I²C PROTOCOL

Handong University

Jong-won Lee

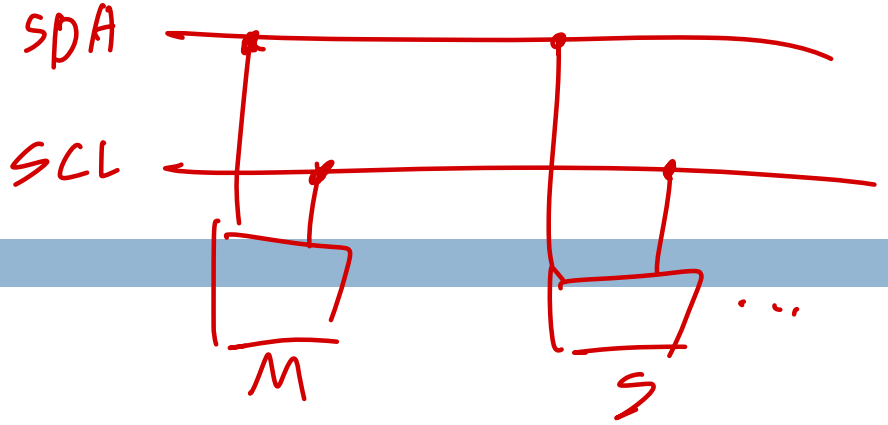
2020-04-18

I2C: Introduction

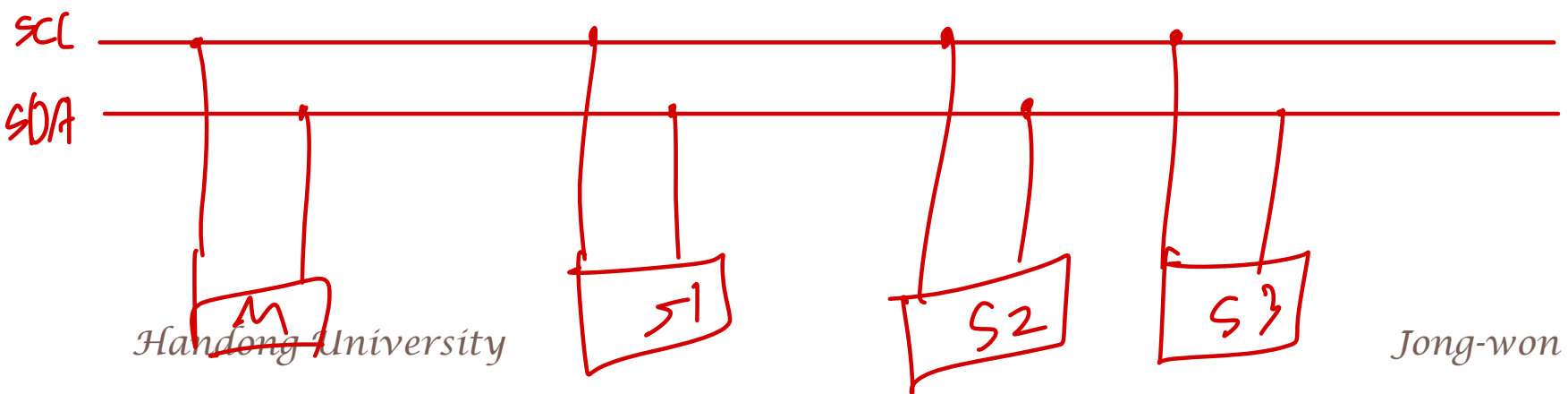
Introduction

- Originally, a bus defined by Philips
  - to provide a simple way to talk between IC's by using a minimum number of pins
  - I2C bus: inter-IC bus
- Features
  - Only two bus lines
    - SDA (a serial data) and SCL (a serial clock)
    - Serial 8-bit oriented bi-directional data transfer
  - Master-slave communication (by poll)
    - A master starts a communication.
    - A master supplies a clock on the SCL line.
  - Multi-master capable bus with arbitration feature

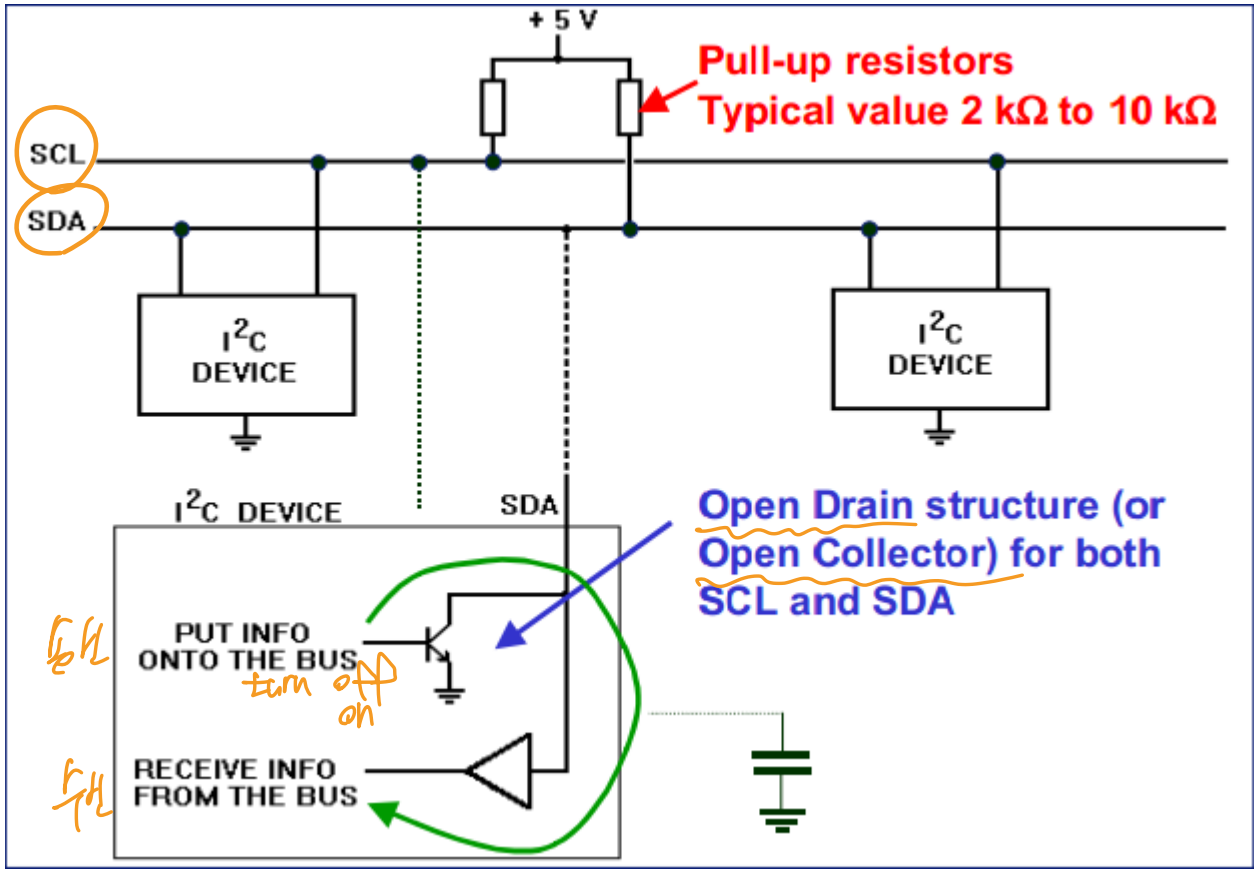
Introduction



- Features (con't)
  - Standard mode: 100 kbps, fast mode: 400 kbps, high-speed mode: 3.4 Mbps
  - Each IC on the bus is identified by its own address
  - The maximum bus capacity: 400 pF

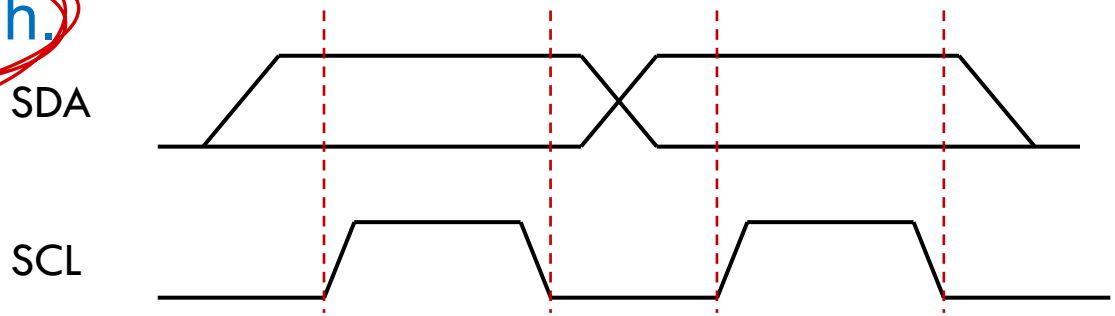


Bus Architecture

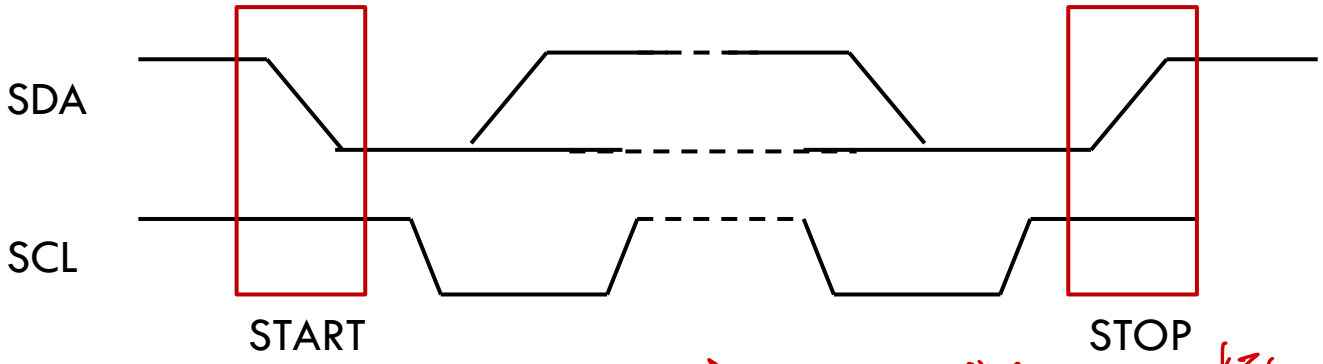


Basic Communication Format

- Data on SDA must be stable when SCL is high. That is, data is not allowed to be changed when the SCL is high.



- Exceptions are the START and STOP conditions

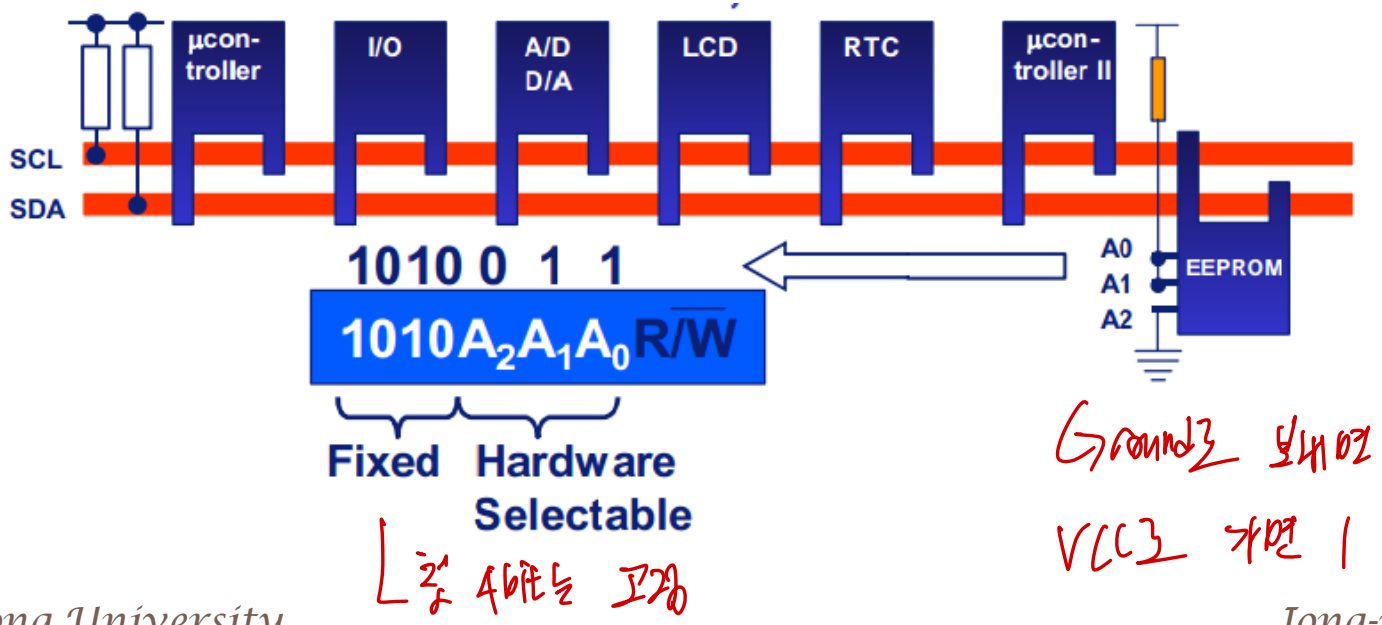


Basic Communication Format

- START condition
  - A HIGH to LOW transition on the SDA while the SCL is HIGH.
- STOP condition
  - A LOW to HIGH transition on the SDA while the SCL is HIGH.
- A master always generates START and STOP conditions.
- The bus is considered idle when SDA and SCL lines are HIGH.

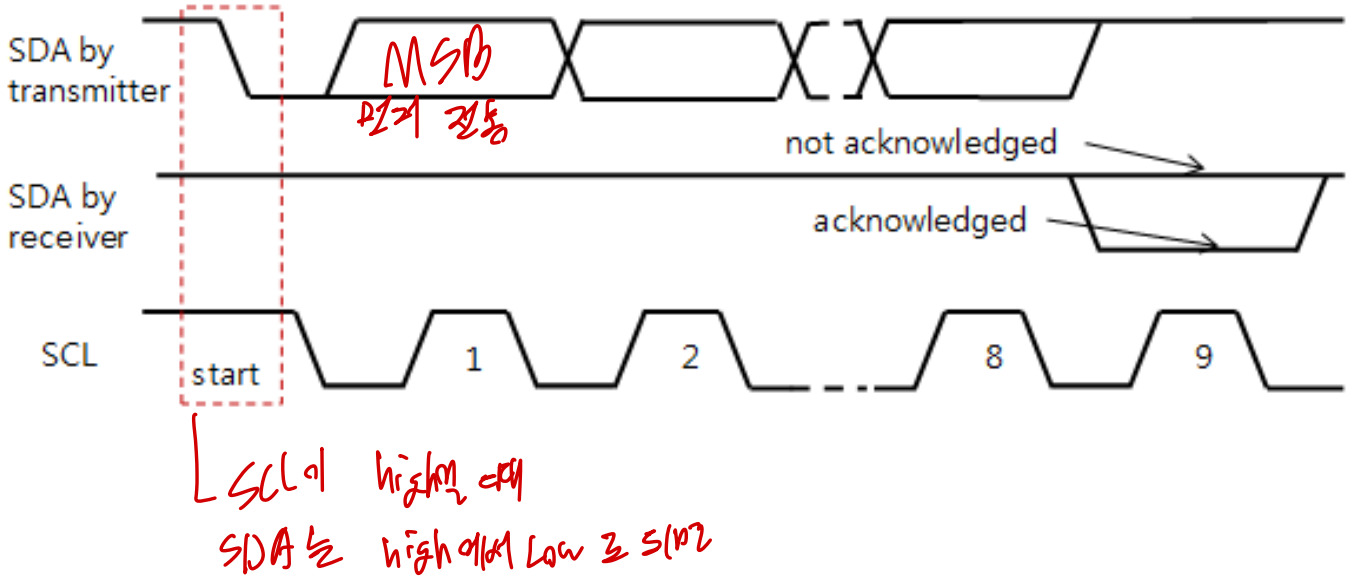
Address: Basic

- Uniqueness of address
  - Fully fixed or with a programmable part through hardware pins
  - In 7-bit address
    - Four most significant bits represent a group of ICs.



Bus Communication

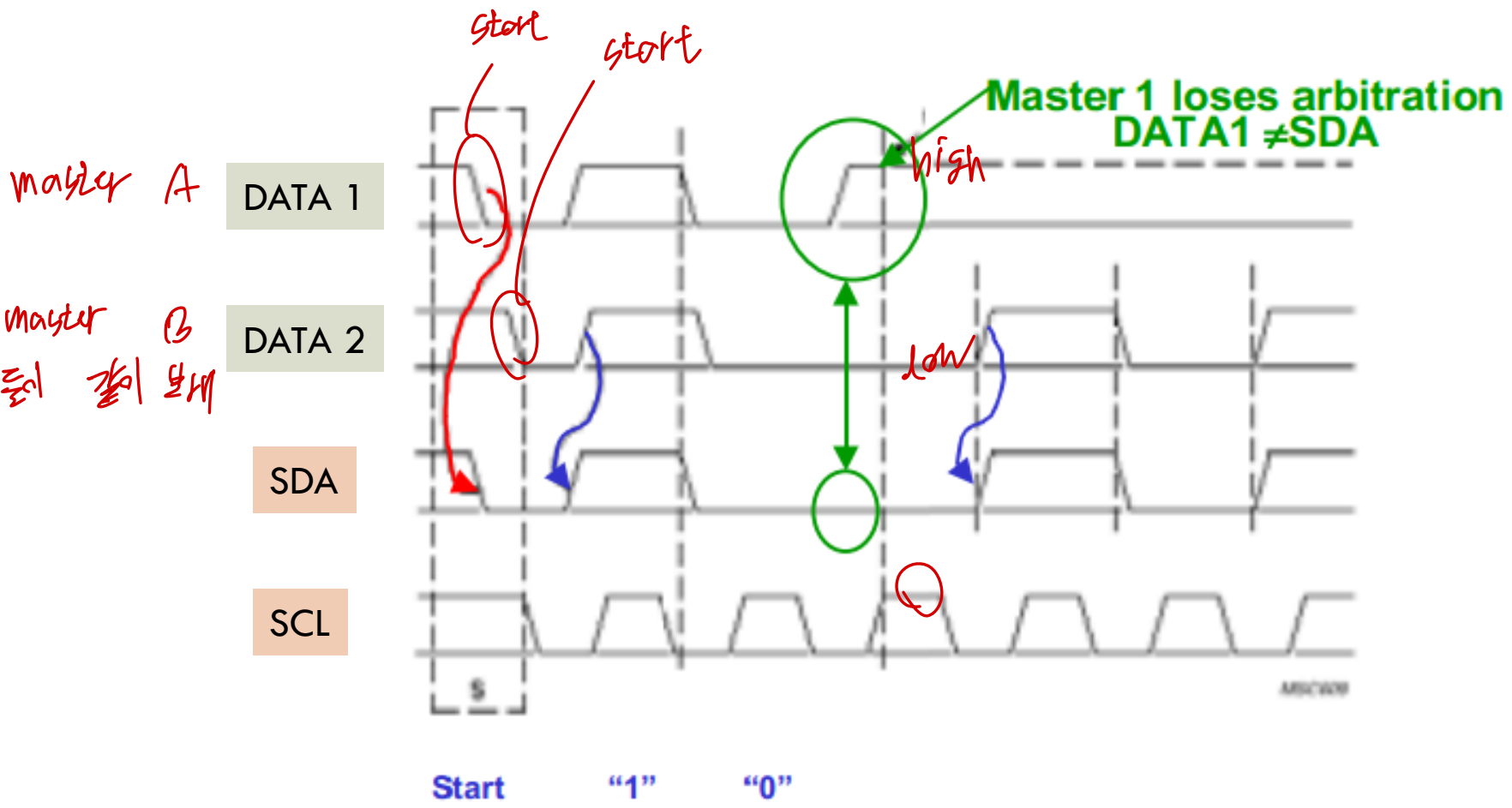
- Communication is established and 8-bit bytes are exchanged, each one being acknowledged using a 9th data bit generated by the receiving party.
  - The MSB of data is transferred first.



Bus Communication

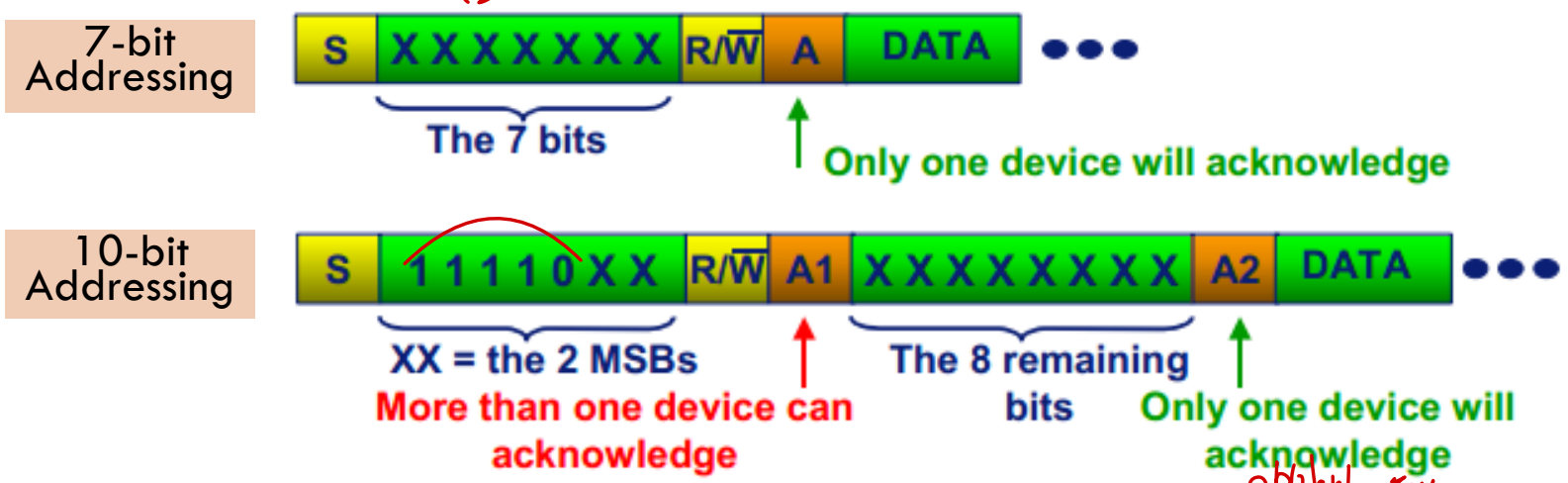
- Arbitration (경쟁)
  - The two masters can even generate a few cycles of the clock and data that 'match', but eventually one will output a 'low' when the other tries for a 'high'. The 'low' wins, so the 'loser' device withdraws.

low 가 더 세


start
start
Master 1 loses arbitration
DATA1 ≠SDA
high
low
master A
master B
DATA 1
DATA 2
SDA
SCL
S
Start
"1"
"0"

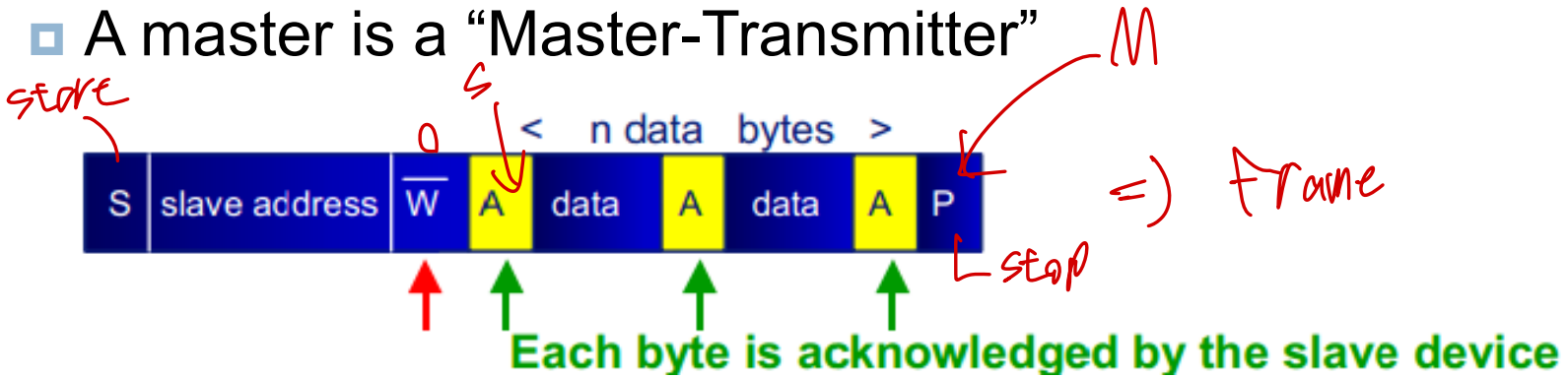
7-bit and 10-bit Address Formats

- The 1st byte after START determines the slave to be addressed.
  - "General call" address: all devices are addressed
    - 0000 000 + R/W
  - Some exceptions
    - 10-bit address: 1111 0XX + R/W

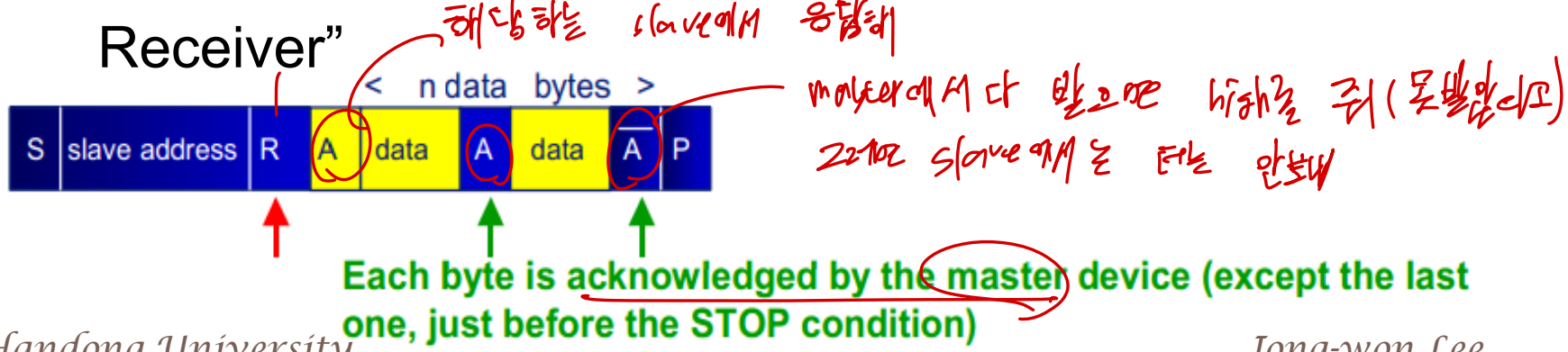


I2C Read and Write Operations

- Write to a slave device
  - A master is a “Master-Transmitter”

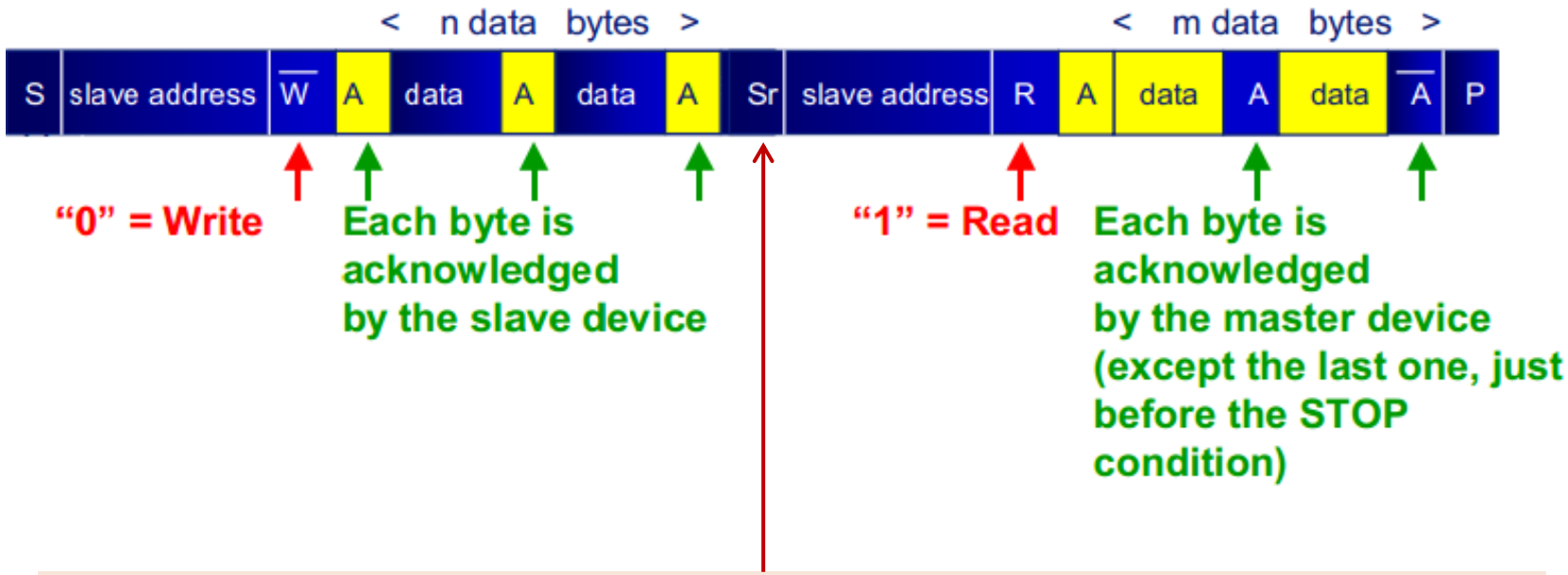


- Read from a slave device
  - A master is a “Master-Transmitter then Master-Receiver”



I2C Read and Write Operations

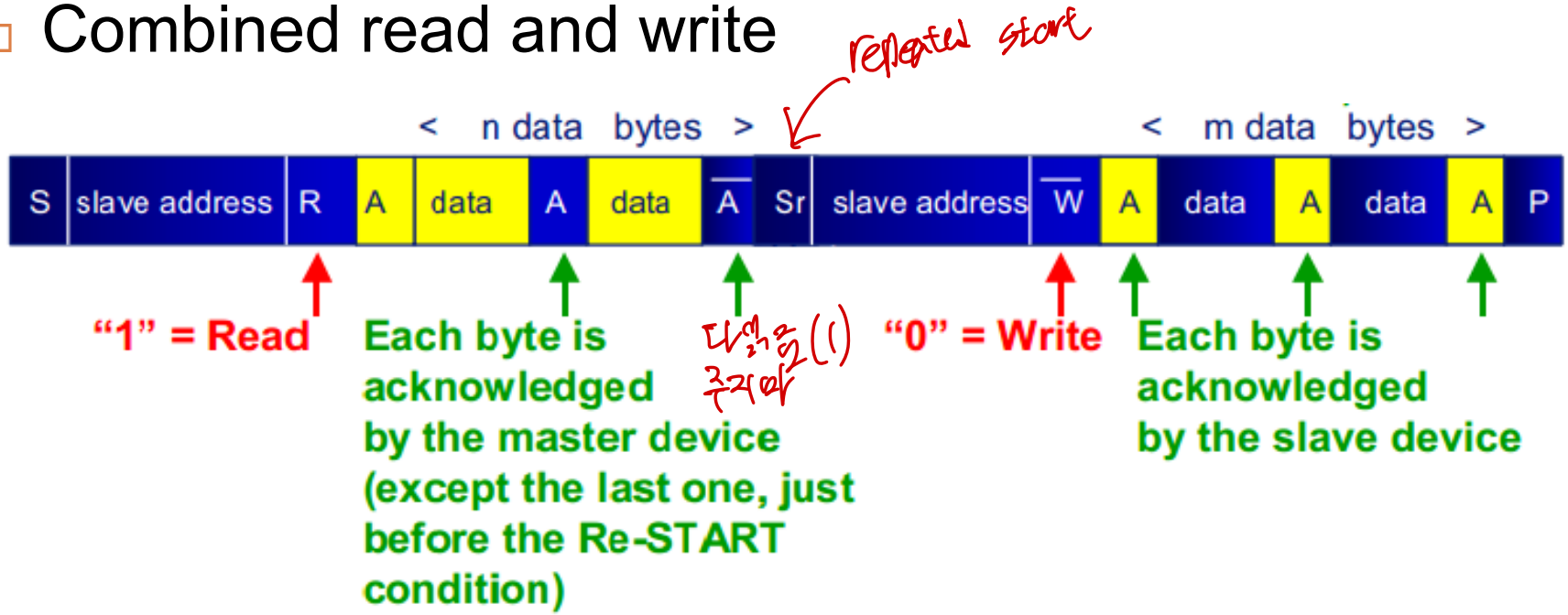
- Combined write and read



Repeated start: Instead of sending the stop condition
it is also allowed to send another start condition again followed by an address (and of course including a read/write bit) and more data.

I2C Read and Write Operations

- Combined read and write

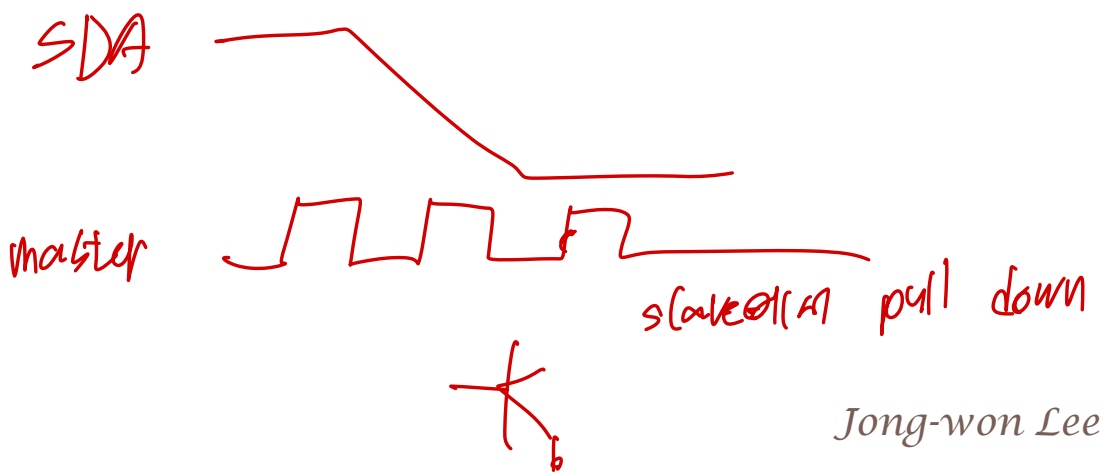


Acknowledge

- ACK
  - On the 9th clock pulse and mandatory
    - Transmitter releases the SDA line
    - Receiver pulls down the SDA line (while SCL is high)
  - Transfer is aborted if no ACK
    - When a slave receiver does not acknowledge the slave address, SDA must be left High by the slave so that the master can generate a STOP condition and abort the current transfer.
  - If the master device is acting as a receiver, it is responsible for acknowledging each transfer made by the slave.
    - Because the master controls the number of bytes in the transfer, it signals the end of data to the slave transmitter by not generating an acknowledge on the last data byte.

Acknowledge

- Clock stretching
  - Slave device can hold the clock line low when performing other functions
  - Master can slow down the clock to accommodate slow device.



HTU21D
HUMIDITY AND TEMPERATURE SENSOR

Handong university

Jong-won Lee

2021.05.08

HTU21D

HTU21D

- Features
  - Interface: I2C (max. 400 kHz)
    - 7 bit address: 0x40H
  - Relative Humidity operating range: 0 ~ 100%
  - Relative Humidity resolution: 12/8/10/11 bits
  - Typical humidity accuracy of ±2%
  - Temperature operating range: -40 ~ 125 °C
  - Temperature resolution: 14/12/13/11 bits
  - Typical temperature accuracy of ±0.3 °C
  - Power supply: 1.5 – 3.6V

HTU21D

- Hold and No Hold Master modes
  - Hold master mode
    - HTU21D blocks SCK line during measurement process.
    - HTU21D pulls down the SCK line while measuring
      - During measurement process, impossible to use I2C bus for other communications.
  - No Hold master mode
    - Allows for processing other I²C communication tasks on a bus while the HTU21D sensor is measuring
    - Has to poll to check the completion of the internal processing of the HTU21D sensor.
      - Send no ACK for its own slave address
      - If the internal processing is finished, the HTU21D sensor acknowledges the poll of the MCU and data can be read by the MCU.

HTU21D

- Commands

Command	Code	Comment
Trigger Temperature Measurement	0xE3	Hold master
Trigger Humidity Measurement	0xE5	Hold master
Trigger Temperature Measurement	0xF3	No Hold master
Trigger Humidity Measurement	0xF5	No Hold master
Write user register	0xE6	
Read user register	0xE7	
Soft Reset	0xFE	

  - Soft reset command: take less than 15 ms

HTU21D

- Commands
  - Temperature measuring time

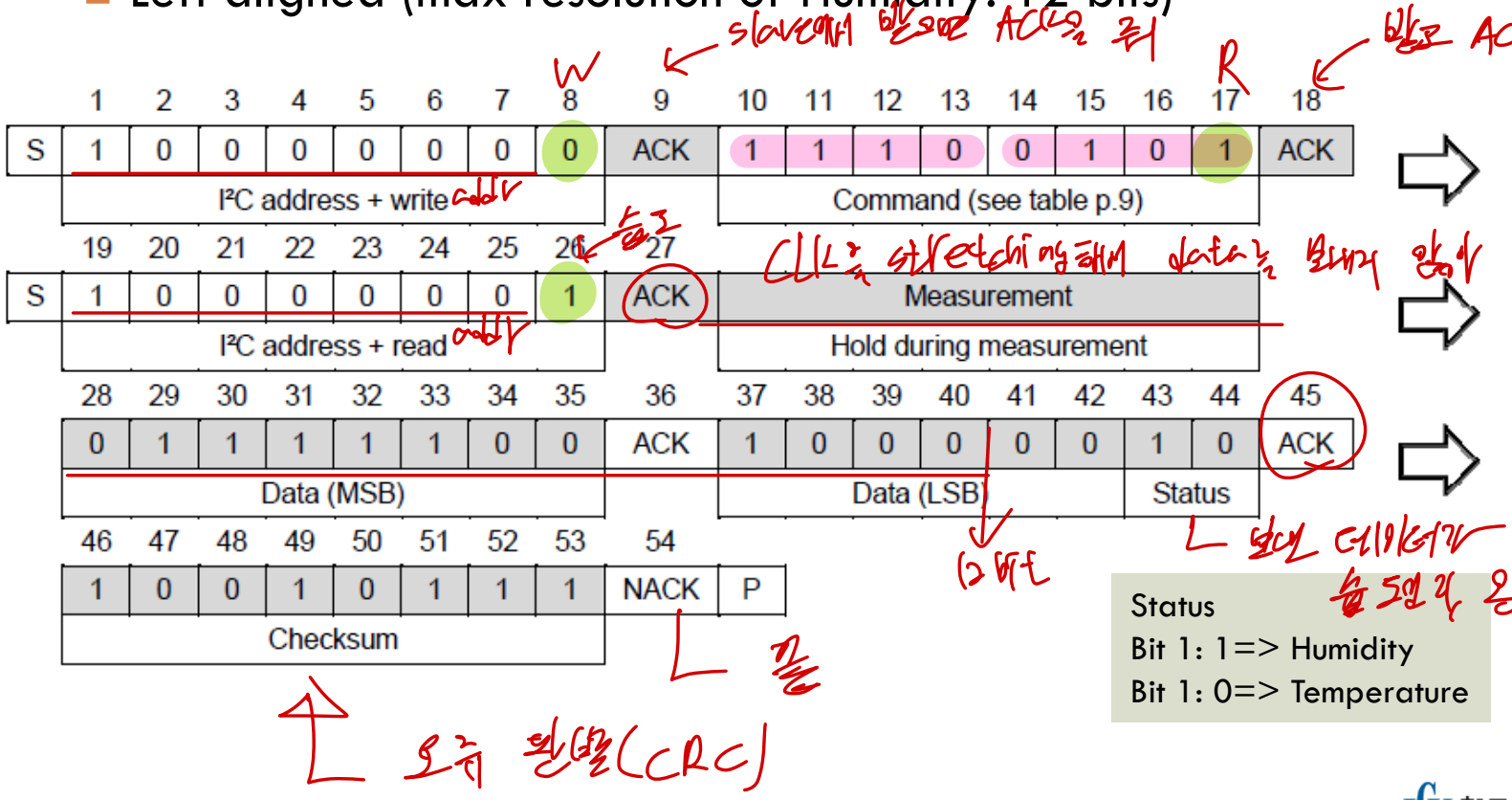
Resolution	Typ [ms]	Max [ms]
14 bits	44	50
13 bits	22	25
12 bits	11	13
11 bits	6	7

  - Relative Humidity measuring time

Resolution	Typ [ms]	Max [ms]
12 bits	14	16
11 bits	7	8
10 bits	4	5
8 bits	2	3

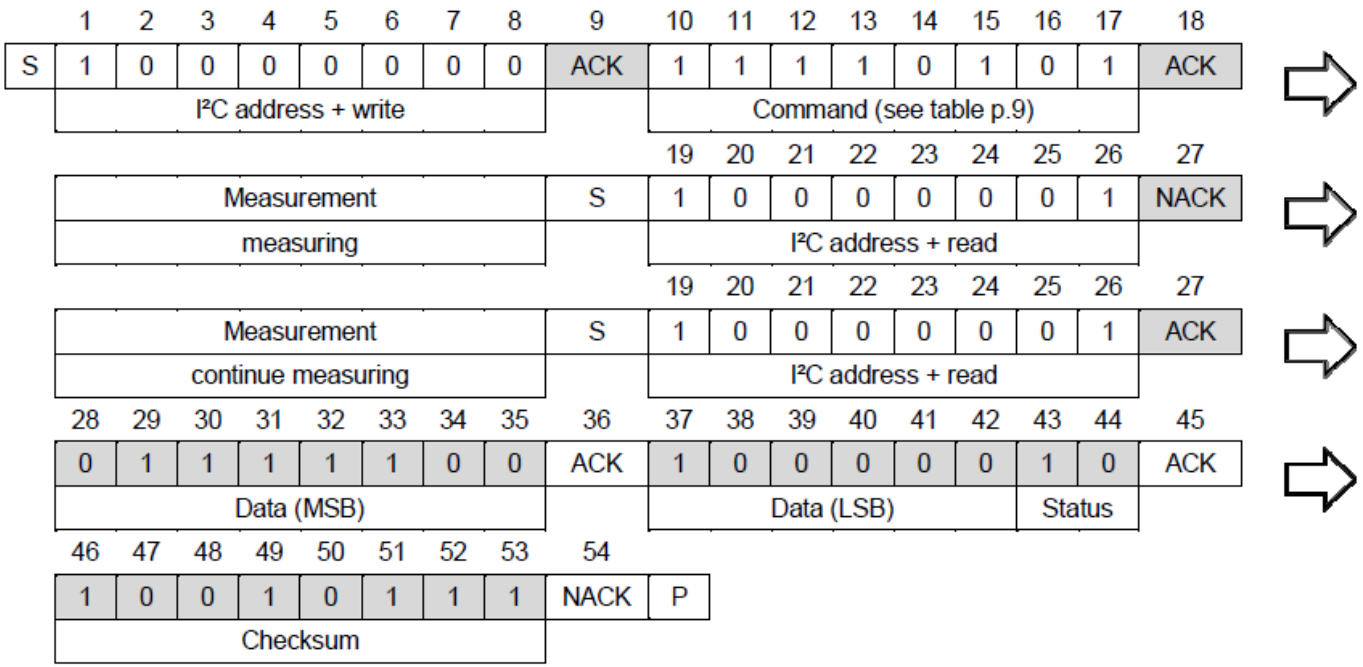
HTU21D

- Ex.: Hold Master communication sequence
  - Hold master Trigger Humidity measurement (0xE5)
    - Left aligned (max resolution of Humidity: 12 bits)



HTU21D

- Ex.: No Hold Master communication sequence
  - No Hold master Trigger Humidity measurement (0xF5)
    - Wrong example… (max resolution of Humidity: 12 bits)



HTU21D

- User register

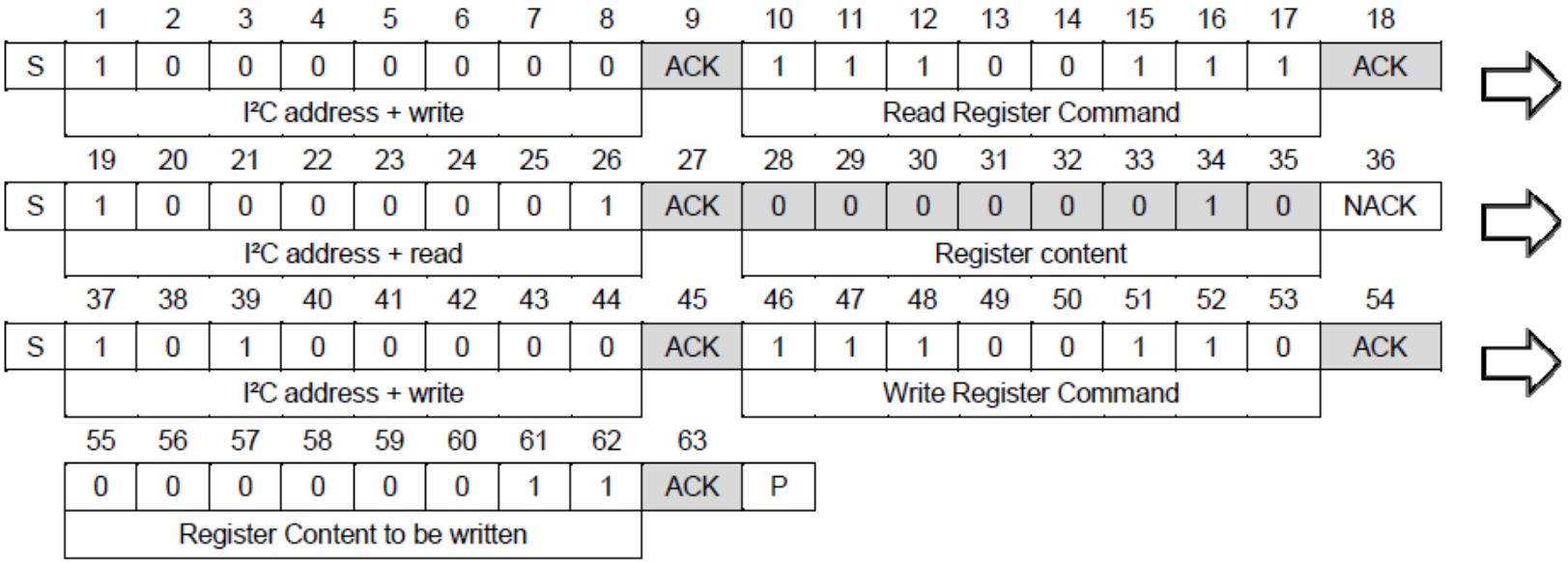
Bit	#Bits	Description/Coding	Default
7,0	2	Measurement resolution Bit 7 / Bit 0 / RH / Temp 0 / 0 / 12 bits / 14 bits 0 / 1 / 8 bits / 12 bits 1 / 0 / 10 bits / 13 bits 1 / 1 / 11 bits / 11 bits	'00'
6	1	Status: End of Battery(1) '0': VDD>2.25V '1': VDD<2.25V	'0'
3, 4, 5	3	Reserved	'0'
2	1	Enable on-chip heater	'0'
1	1	Disable OTP reload	'1'

(1) This status bit is updated after each measurement

- The heater is intended to be used **for functionality diagnosis**:
  - Relative humidity drops upon rising temperature.
  - The heater consumes about 5.5mW and provides a temperature increase of about 0.5-1.5°C.
- OTP reload is a safety feature and load the entire OTP settings to the register, with the exception of the heater bit, before every measurement
  - This feature is disabled per default and it is not recommended for use.
  - Use **soft reset** instead as it contains OTP reload.

HTU21D

- User register (read/write command: 0xE7/0xE6)
  - Example:



HTU21D

- CRC checksum
  - Polynomial: $X^8 + X^5 + X^4 + 1$.
    - Initialization: 0x00
  - Protected data: read data
- Conversion formula
  - Regardless to resolution, data (S_{SH}, S_{Temp}) is left-aligned 16 bit data.
  - Relative humidity

$$RH = -6 + 125 \times \frac{S_{RH}}{2^{16}}$$

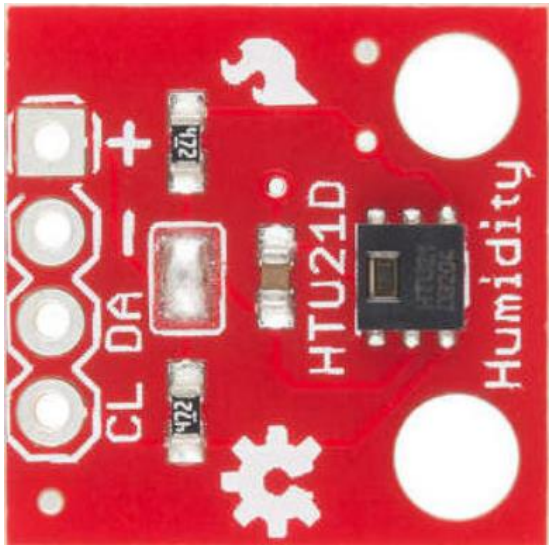
  - Temperature (°C)

$$Temp = -46.85 + 175.72 \times \frac{S_{Temp}}{2^{16}}$$

HTU21D Module

- Has pull-up resistors (4.7 k) on SCL and SDA, but the resistors are not connected to 3.3V.

3.3V
GND
SDA
SCL



I2C IN MBED-OS

Handong university

Jong-won Lee

2021.05.08

I2C in STM32F411RE

I2C in Nucleo-F411RE

- 3 I2C in STM32F411
  - I2C1, I2C2, I2C3
    - Support standard (100 kbps) and fast (400 kbps) modes
      - Up to 1Mbps(fast mode plus)
    - Support the 7/10-bit addressing mode

I2C in Nucleo-F411RE

- I2C in Mbed-OS

I2C 신호	핀 번호	외부 확장 커넥터	I2C 신호	핀 번호	외부 확장 커넥터
I2C1_SDA	**PB_9**, PB_7,	D14, CN7-21	I2C1_SCL	**PB_8**, PB_6	D15, D10
I2C2_SDA	PB_3 **PB_9_ALT0**	D3 (SWO) D14	I2C2_SCL	**PB_10**	D6
I2C3_SDA	PB_4, PB_8, PC_9	D5, D15, CN10-1	~~I2C3_SCL~~	~~PA_8~~	~~D7~~

Mbed I2C API

I2C class

생성자	I2C (PinName sda, PinName scl)
함수	void frequency (int hz)
동작	시리얼 통신에 사용할 SCL 클럭의 속도를 hz 파라미터로 설정한다. 디폴트 속도는 100 kHz이다..
함수	int read (int address, char *data, int length, bool repeated=false)
동작	인자: address[7:1] = 슬레이브 7비트 주소, address[0] = 1, data: 읽은 데이터 저장할 버퍼 포인터, length: 읽을 데이터 개수 repeated: true 이면 “repeated start”로 동작함. 즉 모두 읽은 후 프레임을 끝내지 않고, 다시 시작함. 즉 스톱 심볼 생성하지 않고 스타트 심볼 다시 생성함. 동작: 주어진 슬레이브로부터 데이터 읽음. 반환 인자: ‘0’이면 성공, 아니면 실패.

Mbed I2C API

- I2C class

함수	int write (int address, char *data, int length, bool repeated=false)
동작	인자: address[7:1] = 슬레이브 7비트 주소, address[0] = 0, data: 보낼 데이터 저장한 버퍼 포인터, length: 보낼 데이터 개수 repeated: true 이면 “repeated start”로 동작함. 즉 모두 보낸 후 프레임을 끝내지 않고, 다시 시작함. 즉 스톱 심볼 생성하지 않고 스타트 심볼 다시 생성함. 동작: 주어진 슬레이브로 데이터를 전송함. 반환 인자: ‘0’이면 성공, 아니면 실패.

Mbed I2C API

I2C class

함수	int read(int ack)
동작	인자: ack=1 이면, 데이터 읽은 후에 ACK 보냄. 동작: 한 바이트 읽음. 반환 인자: 읽은 데이터
함수	int write (int data)
동작	인자: data: 전송할 데이터. 동작: 한 바이트 데이터를 전송함. 반환 인자: '0'이면 NAK 수신, '1'이면 ACK 수신, '2'이면 타임아웃.
함수	void start(void)
동작	Creates a start condition on the I2C bus
함수	void stop(void)
동작	Creates a stop condition on the I2C bus

Mbed I2C API

- I2C class

함수	void lock (void),
동작	Acquire exclusive access to this I2C bus
함수	void unlock (void)
동작	Release exclusive access to this I2C bus.

Non-blocking API

Non-blocking Function

- I2C class

int	transfer (int address, const char *tx_buffer, int tx_length, char *rx_buffer, int rx_length, const event_callback_t &callback, int event=I2C_EVENT_TRANSFER_COMPLETE, bool repeated=false);
동작 인자 반환 값	비동기식 방식으로 데이터를 전송하거나/읽는다. address[7:1] = 슬레이브 7비트 주소, address[0] = 0, tx_buffer: 전송할 데이터를 저장한 버퍼 포인터. tx_length: 전송할 데이터 바이트 수. rx_buffer: 읽은 데이터 저장할 버퍼 포인터, rx_length: 읽을 데이터 바이트 수 callback: 이벤트가 발생하였을 때, 불리는 콜백 함수이다. event: 콜백 함수를 부르는 이벤트의 종류. (I2C_EVENET_ERROR, I2C_EVENT_ERROR_NO_LAVE, I2C_TRANSFER_COMPLETE, I2C_EVENT_TRANSFER_EARLY_NACK, I2C_EVENT_ALL) repeated: true 이면 "repeated start"로 동작함. 즉 모두 읽은 후 프레임을 끝내지 않고, 다시 시작함. 즉 스톱 심볼 생성하지 않고 스타트 심볼 다시 생성함. Return value: '0'이면 성공, I2C 주변기기가 다른 일을 하고 있으면 -1.
void	abort_transfer ( );
동작	실행 중인 I2C 동작을 중단 시킴.

Labs

Lab11_1

- 실습 목적
  - I2C 통신 방식을 이해한다.
  - I2C 인터페이스를 지닌 온습도 센서의 동작을 이해한다.
  - I2C 통신 방식을 이용하여 데이터를 전송하고, 데이터를 읽을 수 있다.

- 실습 시나리오.
  - HTU21D로 부터 온도와 습도를 3초 간격으로 읽은 후, 해당 값을 터미널에 표시한다.
    - 소수점 2째 자리까지 출력.
  - Library를 사용하지 말고, mbed-os의 I2C class를 이용하여 프로그램을 작성하시오.

Lab11_1

- 회로 구성

Connections for I2C		
HTU21D	**←→**	**Nucleo**
CL (SCL)	←→	SCL/D15
DA (SDA)	←→	SDA/D14
- (GND)	←→	GND
+ (3.3V)	←→	3.3V

  - 모듈 내 풀업(Pull-up) 저항이 3.3V에 연결되어 있지 않아 SCL, SDA line에 10kΩ(혹은 4.7kΩ) 저항을 풀업 저항으로 달아야 한다.

Lab11_1

- A sample code (part)

```
#include "mbed.h"

#define HTU21D_TEMP_READ (0xF3)  // no hold command
#define HTU21D_HUMI_READ (0xF5)
#define HTU21G_USER_WRITE (0xE6)
#define HTU21D_ADDR      (0x40)

//I2C i2c(PB_9_ALT0, PB_10);   //D14, D6(PB_10)
I2C i2c(PB_9, PB_8);                //D14, D15 (PB_8)
BufferedSerial pc(CONSOLE_TX, CONSOLE_RX, 115200);
```

Lab11_1

- A sample result

```
I2C Test Program for HTU21D Sensor
Temperature = 25.28 [C]
Relative Humidity = 29.78 [%]
Temperature = 25.28 [C]
Relative Humidity = 29.74 [%]
Temperature = 29.06 [C]
Relative Humidity = 45.11 [%]
Temperature = 30.04 [C]
Relative Humidity = 55.21 [%]
Temperature = 30.50 [C]
Relative Humidity = 58.72 [%]
Temperature = 28.62 [C]
Relative Humidity = 28.49 [%]
Temperature = 27.98 [C]
Relative Humidity = 25.52 [%]
Temperature = 27.58 [C]
Relative Humidity = 25.68 [%]
```

Lab11_2

- 실습 시나리오.
  - HUT21D 관련 적당한 library를 import한 다음, 이를 이용하여 온도와 습도를 3초 간격으로 측정하여, 이들을 터미널에 표시하시오.
  - 온도와 습도를 읽을 때에 CRC error 체크를 한 다음에 CRC error가 없는 경우에만 온/습도를 출력하고, CRC error가 있는 경우에는 "CRC error" 메시지를 출력하시오. (필요하면 library를 수정하십시오.)
    - 직접 CRC error check routine을 작성하여도 되고, mbed-os의 MbedCRC class를 이용하여도 된다.

I2C IN MBED-OS

Handong university

Jong-won Lee

2023.05.09

I2C in STM32F411RE

I2C in Nucleo-F411RE

- 3 I2C in STM32F411
  - I2C1, I2C2, I2C3
    - Support standard (100 kbps) and fast (400 kbps) modes
      - Up to 1Mbps(fast mode plus)
    - Support the 7/10-bit addressing mode

I2C in Nucleo-F411RE

- I2C in Mbed-OS

I2C 신호	핀 번호	외부 확장 커넥터	I2C 신호	핀 번호	외부 확장 커넥터
I2C1_SDA	**PB_9**, PB_7,	D14, CN7-21	I2C1_SCL	**PB_8**, PB_6	D15, D10
I2C2_SDA	PB_3 **PB_9_ALT0**	D3 (SWO) D14	I2C2_SCL	**PB_10**	D6
I2C3_SDA	PB_4, PB_8, PC_9	D5, D15, CN10-1	~~I2C3_SCL~~	~~PA_8~~	~~D7~~

Mbed I2C API

I2C class

생성자	I2C (PinName sda, PinName scl)
함수	void frequency (int hz)
동작	시리얼 통신에 사용할 SCL 클럭의 속도를 hz 파라미터로 설정한다. 디폴트 속도는 100 kHz이다..
함수	int read (int address, char *data, int length, bool repeated=false)
동작	인자: address[7:1] = 슬레이브 7비트 주소, address[0] = 1, data: 읽은 데이터 저장할 버퍼 포인터, length: 읽을 데이터 개수 repeated: true 이면 “repeated start”로 동작함. 즉 모두 읽은 후 프레임을 끝내지 않고, 다시 시작함. 즉 스톱 심볼 생성하지 않고 스타트 심볼 다시 생성함. 동작: 주어진 슬레이브로부터 데이터 읽음. 반환 인자: ‘0’이면 성공, 아니면 실패.

Mbed I2C API

- I2C class

함수	int write (int address, char *data, int length, bool repeated=false)
동작	인자: address[7:1] = 슬레이브 7비트 주소, address[0] = 0, data: 보낼 데이터 저장한 버퍼 포인터, length: 보낼 데이터 개수 repeated: true 이면 “repeated start”로 동작함. 즉 모두 보낸 후 프레임을 끝내지 않고, 다시 시작함. 즉 스톱 심볼 생성하지 않고 스타트 심볼 다시 생성함. 동작: 주어진 슬레이브로 데이터를 전송함. 반환 인자: ‘0’이면 성공, 아니면 실패.

Mbed I2C API

I2C class

함수	int read(int ack)
동작	인자: ack=1 이면, 데이터 읽은 후에 ACK 보냄. 동작: 한 바이트 읽음. 반환 인자: 읽은 데이터
함수	int write (int data)
동작	인자: data: 전송할 데이터. 동작: 한 바이트 데이터를 전송함. 반환 인자: '0'이면 NAK 수신, '1'이면 ACK 수신, '2'이면 타임아웃.
함수	void start(void)
동작	Creates a start condition on the I2C bus
함수	void stop(void)
동작	Creates a stop condition on the I2C bus

Mbed I2C API

- I2C class

함수	void lock (void),
동작	Acquire exclusive access to this I2C bus
함수	void unlock (void)
동작	Release exclusive access to this I2C bus.

Non-blocking API

Non-blocking Function

- I2C class

int	transfer (int address, const char *tx_buffer, int tx_length, char *rx_buffer, int rx_length, const event_callback_t &callback, int event=I2C_EVENT_TRANSFER_COMPLETE, bool repeated=false);
동작 인자 반환 값	비동기식 방식으로 데이터를 전송하거나/읽는다. address[7:1] = 슬레이브 7비트 주소, address[0] = 0, tx_buffer: 전송할 데이터를 저장한 버퍼 포인터. tx_length: 전송할 데이터 바이트 수. rx_buffer: 읽은 데이터 저장할 버퍼 포인터, rx_length: 읽을 데이터 바이트 수 callback: 이벤트가 발생하였을 때, 불리는 콜백 함수이다. event: 콜백 함수를 부르는 이벤트의 종류. (I2C_EVENET_ERROR, I2C_EVENT_ERROR_NO_LAVE, I2C_TRANSFER_COMPLETE, I2C_EVENT_TRANSFER_EARLY_NACK, I2C_EVENT_ALL) repeated: true 이면 "repeated start"로 동작함. 즉 모두 읽은 후 프레임을 끝내지 않고, 다시 시작함. 즉 스톱 심볼 생성하지 않고 스타트 심볼 다시 생성함. Return value: '0'이면 성공, I2C 주변기기가 다른 일을 하고 있으면 -1.
void	abort_transfer ( );
동작	실행 중인 I2C 동작을 중단 시킴.

Labs

Lab11-1

- 실습 목적
  - I2C 통신 방식을 이해한다.
  - I2C 인터페이스를 지닌 온습도 센서의 동작을 이해한다.
  - I2C 통신 방식을 이용하여 데이터를 전송하고, 데이터를 읽을 수 있다.

- 실습 시나리오.
  - HTU21D로 부터 온도와 습도를 3초 간격으로 읽은 후, 해당 값을 터미널에 표시한다.
    - 소수점 2째 자리까지 출력.
  - Library를 사용하지 말고, mbed-os의 I2C class를 이용하여 프로그램을 작성하시오.
  - 온도와 습도를 읽을 때에 CRC error 체크를 한 다음에 CRC error가 없는 경우에만 온/습도를 출력하고, CRC error가 있는 경우에는 “CRC error” 메시지를 출력하시오. (필요하면 library를 수정하십시오.)

Lab11-1

- 회로 구성

Connections for I2C		
HTU21D	**←→**	**Nucleo**
CL (SCL)	←→	SCL/D15
DA (SDA)	←→	SDA/D14
- (GND)	←→	GND
+ (3.3V)	←→	3.3V

  - 모듈 내 풀업(Pull-up) 저항이 3.3V에 연결되어 있지 않아 SCL, SDA line에 10kΩ(혹은 4.7kΩ) 저항을 풀업 저항으로 달아야 한다.

Lab11-1

- A sample code (part)

```
#include "mbed.h"

#define HTU21D_TEMP_READ (0xF3)  // no hold command
#define HTU21D_HUMI_READ (0xF5)
#define HTU21G_USER_WRITE (0xE6)
#define HTU21D_ADDR      (0x40)

//I2C i2c(PB_9_ALT0, PB_10);   //D14, D6(PB_10)
I2C i2c(PB_9, PB_8);                //D14, D15 (PB_8)
BufferedSerial pc(CONSOLE_TX, CONSOLE_RX, 115200);
```

Lab11-1

- A sample result

```
I2C Test Program for HTU21D Sensor
Temperature = 25.28 [C]
Relative Humidity = 29.78 [%]
Temperature = 25.28 [C]
Relative Humidity = 29.74 [%]
Temperature = 29.06 [C]
Relative Humidity = 45.11 [%]
Temperature = 30.04 [C]
Relative Humidity = 55.21 [%]
Temperature = 30.50 [C]
Relative Humidity = 58.72 [%]
Temperature = 28.62 [C]
Relative Humidity = 28.49 [%]
Temperature = 27.98 [C]
Relative Humidity = 25.52 [%]
Temperature = 27.58 [C]
Relative Humidity = 25.68 [%]
```