The Embedded-based Teach Pendant optimized for industrial robots.

DTP10-P,D
API Manual

DAINCUBE Corp.
Intel Atom Base Windows System

FORM 140108F – 2018.01.15

DTP10-P,D 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 1.0	2019.03.06	Preliminary version

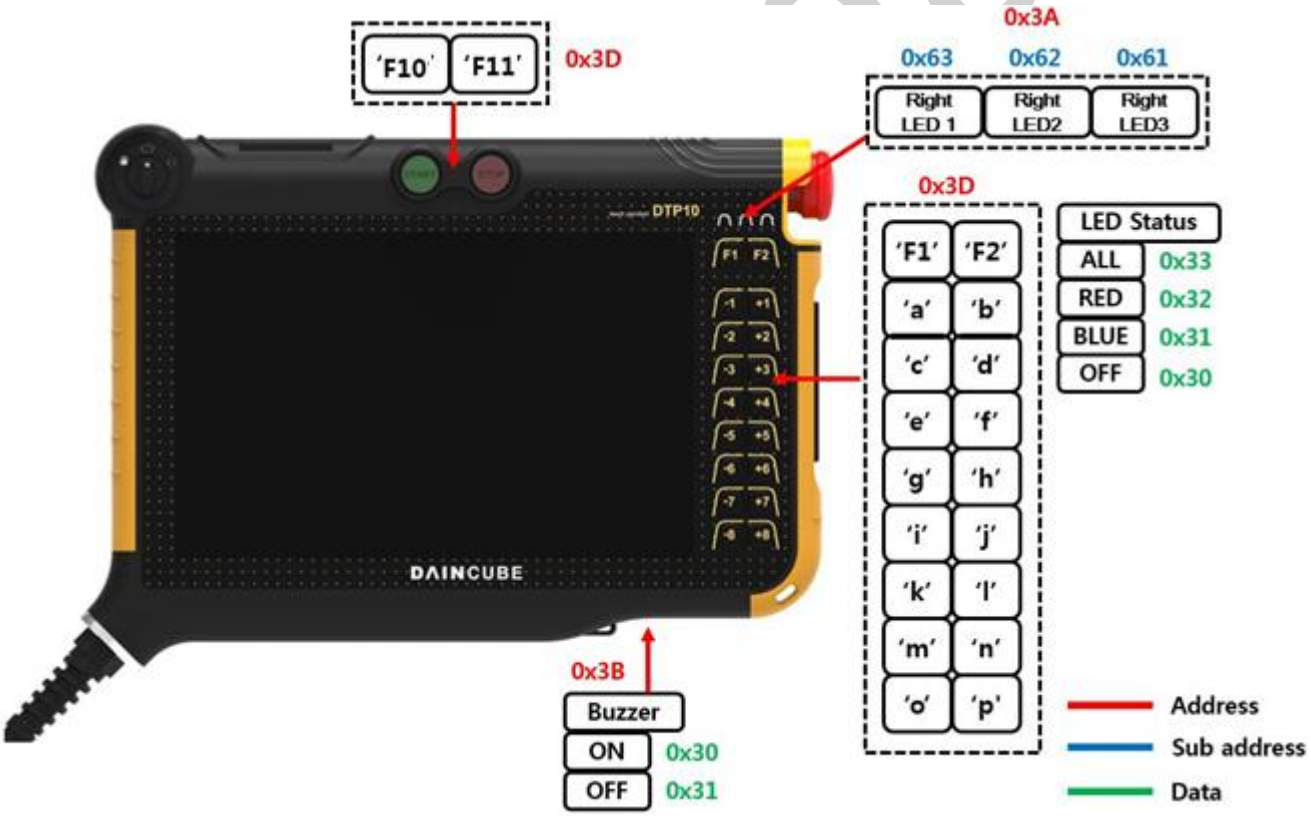
Contents

1. Introduction

This document describes the DTP10-P,D's Key, LED, Buzzer (ETC driver) to help you develop your application more easily. Daincube provides all device drivers and examples for application developers.

2. Serial Daemon application

You can easily use "Keypad, LED and Buzzer" of DTP10-P,D through "Serial Damon application.
To use "Keypad, LED and Buzzer", "Serial Daemon application" must be running. If not, please refer to "User's manual" document.
DTP10-P,D's "Keypad, LED, Buzzer" are mapped to keyboard value for convenience. You can easily develop applications using the Keyboard processing structure.
"Keypad" can receive the keypad value using the keyboard processing structure. "LED, Buzzer" can control "LED, Buzzer" by generating "Virtual Keyboard Event" with each mapped Keycode value.



2.1. Use Keypad

DTP10-P,D's keypads operate the same as the keyboard, so you can use the structure that receives and processes the keyboard value. If you press the keypad, key event of keyboard will occur as below.

2.1.1. Example

```
BOOL CserialDlg::PreTranslateMessage(MSG* pMsg)
{
   // TODO: Add your specialized code here and/or call the base class
   BOOL bHandled = FALSE;

   switch (pMsg->message) {
      case WM_DESTROY:
         break;
      case WM_KEYUP:
         if(pMsg->wParam == VK_F6) {
...... ellipsis
         }
         break;
      case WM_KEYDOWN:
         if(pMsg->wParam == VK_F6) {
...... ellipsis
         }
         break;
      default:
         break;
   }

   return CDialog::PreTranslateMessage(pMsg);
}
```

2.2. Use LED, Buzzer

DTP10-P,D LED and Buzzer can generate "Virtual Keyboard Event" to control LED and Buzzer.

Control LED and Buzzer through each mapped keycode and up/down.

※ LED and Buzzer should be delayed by 1ms before transferring to the serial port since the timer interrupt is confirmed in 1ms cycle.

Device	Color	Virtual Key code	Virtual Key Flags (up/Down)
Right LED1	Blue	0xCA	ON : 0x0 OFF : KEYEVENTF_KEYUP
	Red	0xCB	
	ALL	0xCC	
Right LED2	Blue	0xCD	
	Red	0xCE	
	ALL	0xCF	
Right LED3	Blue	0xD0	
	Red	0xD1	
	ALL	0xD2	
Buzzer		0xD3	

2.2.1. Keybd_event(BYTE bVk, BYTE bScan, DWORD dwFlags, PTR dwExtraInfo)

1. Parameter

bVK

Virtual Key Code value

bScan

Hardware Scan Code value

dwFlags

Operation Specification Flag value

dwExtraInfo

Additional Information value

2.2.2. Example

```
void CserialDlg::OnBnClickedBtLed1()
{
   // TODO: Add your control notification handler code here
   keybd_event(0xC3,0,0,0);                // Left LED1 – ALL - ON
   Sleep(500);
   keybd_event(0xC3,0,KEYEVENTF_KEYUP,0);   // Left LED1 – ALL - OFF
   Sleep(500);
}

void CserialDlg:: OnBnClickedBtBuzzer ()
{
   // TODO: Add your control notification handler code here
   keybd_event(0xD3,0,0,0);                // Buzzer - ON
   Sleep(500);
   keybd_event(0xD3,0,KEYEVENTF_KEYUP,0);   // Buzzer - OFF
   Sleep(500);
}
```

3. Using Serial function

Is it technique for controlling the keypad, LED, Buzzer through the application's serial program.

If you can not use Keypad, LED and Buzzer through "Serial Daemon Application", use Keypad, LED and Buzzer easily with Serial protocol.

3.1. COM Port Open(), Close()

Open and close Serial COM port to enable DTP10-P,D's Key, LED and Buzzer operation.

```
m_comm=  new  CMycomm(_T("₩₩₩₩.₩₩")+m_str_comport,m_str_baudrate,_T("None"),_T("8  Bit"),_T("1 Bit"));
if( m_comm->Create(GetSafeHwnd()) != 0 ) {
    comport_state=true;
} else {
   AfxMessageBox(_T("COM PORT OPEN ERROR!"));
}
```

1. Parameter

Port

Serial port name to use as Serial Daemon.

Baudrate

The baudrate of the serial port.

Parity

Parity method of serial port.

Databit

Databit of serial port.

Stopbit

Stopbit of serial port.

2. Return value

If serial port open success, return handle, If serial port open fail, return 0.

3. Description

DTP10-P,D is additional functions are controlled through serial. Through this manual, you can easily understand Daincube's sample application.

4. Requirement

Function	Header	Reference code
CMycomm() Create()	Mycomm.h	Mycomm.cpp

5. Example

```
void CserialDlg::OnBnClickedBtConnect()
{
    if(comport_state) {     // Close COM port
    if(m_comm) {
      m_comm->Close();
      m_comm = NULL;
      comport_state=false;
  } else {     // Initial COM port
    m_comm= new CMycomm(_T("₩₩₩₩.₩₩")+m_str_comport,m_str_baudrate,_T("None"),_T("8 Bit"),_T("1 Bit"));
    if( m_comm->Create(GetSafeHwnd()) != 0 ) {
      comport_state=true;
    } else {
      AfxMessageBox(_T("COM PORT OPEN ERROR!"));
    }
  }
}
```

3.2. Send()

DTP10-P,D's LED, Buzzer for use to transmit the serial packet.

```
BOOL CMycomm::Send(char *outbuf, DWORD *len);
```

1.Parameter

outbuf

The buffer of the serial packet to be transmitted.

len

The buffer length of the serial packet to be transmitted.

2. Return value

If serial transmit success, return 1. If serial transmit fail, return 0.

3. Description

To turn on or off the LED or Buzzer, control by sending serial packet.

4. Requirement

Function	Header	Reference code
Send()	Mycomm.h	Mycomm.cpp

5. Example

```
void CserialDlg::OnBnClickedBtLed1()
{
   // TODO: Add your control notification handler code here
   char buf_printf[10] = {0, };
   unsigned int crc_buf;
   DWORD dwBytes = 0;

   buf_printf[0] = STX;            // STX
   buf_printf[1] = MOD_SET;          // MOD (get : 0x10, set : 0x11)
   buf_printf[2] = SEL_LED;         // SEL (LED : 0x3A)
   buf_printf[3] = LEFT_LED1;         // Data1
   buf_printf[4] = LED_BLUE; // Data2 (off : 0x30, blue : 0x31, red : 0x32, all : 0x33)
   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);
    m_comm->Send(buf_printf, &dwBytes);
}
```

3.3. Receive()

DTP10-P,D receives a serial packet for use of key

```
int CMycomm::Receive(LPSTR inbuf, int len);
```

1. Parameter

inbuf

The buffer of the serial packet to receive.

len

The buffer length of the serial packet to receive.

2. Return value

If serial packet receive success, return length of packet. If serial packet receive fail, return 0 or -1.

3. Description

Key receives and controls serial packet.

4. Requirement

Function	Header	Reference code
Receive()	Mycomm.h	Mycomm.cpp

5. Example

```
LRESULT CserialDlg::OnReceive(WPARAM length, LPARAM lpara)
{
    if(m_comm && comport_state) {

        while(length--)
        {
            m_comm->Receive(&g_Receive_Buffer[g_Head_Pointer],1);

            if(g_Head_Pointer >= BUFF_MAX-1)
                g_Head_Pointer = 0;
            else
```

```
            g_Head_Pointer++;
        }
    }
    return 0;
}
```

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-P,D 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 = 0x61, RIGHT_LED2 = 0x62, RIGHT_LED3 = 0x63

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

DATA3 : DATA_RESERVED = 0x20

```
void CserialDlg::OnBnClickedBtLed1()
{
    // TODO: Add your control notification handler code here
    char buf_printf[10] = {0, };
    unsigned int crc_buf;
    DWORD dwBytes = 0;
    static char i = 0;

    buf_printf[0] = STX;            // STX
    buf_printf[1] = MOD_SET;        // MOD (get : 0x10, set : 0x11)
    buf_printf[2] = SEL_LED;        // SEL (LED : 0x3A)
    buf_printf[3] = RIGHT_LED1;     // Data1
    if ( i == 3 )
        buf_printf[4] = LED_OFF;    // Data2 (off : 0x30, blue : 0x31, red : 0x32, all : 0x33)
```

```
    else
        buf_printf[4] = LED_BLUE + i;
    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);
    m_comm->Send(buf_printf, &dwBytes);
}
```

4.2. Example using Buzzer

Control the Buzzer of DTP10-P,D 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-P,D 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

```
void CserialDlg::OnBnClickedBtBuzzer()
{
   // TODO: Add your control notification handler code here
   char buf_printf[10] = {0, };
   unsigned int crc_buf;
   DWORD dwBytes = 0;
   static char i = 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 ( i == 0 )
      buf_printf[3] = BUZZ_ON;     // Data1 (off : 0x30, on : 0x31)
   else
      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);
   m_comm->Send(buf_printf, &dwBytes);
}
```

4.3. Examples using Keypad

Receive keypad event of DTP10-P,D 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-P,D 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

```
UINT CserialDlg::OperThread(LPVOID aParam)
{
   CserialDlg *dlg = (CserialDlg*)aParam;
   unsigned int crc_buf;
   DWORD keyevent_buf;

   while(dlg->g_Is_Thread_Run)
   {

      ..... ellipsis

      if ( ((dlg->g_Packet_Buffer[0] != STX) || (dlg->g_Packet_Buffer[8] != ETX)) ){      //STX, ETX Check
         continue;
      }

      if(dlg->g_Packet_Buffer[1] != MOD_GET){     //MOD Check
         continue;
      }

      if ( dlg->g_Packet_Buffer[2] != SEL_KEYPAD ){     // SEL (KEY : 0x3D)
         continue;
      }

      crc_buf = dlg->crc16_append(dlg->g_Packet_Buffer,6);

      if((dlg->g_Packet_Buffer[6]!=(char)((crc_buf>>8)&0xff)) || (dlg-
>g_Packet_Buffer[7]!=(char)(crc_buf&0xff))){ //CRC Check
         continue;
      }

      if ( dlg->g_Packet_Buffer[3] == KEYPAD_DOWN ) {         // Key DOWN
         keyevent_buf = 0;
```

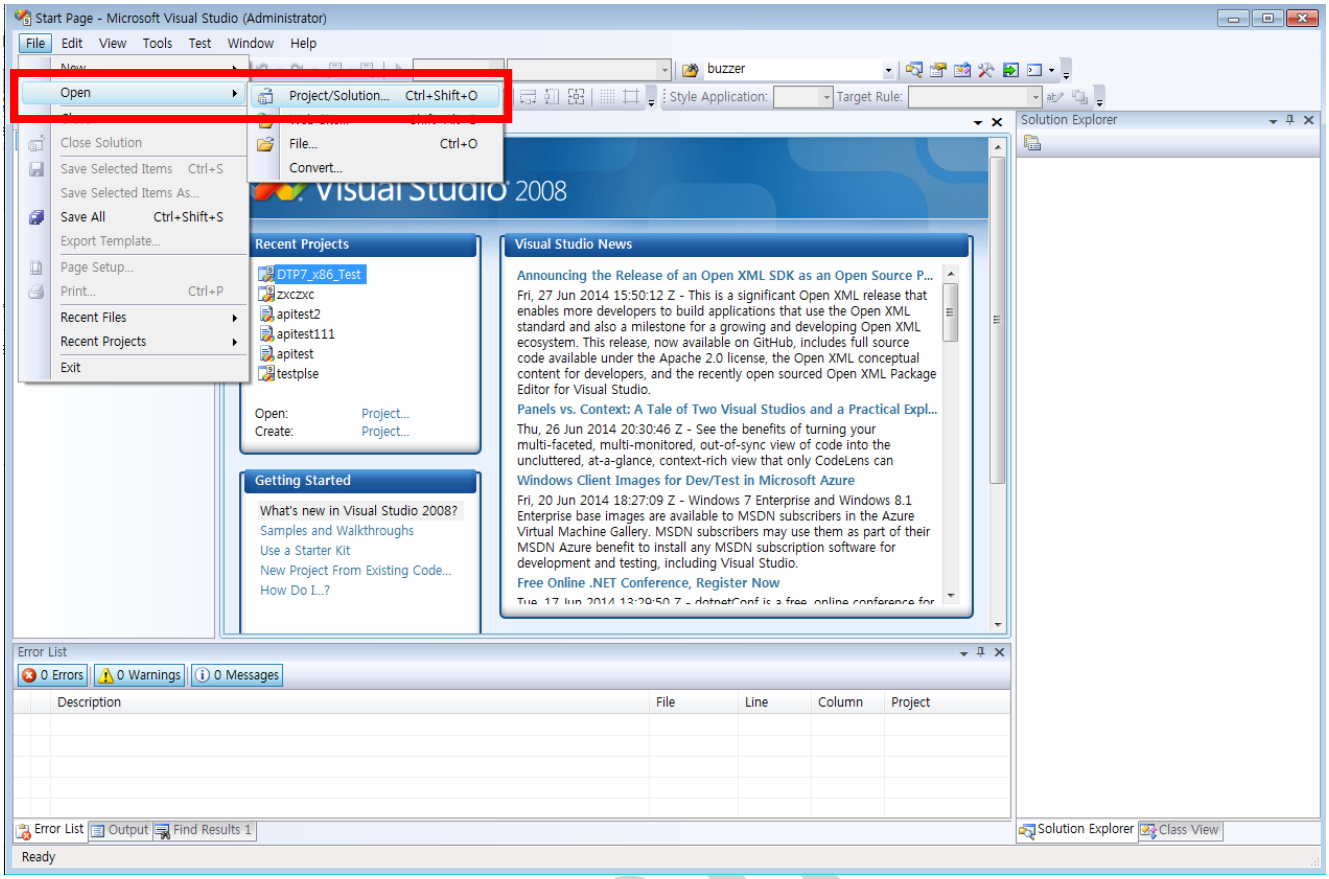
```
    }
    else if ( dlg->g_Packet_Buffer[3] == KEYPAD_UP ) {      // Key UP
      keyevent_buf = KEYEVENTF_KEYUP;
    }

    switch ( dlg->g_Packet_Buffer[4] ) {
      case KEY_A : keybd_event(0x41,0,keyevent_buf,0); break;
      case KEY_B : keybd_event(0x42,0,keyevent_buf,0); break;
      case KEY_C : keybd_event(0x43,0,keyevent_buf,0); break;
      case KEY_D : keybd_event(0x44,0,keyevent_buf,0); break;
      case KEY_E : keybd_event(0x45,0,keyevent_buf,0); break;
      case KEY_F : keybd_event(0x46,0,keyevent_buf,0); break;
      case KEY_G : keybd_event(0x47,0,keyevent_buf,0); break;
      case KEY_H : keybd_event(0x48,0,keyevent_buf,0); break;
      case KEY_I :  keybd_event(0x49,0,keyevent_buf,0); break;
      case KEY_J :  keybd_event(0x4A,0,keyevent_buf,0); break;
      case KEY_K : keybd_event(0x4B,0,keyevent_buf,0); break;
      case KEY_L : keybd_event(0x4C,0,keyevent_buf,0); break;
      case KEY_M :keybd_event(0x4D,0,keyevent_buf,0); break;
      case KEY_N : keybd_event(0x4E,0,keyevent_buf,0); break;
      case KEY_O : keybd_event(0x4F,0,keyevent_buf,0); break;
      case KEY_P : keybd_event(0x50,0,keyevent_buf,0); break;
      case KEY_F1 :   keybd_event(0x70,0,keyevent_buf,0); break;
      case KEY_F2 :   keybd_event(0x71,0,keyevent_buf,0); break;
      case KEY_F10 :  keybd_event(0x79,0,keyevent_buf,0); break;
      case KEY_F11 : keybd_event(0x7A,0,keyevent_buf,0); break;
    }
  }
  return 0;
}
```

5. How to build of Serial Daemon project

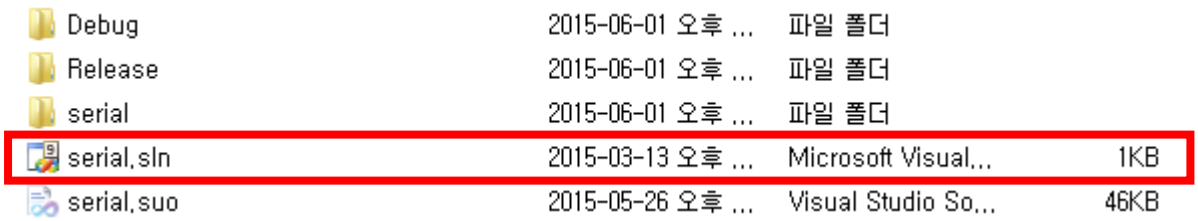
5.1. Project open

Open it using Visual Studio to modify Serial Daemon provided by Daincube.

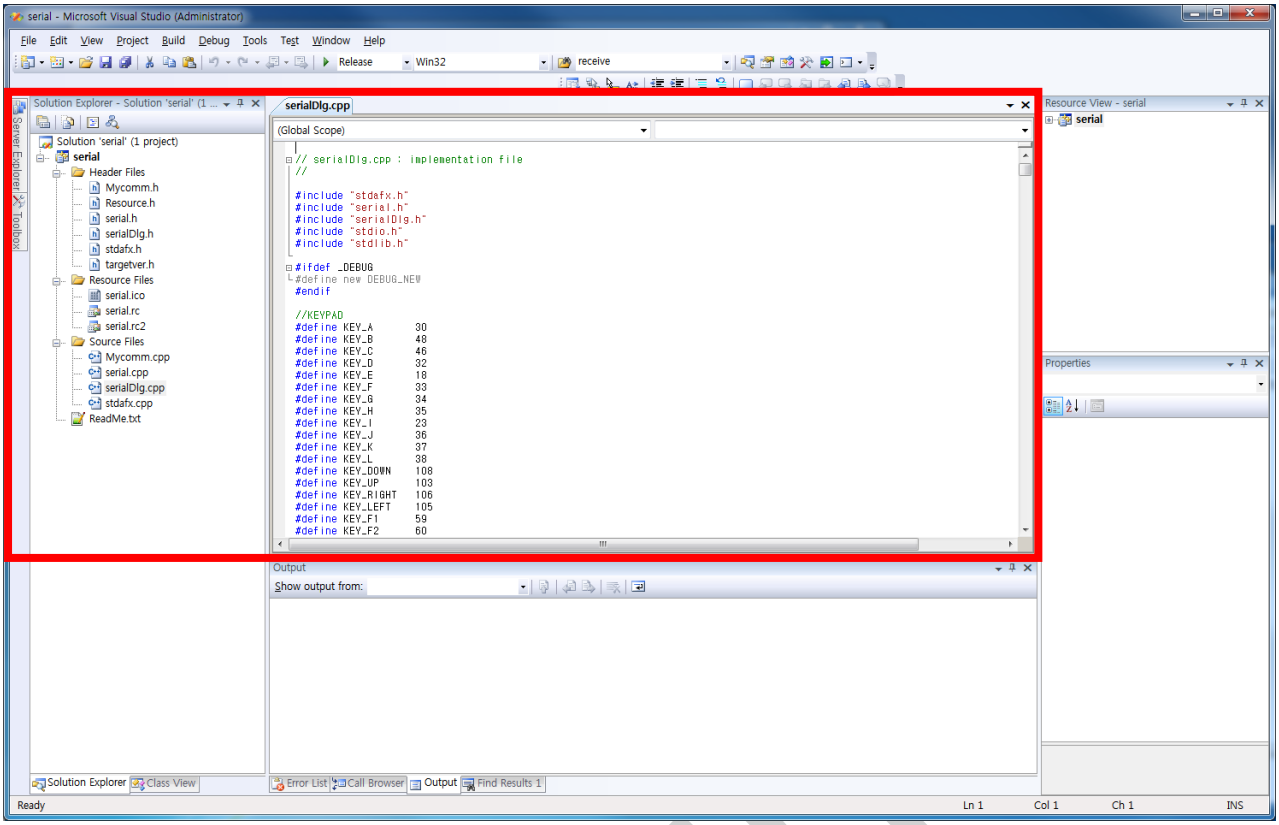
Select File -> Open -> Project/Solution.



Select "02_DTP10-D_SW or 02_DTP10-P_SW >> 03_Example >> 01_DTP10-PD_SerialDaemon_V1.0.0 >> serial.sln Solution files.



When the sample project open is success, the following screen is displayed.



5.2. Project build

Click to Build -> Build Solution. When finished, the following screen appears in the output window.