NTC SAM7 강좌 4. 포트제어 및 printf를 사용하는 방법

뉴테크놀로지 컴패니(N.T.C)
http://www.NewTC.co.kr

1 개요

본 강좌에서는 SAM7S 컨트롤러의 포트를 제어하고 printf 를 사용하는 방법에 관하여 설명하겠습니다. 컴파일러는 IAR 5.20 을 기준으로 작성하였습니다.

2 포트 제어하기

2.1 포트 출력으로 LED 제어하기

AT91SAM7S64(256) 에는 총 32개의 포트가 있으며 이것은 PIO(Parallel IO)로 사용하거나 핀마다 부여된 특수 기능으로 사용될 수 있습니다. 비트 단위로 제어가 가능한 특징이 있습니다. 또한 PIO의 출력을 1로 만들고 0으로 만드는 레지스터가 분리되어 있습니다.

가) PMC에서 PIO로 클럭 공급하기

PMC_PCER 레지스터에 PIO 로 전원을 공급하도록 설정 합니다.
ATMEL 라이브러리 AT91F_PMC_EnablePeriphClock 사용하여 설정할 수 있습니다.
ex) AT91F_PMC_DisablePeriphClock (AT91C_BASE_PMC, 1 << AT91C_ID_PIOA) ;

나) 사용할 PIO 를 출력으로 설정하기

PIO_PER (PIO Enable Register), PIO_OER(PIO Output Enable Register) 를 설정합니다.
ATMEL 라이브러리 AT91F_PIO_CfgOutput 를 사용하여 설정할 수 있습니다.
ex) AT91F_PIO_CfgOutput(AT91C_BASE_PIOA, LED_MASK) ;

다) PIO 출력 레지스터를 설정하여 1 또는 0으로 컨트롤 하기

PIO_SODR(PIO Set Output Data Register) 는 PIO 출력을 1로 만들 때 사용하며
PIO_CODR(PIO Clear Output Data Register) 는 PIO 출력을 0으로 만들 때 사용한다.
AT91F_PIO_SetOutput, AT91F_PIO_ClearOutput 를 사용하여 설정할 수 있습니다.
ex) AT91F_PIO_SetOutput(AT91C_BASE_PIOA, LED0) ; //출력을 1로 만들기
AT91F_PIO_ClearOutput(AT91C_BASE_PIOA, LED0) ; //출력을 0으로 만들기

Example)

```
// LED 관련 선언
#define LED0   (1<<0)
#define LED1   (1<<3)
#define LED2   (1<<2)
#define LED_MASK   (LED0|LED1|LED2)
……

// PIO 클럭을 활성화 시킨다.
AT91F_PMC_EnablePeriphClock ( AT91C_BASE_PMC, 1 << AT91C_ID_PIOA ) ;
// LED 로 연결된 PIO 를 출력으로 설정한다.
AT91F_PIO_CfgOutput( AT91C_BASE_PIOA, LED_MASK ) ;
// LED0 출력을 0로 한다.
AT91F_PIO_ClearOutput( AT91C_BASE_PIOA, LED0) ;
// LED1 출력을 1로 한다.
AT91F_PIO_SetOutput( AT91C_BASE_PIOA, LED1) ;
```

2.2 포트 입력으로 스위치 입력 받기

가) PMC에서 PIO로 클럭 공급하기

PIO 출력과 마찬가지로 PMC_PCER 레지스터에 PIO 로 전원을 공급하도록 설정합니다.

ATMEL 라이브러리 AT91F_PMC_EnablePeriphClock 사용하여 설정할 수 있습니다.

ex) AT91F_PMC_DisablePeriphClock (AT91C_BASE_PMC, 1 << AT91C_ID_PIOA) ;

나) 사용할 PIO 를 입력으로 설정하기

PIO_PER (PIO Enable Register), **PIO_ODR**(PIO Output Disable Register) 를 설정합니다.

ATMEL 라이브러리 AT91F_PIO_CfgInput 를 사용하여 설정할 수 있습니다.

ex) AT91F_PIO_CfgInput(AT91C_BASE_PIOA, SW_MASK) ;

다) PIO의 Pullup Disable 시키기

PIO 내부에 Pull Up 을 시킬 수 있는 기능이 있습니다. 이 기능으로 인해서 입력이 정상적으로 되지 않을 수 있습니다. 따라서 Pull Up 기능을 Disable 시켜주는 것이 좋습니다.

PIO_PUDR(PIO Pull-up Disable Register) 를 설정합니다.

ex) AT91F_PIO_CfgPullupDisable(AT91C_BASE_PIOA, SW_MASK) ;

라) PIO 로 입력 받기

PIO_PDSR (PIO Data Status Register) 는 PIO 라인의 상태를 읽어오는 레지스터 입니다. 이 레지스터의 값을 읽어서 입력이 0인지 1인지 판단하게 됩니다.

AT91F_PIO_GetInput 함수를 사용하여 읽어오게 됩니다. 32비트 데이터 형태이므로 비트 연산을 통하여 해당 PIO 가 0인지 1인지 판단하게 됩니다.

ex) input = AT91F_PIO_GetInput(AT91C_BASE_PIOA);

Example)

```
// SW 관련 선언
#define SW_1       (1<<31)
#define SW_2       (1<<30)
#define SW_MASK (SW_1|SW_2)


……

// PIO 클럭을 활성화 시킨다.
AT91F_PMC_EnablePeriphClock ( AT91C_BASE_PMC, 1 << AT91C_ID_PIOA ) ;
// SW 로 연결된 PIO 를 입력으로 설정한다.
AT91F_PIO_CfgInput( AT91C_BASE_PIOA, SW_MASK) ;
// SW 로 연결된 PIO 의 Pull-up 을 Disable 시킨다.
AT91F_PIO_CfgPullupDisable( AT91C_BASE_PIOA, SW_MASK) ;
// PIO 의 데이터를 입력받아 input 이라는 변수에 저장한다.
input = AT91F_PIO_GetInput(AT91C_BASE_PIOA);

// port_num 번째 포트의 값을 읽어서 출력하는 함수
int SW_IsSet(int port_num)
{
      int input;
      input = AT91F_PIO_GetInput(AT91C_BASE_PIOA);
      if (input & (1<<port_num))
          return 1;
      else
          return 0;
}
```

3 시리얼 포트를 이용하여 디버깅 하기

3.1 시리얼 포트 사용하기

AT91SAM7S64(256)에는 시리얼 통신을 할 수 있는 포트가 Debug 포트와 UART0, UART1 로 총 3개 있습니다. 제공되는 강좌와 예제파일에서는 Debug 포트와 UART0 의 초기화하고 사용하는 방법을 설명하겠습니다.

가) 디버그포트의 초기화 및 함수

디버그포트의 초기화는 DBGU_Configure 함수에서 하도록 되어 있습니다. 해당 파일은 at91lib > peripherals > dbgu > dbgu.c 파일에 들어 있습니다. 자세한 내용은 해당 파일을 참고 하시기 바랍니다.

DBGU_Configure	디버그포트 초기화
DBGU_PutChar	디버그포트로 1바이트 데이터 전송
DBGU_GetChar	디버그포트로 1바이트 데이터 수신

나) UART0 포트 초기화 및 함수

UART0 포트의 초기화는 UART_init 함수에서 하도록 되어 있습니다. 해당 파일은 at91lib > utility > uart.c 파일에 들어 있습니다. 자세한 내용은 해당 파일을 참고하시기 바랍니다.

UART_init	UART0 초기화
UART0_putchar	UART0로 1바이트 데이터 전송
UART0_getchar	UART0로 1바이트 데이터 수신

3.2 **printf 함수 사용하기**

printf 함수를 사용하려면 putchar 함수를 구현해야 합니다. putchar 함수에서 어떻게 동작하느냐에 따라 출력 포트가 결정되게 됩니다. 해당 함수는 main.c 파일에 포함되어 있습니다.

return fputc(c, stdout); 를 사용할 경우 디버그 포트로 데이터를 전송하게 됩니다.

return UART0_putchar(c); 를 사용할 경우 UART0 로 데이터를 전송하게 됩니다.

putchar 함수

```
signed int putchar(signed int c)
{
    return fputc(c, stdout);          // 디버그 포트로 데이터 전송
    //return UART0_putchar(c);        // UART0 로 데이터 전송
}
```

3.3 시리얼 포트를 통하여 입력 받기

예제파일에는 시리얼 포트를 통하여 간단한 입력을 받을 수 있도록 구현되어 있습니다. 기본이 되는 getchar(); 함수를 구현하여 getstr(); (문자열 입력 받은 함수) getint(); (정수형 입력 받는 함수)를 구현 하였습니다. 해당 함수는 main.c 파일에 들어 있습니다.

signed int getchar(void)

: DBGU 또는 UART0에서 1바이트의 데이터를 읽어옵니다.
: 데이터가 수신될 때 까지 대기합니다.
: return 값 : 수신된 데이터 (int 형태이지만 실제로 char 형)

```
signed int getchar(void)
{
    return DBGU_GetChar();          // 디버그 포트로 데이터 수신
    //return UART0_putchar(c);      // UART0 로 데이터 수신
}
```

void getstr(char *sp, int control);

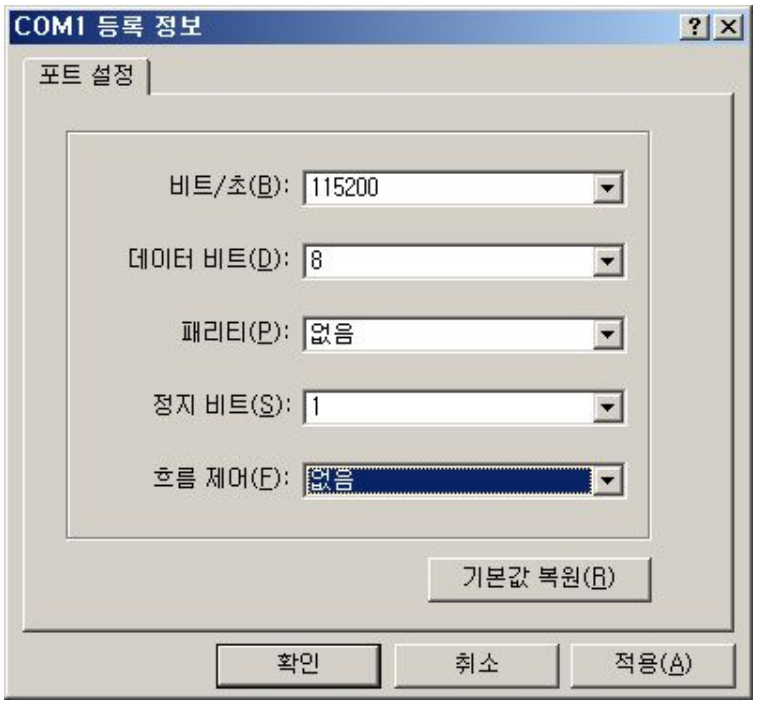
: getchar 에서 지정된 시리얼 포트를 통하여 문자열을 입력 받습니다.
: 하이퍼 터미널의 경우 문자열을 입력 받고 Enter가 들어올 때까지 대기합니다.
: '₩n' '₩r' NULL 문자가 들어올 때까지 입력을 받습니다.
: control 은 입력 받으면서 값을 터미널 화면으로 출력할 것인지 결정합니다.

int getint(int *out, int control)

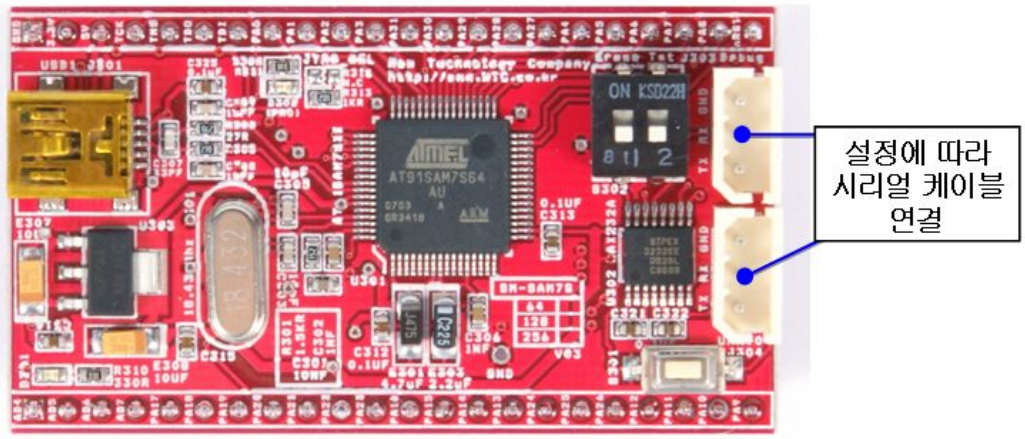
: getchar 에서 지정된 시리얼 포트를 통하여 정수를 입력 받습니다.
: 입력은 int * 형이며 포인터가 가리키는 공간에 입력 받은 정수를 저장 합니다.
: control 은 입력 받으면서 값을 터미널 화면으로 출력할 것인지 결정합니다.
: return 0 : 입력 받은 값이 유효하지 않음
: 1 : 입력 받은 값이 유효함

3.4 하이퍼터미널 설정하기

가) 시리얼 포트를 이용하여 메시지를 확인하시려면 하이퍼터미널과 같은 시리얼 통신 프로그램을 이용해야 합니다. 하이퍼터미널은 “윈도우즈 시작 > 보조프로그램 > 통신” 에 있습니다. 본인의 컴퓨터 설정에 따라 COM 포트를 잡고 통신 속도를 “115200 BPS” 와 흐름제어 “없음”을 선택합니다.



나) 시리얼 케이블을 모듈의 디버그 포트 또는 UART0에 연결하고 반대쪽은 PC에 연결합니다.



다) 설정을 완료하고 프로그램을 실행 시키면 아래와 같은 메시지를 볼 수 있습니다. 보드의 스위치를 누를 경우 [SWx] 메시지가 나타나게 됩니다.