

# 시리얼 통신 SCI 모듈

# SCI Registers

## ■ SCI Baud Rate Register (SCIBD)

	7	6	5	4	3	2	1	0
R	0	0	0	SBR12	SBR11	SBR10	SBR9	SBR8
W								
Reset	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
W								
Reset	0	0	0	0	0	1	0	0

### ■ SCI 모듈의 Baud Rate 설정

■ SCI baud rate =  $\frac{\text{SCI module clock}}{(16 \times \text{BR})}$  (BR: [SBR12..SBR0])

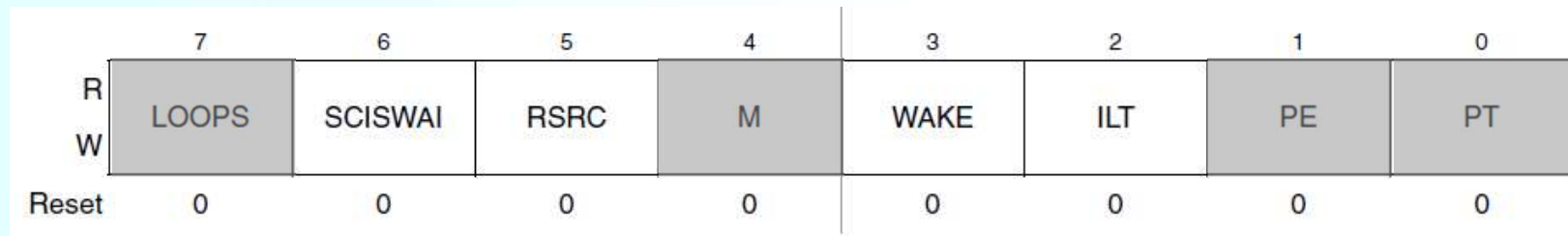
■ SCI module clock =  $\frac{\text{Oscillator frequency}}{2}$

■ [SBR12..SBR0]는 1 ~ 8191까지 설정 가능

■ 기본값: 4

# SCI Registers

## ■ SCI Control Register 1 (SCICR1)



■ **LOOPS** : 송신출력을 수신입력에 연결 (Loop-Back mode 설정)

■ 0 : Normal operation

■ 1 : Loop operation

■ **M** : Data Format Mode Bit

■ 0 : 1 start bit, 8 data bits, 1 stop bit

■ 1 : 1 start bit, 9 data bits, 1 stop bit

# SCI Registers

## ■ SCI Control Register 1 (SCICR1) (계속)

	7	6	5	4	3	2	1	0
R								
W								
	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
Reset	0	0	0	0	0	0	0	0

### ■ PE : Parity Enable Bit

■ 0 : Parity 기능 disabled

■ 1 : Parity 기능 enabled

➤ 가장 중요한 bit 위치에 parity bit 삽입

### ■ PT : Parity Type Bit

■ 0 : 짝수 parity

■ 1 : 홀수 parity

# SCI Registers

## ■ SCI Status Register 1 (SCISR1)

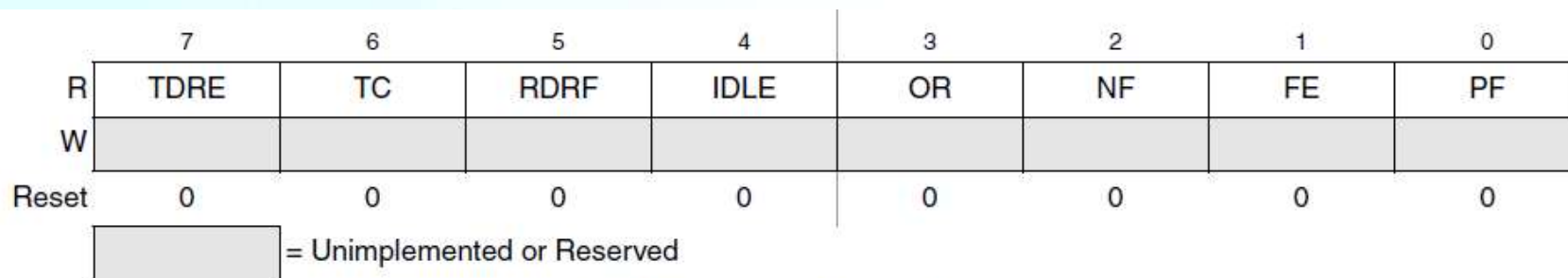


Figure 13-6. SCI Status Register 1 (SCISR1)

Read: Anytime

Write: Has no meaning or effect

### ■ TDRE : Transmit Data Register Empty Flag

- 데이터 송신하면 set
- 0 : 전송할 데이터가 전송되지 아직 않은 상태
- 1 : 전송할 데이터가 비었으므로 새로운 전송 데이터 수신 가능

### ■ TC : Transmit Complete Flag

- 0 : 데이터 전송 중인 상태
- 1 : TDRE = 1, 전송할 데이터가 없는 상태

# SCI Registers

## ■ SCI Status Register 1 (SCISR1) (계속)

	7	6	5	4	3	2	1	0
R	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
W								
Reset	0	0	0	0	0	0	0	0


 = Unimplemented or Reserved

Figure 13-6. SCI Status Register 1 (SCISR1)

Read: Anytime

Write: Has no meaning or effect

### ■ RDRF : Receive Data Register Full Flag

- 데이터를 수신하면 set
- 0 : SCIDR 데이터 접근 불가
- 1 : SCIDR 데이터 접근 가능

# SCI Registers

## ■ SCI Control Register 2 (SCICR2)

	7	6	5	4	3	2	1	0
R								
W								
	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
Reset	0	0	0	0	0	0	0	0

### ■ TIE : Transmitter Interrupt Enable Bit

- 0 : 데이터 전송 인터럽트 요청 disabled
- 1 : 데이터 전송 인터럽트 요청 enabled

### ■ RIE : Receiver Full Interrupt Enable Bit

- 0 : 데이터 수신 인터럽트 요청 disabled
- 1 : 데이터 수신 인터럽트 요청 enabled



# SCI Registers

## ■ SCI Control Register 2 (SCICR2) (계속)

	7	6	5	4	3	2	1	0
R								
W								
	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
Reset	0	0	0	0	0	0	0	0

■ TE : Transmitter Enable Bit

■ 0 : Transmitter disabled

■ 1 : Transmitter enabled

■ RE : Receiver Enabled Bit

■ 0 : Receiver disabled

■ 1 : Receiver enabled



# SCI Registers

## ■ SCI Data Registers (SCIDRH and SCIDRL)

	7	6	5	4	3	2	1	0
R	R8	T8	0	0	0	0	0	0
W								
Reset	0	0	0	0	0	0	0	0

Module Base + 0x\_0007

	7	6	5	4	3	2	1	0
R	R7	R6	R5	R4	R3	R2	R1	R0
W	T7	T6	T5	T4	T3	T2	T1	T0
Reset	0	0	0	0	0	0	0	0

■ Transmit data or receive data

■ R8 & T8

■ 9번째 송수신 데이터

■ 9-bit data format (SCICR1 M=1)일 때 사용

# SCI Example

## ■ Example) 인터럽트를 이용한 Serial communication

### ■ projectvectors.c

- Interrupt handler definition
- software\_trap() function : default interrupt handler

### ■ projectvectors.h

- Interrupt handler declaration
- Include user's header file

# SCI Example

## Interrupt handler 추가

Globals	234	U	•	•	•
projectglobals.h	n/a	n/a	•	•	•
projectvectors.c	234	0	•	•	•
projectvectors.h	n/a	n/a	•	•	•

Open 'projectvectors.c' file

```

//*****
// SCI0 ISR
// DESCRIPTION:
// Interrupt asserted from one of five sources. Two transmit and one receive.
//
//
#pragma CODE_SEG NON_BANKED
interrupt void sci0_isr(void){ (void) software_trap(); }
#pragma CODE_SEG DEFAULT
    
```

software\_trap() function → User's Interrupt handler function

```

//*****
// SCI0 ISR
// DESCRIPTION:
// Interrupt asserted from one of five sources. Two transmit and one receive.
//
//
#pragma CODE_SEG NON_BANKED
interrupt void sci0_isr(void){ (void) sci0_handler(); }
#pragma CODE_SEG DEFAULT
    
```

# SCI Example

## Interrupt handler 추가 (계속)

Globals	234	0	•	•	☑
projectglobals.h	n/a	n/a			☑
projectvectors.c	234	0	•	•	☑
projectvectors.h	n/a	n/a			☑

Open 'projectvectors.h' file

```

#ifndef PROJECTVECTORS_H           /*prevent d
#define PROJECTVECTORS_H

/*Include Files*/
#include "projectglobals.h"

/*****
 * 사용자가 만든 핸들러들 포함시킴 */
#include "sci.h"

/*Local Prototypes*/
void software_trap(void);

#pragma CODE_SEG NON_BANKED
void unused_isr(void);
#pragma CODE_SEG DEFAULT
    
```

# SCI Example

## ■ sci.c 설명

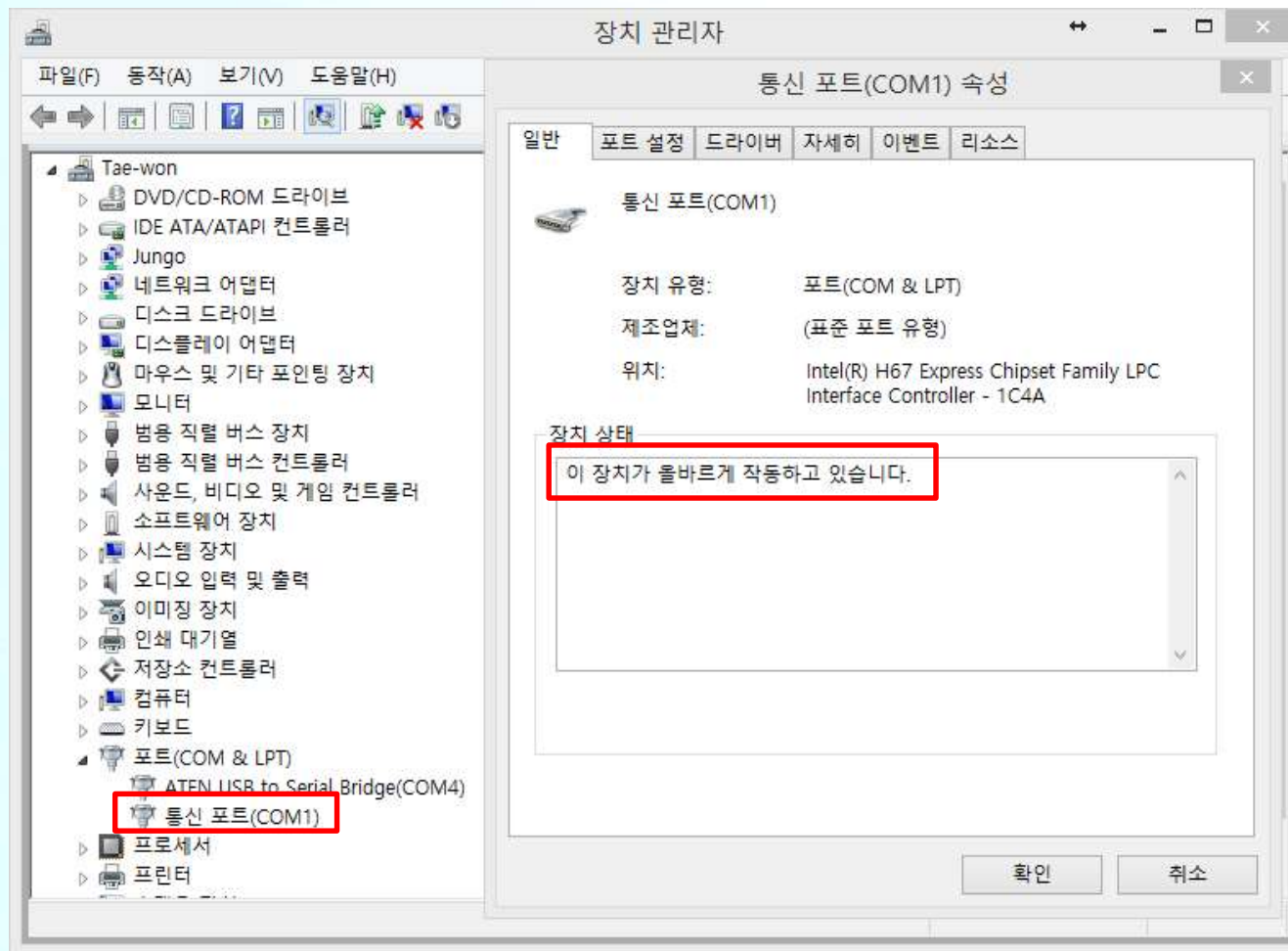
- void init\_sci0(int baud\_rate, char \*rxbuf);
  - baud\_rate : baud rate
  - rxbuf : receive buffer
  - 메시지 수신 인터럽트만 활성화
- Void write\_sci0(unsigned char \*text);
  - text : 전송할 메시지
  - 메시지 전송 함수
  - 전송할 메시지 버퍼에 저장
  - 메시지 송신 인터럽트 활성화
- void sci0\_handler(void);
  - Message transmit & receive interrupt handler
  - 메시지 송신 및 수신 인터럽트 플래그가 발생할 때마다 실행
  - SCI 관련 인터럽트가 반드시 projectvectors에 등록되어야 함



# 시리얼 통신 개론

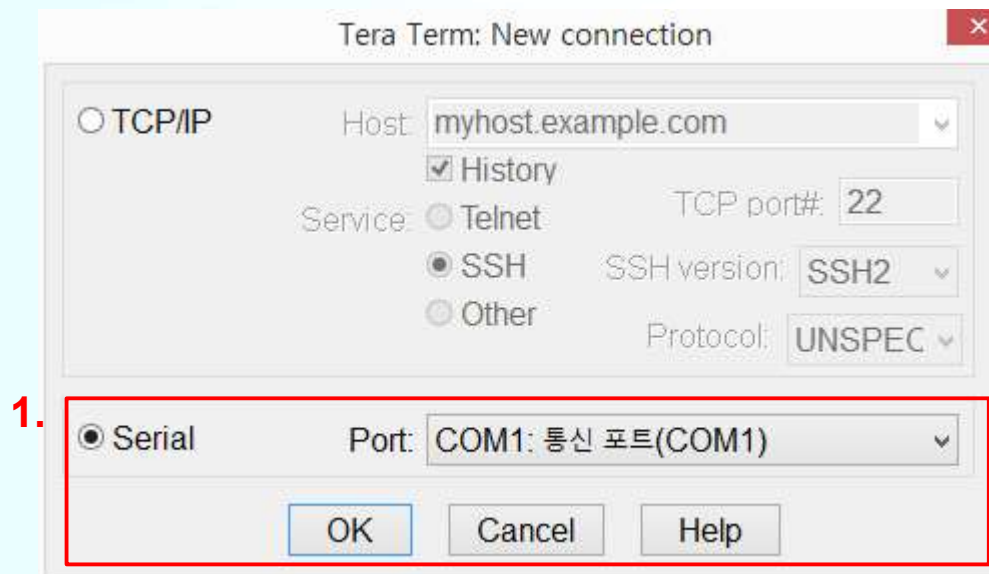
## ■ 하이퍼터미널

### ■ 장치 관리자에서 시리얼 포트 번호 및 상태 확인



# Tera Term 사용법

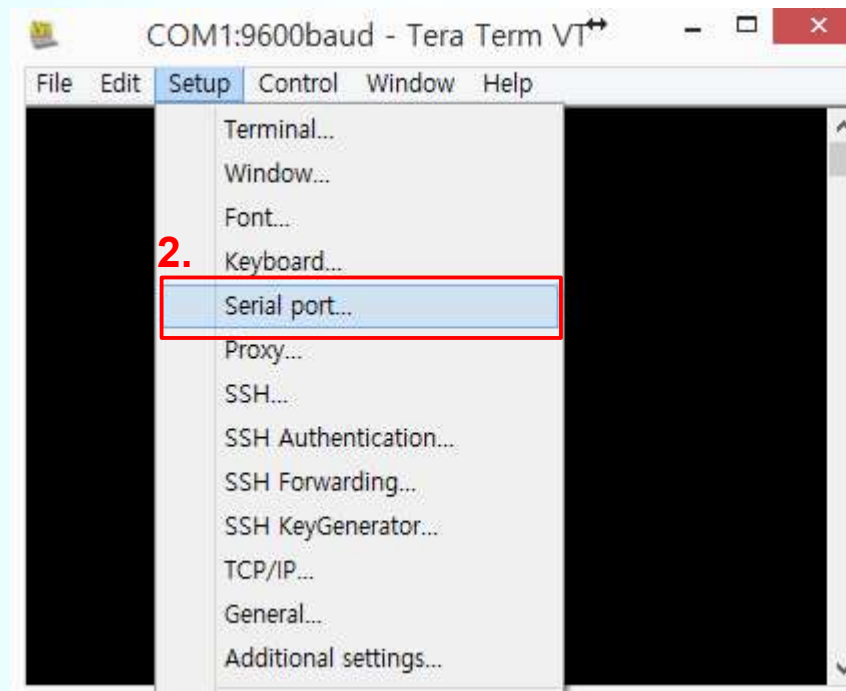
- 다운로드한 Tera Term 프로그램 실행
- Serial 모드로 전환하여 케이블이 연결된 포트 설정
- OK 클릭





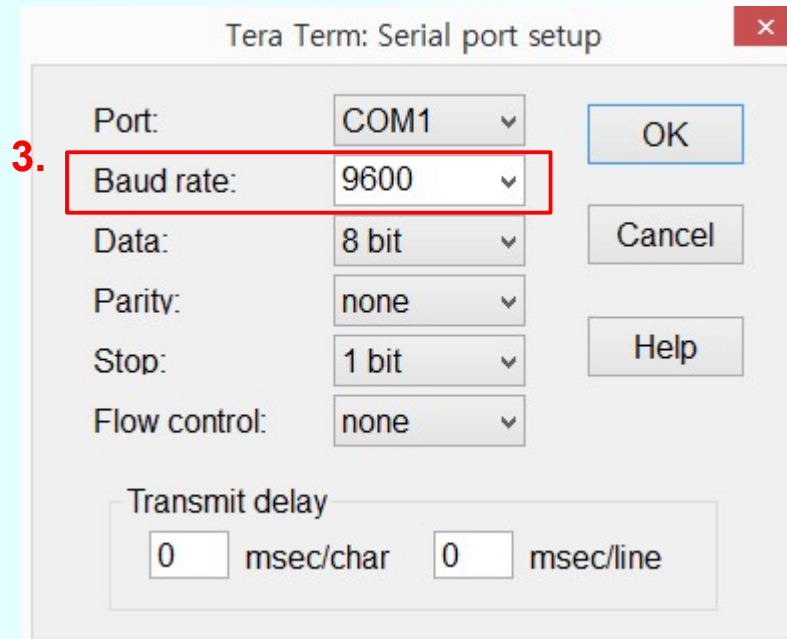
# Tera Term 사용법

- 상단 메뉴의 Setup 내 Serial port 클릭



# Tera Term 사용법

- Baud rate를 HCS12의 통신 속도와 동일하게 설정
- OK 클릭
- Serial 통신 통한 메시지 확인

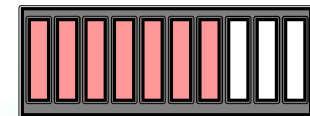
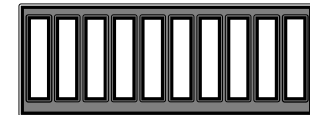
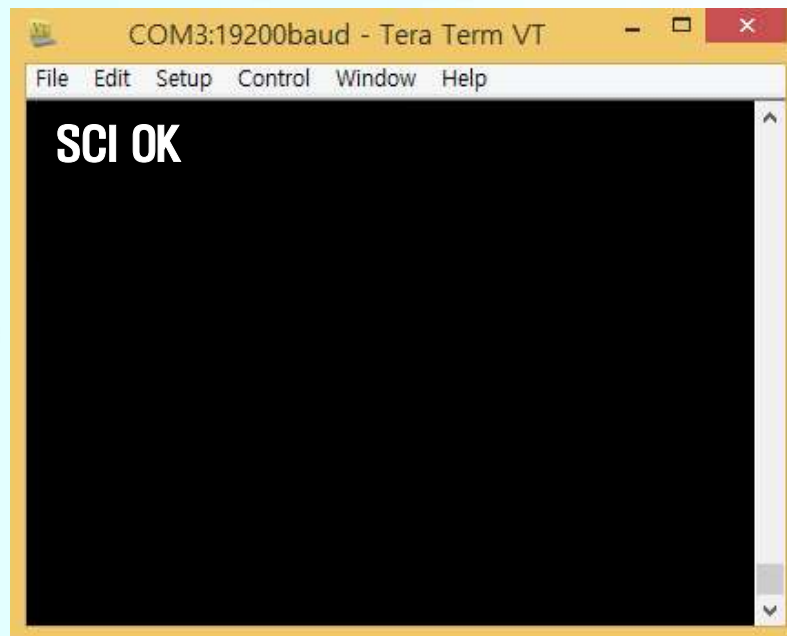


비트/초: **19200bps**  
데이터 비트 : **8bit**  
패리티: 없음  
정지 비트 : **1**  
흐름 제어: 없음

# SCI 모듈 실습

## ■ 실습 1

- PC용 시리얼 프로그램을 통해 PC에서 실습 보드로 숫자를 전송
- 실습 보드는 해당 숫자를 LED 적색 Bar 개수로 출력



# SCI 모듈 실습

## ■ 실습 2

- SW2를 누르면 알파벳 선택 (대/소문자 모두 선택 가능)
- SW3을 누르면 알파벳 선택 완료 및 다음 칸 알파벳 선택
- SW1을 누르면 설정한 문자열의 대/소문자 변경 뒤 PC용 시리얼 프로그램에 출력

