UART 와 LCD

마이크로프로세서

HRI 연구실

김동한

Human-Robot Interaction Laboratory

KYUNG HEE UNIVERSITY

6.3 LCD 제어의 이론 및 실습

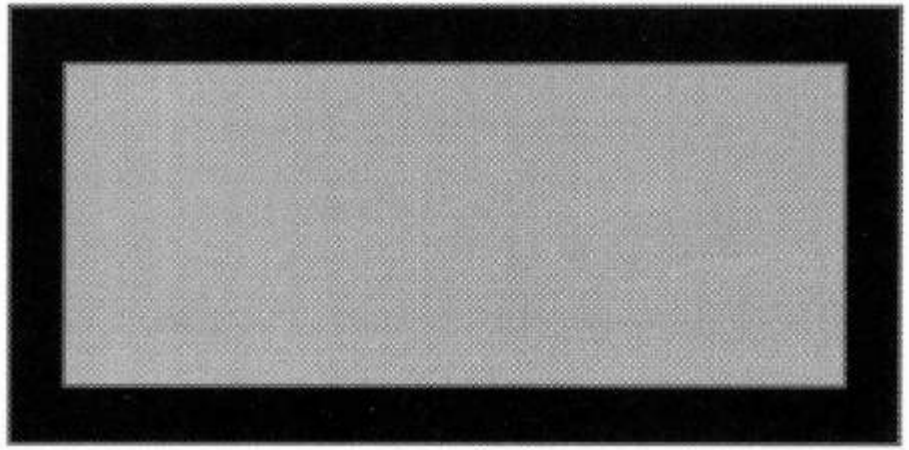
디스플레이의 구분

- CRT, FED : 전자가속 충돌에 의한 형광체
- PDP : 고전압에 의한 가스방전 방식의 디스플레이
- LED, OLED : 전류에 의한 자체발광 물질의 디스플레이
- LCD : 전기 신호에 의한 빛의 통과로 디스플레이

6.3 LCD 제어의 이론 및 실습

LCD (Liquid Crystal Display : 액정 표시 장치)

- LCD는 말 그대로 액정 디스플레이로서 액체와 고체의 중간상태인 액정의 전기 성질을 표시장치에 응용한 장치이다
- 본 교육 키트에 구비된 LCD는 character LCD 이며 A ~ z의 알파벳 영문자와 숫자, 기호 등을 표시할 수 있다.
- 문자는 Black 이며 , Yellow-green back light led로 구성되어 있다.
- 데이터 전송은 UART방식을 이용하여 문자를 표시할 수 있다.



8.1 통신의 개요

8.1.1 통신이란?

- 사람과 사람 사이의 의사소통이나 정보를 교환하는 것

- 데이터 통신
  - 전송한 데이터가 목적지에 정확하게 전달되도록 정해진 순서나 규칙에 따라 데이터를 보내고 받는 과정
  - 전보, 전화 등 전달하고자 하는 목적물을 오류 없이 통신회선을 통하여 2진수로 표시된 디지털 형태의 정보로 교신하는 것

- 데이터 통신 시스템(Data communication system)
  - 데이터 전송 링크를 연결하는 하드웨어와 소프트웨어가 결합된 형태
  - 정보의 이동을 담당하는 전송시스템과 정보의 가공, 처리 및 보관을 위한 데이터 처리시스템(컴퓨터)으로 나눔

8.1 통신의 개요

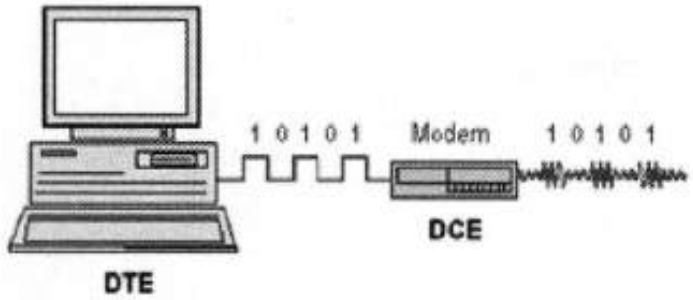
8.1.1 통신이란?

- 데이터 전송(data transmission)이란 정보통신 기기 간에 처리할 데이터를 네트워크 등의 전송매체를 사용하여 한 지점에서 다른 지점으로 보내는 것

- DTE(Data Terminal Equipment) : PC나 단말 등
- DCE(Data Communication Equipment) : 모뎀 등 통신 장치

8.1 통신의 개요

8.1.1 통신이란?



- 단말장치(DTE : Data Terminal Equipment)
  - 데이터 통신망에 연결된 사용자 장치의 일반적인 이름
  - 데이터의 전송 시스템에서 최종적으로 데이터를 송수신 하는 기능과 아울러 입출력 기능, 전송 제어 기능, 기억 기능을 갖고 있음
  - 대표적인 단말장치로는 컴퓨터 시스템이 있음

- DCE(Data Communication Equipment : 데이터 통신 기기)
  - 통신회선에 적합하도록 변환하는 기능과 통신 회선을 통해 수신된 데이터를 원래의 정보로 변환하는 기능을 갖는 장치
  - 일반적으로 컴퓨터가 모뎀이나 기타, 다른 직렬장치를 이용하여 데이터를 교환하기 위한 RS-232C 인터페이스

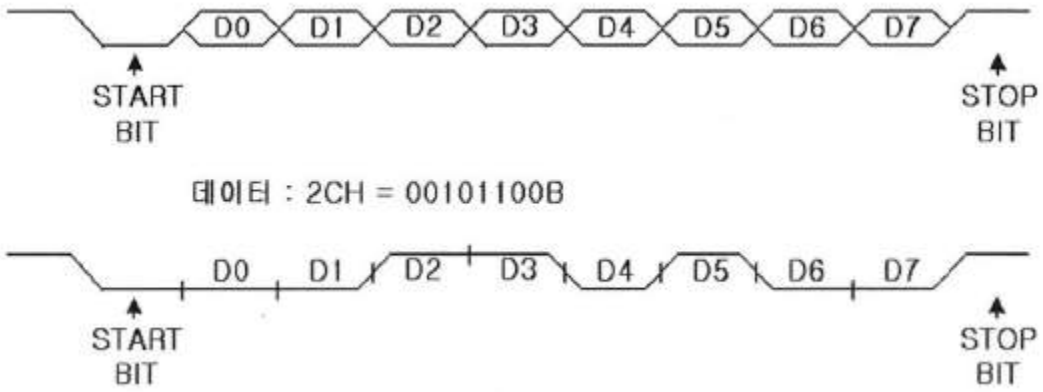
8.1 통신의 개요

8.1.2 통신의 분류

- **UART(Universal Asynchronous Receiver and Transmitter)**
  - 범용 비동기식 시리얼 통신 컨트롤러

8.1 통신의 개요

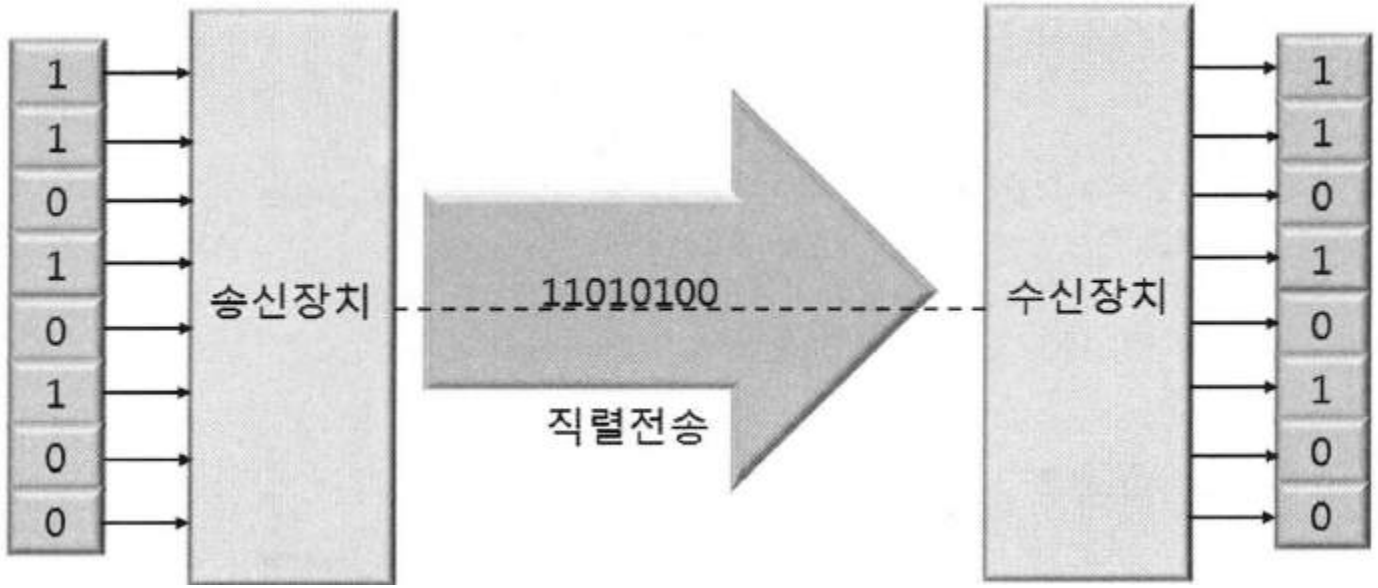
8.1.3 직렬통신과 병렬통신



- 데이터는 LSB부터 MSB 순으로 데이터 전송

8.1 통신의 개요

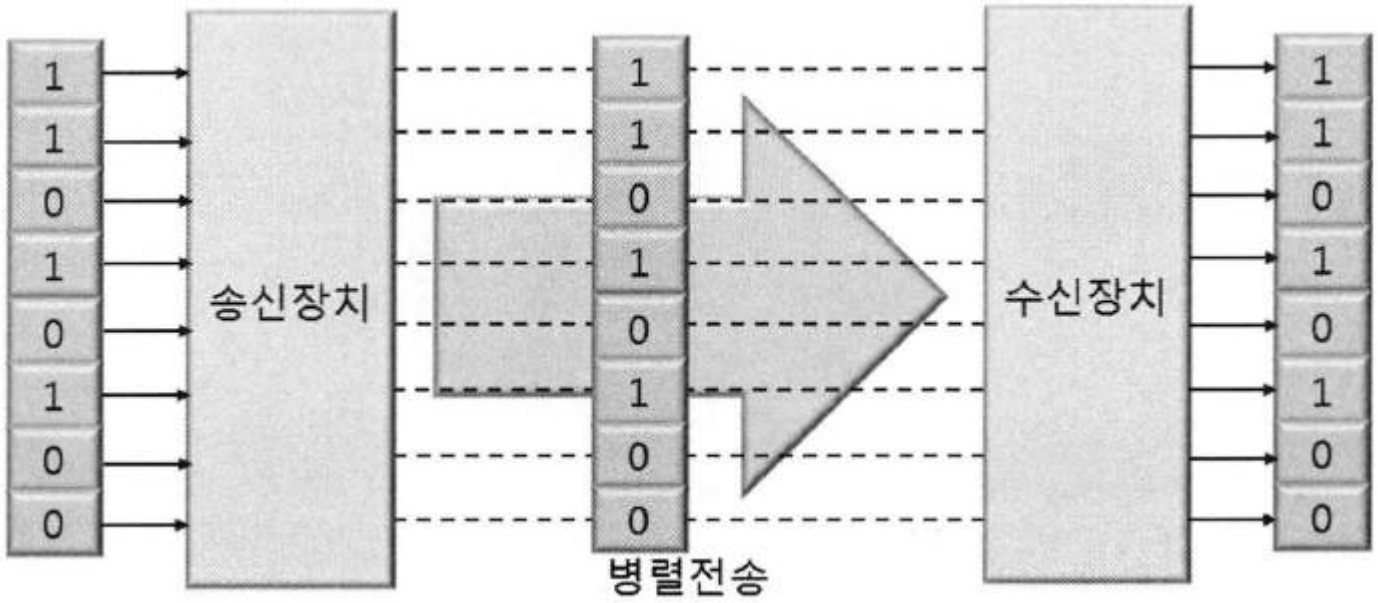
8.1.3 직렬통신과 병렬통신



- **직렬 통신**
  - 컴퓨터 간에 또는 컴퓨터와 주변 장치 간에 한 번에 한 비트(bit)씩 전송하는 통신방식

8.1 통신의 개요

8.1.3 직렬통신과 병렬통신

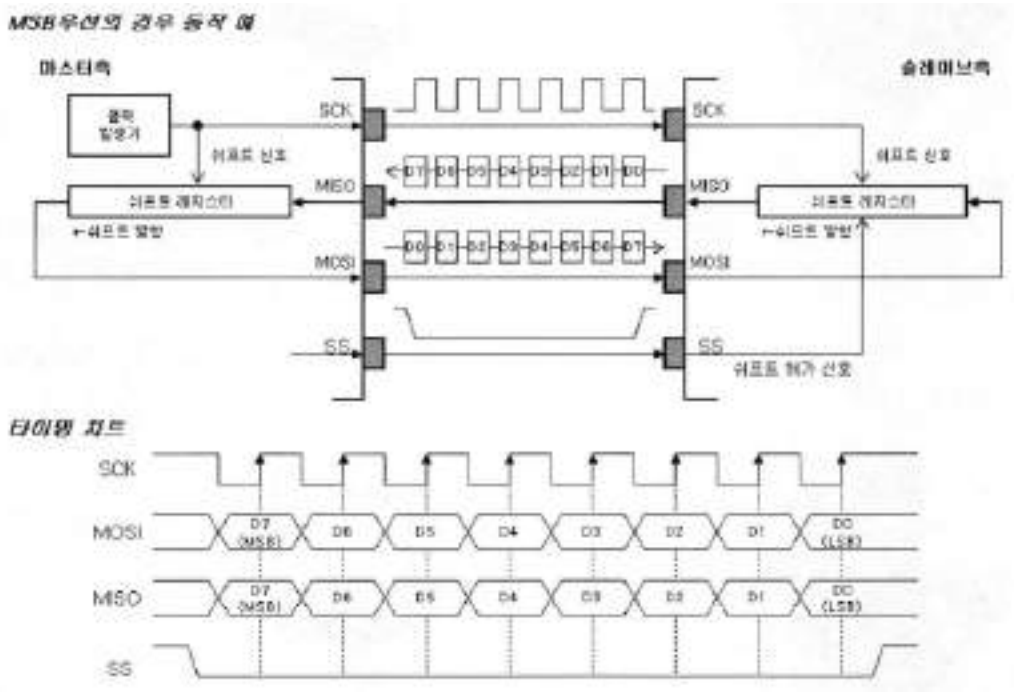


- **병렬 통신**
  - 컴퓨터 간에 보내고자 하는 신호를 몇 개의 선으로 나누어 여러 개의 데이터 비트(data bit)를 동시에 전송하는 방식

8.1 통신의 개요

8.1.4 동기통신과 비동기통신

- **동기(synchronous) 통신**
  - 데이터와는 별도로 송신 측과 수신 즉이 하나의 기준 클럭 (clock)으로 동기신호를 맞추어 동작
  - 이 때, 기준 클럭을 발생시키는 쪽을 마스터라고 하며, 받아서 동작하는 쪽을 슬레이브라고 함
  - 대표적인 동기 통신으로는 12C(혹은 TWI, IIC), SPI 통신 등이 있음
  - 실제 데이터가 전송되기 전에 동기 유지를 위한 문자(동기문자: sync or SCK)를 전송하여 수신 측에서는 이 동기문자를 인식하고 동기를 취함
  - 클럭에 의해 비트를 구별하게 되므로 동기식 전송을 위해서는 데이터와 클럭을 위한 2회선 이상이 필요

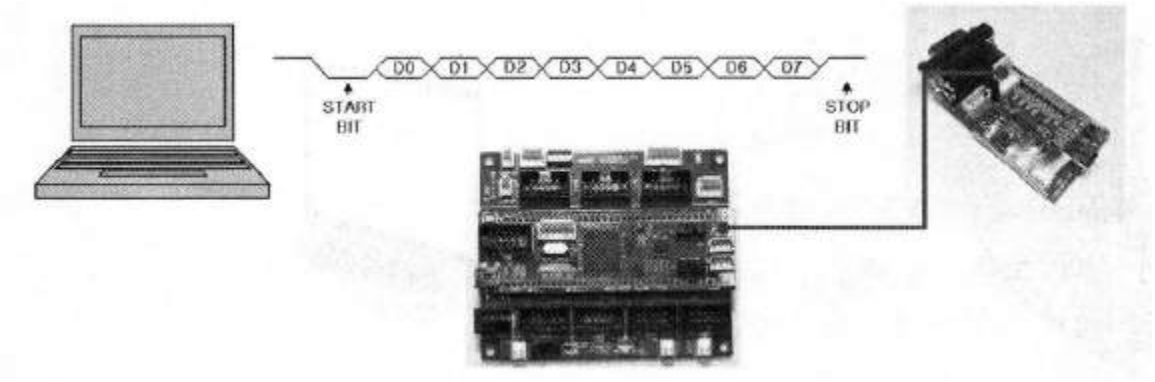


KYUNG HEE UNIVERSITY

8.1 통신의 개요

8.1.4 동기통신과 비동기통신

- **비동기(synchronous) 통신**
  - 송신 컴퓨터는 데이터의 시작을 나타내는 시작 비트(start bit)를 전송하고 데이터 블록이 라는 연속된 비트를 전송함
  - 각 블록의 끝은 하나 이상의 정지 비트(stop bit)를 전송
  - 비동기식 전송은 스타트 비트와 스톱 비트 사이의 간격이 가변적이므로 불규칙적인 전송 에 적합



8.1 통신의 개요

8.1.5 단방향 통신과 양방향 통신

- **단방향 통신(simplex communication)**
  - 오직 한 방향으로만 신호의 전송을 위한 통로가 설정되어 있는 통신 모드로 한 방향으로만 통신이 가능
  - 라디오 송신기, MUX 통신과 PWM 방식

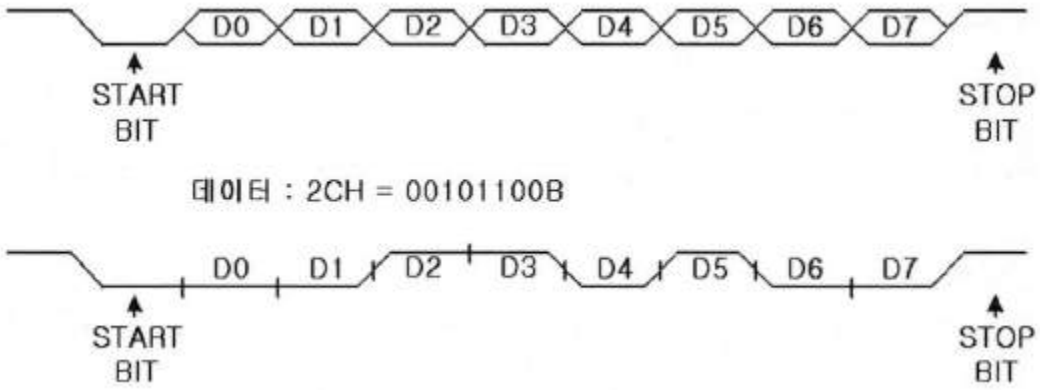
- **반 양방향 통신(Half-duplex communication)**
  - 반이중 모드라고도 하며 전송 통로는 하나이나 방향 선택을 할 수 있는 장치를 부착하여 양방향으로 통신이 가능한 통신 모드로 메시지를 보내고 받을 수 있지만 동시에 할 수는 없음
  - 무전기는 반 양방향 전송의 예, 무전기는 "talk" 버튼을 누르지 않으면 수신 모드에서 대기(CAN 통신)

- **완전 양방향 통신(Full-duplex communication)**
  - 전이중 모드라고도 하며 전송 통로를 통하여 메시지를 동시에 보내고 받는 것이 동시에 가능한 통신 모드
  - 전화는 완전 양방향 통신의 예이고, 대부분의 컴퓨터 통신에서 이용

8.1 통신의 개요

8.1.6 USART 통신

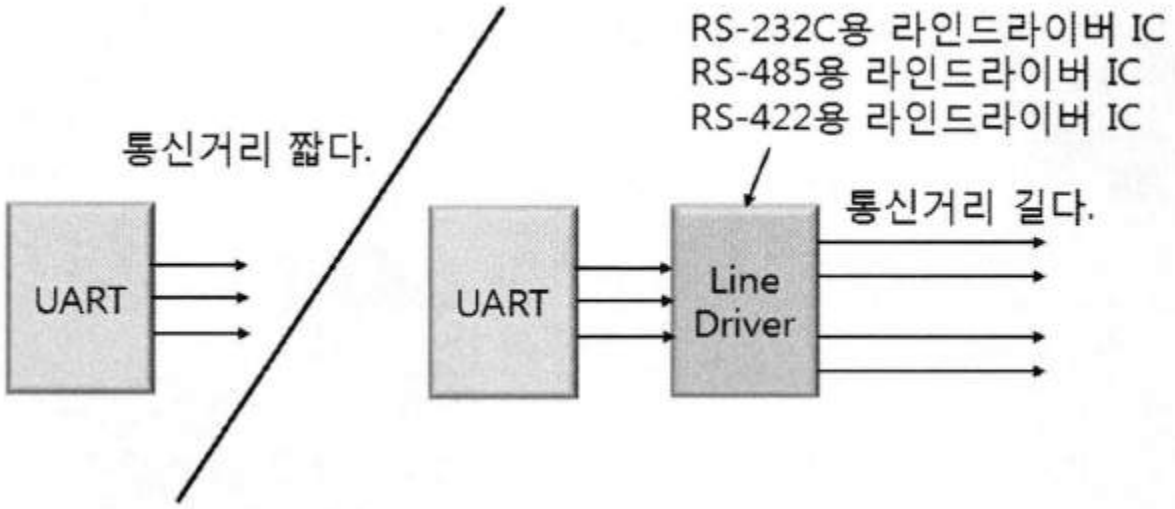
- 비트단위 데이터 전송
  - 송신단 : 1바이트를 8비트로 분리하여 한 번에 1비트씩 통신 선로로 전송
  - 수신단 : 통신 선로를 통해 수신한 비트들을 조립하여 1바이트를 만들어 냄
  - 1바이트 식별을 위해 'Start Bit'와 'Stop Bit'를 사용
  - ERROR 체크를 위해 Parity Bit를 제공하여 에러 검출
  - Parity Bit : 직렬통신시 송수신되는 데이터의 에러 유무를 알려주는 비트(Even parity, Odd parity, Mark & Space Parity)



8.1 통신의 개요

8.1.6 USART 통신

- **Line Transmitter & Receiver**
  - 직렬 출력 신호는 보통 TTL 신호레벨
  - TTL 신호는 노이즈에 약하고 통신거리에 제약
  - TTL 신호를 입력 받아 노이즈에 강하고 멀리 갈 수 있게 해주는 인터페이스 IC인 LINE DRIVER/RECEIVER를 사용
  - 대표적인 Line Driver/Receiver는 RS-232C, RS-422 및 RS-485가 있음



8.1 통신의 개요

8.1.7 RS232C 통신

- EIA(Electronic Industries Association)에 의해 규정
- 데이터단말기(DTE)와 데이터통신기(DCE) 사이의 인터페이스에 대한 규정
- 전기적인 인수, 컨트롤 핸드쉐이킹, 전송속도, 신호 대기시간, 임피던스 인수 등을 정의
- 전송되는 데이터의 포맷과 내용은 지정하지 않음
- DTE간의 인터페이스에 대한 내용도 포함하지 않음
- CCITI(Consultative Committee for International Telegraph and Telephony) 에서도 CCITI V.24에서 DTE와 DCE간의 상호 접속회로의 정의, 핀 번호와 회로의 의미 규정
- GND를 기준점으로 신호선의 크기를 정보로 사용
- 노이즈에 약하여 장거리 전송에 부적합
- 전이중(full duplex, 양방향)방식으로 직렬 접속
- 단지 3 개의 선으로 통신을 하며 Xon/Xoff라 불리는 소프트웨어 적인 방법으로 제어(핸드쉐이크)
- 추가로 선을 사용하여 하드웨어적으로 제어할 수 있음
- Baud Rate: 변조율이나 1초간 통신선의 신호 변경 회수를 가리키는 단어로서 2개의 시리얼 디바이스를 접속한 경우에는 Baud와 BPS는 사실상 똑같음

중요. UART 통신

- UART(Universal Asynchronous Receiver/Transmitter)
  - 시리얼 기반의 통신 방식으로 일반적으로 RS232 프로토콜을 통해 원격지와 통신을 지원하는 방식
  - UART는 컴퓨터에게 RS-232C DTE 인터페이스를 제공함으로써, 모뎀이나 기타 다른 직렬장치들과 통신하거나 데이터를 주고받을 수 있게 한다.
  - UART의 동작
    - 병렬 데이터를 하나의 단일 직렬 비트 스트림으로 변환
    - 직렬 비트 스트림을 컴퓨터가 처리할 수 있도록 바이트로 변환
    - 패리티 비트 처리
    - 시작 비트와 정지 비트 처리
    - 키보드나 마우스로부터 들어오는 인터럽트 처리

중요. UART 통신

- RS-232C
  - 직렬전송을 위한 규격
  - 1969년 미국의 EIA (Electric Industries Associantion)에 의해서 정해진 표준 인터페이스
  - "직렬 2진 데이터의 교환을 하는 데이터 터미널 장비(DTE)와 데이터 통신장비(DCE)간의 인터페이스의 제반을 규정하는 것"
  - RS-232C의 동작
    - 병렬을 직렬로 직렬을 병렬로 바꾸어 주는 작업
    - 스타트 비트와 스톱비트 포함하여 10비트를 1바이트로 보냄
    - RXD, TXD 라인을 통해 신호를 송수신
    - RS232 Transceiver를 통해 전송 전압을 끌어 올려 보다 먼 거리까지 전송

중요. UART 통신

- RS-232C를 이용한 비동기식 전송시 규약
  - 통신속도
    - 시간당 데이터를 전송할 수 있는 양
    - baud rate : 1초당 전송되는 변조된 신호의 수
  - 스톱비트
    - 데이타의 시작과 끝을 알리는 스타트와 스톱 비트를 사용
    - 전송을 시작할 경우 1을 내보내고 8비트를 전송한 후 스톱비트를 전송
    - 스타트 비트는 고정/ 스톱 비트는 1과 1.5, 2비트중 하나를 선택.
  - 패리티
    - 오류 검출을 위해 사용
    - 패리티의 종류는 짝수 및 홀수 방식과 사용하지 않는 경우가 있다.
    - 데이타 길이가 7인 경우에 8번째 비트를 패리티 비트로 이용
  - 자료 길이
    - 하나의 데이타를 전송하는데 필요한 데이타 길이(비트 수)
    - 보통 7과 8 비트가 있다.

중요. UART 통신

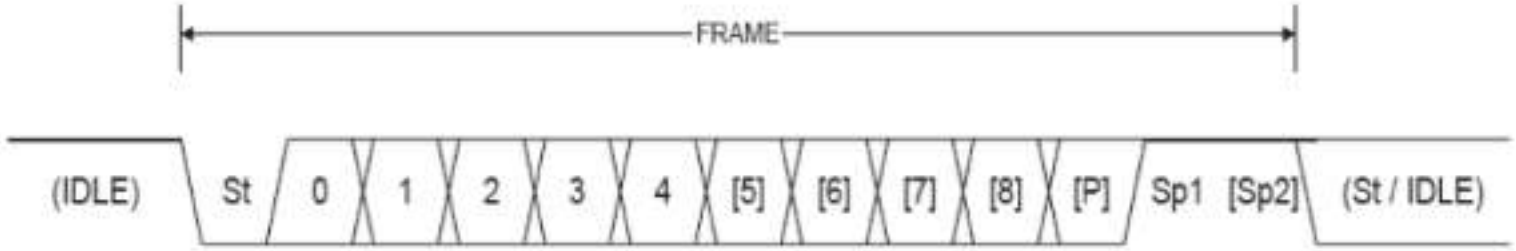
- ATmega128의 직렬통신 포트
  - 직렬통신포트 USART(Universal Synchronous and Asynchronous Receive and Transmitter) 2개 내장
    - USART0
    - USART1
  - 완전 이중방식(Full-Duplex)
  - 동기 및 비동기 전송 가능
  - 멀티 프로세서 통신 모드로 동작 가능
  - 높은 정밀도의 보레이트 발생기 내장
  - 인터럽트
    - 송신 완료(TX Complete)
    - 송신 데이터 레지스터 준비완료(TX Data Register Empty)
    - 수신완료(RX Complete)

중요. UART 통신

ATMega128 USART 데이터 프레임 포맷

- 최소 7비트 최대 13비트로 구성
  - (1 비트의 스타트 비트) +(5,6,7,8,9 비트의 데이터 비트) + (0, 1 비트의 패리티 비트) + (1,2 비트의 스탑비트) 프레임

USART 통신의 데이터 프레임



중요. UART 통신

- ATMega128 USART 데이터 프레임 포맷
  - 스타트 비트
    - 1비트로 이루어 졌으며 항상 0레벨이다. 송신시에 자동적으로 생성된다.
  - 데이터 비트
    - 5,6,7,8,9비트가 가능하다.
  - 패리티 비트
    - 패리티를 사용하지 않을 수도 있고 사용하는 경우 홀수 혹은 짝수 패리티 1비트를 사용한다.
  - 스톱 비트
    - 1,2개의 비트가 가능하며 항상 1레벨이다. 송신시에 자동적으로 생성된다.

중요. UART 통신

- ATMega128 USART 레지스터
  - UDRn(Usart i/o Data Register n)
    - USART I/O 데이터 레지스터 (UDR0, UDR1)
  - UCSRnA(Usart Control and Status Register n A)
    - USART 제어 및 상태 레지스터 A
  - UCSRnB(Usart Control and Status Register n B)
    - USART 제어 및 상태 레지스터 B
  - UCSRnC(Usart Control and Status Register n C)
    - USART 제어 및 상태 레지스터 C
  - UBRRnH/L (USART BAUD RATE REGISTER) :
    - USART baud Rate 레지스터

중요. UART 통신

- UDRn(Usart i/o Data Register n)
  - USART I/O 데이터 레지스터 (UDR0, UDR1)
  - USARTn모듈의 송수신 데이터 버퍼의 기능을 수행하는 8비트 레지스터(n= 0, 1)
    - 송신 데이터를 UDRn에 write하면, 송신 데이터 버퍼 TXB에 저장
    - 수신 데이터를 DRn에서 읽으면 수신 데이터 버퍼 RXB에 수신되어 있는 값이 읽힘

7	6	5	4	3	2	1	0
RXB7	RXB6	RXB5	RXB4	RXB3	RXB2	RXB1	RXB0
TXB7	TXB6	TXB5	TXB4	TXB3	TXB2	TXB1	TXB0

중요. UART 통신

- UCSRnA(Usart Control and Status Register n A)
  - USART 제어 및 상태 레지스터 A
  - USARTn모듈의 송수신 동작을 제어하거나 송수신 상태를 저장하는 기능을 수행하는 8비트 레지스터

7	6	5	4	3	2	1	0
RXCn	TXCn	UDREn	FEn	DORn	PEn	U2Xn	MPCMn

  - 비트 7 : RXCn (USARTn Receiver Complete)
    - 수신버퍼의 상태 플래그
    - 수신버퍼에 수신문자가 있으면 “1” 로 세트
    - 수신 버퍼가 비어있는 상태라면 “0” 으로 클리어

중요. UART 통신

- UCSRnA(Usart Control and Status Register n A)
  - 비트 6 : TXCn (USARTn Transmit Complete)
    - 송신버퍼의 상태 플래그
    - 송신 시프트 레지스터에 있는 송신 데이터가 모두 송신되고 UDRn의 송신 버퍼에 아직 새로운 데이터가 저장되지 않은 상태이면 “1” 로 세트
  - 비트 5 : UDREn (USARTn Data Register Empty)
    - 새로운 송신 데이터를 받기 위한 상태 플래그
    - UDRn의 송신 버퍼에 새로운 송신 데이터를 받을 준비가 되어 있으면 “1” 로 세트
  - 비트 4 : FEn (Frame Error)
    - 수신 프레임 에러 상태 플래그
    - UDRn의 수신 버퍼에 현재 저장되어있는 데이터를 수신하는 동안에 프레임에러가 발생했음을 나타냄

중요. UART 통신

- UCSRnA(Usart Control and Status Register n A)
  - 비트3 : DORn (Data Overrun Error)
    - 수신동작 오버런 에러 상태 플래그
    - 수신동작에서 오버런에러가 발생했음을 나타내는 상태 플래그
  - 비트 2 : UPEn (USARTn Parity Error)
    - UDR의 수신버퍼에 현재 저장되어 있는 데이터를 수신하는 동안에 패리티 에러가 발생하였음을 나타내는 상태 플러그
  - 비트 1 : U2Xn (Double the USARTn Transmission Speed)
    - 비동기 모드에서만 사용가능,
    - 클록의 n분주비를 16에서 8로 1/2만큼 낮추어 전송속도를 2배 높이는 기능
  - 비트 0 : MPCMn (USARTn Multi-Processor Communication Mode)
    - USARTn을 멀티프로세서 통신모드 설정

중요. UART 통신

- UCSRnB(Usart Control and Status Register n B)
  - USART 제어 및 상태 레지스터 B
  - USART 모듈의 송수신 동작 제어/송수신 상태 저장
  - USART0, USART1 포트의 송수신 동작제어, 전송 데이터의 9번째 비트값 저장

7	6	5	4	3	2	1	0
RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n

  - 비트 7 : RXCIEn (USARTn RX Complete Interrupt Enable)
    - 수신완료 인터럽트를 개별적으로 enable
    - 이 비트를 "1" 로 설정하고 SREG레지스터의 I비트가 "1" 이고, UCSRnA 레지스터의 RXCn비트가 "1" 로 설정되어 있으면 수신완료 인터럽트가 발생

중요. UART 통신

- UCSRnB(Usart Control and Status Register n B)
  - 비트 6 : TXCIEn (USARTn TX Complete Interrupt Enable)
    - 송신완료 인터럽트를 개별적으로 enable
    - 이 비트를 “1” 로 설정하고 SREG레지스터의 I비트가 “1” 이고, UCSRnA 레지스터의 RXCn비트가 “1” 로 설정되어 있으면 송신완료 인터럽트가 발생
  - 비트 5 : UDRIEn (USARTn Data Register Empty Interrupt Enable)
    - 송신 데이터 레지스터 준비완료 인터럽트(Data Register Empty)를 개별적으로 Enable
    - “1” 로 설정하고 SREG레지스터의 I비트가 “1” 이고, UCSRnA 레지스터의 UDREn비트가 “1” 로 되면, USARTn Data Register Empty 인터럽트가 발생
  - 비트 4 : RXENn (USARTN Receiver Enable)
    - USARTn 모듈의 수신부가 동작하도록 enable
    - RXDn 핀이 병렬 I/O포트가 아니라 직렬 데이터 수신단자로 동작하도록 설정

중요. UART 통신

UCSRnB(Usart Control and Status Register n B)

- 비트 3 : TXENn (USARTn Transmitter Enable)
  - USARTn 모듈의 송신부가 동작하도록 enable
  - TXDn 핀이 병렬 I/O포트가 아니라 직렬 데이터 송신단자로 동작하도록 설정
- 비트 2 : UCSZn2 (USARTn Character Size)
  - UCSRnC레지스터의 UCSZn1~0비트와 함께 전송문자의 데이터 비트수를 설정
- 비트 1 : RXB8n (USARTn Receiver Data 8Bit)
  - 수신문자가 9비트로 설정된 경우에 수신된 문자의 9번째 비트를 저장
- 비트 0 : TXB8n (USARTn Transmit Data 8Bit)
  - 송신문자가 9비트로 설정된 경우에 송신된 문자의 9번째 비트를 저장

중요. UART 통신

- UCSRnC(Usart Control and Status Register n C)
  - USART 제어 및 상태 레지스터 C
  - USARTn 모듈의 송수신 동작을 제어하거나 송수신 상태를 저장

7	6	5	4	3	2	1	0
–	UMSELn	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn

  - 비트 6 : UMSELn(USARTn Mode Select)
    - USART 모드 설정
    - “1” 이면 USARTn 모듈을 동기 전송모드로 설정하고, “0” 이면 비동기 전송 모드로 설정

중요. UART 통신

- UCSRnC(Usart Control and Status Register n C)
  - 비트 5,4 : UPMn1,0 (USARTn Parity Mode)
    - 패리티 모드 설정
    - UPMn1비트를 "1" 로 설정하면 패리티를 발생
    - 오류가 발생하면 UCSRnA 레지스터의 PE플래그가 "1" 로 세트

패리티 모드 설정표

UPMn1	UPMn0	Parity모드
0	0	Disable
0	1	예약
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

중요. UART 통신

- UCSRnC(Usart Control and Status Register n C)
  - 비트 2,1 : UCSZn1,0(USARTn Character Size)
    - UCSRnB 레지스터의 UCSZn2 비트와 함께 전송문자의 데이터 비트수를 설정

UCSZn에 의한 character Size 설정표

UCSZn2	UCSZn1	UCSZn0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	예약
1	0	1	예약
1	1	0	예약
1	1	1	9-bit

중요. UART 통신

UCSRnC(Usart Control and Status Register n C)

- 비트 0 : UCPOLn (Usart Clock POLarity n)
  - 동기 전송 모드의 슬레이브 동작에서만 유효
  - '1' 로 설정하면 송신 데이터는 클럭의 하강에지에서 새로운 XCKn 값이 출력되고, 수신 문자는 XCKn의 상승에지에서 얻어진다.
  - '0' 로 설정하면 반대로 된다.

UBRRnH/L (USART BAUD RATE REGISTER)

- USART baud Rate 레지스터
- USARTn 모듈의 송수신 속도를 설정
- 16비트중에서 12비트만 사용

15	14	13	12	11	10	9	8
–	–	–	–	UBRRn	UBRRn	UBRRn	UBRRn
7	6	5	4	3	2	1	0
UBRRn7	UBRRn6	UBRRn5	UBRRn4	UBRRn3	UBRRn2	UBRRn1	UBRRn0

- 비트 11~0 : UBRRn11~0
  - 12비트를 이용하여 USARTn의 Baud Rate을 결정
  - UBRRnH의 4비트와 UBRRnL의 8비트가 조합을 이룸.

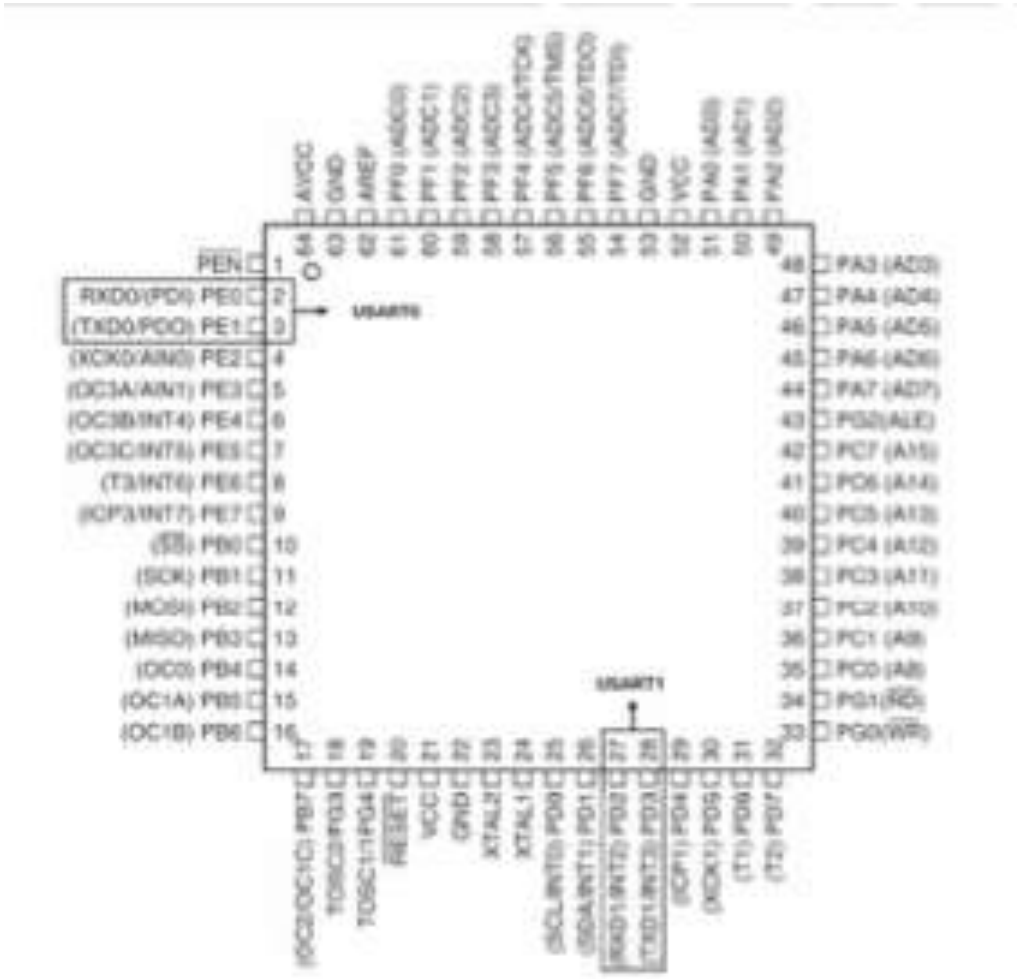
중요. UART 통신

□ UBRRnH/L (USART BAUD RATE REGISTER)

UBRR에 의한 Baud Rate 설정표

Baud Rate (bps)/7.3728MHz	비동기 일반모드 (U2Xn = 0)		비동기 2배속 모드 (U2Xn=1)	
	UBRRn	Error	UBRRn	Error
2400	191	0.0%	383	0.0%
4800	95	0.0%	191	0.0%
9600	47	0.0%	95	0.0%
14,400	31	0.0%	63	0.0%
19,200	23	0.0%	47	0.0%
28,800	15	0.0%	31	0.0%
38,400	11	0.0%	23	0.0%
57,600	7	0.0%	15	0.0%
76.800	5	0.0%	11	0.0%
115,200	3	0.0%	7	0.0%
230,400	1	0.0%	3	0.0%
250,000	1	-7.8%	3	-7.8%
500,000	0	-7.8%	1	-7.8%

중요. UART 통신



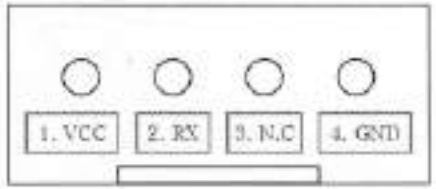
6.3 LCD 제어의 이론 및 실습

실습.1 UART0 데이터 포트에 LCD 하드웨어 설정하기.

(1) 하드웨어 연결하기

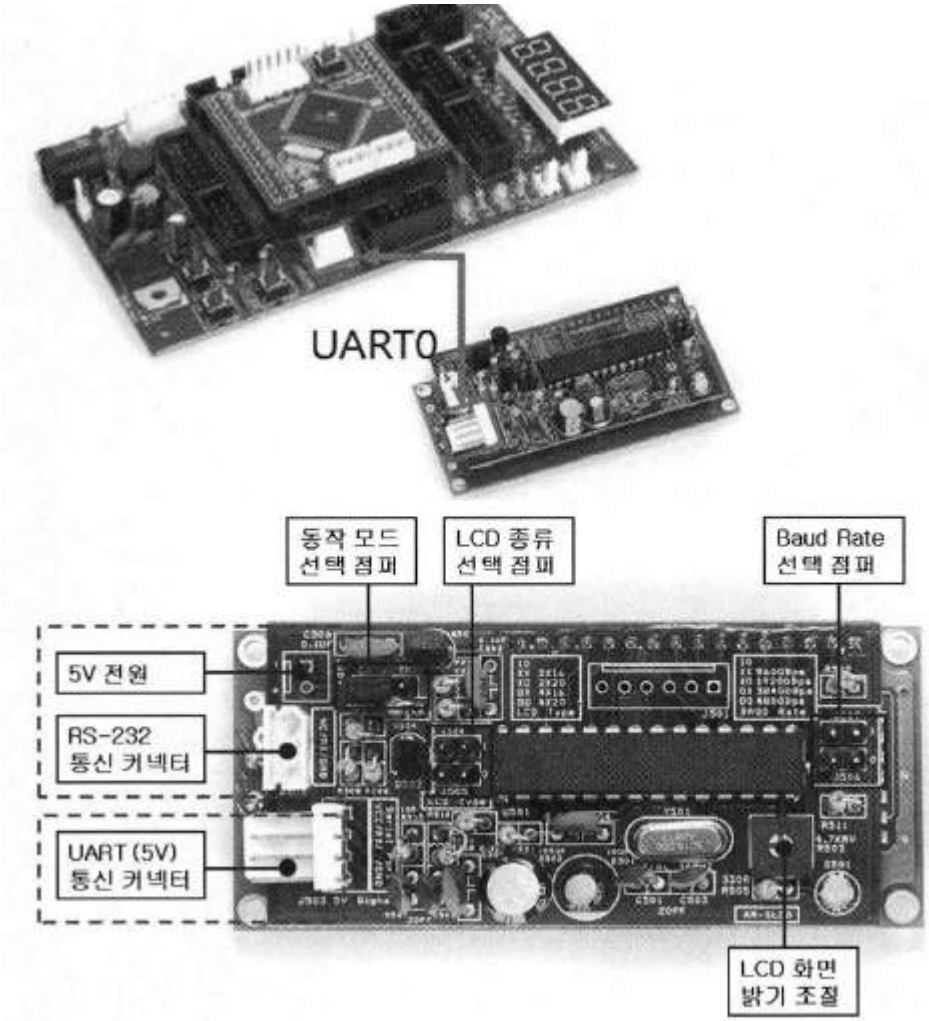
- **ATmega128의 UART0인 데이터 통신포트를 이용하여 LCD를 제어한다.**
- **UART 커넥터를 사용할 경우 데이터 입력과 5V 신호를 커넥터를 통하여 공급 받는다.**

- **제어 신호는 커맨드모드로 printf 를 이용하여 Serial 포트로 출력을 하는 컴파일러를 사용하면 printf를 사용한다.**



커넥터 배선 방법 (본 커넥터로 입력 되는 신호는 TTL Level (5V)입니다.)

1	VCC (5V 출력)
2	UART RX (시리얼 데이터 수신)
3	No Connect
4	GND



6.3 LCD 제어의 이론 및 실습

실습.1 UART0 데이터 포트에 LCD 하드웨어 설정하기.

(2) CodeWizardAVR를 이용한 LCD UART0 통신 설정

- CodeWizardAVR가 ATmega128 에 LCD를 UART0통신을 통하여 연결하는 초기 프로그램을 작성함.
- 만들어진 코드를 main문 안에 넣어 사용.


```
// USART0 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART0 Receiver: Off
// USART0 Transmitter: On
// USART0 Mode: Asynchronous
// USART0 Baud Rate: 9600
UCSR0A=(0<<RXC0) | (0<<TXC0) | (0<<UDRE0) | (0<<FE0) | (0<<DOR0) | (0<<UPE0) | (0<<U2X0) | (0<<MPCM0);
UCSR0B=(0<<RXCIE0) | (0<<TXCIE0) | (0<<UDRIE0) | (0<<RXEN0) | (1<<TXEN0) | (0<<UCSZ02) | (0<<RXB80) | (0<<TXB80);
UCSR0C=(0<<UMSEL0) | (0<<UPM01) | (0<<UPM00) | (0<<USBS0) | (1<<UCSZ01) | (1<<UCSZ00) | (0<<UCPOL0);
UBRR0H=0x00;
UBRR0L=0x67;
```

6.3 LCD 제어의 이론 및 실습

실습.1 UART0 데이터 포트에 LCD 하드웨어 설정하기.

(3) Printf를 이용한 초기 프로그램 설정하기

- #include <stdio.h> : print문을 사용하기 위해 인클루드 해야함.

- 백라이트를 끄고 실습하는 것을 권장함
- UART 포트에 LCD 연결 시 전원 끌 것!

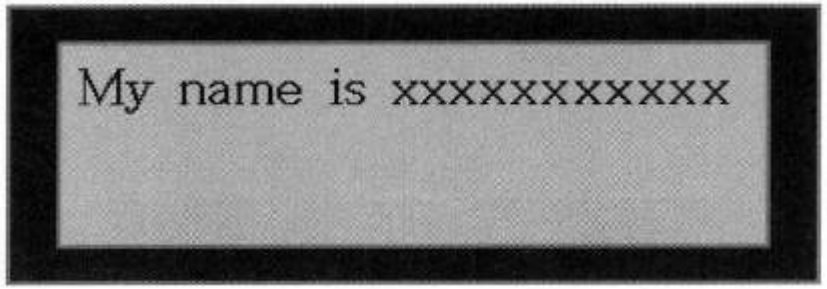
```
printf("$T, MICRO PROCESSOR \r"); //LCD화면에 텍스트 출력
printf("$T, Test Data: %d\r", 1);
printf("$C\r"); //LCD 화면 지우기
printf("$G, 1, 1\r"); //커서를 1,1위치로 이동
printf("$L,1\r");      //백라이트 켜기
printf("$L,0\r");      //백라이트 끄기
```

구분	명령	Data1	Data2	End	Example
초기화	$I			<CR>	$I<CR>
화면 Clear	$C			<CR>	$C<CR>
커서 위치 지정	$G	행 위치(1-4)	열 위치(1-20)	<CR>	$G,1,1<CR>
문자열 출력	$T	Text		<CR>	$T,Testing...<CR>
커서 OFF/ON/Blink	$B	0/1/B		<CR>	$D,1<CR>
Display(On/Off)	$D	1/0		<CR>	$L,1<CR>
Back Light (On/Off)	$L	1/0		<CR>	$L,1<CR>
Display Shift Left/Right	$S	L/R		<CR>	$S,R<CR>

6.3 LCD 제어의 이론 및 실습

실습.2 자기소개 My name is ( ) 가 표현될 수 있도록 프로그램

(1,1) (1,16)



- 100ms 마다 특정 문자가 왼쪽에서 오른쪽으로 Shift 한다
- (1, 16) 까지 Shift 후 다시 (1, 1)로 이동한다

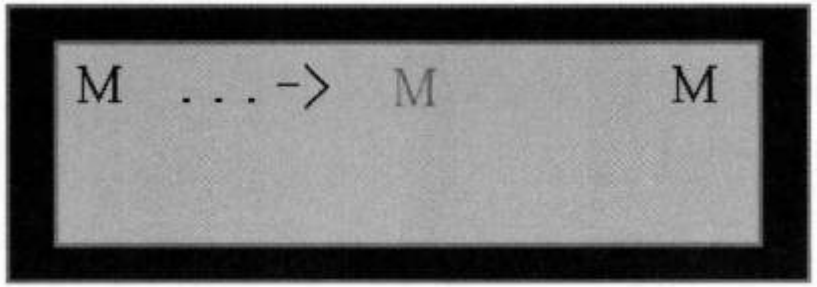
```
printf("$I\r");

while(1)
{
    // Place your code here

    printf("$G, 1, 1\r");
    printf("$T, My name is xxxxxxxxxxx\r);
}
```

KYUNG HEE UNIVERSITY

6.3 LCD 제어의 이론 및 실습

실습.3 LCD에서 오른쪽으로 M이 이동하는 프로그램을 작성하시오

(1,1) (1,16)



- 100ms 마다 특정 문자가 왼쪽에서 오른쪽으로 쉬프트한다.
- 초기 문자는 (1, 1)에서 출발한다.
- (1, 16) 까지 쉬프트 후 다시 (1, 1)로 이동한다

```
while(1)
{
    // Place your code here

    printf("$C\r");
    printf("$G, 1, %d\r", i++);
    printf("$T, M\r);

    if( i >= 16 ) i = 1;

    delay_ms(100);
}
```

6.3 LCD 제어의 이론 및 실습

[과제6] LCD를 이용한 디지털 시계 제작하기

(1) CodeWizardAVR를 이용한 디지털시계 초기 설정

- Timer0를 이용하여 디지털시계에서 가장 기초가 되는 1초를 만든다.
- LCD를 마이크로프로세서에 연결하는 UART0을 초기화하는 작업 수행.
- 이 때 생성되는 프로그램은 타이머루틴과 UART0값이 설정된다.
- 타이머 분주비는 250KHz(1/64)로 설정한다. (1/128 로 해도 상관없음)

1.MCU(ATmega128_의 기본 클럭 : 16MHz, 인터럽트 주기는 1ms 로 설정한다.

2. 분주비 (Prescaler) 를 이용한 Timer0의 클럭 설정 : 16MHz/분주비

   → 16MHz(1), 2MHz(8), 500kHz(32), **250kHz(64), 125kHz(128),** 62.5kHz(256), 15.625kHz(1024)

4. 클럭의 주기 : 1/클럭 62.5ns(1), 500ns(8), 2us(32), **4us(64), 8us(128),** 16us(256), 64us(1024)

5. 오버플로우가 발생하려면 최대 256개의 클럭이 필요 (8 bit, 256번 : 0 → 1 → ... → 255 → 0)

6. 각 주기의 최대 인터럽트 시간 : 주기*256

   → 16us(1), 128us(8), 512us(32), **1.024ms(64), 2.048ms(128),** 4.096ms(256), 16.384ms(1024)

7. Timer/Counter 0에서는 분주비 1, 8, 32는 불가능하고 64,128,256,1024 사용 가능함.

8. 각 클럭의 주기에 따라 카운터하는 클럭의 개수 계산 : 1ms/주기

   → **250번(64), 125번(128),** 62.5번(256), 31.25번(1024)

9. 분주비 256, 1024는 정확하지 않으므로 제외, 분주비 64, 128 중 선정

10. 분주비에 따른 카운터 횟수 설정 : 256 - 250 = 6 (64), 256 - 125 = 131 (128)

KYUNG HEE UNIVERSITY

6.3 LCD 제어의 이론 및 실습

[과제6] LCD를 이용한 디지털 시계 제작하기

(2) Timer0에 의해 설정된 interrupt routine 이해하기

- cnt를 1씩 증가시켜 1000 번 증가시키면 이 때 one_sec_flag 가 활성화 되면서 1초가 되어 one_sec_cnt가 1 이 증가한다.
- 지속적으로 one_sec_cnt 가 증가하고 60이 되면 다시 초기화되고, 이때에 one_min_cnt 가 다시 1이 증가한다.
- 그리고 one_min_cnt가 60이 되면 다시 초기화된다.
- 이 결과를 지속적으로 반복하면 60분 60초를 나타내는 프로그램이 완성된다.
- 타이머 문은 main 문 밖에 작성한다.(타이머 인터럽트)

```
23  #include <io.h>
24  #include <mega128.h>
25  #include <stdio.h>
26
27  bit one_sec_flag;
28  int cnt, one_min_cnt = 0, one_sec_cnt = 0;
29
30  interrupt [TIM0_OVF] void timer0_ovf_isr(void)
31  {
32  // Reinitialize Timer 0 value
33  TCNT0=0x06;
34  // Place your code here
35      cnt++;
36      if(cnt >= 1000)
37      {
38          cnt = 0;
39          one_sec_flag = 1;
40          one_sec_cnt++;
41          if(one_sec_cnt == 60)
42          {
43              one_sec_cnt = 0;
44              one_min_cnt++;
45              if(one_min_cnt == 60)
46              {
47                  one_min_cnt = 0;
48              }
49          }
50      }
51  }
```

6.3 LCD 제어의 이론 및 실습

[과제6] LCD를 이용한 디지털 시계 제작하기

(3) LCD에 분:초 시계 나타내는 프로그램 결과

#asm("sei") 는 어셈블리 언어로 모든 인터럽트를 허용한다는 의미.

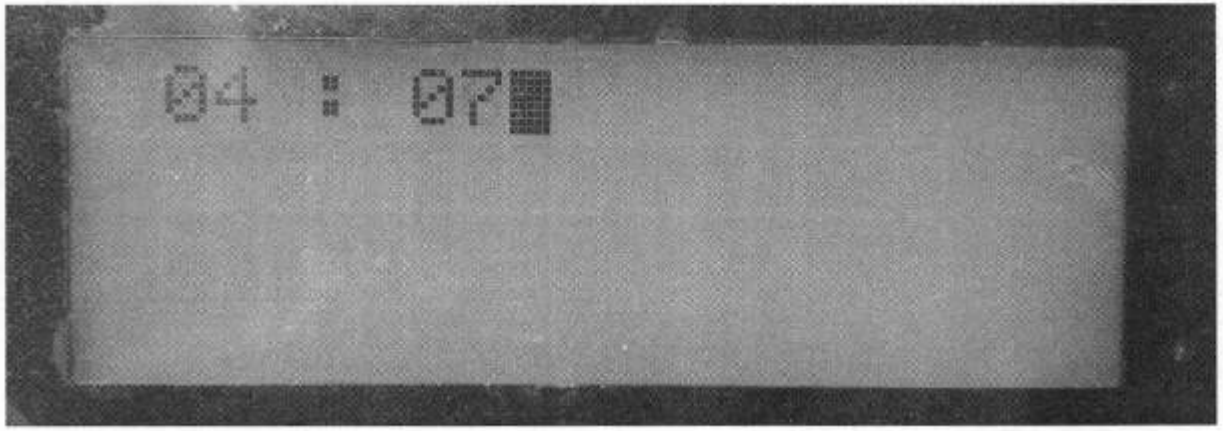
```
void main(void)
{
    ASSR=0<<AS0;
    TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) |
          (0<<WGM01) | (1<<CS02) | (0<<CS01) |
          (0<<CS00); /* 0x04 */
    TCNT0=0x06;
    OCR0=0x00;

    TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) |
          (0<<OCIE1A) | (0<<OCIE1B) | (0<<TOIE1) |
          (0<<OCIE0) | (1<<TOIE0); /* 0x01 */

    UCSR0A=(0<<RXC0) | (0<<TXC0) | (0<<UDRE0) |
           (0<<FE0) | (0<<DOR0) | (0<<UPE0) |
           (0<<U2X0) | (0<<MPCM0); /* 0x00 */
    UCSR0B=(0<<RXCIE0) | (0<<TXCIE0) | (0<<UDRIE0) |
           (0<<RXEN0) | (1<<TXEN0) | (0<<UCSZ02) |
           (0<<RXB80) | (0<<TXB80); /* 0x08 */
    UCSR0C=(0<<UMSEL0) | (0<<UPM01) | (0<<UPM00) |
           (0<<USBS0) | (1<<UCSZ01) | (1<<UCSZ00) |
           (0<<UCPOL0); /* 0x06 */
    UBRR0H=0x00;
    UBRR0L=0x67;

    #asm("sei")

    printf("$I\r");
    while (1)
    {
    // Place your code here
        if(one_sec_flag == 1)
        {
            one_sec_flag = 0;
            printf("$G, 1, 1\r");
            printf("$T, %02d : %02d\r",
                    one_min_cnt, one_sec_cnt);
        }
    }
}
```

KYUNG HEE UNIVERSITY

6.3 LCD 제어의 이론 및 실습

[과제6] LCD를 이용한 디지털 시계 제작하기

(3-1) LCD에 분:초 시계 나타내는 프로그램 결과

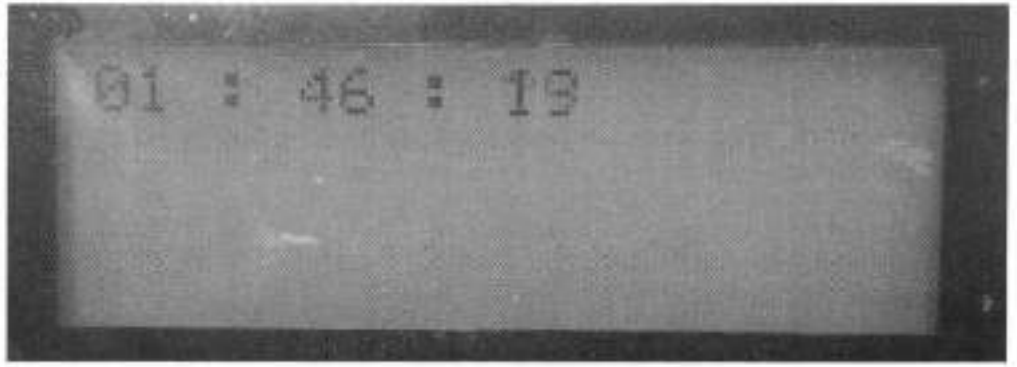


6.3 LCD 제어의 이론 및 실습

[과제6] LCD를 이용한 디지털 시계 제작하기

(4) LCD에 분:초:밀리초 시계 나타내는 프로그램 결과

```
28  int cnt, one_min_cnt = 0, one_sec_cnt = 0, one_msec_cnt = 0;
29
30  interrupt [TIM0_OVF] void timer0_ovf_isr(void)
31  {
32  // Reinitialize Timer 0 value
33  TCNT0=0x06;
34  // Place your code here
35      cnt++;
36      if(cnt >= 10)
37      {
38          cnt = 0;
39          one_sec_flag = 1;
40          one_msec_cnt++;
41
42          if(one_msec_cnt >= 100)
43          {
44              one_msec_cnt = 0;
45              one_sec_cnt++;
46              if(one_sec_cnt == 60)
47              {
48                  one_sec_cnt = 0;
49                  one_min_cnt++;
50                  if(one_min_cnt == 60)
51                  {
52                      one_min_cnt = 0;
53                  }
54              }
55          }
56      }
57  }
```



결과

```
88  while (1)
89  {
90  // Place your code here
91      if(one_sec_flag == 1)
92      {
93          one_sec_flag = 0;
94          printf("$G, 1, 1\r");
95          printf("$T,%02d : %02d : %02d\r",
96                  one_min_cnt, one_sec_cnt, one_msec_cnt);
97      }
98  }
```