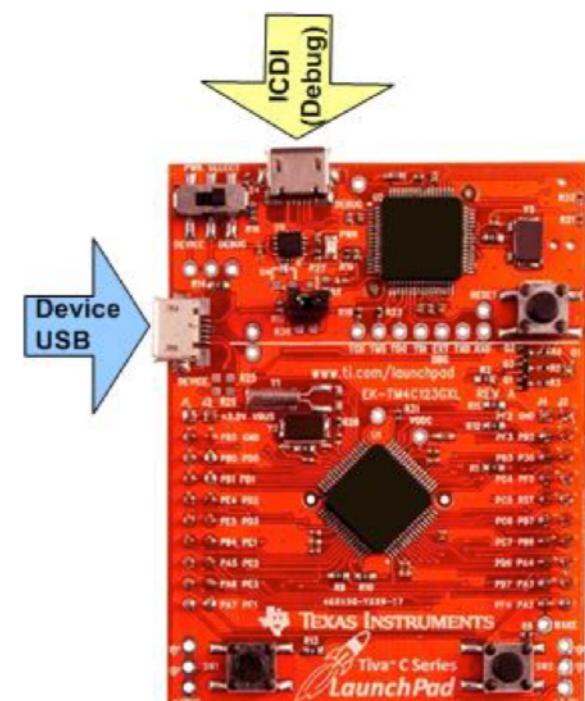


# **UART Serial Port Programming**

**Lesson 03**

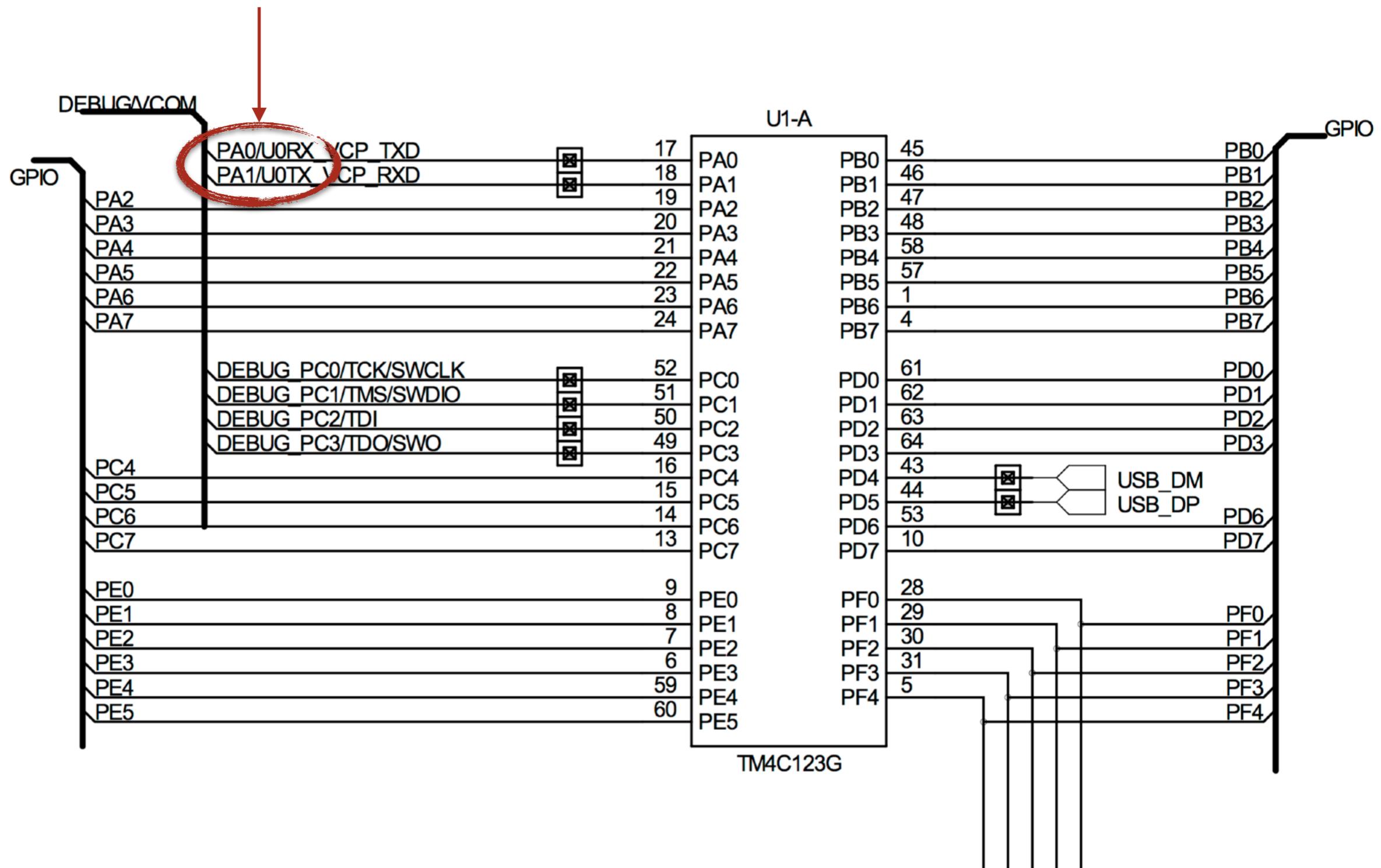
# Serial Ports in TI Launchpad

- **8 UART ports: UART0 ~ UART7**
- **UART0 포트는 ICDI (In-Circuit Debug Interface)에 연결**
  - Flash Programming 소프트웨어를 이용한 프로그래밍(다운로딩)
  - JTAG을 이용한 디버깅
  - Serial COM 포트로 사용 (USB 케이블로 연결하면  
노트북에서 serial com 포트로 인식됨)



# ICDI USB Port

GPIO Port A의 pin 0와 1이 UART0의 RX와 TX

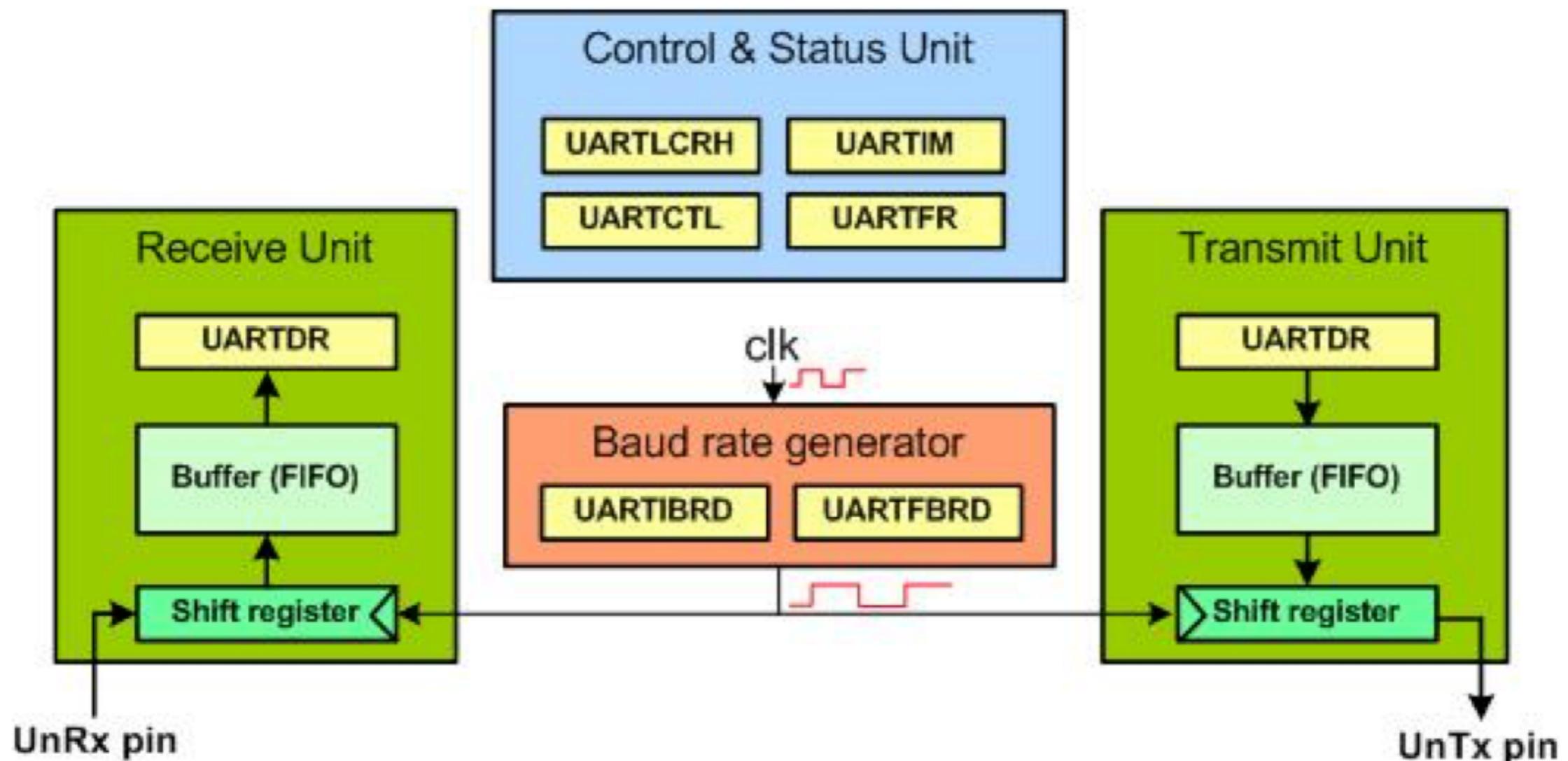


# UARTs Base Address

**Table 2-4. Memory Map (continued)**

Start	End	Description	For details, see page ...
0x4000.5000	0x4000.5FFF	GPIO Port B	658
0x4000.6000	0x4000.6FFF	GPIO Port C	658
0x4000.7000	0x4000.7FFF	GPIO Port D	658
0x4000.8000	0x4000.8FFF	SSI0	963
0x4000.9000	0x4000.9FFF	SSI1	963
0x4000.A000	0x4000.AFFF	SSI2	963
0x4000.B000	0x4000.BFFF	SSI3	963
0x4000.C000	0x4000.CFFF	UART0	902
0x4000.D000	0x4000.DFFF	UART1	902
0x4000.E000	0x4000.EFFF	UART2	902
0x4000.F000	0x4000.FFFF	UART3	902
0x4001.0000	0x4001.0FFF	UART4	902
0x4001.1000	0x4001.1FFF	UART5	902
0x4001.2000	0x4001.2FFF	UART6	902
0x4001.3000	0x4001.3FFF	UART7	902
0x4001.4000	0x4001.FFFF	Reserved	-

# Block Diagram of UARTn

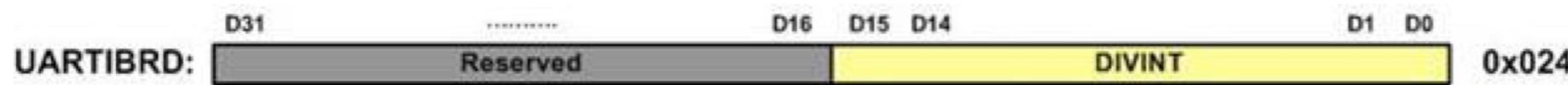


# UART Registers

1. **UARTIBRD와 UARTFBRD: Baud rate 설정**
2. **UARTCTL: UART, Rx, Tx enabling and disabling**
3. **UARTLCRH: data bit and stop bit length, FIFO enable/disable**
4. **UARTFR: FIFO empty/full check (수신 및 송신가능 여부 체크)**
5. **UARTDR: 데이터 레지스터, 수신한/전송할 데이터가 저장**

# 1. Baud-rate Generator

- 2개의 레지스터가 **Baud rate**를 설정하기 위한 **ClkDiv** 값을 저장
  - UARTIBRD (UART Integer Baud-Rate Divisor) - ClkDiv의 정수 부분
  - UARTFBRD (UART Fractional Baud-Rate Divisor) - ClkDiv의 소수 부분



UARTIBRD의 하위 16비트에 ClkDiv의 정수부분 저장



UARTFBRD의 하위 6비트에  
ClkDiv의 소수부분 저장

$$\text{Desired Baud Rate} = \text{SysClk} / (16 * \text{ClkDiv})$$

↓  
16MHz

## Baud-rate Generator: 예

$$\text{Desired Baud Rate} = \text{SysClk} / (16 * \text{ClkDiv})$$

↓  
16MHz

$$\text{ClkDiv} = 1\text{MHz} / \text{Desired Baud Rate}$$

• **1MHz/4800 = 208.3333**

• UARTIBRD = 208, UARTFBRD =  $(0.3333*64)+0.5 = 21.8312 = 21$

• **1MHz/9600 = 104.166666**

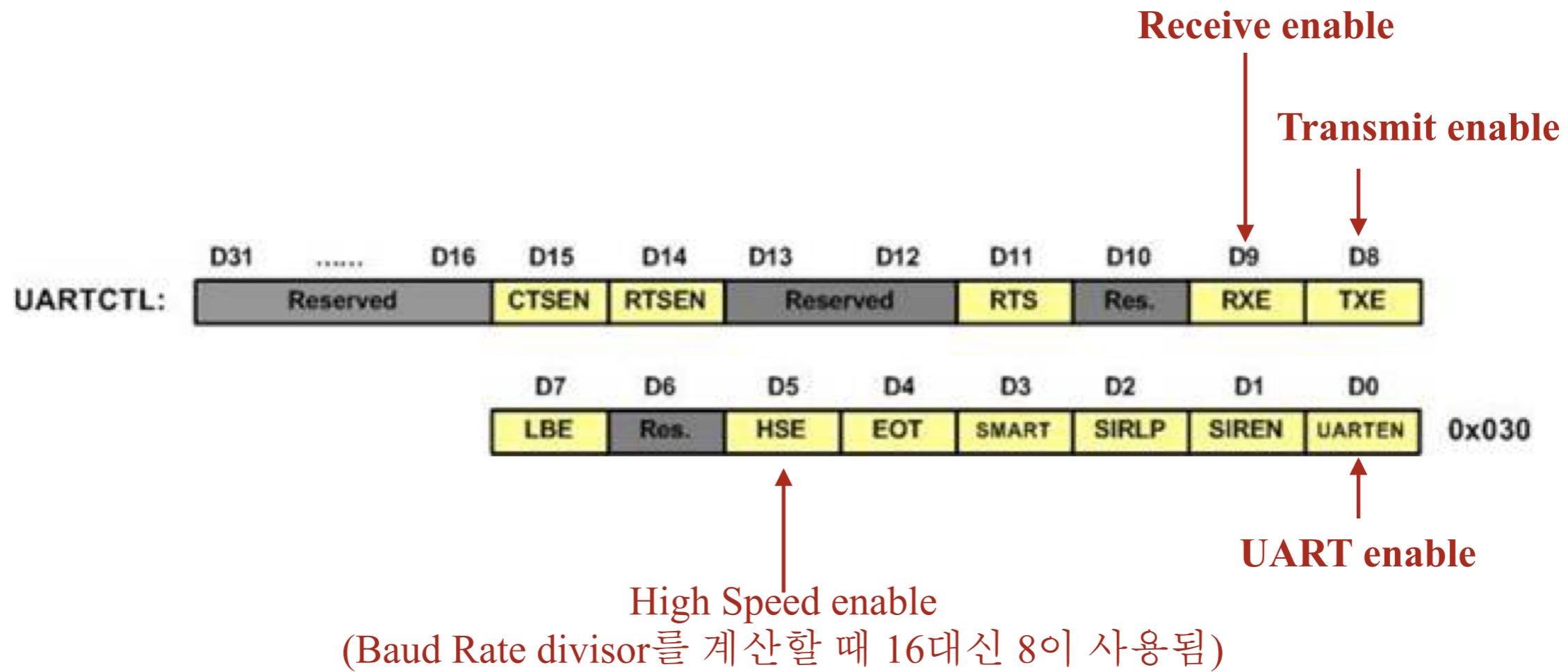
• UARTIBRD = **104**, UARTFBRD =  $(0.1666*64)+0.5 = 11$

• **1MHz/115200 = 8.680**

• UARTIBRD = 8, UARTFBRD =  $(0.680*64)+0.5 = 44$

```
UART0->IBRD = 104; /* 16MHz/16=1MHz, 1MHz/104=9600 baud rate */
UART0->FBRD = 11; /* fraction part */
```

## 2. UART Control (UARTCTL) 레지스터

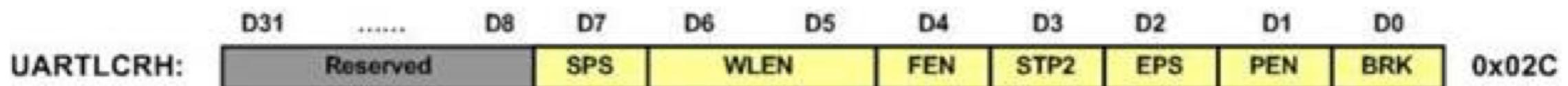


```
UART0->CTL = 0;          /* disable UART0 */  
...  
UART0->CTL = 0x301;       /* modifying some UART registers */  
                           /* enable UART0, TXE, RXE */
```

### 3. UART Line Control 레지스터

#### ➊ 프레임 당 데이터 비트 수와 stop 비트 수를 설정

STP2 = 1이면 2비트의 stop비트를 사용  
(느린 디바이스의 경우에 사용)



word length (# of data bits)

D6	D5	
0	0	5 bits
0	1	6 bits
1	0	7 bits
1	1	8 bits

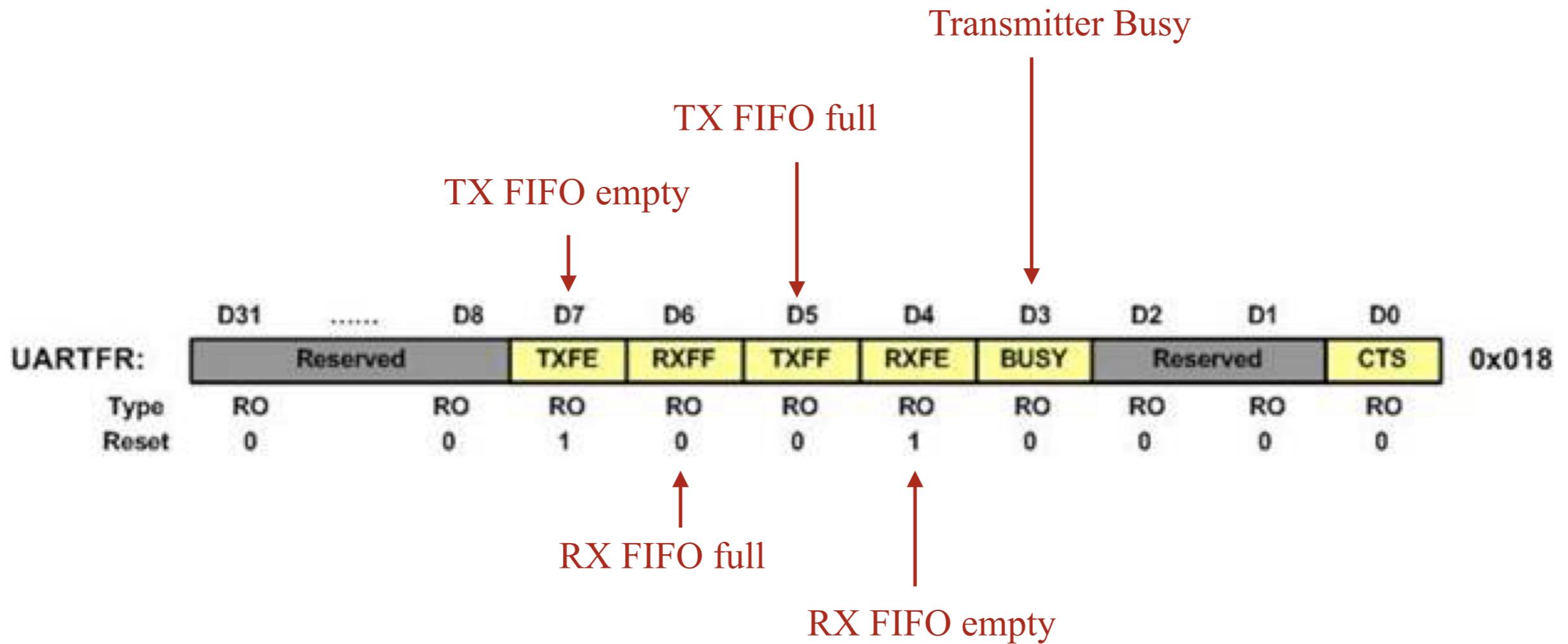
TI Tiva Launchpad는 전송 및 수신을 위해서  
각각 16-byte FIFO 버퍼를 사용

enable FIFO

data length 8bits

UART0->LCRH = (0x3<<5)|(1<<4);

## 4. UARTFR: UART Flag Register



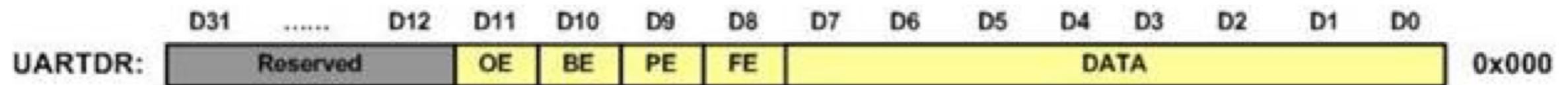
전송시:

```
while (UART0->FR & (1<<5) != 0);  
UART0->DR = c;
```

수신시:

```
while (UART0->FR & (1<<4)) != 0);  
c = UART0->DR;
```

## 5. UART Data Register



- ④ 전송할/수신된 데이터가 저장
- ④ 하위 8비트에 데이터가 저장됨
- ④ 나머지 네 비트(D8~D11)은 오류검출 용도로 사용됨

# RCGCUART: UART Run Mode Clock Gating Control

- UART에 clock을 enable함
- UART0 ~ UART7 각각에 1비트씩

UART0를 사용하려면  
이 bit를 enable한다.

(SYSCTL->RCGCUART |= 1;)

그냥 1을 넣어주면 앞의 부분이 다 0이 되기 때문에  
or 연산자를 이용해야한다.  
원하는 부분만 1이 되고 나머지는 그대로 있음

	D31	.....	D8	D7	D6	D5	D4	D3	D2	D1	D0	
RCGCUART:		Reserved		R7	R6	R5	R4	R3	R2	R1	R0	0x618
Type	RO		RO	R/W								

# GPIO pins used for UART TxD and RxD

- I/O 핀을 UART 용도로 사용하도록 설정해야 함
- 5가지 레지스터를 설정해주어야 함

- PORTx Run Mode Clock Gating Control ← SYSCTL->RCGCGPIO |= (1<<0);  
a port에 1을 넣어준다.  
첫 포트에..
- PORTx Digital Enable ← GPIOA->DEN = (1<<0) | (1<<1);
- PORTx ADC Mode Selection ← UART와 A/D 컨버터 기능을 공유하는 핀의 경우  
ADC를 disable해야 함 (UART0에는 해당없음)
- PORTx Alternate Selection
- PORTx Port Control registers  
여러가지 기능 pin mux =하나의 핀에 여러가지 기능을 여러가지 중첩시켜 놓음

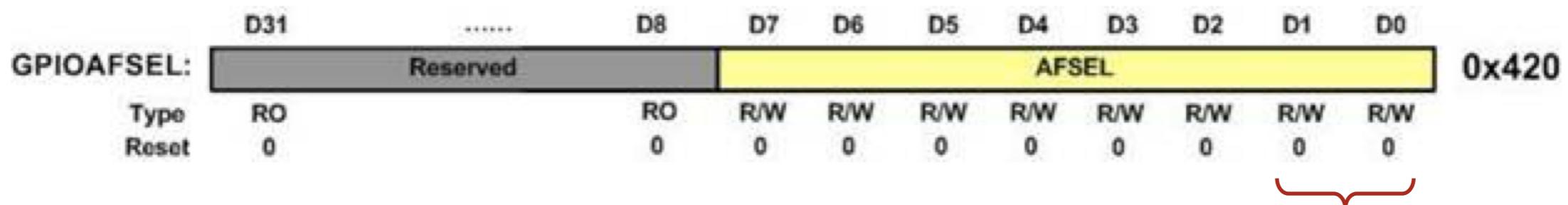
# GPIOAMSEL: GPIO Analog Mode Selection

- ADC (analog-to-digital)에 연결된 핀을 UART로 사용하기 위해서는 ADC 기능을 disable해야 함
- PE0~PE5, PB4~PB5, PD0~PD30이 analog input을 지원함
- Reset시에 PORTxAMSEL 레지스터는 모두 0으로 리셋됨 (즉 disable됨).
- PA0와 PA1에 대해서는 설정이 불필요

GPIOAMSEL:										0x528
Type	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	
Reset	0	0	0	0	0	0	0	0	0	
D31	.....	D8	D7	D6	D5	D4	D3	D2	D1	D0
Reserved										

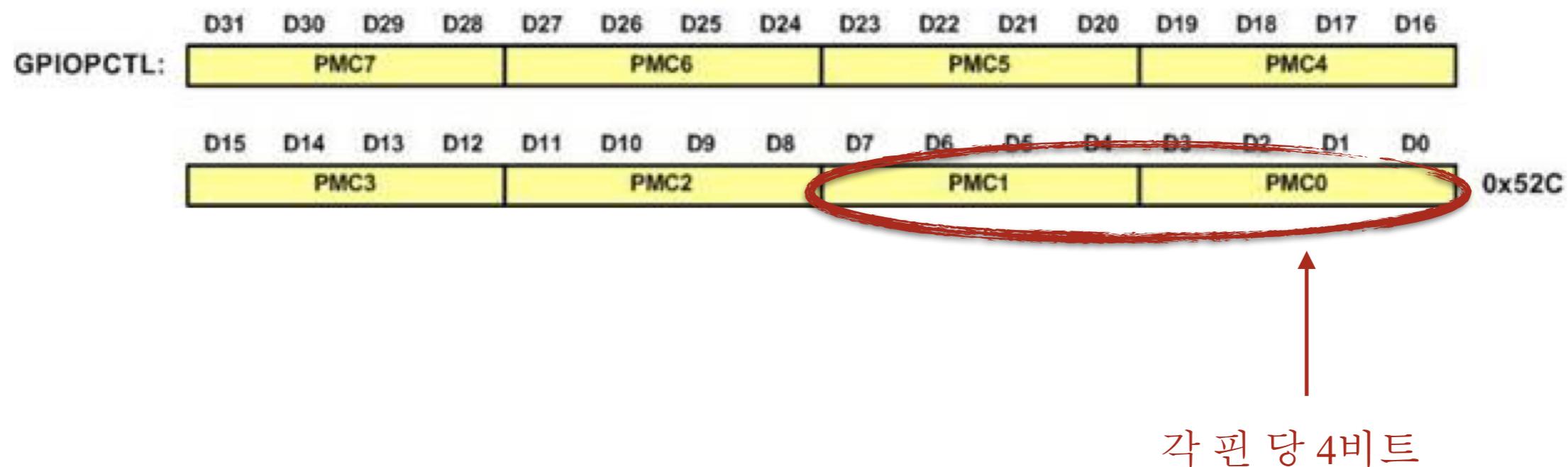
# GPIOAFSEL: GPIO Alternate Function Select

- Reset시에 모든 GPIO 핀들은 단순 I/O로 설정됨
- UART와 같은 alternate function을 사용하려면
  - 먼저 GPIOAFSEL 레지스터에서 alternate function을 enable하고
  - GPIOPCTL 레지스터에서 어떤 alternate function을 사용할지 지정해야함 (하나의 핀에 2개 이상의 alternate function이 부여되어 있을 수 있음)



UART0를 사용하려면  
2 비트를 enable한다.  
(GPIOA->AFSEL = (1<<1)|(1<<0);  
핀 0과 1을 1로 설정

# GPIOPCTL 레지스터



## GPIOPCTL 레지스터

PA0와 PA1을 UART Rx와 Tx로 사용하려면 0001을  
GPIOPCTL의 D0~D3와 D4~D7에 각각 저장

GPIOA->PCTL = (1<<0) | (1<<4);

마지막 비트를 1로 하고  
1을 4비트 쉬프트  
10001 - 마지막

Pin	Digital Function (GPIOPCTL PMCx)															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
17	PA0	U0Rx								CAN1Rx						
18	PA1	U0Tx								CAN1Tx						
45	PB0	U1Rx							T2CCPO							
46	PB1	U1Tx							T2CCP1							
16	PC4	U4Rx	U1Rx		M0PWM6			IDX1	WT0CCPO	U1RTS						
15	PC5	U4Tx	U1Tx		M0PWM7			PhA1	WT0CCP1	U1CTS						
14	PC6	U3Rx						PhB1	WT1CCPO	USB0EPEN						
13	PC7	U3Tx						WT1CCP1	USB0PFLT							
43	PD4	U6Rx						WT4CCPO								
44	PD5	U6Tx						WT4CCP1								
53	PD6	U2Rx		M0FAULT0			phA0	WT5CCPO								
10	PD7	U2Tx					phB0	WT5CCP1	NMI							
9	PE0	U7Rx								USB0EPEN						
8	PE1	U7Tx								USB0PFLT						
59	PE4	U5Rx		I2C2SCL	M0PWM4	M1PWM2				CAN0Rx						
60	PE5	U5Tx		I2C2SDA	M0PWM5	M1PWM3				CAN0Tx						
28	PF0	U1RTS	SSI1Rx	CAN0Rx		M1PWM4	PhA0	T0CCPO	NMI							
29	PF1	U1CTS	SSI1Tx			M1PWM5	PhB0	T0CCP1								

## 데이터 전송 절차 (pp.902 of Data Sheet)

1. RCGCUART에 1을 write하여 UART0에 clock을 공급한다.
2. RCGCGPIO에 1을 write하여 PORTA에 clock을 공급한다.
3. UARTCTL에 0을 write하여 UART0를 disable한다.
4. UARTIBRD와 UARTRFBRD에 baud rate를 설정한다.
5. UART0의 UARTCC에 0을 write하여 system clock을 UART의 clock source로 설정한다.
6. UARTLCRH에 0x60을 저장하여 1 stop bit, no FIFO, no interrupt, no parity, 8-bit 데이터 크기를 지정한다.
7. UARTCTL의 TxE와 RxE 비트를 1로 설정하여 Tx와 Rx를 enable한다.
8. UARTCTL의 UARLEN 비트를 1로 설정하여 UART0를 enable한다.

## 데이터 전송 절차 (pp.902 of Data Sheet)

9. PA0와 PA1 핀을 Digital I/O로 사용하도록 설정한다.
10. GPIOSEL에서 PA0와 PA1을 alternate 기능(RxD와 TxD)으로 사용한다.
11. CPIOPCTL에서 PA0와 PA1을 UART 기능으로 설정한다.
12. TxD가 idle high가 되기를 기다린다.
13. UART Flag 레지스터의 TXFF flag를 monitor하다가 LOW (buffer not full) 이 되면 전송할 1 바이트의 데이터를 Data 레지스터에 쓴다.

# Transmitting

```
/* Sending "YES" to UART0 on TI ARM Launchpad (TM4C123GH6PM) */
/* UART0 is on USB/Debug */

#include <stdint.h>
#include “TM4C123GH6PM.h”

void UART0Tx(char c);
void delayMs(int n);

int main(void)
{
    uart 클럭게이팅레지스터
    SYSCTL->RCGCUART |= 1; /* ① provide clock to UART0 */
    SYSCTL->RCGCGPIO |= 1; /* ② enable clock to PORTA */

    /* UART0 initialization */
    UART0->CTL = 0;           /* ③ disable UART0 */ 0으로 초기화를 해서 셋팅을 시작
    UART0->IBRD = 104;        /* ④ 16MHz/16=1MHz, 1MHz/104=9600 baud rate */
    UART0->FBRD = 11;         /* ④ fraction part, see Example 4-4 */
    UART0->CC = 0;            /* ⑤ use system clock */
    UART0->LCRH = 0x60;       /* ⑥ 8-bit, no parity, 1-stop bit, no FIFO */
    UART0->CTL = 0x301;       /* ⑦,⑧ enable UART0, TXE, RXE */
```

# Transmitting

```
/* UART0 TX0 and RX0 use PA0 and PA1. Set them up. */
GPIOA->DEN = 0x03;          /* ⑨ Make PA0 and PA1 as digital */
GPIOA->AFSEL = 0x03;          /* ⑩ Use PA0,PA1 alternate function */ GPIO기능 말고 UART 기능으로 사용
GPIOA->PCTL = 0x11;          /* ⑪ configure PA0 and PA1 for UART */

    핀 하나당 4비트씩

delayMs(1);                  /* wait for output line to stabilize */

for(;;)
{
    UART0Tx('Y');
    UART0Tx('E');
    UART0Tx('S');
    UART0Tx(' ');
}

}
```

# Transmitting

```
/* UART0 Transmit */
/* This function waits until the transmit buffer is available then */
/* writes the character in the transmit buffer. It does not wait */
/* for transmission to complete. */
void UART0Tx(char c)
{
    while((UART0->FR & 0x20) != 0); /* ⑫ wait until Tx buffer not full */
    UART0->DR = c;                  /* ⑬ before giving it another byte */
}

void delayMs(int n)
{
    int i, j;
    for(i = 0 ; i < n; i++)
        for(j = 0; j < 3180; j++)
            {} /* do nothing for 1 ms */
}
```

## 데이터 수신 절차

1~11은 송신 절차와 동일함

12. **UART Flag** 레지스터의 **RXFE flag**를 **monitor**하다가 **LOW (buffer not empty)**가 되면 **Data** 레지스터로부터 **1바이트**를 읽어 온다.

# Receiving

```
/* Read data from UART0 and display it at the tri-color LEDs. */
/* The LEDs are connected to Port F 3-1. */
/* Press any A-z, a-z, 0-9 key at the terminal emulator */
/* and see ASCII value in binary is displayed on LEDs of PORTF. */

#include <stdint.h>
#include “TM4C123GH6PM.h”

char UART0Rx(void);
void delayMs(int n);

int main(void)
{
    char c;

    SYSCTL->RCGCUART |= 1; /* provide clock to UART0 */
    SYSCTL->RCGCGPIO |= 1; /* enable clock to PORTA */
    SYSCTL->RCGCGPIO |= 0x20; /* enable clock to PORTF */
```

# Receiving

```
/* UART0 initialization */
UART0->CTL = 0;                      /* disable UART0 */
UART0->IBRD = 104;                    /* 16MHz/16=1MHz, 1MHz/104=9600 baud rate */
UART0->FBRD = 11;                     /* fraction part, see Example 4-4 */
UART0->CC = 0;                        /* use system clock */
UART0->LCRH = 0x60;                   /* 8-bit, no parity, 1-stop bit, no FIFO */
UART0->CTL = 0x301;                   /* enable UART0, TXE, RXE */

/* UART0 TX0 and RX0 use PA0 and PA1. Set them up. */
GPIOA->DEN = 0x03;                    /* Make PA0 and PA1 as digital */
GPIOA->AFSEL = 0x03;                  /* Use PA0,PA1 alternate function */
GPIOA->PCTL = 0x11;                   /* configure PA0 and PA1 for UART */

GPIOF->DIR = 0x0E;                    /* configure Port F to control the LEDs */
GPIOF->DEN = 0x0E;
GPIOF->DATA = 0;
```

# Receiving

```
for(;;)
{
    c = UART0Rx();           /* get a character from UART */
    GPIOF->DATA = c << 1;   /* shift left and write it to LEDs */
}

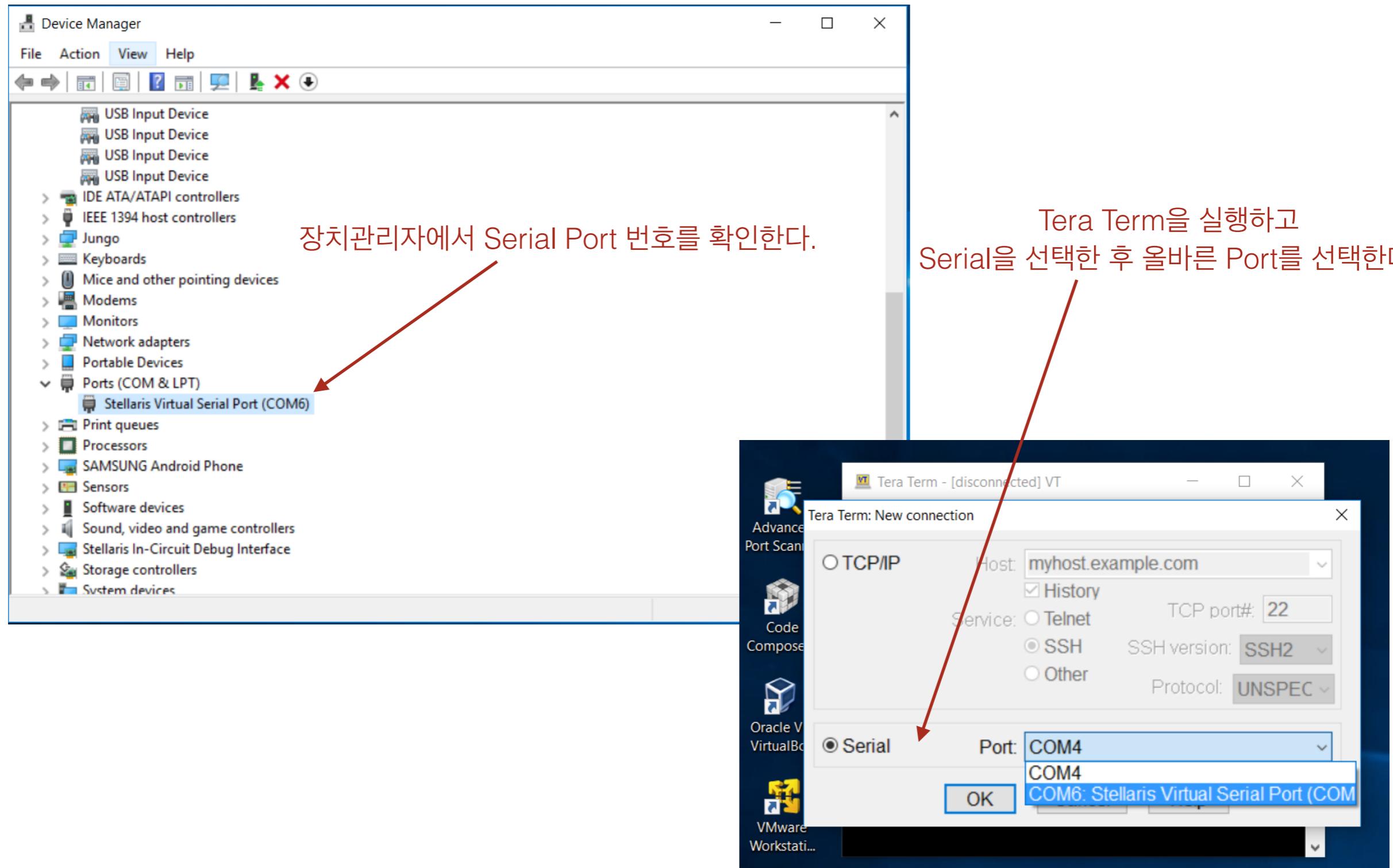
/* UART0 Receive */
/* This function waits until a character is received then returns it. */
char UART0Rx(void)
{
    char c;
    while((UART0->FR & 0x10) != 0); /* wait until the buffer is not empty */
    c = UART0->DR;                  /* read the received data */
    return c;                       /* and return it */
}

/* Append delay functions here */
```

# Serial Terminal

- ☞ **Tera Term**
  - ☞ 다운로드 주소
  - ☞ [http://microdigitaled.com/tutorials/Tera\\_Terminal.pdf](http://microdigitaled.com/tutorials/Tera_Terminal.pdf)
- ☞ **Or Arduino Serial Terminal is OK.**

# TeraTerm 사용



- Terminal을 열고 다음과 같이 screen 명령으로 접속한다.

```
$ ls /dev/tty.* ← Is 명령으로  
...  
디바이스 파일명을 확인한다.  
  
/dev/tty.usbmodem0E21CDA1  
  
...  
$ screen /dev/tty.usbmodem0E21CDA1 9600
```

- 연결을 종료할 때는 Control-a, 그리고 Control-\를 누르고 yes를 입력한다.
- 그냥 터미널 창을 닫는 방식으로 종료하면 프로세스가 포트를 점유하고 있어서 재연결이 안됨
- 그런 경우에는 다음과 같이 kill -9 명령으로 프로세스를 종료하고 다시 연결한다.

```
$ ps ← ps 명령으로 프로세스 id를 알아낸다.  
 PID TTY          TIME CMD  
 80593 ttys000    0:00.01 -bash  
 80580 tty.usbmodem0E21CDA1  0:00.00 SCREEN /dev/tty.usbmodem0E21CDA1  
  
$ kill -9 80580 ← 80580번 프로세스를 종료한다.
```

## Serial Echo: main.c

```
#include "TM4C123GH6PM.h"
#include "serial.h"
#define MAX 100

int main(void)
{
    init_serial();
    while(1)
    {
        printString("Type something and press enter: ");
        char *string = readString('\r');
        printString("\n\r");
        printString("You typed: ");
        printString(string);
        printString("\n\r");
        free(string);
    }
}
```

# serial.h

```
#include "TM4C123GH6PM.h"
#include <stdlib.h>

void init_serial(void);
char readChar(void);
void printChar(char c);
void printString(char * string);
char *readString(char);
```

```
#include "serial.h"

void init_serial()
{
    // 1. Enable the UART module using the RCGCUART register (see page 344).
    SYSCTL->RCGCUART |= (1<<0);

    // 2. Enable the clock to the appropriate GPIO module via
    // the RCGCGPIO register (see page 340). To find out which GPIO port
    // to enable, refer to Table 23-5 on page 1351.
    SYSCTL->RCGCGPIO |= (1<<0);

    // 3. Set the GPIO AFSEL bits for the appropriate pins (see page 671).
    // To determine which GPIOs to configure, see Table 23-4 on page 1344
    GPIOA->AFSEL = (1<<1)|(1<<0);

    // 4. Configure the GPIO current level and/or slew rate as specified
    // for the mode selected (see page 673 and page 681).
```

```
// 5. Configure the PMCn fields in the GPIOPCTL register to assign
// the UART signals to the appropriate
// pins (see page 688 and Table 23-5 on page 1351).
GPIOA->PCTL = (1<<0)|(1<<4);

GPIOA->DEN = (1<<0)|(1<<1);

// Find the Baud-Rate Divisor
// BRD = 16,000,000 / (16 * 9600) = 104.1666666
// UARTFBRD[DIVFRAC] = integer(0.166667 * 64 + 0.5) = 11

// With the BRD values in hand, the UART configuration is written to
// the module in the following order

// 1. Disable the UART by clearing the UARTEN bit in the UARTCTL register
UART0->CTL &= ~(1<<0);

// 2. Write the integer portion of the BRD to the UARTIBRD register
UART0->IBRD = 104;
// 3. Write the fractional portion of the BRD to the UARTFBRD register.
UART0->FBRD = 11;
```

```
// 4. Write the desired serial parameters to the UARTLCRH register  
// (in this case, a value of 0x0000.0060)  
UART0->LCRH = (0x3<<5)|(1<<4);      // 8-bit, no parity, 1-stop bit  
  
// 5. Configure the UART clock source by writing to the UARTCC register  
UART0->CC = 0x0;  
  
// 6. Optionally, configure the DMA channel (see page 585)  
// and enable the DMA option(s) in the UARTDMACTL register  
  
// 7. Enable the UART by setting the UARTEN bit in the UARTCTL register.  
UART0->CTL = (1<<0)|(1<<8)|(1<<9);  
  
// Configure LED pins  
SYSCTL->RCGCGPIO |= (1<<5);    // enable clock on PortF  
GPIOF->DIR = (1<<1)|(1<<2)|(1<<3); // make LED pins (PF1,PF2,PF3) outputs  
GPIOF->DEN = (1<<1)|(1<<2)|(1<<3); // enable digital function on LED pins  
GPIOF->DATA &= ~((1<<1)|(1<<2)|(1<<3)); // turn off leds  
}
```

```
char readChar(void)
{
    char c;          플래그 레지스터에 4번째 비트를 검사
    while((UART0->FR & (1<<4)) != 0);
    c = UART0->DR;
    return c;
}

void printChar(char c)
{
    while((UART0->FR & (1<<5)) != 0);
    UART0->DR = c;
}

void printString(char * string)
{
    while(string != 0 && *string)
    {
        printChar(*string++);
    }
}
```

```
char* readString(char delimiter)      {
    int stringSize = 0;
    char* string = (char*)calloc(10, sizeof(char));
    char c = readChar();
    printChar(c); 내가 적은 것을 터미널에서 보기위한 함수

    while(c!=delimiter)
    {
        *(string+stringSize) = c; // put the new character at the end of the array
        stringSize++;
        if((stringSize%10) == 0) // string length has reached multiple of 10
            string = (char*)realloc(string, (stringSize+10)*sizeof(char));
        c = readChar();
        printChar(c); // display the character the user typed
    }
    if(stringSize == 0)
        return '\0'; // null char
    return string;
}
```