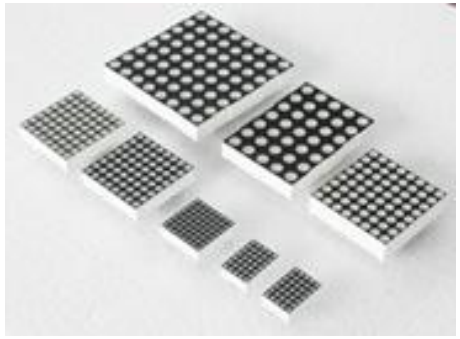
㈜한백전자 교육사업부

Chapter 19
DOT Matrix 제어

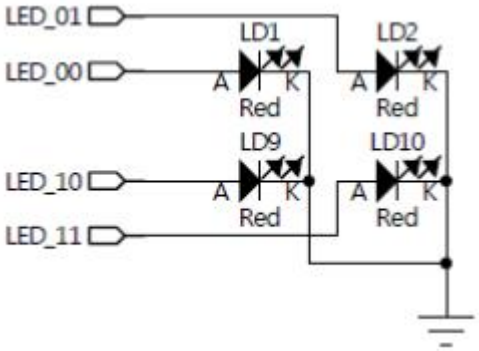
19-1 DOT MATRIX 제어

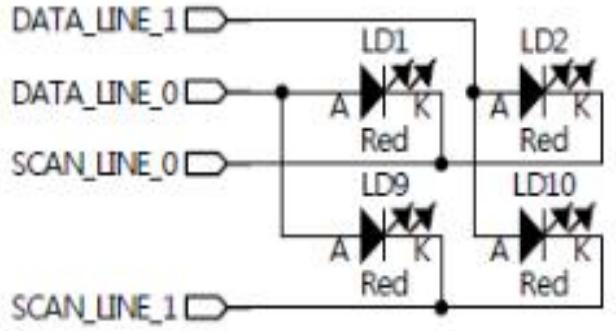
19-1-가 DOT Matrix

- DOT Matrix 모듈의 구조
  - 도트매트릭스 LED의 형태에서 5*7형은 주로 글씨를 표시하기 위한 용도로 주로 사용되며 8*8, 16*16형은 주로 이미지를 표시하기 위해 주로 사용된다.



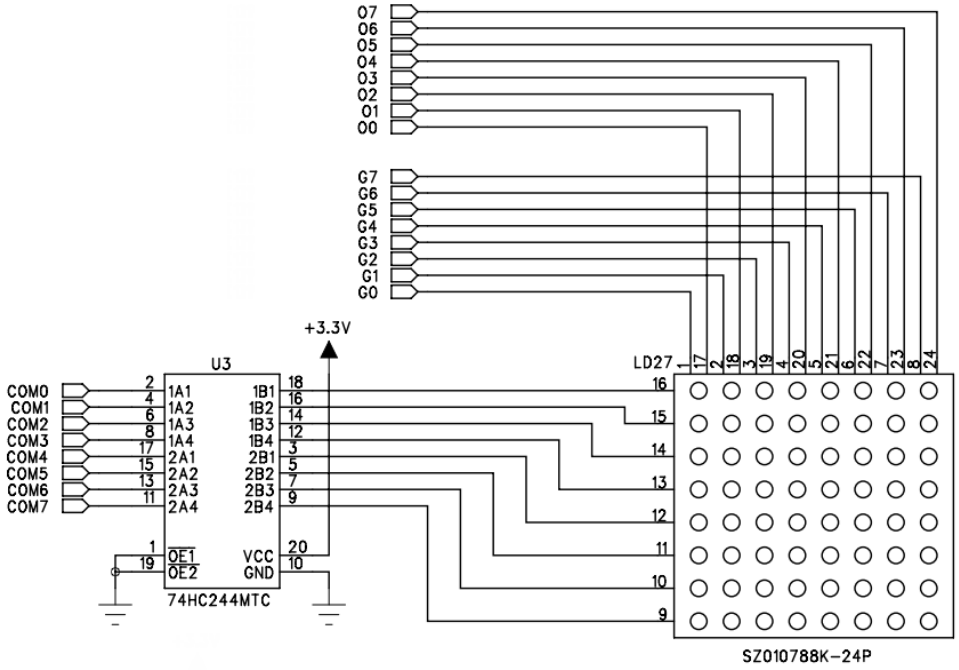
  - 도트매트릭스 LED를 동작하기 위한 구동 회로는 크게 스태틱 구동 방식과 다이나믹 구동 방식으로 나눌 수 있다.





19-1-가 DOT Matrix

- 3 Color Dot Matrix LED의 구동 회로
  - LD1071 IC를 이용하여 3 Color Dot Matrix의 색상 및 점멸을 제어
  - 3 Color LED는 RED와 GREEN의 신호를 이용하여 RED, GREEN, ORANGE 총 3가지 색상을 표현
  - LD1071은 16채널 LED 드라이버 IC로 OUT0 ~ OUT7 의 핀에서 나온 출력으로 3 Color LED의 RED 값을 조절하게 되고 OUT8 ~OUT15까지의 핀에서 나온 출력으로 3 Color LED의 GREEN 값을 조절



19-1-가 DOT Matrix

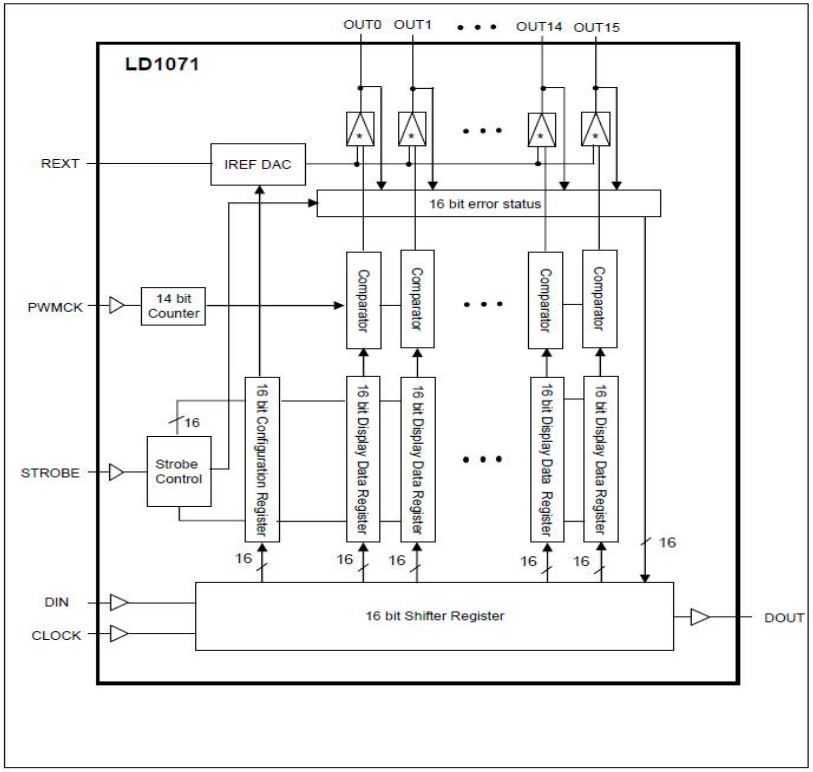
- LED 구동 IC LD1071
  - LD1071은 14비트 PWM을 내장하고 있는 16채널 LED 드라이버로 3 Color Dot Matrix 구동 IC로 널리 사용
  - DIN, CLOCK, STROBE, PWMCK핀을 제어해서 OUT0~15 핀의 출력을 제어
  - 출력이 3 Color Dot Matrix의 핀에 입력 되어서 색상을 조절

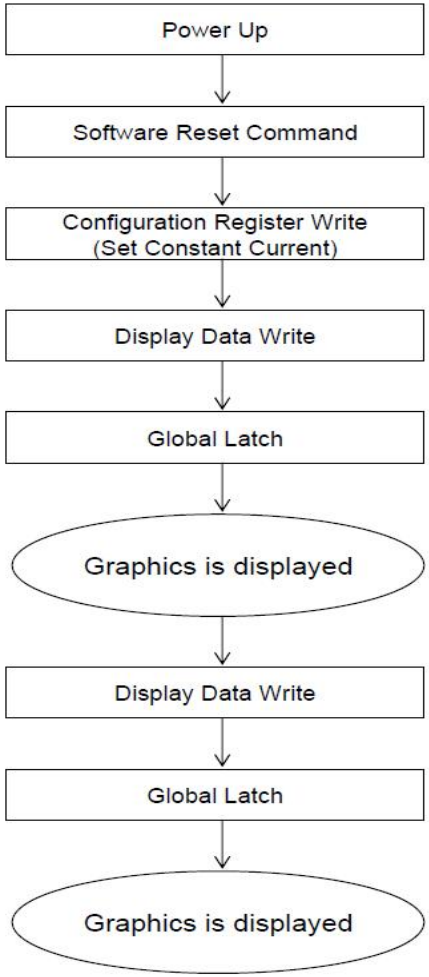
PIN NO.	PIN NAME	DESCRIPTION
1	GND	GND
2	DIN	직렬 데이터 입력
3	CLOCK	DIN으로 직렬 데이터를 받기 위한 쉬프트 입력 클럭 상승에지에서 구동)
4	STROBE	데이터 스트로브 신호로 이 핀은 pull-down 되어 있다.
5~12 13~20	OUTn	전류 출력 n = 0~15
21	PWMCK	PWM 클럭 시그널. 이 핀은 pull-up 으로 되어 있다.
22	DOUT	직렬 데이터 출력
23	REXT	출력핀의 전류를 조절하기 위해 GND와 사이에 저항을 연결하도록 하는 핀이다.
24	VDD	전원 핀 (3.3V/5V)

19-1-가 DOT Matrix

– LD1071 블록도와 제어 순서도

BLOCK DIAGRAM





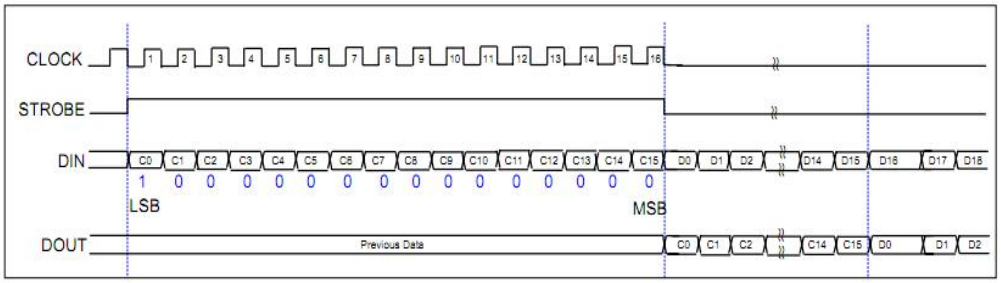
19-1-가 DOT Matrix

- LD1071의 명령어 코드

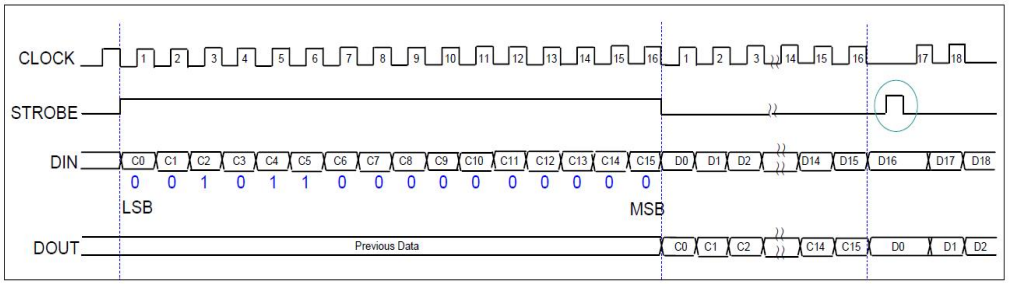
명령어 이름	명령어 코드	동작
Software Reset Command	0001h	이 명령은 전원이 입력된 후에 바로 넣어준다. 이 명령은 Configuration Register, internal shift register, Display Register를 모두 '0'으로 클리어 해 준다.
Configuration Register Write Command	0034h	이 명령은 Configuration Register의 내용을 Update 하는데 사용한다.
Configuration Register Read Command	0038h	이 명령은 Configuration Register의 내용을 읽어내는데 사용한다.
Error Status Read Command	003Ch	이 명령은 LED 에러 상태(Open/Short)를 체크하는데 사용한다.
Display Data Write Command	0084h	이 명령은 이미지를 LED에 디스플레이할 때 사용된다.
Global Latch Command	-	이 명령은 PWM이 출력전류를 만들기 위해서 내부 디스플레이 레지스터의 출력을 Display Data Write Command에 의해 래치시키는데 사용된다. 이 Global Latch Command는 마지막 디스플레이 데이터가 보내준 후에 전송되어야 한다. Global Latch Command의 경우 명령 값이 없으나 이 명령은 실행 과정에서 반드시 필요한 명령이다.

19-1-가 DOT Matrix

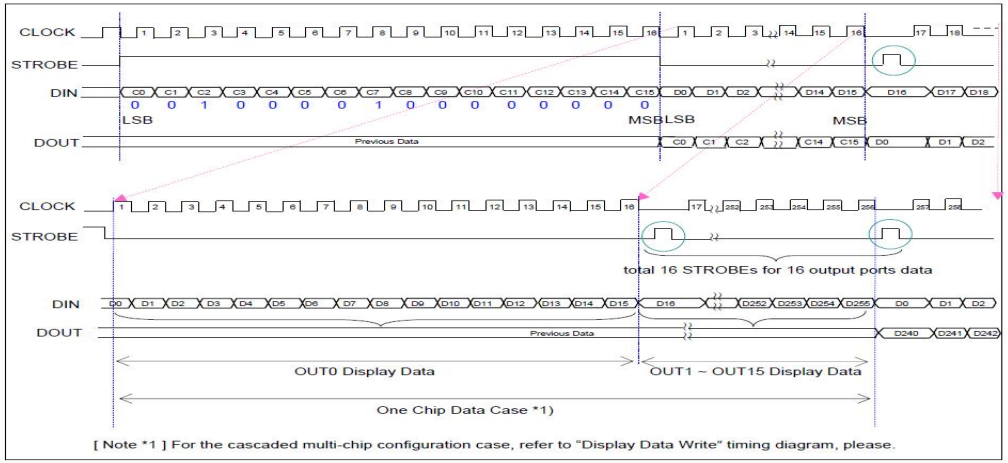
Software Reset Command (0001h)



Configuration Register Write Command (0034h)



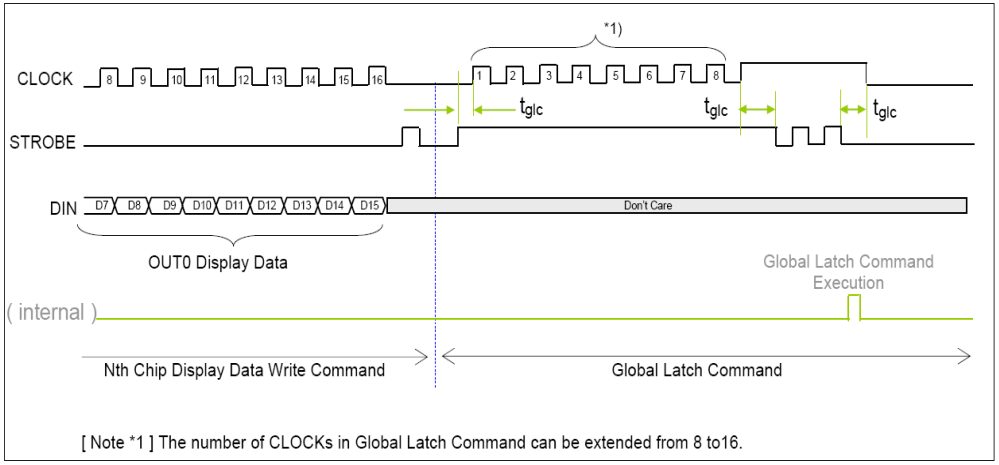
Display Data Write Command (0084h)



[Note *1] For the cascaded multi-chip configuration case, refer to "Display Data Write" timing diagram, please.

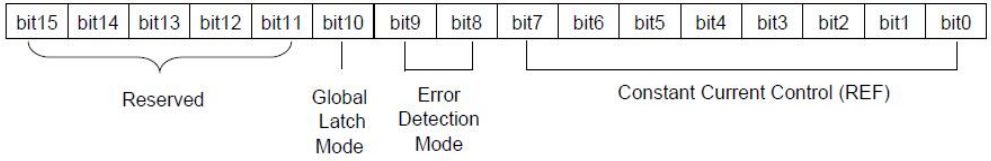
19-1-가 DOT Matrix

Global Latch Command



[Note *1] The number of CLOCKs in Global Latch Command can be extended from 8 to16.

CONFIGURTION REGISTER (CR[15:0])



Constant Current Control **(CR[7:0] Default 00h)**

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Current Value
0	0	0	0	0	0	0	0	IOUT/256
0	0	0	0	0	0	0	1	IOUT*2/256
-	-	-	-	-	-	-	-	-
0	1	1	1	1	1	1	1	IOUT*128/256
-	-	-	-	-	-	-	-	-
1	1	1	1	1	1	1	0	IOUT*255/256
1	1	1	1	1	1	1	1	IOUT

19-1-가 DOT Matrix

- 순서도를 따라가면서 프로그램
  - 전원 입력한다(전원 투입 후 최소한 0.1[ms] 이상의 시간 지연을 주어야 한다).
  - LD1071 IC를 초기화하기 위해 Software Reset Command를 전송해 준다. 위의 타이밍도에 나타나 있는 것과 같이 DIN으로 Software Reset Command (0x0001) 을 전송해 준다. 이때 LSB부터 전송을 하게 된다. T1071을 프로그램하기 위한 기본 함수는 다음과 같이 구성된다.
  - Software Reset Command (0x0001)을 전송해 주는 함수

```
void LD1071_Reset(void)
{
// Software Reset Command를 전송해 준다.
LD1071_Tx_CMD(0x0001);
Delay(100);
}
```

19-1-가 DOT Matrix

– 명령(Command)을 전송해 주는 함수

```
void LD1071_Tx_CMD(unsigned int cmd)
{
unsigned int i = 0;
// Strobe 신호를 'high'=> PORTB.2 = 1;
GPIO_SetBits(GPIOB, LD1071_STB);
// 16비트 데이터인 cmd를 D0부터 1비트씩 직렬로 전송한다.
for( i = 0 ; i< 16 ; i++ )
{
// PORTB.0 = 1;
if(cmd & 0x0001) GPIO_SetBits(GPIOB, LD1071_DIN);
// PORTB.0 = 0;
else GPIO_ResetBits(GPIOB, LD1071_DIN);
// PORTB.1 = 0;
GPIO_ResetBits(GPIOB, LD1071_CLK);
// PORTB.1 = 1;
GPIO_SetBits(GPIOB, LD1071_CLK);
// LSB부터 송신
cmd = cmd >> 1 ;
}
// PORTB.1 = 0;
```

```
// PORTB.0 = 0;
else GPIO_ResetBits(GPIOB, LD1071_DIN);
void LD1071_Tx_CMD(unsigned int cmd)
{
unsigned int i = 0;
// Strobe 신호를 'high'=> PORTB.2 = 1;
GPIO_SetBits(GPIOB, LD1071_STB);
// 16비트 데이터인 cmd를 D0부터 1비트씩 직렬로 전송한다.
for( i = 0 ; i< 16 ; i++ )
{
// PORTB.0 = 1;
if(cmd & 0x0001) GPIO_SetBits(GPIOB, LD1071_DIN);
// PORTB.0 = 0;
else GPIO_ResetBits(GPIOB, LD1071_DIN);
// PORTB.1 = 0;
GPIO_ResetBits(GPIOB, LD1071_CLK);
// PORTB.1 = 1;
GPIO_SetBits(GPIOB, LD1071_CLK);
// LSB부터 송신
cmd = cmd >> 1 ;
}
// PORTB.1 = 0;
GPIO_ResetBits(GPIOB, LD1071_CLK);
// PORTB.2 = 0;
GPIO_ResetBits(GPIOB, LD1071_STB);
}
```

19-1-가 DOT Matrix

- configuation register로 데이터를 전송해 주는 함수

```
void LD1071_Tx_Data(unsigned int data)
{
unsigned int i;
// PORTB.2 = 0;
GPIO_ResetBits(GPIOB, LD1071_STB);
for( i =0 ; i < 16; i++ )
{
// PORTB.0 = 1;
if( data & 0x0001) GPIO_SetBits(GPIOB, LD1071_DIN);
// PORTB.0 = 0;
else GPIO_ResetBits(GPIOB, LD1071_DIN);
// PORTB.1 = 0;
GPIO_ResetBits(GPIOB, LD1071_CLK);
// PORTB.1 = 1;
GPIO_SetBits(GPIOB, LD1071_CLK);
// LSB부터 송신
data = data >> 1;
}
// PORTB.0 = 0;
else GPIO_ResetBits(GPIOB, LD1071_DIN);
// PORTB.1 = 0;
GPIO_ResetBits(GPIOB, LD1071_CLK);
// PORTB.1 = 1;
GPIO_SetBits(GPIOB, LD1071_CLK);
// LSB부터 송신
data = data >> 1;
}
```

19-1-가 DOT Matrix

- 디스플레이로 데이터를 전송해 주는 함수
  - LD1071_Tx_Display(unsigned int data)에서 입력 변수 data의 구성을 보면 다음과 같다.

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
G7	G6	G5	G4	G3	G2	G1	G0	R7	R6	R5	R4	R3	R2	R1	R0

  - LED 위치 별 GREEN 색상 제어 주소

	COM0=1	COM1=1	COM2=1	COM3=1	COM4=1	COM5=1	COM6=1	COM7=1
G0=1	(0,0)	(1,0)	(2,0)	(3,0)	(4,0)	(5,0)	(6,0)	(7,0)
G1=1	(0,1)	(1,1)	(2,1)	(3,1)	(4,1)	(5,1)	(6,1)	(7,1)
G2=1	(0,2)	(1,2)	(2,2)	(3,2)	(4,2)	(5,2)	(6,2)	(7,2)
G3=1	(0,3)	(1,3)	(2,3)	(3,3)	(4,3)	(5,3)	(6,3)	(7,3)
G4=1	(0,4)	(1,4)	(2,4)	(3,4)	(4,4)	(5,4)	(6,4)	(7,4)
G5=1	(0,5)	(1,5)	(2,5)	(3,5)	(4,5)	(5,5)	(6,5)	(7,5)
G6=1	(0,6)	(1,6)	(2,6)	(3,6)	(4,6)	(5,6)	(6,6)	(7,6)
G7=1	(0,7)	(1,7)	(2,7)	(3,7)	(4,7)	(5,7)	(6,7)	(7,7)

19-1-가 DOT Matrix

– LED 위치 별 RED 색상 제어 주소

	COM0=1	COM1=1	COM2=1	COM3=1	COM4=1	COM5=1	COM6=1	COM7=1
R0=1	(0,0)	(1,0)	(2,0)	(3,0)	(4,0)	(5,0)	(6,0)	(7,0)
R1=1	(0,1)	(1,1)	(2,1)	(3,1)	(4,1)	(5,1)	(6,1)	(7,1)
R2=1	(0,2)	(1,2)	(2,2)	(3,2)	(4,2)	(5,2)	(6,2)	(7,2)
R3=1	(0,3)	(1,3)	(2,3)	(3,3)	(4,3)	(5,3)	(6,3)	(7,3)
R4=1	(0,4)	(1,4)	(2,4)	(3,4)	(4,4)	(5,4)	(6,4)	(7,4)
R5=1	(0,5)	(1,5)	(2,5)	(3,5)	(4,5)	(5,5)	(6,5)	(7,5)
R6=1	(0,6)	(1,6)	(2,6)	(3,6)	(4,6)	(5,6)	(6,6)	(7,6)
R7=1	(0,7)	(1,7)	(2,7)	(3,7)	(4,7)	(5,7)	(6,7)	(7,7)

```
// 입력변수 data의 D0~D7까지는 RED 색 데이터 이고 D8~D15까지는
// GREEN색 데이터 이다.
void LD1071_Tx_Display( unsigned int data)
{
unsigned int i , mask = 0x0001;
LD1071_Tx_CMD( 0x0084);
for(i = 0 ; i <16 ; i++)
{
if(data & mask ) LD1071_Tx_Data(0xFFFF);
else LD1071_Tx_Data(0x0000);
mask = mask << 1;
}
LD1071_Global_Latch();
}
```

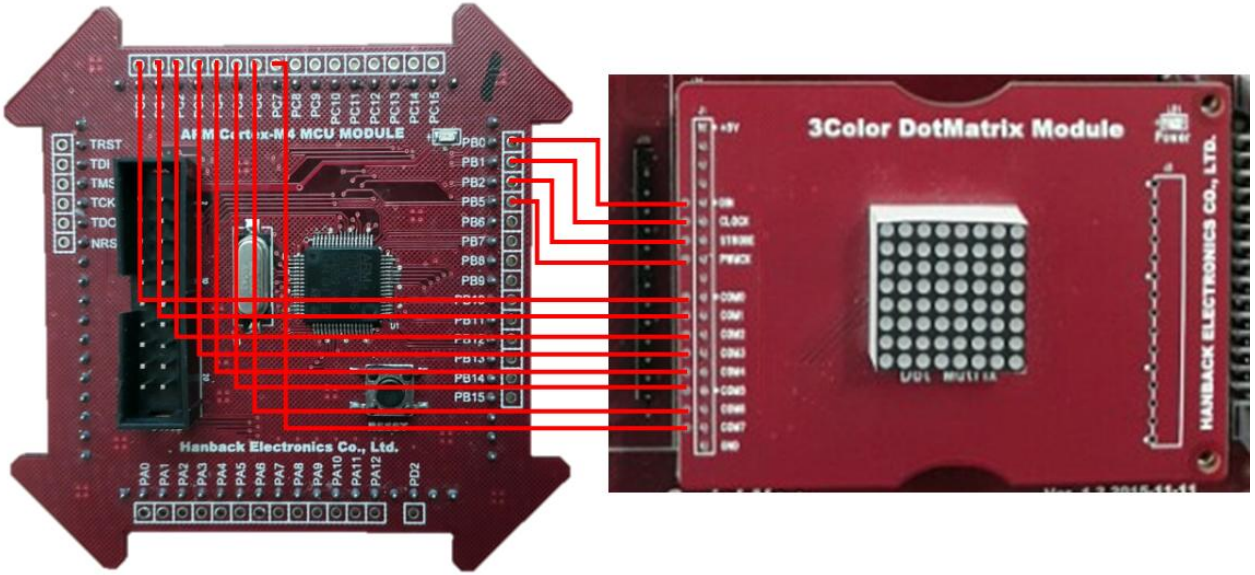
19-1-가 DOT Matrix

- Global Latch Command 함수

```
void LD1071_Global_Latch(void)
{
unsigned int i = 0;
// PORTB.2 = 1;
GPIO_SetBits(GPIOB, LD1071_STB);
// PORTB.0 = 0;
GPIO_ResetBits(GPIOB, LD1071_DIN);
for( i = 0 ; i < 9 ; i++ )
{
// PORTB.1 = 0;
GPIO_ResetBits(GPIOB, LD1071_CLK);
// PORTB.1 = 1;
GPIO_SetBits(GPIOB, LD1071_CLK);
}
// PORTB.2 = 0;
GPIO_ResetBits(GPIOB, LD1071_STB);
// PORTB.2 = 1;
GPIO_SetBits(GPIOB, LD1071_STB);
// PORTB.2 = 0;
GPIO_ResetBits(GPIOB, LD1071_STB);
// PORTB.2 = 1;
GPIO_SetBits(GPIOB, LD1071_STB);
// PORTB.2 = 0;
GPIO_ResetBits(GPIOB, LD1071_STB);
// PORTB.1 = 0;
GPIO_ResetBits(GPIOB, LD1071_CLK);
}
// PORTB.2 = 1;
GPIO_SetBits(GPIOB, LD1071_STB);
// PORTB.2 = 0;
GPIO_ResetBits(GPIOB, LD1071_STB);
// PORTB.2 = 1;
GPIO_SetBits(GPIOB, LD1071_STB);
// PORTB.2 = 0;
GPIO_ResetBits(GPIOB, LD1071_STB);
// PORTB.1 = 0;
GPIO_ResetBits(GPIOB, LD1071_CLK);
}
```

19-1-실험1 : 3 Color DOT Matrix 제어

- 실험 내용 : '0' 번째 줄 (0,0)~(7,0)을 Green 색으로, '1' 번째 줄 (0,1)~(7,1)을 RED로, '2' 번째 줄 (0,2)~(7,2)를 ORANGE 색으로 켜는 프로그램을 작성
- 실험 방법
  - 포트 PB0~PB2, PB5를 DIN, CLOCK, STROBE, PWM 핀에 연결
  - 포트 PC0~PC7까지를 COM0~COM7까지로 연결
  - LD1071 결선



19-1-실험1 : 3 Color DOT Matrix 제어

- 소스 파일 – (1)

```
포트 연결:
1) ARM Cortex-M4 모듈의 포트B (PB0 ~ PB2, PB5)를 4핀 케이블을 이용해서
3Color DotMatrix 모듈의 DIN, CLOCK, STROBE, PWMCK에 연결한다.
2) ARM Cortex-M4 모듈의 포트C (PC0 ~ PC7)를 8핀 케이블을 이용해서
Array FND모듈의 SA_A ~ SA_H에 연결한다. (SA_A가 PC0로 연결돼야 한다.)
3) ARM Cortex-M4 모듈의 포트C (PC8 ~ PC11)를 4핀 케이블을 이용해서
Array FND모듈의 C0 ~ C3에 연결한다. (C0가 PC8로 연결돼야 한다.)
*******************************************************************************/
// stm32f4xx의 각 레지스터들을 정의한 헤더파일
#include "stm32f4xx.h"
#define LD1071_DIN GPIO_Pin_0
#define LD1071_CLK GPIO_Pin_1
#define LD1071_STB GPIO_Pin_2
#define LD1071_PWM GPIO_Pin_5
// delay 함수
static void Delay(const uint32_t Count)
{
__IO uint32_t index = 0;
for(index = (16800 * Count); index != 0; index--);
}
void LD1071_Tx_CMD(unsigned int cmd)
{
unsigned int i = 0;
// Strobe 신호를 'high'=> PORTB.2 = 1;
GPIO_SetBits(GPIOB, LD1071_STB);
// 16비트 데이터인 cmd를 D0부터 1비트씩 직렬로 전송한다.
for( i = 0 ; i< 16 ; i++ )
{
// PORTB.0 = 1;
if(cmd & 0x0001) GPIO_SetBits(GPIOB, LD1071_DIN);
// PORTB.0 = 0;
else GPIO_ResetBits(GPIOB, LD1071_DIN);
// PORTB.1 = 0;
GPIO_ResetBits(GPIOB, LD1071_CLK);
```

19-1-실험1 : 3 Color DOT Matrix 제어

- 소스 파일 – (2)

```
// PORTB.1 = 1;
GPIO_SetBits(GPIOB, LD1071_CLK);
// LSB부터 송신
cmd = cmd >> 1 ;
}
// PORTB.1 = 0;
GPIO_ResetBits(GPIOB, LD1071_CLK);
// PORTB.2 = 0;
GPIO_ResetBits(GPIOB, LD1071_STB);
}
void LD1071_Tx_Data(unsigned int data)
{
unsigned int i;
// PORTB.2 = 0;
GPIO_ResetBits(GPIOB, LD1071_STB);
for( i =0 ; i < 16; i++ )
{
// PORTB.0 = 1;
if( data & 0x0001) GPIO_SetBits(GPIOB, LD1071_DIN);
// PORTB.0 = 0;
else GPIO_ResetBits(GPIOB, LD1071_DIN);
// PORTB.1 = 0;
GPIO_ResetBits(GPIOB, LD1071_CLK);
// PORTB.1 = 1;
GPIO_SetBits(GPIOB, LD1071_CLK);
// LSB부터 송신
data = data >> 1;
}
// PORTB.1 = 0;
GPIO_ResetBits(GPIOB, LD1071_CLK);
// PORTB.2 = 1;
GPIO_SetBits(GPIOB, LD1071_STB);
```

19-1-실험1 : 3 Color DOT Matrix 제어

- 소스 파일 – (3)

```
// PORTB.2 = 0;
GPIO_ResetBits(GPIOB, LD1071_STB);
}
void LD1071_Global_Latch(void)
{
unsigned int i = 0;
// PORTB.2 = 1;
GPIO_SetBits(GPIOB, LD1071_STB);
// PORTB.0 = 0;
GPIO_ResetBits(GPIOB, LD1071_DIN);
for( i = 0 ; i < 9 ; i++ )
{
// PORTB.1 = 0;
GPIO_ResetBits(GPIOB, LD1071_CLK);
// PORTB.1 = 1;
GPIO_SetBits(GPIOB, LD1071_CLK);
}
// PORTB.2 = 0;
GPIO_ResetBits(GPIOB, LD1071_STB);
// PORTB.2 = 1;
GPIO_SetBits(GPIOB, LD1071_STB);
// PORTB.2 = 0;
GPIO_ResetBits(GPIOB, LD1071_STB);
// PORTB.2 = 1;
GPIO_SetBits(GPIOB, LD1071_STB);
// PORTB.2 = 0;
GPIO_ResetBits(GPIOB, LD1071_STB);
// PORTB.1 = 0;
GPIO_ResetBits(GPIOB, LD1071_CLK);
}
```

19-1-실험1 : 3 Color DOT Matrix 제어

- 소스 파일 – (4)

```
void LD1071_Tx_Display( unsigned int data)
{
unsigned int i , mask = 0x0001;
LD1071_Tx_CMD( 0x0084);
for(i = 0 ; i <16 ; i++)
{
if(data & mask ) LD1071_Tx_Data(0xFFFF);
else LD1071_Tx_Data(0x0000);
mask = mask << 1;
}
LD1071_Global_Latch();
}
void LD1071_Reset(void)
{
// Software Reset Command를 전송해 준다
LD1071_Tx_CMD(0x0001);
Delay(100);
}
void Init_Port(void)
{
GPIO_InitTypeDef GPIO_InitStructure;
TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
TIM_OCInitTypeDef TIM_OCInitStructure;
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOB|RCC_AHB1Periph_GPIOC, ENABLE);
RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
// COM0 ~ COM7
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2|GPIO_Pin_3|
GPIO_Pin_4|GPIO_Pin_5|GPIO_Pin_6|GPIO_Pin_7;
GPIO_Init(GPIOC, &GPIO_InitStructure);
```

19-1-실험1 : 3 Color DOT Matrix 제어

- 소스 파일 – (5)

```
// LD1071_DIN, LD1071_CLK, LD1071_STB
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2;
GPIO_Init(GPIOB, &GPIO_InitStructure);
// LD1071_PWM(PB5, TIM3_CH2)
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
GPIO_Init(GPIOB, &GPIO_InitStructure);
//TIM3_CH2
GPIO_PinAFConfig(GPIOB, GPIO_PinSource5, GPIO_AF_TIM3);
// 타이머3 PWM 설정
//(168Mhz/2)/1 = 84MHz
TIM_TimeBaseStructure.TIM_Prescaler = 0;
//(42(41+1) /84MHz = 0.5us), 2MHz
TIM_TimeBaseStructure.TIM_Period = 42-1;
TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;
TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);
TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
TIM_OCInitStructure.TIM_Pulse = 21;
TIM_OC2Init(TIM3, &TIM_OCInitStructure);
//타이머3을 동작시킨다.
TIM_Cmd(TIM3, ENABLE);
}
int main()
{
Init_Port();
LD1071_Reset();
// Configuration Register Update Command
LD1071_Tx_CMD(0x0034);
// Configuration Register
LD1071_Tx_Data(0x04FF);
```

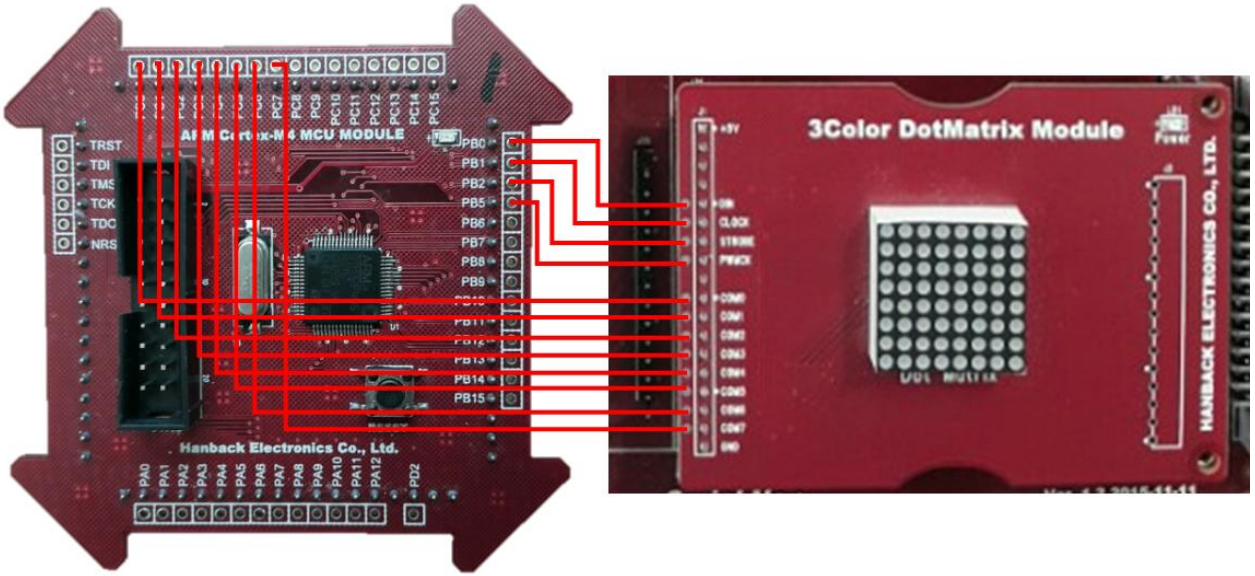
19-1-실험1 : 3 Color DOT Matrix 제어

- 소스 파일 – (6)

```
while(1)
{
LD1071_Tx_Display(0x0506);
GPIO_Write(GPIOC, 0xff);
}
}
```

19-1-실험2 : 3 Color DOT Matrix 제어

- 실험 내용 : Dot Matrix 좌표 (0,0) LED를 GREEN으로, 좌표 (2,0) LED를 RED로, 좌표 (2,4)의 LED를 ORANGE로 켜주는 프로그램을 작성
- 실험 방법
  - 포트 PB0 ~ PB2, PB5를 DIN, CLOCK, STROBE, PWM 핀에 연결
  - 포트 PC0 ~ PC7까지를 COM0 ~ COM7까지로 연결
  - LD1071 결선



19-1-실험2 : 3 Color DOT Matrix 제어

- 소스 파일 – (1)

```
포트 연결:
1) ARM Cortex-M4 모듈의 포트B (PB0 ~ PB2, PB5)를 4핀 케이블을 이용해서
3Color DotMatrix 모듈의 DIN, CLOCK, STROBE, PWMCK에 연결한다.
2) ARM Cortex-M4 모듈의 포트C (PC0 ~ PC7)를 8핀 케이블을 이용해서
Array FND모듈의 SA_A ~ SA_H에 연결한다. (SA_A가 PC0로 연결돼야 한다.)
3) ARM Cortex-M4 모듈의 포트C (PC8 ~ PC11)를 4핀 케이블을 이용해서
Array FND모듈의 C0 ~ C3에 연결한다. (C0가 PC8로 연결돼야 한다.)
*******************************************************************************/
// stm32f4xx의 각 레지스터들을 정의한 헤더파일
#include "stm32f4xx.h"
#define LD1071_DIN GPIO_Pin_0
#define LD1071_CLK GPIO_Pin_1
#define LD1071_STB GPIO_Pin_2
#define LD1071_PWM GPIO_Pin_5
// delay 함수
static void Delay(const uint32_t Count)
{
__IO uint32_t index = 0;
for(index = (16800 * Count); index != 0; index--);
}
void LD1071_Tx_CMD(unsigned int cmd)
{
unsigned int i = 0;
// Strobe 신호를 'high'=> PORTB.2 = 1;
GPIO_SetBits(GPIOB, LD1071_STB);
// 16비트 데이터인 cmd를 D0부터 1비트씩 직렬로 전송한다.
for( i = 0 ; i< 16 ; i++ )
{
// PORTB.0 = 1;
if(cmd & 0x0001) GPIO_SetBits(GPIOB, LD1071_DIN);
// PORTB.0 = 0;
else GPIO_ResetBits(GPIOB, LD1071_DIN);
```

19-1-실험2 : 3 Color DOT Matrix 제어

- 소스 파일 – (2)

```
// PORTB.1 = 0;
GPIO_ResetBits(GPIOB, LD1071_CLK);
// PORTB.1 = 1;
GPIO_SetBits(GPIOB, LD1071_CLK);
// LSB부터 송신
cmd = cmd >> 1 ;
}
// PORTB.1 = 0;
GPIO_ResetBits(GPIOB, LD1071_CLK);
// PORTB.2 = 0;
GPIO_ResetBits(GPIOB, LD1071_STB);
}
void LD1071_Tx_Data(unsigned int data)
{
unsigned int i;
// PORTB.2 = 0;
GPIO_ResetBits(GPIOB, LD1071_STB);
for( i =0 ; i < 16; i++ )
{
// PORTB.0 = 1;
if( data & 0x0001) GPIO_SetBits(GPIOB, LD1071_DIN);
// PORTB.0 = 0;
else GPIO_ResetBits(GPIOB, LD1071_DIN);
// PORTB.1 = 0;
GPIO_ResetBits(GPIOB, LD1071_CLK);
// PORTB.1 = 1;
GPIO_SetBits(GPIOB, LD1071_CLK);
// LSB부터 송신
data = data >> 1;
}
```

19-1-실험2 : 3 Color DOT Matrix 제어

- 소스 파일 – (3)

```
// PORTB.1 = 0;
GPIO_ResetBits(GPIOB, LD1071_CLK);
// PORTB.2 = 1;
GPIO_SetBits(GPIOB, LD1071_STB);
// PORTB.2 = 0;
GPIO_ResetBits(GPIOB, LD1071_STB);
}
void LD1071_Global_Latch(void)
{
unsigned int i = 0;
// PORTB.2 = 1;
GPIO_SetBits(GPIOB, LD1071_STB);
// PORTB.0 = 0;
GPIO_ResetBits(GPIOB, LD1071_DIN);
for( i = 0 ; i < 9 ; i++ )
{
// PORTB.1 = 0;
GPIO_ResetBits(GPIOB, LD1071_CLK);
// PORTB.1 = 1;
GPIO_SetBits(GPIOB, LD1071_CLK);
}
// PORTB.2 = 0;
GPIO_ResetBits(GPIOB, LD1071_STB);
// PORTB.2 = 1;
GPIO_SetBits(GPIOB, LD1071_STB);
// PORTB.2 = 0;
GPIO_ResetBits(GPIOB, LD1071_STB);
// PORTB.2 = 1;
GPIO_SetBits(GPIOB, LD1071_STB);
// PORTB.2 = 0;
GPIO_ResetBits(GPIOB, LD1071_STB);
// PORTB.1 = 0;
GPIO_ResetBits(GPIOB, LD1071_CLK);
}
```

19-1-실험2 : 3 Color DOT Matrix 제어

- 소스 파일 – (4)

```
void LD1071_Tx_Display( unsigned int data)
{
unsigned int i , mask = 0x0001;
LD1071_Tx_CMD( 0x0084);
for(i = 0 ; i <16 ; i++)
{
if(data & mask ) LD1071_Tx_Data(0xFFFF);
else LD1071_Tx_Data(0x0000);
mask = mask << 1;
}
LD1071_Global_Latch();
}
void LD1071_Reset(void)
{
// Software Reset Command를 전송해 준다.
LD1071_Tx_CMD(0x0001);
Delay(100);
}
void Init_Port(void)
{
GPIO_InitTypeDef GPIO_InitStructure;
TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
TIM_OCInitTypeDef TIM_OCInitStructure;
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOB|RCC_AHB1Periph_GPIOC, ENABLE);
RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
```

19-1-실험2 : 3 Color DOT Matrix 제어

- 소스 파일 – (5)

```
// COM0 ~ COM7
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2|GPIO_Pin_3|
GPIO_Pin_4|GPIO_Pin_5|GPIO_Pin_6|GPIO_Pin_7;
GPIO_Init(GPIOC, &GPIO_InitStructure);
// LD1071_DIN, LD1071_CLK, LD1071_STB
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2;
GPIO_Init(GPIOB, &GPIO_InitStructure);
// LD1071_PWM(PB5, TIM3_CH2)
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
GPIO_Init(GPIOB, &GPIO_InitStructure);
//TIM3_CH2
GPIO_PinAFConfig(GPIOB, GPIO_PinSource5, GPIO_AF_TIM3);
// 타이머3 PWM 설정
//(168Mhz/2)/1 = 84MHz
TIM_TimeBaseStructure.TIM_Prescaler = 0;
//(42(41+1) /84MHz = 0.5us), 2MHz
TIM_TimeBaseStructure.TIM_Period = 42-1;
TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;
TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);
TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
TIM_OCInitStructure.TIM_Pulse = 21;
TIM_OC2Init(TIM3, &TIM_OCInitStructure);
//타이머3을 동작시킨다.
TIM_Cmd(TIM3, ENABLE);
}
int main()
{
Init_Port();
LD1071_Reset();
```

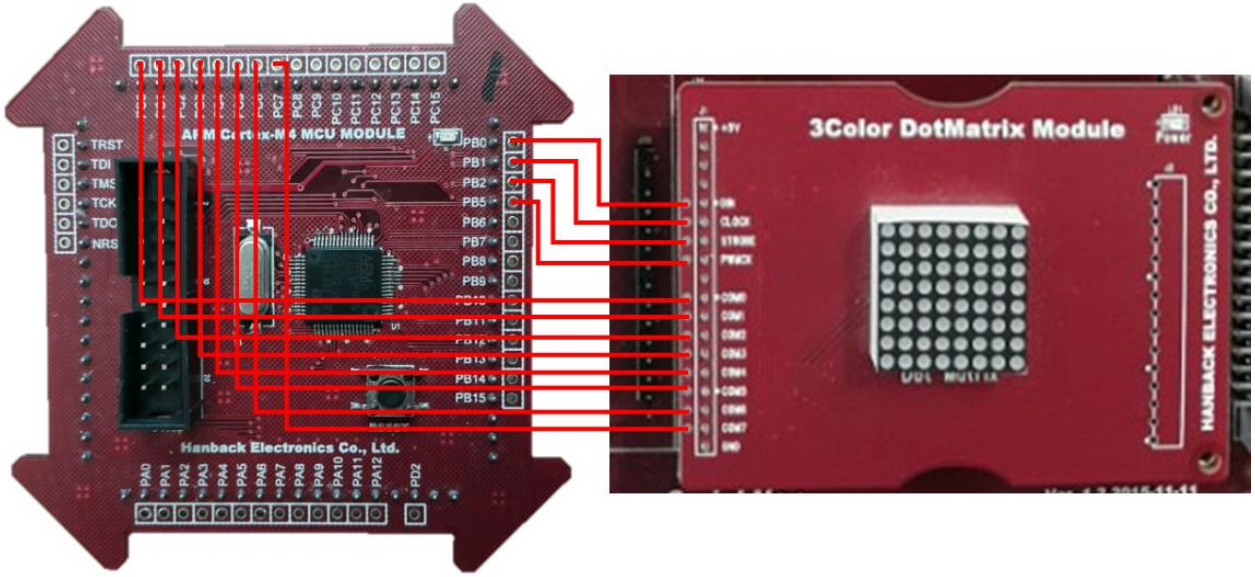
19-1-실험2 : 3 Color DOT Matrix 제어

- 소스 파일 – (6)

```
LD1071_Tx_CMD(0x0034); // Configuration Register Update Command
LD1071_Tx_Data(0x04FF); // Configuration Register
while(1)
{
LD1071_Tx_Display(0x0100); // Dot matrix 좌표 (0,0)을 GREEN
GPIO_Write(GPIOC, 0x01);
LD1071_Tx_Display(0x0001); // Dot matrix 좌표 (2,0)을 RED
GPIO_Write(GPIOC, 0x04);
LD1071_Tx_Display(0x1010); // Dot matrix 좌표 (2,4)를 ORANGE
GPIO_Write(GPIOC, 0x04);
}
}
```

19-1-실험3 : 3 Color DOT Matrix 제어

- 실험 내용 : 실험 방법세로 '0' 번째 줄 (0,0)~(0,7)을 Green 색으로, 세로 '1' 번째 줄 (1,1)~(1,7)을 RED로, 세로 '2' 번째 줄 (2,0) ~(2,7)를 GREEN 색으로, 세로 '3' 번째 줄 (3,0)~(3,7)은 RED로 켜는 프로그램을 작성
- 실험 방법
  - 포트 PB0~PB2, PB5를 DIN, CLOCK, STROBE, PWM 핀에 연결
  - 포트 PC0~PC7까지를 COM0~COM7까지로 연결
  - LD1071 결선



19-1-실험3 : 3 Color DOT Matrix 제어

- 소스 파일 – (1)

```
포트 연결:
1) ARM Cortex-M4 모듈의 포트B (PB0 ~ PB2, PB5)를 4핀 케이블을 이용해서
3Color DotMatrix 모듈의 DIN, CLOCK, STROBE, PWMCK에 연결한다.
2) ARM Cortex-M4 모듈의 포트C (PC0 ~ PC7)를 8핀 케이블을 이용해서
Array FND모듈의 SA_A ~ SA_H에 연결한다. (SA_A가 PC0로 연결돼야 한다.) 3) ARM Cortex-M4 모듈의 포트C (PC8 ~
PC11)를 4핀 케이블을 이용해서
Array FND모듈의 C0 ~ C3에 연결한다. (C0가 PC8로 연결돼야 한다.)
*******************************************************************************/
// stm32f4xx의 각 레지스터들을 정의한 헤더파일
#include "stm32f4xx.h"
#define LD1071_DIN GPIO_Pin_0
#define LD1071_CLK GPIO_Pin_1
#define LD1071_STB GPIO_Pin_2
#define LD1071_PWM GPIO_Pin_5
// delay 함수
static void Delay(const uint32_t Count)
{
__IO uint32_t index = 0;
for(index = (16800 * Count); index != 0; index--);
}
void LD1071_Tx_CMD(unsigned int cmd)
{
unsigned int i = 0;
// Strobe 신호를 'high'=> PORTB.2 = 1;
GPIO_SetBits(GPIOB, LD1071_STB);
// 16비트 데이터인 cmd를 D0부터 1비트씩 직렬로 전송한다.
for( i = 0 ; i< 16 ; i++ )
{
// PORTB.0 = 1;
if(cmd & 0x0001) GPIO_SetBits(GPIOB, LD1071_DIN);
// PORTB.0 = 0;
else GPIO_ResetBits(GPIOB, LD1071_DIN);
```

19-1-실험3 : 3 Color DOT Matrix 제어

- 소스 파일 – (2)

```
// PORTB.1 = 0;
GPIO_ResetBits(GPIOB, LD1071_CLK);
// PORTB.1 = 1;
GPIO_SetBits(GPIOB, LD1071_CLK);
// LSB부터 송신
cmd = cmd >> 1 ;
}
// PORTB.1 = 0;
GPIO_ResetBits(GPIOB, LD1071_CLK);
// PORTB.2 = 0;
GPIO_ResetBits(GPIOB, LD1071_STB);
}
void LD1071_Tx_Data(unsigned int data)
{
unsigned int i;
// PORTB.2 = 0;
GPIO_ResetBits(GPIOB, LD1071_STB);
for( i =0 ; i < 16; i++ )
{
// PORTB.0 = 1;
if( data & 0x0001) GPIO_SetBits(GPIOB, LD1071_DIN);
// PORTB.0 = 0;
else GPIO_ResetBits(GPIOB, LD1071_DIN);
// PORTB.1 = 0;
GPIO_ResetBits(GPIOB, LD1071_CLK);
// PORTB.1 = 1;
GPIO_SetBits(GPIOB, LD1071_CLK);
// LSB부터 송신
data = data >> 1;
}
```

19-1-실험3 : 3 Color DOT Matrix 제어

- 소스 파일 – (3)

```
// PORTB.1 = 0;
GPIO_ResetBits(GPIOB, LD1071_CLK);
// PORTB.2 = 1;
GPIO_SetBits(GPIOB, LD1071_STB);
// PORTB.2 = 0;
GPIO_ResetBits(GPIOB, LD1071_STB);
}
void LD1071_Global_Latch(void)
{
unsigned int i = 0;
// PORTB.2 = 1;
GPIO_SetBits(GPIOB, LD1071_STB);
// PORTB.0 = 0;
GPIO_ResetBits(GPIOB, LD1071_DIN);
for( i = 0 ; i < 9 ; i++ )
{
// PORTB.1 = 0;
GPIO_ResetBits(GPIOB, LD1071_CLK);
// PORTB.1 = 1;
GPIO_SetBits(GPIOB, LD1071_CLK);
}
// PORTB.2 = 0;
GPIO_ResetBits(GPIOB, LD1071_STB);
// PORTB.2 = 1;
GPIO_SetBits(GPIOB, LD1071_STB);
// PORTB.2 = 0;
GPIO_ResetBits(GPIOB, LD1071_STB);
// PORTB.2 = 1;
GPIO_SetBits(GPIOB, LD1071_STB);
// PORTB.2 = 0;
GPIO_ResetBits(GPIOB, LD1071_STB);
// PORTB.1 = 0;
GPIO_ResetBits(GPIOB, LD1071_CLK);
}
```

19-1-실험3 : 3 Color DOT Matrix 제어

- 소스 파일 – (4)

```
void LD1071_Tx_Display( unsigned int data)
{
unsigned int i , mask = 0x0001;
LD1071_Tx_CMD( 0x0084);
for(i = 0 ; i <16 ; i++)
{
if(data & mask ) LD1071_Tx_Data(0xFFFF);
else LD1071_Tx_Data(0x0000);
mask = mask << 1;
}
LD1071_Global_Latch();
}
void LD1071_Reset(void)
{
// Software Reset Command를 전송해 준다.
LD1071_Tx_CMD(0x0001);
Delay(100);
}
void Init_Port(void)
{
GPIO_InitTypeDef GPIO_InitStructure;
TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
TIM_OCInitTypeDef TIM_OCInitStructure;
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOB|RCC_AHB1Periph_GPIOC, ENABLE);
RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
// COM0 ~ COM7
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2|GPIO_Pin_3|
GPIO_Pin_4|GPIO_Pin_5|GPIO_Pin_6|GPIO_Pin_7;
GPIO_Init(GPIOC, &GPIO_InitStructure);
```

19-1-실험3 : 3 Color DOT Matrix 제어

- 소스 파일 – (5)

```
// LD1071_DIN, LD1071_CLK, LD1071_STB
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2;
GPIO_Init(GPIOB, &GPIO_InitStructure);
// LD1071_PWM(PB5, TIM3_CH2)
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
GPIO_Init(GPIOB, &GPIO_InitStructure);
//TIM3_CH2
GPIO_PinAFConfig(GPIOB, GPIO_PinSource5, GPIO_AF_TIM3);
// 타이머3 PWM 설정
//(168Mhz/2)/1 = 84MHz
TIM_TimeBaseStructure.TIM_Prescaler = 0;
//(42(41+1) /84MHz = 0.5us), 2MHz
TIM_TimeBaseStructure.TIM_Period = 42-1;
TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;
TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);
TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
TIM_OCInitStructure.TIM_Pulse = 21;
TIM_OC2Init(TIM3, &TIM_OCInitStructure);
//타이머3을 동작시킨다.
TIM_Cmd(TIM3, ENABLE);
}
int main()
{
Init_Port();
LD1071_Reset();
// Configuration Register Update Command
LD1071_Tx_CMD(0x0034);
// Configuration Register
LD1071_Tx_Data(0x04FF);
```

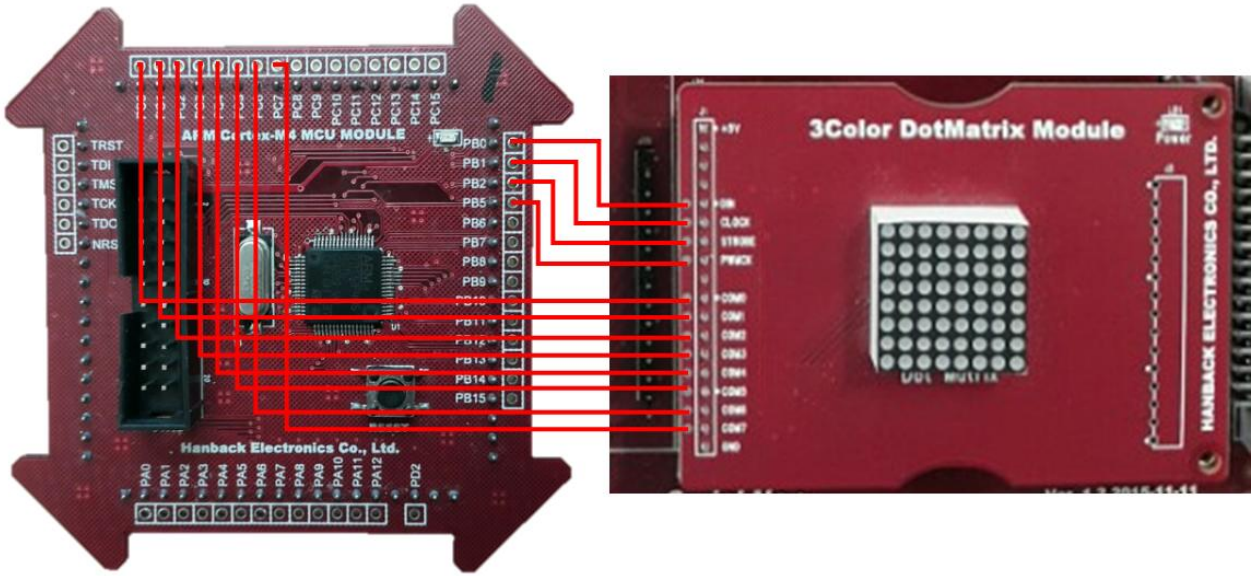
19-1-실험3 : 3 Color DOT Matrix 제어

- 소스 파일 – (6)

```
while(1)
{
// 세로 '0'번째 줄이 GREEN
LD1071_Tx_Display(0xff00);
GPIO_Write(GPIOC, 0x01);
// 세로 '1'번째 줄이 RED
LD1071_Tx_Display(0x00ff);
GPIO_Write(GPIOC, 0x02);
// 세로 '2'번째 줄이 GREEN
LD1071_Tx_Display(0xff00);
GPIO_Write(GPIOC, 0x04);
// 세로 '3'번째 줄이 RED
LD1071_Tx_Display(0x00ff);
GPIO_Write(GPIOC, 0x08);
}
}
```

19-1-실험4 : 3 Color DOT Matrix 제어

- 실험 내용 : Dot matrix 각 LED를 제어하여 아래 그림과 같이 나타나도록 하시오
- 실험 방법
  - 포트 PB0 ~ PB2, PB5를 DIN, CLOCK, STROBE, PWM 핀에 연결
  - 포트 PC0 ~ PC7까지를 COM0 ~ COM7까지로 연결
  - LD1071 결선



19-1-실험4 : 3 Color DOT Matrix 제어

G (0,0)	G (1,0)	G (2,0)	G (3,0)	R (4,0)	R (5,0)	G (6,0)	G (7,0)
G (0,1)	G (1,1)	G (2,1)	G (3,1)	R (4,1)	R (5,1)	G (6,1)	G (7,1)
R (0,2)	R (1,2)	R (2,2)	R (3,2)	G (4,2)	G (5,2)	G (6,2)	G (7,2)
R (0,3)	R (1,3)	R (2,3)	R (3,3)	G (4,3)	G (5,3)	G (6,3)	G (7,3)
G (0,4)	G (1,4)	G (2,4)	G (3,4)	R (4,4)	R (5,4)	R (6,4)	R (7,4)
G (0,5)	G (1,5)	G (2,5)	G (3,5)	R (4,5)	R (5,5)	R (6,5)	R (7,5)
G (0,6)	G (1,6)	R (2,6)	R (3,6)	G (4,6)	G (5,6)	G (6,6)	G (7,6)
G (0,7)	G (1,7)	R (2,7)	R (3,7)	G (4,7)	G (5,7)	G (6,7)	G (7,7)

19-1-실험4 : 3 Color DOT Matrix 제어

- 소스 파일 – (1)

```
포트 연결:
1) ARM Cortex-M4 모듈의 포트B (PB0 ~ PB2, PB5)를 4핀 케이블을 이용해서
3Color DotMatrix 모듈의 DIN, CLOCK, STROBE, PWMCK에 연결한다.
2) ARM Cortex-M4 모듈의 포트C (PC0 ~ PC7)를 8핀 케이블을 이용해서
Array FND모듈의 SA_A ~ SA_H에 연결한다. (SA_A가 PC0로 연결돼야 한다.)
3) ARM Cortex-M4 모듈의 포트C (PC8 ~ PC11)를 4핀 케이블을 이용해서
Array FND모듈의 C0 ~ C3에 연결한다. (C0가 PC8로 연결돼야 한다.)
*******************************************************************************/
// stm32f4xx의 각 레지스터들을 정의한 헤더파일
#include "stm32f4xx.h"
#define LD1071_DIN GPIO_Pin_0
#define LD1071_CLK GPIO_Pin_1
#define LD1071_STB GPIO_Pin_2
#define LD1071_PWM GPIO_Pin_5
// delay 함수
static void Delay(const uint32_t Count)
{
__IO uint32_t index = 0;
for(index = (16800 * Count); index != 0; index--);
}
void LD1071_Tx_CMD(unsigned int cmd)
{
unsigned int i = 0;
// Strobe 신호를 'high'=> PORTB.2 = 1;
GPIO_SetBits(GPIOB, LD1071_STB);
// 16비트 데이터인 cmd를 D0부터 1비트씩 직렬로 전송한다.
for( i = 0 ; i< 16 ; i++ )
{
// PORTB.0 = 1;
if(cmd & 0x0001) GPIO_SetBits(GPIOB, LD1071_DIN);
```

19-1-실험4 : 3 Color DOT Matrix 제어

- 소스 파일 – (2)

```
// PORTB.0 = 0;
else GPIO_ResetBits(GPIOB, LD1071_DIN);
// PORTB.1 = 0;
GPIO_ResetBits(GPIOB, LD1071_CLK);
// PORTB.1 = 1;
GPIO_SetBits(GPIOB, LD1071_CLK);
// LSB부터 송신
cmd = cmd >> 1 ;
}
// PORTB.1 = 0;
GPIO_ResetBits(GPIOB, LD1071_CLK);
// PORTB.2 = 0;
GPIO_ResetBits(GPIOB, LD1071_STB);
}
void LD1071_Tx_Data(unsigned int data)
{
unsigned int i;
// PORTB.2 = 0;
GPIO_ResetBits(GPIOB, LD1071_STB);
for( i =0 ; i < 16; i++ )
{
// PORTB.0 = 1;
if( data & 0x0001) GPIO_SetBits(GPIOB, LD1071_DIN);
// PORTB.0 = 0;
else GPIO_ResetBits(GPIOB, LD1071_DIN);
// PORTB.1 = 0;
GPIO_ResetBits(GPIOB, LD1071_CLK);
// PORTB.1 = 1;
GPIO_SetBits(GPIOB, LD1071_CLK);
// LSB부터 송신
data = data >> 1;
}
```

19-1-실험4 : 3 Color DOT Matrix 제어

- 소스 파일 – (3)

```
// PORTB.1 = 0;
GPIO_ResetBits(GPIOB, LD1071_CLK);
// PORTB.2 = 1;
GPIO_SetBits(GPIOB, LD1071_STB);
// PORTB.2 = 0;
GPIO_ResetBits(GPIOB, LD1071_STB);
}
void LD1071_Global_Latch(void)
{
unsigned int i = 0;
// PORTB.2 = 1;
GPIO_SetBits(GPIOB, LD1071_STB);
// PORTB.0 = 0;
GPIO_ResetBits(GPIOB, LD1071_DIN);
for( i = 0 ; i < 9 ; i++ )
{
// PORTB.1 = 0;
GPIO_ResetBits(GPIOB, LD1071_CLK);
// PORTB.1 = 1;
GPIO_SetBits(GPIOB, LD1071_CLK);
}
// PORTB.2 = 0;
GPIO_ResetBits(GPIOB, LD1071_STB);
// PORTB.2 = 1;
GPIO_SetBits(GPIOB, LD1071_STB);
// PORTB.2 = 0;
GPIO_ResetBits(GPIOB, LD1071_STB);
// PORTB.2 = 1;
GPIO_SetBits(GPIOB, LD1071_STB);
// PORTB.2 = 0;
GPIO_ResetBits(GPIOB, LD1071_STB);
// PORTB.1 = 0;
GPIO_ResetBits(GPIOB, LD1071_CLK);
}
```

19-1-실험4 : 3 Color DOT Matrix 제어

- 소스 파일 – (4)

```
void LD1071_Tx_Display( unsigned int data)
{
unsigned int i , mask = 0x0001;
LD1071_Tx_CMD( 0x0084);
for(i = 0 ; i <16 ; i++)
{
if(data & mask ) LD1071_Tx_Data(0xFFFF);
else LD1071_Tx_Data(0x0000);
mask = mask << 1;
}
LD1071_Global_Latch();
}
void LD1071_Reset(void)
{
// Software Reset Command를 전송해 준다.
LD1071_Tx_CMD(0x0001);
Delay(100);
}
void Init_Port(void)
{
GPIO_InitTypeDef GPIO_InitStructure;
TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
TIM_OCInitTypeDef TIM_OCInitStructure;
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOB|RCC_AHB1Periph_GPIOC, ENABLE);
RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
// COM0 ~ COM7
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2|GPIO_Pin_3|
GPIO_Pin_4|GPIO_Pin_5|GPIO_Pin_6|GPIO_Pin_7;
GPIO_Init(GPIOC, &GPIO_InitStructure);
```

19-1-실험4 : 3 Color DOT Matrix 제어

- 소스 파일 – (5)

```
// LD1071_DIN, LD1071_CLK, LD1071_STB
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2;
GPIO_Init(GPIOB, &GPIO_InitStructure);
// LD1071_PWM(PB5, TIM3_CH2)
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
GPIO_Init(GPIOB, &GPIO_InitStructure);
//TIM3_CH2
GPIO_PinAFConfig(GPIOB, GPIO_PinSource5, GPIO_AF_TIM3);
// 타이머3 PWM 설정
//(168Mhz/2)/1 = 84MHz
TIM_TimeBaseStructure.TIM_Prescaler = 0;
//(42(41+1) /84MHz = 0.5us), 2MHz
TIM_TimeBaseStructure.TIM_Period = 42-1;
TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;
TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);
TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
TIM_OCInitStructure.TIM_Pulse = 21;
TIM_OC2Init(TIM3, &TIM_OCInitStructure);
//타이머3을 동작시킨다.
TIM_Cmd(TIM3, ENABLE);
}
int main()
{
Init_Port();
LD1071_Reset();
// Configuration Register Update Command
LD1071_Tx_CMD(0x0034);
// Configuration Register
LD1071_Tx_Data(0x04FF);
```

19-1-실험4 : 3 Color DOT Matrix 제어

- 소스 파일 – (6)

```
while(1)
{
LD1071_Tx_Display(0xF30C);
// COM0
GPIO_Write(GPIOC, 0x01);
LD1071_Tx_Display(0xF30C);
// COM1
GPIO_Write(GPIOC, 0x02);
LD1071_Tx_Display(0x33CC);
// COM2
GPIO_Write(GPIOC, 0x04);
LD1071_Tx_Display(0x33CC);
// COM3
GPIO_Write(GPIOC, 0x08);
LD1071_Tx_Display(0xCC33);
// COM4
GPIO_Write(GPIOC, 0x10);
LD1071_Tx_Display(0xCC33);
// COM5
GPIO_Write(GPIOC, 0x20);
LD1071_Tx_Display(0xCF30);
// COM6
GPIO_Write(GPIOC, 0x40);
LD1071_Tx_Display(0xCF30);
// COM7
GPIO_Write(GPIOC, 0x80);
}
}
```