NUC240 Control EEPROM via GPIO Emulated I²C

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	This example implements software emulated I²C by using GPIO to control EEPROM.
BSP Version	NUC230_240 Series BSP CMSIS v3.01.001
Hardware	NuEdu-EVB-NUC240 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

When user needs more I²C channels then the microcontroller provides, user can use GPIO to emulate I²C waveform. The example is a software-based I²C EEPROM solution for NUC240. This sample code provides two APIs for user:

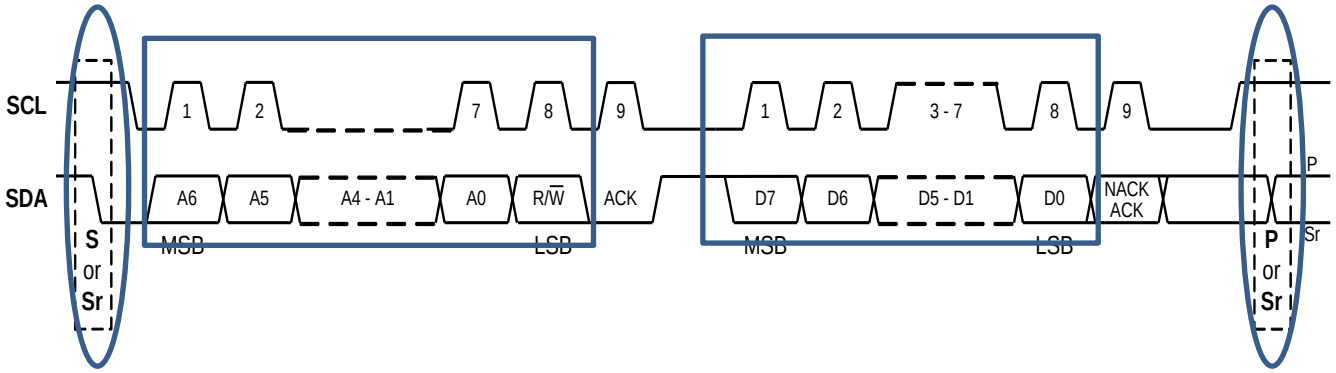
API	Function
void I2C_GPIO_Write_Byte(uint16_t u16Addr, uint8_t u8Data)	Send Character
void I2C_GPIO_Read_Byte(uint16_t u16Addr, uint8_t* pu8Data)	Receive Character

1.2 Principle

I²C bus is a two-wire bidirectional serial bus that requires a serial data line and a serial clock line to communicate.

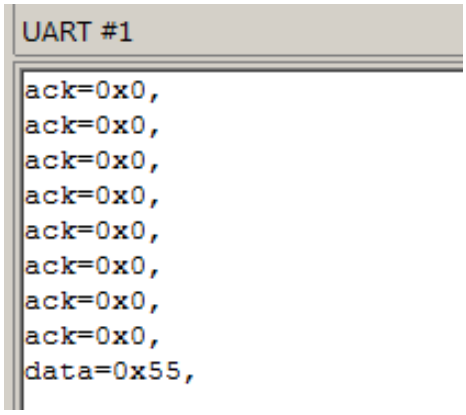
I²C standard communication consists of four stages:

1. START or Repeated START signal generation
2. 7-bit slave address transfer and 1-bit R/W bit transfer
3. 8-bit data transfer
4. STOP signal generation



1.3 Demo Result

NUC240 uses PA.10 and PA.11 to emulate SDA and SCL signals. Compile the project and enter debug mode, the serial window UART #1 will show the EEPROM's ACK and read/write data.


UART #1
ack=0x0,
ack=0x0,
ack=0x0,
ack=0x0,
ack=0x0,
ack=0x0,
ack=0x0,
ack=0x0,
data=0x55,

2 Code Description

2.1 Initial Software I²C

I2C_Init() initializes software I²C. It configures two selected GPIOs to output high on SDA and SCL lines in idle state.

```
void I2C_Init(void)
{
       /* Configure PA10 as open-drain mode */
       GPIO_SetMode(PA, BIT10, GPIO_PMD_OPEN_DRAIN);

       /* Configure PA11 as open-drain mode */
       GPIO_SetMode(PA, BIT11, GPIO_PMD_OPEN_DRAIN);

       /* Control PA11 output status */
       I2C_CLK = 1;

       /* Control PA10 output status */
       I2C_DAT = 1;
}
```

2.2 Send I²C Character

I2C_GPIO_Write_Byte() sends I²C character. Transmit the write control byte, word address and the first data byte to the EEPROM in the same way as in a byte write. Please refer to EEPROM specification for details.

```
void I2C_GPIO_Write_Byte(uint16_t u16Addr, uint8_t u8Data)
{
       uint8_t u8AddrH, u8AddrL;

       u8AddrH = u16Addr >> 8;
       u8AddrL = (uint8_t)u16Addr;

       /* Send Start bit to I2C EEPROM */
       I2C_Start();

       /* Send control byte to I2C EEPROM */
       printf("ack=0x%x,\n\r",I2C_Write(I2C_ADDRESS_W));
```

```
	/* Send I2C EEPROM's High Byte Address */
	printf("ack=0x%x,\n\r",I2C_Write(u8AddrH));

	/* Send I2C EEPROM's Low Byte Address */
	printf("ack=0x%x,\n\r",I2C_Write(u8AddrL));

	/* Send data byte to I2C EEPROM */
	printf("ack=0x%x,\n\r",I2C_Write(u8Data));

	/* Send Stop bit to I2C EEPROM */
	I2C_Stop();
}
```

2.3 Receive I²C Character

I2C_GPIO_Read_Byte() receives I²C character, includes the write control byte, word address and the first data byte from the EEPROM. Please refer to EEPROM specification for details.

```
void I2C_GPIO_Read_Byte(uint16_t u16Addr, uint8_t* pu8Data)
{
	uint8_t u8AddrH, u8AddrL;

	u8AddrH = u16Addr >> 8;
	u8AddrL = (uint8_t)u16Addr;

	/* Send Start bit to I2C EEPROM */
	I2C_Start();

	/* Send control byte to I2C EEPROM */
	printf("ack=0x%x,\n\r",I2C_Write(I2C_ADDRESS_W));

	/* Send I2C EEPROM's High Byte Address */
	printf("ack=0x%x,\n\r",I2C_Write(u8AddrH));

	/* Send I2C EEPROM's Low Byte Address */
	printf("ack=0x%x,\n\r",I2C_Write(u8AddrL));

	/* Send Start bit to I2C EEPROM */
	I2C_Start();
```

```
        /* Send control byte to I2C EEPROM */
        printf("ack=0x%x,\n\r",I2C_Write(I2C_ADDRESS_R));

        /* Read data byte from EEPROM */
        *pu8Data = I2C_Read(NO_ACK);

        /* Send Stop bit to I2C EEPROM */
        I2C_Stop();
}
```

3 Software and Hardware Environment

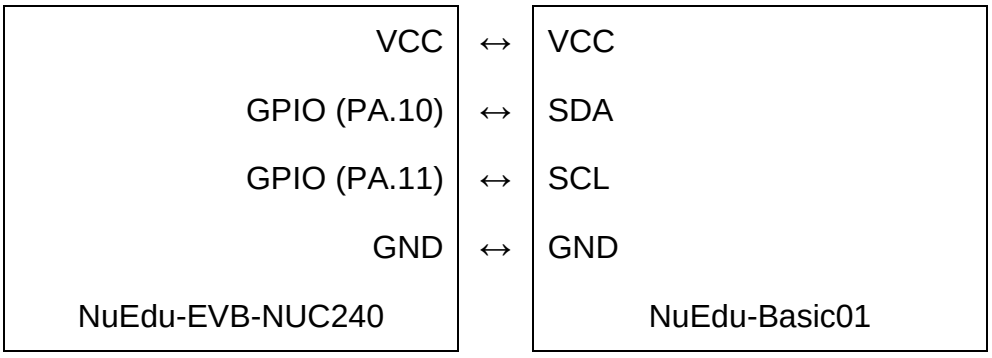
Software Environment

- BSP version
  - NUC230_240 Series BSP CMSIS v3.01.001
- IDE version
  - Keil uVersion 4.74

Hardware Environment

- Circuit components
  - NuEdu-EVB-NUC240 V1.0
  - NuEdu-Basic01 V2.1
- Diagram

  NuEdu-EVB-NUC240 uses PA.10 and PA.11 to simulate I^2C waveform and control the EEPROM on NuEdu-Basic01. Connect Nu-Bridge with NuEdu-Basic01 to print the debug message on the terminal.



4 Directory Information

- EC_ NUC240_GPIO_I2C_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
    - NuEdu — Library for NuEdu-SDK-NUC240 board
    - 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 NUC240_GPIO_I2C.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. 1, 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.