

# PT/HT630 Portable Terminal Programming Reference Guide



---

<b>CHAPTER 1.BIOS AND SYSTEM FUNCTIONS .....</b>	<b>5</b>
1.1INTERRUPT VECTOR ASSIGNMENT FOR I/OS.....	5
1.2DISPLAY FONT FUNCTIONS ( INT 09H ) .....	5
1.2.1 Select Large Font .....	5
1.2.2 Select Small Font .....	5
1.2.3 Set Font Type .....	6
1.2.4 Get Font Type .....	6
1.2.5 Set User-Defined Font for All Characters .....	6
1.2.6 Get Font Data for All Characters .....	7
1.2.7 Set User-Defined Font for One Character .....	7
1.2.8 Reverse character display (only available on small font mode).....	7
1.2.9 Get Font Data for One Characters .....	8
1.3KERMIT SERVER FUNCTION .....	8
1.4LCD CONTROL FUNCTIONS ( INT 10H ) .....	9
1.4.1 Clear LCD Screen.....	9
1.4.2 Enable/Disable Screen Scroll .....	9
1.4.3 Set Cursor Position.....	9
1.4.4 Get Cursor Position.....	10
1.4.5 Display 16x16 Bitmap .....	10
1.4.6 Set Cursor On/Off.....	12
1.4.7 Enable/Disable Power-on Logo Display .....	12
1.4.8 Display Character .....	13
1.4.9 Reverse display.....	13
1.4.10 Display 16*16 Bitmap at Current Cursor Position .....	13
1.4.11 Read Pixel from screen .....	13
1.4.12 Write Pixel to screen.....	14
1.5SYSTEM FUNCTIONS ( INT 21H ) .....	14
1.5.1 Terminate Program.....	14
1.5.2 Read Keypad (wait if no key) and Write to LCD.....	14
1.5.3 Write LCD.....	15
1.5.4 Read RS232 (wait if no character) .....	15
1.5.5 Write RS232 .....	15
1.5.6 Direct Console I/O .....	15
1.5.7 Read Keypad Not Echo(wait if no key).....	16
1.5.8 Read Keypad Echo(wait if no key).....	16
1.5.9 Write Character String to LCD.....	16
1.5.10 Buffered Keypad Input .....	17
1.5.11 Check Keypad Status .....	17
1.5.12 Device control function.....	17
1.5.13 General Communication Setting.....	23
1.5.14 Terminal mode setting function.....	26
1.5.15 Set User-defined Key-map.....	29
1.5.16 Get System Key-map.....	30
1.5.17 Barcode symbology setting function .....	31
1.5.18 Set Interrupt Vector.....	34
1.5.19 Get System Date .....	35
1.5.20 Set System Date .....	35
1.5.21 Get System Time.....	35
1.5.22 Set System Time .....	35
1.5.23 Set Alarm Date.....	36
1.5.24 Set Alarm Time.....	36
1.5.25 Get DOS and Firmware Version Number .....	37
1.5.26 Get Interrupt Vector.....	37
1.5.27 Get Free Disk Cluster .....	37
1.5.28 File operation function.....	38

1.5.29 Allocate Memory.....	45
1.5.30 Free Allocated Memory .....	45
1.5.31 Modify Allocated Memory Block .....	46
1.5.32 Run specified program.....	46
1.5.33 End Program .....	46
1.5.34 Read Data from Scanner Port .....	47
1.5.35 Set Scanner Port.....	47
1.5.36 Receive Data from RS232 Port in MULTIPOINT Protocol.....	48
1.5.37 Send Data to RS232 Port in MULTIPOINT Protocol (Buffered Output).....	48
1.5.38 Check RS232 Output Buffer Status in MULTIPOINT Protocol .....	49
1.6POWER MANAGEMENT FUNCTION ( INT 22H ) .....	50
1.7BEEPER CONTROL FUNCTION ( INT 31H ) .....	51
1.8COM1 RS232 CONTROL FUNCTIONS ( INT 33H ).....	52
1.8.1 Set Communication Parameters .....	52
1.8.2 Get Character from RS232 Port .....	53
1.8.3 Send Character to RS232 Port .....	54
1.8.4 Enable RS-232 Port.....	54
1.8.5 Disable RS-232 Port.....	54
1.8.6 Set RTS/DTR Signal of RS232 Port .....	55
1.8.7 Get CTS/DSR Signal Status of RS232 Port.....	55
1.8.8 Flush RS232 RX buffer .....	55
1.9COM2 (INTERNAL COM PORT) RS232 CONTROL FUNCTIONS ( INT 14H ).....	56
1.9.1 Set Communication Parameters .....	56
1.9.2 Get Character from COM 2 RS232 Port .....	58
1.9.3 Send Character to COM2 RS232 Port.....	58
1.9.4 Enable COM2 RS-232 Port.....	58
1.9.5 Disable COM2 RS-232 Port.....	58
1.9.6 Set RTS/DTR Signal of RS232 Port .....	59
1.9.7 Get CTS/DSR Signal Status of RS232 Port.....	59
1.9.8 Flush RS232 RX buffer .....	59
1.10COM2 (INTERNAL COM PORT) BLUETOOTH CONTROL FUNCTIONS ( INT 35H ).....	57
1.10.1 Set COM2 Parameters.....	57
1.10.2 Get Character from Bluetooth via COM2 port.....	58
1.10.3 Send Character to Bluetooth via COM2 port .....	58
1.10.4 Enable COM2 RS-232 Port & Bluetooth module .....	58
1.10.5 Disable COM2 RS-232 Port & Bluetooth module .....	59
1.10.6 Set RTS/DTR Signal of COM2 port .....	59
1.10.7 Get CTS/DSR Signal Status of COM2 Port .....	59
1.10.8 Clear Receiving Buffer of COM2 port.....	59
1.10.9 Warm start BT module .....	60
1.10.10 Cold start BT module.....	60
1.10.11 Send BT AT command to BT module via COM2 port.....	60
<b>CHAPTER 2.MULTIPOINT COMMUNICATION .....</b>	<b>65</b>
2.1MULTIPOINT COMMUNICATION PROTOCOL .....	65
2.1.1 Symbols in Transmission Packet.....	65
2.1.2 Fast checking whether terminal is on-line ( BELL).....	65
2.1.3 Host Transmission Packet .....	66
2.1.4 PT/HT630 Transmissions Packet .....	66
2.1.5 Data Conversion Rules in Packet.....	66
2.1.6 Data Checksum Calculation in Packet .....	66
2.1.7 Example packet of command to send the file named A.EXE to PT/HT630 with address 'A' .....	66
2.2DATA COMMUNICATION IN MULTIPOINT PROTOCOL.....	68
2.2.1 Host Send Data to PT/HT630 (ESC 0) .....	68
2.2.2 Host Requests Data from the PT/HT630 ( Polling ) .....	68
2.3FILE COMMUNICATION IN MULTIPOINT PROTOCOL .....	68
2.3.1 Host Send File to PT/HT630 ( Download ) .....	68
2.3.2 Host Send File to PT/HT630 ( Download ) from specified file position .....	69

---

2.3.3 Cancel Current Downloading File Command (ESC z).....	69
2.3.4 Host Requests File from the PT/HT630 ( Upload ) .....	69
2.3.5 Host Requests File from the PT/HT630 ( Upload ) from specified file positon .....	69
2.3.6 Cancel Current Uploading File Command (ESC y).....	69
2.4HOST COMMANDS.....	70
2.4.1 Set MULTIPOINT Address ( ESC 5).....	70
2.4.2 Warm Start ( ESC A ) .....	70
2.4.3 Enable/Disable Barcode Symbolologies ( ESC B ).....	70
2.4.4 Communication Configuration ( ESC C ).....	71
2.4.5 Get File Directory in RAM Disk ( ESC D ).....	71
2.4.6 Get Program Name in FLASH Memory (ESC D/ROM) .....	72
2.4.7 Erase File ( ESC E ) .....	72
2.4.8 Change EXEC Area Size (ESC F).....	72
2.4.9 Get RAM Memory Configuration (ESC G).....	72
2.4.10 Get FLASH Memory Configuration (ESC G/ROM) .....	72
2.4.11 Cold Start ( ESC H ) .....	72
2.4.12 Get the Name of Current Running Program (ESC I).....	73
2.4.13 Check whether File/Program Exists (ESC J).....	73
2.4.14 Enter Kermit Server Mode (ESC k) .....	73
2.4.15 Set Date/Time ( ESC M ).....	74
2.4.16 Set Buzzer Volume (ESC N) .....	74
2.4.17 Change Password ( ESC P ) .....	74
2.4.18 Get Terminal ID (ESC R).....	74
2.4.19 Terminal Mode Configuration ( ESC T ).....	75
2.4.20 Device Configuration ( ESC V ).....	75
2.4.21 Get Portable Model Number and BIOS Version Number (ESC v).....	76
2.4.22 Run Program in RAM or FLASH memory ( ESC X ).....	76
2.4.23 Run Program in FLASH memory (ESC X/ROM).....	76
<b>CHAPTER 3.UPDATE NOTE.....</b>	<b>77</b>

## Chapter 1. BIOS and System Functions

The PT/HT630 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
PT630	HT630	BIOS Function
09 H	09 H	Display Font
0F H	1F H	Kermit Server
10 H	10 H	LCD Control
21 H	21 H	System Functions
22 H	22 H	Power Manager
31 H	31 H	Beeper Control
33 H	33 H	RS232 Control
	35H	Bluetooth Control

### 1.2 Display Font Functions ( INT 09H )

#### 1.2.1. Select Large Font

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

Return Value: None

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

#### 1.2.2. Select Small Font

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

Return Value: None

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

**1.2.3. Set Font Type**

Entry Parameter: AH= 2  
AL= 0/1 ; set to large/small font  
Return Value: None

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

**1.2.4. Get Font Type**

Entry Parameter: AH= 3  
Return Value: AL= 0/1 ; large/small font

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

**1.2.5. 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

```
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);
}
```

**1.2.6. 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);
}
```

**1.2.7. 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 )

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);
}
```

**1.2.8. Reverse character display (only available on small font mode)**

Use printf or character display to output "%c7r[" to force PT600 to display reverse character display. E.x

```
printf("%c7r[" ,27); ---- show REVERSE character
```

Use printf or character display to output "%c8r[" to force PT600 to show normal character. E.x

```
printf("%c0r[" ,27); ---- show normal character
```

**1.2.9. 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**

Note :

Due to system issue, PT630 and HT630 use different INT vector for Kermit function

PT630 = **INT 0FH**

HT630 = **INT 1FH**

Entry Parameter: None

Return Value: None

When INT 0FH is called in user application, PT/HT630 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 )

### 1.4.1. Clear LCD Screen

Entry Parameter: AH = 0

Return Value: None

**Example:**

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

### 1.4.2. 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);
}
```

### 1.4.3. Set Cursor Position

Entry Parameter: AH = 2

DH = Row

DL = Column

	PT/HT630	
	6x8	8x16
Row	0~7	0~3
Column	0~19	0~15

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);
}
```

---

#### 1.4.4. 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;
}
```

#### 1.4.5. 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.**

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
content	FF	FF	3E	1C	C9	E3	F7	FF								
7	█	█			█	█	█	█	█	█	█	█	█	█	█	█
6	█	█	█			█	█	█	█	█	█	█	█	█	█	█
Bit	█	█	█	█			█	█	█	█	█	█	█	█	█	█
Count	█	█	█	█	█			█	█	█	█	█	█	█	█	█
3	█	█	█	█			█	█	█	█	█	█	█	█	█	█
2	█	█	█			█	█	█	█	█	█	█	█	█	█	█
1	█	█			█	█	█	█	█	█	█	█	█	█	█	█
0	█	█		█	█	█	█	█	█	█	█	█	█	█	█	█

  

7	█	█	█			█	█	█	█	█	█	█	█	█	█	█
6	█	█	█	█			█	█	█	█	█	█		█	█	█
Bit	█	█	█	█	█			█	█	█	█				█	█
Count	█	█	█	█	█	█			█	█			█			█
3	█	█	█	█	█	█	█					█	█	█		
2	█	█	█	█	█	█	█	█			█	█	█	█	█	
1	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█
0	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█
Byte count	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Content	FF	FF	FF	FE	FC	F9	F3	E7	CF	CF	E7	F3	F9	F3	E7	CF

```
#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();
}

```

#### 1.4.6. 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);
}

```

#### 1.4.7. Enable/Disable Power-on Logo Display

Entry Parameter: AH = 6  
AL = 0/1 ; disable/enable

Return Value: None

##### **Example**

```

void TL_Logo_Dispatch(int status)
{
    regs.h.ah = 6;
    regs.h.al = (unsigned char)status;
    int86(0x10,&regs,&regs);
}

```

#### 1.4.8. Display Character

Entry Parameter: AH = 0AH  
AL = 0 - 255 ; character to display

Return Value: None

##### **Example**

```

void TL_Dispatch_Char(int cc)
{

```

```

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

```

#### 1.4.9. Reverse display

Entry Parameter: AH = 0bH  
AL = 0 disable ; disable reverse  
1 Enable ; Enable reverse function

Return Value: None

##### Example

```

void TL_Reverse_Disp(int cc)
{
regs.h.ah = 0x0B;
regs.h.al = (byte)cc;
int86(0x10,&regs,&regs);
}

```

#### 1.4.10. 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);
}

```

#### 1.4.11. Read Pixel from screen

Entry Parameter: INT 0x10  
AH= 0x40

Return DH= row ; 0-63  
DL=column ; 0-127  
AL=0 ; No Pixel  
1 ; With Pixel

##### Example:

```

int TL_get_pixel(int x, int y)
{

```

```

regs.h.ah= 0x40;
regs.h.dl= (unsigned char)x;
regs.h.dh= (unsigned char)y;
int86(0x10,&regs,&regs);
return(regs.h.al);
}

```

#### 1.4.12. Write Pixel to screen

Entry Parameter: INT 0x10  
AH= 0x41  
AL=0 ; No Pixel  
1 ; With Pixel

Return DH= row ; 0-63  
DL=column ; 0-127

##### **Example:**

```

void TL_set_pixel(int x,int y,int status)
{
regs.h.ah= 0x41;
regs.h.al= (unsigned char)status;
regs.h.dl= (unsigned char)x;
regs.h.dh= (unsigned char)y;
int86(0x10,&regs,&regs);
}

```

### 1.5 System Functions ( INT 21H )

#### 1.5.1. Terminate Program

Entry Parameter: AH = 0  
Return Value: None

##### **Example:**

```

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

```

#### 1.5.2. 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);
}
```

**1.5.3. 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;
}
```

**1.5.4. 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);
}
```

**1.5.5. 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;
}
```

**1.5.6. 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);
}
```

**1.5.7. Read Keypad Not Echo(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);
}
```

**1.5.8. Read Keypad Echo(wait if no key)**

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

**1.5.9. 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;
}

```

### 1.5.10. 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)
{
    segreg.ds = FP_SEG(str);
    regs.x.dx = FP_OFF(str);
    regs.h.ah= 0x0a;
    int86x(0x21,&regs,&regs,&segregs);
    return;
}

```

### 1.5.11. 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;
    int86x(0x21,&regs,&regs);
    return(regs.h.al);
}

```

### 1.5.12. Device control function

#### 1.5.12.1 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);
}
```

### 1.5.12.2 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);
}
```

### 1.5.12.3 Beeper Volume

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

Return Value: None

**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);
}
```

### 1.5.12.4 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);
}
```

**1.5.12.5 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);
}
```

**1.5.12.6 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
	ALPHA-1	5B	@

0. English	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	ı
	ALPHA-2	A5	Ñ
	ALPHA-3	A8	ı
4. French	ALPHA-1	F8	°
	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);
}
```

**1.5.12.7 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);
}
```

**1.5.12.8 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);
}
```

**1.5.12.9 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);
}
```

**1.5.12.10 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);
}
```

**1.5.12.11 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);
}
```

**1.5.12.12 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);
}
```

**1.5.12.13 Set Aim mode for Long Range engine**

Entry Parameter: AH = 1AH

BH = 0FH

Return Value: AL = 0 ; Disable

1 ; Enable

**1.5.12.14 Set Alpha input mode**

Entry Parameter: AH = 1AH

BH = 0x11H

**1.5.12.15 Set keypad input statusr**

Entry Parameter: AH = 1AH

BH = 0x12H

AL = 0 ; Alpha mode

1 ; Non-Aplha mode

**1.5.12.16 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_Scanner_Status()
{
```

```

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

```

### 1.5.13. General Communication Setting

#### 1.5.13.1 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);
}

```

#### 1.5.13.2 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);
}

```

### 1.5.13.3 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);
}

```

#### 1.5.13.4 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

#### 1.5.13.5 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);
}

```

#### 1.5.13.6 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);
}

```

### 1.5.13.7 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);
}

```

### 1.5.13.8 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);
}

```

## 1.5.14. Terminal mode setting function

### 1.5.14.1 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);  
}
```

#### 1.5.14.2 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);  
}
```

#### 1.5.14.3 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);  
}
```

#### 1.5.14.4 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);  
}
```

**1.5.14.5 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);
}
```

**1.5.14.6 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);
}
```

**1.5.14.7 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);
}
```

**1.5.14.8 Get Terminal ID**

Entry Parameter: AH = 1DH

BH = 0x10

Returned Value: DS:DX = output buffer pointer

**1.5.15. Set User-defined Key-map**

Entry Parameter: AH = 1EH

AL = 0

DS:DX

; pointer to Key-map buffer with 5\*32 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 Mode			CMD Mode		
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	7	37	32	S	53	64	T	54	96	U	55	128	-	2D
1	4	34	33	J	4A	65	K	4B	97	L	4C	129	:	3A
2	1	31	34	A	41	66	B	42	98	C	43	130	#	23
3	CLR	08	35	CLR	08	67	CLR	08	99	CLR	08	131	\	5c
4		0	36		0	68		0	100		0	132		84
5	F1	86	37	F1	86	69	F1	86	101	F1	86	133	F5	8A
6	ENTER	0D	38	ENTER	0D	70	ENTER	0D	102	ENTER	0D	134	ENTER	0D
7		0	39		0	71		0	103		0	135		0
8	8	38	40	V	56	72	W	57	104	X	58	136	+	2B
9	5	35	41	M	4D	73	N	4E	105	Q	4F	137	= 3D	24
10	2	32	42	D	44	74	E	45	106	F	46	138	\$	24
11	0	30	43	@	40	75	?	3F	107	&	26	139	'	27
12	◀□	11	44	◀□	11	76	◀□	11	108	◀□	11	140	◀□	11
13	▲□	13	45	▲□	13	77	▲□	13	109	▲□	13	141	PgDn	93
14	F2	87	46	F2	87	78	F2	87	110	F2	87	142	F6	8B
15		0	47		0	79		0	111		0	143		00
16	9	39	48	Y	59	80	Z	5A	112	-	5F	144	*	2A
17	6	36	49	P	50	81	Q	51	113	R	52	145	/	2F
18	3	33	50	G	47	82	H	48	114	I	49	146	%	25
19	.	2E	51	:	3B	83	.	2E	115	.	2C	147	!	21
20	▶□	10	52	▶□	10	84	▶□	10	116	▶□	10	148	▶□	10
21	▼□	12	53	▼□	12	85	▼□	12	117	▼□	12	149	PgUp	92
22	F3	88	54	F3	88	86	F3	88	118	F3	88	150	F7	8C
23		0	55		0	87		0	119		0	151		0
24		0	56		0	88		0	120		0	152		0
25		0	57		0	89		0	121		0	153		0
26		0	58		0	90		0	122		0	154		0
27		84	59		84	91		84	123		84	155		84
28	SP	20	60	SP	20	92	SP	20	124	SP	20	156	SP	20
29		80	61		00	93		00	125		00	157		80
30	F4	89	62	F4	89	94	F4	89	126	F4	89	158	F8	8D
31		0	63		0	95		0	127		0	159		0

☾ : LCD Backlight  
 ★ : LCD Contrast  
 ⚙ : Beeper Volume

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

### **1.5.16. 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;
    int86(0x21,&regs,&regs,&segregs);
    return(regs.h.al);
}
```

### **1.5.17. Barcode symbology setting function**

#### **1.5.17.1 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);
}
```

#### **1.5.17.2 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);
}
```

**1.5.17.3 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)decoder_type;
    regs.h.al = (unsigned char)setting;
    int86(0x21,&regs,&regs);
}
```

**1.5.17.4 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:

### 1.5.17.5 Codabar Settings

Entry Parameter: AH = 1FH  
 BH = 3  
 BL = 3  
 AL bit 0= 0/1 ; disable/enable Codabar 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

### 1.5.17.6 UPC/EAN Add-on

Entry Parameter: AH = 1FH  
 BH = 3  
 BL = 4  
 CL=0 ; Disable  
 1 ; Optional  
 2 ; Required

Return Value: None

### 1.5.17.7 Code 128 Setting

Entry Parameter: AH = 1FH  
 BH = 3  
 BL = 5  
 AL bit 0= 0/1 : disable/enable Code 128 decoding

Return Value: None

### 1.5.17.8 EAN 128 Setting

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

Return Value: None

### 1.5.17.9 Code 93 Setting

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

Return Value: None

### 1.5.17.10 Code 32 Setting

Entry Parameter: AH = 1FH  
 BH = 3  
 BL = 8  
 AL bit 0= 0/1 ; disable/enable Code 32 decoding  
 AL bit 1= 0/1 ; not send/send check digit  
 AL bit 2= 0/1 ; not send/send leading character 'A'

Return Value: None

**1.5.17.11 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

**1.5.17.12 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

**1.5.17.13 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

**1.5.17.14 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

**1.5.17.15 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

**1.5.17.16 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

### **1.5.18. 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);
}
```

### **1.5.19. 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;
}
```

### **1.5.20. 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);
}
```

**1.5.21. 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;
}
```

**1.5.22. 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);
}
```

**1.5.23. Set Alarm Date**

Entry Parameter: AH = 2EH  
                   AL = 0 ; disable alarm  
                   ; enable alarm everyday  
                   ; 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);
}
```

**1.5.24. 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);
}
```

**1.5.25. 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);
}
```

**1.5.26. 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 = segregregs.es;
    *bx = regs.x.bx;
}
```

### 1.5.27. 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);
}
```

### 1.5.28. File operation function

#### 1.5.28.1 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 = 0x3c;
    regs.x.dx = (int)fn;
    // segregregs.ds = FP_SEG(fn);
    segread(&segregs);
    intdosx(&regs, &regs, &segregs);
}
```

#### 1.5.28.2 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 = 0x3D;
  regs.h.al = (BYTE)mode;
  regs.x.dx = (int)fn;
  //segregs.ds = FP_SEG(fn);
  segread(&segregs);
  int86(0x21, &regs, &regs, &segregs);
  if ((regs.x.cflag & 0x01) == 1) return(-1);
  else return(regs.x.ax);
}
```

**1.5.28.3 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);
}
```

**1.5.28.4 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)
{
  regs.x.dx = (int)pszBuff;
```

```

//segregs.ds = FP_SEG(pszBuff);
regs.h.ah = 0x3f;
regs.x.cx = nSize;
regs.x.bx = hdl;
segread(&segregs);
int86x(0x21, &regs, &regs, &segregs);

if ((regs.x.cflag & 0x01) == 0) return(regs.x.ax);
else return(-1);
}

```

### 1.5.28.5 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)
{
regs.x.dx = (int)pszBuff;
//segregs.ds = FP_SEG(pszBuff);
regs.h.ah = 0x40;
regs.x.bx = hdl;
regs.x.cx = nSize;
segread(&segregs);
int86x(0x21,&regs,&regs,&segregs);

if ((regs.x.cflag & 0x01) == 0) return(regs.x.ax);
else return(-1);
}

```

### 1.5.28.6 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);
}

```

### 1.5.28.7 **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);
}

```

### 1.5.28.8 **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)

If character is not found:

Carry flag = set

CX = total matched times

DX:AX = pointer to current file position (not changed)

### 1.5.28.9 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)

### 1.5.28.10 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.

**1.5.28.11 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);
}
```

**1.5.28.12 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 flag 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);
    segread(&segregs);
    if ((regs.x.cflag & 0x01) == 1) return(-1);
    else return(regs.x.ax);
}
```

**1.5.29. 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);
}

```

### **1.5.30. 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);
    int86(0x21,&regs,&regs,&segregs);
    if ((regs.x.cflag & 0x01) == 0) return(1);
    else return(-1);
}

```

### **1.5.31. 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);
}

```

### **1.5.32. Run specified program**

Entry Parameter: AH = 4BH  
AL = 3  
DS:DX ; pointer to program name

Return Value: if success, AX = Handle and CARRY flag is cleared  
if failed , AX = 2 and CARRY flag is set

```

void TS_Run(char far *str)
{
    regs.h.ah=0x4b;
    regs.h.al=3;
    segregs.ds=FP_SEG(str);
    segregs.x.dx=FP_OFF(str);
    int86x(0x21, &regs, &regs, &segregs);
}

```

### 1.5.33. 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);
}

```

### 1.5.34. Read Data from Scanner Port

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);
        segregs.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);
}

```

### 1.5.35. 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);
}

```

### 1.5.36. Receive Data from RS232 Port in MULTIPOINT Protocol

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);
}

```

---

**1.5.37. Send Data to RS232 Port in MULTIPOINT Protocol (Buffered Output)**

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) PT/HT630 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);
}
```

**1.5.38. Check RS232 Output Buffer Status in MULTIPOINT Protocol**

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 ; awoken by key  
                   1 ; awoken by barcode scanner  
                   2 ; awoken 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 COM1 RS232 Control Functions ( INT 33H )****1.8.1. 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 960 : cc= 0x80; break;
        case 1920 : cc= 0x90; break;
        case 3840 : cc= 0xA0; 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);
}

```

### 1.8.2. 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);
}

```

---

### 1.8.3. 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);
}
```

### 1.8.4. Enable RS-232 Port

Entry Parameter: AH = 3

Return Value: None

**Example:**

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

### 1.8.5. Disable RS-232 Port

Entry Parameter: AH = 4

Returned Value: None

**Example:**

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

**1.8.6. 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);
}
```

**1.8.7. Get CTS/DSR Signal Status of RS232 Port**

Entry Parameter: AH = 6  
                   AL = 1 ; get CTS  
                   AL = 2 ; get DSR  
 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);
}
```

**1.8.8. Flush RS232 RX buffer**

Entry Parameter: AH = 8

Return Value: None

## 1.9 COM2 (internal COM port) RS232 Control Functions ( INT 14H )

### 1.9.1. 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 960 : cc= 0x80; break;
        case 1920 : cc= 0x90; break;
        case 3840 : cc= 0xA0; 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(0x14,&regs,&regs);
}
```

**1.9.2. Get Character from COM 2 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(0x14,&regs,&regs);
    if (regs.h.ah == 0)
    {
        return(regs.h.al);
    }
    return(255);
}

```

**1.9.3. Send Character to COM2 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(0x14,&regs,&regs);
}

```

**1.9.4. Enable COM2 RS-232 Port**

Entry Parameter: AH = 3

Return Value: None

**Example:**

```

void TC_232_enable()
{
    regs.h.ah = 3;
    int86(0x14,&regs,&regs);
}

```

**1.9.5. Disable COM2 RS-232 Port**

Entry Parameter: AH = 4

Returned Value: None

**Example:**

```

void TC_232_disable()
{
    regs.h.ah = 4;
    int86(0x14,&regs,&regs);
}

```

**1.9.6. 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(0X14,&regs,&regs);
}
void TC_232_DTR(int dtr)
{
  regs.h.ah = 5;
  regs.h.al = 1;
  regs.h.dh = (unsigned char)dtr;
  int86(0X14,&regs,&regs);
}
```

**1.9.7. Get CTS/DSR Signal Status of RS232 Port**

Entry Parameter: AH = 6  
                   AL = 1 ; get CTS  
                   AL = 2 ; get DSR  
 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(0X14,&regs,&regs);
  return((int)regs.h.dh);
}
int TC_232_DSR()
{
  regs.h.ah = 6;
  regs.h.al = 1;
  int86(0X14,&regs,&regs);
  return((int)regs.h.dh);
}
```

**1.9.8. Flush RS232 RX buffer**

Entry Parameter: AH = 8  
 Return Value: None

**1.10. COM2 (internal COM port) Bluetooth Control Functions ( INT 35H )**

HT630 with BT module FW use this COM2 port to communicate with BT module

**1.10.1. Set COM2 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:	xxxxxx07 data bits
		xxxxxx18 data bits

Return Value: None

None: The functions only change the parameters for HT630 COM2.  
To set the parameters for BT module, please use AT commands.

**Example:**

```
int BT_232_param(long lnBaud, int nParity, int nStop, int nData)
{
    BYTE cc = 0;
    unsigned int i_baud;

    i_baud = (int)(lnBaud / 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 (nParity)
    {
        case 0 : break;
        case 1 : cc=cc|0x04; break;
        case 2 : cc=cc|0x0c; break;
    }
}
```

```

switch (nStop)
{
    case 1 : break;
    case 2 : cc=cc10x02; break;
}
switch (nData)
{
    case 7 : break;
    case 8 : cc=cc10x01; break;
}
regs.h.ah = 0;
regs.h.al = cc;
int86(0x35, &regs, &regs);
}

```

### 1.10.2. Get Character From Bluetooth via COM2 port

Entry Parameter: AH = 1

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

Example:

```

BYTE BT_Get_232_Char()
{
    regs.h.ah = 1;
    int86(0x35, &regs, &regs);
    if (regs.h.ah == 0)
    {
        return(regs.h.al);
    }

    return(255);
}

```

### 1.10.3. Send Character To Bluetooth via COM2 port

Entry Parameter: AH = 2

AL = character

Return Value: None

Example:

```

void BT_Send_232_Char(BYTE ch)
{
    regs.h.ah = 2;
    regs.h.al = ch;
    int86(0x35, &regs, &regs);
}

```

### 1.10.4. Enable COM2 & BT Module

Entry Parameter: AH = 3

Return Value: None

Example:

```

void BT_Enable_232()
{
    regs.h.ah = 3;
}

```

```

    int86(0x35, &regs, &regs);
}

```

#### 1.10.5. Disable COM2 & BT Module

Entry Parameter: AH = 4

Returned Value: None

Example:

```

void BT_Disable_232()
{
    regs.h.ah = 4;
    int86(0x35, &regs, &regs);
}

```

#### 1.10.6. Set RTS Signal for COM2 port

Entry parameter: AH = 5

AL = 2

DH = 0 ; set RTS high (enable)

1 ; set RTS low (disable)

Return Value: None

Example:

```

void BT_Set_232_RTS(int nRTS)
{
    regs.h.ah = 5;
    regs.h.al = 2;
    regs.h.dh = (BYTE)nRTS;
    int86(0x35, &regs, &regs);
}

```

#### 1.10.7. Get CTS Signal Status of COM2 port

Entry Parameter: AH = 6

AL = 1 ; get CTS

Return Value: DH = 0 ; CTS high (enable)

; CTS low (disable)

Example:

```

int BT_Get_232_CTS()
{
    regs.h.ah = 6;
    regs.h.al = 1;
    int86(0x35, &regs, &regs);
    return((int)regs.h.dh);
}

```

#### 1.10.8. Clear Receiving Buffer of COM2

Entry Parameter: AH = 8

Return Value: None

Example:

```

void BT_Clear_232_buff()
{
    regs.h.ah = 8;
    int86(0x35, &regs, &regs);
}

```

#### 1.10.9. Warm Start BT Module

Entry Parameter: AH = 0x10

---

**Return Value:** None

**Note:** BT module will reset and start from beginning without no setting change.

**Example:**

```
void BT_WarmStart()
{
    regs.h.ah = 0x10;
    int86(0x35, &regs, &regs);
}
```

#### **1.10.10. Cold Start BT Module**

**Entry Parameter:** AH = 0x11

**Return Value:** None

**Note:** BT module will reset and start from beginning with all settings restored to the factory default values.

**Example:**

```
void BT_ColdStart()
{
    regs.h.ah = 0x11;
    int86(0x35, &regs, &regs);
}
```

#### **1.10.11. Send BT AT Command to BT Module via COM2 Port**

**Entry Parameter:** AH = 0x12  
DS:DX → command buffer

**Return Value:** None

**Example:** DS:DX → “ATB?\0”

**Note:** When send AT command to BT module it will response Result Code. (pls refer to ATQ command)

```
void BT_Send_Command(BYTE *pszData)
{
    regs.h.ah = 0x12;
    regs.x.dx = (int) pszData;
    int86(0x35, &regs, &regs);
}
```

## SPP AT Command Sets

A (Establish Connection)	When it's in master mode. This command establish a connection. When it's in slave mode, this command will be rejected.	
	Command	Description
	A	Connect to Bluetooth Device(It's only available when "ATD=XXXXXXXXXXXX" assigned)
	A1~A8	Connect to Bluetooth neighbourhood device 1~8(ATF? Will list all neighbourhood device)
B (Display local BD address)	This command display the local device BD address	
	Command	Description
	B?	Inquire the Local BD address
C (Flow Control)	This command enable or disable flow control signals(CTS/RTS) of the COM port. Note, the setting is not affected by ATZ0.	
	Command	Description
	C0	Disable flow control
	C1(Default)	Enable flow control
	C?	Inquire the current setting
D (Set Remote BD Address)	For security purpose, We can specifies the unique remote device can be connected. In master role, it automatically inquire and search the slave even the slave is undiscoverable. In slave role, the command should be as a filter condition to accept the master's inquiry.	
	Command	Description
	D=xxxxxxxxxx xx	"xxxx-xx-xxxxxx" is 12 digit hex symbol
	D0(Default)	Clear remote BD address setting, inquire any slave in master or accept any master in slave mode.
	D?	Inquire the remote BD address setting
E (Local Echo)	This command specifies whether the device should echo characters received from the UART back to the DTE/DCE	
	Command	Description
	E0	Command characters received from the UART are not echoed back to the DTE/DCE

	E1(Default)	Command characters received from the UART are echoed back to the DTE/DCE.
	E?	Inquire the current setting.
F (Find Bluetooth device)	This command is used to find any bluetooth device in neighborhood within 60 seconds timeout. If any device is found, its name and address will be listed. The search ends with a message “Inquiry ends, xx devices found.” This command is available only the device is in master role.	
	Command	Description
	F?	Inquire scan Bluetooth neighborhood devices.
H (Discoverable Control)	This command specifies whether the device could be discovered by remote master device. Note:waiting for 15 seconds after ATH1 command to take the effect.	
	Command	Description
	H0	The device enters undiscoverable mode. If a pair have been made, the original connection could be connected again. Other remote master device can not discovery this device.
	H1	The device enters discoverable mode.
	H?	Inquire the current setting
I (Information)	This command is used to inquire the F/W version	
	Command	Description
	I?	Inquire the F/W version
K (Stop bits setting)	This command is used to specify one or two bits of COM port.	
	Command	Description
	K0(Default)	One Stop bit
	K1	Two stop bit
	K?	Inquire the current setting.
L (Baud Rate Control)	This command is used to specify the baud rate of COM port	
	Command	Description
	L0	4800bps
	L1	9600bps
	L2(Default)	19200bps
	L3	38400bps

	L4	57600bps
	L5	115200bps
	L6	230.4Kbps
	L7	460.8Kbps
	L8	921.bKbps
	L?	Inquire the current setting
M (Parity bits setting)	This command is used to specify the parity bit setting of COM port	
	Command	Description
	M0(Default)	None parity bit
	M1	Odd parity setting
	M2	Even parity setting
	M?	Inquire the current setting.
N (Set device Name)	We can specified the device a friendly name using 0 to 9,A to Z, a to z,space and '-', which are all valid characters. Note that “first space or -, last space or – isn't permitted”. The default name is “Serial Adaptor”	
	Command	Description
	N=XXX	“xxx” is a character string, maxima length is 16
	N?	Inquire the device name
O (Auto connect setting)	When it's in master mode, This command is used to enable or disable auto-connection feature. When it's in slave mode, this command is rejected.	
	Command	Description
	O0 (Default)	Automatically connecting to a device which is assigned in “ATD” or any available device if “ATD” was not assigned.
	O1	Disable auto-connecting feature. User should manually use “ATA” command to connect a remote device.
O?	Inquire the current setting	
P (Set PIN code)	This command specifies the PIN number. It control to off the PIN code authorization that allow to establish a connection without PIN code. Default PIN number is “1234”	
	Command	Description

	P=XXXX (Default)	“xxxx” is 4-8 digit string
	P0	Turn off the PIN code authorization
	P?	Inquire the PIN number
Q (Result Code Supression)	This command is used to determine if result Codes should be sent to the DTE/DCE. When reuslt codes are supressed, the devide does not generate any characters in response to the completion of a command or when an event occurs. Four Resutl Codes:OK,CONNECT,DISCONNECT,ERROR	
	Command	Description
	Q0(Default)	The device will send Result Code to DTE/DCE
	Q1	The device will not send Result Code to DTE/DCE
	Q?	Inquire the current setting
R (Set Role)	This command specifies whether the device should be master or slave device. If change the role, the adaptor will warm start and clear all paired address.	
	R0	The device as master role
	R1	The device as slave role
	R?	Inquire the current setting
U (F/W upgrade)	This command will prompt “Enter DFU mode, Are you sure(y/n)?” message, then press Y to confirm the command. Then you should connect USB cable to PC and run DFU wizard(DFU wizard please contact us <a href="http://www.rayson.com">www.rayson.com</a> )	
	Command	Description
	U=password	Password=RaysonUpgrade, Go into upgrae F/W Mode.

The Factory Settings of Bluetooth module are as follows:

Baud rate:19200

Data bit:8

Parity:none

Stop bit:1

Flow control:H/W or none

## Chapter 2. MULTIPOINT Communication

### 2.1 MULTIPOINT Communication Protocol

MULTIPOINT communication protocol is supported by built-in communication tool on PT/HT630. 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 PT/HT630 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 PT/HT630 terminals, while each PT/HT630 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 PT/HT630 to response. The maximum packet size including protocol control characters is 256 bytes.

#### 2.1.1 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

#### 2.1.2 Fast checking whether terminal is on-line ( BELL )

Format:

⇒ BELL ADDR ; <addr>='A'~'Y' or '0'~'6'  
 ⇐ STX ESC BELL ? ETX ;  
 ; = 01H error  
 ⇒ ACK or NAK

Description: The command will change PT/HT630 MULTIPOINT address.

**2.1.3 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

**2.1.4 PT/HT630 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

**2.1.5 Data Conversion Rules in Packet**

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

\	convert to	\\
<u>00 hex</u> -- <u>1F hex</u>	convert to	\ <u>80 hex</u> -- \ <u>9F hex</u>
<u>A0 hex</u> -- <u>FF hex</u>	convert to	\ <u>20 hex</u> -- \ <u>7F hex</u>

(excluding DC hex)

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

**2.1.6 Data Checksum Calculation in Packet**

- CS = [sum of all characters (excluding STX and ETX) + packet length (excluding STX and ADDR/ETX)] MOD 256
- CS1 = high nibble of CS + 40H
- CS2 = low nibble of CS + 40H

**2.1.7 Example packet of command to send the file named A.EXE to PT/HT630 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



**2.3.2 Host Send File to PT/HT630 ( Download ) from specified file position**

- ⇒ STX ESC 'L' <filename> [offset] 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

Note : [offset] mean file's starting pointer. [offset] E.x. [1000] mean to start download specified file from 1000st bytes.

**2.3.3 Cancel Current Downloading File Command (ESC z)**

- Format: ⇒ STX ESC 'z' CS1 CS2 ADDR
- ⇐ ACK or NAK

**2.3.4 Host Requests File from the PT/HT630 ( 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
- ⇒ ACK or NAK
- ⇐ STX ESC 'Z' CS1 CS2 ETX ; end of file transfer
- ⇒ ACK or NAK

Description: For uploading, the host must know the filename on PT/HT630. Get Directory Command (ESC 'D') can be used to view all filenames on PT/HT630 (see 4.4 Host Commands).

**2.3.5 Host Requests File from the PT/HT630 ( Upload ) from specified file positon**

- Format: ⇒ STX ESC 'U' <filename> [offset] 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
- ⇒ ACK or NAK
- ⇐ STX ESC 'Z' CS1 CS2 ETX ; end of file transfer
- ⇒ ACK or NAK

Note : [offset] mean file's starting pointer. [offset] E.x. [1000] mean to start download specified file from 1000st bytes.

Description: For uploading, the host must know the filename on PT/HT630. Get Directory Command (ESC 'D') can be used to view all filenames on PT/HT630 (see 4.4 Host Commands).

**2.3.6 Cancel Current Uploading File Command (ESC y)**

- Format: ⇒ STX ESC 'y' CS1 CS2 ADDR
- ⇐ ACK or NAK



#### 2.4.4 Communication Configuration ( ESC C )

Format: ⇒ STX ESC 'C' <comtable> CS1 CS2 ADDR  
 ⇐ STX ESC 'C' < retcode> CS1 CS2 ETX ;< retcode> = 00H successful  
 ; 01H error  
 ⇒ ACK or NAK

Description: The command writes <comtable> to PT/HT630 communication control table.

< comtable> is defined as:

```
typedef struct {
    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 PT/HT630 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 PT/HT630 will switch to 1200 right after the command has been received. The next host communication will use 1200 baud rate.

#### 2.4.5 Get File Directory in RAM Disk ( ESC D )

Format: ⇒ STX ESC 'D' CS1 CS2 ADDR  
 ⇐ STX ESC 'D' < filename,filename... > CS1 CS2 ETX  
 ⇒ ACK or NAK

if If the last character of < **filename,filename...** > is it mean directory list is too big for single packet. Please issue ESC D command again  
 ⇒ STX ESC 'D' CS1 CS2 ADDR  
 ⇐ STX ESC 'D' < filename,filename... > CS1 CS2 ETX  
 ⇒ ACK or NAK

Description: PT/HT630 will send its file directory to the host.

**2.4.6 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  
 ⇒ ACK or NAK

Description: PT/HT630 will send program names in FLASH memory to the host.

**2.4.7 Erase File (ESC E)**

Format: ⇒ STX ESC 'E' <filename> CS1 CS2 ADDR  
 ⇐ STX ESC 'E' < retcode> CS1 CS2 ETX ; < retcode> = 00H successful  
 ; 01H error  
 ⇒ ACK or NAK

Description: Used to erase file in PT/HT630.

**2.4.8 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  
 ⇒ ACK or NAK

Description: If PT/HT630 RAM disk has files, its size can't be changed (< retcode> = 01H).

**2.4.9 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  
 ⇒ ACK or NAK

**2.4.10 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  
 ⇒ ACK or NAK

**2.4.11 Cold Start (ESC H)**

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

Description: The command clears all PT/HT630 RAM memory content. It performs tests on all major hardware devices. Programs or data that have previously accumulated in the PT/HT630 or previously been downloaded by the host will be purged from the memory. Default system parameters are restored from the EPROM.

**2.4.12 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  
 or ⇐ STX ESC 'I' < prg-name/ROM> CS1 CS2 ETX  
     ; if FLASH program is running  
 or ⇐ STX ESC 'I' < retcode> CS1 CS2 ETX  
     ; if no program is running < retcode> = 01H  
 ⇒ ACK or NAK

**2.4.13 Check whether File/Program Exists (ESC J)**

Format: ⇒ STX ESC 'J' <filename> CS1 CS2 ADDR  
 ⇐ STX ESC 'J' < retcode> < length> CS1 CS2 ETX  
     ;< retcode> = 0 file in RAM  
     ;          1 no file  
     ;          2 file in FLASH  
     ;< length> : string if < retcode> = 0 or 2.

⇒ ACK or NAK

**2.4.14 Enter Kermit Server Mode (ESC k)**

Format: ⇒ STX ESC 'k' CS1 CS2 ADDR  
 ⇐ ACK or NAK

Description: PT/HT630 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.

**2.4.15 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  
 ⇒ ACK or NAK

Description: This command allows the host system to set PT/HT630 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 PT/HT630 clock to September 26, 1990. The time is 11:45 PM. The PT/HT630 reconfigures the real time clock chip as soon as the command has been successfully received.

**2.4.16 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

**2.4.17 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  
 ⇒ ACK or NAK

Description: The command changes the password for PT/HT630 Supervisor Mode.

**2.4.18 Get Terminal ID (ESC R)**

Format: ⇒ STX ESC 'R' CS1 CS2 ADDR  
 ⇐ STX ESC 'R' < ID> CS1 CS2 ETX  
 ⇒ ACK or NAK

Description: <ID> is 8-characters fixed length string and its default value is "PT/HT630".

### 2.4.19 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  
 ⇒ ACK or NAK

Description: The command set the configuration for PT/HT630 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.

### 2.4.20 Device Configuration ( ESC V )

Format: ⇒ STX ESC 'V' <devtable> CS1 CS2 ADDR  
 ⇐ STX ESC 'V' <retcode> CS1 CS2 ETX ;< retcode> = 00H success  
 ; 01H error  
 ⇒ ACK or NAK

Description: The command sets configuration of PT/HT630 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.

**2.4.21 Get Portable Model Number and BIOS Version Number (ESC v)**

Format: ⇒ STX ESC 'v' CS1 CS2 ADDR  
 ⇐ STX ESC 'v' 'PT/HT630 V5.00' CS1 CS2 ETX  
 ⇒ ACK or NAK

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

**2.4.22 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  
 ⇒ ACK or NAK

**2.4.23 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  
 ⇒ ACK or NAK

## Chapter 3. Update Note

### 1. Correct error return value on BX on section 1.5.25 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= 830H

### 2. Correct error entry value on 1.2.1 **Select Large Font**

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

Return Value: None

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

### 3. Correct error entry value for DSR 1.8.7 Get CTS/DSR Signal Status of RS232 Port

Entry Parameter: AH = 6

AL = ~~2~~1 ; get CTS

AL = ~~1~~2 ; get DSR

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);
}
```

**4. Error entry value on 1.8.8 Flush RS232 RX buffer**

Entry Parameter: AH = 08

Return Value: None

**5. Add COM2 RS232 function call on section 1.9**

**6. Add reverse display under small font mode on 1.2.9**

**7. Keymap table error on 1.5.15 Set User-defined Key-map**

Entry Parameter: AH = 1EH

AL = 0

DS:DX ; pointer to Key-map buffer with 5\*32 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 Mode			CMD Mode		
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	7	37	32	S	53	64	T	54	96	U	55	128	-	2D
1	4	34	33	J	4A	65	K	4B	97	L	4C	129	:	3A
2	1	31	34	A	41	66	B	42	98	C	43	130	#	23
3	CLR	08	35	CLR	08	67	CLR	08	99	CLR	08	131	\	5c
4		0	36		0	68		0	100		0	132		84
5	F1	86	37	F1	86	69	F1	86	101	F1	86	133	F5	8A
6	ENTER	0D	38	ENTER	0D	70	ENTER	0D	102	ENTER	0D	134	ENTER	0D
7		0	39		0	71		0	103		0	135		0
8	8	38	40	V	56	72	W	57	104	X	58	136	+	2B
9	5	35	41	M	4D	73	N	4E	105	Q	4F	137	= 3D	24
10	2	32	42	D	44	74	E	45	106	F	46	138	\$	24
11	0	30	43	@	40	75	?	3F	107	&	26	139	'	27
12	◀□	11	44	◀□	11	76	◀□	11	108	◀□	11	140	◀□	11
13	▲□	13	45	▲□	13	77	▲□	13	109	▲□	13	141	PgDn	93
14	F2	87	46	F2	87	78	F2	87	110	F2	87	142	F6	8B
15		0	47		0	79		0	111		0	143		00
16	9	39	48	Y	59	80	Z	5A	112	-	5F	144	*	2A
17	6	36	49	P	50	81	Q	51	113	R	52	145	/	2F
18	3	33	50	G	47	82	H	48	114	I	49	146	%	25
19	.	2E	51	:	3B	83	.	2E	115	.	2C	147	!	21
20	▶□	10	52	▶□	10	84	▶□	10	116	▶□	10	148	▶□	10
21	▼□	12	53	▼□	12	85	▼□	12	117	▼□	12	149	PgUp	92
22	F3	88	54	F3	88	86	F3	88	118	F3	88	150	F7	8C
23		0	55		0	87		0	119		0	151		0
24		0	56		0	88		0	120		0	152		0
25		0	57		0	89		0	121		0	153		0
26		0	58		0	90		0	122		0	154		0
27		84	59		84	91		84	123		84	155		84
28	SP	20	60	SP	20	92	SP	20	124	SP	20	156	SP	20
29		80	61		00	93		00	125		00	157		80
30	F4	89	62	F4	89	94	F4	89	126	F4	89	158	F8	8D
31		0	63		0	95		0	127		0	159		0



- 
8. Add Multi-protocol command “Bell” on section **Error! Reference source not found.**
  9. Add Aim mode for long range engine on **1.5.12.13 Set Aim mode for Long Range engine**
  10. Add alpha lower/upper case mode on **1.5.12.14 Set Alpha input mode**
  11. Add Code 32 function call on 1.5.17.10
  12. Add Bluetooth Control function on 1.10

## V1.3

1. Add reverse display on section v1.4.9

## V1.4

Combine HT630 and PT630 to single document.. Basically, HT630 and PT630 is fully compatible except KERMIT function on chapter 1.3

## V1.5

1. Add BT control function on chapter 1.10