M252 PSIO 模拟 MDIO 波形

NuMicro® 32位系列微控制器范例代码介绍

文件信息

应用简述	本范例代码使用 M252 PSIO 模拟 MDIO 波形用以控制以太网络 PHY (IP101GR)。
BSP 版本	M251_M252_M254_M256_M258_Series_BSP_CMSIS_V3.02.006
开发平台	NuMaker-M252SD V1.2

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

本范例代码使用 M252 PSIO 接口来实作控制 MDIO 时序并实际用以控制以太网络 PHY。

PSIO 提供了一种简单的方法来接收和传送各种串行传输，透过设置时隙 (Slot)、检查点 (Check point) 与引脚相对应的电压来控制引脚在特定的时间输出高电平或低电平，以及电平状态要维持多长时间。

MDIO 接口包含两条线，MDC 与 MDIO，其中 MDIO 是双向资料线而 MDC 为时钟线。

大多数 NuMicro® 家族微控制器不支持 MDIO 接口，本范例代码提供一种存取以太网络 PHY 的解决方案，可广泛应用于 NuMicro® 家族微控制器。

本范例代码需另接外部硬件模块。包含一九旸电子的以太网络 PHY(IP101GR)。

1.1 原理

1.1.1 MDIO 数据格式

MDIO 数据格式如图 1-1 所示，字段说明如表 1-1 所示。

Preamble	ST	OP	PHY_ADR	REG_ADR	TA	DATA
32-bit	2-bit	2-bit	5-bit	5-bit	2-bit	16-bit

图 1-1 MDIO 数据格式

字段	数据宽度	说明
Preamble	32-bit	32 个连续 2’1
ST	2-bit	帧开始标志 2’01
OP	2-bit	操作码 01 = 写操作 10 = 读操作
PHY_ADR	5-bit	PHY 地址

REG_ADR	5-bit	寄存器地址
TA	2-bit	状态转换时间
DATA	16-bit	MDIO 资料

表 1-1 MDIO 数据字段

1.1.2 MDIO 时序

使用 PSIO 控制 MDIO 装置，用户必须先设置时隙 (Slot)、检查点 (Check point) 与引脚相对应的电压来控制引脚在特定的时间输出高电平或低电平，以及电平状态要维持多长时间。

PSIO0 CH3 (PC.2) 引脚用于模拟 MDIO 波形输出输入引脚; PSIO0 CH1(PC.4) 引脚用于模拟 MDC 波形输出引脚。要注意的是，在读操作中，程序必须先将 PSIO0 CH3 (PC.2) 引脚配置为输入模式，才能读取数据。

图 1-2 与图 1-3 分别标示 PSIO 在读/写操作的相关配置。本范例代码设定 MDC 为 500 kHz，其 MDC 时序计算如下所示：

- PSIO 时钟源设定为 HIRC，并除频为 8 MHz; 一个时隙的时间为 0.125 µs (= 1 / 8 MHz)。
- 设定时隙长度为 8，则 MDC 半个脉波宽度时间为 0.125 µs x 8 = 1 µs

基于以上设定，PSIO0 CH1(PC.4) 引脚即可产生 500 kHz 的方波波形，PSIO0 CH3 (PC.2) 引脚仅须在检查点 0 时设定为输出或输入缓存。

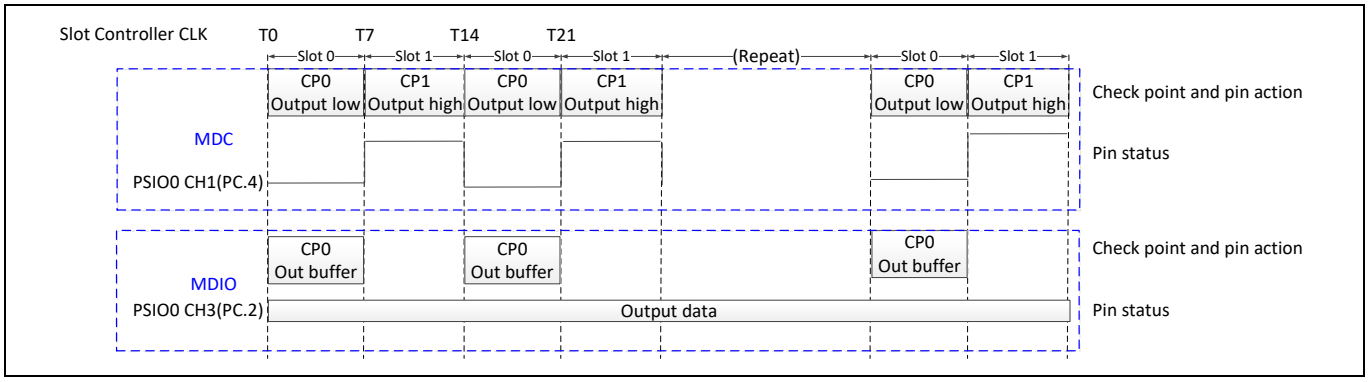

图 1-2 PSIO 写操作相关设定

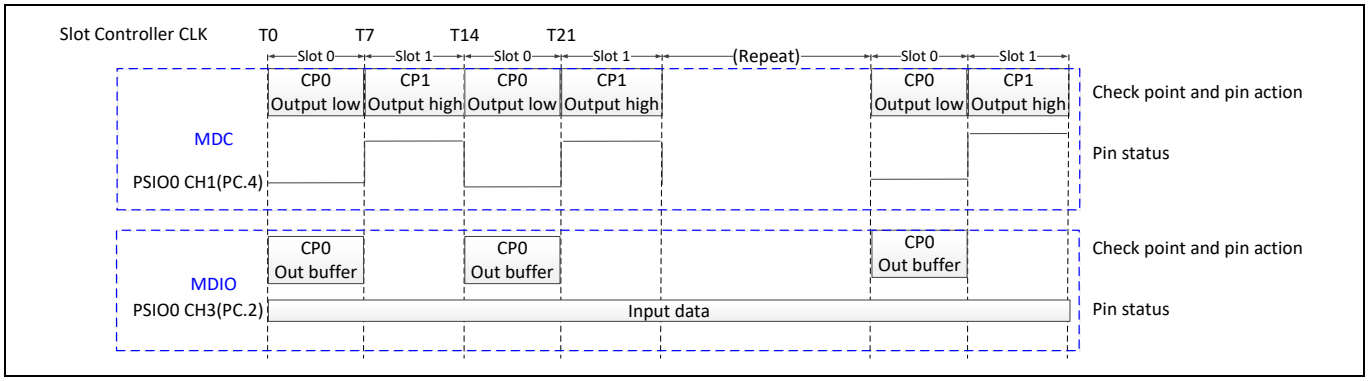

图 1-3 PSIO 读操作相关设定

1.2 执行结果

开启计算机串口调试助手软件，设置 UART 传输速度为 115200bps，无校验位，数据长度为 8 位，停止位长度为 1 位。执行程序后，串口调试助手显示操作指示。

图 1-4 串口调试助手屏幕截图

本范例代码执行编译结果如图 1-5 与图 1-6 所示，分别对应 MDIO 写操作和读操作波形。演示将寄存器 0x04 配置为 0x0001 并进行验证。

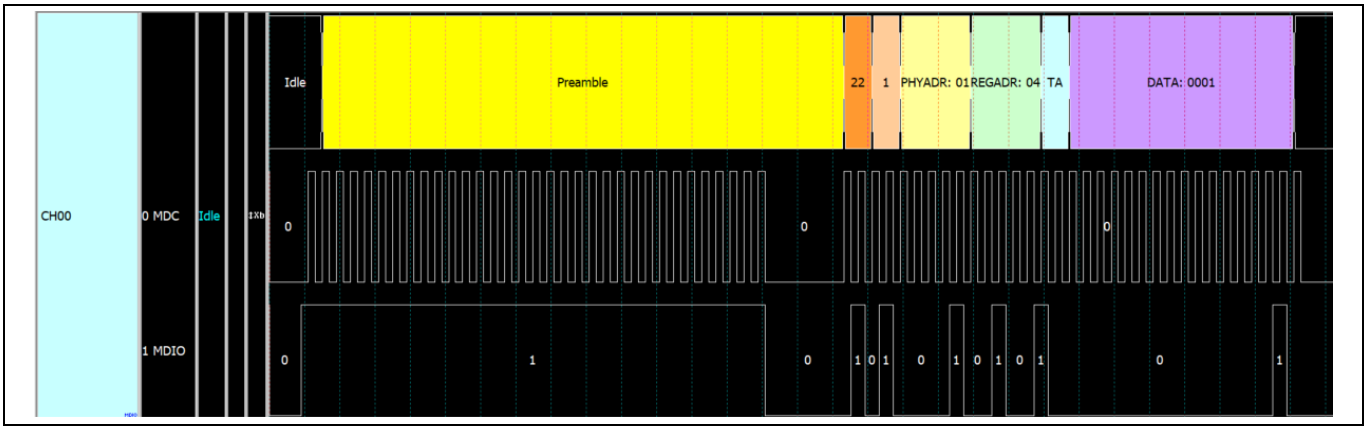

图 1-5 MDIO 写操作波形

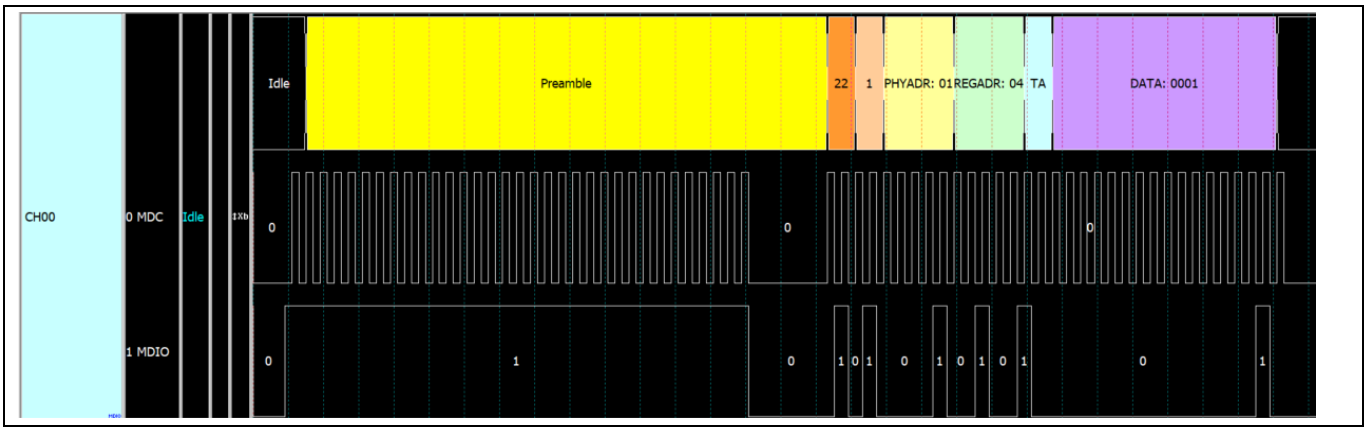

图 1-6 MDIO 读操作波形

2. 代码介绍

写操作与读操作流程代码位于 *main.c*。

写操作

- 传送 32 位 Preamble。
- 传送 ST (2 位)、opcode (2 位)、PHY 地址 (5 位)、寄存器地址 (5 位)、TA (2 位) 与 寄存器数据 (16 位)。

```
int main ( void)
{
    …

    /* Prepare the MDIO package with 32-bit data, including 2-bit ST, 2-bit Op code, 5-bit
       PHY address, 5-bit Register address,  2-bit TA and 16-bit Data */
    u32TxData = (MDIO_ST_CAUSE_22 << MDIO_ST_OFFSET) | (MDIO_OP_W << MDIO_OP_OFFSET) |
                (MDIO_PHY_ADDR << MDIO_PHY_OFFSET) | (u32InputRegAdr << MDIO_REG_OFFSET) |
                (MDIO_TA << MDIO_TA_OFFSET) | u32InputData;

    /* Write Preamble field to Ethernet PHY */
    PSIO_MDIO_Write(&sConfig, &u32Preamble, 32);

    /* Write ST/Op code/PHY adress/TA fields to Ethernet PHY */
    PSIO_MDIO_Write(&sConfig, &u32TxData, 32);

    …
}
```

读操作

- 传送 32 位 Preamble。
- 传送 ST (2 位)、opcode(2 位)、PHY 地址(5 位)、寄存器地址(5 位)、TA (2 位)。
- 接收 16 位数据。

```
int main(void)
{
    …

    /* Prepare the MDIO package with 32-bit data, including 2-bit ST, 2-bit Op code, 5-bit
       PHY address, 5-bit Register address and 2-bit TA */
    u32TxData = (MDIO_ST_CAUSE_22 << MDIO_ST_OFFSET) | (MDIO_OP_R << MDIO_OP_OFFSET) |
    (MDIO_PHY_ADDR << MDIO_PHY_OFFSET) | (u32InputRegAdr << MDIO_REG_OFFSET) | (MDIO_TA <<
    MDIO_TA_OFFSET);
    u32TxData = u32TxData >> 16;
```

```
    /* Write Preamble field to Ethernet PHY */
    PSIO_MDIO_Write(&sConfig, &u32Preamble, 32);

    /* Write ST/Op code/PHY adress/TA fields to Ethernet PHY */
    PSIO_MDIO_Write(&sConfig, &u32TxData, 16);

    /* Read data from Ethernet PHY */
    PSIO_MDIO_Read(&sConfig, &u32InputData);

    …
}
```

PSIO 初始化设定、写操作与读操作代码位于 *MDIO_driver*.c。

使用 PSIO_MDIO_Init( ) 来设定每一个时隙 (Slot) 的宽度与检查点 (Check point) 相对应的引脚状态。

```
void PSIO_MDIO_Init(S_PSIO_MDIO *psConfig)
{
    const S_PSIO_CP_CONFIG sMDCConfig =
    {

             /* Check Point0 */ /* Check Point1 */  /* Check Point2 */  /* Check Point3 */
/* Slot */   PSIO_SLOT0,        PSIO_SLOT1,         PSIO_SLOT_DISABLE,  PSIO_SLOT_DISABLE,
/* Action */ PSIO_OUT_LOW,      PSIO_OUT_HIGH,      PSIO_NO_ACTION,     PSIO_NO_ACTION,

             /* Check Point4 */ /* Check Point5 */  /* Check Point6 */  /* Check Point7 */
/* Slot */   PSIO_SLOT_DISABLE, PSIO_SLOT_DISABLE,  PSIO_SLOT_DISABLE,  PSIO_SLOT_DISABLE,
/* Action */ PSIO_NO_ACTION,    PSIO_NO_ACTION,     PSIO_NO_ACTION,     PSIO_NO_ACTION
    };

    /* Enable MDC pin and setting general configuration */
    PSIO_SET_GENCTL(PSIO, psConfig->u8MDC, PSIO_PIN_ENABLE, psConfig->u8SlotCtrl
                    , PSIO_OUTPUT_MODE, PSIO_LOW_LEVEL, PSIO_LOW_LEVEL);

    /* Set data order and MSB */
    PSIO_SET_ORDER(PSIO, psConfig->u8MDC, PSIO_MSB);
    PSIO_SET_ORDER(PSIO, psConfig->u8MDIO, PSIO_MSB);

    /* Set slot counter number */
    PSIO_SCSLOT_SET_SLOT(PSIO, psConfig->u8SlotCtrl, PSIO_SLOT0, 8);
    PSIO_SCSLOT_SET_SLOT(PSIO, psConfig->u8SlotCtrl, PSIO_SLOT1, 8);

    /* Set check point configuration */
    PSIO_SET_CP_CONFIG(PSIO, psConfig->u8MDC, &sMDCConfig);
}
```

使用 PSIO_MDIO_Write( ) 产生输出波形，其时序可参考图 1-2。

```
void PSIO_MDIO_Write(S_PSIO_MDIO *psConfig, uint32_t *pu32Data, uint32_t bit_length)
{
    const S_PSIO_CP_CONFIG sDataOutConfig =
    {
            /* Check Point0 */ /* Check Point1 */  /* Check Point2 */  /* Check Point3 */
```

```
/* Slot */  PSIO_SLOT0,         PSIO_SLOT_DISABLE,  PSIO_SLOT_DISABLE,  PSIO_SLOT_DISABLE,
/* Action */PSIO_OUT_BUFFER,    PSIO_NO_ACTION,     PSIO_NO_ACTION,     PSIO_NO_ACTION,

            /* Check Point4 */ /* Check Point5 */  /* Check Point6 */  /* Check Point7 */
/* Slot */   PSIO_SLOT_DISABLE, PSIO_SLOT_DISABLE,  PSIO_SLOT_DISABLE,  PSIO_SLOT_DISABLE,
/* Action */ PSIO_NO_ACTION,    PSIO_NO_ACTION,     PSIO_NO_ACTION,     PSIO_NO_ACTION
    };

    /* Enable MDIO pin and setting general configuration */
    PSIO_SET_GENCTL(PSIO, psConfig->u8MDIO, PSIO_PIN_ENABLE, psConfig->u8SlotCtrl
                    , PSIO_OUTPUT_MODE, PSIO_LOW_LEVEL, PSIO_LOW_LEVEL);

    PSIO_SET_CP_CONFIG(PSIO, psConfig->u8MDIO, &sDataOutConfig);

    /* Loop slot0~slot1 times */
    PSIO_SET_SCCTL(PSIO, psConfig->u8SlotCtrl, PSIO_SLOT0, PSIO_SLOT1, bit_length - 1,
PSIO_REPEAT_DISABLE);

    /* Set output data width */
    PSIO_SET_WIDTH(PSIO, psConfig->u8MDIO, 0, bit_length);

    /* Set output data */
    PSIO_SET_OUTPUT_DATA(PSIO, psConfig->u8MDIO, *pu32Data);

    /* Trigger slot controller */
    PSIO_START_SC(PSIO, psConfig->u8SlotCtrl);

    /* Wait for slot controller is not busy */
    while (PSIO_GET_BUSY_FLAG(PSIO, psConfig->u8SlotCtrl));
}
```

使用 PSIO_MDIO_Read( ) 产生读取波形，其时序可参考图 1-3。

```
void PSIO_MDIO_Read(S_PSIO_MDIO *psConfig, uint32_t *pu32Data)
{
    uint32_t u32Data;

    const S_PSIO_CP_CONFIG sDataInConfig =
    {
            /* Check Point0 */ /* Check Point1 */  /* Check Point2 */  /* Check Point3 */
/* Slot */   PSIO_SLOT0,         PSIO_SLOT_DISABLE,  PSIO_SLOT_DISABLE,  PSIO_SLOT_DISABLE,
/* Action */ PSIO_IN_BUFFER,    PSIO_NO_ACTION,     PSIO_NO_ACTION,     PSIO_NO_ACTION,

            /* Check Point4 */ /* Check Point5 */  /* Check Point6 */  /* Check Point7 */
/* Slot */   PSIO_SLOT_DISABLE, PSIO_SLOT_DISABLE,  PSIO_SLOT_DISABLE,  PSIO_SLOT_DISABLE,
/* Action */ PSIO_NO_ACTION,    PSIO_NO_ACTION,     PSIO_NO_ACTION,     PSIO_NO_ACTION
    };

    /* Enable MDIO pin and setting general configuration */
    PSIO_SET_GENCTL(PSIO, psConfig->u8MDIO, PSIO_PIN_ENABLE, psConfig->u8SlotCtrl
                    , PSIO_INPUT_MODE, PSIO_LOW_LEVEL, PSIO_LOW_LEVEL);

    PSIO_SET_CP_CONFIG(PSIO, psConfig->u8MDIO, &sDataInConfig);

    /* Loop slot 0~slot 1 15 times */
```

```
    PSIO_SET_SCCTL(PSIO, psConfig->u8SlotCtrl, PSIO_SLOT0, PSIO_SLOT1, 15,
PSIO_REPEAT_DISABLE);

    /* Set input data width as 16 */
    PSIO_SET_WIDTH(PSIO, psConfig->u8MDIO, 16, 0);

    /* Set input data depth as 1 */
    PSIO_SET_INPUT_DEPTH(PSIO, psConfig->u8MDIO, PSIO_DEPTH1);

    /* Trigger slot controller */
    PSIO_START_SC(PSIO, psConfig->u8SlotCtrl);

    /* Wait for data buffer is full */
    while (!PSIO_GET_TRANSFER_STATUS(PSIO, PSIO_TRANSTS_INFULL0_Msk << (psConfig->u8MDIO *
4)));

    /* Read data from device */
    u32Data = PSIO_GET_INPUT_DATA(PSIO, psConfig->u8MDIO);
    *pu32Data = u32Data;
}
```

3. 软件与硬件需求

3.1 软件需求

- BSP 版本
  - M251_M252_M254_M256_M258_Series_BSP_CMSIS_V3.02.006
- IDE 版本
  - Keil uVersion 5.28

3.2 硬件需求

- 电路组件
  - NuMaker-M252SD V1.2
- 线路示意图
  - PSIO0_CH3 (PC.2) 引脚连接到以太网络 PHY MDIO 引脚。
  - PSIO0_CH1 (PC.4) 引脚连接到以太网络 PHY MDC 引脚。

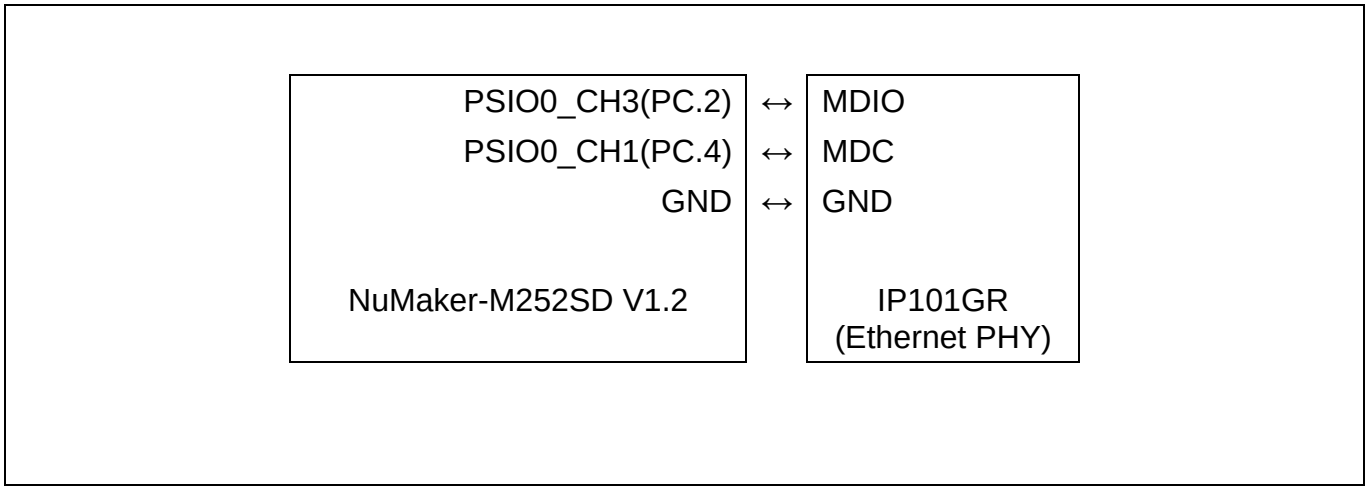

图 3-1 引脚连接图

4. 目录信息

EC_M252_PSIO_Emulate_MDIO_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	
Project	Source file of example code

图 4-1 目录信息

5. 范例程序执行

1. 根据目录信息章节进入 ExampleCode 路径中的 KEIL 文件夹，双击 *M252_PSIO_Emulate_MDIO.uvprojx*。
2. 进入编译模式界面
   - 编译
   - 下载代码至内存
   - 进入 / 离开仿真模式
3. 进入仿真模式界面
   - 执行代码

6. 修订纪录

Date	Revision	Description
2024.08.22	1.00	初始发布。

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.