AVR – UART 통신

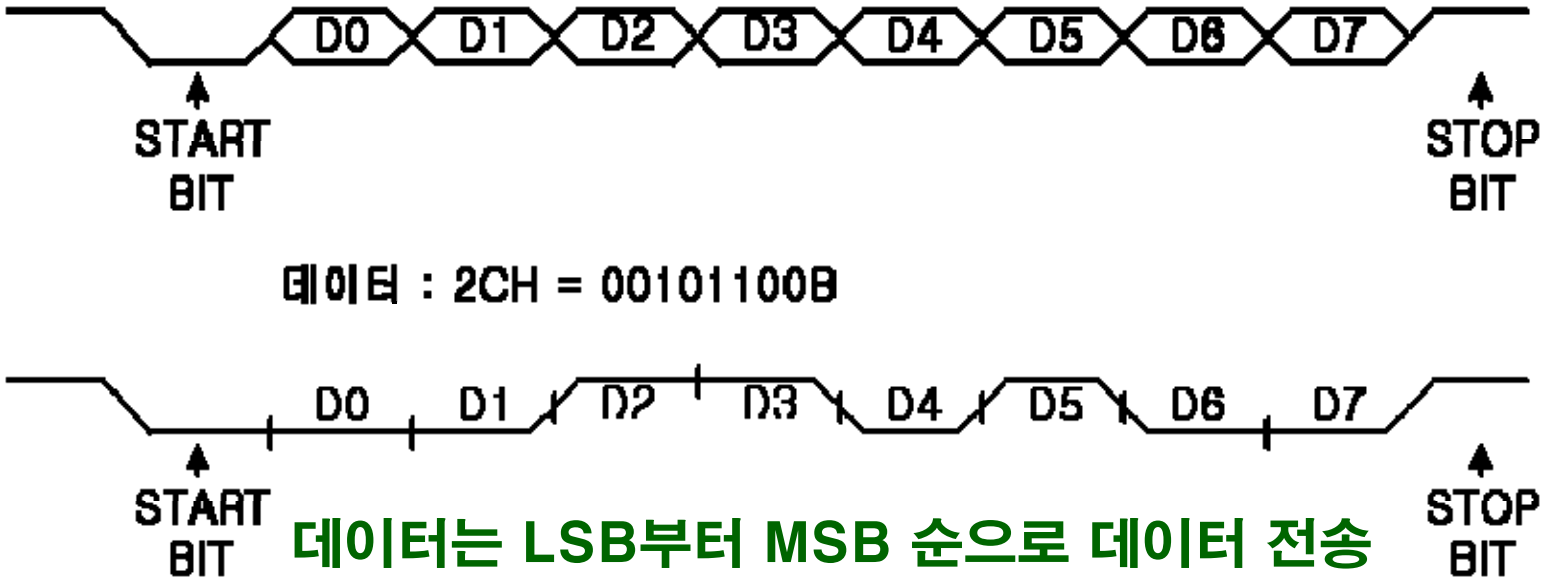
류 대 우

davidryu@newtc.co.kr

[시리얼(Serial) 통신이란?]

- Serial 통신은 하나의 신호 선을 이용해서 데이터를 비트 단위로 보내는 방식
  - 8비트 비 동기식 통신 콘트롤러

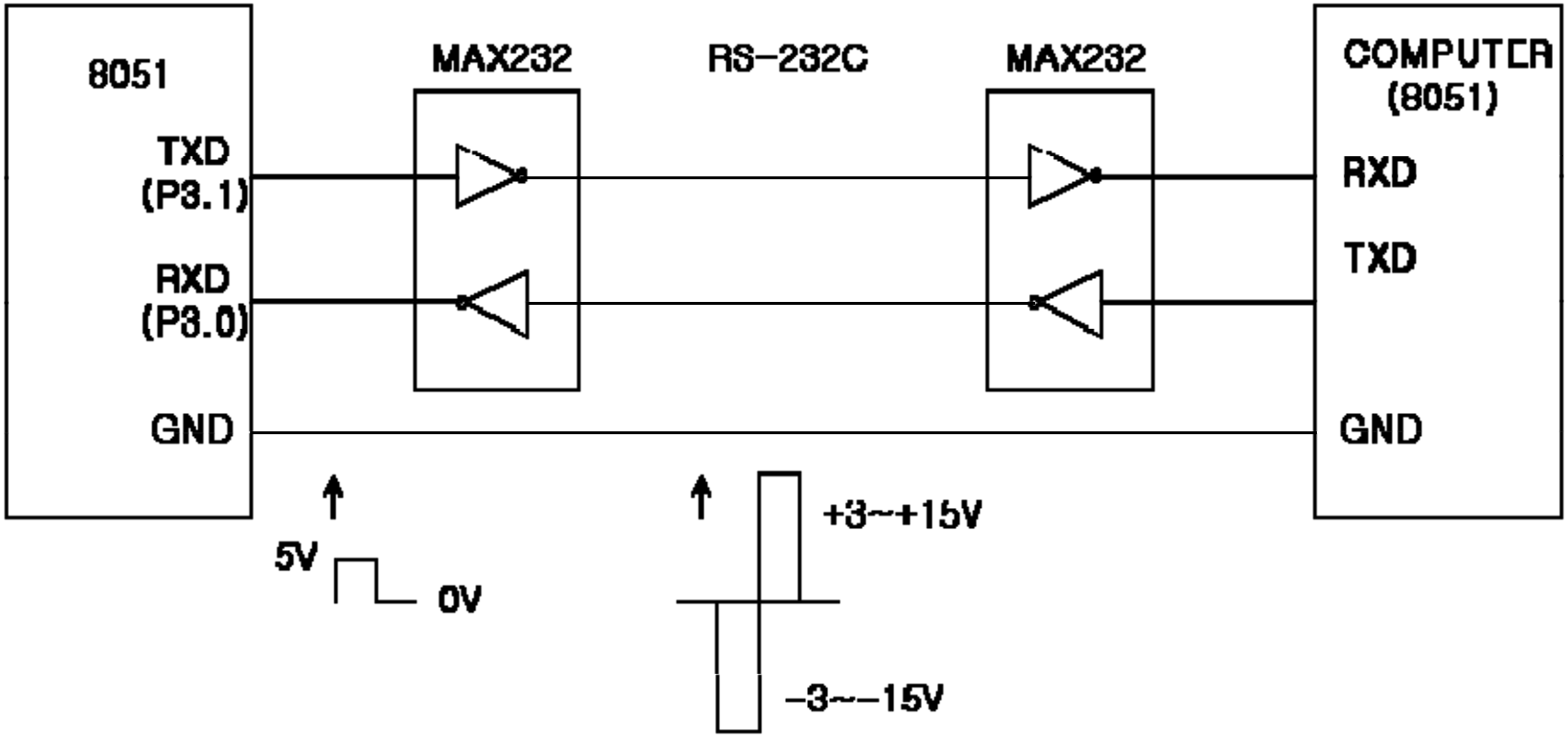
(UART : Universal Asynchronous Receiver Transmitter)



데이터는 LSB부터 MSB 순으로 데이터 전송

[Serial interface 통신 규격]

- **Serial interface 통신 규격**
  - **RS-232C, RS-422, RS-423, RS-485…**



< RS-232C의 예>

[시리얼(Serial) 통신]

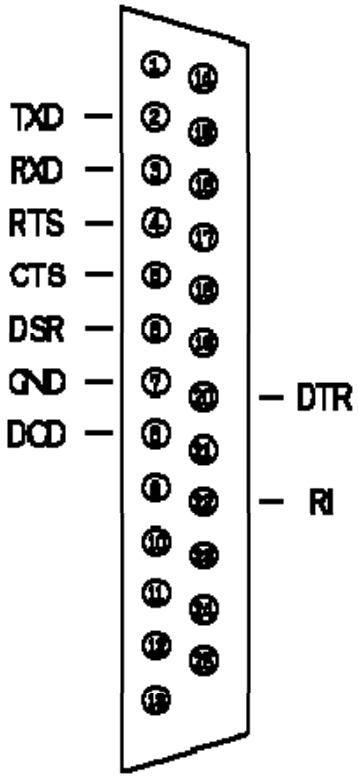
- Serial 통신 이점
  - 적은 수의 통신 라인 사용과 먼 거리 전송
- Serial 통신 전송 방식
  - Simplex
    - 단방향 전송 방식 - 라디오
  - half duplex
    - 반이중 전송 방식(서로 다른 시간 양방향 전송 방식) - 무전기
  - (full) duplex
    - 전이중 전송 방식(동시 양방향 전송 방식) - 8051
- 속도
  - bps(bits per second) : 1초당 전송되는 비트의 수
  - baud rate : 1초당 전송되는 변조된 신호의 수

  * 8051에서는 하나의 비트가 하나의 신호이므로 같은 의미

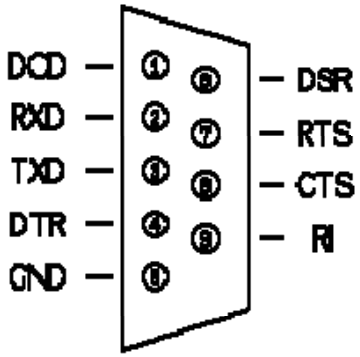
[시리얼(Serial) 종류]

Specification	RS232C	RS423	RS422	RS485
동작 모드	Single-Ended	Single-Ended	Differential	Differential
최대 Driver / Receiver 수	1 Driver 1 Receiver	1 Driver 10 Receivers	1 Driver 32 Receivers	32 Drivers 32 Receivers
최대 통달거리	약 15 m	약 1.2 km	약 1.2 km	약 1.2 km
최고 통신속도	20 Kb/s	100 Kb/s	10 Mb/s	10 Mb/s
지원 전송방식	Full Duplex	Full Duplex	Full Duplex	Half Duplex
최대 출력전압	±25V	±6V	-0.25V to +6V	-7V to +12V
최대 입력전압	±15V	±12V	-7V to +7V	-7V to +12V

[RS-232C]



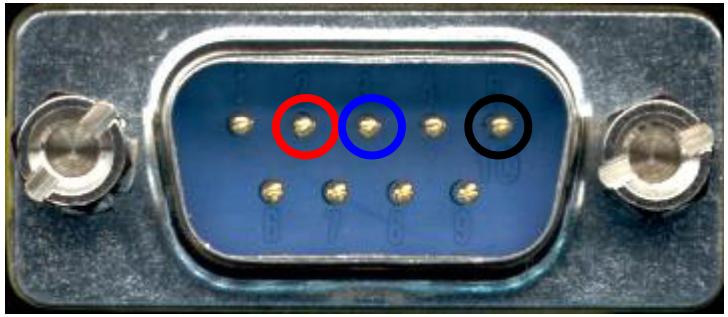
DB-25 RS232 코넥터

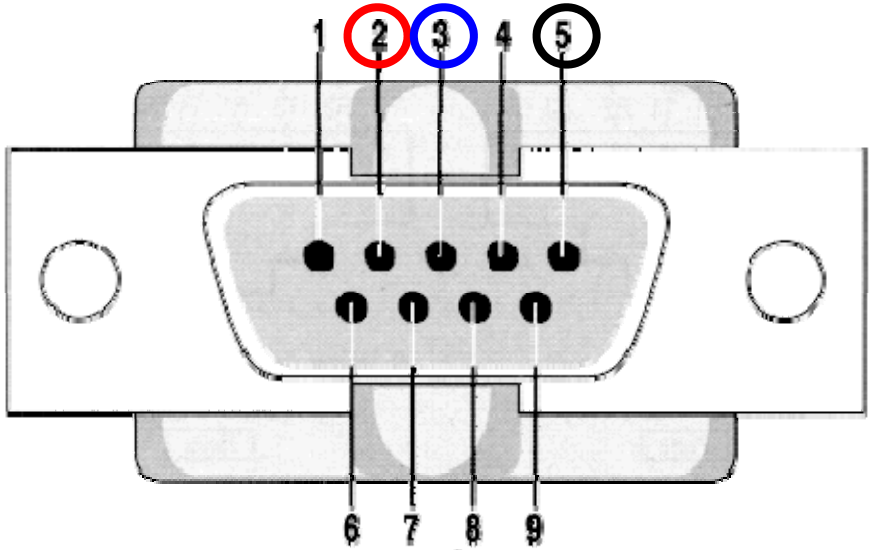


DB-9 RS232 코넥터

- **TXD - Transmit Data**
  - 직렬통신 데이터가 나오는 신호선
- **RXD - Receive Data**
  - 직렬통신 데이터를 입력받는 신호선
- **RTS - Ready To Send**
  - DTE장치가 DCE장치에게 준비가 됐음을 나타내는 신호선
- **CTS - Clear To Send**
  - DCE장치가 DTE장치에게 준비가 됐음을 나타내는 신호선
- **DTR - Data Terminal Ready**
  - 터미널이 모뎀에게 자신이 송수신 가능한 상태임을 알리는 신호선
- **DSR - Data Set Ready**
  - 모뎀이 터미널에게 자신이 송수신 가능한 상태임을 알려주는 신호선
- **DCD - Data Carrier Detect**
  - 모뎀이 상대편 모뎀과 전화선등을 통해서 접속이 완료되었을 때 상대편 모뎀이 캐리어신호를 보내오며 이신호를 검출하였음을 컴퓨터 또는 터미널에 알려주는 신호선
- **RI - Ring Indicator**
  - 상대편 모뎀이 통신을 하기위해서 먼저 전화를 걸어오면 전화 벨이 울리게 된다. 이때 이신호를 모뎀이 인식하여 컴퓨터 또는 터미널에 알려주는 신호선이다.

[RS-232C]





Pin	Signal	Pin	Signal
1	Data Carrier Detect	6	Data Set Ready
2	Received Data	7	Request to Send
3	Transmitted Data	8	Clear to Send
4	Data Terminal Ready	9	Ring Indicator
5	Signal Ground		

[시리얼 포트 보우 레이트]

- U2X = 0 으로 하였을 때의 보우 레이트 계산식, 아래와 같이 계산하면 UBRR 에 세팅 해야 하는 값이 나옴

$$\frac{CPU\text{클럭속도}}{UART-BAUD-RATE*16}-1$$

- 코딩은 아래와 같이 하면 편리하다
  - (UBRR 값이 255 보다 크게 나올경우는 주의)

$$UBRRL=\frac{F-CPU}{UART-BAUD-RATE*16L}-1$$

[시리얼 보우 레이트]

Baud Rate (bps)	f_{osc} = 1.0000 MHz				f_{osc} = 1.8432 MHz				f_{osc} = 2.0000 MHz			
	U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1	
	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error
2400	25	0.2%	51	0.2%	47	0.0%	95	0.0%	51	0.2%	103	0.2%
4800	12	0.2%	25	0.2%	23	0.0%	47	0.0%	25	0.2%	51	0.2%
9600	6	-7.0%	12	0.2%	11	0.0%	23	0.0%	12	0.2%	25	0.2%
14.4k	3	8.5%	8	-3.5%	7	0.0%	15	0.0%	8	-3.5%	16	2.1%
19.2k	2	8.5%	6	-7.0%	5	0.0%	11	0.0%	6	-7.0%	12	0.2%
28.8k	1	8.5%	3	8.5%	3	0.0%	7	0.0%	3	8.5%	8	-3.5%
38.4k	1	-18.6%	2	8.5%	2	0.0%	5	0.0%	2	8.5%	6	-7.0%
57.6k	0	8.5%	1	8.5%	1	0.0%	3	0.0%	1	8.5%	3	8.5%
76.8k	–	–	1	-18.6%	1	-25.0%	2	0.0%	1	-18.6%	2	8.5%
115.2k	–	–	0	8.5%	0	0.0%	1	0.0%	0	8.5%	1	8.5%
230.4k	–	–	–	–	–	–	0	0.0%	–	–	–	–
250k	–	–	–	–	–	–	–	–	–	–	0	0.0%
Max.(1)	62.5 kbps		125 kbps		115.2 kbps		230.4 kbps		125 kbps		250 kbps	

[시리얼 보우 레이트]

Baud Rate (bps)	f_{osc} = 3.6864 MHz				f_{osc} = 4.0000 MHz				f_{osc} = 7.3728 MHz			
	U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1	
	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error
2400	95	0.0%	191	0.0%	103	0.2%	207	0.2%	191	0.0%	383	0.0%
4800	47	0.0%	95	0.0%	51	0.2%	103	0.2%	95	0.0%	191	0.0%
9600	23	0.0%	47	0.0%	25	0.2%	51	0.2%	47	0.0%	95	0.0%
14.4k	15	0.0%	31	0.0%	16	2.1%	34	-0.8%	31	0.0%	63	0.0%
19.2k	11	0.0%	23	0.0%	12	0.2%	25	0.2%	23	0.0%	47	0.0%
28.8k	7	0.0%	15	0.0%	8	-3.5%	16	2.1%	15	0.0%	31	0.0%
38.4k	5	0.0%	11	0.0%	6	-7.0%	12	0.2%	11	0.0%	23	0.0%
57.6k	3	0.0%	7	0.0%	3	8.5%	8	-3.5%	7	0.0%	15	0.0%
76.8k	2	0.0%	5	0.0%	2	8.5%	6	-7.0%	5	0.0%	11	0.0%
115.2k	1	0.0%	3	0.0%	1	8.5%	3	8.5%	3	0.0%	7	0.0%
230.4k	0	0.0%	1	0.0%	0	8.5%	1	8.5%	1	0.0%	3	0.0%
250k	0	-7.8%	1	-7.8%	0	0.0%	1	0.0%	1	-7.8%	3	-7.8%
0.5M	–	–	0	-7.8%	–	–	0	0.0%	0	-7.8%	1	-7.8%
1M	–	–	–	–	–	–	–	–	–	–	0	-7.8%
Max. (1)	230.4 kbps		460.8 kbps		250 kbps		0.5 Mbps		460.8 kbps		921.6 kbps	

[시리얼 보우 레이트]

Baud Rate (bps)	f_{osc} = 8.0000 MHz				f_{osc} = 11.0592 MHz				f_{osc} = 14.7456 MHz			
	U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1	
	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error
2400	207	0.2%	416	-0.1%	287	0.0%	575	0.0%	383	0.0%	767	0.0%
4800	103	0.2%	207	0.2%	143	0.0%	287	0.0%	191	0.0%	383	0.0%
9600	51	0.2%	103	0.2%	71	0.0%	143	0.0%	95	0.0%	191	0.0%
14.4k	34	-0.8%	68	0.6%	47	0.0%	95	0.0%	63	0.0%	127	0.0%
19.2k	25	0.2%	51	0.2%	35	0.0%	71	0.0%	47	0.0%	95	0.0%
28.8k	16	2.1%	34	-0.8%	23	0.0%	47	0.0%	31	0.0%	63	0.0%
38.4k	12	0.2%	25	0.2%	17	0.0%	35	0.0%	23	0.0%	47	0.0%
57.6k	8	-3.5%	16	2.1%	11	0.0%	23	0.0%	15	0.0%	31	0.0%
76.8k	6	-7.0%	12	0.2%	8	0.0%	17	0.0%	11	0.0%	23	0.0%
115.2k	3	8.5%	8	-3.5%	5	0.0%	11	0.0%	7	0.0%	15	0.0%
230.4k	1	8.5%	3	8.5%	2	0.0%	5	0.0%	3	0.0%	7	0.0%
250k	1	0.0%	3	0.0%	2	-7.8%	5	-7.8%	3	-7.8%	6	5.3%
0.5M	0	0.0%	1	0.0%	–	–	2	-7.8%	1	-7.8%	3	-7.8%
1M	–	–	0	0.0%	–	–	–	–	0	-7.8%	1	-7.8%
Max. (1)	0.5 Mbps		1 Mbps		691.2 kbps		1.3824 Mbps		921.6 kbps		1.8432 Mbps	

[시리얼 보우 레이트]

Baud Rate (bps)	f_{osc} = 16.0000 MHz				f_{osc} = 18.4320 MHz				f_{osc} = 20.0000 MHz			
	U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1	
	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error
2400	416	-0.1%	832	0.0%	479	0.0%	959	0.0%	520	0.0%	1041	0.0%
4800	207	0.2%	416	-0.1%	239	0.0%	479	0.0%	259	0.2%	520	0.0%
9600	103	0.2%	207	0.2%	119	0.0%	239	0.0%	129	0.2%	259	0.2%
14.4k	68	0.6%	138	-0.1%	79	0.0%	159	0.0%	86	-0.2%	173	-0.2%
19.2k	51	0.2%	103	0.2%	59	0.0%	119	0.0%	64	0.2%	129	0.2%
28.8k	34	-0.8%	68	0.6%	39	0.0%	79	0.0%	42	0.9%	86	-0.2%
38.4k	25	0.2%	51	0.2%	29	0.0%	59	0.0%	32	-1.4%	64	0.2%
57.6k	16	2.1%	34	-0.8%	19	0.0%	39	0.0%	21	-1.4%	42	0.9%
76.8k	12	0.2%	25	0.2%	14	0.0%	29	0.0%	15	1.7%	32	-1.4%
115.2k	8	-3.5%	16	2.1%	9	0.0%	19	0.0%	10	-1.4%	21	-1.4%
230.4k	3	8.5%	8	-3.5%	4	0.0%	9	0.0%	4	8.5%	10	-1.4%
250k	3	0.0%	7	0.0%	4	-7.8%	8	2.4%	4	0.0%	9	0.0%
0.5M	1	0.0%	3	0.0%	–	–	4	-7.8%	–	–	4	0.0%
1M	0	0.0%	1	0.0%	–	–	–	–	–	–	–	–
Max. (1)	1 Mbps		2 Mbps		1.152 Mbps		2.304 Mbps		1.25 Mbps		2.5 Mbps	

[ATMEGA128_관련레지스터]

- **USRnA(USARTn Control & Status Register A)**
  - **BIT 7 : RXCn(USART Receive Complete)**
    - RXCn(USARTn Receive Complete) 비트는 수신버퍼에 읽혀지지 않은 수신 문자가 들어있으면 "1"로 셋 되고 CPU가 이를 읽어서 수신 버퍼가 비어있는 상태라면 "0" 으로 클리어 되었음을 나타내는 상태 플래그이다. RXCn 비트는 수신 완료 인터럽트를 발생시킬 때 사용한다.
  - **BIT 6 : RXCn(USARTn Transmit Complete)**
    - TXCn(USARTn Transmit Complete) 비트는 송신 시 포트 레지스터에 있는 송신 데이터가 모두 송신되고 UDRn의 송신 버퍼에 아직 새로운 송신 데이터가 라이트 되지 않은 상태 이면"1"로 셋 되는 상태를 지시 하는 플래그이다. TXCn 비트는 송신 완료 인터럽트를 발생 시킬 때 사용된다

[ATMEGA128_관련레지스터]

- **BIT 5 : UDREn(USARTn Data Register Empty)**
  - UDREn(USARTn Data Register Empty)비트는 UDRn의 송신 버퍼에 새로운 송신 데이터를 받을 준비가 되어 있으면 "1"로 셋 되는 상태 플래그이다 UDREn 비트는 UDARTn Data Register Empty 인터럽트를 발생할 때 사용한다.

- **BIT 4 : FEn(Frame Error)**
  - FEn(USARTn Frame Error)비트는 UDRn의 수신 버퍼에 현재 저장 되어있는 데이터를 수신 하는 동안 프레임 에러가 발생 하였음을 나타내는 상태 플래그로, 프레임에러는 수신 문자의 첫 번째 스톱비트가"0"으로 검출 되면 발생(1로 셋 되고) 하고 UCSRAn 레지스터를 라이트하면 이 비트는 "0"으로 클리어된다

- **BIT 3 : DORn*Data OverRun**
  - DORn(USARTn Data Overrun Error) 비트는 수신 동작에서 오버런 에러가 발생하였음을 나타내는 상태 플래그로 OverRun Error는 UDRn의 수신 버퍼에 현재 읽지 않은 수신 문자가 들어있는 상태에서, 수신 시 포트레지스터에 새로운 문자가 수신 완료되면, 다시 그 다음 수신 데이터인 3번째 문자의 스타트비트가 검출되면 발생한다. UCSRnA 레지스터를 라이트 하면 이 비트는 "0"으로 클리어된다.

[ATMEGA128_관련레지스터]

- **BIT 2 : PEn(Parity Error)**
  - PEn(Parity Error)비트는 UDR의 수신 버퍼에 현재 저장 되어 있는 데이터를 수신 하는 동안 패리티에러가 발생 하였음을 나타내는 상태플래그로 패리티에러는 UCSRnC 레지스터의UPMn1 비트를"1" 설정하여 패리티 비트를 사용하도록 설정 한 경우에만 발생할 수 있다. UCSRnA 레지스터를 라이트 하면 이 비트는"0"으로 클리어된다

- **BIT 1 : UCXn(Double the USARTn Transmission Speed)**
  - U2Xn(Double the USART Transmisson Speed) 비트는 비동기모드에서만 사용가능 한 것으로서 클록의 n 분주비를 16에서 8로 1/2만큼 낮추어 전송 속도를 2배 높이는 기능을 한다

- **BIT 0 : MPCMn(USART Multi-Processor Communication Mode)**
  - 비트는 USARTn을 멀티프로세서 통신 모드로 설정하고 멀티프로세서통신 어드레스정보를 포함 하지 않는 모든 수신 데이터는 수신부에 의하여 무시 된다.

[ATMEGA128_관련레지스터]

- **USRnC(USARTTn Control & Status Register C)**
  - **BIT 7**
    - 예약
  - **BIT 6**
    - UMSELn(USARTn Mode Select) UMSELn(USARTn Mode Select) 비트는 "1"이면 USARTn 모듈을 동기 전송 모드로 설정 하고 "0"이면 비동기 전송 모드로 설정한다
  - **BIT5,4**
    - UPMn1, 0(Parity Mode)
    - 이 비트를 "1"로 설정 하면 패리티를 발생 시키고 검사를 할 수 있고 송신기는 자동적으로 각 프레임의 송신 데이터에 페리티 비트를 더하여 송신한다. 수신기는 UPM0 비트와 수신된 데이터를 비교한다. 만약에 오류가 발생하면UCSRnA 레지스터의 PE 플래그가"1"로셋된다
    - UPMn 비트설정표
    - ---
    - UPMn1 UPMn0 Parity모드
    - ---
    - 0 0 Disable
    - 0 1 예약
    - 1 0 Enabled Even Panrity
    - 1 1 Enabled Odd Parity

[ATMEGA128_관련레지스터]

- **BIT 3 : USBSn(Stop Bit Select)**
  - USBSn(Stop Bit Select) 비트가 "0“ 이면 USARTn 모듈에서 데이터 포맷을 구성하는 스톱 비트를 1개로 설정하고 "1"이면 스톱 비트를 2개로 설정한다.

USBSn	StopBIt
0	1-bit
1	2-bit

- **BIT 2,1 : UCSZn1,0(Character Size) - 데이터비트수를설정**

UPMn1	UPMn0	UCSZn0	Parity모드
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	에약
1	0	1	에약
1	1	0	에약
1	1	1	9-bit

[ATMEGA128_관련레지스터]

- **BIT 0 : UCPOLn**
  - ---
  - USBSn TxDn의출력StopBIt
  - ---
  - 0 XCKn 상승 XCKn 하강
  - 1 XCKn 하강 XCKn 상승

[시리얼 프로그래밍]

- **UBRR0H = 0x00;**
  - UBRnH/L(USARTTn Baud Rate Register) 레지스터는 16비트 중에서 12비트만 사용하여 USARTTn 모듈의 송·수신 속도를 설정하는 기능을 한다.
  - BIT 15~23 : 에약비트
  - BIT 11~0 : UBRRn11~0(USARTn Baud Rate Register)
  - 2비트를 이용하여 USARTn의 보율(baud rate)을 결정 하는데 UBRRnH의 4비트와 UBRRnL의8비트가조합을이루고있다

[시리얼 프로그래밍]

- **USRnB(USARTTn Control & Status Register B)**
  - **BIT 7 : RXCIEn(RX Complete Interrupt enable)**
    - EXCIEn(USARTn RX Complete Interrupt Enable) 비트는 수신 완료 인터럽트를 개별적으로 enable하고 "1"로 설정하고 SREG 레지스터의 비트가"1"0이고 UCSRnA 레지스터의 RXCn 비트가"1"로 설정되어 있으면 수신 완료 인터럽트가 발생한다
  - **BIT 6 : TXCIEn(TX Complete Interrupt Enable)**
    - TXCIEn(USARTn TX Complete Interrupt Enable)비트는 수신완료 인터럽트를 개별적으로 enable(1)로 설정하고 SREG 레지스터의 비트가 "1"0이고 UCSRnA와 UCSRnA레지트터의 TXCn 비트가"1"로 설정 되어 있으면 송신 완료 인터럽트가 발생한다

[시리얼 프로그래밍]

- **BIT 5 : UDRIEn(USARTn Data Register Empty Interrupt Enable) 비트**
  - USARTn Data Register Empty 인터럽트를 개별적으로 enable하는 비트로 "1"로 설정하고 SREG 레지스터의 비트가 "1"이고 UCSRnA 레지스터의 UDREn 비트가 "1"로 되면 USARTn Data Register Empty 인터럽트가발생된다
- **BIT 4 : RXENn(Receiver Enable)**
  - RXEN(Receiver Enable) 비트는USARTn 모듈의 수신부가 동작하도록 enable하는 것 으로RXDn 핀이 병렬 I/O 포트가 아니라 직렬 데이터 수신단자로 동작 하도록 설정한다.
- **BIT 3 : TXENn(Transmitter Enable)**
  - TXENn(Transmitter Enable) 비트는 USARTn 모듈의 송신부가 동작하도록 enable하는 것으로 TXDn핀이 병렬 I/O 포트가 아니라 직렬 데이터 송신단자로 동작하도록 설정한다.
- **BIT 2 : UCSZn2(Character Size)**
  - UCSZn2(USARTn Character Size)비트는 UCSRnC레지스터의 UCSZn1~0비트와 함께 전송 문자의 데이터비트수를설정하는데사용된다

[시리얼 프로그래밍]

- **BIT 1 : RXB8n(Transmit data Bit 8)**
  - **RXB8n(Receive Data Bit 8)은 수신 문자가 9비트로 설정 된 경우에 수신된 문자의 9번째 비트(MSB)를 저장한다**
- **BIT 0 : TXB8n(Transmit data BIt 8)**
  - **송신문자가 8비트로 설정 된 경우에 송신된 문자의9번째 비트 (MSB)를 저장한다**

[시리얼 프로그래밍]

- #pragma interrupt_handler uart0_rx_isr:19
- void uart0_rx_isr(void)
- {
- //uart has received a character in UDR
- cmd=UDR0;
  - //시리얼 통신에서 받은 데이터(UART Data Register)를 cmd변수에 저장
- }

[volatile]

- volatile은 휘발성 메모리라는 뜻
- 컴파일러가 최적화(Optimize)하더라도 프로그래머의 의도대로 되도록 이부분을 최적화(Optimize)하지 말라는 뜻
- 변수값을 메모리에서 읽어 들이도록 함.

[시리얼 통신 문자열 처리]

- 시리얼은 한 비트씩 한 바이트가 한 패킷이 되어 들어오는 통신

- 문제:
  - 시리얼 통신에서 사용자가 터미널에 어떠한 문자를 입력하고, 이 문자를 다 친 후에 1을 입력하면 이를 출력하게 하여라
    - 치는 동안에는 ATmega128에서 기억하고 있게 한다.
  - 예 : asdlfjks1sdljflasdjk1

[시리얼 통신 문자열 처리]

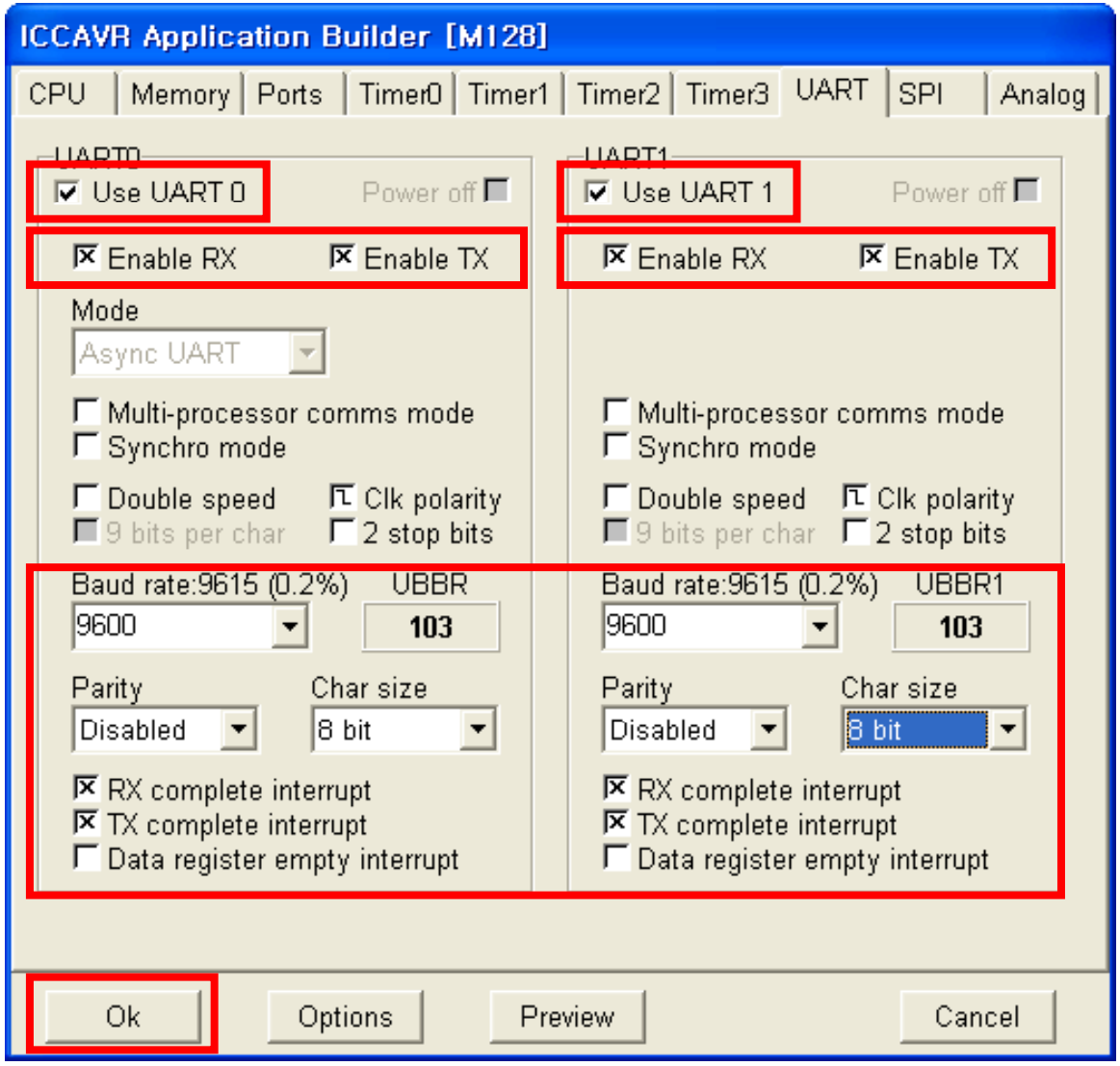
```
char data[100] = {0};
volatile int cnt = 0;

#pragma interrupt_handler uart0_rx_isr:19
void uart0_rx_isr(void)
{
      data[cnt] = UDR0;

      if(data[cnt] == '1'){
                  cnt = -1;
                  data[cnt+1] = 0x00;
                  printf("%s", data);

      }
      cnt++;
 //uart has received a character in UDR
}
```

[시리얼 포워더 만들기]

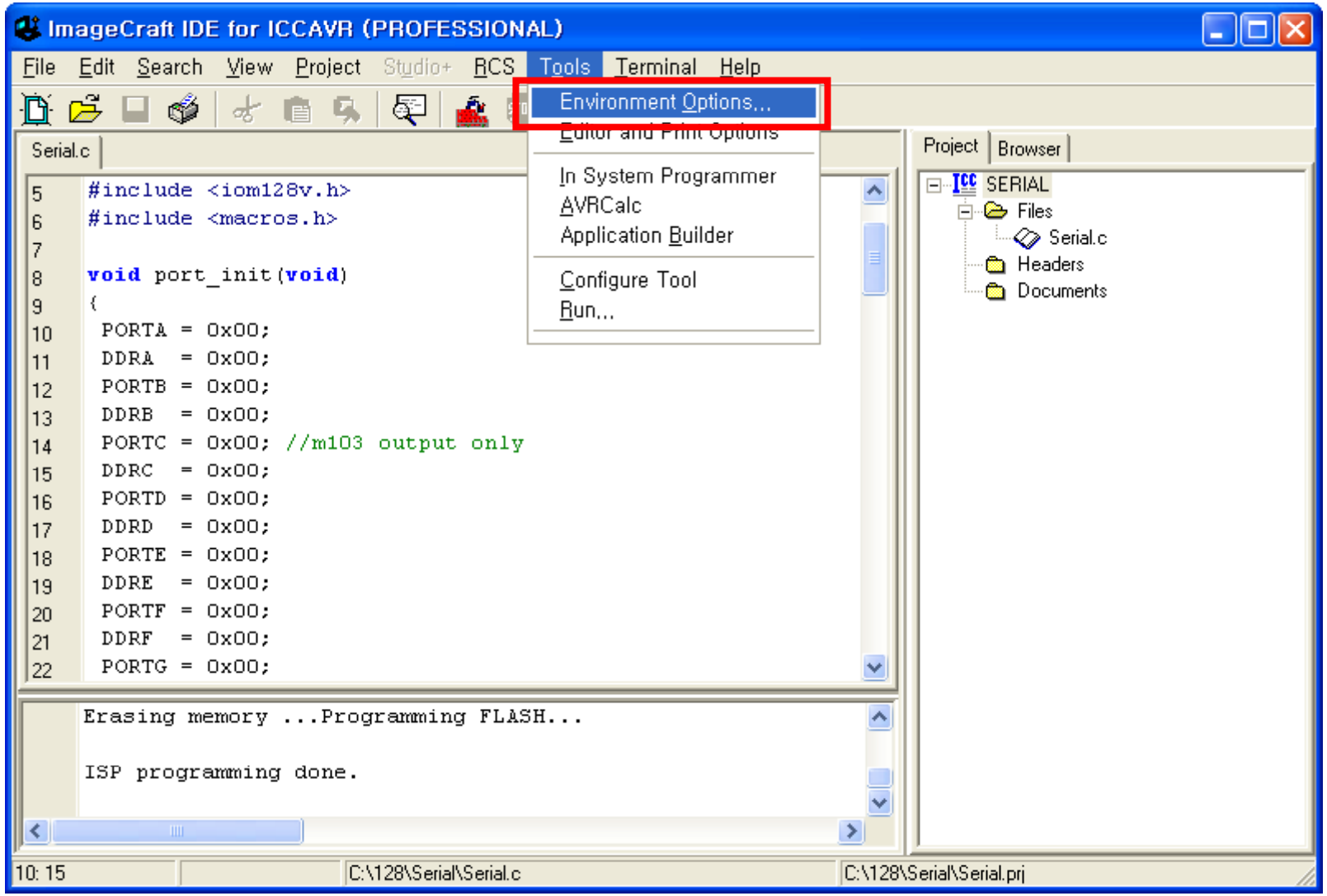

ICCAVR Application Builder [M128]
CPU
Memory
Ports
Timer0
Timer1
Timer2
Timer3
UART
SPI
Analog
UART0
Use UART 0
Power off
Enable RX
Enable TX
Mode
Async UART
Multi-processor comms mode
Synchro mode
Double speed
Clk polarity
9 bits per char
2 stop bits
Baud rate:9615 (0.2%)
UBBR
9600
103
Parity
Disabled
Char size
8 bit
RX complete interrupt
TX complete interrupt
Data register empty interrupt
UART1
Use UART 1
Power off
Enable RX
Enable TX
Multi-processor comms mode
Synchro mode
Double speed
Clk polarity
9 bits per char
2 stop bits
Baud rate:9615 (0.2%)
UBBR1
9600
103
Parity
Disabled
Char size
8 bit
RX complete interrupt
TX complete interrupt
Data register empty interrupt
Ok
Options
Preview
Cancel

[시리얼 포워더 만들기]

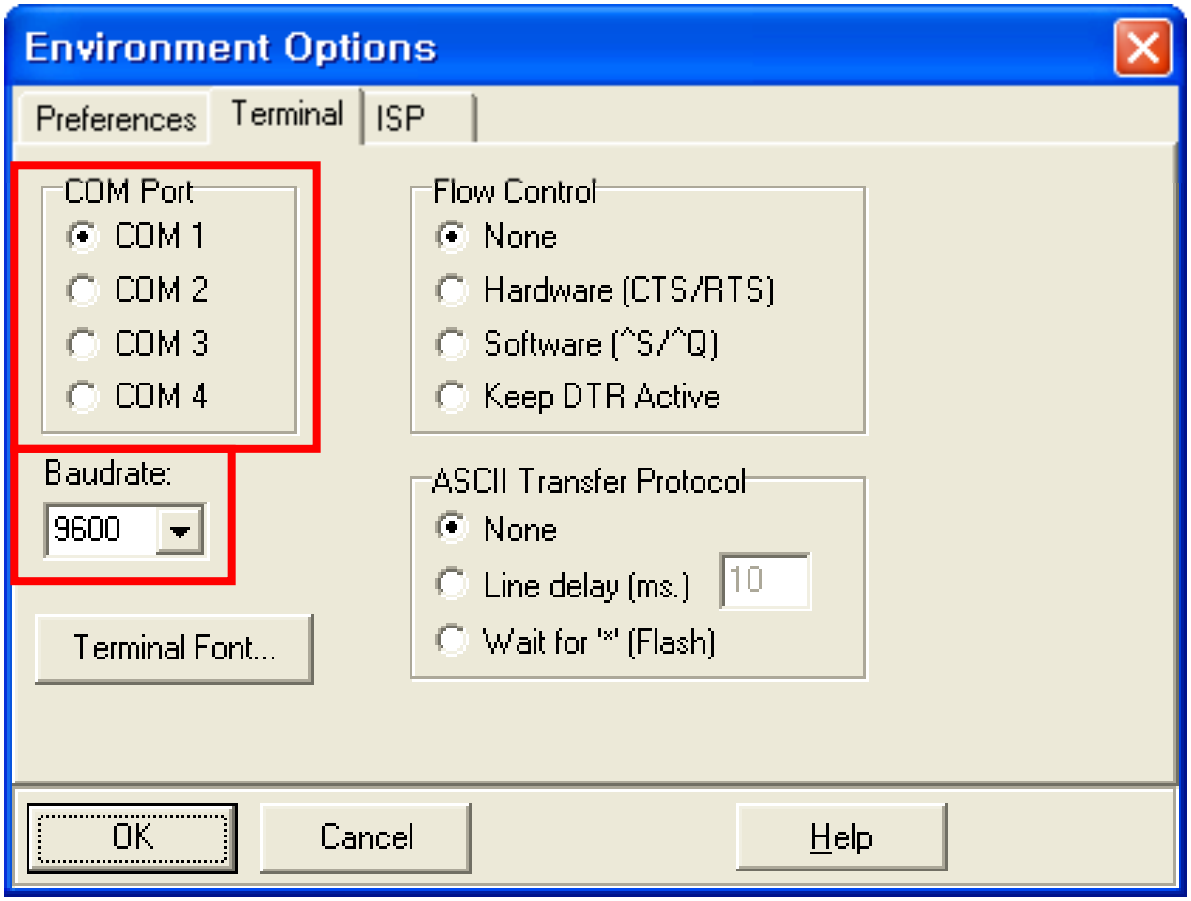
- ```
  #pragma interrupt_handler uart0_rx_isr:19
  void uart0_rx_isr(void)
  {
      //uart has received a character in UDR
     UDR0 = UDR0;
  }
  ```

//컴파일 후 ATmega128에 포팅한다.
  ```

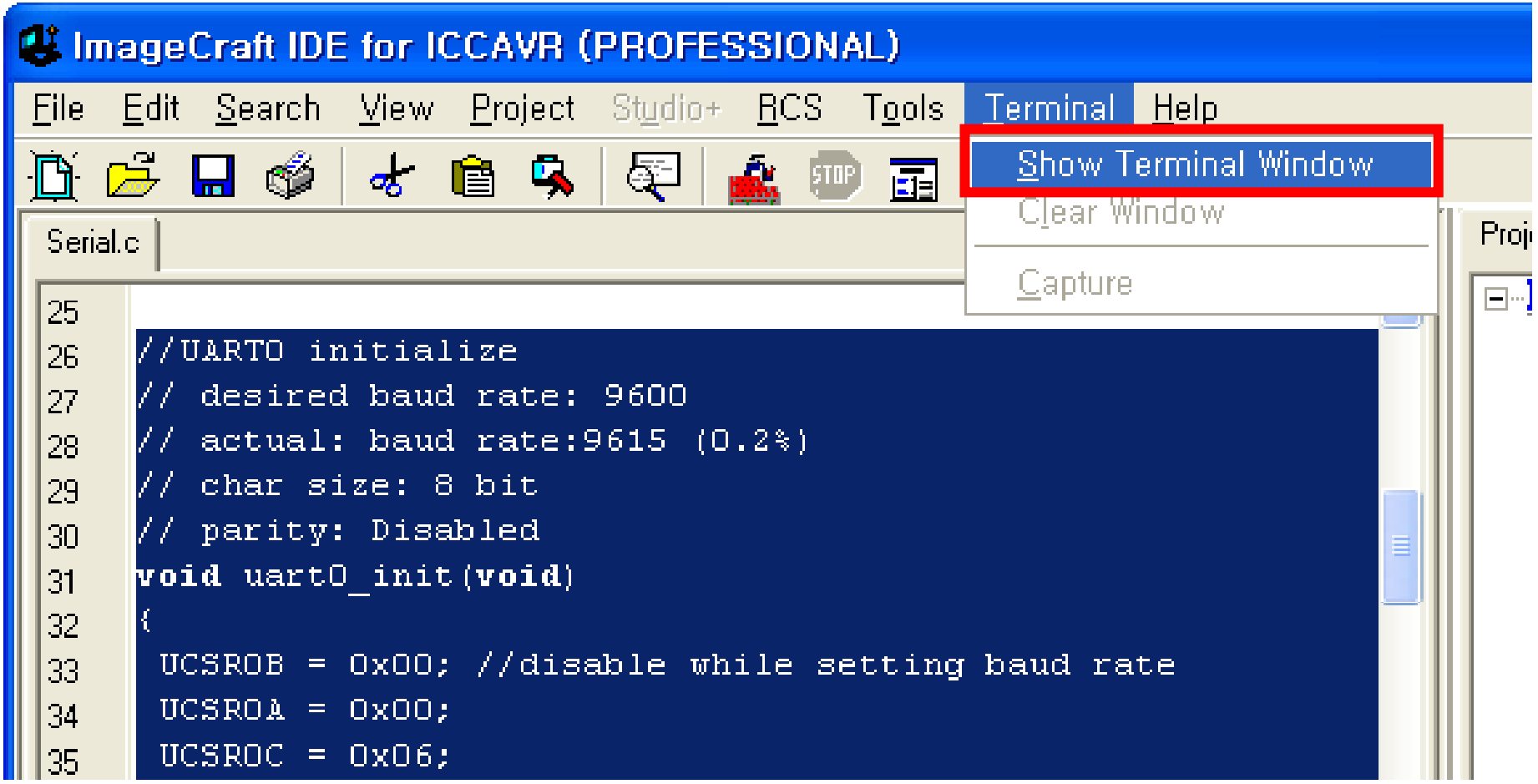
[시리얼 포워더 만들기]


ImageCraft IDE for ICCAVR (PROFESSIONAL)
File Edit Search View Project Studio+ RCS Tools Terminal Help
Environment Options...
Editor and Print Options
In System Programmer
AVRCalc
Application Builder
Configure Tool
Run...
Serial.c
#include <iom128v.h>
#include <macros.h>
void port_init(void)
{
PORTA = 0x00;
DDRA = 0x00;
PORTB = 0x00;
DDRB = 0x00;
PORTC = 0x00; //m103 output only
DDRC = 0x00;
PORTD = 0x00;
DDRD = 0x00;
PORTE = 0x00;
DDRE = 0x00;
PORTF = 0x00;
DDRF = 0x00;
PORTG = 0x00;
Erasing memory ...Programming FLASH...
ISP programming done.
Project Browser
SERIAL
Files
Serial.c
Headers
Documents
10: 15
C:\128\Serial\Serial.c
C:\128\Serial\Serial.prj

[시리얼 포워더 만들기]

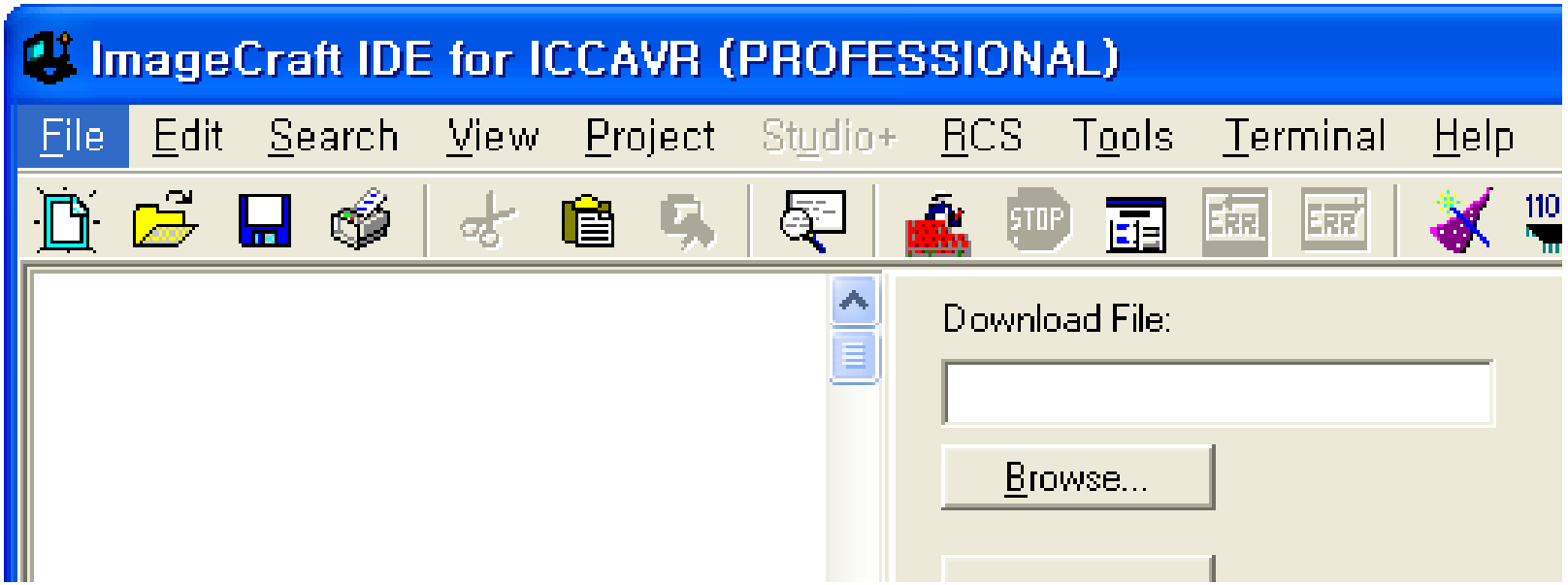

Environment Options
Preferences
Terminal
ISP
COM Port
COM 1
COM 2
COM 3
COM 4
Baudrate:
9600
Terminal Font...
Flow Control
None
Hardware (CTS/RTS)
Software (^S/^Q)
Keep DTR Active
ASCII Transfer Protocol
None
Line delay (ms.)
10
Wait for '*' (Flash)
OK
Cancel
Help

[시리얼 포워더 만들기]


ImageCraft IDE for ICCAVR (PROFESSIONAL)
File Edit Search View Project Studio+ RCS Tools Terminal Help
Show Terminal Window
Clear Window
Capture
Serial.c
25
26 //UART0 initialize
27 // desired baud rate: 9600
28 // actual: baud rate:9615 (0.2%)
29 // char size: 8 bit
30 // parity: Disabled
31 void uart0_init(void)
32 {
33 UCSR0B = 0x00; //disable while setting baud rate
34 UCSR0A = 0x00;
35 UCSR0C = 0x06;

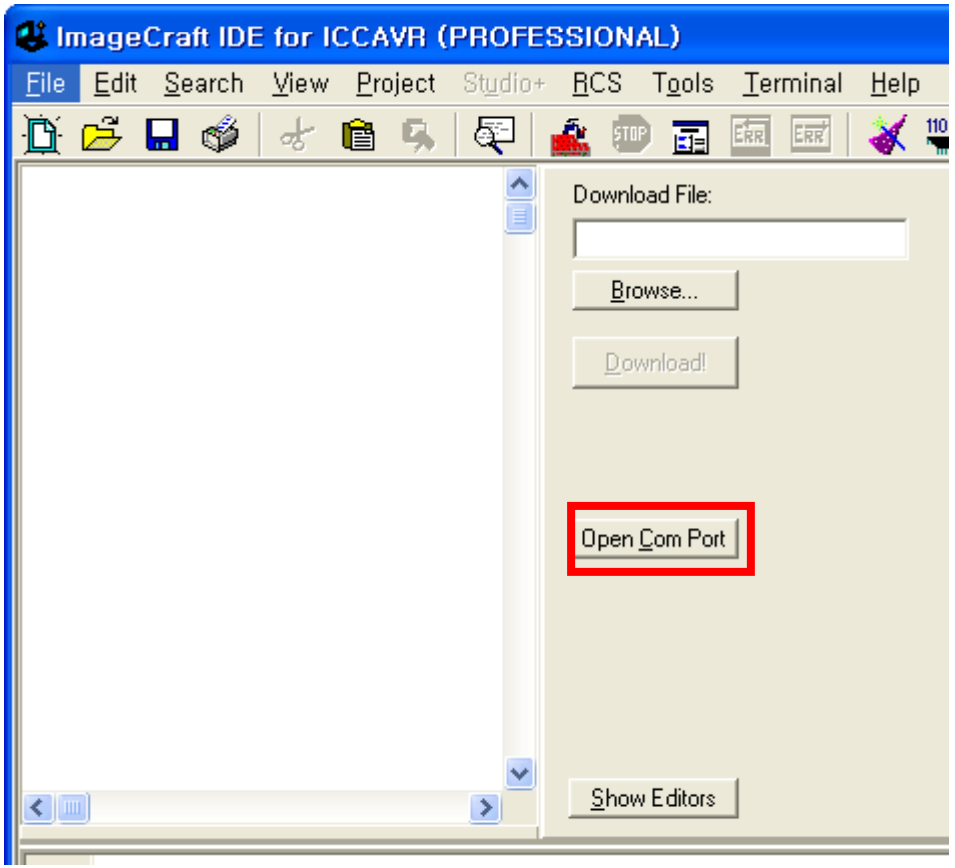
[시리얼 포워더 만들기]

- 키보드를 눌러도 아무런 반응이 없다.

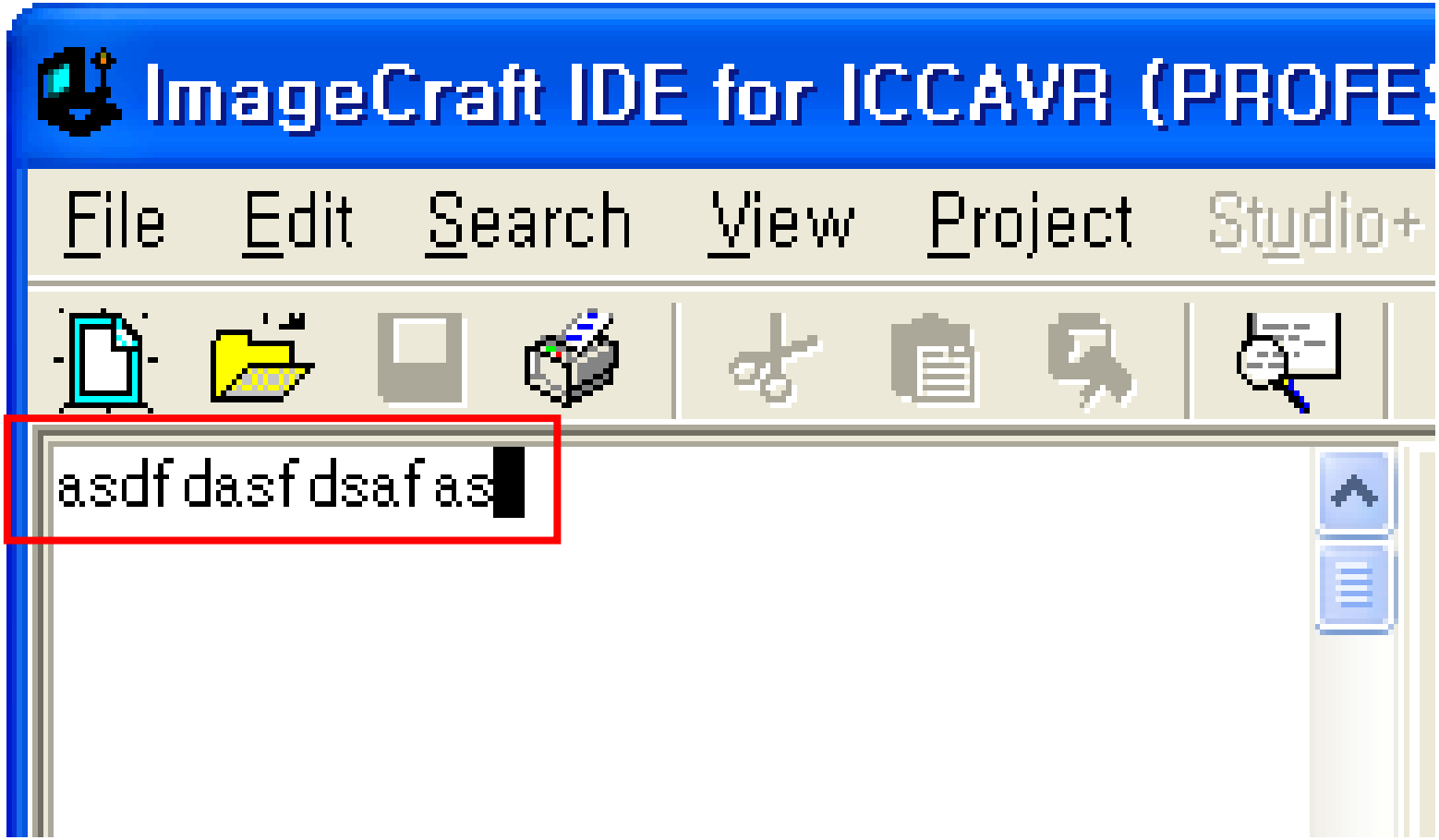


[시리얼 포워더 만들기]

- **Open Com Port를 눌른다.**



[시리얼 포워더 만들기]


ImageCraft IDE for ICCAVR (PROFE
File Edit Search View Project Studio+
asdf dasf dsaf as

[Printf 사용하기]

```
void main(){
    init_devices();

    printf("Serial Forwarder v0.1|r|n");
    while(1){
            ;
    }
}
```

[컴파일러 별 차이점]

- **AVR Studio + WINAVR**
  - printf를 사용하기 위하여
  - 초기화 루틴에 fdevopen(Putchar,0); 를 적어준다.
  - #include <avr/io.h>
  - #include <avr/interrupt.h>
  - #include "c:/WinAVR/avr/include/avr/iom128.h"
  - #include <stdio.h>
  - static int Putchar(char message, FILE *stream)
  - { while (((UCSR0A>>UDRE0)&0x01) == 0) ;
  - // UDRE, data register empty
  - UDR0 = message; }

[컴파일러 별 차이점]

- **Codevision**
  - #include <mega128.h>
  - #include <delay.h>
  - #include <string.h>
  - #include <stdio.h>

  - 그냥 printf를 사용하면 된다.
  - 내장 함수를 별도로 써주지 않아도 됨.

[컴파일러 별 차이점]

- **IAR EwAVR**
  - #include <iom128.h>
  - #include <stdio.h>

  - int getchar(void){
  - int in;
  - while ((UCSR0A & 0x80) == 0);
  - in = UDR0; return in; }
  - int putchar(int c){
  - while (((UCSR0A>>5)&0x01) == 0) ;
  - // UDRE, data register empty
  - UDR0 = c; return c; }

[컴파일러 별 차이점]

- **ICCAVR**
  - #include <iom128v.h>
  - #include <macros.h>
  - #include <stdio.h>
  - int putchar(char c) // printf 함수 사용시 추가할 것.
  - { while (((UCSR0A>>UDRE0)&0x01) == 0) ;
  - UDR0 = c;
  - return c; }
  - int getchar(void){ // scanf 함수 사용시 추가할 것.
  - while ((UCSR0A & 0x80) == 0);
  - return UDR0; }