Application Note

32-bit Cortex™-M0 MCU
NuMicro® Family

How to access 24C64 via IIC in NUC1xx?

Table of Contents-

1 INTRODUCTION 2
2 I2C FEATURES 2
3 I2C FUNCTION DESCRIPTION 3
3.1 Address Registers (I2ADDR) 3
3.2 Data Register (I2DAT) 3
3.3 Control Register (I2CON) 4
3.4 Status Register (I2STATUS) 5
3.5 I2C Clock Baud Rate Bits (I2CLK) 5
3.6 The I2C Time-out Counter Register (I2TOC) 6
4 EEPROM 7
5 CIRCUIT 7
6 SAMPLE CODE 8
7 REVISION HISTORY 11

1 INTRODUCTION

This application note describes the deisign of I2C EEPROM using the NUC1XX device.

2 I2C FEATURES

- AMBA APB interface compatible
- Compatible with Philips I2C standard, support master mode
- Master/Slave up to 1Mbit/s (Fast-mode Plus)
- Bidirectional data transfer between masters and slaves
- Multi-master bus (no central master)
- Arbitration between simultaneously transmitting masters without corruption of serial data on the bus
- Serial clock synchronization allows devices with different bit rates to communicate via one serial bus
- Serial clock synchronization can be used as a handshake mechanism to suspend and resume serial transfer
- Built-in a 14-bit time-out counter will request the I2C interrupt if the I2C bus hangs up and timer-out counter overflows.
- External pull-up are needed for high output but do not support powering off of individual devices connected to the same bus
- Programmable clocks allow versatile rate control
- Supports 7 bit addressing mode
- I2C-bus controllers support multiple address recognition (Four slave address with mask option)

3 I2C FUNCTION DESCRIPTION

The CPU interfaces to the SIO port through the following six special function registers: **I2CON** (control register, C0H), **I2STATUS** (status register, BDH), **I2DAT** (data register, BCH), **I2ADDR** (address registers, C1H), **I2CLK** (clock rate register BEH) and **I2TOC** (Time-out counter register, BFH) All bit 31~ bit 8 of the I2C function register are reserved. These bits do not have any function and read back is all zero.

When I2C port is enabled by setting ENS1 to high, the internal states will be controlled by I2CON and I2C logic hardware. Once a new status code is generated and stored in I2STATUS, the I2C interrupt flag (SI) will be set automatically. If EI2 is set high at this time, the I2C interrupt will be requested. The 5 most significant bits of I2STATUS stores the internal state code, the lowest 3 bits are always zero and the content keeps stable until SI is cleared by software. The base address on NUC1XX is 4002_0000 and 4012_0000.

3.1 Address Registers (I2ADDR)

I2C port is equipped with four slave address register. The contents of the register are irrelevant when I2C is in master mode. In the slave mode, the seven most significant bits must be loaded with the MCU's own slave address. The I2C hardware will react if the contents of I2ADDR are matched with the received slave address.

The I2C ports support the "General Call" function. If the GC bit is set the I2C port hardware will respond to General Call address (00H). Clear GC bit to disable general call function.

When GC bit is set, the I2C is in Slave mode, it can be received the general call address by 00H after Master send general call address to I2C bus, then it will follow status of GC mode. If it is in Master mode, the AA bit must be cleared when it will send general call address of 00H to I2C bus.

I2C-bus controllers support multiple address recognition with four address mask reg. When the bit in the address mask register is set to one, that means the received corresponding address bit is don't care. If the bit is set to zero, that means the received corresponding register bit should be exact the same as address register.

SYMBOL	DEFINITION	ADDRESS	MSB BIT ADDRESS, SYMBOL LSB			RESET
			Bit 31	Bit7	Bit0	
I2ADDR0	I2C ADDRESS0	I2C_BA+04H	Reserved	ADDR0.6 ~ ADDR0.0	GC0	0000 0000H
I2ADDR1	I2C ADDRESS1	I2C_BA+18H	Reserved	ADDR1.6 ~ ADDR1.0	GC1	0000 0000H
I2ADDR2	I2C ADDRESS2	I2C_BA+1cH	Reserved	ADDR2.6 ~ ADDR2.0	GC2	0000 0000H
I2ADDR3	I2C ADDRESS3	I2C_BA+20H	Reserved	ADDR3.6 ~ ADDR3.0	GC3	0000 0000H
I2ADRM0	I2C ADDRESS MASK0	I2C_BA+24H	Reserved	ADRM0.6 ~ ADRM0.0	Reserved	0000 000XH
I2ADRM1	I2C ADDRESS MASK1	I2C_BA+28H	Reserved	ADRM1.6 ~ ADRM1.0	Reserved	0000 000XH
I2ADRM2	I2C ADDRESS MASK2	I2C_BA+2CH	Reserved	ADRM2.6 ~ ADRM2.0	Reserved	0000 000XH
I2ADRM3	I2C ADDRESS MASK3	I2C_BA+30H	Reserved	ADRM3.6 ~ ADRM3.0	Reserved	0000 000XH

3.2 Data Register (I2DAT)

This register contains a byte of serial data to be transmitted or a byte which has just been received. The CPU can read from or write to this 8-bit directly addressable SFR while it is not in the process of shifting a byte. This occurs when SIO is in a defined state and the serial interrupt flag (SI) is set. Data in I2DAT remains stable as long as SI bit is set. While data is being shifted out, data on the bus is simultaneously being shifted in; I2DAT always contains the last data byte present on the bus. Thus, in

the event of arbitration lost, the transition from master transmitter to slave receiver is made with the correct data in I2DAT.

I2DAT and the acknowledge bit form a 9-bit shift register, the acknowledge bit is controlled by the SIO hardware and cannot be accessed by the CPU. Serial data is shifted through the acknowledge bit into I2DAT on the rising edges of serial clock pulses on the SCL line. When a byte has been shifted into I2DAT, the serial data is available in I2DAT, and the acknowledge bit (ACK or NACK) is returned by the control logic during the ninth clock pulse. Serial data is shifted out from I2DAT on the falling edges of SCL clock pulses, and is shifted into I2DAT on the rising edges of SCL clock pulses.

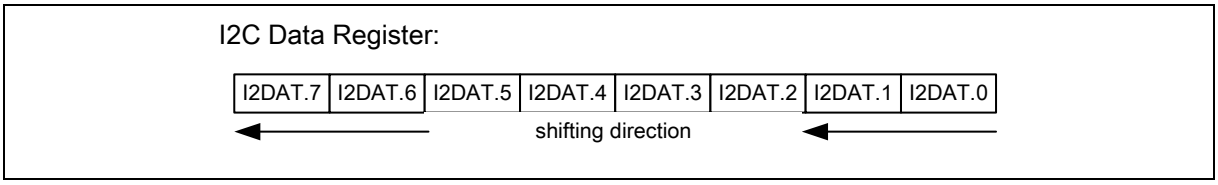


Figure 3-1 I2C Data Shifting Direction

SYMBOL	DEFINITION	ADDRESS	MSB BIT ADDRESS, SYMBOL LSB		RESET
			Bit31	Bit7 Bit0	
I2DAT	I2C DATA REGISTER	I2C_BA+08H	Reserved	I2DAT.7 ~ I2DAT.0	000000xxh

3.3 Control Register (I2CON)

The CPU can read from and write to this 8-bit, directly addressable SFR. Two bits are affected by hardware: the SI bit is set when the I2C hardware requests a serial interrupt, and the STO bit is cleared when a STOP condition is present on the bus. The STO bit is also cleared when ENS1 = "0".

EI Enable interrupt.

ENSI Set to enable I2C serial function block. When ENS=1 the I2C serial function enables. The port latches of SDA1 and SCL1 must be set to logic high.

STA I2C START Flag. Setting STA to logic 1 to enter master mode, the I2C hardware sends a START or repeat START condition to bus when the bus is free.

STO I2C STOP Flag. In master mode, setting STO to transmit a STOP condition to bus then I2C hardware will check the bus condition if a STOP condition is detected this flag will be cleared by hardware automatically. In a slave mode, setting STO resets I2C hardware to the defined “not addressed” slave mode. This means it is NO LONGER in the slave receiver mode to receive data from the master transmit device.

SI I2C Interrupt Flag. When a new SIO state is present in the I2STATUS register, the SI flag is set by hardware, and if the EA and EI2 bits are both set, the I2C interrupt is requested. SI must be cleared by software. Clear SI is by writing one to this bit.

AA Assert Acknowledge control bit. When AA=1 prior to address or data received, an acknowledged (low level to SDA) will be returned during the acknowledge clock pulse on the SCL line when 1.) A slave is acknowledging the address sent from master, 2.) The receiver devices are acknowledging the data sent by transmitter. When AA=0 prior to address or data received, a Not acknowledged (high level to SDA) will be returned during the acknowledge clock pulse on the SCL line.

SYMBOL	DEFINITION	ADDRESS	MSB	BIT ADDRESS, SYMBOL	LSB	RESET

			Bit31	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit0	
I2CON	I2C CONTROL REGISTER	I2C_BA+00H	Reserved	EI	ENS1	STA	STO	SI	AA	- -	0000 0000H

3.4 Status Register (I2STATUS)

I2STATUS is an 8-bit read-only register. The three least significant bits are always 0. The five most significant bits contain the status code. There are 26 possible status codes. When I2STATUS contains F8H, no serial interrupt is requested. All other I2STATUS values correspond to defined SIO states. When each of these states is entered, a status interrupt is requested (SI = 1). A valid status code is present in I2STATUS one machine cycle after SI is set by hardware and is still present one machine cycle after SI has been reset by software.

In addition, state 00H stands for a Bus Error. A Bus Error occurs when a START or STOP condition is present at an illegal position in the format frame. Examples of illegal positions are during the serial transfer of an address byte, a data byte or an acknowledge bit. To recover I2C from bus error, STO should be set and SI should be clear to enter not addressed slave mode. Then clear STO to release bus and to wait new communication. I2C bus can not recognize stop condition during this action when bus error occurs.

SYMBOL	DEFINITION	ADDRESS	MSB	BIT ADDRESS, SYMBOL	LSB	RESET
			Bit31	Bit8 Bit7	Bit0	
I2STATUS	I2C STATUS REGISTER	I2C_BA+0CH	Reserved	8 bit status data		000000F8H

3.5 I2C Clock Baud Rate Bits (I2CLK)

The data baud rate of I2C is determines by I2CLK register when SIO is in a master mode. It is not important when SIO is in a slave mode. In the slave modes, SIO will automatically synchronize with any clock frequency up to 400 KHz from master I2C device.

The data baud rate of I2C setting is Data Baud Rate of I2C = PCLK /(4x(I2CLK+1)). If PCLK=16MHz, the I2CLK = 40(28H), so data baud rate of I2C = 16MHz/(4X (40 +1)) = 97.5Kbits/sec. The block diagram is as below figure.

SYMBOL	DEFINITION	ADDRESS	MSB	BIT ADDRESS, SYMBOL	LSB	RESET
			Bit31	Bit8 Bit7	Bit0	
I2CLK	I2C CLOCK RATE	I2C_BA+10H	Reserved	I2CLK.7 ~ I2CLK.0		0000 0000H

3.6 The I2C Time-out Counter Register (I2TOC)

There is a 14-bit time-out counter which can be sued to deal with the I2C bus hang-up. If the time-out counter is enabled, the counter starts up counting until it overflows(TIF=1) and requests I2C interrupt from CPU or stops counting by clearing ENTI to 0. When time-out counter is enabled, setting flag SI to high will reset counter and re-start up counting after SI is cleared. If I2C bus hangs up, it causes the I2STATUS and flag SI are not updated for a period, the 14-bit time-out counter may overflow and acknowledge CPU the I2C interrupt. Refer to Figure 3-2 for the 14-bit time-out counter. User may clear TIF by write one to this bit.

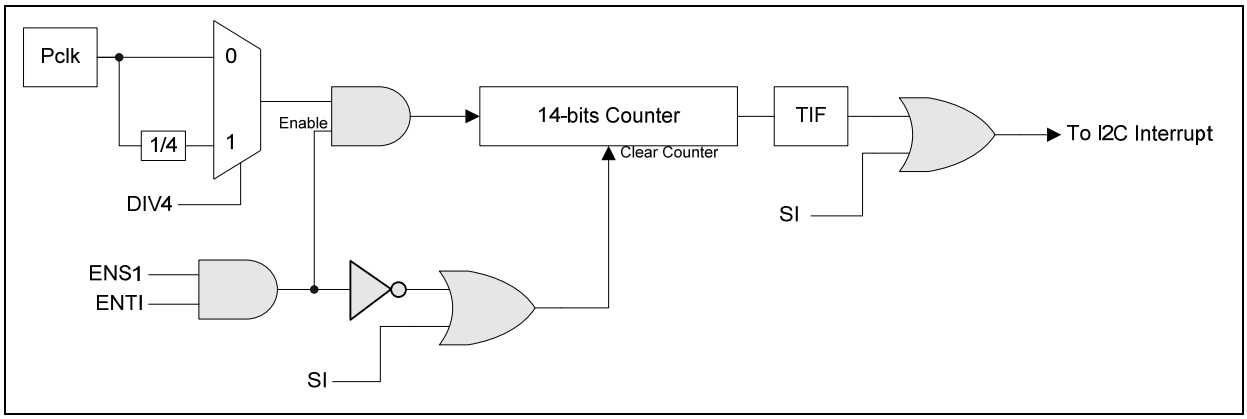


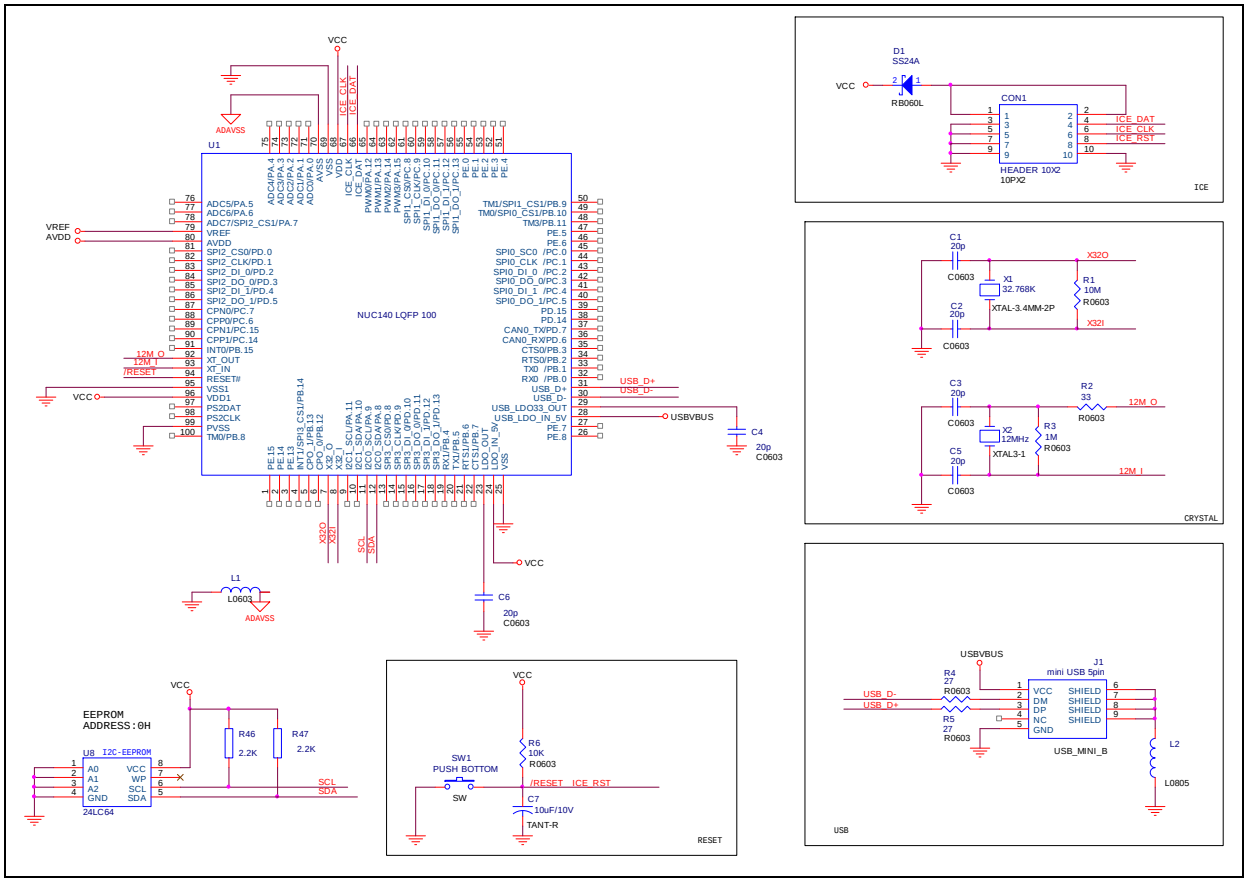
Figure 3-2: I2C Time-out Count Block Diagram

SYMBOL	DEFINITION	ADDRESS	MSB	BIT ADDRESS, SYMBOL			LSB	RESET
			Bit31	Bit2	Bit1	Bit0		
I2TOC	I2C TIME-OUT COUNTER REGISTER	I2C_BA+14H	Reserved	ENTI	DIV4	TIF		0000 0000B

4 EEPROM

This I2C EEPROM sampe for 24LC64, please refer the 24LC64 SPEC.

5 CIRCUIT



6 SAMPLE CODE

```
#include <stdio.h>
#include "NUC1xx.h"
#include "Driver\DrvGPIO.h"
#include "Driver\DrvI2C.h"
#include "Driver\DrvSYS.h"
#include "Driver\DrvUART.h"

/*---------------------------------------------------------*/
/*                    MAIN Function                        */
/*---------------------------------------------------------*/
int main (void)
{
      uint32_t u32HCLK;
      /* SYSCLK =>12Mhz*/
      UNLOCKREG();
       SYSCLK->PWRCON.XTL12M_EN = 1;
      u32HCLK = DrvSYS_GetHCLK() * 1000;


      /* Set I2C I/O */
      DrvGPIO_InitFunction(FUNC_I2C0);

      /* Open I2C0 and set clock = 100Kbps */
      DrvI2C_Open(I2C_PORT0, u32HCLK, 100000);

      //send i2c start
      DrvI2C_Ctrl(I2C_PORT0, 1, 0, 0, 0);     //set start
      while (I2C0->CON.SI == 0);              //poll si flag
      //send writer command
      I2C0->DATA = 0XA0;                      //send writer command
      DrvI2C_Ctrl(I2C_PORT0, 0, 0, 1, 0);     //clr si flag
      while( I2C0->CON.SI == 0 );             //poll si flag
      //send address high
      I2C0->DATA = 0X00;
      DrvI2C_Ctrl(I2C_PORT0, 0, 0, 1, 1);     //clr si and set ack
```

```
        while( I2C0->CON.SI == 0 );             //poll si flag
        //send address low
        I2C0->DATA = 0X01;
        DrvI2C_Ctrl(I2C_PORT0, 0, 0, 1, 1);     //clr si and set ack

        while( I2C0->CON.SI == 0 );             //poll si flag
        //send data
        I2C0->DATA = 0X55;                      //write data to
        DrvI2C_Ctrl(I2C_PORT0, 0, 0, 1, 1);     //clr si and set ack

        while( I2C0->CON.SI == 0 );             //poll si flag
        //send i2c stop
        DrvI2C_Ctrl(I2C_PORT0, 0, 1, 1, 0);     //send stop
        DrvI2C_Close(I2C_PORT0);



        /* Open I2C0 and set clock = 100Kbps */
        DrvI2C_Open(I2C_PORT0, u32HCLK, 100000);
        //send i2c start
        DrvI2C_Ctrl(I2C_PORT0, 1, 0, 0, 0);     //set start
        while (I2C0->CON.SI == 0);              //poll si flag

        //send writer command
        I2C0->DATA = 0XA0;
        DrvI2C_Ctrl(I2C_PORT0, 0, 0, 1, 0);     //clr si
        while( I2C0->CON.SI == 0 );             //poll si flag

        //send address high
        I2C0->DATA = 0X00;
        DrvI2C_Ctrl(I2C_PORT0, 0, 0, 1, 1);     //clr si and set ack
        while( I2C0->CON.SI == 0 );             //poll si flag

        //send address low
        I2C0->DATA = 0X01;
        DrvI2C_Ctrl(I2C_PORT0, 0, 0, 1, 1);     //clr si and set ack
        while( I2C0->CON.SI == 0 );             //poll si flag

        //send start flag
```

```
        DrvI2C_Ctrl(I2C_PORT0, 1, 0, 1, 0);     //clr si and send start

        while( I2C0->CON.SI == 0 );             //poll si flag

        //send read command
        I2C0->DATA = 0XA1;
        DrvI2C_Ctrl(I2C_PORT0, 0, 0, 1, 0);     //clr si
         while( I2C0->CON.SI == 0 );            //poll si flag

        //resive data
        I2C0->DATA = 0X00;
        DrvI2C_Ctrl(I2C_PORT0, 0, 0, 1, 0);     //clr si
        while( I2C0->CON.SI == 0 );             //poll si flag
        //send i2c stop
        DrvI2C_Ctrl(I2C_PORT0, 0, 1, 1, 0);     //clr si and set stpo
        DrvI2C_Close(I2C_PORT0);

         while(1);
}
```

7 REVISION HISTORY

REV.	DATE	DESCRIPTION
0.01	March 11, 2010	1. Initially issued.

Important Notice

Nuvoton products are not designed, intended, authorized or warranted for use as components in systems or equipment intended for surgical implantation, atomic energy control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, combustion control instruments, or for other applications intended to support or sustain life. Further more, Nuvoton products are not intended for applications wherein failure of Nuvoton products could result or lead to a situation wherein personal injury, death or severe property or environmental damage could occur.

Nuvoton customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Nuvoton for any damages resulting from such improper use or sales.

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.