HID Touch Device

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	Demonstrate how to implement a USB HID touch device.
BSP Version	NUC123 Series BSP CMSIS V3.01.001
Hardware	NuTiny-EVB-NUC123-LQFP64 v1.0

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1 Function Description

1.1 Introduction

This example code implements a HID touch device with two contacts. It can simulate two contact input drawing two parallel lines.

1.2 Principle

The device is with a control pipe for control transfer. The control pipe (endpoint 0) is made of two hardware endpoints 0 and 1 in device. USB Host only see there is only a control endpoint 0 in device. The only interface is for HID touch data transmission. The Interface contains a HID interrupt IN endpoint for data transmission. The interface and endpoint configuration are shown in Table 1.

Endpoints in NUC123	Composite Device	Function
Endpoint 0 Control IN **Endpoint 1 Control OUT**	Control Pipe (Endpoint 0)	Control Transfer
Endpoint 2	Interface 0: Endpoint 1: Interrupt IN	HID Touch

Table 1:Interface and Endpoint Configuration of HID Touch Device

Touch data in interface 0 is transferred in the format shown in Table 2. This table is the format defined in HID report descriptor. It defines 2 sets of touch data structure so the device can return 2 contact data simultaneously. For further detail about the descriptor, user can refer to the Microsoft official site.

Byte	Bits	Description
0	0	Tip Switch
0	1	In Range
0	2	Touch Valid
0	3 ~ 7	Padding
1	0 ~ 7	Contact Identifier
2 ~ 3	0 ~ 7	X-axis
4 ~ 5	0 ~ 7	Y-axis
6	0	Tip Switch
6	1	In Range
6	2	Touch Valid
6	3 ~ 7	Padding
7	0 ~ 7	Contact Identifier
8 ~ 9	0 ~ 7	X-axis
10 ~ 11	0 ~ 7	Y-axis

Table 2: Report Format of HID Touch Device

1.3 Demo Result

When the example code is running and connected PC, HID-compliant touch screen device will be shown on device tree in device manager as shown in Figure 1.

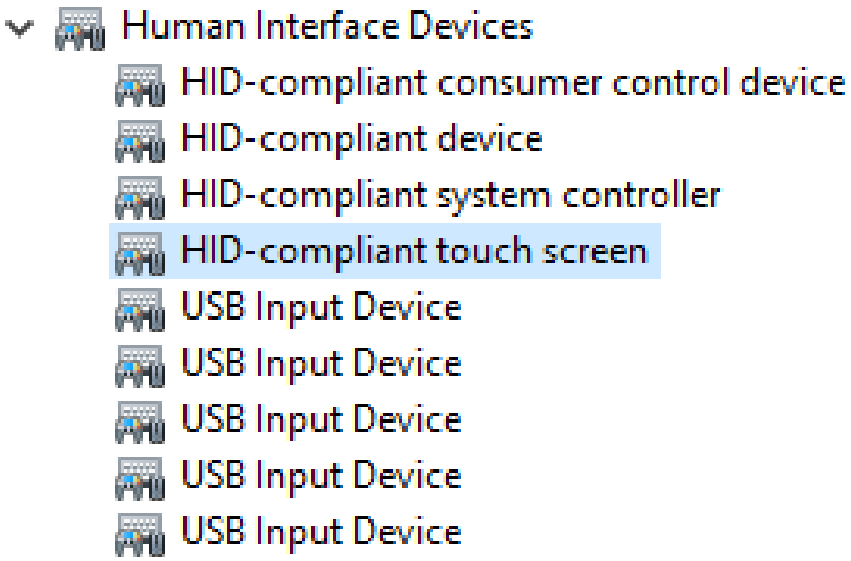


Figure 1: Device Tree in Device Manager

Figure 2 shows there is 14 bytes payload transmission in each transaction through USB packet analyzer.

Transaction	F	IN	ADDR	ENDP	T	Data	ACK	Time	Time Stamp
196	S	0x96	60	1	1	01 00 00 00 00 00 00 00 01 00 00 00 00 00	0x4B	1.000 ms	1 . 722 543 132
197	S	0x96	60	1	0	01 00 00 00 00 00 00 00 01 00 00 00 00 00	0x4B	999.982 us	1 . 723 543 150
198	S	0x96	60	1	1	01 00 00 00 00 00 00 00 01 00 00 00 00 00	0x4B	1.000 ms	1 . 724 543 132
199	S	0x96	60	1	0	01 00 00 00 00 00 00 00 01 00 00 00 00 00	0x4B	1.000 ms	1 . 725 543 132
200	S	0x96	60	1	1	01 00 00 00 00 00 00 00 01 00 00 00 00 00	0x4B	999.982 us	1 . 726 543 150
201	S	0x96	60	1	0	01 07 00 02 02 00 01 07 01 02 02 20 01 02	0x4B	1.000 ms	1 . 727 543 132
202	S	0x96	60	1	1	01 07 00 02 02 00 01 07 01 02 02 20 01 02	0x4B	1.000 ms	1 . 728 543 132
203	S	0x96	60	1	0	01 07 00 02 02 00 01 07 01 02 02 20 01 02	0x4B	1.000 ms	1 . 729 543 132
204	S	0x96	60	1	1	01 07 00 02 02 00 01 07 01 02 02 20 01 02	0x4B	1.000 ms	1 . 730 543 132
205	S	0x96	60	1	0	01 07 00 02 02 00 01 07 01 02 02 20 01 02	0x4B	1.000 ms	1 . 731 543 132

Figure 2: Transaction of HID Touch Device

Connect the device to desktop, then open software, paint, the device can draw parallel lines as the Figure 3.

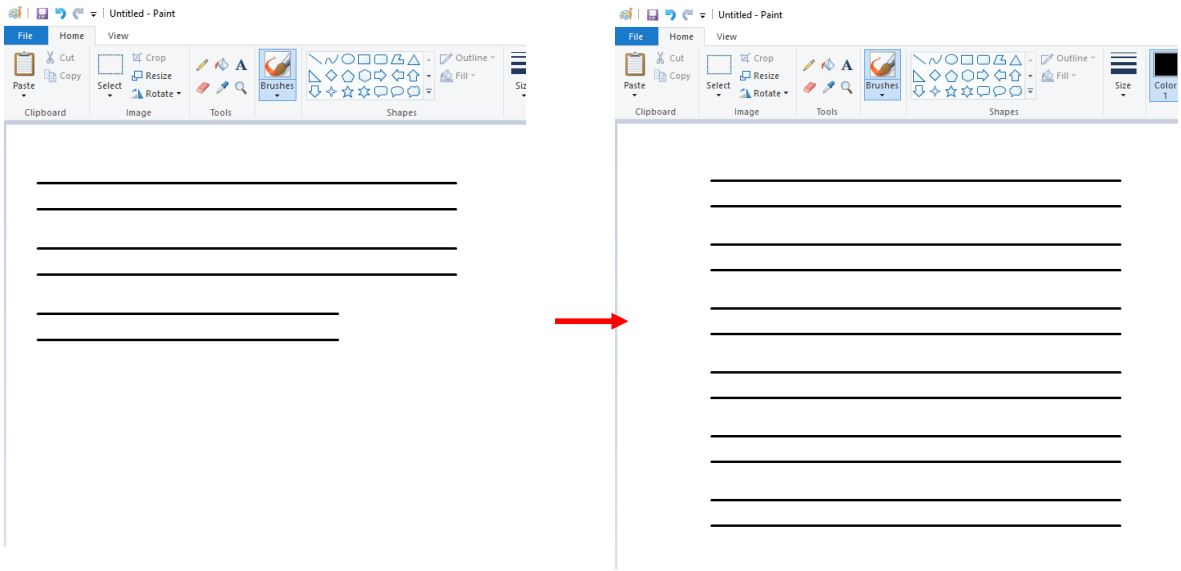


Figure 3: HID Touch Device Draws Parallel Lines in Software, Paint.

2 Code Description

The example implements a touch device by HID standard. The IN endpoint sends the data defined in HID report descriptor to the host as shown in Figure 2. The section 2.1 ~ 2.2 will introduce the processing procedure.

2.1 Main.c

Function, HID_UpdateTouchData(), in while loop is for data transmission. It implements interrupt IN transfer processing logic.

```
    while(1)
    {
       HID_UpdateTouchData();
    }
```

2.2 hid_touch.c

Desktop (USB Host) will issue IN token to request the data from the endpoint 2. While IN token received, USBD interrupt handler of endpoint 2 will set g_u8EP2Ready. Function, HID_UpdateTouchData(), will setup the payload and do data transmission if flag g_u8EP2Ready is set.
The device supports two contacts data transmission simultaneously. Each contact data length is 6 bytes plus a byte for Report ID for payload transmission so total length of payload is 14 bytes.

```
/* two lines demo in Paint */
uint32_t u32ReportCount = 0;
uint8_t u8IsX1Send04 = 0, u8IsX2Send04 = 0;
void HID_UpdateTouchData(void)
{
    uint8_t *pu8buf;
    static uint16_t u16X1 = 0x01f0, u16Y1 = 0x0100;

    if(g_u8EP2Ready)
    {

        pu8buf = (uint8_t *)(USBD_BUF_BASE + USBD_GET_EP_BUF_ADDR(EP2));

        /* Report ID 1 */
        pu8buf[0] = 1;
        pu8buf[2] = 0;
        pu8buf[8] = 1;
```

Below program implements two lines drawing. It sets the touch data in related fields. The code can simulate different trace effect by changing these fields.

```
         u32ReportCount++;

        if ((u16X1 >= 0x0200) && (u16X1 <= 0x0400)) /* touchDown */
        {
            u8IsX1Send04 = 1;

            pu8buf[1] = 0x07;
```

```
            pu8buf[3] = u16X1 & 0xff;
            pu8buf[4] = (u16X1 >> 8) & 0xff;
            pu8buf[5] = u16Y1 & 0xff;
            pu8buf[6] = (u16Y1 >> 8) & 0xff;
            pu8buf[7] = 0x07;
            pu8buf[9] = u16X1 & 0xff;
            pu8buf[10] = (u16X1 >> 8) & 0xff;
            pu8buf[11] = (u16Y1+0x20) & 0xff;
            pu8buf[12] = ((u16Y1+0x20) >> 8) & 0xff;
            pu8buf[13] = 2;
        }
        else if (u8IsX1Send04)     /* touchUp */
        {
            u8IsX1Send04 = 0;
            pu8buf[1] = 0x04;
            pu8buf[3] = 0x00;
            pu8buf[4] = 0x04;
            pu8buf[5] = u16Y1 & 0xff;
            pu8buf[6] = (u16Y1 >> 8) & 0xff;
            pu8buf[7] = 0x04;
            pu8buf[9] = 0x00;
            pu8buf[10] = 0x04;
            pu8buf[11] = (u16Y1+0x20) & 0xff;
            pu8buf[12] = ((u16Y1+0x20) >> 8) & 0xff;
            pu8buf[13] = 2;

            if (u16Y1 == 0x0100)
            {
                pu8buf[5] = 0xE0;
                pu8buf[6] = 0x02;
                pu8buf[11] = 0x00;
                pu8buf[12] = 0x03;
            }
            else
            {
                pu8buf[5] = (u16Y1-0x50) & 0xff;
                pu8buf[6] = ((u16Y1-0x50) >> 8) & 0xff;
                pu8buf[11] = ((u16Y1+0x20)-0x50) & 0xff;
                pu8buf[12] = (((u16Y1+0x20)-0x50) >> 8) & 0xff;
            }
        }
        else
        {
            pu8buf[1] = 0;
            pu8buf[3] = 0;
            pu8buf[4] = 0;
            pu8buf[5] = 0;
            pu8buf[6] = 0;
            pu8buf[7] = 0;
            pu8buf[9] = 0;
            pu8buf[10] = 0;
            pu8buf[11] = 0;
            pu8buf[12] = 0;
            pu8buf[13] = 0;
        }

        if ((u32ReportCount % 6) == 0)
            u16X1 += 0x3;

        if (u16X1 > 0x0400)
        {
            u16X1 = 0x01f0;
            u16Y1 += 0x50;

            if (u16Y1 > 0x0300)
                u16Y1 = 0x0100;
```

```
        }
```

Finally set the length of payload for endpoint 2 to send data.

```
        g_u8EP2Ready = 0;
        USBD_SET_PAYLOAD_LEN(EP2, 14);
    }
}
```

2.3 Descriptor.c

2.3.1 HID Report Descriptor

HID_DigitizerReportDescriptor is the HID report descriptor of the touch device. It defines the data format listed in Table 2.

The detail about these fields refer the document on Microsoft official site, and the document, HID Usage Table, on USB-IF site.

```
/*!<USB HID Report Descriptor */
const uint8_t HID_DigitizerReportDescriptor[] =
{
    0x05, 0x0D,         // USAGE_PAGE (Digitizers)
    0x09, 0x04,         // USAGE (Touch Screen)
    0xA1, 0x01,         // COLLECTION (Application)
    0x85, 0x01,         // REPORT_ID (1)
    0x09, 0x22,         // USAGE (Finger)

    0xA1, 0x02,         // COLLECTION (Logical)
    0x09, 0x42,         // USAGE (Tip Switch)
    0x15, 0x00,         // LOGICAL_MINIMUM (0)
    0x25, 0x01,         // LOGICAL_MAXIMUM (1)
    0x75, 0x01,         // REPORT_SIZE (1)
    0x95, 0x01,         // REPORT_COUNT (1)
    0x81, 0x02,         // INPUT (Data,Var,Abs)
    0x09, 0x32,         // USAGE (In Range)
    0x81, 0x02,         // INPUT (Data,Var,Abs)
    0x09, 0x47,         // USAGE (Touch Valid)
    0x81, 0x02,         // INPUT (Data,Var,Abs)
    0x95, 0x05,         // REPORT_COUNT (5)
    0x81, 0x03,         // INPUT (Cnst,Ary,Abs)
    0x75, 0x08,         // REPORT_SIZE (8)
    0x09, 0x51,         // USAGE (Contact Identifier)
    0x95, 0x01,         // REPORT_COUNT (1)
    0x81, 0x02,         // INPUT (Data,Var,Abs)
    0x05, 0x01,         // USAGE_PAGE (Generic Desktop)
    0x75, 0x10,         // REPORT_SIZE (16)
    0x55, 0x0E,         // UNIT_EXPONENT (-2)
    0x65, 0x11,         // UNIT (Centimeter)
    0x09, 0x30,         // USAGE (X)
    0x35, 0x00,         // PHYSICAL_MINIMUM (0)
    0x46, 0xE3, 0x13,   // PHYSICAL_MAXIMUM (0x13E3)
    0x26, 0x7F, 0x07,   // LOGICAL_MAXIMUM (0x77F)
    0x81, 0x02,         // INPUT (Data,Var,Abs)
    0x09, 0x31,         // USAGE (Y)
    0x46, 0x2F, 0x0B,   // PHYSICAL_MAXIMUM (0xB2F)
    0x26, 0x37, 0x04,   // LOGICAL_MAXIMUM (0x437)
    0x81, 0x02,         // INPUT (Data,Var,Abs)
```

```
    0xC0,               // END_COLLECTION

    0xA1, 0x02,         // COLLECTION (Logical)
    0x05, 0x0D,         // USAGE_PAGE (Digitizers)
    0x09, 0x42,         // USAGE (Tip Switch)
    0x15, 0x00,         // LOGICAL_MINIMUM (0)
    0x25, 0x01,         // LOGICAL_MAXIMUM (1)
    0x75, 0x01,         // REPORT_SIZE (1)
    0x95, 0x01,         // REPORT_COUNT (1)
    0x81, 0x02,         // INPUT (Data,Var,Abs)
    0x09, 0x32,         // USAGE (In Range)
    0x81, 0x02,         // INPUT (Data,Var,Abs)
    0x09, 0x47,         // USAGE (Touch Valid)
    0x81, 0x02,         // INPUT (Data,Var,Abs)
    0x95, 0x05,         // REPORT_COUNT (5)
    0x81, 0x03,         // INPUT (Cnst,Ary,Abs)
    0x75, 0x08,         // REPORT_SIZE (8)
    0x09, 0x51,         // USAGE (Contact Identifier)
    0x95, 0x01,         // REPORT_COUNT (1)
    0x81, 0x02,         // INPUT (Data,Var,Abs)
    0x05, 0x01,         // USAGE_PAGE (Generic Desktop)
    0x75, 0x10,         // REPORT_SIZE (16)
    0x55, 0x0E,         // UNIT_EXPONENT (-2)
    0x65, 0x11,         // UNIT (Centimeter)
    0x09, 0x30,         // USAGE (X)
    0x35, 0x00,         // PHYSICAL_MINIMUM (0)
    0x46, 0xE3, 0x13,   // PHYSICAL_MAXIMUM (0x13E3)
    0x26, 0x7F, 0x07,   // LOGICAL_MAXIMUM (0x77F)
    0x81, 0x02,         // INPUT (Data,Var,Abs)
    0x46, 0x2F, 0x0B,   // PHYSICAL_MAXIMUM (0xB2F)
    0x26, 0x37, 0x04,   // LOGICAL_MAXIMUM (0x437)
    0x09, 0x31,         // USAGE (Y)
    0x81, 0x02,         // INPUT (Data,Var,Abs)
    0xC0,               // END_COLLECTION

    0x05, 0x0D,         // USAGE_PAGE (Digitizers)
    0x09, 0x54,         // USAGE (Contact Count)
    0x15, 0x00,         // LOGICAL_MINIMUM (0)
    0x26, 0xFF, 0x00,       // LOGICAL_MAXIMUM (0x00FF)
    0x95, 0x01,         // REPORT_COUNT (1)
    0x75, 0x08,         // REPORT_SIZE (8)
    0x81, 0x02,         // INPUT (Data,Var,Abs)
    0x09, 0x55,         // USAGE (Contact Count Maximum)
    0x25, 0x02,         // LOGICAL_MAXIMUM (2)
    0x95, 0x01,         // REPORT_COUNT (1)
    0x85, 0x02,         // REPORT_ID (2)
    0xB1, 0x02,         // FEATURE (Data,Var,Abs)
    0xC0                // END_COLLECTION
};
```

2.3.2 Configuration Descriptor

USB host needs configuration descriptor for device enumeration. The array, gu8ConfigDescriptor, is the configuration descriptor with 1 interfaces. Field “bNumInterfaces” set as 1 to tell USB host there is 1 interface in this device. Interface descriptor, HID descriptor and endpoint descriptor are listed after configuration descriptor.

```
/*!<USB Configure Descriptor */
const uint8_t gu8ConfigDescriptor[] =
{
    LEN_CONFIG,     /* bLength */
```

```
    DESC_CONFIG,    /* bDescriptorType */
    /* wTotalLength */
    LEN_CONFIG_AND_SUBORDINATE & 0x00FF,
    ((LEN_CONFIG_AND_SUBORDINATE & 0xFF00) >> 8),
    0x01,           /* bNumInterfaces */
    0x01,           /* bConfigurationValue */
    0x00,           /* iConfiguration */
    0x80 | (USBD_SELF_POWERED << 6) | (USBD_REMOTE_WAKEUP << 5),/* bmAttributes */
    USBD_MAX_POWER, /* MaxPower */

    /* I/F descr: HID */
    LEN_INTERFACE,  /* bLength */
    DESC_INTERFACE, /* bDescriptorType */
    0x00,           /* bInterfaceNumber */
    0x00,           /* bAlternateSetting */
    0x01,           /* bNumEndpoints */
    0x03,           /* bInterfaceClass */
    0x00,           /* bInterfaceSubClass */
    HID_NONE,       /* bInterfaceProtocol */
    0x00,           /* iInterface */

    /* HID Descriptor */
    LEN_HID,        /* Size of this descriptor in UINT8s. */
    DESC_HID,       /* HID descriptor type. */
    0x10, 0x01,     /* HID Class Spec. release number. */
    0x00,           /* H/W target country. */
    0x01,           /* Number of HID class descriptors to follow. */
    DESC_HID_RPT,   /* Descriptor type. */
    /* Total length of report descriptor. */
    sizeof(HID_DigitizerReportDescriptor) & 0x00FF,
    ((sizeof(HID_DigitizerReportDescriptor) & 0xFF00) >> 8),

    /* EP Descriptor: interrupt in. */
    LEN_ENDPOINT,   /* bLength */
    DESC_ENDPOINT,  /* bDescriptorType */
    (INT_IN_EP_NUM | EP_INPUT), /* bEndpointAddress */
    EP_INT,         /* bmAttributes */
    /* wMaxPacketSize */
    EP2_MAX_PKT_SIZE & 0x00FF,
    ((EP2_MAX_PKT_SIZE & 0xFF00) >> 8),
    HID_DEFAULT_INT_IN_INTERVAL     /* bInterval */
};
```

3 Software and Hardware Environment

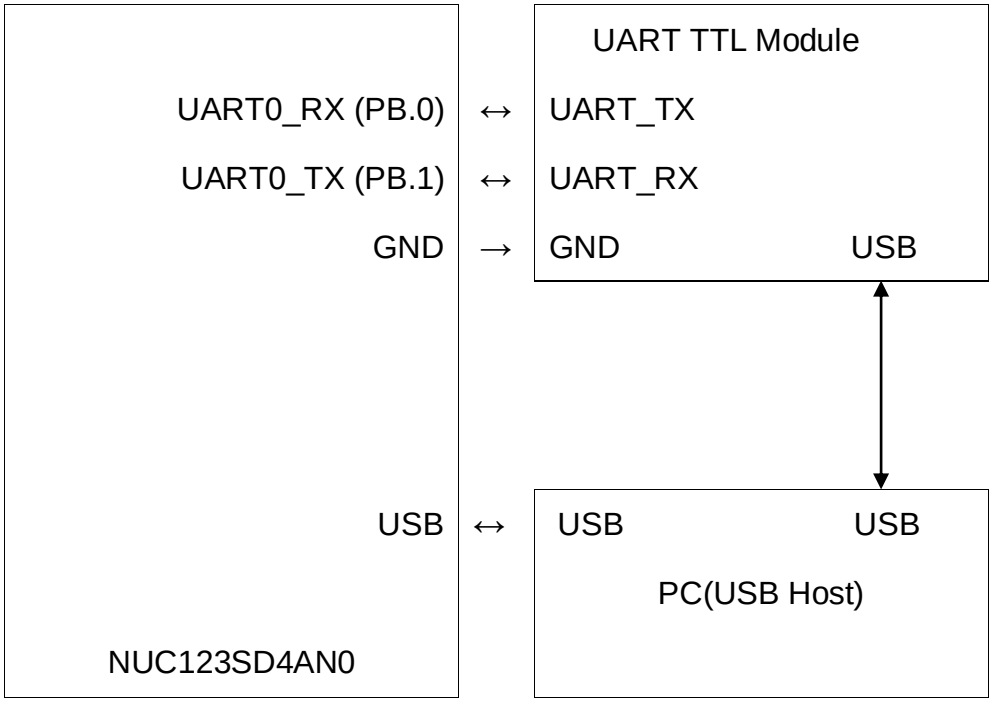
Software Environment

- BSP version
  - NUC123 Series BSP CMSIS V3.01.001
- IDE version
  - Keil uVersion 5.26
- Tool for USB packet sniffing and device tree display
  - USBLyzer (http://www.usblyzer.com/)

Hardware Environment

- Circuit components
  - NuTiny-EVB-NUC123-LQFP64 v1.0
  - USB mini USB cable
  - USB-UART TTL(Optional)
  - USB packet analyzer(Optional)

- Diagram


UART TTL Module
UART0_RX (PB.0)
↔
UART_TX
UART0_TX (PB.1)
↔
UART_RX
GND
→
GND
USB
USB
↔
USB
USB
PC(USB Host)
NUC123SD4AN0

4 Directory Information

- EC_NUC123_USBD_HID_Touch_V1.00
  - Library — Sample code header and source files
    - CMSIS — Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
    - Device — CMSIS compliant device header file
    - StdDriver — All peripheral driver header and source files
  - SampleCode
    - ExampleCode — Source file of example code

5 How to Execute Example Code

1. Browsing into sample code folder by Directory Information(section 4) and double click NUC123_USBD_HID_Touch.uvproj.
2. Enter Keil compile mode
   a. Build
   b. Download
   c. Start/Stop debug session
3. Enter debug mode
   a. Run

6 Revision History

Date	Revision	Description
Nov.01 , 2019	1.00	1. Initially issued.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.