setting(int decoder_type,int setting)
{
  regs.h.ah = 0x1F;
  regs.h.bh = 3;
  regs.h.bl = (unsigned char)decode_type;
  regs.h.al = (unsigned char)setting;
  int86(0x21,&regs,&regs);
}
```

1F Interleaved 2 of 5 Settings

Entry Parameter: AH = 1FH

BH = 3

BL = 2

AL bit 0= 0/1 ; disable/enable Interleaved 2 of 5 decoding
1= 0/1 ; disable/enable Check Digit verification
2= 0/1 ; no-send/send Check Digit
Return Value: None

Example:
Same with Code 39's example:

1F Codabar Settings
Entry Parameter: AH = 1FH
BH = 3
BL = 3
AL bit 0= 0/1 ; disable/enable Codebar decoding
1= 0/1 ; disable/enable Check Digit verification
2= 0/1 ; no-send/send Check Digit
3= 0/1 ; no-send/send Start/Stop characters
Return Value: None

1F Code 128 Setting
Entry Parameter: AH = 1FH
BH = 3
BL = 5
AL bit 0= 0/1 : disable/enable Code 128 decoding
Return Value: None

Example:
Same with Code 39's example:

1F EAN 128 Setting
Entry Parameter: AH = 1FH
BH = 3
BL = 6
AL bit 0= 0/1 ; disable/enable EAN 128 decoding

Return Value: None

Example:
Same with Code 39's example:

1F Code 93 Setting

Entry Parameter: AH = 1FH
BH = 3
BL = 7
AL bit 0= 0/1 ; disable/enable Code 93 decoding

Return Value: None

Example:
Same with Code 39's example:

1F UPC-A Settings

Entry Parameter: AH = 1FH
BH = 3
BL = 11H
AL bit 0= 0/1 ; disable/enable UPC-A decoding
2= 0/1 ; no-send/send Check Digit
3= 0/1 ; no-send/send Leading Digit

Return Value: None

Example:
Same with Code 39's example:

1F UPC-E Settings

Entry parameter: AH = 1FH
BH = 3
BL = 12H
AL bit 0= 0/1 ; disable/enable UPC-E decoding
2= 0/1 ; no-send/send Check Digit
3= 0/1 ; no-send/send Leading Digit
4= 0/1 ; disable/enable Zero Expansion

Return Value: None

Example:
Same with Code 39's example:

1F EAN-13 Settings

Entry Parameter: AH = 1FH
BH = 3
BL = 13H
AL bit 0= 0/1 ; disable/enable EAN-13 decoding
2= 0/1 ; no-send/send Check Digit
3= 0/1 ; no-send/send Leading Digit

Return Value: None

Example:
Same with Code 39's example:

1F EAN-8 Settings

Entry Parameter: AH = 1FH
BH = 3
BL = 14H
AL bit 0= 0/1 ; disable/enable UPC-E decoding
2= 0/1 ; no-send/send Check Digit

Return Value: None

Example:
Same with Code 39's example:

1F User Code 1 Setting (TRIOPTIC Code)

Entry Parameter: AH = 1FH
BH = 3
BL = 21H
AL bit 0= 0/1 : disable/enable User Code 1 decoding

Return Value: None

Example:
Same with Code 39's example:

1F User Code 2 Setting (TOSHIBA code)

Entry Parameter: AH = 1FH
BH = 3
BL = 22H
AL bit 0= 0/1 : disable/enable User Code 2 decoding

Return Value: None

Example:
Same with Code 39's example:

25 Set Interrupt Vector

Entry Parameter: AH = 25H
AL = interrupt number
DS:BX = address of interrupt routine

Return Value: None

Example:

```
void TS_set_interrupt_vector(int vect,unsigned int ds,unsigned int dx)
{
  regs.h.ah= 0x25;
  regs.h.al= (unsigned char)vect;
  segregs.ds=ds;
  regs.x.dx=dx;
  int86x(0x21,&regs,&regs,&segregs);
}
```

2A Get System Date

Entry Parameter: AH = 2AH

Return Value: CX = year (1980 - 2079)
DH = month (1 - 12)
DL = day (1 - 31)
AL = weekday (0 - 6)

Example:

```
void TS_get_date(int *year,int *month,int *day,int *week)
{
  TD_int_dos1(0x2a,0,0,0);
  *year = regs.x.cx;
  *month = regs.h.dh;
  *day = regs.h.dl;
  *week = regs.h.al;
}
```

2B Set System Date

Entry Parameter: AH = 2BH
CX = year (1980 - 2079)
DH = month (1 - 12)
DL = day (1 - 31)
Return Value: AL = 0/FFH ; OK/Setting error

Example:

```
int TS_set_date(int year,int month,int day)
{
  regs.h.ah = 0x2b;
  regs.x.cx = year;
  regs.h.dh = month;
  regs.h.dl = day;
  int86(0x21,&regs,&regs);
      return((int)regs.h.al);
}
```

2C Get System Time

Entry Parameter: AH = 2CH
Return Value: CH = hour (0 - 23)
CL = minute (0 - 59)
DH = second (0 - 59)

Example:

```
void TS_get_time(int *hour,int *minute,int *second,int *mini_sec)
{
  TD_int_dos1(0x2c,0,0,0);
  *hour = (int)regs.h.ch;
  *minute = (int)regs.h.cl;
  *second = (int)regs.h.dh;
  *mini_sec = (int)regs.h.dl;
```

```
}
```

2D Set System Time

Entry Parameter: AH = 2DH
CH = hour (0 - 23)
CL = minute (0 - 59)
DH = second (0 - 59)
Return Value: AL = 0/FFH ; OK/Setting error

Example:

```
int TS_set_time(int hour,int minute,int second)
{
  regs.h.ah = 0x2d;
  regs.h.ch = hour;
  regs.h.cl = minute;
  regs.h.dh = second;
  int86(0x21,&regs,&regs);
        return((int)regs.h.al);
}
```

2E Set Alarm Date

Entry Parameter: AH = 2EH
AL = 0 ; disable alarm
1 ; enable alarm everyday
2 ; enable alarm by date
If AL = 2:
CX = year (1980 - 2079)
DH = month (1 - 12)
DL = day (1 - 31)
Return Value: AL = 0/FFH ; OK/Setting error

Example:

```
int TS_alarm_date(int status,int year,int month,int day)
{
  regs.h.ah = 0x2E;
  regs.h.al = (unsigned char)status;
  regs.x.cx = year;
  regs.h.dh = (unsigned char)month;
  regs.h.dl = (unsigned char)day;
  int86(0x21,&regs,&regs);
```

```
  return((int)regs.h.al);
}
```

2F Set Alarm Time

Entry Parameter: AH = 2FH
CH = hour (0 - 23)
CL = minute (0 - 59)
DH = second (0 - 59)
Return Value: AL = 0/FFH ; OK/Setting error

Example:

```
int TS_alarm_time(int hour,int minute,int second)
{
  regs.h.ah = 0x2F;
  regs.h.ch = hour;
  regs.h.cl = minute;
  regs.h.dh = second;
  int86(0x21,&regs,&regs);
  return((int)regs.h.al);
}
```

30 Get DOS and Firmware Version Number

Entry Parameter: AH = 30H
AL = 0/1 ; with/without OEM version code
Return Value: if AL = 0 when call:
AL= major DOS version number (=2)
AH= minor DOS version number (=10)
CL= major firmware version number
CH= minor firmware version number
BX= OEM firmware version code (=0 for standard version)
if AL = 1 when call:
AL= major DOS version number (=2)
AH= minor DOS version number (=10)
CL= major firmware version number
CH= minor firmware version number
BX= 810H

Example:

```
int TS_version1(int *ver,int *firm,int *oem)
```

```
{
    regs.h.ah= 0x30;
    regs.h.al= 0;
    int86(0x21,&regs,&regs);
    *ver = regs.h.al * 100 + regs.h.ah;
    *firm= regs.h.cl * 100 + regs.h.ch;
    *oem = regs.x.bx;
    return(regs.h.al * 100 + regs.h.ah);
}
```

35 Get Interrupt Vector

Entry Parameter: AH = 35H
AL = interrupt number
Return Value: DS:BX = address of interrupt routine

Example:

```
void TS_get_interrupt_vector(int vect,unsigned int *es,unsigned int *bx)
{
  regs.h.ah= 0x35;
  regs.h.al= (unsigned char)vect;
  int86x(0x21,&regs,&regs,&segregs);
  *es = segregs.es;
  *bx = regs.x.bx;
}
```

36 Get Free Disk Cluster

Entry Parameter: AH = 36H
Return Value: AX = 1 (number of sectors per cluster)
BX = number of available clusters
CX = 1024 (number of bytes per sector)
DX = number of total clusters in RAM disk

Example:

```
long TS_free_disk(int drive)
{
  regs.h.al = (unsigned char)drive;
  regs.h.ah = 0x36;
  int86(0x21,&regs,&regs);
  return((long)regs.x.bx*(long)regs.x.cx);
}
```

3C Create File with Handle

Entry Parameter: AH = 3CH
DS:DX ; pointer to file name buffer
Return Value: if success, AX = Handle and CARRY flag is cleared
if failed, AX = 3 and CARRY flag is set
Note: If the file already exists, the function will truncate it to zero length.

Example:

```
int TS_create_file(char *fn)
{
  regs.h.ah=0x3Ch;
  segregs.ds = FP_SEG(fn);
  regs.x.dx = FP_OFF(fn);
  int86x(0x21,&regs,&regs,&segregs);
  if ((regs.x.cflag & 0x01) == 1) return(-1);
  else return(regs.x.ax);
}
```

3D Open File with Handle

Entry Parameter: AH = 3DH
AL = 0 ; read only
1 ; write only
2 ; read and write
DS:DX ; pointer to file name buffer
Return Value: if success, AX = Handle and CARRY flag is cleared
if failed , AX = 2 and CARRY flag is set

Example:

```
int TS_open_file(char *fn,int mode)
{
  regs.h.ah=0x3Dh;
  regs.h.al=(unsigned char)mode;
  segregs.ds = FP_SEG(fn);
  regs.x.dx = FP_OFF(fn);
  int86x(0x21,&regs,&regs,&segregs);
  if ((regs.x.cflag & 0x01) == 1) return(-1);
  else return(regs.x.ax);
}
```

3E Close File with Handle

Entry Parameter: AH = 3EH
BX = Handle of file
Return Value: if success, CARRY flag is cleared
if failed, CARRY flag is set

Example:

```
void TS_close_file(int hdl)
{
  regs.h.ah= 0x3e;
  regs.x.bx= hdl;
  int86(0x21,&regs,&regs);
  if ((regs.x.cflag & 0x01) == 1) return(-1);
  else return(1);
}
```

3F Read File with Handle

Entry Parameter: AH = 3FH
BX = Handle of file
CX = number of bytes to read
DS:DX ; pointer to data buffer
Return Value: if success, AX = number of bytes read and CARRY flag is cleared,
data in buffer
if failed, AX = 6 and CARRY flag is set

Example:

```
int TS_read_file(int hdl,int cnt,char *str)
{
  segregs.ds = FP_SEG(str);
  regs.x.dx = FP_OFF(str);
  regs.h.ah=0x3f;
  regs.x.cx=cnt;
  regs.x.bx=hdl;
  int86x(0x21,&regs,&regs,&segregs);
  if ((regs.x.cflag & 0x01) == 0) return(regs.x.ax);
  else return(-1);
}
```

40 Write File with Handle

Entry Parameter: AH = 40H
BX = Handle of file
CX = number of bytes to write
DS:DX ; pointer to data buffer
Returned Value: if success, AX = number of written bytes and CARRY flag is cleared
if failed, AX = 6 and CARRY flag is set

Example:

```
int TS_write_file(int hdl,int cnt,char *str)
{
  segregs.ds = FP_SEG(str);
  regs.x.dx = FP_OFF(str);
  regs.h.ah=0x40;
  regs.x.bx=hdl;
  regs.x.cx=cnt;
  int86x(0x21,&regs,&regs,&segregs);
  if ((regs.x.cflag & 0x01) == 0) return(regs.x.ax);
  else return(-1);
}
```

41 Delete File

Entry Parameter: AH = 41H
DS:DX ; pointer to file name buffer
Return Value: if success, CARRY flag is cleared
if failed, AX = 2 and CARRY flag is set

Example:

```
int TS_delete_file(char *fn)
{
  TD_intdos_O(0x41,0,fn);
  if ((regs.x.cflag & 0x01) == 0) return(1);
  else return(-1);
}
```

42 Move File Pointer

Entry Parameter: AH = 42H
AL = 0 ; offset from beginning
1 ; offset from current
2 ; offset from end

BX = Handle of file
CX = most significant half of 32 bits offset
DX = least significant half of 32 bits offset

Return Value: if success, CARRY flag is cleared
AX = least significant half of new current position
DX = most significant half of new current position
if failed, CARRY flag is set
AX = 6

Example:

```
long TS_seek_file(int hdl,int type,long loc)
{
  union LONG_III aa;

  regs.h.ah=0x42;
  regs.h.al=(unsigned char)type;
  regs.x.bx=hdl;
  aa.l.ll = loc;
  regs.x.cx=aa.i.ii2;
  regs.x.dx=aa.i.ii1;
  int86(0x21,&regs,&regs);
  aa.i.ii2=regs.x.dx;
  aa.i.ii1=regs.x.ax;
  if ((regs.x.cflag & 0x01) == 0) return(aa.l.ll);
  else return(-1L);
}
```

42. Search Character Beginning at the Current File Position

Entry Parameter: AH = 42H
AL = 3 ; search forward (to the end of file)
4 ; search backward (to the beginning of file)
BX = file handle
CX = n ; search nth matched character
DL = character

Return Value: 1) If character is found:
Carry flag = clear
DX:AX = pointer to current file position (at the position of
matched character)
2) If character is not found:
Carry flag = set

CX = total matched times
DX:AX = pointer to current file position (not changed)

42 Search String in Formatted Data File Beginning at the Current Position

Entry Parameter: AH = 42H
AL = 5 ; search forward (to the end of file)
6 ; search backward (to the beginning of file)
BX = file handle
CH = n ; Total field number in the data record
CL = m ; search mth field
DS:DX = pointer to parameter block
Structure of parameter block for non-fixed length record data file:
String length : 1 byte
String without '\0' terminator : N bytes
0x00 1 byte
Field separator character : 1 byte
Structure of parameter block for fixed length record data file:
String length : 1 byte
String without '\0' terminator : N bytes
Field #1 length: 1 byte
Field #2 length: 1 byte
. .
. .
. .
Field #n length: 1 byte

Return Value: 1) If string is found:
Carry flag = clear
DX:AX = pointer to current file position (at the beginning of the matched string)
2) If string is not found:
Carry flag = set
DX:AX = pointer to current file position (not changed)

42 Insert / Delete Data Block to / from File at the Current Position

Entry Parameter: AH = 42H
AL = 7 ; insert
8 ; delete

BX = file handle
CX = block length in bytes

Return Value: 1) If the function is successful:
Carry flag = clear
DX:AX = pointer to current file position (not changed)
2) If the function fails:
Carry flag = set
DX:AX = pointer to current file position (not changed)

Note: For insertion, the content of the inserted data block is undefined.

48 Allocate Memory

Entry Parameter: AH = 48H
BX = block size in paragraphs (16 bytes) to be allocated

Return Value: if success, CARRY flag is cleared
AX = segment address of block allocated
if failed , CARRY flag is set
AX = error code
BX = largest available memory block in paragraphs

Example:

```
int TS_alloc_mem(unsigned int size,unsigned char far *str,int *free)
{
  unsigned long aa;

  regs.h.ah=0x48;
  regs.x.bx=size;
  int86(0x21,&regs,&regs);
  *free = regs.x.bx;
  aa = (unsigned long)regs.x.ax * 10000L;
  str = (unsigned char far *)aa;
  if ((regs.x.cflag & 0x01) == 0) return(1);
  else return(-1);
}
```

49 Free Allocated Memory

Entry Parameter: AH = 49H
ES = segment address of block to be freed

Return Value: if success , CARRY flag is cleared
if failed , CARRY flag is set
AX = error code

Example:

```
int TS_free_mem(unsigned char far *str)
{
  unsigned long aa;

  regs.h.ah=0x49;
  segregs.es=FP_SEG(str);
  int86x(0x21,&regs,&regs,&segregs);
  if ((regs.x.cflag & 0x01) == 0) return(1);
  else return(-1);
}
```

4A Modify Allocated Memory Block

Entry Parameter: AH = 4AH
ES = segment address of the block to resize
BX = new block size in paragraphs (16 bytes)
Returned Value: if success , CARRY flag is cleared
if failed , CARRY flag is set
AX = error code

Example:

```
int TS_modify_alloc_mem(unsigned int size,unsigned char far *str,int
*free)
{
  unsigned long aa;

  regs.h.ah=0x4a;
  regs.x.bx=size;
  segregs.ds = FP_SEG(str);
  int86(0x21,&regs,&regs);
  *free = regs.x.bx;
  if ((regs.x.cflag & 0x01) == 0) return(1);
  else return(-1);
}
```

4C End Program

Entry Parameter: AH = 4CH
AL = program-defined return value

Return Value: None

Example:

```
void TS_end_program(int ret_code)
{
    regs.h.ah= 0x4C;
    regs.h.al= (unsigned char)ret_code;
  int86(0x21,&regs,&regs);
}
```

50 Read Data from Scanner Port or RFID

```
Entry Parameter:        AH = 50H
                DS:DX           ; pointer to data buffer
Return Value:   1)  AX =0               ;  has data input
                        DS:DX ; pointer to input data string
                        BL = 1 ; Code 39
                             2      ; Interleaved 2 of 5
                             3      ; Codabar
                             5      ; Code 128
                             6      ; EAN 128
                             7      ; Code 93
                            11H     ; UPC-A
                            12H     ; UPC-E
                            13H     ; EAN-13
                            14H     ; EAN-8
                            21H     ; User Code 1 (TRIOPTIC)
                            22H     ; User Code 2 (TOSHIBA)
                        CL = 0 ; scan direction from left to right
                               1 ; scan direction from right to left
                2)  AX = 1              ; no data input
```

Example:

```
int TD_get_bar1(unsigned char *str,int wait,int *type,int *dir)
{
  int i;
  do
  {
        regs.h.ah=0x50;
        segregs.ds = FP_SEG(str);
        regs.x.dx = FP_OFF(str);
        int86x(0x21,&regs,&regs,&segregs);
        i = regs.x.ax;
```

```
        *type = regs.h.bl;
        *dir = regs.h.cl;
    } while (wait && i);
    return(i);
}
```

51 Set Scanner Port

Entry Parameter: AH = 51H
AL = 0 ; disable scanner port
1 ; enable scanner
Return Value: None

Example:

```
void TD_set_bar(int status)
{
  regs.h.ah= 0x51;
  regs.h.al= (unsigned char)status;
  int86(0x21,&regs,&regs);
}
```

56 Rename File

Entry Parameter: AH = 56H
DS:DX ; pointer to old file name string
ES:DI ; pointer to new file name string
Return Value: if success, AH = 0 and CARRY flag is cleared
if failed, AH = 1 and CARRY flag is set

Example:

```
int TS_rename_file(char *inf,char far *outf)
{
  segregs.ds = FP_SEG(inf);
  regs.x.dx = FP_OFF(inf);
  segregs.es = FP_SEG(outf);
  regs.x.di = FP_OFF(outf);
  regs.h.ah=0x56;
  int86x(0x21,&regs,&regs,&segregs);
  if ((regs.x.cflag & 0x01) == 0) return(regs.x.ax);
  else return(-1);
}
```

5B Create New File

Entry Parameter: AH = 5BH
DS:DX ; pointer to file name string
Return Value: if success, CARRY flag is cleared
AX = file Handle
if failed, CARRY flay is set
AX = 04H ; too many open files
50H ; file exists
Note: If the specified file already exists, the function fails.

Example:

```
int TS_create_new_file(char *fn)
{
  regs.h.ah=0x5B;
  segregs.ds = FP_SEG(fn);
  regs.x.dx = FP_OFF(fn);
  if ((regs.x.cflag & 0x01) == 1) return(-1);
  else return(regs.x.ax);
}
```

5C Receive Data from RS232 Port in MULTIPOINT Protocol ---for Host Port Control

Entry Parameter: AH = 5CH
DS:DX ; pointer to data buffer (max. 256 bytes)
Returned Value: AL = 0 ; input successful
1 ; no data
DS:DX = input buffer pointer
Note: 1) Only for MULTIPOINT communication protocol
2) Data is sent by Host computer using ESC '0' command

Example:

```
int TC_str_I(unsigned char *str,int wait)
{
  do {
      regs.h.ah=0x5C
      segregs.ds = FP_SEG(str);
      regs.x.dx = FP_OFF(str);
      int86x(0x21,&regs,&regs,&segregs);
  } while (wait && regs.h.al);
  return(regs.h.al);
}
```

5D Send Data to RS232 Port in MULTIPOINT Protocol (Buffered Output)

---for Host Port Control

Entry Parameter: AH = 5DH

DS:DX ; pointer to buffer with output data (max. 256 bytes)

Return Value: AL = 0 ; successful

1 ; failed (output buffer already has data to be sent)

Note: 1) Only for MULTIPOINT communication protocol

2) PT600 will send data out when polled by Host computer

Example:

```
int TC_str_O(unsigned char *str,int wait)
{
  do {
       segregs.ds = FP_SEG(str);
       regs.x.dx = FP_OFF(str);
       regs.h.ah=0x5D;
       int86x(0x21,&regs,&regs,&segregs);
       return(regs.h.al);
  } while (wait && regs.h.al);
  return(regs.h.al);
}
```

5E Check RS232 Output Buffer Status in MULTIPOINT Protocol

---for Host Port Control

Entry Parameter: AH = 5EH

Returned Value: AL = 0 ; output buffer is empty

1 ; output buffer has data

Note: 1) Only for MULTIPOINT communication protocol

2) The function can be used to check if data has been sent to Host computer.

Example:

```
int TC_ready(int wait)
{
  do {
       regs.h.ah= 0x5E;
       int86(0x21,&regs,&regs);
  } while (wait && regs.h.al);
  return(regs.h.al);
```

```
}
```

1.6 Power Management Function (INT 22H)

Entry Parameter: None
Return Value: AX = 0 ; awaken by key
1 ; awaken by barcode scanner
2 ; awaken by RS232

The system will enter HALT power saving mode from this BIOS call, on return the value in AX register will define the type of device that awakes the system back to ACTIVE mode.

Sample C program with INT22

```
int get_barcode(char * str_bar)
{
union REGS regs;
struct SREGS sregs;

  regs.x.dx=(int)str_bar;
  segread(&sregs);
  regs.h.ah=0x50;
  intdosx(&regs,&regs,&sregs);
  if (regs.x.ax==1)
    return(0);        /* no data input from barcode port */
  else
    return(1);        /* has data input from barcode port */
}

int get_keyboard()
{
  if (kbhit()==0)
    return(0);        /* no input from key pressing */
return(1);     /* has input from keyboard */
}

while (       !get_barcode()      &&    /* check if data input from barcode
port */
              !get_keyboard()     )           /* check if data input from
keyboard      */
{
```

```
 int86(0x22,&regs,&regs);          /* enter stand-by state if no input from
                                      either barcode port or keyboard */
}
```

1.7 Beeper Control Function (INT 31H)

Entry Parameter:

AX= frequency		BX=time duration	
AX	Frequency(Hz)	BX	Time Duration(ms)
0	200	0	10
1	400	1	50
2	600	2	100
3	800	3	200
4	1K	4	500
5	2K	5	800
6	2.5K	6	1K
7	3K	7	1.5K
7	5K	8	2K

Return Value: None

Example:

```
void TD_beep_new(int fz,int tm)
   {
     regs.x.ax = fz;
     regs.x.bx = tm;
     int86(0x31,&regs,&regs);
   }
```

1.8 RS232 Control Functions (INT 33H)

Function #

00 Set Communication Parameters

Entry Parameter: AH = 0

AL bits 7-4: 0001xxxx baud 150

0010xxxx baud 300

0011xxxx baud 600

0100xxxx baud 1200

0101xxxx baud 2400

0110xxxx baud 4800

0111xxxx baud 9600

1000xxxx baud 19200

1001xxxx baud 38400

1010xxxx baud 57600

bits 3-2: xxxx00xx none parity

xxxx01xx odd parity

xxxx11xx even parity

bit 1: xxxxxx0x one stop bit

xxxxxx1x two stop bits

bit 0: xxxxxxx0 7 data bits

xxxxxxx1 8 data bits

Return Value: None

Example:

```
int TC_232_parameter(long baud,int parity,int stop,int data)
{
  unsigned char cc=0;
  unsigned int i_baud;
```

```
  i_baud = (int)(baud / 10L);
  switch (i_baud)
  {
    case   11 : cc=0x00; break;
    case   15 : cc=0x10; break;
    case   30 : cc=0x20; break;
    case   60 : cc=0x30; break;
    case  120 : cc=0x40; break;
    case  240 : cc=0x50; break;
    case  480 : cc=0x60; break;
    case 1920 : cc=0x80; break;
    case 3840 : cc=0x90; break;
    case 5760 : cc=0xA0; break;
    default: cc=0x70;  break;
  }
  switch (parity)
  {
    case 0 : break;
    case 1 : cc=cc|0x04; break;
    case 2 : cc=cc|0x0c; break;
  }
  switch (stop)
  {
    case 1 : break;
    case 2 : cc=cc|0x02; break;
  }
  switch (data)
  {
    case 7 : break;
    case 8 : cc=cc|0x01; break;
  }
  regs.h.ah = 0;
  regs.h.al  = cc;
  int86(0x33,&regs,&regs);
}
```

01 Get Character from RS232 Port

Entry Parameter: AH = 1

Return Value: 1) AH = 0 ; has character input
AL = character
2) AH = 1 ; no character input

Note: Only for NONE communication protocol

Example:

```
unsigned char TC_232_char_I()
{
  regs.h.ah = 1;
  int86(0x33,&regs,&regs);
  if (regs.h.ah == 0)
  {
   return(regs.h.al);
   }
   return(255);
}
```

02 Send Character to RS232 Port

Entry Parameter: AH = 2
AL = character
Return Value: None
Note: Only for NONE communication protocol

Example:

```
void TC_232_char_O(unsigned char ch)
{
  regs.h.ah = 2;
  regs.h.al = ch;
  int86(0X33&regs,&regs);
}
```

03 Enable RS-232 Port

Entry Parameter: AH = 3
Return Value: None

Example:

```
void TC_232_enable()
{
  regs.h.ah = 3;
  int86(0x33,&regs,&regs);
}
```

04 Disable RS-232 Port

Entry Parameter: AH = 4
Returned Value: None

Example:

```
void TC_232_disable()
{
  regs.h.ah = 4;
  int86(0X33&regs,&regs);
}
```

05 Set RTS/DTR Signal of RS232 Port

Entry parameter: AH = 5
AL = 1 ; set DTR
2 ; set RTS
DH = 0 ; set Enable (High voltage)
1 ; set Disable (Low voltage)
Return Value: None
Note: Only for NONE communication protocol

Example:

```
void TC_232_RTS(int rts)
{
  regs.h.ah = 5;
  regs.h.al = 2;
  regs.h.dh = (unsigned char)rts;
  int86(0X33&regs,&regs);
}
void TC_232_DTR(int dtr)
{
  regs.h.ah = 5;
  regs.h.al = 1;
  regs.h.dh = (unsigned char)dtr;
  int86(0X33&regs,&regs);
}
```

06 Get CTS Signal Status of RS232 Port

Entry Parameter: AH = 6
AL = 1 ; get CTS
Return Value: DH = 0 ; High voltage on signal
1 ; Low voltage on signal
Note: Only for NONE communication protocol

Example:

```
int TC_232_CTS()
{
  regs.h.ah = 6;
  regs.h.al = 2;
  int86(0X33&regs,&regs);
  return((int)regs.h.dh);
}
int TC_232_DSR()
{
  regs.h.ah = 6;
  regs.h.al = 1;
  int86(0X33&regs,&regs);
  return((int)regs.h.dh);
}
```

Chapter 2 MULTIPOINT Communication

2.1 MULTIPOINT Communication Protocol

MULTIPOINT communication protocol is supported by built-in communication tool on PT600. It is the default setting for RS232 port. User can enable/disable MULTIPOINT protocols by keypad setting or system call in the application.

If RS232 port is set to NONE communication protocol, that means MULTIPOINT protocol is disabled and the application can treat RS232 port as a character I/O device like COM ports on PC. Otherwise, the MULTIPOINT is active and PT600 will interpret RS232 input data as on-line commands from host computer and send data to host computer in data packet according to MULTIPOINT protocol.

MULTIPOINT protocol is an asynchronous serial multi-drop protocol for communication with the host computer. One host computer can be connected up to 32 PT600 terminals, while each PT600 has a unique address. The address is an ASCII character ('A' – 'Y' or '0' – '6') plus 80H.

In MULTIPOINT protocol, host computer always issues POLLING DATA request or ESC commands first, then wait for the addressed PT600 to response. The maximum packet size including protocol control characters is 256 bytes.

Symbols in Transmission Packet

STX = 02H
ETX = 03H
EOT = 04H
ACK = 06H
NAK = 15H
ESC = 1BH
CMD = command character
ADDR = address ('A' – 'Y' or '0' – '6') + 80H
CS1 CS2= 2 checksum characters
<data>/<parameters> = character string

Host Transmission Packet

Transmission	Format
Poll	STX ADDR
Host Data	STX ESC '0' <data> CS1 CS2 ADDR

Host Command	STX ESC CMD <parameters> CS1 CS2 ADDR
Acknowledgement	ACK
Negative ACK	NAK

PT600 Transmissions Packet

Transmission	Format
Terminal Data	STX <data> CS1 CS2 ETX
Terminal No Data	EOT
Terminal response	STX CMD <parameters> CS1 CS2 ETX
Acknowledgement	ACK
Negative ACK	NAK

Data Conversion Rules in Packet

1) One-byte data converted to two-byte data:

\ convert to \\
00 hex -- 1F hex convert to \ 80 hex -- \ 9F hex
A0 hex -- FF hex convert to \ 20 hex -- \ 7F hex
(excluding DC hex)

2) One-byte data transmitted as original data without converting:
all codes not in 1)
(including DC hex)

Data Checksum Calculation in Packet

1) CS =[sum of all characters (excluding STX and ETX) + packet length (excluding STX and ADDR/ETX)] MOD 256

2) CS1 = high nibble of CS + 40H

3) CS2 = low nibble of CS + 40H

Example packet of command to send the file named A.EXE to PT600 with address 'A'

Packet: STX ESC CMD <parameters> CS1 CS2 ADDR

CMD = 'L'

<parameters> = "A . E X E "

Packet length = total characters of ESC, CMD and <parameters> = 7

ADDR = 'A' + 80H = C1H

CS = [(ESC+'L'+'A'+'.'+'E'+'X'+'E'+ADDR)+Packet length]

MOD 256

= [(1BH+4CH+41H+2EH+45H+58H+45H+C1H)+7] MOD 256

= 80H

CS1 = high nibble of CS + 40H = 08H + 40H = 48H

CS2 = low nibble of CS + 40H = 00H + 40H = 40H

Example Packet in Hex Format:

STX ESC CMD <parameters> CS1 CS2 ADDR
02H 1BH 4CH 41H 2EH 45H 58H 45H 48H 40H C1H

2.2 Data Communication in MULTIPOINT Protocol

1. Host Send Data to PT600 (ESC 0)

Format :
- ✍ STX ESC '0' <data> CS1 CS2 ADDR
- ✍ ACK or NAK

Description: PT600 will put the received data in the buffer. The application on PT600 can:
1) Use system call INT 21H/AH=5CH to get data.
2) Use system call INT 21H/AH=5EH to check the buffer status.

2. Host Requests Data from the PT600 (Polling)

Format
- ✍ STX ADDR ; polling
- ✍ EOT ; if PT600 has no data to send

Or
- ✍ STX <data> CS1 CS2 ETX ; if there is data in PT600 buffer

Description: The application on PT600 uses system call INT 21H/AH= 5DH to put data in buffer. Once polled by host computer, PT600 system will send data out.

2.3 File Communication in MULTIPOINT Protocol

1. Host Send File to PT600 (Download)

Format :

? STX ESC ‘L’ <filename> CS1 CS2 ADDR
? ACK or NAK ; if NAK, retry or abort the command
? STX ESC ‘Y’ <data> CS1 CS2 ADDR ; if ACK, loops until file is transmitted
? ACK or NAK ; if NAK, send the packet again
? STX ESC ‘Z’ CS1 CS2 ADDR ; end of file transfer
? ? ACK or NAK

Description: The download command is used to send binary executable program (*.exe file) or data file from host system to the PT600. When the PT600 receives the download command it sends an ACK response back to the host and immediately put itself into file receiving state. The host system may start transmission of the file as soon as the ACK response has arrived.

2. Cancel Current Downloading File Command (ESC z)

Format: ? **STX ESC ‘z’ CS1 CS2 ADDR**
? ACK or NAK

3. Host Requests File from the PT600 (Upload)

Format: ? **STX ESC ‘U’ <filename> CS1 CS2 ADDR**
? ACK or NAK ; if NAK, retry or abort
? STX ESC ‘Y’ CS1 CS2 ADDR ; loops until ESC ‘Z’ is
? STX ESC ‘Y’ <data> CS1 CS2 ETX ; received
? STX ESC ‘Z’ CS1 CS2 ETX ; end of file transfer
? ACK or NAK

Description: For uploading, the host must know the filename on PT600. Get Directory Command (ESC ‘D’) can be used to view all filenames on PT600 (see 4.4 Host Commands).

4. Cancel Current Uploading File Command (ESC y)

Format: ? STX ESC ‘y’ CS1 CS2 ADDR
? ACK or NAK

2.4 Host Commands

1. Set MULTIPOINT Address (ESC 5)

Format:
? **STX ESC ‘5’ <addr> CS1 CS2 ADDR ;<addr>=‘A’~’Y’ or ’0’~’6’**
? STX ESC ‘5’ <retcode> CS1 CS2 ETX ;<retcode>= 00H successful
; = 01H error

7 Description: The command will change PT600 MULTIPOINT address.

2. Warm Start (ESC A)

Format: ? **STX ESC ‘A’ CS1 CS2 ADDR**
? ACK or NAK

Description: The command causes PT600 stop program execution and return to Ready Mode. Programs and data that have been stored in the PT600 RAM space are preserved. The system parameters remain unchanged. The equivalent keypad invoking sequence is pressing CMD and then the ALPHA key.

3. Enable/Disable Barcode Symbologies (ESC B)

Format: ? STX ESC 'B' <C_1 C_2 C_3 C_4 C_5 C_6 C_7 C_8> CS1 CS2 ADDR

C_1 = 'E' Enable Code 39
'F' Enable Code 39 full ASCII
'D' Disable Code 39

C_2 - C_8 = 'E' Enable Interleaved 2 of 5, Codabar, UPC/EAN, Code 128, EAN 128, Code 93, Trioptic Code respectively
'D' Disable

? STX ESC 'B' <retcode>CS1 CS2 ETX
;<retcode>= 00H successful
;
01H error

4. Communication Configuration (ESC C)

Format: ? STX ESC 'C' <comtable> CS1 CS2 ADDR
? STX ESC 'C ' <retcode> CS1 CS2 ETX ;<retcode>= 00H successful
; 01H error

Description: The command writes <comtable> to PT600 communication control table.

<comtable> is defined as:

```
typedef strut {
        char   baud_rate;   //                          '0'~'9'/'A': 110/150/300/600/1200/2400
                            //                          4800/9600/9200/38400/57600
        char   stop_bit,    // '1'/'2':   1/2 stop bits
        char   data_bit     // '7'/'8':   7/8 data bits
        char   parity;      // 'N'/'O'/'P':   None/Odd /Even parity
        char   flow_control;// 'X'/'C'/'F':   XON XOFF/RTS CTS/NONE
        char   protocol;    // 'M'/'F'  MULTIPOINT/None
        char   address      // 'A'~'Y' or '0'~'6'
        char   reserved;
        char   reserved;
        char   timeout  ;   // communication time-out
} comtable;
```

The new communication control table takes effect immediately after the command has been successfully received. The PT600 will reinitialize the corresponding communication port with its new parameters. For **Example**, if the command instructs the RS-232 port to change the baud rate from 9600 to 1200 the PT600 will switch to 1200 right after the command has been received. The next host communication will use 1200 baud rate.

5. Get File Directory in RAM Disk (ESC D)

Format: ? STX ESC 'D' CS1 CS2 ADDR
? STX ESC 'D' <filename,filename…> CS1 CS2 ETX

Description: PT600 will send its file directory to the host.

6. Get Program Name in FLASH Memory (ESC D/ROM)

Format: ? STX ESC 'D/ROM' CS1 CS2 ADDR
? STX ESC 'D' <program-name,program-name…> CS1 CS2 ETX

Description: PT600 will send program names in FLASH memory to the host.

7. Erase File (ESC E)

Format: ? STX ESC 'E' <filename> CS1 CS2 ADDR
? STX ESC 'E' <retcode> CS1 CS2 ETX ;<retcode>=00H successful
; 01H error

Description: Used to erase file in PT600.

8. Change EXEC Area Size (ESC F)

Format: ? STX ESC 'F' <nnn> CS1 CS2 ADDR ; <nnn>: 3 digits in ASCII
? STX ESC 'F' <retcode> CS1 CS2 ETX ; <retcode>= 00H successful
;
01H can't change

;

02H invalid value

Description: If PT600 RAM disk has files, its size can't be changed (<retcode>=01H).

9. Get RAM Memory Configuration (ESC G)

Format: ? STX ESC 'G' CS1 CS2 ADDR
? STX ESC 'G' <s1><s2><s3> CS1 CS2 ETX
; <s1> total RAM memory size in 4 ASCII digits
; <s2> EXEC memory size in 4 ASCII digits
; <s3> free memory size in 4 ASCII digits

10. Get FLASH Memory Configuration (ESC G/ROM)

Format: ? STX ESC 'G/ROM' CS1 CS2 ADDR
? STX ESC 'G' <s1><s2>CS1 CS2 ETX
; <s1> total FLASH memory size in 4 ASCII digits
; <s2> free FLASH memory size in 4 ASCII digits

11. Cold Start (ESC H)

Format: ? STX ESC 'H' CS1 CS2 ADDR
? ACK or NAK

Description: The command clears all PT600 RAM memory content. It performs tests on all major hardware devices. Programs or data that have previously accumulated in the PT600 or previously been downloaded by the host will be purged from the memory. Default system parameters are restored from the EPROM.

12. Get the Name of Current Running Program (ESC I)

Format:

? **STX ESC 'I' CS1 CS2 ADDR**

? STX ESC 'I' <prg-name> CS1 CS2 ETX ; if RAM program is running

? STX ESC 'I' <prg-name/ROM> CS1 CS2 ETX ; if FLASH program is running

? STX ESC 'I' <retcode> CS1 CS2 ETX ; if no program is running

;<retcode> = 01H

13. Check whether File/Program Exists (ESC J)

Format: ? STX ESC 'J' <filename> CS1 CS2 ADDR

? STX ESC 'J' <retcode> CS1 CS2 ETX ;<retcode>=0 file in RAM

; 1 no file

; 2 file in FLASH

14. Enter Kermit Server Mode (ESC k)

Format: ? STX ESC 'k' CS1 CS2 ADDR

? ACK or NAK

Description: PT600 will enter Kermit Server Mode if the command is received. The MULTIPOINT protocol will be disabled until BYE or EXIT command is issued in Kermit Server Mode.

15. Set Date/Time (ESC M)

Format: ? STX ESC 'M'<datetime> CS1 CS2 ADDR ;<datetime>=ASCII string

; in "yyyymmddhhmmss"

? STX ESC' M' <retcode> CS1 CS2 ETX ;<retcode>= 00H success

; 01H error

Description: This command allows the host system to set PT600 real time clock. The parameter <datetime> is an ASCII character string. The first four characters represent the year. The next two characters represent the month. The fields after the month field represent the day of the month, hour (24-hour format), minute, and second,

respectively. For **Example**, string "19900926234500" will set PT600 clock to September 26, 1990. The time is 11:45 PM. The PT600 reconfigures the real time clock chip as soon as the command has been successfully received.

16. Set Buzzer Volume (ESC N)

Format: ? STX ESC 'N' <n> CS1 CS2 ADDR ; <n>='0' low volume
;
'5' medium volume
;
'9' high volume
? ACK or NAK

17. Change Password (ESC P)

Format: ? STX ESC 'P'<password>CS1 CS2 ADDR ; <password> 10 characters

; maximum
? STX ESC 'P' <retcode> CS1 CS2 ETX ; <retcode>=00H success
; 01H error

Description: The command changes the password for PT600 Supervisor Mode.

18. Get Terminal ID (ESC R)

Format: ? STX ESC 'R' CS1 CS2 ADDR
? STX ESC 'R' <ID> CS1 CS2 ETX

Description: <ID> is 8-characters fixed length string and its default value is "PT600".

19. Terminal Mode Configuration (ESC T)

Format: ? **STX ESC 'T' <termtable> CS1 CS2 ADDR**
? STX ESC 'T' <retcode> CS1 CS2 ETX ;<retcode>=00H success
; 01H error

Description: The command set the configuration for PT600 Terminal Mode.

<termtable> is defined as:

```
typedef struct {
        char   id[8];        // terminal ID      =  fixed  8  characters string
        char   online;       //  'R' /' L'       = remote / local
        char   echo;         //  'N' /' F'  =echo on / echo off
        char   autolf;       //  'N' / 'F'  =autoLF / no autoLF
        char   mode;         //  'C' /' B'  =character / block mode
        char   linepage;     //  'L' /' P' / 'B' = block defined as line / page / both
        char   lineterm;     //  line mode terminator character
        char   pageterm;     //  page mode terminator character
} termtable;
```

The new terminal control table takes effect immediately after the command has been successfully received.

20. Device Configuration (ESC V)

Format: ? **STX ESC 'V' <devtable> CS1 CS2 ADDR**

? STX ESC 'V' <retcode> CS1 CS2 ETX ;<retcode>=00H success
; 01H error

Description: The command sets configuration of PT600 device control table.

<devtable> is defined as:

```
typedef struct {
        char   scanner_type;    //  'P' /'A'/'D' = Pen/Auto/Disable
        char   lcd_backlit;  //  'N' / 'F'     = LCD backlight  ON /OFF
        char   buzzer;          //  'N' /'F'      = Buzzer ON /OFF
        char   reserved;
        char   beepvol;         // '0' /'5' /'9'      =          Vol. Low/Medium/High
        char   serial_port;  //  'N' /'F'   = RS232 port  Enable/Disable
        char   auto_off;        //  '0'       = disable auto-off
                                // '1' ~ '9'   auto-off    timer    (1~9 minutes)
        char   reserved;
} devtable;
```

The new device control table takes effect immediately after the command has been successfully received.

21. Get Portable Model Number and BIOS Version Number (ESC v)

Format: ? STX ESC 'v' CS1 CS2 ADDR

? STX ESC 'v' 'PT600 V5.00' CS1 CS2 ETX

Description: The model and BIOS version numbers are separated by a space character.

22. Run Program in RAM or FLASH memory (ESC X)

Format: ? STX ESC X <prg-name> CS1 CS2 ADDR

? STX ESC X <retcode> CS1 CS2 ETX ;<retcode>=00H run from RAM

; 01H no program

; 02H run from FLASH

23. Run Program in FLASH memory (ESC X/ROM)

Format: ? STX ESC X <prg-name/ROM> CS1 CS2 ADDR

? STX ESC X <retcode> CS1 CS2 ETX ;<retcode>=01H no program

; 02H run from FLASH