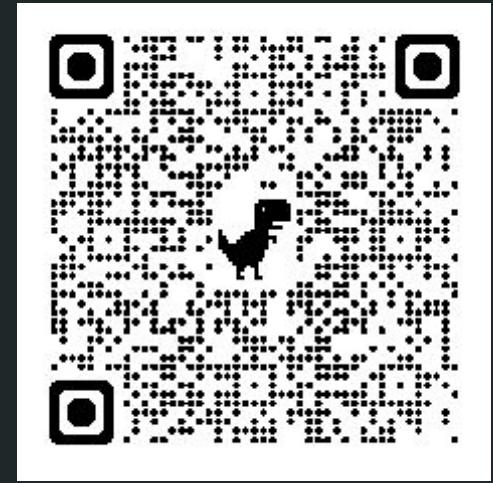


강의자료 QR코드



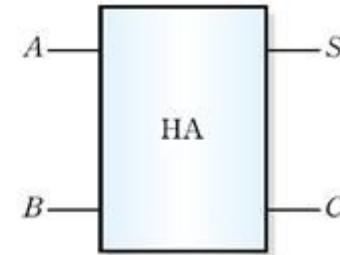
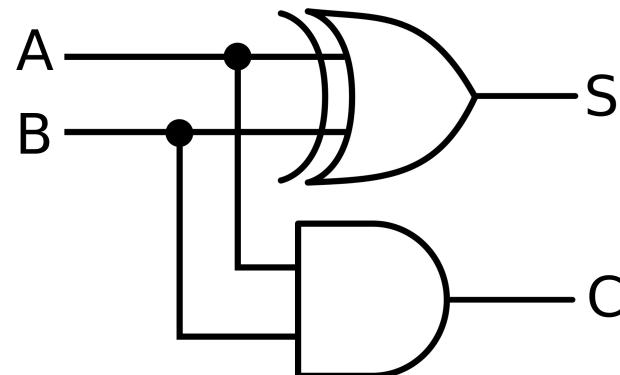
STM32

TechAbyss - Steve Kim

반가산기

➤ Half Adder

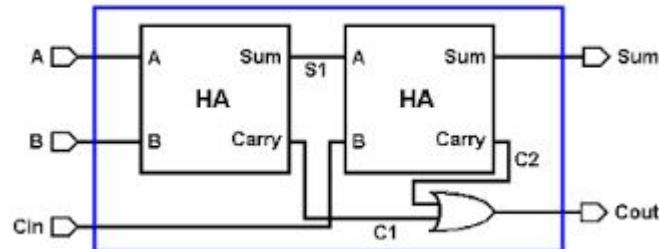
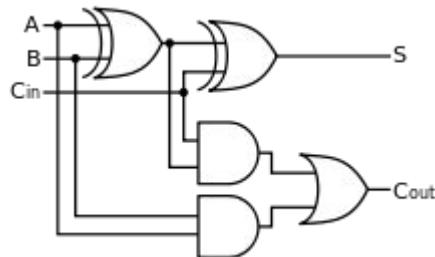
X	Y	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



전가산기

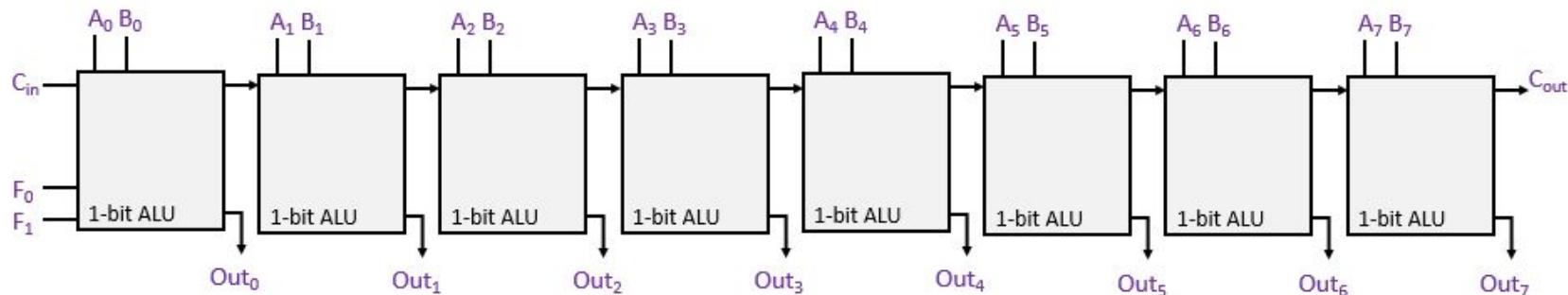
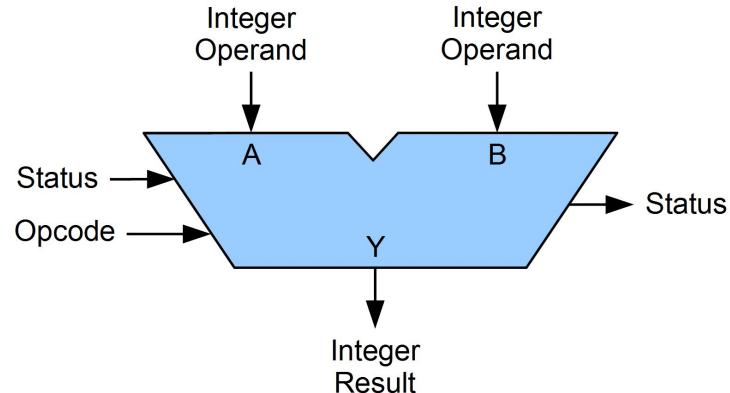
➤ Full Adder

X	Y	Z	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



ALU

- Arithmetic and Logical Unit



연산

5+3

$$0101 + 0011 = 1000$$

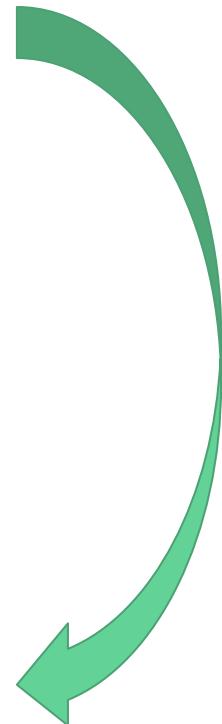
$$5-3 = 5 + (-3)$$

3의 2의 보수 : 1101

$$0101 + 1101 = 0010$$

원형의 코드 테이블에서 3칸 뒤로 후퇴한 상태를 더한다

7	:	0111
6	:	0110
5	:	0101
4	:	0100
3	:	0011
2	:	0010
1	:	0001
0	:	0000
-1	:	1111
-2	:	1110
-3	:	1101
-4	:	1100
-5	:	1011
-6	:	1010
-7	:	1001
-8	:	1000



Processor

- CPU : Central Processing Unit 중앙 처리 장치
 - ALU : 연산
 - 제어부
- MPU : Micro Processing Unit 소형 처리 장치
 - CPU에 주변장치가 포함됨
 - RAM : Random Access Memory : 휘발성 : 데이터
 - ROM : Read Only Memory : 비휘발성 : 프로그램 코드
- GPU : Graphic Processing Unit 그래픽 처리 장치
 - 그래픽카드의 3차원 연산을 담당
 - 50개 이상의 ALU를 가짐
 - nVidia CUDA

Memory

RAM

SRAM : Static RAM

DRAM : Dynamic RAM

DDRAM, FRAM

ROM

PROM : Programmable ROM

EPROM : Erase Programmable ROM

EEPROM : Electronic Erase Programmable ROM

Flash

Code, Data

Code

작성된 프로그램 코드

Data

프로그램 동작 과정에서 생성되는 데이터

CISC vs RISC

➤ CISC (Complex Instruction Set Computer)

- 복잡하고 많은 종류의 명령어와 주소 지정 모드를 사용
- 가변 길이 명령어 형식
- 100~250개의 명령어
- Intel사

➤ RISC (Reduced Instruction Set Computer)

- 간단하고 적은 종류의 명령어와 주소 지정 모드를 사용
- 고정 길이 명령어 형식
- 100개 이하의 명령어
- ARM사

Example:

Multiply value in memory location "X" by value in memory location "Y"; store result back into location "X". Registers "A" and "B" are available.

CISC Assembly:

IMUL X, Y

RISC Assembly:

LOAD A, X

LOAD B, Y

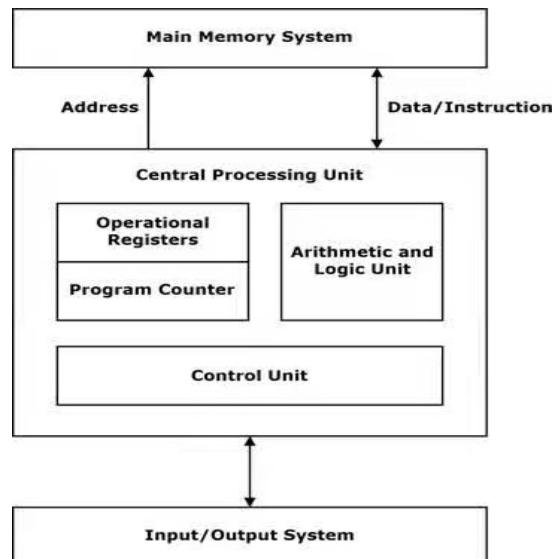
PROD A, B

STORE X, A

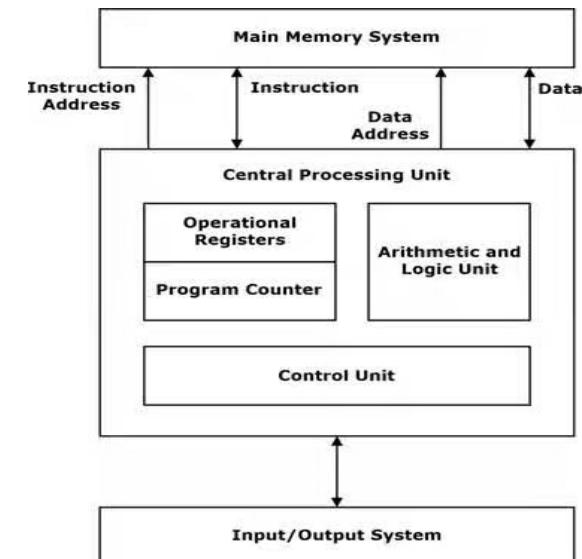
CPU architecture comparison

Von Neumann architecture

Harvard architecture



Von Neumann Architecture



Harvard Architecture

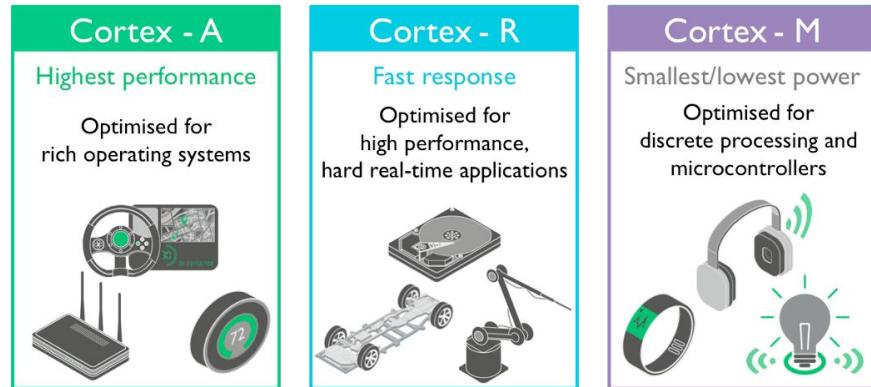
CPU

- Z-80 (Zilog), 6502 (MOS)
- 8088, 8086, 8087
- 80286, 80287
- 80386 (32bit), 80387
- 80486 with co-processor
- 80586... Pentium (64bit)
- i3, i5, i7
- DSP - Digital Signal Processing
 - 실시간 소수점 연산



MPU

- Z-80 (Zilog), 6502 (MOS)
- MCS51, 52 (8051, 8751, 8951)
- NEC V25, V55
- PIC, AVR
- ARM7, ARM9, ARM11
- Cortex-M, Cortex-R, Cortex-A
 - M0, M0+, M3, M4, M7
- RISC-V
 - Open source architecture
 - FPGA에 결합하여 사용



프로그램 설치

- st.com 접속 후 회원 가입
- cubeide 검색 후 프로그램 다운로드
- 설치 오류 발생시 언어 로케일 변경

The screenshot shows a web browser displaying the STMicroelectronics website at [st.com/en/development-tools/stm32cubeide.html](https://www.st.com/en/development-tools/stm32cubeide.html). The page title is "STM32CubeIDE - Integrated Dev...". The main content area is titled "All features" and lists the following points:

- Integration of services from STM32CubeMX STM32 microcontroller, microprocessor, development platform and example project selectionPinout, clock, peripheral, and middleware configurationProject creation and generation of the initialization codeSoftware and middleware completed with enhanced STM32Cube Expansion Packages
- Based on Eclipse®/CDT™, with support for Eclipse® add-ons, GNU C/C++ for Arm® toolchain and GDB debugger

Below this, there is a "Read less" link. Further down, a "Get Software" section is shown with a table of software packages:

Part Number	General Description	Supplier	Download	All versions
STM32CubeIDE-DEB	STM32CubeIDE Debian Linux Installer	ST	Get latest	Select version
STM32CubeIDE-Lin	STM32CubeIDE Generic Linux Installer	ST	Get latest	Select version
STM32CubeIDE-Mac	STM32CubeIDE macOS Installer	ST	Get latest	Select version
STM32CubeIDE-RPM	STM32CubeIDE RPM Linux Installer	ST	Get latest	Select version
STM32CubeIDE-Win	STM32CubeIDE Windows Installer	ST	Get latest	Select version

eclipse

- 자바를 비롯하여 다양한 언어를 지원하는 프로그래밍 통합 개발 환경
- 이클립스 재단에서 개발
- 415가지 이상의 프로젝트에 사용됨
- Visual Studio Code, ATOM

gcc

➤ GNU General Public License

➤ GNU Compiler Collection

➤ VS

- Keil

- 세계최초 8051용 C 컴파일러 개발 / 독일
- 8051, 80251, ARM, XC16x C16x, ST10

- IAR

- 1983년 스웨덴
 - 8051, ARM, AVR, H8, MSP430, RISC-V, STM8, SAM8, ARM7, ARM9, ARM11
- Mbed, MongoosOS, Amazon IoT, Azer,...



Keil MDK



VS

IAR Embedded
Workbench



환경 설정

- 코딩 스타일 변경
 - Windows / Preferences 명령의 C/C++ > Code Style > Formatter
 - K&R BSD GNU Whitesmiths
- 스타일 반영
 - Ctrl + A 모두 선택
 - Ctrl + Shift + F 스타일 반영
- TODO 사용하기
 - Windows / Preferences 명령의 C/C++ > Task Tag 에서 순위 변경
 - Windows / Show View / Other 명령의 General > Tasks 패널 호출
 - 코드내 // TODO 설명 문구

편집기 사용

- Perspective : 관점
 - 편집기의 패널, 상태표시 창 등의 위치를 사용자 정의
- 최소화, 최대화, 이전크기

언어

➤ 기계어, 어셈블러

- 기계어 : 1011 0110 1000 1110
- 어셈블러 : LD R3, 10

High-level Language

```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

C/Java Compiler

```
TEMP = V(K)
V(K) = V(K+1)
V(K+1) = TEMP
```

Fortran Compiler

```
lw $t0, 0($2)
lw $t1, 4($2)
sw $t1, 0($2)
sw $t0, 4($2)
```

MIPS Assembler

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```

Assembly Language

➤ 컴파일언어

- 컴파일러를 이용하여 일괄 변환
- C, Pascal, C++, C#, Java

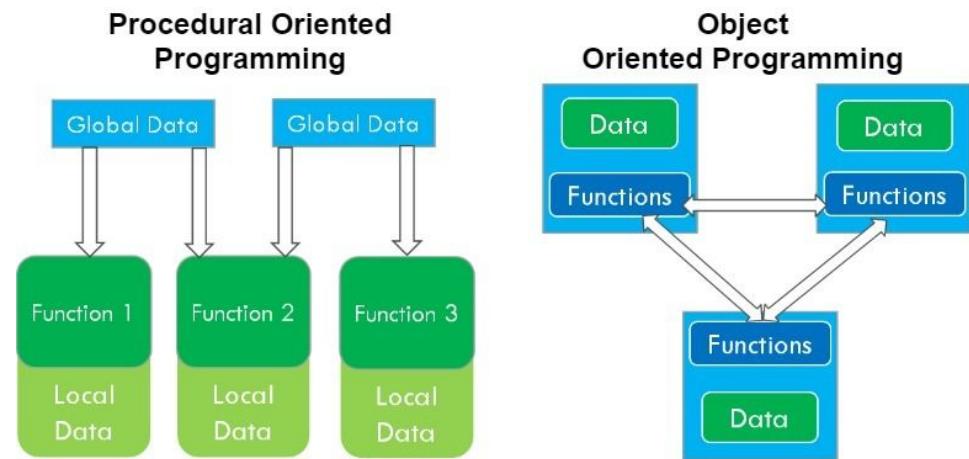
Machine Language

➤ 스크립트언어

- 인터프리터를 이용하여 한줄씩 변환하여 실행
- Basic, Python, Javascript, Perl, Skill

언어

- 기계어
 - 어셈블러를 디코딩하여 코드로 변환
- 절차언어
 - C, 코볼, 파스칼, 베이직
- 객체지향언어
 - C++, C#, Java, Javascript, Python
- MVC



소스 수정

STM32CubeIDE에서 자동 생성된 코드는 /* ~ BEGIN ~ */ 와 /* ~ END ~ */ 사이에 작성해당 문구 사이의 밖에 코드를 작성시 칩환경 설정의 변경에 따른 코드의 생성시 삭제됨

주석 및 기타 한글 문자는 코드 자동 생성시 인코딩과정으로 깨짐에 주의

```
④ int main(void)
{
    /* USER CODE BEGIN 1 */

    /* USER CODE END 1 */

    /* MCU Configuration-----*/
    /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
    HAL_Init();
}
```

주석, 괄호

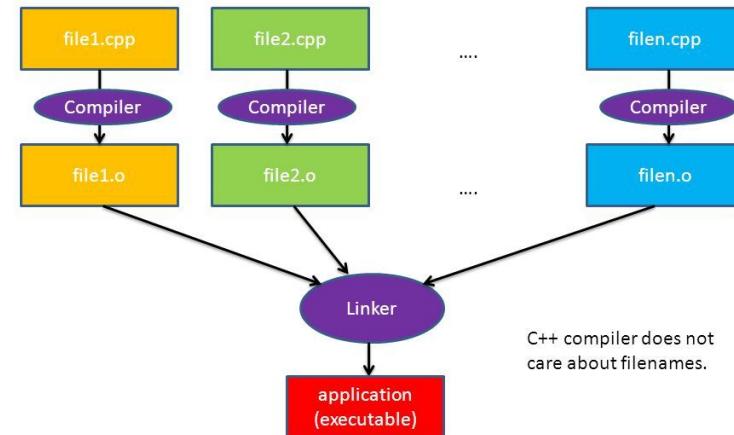
- 주석
 - // 한줄 주석처리
 - /* 시작부터 */ 종료까지
 - //*/활용
- {}
 - 그룹 괄호
- []
 - 배열 괄호
- ()
 - 함수 괄호
 - 연산 우선순위
- <>
 - #include <>, #include “”

컴파일러의 동작

Compile : C, Header를 Object파일로 컴파일 *.o

Link : *.o 결합하여 실행파일로 변환

Compiler & Linker expectations



전기처리

- 컴파일시 할당되는 명령
- #으로 시작하고 ;가 없음
- **#include <stdio.h>**
 - 시스템에서 기본적으로 제공하는 라이브러리
 - stdlib, string, math, mem
- **#include “lcd.h”**
 - 작업폴더에 존재하는 라이브러리
 - 서브 폴더 지정 가능
 - #include “sub/motor.h”
- **#define delay(x) HAL_Delay(x)**
 - 전달자 사용 가능
- **#if, #ifndef, #ifdef, #endif #else**
 - #ifdef DEBUG

변수형

- boolean (_Bool)
 - 1 bit 0, 1 참, 거짓
- char
 - 문자, -128~0~127
- int
 - 정수형, 0~2^32
- long
 - 배정도, int x 2
- float
 - 실수, 소수 32bit
- double
 - float x 2

embedded 전용 변수형

변수형		범위
uint8_t	unsigned integer 8bit	0 ~ 255
uint16_t	unsigned integer 16bit	0 ~ 65,535
uint32_t	unsigned integer 32bit	0 ~ 4,294,967,296
int8_t	signed integer 8bit	-128 ~ 127
int16_t	signed integer 16bit	-32,768 ~ 32,767
int32_t	signed integer 32bit	-2,147,483,648 ~ 2,147,483,647

little endian vs big endian

little endian : 가치가 낮은 8bit를 먼저 저장

big endian : 가치가 높은 8bit를 먼저 저장



변수 형식

- static
 - 변수의 사용이 종료된 뒤에도 값을 기억
 - 해당 블럭의 재진입시 이전 값을 기억하여 연속적인 기록
 - 전역 변수를 대체
- volatile
 - 컴파일러에 의한 최적화 기능을 배제
- extern
 - 다른 파일에서 선언된 변수를 재정의

float and double

32bit = 부호1 + 지수8 + 가수 23

$$\text{Value} = (-1)^S \times M \times 2^{(E-127)}, \text{ or Value} = (-1)^S \times M \times 2^{(E-1023)}$$

Figure 1. Single precision number format

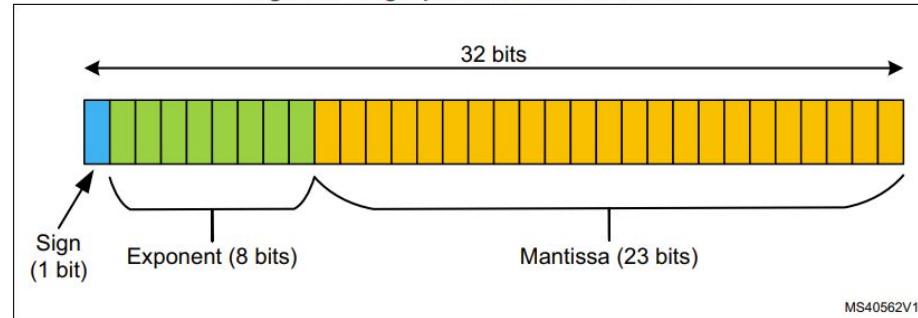
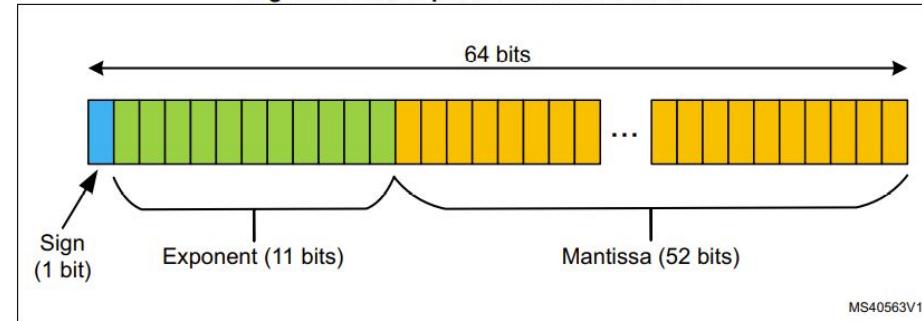


Figure 2. Double precision number format



float의 사용

float value = 10.34f;

연산자

➤ 사칙 연산

- + - * %

➤ 대입

- =

➤ 비교

- < > <= >= ! <> == !=

➤ 논리 연산자

- && ||

➤ 비트 연산자

- & | ^

➤ 비트 회전

- >> <<

Precedence	Operator	Description	Associativity
1	++ --	Suffix/postfix increment and decrement	Left-to-right
	()	Function call	
	[]	Array subscripting	
	.	Structure and union member access	
	->	Structure and union member access through pointer	
2	(type){list}	Compound literal <small>(c99)</small>	
	++ --	Prefix increment and decrement <small>[note 1]</small>	Right-to-left
	+ -	Unary plus and minus	
	! ~	Logical NOT and bitwise NOT	
	(type)	Cast	
	*	Indirection (dereference)	
	&	Address-of	
3	sizeof	Size-of <small>[note 2]</small>	Left-to-right
	_Alignof	Alignment requirement <small>(c11)</small>	
4	* / %	Multiplication, division, and remainder	
5	+ -	Addition and subtraction	
6	<<>>	Bitwise left shift and right shift	
7	< <=	For relational operators < and ≤ respectively	
8	> >=	For relational operators > and ≥ respectively	
9	== !=	For relational = and ≠ respectively	
10	&	Bitwise AND	
11	^	Bitwise XOR (exclusive or)	
12		Bitwise OR (inclusive or)	
13	&&	Logical AND	
14		Logical OR	
14 <small>[note 4]</small>	? :	Ternary conditional <small>[note 3]</small>	Right-to-left
	=	Simple assignment	
	+= -=	Assignment by sum and difference	
	*= /= %=	Assignment by product, quotient, and remainder	
	<<= >>=	Assignment by bitwise left shift and right shift	
15	&= ^= =	Assignment by bitwise AND, XOR, and OR	Left-to-right
	,	Comma	

함수

- **function** : 함수
 - C언어, 파스칼
- **procedure** : 절차
 - 파스칼
- **method** : 방법
 - 객체지향 언어

함수

리턴형식 함수명(매개변수)

```
int sub(int a);
```

```
int main(void) {
```

```
    int b = 1;
```

```
    b = sub(b);
```

```
}
```

```
int sub(int a) {
```

```
    a = a + 2;
```

```
    return a;
```

```
}
```

printf, sprintf, snprintf, fprintf

printf(문자열, 변수...);

formatter

%d - 10진수의 수로 표현

%x - 16진수의 수로 표현

%f - float형식으로 표현

%c - 문자로 표현

%s - 문자열로 표현

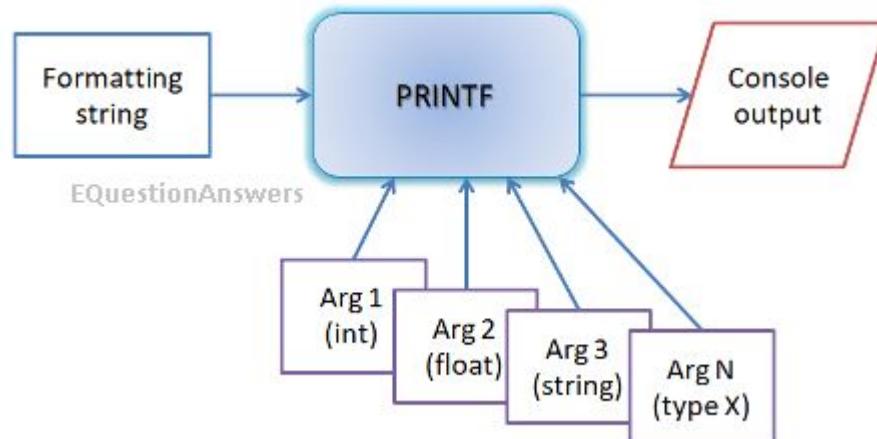
specification character

\n - new line, 0x0a

\r - cr, 0x0d

\t - tab, 0x09

Project 우측버튼 > Properties > C/C++ Build > Settings > Tool Settings > MCU Settings > “Use float with printf from newlib-nano (-u_printf_float)”



비교문

```
if (조건 수식) {
```

참인 경우 실행될 코드

```
}
```

```
else {
```

거짓인 경우 실행될 코드

```
}
```

비교문

```
switch (비교변수) {  
    case 값:  
        변수와 값이 동일한 경우 실행될 코드  
        break;  
    case 값:  
        변수와 값이 동일한 경우 실행될 코드  
        break;  
}
```

반복문 (for)

for(초기값 ; 비교문 ; 증분) {

 비교문 만족시 실행될 코드

}

 실행순서

for (1; 2, 5 ; 4) {

 3, 6

}

반복문 (while)

```
while (조건식) {
```

```
}
```

```
int a = 0;
```

```
while (a < 10) {
```

```
    printf("%d\n", a);
```

```
    a++;
```

```
}
```

```
break; 반복 구간 탈출
```

```
continue; 반복 구간의 시작점으로 이동
```

반복문 (do while)

```
do {  
    반복 실행 코드  
} while(조건식);
```

3항 연산자

(조건) ? 참인 경우 실행코드 : 거짓인 경우 실행코드;

$a > 0 ? \text{printf}(\text{"is true"}) : \text{printf}(\text{"is false"})$;

잦은 실수

- 괄호 개수 누락
 - 마우스로 괄호를 클릭하여 시작, 종료지점 확인
- 세미콜론 누락
 - 오류발생지점 또는 오류발생 앞지점
- 오타
 - 자동완성(Ctrl + Spacebar) 기능의 활용
- 줄 맞춤
 - 모두 선택 (Ctrl + A)
 - Ctrl + Shift + F
- 이름 동시 변경
 - Ctrl + 1

컴파일, 실행, 디버그

컴파일 - 파일의 저장, 컴파일, 링크 과정을 진행

실행 - 컴파일 및 다운로드 과정을 진행

디버그 - 컴파일 및 다운로드 후 디버그 시작



다중 파일

main.c

```
#include "main.h"

void init() {
    initLcd();
    initMotor();
}

void main() {
    init();
    while(1) {
    }
}
```

main.h

```
#ifndef __main_h__
#define __main_h__

int valueGlobal;

#include "lcd.h"
#include "motor.h"

#endif
```

lcd.c

```
#include "lcd.h"

extern int stateMotor;

void initLcd() {
}

void putLcd() {
```

lcd.h

```
#ifndef __lcd_h__
#define __lcd_h__

#include "main.h"

int stateLcd;

void initLcd();
void putLcd();

#endif
```

motor.c

```
#include "motor.h"

int stateMotor;

void initMotor() {
}

void runMotor(int dir) {
```

motor.h

```
#ifndef __motor_h__
#define __motor_h__

#include "main.h"
#include "lcd.h"

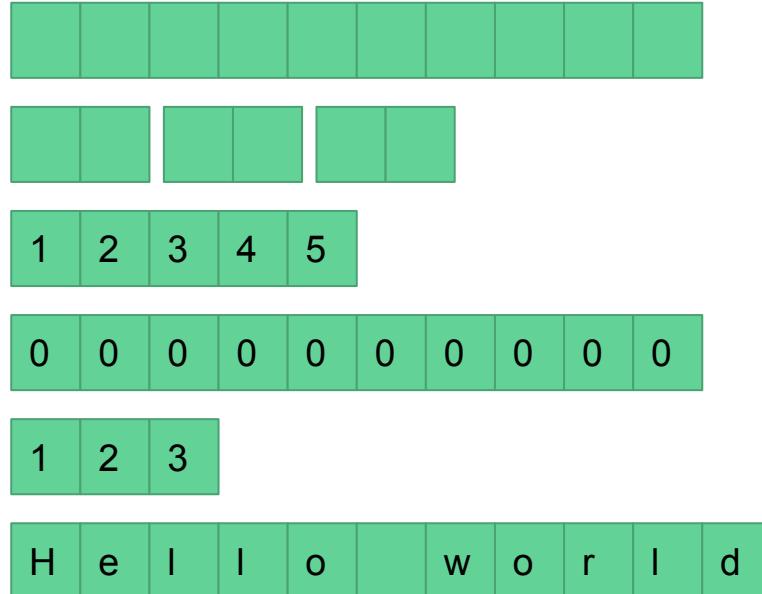
void initMotor();
void runMotor(int);

#endif
```

배열

➤ 자료형 변수 이름[개수]

- int array[10];
- int array[2][3];
- int array[5] = {1, 2, 3, 4, 5};
- int array[10] = {0,};
- int array[] = {1, 2, 3};
- int array[] = {"Hello world"};



배열의 길이?

배열의 길이를 알아옴

`sizeof();`

문자열의 길이를 알아옴

`strlen();`

포인터 입문

&연산자는 변수의 주소를 반환한다

*연산자는 포인터 변수를 선언한다

포인터란?

변수의 주소를 가르키는 공간, 자신의 공간을 가지고 있지 않음

배열 = 포인터

```
volatile unsigned int *myPointer = (volatile unsigned int *)0x12345678;
```

포인터 관리

```
char *pt;  
  
pt = malloc(10);           // 10개의 저장공간 확보  
  
pt = realloc(pt, 20);     // 20개의 저장공간으로 확장  
  
free(pt);                // 모든 저장공간 해제
```

메모리 관리

memset(pt, 0, 10); // pt의 저장공간에 0으로 10개를 채움 (8bit 기준)

memcpy(pt, str, 5); // str의 데이터 5개를 pt에 복사 (8bit 기준)

구조체 struct

형 선언

```
typedef 기존형 새로운형;
```

구조체 선언 : struct

```
typedef struct {
```

```
    char a;
```

```
    char b;
```

```
} newtype_t;
```

```
newtype_t var;
```

```
var.a = 10;
```

```
var.b = 20;
```

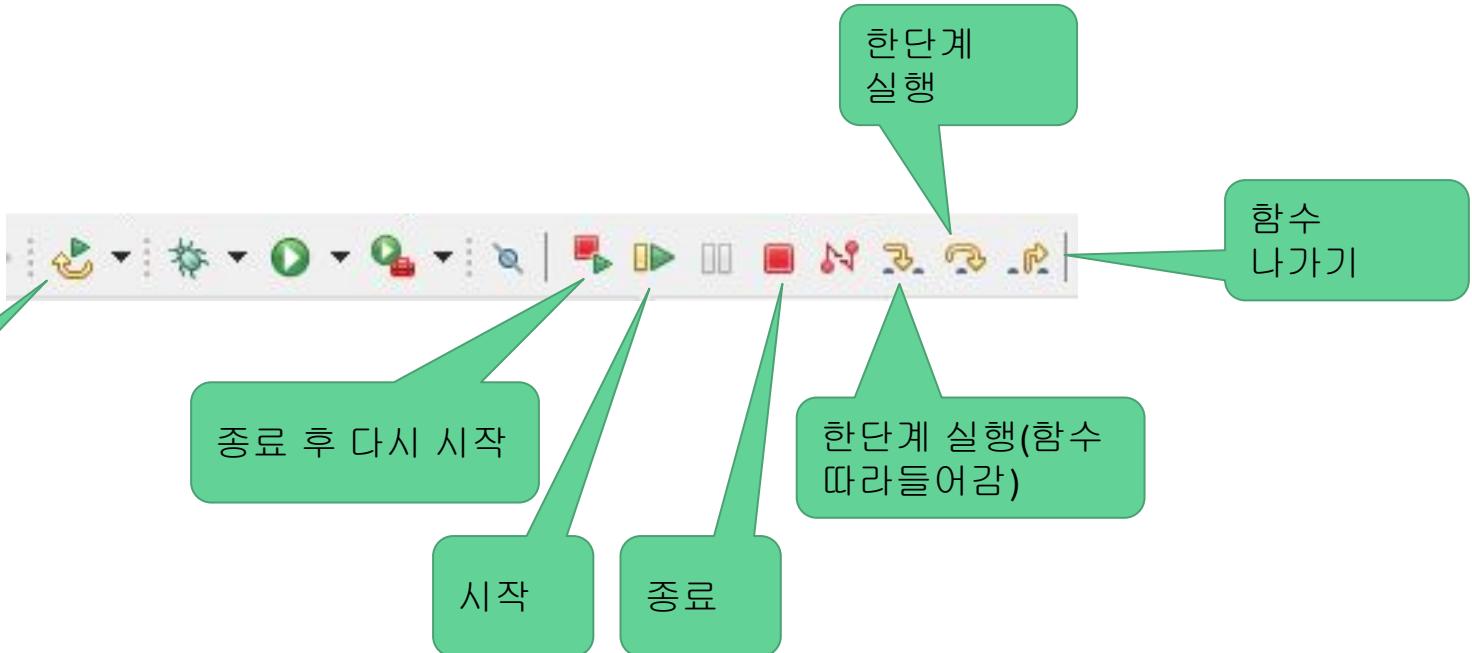
구조체 포인터

```
typedef struct {  
    char a;  
    char b;  
} newtype_t;  
  
newtype_t var;  
  
newtype_t *pt;  
  
pt = &var;  
  
pt->a = 10;  
  
pt->b = 20;
```

함수 포인터

```
1 #include <stdio.h>
2
3 // 합수포인터 타입 정의
4 typedef int (*calcFuncPtr)(int, int);
5 // 덧셈 합수
6 int plus (int first, int second) {
7     return first + second;
8 }
9 // 뺄셈 합수
10 int minus (int first, int second) {
11     return first - second;
12 }
13 // 곱셈 합수
14 int multiple (int first, int second) {
15     return first * second;
16 }
17 // 나눗셈 합수
18 int division (int first, int second) {
19     return first / second;
20 }
21
22 void main(void) {
23     calcFuncPtr calc = NULL;
24     int a = 0, b = 0;
25     char op = 0;
26     int result = 0;
27
28     scanf ("%d %c %d", &a, &op, &b);
29
30     switch (op) // 합수포인터 calc에 op에 맞는 합수들의 주소를 넣음
31    {
32         case '+':
33             calc = plus;
34             break;
35
36         case '-':
37             calc = minus;
38             break;
39
40         case '*':
41             calc = multiple;
42             break;
43
44         case '/':
45             calc = division;
46             break;
47     }
48
49     result = calc(a, b);
50
51     printf ("result : %d", result);
52 }
```

Debug



Debug

Debug

함수의 실행 위치 추적



통신

➤ 무선 / 유선

- 무선 : WiFi, Bluetooth, Zigbee, Lora, NB-IoT
- 유선 : 병렬, 직렬
 - 병렬 :
 - 직렬 : 동기, 비동기 (동기 = 클럭)
 - 비동기 : RS-232-C, RS-485, CAN (시간, 통신속도 Baud rate)
 - 동기 : I2C, SPI, Serial Wire, I2S, QSPI

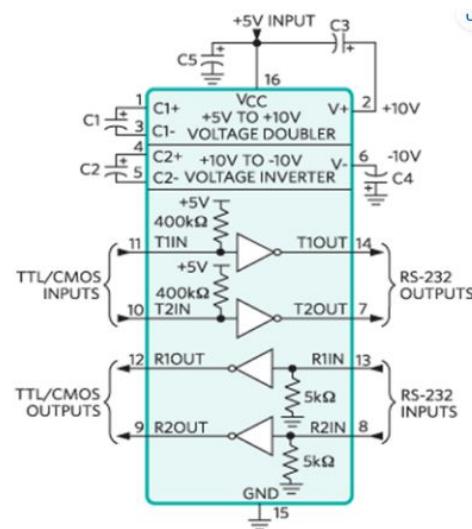
RS-232-C / RS-485

➤ RS-232-C

- 1:1 통신
- 유선 비동기 통신 규격
- 전이중(Full Duplex)
- Baud Rate : 통신 속도 300bps ~ 115200bps
- Driver IC : MAX232

➤ RS-485

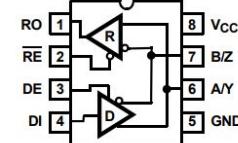
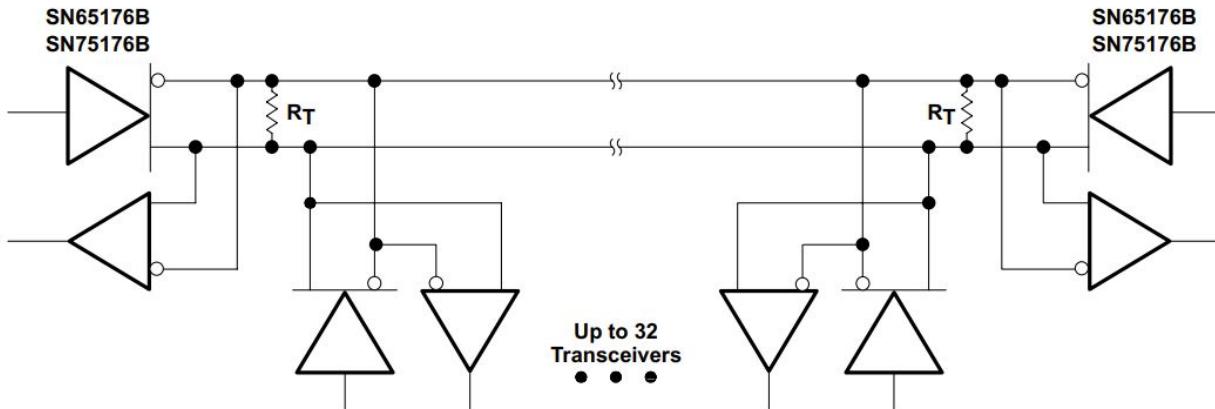
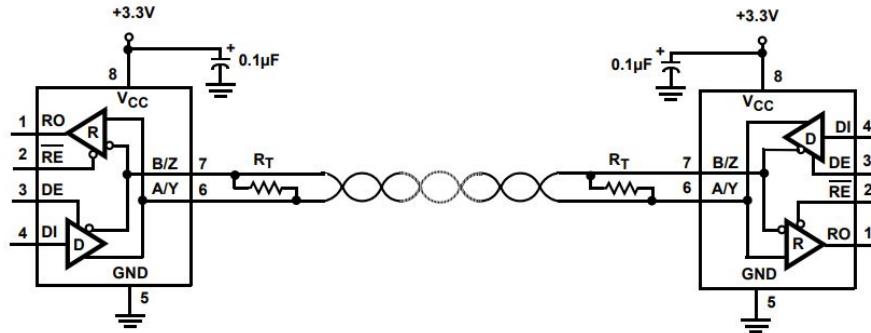
- 1:n 통신
- 반이중(Half Duplex), 송수신 모드 설정 핀 필요
- CAN
- Driver IC : 75176, 전압, 전류 고려



RS-485 연결

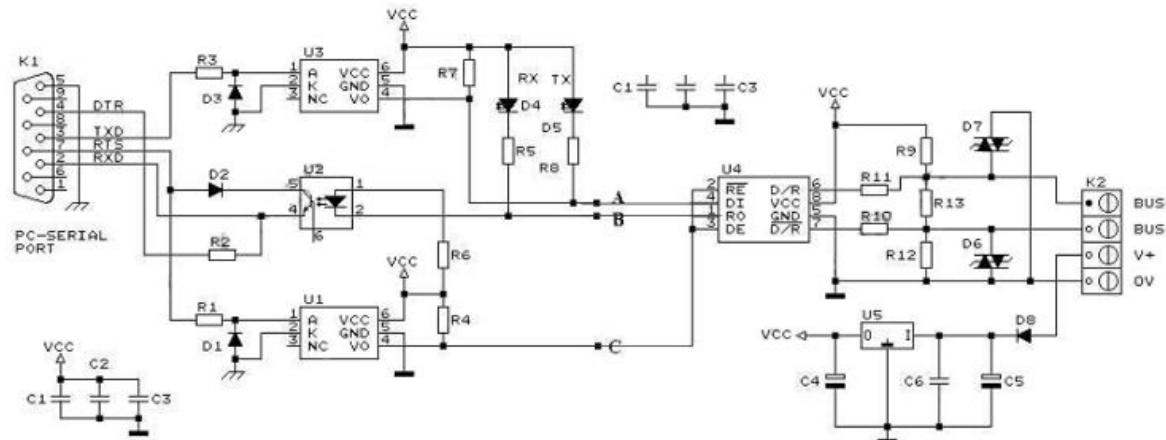
- 전압
- 전류

[SNx5176B Differential Bus Transceivers datasheet \(Rev. H\)](#)



회로구성

- 터미네이션 저항
- 풀업, 풀다운
- TVS다이오드



Uart 송수신

- 송신
 - HAL_UART_Transmit(&handle, pTxBuf, length, timeout);
- 수신
 - Polling
 - HAL_UART_Receive(&handle, pRxBuf, length, timeout);
 - Interrupt
 - HAL_UART_Receive_IT(&handle, pRxBuf, length);
 - length의 개수만큼 수신시 HAL_UART_RxCpltCallback 호출, 인터럽트 재설정 필요
 - DMA
 - HAL_UART_ReceiveDMA(&handle, pRxBuf, length);

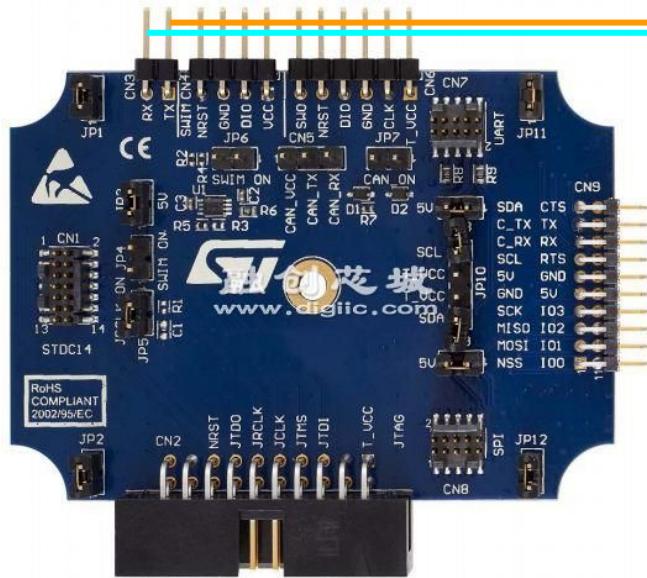
printf 재설정

```
#include <stdio.h>

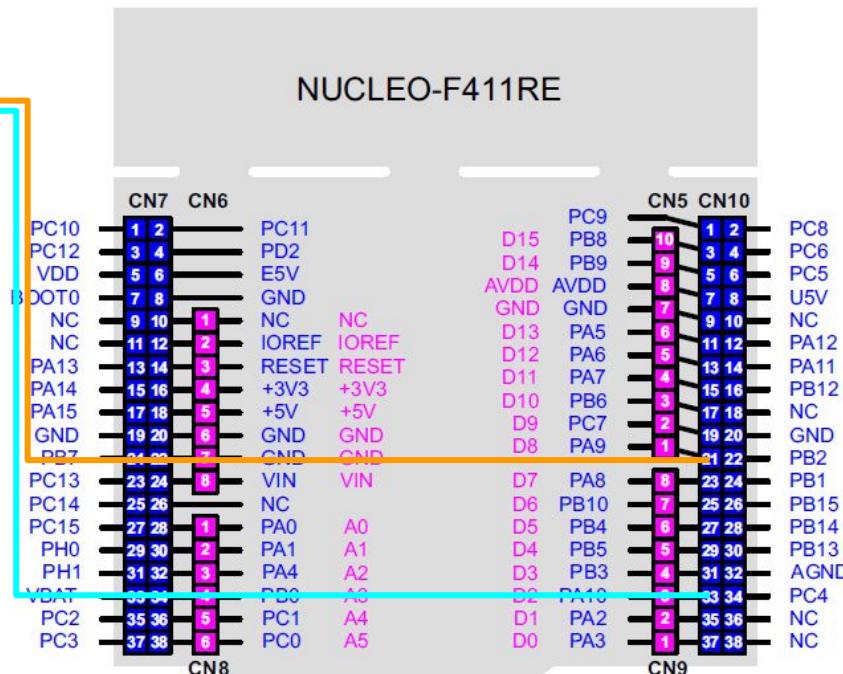
int _write(int file, char* p, int len){
    HAL_UART_Transmit(&huart2, p, len, 10);

    return len;
}
```

F411RE UART



NUCLEO-F411RE



Arduino

Morpho

유용한 터미널

SerialPlot

<https://hackaday.io/project/5334-serialplot-realtime-plotting-software>

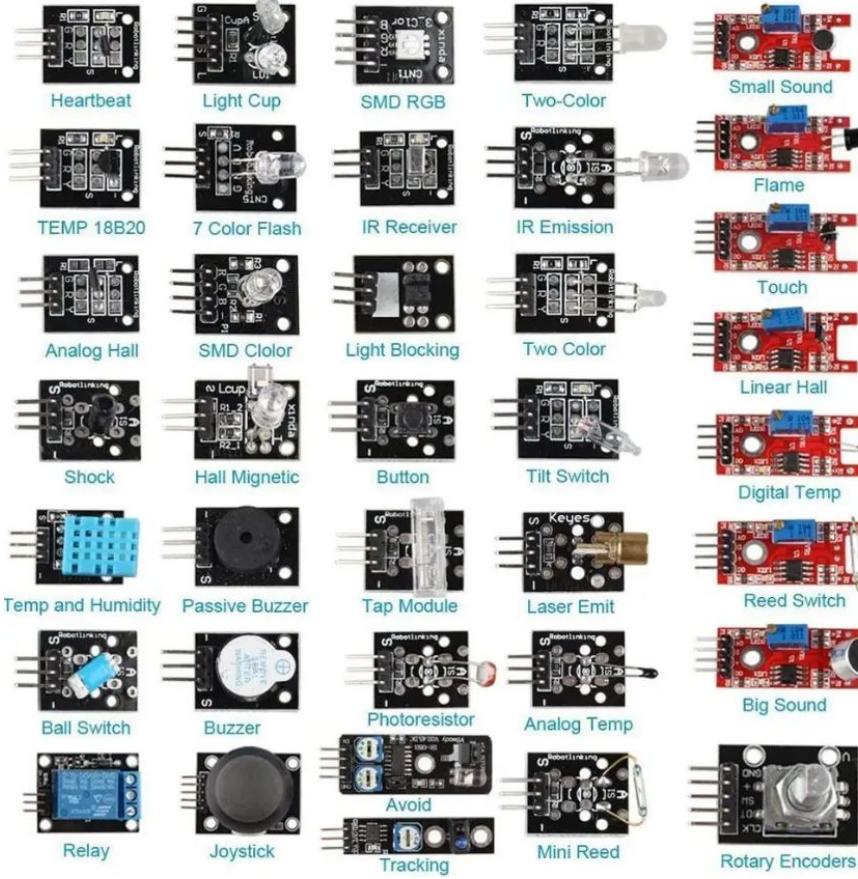
CoolTerm

<https://freeware.the-meiers.org/>

SerialPortMon

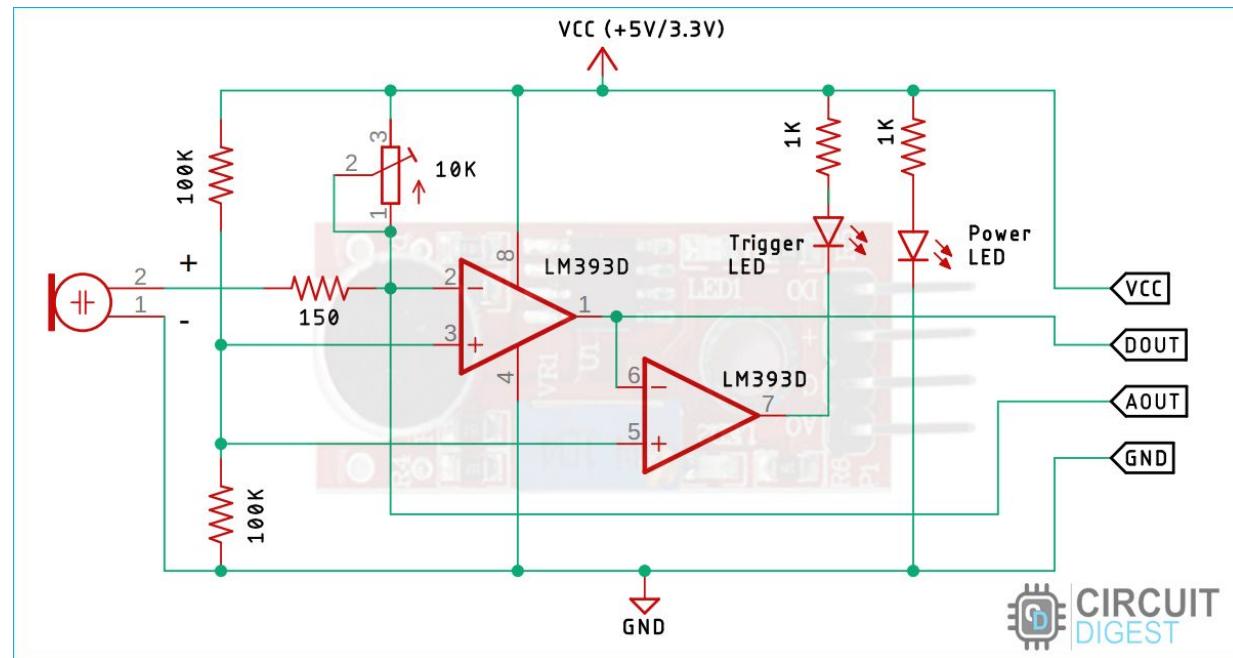
https://drive.google.com/file/d/195U9P-KiFRDWI0i-d7MuGajTTVZJuNXb/view?usp=drive_link

센서



OpAmp

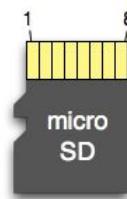
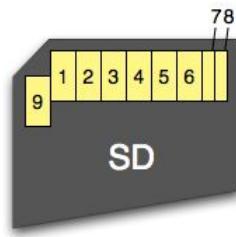
- 차동증폭기
- 비교기



SD메모리

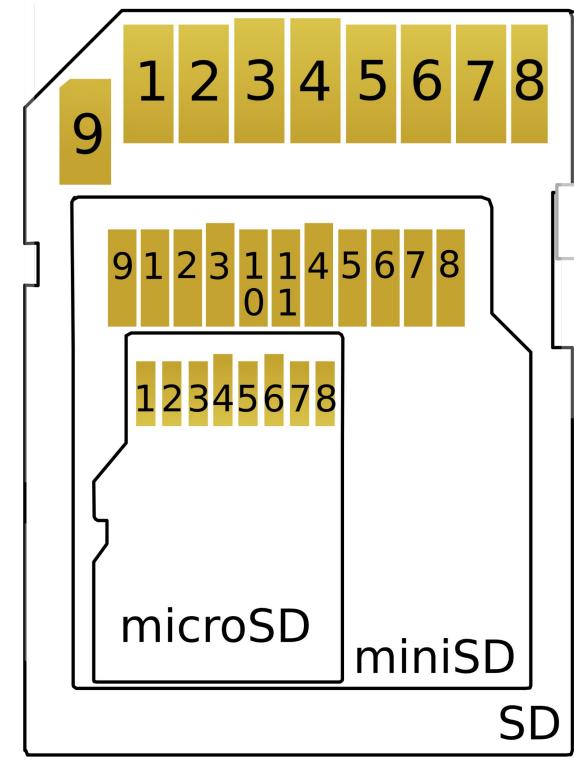
3.3V전압

SPI통신을 이용함



Pin	SD	SPI
1	CD/DAT3	CS
2	CMD	DI
3	VSS1	VSS1
4	VDD	VDD
5	CLK	SCLK
6	VSS2	VSS2
7	DAT0	DO
8	DAT1	X
9	DAT2	X

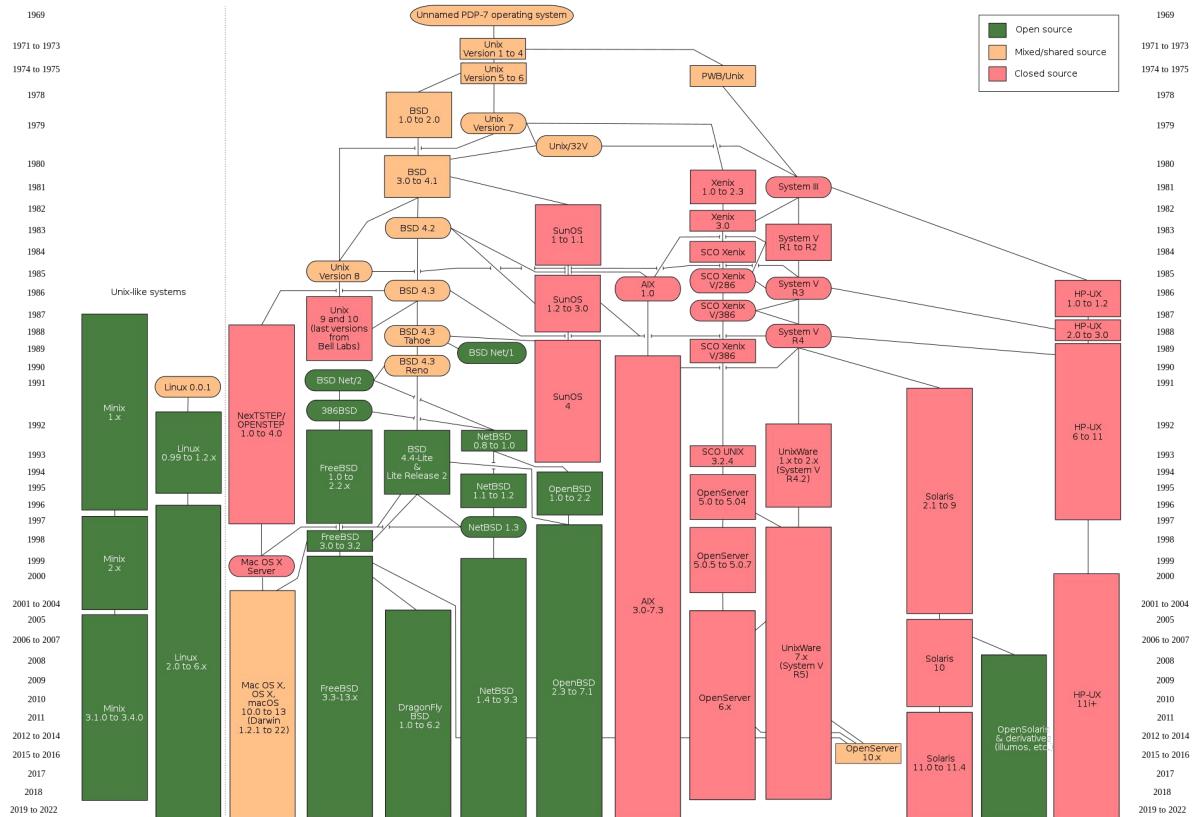
Pin	SD	SPI
1	DAT2	X
2	CD/DAT3	CS
3	CMD	DI
4	VDD	VDD
5	CLK	SCLK
6	VSS	VSS
7	DAT0	DO
8	DAT1	X



Operating System

1981년 IBM-PC용
MS-DOS를 기반으로
Windows가 출시됨

MS vs Etc



File system

컴퓨터에서 파일이나 자료를 쉽게 찾고 접근할 수 있도록 보관 또는 조작하는 체계
하드디스크, CD-ROM, USB, SD카드 등에서 사용
FAT16, FAT32, NTFS, exFAT, POSIX



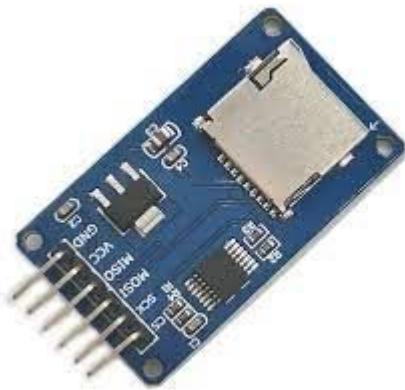
embedded용 파일시스템

emFile : 독일 segger사에서 제작

FatFs : 일본 사이타마의 Cha N가 제작(공개소스)

<http://elm-chan.org/fsw/ff/>

SD card module



FatFs

```
f_mount(&FatFs, "", 0);  
f_open(&fil, "read.me", FA_READ | FA_WRITE);  
f_read(&fil, buffer, sizeof(buffer), &br);  
f_write(&fil, buffer, sizeof(buffer), &br);  
f_close(&fil);  
f_unmount("0:");
```

Hexa editor

<https://mh-nexus.de/en/hxd/>

The screenshot shows the HxD hex editor interface. The main window displays the source code of `main.c` in ANSI mode. The code is a C program for a STM32CubeIDE project, specifically the HelloWorld example. It includes definitions for `ClockConfig.h`, `RCC.h`, and `ClkInitStruct`, and initializes various peripherals like TIM2 and USART1. The `main` function sets up the system clock, initializes peripherals, and enters an infinite loop. The `Data inspector` panel on the right shows memory dump details for the current selection.

Offset(h)	Decoded text
000011B0	43 6C 6F 63 6B 43 6F 6E 66 69 67 28 26 52 43 43
000011C0	ClockConfig.h
000011D0	5F 43 6C 6B 49 6E 69 74 53 74 72 75 63 74 2C 20
000011E0	_ClkInitStruct,
000011F0	46 4C 41 53 48 5F 4C 41 54 45 4E 43 59 5F 30 29
00001200	FLASH_LATENCY_0)
00001210	20 21 3D 20 49 41 4C 5F 4F 48 29 0D 0A 20 20 78 != HAL_OK).. {
00001220	0D 0A 20 20 20 45 72 72 6F 72 5F 46 61 6E 64 .. Error_Hand
00001230	6C 65 72 28 29 3B 0D 0A 20 20 7D 0D 02 7D 0D 08 lex()...)...
00001240	0D 0A 2F 28 2A 0D 02 20 20 20 40 62 72 69 65 65 ../*... * @brief
00001250	06 20 54 49 4D 32 20 49 6E 69 74 69 61 6C 69 74 f TIM2 Initialization Function..
00001260	61 74 69 6F 6E 20 46 75 6E 63 74 69 6F 6E 0D 0A . * @param None.
00001270	20 20 2A 20 40 70 61 72 61 6D 20 4E 6F 6E 65 0D . * @retval Non
00001280	0A 20 20 20 40 72 65 74 76 61 6C 20 4E 6F 6E . * @retval Non
00001290	65 0D 0A 20 20 2A 2F 0D 0A 73 74 61 74 69 63 20 e.. /*... static
000012A0	76 6F 69 60 20 9D 55 5F 54 49 4D 32 5F 49 6E 69 void MX_TIM2_Ini
000012B0	74 28 76 6F 69 64 28 0D 0A 7B 0D 0A 0B 20 20 t(void)....
000012C0	2F 2A 20 55 53 45 52 20 43 4F 44 45 20 42 45 47 /* USER CODE BEG
000012D0	000012A0 49 4E 20 54 49 4D 32 5F 49 6E 69 74 20 20 2A IN TIM2_Init 0 *
000012E0	2F 2A 20 55 53 45 52 20 43 4F 44 45 20 42 45 47 /* USER CODE BEG
000012F0	000012D0 2F 0D 0A 00 20 20 2F 2A 20 55 53 45 52 20 43 /.... /* USER C
00001300	000012E0 4F 44 45 4D 44 45 4F 54 49 4D 32 5F 49 6E 69 ODE END TIM2_Init
00001310	000012F0 74 20 30 20 2A 2F 0D 0A 0D 0A 20 20 54 49 4D 5F t 0 /*.... TIM
00001320	00001300 43 6C 6F 63 6B 43 6F 6E 66 69 67 54 79 70 65 44 ClockConfigTypeD
00001330	00001310 65 66 20 73 43 6F 63 6B 53 6F 75 72 63 65 45 ef sclockSourceC
00001340	00001320 6F 6E 66 69 67 20 3D 20 7B 3D 0D 0A 20 20 config = {0}....
00001350	00001330 54 79 70 65 44 65 60 20 73 4D 61 73 74 65 72 43 TypeDef sMasterC
00001360	00001340 20 20 2F 2A 20 55 53 45 52 20 43 4F 44 45 20 42 /* USER CODE B
00001370	00001350 45 47 49 4E 20 54 49 4D 32 5F 49 6E 69 74 20 31 EGIN TIM2_Init 1
00001380	00001360 20 2A 2F 0D 0A 0D 0A 20 20 2F 2A 20 55 53 45 52 /*.... /* USER
00001390	00001370 20 43 4F 44 45 20 45 4E 44 20 54 49 4F 44 45 5F 49 CODE END TIM2_I
000013A0	00001380 6E 69 74 20 31 20 2A 2F 0D 0A 20 20 60 74 69 6D nit 1 /*... htim
000013B0	00001390 32 2E 49 6E 73 74 61 6E 63 65 20 3D 20 54 49 4D 2.Instance = TIM
000013C0	000013A0 32 3B 0D 0B 20 20 68 74 69 6D 32 2E 4E 69 74 2... htim2.Init
000013D0	000013B0 2E 50 72 65 73 63 61 6C 65 72 20 3D 20 30 3B 0D .Prescaler = 0;
000013E0	000013C0 0A 20 20 68 74 69 6D 32 2E 4E 69 74 2E 43 6F . htim2.Init.Co
000013F0	000013D0 75 6E 74 65 72 4D 6F 64 65 20 3D 20 54 49 4D 5F unterMode = TIM_
00001400	000013E0 43 4F 55 4E 54 55 52 4D 4F 44 45 5F 55 50 3B 0D COUNTERMODE_UP_
00001410	00001400 0A 20 20 68 74 69 6D 32 2E 49 6E 69 74 2E 50 65 . htim2.Init.Pe

이미지

무압축 : bitmap

압축 : jpeg, png, tiff, gif

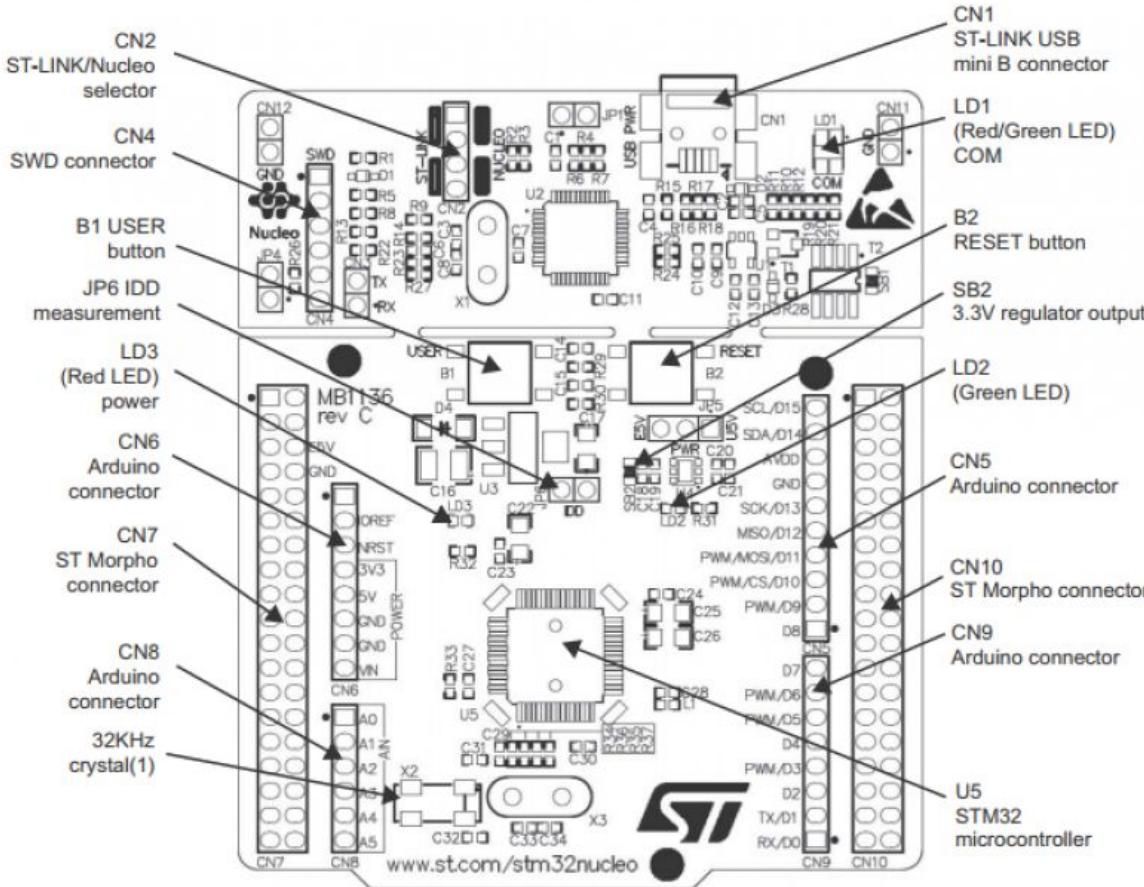
BMP file format

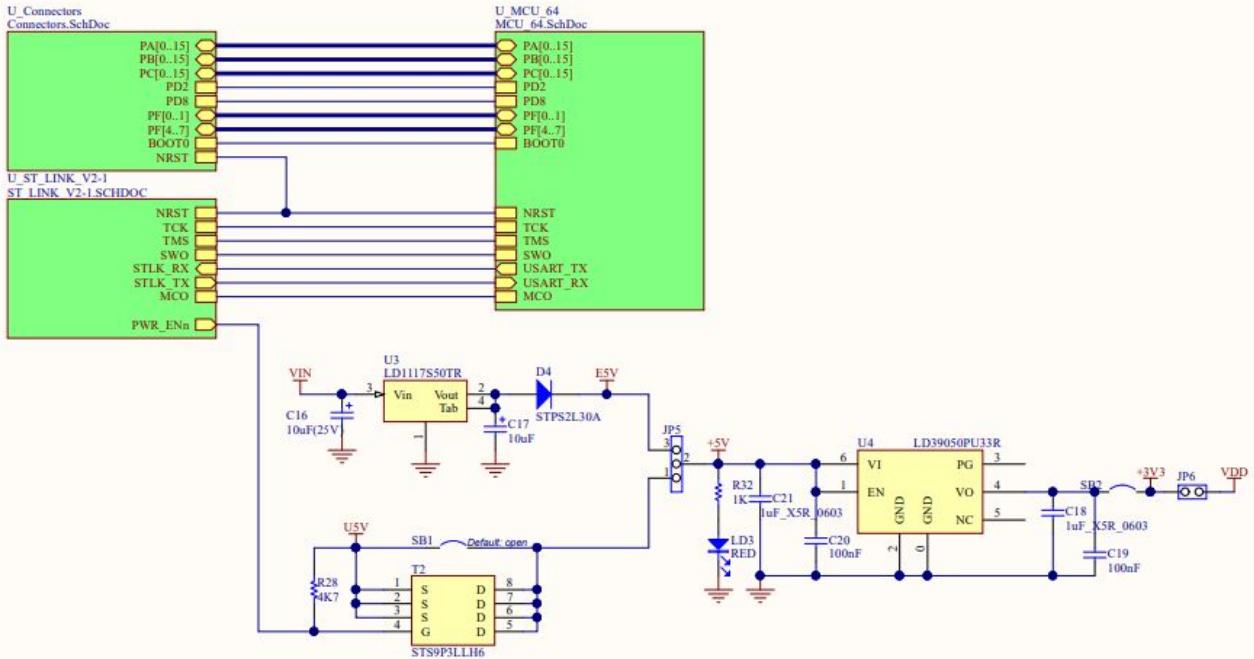
https://en.wikipedia.org/wiki/BMP_file_format

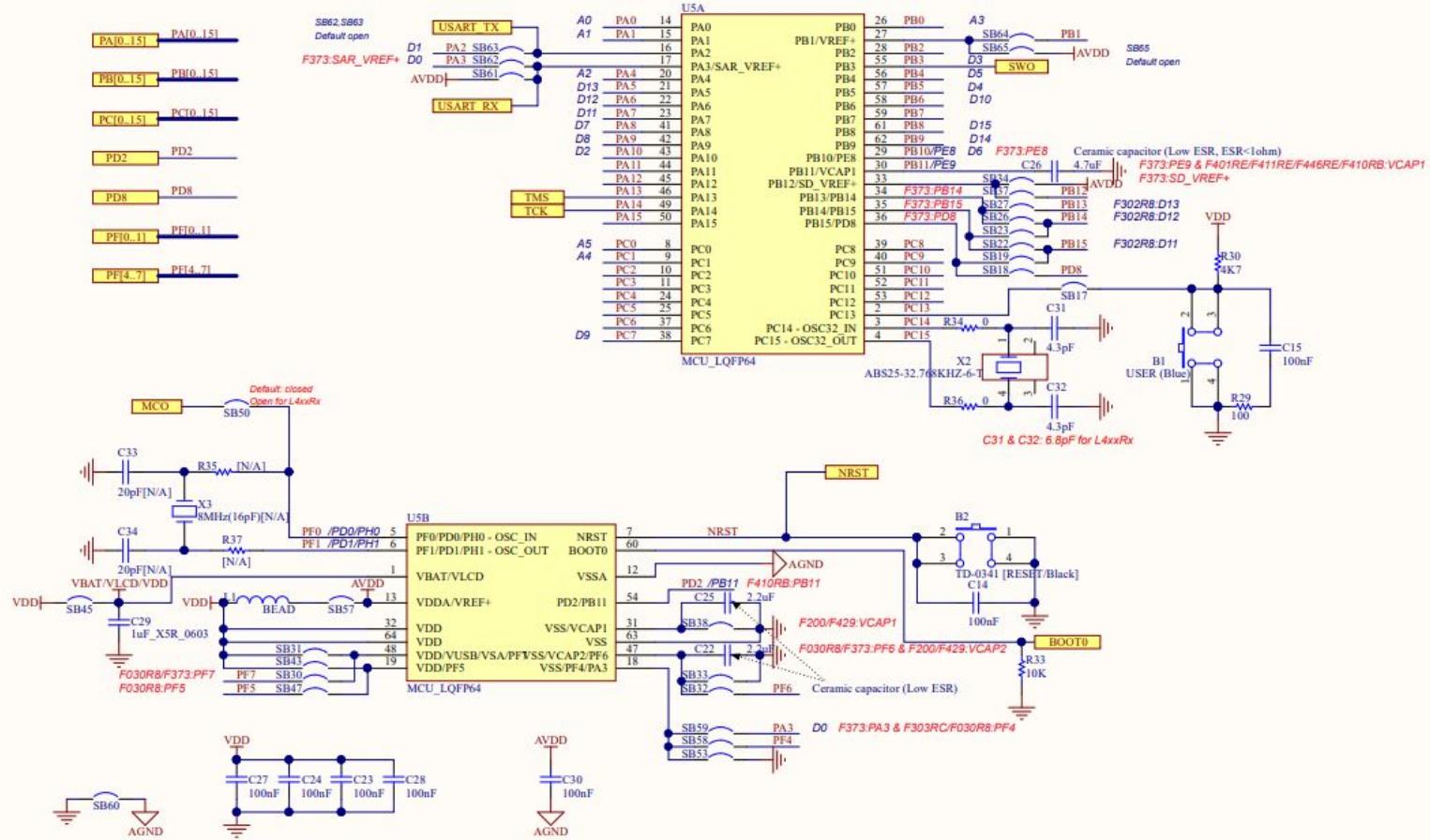
wave

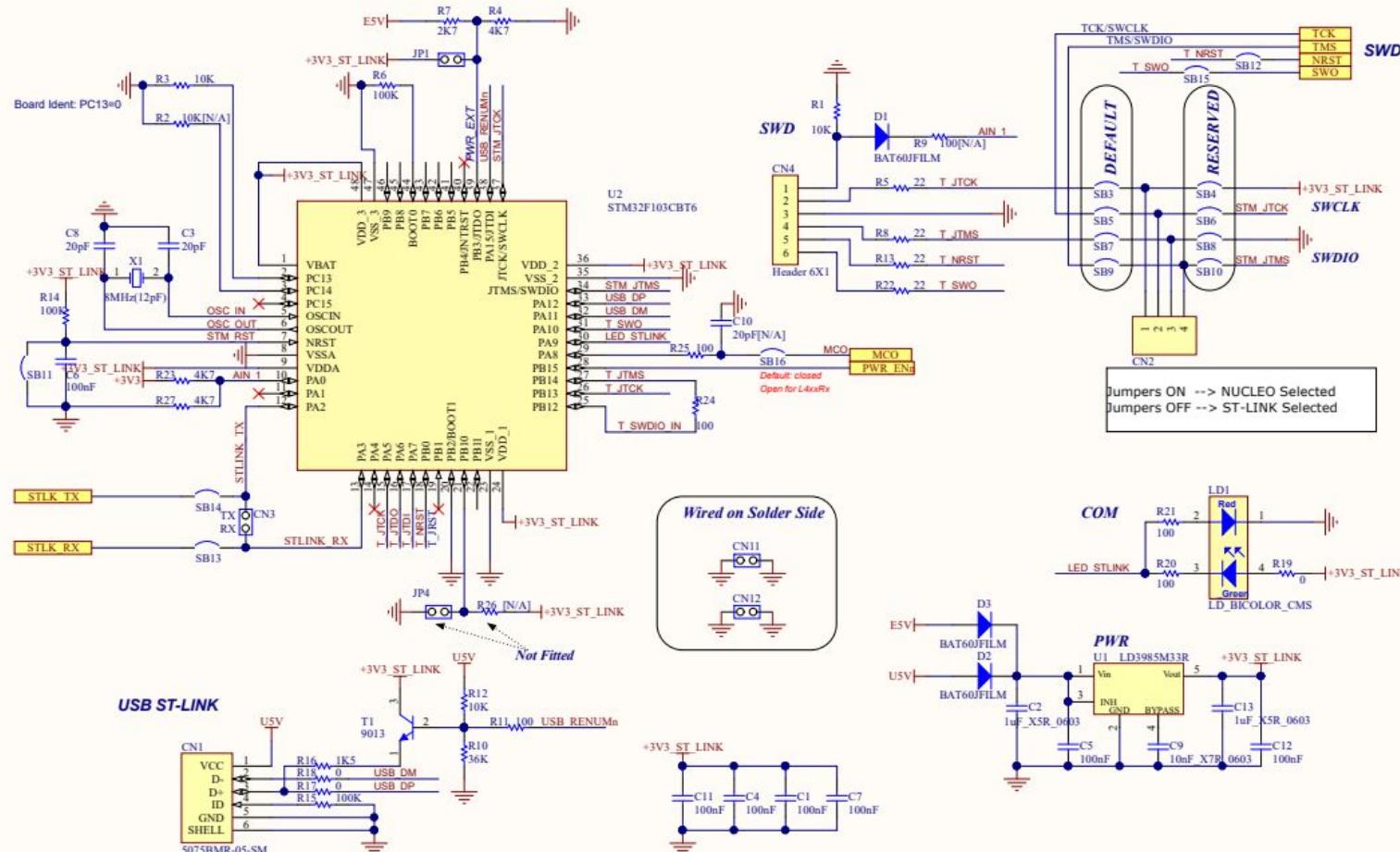
<https://en.wikipedia.org/wiki/WAV>

<http://soundfile.sapp.org/doc/WaveFormat/>







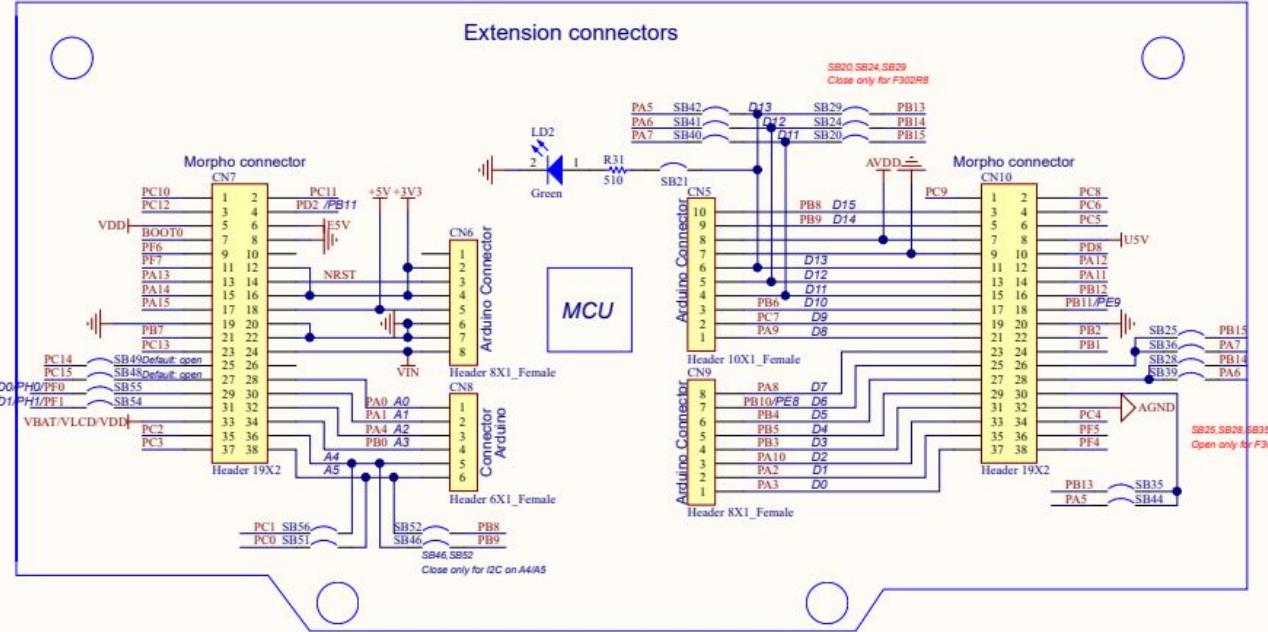


PA[0..15]	PA0..151
PB[0..15]	PB0..151
PC[0..15]	PC1..151
PD2	PD2
PD8	PD8
PF[0..1]	PF10..11
PF[4..7]	PF4..7
NRST	NRST
BOOT0	BOOT0

SB55 Default open
Closed for 4xRx PDDO PH0/PF0

PD1 PH1/PF1 SB54

VBAT/VLCD/VDD



전원공급

VDD : 1.8V ~ 3.3V

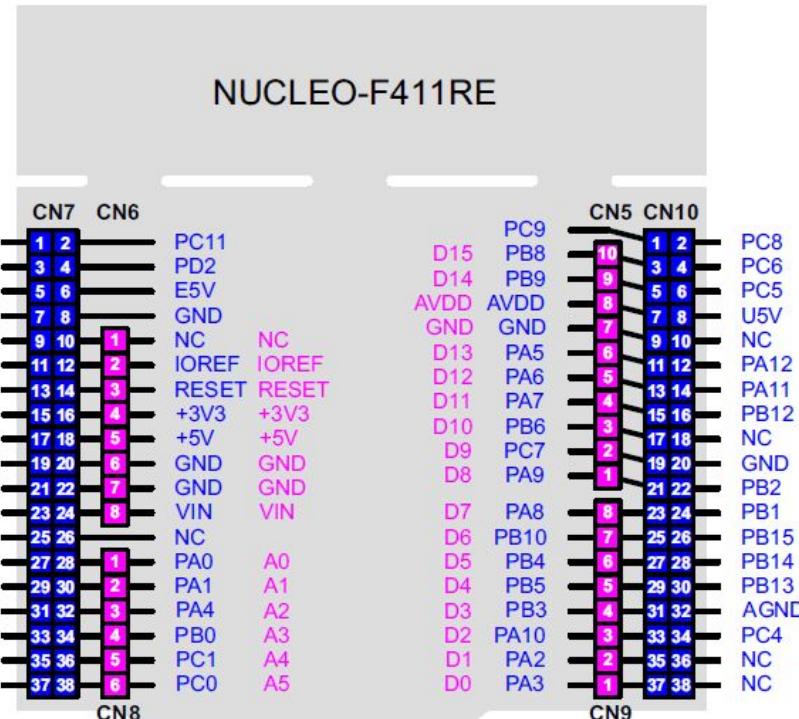
VSS : 0V, GND

VDDA : 아날로그 전원

VSSA : 아날로그 접지

POR : Power On Reset

BOR : Brown Out Reset

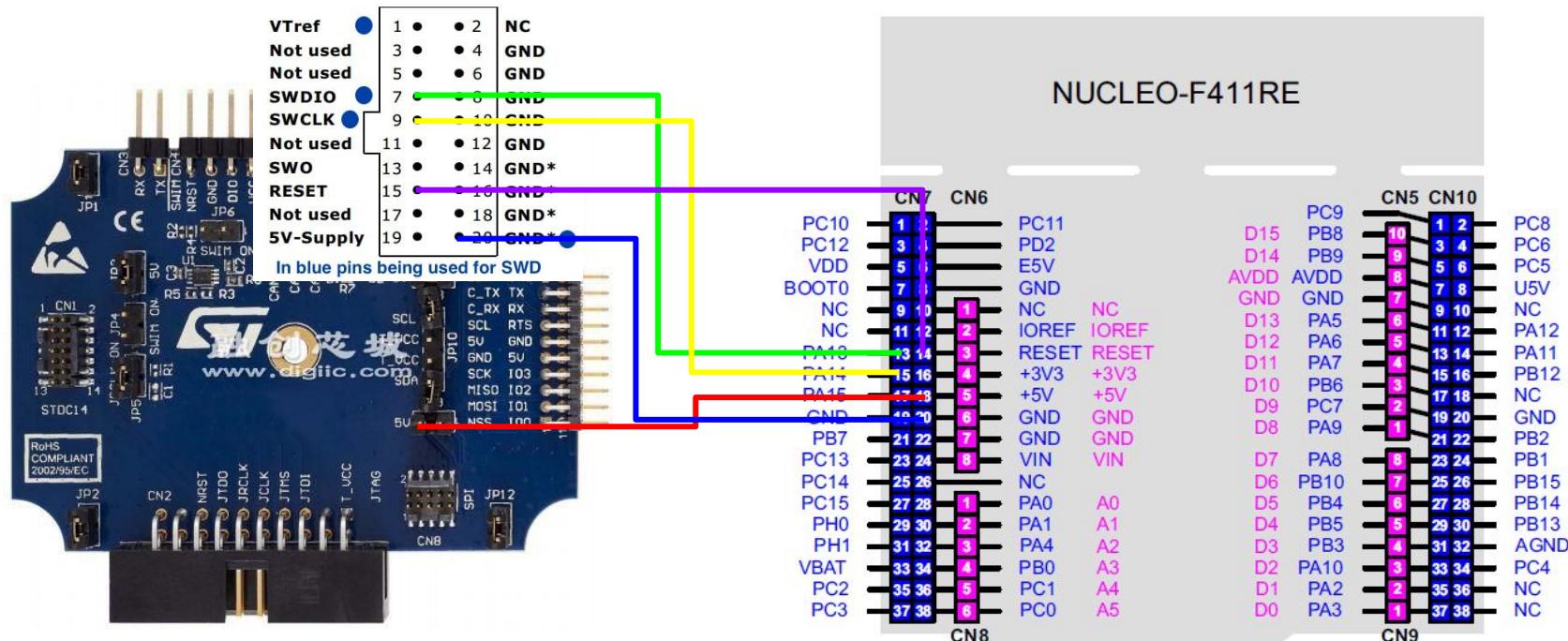


Arduino

Morpho

JPRALVES.NET

ST-Link V3 and F411RE



■ Arduino

■ Morpho

JPRALVES.NET

클럭설정

- RCC (Reset Clock Controller)
 - HSE : High Speed Clock
 - 4~25MHz의 클럭을 외부에 부착하여 동작 클럭으로 사용
 - LSE : Low Speed Clock
 - 낮은 동작을 위한 클럭으로 외부에 32.768KHz 사용
 - RTC를 위한 클럭으로 사용

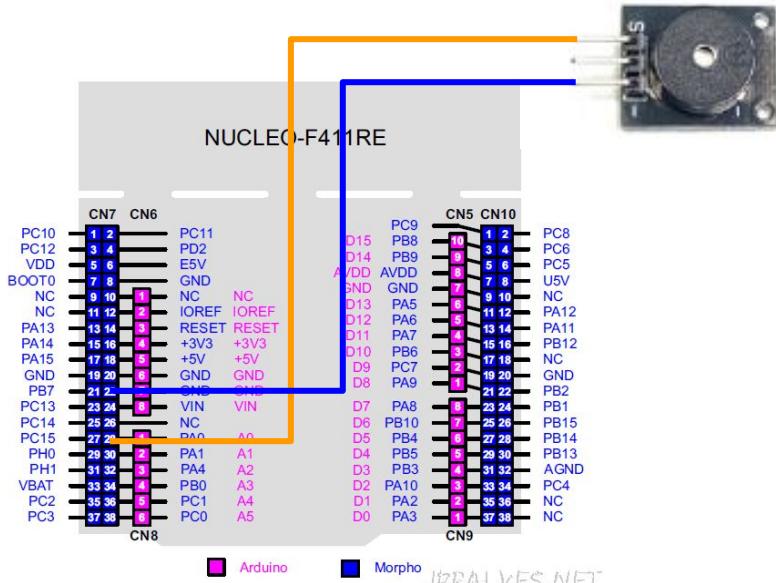
LED 켜기

```
HAL_GPIO_WritePin(Port, Pin, State);
```

```
HAL_GPIO_TogglePin(Port, Pin);
```

Buzzer

- 버저 : 압전소자에 전극을 가하여 흔을 발생시켜 소리를 만들어 낸다
- 능동버저
 - 발진회로를 내장하여 주파수 변경 불가
- 수동버저
 - 발진회로가 없으나 다양한 주파수를 활용 가능



Timer

Counter : 7490 (10진 카운터 0~9), 7493 (16진 카운터 0~15)

Prescaler : 사전 분주비

ARR : Auto Reload Register

카운트가 리셋될 값

CNT : Counter

실제 펄스를 입력 받아 카운트 되는 값

CCR : Channel Compare Register

파형의 기준값(Duty)으로 ARR의 1/2값을 준다

PWM (Pulse Width Modulation)

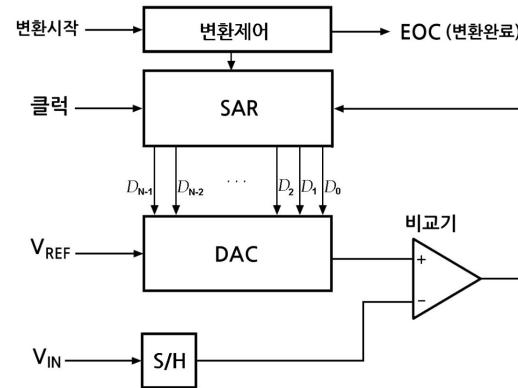
- 주파수설정 : Clock / Prescaler / Counter Period(ARR)
- Duty 설정 : Pulse(CCRx) / Counter Period(ARR) * 100

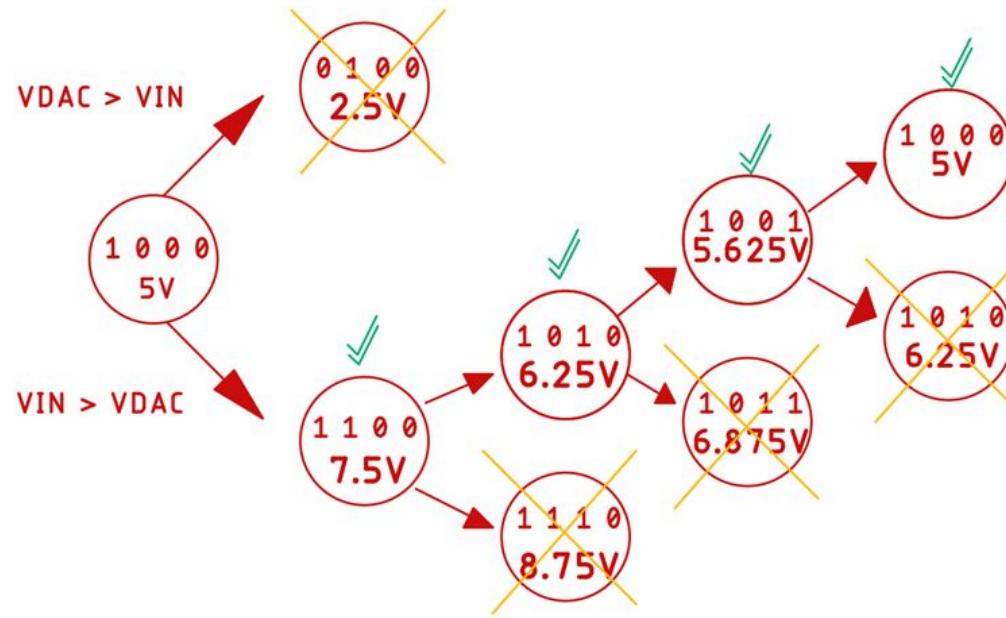
스위치 입력

```
HAL_GPIO_ReadPin();
```

ADC (Analog to Digital Conversion)

- SAR (Successive-approximation ADC) : 축차 비교형 아날로그 디지털 변환으로 이진 탐색 방식으로 양자화하는 방식
- 내부의 DAC와 비교기를 사용하여 각각의 비트에 대한 클럭에 상위(MSB)부터 하위(LSB)로 결정해가는 방식





SAR vs Delta-Sigma

SAR(Successive Approximation Register) ADC와 Delta-Sigma ADC는 각각 다른 원리로 동작하는 아날로그-디지털 변환기입니다. 이 둘의 주요 차이점은 다음과 같습니다.

원리: SAR ADC는 순차 근사 방법을 사용하여 입력 신호를 비교하며, Delta-Sigma ADC는 고차 노이즈 쉐이핑 및 오버샘플링 기술을 사용합니다.

성능: Delta-Sigma ADC는 SAR ADC보다 더 높은 해상도를 제공할 수 있습니다. 그러나 Delta-Sigma ADC는 전력 소비가 높아 SAR ADC보다 전력 효율성이 떨어질 수 있습니다.

입력 범위: SAR ADC는 입력 범위가 작을 수 있지만, Delta-Sigma ADC는 대부분의 입력 범위에서 작동할 수 있습니다.

속도: SAR ADC는 매우 높은 샘플링 속도를 가질 수 있지만, Delta-Sigma ADC는 오버샘플링을 사용하여 상대적으로 느린 샘플링 속도를 가집니다.

가격: 일반적으로 SAR ADC는 더 저렴하고, Delta-Sigma ADC는 더 비싸지만, 고해상도와 더 나은 성능을 제공합니다.

따라서 SAR ADC는 높은 속도와 저전력 소비가 필요한 응용 프로그램에서 유용하며, Delta-Sigma ADC는 고해상도 및 높은 성능이 필요한 응용 프로그램에서 사용됩니다.

Filter

- Low pass filter
 - 급격한 변화를 버리고 미세한 변화만 통과
- Average filter
 - 일정 수만큼의 값을 누적하고 평균을 산출
- Kalman filter
 - 이전 상태를 바탕으로 예측결과를 산출하여 비교 및 보정

Kalman filter

칼만 필터(Kalman filter)는 잡음이 포함되어 있는 측정치를 바탕으로 선형 역학계의 상태를 추정하는 재귀 필터로, 루돌프 칼만이 개발하였다. 칼만 필터는 컴퓨터 비전, 로봇 공학, 레이다 등의 여러 분야에 사용된다. 칼만 필터는 과거에 수행한 측정값을 바탕으로 현재의 상태 변수의 결합분포를 추정한다.

알고리즘은 예측과 업데이트의 두 단계로 이루어진다. 예측 단계에서는 현재 상태 변수의 값과 정확도를 예측한다. 현재 상태 변수의 값이 실제로 측정된 이후, 업데이트 단계에서는 이전에 추정한 상태 변수를 기반으로 예측한 측정치와 실제 측정치의 차이를 반영해 현재의 상태 변수를 업데이트한다.

확장 칼만 필터는 비선형 시스템을 기반으로 동작한다.

```
double Kalman(double measurement) {
    // Kalman filter setup
    static double P = 1.0;
    static double varP = 0.0001; // pow(0.01, 2)
    static double R = 0.25;//pow(0.5, 2);
    static double K = 1.0;
    static double X = 20.0;

    // Kalman Simple Filter
    P = P + varP;
    K = P / (P + R);
    X = (K * measurement) + (1 - K) * X;
    P = (1 - K) * P;

    return X;
}
```

System tick

- 순차적인 실행코드의 Delay()함수를 사용하는 경우 문제점
- ioc파일을 열고 Systick을 설정 후 코드생성
- stm32l0xx_it.c 열기
- void SysTick_Handler()에 CallbackSysTick등록
- main.c에 CallbackSysTick등록

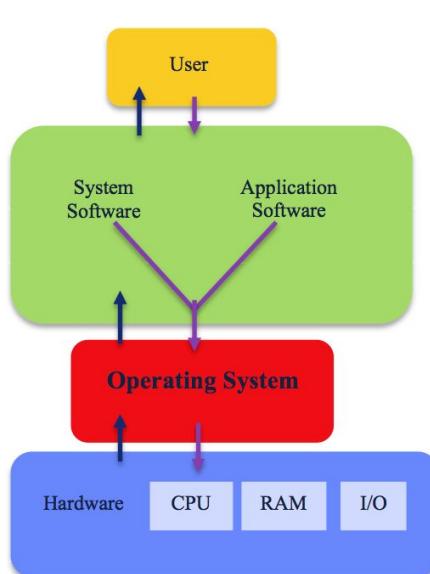
5. 아래코드 작성 USER_CODE_BEGIN_0

```
int count300, count400;
void Callback_SysTick(void) { // 1ms마다 실행
    if(count300 > 0) count300--;
    if(count400 > 0) count400--;
}
```
6. main함수의 while문 내에 다음 코드 추가

```
while(1) {
    if(count300 == 0) {
        count300 = 300;
        // A 행동 추가
    }
    if(count400 == 0) {
        count400 = 400;
        // B 행동 추가
    }
}
```

RTOS (Real Time Operating System)

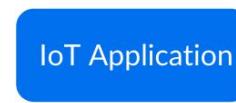
- OS : Windows, Linux(MacOS, iOS, Android)
- RTOS : embedded system



Bare Metal



RTOS



Comparison of commonly used RTOS, ●: true, ○: false, ⓠ: announced but not yet implemented, ⓡ: no information given, ¹: sold separately, ²: SIL-4

freeRTOS

- freeRTOS는
 - FreeRTOS 는 35개의 마이크로컨트롤러 플랫폼 으로 포팅 된 임베디드 장치 용 실시간 운영 체제 커널입니다. MIT 라이선스 에 따라 배포됩니다 .
- 연혁
 - FreeRTOS 커널은 원래 Richard Barry가 2003년경에 개발했으며 나중에 Barry의 회사인 Real Time Engineers Ltd에서 개발 및 유지 관리했습니다. 2017년에 이 회사는 FreeRTOS 프로젝트의 관리 권한을 Amazon Web Services (AWS)에 넘겼습니다. Barry는 AWS 팀의 일원으로 FreeRTOS 작업을 계속하고 있습니다.

FreeRTOS 사용

- main 함수에서 태스크 실행
 - osThreadNew(functionA, 0, &TaskA_attributes);
- main의 loop를 사용하지 않는다
- 기존 HAL_Delay함수대신 지정된 (osDelay, vTaskDelay)함수를 사용한다
- 모든 loop에는 osDelay(1)이상을 포함해야 한다.

WS2812

➤ WS2811

- 8핀 IC
- 5V, 12V 전원 사용 가능
- 외부 별도 LED 부착 필요

➤ WS2812

- 6핀 5x5 패키지

➤ WS2812B

- 4핀 5x5 패키지

Display

- CRT : Cathode Ray Tube (음극 선 관)
- LCD : Liquid Crystal Display (액정 표시 장치)
 - 빛의 편광을 이용(편광 필터)
- OLED : Organic Light Emitting Diode(유기 발광 다이오드)
 - 열에 취약, 번인 현상
- Nano(Micro) LED
 - 반 영구적
- Driver
 - COB : Chip On Board
 - COG : Chip On Glass
 - COF : Chip On Flexible

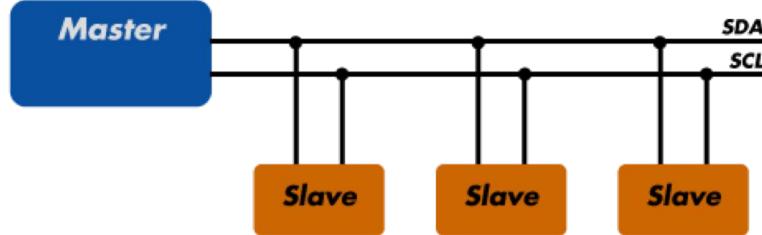
Display Driver IC

- HD44780
 - LCD 문자 표시용 드라이버
 - 히다치
- SSD1306
 - OLED 용 그래픽 드라이버
 - 128 x 64
- SSD1322
 - OLED
 - 256 x 64
- ILI9341
 - Color LCD Graphic Driver

I2C / SMBus

➤ 핀

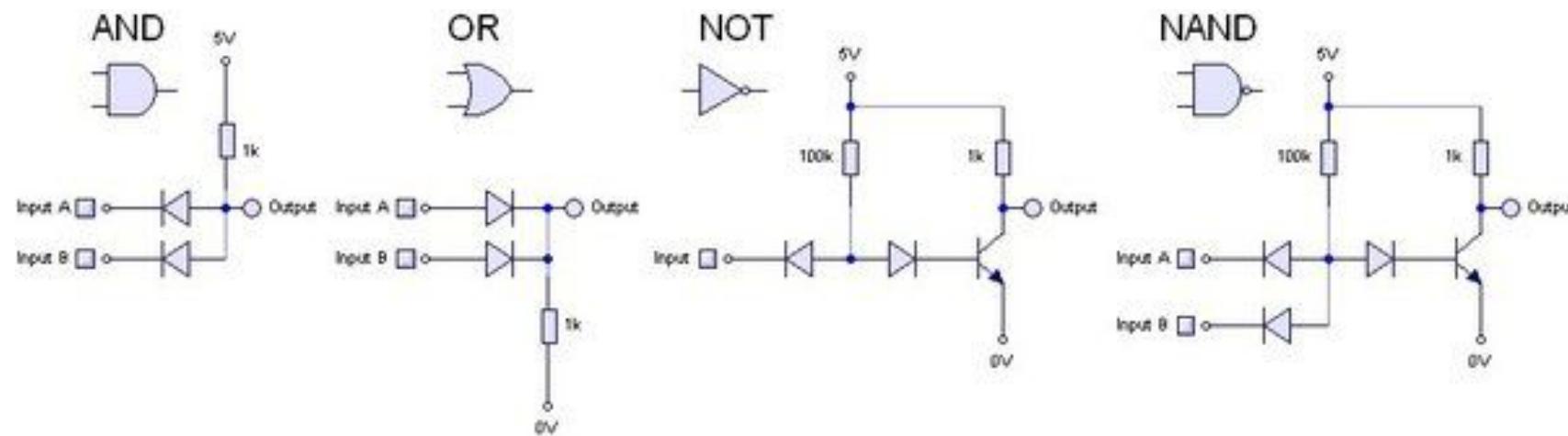
- SDA : Serial DAta
 - 데이터의 전송
 - 양방향
- SCL : Serial CLock
 - 동기용 클럭
 - 단방향(마스터)

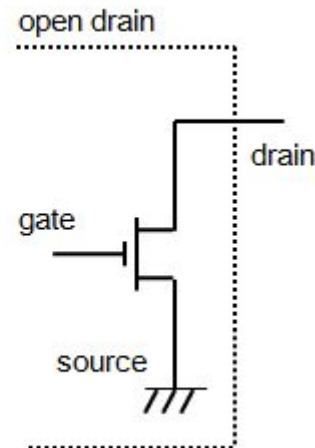
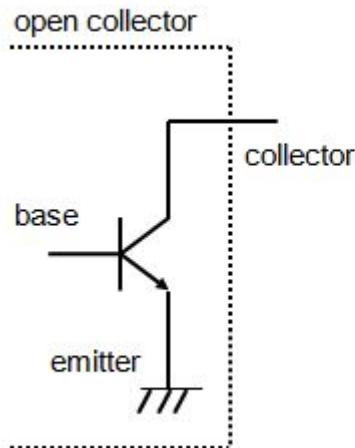
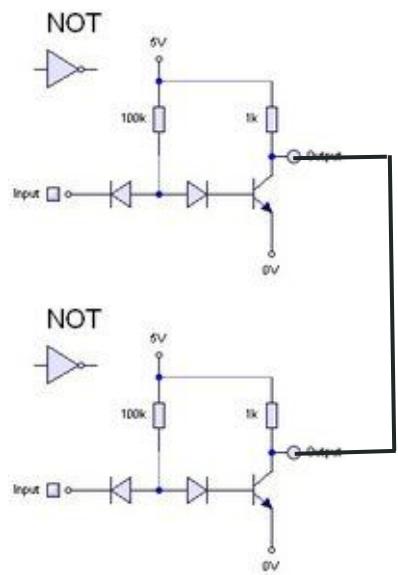


➤ 속도

- 50KHz, 100KHz, 400KHz, 1000KHz
- 속도에 따라 풀업 저항값 변경
 - 10K, 4.7K, 1K, 680

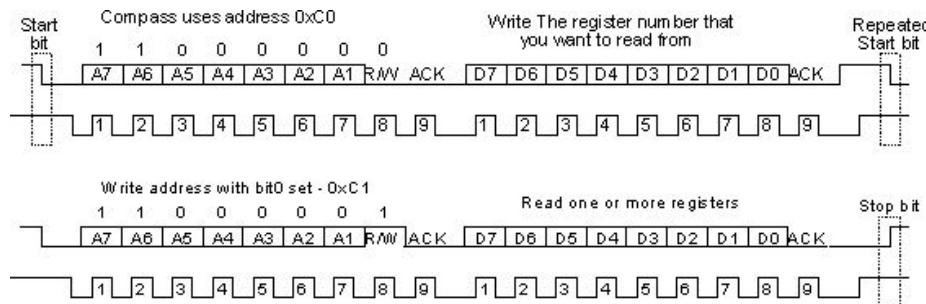
Logic gate with diode and transistor





I2C Protocol

- Idle
 - SDA, SCL은 High 상태
- Start
 - SDA가 Low로 변경 후 SCL이 Low로 변경
- Slave Address + R/W
 - SCL의 동기에 맞춰 SDA 데이터 송신
 - MSB 우선
 - 7Bit Data + R/W 1Bit
 - Acknowledge / No acknowledge
 - 데이터의 수신 측에서 전송
 - Low = Ack, High = Nak
- Data
 - 8Bit의 데이터 전송
 - MSB 우선
 - Acknowledge / No acknowledge
 - 데이터의 수신 측에서 전송
 - Low = Ack, High = Nak
- Stop
 - Start의 역순, SCL이 High로 변경 후 SDA가 High로 변경



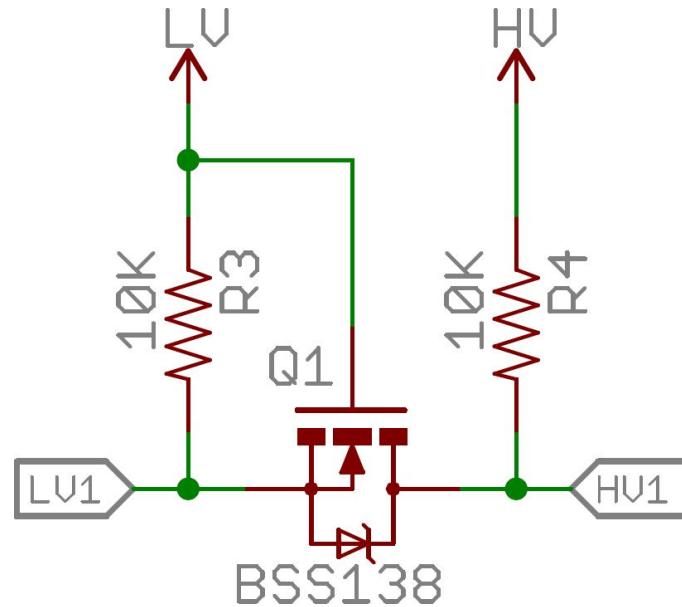
24LC01 (EEPROM, 1KBit = 1024Bit, Byte = 128)

24LC64 (64KBit, 8KByte)

8KByte에 필요한 주소? 8192Byte, 2^{13} , 13Bit가 주소 표현에 필요함

I2C Level Shift

- MOSFET N Channel



Delay Us

타이머1, 1MHz설정

```
void delay_us(uint16_t time) {
    __HAL_TIM_SET_COUNTER(&htim1, 0);
    while((__HAL_TIM_GET_COUNTER(&htim1))<time);
}
```

```
main.c
HAL_TIM_Base_Start(&htim1);
```

DHT11

```
// 온도, 습도 저장하는 전역변수 선언
int Temperature = 0;
int Humidity = 0;

int dht11_read (void) {
    //---- Start Signal 전송
    // 포트를 출력으로 설정
    GPIO_InitTypeDef GPIO_InitStruct = {0};
    GPIO_InitStruct.Pin = DHT11_PIN;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(DHT11_PORT, &GPIO_InitStruct);

    // Low 18ms, High 20us 풀스 생성
    HAL_GPIO_WritePin(DHT11_PORT, DHT11_PIN, 0);
    delay_us(18000);
    HAL_GPIO_WritePin(DHT11_PORT, DHT11_PIN, 1);
    delay_us(20);

    // 포트를 입력으로 설정
    GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    HAL_GPIO_Init(DHT11_PORT, &GPIO_InitStruct);

    //---- DHT11 응답 확인
    delay_us(40); // 40us 대기
    if(!HAL_GPIO_ReadPin(DHT11_PORT, DHT11_PIN)) { // DHT11 응답 체크(Low)
        delay_us(80);
        if(!HAL_GPIO_ReadPin(DHT11_PORT, DHT11_PIN)) // 80us 뒤 DHT11 High 응답 없으면 timeout으로 간주
            return -1;
    }
    if(wait_pulse(GPIO_PIN_RESET) == 0) // 데이터 전송 시작 대기
        return -1; // timeout

    //---- DHT11 데이터 읽기
    uint8_t out[5], i, j;
    for(i = 0; i < 5; i++) {
        // 습도 정수자리, 습도 소수자리, 온도 정수자리, 온도 소수자리, 체크섬 순으로 읽음
        for(j = 0; j < 8; j++) { // 하나의 데이터는 8비트로 구성되며, 최상위 비트부터 하나씩 읽기 시작함
            if(wait_pulse(GPIO_PIN_SET)) // 데이터 전송 시작 까지 대기
                return -1;

            delay_us(40); // 40us 대기 후 High 상태이면 0 수신
            if(!HAL_GPIO_ReadPin(DHT11_PORT, DHT11_PIN)) // Low일 경우 0
                out[i] |= ~(1<<(7-j));
            else
                out[i] |= (1<<(7-j));

            if(!wait_pulse(GPIO_PIN_RESET)) // 다음 데이터 전송 시작 까지 대기
                return -1;
        }
    }

    //---- 체크섬 판별
    if(out[4] != (out[0] + out[1] + out[2] + out[3]))
        return -2;

    //---- 필요 데이터 분리
    Temperature = out[2];
    Humidity = out[0];
}

//---- 필요 데이터 분리
Temperature = out[2];
Humidity = out[0];

return 1;
```

DS18B20

SK6812

- 1bit : 1.25us +- 0.6us, 0.65us ~ 1.85us, 800KHz, 540KHz ~ 1.538MHz
- Low : 0.3us + 0.9us 1/4
- High : 0.6us + 0.6us 2/4
- Reset : 80us

SPI DMA 설정

- Clock : 24MHz
- SPI Clock : 3MHz(3.2MHz), 750KHz(800Khz)
- 1bit 주기 : 1.3us(1.25us)
- Reset 80us : 61개의 4비트 펄스 인가, 80개
- Transmit Only Master
- MOSI, MISO, SCLK, SS(CS)

Display

FND : Flexible Numeric Display, LED를 이용한 숫자 표시장치

Dotmatrix : LED를 Array, 드라이버 내장된 HUB75E

CRT : 음극선관(브라운관)

VFD : Vacuum Fluorescent Display, 진공 형광 표시장치

LCD : Liquid Crystal Display

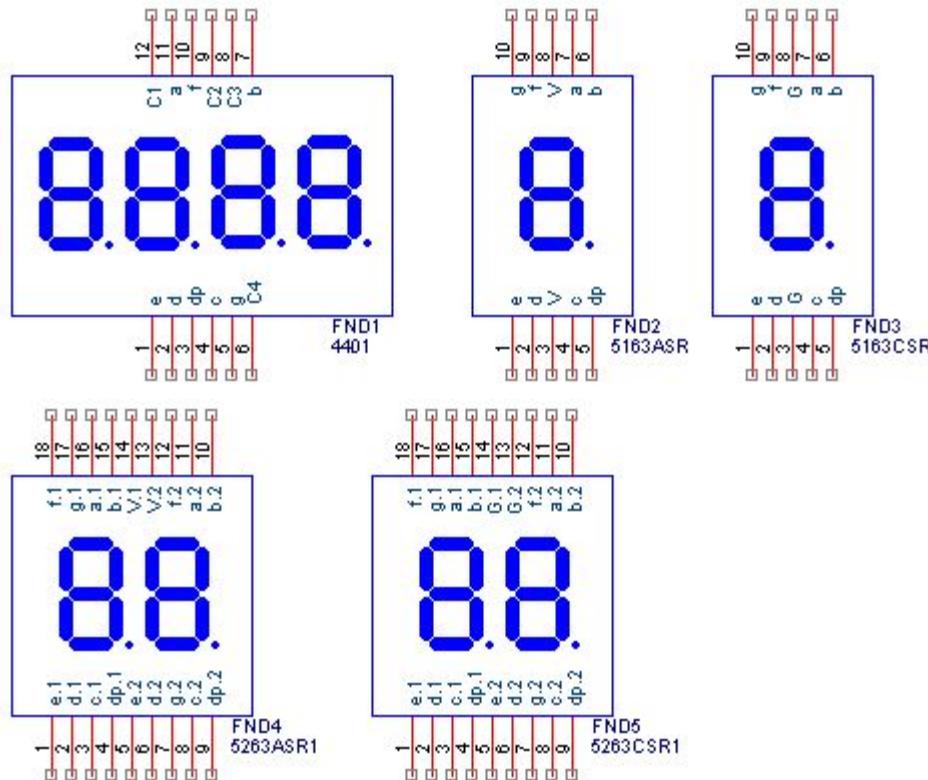
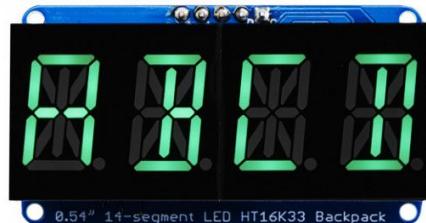
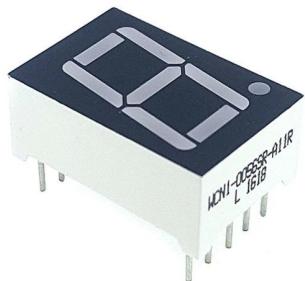
GLCD : Graphic LCD

OLED : Organic LED

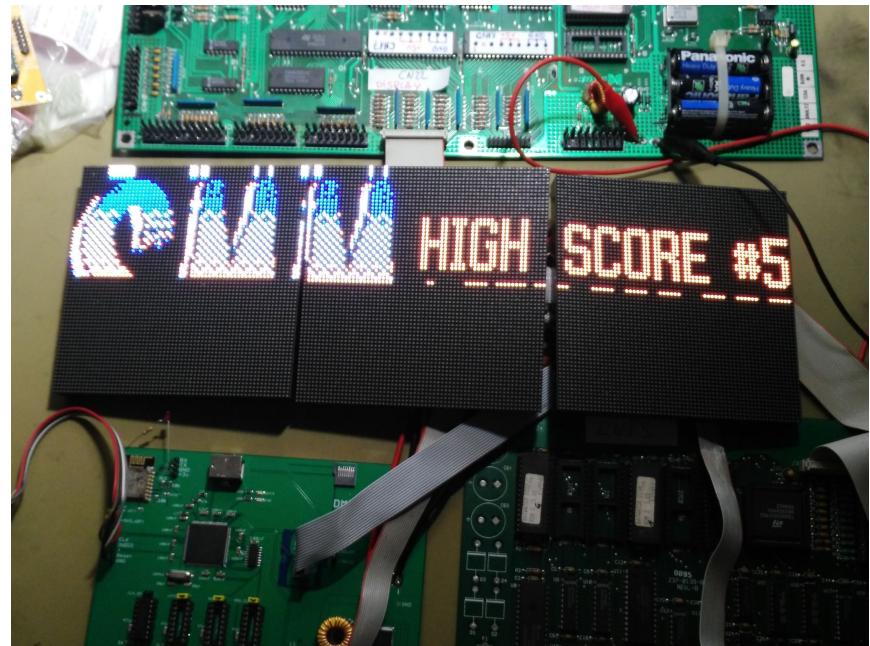
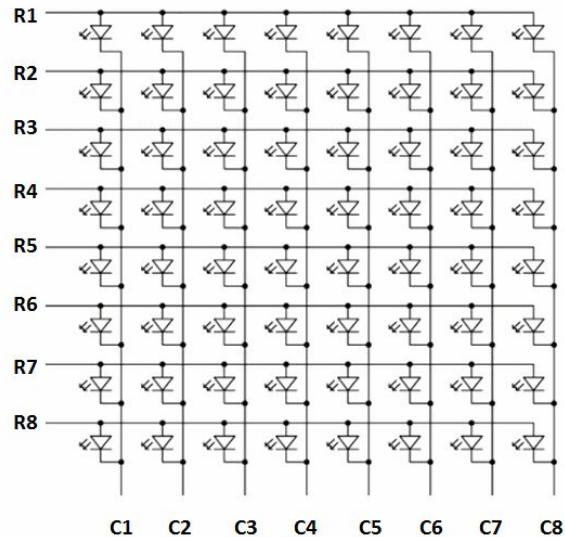
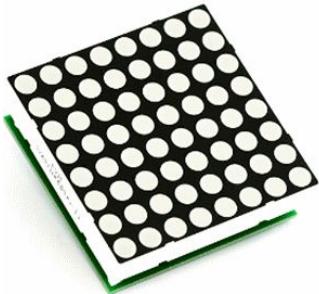
Color LCD

HMI

FND

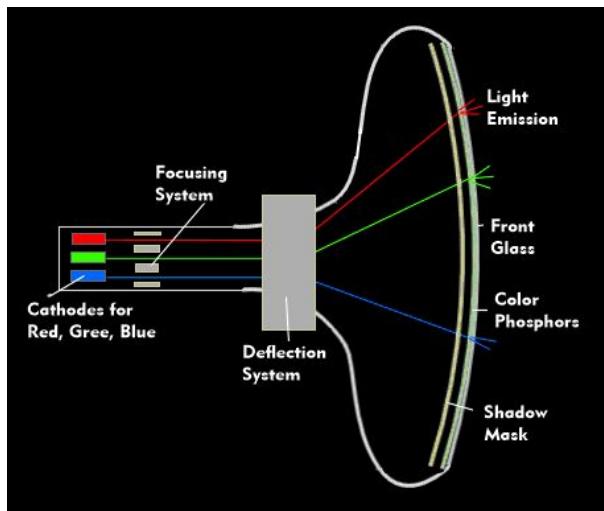


Dotmatrix



CRT

진공의 높은 전압($15kV \sim 30kV$)을 이용

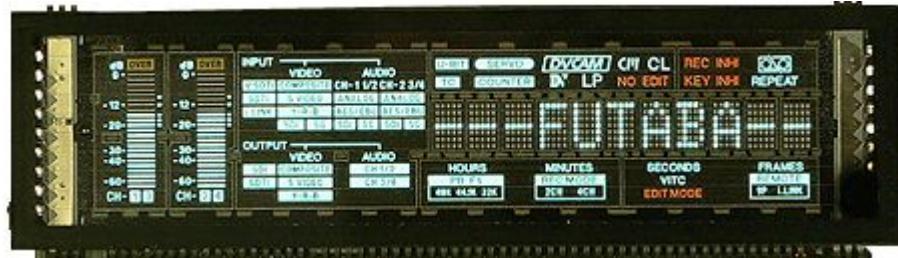


VFD

CRT와 비슷한 원리

CRT보다 낮은 전압

인으로 코팅된 양극과 음극 필라멘트에서 방출되는 전자로 발광



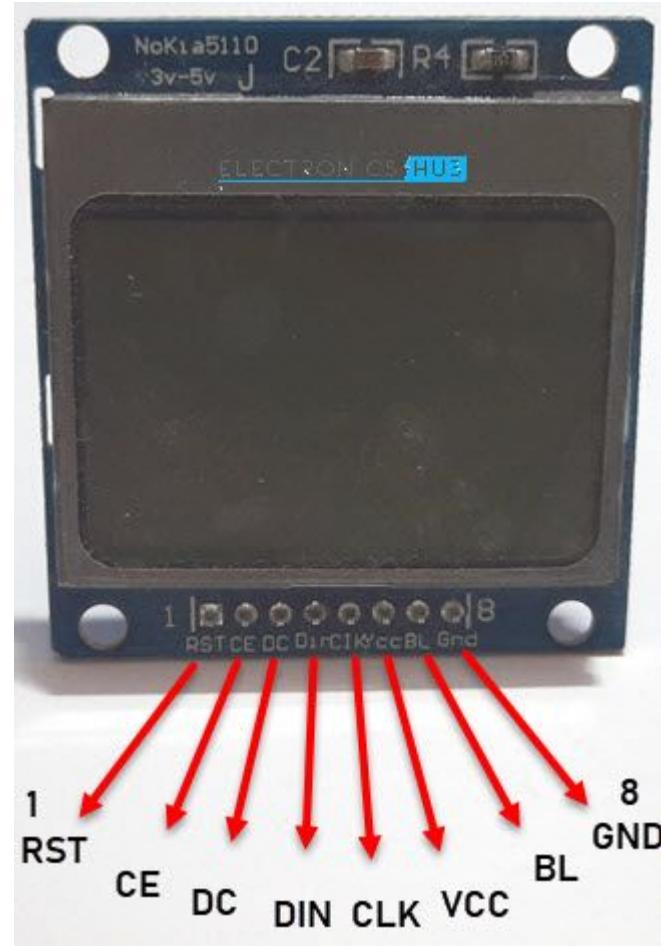
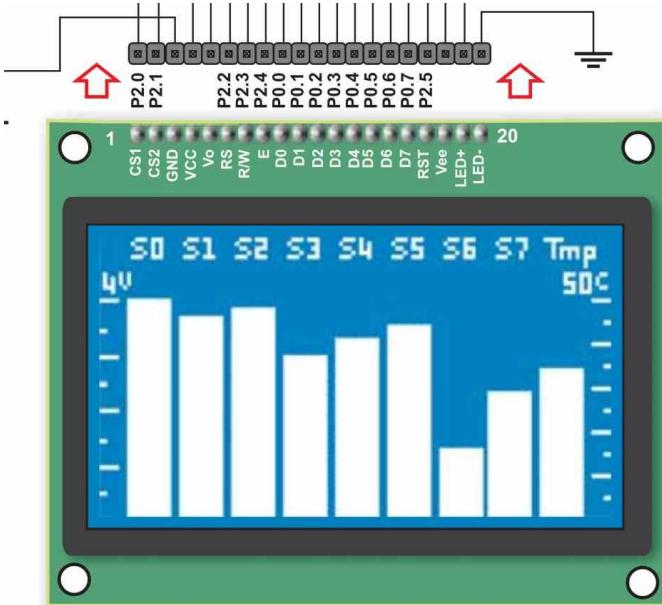
LCD

HD4478

<https://www.sparkfun.com/datasheets/LCD/HD44780.pdf>



GLCD



OLED

SSD1306

<https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf>



Color LCD

ILI9341

<https://cdn-shop.adafruit.com/datasheets/ILI9341.pdf>

ST7735R

<https://www.displayfuture.com/Display/datasheet/controller/ST7735.pdf>



HMI

FT813



Adafruit_GFX

다양한 그래픽 라이브러리

<https://github.com/adafruit/Adafruit-GFX-Library>

font

<https://www.mikroe.com/glcd-font-creator>

영문 vs 한글

인코딩

ASCII, ANSI, ISO8859, KS-X-1001, EUC-KR, ISO-2022-KR, UTF-8

조합형(ks-x-1001) vs 완성형(EUC-KR)

조합형 인코딩

https://ko.wikipedia.org/wiki/%ED%95%9C%EA%B8%80_%EC%83%81%EC%9A%A9_%EC%A1%B0%ED%95%A9%ED%98%95_%EC%9D%B8%EC%BD%94%EB%94%A9

완성형 한글

≈ 20 x14

03 x03

L 04 x04

110100 00011 00100

1101 0000 0110 0100 d0 64

코드	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		
초성	-	채움	ㄱ	ㄲ	ㄴ	ㄷ	ㄸ	ㄹ	ㅁ	ㅂ	ㅃ	ㅅ	ㅆ	ㅇ	ㅈ	ㅉ	ㅊ	ㅋ	ㅌ	ㅍ	ㅎ	-	-	-	-	-	-	-	-	-	-			
중성	-	-	채움	ㅏ	ㅐ	ㅑ	ㅒ	ㅓ	ㅔ	ㅕ	ㅖ	ㅗ	ㅕ	ㅘ	ㅕ	ㅚ	-	-	ㅚ	ㅞ	ㅜ	ㅙ	ㅢ	ㅟ	ㅡ	ㅢ	ㅡ	ㅢ	ㅡ	ㅢ	-	-		
종성	-	채움	ㄱ	ㄲ	ㄳ	ㄴ	ㄵ	ㄶ	ㄷ	ㄹ	ㄺ	ㅁ	ㄻ	ㄮ	ㄹ	ㄱ	ㄲ	ㄳ	㄰	ㄴ	ㄵ	ㄶ	ㄷ	ㄸ	ㄹ	ㄷ	ㄸ	ㄹ	ㄷ	ㄸ	ㄹ	ㄷ	-	-

완성형 > 조합형

Compound Document

<https://www.openoffice.org/sc/compdocfileformat.pdf>

CMSIS (<https://community.st.com/t5/stm32-mcus/configuring-dsp-libraries-on-stm32cubeide/ta-p/49637>)

- 다운로드
 - <https://developer.arm.com/tools-and-software/embedded/cmsis/cmsis-packs>
 - cmsis로 검색 후 “cmsis”를 클릭 후 5.8버전 다운로드
 - ARM CMSIS 5.8.0.pack
- CubeIDE에 파일 추가
 - Software Packs / Manage Software Packs
 - From Local로 파일 추가
- 프로젝트 파일 추가
 - C:\Users\사용자\STM32Cube\Repository\STM32Cube_FW_F4_V1.27.1\Drivers\CMSIS
 - Project / **Drivers / CMSIS** / DSP / Include
 - Project / **Drivers / CMSIS** / Lib
- 프로젝트 설정 추가
 - Properties / C/C++ Build / Settings / Tool Settings / MCU GCC Linker / General
 - Libraries > arm_cortexM4If_math
 - Library search path > ../Drivers/CMSIS/Lib/GCC
 - Properties / C/C++ Build / Settings / Tool Settings / MCU GCC Compiler / Preprocessor > ARM_MATH_CM4
 - Properties / C/C++ Build / Settings / Tool Settings / MCU GCC Compiler / Include paths > ../Drivers/CMSIS/DSP/Include

FFT Test Code

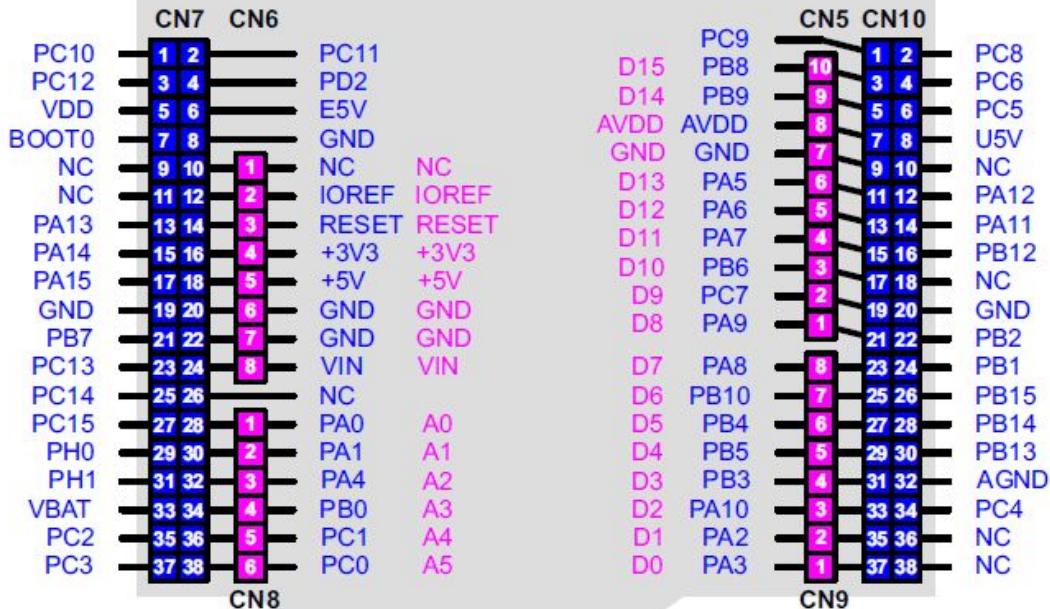
<https://community.st.com/s/article/configuring-dsp-libraries-on-stm32cubeide>

```
/* USER CODE BEGIN PV */  
float32_t FFT_Input_Q15_f[50];  
float32_t aFFT_Input_Q15[50];  
/* USER CODE END PV */  
  
/* USER CODE BEGIN PD */  
#define FFT_Length 1024  
/* USER CODE END PD */
```

Let's take the example of using the “arm_float_to_q15” function:

```
/* USER CODE BEGIN 1 */  
    arm_float_to_q15((float32_t *)&FFT_Input_Q15_f[0], (q15_t *)&aFFT_Input_Q15[0], FFT_Length*2);  
/* USER CODE END 1 */
```

NUCLEO-F411RE



Small Sound

Arduino

Morpho

FFT Test Code2

<https://www.digikey.kr/ko/blog/audio-processing-with-stm32>

FIR vs IIR

FIR과 IIR은 디지털 신호 처리에서 사용되는 필터의 두 가지 일반적인 유형입니다.

FIR 필터는 Finite Impulse Response(유한 임펄스 응답)의 약어로, 입력 신호에 대한 응답이 유한한 길이의 시간만큼 지속되는 필터입니다. 이 필터는 고정된 크기의 임펄스 응답으로 구성되며, 이 응답은 필터의 계수에 의해 결정됩니다. FIR 필터는 일반적으로 안정적이고 선형 상태이며, 대역폭 제한 및 잡음 제거에 많이 사용됩니다.

반면, IIR 필터는 Infinite Impulse Response(무한 임펄스 응답)의 약어로, 입력 신호에 대한 응답이 무한히 계속되는 필터입니다. 이 필터는 현재 및 이전 입력 및 출력 값을 사용하여 계산됩니다. IIR 필터는 FIR 필터보다 계산 비용이 적지만, 일반적으로 FIR 필터보다 불안정하며 비선형 상태를 가질 수 있습니다. IIR 필터는 주로 필터링, 샘플링, 시스템 식별 등에 사용됩니다.

FIR 필터와 IIR 필터 모두 실제 시스템에서 사용됩니다. 디지털 신호 처리의 응용 분야에 따라 적합한 필터를 선택하여 사용해야 합니다.

```

/* USER CODE BEGIN Header */
/**
 * @file          : main.c
 * @brief         : Main program body
 * @attention
 *
 * Copyright (c) 2024 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 */
/* USER CODE END Header */

/* Includes -----*/
#include "main.h"
/* Private includes -----*/
/* USER CODE BEGIN Includes */
#include "uart.h"
/* USER CODE END Includes */
/* Private typedef -----*/
/* USER CODE BEGIN PTD */
/* USER CODE END PTD */
/* Private define -----*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */
/* Private macro -----*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private function prototypes -----*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_TIM1_Init(void);
static void MX_TIM2_Init(void);
static void MX_USART2_UART_Init(void);
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* Private user code -----*/
/* USER CODE BEGIN 0 */
uint32_t IC_Val1 = 0;
uint32_t IC_Val2 = 0;
uint32_t Distance = 0;
uint8_t IsFirstCaptured = 0;
uint8_t DistanceCm = 0;
void delay_us(uint16_t time) {
    htim2.Instance->CNT = 0;
    while(htim3.Instance->CNT < time);
}
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim) {
    if(htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1) {
        if(IsFirstCaptured == 0) {
            IC_Val1 = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_1);
            IsFirstCaptured = 1;
            __HAL_TIM_SET_CAPTUREPOLARITY(htim, TIM_CHANNEL_1, TIM_INPUTCHANNELPOLARITY_FALLING);
        }
        else if(IsFirstCaptured == 1) {
            IC_Val2 = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_1);
            __HAL_TIM_SET_CAPTUREPOLARITY(htim, TIM_CHANNEL_1, TIM_INPUTCHANNELPOLARITY_RISING);
        }
    }
}

```