마이크로프로세서 응용 보고서 (SPI)

자동차IT융합학과 20183376

박선재

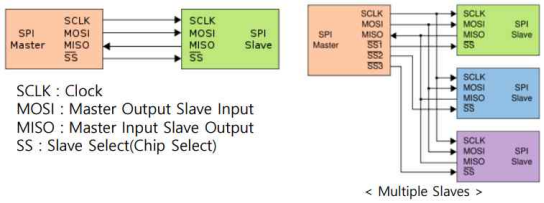
*주석으로 설명 대체

```
#define SPImode 0                // 1: Master로 사용할지, 0 : Slave로 사용
void Register0_set(void); void LED_ctrl(void); void init_DSPI_1(void);
void MasterDSPI1(void); void SlaveDSPI1(void); void init_ADC1(void);
void ADCRead_1(void); int MasterCnt = 0; int SlaveCnt = 0; uint16_t Send_Data = 0;
uint16_t Read_Data = 0; int R_adc = 0; int LED_dis = 0;
char LED[4] = {1, 1, 1, 1}; //LED  uint32_t Pit1cnt = 0;
int main(void)
{
        init_PIT(); init_DSPI_1(); init_ADC1();
        // Timer 1의 interrupt가 발생하면 PIT1ISR function으로 이동함
        // 6 : priority level, 60 : Vector number
        // priority level이 더 큰 것이 우선 동작, 우선 순위가 같으면 Vector Number가 덛 큰 것이 우선 동작
        INTC_InstallINTCInterruptHandler(PIT1ISR, 60, 6);
        Register0_set();
        /* Loop forever */
        for (;;)
        {
                LED_ctrl(); ADCRead_1();
        }
}
void Register0_set(void)
{
        // LED를 Output으로 사용하도록 세팅
        // .B => Register의 특정 비트의 특정 위치
        // .R => Register all
        // Pad Configuration Registers(Port D)의 OBE bit를 enable하여 output으로 사용하도록 설정
        SIU.PCR[52].R = 0x0200;  // LED1 // Register 전체 중 OBE bit만 따로 enable
        SIU.PCR[53].R = 0x0200;  // LED2
        SIU.PCR[54].R = 0x0200;  // LED3
        SIU.PCR[55].B.OBE = 0b1; // LED4 // OBE bit만 따로 불러와 enable
}
void LED_ctrl(void)
{
        //LED Output (Pad data output)
```

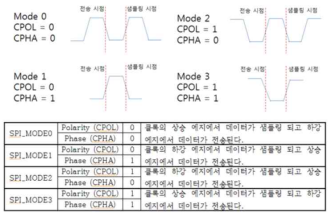
```
        SIU.GPDO[52].B.PDO = LED[0]; // LED1
        SIU.GPDO[53].B.PDO = LED[1]; // LED2
        SIU.GPDO[54].B.PDO = LED[2]; // LED3
        SIU.GPDO[55].B.PDO = LED[3]; // LED4
}
void init_PIT(void)
{
        //Enable PIT and Config Stop in debug mode
        PIT.PITMCR.R = 0x00000001;
        // Register에 설정 된 value에서 system clock의 주기로 0까지 count
        // Timeout = (6.4M) x 1sec / 64M sysclks = 100ms
        PIT.CH[1].LDVAL.R = 6400000;
        //Enable PIT1 Interrupt Enabled & Start, PIT counting
        PIT.CH[1].TCTRL.R = 0x00000003;
}
//Interrupt service routine
void PIT1ISR(void)
{
        Pit1cnt++; //Interrupt 발생 시 +1씩 연산
        //추가하지 않으면 interrput가 계속 살아 있어서 통신 interrupt를 받지 못함
        PIT.CH[1].TFLG.B.TIF = 1;
        MasterDSPI1();          //통신도 clock으로 PIT사용하기 때문에 여기서 호출
}
void init_ADC1(void)            //가변저항 초기화
{
        ADC_1.MCR.B.ABORT = 1;
        ADC_1.MCR.B.OWREN = 0; //disable overwritting
        ADC_1.MCR.B.WLSIDE = 0;
        ADC_1.MCR.B.MODE = 0;
        ADC_1.MCR.B.CTUEN = 0;
        ADC_1.MCR.B.ADCLKSEL = 0;
        ADC_1.MCR.B.ACK0 = 0;
        ADC_1.MCR.B.PWDN = 0;
        ADC_1.CTR[0].R = 0x00008208;
        ADC_1.NCMR[0].R = 0x00000020;
        ADC_1.CDR[0].R = 0x00000000;
        ADC_1.MCR.B.ABORT = 0; //Exit Abort ADC_1

}
void ADCRead_1(void)            //가변저항값 읽기
{
```

```
        ADC_1.MCR.B.NSTART = 1;
        asm("nop");                 //읽기 전까지 대기
        while(ADC_1.MCR.B.NSTART) asm("nop");
        R_adc = ADC_1.CDR[5].B.CDATA;
}
void init_DSPI_1(void)   //SPI : 모토롤라가 개발한 단순하고 신뢰성 높은 직렬 통신
{          // Clock을 사용하여 동기화된 통신을 하고 대역폭은 낮으나 신호 간섭이 적음
#if SPImode     //MASTER 초기화
        DSPI_1.MCR.R = 0x80010001;            // Configure DSPI_0 as master
        DSPI_1.CTAR[0].R = 0x7A0A7727;        // Configure CTAR0
        DSPI_1.MCR.B.HALT = 0x0;      //Exit HALT mode
        SIU.PCR[5].R = 0x0604;          //Config pad as DSPI_1 CS0 output
        SIU.PCR[6].R = 0x0604;          //Config pad as DSPI_1 SCK output
        SIU.PCR[7].R = 0x0604;          //Config pad as DSPI_1 SOUT output
        SIU.PCR[8].R = 0x0103;          //Config pad as DSPI_1 SIN input
#else           //SLAVE 초기화
        DSPI_1.MCR.R = 0x00010001;            // Configure DSPI_0 as slave
        DSPI_1.CTAR[0].R = 0x7A000000;        // Configure CTAR0
        DSPI_1.RSER.B.RFDFRE = 1;     // Recieve FIFO drain request enable
        DSPI_1.MCR.B.HALT = 0x0;      //Exit HALT mode
        SIU.PCR[5].R = 0x0504;          //Config pad as DSPI_1 CS0 input
        SIU.PCR[6].R = 0x0504;          //Config pad as DSPI_1 SCK input
        SIU.PCR[7].R = 0x0504;          //Config pad as DSPI_1 SOUT output
        SIU.PCR[8].R = 0x0103;          //Config pad as DSPI_1 SIN output
        INTC_InstallINTCInterruptHandler(SlaveDSPI1, 98, 5);    //수신 시 인터럽트로
떠서 받기 위해
#endif
}
void MasterDSPI1(void) //Master 통신에서 값 보내기
{
   MasterCnt++;
   Send_Data = R_adc; //Send할 값 (가변저항)
   DSPI_1.PUSHR.R = (0x00010000 | Send_Data);
   Read_Data = (uint16_t)(DSPI_1.POPR.R&0xffff);
   DSPI_1.SR.R = 0x80020000; //Clear TCF, RDRF flags by writing 1 to them
}
void SlaveDSPI1(void)  //Slave 통신에서 값 보내기
{
        SlaveCnt++;
        DSPI_1.PUSHR.R = Send_Data;  //Read data received by SPI
        Read_Data = (uint16_t)(DSPI_1.POPR.R&0xffff);  //Recieve할 값
```

```
        DSPI_1.SR.R = 0x80020000; //Clear TCF, RDRF flags by writing 1 to them
        LED_dis = Read_Data>>6;         //LED enable
        LED[0] = !(LED_dis&0x08);
        LED[1] = !(LED_dis&0x04);
        LED[2] = !(LED_dis&0x02);
        LED[3] = !(LED_dis&0x01);
}
```



- SPI에서의 사용신호



SPI_MODE0	Polarity (CPOL)	0	클록의 상승 에지에서 데이터가 샘플링 되고 하강
	Phase (CPHA)	0	에지에서 데이터가 전송됩니다.
SPI_MODE1	Polarity (CPOL)	0	클록의 하강 에지에서 데이터가 샘플링 되고 상승
	Phase (CPHA)	1	에지에서 데이터가 전송됩니다.
SPI_MODE2	Polarity (CPOL)	1	클록의 하강 에지에서 데이터가 샘플링 되고 상승
	Phase (CPHA)	0	에지에서 데이터가 전송됩니다.
SPI_MODE3	Polarity (CPOL)	1	클록의 상승 에지에서 데이터가 샘플링 되고 하강
	Phase (CPHA)	1	에지에서 데이터가 전송됩니다.

- CPOL과 CPHA

• DSPI Module Configuration Registers(DSPIx_MCR)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MSTR	MONT_SCKE	DCONF[0:1]		FRZ	MTFE	PCSSE	ROOE	PCSIS7	PCSIS6	PCSIS5	PCSIS4	PCSIS3	PCSIS2	PCSIS1	PCSIS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	MDIS	DIS_TXF	DIS_RXF	CLR_TXF	CLR_RXF	SMPL_PT[0:1]		0	0	0	0	0	0	0	HALT
W					w1c	w1c										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Port GPIO Pad Data Output Register 0-3(GPDO0_3)

Field	Description	
MSTR	Master/slave mode select DSPI모듈을 master/slave 모드 선택	1 : Master 모드 0 : Slave 모드
PCSISx	Peripheral chip select inactive state CS의 비활성화 상태 결정	1 : CS0_x가 high일 때 비활성화 0 : CS0_x가 low일 때 비활성화
HALT	Halt DSPI의 전송을 start/stop (Register 설정 시 stop 후 설정)	1 : Stop 0 : Start

- DSPI Clock and Transfer Attributes Registers(DSPIx_CTARn)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DBR		FMSZ				CPOL	CPHA	LSBFE	PCSSCK		PASC		PDT		PBR
W																
Reset	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CSSCK				ASC				DT				BR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Port GPIO Pad Data Output Register 0-3(GPDO0_3)

Field	Description	
FMSZ	Frame Size 통신 data size 설정	데이터 크기 = FMSZ+1(FMSZ>3)

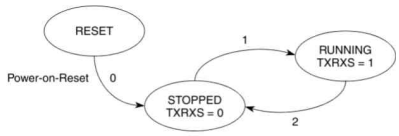


Figure 21-13. DSPI start and stop state diagram

Table 21-18. State transitions for start and stop of DSPI transfers

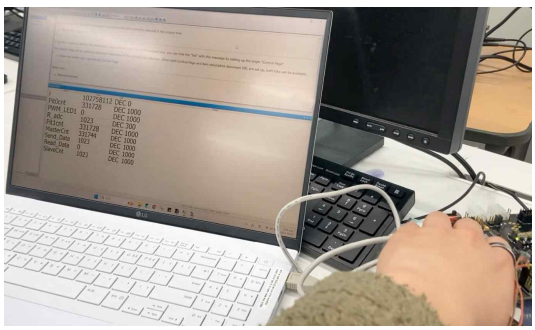
Transition #	Current State	Next State	Description
0	RESET	STOPPED	Generic power-on-reset transition
1	STOPPED	RUNNING	The DSPI starts (transitions from STOPPED to RUNNING) when all of the following conditions are true: • EOQF bit is clear • Debug mode is not selected or the FRZ bit is clear • HALT bit is clear
2	RUNNING	STOPPED	The DSPI stops (transitions from RUNNING to STOPPED) after the current frame for any one of the following conditions: • EOQF bit is set • Debug mode is selected and the FRZ bit is set • HALT bit is set

- 데이터 송신의 상태천이도

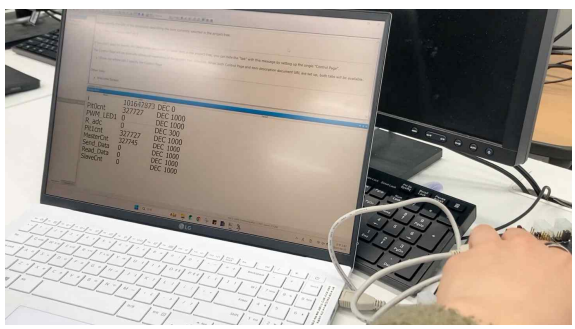
A[5]	PCR[5]	ALT0 ALT1 ALT2 ALT3 —	GPIO[5] CS0 ETC[5] CS7 EIRQ[5]	SIUL DSPI_1 eTimer_1 DSPI_0 SIUL	I/O I/O I/O O I	Slow	Medium	8	14
A[6]	PCR[6]	ALT0 ALT1 ALT2 ALT3 —	GPIO[6] SCK — — EIRQ[6]	SIUL DSPI_1 — — SIUL	I/O I/O — — I	Slow	Medium	2	2
A[7]	PCR[7]	ALT0 ALT1 ALT2 ALT3 —	GPIO[7] SOUT — — EIRQ[7]	SIUL DSPI_1 — — SIUL	I/O O — — I	Slow	Medium	4	10
A[8]	PCR[8]	ALT0 ALT1 ALT2 ALT3 — —	GPIO[8] — — — SIN EIRQ[8]	SIUL — — — DSPI_1 SIUL	I/O — — — I I	Slow	Medium	6	12

DSPI1			
94	0x0978	DSPI_SR[TFUF] DSPI_SR[RFOF]	DSPI 1
95	0x097C	DSPI_SR[EOQF]	DSPI 1
96	0x0980	DSPI_SR[TFFF]	DSPI 1
97	0x0984	DSPI_SR[TCF]	DSPI 1
98	0x0988	DSPI_SR[RFDF]	DSPI 1

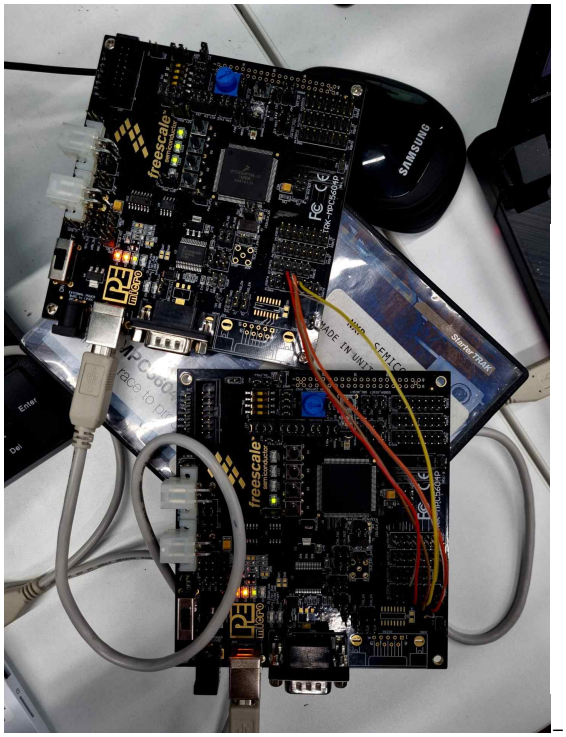
- PAD, 인터럽트 벡터 세팅



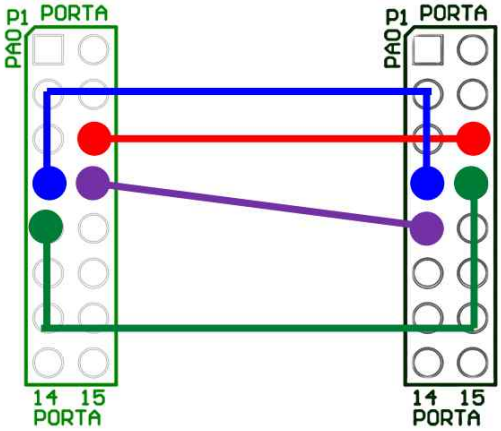
- Master의 가변저항(1023)에서 Master



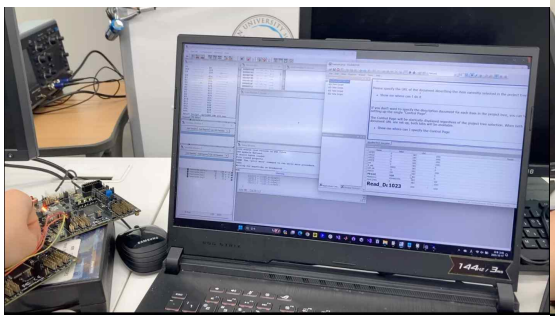
- Master의 가변저항(0)에서 Master



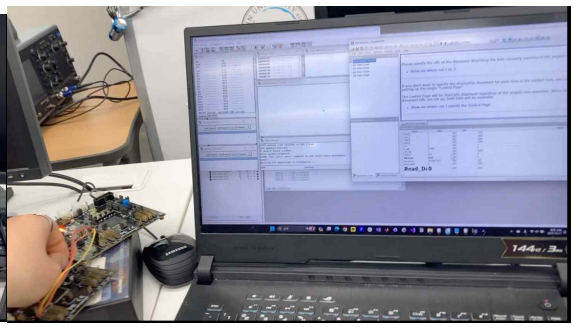
- Master와 Slave 연결



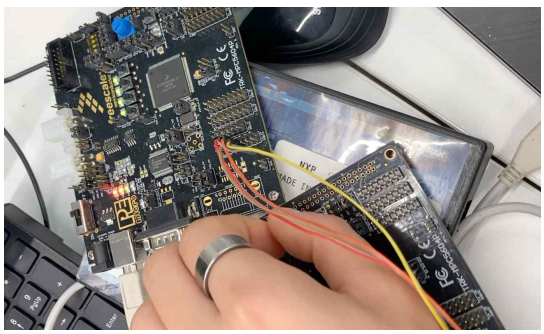
- Master와 Slave 연결



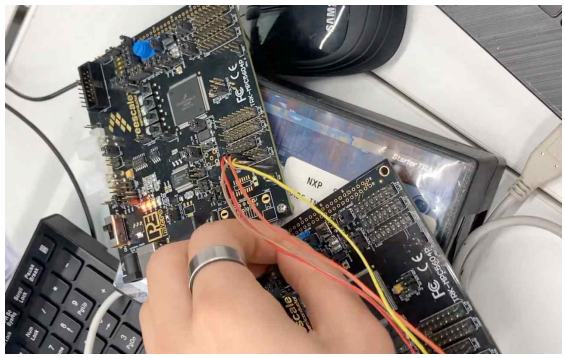
- Master의 가변저항(1023)에서 Slave



- Master에서 가변저항(0)에서 Slave



- Master의 가변저항(1023)에서 Slave LED



- Master의 가변저항(0)에서 Slave LED