EDDI
Electronic Design
Development Institute

에디로봇아카데미

임베디드 마스터 Lv2 과정

제 1기

2022. 03. 04

손표훈

CONTENTS

EDDI
Electronic Design
Development Institute

TMS570LC43x의 ADC 특징

1. 7bit, 10bit addressing modes를 따르고 bit/byte 형태로 전송이 가능하다
2. 다중 마스터/슬레이브 수신/송신 모드가 가능하다
3. 전송속도는 10kbps~400kbps까지 가능하다
4. Free data format 기능을 지원한다
5. DMA를 통해 module 데이터 전송이 가능하다
6. VBUS(I2C의 데이터 버스)의 주파수는 6.7MHz이상 지원한다
7. I2C 모듈에 인가되는 주파수는 6.7MHz ~ 13.3MHz까지 가능하다
8. I2C SDA/SCL을 open drain구조로 구성할 수 있어 출력레벨변환이 가능하다
9. SDA/SCL을 pull-up/down구조로 구성할 수 있다

* Open Drain 구조이지만 datasheet상 3~3.6V를 권장한다(Max = 4.6V)
SDA/SCL의 전압은 VCCIO에 의해 공급된다

31.1 Overview

The I2C has the following features:

- Compliance to the Philips I²C bus specification, v2.1 (*The I2C Specification*, Philips document number 9398 393 40011)
  - Bit/Byte format transfer
  - 7-bit and 10-bit device addressing modes
  - General call
  - START byte
  - Multi-master transmitter/ slave receiver mode
  - Multi-master receiver/ slave transmitter mode
  - Combined master transmit/receive and receive/transmit mode
  - Transfer rates of 10 kbps up to 400 kbps (Phillips fast-mode rate)
- Free data format
- Two DMA events (transmit and receive)
- DMA event enable/disable capability
- Seven interrupts that can be used by the CPU
- Operates with VBUS frequency from 6.7 MHz up
- Operates with module frequency between 6.7 MHz to 13.3 MHz
- Module enable/disable capability
- The SDA and SCL are optionally configurable as general purpose I/O
- Slew rate control of the outputs
- Open drain control of the outputs
- Programmable pullup/pulldown capability on the inputs
- Supports Ignore NACK mode

5.4 Device Recommended Operating Conditions(1)

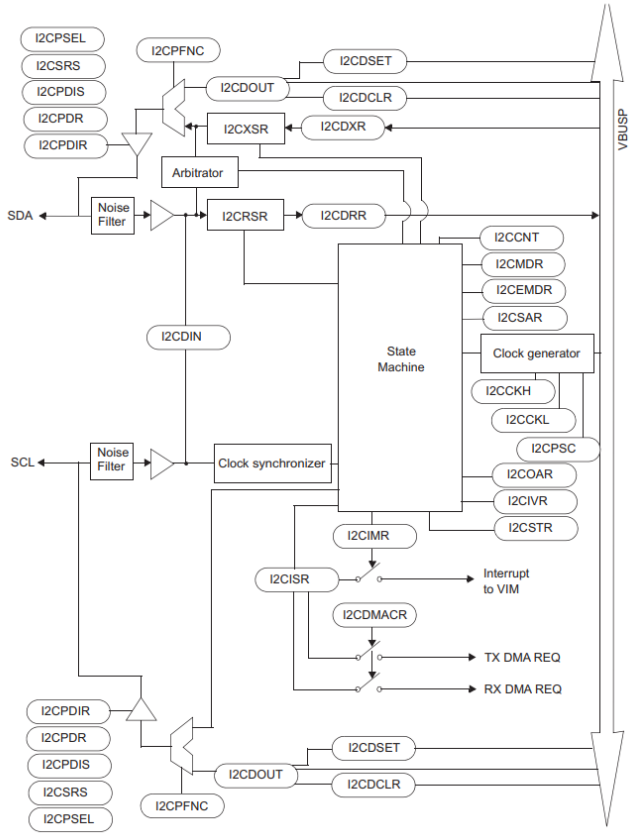
		MIN	NOM	MAX	UNIT
V_{CC}	Digital logic supply voltage (Core)	1.14	1.2	1.32	V
V_{CCP}	PLL supply voltage	1.14	1.2	1.32	V
V_{CCIO}	Digital logic supply voltage (I/O)	3	3.3	3.6	V
V_{CCAD}	MibADC supply voltage	3	3.3	3.6	V
V_{CCP}	Flash pump supply voltage	3	3.3	3.6	V
V_{SS}	Digital logic supply ground		0		V
V_{SSAD}	MibADC supply ground	–0.1		0.1	V
$V_{ADREFHI}$	Analog-to-Digital (A-to-D) high-voltage reference source	V_{SSAD}		V_{CCAD}	V
$V_{ADREFLO}$	A-to-D low-voltage reference source	V_{SSAD}		V_{CCAD}	V
T_A	Operating free-air temperature	–40		125	°C
T_J	Operating junction temperature	–40		150	°C

(1) All voltages are with respect to V_{SS}, except V_{CCAD}, which is with respect to V_{SSAD}.

I2C의 H/W 구성

EDDI
Electronic Design
Development Institute

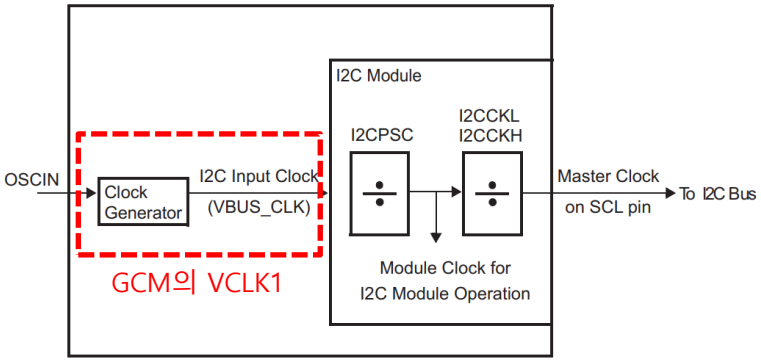
Figure 31-2. Simple I2C Block Diagram



I2C의 H/W 구성

➢ I2C의 Clock Generation

Figure 31-3. Clocking Diagram for the I2C Module



The resulting frequency is:

$$MasterClockFrequency = \frac{ModuleClockFrequency}{(I2CCKL + d) + (I2CCKH + d)}$$

$$MasterClockFrequency = \frac{I2CInputClockFrequency}{(I2CPSC + 1)((I2CCKL + d) + (I2CCKH + d))}$$

where *d* depends on the value of I2CPSC:

I2CPSC	d
0	7
1	6
Greater than 1	5

→ GCM의 VCLK1으로 부터 클럭 소스를 입력받아 module clock을 I2CPSC레지스터를 통해 설정한다

→ I2CCKL, I2CCKH레지스터 설정을 통해 SCL의 데이터 전송 클럭을 설정한다

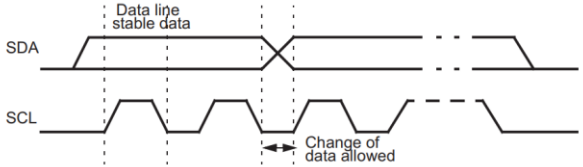
I2C 동작

→ TMS570LC43x의 I2C가 어떻게 동작하는지 살펴보자

(1) I2C module reset : I2CMDR레지스터의 IRS bit를 0으로 설정하므로 reset을 한다

(2) Data validity : I2CMDR레지스터의 IRS bit를 0으로 설정하므로 reset을 한다

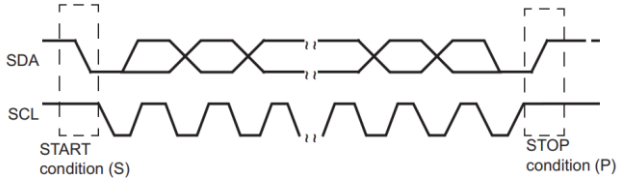
Figure 31-4. Bit Transfer on the I2C Bus



→ Data는 SCL이 H 동안 유지상태에 있어야 하며, SCL이 L 상태에서 데이터 변환이 가능하다

(3) Start/Stop Condition : I2C 모듈은

Figure 31-5. I2C Module START and STOP Conditions



→ SCL이 H 상태에서 SDA가H -> L인 조건이 Start 상태, SCL이 H 상태에서 SDA L -> H가 되는 조건이 Stop 상태가 된다

→ I2CMDR MST bit와 STT bit를 set하면서 데이터 전송 시작 상태를 설정할 수 있다

→ 반대로 I2CMDR의 STP bit를 set하면서 데이터 전송 중지 상태를 설정할 수 있다

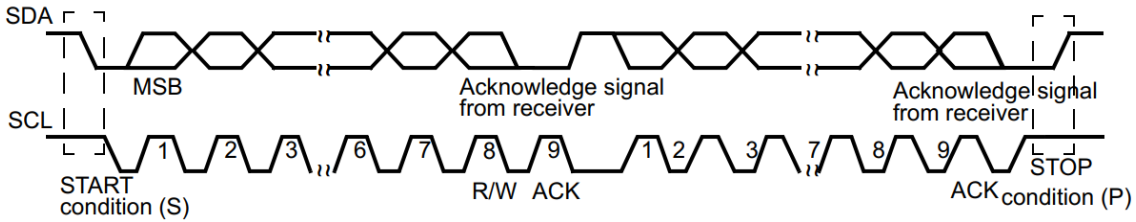
I2C 동작

EDDI
Electronic Design Development Institute

→ TMS570LC43x의 I2C가 어떻게 동작하는지 살펴보자

(4) Serial data transfer

Figure 31-6. I2C Module Data Transfer



→ I2C를 2~8bit까지 data길이를 설정한다

→ Start -> data (2~8bit) -> R/W(R : 1, W : 0) -> ACK의 순서로 데이터를 SCL에 동기화하여 보내게 된다

(5) I2C의 Addressing mode는 2가지이다 : 7bit / 10bit

Figure 31-7. I2C Module 7-Bit Addressing Format

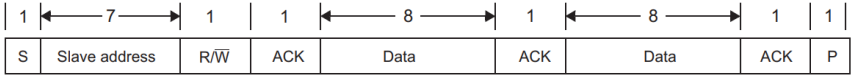
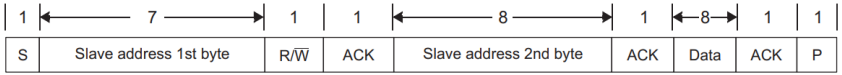


Figure 31-8. I2C Module 10-bit Addressing Format



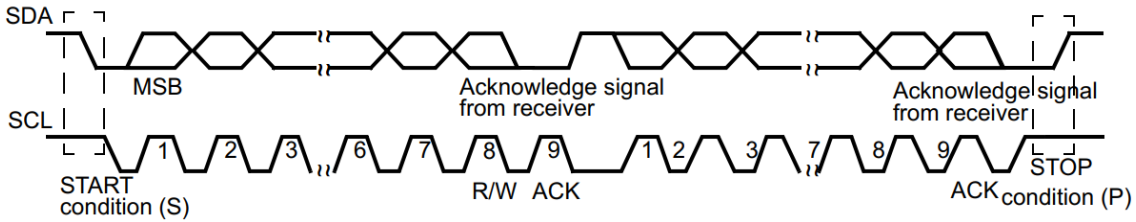
I2C 동작

EDDI
Electronic Design Development Institute

→ TMS570LC43x의 I2C가 어떻게 동작하는지 살펴보자

(4) Serial data transfer

Figure 31-6. I2C Module Data Transfer



→ I2C를 2~8bit까지 data길이를 설정한다

→ Start -> data (2~8bit) -> R/W(R : 1, W : 0) -> ACK의 순서로 데이터를 SCL에 동기화하여 보내게 된다

(5) I2C의 Addressing mode는 2가지이다 : 7bit / 10bit

Figure 31-7. I2C Module 7-Bit Addressing Format

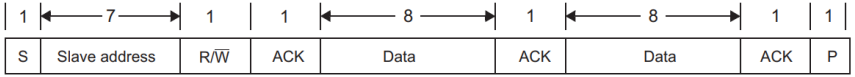
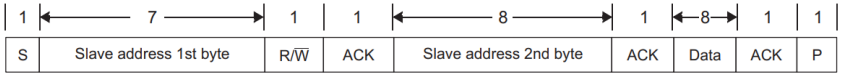


Figure 31-8. I2C Module 10-bit Addressing Format



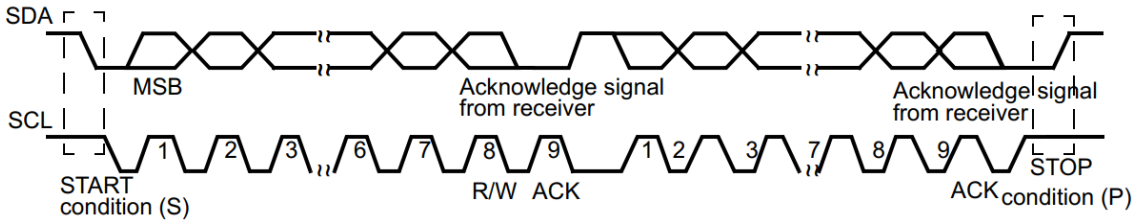
I2C 동작

EDDI
Electronic Design Development Institute

→ TMS570LC43x의 I2C가 어떻게 동작하는지 살펴보자

(4) Serial data transfer

Figure 31-6. I2C Module Data Transfer



→ I2C를 2~8bit까지 data길이를 설정한다

→ Start -> data (2~8bit) -> R/W(R : 1, W : 0) -> ACK의 순서로 데이터를 SCL에 동기화하여 보내게 된다

(5) I2C의 Addressing mode는 2가지이다 : 7bit / 10bit

Figure 31-7. I2C Module 7-Bit Addressing Format

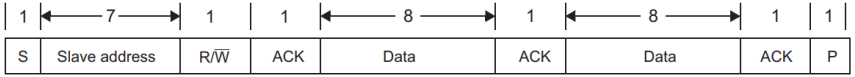
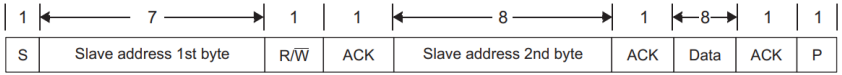


Figure 31-8. I2C Module 10-bit Addressing Format

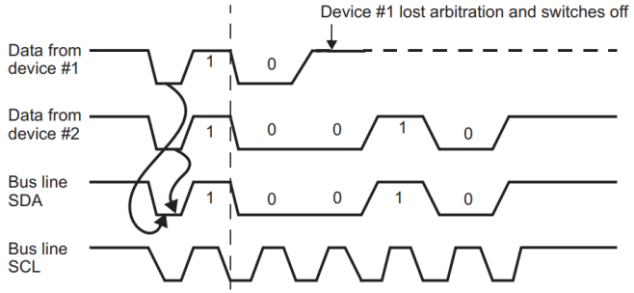


I2C 동작

→ TMS570LC43x의 I2C가 어떻게 동작하는지 살펴보자

(6) multi-master의 Arbitration 동작

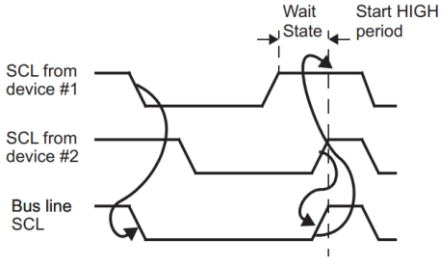
Figure 31-11. Arbitration Procedure Between Two Master Transmitters



(1) 두 개 이상의 마스터가 동시에 데이터를 전송할 경우 중재가 발생한다
(2) 데이터가 더 낮은 값이 BUS에서 dominant하다
(3) 옆 그림에서 보면 device#1의 데이터 보다 device#2의 데이터가 더 낮은 값을 가지므로 bus에서 dominant하다
(4) 이 때 device#1은 lost arbitration상태가 되고 I2CSTR의 AL flag bit를 set한다
(5) arbitration lost interrupt를 요청한다
(6) interrupt 또는 flag bit를 이용하여 후속 처리를 해준다

(7) multi-master의 clock sync 동작

Figure 31-12. Synchronization of Two I2C Clock Generators During Arbitration



(1) 두개 이상의 마스터가 서로 다른 시간에 데이터를 보낸 상황이거나
한 개의 마스터의 SCL low 시간이 긴 경우 해당 마스터가 SCL line을 점유한다
(2) 다른 장치들은 SCL wait 상태로 만든 뒤 SCL line을 점유하고 있는
마스터 클럭에 동기하여 데이터 전송을 시작할 준비를 한다

I2C 동작

EDDI
Electronic Design Development Institute

→ TMS570LC43x의 I2C가 어떻게 동작하는지 살펴보자

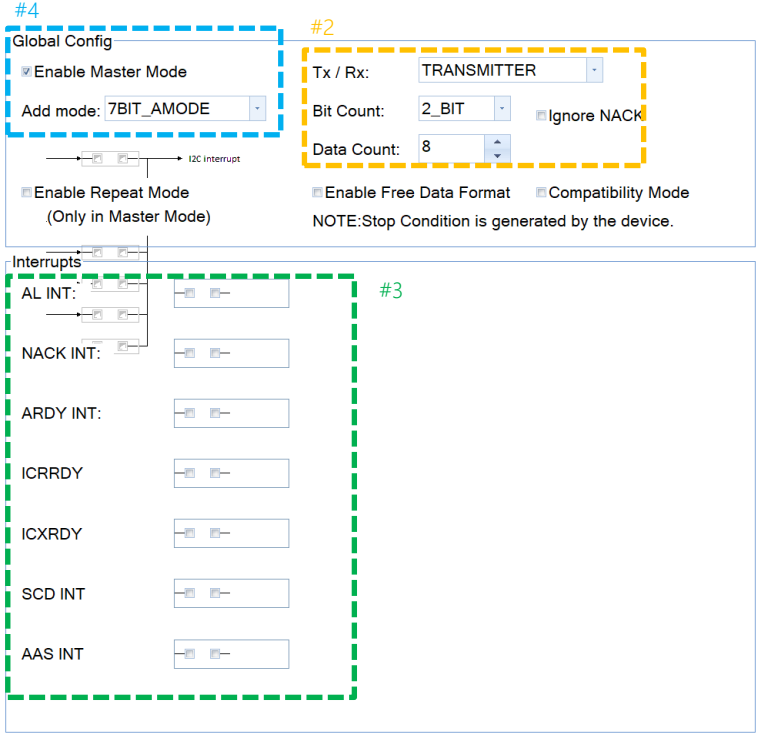
(8) I2C의 인터럽트 종류

Table 31-2. Interrupt Requests Generated by I2C Module

Flag	Name	Generated
AL	Arbitration-lost interrupt	Generated when the I2C module has lost an arbitration contest with another master-transmitter
NACK	No-acknowledge interrupt	Generated when the master I2C does not receive an acknowledge from the receiver
ARDY	Register-access-ready interrupt	Generated when the previously programmed address, data and command have been performed and the status bits have been updated. The interrupt is used to notify the device that the I2C registers are ready to be accessed.
RXRDY	Receive-data-ready interrupt	Generated when the received data in the receive-shift register (I2CSR) has been copied into the data receive register (I2CDRR). The RXRDY bit can also be polled by the device to determine when to read the received data in the I2CDRR.
TXRDY	Transmit-data-ready interrupt	Generated when the transmitted data has been copied from the data transmit register (I2CDXR) into the transmit-shift register (I2CXSR). The TXRDY bit can also be polled by the device to determine when to write the next data into I2CDXR.
SCD	Stop-condition-detect interrupt	Generated when a STOP condition has been detected.
AAS	Address-as-slave interrupt	Generated when the I2C has recognized its own slave address or an address of all zeroes.

Halcogen I2C 설정

→ I2C의 기본동작 설정 절차



#1

#1. VCLK1을 분주비를 통해 Baudrate를 만든다(10Kbps ~ 400Kbps)
#2. I2C의 전송/수신 모드와 data 길이를 설정한다
#3. I2C의 interrupt를 설정한다
#4. I2C의 마스터 활성화 및 address bit 모드를 설정한다

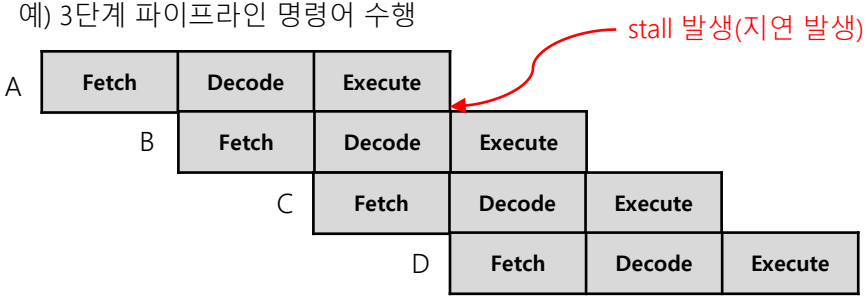
Memory barrier

(1) Memory barrier란?

→ 여러 개의 명령어 처리(multi-thread)를 처리 할 때 특정 명령어가 interrupt에 의해 수행되는 경우 interrupt enable이 먼저 수행 되어야 하나 instruction scheduling에 의해 명령어 수행 순서가 조정되면서 원하는 결과를 얻지 못하는 상황을 막기 위한 처리 상황을 memory barrier라 한다 이 memory barrier처리에 volatile 키워드가 사용된다

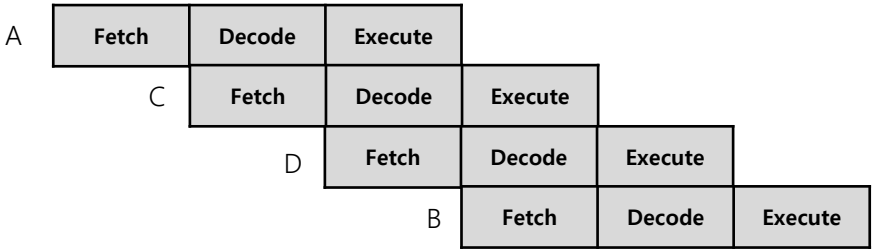
* Instruction schedulin이란?

예) 3단계 파이프라인 명령어 수행



A : AA = BB * CC
B : DD = AA + BB
C : EE = FF * GG
D : HH = BB + CC

→ 위 와 같이 A,B,C,D 4개의 명령어 수행동작 중 A,B 명령어는 AA에 의해 true dependency가 된다(B명령어처리를 위해 A의 결과가 필요함) 이 때 컴파일러는 CPU성능 최적화를 위해 서로 의존성이 없는 명령어인 C,D를 먼저 처리하게 명령어 순서를 재배치 한다



Memory barrier

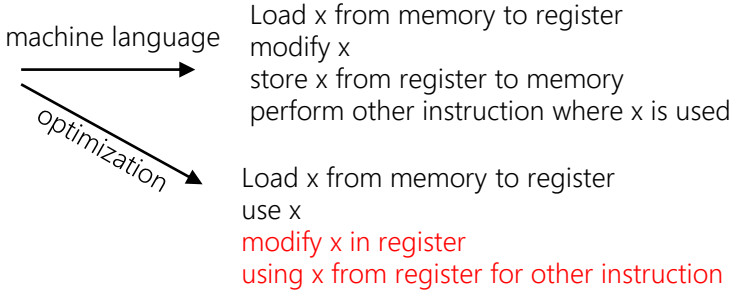
(2) volatile이란?

→ volatile 키워드는 volatile로 선언된 변수를 무조건 메모리에서 load/store 동작을 수행하게 한다

(3) volatile의 사용 예 – 컴파일러 최적화 방지

→ 예를 들어 아래와 같은 코드를 수행 한다고 하자

```
int x;
modify_value(x);
interrupt_service_routine()
{
  if(x == 0)
    return false;
}
```



→ 최적화 동작에 의해 변수의 변경이 레지스터에서 수행되고 다른 명령어 수행을 위해 레지스터로부터 변수 값을 읽어온다

→ 이 때 인터럽트에서 변수 값을 메모리에서 읽는다면 변경되지 않은 값으로 인해 오동작을 일으킨다