Using UART to Simulate ISO7816-3 Interface

Example Code Introduction for 32-bit NuMicro® Family

Document Information

Application	This example code uses M251 UART to simulate a Smart Card interface based on ISO7816-3 standard.
BSP Version	M251_M252_M254_M256_M258_Series_BSP_CMSIS_V3.02.007
Hardware	NuMaker-M251SD 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 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. Overview

This example code will demonstrate how to use M251 UART and CLKO to simulate a Smart Card interface based on ISO7816-3 standard, and implement operations such as activation, cold reset, warm reset, transactions, and release sequence of Smart Card. The transactions use T0 asynchronous half-duplex character transmission protocol.

1.1 Principle

This example code mainly uses four peripherals: UART0, UART1, CLKO and TIMER0.

- UART0 is used as a debugging serial port and is mainly used to print debugging information, Smart Card response data, and input interaction information.
- UART1 is used to simulate ISO7816-3 DAT. ISO7816-3 uses asynchronous half-duplex communication. Transmitting and receiving use a common DAT line, so the TX and RX of UART1 need to be connected.
- CLKO is used to output the ISO7816-3 clock CLK.
- TIMER0 is used to implement various delays in the Smart Card operation sequence.

The character frame format in ISO7816-3 transmission is shown in Figure 1-1.

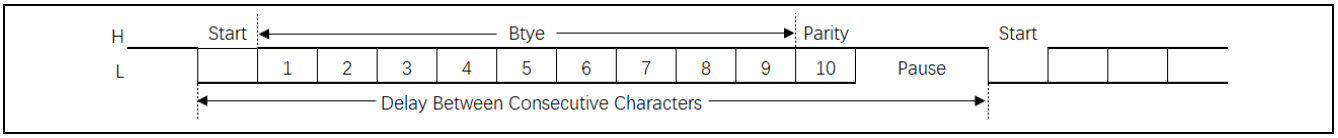


Figure 1-1 Character Frame

The elementary time unit refers to the time required to transmit one bit of the character frame. The calculation formula is as follows:

$$1\ etu = \frac{F}{D} \times \frac{1}{f}$$

The parameter f is the CLK output frequency, which is output by CLKO in the example and is set to 3 MHz.

The parameter F is the clock conversion factor, and parameter D is the baud rate adjustment factor. The elementary time unit used in the communication depends on the actual values of F and D. During Answer to Request (ATR), F and D use default values, where F = 372 and D = 1.

It can be seen from the above that the baud rate of UART1 is 1/(1 etu).

Timer0's CLK is also set to 3 MHz.

When receiving, set PB.6 pin multiplexing function to UART1_RX and PB.7 to GPIO input. Keeping both pins in input mode will not affect the Smart Card output data. It is also necessary to enable the receive data available interrupt and RX timeout interrupt. After the reception is completed, interrupts need to be disabled.

When sending, keep PB.6 as UART1_RX function, because UART1_RX is input and will not affect the data on DAT. Set PB.7 pin multiplexing function to UART1_TX, enable the transmit holding register empty interrupt, and complete the data transmission in the interrupt function. After the transmission is completed, it is necessary to disable the transmit holding register empty interrupt and set PB.7 to GPIO input mode again to prevent PB.7 and Smart Card from outputting at the same time.

The steps for this example to operate Smart Card are as follows:

1) Activate Smart Card
2) Cold reset
3) Warm reset if cold reset fails
4) Receive Answer to Request (ATR)
5) PPS exchange
6) Transaction operations: select file, get response
7) Release sequence

1.2 Demo Result

The compilation results of this example code are shown in Figure 1-2. Open the serial port tool on the PC and connect to UART0 to print debugging information.

```
[2025-01-16 14:01:49.646]# RECV ASCII>
+-------------------------------------------------------------------------+
|    M251 UART Emulate ISO7816 Sample Code                                |
+-------------------------------------------------------------------------+

>> The I/O connection: <<
   M251             Smart Card Slot
   GPIO(PA.1)             VCC
   GPIO(PA.2)             RST
   CLKO(PA.3)             CLK
   UART1_RXD(PB.6)        DAT
   UART1_TXD(PB.7)        DAT
   GND                    GND

Press any key to start test.

[2025-01-16 14:01:58.014]# SEND ASCII>
1

[2025-01-16 14:01:58.255]# RECV ASCII>
ATR: 3b 79 18 00 00 50 53 4d 5f 54 43 31 30 30

Select file
61 17

Get response
6f 15 84 0e 31 50 41 59 2e 53 59 53 2e 44 44 61 08

Exit Sample code.
```

Figure 1-2 Compilation Results

2. Code Description

This example code is mainly divided into three parts: hardware driver, Smart Card protocol library, and main function.

Hardware driver

For peripheral and pin definitions, the code is located in *sc_drv_config.h*. If using other pins, UART, or Timer, only the definition here needs to be modified, without modifying the driver code.

```
/* Define function of multi-function pin */
#define SET_CLKO_PA3()       SYS->GPA_MFPL = (SYS->GPA_MFPL & ~SYS_GPA_MFPL_PA3MFP_Msk) |
                             SYS_GPA_MFPL_PA3MFP_CLKO
#define SET_UART1_RX_PB6()   SYS->GPB_MFPL = (SYS->GPB_MFPL & ~SYS_GPB_MFPL_PB6MFP_Msk) |
                             SYS_GPB_MFPL_PB6MFP_UART1_RXD
#define SET_UART1_TX_PB7()   SYS->GPB_MFPL = (SYS->GPB_MFPL & ~SYS_GPB_MFPL_PB7MFP_Msk) |
                             SYS_GPB_MFPL_PB7MFP_UART1_TXD
#define SET_GPIO_PB7()       SYS->GPB_MFPL = (SYS->GPB_MFPL & ~SYS_GPB_MFPL_PB7MFP_Msk)

/* VCC */
#define VCC       PA1  /* pin */
#define VCC_PORT  PA   /* port */
#define VCC_BIT   BIT1 /* bit */

/* RST */
#define RST       PA2  /* pin */
#define RST_PORT  PA   /* port */
#define RST_BIT   BIT2 /* bit */

/* CLK */
#define SC_CLK        PA3  /* pin */
#define CLK_PORT      PA   /* port */
#define CLK_BIT       BIT3 /* bit */
#define SET_CLKO_PIN  SET_CLKO_PA3 /* multi-function = CLKO */

/* UART_TX */
#define UART_TX             PB7   /* pin */
#define UART_TX_PORT        PB    /* port */
#define UART_TX_BIT         BIT7  /* bit */
#define SET_UART_TX_PIN     SET_UART1_TX_PB7 /* multi-function = UART1_TX */
#define CANCEL_UART_TX_PIN  SET_GPIO_PB7     /* multi-function = GPIO */

/* UART_RX */
#define UART_RX          PB6  /* pin */
#define UART_RX_PORT     PB   /* port */
#define UART_RX_BIT      BIT6 /* bit */
#define SET_UART_RX_PIN  SET_UART1_RX_PB6 /* multi-function = UART1_RX */

/* UART */
#define UART      UART1      /* Uart select */
#define UART_IRQ  UART1_IRQn /* IRQ */
#define UART_IRQHandler UART1_IRQHandler /* Interrupt handling function */
#define UART_CLK_SELECT()   CLK_SetModuleClock(UART1_MODULE, CLK_CLKSEL1_UART1SEL_HIRC,
 CLK_CLKDIV0_UART1(1)) /* Clock source */
```

```
#define UART_CLK_ENABLE()   CLK_EnableModuleClock(UART1_MODULE) /* Enable clock */

/* Timer */
#define TIMER   TIMER0
#define TIMER_CLK_SELECT()   CLK_SetModuleClock(TMR0_MODULE, CLK_CLKSEL1_TMR0SEL_HIRC, 0)
#define TIMER_CLK_ENABLE()   CLK_EnableModuleClock(TMR0_MODULE) /* Enable clock */
```

For hardware driver function, the code is located in *sc_drv.c.*

- Smart Card interface initialization

```
void SC_Init(void)
{
    /* 1. Set pin multi-function */
    SC_Pin_Init();

    /* 2. UART init */
    UART_Open(UART, 8065);/* By default, F=372 D = 1, Baud Rate = f/372 = 3M/372 = 8065 */

    /* Set Rx Time-out counter */
    UART_SetTimeoutCnt(UART, 0x10);

    /* Set Rx FIFO Interrupt Trigger Level */
    UART->FIFO &= ~ UART_FIFO_RFITL_Msk;

    UART->FIFO |= UART_FIFO_RFITL_4BYTES;

    NVIC_EnableIRQ(UART_IRQn);

    /* 3. Set the clock output of the card reader using clko. f = 3M Hz */
    CLK_EnableModuleClock(CLKO_MODULE); /* Enable clk */

    /* Select CKO clock source. clock source = HCLK, No frequency division */
    CLK_SetModuleClock(CLKO_MODULE, CLK_CLKSEL1_CLKOSEL_HCLK, NULL);

    /* CKO = clock source / 2^(u32ClkDiv + 1) */
    CLK->CLKOCTL = (CLK->CLKOCTL & ~(CLK_CLKOCTL_FREQSEL_Msk | CLK_CLKOCTL_DIV1EN_Msk))
                   | (3 << CLK_CLKOCTL_FREQSEL_Pos) /* div = 2^(3+1) */
                   | (0 << CLK_CLKOCTL_DIV1EN_Pos); /* div enable   f = 48/2^3 = 3M */

    /* 4. Set timer for delay. */
    /* Set timer frequency to SC_CLK (3M Hz) */
    TIMER->CTL = TIMER_CONTINUOUS_MODE | 15;/* 3M = HCLK/(15+1) */
}
```

- DAT output enable function

```
void SC_DAT_OutPut_Enable(void)
{
    /* Set multi-function pin for UART1 TXD */
    SET_UART_TX_PIN();

    /* Enable transmit holding register empty interrupt */
    UART_EnableInt(UART, UART_INTEN_THREIEN_Msk);
}
```

- DAT output disable function

```
void SC_DAT_OutPut_Disable(void)
{
    /* Disable TX interrupt */
    UART_DisableInt(UART, UART_INTEN_THREIEN_Msk);

    /* Wait for transmission to end */
    while (!UART_IS_TX_EMPTY(UART));

    /* Disable TX pin. Set multi-function pin for GPIO */
    CANCEL_UART_TX_PIN();
}
```

- DAT input enable function

```
void SC_DAT_Input_Enable(void)
{
    /* Enable Receive Data Available Interrupt and RX Time-out Interrupt. */
    UART_EnableInt(UART, (UART_INTEN_RDAIEN_Msk | UART_INTEN_RXTOIEN_Msk));
}
```

- DAT input disable function

```
void SC_DAT_Input_Disable(void)
{
    UART_DisableInt(UART, (UART_INTEN_RDAIEN_Msk | UART_INTEN_RXTOIEN_Msk));
}
```

- Delay function

```
void SC_Timer_Delay(uint32_t ticks)
{
    uint32_t t1;

    t1 = TIMER->CNT;

    while(TIMER->CNT - t1 < ticks);
}
```

Smart Card Protocol Library

The Smart Card protocol library is located in SmartcardT0Lib folder, which contains API interface functions for application calls, implementation code for protocols, and functions for internal protocol calls.

The API interface functions of the Smart Card protocol library can be found in the *SmartcardT0Lib\Source\sclib-api.c* file.

- Card Activation

```
int32_t SCLIB_Activate(uint32_t num, uint32_t u32EMVCheck)
{
    SCLIB_DEV_T *dev = &_scDev[num];
    int32_t retval, clk;

    /* 1. Read clock of card slot. */
    clk = _SCLIB_GetInterfaceClock(num);

    /* Upper and lower clock limits */
```

```
    if(clk < 1000 || clk > 5000)
        return SCLIB_ERR_CLOCK;

    /* 2. Set card status=card present */
    dev->bCardRemoved = SCLIB_CARD_PRESENT;

    /* 3. Set whether to check EMV */
    dev->EMV = u32EMVCheck;

    /* 4. Cold Reset */
    retval = SCLIB_ColdReset(num);

    /* 5. Warm Reset */
    if(retval == SCLIB_ERR_ATR_INVALID_PARAM) /* Cold Reset failed. */
        retval = SCLIB_WarmReset(num); /* Warm Reset */

    /* 6. Set Tx and Rx buffers */
    dev->snd_buf = &g_TxBUF[0]; /* Tx buffer */

    dev->rcv_buf = &g_RxBUF[0]; /* Rx buffer */

    return(retval);
}
```

- Cold Reset

```
int32_t SCLIB_ColdReset(uint32_t num)
{
    SCLIB_DEV_T *dev = &_scDev[num]; /* Device pointer */

    /* 1. Cold Reset */
    dev->errno = _SCLIB_ResetCard(num, SCLIB_COLDRESET);/* Smartcard interface number,
    Cold Reset*/

    /* Cold Reset failed. */
    if(dev->errno != SCLIB_SUCCESS)
        return (dev->errno);

    /* 2. Update card Capabilities, read from ATR. */
    dev->pps_complete = 0;

    dev->errno = _SCLIB_UpdateCardCapabilities(num);

    /* Update failed */
    if(dev->errno != SCLIB_SUCCESS)
        return (dev->errno);

    /* 3. Parsing ATR, and PPS exchange */
    dev->errno = _SCLIB_ParseATR(num, SCLIB_COLDRESET);/* Smartcard interface number,
    Cold Reset */

    if(dev->errno != SCLIB_SUCCESS) /* Failed */
        SCLIB_ERR("Cold Reset Parse ATR..Error :%d \n", dev->errno);

    if(dev->errno == SCLIB_SUCCESS) /* Success */
        /* Smartcard has been activated. */
        dev->openflag = 1;
```

```
    return (dev->errno);
}
```

- Start Transfer

```
int32_t SCLIB_StartTransmission(uint32_t num, uint8_t *cmdBuf, uint32_t cmdLen, uint8_t
*rspBuf, uint32_t *rspLen)
{
    int32_t retval = SCLIB_SUCCESS;
    SCLIB_DEV_T *dev = &_scDev[num];

   /* 1. Not activate yet.*/
    if(dev->openflag == 0)      /* Try to access smartcard before activation. */
        return SCLIB_ERR_DEACTIVE;

    /* 2. Smartcard busy. */
    if(dev->op_state)           /* Smartcard busy, previous transmission not complete. */
        return SCLIB_ERR_CARDBUSY;

    /* Give up data remain in dev->buf, if anything remains... */
    dev->snd_pos = 0; /* Send buffer offset = 0 */

    dev->rcv_pos = 0; /* Receive buffer offset = 0 */

    /* 3. Send a command and get response from smartcard according to protocol */
    if(dev->CardCapabilities.Protocol.Selected == SCLIB_PROTOCOL_T0) /* T0 */
    {
        /* Using T=0 protocol */
        retval = _SCLIB_T0Transmit(num,     /* Smartcard interface number */
                                   cmdBuf,  /* cmd buffer */
                                   cmdLen,  /* cmd len */
                                   rspBuf,  /* response buffer */
                                   rspLen); /* response len */
    }
    else if(dev->CardCapabilities.Protocol.Selected == SCLIB_PROTOCOL_T1) /* T1 */
    {
        retval = _SCLIB_T1Transmit(num, cmdBuf, cmdLen, rspBuf, rspLen);
    }

    /* 4. Stop all timers here, the transmission is complete no matter success or not. */
    SC_Timer_Stop();

    return retval;
}
```

- Card Transfer Event Handling

This function needs to be called in the UART interrupt handling function.

```
uint32_t SCLIB_CheckTxRxEvent(uint32_t num)
{
   ……
    /* 1. Transmit buffer empty interrupt */
    if (SC_GetTxIntFlag())
    {
        retval = 1;
```

```
        if( dev->op_state == SCLIB_OP_WRITE )/* Operation = Write */
        {
            ……
        }
    }
    /* 2. Receive Data Available Interrupt or Rx TimeOut interrupt. */
    if(SC_GetRxIntFlag())
    {
        ……
        if(dev->op_state == SCLIB_OP_READ) /* Operation = Read */
        {
            ……
        }
        else if(dev->op_state == SCLIB_OP_ATR_READ)  /* Operation = Read ATR */
        {
            ……
        }
        else {
            ……
        }
    }
out:
    return retval;
}
```

In addition, the implementation code of T0 asynchronous half duplex character transmission protocol is located in *SmartcardT0Lib\Source\sclib_t0protocol.c*. The internal function code of the Smart Card protocol library is located in *SmartcardT0Lib\Source\sclib_util.c*.

Main Function

```
int32_t main(void)
{
    SCLIB_CARD_INFO_T s_info;
    int retval, i;
    uint32_t u32Ret;

    /* Unlock protected registers */
    SYS_UnlockReg();

    /* Init System, peripheral clock and multi-function I/O */
    SYS_Init();

    /* Lock protected registers */
    SYS_LockReg();

#if defined (__GNUC__) && !defined(__ARMCC_VERSION) && defined(OS_USE_SEMIHOSTING)
    initialise_monitor_handles();
#endif

    /* Init UART0 for printf and test */
    UART0_Init();

    printf("+------------------------------------------------------------------------+\n");
    printf("|     M251 UART Emulate ISO7816 Sample Code                              |\n");
```

```
    printf("+-------------------------------------------------------------------------+\n\n");
    printf(">> The I/O connection: <<\n");
    printf("    M251              Smart Card Slot\n");
    printf("    GPIO(PA.1)              VCC\n");
    printf("    GPIO(PA.2)              RST\n");
    printf("    CLKO(PA.3)              CLK\n");
    printf("    UART1_RXD(PB.6)         DAT\n");
    printf("    UART1_TXD(PB.7)         DAT\n");
    printf("    GND                     GND\n");
    printf("\nPress any key to start test.\n");

    getchar();

    /* Init ISO7816 */
    SC_Init();

    /* Activate Smartcard interface 0 */
    retval = SCLIB_ActivateDelay(0,      /* Smartcard interface number */
                                 FALSE,  /* Disable EMV error check */
                                 10000); /* Delay clk */

    if (retval == SCLIB_SUCCESS) /* Activated */
    {
        /* Copy ATR information to variable. */
        SCLIB_GetCardInfo(0, &s_info);  /* Smartcard interface number, variable */

        /* Print ATR information */
        printf("ATR: ");

        for (i = 0; i < s_info.ATR_Len; i++)
            printf("%02x ", s_info.ATR_Buf[i]);

        printf("\n\n");

        /* Smartcard transmissiontransmission test: Select file MF */
        u32Ret = SCLIB_StartTransmission(0,              /* Smartcard interface number */
                                         g_cmd11buf,     /* Cmd buffer */
                                         g_cmd1len,      /* Cmd len */
                                         g_cmdrsp,       /* Response buffer */
                                         &g_cmd1rsplen); /* Response len */

        if (u32Ret != 0) /* Error */
        {
            dwErrorNum++;
        }
        else
        {
            /* Print cmd response data */
            printf("Select file\n");

            for (i = 0; i < g_cmd1rsplen; i++)
                printf("%02x ", g_cmdrsp[i]);

            printf("\n\n");

            /* Smartcard transmission test: get response */
```

```
            u32Ret = SCLIB_StartTransmission(0,           /* Smartcard interface number */
                                             g_cmd3buf,      /* Cmd buffer */
                                             g_cmd3len,      /* Cmd len */
                                             g_cmdrsp,       /* Response buffer */
                                             &g_cmd1rsplen); /* Response len */

            if (u32Ret != 0)
            {
                dwErrorNum++;
            }
            else
            {
                /* Print cmd response data */
                printf("Get response\n");

                for (i = 0; i < g_cmd1rsplen; i++)
                    printf("%02x ", g_cmdrsp[i]);

                printf("\n");
            }
        }
    }
    else
    {
        printf("Smartcard activate failed\n");
    }

    printf("\nExit Sample code.\n");

    while (1);

}
```

3. Software and Hardware Requirements

3.1 Software Requirements

- BSP version
  - M251_M252_M254_M256_M258_Series_BSP_CMSIS_V3.02.007
- IDE version
  - Keil uVision 5.40.0.0

3.2 Hardware Requirements

- Circuit components
  - NuMaker-M251SD V1.0
- Pin Connect
  - Connect PA.1/PA.2/PA.3 to VCC/RST/CLK of the Smart Card slot respectively.
  - Connect the UART1 TX (PB.7) pin and UART1_RX (PB.6) to the DAT pin of the Smart Card slot.
  - Connect the UART0 TX (PB.13) pin and UART0 RX (PB.12) pin to the PC via Nu-Link-me.

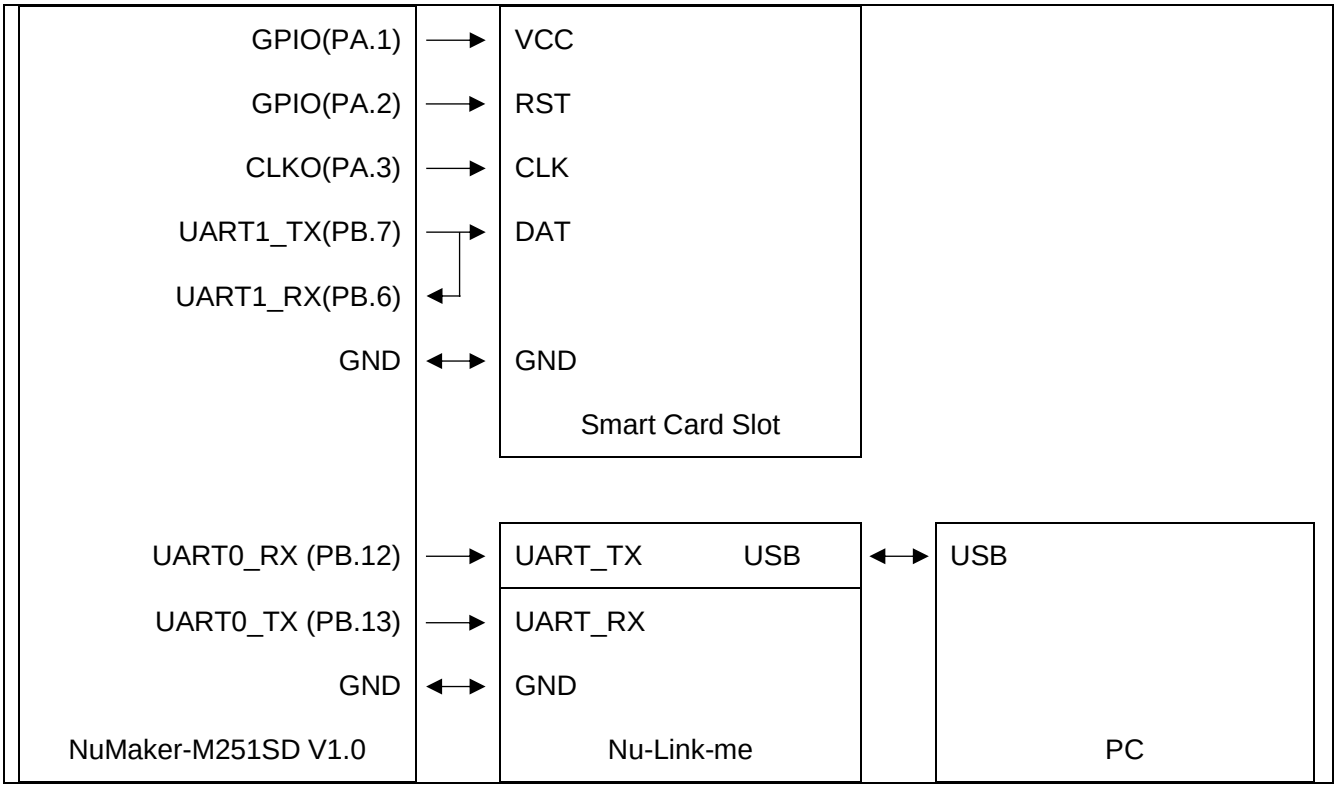


Figure 3-1 Pin Connect

4. Directory Information

The directory structure is shown below.

EC_M251_UART_Emulate_ISO7816_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

Figure 4-1 Directory Structure

5. Example Code Execution

1. Browse the sample code folder as described in the Directory Information section and double-click *M251_UART_Emulate_ISO7816.uvprojx*.
2. Enter Keil compile mode.
   - Build
   - Download
   - Start/Stop debug session
3. Enter debug mode.
   - Run

6. Revision History

Date	Revision	Description
2025.02.11	1.00	Initial version.

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.