NUC240 Controls DS3231 via I^{2}C

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	The example use NUC240 to control DS3231 by using I^{2}C
BSP Version	NUC230_240 Series BSP CMSIS v3.01.001
Hardware	NuEdu-EVB-NUC240 v1.0

1 Function Description

1.1 Introduction

The example is the driver of DS3231 based on NUC240 series. NUC240 series read RTC information from DS3231 via I^{2}C interface, and display on the terminal. This sample code provides two API for user:

API	Function
void DS3231_RTC_Write(uint8_t u8Reg, uint8_t u8Value)	Set the time to DS3231
uint8_t DS3231_RTC_Read(uint8_t u8Reg)	Get the time from DS3231

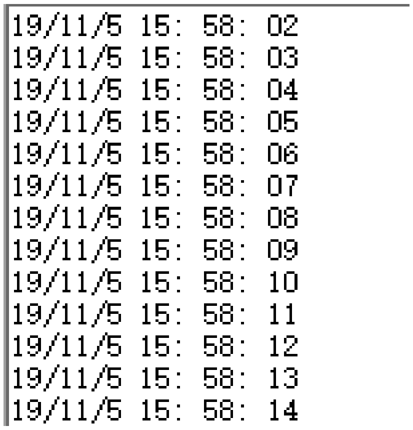
1.2 Principle

DS3231 features:

- Low cost high accuracy real time clock.
- Real time clock generator for seconds, minutes, hours, day of week, date, month, and year timing.
- I^{2}C interface
- The accuracy from 0 °C to +40 °C is ±2 ppm and -40 °C to +85 °C is ±3.5 ppm.

1.3 Demo Result

NUC240 controls DS3231 via I^{2}C interface. Compile the project and enter debug mode, the serial window UART #1 will show the current time continuously.



```
19/11/5 15: 58: 02
19/11/5 15: 58: 03
19/11/5 15: 58: 04
19/11/5 15: 58: 05
19/11/5 15: 58: 06
19/11/5 15: 58: 07
19/11/5 15: 58: 08
19/11/5 15: 58: 09
19/11/5 15: 58: 10
19/11/5 15: 58: 11
19/11/5 15: 58: 12
19/11/5 15: 58: 13
19/11/5 15: 58: 14
```

2 Code Description

2.1 Write time to DS3231

Use DS3231_RTC_Write() function to set DS3231 default information. DS3231 has an I^{2}C address 0x68 by default and cannot be changed.

```
void DS3231_RTC_Write(uint8_t u8Reg, uint8_t u8Value)
{
    /* DS3231 I2C slave address */
    g_u8DeviceAddr = 0x68;

    /* DS3231 address map */
    g_au8TxData[0] = u8Reg;

    /* Write data to DS3231 */
    g_au8TxData[1] = dec_to_bcd(u8Value);
    g_u8DataLen = 0;
    g_u8EndFlag = 0;

    /* I2C function to write data to slave */
    s_I2C0HandlerFn = (I2C_FUNC)I2C_MasterTx;

    /* I2C as master sends START signal */
    I2C_SET_CONTROL_REG(I2C0, I2C_I2CON_STA);

    /* Wait I2C Tx Finish */
    while (g_u8EndFlag == 0);

    g_u8EndFlag = 0;
}
```

2.2 Read time from DS3231

Use DS3231_RTC_Read() function to get the information from DS3231. It will update the correct time and display on the terminal continuously.

```
uint8_t DS3231_RTC_Read(uint8_t u8Reg)
{
    /* DS3231 I2C slave address */
    g_u8DeviceAddr = 0x68;
    /* DS3231 address map */
    g_au8TxData[0] = u8Reg;

    g_u8DataLen = 0;
    g_u8EndFlag = 0;

    /* I2C function to read data from slave */
    s_I2C0HandlerFn = (I2C_FUNC)I2C_MasterRx;

    /* I2C as master sends START signal */
    I2C_SET_CONTROL_REG(I2C0, I2C_I2CON_STA);

    /* Wait I2C Rx Finish */
    while (g_u8EndFlag == 0);

    g_u8EndFlag = 0;

    return bcd_to_dec(g_u8RxData);
}
```

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
    - DS3231
  - Diagram

The pin connections between the NUC240 and the DS3231 are detailed below.

NUC240		DS3231
VCC	↔	VCC
I2C0_SCL (PA.9)	↔	SCL
I2C0_SDA (PA.8)	↔	SDA
GND	↔	GND

4 Directory Information

🗁 EC_NUC240_I2C_DS3231_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_I2C_DS3231.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.