M3334 UART 模擬 LIN 匯流排主機代碼

NuMicro® 32 位元系列微控制器範例代碼介紹

檔案資訊

應用簡述	本代碼用 M3334 的 UART1 實現 LIN 匯流排主機收發代碼
BSP 版本	M3334_Series_BSP_CMSIS_V3.00.002
開發平臺	NuMaker-M3334KI V2.0

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 and microprocessor 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. 概述

LIN 匯流排是基於 UART/SCI 的、單主多從結構的低成本介面，最高速率 20Kbps，常用速率 19200bps，9600bps。其報文幀結構以 13 位元 0 開始，後續跟著標準 UART 通信的多個位元組： 0x55+ID+資料+校驗和。資料個數雙方約定好，最多 8 位元組。13bit0+0x55+ID 是幀頭，只能主機發送。從機只能在收到與自己 ID 匹配的幀頭後，把資料發出。故主機必須週期性的發送幀頭，以便從機及時把資料發出。

本代碼用 UART 實現了 LIN 主機功能。代碼中定義了一個含六條報文的報文列表，列表內的報文迴圈發送，發送間隔是 20 ms，由 Timer 完成定時。每次幀頭髮完後，後續若是主機發送就繼續發送資料，後續若是主機接收，就啟動資料接收，匯流排時序如圖 1-1。

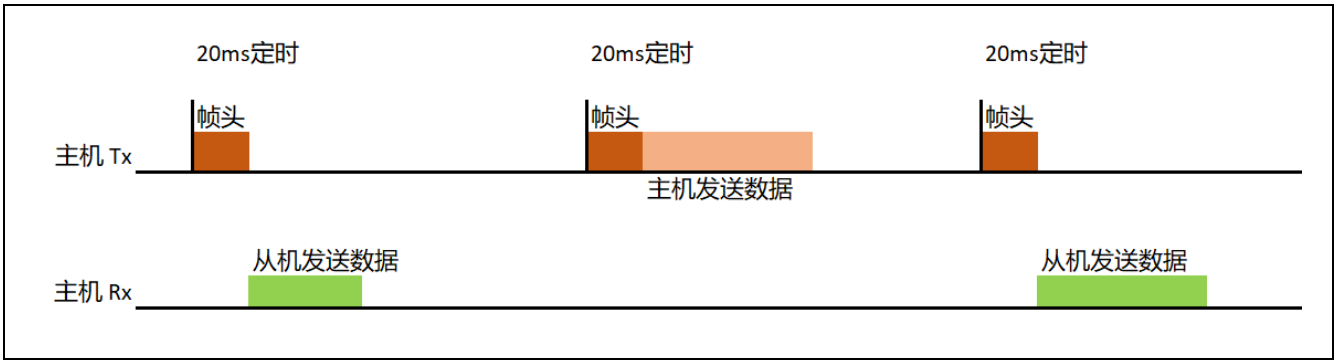


圖 1-1 LIN 匯流排時序

1.1 原理

幀頭結構如 圖 1-2 所示。

- 幀 ID 取值 0~0x3F 共 64 個值，放在 ID 段的 bit0~bit5。
- ID 段 bit6 = bit0，bit1，bit2，bit4 的異或值。Bit7=bit1，bit3，bit4，bit5 異或值再取反。
- 串列傳輸速率最高 20 kbps，本代碼中取 19200 bps。

在 Timer 的 20 ms 週期中斷裡，把串列傳輸速率降為 9/13 發送一個位元組 0，在匯流排上產生正常串列傳輸速率的 13 位時長的 0。13 位 0 發送結束後，進 UART 中斷發送 0x55 和 ID，到此幀頭發送結束。

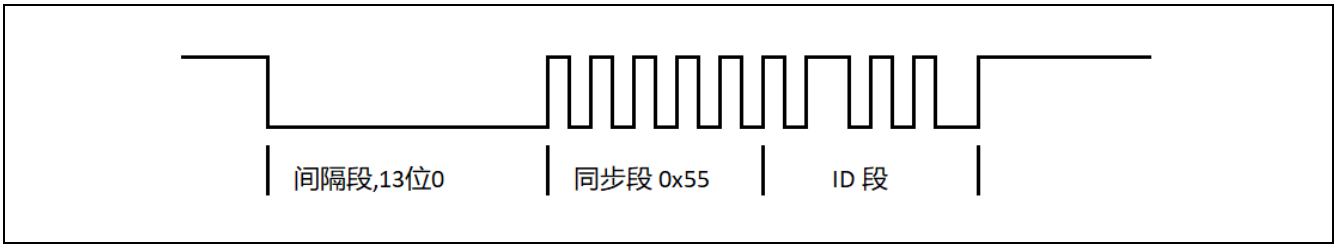


圖 1-2　LIN 匯流排幀頭

UART 中斷裡發送完 0x55 和 ID 後，再判斷當前幀若是主機發送，就繼續發送資料；當前幀若是主機接收，則配置接收長度、做接收前的準備。

1.2 執行結果

代碼中定義的六條報文 Message_Buffer[6]，四條發送，兩條接收。為方便查看本代碼執行效果，main 函數中週期性的啟動了 ID 等於 0x05 的報文的發送，列印資訊如圖 1-3 所示。



圖 1-3 列印資訊 1

若掛個 LIN 從機，列印介面還可會列印出接收報文的資訊，如圖 1-4 所示。

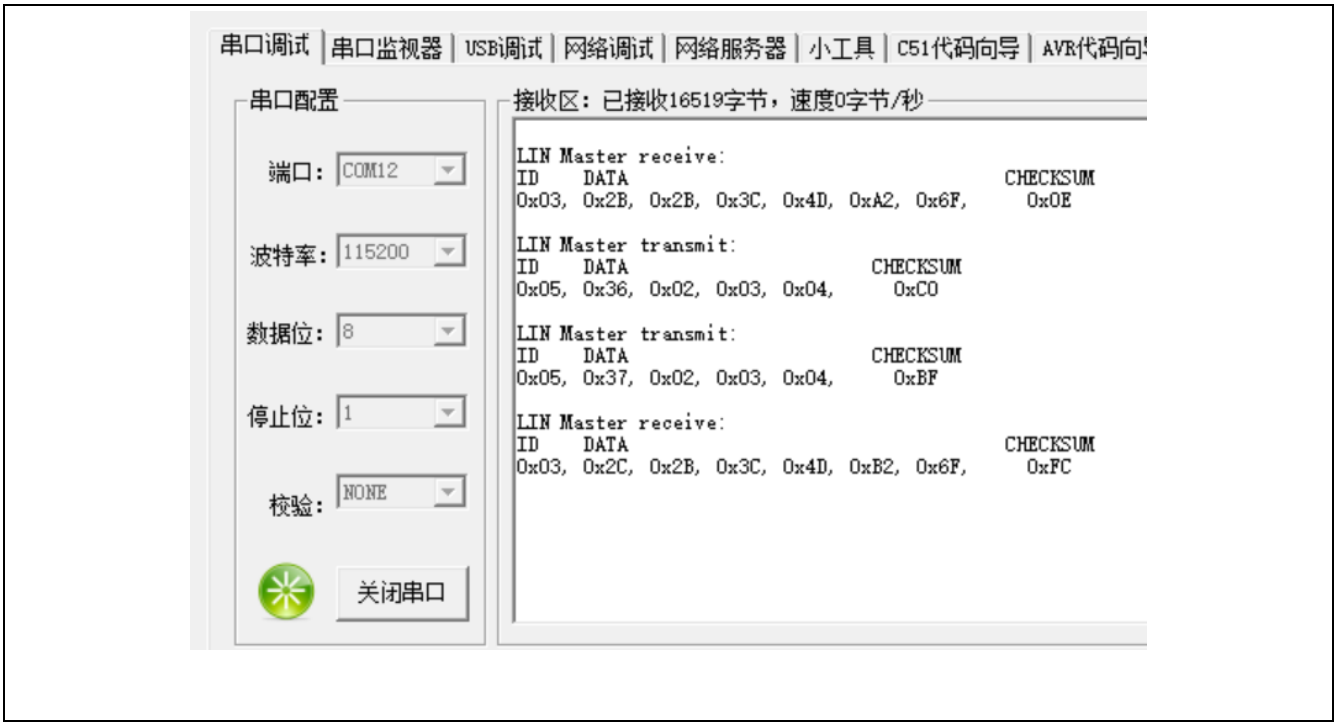

串口调试
串口监视器
USB调试
网络调试
网络服务器
小工具
C51代码向导
AVR代码向
串口配置
端口: COM12
波特率: 115200
数据位: 8
停止位: 1
校验: NONE
关闭串口
接收区：已接收16519字节，速度0字节/秒
LIN Master receive:
ID DATA CHECKSUM
0x03, 0x2B, 0x2B, 0x3C, 0x4D, 0xA2, 0x6F, 0x0E
LIN Master transmit:
ID DATA CHECKSUM
0x05, 0x36, 0x02, 0x03, 0x04, 0xC0
LIN Master transmit:
ID DATA CHECKSUM
0x05, 0x37, 0x02, 0x03, 0x04, 0xBF
LIN Master receive:
ID DATA CHECKSUM
0x03, 0x2C, 0x2B, 0x3C, 0x4D, 0xB2, 0x6F, 0xFC

圖 1-4 列印資訊 2

2. 代碼介紹

Timer1 20 ms 中斷代碼

進中斷先把串列傳輸速率降為 9/13 發一個位元組 0，如此匯流排上產生正常串列傳輸速率的 13 位 0。發送 ID 前重新計算一下 ID 的 bit6 bit7，如此，代碼中修改報文 ID 不必手算 ID 校驗碼。13 位 0 發送結束後產生能 UART 發送結束中斷，由 UART 中斷收發後續資料。

```
void TMR1_IRQHandler(void)
{
    int32_t temp32, temp32_1;

    Timer1_Period_Dec --;

    /*
     * Generate LIN Break by transmitting UART 0x00 at a reduced
     * baud rate. UART 0x00 has 9 LOW bits (start + 8 data bits).
     * Scaling the baud rate makes these approximately 13 LIN bits.
     *
     * At 19.2 kbps:
     *     13 LIN bits = 677.08 us
     *
     * Verify the actual waveform because the UART baud-rate divider
     * may introduce a small timing error.
     */
    UART_Open(UART1, LIN_BREAK_BAUD_RATE);
    UART1->DAT = 0x00;
    Frame_Head = TRUE;

    if (++ptr_Msg > Message_End) ptr_Msg = Message_Buffer;

    /* Clear TIMER1 time-out interrupt flag */
    TIMER_ClearIntFlag(TIMER1);
    /* bit6 = bit1 ^ bit4 */
    temp32   = (ptr_Msg->PID << 5) ^ (ptr_Msg->PID << 2);
    /* bit2 = bit0 ^ bit2 */
    temp32_1 = (ptr_Msg->PID << 6) ^ (ptr_Msg->PID << 4);
    /* bit6 = bit0 ^ bit1 ^ bit2 ^ bit4 */
    temp32_1 = (temp32 ^ temp32_1) & 0x40;

    temp32 <<= 1;
    temp32  ^= ptr_Msg->PID << 4;
    temp32  ^= ptr_Msg->PID << 2;
    temp32  ^= 0x80;

    /* bit7 = ~(bit1 ^ bit3 ^ bit4 ^ bit5) */
    temp32 &= 0x80;
    /* Calculate bit6,7 of PID */
    ptr_Msg->PID = (ptr_Msg->PID & 0x3F) | temp32 | temp32_1;

    if (ptr_Msg->DataNum > 8) ptr_Msg->DataNum = 8;
```

```
    /* Last frame stop receiving */
    UART_DisableInt(UART1, UART_INTEN_RXTOIEN_Msk);

    /* 13bit zero finishing will occur interrupt */
    UART_EnableInt(UART1, UART_INTEN_TXENDIEN_Msk);
}
```

幀資料結構

成員 Tx_Frame_Cnt 只用于發送資料幀，報文資料每發一次減 1，減到 0 對應 ID 的資料幀不再發送資料，但仍會發送幀頭。故此值=0 時可更新待發送的資料，不會引起發送資料混疊。

Rx_Frame_Num 只用於接收資料幀，每收到一幀對應 ID 的完整報文資料，就加 1。此值=0 時收到的資料不更新到幀清單結構體中，因此把此值寫 0 再讀清單內的資料可避免接收資料混疊。

```
typedef struct
{
    /* 1: This frame will transmit data.  0: This frame is for reception */
    uint8_t  Flag_Master_Tx;
    /* Message ID */
    uint8_t  PID;
    /* Data for transmiting */
    uint8_t  TxData[9];
    /* data number, Max =8 */
    uint8_t  DataNum;
    /* received (data + Checksum) */
    uint8_t  RxData[12];
    /* when Tx_Frame_Cnt != 0, data TxData[] will be transmited */
    uint8_t  Tx_Frame_Cnt;
    /* If Rx_Frame_Num=0, RxData[] won't be updated. */
    uint8_t  Rx_Frame_Num;
    /* If this equals 0 checksum is classic */
    uint8_t  CheckSum_Type;
} Message_Type;
```

UART 中斷函數

進中斷後若 Frame_Head 非 0 表明前面 Timer 剛發完 13 位 0，恢復 19200 串列傳輸速率繼續發幀頭資料 0x55 和 幀 ID。

接下來按幀結構資料中的 Flag_Master_Tx 判斷當前幀是主機發送、還是主機接收。因 M3334 UART 有 16 級緩存，而 LIN 匯流排最多 8 位元組資料加校驗碼。故若是主機發送，則可把資料一次性全部寫入發送緩存。若是主機接收，則要等 0x55+ID 發送結束後再啟動接收，目的是單線收發時，不接收自己發的 0x55+ID，因此無論單線還是雙線應用，統一都只接收報文的資料欄。

```
void UART1_IRQHandler(void)
```

```
{
    int32_t static TxCnt, RxCnt, CntMax;
    uint8_t static RxStr_Buf[11];
    uint8_t temp8u;

    /* If frame header has just been transmitted out. */
    if (Frame_Head == TRUE)
    {
        /* Transmit frame head */
        UART_Open(UART1, LIN_BAUD_RATE);
        UART1->DAT = 0x55;
        UART1->DAT = ptr_Msg->PID;

        /* Disable UART Tx interrupt */
        UART_DisableInt(UART1, UART_INTEN_TXENDIEN_Msk);
        Frame_Head = FALSE;

        CntMax = ptr_Msg->DataNum;

        /* Master transmits data subsequently */
        if (ptr_Msg->Flag_Master_Tx)
        {
            /* When CPU updates the data, please set Tx_Frame_Cnt = 0 */
            if (ptr_Msg->Tx_Frame_Cnt)
            {
                /* Decrease every transmision */
                ptr_Msg->Tx_Frame_Cnt --;
                TxCnt = 0;

                while (TxCnt <= CntMax)
                {
                    /* Transmits data and CheckSum */
                    UART1->DAT = ptr_Msg->TxData[TxCnt++];
                }
            }
        }
        /* Master receive data subsequently. */
        else
        {
            /* Wait for 0x55+PID is transmitted completely */
            while ((UART1->FIFOSTS & UART_FIFOSTS_TXEMPTYF_Msk) == 0);

            RxCnt = 0;
            /* Clear Rx FIFO for acceptance */
            UART1->FIFO |= UART_FIFO_RXRST_Msk;

            /* Waitting for Rx FIFO clear completion */
            while ((UART1->FIFOSTS & UART_FIFOSTS_RXIDLE_Msk) == 0) __NOP();

            /* Enable TxD Timer over interrupt */
            UART_EnableInt(UART1, UART_INTEN_RXTOIEN_Msk);
        }

    }
    /* Rx data process */
    else
```

```
    {
        /* RxD Timer over interrupt */
        while (UART1->INTSTS & UART_INTSTS_RXTOINT_Msk)
        {
            while ((UART1->FIFOSTS & UART_FIFOSTS_RXEMPTY_Msk) == 0)
            {
                /* Read out received data */
                temp8u = UART1->DAT;

                /* Copy data and CheckSum */
                if (RxCnt <= CntMax) RxStr_Buf[RxCnt++] = temp8u;
                /* After receiving this frame Data field, disable UART1 TO interrupt */
                else UART_DisableInt(UART1, UART_INTEN_RXTOIEN_Msk);
            }

            /* Last byte, checksum has been received */
            if (RxCnt == CntMax + 1)
            {
                /*
                 * Accept the response only when the LIN checksum valid.
                 */
                if (LIN_Checksum_Verify(ptr_Msg->PID,
                                        RxStr_Buf,
                                        CntMax,
                                        ptr_Msg->CheckSum_Type))
                {
                    /* If Rx_Frame_Num=0, data updates here are prohibited */
                    if (ptr_Msg->Rx_Frame_Num)
                    {
                        for (RxCnt = 0; RxCnt <= CntMax; ++RxCnt)
                        {
                            /* Copy data and CheckSum */
                            ptr_Msg->RxData[RxCnt] = RxStr_Buf[RxCnt];
                        }

                        if (++ptr_Msg->Rx_Frame_Num == 0) ptr_Msg->Rx_Frame_Num = 1;
                    }
                }
                else
                {
                    /* Invalid LIN checksum: discard the response. */
                    LIN_Rx_Checksum_Error++;
                }
            }
        }
    }
}
```

當收到資料個數等於幀資料長度再多一位元組時，表明已收到校驗和位元組，可把收到的資料從 RxStr_Buf [ ] 轉存到幀清單對應 ID 陣列中。成員 Rx_Frame_Num=0 時不轉存資料，故從幀清單中讀數據時，先寫 Rx_Frame_Num=0，可防止讀報文資料時發生接收資料混疊。

3. 軟體與硬體需求

3.1 軟體需求

- BSP 版本
  - M3334_Series_BSP_CMSIS_V3.00.002
- IDE 版本
  - Keil uVision 5.43.1.0
  - VS Code 1.115.3

3.2 硬體需求

- 電路元件
  - NuMaker-M3334KI V2.0。
  - 若無其它測試從機,可用 EC_M2003_UART_LIN_Slave_V1.00 做測試從機。
- 線路示意圖

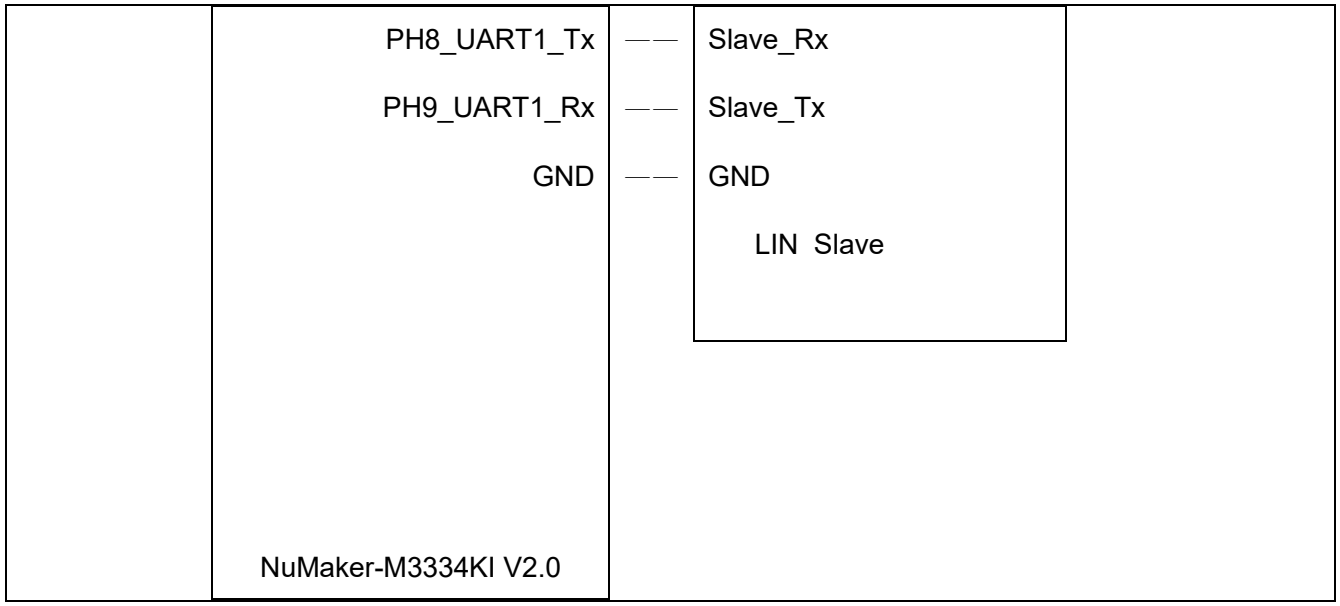


圖 3-1 線路示意圖

4. 目錄資訊

目錄結構如下圖所示。

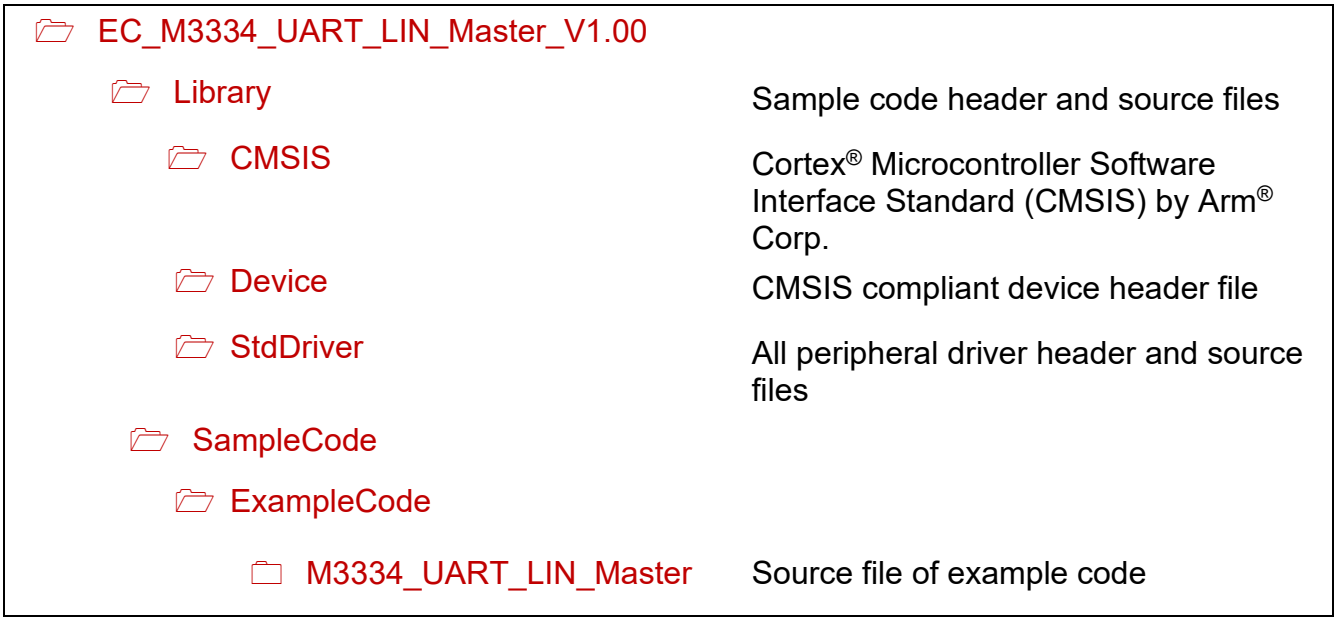


圖 4-1 目錄資訊

5. 範例程式執行

5.1 Keil 平臺

1. 在 Windows 資源管理器中進入 Keil 工程資料夾，再開啟 *M3334_UART_LIN_Master.uvprojx* 工程檔。
2. 編譯代碼
3. 下載代碼
4. 進入模擬模式
   - 執行代碼

5.2 VS Code 平臺

1. 在 VS Code IDE 中開啟 File 菜單
   - 按一下**Open Folder**
   - 選取工程目錄下的**VSCode** 資料夾
2. 選擇 CMSIS
   - 按一下**Build solution**編譯工程
   - 按一下**Load & Run application**下載映射檔到目標晶片
   - 按一下**Load & Debug application**進入模擬模式

6. 修訂記錄

Date	Revision	Description
2026.09.19	1.00	初始發佈。

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.