The Embedded-based Teach Pendant optimized for industrial robots.

DTP10-L
API Manual

DAINCUBE Corp.
ARM Cortex-A9 Linux system

FORM 140108F – 2018.01.11

DTP10-L API manual
Form 140108F-171206— January, 2018

DAINCUBE Corp.
Web: www.daincube.com
E-mail: sales@daincube.com
Tel: 82-32-329-9783~4
Fax: 82-32-329-9785

#401-701, Bucheon TechnoPark 4-Danji,
655 Pyeongcheon-ro, Wonmi-gu, Bucheon-Si,
Gyeonggi-Do, Republic of Korea

Preface

Copyright notice

Important information

This documentation is intended for qualified audience only. The product described herein is not an end user product. It was developed and manufactured for further processing by trained personnel.

Disclaimer

Although this document has been generated with the utmost care no warranty or liability for correctness or suitability for any particular purpose is implied. The information in this document is provided “as is” and is subject to change without notice.

Trademarks

All used product names, logos or trademarks are property of their respective owners.

Product support

DAINCUBE Corp.

Web: www.daincube.com

E - MAIL: sales@daincube.com

Safety precautions

Be sure to observe all of the following safety precautions.

Strict observance of these warning and caution indications are a MUST for preventing accidents, which could result in bodily injury and substantial property damage. Make sure you fully understand all definitions of these terms and related symbols given below, before you proceed to the manual.

Safety precautions

The following symbols may be used in this specification:

⚠ Warning:

Warnings indicate conditions that, if not observed, can cause personal injury.

⚠ Caution :

Cautions warn the user about how to prevent damage to hardware or loss of data.

✎ Note:

Notes call attention to important information that should be observed.

Revision history

Revision	Data	Comment
Version 0.1 (Preliminary)	2018.01.11	Preliminary version

Contents

1. Introduction

Thank you very much for purchasing our products.
This document describes an example program for controlling Key, LED and Buzzer on DTP10-L.
If you will control Key, LED and Buzzer, you need to develop serial communication program. However, if it is difficult to develop serial communication program, refer to the event filter function in this document. (Event filter function can only control LED and Buzzer.) This program is only sample program. So, you must to modify it for your application.

2. How to use Serial Daemon

2.1. COM port open

Key, LED and Buzzer use ttyACM0 COM port on DTP10-L. Therefore ttyACM0 port must be opened and function used are as follows.

```
fd= open ( SERPORT, O_RDWR | O_NOCTTY);
if( fd < 0 ) {
    // Serial Port Open Fail
} else {
  // Serial Port Open Success
}
```

If the serial port is successfully opened, set additional settings for serial communication.
Example)

```
/*
 * Function name : Serial_Port_Open
 * Description   : Open serial port (ttyACM0) and initialize of serial port.
 */
void Serial_Daemon::Serial_Port_Open()
{
    struct termios newtio;

    /* Serial port open */
    fd = open( SERPORT, O_RDWR | O_NOCTTY);
    if ( fd < 0 )
    {
        /* Serial port open fail */
        g_com_flag = 0;
        ui->Btn_Open->setText("OPEN");
        ui->Btn_Open->setEnabled(1);
        return;
    }
    else
    {
        /* Serial port init */
        memset(&newtio, 0, sizeof(newtio));
        newtio.c_iflag = IGNPAR; // non-parity
        newtio.c_oflag = 0;
        newtio.c_cflag = CS8 | CLOCAL | CREAD | BAUD_RATE; // NO-RTS/CTS

        /* Set input mode (non-canonical, no echo,.....) */
        newtio.c_lflag = 0;
        newtio.c_cc[VTIME] = TIME;
        newtio.c_cc[VMIN] = MIN;
```

```
        /* Flush the input and output */
        tcflush ( fd, TCIOFLUSH );
        /* Set the new options for the port */
        tcsetattr( fd, TCSANOW, &newtio );
        notRsRead = new QSocketNotifier(fd, QSocketNotifier::Read, this);
        connect(notRsRead, SIGNAL(activated(int)), this, SLOT(Receive_Event()));
        ui->Btn_Open->setEnabled(1);
        /* Ring buffer create */
        m_abyBuffer.Create(BUFF_MAX);
        Open_uinput_port();
    }
}
```

2.2. Write packet to serial port

The method of writing packet to serial port is as follows. Transmit serial packet through the write function.
Example)

```
/*
 * Function name : on_Btn_LED1_clicked
 * Description   : This function is control Right LED 1.
 */
void Serial_Daemon::on_Btn_LED1_clicked()
{
    char LED_Packet[10] = {0, };
    unsigned int crc_buf;
    unsigned long dwBytes = 0;
    static char i = 0;

    LED_Packet[0] = STX;          // STX
    LED_Packet[1] = MOD_SET;      // MOD (get : 0x10, set : 0x11)
    LED_Packet[2] = SEL_LED;      // SEL (LED : 0x3A)
    LED_Packet[3] = RIGHT_LED1;           // Data1 (SUB ADDRESS)
    if ( i == 3 )
        LED_Packet[4] = LED_OFF;      // Data2 (off : 0x30, blue : 0x31, red : 0x32, all :
0x33)
    else
        LED_Packet[4] = LED_BLUE + i;
    LED_Packet[5] = DATA_RESERVED;        // Data3 (Reserved : 0x20)
    crc_buf = crc16_append(LED_Packet,6);
    LED_Packet[6] = (char)(crc_buf>>8)&0xff;
    LED_Packet[7] = (char)crc_buf&0xff;
    LED_Packet[8] = ETX;          // ETX
    LED_Packet[9] = '₩0';

    dwBytes = strlen(LED_Packet);
    write( fd, LED_Packet, dwBytes );
    usleep(1000);

    switch (i) {
        case 0 :
            ui->Btn_LED1->setText("BLUE");
            i++;
            break;
        case 1 :
            ui->Btn_LED1->setText("RED");
            i++;
```

```
            break;
        case 2 :
            ui->Btn_LED1->setText("ALL");
            i++;
            break;
        case 3 :
            ui->Btn_LED1->setText("OFF");
            i = 0;
            break;
    }
}
```

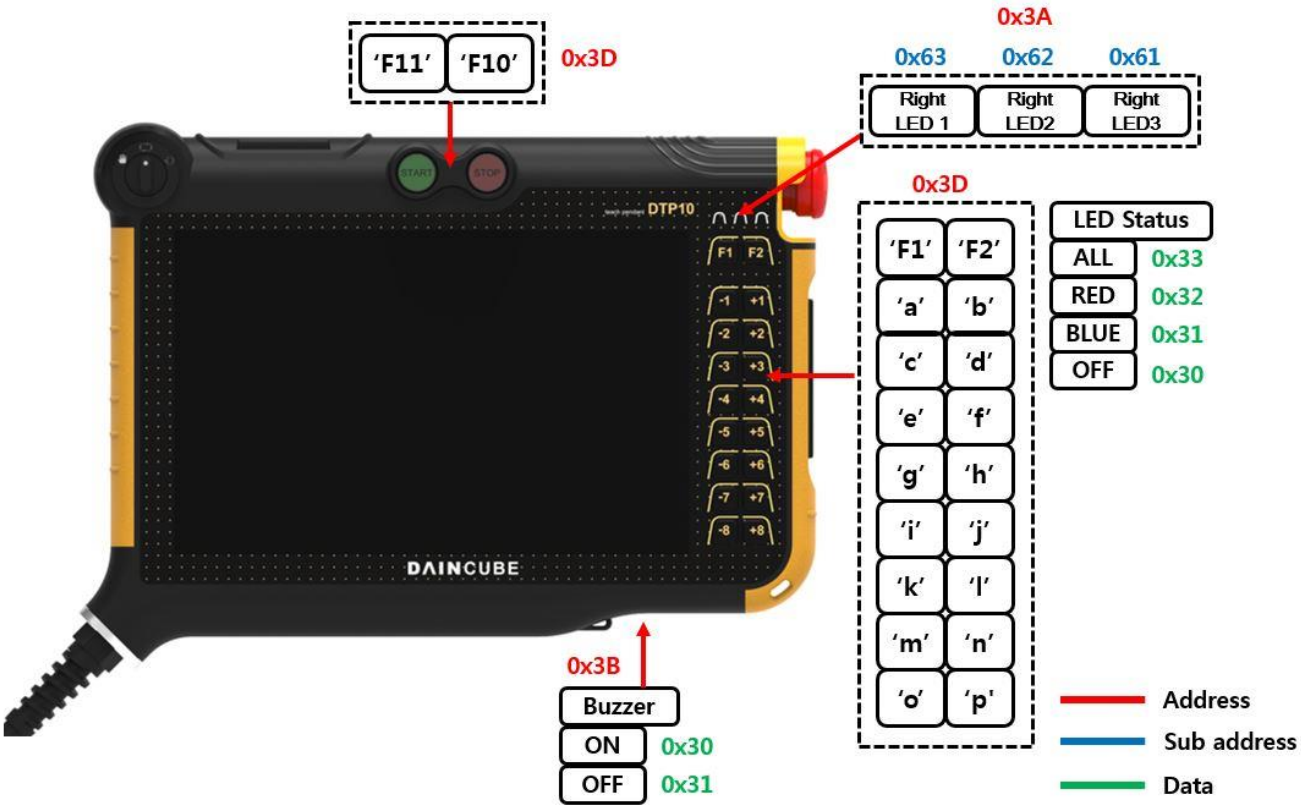
2.3. Read from serial port and write ring buffer

Read to serial packet and fill in the ring buffer.

```
/*
 * Function name : Receive_Event
 * Description   : This function takes packet and stacks it in the ring buffer.
 */
void Serial_Daemon::Receive_Event()
{
    char Receive_Packet[BUFF_MAX];
    int temp;
    memset(Receive_Packet,0,sizeof(Receive_Packet));
    read(fd, Receive_Packet, BUFF_MAX);
    m_abyBuffer.WriteBinary(Receive_Packet, BUFF_SIZE); //write ring buffer
    if(m_abyBuffer.FindChar(STX, temp)){
        if(temp == 10){
            return;
        }
        mThreadKEY->start();
    }
}
```

3. Defines of ETC values

3.1. Description



4. How to control ETC driver

4.1. Example using LED

Control the LED of DTP10-L by using serial daemon program provided by Daincube.

- Serial COM port open
- Packet buffer create.
- CRC create and fill in the packet buffer
- Transmit serial packet
- Serial COM port close

You can control LED of DTP10-L by transmit serial packet as below.

STX	MOD	SEL	DATA1	DATA2	DATA3	CRC_H	CRC_L	ETX
0x02	0x11	0x3A	0x63	0x33	0x20	0xXX	0xXX	0x03
1BYTE	1BYTE	1BYTE	1BYTE	1BYTE	1BYTE	1BYTE	1BYTE	1BYTE

MOD : MOD_GET = 0x10, MOD_SET = 0x11

SEL : SEL_LED = 0x3A

DATA1 : RIGHT_LED1 = 0x63, RIGHT_LED2 = 0x62, RIGHT_LED3 = 0x61

DATA2 : LED_OFF = 0x30, LED_BLUE = 0x31, LED_RED = 0x32, LED_ALL_ON = 0x33

DATA3 : DATA_RESERVED = 0x20

```
/*
 * Function name : LED_Set
 * Description  : This function is control of LED and each led has its own value.
 *                If you control LED, using this function.
 */
void Serial_Daemon::LED_Set(char sel, char data)
```

```
{
    char LED_Packet[10] = {0, };
    unsigned int crc_buf;
    unsigned long dwBytes = 0;
    LED_Packet[0] = STX;          // STX
    LED_Packet[1] = MOD_SET;        // MOD (get : 0x10, set : 0x11)
    LED_Packet[2] = SEL_LED;        // SEL (LED : 0x3A)
    LED_Packet[5] = DATA_RESERVED;      // Data3 (Reserved : 0x20)
    LED_Packet[8] = ETX;          // ETX
    LED_Packet[9] = '₩0';

    switch(sel) {
        case RIGHT_LED1:
            LED_Packet[3] = RIGHT_LED1;
            break;
        case RIGHT_LED2:
            LED_Packet[3] = RIGHT_LED2;
            break;
        case RIGHT_LED3:
            LED_Packet[3] = RIGHT_LED3;
            break;
        case LED_ALL:
            LED_Packet[3] = LED_ALL;
    }

    if (data & 0x1) {                 // Blue color
        LED_Packet[4] = LED_BLUE;
    } else if (data & 0x2) {             // Red color
        LED_Packet[4] = LED_RED;
    } else if (data & 0x4) {             // All color
        LED_Packet[4] = LED_ALL_ON;
    } else {                  // LED Off
        LED_Packet[4] = LED_OFF;
    }

    crc_buf = crc16_append(LED_Packet,6);
    LED_Packet[6] = (char)(crc_buf>>8)&0xff;
    LED_Packet[7] = (char)crc_buf&0xff;

    dwBytes = strlen(LED_Packet);
    write( fd, LED_Packet, dwBytes );
    usleep(1000);
}
```

4.2. Examples using Buzzer

Control the Buzzer of DTP10-L by using serial daemon program provided by Daincube.

- Serial COM port open
- Packet buffer create.
- CRC create and fill in the packet buffer
- Transmit serial packet
- Serial COM port close

You can control Buzzer of DTP10-L by transmit serial packet as below.

STX	MOD	SEL	DATA1	DATA2	DATA3	CRC_H	CRC_L	ETX
0x02	0x11	0x3B	0x31	0x20	0x20	0xXX	0xXX	0x03

1BYTE	1BYTE	1BYTE	1BYTE	1BYTE	1BYTE	1BYTE	1BYTE	1BYTE

MOD : MOD_GET = 0x10, MOD_SET = 0x11

SEL : SEL_BUZZ = 0x3B

DATA1 : BUZZ_OFF = 0x30, BUZZ_ON = 0x31

DATA2 : 0x20 = DATA_RESERVED

DATA3 : 0x20 = DATA_RESERVED

```
/*
 * Function name : BUZ_Set
 * Description   : This function is control of Buzzer and Buzzer on,off has its own value.
 *                 If you control Buzzer, using this function.
 */
void Serial_Daemon::BUZ_Set(char data)
{
    char buf_printf[10] = {0, };
    unsigned int crc_buf;
    unsigned long dwBytes = 0;

    buf_printf[0] = STX;            // STX
    buf_printf[1] = MOD_SET;          // MOD (get : 0x10, set : 0x11)
    buf_printf[2] = SEL_BUZZ;         // SEL (BUZZ : 0x3B)
    if ( data == BUZZ_ON )
        buf_printf[3] = BUZZ_ON;        // Data1 (on : 0x31, off : 0x30)
    else if ( data == BUZZ_OFF )
        buf_printf[3] = BUZZ_OFF;
    buf_printf[4] = DATA_RESERVED;        // Data2 (Reserved : 0x20)
    buf_printf[5] = DATA_RESERVED;        // Data3 (Reserved : 0x20)
    crc_buf = crc16_append(buf_printf,6);
    buf_printf[6] = (char)(crc_buf>>8)&0xff;
    buf_printf[7] = (char)crc_buf&0xff;
    buf_printf[8] = ETX;            // ETX
    buf_printf[9] = '₩0';

    dwBytes = strlen(buf_printf);
    write( fd, buf_printf, dwBytes );
    usleep(1000);
}
```

4.3. Examples using Keypad

Receive keypad event of DTP10-L by using serial daemon program provided by Daincube.

- Serial COM port open
- Packet buffer create.
- CRC create and fill in the packet buffer
- Receive serial packet
- Receive packet check and parsing
- Serial COM port close

You can check the keypad status of DTP10-L by transmit/receive serial packet as below.

STX	MOD	SEL	DATA1	DATA2	DATA3	CRC_H	CRC_L	ETX
0x02	0x10	0x3D	0x30	0x33	0x30	0xXX	0xXX	0x03
1BYTE	1BYTE	1BYTE	1BYTE	1BYTE	1BYTE	1BYTE	1BYTE	1BYTE

MOD : MOD_GET = 0x10

SEL : SEL_KEYPAD = 0x3D

DATA1 : KEYPAD_UP = 0x30, KEYPAD_DOWN = 0x31
DATA2 : KEY_A = 30, KEY_B = 48, KEY_C = 46, KEY_D = 32, KEY_E = 18, KEY_F = 33, KEY_G = 34,
KEY_H = 35, KEY_I = 23, KEY_J = 36, KEY_K = 37, KEY_L = 38, KEY_M = 50, KEY_N = 49,
KEY_O = 24, KEY_P = 25, KEY_F1 = 59, KEY_F2 = 60, KEY_F10 = 68, KEY_F11 = 87
DATA3 : DATA_RESERVED = 0x20

```
/*
 * Function name : OperThread
 * Description   : This function is a parsing packets in ring buffer.
 */
void Serial_Daemon::OperThread()
{
    int ptr_start, ptr_end;
    unsigned int crc_buf;
    /* check STX and ETX in packet */
    m_abyBuffer.FindChar(STX,ptr_start);
    m_abyBuffer.FindChar(ETX,ptr_end);
    if((ptr_start >= ptr_end) && (ptr_end - ptr_start != 8)){
        return;
    }
    ... 중략 ...
    /* Pasing packet */
    if(g_Packet_Buffer[1] != MOD_GET){
        return;
    }
    if(g_Packet_Buffer[2] != SEL_KEYPAD){
        return;
    }
    crc_buf = crc16_append(g_Packet_Buffer,6);
    if((g_Packet_Buffer[6] !=(char)((crc_buf>>8)&0xff)) ||
(g_Packet_Buffer[7]!=(char)(crc_buf&0xff))){  //CRC Check
        return;
    }
    if(g_Packet_Buffer[3] == KEYPAD_DOWN){
        Keydown_Event(g_Packet_Buffer);
    }
    if(g_Packet_Buffer[3] == KEYPAD_UP){
        Keyup_Event(g_Packet_Buffer);
    }
}

/*
 * Function name : Keydown_Event
 * Description   : This functin is keydown event.
 *                 If you want to output consecutive key values, add a sync event.
 */
void Serial_Daemon::Keydown_Event(char key_down[10])
{
    struct input_event event;
    char Receive_Packetf[10] = {0, };
    strcpy(Receive_Packetf,key_down);
    //Input key event
    memset(&event,0,sizeof(event));
    event.type = EV_KEY;
    event.code = Receive_Packetf[4];
```

```
    event.value = 1;
    write(key_fd,&event,sizeof(event));
}

/*
 * Function name : Keyup_Event
 * Description   : This function is keyup event.
 */
void Serial_Daemon::Keyup_Event(char key_up[10])
{
    struct input_event event;
    char Receive_Packetf[10] = {0, };
    strcpy(Receive_Packetf,key_up);
    //Input key event
    memset(&event,0,sizeof(event));
    event.type = EV_KEY;
    event.code = Receive_Packetf[4];
    event.value = 0;
    write(key_fd,&event,sizeof(event));
    //Sync event
    memset(&event,0,sizeof(event));
    event.type = EV_SYN;
    event.code = SYN_REPORT;
    event.value = 0;
    write(key_fd,&event,sizeof(event));
}
```

4.4. Examples using event filter

Control LED and Buzzer of DTP10-L by using event filter of serial daemon program provided by Daincube. If you makes keyboard event of predefined key code, you can control LED and buzzer. **However, it will work only active mode. Do not work inactive mode (hide mode, hidden mode, etc.)** Therefore you can refer to the event filter section of serial daemon ini your application to control the LE and buzzer through keyboard events.

```
/*  Function name : eventFilter
 *  description   : This function is receive the keyboard events and control the LED or Buzzer.
 *                  It is possible only when serial daemon application is active mode.
 *                  If you want to control the LED or Buzzer using keyboard events in your
application, refer to this function.
*/
bool Serial_Daemon::eventFilter(QObject *object, QEvent *event)
{
    char sel, data;
    QKeyEvent *Keyevent = static_cast <QKeyEvent *>(event);
    if(event->type() == QEvent::KeyPress)
    {
        if((Keyevent->key() == Qt::Key_A) ||
            (Keyevent->key() == Qt::Key_B) ||
            (Keyevent->key() == Qt::Key_C) ){
            data = 0x1; //blue
        }
        else if((Keyevent->key() == Qt::Key_D) ||
                (Keyevent->key() == Qt::Key_E) ||
                (Keyevent->key() == Qt::Key_F) ){
```

```
                data = 0x2; //red
        }
        else if((Keyevent->key() == Qt::Key_G) ||
                (Keyevent->key() == Qt::Key_H) ||
                (Keyevent->key() == Qt::Key_I) ){
                data = 0x4; //all
        }
        else if((Keyevent->key() == Qt::Key_J)){
            BUZ_Set(BUZZ_ON);
            usleep(1000);
        }
    }
    else if (event->type() == QEvent::KeyRelease){
        data = 0x0;
        LED_Set(sel,data);
        usleep(1000);
        BUZ_Set(BUZZ_OFF);
        usleep(1000);
    }
    switch(Keyevent->key()){
        case Qt::Key_A:
        case Qt::Key_D:
        case Qt::Key_G:
            sel = RIGHT_LED1;
            break;
        case Qt::Key_B:
        case Qt::Key_E:
        case Qt::Key_H:
            sel = RIGHT_LED2;
            break;
        case Qt::Key_C:
        case Qt::Key_F:
        case Qt::Key_I:
            sel = RIGHT_LED3;
            break;
        default:
            break;
    }
    LED_Set(sel,data);
    return QObject::eventFilter(object,event);
}
```