실험7. LCD에 글자쓰기(직렬통신 이해하기)

1. LCD란

일반적인 응용에서는 많은 숫자를 표시할 필요가 있습니다. 그러나 이를 직접 프로세서가 관리하면서 다른 일을 하기에는 어려움이 많이 있지요. 그래서 숫자나 글자를 전담해서 나타내 주는 단위 기능을 갖는 장치를 많이 사용합니다. 대표적인 것이 LCD display module입니다. 즉 액정표시기를 사용하여 많은 양의 숫자나 글자를 표시해주는 장치이지요. 이것도 기본적으로는 우리가 실험해 보았던 scanning 방법을 이용하여 많은 글자를 표시해 주는 것입니다. 따라서 한 개 이상의 프로세서가 사용되고 있습니다. 이번 실험은 LCD module에 글자를 써 보도록 하는 것입니다. 상용으로 판매되고 있는 기본 장치는 병렬 데이터 전송 방법으로 표시하고자 하는 글자를 전송합니다. 왜냐하면 데이터 전송속도가 빠르기 때문이지요. 그러나 외부와의 연결 pin이 많아져 하드웨어가 복잡해지므로 우리가 사용하는 실습보드에는 병렬 전송을 직렬 전송으로 변환시켜 주는 회로를 추가하여(LCD 아래에 별도로 부착된 기판) 사용하고 있습니다. 그리고 한글 표시는 표준 글자 표시가 아니고 조합으로 만들어 표시하므로 한글을 일반 LCD에 직접 표시하기에는 어려움이 많이 있지요. 그래서 한글을 표준 코드로 주면 글자로 표시해 주는 역할도 이 추가 보드가 하게됩니다. 우리는 이것을 한글 LCD라 부릅니다. 한글 LCD의 사용은 직렬 통신을 이해하기 위한 수단으로 사용하는 것이므로 설명은 이 정도만 하겠습니다. 구체적인 관심이 있는 사람은 다른 자료를 참고하시오.

2. 직렬 통신이란

장치간에 정보를 주고받기 위해서는 신호가 다닐 수 있는 선을 만들어야 합니다. 그리고 주고받는 정보가 서로 같은 의미로 사용되기 위해서는 표준 코드를 사용합니다. 가장 일반적인 표준 코드는 ASCII(American Standard Code for Information Interchange)로 기호, 숫자, 대소영문자를 나타내는 표준 코드입니다. 앞의 개수는 총 128개이며, 2진수 7bit로 표현됩니다. 그러면 컴퓨

표 E7-1. ASCII TABLE

Ctrl	Dec	Hex	Char	Code
^@	0	00		NUL
^A	1	01	☺	SOH
^B	2	02	☻	SIX
^C	3	03	♥	EIX
^D	4	04	♦	EOI
^E	5	05	♣	ENQ
^F	6	06	♠	ACK
^G	7	07	•	BEL
^H	8	08	◘	BS
^I	9	09	○	HI
^J	10	0A	◙	LF
^K	11	0B	♂	VI
^L	12	0C	♀	FF
^M	13	0D	♪	CR
^N	14	0E	♫	SO
^O	15	0F	☼	SI
^P	16	10	►	SLE
^Q	17	11	◄	CS1
^R	18	12	↕	DC2
^S	19	13	‼	DC3
^T	20	14	¶	DC4
^U	21	15	§	NAK
^V	22	16	▬	SYN
^W	23	17	↨	EIB
^X	24	18	↑	CAN
^Y	25	19	↓	EM
^Z	26	1A	→	SIB
^[	27	1B	←	ESC
^\	28	1C	∟	FS
^]	29	1D	↔	GS
^^	30	1E	▲	RS
^_	31	1F	▼	US

Dec	Hex	Char
32	20	sp
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?

Dec	Hex	Char
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	/
93	5D	]
94	5E	^
95	5F	_

Dec	Hex	Char
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	⌂†

† ASCII code 127 has the code DEL. Under MS-DOS, this code has the same effect as ASCII 8 (BS). The DEL code can be generated by the CTRL+BKSP key.

Dec	Hex	Char
128	80	Ç
129	81	ü
130	82	é
131	83	â
132	84	ä
133	85	à
134	86	å
135	87	ç
136	88	ê
137	89	ë
138	8A	è
139	8B	ï
140	8C	î
141	8D	ì
142	8E	Ä
143	8F	Å
144	90	É
145	91	æ
146	92	Æ
147	93	ô
148	94	ö
149	95	ò
150	96	û
151	97	ù
152	98	ÿ
153	99	Ö
154	9A	Ü
155	9B	¢
156	9C	£
157	9D	¥
158	9E	₧
159	9F	ƒ

Dec	Hex	Char
160	A0	á
161	A1	í
162	A2	ó
163	A3	ú
164	A4	ñ
165	A5	Ñ
166	A6	ª
167	A7	º
168	A8	¿
169	A9	⌐
170	AA	¬
171	AB	½
172	AC	¼
173	AD	¡
174	AE	«
175	AF	»
176	B0	▒
177	B1	▒
178	B2	▒
179	B3	│
180	B4	┤
181	B5	╡
182	B6	╢
183	B7	╖
184	B8	╕
185	B9	╣
186	BA	║
187	BB	╗
188	BC	╝
189	BD	╜
190	BE	╛
191	BF	┐

Dec	Hex	Char
192	C0	└
193	C1	┴
194	C2	┬
195	C3	├
196	C4	─
197	C5	┼
198	C6	╞
199	C7	╟
200	C8	╚
201	C9	╔
202	CA	╩
203	CB	╦
204	CC	╠
205	CD	═
206	CE	╬
207	CF	╧
208	D0	╨
209	D1	╤
210	D2	╥
211	D3	╙
212	D4	╘
213	D5	╒
214	D6	╓
215	D7	╫
216	D8	╪
217	D9	┘
218	DA	┌
219	DB	▒
220	DC	▒
221	DD	▒
222	DE	▒
223	DF	▒

Dec	Hex	Char
224	E0	α
225	E1	ß
226	E2	Γ
227	E3	π
228	E4	Σ
229	E5	σ
230	E6	µ
231	E7	τ
232	E8	Φ
233	E9	Θ
234	EA	Ω
235	EB	δ
236	EC	∞
237	ED	φ
238	EE	ε
239	EF	∩
240	F0	≡
241	F1	±
242	F2	≥
243	F3	≤
244	F4	⌠
245	F5	⌡
246	F6	÷
247	F7	≈
248	F8	°
249	F9	∙
250	FA	·
251	FB	√
252	FC	ⁿ
253	FD	²
254	FE	■
255	FF	

터에서 표준으로 사용하는 8bit(1byte)와는 1bit 차이가 있는데, 이1bit는 코드 확장용으로 사용합니다. 즉 ASCII는 8bit code로 완전히 정의된 영역은 128개이고 나머지 128개는 확장용이라고 생각하시면 됩니다. 그러므로 영어를 제외한 글자는 이 확장 영역을 이용하여 만들어진다고 생각하시면 됩니다.

따라서 의미 있는 정보가 전송되기 위해서는 최소한 8bit의 데이터가 전송되어야 합니다. 8bit를 직접 전송하려면 전송선로가 8개로 만들어져야 하지요. 짧은 거리는 가능하겠지만 장거리는 선로 비용이 너무 많이 들겠지요. 그래서 신호를 조작하게 되지요. 즉 8bit를 동시에 보내지 않고 1선만을 사용하여 1bit를 보낸 후 다음 1bit를 순서대로 보내는 방법을 사용하지요. 이것을 직렬통신(serial communication)이라고 하며, 전자를 병렬통신(parallel communication)이라고 합니다. 여러분이 흔히 사용하는 프린터는 병렬을 많이 사용하며, 인터넷은 당연히 장거리이므로 직렬이지요. 직렬이 하드웨어의 비용을 줄일 수 있으므로 앞으로 프린터도 점차 직렬로 바뀔 것입니다.

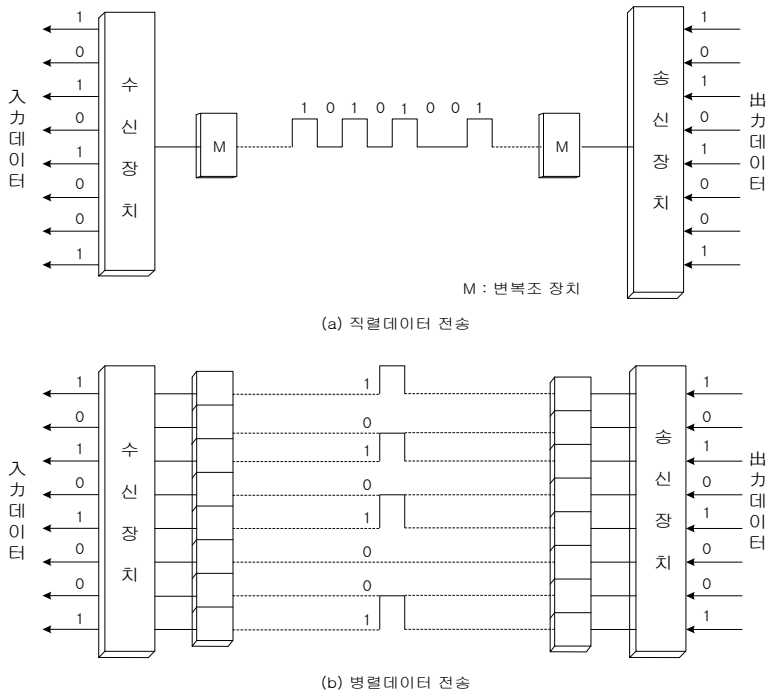


그림 E7-1. 병렬과 직렬 데이터 전송방식

직렬로 데이터를 전송할 때도 어떤 표준이 필요하겠지요. 크게 전기적인 신호의 형태와 전송속도로 나누어집니다. 전송속도를 나타내는 용어가 있으며, 이를 baud rate라고 합니다. 즉 1초 동안 몇 bit를 전송하느냐를 나타내는 용어입니다. baud rate가 9600bps(bit per second)라는 것은 1초에 9600개의 bit를 전송한다는 의미입니다. 그러면 8bit가 데이터 한 개가 되므로 1200byte를 전송하는 것이 될까요? 아닙니다. 데이터만 연속적으로 전송하면 어디가 시작이고 끝인가를 알 수가 없지요. 그러면 데이터가 정상적으로 전송될 수 있을까요? 따라서 정보를 보내기 위해서는 앞뒤에 특별한 인식 코드를 사용합니다.

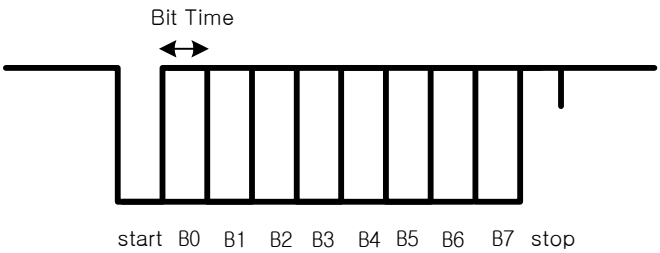


그림 E7-2. 직렬 통신 데이터의 구성

그림에서 알 수 있듯이 start bit와 stop bit 이지요. 그리고 이런 bit가 꼭 1bit 일 필요는 없지요. 이런 것을 서로 약속하여 사용하며, 이런 약속을 'protocol' 이라고 합니다. 아주 중요한 말입니다. 따라서 통신을 하기 전에 두 장치 사이에 protocol를 먼저 통일 시켜야 합니다. 또 기계적인 구조나 전기적인 신호 형태도 당연히 통일되어 있어야 하겠지요. 직렬 통신에서 가장 오래되고 기본이 되는 것이 RS232C 라는 통신 규약입니다.

RS232C는 모뎀과 터미널간의 인터페이스 방법으로 EIA(세계 전자공업협회)의 RS-x(recommanded standards-x) 규격으로 그 사용법이 간단하여 비동기식 데이터 통신으로 널리 사용되고 있습니다. 원래는 모뎀 제어선을 포함하여 25pin(DB_25라고 함)으로 시작되었으나, 사용하지 않는 pin을 제거하여 9pin(DB_9라고 함)으로 축소하여 사용하고 있습니다.

9pin 중 순수한 데이터 전송선은 2번 pin과 3번 pin입니다. 나머지는 모뎀 제어 선입니다. 그리고 당연히 전기는 2선이 있어야 흐르므로 5번 선인 GND도 사용됩니다. 여기서 2번과 3번 pin은 장치가 주가 되느냐 종이 되느냐에 따라서 동작이 정 반대로 되므로 사용할 때 신경을 써야할 부분입니다. 즉 컴

퓨터 측(master 입장, DCE : data communication equipment)에서는 2번 pin은 신호를 받는 입력단자입니다. 그러나 외부 장치(slave 입장, DTE : data terminal equipment)에서 신호를 내 보내는 단자가 됩니다. 3번 pin도 같은 의미로 사용됩니다.

표 E7-2. RS232C의 신호선 종류 및 의미 (9pin : DCE 경우)

명 칭	신호명	핀 번호	비 고
Carrier Detect	CD	1	캐리어 검출
Receive Data	RD	2	수신 데이터
Transmitted Data	TD	3	송신 데이터
Data Terminal Ready	CTR	4	단말 장치의 송·수신 가능
Signal Ground	GND	5	신호 접지
Data Set Ready	DST	6	모뎀의 송·수신 가능
Request to Send	RTS	7	송신 요구
Clear to Send	CTS	8	송신 가능
Ring Indicator	RI	9	호출 표시

그리고 RS232C는 전기적인 규격이 0에서 5V 크기를 갖는 신호로직 신호 출력을 바로 사용하는 것이 아니고, 잡음에 강하도록 신호의 크기를 크게 하여 사용합니다. 즉 -5V(이 보다 적어야함)에서 +5V(이 보다 커야함)가 사용됩니다. 따라서 신호 변환 소자가 필요합니다. 대표적인 소자로 MAX232C가 있으며, 5V 신호를 ±8V로 변환합니다. 자세한 것은 maxim 이라는 회사에 들어가 자료를 찾아보세요.

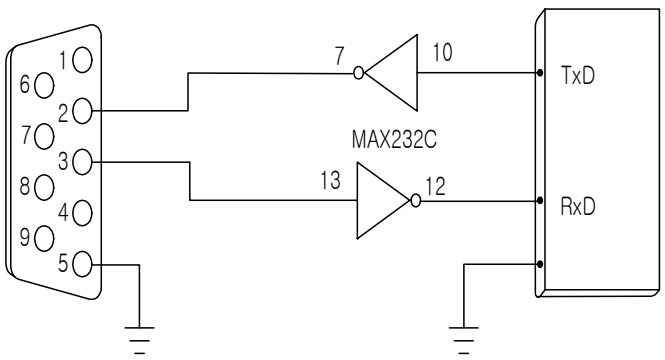


그림 E7-3. 데모보드에 구성된 RS232C 회로도 (전원회로 생략)

3. 직렬 통신용 보내는 신호 만들기

직렬 통신을 하기 위해서는 통신 규약을 서로 통일하여야 한다고 설명하였습니다. 우리가 사용하려는 LCD모듈은 4800bps에 1 start bit, 8bit data, no parity, 1 stop bit의 통신 규약을 가지므로 이 format의 serial 통신 신호 파형을 만들어 보겠습니다.

먼저 4800bps는 1초당 4800 bit를 내보내는 것을 의미하며, 1 bit는 한 상태를 나타내므로, 결국 한 상태가 유지되는 시간은 1/4800 sec가 되지요. 즉 1 bit 데이터를 출력하고 1/4800 sec 동안 유지한 후에 다음 bit의 상태를 출력한다는 것입니다. 구체적으로 직렬 통신 출력으로 내보내는 bit 상태는 그림 7.2에 나타낸 것처럼 먼저 start bit가 출력됩니다. Start bit는 '0'이 출력됩니다. 그리고 전송하고자 하는 데이터를 LSB부터 차례로 내 보내면 됩니다. 마지막 7bit인 MSB가 출력되면 parity bit를 출력하는데, 우리는 no parity로 선언했으므로 출력하지 않는다는 말입니다. 따라서 마지막으로 stop bit를 1bit 내 보내면 1개의 데이터 전체가 전송됩니다. 즉 1개의 데이터는 10bit로 구성된다는 것이며, 최대 전송 속도는 초당 480자가된다는 것입니다. (데이터와 데이터 사이의 간격으로 실제는 이것보다 약간 적어짐) 이러한 역할을 프로그램으로 만들어 보겠습니다.

예1) COM_B 라는 변수에 들어 있는 데이터를 직렬로 출력한다.

1) Start bit로 직렬 통신용 I/O port에 '0'을 출력한다
2) Baud rate에 해당하는 시간 지연을 시킨다.
3) COM_B의 bit0의 값을 확인하여 직렬 통신용 I/O port에 출력한다
4) Baud rate에 해당하는 시간 지연을 시킨다.
5) Bit 위치를 0에서 7까지 변경하여 3,4)를 반복한다.
6) Stop bit로 직렬 통신용 I/O port에 '1'을 출력한다.
7) Baud rate에 해당하는 시간 지연을 시킨다.

위 단계를 전송하고자 하는 데이터에 대하여 반복하면 직렬 통신이 가능하게 됩니다. 당연히 부 프로그램으로 작성하는 것이 유리하겠지요. 부 프로그램의 이름은 'RS232C'로 하고, RA0 pin을 직렬 통신 출력 단자로 사용할 때를 기준으로 작성 해 보겠습니다.

예2) 예1)의 프로그램

```
RS232C  BCF     PORTA,0     ; start bit=0
        CALL    DELAYB      ; 1/4800 sec delay
        BTFSS   COM_B,0     ; 전송 데이터 0bit 확인
        GOTO    BIT0_0
        ; 0 bit=1
        BSF     PORTA,0
        CALL    DELAYB
        GOTO    BIT1
BIT0_0  BCF     PORTA,0
        CALL    DELAYB
; 1 bit 확인
BIT1    .
        .
        .
; 7 bit 확인
BIT7    BTFSS   COM_B,7     ; 전송 데이터 7 bit 확인
        GOTO    BIT7_0
        ; 7 bit=1
        BSF     PORTA,0
        CALL    DELAYB
        GOTO    BIT8
BIT7_0  BCF     PORTA,0
        CALL    DELAYB
 ; stop bit 출력
BIT8    BSF     PORTA,0
        CALL    DELAYB
        RETURN              ; 부 프로그램이므로
```

프로그램이 너무 지저분하지요. 반복 부분을 간단히 하여 보도록 합시다. 역시 반복 횟수를 체크하는 buffer가 필요하지요. 'COM_C' 라고 선언하지요. 그리고 데이터의 bit 위치를 직접 확인하는 형태는 반복으로 바꿀 수가 없지요. 왜냐하면 동일한 프로그램이 아니니까요. 그러면 어떻게 해야 하나요? 데

이터 위치에 관계없이 동일한 프로그램 형태가 되려면 위치를 통일해야 되지요. 즉 오른쪽으로 shift하면서 확인하면 동일한 위치에서 확인이 가능하지요. 이해됩니까? 예를 보이면, 데이터의 bit 0을 확인하여 통신 port로 출력하고 데이터를 오른쪽으로 shift하면 데이터 bit 1의 값이 bit 0으로 이동하므로 항상 bit 0만 확인하면 되지요.

예3)

```
RS232C  MOVLW   8
        MOVWF   COM_C
        BCF     PORTA,0     ; start bit=0
        CALL    DELAYB      ; 1/4800 sec delay
; 데이터 출력 LOOP
R_LP    BTFSS   COM_B,0     ; 전송 데이터 0bit 확인
        GOTO    BITX
        ; 0 bit=1
        BSF     PORTA,0
        GOTO    NEXT1
BITX    BCF     PORTA,0
NEXT1   CALL    DELAYB
; 다음 데이터 만들기 및 마지막 확인
        RRF     COM_B,F
        DECFSZ  COM_C
        GOTO    R_LP
; stop bit 출력
        BSF     PORTA,0
        CALL    DELAYB
        RETURN
```

☞ 참고: 시간지연 프로그램은 앞에서 사용한 것을 사용하면 된다. 다만 계산식에 의한 시간지연이 loop 수행시간까지 고려하여 계산된 것이 아니므로 적절히 증/감하여 확인하시오.

☞ 참고: 그림 E7-2의 직렬통신 데이터의 그림을 보면 '1'인 상태에서 start

bit는 항상 '0'이여야 하므로 RS232C를 부르기 전에 통신 port는 '1'이 되어 있어야 첫문자가 정상적으로 출력된다.

4. LCD를 동작시키기

앞에서 설명하였듯이 데모 보드에 있는 한글 LCD는 직렬 통신으로 동작됩니다. LCD에 글자를 쓰기 위해서는 먼저 통신 규약(protocol)을 알아야합니다. 통신규약에는 기초적인 것으로 하드웨어적인 전기신호규약과 정보를 정상적으로 주고받기 소프트웨어적인 규약이 있습니다. LCD모듈의 전기신호규약은 앞에서 설명한 **4800bps에 1 start bit, 8bit data, no parity, 1 stop bit의 통신 규약**을 가지며, 이를 만들기 위한 방법은 설명하였습니다. 그러나 보다 더 중요한 것은 주고받는 정보에 대한 소프트웨어적인 규약입니다. 먼저 글자에 대한 code를 알아야 합니다. 영문자 등은 표준 code인 ASCII를 따릅니다. 암호화된 글자는 당연히 이 code도 따르지 않겠지요. 그리고 한글 코드를 알아야 합니다. 한글 코드는 아래와 같습니다.

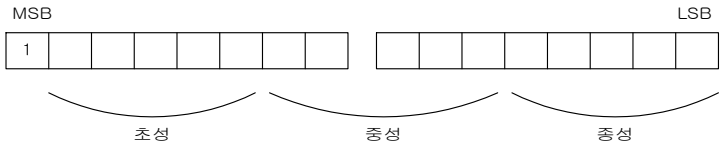


그림 E7-4. 한글 코드 bit 구성

한글을 나타내는 방법은 여러 가지가 있으며, 가장 많이 사용하는 것이 2 byte 조합형 코드입니다. 즉 2 byte를 묶어서 한글 1 자를 나타내는 방식입니다. 앞에서 약간 설명하였지만, 표준형 ASCII에서 벗어나므로 MSB를 1로 하여 만들어지는 코드를 사용합니다. 초성 중성 종성 각 5 bit 가 할당되며, 따라서 1+3*5=16bit 가 되지요.

"아름다운 우리나라"를 조합형 코드로 표현하면 다음과 같습니다.

아	름	다	운	우	리	나	라
0B4H, 061H	09FH, 071H	094H, 061H	0B6H, 085H	0B6H, 081H	09FH, 0A1H	090H, 061H	09CH, 061H

표 E7-3. 한글 LCD모듈에 들어있는 코드표

코드(16진)	초 성	중 성	종 성
0			
1			종성이 없는 경우
2	ㄱ		ㄱ
3	ㄲ	ㅏ	ㄲ
4	ㄴ	ㅐ	
5	ㄷ	ㅑ	ㄴ
6	ㄸ	ㅒ	ㄵ
7	ㄹ	ㅓ	ㄶ
8	ㅁ		ㄷ
9	ㅂ		ㄹ
A	ㅃ	ㅔ	ㄺ
B	ㅅ	ㅕ	ㄻ
C	ㅆ	ㅖ	ㄼ
D	ㅇ	ㅗ	ㄽ
E	ㅈ	ㅘ	
F	ㅉ	ㅙ	

코드(16진)	초 성	중 성	종 성
10	ㅊ		ㄾ
11	ㅋ		ㅁ
12	ㅌ	ㅚ	
13	ㅍ	ㅛ	ㅂ
14	ㅎ	ㅜ	ㅄ
15		ㅝ	ㅅ
16		ㅞ	ㅆ
17		ㅟ	ㅇ
18			ㅈ
19			ㅊ
1A		ㅠ	ㅋ
1B		ㅡ	ㅌ
1C		ㅢ	ㅍ
1D		ㅣ	ㅎ
1E			
1F			

그러면 구체적으로 LCD에 글자를 써 보도록 합시다. LCD가 한글이므로 2번 데이터를 내 보내야 한글 1자가 나옵니다. 데모보드의 LCD는 근거리용으로 사용하므로 잡음의 영향을 고려할 필요가 없어 신호 변환과정을 거치지 않습니다(RS232C가 아니지요). 즉 로직 신호 출력으로 바로 직렬 통신이 가능하다는 이야기입니다. 따라서 마이크로프로세서에서 출력되는 RA0을 LCD

module의 RX 단자에 직접 연결하면 됩니다(데모보드의 0V단자는 공통이므로 0V연결은 어느 한 곳만 해주면 됨).

예4) LCD에 '우리' 이라는 글자를 써보자.
'우리'의 코드는 '1' '01101' '10,100' '00001' + '1' '00111' '11,101' '00001' 이다.

```
        MOVLW       B'10110110'
        MOVWF       COM_B
        CALL        RS232C
        CALL        DELAY       ; 안정된 동작을 위하여
        MOVLW       B'10000001'
        MOVWF       COM_B
        CALL        RS232C
        CALL        DELAY       ; 안정된 동작을 위하여
        MOVLW       B'10011111'
        MOVWF       COM_B
        CALL        RS232C
        CALL        DELAY       ; 안정된 동작을 위하여
        MOVLW       B'10100001'
        MOVWF       COM_B
        CALL        RS232C
        CALL        DELAY       ; 안정된 동작을 위하여
BBLP
        GOTO        BBLP        ; 프로그램 종료
```

위 프로그램을 수행시키면 동작될까요? 아닙니다.

왜냐고요? 데이터만 보내준다고 전부 동작하는 것이 아닙니다. LCD모듈을 초기화시키고, 글자를 쓰는 위치를 지정해주고, 또 다양한 형태의 글자나 글자체를 나타내기 위해서는 LCD에 글자 정보가 아닌 제어 정보가 필요하지요. 쓰이는 글자의 형태를 바꾸어 주는 제어 명령어도 별도로 필요합니다. 그리고 당연히 어떤 순서에 따라서 데이터를 보내야 정상적인 동작이 되겠지요. 다음은 한글 LCD 제어코드입니다.

표 E7-4. 한글 LCD 제어코드

명 령	설 명	비 고
0A0H	LCD를 초기화합니다 (A와 0을 보내는 것이 아니라 숫자 A0을 보내야 합니다. 즉, 41H, 30H를 보내는 것이 아닌, 0A0H를 보냅니다.)	
0A3H, 1	화면을 클리어 합니다	
0A1H, X, Y	표시위치를 지정합니다 (X: dot 단위값, Y: 줄 단위값)	
0A2H,STRING, 0	STRING변수에들어있는 코드를 LCD에 글자로 표시합니다.	사용상에 주의필요
0B0H, 0	9×16 샘물체 폰트로 설정	
OB0H, 1	9×16 고딕체 폰트로 설정	
0B0H, 2	16×16 명조체 폰트로 설정	
0B0H, 3	16×16 태고딕체 폰트로 설정	
0C0H, 1	가로 2배 확대	HLCD 112, 114 전용
0C0H, 0	가로 2배 확대를 취소 (본래크기로)	HLCD 112, 114 전용
0D0H, 0	반전 ON (반전을 이용하면 커서를 구현할 수 있습니다.)	
0D0H, 1	반전 OFF	
0A4H, DATA 32개	그래픽 데이터 표시 (n은 32개) 16×16으로 그래픽을 표시	

☞ 실험 데모보드에 장착된 LCD는 HLCD114로 4줄로 21자(영문자), 11자(한글)를 표시할 수 있습니다.

초기화는 일반적으로 전원이 ON되면 이루어지므로 꼭하지 않아도 좋으나, LCD에 글자를 나타내기 위해서는 지금 나가는 데이터가 글자 데이터라는 것을 꼭 알려 주어야 하지요. 무슨 code입니까?(표에서 찾아보세요) '0A2H'라는 code이지요. 즉 글자를 보내기 전에 '0A2H'를 보내고 표시하고자 하는 글자

code를 보내면 됩니다. 또 당연히 **표시할 데이터가 끝났다는 code도 필요하지요. 이것은 '00'입니다**. 제어 code 표의 4번째 줄의 내용입니다. 다른 제어 code도 한번씩 사용해 보세요. 우리가 사용하는 모듈에서는 실행되지 않는 기능도 있을 수 있습니다. 그리고 가장 마지막 제어 명령어는 여러분이 직접 16×16 dot로 점을 찍어서 무엇인가를 그리는 그래픽 코드이지요. 실제로 모든 한글은 이런 기능으로 만들어집니다. 이러한 제어코드는 복잡한 기능을 가지는 것들이므로 코드를 받아서 처리하는데 시간이 걸리겠지요. 실험장비에 있는 한글제어기는 대략 제어코드를 처리하는데 최대 약 5msec의 시간이 걸립니다. 따라서 여러분이 제어코드를 출력할 때는 출력후 5msec동안은 다시 새로운 제어코드를 보내면 안됩니다.

이렇게 직렬통신을 프로그램으로 전부 구현하면 너무나 프로그램이 하는 일이 많아서 다른 일을 할 수가 없습니다. 그래서 CPU의 효율성을 높이기 위해서 하드웨어로 데이터를 직렬데이터로 만들어 보내는 기능을 구현하여 사용하지요. 우리가 접하는 거의 모든 MCU에서는 하드웨어로 만들어진 통신회로가 내장되어 있으며, 일반적인 명칭은 SPI/ I^2C/SCI/, UART 등으로 다양한 구조와 기능들이 있지요. 이 경우는 사용자는 단순히 주고받을 데이터를 특정 버퍼(메모리)에 기록하는 것으로 통신은 이루어집니다. 다만 통신 규약에 따르는 초기 설정은 당연히 필요하지요.

5. LCD에 표시하는 부 프로그램 만들기

앞의 예4)는 생각은 단순하나 단순 반복이므로 프로그램이 길어지며 표시되는 값을 바꾸기가 어렵지요. 따라서 항상 프로그램을 작성할 때에는 반복되는 부분이 없도록 만들고, 또 데이터의 변경이 용이하도록 데이터를 별도로 관리할 필요가 있습니다. 즉 데이터 buffer에 들어있는 데이터를 순서대로 불러내어 동작하도록 하는 방법을 사용합니다. 불러내어 사용하는 방법에는

1) 시작 위치와 끝 위치 또는 출력할 개수를 지정하여 사용하는 방법
2) 시작 위치와 끝 코드를 사용하여 출력하는 방법

으로 나눌 수 있습니다. 단순하게 생각하면, 1)이 2)보다 편할 것처럼 보이나

꼭 그런 것은 아닙니다. 1)의 경우는 변수가 1개 더 사용되며, 항상 출력할 데이터 개수를 미리 알아야 합니다. 그러나 2)는 시작 위치만 있고, 끝은 buffer에 데이터를 넣을 때 데이터의 마지막위치에 특별한 코드를 삽입하고, 출력시 이를 확인하여 끝내는 것입니다.

☞ 이 방법은 C-language에서 string 변수를 선언하고 출력하는 것과 같습니다. char str1[]=" I LOVE U_P " 이라는 문자 데이터 변수를 선언하면, buffer에 hex code로 "20, 49, 20, 4C, 4F, 56, 45, 20, 55, 5F, 50" 이라는 데이터와 데이터의 끝을 나타내는 '\0' (코드는 NULL로 hex code는 00 임) 이 추가됩니다. 따라서 str1 변수는 데이터가 들어있는 시작번지만을 가리키고 있으며, buffer에는 "20, 49, 20, 4C, 4F, 56, 45, 20, 55, 5F, 50, 00" 이 들어 있지요.

위의 설명처럼 프로그램하기 위해서는 데이터 buffer의 시작 위치만 주어지고 다음부터는 위치(메모리 주소)를 1씩 증가시키면서 마지막까지 읽어야 합니다. 그러나 우리가 사용하는 PIC 계열은 명령어 표를 보면 알 수 있지만, 명령어 구조상 직접 buffer의 주소 값을 증가시켜가면서 순차적으로 읽어오는 명령어가 없지요.

☞ Intel 등의 다른 마이크로프로세서에서는 있습니다.
(memory pointer M 또는 HL register)

그래서 특별히 이런 약점을 보완하기 위하여 INDF 와 FSR 이라는 register를 두고 있습니다. 이는 직접 데이터 buffer의 주소를 지정할 수 없으므로 간접적으로 지정하여 사용하는 방법이며(용어로는 indirect addressing이라 함), 약간 어려우므로 이렇게 사용한다고 받아들이고 data sheet에서 꼭 확인하여 보시기 바랍니다.

예5) 데이터 RAM 20H에서 2FH의 내용을 전부 '00'으로 만드는 프로그램

```
        MOVLW   20H     ; initialize pointer 의 초기값
        MOVWF   FSR     ; 초기값을 initialize pointer에 넣음
NEX     CLRF    INDF    ; clear INDF register
        INCF    FSR,F   ; inc pointer
        BTFSS   FSR,4   ; 30H 번지가 되었는가 확인
```

```
                        ; 2F 번지까지 수행하고 1 증가되면
                        ; 30H이 됨
        GOTO    NEX
        CON             ; 다음 작업 영역
```

☞ FSR 및 INDF register는 RAM 04번지와 00번지로 INDF는 물리적으로 존재하는 주소는 아닙니다. 다만 FSR에 들어있는 값을 나타내는 것이며, 따라서 간접 지정 방식이라고 합니다. 즉 위의 CLRF INDF 라는 명령어는 00번지를 clear 하라는 명령어가 아닙니다. INDF의 구체적인 주소는 FSR입니다. 이것이 C-language에서 여러분이 가장 어렵게 생각하는 pointer 변수입니다. 아이구! 어렵다. C-language까지 설명해야 되니... 아니지요, C-language도 compile되어 수행될 때는 assembly language(또는 기계어)로 돌아갑니다. 막연했던 C의 pointer 변수가 이해되지요.

☞ 여기서 이야기하는 데이터 buffer는 구체적으로 무엇일까요? 모르시겠어요? 데이터 RAM 영역이며, file register 영역 중 사용자 영역이지요. 구체적인 주소는 0CH부터 4FH까지입니다. 알고 있었지요. 모른다면 기본이 안되어 있는 것인데 더 이상 진도 나가지 말고 처음 내용을 잘 읽어보고 이해하도록 하고, 모르는게 있으면 질문... 또 질문해야지, 적당히 넘어가면 안됩니다.

예6) buffer에 들어있는 내용을 LCD에 출력하는 부 프로그램

```
LCD_OUT
        MOVLW   BUFFER  ; initialize pointer 의 초기값
        MOVWF   FSR     ; 초기값을 initialize pointer에 넣음
        ; String 데이터를 출력하기 위한 시작 코드 출력
        MOVLW   0A2H
        MOVWF   COM_B
        CALL    RS232C
NEX     MOVF    INDF,W  ; buffer 내용을 W로 옮김
        MOVWF   COM_B   ; W를 통신 출력 변수로 이동
        CALL    RS232C
        CALL    DELAYX  ; 안정된 동작을 위한 시간지연
```

```
; 마지막을 확인함
MOVF        INDF,W        ; buffer 내용을 W로 옮김
BTFSC       STATUS,ZF     ; '00' 인가 확인
RETURN                    ;끝 코드가 출력됨
INCF        FSR,F      ; inc pointer
GOTO        NEX
```

우리가 사용하는 프로세서는 1 chip으로 데이터 메모리가 내장되어 있을 뿐만 아니라, 명령어로 모든 데이터 메모리를 직접 접근할 수 있다. 따라서 고정된 값을 데이터 buffer를 이용하여 출력하기에는 오히려 번거롭다. 하지만 연산 결과 나오는 값이나, 외부에서 들어오는 변하는 값을 표시할 때에는 아주 유용한 프로그램이 된다. 우리가 실험할 때는 먼저 데이터 buffer에 표시하고자 하는 글자를 써 넣어주고 동작시켜야 한다.

예7) 부 프로그램 LCD_OUT를 이용하여 글자 표시하기

```
;  표시 글자: ' I LOVE 1'
;  앞 부분, 변수설정 등은 당연히 필요하지요.
;  그리고 서로 중복되지 않게 설정해야됩니다.

BUFFER      EQU   ??H

; 데이터 buffer에 글자를 넣은 프로그램
        MOVLW       ' '
        MOVWF       BUFFER
        MOVLW       'I'
        MOVWF       BUFFER+1
        MOVLW       ' '
        MOVWF       BUFFER+2
        MOVLW       'L'
        MOVWF       BUFFER+3
        MOVLW       'O'
        MOVWF       BUFFER+4
```

```
        MOVLW   'V'
        MOVWF   BUFFER+5
        MOVLW   'E'
        MOVWF   BUFFER+6
        MOVLW   ' '
        MOVWF   BUFFER+7
        MOVLW   '1'
        MOVWF   BUFFER+8
        MOVLW   00          ; end code
        MOVWF   BUFFER+9

; 데이터 buffer에 들어 있는 내용을 출력하는 부분
        MOVLW   BUFFER      ; initialize pointer 의 초기값
        MOVWF   FSR         ; 초기값을 initialize pointer에
                              넣음
        CALL    LCD_OUT
BBLP    GOTO    BBLP        ; 프로그램 종료

        END
```

☞ 주의사항 : BBLP GOTO BBLP 명령어가 필요한 이유를 알겠어요?

(1) 항상 프로그램을 작성할 때는 끝이 있어야 합니다. 잘 알고 있겠지만 프로그램 메모리에 있는 모든 값은 명령어가 될 수 있습니다. 따라서 우리가 원하는 명령어까지 수행시키려면 더 이상 프로그램이 진행되지 못하게 만들어야 합니다.

(2) 위의 기능을 가장 쉽게 구현하는 방법이 위 프로그램처럼 무한 Loop가 되게 하는 것입니다. 이 방법 외에 하드웨어적으로 (마이크로프로세서 자체가 가지는 기능으로)구현된 것은 없을까요? 명령어에서 SLEEP이 있다. 자세한 것은 data sheet를 참고하기 바란다.(다른 프로세서는 halt 라는 명령어도 있다)

6. 직렬 통신 수신 프로그램 만들기

직렬 통신 신호 출력하기와 같은 방법으로 직렬 통신으로 들어오는 신호를 받아서 처리할 수 있습니다. 받는 부분은 보내는 부분보다 조금 더 복잡합니다. 왜냐하면, slave 이기 때문이지요. 신호를 받아야 할 뿐만 아니라 받는 데이터를 처리해야 하기 때문이지요. 즉 처리하고 있는 도중에도, 이때는 들어오는 신호를 받을 수 없어 신호를 잃어버림으로 통신 오류가 발생됨, 신호는 계속 들어올 수 있기 때문이지요. 그래서 앞 예에서 데이터를 내 보내고 약간 쉬도록 CALL DELAYX 라는 명령어를 추가했지요. 즉 수신측이 데이터를 받아서 처리할 수 있는 시간을 갖도록 해 주는 것입니다. 따라서 여러분도 실험 도중에 데이터 오류가 발생하면 이 시간을 약간 길게 해 주기바랍니다.

직렬 통신 신호를 받기 위해서는 먼저 start bit가 들어왔는가를 확인하는 것이 필요합니다. 만약 start bit가 들어오면, 신호가 안정될 때까지 기다리는 의미에서 baud rate의 절반에 해당하는 시간 동안 시간 지연 후 start bit를 다시 확인합니다. 이때도 start bit가 들어온 것으로 확인되면 데이터를 받는 부분으로 들어갑니다. 따라서 수신 buffer를 초기화시키며, 다시 baud rate 시간만큼 지연 후 입력 값을 읽어서 데이터 bit 0에 넣습니다. 이것을 나열하면 되는데 역시 프로그램이 길어지지요. 반복하는 부분으로 변경하려면, 출력과 마찬가지로 들어오는 데이터 bit를 동일 위치에서 수신 buffer에 넣어야 하지요. 역시 shift 명령어를 사용하면 됩니다. 바로 프로그램으로 작성해 보겠습니다.

예8) 직렬 통신 받기

```
RS232R  MOVLW   8
        MOVWF   COM_C
RR11    BTFSC   PORTA,1     ; start bit=0 가 들어옴 확인
        GOTO    RR11
        CALL    DELAY1      ; 1/4800/2 sec delay
        BTFSC   PORTA,1     ; start bit=0 가 들어옴 확인
        GOTO    RR11
```

```
        ; 데이터가 들어옴
T_LP    CALL    DELAY2      ; 1/4800 sec delay
        RRF     RX_DATA,F   ; 위치 이동
        BCF     RX_DATA,7   ; 수신 데이터를 bit 위치에 넣음
        BTFSC   PORTA,1     ; 데이터를 읽음
        BSF     RX_DATA,7   ; 수신 데이터를 bit 위치에 넣음

        DECFSZ      COM_C
        GOTO        T_LP
        CALL        DELAY2
      ; stop bit 확인
        BTFSC       PORTA,1
        RETURN
      ; data error -->'00' code return
        CLRF        RX_DATA     ; 받는 buffer clear
        RETURN
```

직렬통신 받기를 실험하려면 당연히 보내기가 있어야 합니다. 즉 보내는 장치가 있어야 하며 이를 앞에서 실험한 데모보드로 구현하거나 PC를 사용하여 구현할 수가 있습니다. 실험 데모보드를 사용할 경우는 두 조가 합동으로 해야하고 두 종류 프로그램을 모두 여러분이 작성하였으므로 신뢰성이 없어(?) PC와 연결하도록 하겠습니다. 실험장비가 windows 상에서 돌아가므로 한편으로는 **PC를 터미널로 만들어** 놓고, 또 한편으로는 여러분이 작성한 PIC 프로그램을 동작시키면 됩니다. PC를 터미널로 만드는 프로그램 중 hyper terminal이라는 것이 있으며(PC의 보조프로그램중 통신에서 hypertrm.exe 수행), 이것을 사용시도 통신 규약을 서로 통일시켜야 합니다. 통신 케이블은 DCE와 DTE를 연결하는 것이므로 1:1(2번 은 2번 pin, 3번은 3번 pin에, 그리고 5번 연결)로 하면 되고, 연결단자는 DB9의 암과 수로 한쌍이 되면 된다.

☞ 만약 데모보드 2대로 여러분이 작성한 프로그램으로 주고받기 위해서는 RS232C 회로 부분을 이용할 경우는 pin 번호가 2번이 TX, 3번이 RX로 되어 있습니다. 통신 케이블은 2번과 3번이 서로 꼬여 있어야 정상적으로 연결됩니다. 왜냐하면 2개가 모두 같은 구조로 되어 있기 때문이지요.

☞ LCD모듈은 표준형이 직렬통신형이 아니고, 병렬통신형입니다. 다만 여러분이 쉽게 접근할 수 있도록 연결 pin도 줄이고, 직렬통신도 이해하기 위해 직렬통신으로 변환하는 장치를 추가하여 사용하고 있습니다. LCD모듈 자체에는 영문자 코드는 들어 있으나, 한글은 없습니다. 왜냐하면 한글은 고정된 글자가 아니라 조합으로 만들어지는 조합형이기 때문입니다. 따라서 한글은 표시하려면 글자에 해당하는 점을 만들어 표시해야 하므로 여러분이 사용하기에는 어렵지요. 따라서 이런 기능을 별도의 프로세서를 사용하여 구현해두고, 한글코드만 직렬로 받으면 동작되도록 하고있지요. 구체적으로 데모보드의 LCD를 보면 전면에 아주 많은 pin이(병렬데이터 연결) 아래의 아주 작은 보드에 직결되어 있으며, 이 보드의 3pin 단자가 직렬통신용으로 LCD용 글자를 직렬로 받아서 병렬로 변환시켜주는 기능을 가지고 있다.

☞ 통신에서 RX냐, TX냐는 각 장치의 관점에서 본 것이므로 LCD에 연결할 때는 여러분이 만든 출력을 LCD의 RX에 연결하고, PC와 연결할 때는 RS232C의 TX에 연결해야 합니다.

PC를 터미널로 사용하겠다는 의미를 이해합니까? '실험에 들어오기'를 다시 참고하세요.

■ 예비문제 7-1

1) 다음 용어를 정리해 봅시다.
   ① Modem이란 무엇인가?
   ② Interface란 무엇인가?
   ③ 비동기식 데이터 전송이란?
   ④ Parity bit란?

2) 표준 데이터 코드인 ASCII와 한글코드를 어떻게 구분하는가를 설명해 보시오.

3) 예3)의 프로그램을 자세히 설명해 보시오.
4) 예8)의 프로그램을 자세히 설명해 보시오.

5) 표준 DELAY 프로그램을 적고, 4800bps와 9600bps 에 해당하는 지연시간을 만들어 보시오.

■ 실 험 7-1

1) 예4)를 실험해 봅시다.

2) 예4)를 변경하여 각자 이름을 써 봅시다.

☞ LCD초기화 명령이 정상적으로 동작 안될 때는 여러분이 잘못하여 문자끝 코드가 나가지 않을 때입니다. 이 경우는 처음에 '00'코드를 별도로 출력시키면 LCD는 글자를 받는 모드에서 벗어나 정상적으로 동작하게 됩니다.

3) 예4)를 이용하여 LCD 화면에서 특정 위치에서부터 학번과 이름을 써 봅시다.

4) 예7)를 실험으로 확인하여 봅시다.

5) 예7)에서 표시되는 '1'을 1초에 1씩 증가시켜 봅시다.

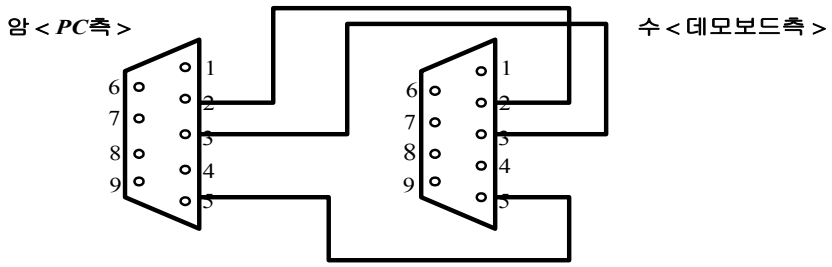
6) 4)를 이용하여 LCD 화면에서 특정 위치에서부터 학번과 이름을 써 봅시다.

7) PC의 직렬포트와 실험보드의 직렬포트를 RS232C 케이블을 이용하여 연결하고, PC와 PIC16F876 사이에 직렬통신으로 데이터를 전달하세요. 즉, PC에서 keyboard를 이용하여 입력한 문자가 실험보드의 LCD에 나타나도록 하고, PIC16F876에서 전송한 문자가 PC 모니터에 나타나도록 하세요. 이때 PC 쪽의 직렬통신 프로그램으로는 Windows에 포함되어 있는 하이퍼터미널을 이용하고, PIC16F876의 프로그램은 본문에 있는 예제 프

로그램을 적절히 활용하면 됩니다.

☞ 회로연결 시 RS232C의 TX 단자가 데모보드에서 외부로 내보내주는 신호이며, RX 단자가 외부 신호를 받아들이는 신호가 됩니다. 따라서 LCD RX 단자와 RS232C TX 단자를 서로 연결하면 LCD에 표시되는 글자가 외부로도 전달됩니다. 먼저 이렇게 PC로 출력하는 것을 확인하고 결선과 프로그램을 변경하여 직렬 통신을 받아서 LCD에 표시하도록 하시오.

PC Windows는 multi-task를 수행할 수 있으므로 현재의 에뮬레이터 환경을 완전히 벗어나지 말고 하이퍼터미널용 프로그램을 수행할 수 있다. 구체적인 것은 부록을 참고하고, 이때의 통신케이블은 다음과 같이 3선을 이용하여 만들거나, 실험실에 미리 만들어진 케이블을 사용하시오(암수 1:1로 결선된 것이면 됨).



■ HOME WORK 7 -1

1) 실험 못한 내용은 다음 시간에 할 수 있도록 프로그램을 미리 작성해 보시오.

2) 직렬통신을 프로그램으로 구현하기에는 여러 가지 문제점이 있지요. 전용 하드웨어가 있는데 이름은 UART라고 합니다. 인터넷에서 자료를 찾아서 정리해 봅시다.

3) 지금까지 실험한 내용 중 가장 어려운 것과 이해가 안된 부분을 3가지씩 적어 봅시다.
(어떤 것이, 무엇을, 왜 등으로....)

7. 내장된 통신기능을 사용하기

앞의 실험은 외부 장치와 통신으로 정보를 주고 받는 프로그램을 여러분이 일일이 만들어서 사용한 예이며, 이렇게 동작하면 컴퓨터는 이 일 이외에 다른 일을 할 수가 없습니다. 따라서 가능하면 복잡한 기능을 모두 프로그램으로 처리하기에는 한계가 있어 효율적인 사용과 처리속도를 빠르게 하기 위해여 별도의 하드웨어로 구현하여 사용하게 되며, 따라서 마이크로프로세서 주변에는 이런 용도의 많은 회로가 추가되게 됩니다.

PIC16F876에는 직렬통신용으로 특별하게 구현된 하드웨어가 3종류가 있는데, 그중 가장 기본이 되는 RS232C 통신 용 비동기식 통신(USART)을 사용해 보도록 하겠습니다.

■ PIC16F876의 USART 사용하기

PIC16F876에 들어있는 USART 또는 SCI(serial communications interface)는 아래와 같은 동작을 하며, 가장 기초적인 동작인 비동기식 기능인 UART에 대하여 설명한다.

- Asynchronous operation: full duplex
- Synchronous operation: master, half duplex
- Synchronous operation: slave, half duplex

UART는 직렬 비동기식이므로 크게 다음의 2가지 조건을 만족하도록 설정하여야 한다. 첫째로 송신이나 수신의 조건 설정이며, 다음이 통신 속도 baud rate 설정이다. 전자를 관장하는 register는 bank 1의 TXSTA와 bank 0의 RCSTA이며, 후자를 관리하는 register가 bank 1의 SPBRG이다.

◎ TXSTA(TRANSMIT STATUS AND CONTROL REGISTER)

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
CSRC	TX9	TXEN	SYNC		BRGH	TRMT	TX9D

BIT0	9비트 전송시 9번째 비트
BIT1	TSR 상태 비트(1 : TSR empty, 0 : TSR full)
BIT2	비동기 모드에서(1 : 고속, 0 : 저속)
BIT3	
BIT4	(1 : 동기 모드, 0 : 비동기 모드)
BIT5	(1 : 전송허가, 0 : 전송불가)
BIT6	(1 : 9 비트 전송, 0 : 8 비트 전송)
BIT7	동기모드에서(1 : 마스터,0 : 슬레이브)

☞ Bit2의 BRGH는 통신으로 들어온 신호 중에서 데이터를 재생하는 위치를 결정하는 방법에 따른 의미로 신호선에서 delay time이 존재하거나, 잡음이 많은 경우에는 저속 모드를 선택하는 것이 좋다. 그러나 프로세서의 시스템 clock의 주파수가 baud rate와 일치되지 못하는 경우는 오히려 표준 baud rate 값과 더 큰 오차를 가질 수 있다. 자세한 것은 다음에 나오는 baud rate register의 내용을 참조하시오.

◎ RCSTA REG

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
SPEN	RX9	SREN	CREN		FERR	OERR	RX9D

BIT0	수신된 9번째 비트가 들어 있음
BIT1	오버런 에러시 1
BIT2	프레임 에러시 1
BIT3	
BIT4	(1 : 연속수신허가, 0 : 연속수신불가)
BIT5	동기-마스터 모드(1 : 수신허가, 0 : 수신불가)
BIT6	9번째 비트 수신 허가 비트 (1 : 허가, 0 : 불허)
BIT7	직렬포트 허가 비트 (1 : 허가, 0 : 불허)

◎ SPBRG REG

Baud rate를 설정하기 위한 레지스터의 구체적인 값은 cpu clock 과 TXSTA(BRGH<BIT2>=1 고속, 0=저속)의 값에 따라 다른 값을 가진다. 주의점은 통상적인 클럭 주파수를 사용하는 경우 설정한 baud rate와 실제로 구현된 baud rate 사이에 에러가 있음을 확인해야 하며, 특히 사용 주파수가 낮고 높은 baud rate를 설정할 경우에 큰 값을 가진다. 자세한 조건은 아래 표를 참조하기 바란다.

☞ 아주 저속 통신이 아닌 경우는 BRGH를 1로 설정하는 것이 baud rate 오차를 줄일 수 있으므로 이를 사용하기 바란다.

1 송신

그림 6-6은 RS232C 송신부에 대한 블록도를 나타낸다. 블록도를 보면 알 수 있겠지만, 8bit 데이터를 TSR register에 넣고, 앞뒤에 필요한 제어 bit를 추가하며 이를 하드웨어적으로 오른쪽으로 1bit씩 shift하여 출력 pin RC6으로 나가도록 만들어진 회로이다. Shift는 오른쪽의 baud rate clock에 의해서 일어나며, data는 LSB부터 출력된다.

표 E7-5. BAUD RATE REGISTER

BAUD RATES FOR ASYNCHRONOUS MODE (BRGH = 1)

BAUD RATE (K)	Fosc = 20 MHz			Fosc = 16 MHz			Fosc = 10 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	-	-	-	-	-	-	-	-	-
1.2	-	-	-	-	-	-	-	-	-
2.4	-	-	-	-	-	-	2.441	1.71	255
9.6	9.615	0.16	129	9.615	0.16	103	9.615	0.16	64
19.2	19.231	0.16	64	19.231	0.16	51	19.531	1.72	31
28.8	29.070	0.94	42	29.412	2.13	33	28.409	1.36	21
33.6	33.784	0.55	36	33.333	0.79	29	32.895	2.10	18
57.6	59.524	3.34	20	58.824	2.13	16	56.818	1.36	10
HIGH	4.883	-	255	3.906	-	255	2.441	-	255
LOW	1250.000	-	0	1000.000		0	625.000	-	0

BAUD RATE (K)	Fosc = 4 MHz			Fosc = 3.6864 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	-	-	-	-	-	-
1.2	1.202	0.17	207	1.2	0	191
2.4	2.404	0.17	103	2.4	0	95
9.6	9.615	0.16	25	9.6	0	23
19.2	19.231	0.16	12	19.2	0	11
28.8	27.798	3.55	8	28.8	0	7
33.6	35.714	6.29	6	32.9	2.04	6
57.6	62.500	8.51	3	57.6	0	3
HIGH	0.977	-	255	0.9	-	255
LOW	250.000	-	0	230.4	-	0

BAUD RATES FOR ASYNCHRONOUS MODE (BRGH = 0)

BAUD RATE (K)	Fosc = 20 MHz			Fosc = 16 MHz			Fosc = 10 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	-	-	-	-	-	-	-	-	-
1.2	1.221	1.75	255	1.202	0.17	207	1.202	0.17	129
2.4	2.404	0.17	129	2.404	0.17	103	2.404	0.17	64
9.6	9.766	1.73	31	9.615	0.16	25	9.766	1.73	15
19.2	19.531	1.72	15	19.231	0.16	12	19.531	1.72	7
28.8	31.250	8.51	9	27.778	3.55	8	31.250	8.51	4
33.6	34.722	3.34	8	35.714	6.29	6	31.250	6.99	4
57.6	62.500	8.51	4	62.500	8.51	3	52.083	9.58	2
HIGH	1.221	-	255	0.977	-	255	0.610	-	255
LOW	312.500	-	0	250.000	-	0	156.250	-	0

BAUD RATE (K)	Fosc = 4 MHz			Fosc = 3.6864 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	0.300	0	207	0.3	0	191
1.2	1.202	0.17	51	1.2	0	47
2.4	2.404	0.17	25	2.4	0	23
9.6	8.929	6.99	6	9.6	0	5
19.2	20.833	8.51	2	19.2	0	2
28.8	31.250	8.51	1	28.8	0	1
33.6	-	-	-	-	-	-
57.6	62.500	8.51	0	57.6	0	0
HIGH	0.244	-	255	0.225	-	255
LOW	62.500	-	0	57.6	-	0

원하는 데이터를 통신하기 위한 송신 순서는 먼저 **전송버퍼인 TXREG에 전송하고자 하는 값을 로드하면 TRS 버퍼가 비워져 있으면 TSR로 전송되고 BAUD RATE GENERATOR에 의해 TX 핀을 통해 START 비트, DATA 비트, STOP 비트 순으로 한 비트씩 송신** 한다. 전송이 끝나면 TRMT(TSR buffer가 비었음을 나타내는 flag) bit가 1로 되고, TSR은 TXREG로부터 다음 보낼 data를 가져오고, TXREG로부터 TSR에 byte data가 옮겨가면 TXREG는 비어있는 상태(과거의 값이 남아있음)로 되며, TXIF 플래그 비트를 set 한다.

USART TRANSMIT BLOCK DIAGRAM

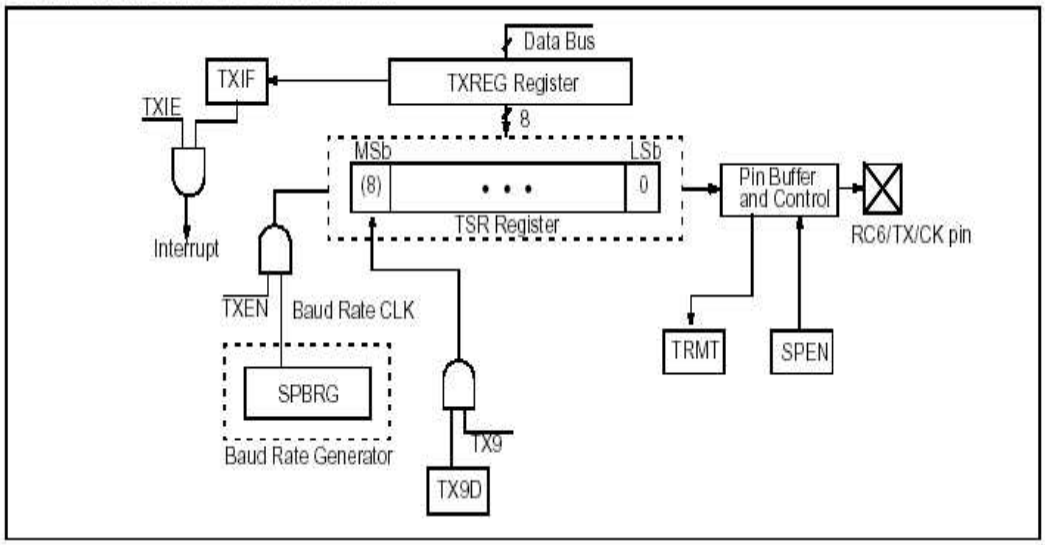


그림 E7-5. RS232C 송신부의 블록선도

표 E7-6. 송신에 사용되는 REGISTERS

REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
0Bh, 8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	R0IF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
19h	TXREG	USART Transmit Register								0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented locations read as '0'. Shaded cells are not used for asynchronous transmission.
Note 1: Bits PSPIE and PSPIF are reserved on the PIC16F873/876; always maintain these bits clear.

만약 데이터 전송이 완료되었을 경우 자동적으로 인터럽트를 걸도록 하여 반복 송신이 되도록 하려면, 전송 완료 인터럽트를 사용하기 위한 PIE1 레지스터에 있는 TXIE(PIE1<BIT4>) 프레그를 1로 만들어두어야 한다. 물론 GIE(INTCON<BIT7>)와 PEIE(INTCON<BIT6>)를 set 해두어야 한다. 그러

나 일반적으로 전송완료 인터럽트는 잘 사용하지 않는다.

☞ 송신 신호에 오류가 있는가를 확인하기 위한 parity bit를 추가 하고자 할때는 TX9를 1로 설정하고 TX9D에 출력할 parity bit 값을 넣으면 된다.

□ 송신을 위한 프로그램 순서

1. TX PORT를 출력상태로 만든다.
2. SPBRG에 BAUD RATE 값을 써 넣는다.
3. TXSTA의 SYNC=0, RCSTA SPEN=1로 만든다.(비동기식, 직렬포트 사용허가)
4. CREN=0(수신금지): * interrupt로 수신을 할 경우에 사용
5. TXEN을 SET 한다. (전송허가)
6. TXREG에 DATA를 넣는다. 넣으면 자동으로 출력된다.
7. 전송이 끝날 때 까지 대기한다. (TRMT BIT가 1이 될 때까지)

□ 송신을 위한 프로그램

예 1)
```
        ; ---- 송신 설정 부 -----
        BSF     STATUS,RP0  ;BANK 1
        BCF     TRISC,6     ;TX 단자를 출력으로
        MOVLW   B'00100100' ;BIT2(BRGH=1),BIT5(TXEN=1)
        MOVWF   TXSTA       ;전송허가, 고속
        MOVLW   .51         ;BAUD RATE->4M기준(4800bps)
        MOVWF   SPBRG
        BCF     STATUS,RP0  ;BANK 0
        MOVLW   B'10000000'  ;BIT7(SPEN=1 : 직렬포트허가)
        MOVWF   RCSTA;비동기 직력 포트 인에이블
        ;
        ; ---- 송신 데이터 이동 -----
        MOVLW   33H         ;송신데이터
        MOVWF   TXREG       ;송신 데이터 로드
        ; ---- 송신이 완료되었는가를 확인 -----
LP      BSF     STATUS,RP0  ;BANK1
```

```
        BTFSS TXSTA,TRMT; 송신완료 확인
        GOTO        LP
        BCF         STATUS,RPO ;BANK0
        ; ---- 송신 완료 -----
        ;
```

☞ 송신부를 부프로그램으로 만드려면

```
TX_OUT      ; W-reg 내용이 출력됨
        MOVWF       TXREG       ;송신 데이터 로드
        ; ---- 송신이 완료되었는가를 확인 -----
LP      BSF         STATUS,RP0  ;BANK1
        CLRWDT                  ; watch-dog 사용시 넣어야 함
        BTFSS TXSTA,TRMT; 송신완료 확인
        GOTO        LP
        BCF         STATUS,RPO ;BANK0
        NOP         ; 약간의 지연이 필요할 수도 있음
        RETURN
```

2 수신

그림 6-7는 RS232C의 수신부의 블록다이어그램을 나타낸다. 수신부의 중심은 RSR 레지스터이다. 수신된 data는 RX 핀을 통하여 data recovery에 들어오고, 여기에서 start bit가 샘플되면 data는 다시 RSR 레지스터에 옮겨갔다가 수신이 끝나면 RCREG에 다시 옮겨 간다. Stop 비트가 샘플됨으로써 하나의 바이트에 대한 수신이 끝나면 RSR 레지스터의 내용이 RCREG에 옮겨지고, 이때 수신 인터럽트 플래그 RCIF(PIR<BIT5>)가 set 된다. 만약 RCIE(PIE<BIT5>)가 set 되어서 수신 인터럽트가 개별적으로 허용되어 있고, GIE(INTCON<BIT7>)와 PEIE(INTCON<BIT6>)가 set 되어 있어서 전체적으로 인터럽트가 허용되어 있으면 직렬 수신 인터럽트가 발생한다. 이 인터럽트를 이용해서 계속 수신을 할 수 있다. 물론 수신을 인터럽트로 처리하지 않고 RCIF 플래그가 set 되는 것을 체크 하는 방법에 의해서 같은 일을 할 수 있다. 그러나 RCIF 플레그를 확인하는 프로그램 loop가 데이터 수신 시간보다도 더 걸리게 되면, 과거에 수신된 내용이 사라지게 된다. 따라서 이러한 문제점을 최소화하기 위하여 수신 레지스터인 RCREG의 메모리 용량을 1개가 아

니고 여러개 만들며, 이러한 메모리 구조를 FIFO(first in first out)라고 부르며, PIC16F87X에서는 2개가 들어있다.

USART RECEIVE BLOCK DIAGRAM

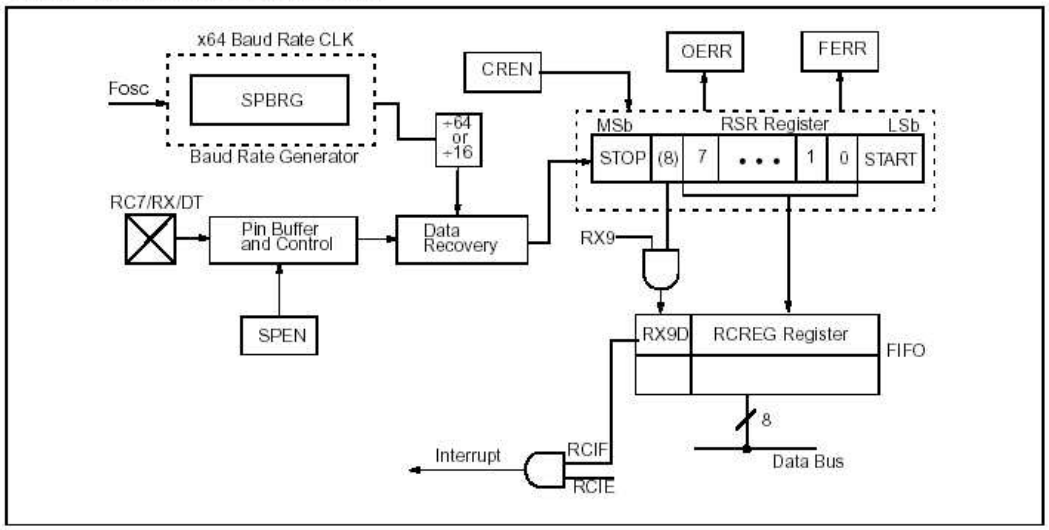


그림 E7-6. RS232C 수신부의 블록선도

표 E7-7.. 수신에 사용되는 REGISTERS

REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
0Bh, 8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	R0IF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
1Ah	RCREG	USART Receive Register								0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented locations read as '0'. Shaded cells are not used for asynchronous reception.
Note 1: Bits PSPIE and PSPIF are reserved on PIC16F873/876 devices; always maintain these bits clear.

☐ 수신을 위한 프로그램 순서

1. RX PORT를 입력상태로 만든다.
2. SPBRG에 baud rate 값을 써 넣는다.
3. TXSTA의 SYNC=0, RCSTA SPEN=1로 만든다.(비동기식 직렬포트 사용

허가)
4. CREN=1 (수신 가능상태)
5. RCIF 비트가 set 되면 한 byte가 수신된 것이다.
6. RCREG를 읽는다.

□ 수신을 위한 프로그램

1. 인터럽트를 이용하지 않는 수신

예 2)

```
        ; 수신을 위한 초기 조건 설정
        BSF     STATUS,5        ;BANK 1
        BSF     TRISC,7         ;RX PIN을 입력으로
        MOVLW   B'00100100'     ;전송허가 TXEN=1
        MOVWF   TXSTA           ;비동기모드 고속 BRGH=1
        MOVLW   .25             ;9600BPS
        MOVWF   SPBRG           ;BAUD RATE 설정
        BCF     STATUS,RP0      ;BANK0
        MOVLW   B'10010000'     ;직렬 포트 허가 SPEN=1
        MOVWF   RCSTA           ;연속 수신 허가 CREN=1
        ; 이 이하부터 수신할 수 있음

; 수신을 위한 부프로그램 - 결과는 READ_BUF에 저장됨
LP      CLRWDT      ; Interrupt를 사용하는 경우에는 넣어야 함
        BTFSS PIR1,RCIF     ;수신대기 RCIF=1 이면 수신완료
        GOTO    LP          ;수신완료시 까지 대기
        MOVF    RCREG,W     ;
        MOVWF   READ_BUF    ;
        RETURN
```

2. 인터럽트를 이용한 수신

예 3)

```
        BSF     STATUS,5        ;BANK 1
        BSF     TRISC,7         ;RX PIN을 입력으로
```

```
        MOVLW       B'00100100'             ;전송허가 TXEN=1
        MOVWF       TXSTA                   ;비동기모드 고속 BRGH=1
        MOVLW       .25                     ;9600BPS
        MOVWF       SPBRG           ;BAUD RATE 설정
        BSF         PIE1,RCIE               ;수신 인터럽트 허용
        BSF         INTCON,PEIE ;주변 인터럽트 허용
        BSF         INTCON,GIE              ;전체 인터럽트 허용
        BCF         STATUS,RP0              ;BANK0
        MOVLW       B'10010000'             ;직렬 포트 허가 SPEN=1
        MOVWF       RCSTA           ;연속 수신 허가 CREN=1
        ; ---- 초기설정이 끝남 -------------
;  main loop start
LOOP  ...


        GOTO        LOOP

; ISR: interrupt로 수신하여 결과를 READ_BUF에 넣음
RX_ISR
        BCF         PIR1,RCIF       ;수신 인터럽트 플래그 클리어
        MOVF        RCREG,W                 ;
        MOVWF       READ_BUF        ;RCREG의 DATA를 읽어서 저장
        GOTO        RETIF           ; interrupt return 부분
```

■ LCD를 동작시키기

여기서는 앞서 했던 데모 보드에 있는 한글 LCD는 직렬 통신 실험을 이제는 내장된 통신회로를 통해서 동작시켜보도록 하겠습니다.

예4) LCD에 '우리' 이라는 글자를 써보자.

'우리'의 코드는 '1' '01101' '10,100' '00001' + '1' '00111' '11,101' '00001' 이다.

```
        MOVLW       B'10110110'
        CALL        TX_OUT
        CALL        DELAY; 안정된 동작을 위하여
```

```
        MOVLW       B'10000001'
        CALL        TX_OUT
        CALL        DELAY; 안정된 동작을 위하여
        MOVLW       B'10011111'
        CALL        TX_OUT
        CALL        DELAY; 안정된 동작을 위하여
        MOVLW       B'10100001'
        CALL        TX_OUT
        CALL        DELAY; 안정된 동작을 위하여
BBLP
        GOTO        BBLP            ; 프로그램 종료
```

■ LCD에 표시하는 부프로그램 만들기

앞의 예4)는 생각은 단순하나 단순 반복이므로 프로그램이 길어지며 표시되는 값을 바꾸기가 어렵지요. 따라서 항상 프로그램을 작성할 때에는 반복되는 부분이 없도록 만들고, 또 데이터의 변경이 용이하도록 데이터를 별도로 관리할 필요가 있습니다. 즉 데이터 buffer에 들어있는 데이터를 순서대로 불러내어 동작하도록 하는 방법을 사용합니다. 불러내어 사용하는 방법에는

1) 시작 위치와 끝 위치 또는 출력할 개수를 지정하여 사용하는 방법

2) 시작 위치와 끝 코드를 사용하여 출력하는 방법

으로 나눌 수 있습니다. 단순하게 생각하면, 1)이 2)보다 편할 것처럼 보이나 꼭 그런 것은 아닙니다. 1)의 경우는 변수가 1개 더 사용되며, 항상 출력할 데이터 개수를 미리 알아야 합니다. 그러나 2)는 시작 위치만 있고, 끝은 buffer에 데이터를 넣을 때 데이터의 마지막 위치에 특별한 코드를 삽입하고, 출력시 이를 확인하여 끝내는 것입니다. 따라서 후자가 더 표준적인 사용법입니다.

☞ 이 방법은 C-language에서 string 변수를 선언하고 출력하는 것과 같습니다. char str1[]=" I LOVE u_P " 이라는 문자 데이터 변수를 선언하면, buffer에 hex code로 "20,49,20,4C,4F,56,45,20,55,75,50" 이라는 데이터와 데이터의 끝을 나타내는 '\0' (코드는 NULL로 hex code는 00 임) 이 추가됩니다. 따라서 str1 변수는 데이터가 들어있는 시작번지 만을 가리키고 있으며, buffer에는 "20,49,20,4C,4F,56,45,20,55,75,50,00" 이 들어 있지요.

위의 설명처럼 프로그램하기 위해서는 데이터 buffer의 시작 위치만 주어지고 다음부터는 위치(메모리 주소)를 1씩 증가시키면서 마지막까지 읽어야 합니다. 그러나 우리가 사용하는 PIC 계열은 명령어 표를 보면 알 수 있지만, 명령어 구조상 직접 buffer의 주소 값을 증가시켜가면서 순차적으로 읽어오는 명령어가 없지요. (Intel 등의 다른 마이크로프로세서에서는 있음: memory pointer M 또는 HL register) 그래서 특별히 이런 약점을 보완하기 위하여 INDF 와 FSR 이라는 register를 두고 있습니다. 이는 직접 데이터 buffer의 주소를 지정할 수 없으므로 간접적으로 지정하여 사용하는 방법이며(용어로는 indirect addressing 이라함), 약간 어려우므로 이렇게 사용한다고 받아들이고 data sheet에서 꼭 확인하여 보시기 바랍니다.

예5) 데이터 RAM 20H에서 2FH의 내용을 전부 '00'으로 만드는 프로그램

```
        MOVLW   20H     ; initialize pointer 의 초기값
        MOVWF   FSR     ; 초기값을 initialize pointer에 넣음
NEX     CLRF    INDF    ; clear INDF register
        INCF    FSR,F   ; inc pointer
        BTFSS FSR,4     ; 30H 번지가 되었는가 확인
                ; 2F 번지까지 수행하고 1 증가되면 30H이 됨
        GOTO    NEX
CON     ; 다음 작업 영역
```

☞ FSR 및 INDF register는 RAM 04번지와 00번지로 INDF는 물리적으로 존재하는 주소는 아닙니다. 다만 FSR에 들어있는 값을 나타내는 것이며, 따라서 간접 지정 방식이라고 합니다. 즉 위의 CLRF INDF 라는 명령어는 00번지를 clear 하라는 명령어가 아닙니다. INDF의 구체적인 주소는 FSR입니다. 이것이 C-language에서 여러분이 가장 어렵게 생각하는 pointer 변수입니다. 아이구 어렵다 C-language까지 설명해야 되니... 아니지요, C-language도 compile되어 수행될 때는 assembly language(또는 기계어)로 돌아갑니다. C의 pointer 변수가 이해되지요.

☞ 여기서 이야기하는 데이터 buffer는 구체적으로 무엇일까요. 모르시겠어요. 데이터 RAM 영역이며, file register 영역중 사용자 영역이지요. 구체적인 주소는 bank 0에서는 020H부터 07FH까지입니다. PIC16F873에는 bank가 있으므로 bank를 변경하면 더 늘릴 수 있습니다.

예6) buffer에 들어있는 내용을 LCD에 출력하는 부프로그램

```
;       MOVLW       BUFFER  ; initialize pointer 의 초기값
;       MOVWF       FSR         ; 초기값을 initialize pointer에
                                ; 넣음
LCD_OUT
NEX     MOVF        INDF,W  ; buffer 내용을 W 로 옮김
        CALL        TX_OUT
        CALL        DELAYX  ; 안정된 동작을 위한 시간지연
        INCF        FSR,F       ; inc pointer
;  마지막을 확인함
        MOVF        INDF,W      ; buffer 내용을 W 로 옮김
        BTFSS STATUS,ZF   ; '00' 인가 확인
        GOTO        NEX
; 마지막 코드인 00임 -- 글자 끝 코드 출력
        MOVLW       0           ; 00을 통신 출력 변수로 이동
        CALL        TX_OUT
        CALL        DELAYX  ; 안정된 동작을 위한 시간지연
        RETURN
```

우리가 사용하는 프로세서는 1 chip으로 데이터 메모리가 내장되어 있을 뿐만 아니라, 명령어로 모든 데이터 메모리를 직접 접근할 수 있다. 따라서 고정된 값을 데이터 buffer를 이용하여 출력하기에는 오히려 번거롭다. 하지만 연산 결과 나오는 값이나, 외부에서 들어오는 변하는 값을 표시할 때에는 아주 유용한 프로그램이다. 사용할 때는 먼저 데이터 buffer에 표시하고자 하는 글자를 써 넣어주고 동작시켜야 한다.

예7) 부프로그램 LCD_OUT를 이용하여 글자 표시하기

```
; 표시 글자: ' I LOVE 1'
;  앞 부분, 변수설정 등은 당연히 필요하지요.
;           그리고 서로 중복되지 않게 설정해야 됩니다.

BUFFER      EQU   ??H
```

```
; 데이터 buffer에 글자를 넣은 프로그램
        MOVLW       ' '
        MOVWF       BUFFER
        MOVLW       'I'
        MOVWF       BUFFER+1
        MOVLW       ' '
        MOVWF       BUFFER+2
        MOVLW       'L'
        MOVWF       BUFFER+3
        MOVLW       'O'
        MOVWF       BUFFER+4
        MOVLW       'V'
        MOVWF       BUFFER+5
        MOVLW       'E'
        MOVWF       BUFFER+6
        MOVLW       ' '
        MOVWF       BUFFER+7
        MOVLW       '1'
        MOVWF       BUFFER+8
        MOVLW       00              ; end code
        MOVWF       BUFFER+9

; 데이터 buffer에 들어 있는 내용을 출력하는 부분
        MOVLW       BUFFER  ; initialize pointer 의 초기값
        MOVWF       FSR         ;  초기값을  initialize  pointer에
                                ;넣음
        CALL        LCD_OUT
BBLP    GOTO        BBLP            ; 프로그램 종료
        END
```

■ 직렬 통신 수신 프로그램 만들기

통신 전용 하드웨어를 사용하지 않고 프로그램으로 직렬 통신을 구현하면 동시에 보내는 것과 받는 2가지 일을 할 수가 없어 이상적인 통신이 되지 않습니다. 일반 입출력 단자를 사용하여 직렬 통신 신호 출력하기와 직렬 통신 받기를 한다면, 일방적인 보내기와 받기는 가능하지만 데이터를 임으로 주고받기는 데이터의 손실이 발생합니다. 그 이유는 신호를 보내야하고 받아야 할 뿐만 아니라 받는 데이터를 처리해야 하기 때문이지요. 즉 보내고 있는 도중에도 통신으로 신호는 계속 들어올 수 있기 때문에 이때는 들어오는 신호를 받을 수 없어 신호를 잃어버림으로 통신 오류가 발생됩니다. 이러한 의미를 full 또는 half duplex 라고 한다는 것을 앞에서 설명하였지요. 또 **데이터를 받아서 처리 하는데도 시간이 걸리므로 앞 예에서 데이터를 내 보내고 약간 쉬도록 CALL DELAYX 라는 명령어를 추가**했지요. 즉 수신측이 데이터를 받아서 처리할 수 있는 시간을 갖도록 해 주는 것입니다. 따라서 여러분도 실험 도중에 데이터 오류가 발생하면 이 시간을 약간 길게 해 주기바랍니다.

직렬통신을 효과적으로 하기 위해서는 보내는 것은 메인 프로그램에서 하고, 받는 것은 interrupt를 이용하는 것입니다. 들어오는 것은 언제 들어올 줄 모르기 때문에 프로그램으로 기다리고 있으면 다른 일을 할 수가 없기 때문입니다. 그러나 보내는 것은 내가 주가 되어 하는 일이므로 바로 출력하는 것이 가능합니다. 만약 보내고 받고 또 다른 일을 하려고 한다면 어떻게 해야 합니까? 즉 **동시에 여러 가지 일을 해야 하거나 많은 보내기(출력)를 해야 할 때는 보내기도 interrupt를 이용**합니다. Interrupt를 이용한 보내기는 보내는 데이터를 저장할 버퍼(queue)를 만들어 여기에 순차적으로 넣고, interrupt에서는 이 버퍼에 들어 있는 내용을 순차적으로 출력하도록 합니다. 이때에는 데이터의 끝으로 나타내거나 출력할 데이터의 개수를 확인하여 출력하도록 하면 됩니다. 예6)의 출력프로그램을 바로 interrupt에서 불러 쓰면 됩니다. 이런 유사한 내용이 PC에서 프린터에 출력하기 입니다.

직렬통신 받기를 실험하려면 당연히 보내기가 있어야 합니다. 즉 보내는 장치가 있어야 하며 이를 앞에서 실험한 데모보드로 구현하거나 PC를 사용하여 구현할 수가 있습니다. 실험 데모보드를 사용할 경우는 두 조가 합동으로 해야하고 두 종류 프로그램을 모두 여러분이 작성하였으므로 신뢰성이 없어(?) PC와 연결하도록 하겠습니다. 실험장비가 windows 상에서 돌아가므로 한

편으로는 PC를 터미널로 만들어 놓고, 또 한편으로는 여러분이 작성한 PIC 프로그램을 동작시키면 됩니다. PC를 터미널로 만드는 프로그램 중 hyper terminal이라는 것이 있으며(PC의 보조프로그램 중 통신에서 hypertrm.exe 수행), 이것을 사용시도 통신 규약을 서로 통일시켜야 합니다. 통신 케이블은 DCE와 DTE를 연결하는 것이므로 1:1(2번 은 2번 pin, 3번은 3번 pin에, 그리고 5번 연결: 실험시간에 나누어줌)로 하면 되고, 연결단자는 DB9의 암과 수로 한쌍이 되면 된다.

☞ 만약 데모보드 2대로 여러분이 작성한 프로그램으로 주고받기 위해서는 RS232C 회로 부분을 이용할 경우는 pin 번호가 2번이 TX, 3번이 RX로 되어 있습니다. 통신 케이블은 2번과 3번이 서로 꼬여 있어야 정상적으로 연결됩니다. 왜냐하면 2개가 모두 같은 구조로 되어 있기 때문이지요.

♧ LCD모듈은 표준형이 직렬통신 형이 아니고, 병렬통신 형입니다. 다만 여러분이 쉽게 접근할 수 있도록 연결 pin도 줄이고, 직렬통신도 이해하고 등으로 직렬통신으로 변환하는 장치를 추가하여 사용하고 있습니다. LCD모듈 자체에는 영문자 코드는 들어 있으나, 한글은 없습니다. 왜냐하면 한글은 고정된 글자가 아니라 조합으로 만들어지는 조합형이기 때문입니다. 따라서 한글은 표시하려면 글자에 해당하는 점을 만들어 표시해야 하므로 여러분이 사용하기에는 어렵지요. 따라서 이런 기능을 별도의 프로세서를 사용하여 구현해두고, 한글코드만 직렬로 받으면 동작되도록 하고 있지요.

■ 예비문제 7-2

1) 다음 용어를 정리해 봅시다.
   - 비동기식 데이터 전송이란?
   - Parity bit란?
2) 표준 데이터코드인 ASCII와 한글코드를 어떻게 구분하는가를 설명하시오.
3) 예3)의 프로그램을 자세히 설명해 보시오.
4) 예7)의 프로그램을 자세히 설명해 보시오.

■ 실 험 7-2

1. 예4)를 실험해 봅시다.
2. 예4)를 변경하여 LCD 화면에서 특정 위치에서부터 학번과 이름을 쓰시오.
3. 예6)를 사용하여 interrupt로 출력하는 프로그램을 작성해 보시오.

4. 예7)를 실험으로 확인하여 봅시다.
   ☞ 초기화가 정상적으로 안되는 경우는 문자 출력 후 문자 끝 코드인 '00'이 나가자 않았기 때문입니다. 즉 LCD 회로는 문자 끝 코드가 들어오지 않았으므로 들어오는 모든 코드를 글자코드로 인식하기 때문이지요. 따라서 이 부분을 다시 확인하여 봅시다.
5. 예7)에서 표시되는 '1'를 1초에 1씩 증가 분-초 시계를 만들어 봅시다.
6. PC의 직렬포트와 실험보드의 직렬포트를 RS232C 케이블을 이용하여 연결하고, PC와 PIC16F876 사이에 직렬통신으로 데이터를 전달하세요. 즉, PC에서 keyboard를 이용하여 입력한 문자가 실험보드의 LCD에 나타나도록 하고, PIC16F876에서 전송한 문자가 PC 모니터에 나타나도록 하세요. 이때 PC 쪽의 직렬통신 프로그램으로는 Windows에 포함되어 있는 하이퍼터미널을 이용하고, PIC16F876의 프로그램은 본문에 있는 예제 프로그램을 적절히 활용하면 됩니다.
   ☞ 회로 연결시 RS232C의 TX 단자가 데모보드에서 외부로 내보내주는 신호이며, RX 단자가 외부 신호를 받아드리는 신호가 됩니다. 따라서 LCD RX 단자와 RS232C TX 단자를 서로 연결하면 LCD에 표시되는 글자가 외부로도 전달됩니다. 먼저 이렇게 PC로 출력하는 것을 확인하고 결선과 프로그램을 변경하여 직렬 통신을 받아서 LCD에 표시하도록 하시오.
7. 실험5)를 변경하여 PC에서 'M' 글자를 치면 분표시가 0으로, 'S' 글자를 치면 초표시가 0으로 되는 분-초 시계를 만들어 봅시다.
   ☞ 입력을 interrupt에서 구현 해야됨.

■ HOME WORK 7-2

1. 우리가 사용하는 UART의 내부 기능을 설명하여 보시오.

2. 지금까지 실험한 내용 중 가장 어려운 것과 이해가 안된 부분을 3가지씩 적어 봅시다.
   - 어떤 것이, 무엇을, 왜 등으로

참고. PC의 Hyper terminal 사용법

1. 하이퍼 터미널을 실행한다.

PC에서 시작→프로그램→보조프로그램→통신→하이퍼터미널 로 가서 hypertrm.exe 을 실행한다. 위의 메뉴가 없으면 hypertrm.exe을 찾아서 실행한다.

2. 혹시 모뎀 설정 화면이 나오면 [아니오]를 클릭. 안나오면 3번으로.

3. [새연결] 화면에서 이름을 입력하고 아이콘을 선택한 후 [확인] 클릭.

4. [연결대상] 화면에서 "연결에 사용할 모뎀"을 [Com1에 직접 연결] 선택 후 [확인] 클릭.

5. [포트설정] 화면에서 초당 비트수=4800, 데이터비트=8, 패리티=없음, 정지비트=1, 흐름 콘트롤=없음 으로 설정한 후 [확인] 클릭.

★ 꼭 , "흐름 콘트롤"은 [없음]으로 설정해야 합니다.

★ 이때 흐름 콘트롤을 제외한 다른 변수들은 PIC프로그램과 일치하기만 하면 됩니다. 초당 비트수를 9600이 되도록 프로그램을 작성했다면 그렇게 설정하면 됩니다.

6. 여기까지 하면 설정은 끝났습니다.

PIC에서 직렬 통신 데이터를 보내는 프로그램을 실행시키면 보낸 데이터의 ASCII 코드에 해당하는 글자가 화면에 나타날 겁니다. 이때 LCD에 한글을 쓰기 위해 작성한 한글코드를 그대로 PC로 보내면 화면에는 이상한 글자가 나타납니다. 프로그램을 잘못 작성해서 이상한 글자가 나오는 것이 아니고 LCD의 한글 코드와 windows의 한글 코드가 다르기 때문에 그런 현상이 나타나게 됩니다. 따라서 보내는 실험은 일단은 표준 ASCII code 만을 사용하도록 하고, 능력이 되면 한글도 표시되도록 하여 보세요.

그리고, 다른 조작없이 하이퍼터미널 상에서 키보드를 누르면 그 문자에 해당하는 아스키 코드가 직렬통신 포트로 바로 전송됩니다. PIC에서 직렬 통신 포트로 받은 데이터를 LCD에 나타나게 하는 프로그램을 실행시키면 키보드로 입력한 문자가 LCD에 나타나게 할 수 있습니다.

★ 하이퍼터미널의 포트설정을 변경하고자 한다면, 하이퍼터미널을 종료하고 재실행하면서 [포트설정] 메뉴에서 변경하세요. 하이퍼터미널이 실행중일 때 변경한 사항은 제대로 반영되지 않습니다.