Intelligent LCD Display Module

IntelliLCD

인텔리 엘씨디

사용설명서 Version: 2.0.6 a

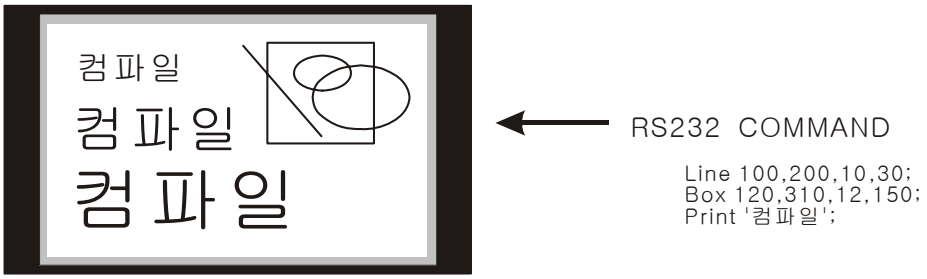
[펌웨어 v2.0.6 기준]

COMFILE TECHNOLOGY

컴파일 테크놀로지 주식회사

www.comfile.co.kr

IntelliLCD는 Color LCD를 편리하게 사용할 수 있도록 만들어진 제품입니다. Color LCD를 적용하기 위해서는 고성능(32Bit)의 CPU를 사용해야만 했습니다. 보통 32비트급 CPU정도 되어야 Color LCD 인터페이스 기능을 갖추고 있기 때문입니다. Color LCD가 필요한 어플리케이션 개발시, 이것은 개발자에게 큰 부담을 주게 됩니다. Color LCD에 그래픽을 표현 하기위한 복잡한 코드를 작성 해야 했고, 폰트구현과 터치입력 등을 처리하기 위해 많은 노력을 필요로 합니다.



저희 컴파일 테크놀로지의 IntelliLCD를 사용하게 되면 위 같은 문제 점들을 쉽게 해결하실 수 있습니다. 단순히 RS232커맨드를 송신하는 것 만으로, 컬러 LCD에서 표시하고자 하는 문자 및 그래픽을 표시할 수 있으며, 그 외에도 다음과 같은 장점들을 경험할 수 있습니다.

1. 인터페이스를 위한 CPU의 성능을 고려할 필요가 없습니다.
   - 저가의 8Bit MCU를 사용하여 사용자가 원하는 그래픽을 표현할 수 있습니다.
   - 컴파일 테크놀로지의 CUBLOC 모듈을 사용하시면 더욱 쉽게 IntelliLCD를 구동할 수 있습니다.
2. 기본적으로 RS232로 커맨드를 받아, LCD상에 표시하는 방법으로 구동합니다.
3. 한글/한자를 자유롭게 표시할 수 있습니다. 기본적으로 트루타입 (벡터) 폰트를 사용합니다. 비트맵폰트보다 미려한 글자체를 표시할 수 있습니다.
4. 다양한 그래픽 커맨드를 지원하고 있어, 그래픽을 표현하기 위한 프로그램을 최소화 할 수 있으며 그 외에도 다양한 사용자 컨트롤들이 준비되어 있습니다.
5. 그 밖의 부가적인 기능들도 갖추고 있습니다.
   - 터치 스크린 기본제공
   - SD 메모리 카드 (별도 구입품, 최대 2GByte까지 사용 가능)
   - Windows가 사용하는 트루 타입 폰트를 인텔리LCD에 옮겨서 사용할 수 있습니다.
   - 외부 USB Mouse, Keyboard 지원
   - 7 / 10.2 Inch wide 18 Bit Color LCD (800*480)
   - RTC 내장 (배터리 내장되어 있어, 전원 OFF시에도 시간/날짜가 기억됨)
     (배터리는 iTL720 또는 Cuwin의 iTL모드에서만 지원)
   - 사운드/음성 출력기능(Wave Play기능), 별도의 사운드칩 없이 안내맨트를 구현할 수 있습니다.
     (iTL720 또는 Cuwin의 iTL모드에서만 지원)

인텔리 LCD 모델

모델에 따라 화면크기와 기능차이가 있습니다.

모델명	iTL840K/S	iTL740	CUWIN3200/3500/4300 (intelliLCD로 사용가능한 CUWIN모델)
화면크기	10.2인치 와이드	7인치 와이드	7인치 / 10.2인치 와이드
전면케이스 (bezel)	있음 K=검정, S=은색	있음	3200 : BEZEL 케이스 3500 : 전면부 방수 케이스 4300 : BEZEL 케이스(검정/은색)
해상도	800 x 480	800 x 480	800 x 480
지원칼라수	26만칼라	26만칼라	26만칼라
사운드출력	X	X	O
리얼타임클럭	X	X	O
이더넷 포트	X	X	O
HTML표시 기능	X	X	O
터치입력	지원함	지원함	지원함
출시여부	구입가능	구입가능	구입가능

*각 모델의 START KIT 에는 매뉴얼, CD, 케이블, 2G SD CARD가 들어있습니다. 최초 구매시에만 START KIT를 구입하시고, 양산시에는 본체만 구입하시기 바랍니다.

*사운드 기능과 RTC기능을 원하시는 분은 CUWIN 시리즈를 구입하시기 바랍니다. CUWIN 시리즈는 intelliLCD 모드로도 사용할 수 있습니다.

타사의 시리얼 그래픽 LCD모듈과의 비교

인텔리LCD는 기존제품과 차원을 달리하는 제품입니다. 타사 제품과는 다음과 같은 차이점이 있습니다.

	A사 제품	IntelliLCD iTLXXX
터치기능	지원안함	지원함
폰트	비트맵폰트	트루타입폰트
윈도우폰트지원	안함	함
글자확대	4배까지	자유자재
키보드/마우스	지원안함	함
추가 메모리카드	지원안함	함
시뮬레이터	지원안함	지원함

타사의 제품과 가장 큰 차이점은 폰트시스템에 있습니다. IntelliLCD는 기본적으로 윈도우에서 사용하는 트루타입폰트체계를 지원하고 있으므로, 좀더 미려한 화면을 구성하실 수 있습니다. 이에 반해 타사의 제품은 비트맵방식이므로, 글자가 미려하지 못하고, 사이즈에 제한이 있습니다.

예를들어, LCD화면 가득히 글자 하나를 그렸을 경우, 비트맵은 외곽선이 울퉁불퉁하지만, 트루타입폰트는 부드러운 곡선으로 처리됩니다.

등록상표

WINDOWS는 Microsoft Corporation의 등록상표입니다.
CUBLOC은 Comfile Technology의 등록상표입니다.
IntelliLCD은 Comfile Technology의 등록상표입니다.
기타 다른 상표는 해당회사의 등록상표입니다.

알림

본 설명서의 내용은 사전 통보 없이 변경될 수 있습니다. 본 제품의 기능은 성능 개선을 위하여 사전 통보 없이 변경될 수 있습니다. 본 제품을 이 자료에서 설명한 용도 외에서 사용할 경우, 폐사에서는 어떠한 책임도 지지 않으므로 주의하시기 바랍니다. 본 제품은 컴파일 테크놀로지의 고유 기술을 사용하여 개발된 제품으로 저작권법에 의한 보호를 받고 있습니다. 따라서 본 제품 (제품에 대한 아이디어 및 설명서 및 기타 포함)의 어떠한 부분도 사전에 폐사와의 문서 동의 없이 복사되거나 변경, 재 생산할 수 없으며 또한 다른 언어로도 번역될 수 없습니다.

주의사항

인쇄된 설명서는 인쇄된 시점에서는 최신 버젼이지만, 인쇄된 후 시간이 경과된 뒤에 새로운 내용이 추가되거나, 기존내용이 바뀔 가능성이 있습니다. 최신 버젼의 설명서는 항상 인터넷 홈페이지 (www.comfile.co.kr)에서 확인하시기 바랍니다.
본 제품을 사용하시다가 생긴 손해 및 손실에 대하여 저희 컴파일 테크놀로지 주식회사는 어떠한 책임도 없음을 명시하는 바입니다. 본 제품을 사용하기 이전에 반드시 본 사용설명서를 읽어본 뒤 사용하시기 바랍니다. 본 사용설명서를 충분히 읽어보지 않은 상태로 본 제품을 사용하는 것으로 인해 발생된 피해에도 저희 회사에서는 어떠한 책임도 없음을 명시합니다.

차 례

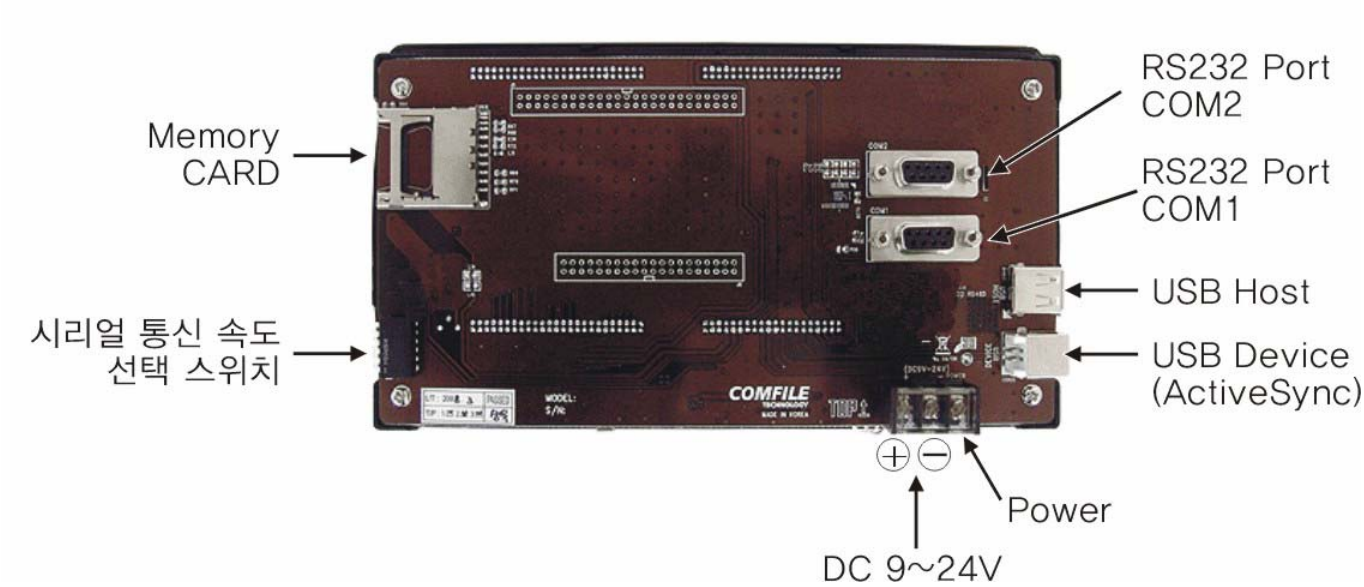
인텔리 LCD 모델 3
타사의 시리얼 그래픽 LCD모듈과의 비교 3
차 례 5
제1장 IntelliLCD 구성 7
1-1. 각부 명칭 7
1-2. 외형치수 8
1-3. 하드웨어 사양 10
제2장 IntelliLCD 통신 11
2-1. 환경 설정 11
2-1-1. 통신 설정 11
2-1-2. 연결 방법 11
제3장 IntelliLCD 명령 구성 12
3-1. 기본 사항 12
3-2. 커맨드 목록 13
*** 다국어 출력** 18
*** Screen 기능** 44
*** Screen script 파일 작성법** 45
3-3. 이벤트 46
3-3-1. 아스키모드와 바이너리 모드 47
3-3-2. 컨트롤 입력 관련 이벤트 48
3-3-3. 유저 입력 반환 이벤트 49
3-3-4. RTC관련 이벤트 50
3-3-5. Screen 관련 이벤트 50
3-3-6. 기타 이벤트 51
제4장 부가적 기능 54
4-1. 스피커 연결 54
4-2. USB Key & Mouse 연결 54
4-3. SD 메모리 카드 54
4-4. 펌웨어 업그레이드 54
4-5. 사용자 폰트 추가 55
4-6. ActiveSync 사용법 56
4-7. ActiveSync 실행 58
4-8. IntelliLCD의 저장 장치 접근하기 60
4-9. Demo 화면 설정, 바꾸기 62
제5장 큐블록에서의 사용법 63
5-1. 큐블록 스터디보드와 IntelliLCD의 연결방법 63

5-2. 큐블록에서 IntelliLCD로 커맨드 송신방법 64
5-3. 큐블록에서 이벤트수신 방법 68
제6장 인텔리 LCD 시뮬레이터 69
제7장 인텔리 LCD TestBox 70
제8장 인텔리 Unicode Generator 71
8-1. 폰트 설치 71
8-2. IME 설치 72
제9장 시스템 변수 75
9-1. 기본 시스템 변수 75
9-2. 이벤트 관련 시스템 변수 75

제1장 IntelliLCD 구성

1-1. 각부 명칭

[iTL740/840]

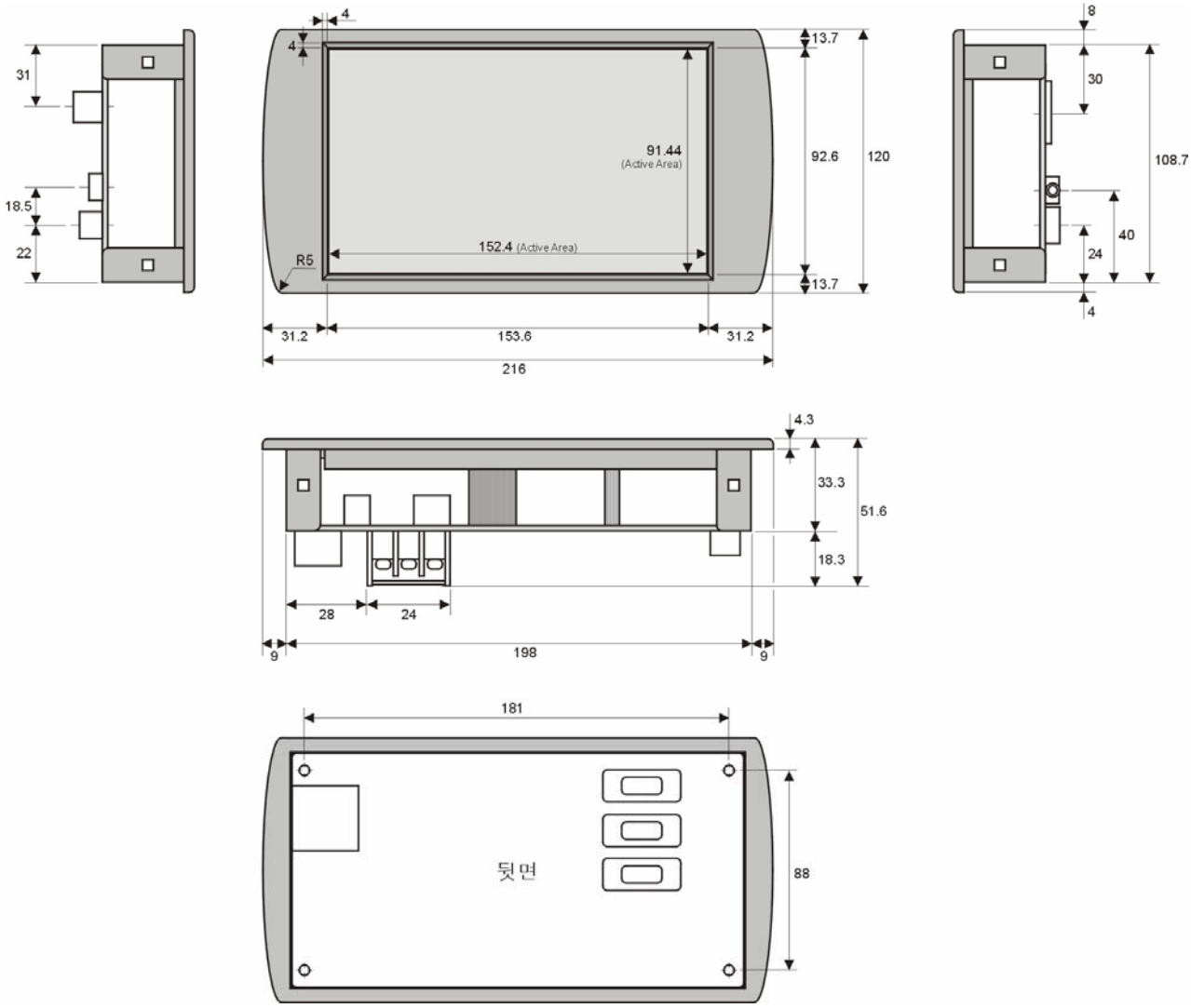


*전원연결시 극성에 주의하시기 바랍니다. 잘못된 극성으로 연결했을 경우, 제품에 손상이 있을 수 있습니다.

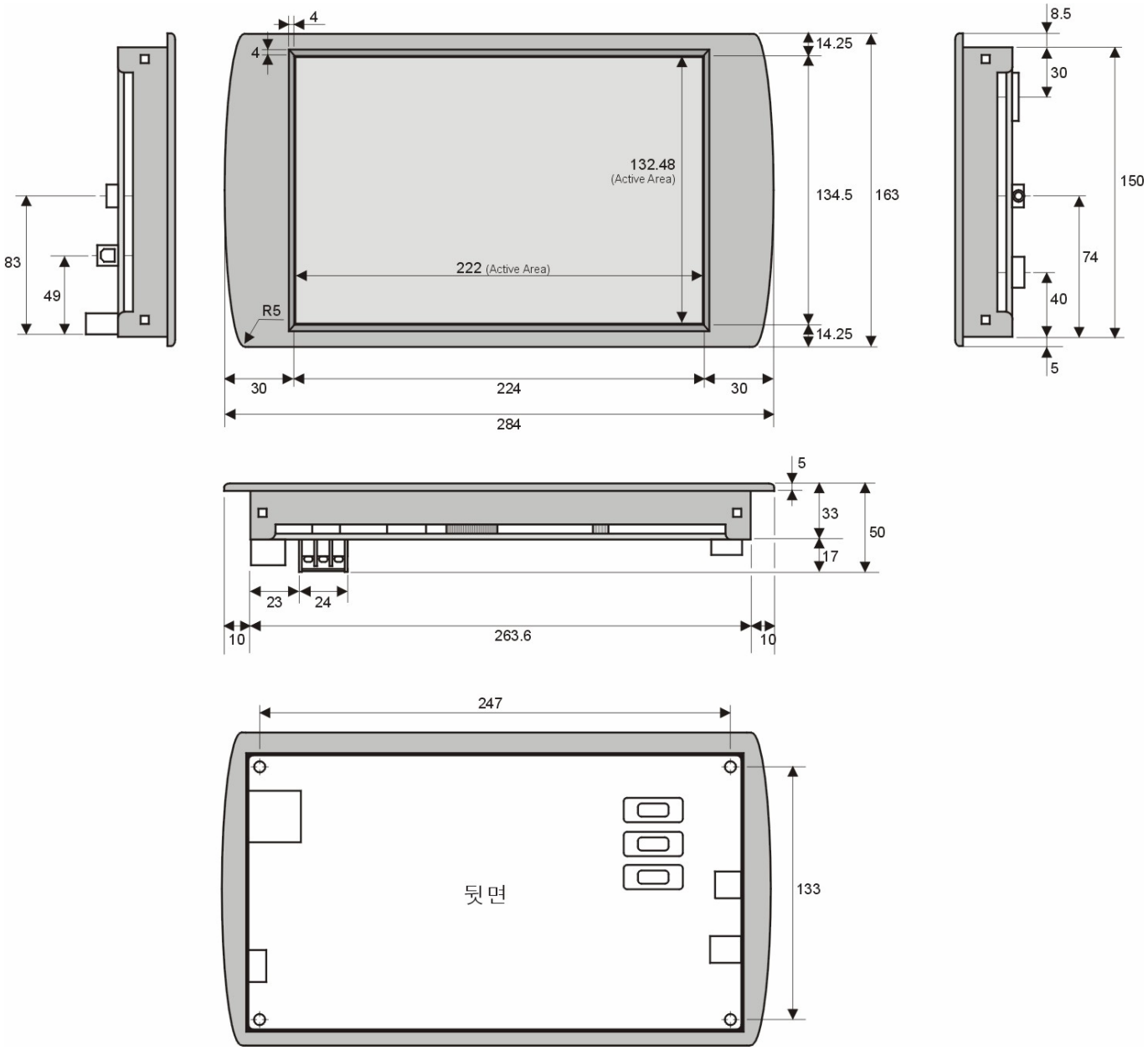
* Audio-Out 단자는 CUWIN에서만 지원합니다

1-2. 외형치수

[iTL740]



[iTL840]


4
4
14.25
132.48
(Active Area)
134.5
163
222 (Active Area)
R5
14.25
30
224
30
284
83
49
8.5
30
150
74
40
5
5
33
50
17
23
24
10
263.6
10
247
133
뒷면

1-3. 하드웨어 사양

- 32Bit RISC CPU (ARM920T Core) : 266MHz
- RAM : 64MB , NAND Flash : 64MB , NOR Flash : 1MB
- TFT Color LCD(26만 칼라) ; 800*480
- iTL740: 7 Inch Wide, iTL840: 10.2 Inch Wide
- LED Backlight
- Serial Port 2 (RS232C, 5V-TTL)
- Touch Pad
- Stereo Sound 출력 (CUWIN)
- USB (Host, Device), USB 메모리 스틱 지원
- SD 메모리 CARD 장착가능 (최대 2GB까지 지원)
- RTC 기능 내장 (CUWIN)
- 입력 전원 DC9~24V(500mA)
- 사용온도 -10도~40도 (상온)

제2장 IntelliLCD 통신

2-1. 환경 설정

2-1-1. 통신 설정

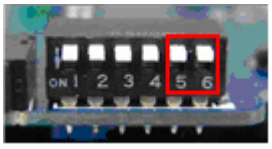
- 포트설정 : 데이터비트 8, 패리티 없음, 정지비트 1, 흐름제어 없음
- 지원 Baud Rate : 115200, 38400, 19200, 9600 bps

(주의 : 115200 bps는 연속적으로 대량의 데이터를 송신할 경우 안정성이 보장되지 않습니다. 따라서 38400 bps 이하 사용을 권장합니다.)

- Baud Rate변경 방법

다음과 같이 Dip SW의 스위치 조작으로 변경 가능 합니다. 변경 후 기기의 전원을 OFF후 다시 On 해주어야 변경됩니다.

iTL740/840은 DIP 스위치 위치가 측면에 있으며, DIP스위치 번호가 5,6번입니다.



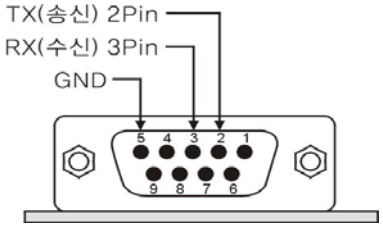
DIP스위치 상태	Baud Rate
ON 5 6	9600
ON 5 6	19200
ON 5 6	38400
ON 5 6	115200

2-1-2. 연결 방법

- 통신 포트는 RS232C가 있습니다.

다음 그림과 같이 사용목적 및 대상 기기와의 전기적인 인터페이스 사양을 확인 후 서로 연결하시기 바랍니다.

- 제품구입시 제공되는 케이블을 사용하여 연결하시면 됩니다.
- 각 컨넥터 핀 구성(보드측 컨넥터 기준)

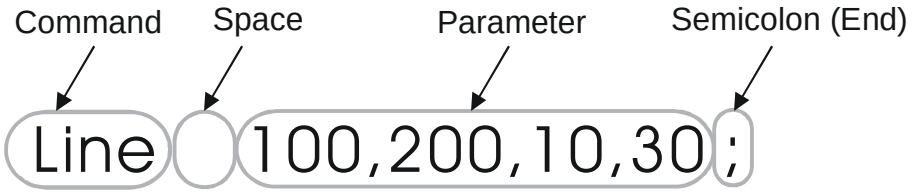


[그림 RS232C 컨넥터]

제3장 IntelliLCD 명령 구성

3-1. 기본 사항

각 커맨드는 기본적으로 1개의 명령어로 시작하여야 하며 뒤에 파라미터들을 쉼표로 구분하여 붙일 수 있습니다. 파라미터의 개수는 가변적인 경우가 있습니다. **명령과 파라미터 사이는 공백문자로 구분됩니다**.
커멘드는 대소문자를 구분하지 않습니다. 모두 대문자로 쓰거나, 소문자를써도 인식합니다.



(각 파라미터의 최대 길이는 1024 byte이며 그 이상이 올 경우 해당 커맨드는 무효화됩니다.)
하나의 커맨드는 반드시 세미콜론(;)으로 마쳐야 하며 그 후에 다시 다른 커맨드를 이어 붙일 수 있습니다.
세미콜론을 생략했을 경우 예기치 못한 **심각한 오류**가 발생할 수 있습니다.
세미콜론 대신 슬래쉬(/)를 사용할 수도 있습니다.

예) Box 50,50,100,100;Line 20,20,40,40;

3-2. 커맨드 목록

PSet

PSet *x1,y1*
x1,y1 지점에 점을 찍습니다.
예) PSet 42,56;

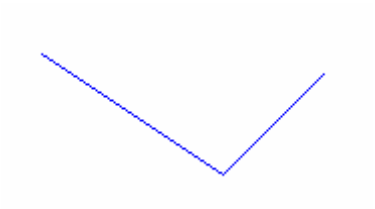
Line

Line *x1,y1,x2,y2*
x1,y1 지점으로부터 *x2,y2* 지점까지 직선을 표시합니다.
예) Line 30,30,200,200;

MoveTo

MoveTo *x,y*
선을 그릴 때 시작점을 지정합니다. LineTo에 의해 완성되어 선이 그려집니다.
예) MoveTo 300,300;

LineTo



LineTo *x,y*
계속해서 선을 그릴 경우, 마지막 지점으로부터 이어질 선을 그립니다.
예) LineTo 300,300;

Color

Color *color*

이후 그래픽 명령에 적용될 색상을 지정해둡니다. 화면 상에 바로 변화되는 것은 없습니다.

색상은 16진수 형태로 써야 하며 반드시 &H로 시작하고 뒤에 6자리가 와야 합니다.

예) Color &H0000FF; ⇒ 적색 (R)

Color &H00FF00; ⇒ 녹색 (G)

Color &HFF0000; ⇒ 청색 (B)

Color &HFF00FF; ⇒ 보라색

아래서부터 8비트씩 R, G, B에 할당되어 있습니다. 숫자가 높을수록 밝은 색이 됩니다.

Clear

Clear *[back_color]*

화면을 초기화시키고 컨트롤들을 모두 삭제합니다. *back_color*를 지정할 경우 전체 배경색으로 지정하여 초기화합니다. 처음 상태의 전체 배경색은 흰색입니다.

참고로, Clear를 실행하면 자동으로 EndScreen 기능도 수행합니다.

예) Clear;

Clear &HA00000;

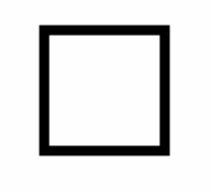
DotSize

DotSize *value*

그래픽 명령에서 사용될 점의 크기를 미리 지정합니다. 최저값은 1이고 최대값은 255입니다. 단, Arc와 Bezier의 경우는 1이 최대치입니다.

예) DotSize 4;

Box



Box *x1,y1,x2,y2*

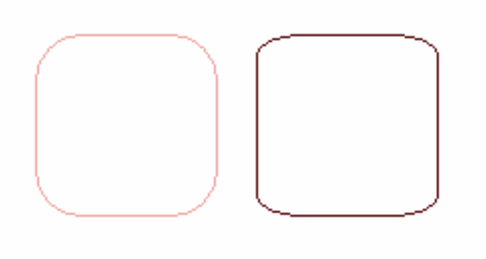
x1,y1 지점에서 *x2,y2* 지점까지 대각선으로 하는 사각형을 표시합니다.

예) Box 30,30,150,150;

BoxFill

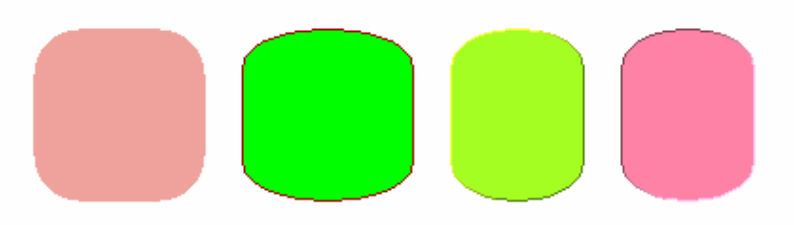
BoxFill *[x1,y1,x2,y2] [,border_colorstyle] [,border_dotsize] [,border_depth]*
x1,y1 지점에서 *x2,y2* 지점까지 대각선으로 하는 채워진 사각형을 표시합니다
*border_colorstyle*을 지정할 경우 지정한 색상으로 **테두리**를 두릅니다.(예: 적색 - &H0000FF)
특정 색상 대신 $push(눌린 모양), $pop(튀어 나온 모양)의 3D 효과를 지정할 수 있습니다.
*border_dotsize*를 지정할 경우 테두리의 두께를 지정할 수 있으며(생략시 1),
*border_depth*로 3D 효과의 입체감 정도(1~100)를 지정할 수 있습니다(생략시 50).
*x1,y1,x2,y2*를 생략할 경우 전체 화면 영역이 됩니다.(0,0,799,479)
예) BoxFill 30,30,150,150;
BoxFill 30,30,150,150,&H00FF40,3;
BoxFill 30,30,150,150,$pop,5,60;

RoundBox



RoundBox *x1, y1, x2, y2, corner_width [, corner_height]*
x1,y1 지점에서 *x2,y2* 지점까지 대각선으로 하는 모서리가 둥근 사각형을 표시합니다. 둥근 모서리의 폭은 *corner_width*로, 높이는 *corner_height*로 지정할 수 있으며 *corner_height*를 생략할 경우 *corner_width*를 *corner_height*로 간주합니다.
예) RoundBox 30,30,150,150,4;
RoundBox 30,30,150,150,4,7;

RoundBoxFill



RoundBoxFill *x1, y1, x2, y2, corner_width [, corner_height] [,border_colorstyle] [,border_depth]*

x1,y1 지점에서 *x2,y2* 지점까지 대각선으로 하는 모서리가 둥근 채워진 사각형을 표시합니다.

*corner_height*를 생략할 경우 *corner_width*를 *corner_height*로 간주합니다.

*border_colorstyle*을 지정할 경우 지정한 색상으로 굵기 1의 테두리를 두릅니다. (예: 청색 - &HFF0000)

색상 대신 $push(눌린 모양), $pop(튀어 나온 모양)의 3D 효과를 지정할 수 있습니다.

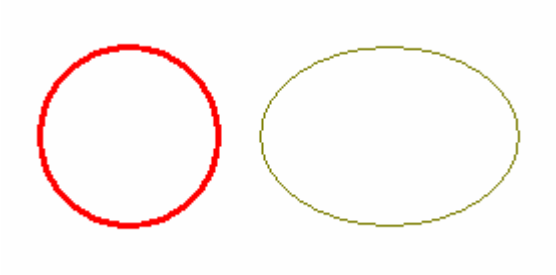
*border_depth*로 입체감의 정도(1~100)를 지정할 수 있습니다(생략시 50).

예) RoundBoxFill 30,30,150,150,4;

RoundBoxFill 30,30,150,150,4,5,&HDFFF00;

RoundBoxFill 30,30,150,150,4,5,$pop,60;

Circle

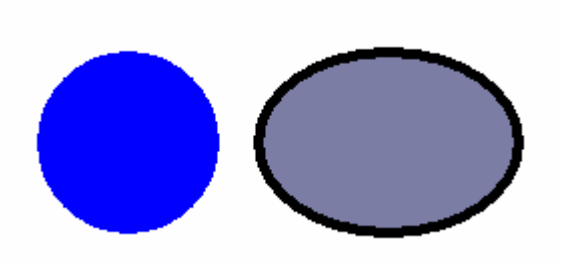


Circle *x1, y1, x2, y2*

x1,y1 지점에서 *x2,y2* 지점까지 대각선으로 하는 사각 영역에 채워지는 원을 그립니다.

예) Circle 30,30,150,150;

CircleFill



CircleFill *x1, y1, x2, y2 [,border_color] [,border_dotsize]*
x1,y1 지점에서 *x2,y2* 지점까지 대각선으로 하는 사각 영역에 채워진 원을 그립니다.
*border_color*를 지정할 경우 지정한 색상으로 테두리를 그립니다.
*border_dotsize*로 테두리의 두께를 지정할 수 있으며 생략할 경우 두께는 1입니다.
예) CircleFill 30,30,150,150;
CircleFill 30,30,150,150,&HA0B030,5;

Arc

Arc *x1, y1, x2, y2, start_angle, end_angle*
x1,y1 지점에서 x2,y2 지점까지 대각선으로 하는 사각 영역에 채워지는 호를 그립니다.
시작각은 *start_angle*이고 종료각은 *end_angle*이며 각도는 0~360의 값을 가집니다.
DotSize는 1로 지정되어 있어야 합니다.
예) Arc 30,30,150,150,60,320;

Bezier

Bezier *x1, y1, x2, y2, x3, y3, x4, y4*
4개의 지점을 참조로 하여 베지어 곡선을 그립니다.
DotSize는 1로 지정되어 있어야 합니다.
예) Bezier 30,30,50,60,120,80,100,130;

Print

Print *string [,x ,y]*

텍스트를 출력합니다. string은 어퍼스트로피(‘)로 앞뒤를 감싸야 합니다.

텍스트의 최대 길이는 어퍼스트로피 포함 1024 byte입니다.

뒤의 *x, y*를 붙여줄 경우 그 지점을 왼쪽 상단으로 일시적으로 간주하여 출력하며

*x, y*가 없을 경우는 TextPos에서 미리 지정한 위치에 출력합니다.

예) Print ‘Hello! Comfile Technology’;

Print ‘Hello! Comfile Technology’,50,50;

SetVar #value,’Hello’; Print #value,50,50;

* 다국어 출력

IntelliLCD는 사용자가 폰트만 설치하면 전세계의 모든 언어를 표현할 수 있습니다.(유니코드 사용)

자세한 내용은 <제 9 장 인텔리 LCD Unicode Generator (다국어 지원)>을 참조하십시오

예) Print 'U::66 5B 1F 75';

Print 'U::665B1F75';

Print 'U::66,5B,1F,75';

Print 'U::66-5B-1F-75';

PrintChr

PrintChr *asc_value [,x ,y]*

한 개의 문자를 출력합니다. *asc_alue*는 아스키 코드 값으로서 최저값은 32이고 최대값은 255입니다.

뒤의 *x, y*를 붙여줄 경우 그 지점을 왼쪽 상단으로 일시적으로 간주하여 출력하며

x, y가 없을 경우는 TextPos에서 미리 지정한 위치에 출력합니다.

예) PrintChr 96;

PrintChr 65,200,100;

TextPos

TextPos *x1, y1 [,x2 ,y2][,align]*
이후 표시될 텍스트의 위치를 지정합니다.
*x1*과 *y1*만 지정할 경우 그 지점을 새로운 텍스트의 왼쪽 상단으로 지정하고
x2, *y2*까지 지정할 경우 사각 영역의 내부에 텍스트를 정렬합니다.
*align*은 텍스트를 어느 쪽으로 정렬할지를 결정하며 $left, $center, $right 중의 하나를 지정할 수 있습니다. 처음 상태는 $center이며, 생략하면 가장 최근에 지정했던 설정을 따릅니다.
Print문을 사용하기 전에 해야만 효과가 있습니다. 이미 출력된 텍스트의 위치는 변경하지 못합니다.
예) TextPos 30,30;
예) TextPos 30,30,200,70;

TextFont

TextFont *[font_size, font_face, font_bold, font_italic, font_underlined]*
이후 표시할 텍스트의 폰트를 지정합니다. 크기, 글자체, 굵게, 이탤릭, 밑줄의 순입니다.
*font_face*는 어퍼스트로피(‘)로 앞뒤를 감싸야 하며 나머지는 1(true)/0(false)의 값을 가집니다.
변경하지 않고 유지하고 싶은 속성은 생략할 수 있습니다.
사용할 수 있는 기본 폰트 종류 외에 SD 카드를 이용하여 폰트를 추가할 수 있습니다.
Print문을 사용하기 전에 해야만 효과가 있습니다. 이미 출력된 텍스트의 폰트는 변경하지 못합니다.
예) TextFont 12,‘Courier New’;
TextFont ,,,,1;
TextFont ,’Tahoma’,,1;
최초구입시 사용할수 있는 폰트의 종류는 다음과 같습니다.
- 굴림체, 굴림, 돋움체, 돋움, 바탕체, 바탕, 궁서체, 궁서, Tahoma, Courier New

CtrlFont

CtrlFont *[font_size, font_face, font_bold, font_italic, font_underlined]*
이후에 표시할 컨트롤 내부에서 쓰는 텍스트의 폰트를 지정합니다.
파라미터는 TextFont의 경우와 동일합니다.
컨트롤을 생성하기 전에 해야만 효과가 있습니다. 이미 생성된 컨트롤의 폰트는 변경하지 못합니다.
(컨트롤이란? 화면상에 생성되는 독립기능을 갖춘 객체를 의미합니다. 버튼, 체크박스등)

TextColor

TextColor *[font_color, back_color]*

텍스트의 폰트 색상과 배경색을 지정합니다. 색상은 16진수 형태로 써야 하며 반드시 &H로 시작하고 뒤에 6자리가 와야 합니다. 배경색이 투명일 경우는 $transparent로 지정합니다.

Print문을 사용하기 전에 해야만 효과가 있습니다. 이미 출력된 텍스트의 색상은 변경하지 못합니다.

예) TextColor &HFF0000;
TextColor ,$transparent;
TextColor &H000000,&HA0A0A0;

CtrlColor

CtrlColor *[font_color, back_color]*

컨트롤 내부의 폰트 색상과 배경색을 지정합니다. 색상은 16진수 형태로 써야 하며 반드시 &H로 시작하고 뒤에 6자리가 와야 합니다. 투명색은 지정할 수 없습니다.

컨트롤을 생성하기 전에 해야만 효과가 있습니다. 이미 생성된 컨트롤의 색상을 변경하려면 SetCtrl 명령을 사용하십시오.

font_color가 적용되는 컨트롤들 : Button, Checkbox, Static

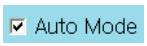
back_color가 적용되는 컨트롤들 : Button, Checkbox, Static, XSlider, YSlider

예) CtrlColor &HFF0000;
CtrlColor &H000000,&H0000FF;
CtrlColor ,&H0000FF;

Button

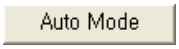
Button *id, x1, y1, x2, y2 [,caption] [,chattering_delay]*
x1,y1 지점에서 *x2,y2* 지점까지 대각선으로 하는 영역에 정해진 *id*를 가진 버튼을 생성합니다.
*id*는 1~255의 숫자이며 다른 컨트롤과 중복될 수 없습니다.
버튼에 처음 표시될 내용은 *caption*으로 지정 가능하며 어퍼스트로피(‘)로 앞뒤를 감싸야 합니다.
부정확한 터치에 의한 원치 않는 연속적인 클릭(채터링 현상)을 방지하기 원한다면 최소 지연 시간(1/1000초 단위)을 *chattering_delay*를 통하여 지정합니다. 생략하면 그 값은 0입니다.
채터링 방지 기능은 이벤트 발생에 대해 적용되는 것이며 화면에 대하여 적용되는 것은 아닙니다.
또한 ButtonAutoRepeat 기능이 활성화되어 있을 때에는 효력이 없습니다.
<관련 이벤트: $on_click 참조>
예) Button 1,40,40,120,60,‘Start’;
Button 1,40,40,120,60,‘Start’,800;
2줄 이상의 캡션도 지정 가능하며 행바꿈은 ‘|’문자로 나타냅니다.
예) Button 1,40,40,120,100,‘지금|시작’;

CheckBox



CheckBox *id, x1, y1, x2, y2 [,caption]*
x1,y1 지점에서 *x2,y2* 지점까지 대각선으로 하는 영역에 정해진 *id*를 가진 체크박스를 생성합니다.
*id*는 1~255의 숫자이며 다른 컨트롤과 중복되는 숫자를 사용할 수 없습니다.
체크박스에 처음 표시될 내용은 *caption*이며 어퍼스트로피(‘)로 앞뒤를 감싸야 합니다.
예) CtrlColor ,&H0000FF;Checkbox 2,100,315,170,135,’Auto Mode’;
<관련 이벤트: $on_check 참조>

Toggle



Toggle *id, x1, y1, x2, y2 [,caption]*
x1, y1 지점에서 *x2, y2* 지점까지 대각선으로 하는 영역에 정해진 *id*를 가진 토글 버튼을 생성합니다.
토글 버튼이란 한 번 눌렀다 떼면 눌려진 상태가 유지되고 다시 눌렀다 떼면 튀어나온 상태가 유지되는 버튼을 말합니다. 생김새는 버튼과 같고 **수행 동작과 이벤트는 체크박스와 같습니다.**
<관련 이벤트: $on_check 참조>
예) Toggle 1,40,40,120,60,‘Start’;

Static

Static *id, x1, y1, x2, y2 [,caption] [,align]*

x1,y1 지점에서 *x2,y2* 지점까지 대각선으로 하는 영역에 정해진 ID(id)를 가진 스태틱 컨트롤을 생성합니다.

*id*는 1~255의 숫자이며 다른 컨트롤과 중복되는 숫자를 사용할 수 없습니다.

*align*은 정렬 방식이며 $left,$center,$right 중 하나를 지정할 수 있습니다.

수직 정렬은 위쪽 방향으로 고정되어 있으며 변경할 수 없습니다.

예) CtrlColor &H000000,&HD0D0D0; Static 1,250,100,375,130,'COMFILE',$center;

XProgress

XProgress *id, x1, y1, x2, y2, max_range*

x1,y1 지점에서 *x2,y2* 지점까지 대각선으로 하는 영역에 특정한 ID(*id*)를 가진 가로 방향 프로그레스 컨트롤을 생성합니다. 진행 상황 등을 표시하기 위해 사용됩니다.

*id*는 최소 1 이상의 숫자이며 다른 컨트롤과 중복되는 숫자를 사용할 수 없습니다.

최대 수치(*max_range*)를 65535 이하의 값으로 지정해야 하며 실제 픽셀값이 아닌 **가상적인** 값입니다.

진행된 상황 값을 표현하기 위해서는 SetCtrl을 사용합니다.

우측으로 진행될 수록 높은 값을 표현합니다.

예) XProgress 4,250,100,315,120,10000;

예) SetCtrl 4,$position,5000;

YProgress

YProgress *id, x1, y1, x2, y2, max_range*

XProgress와 같으며 세로 방향으로 표시된다는 점만 다릅니다.

상단으로 진행될수록 높은 값을 표현합니다.

예) YProgress 4,250,100,275,180,10000;

XSlider

XSlider *id, x1, y1, x2, y2, max_range [,highlight_color] [,track_width] [,grip_color]*
x1,y1 지점에서 *x2,y2* 지점까지 대각선으로 하는 영역에 정해진 id를 가진 가로 방향 슬라이더 컨트롤을 생성합니다. *id*는 1~255의 숫자이며 다른 컨트롤과 중복되는 숫자를 사용할 수 없습니다.
최대 수치(*max_range*)를 65535 이하의 값으로 지정해야 하며 실제 픽셀값이 아닌 **가상적인** 값입니다.
우측으로 옮길수록 높은 값을 표현합니다.
특별히 *highlight_color*를 지정하면 진행된 부분을 해당 색상으로 강조하여 그립니다.
*track_width*로 라인의 굵기를 지정할 수 있습니다. (생략시 4)
*grip_color*로 그립 버튼의 색상을 지정할 수 있습니다. (생략시 회색)
예) CtrlColor ,&HD0D0D0; XSlider 4,250,100,315,120,10000;
<관련 이벤트: $on_position 참조>

YSlider

YSlider *id, x1, y1, x2, y2, max_range [,highlight_color] [,track_width] [,grip_color]*
XSlider와 같으며 세로 방향으로 표시된다는 점만 다릅니다.
상단으로 올릴수록 높은 값을 표현합니다.
예) CtrlColor ,&H0000FF; YSlider 4,250,100,275,180,10000,&H4040D0;

FCreate

FCreate *file_name [,content]*
특정 파일을 생성합니다. 이미 같은 파일명의 파일이 있을 경우 대체합니다.
*file_name*은 SD카드의 루트 디렉터리(폴더명 'storage card')에 존재하는 파일을 가리킵니다.
내용(*content*)을 지정할 경우 그 내용이 기록되어지고 지정하지 않을 경우 파일 크기가 0인 파일로 생성합니다.
file_name과 content는 어퍼스트로피(')로 앞뒤를 감싸야 합니다.
예) FCreate 'record.dat','279432';
FCreate 'record.dat',#string2;
SetVar #data,'hello'; FCreate 'record.dat',#data;

FAppend

FAppend *file_name ,content*
특정 경로의 파일 뒷부분에 내용을 추가합니다. 파일이 없을 경우 자동으로 새로 만듭니다.
*file_name*은 SD카드의 루트 디렉터리(폴더명 'storage card')에 존재하는 파일을 가리킵니다.
*file_name*과 *content*는 어퍼스트로피(')로 앞뒤를 감싸야 합니다.
content 대신에 $newline이라고 쓸 경우 행바꿈 (CR-LF)이 기록됩니다.
예) FAppend 'record.dat','279432';
FAppend 'record.dat',$newline;
SetVar #data,'hello'; FAppend 'record.dat',#data;

FDelete

FDelete *file_name*
특정한 파일을 삭제합니다.
*file_name*은 SD카드의 루트 디렉터리(폴더명 'storage card')에 존재하는 파일을 가리킵니다.
*file_name*은 어퍼스트로피(')로 앞뒤를 감싸야 합니다.
예) FDelete 'record.dat';

FRead

FRead *file_name [,start_pos] [,max_bytes] [,query_id]*
특정한 파일의 내용을 읽습니다.
이 커맨드가 수행되면 $on_file_read 이벤트가 바로 발생되어 파일 내용을 조회할 수 있습니다.
*file_name*은 SD카드의 루트 디렉터리(폴더명 'storage card')에 존재하는 파일을 가리킵니다.
*start_pos*는 해당 파일의 몇 번째 바이트를 서두로 해서 읽을지를 나타내는 위치값(생략시 0)이며
*max_bytes*는 결과적으로 읽기를 원하는 데이터의 최대 바이트수(생략시 끝까지 읽음)를나타냅니다.
*query_id*는 0~255의 값이며 나중에 $on_file_read 이벤트와 일치하는지 확실히 확인하기 위한값이며 생략시 0으로 할당됩니다.
예) FRead 'record.bin';
FRead 'record.txt',100,20;
FRead 'record.txt',,50;

FCopy

FCopy *existing_file_name, new_file_name*
특정 파일을 복사합니다. 이미 같은 파일명의 파일이 있을 경우 대체합니다.
*existing_file_name*과 *new_file_name*은 어퍼스트로피(')로 앞뒤를 감싸야 하며 SD카드의 루트 디렉터리(폴더명 'storage card')에 존재하는 파일을 가리킵니다.
예) FCopy 'file1.bin','file2.bin';

FReadToVar

FReadToVar *file_name, var_key, [,start_pos] [,max_bytes]*
특정한 파일의 내용을 읽어서 메모리 상의 변수에 기억합니다.
이 커맨드는 이벤트를 발생시키지 않습니다.
*var_key*는 변수명을 뜻하며 #문자로 시작해야 합니다.
*file_name*은 SD카드의 루트 디렉터리(폴더명 'storage card')에 존재하는 파일을 가리킵니다.
*start_pos*는 해당 파일의 몇 번째 바이트를 서두로 해서 읽을지를 나타내는 위치값(생략시 0)이며
*max_bytes*는 결과적으로 읽기를 원하는 데이터의 최대 바이트수(생략시 끝까지 읽음)를 나타냅니다.
예) FReadToVar 'record.bin',#string2;
FReadToVar 'record.txt',#string2,100,20;
FReadToVar 'record.txt',#string,,50

SetVar

SetVar *var_key, value*

Set *var_key, value*

메모리의 특정 변수에 값을 저장합니다.

저장될 값은 문자열일 수도 있고 숫자(음수도 지원)일 수도 있습니다.

재부팅하면 값이 사라집니다.

*var_key*는 변수명으로서 #문자로 시작해야 하며 대소문자를 가리지 않습니다.

*value*는 어퍼스트로피(‘)로 앞뒤를 감쌀 경우 문자열로 간주하고 그렇지 않으면 숫자로 간주합니다.

*value*가 다른 변수명이 될 수도 있습니다.

이렇게 저장된 변수값은 IntelliLCD의 모든 커맨드의 모든 파라미터에서 사용 가능합니다.

예) SetVar #a,’hello’; print #a;

SetVar #b,#a;

SetVar #left,30; line #left,30,200,200;

DelVar

DelVar *var_key*

특정 변수를 메모리에서 삭제합니다.

*var_key*는 변수명으로서 #문자로 시작해야 합니다.

예) DelVar #value;

DelVarAll

DelVarAll

모든 변수를 메모리에서 삭제합니다.

예) DelVarAll;

Add

Add *var_key, value;*
변수에 특정한 값을 더합니다.
더할 값은 정수여야하며 다른 변수명을 지정할 수도 있습니다.
var_key는 변수명을 뜻하며 #문자로 시작해야 합니다.
예) Add #value,3;
Add #value,#k;

Sub

Sub *var_key, value;*
변수에 특정한 값을 뺍니다.
뺄 값은 정수여야하며 다른 변수명을 지정할 수도 있습니다.
var_key는 변수명을 뜻하며 #문자로 시작해야 합니다.
예) Sub #value,3;
Sub #value,#k;

Mul

Mul *var_key, value;*
변수에 특정한 값을 곱합니다.
곱할 값은 정수여야하며 다른 변수명을 지정할 수도 있습니다.
var_key는 변수명을 뜻하며 #문자로 시작해야 합니다.
예) Mul #value,3;
Mul #value,#k;

Div

Div *var_key, value;*
변수에 특정한 값을 나눕니다.
나눌 값은 정수여야하며 다른 변수명을 지정할 수도 있습니다.
var_key는 변수명을 뜻하며 #문자로 시작해야 합니다.
예) Div #value,3;
Div #value,#k;

Mod

Mod *var_key, value;*
변수에 특정한 값을 나눈 나머지를 세팅합니다.
나눌 값은 정수여야하며 다른 변수명을 지정할 수도 있습니다.
var_key는 변수명을 뜻하며 #문자로 시작해야 합니다.
예) Mod #value,3;

Attach

Attach *var_key, string*
변수값 뒤에 문자열을 덧붙입니다.
예) SetVar #a,'pine'; Attach #a,'apple'; print #a;

AttachChr

AttachChr *var_key, asc_value*
한 개의 문자를 변수값이 덧붙입니다.
*asc_alue*는 아스키 코드 값으로서 최저값은 32이고 최대값은 255입니다.
예) SetVar #a,'appl'; Attach #a,69; print #a;

Sound

Sound *file_name*
이 명령은 사운드지원기능이 있는 모델에서만 사용가능합니다.(Cuwin의 iTL모드에서는 지원)
음성 파일을 재생하여 소리를 냅니다. WAV 포맷을 지원합니다.
*file_name*은 SD카드의 루트 디렉터리(폴더명 'storage card')에 존재하는 파일을 가리킵니다.
*file_name*은 어퍼스트로피(')로 앞뒤를 감싸야 합니다.
예) Sound 'ring.wav';

StopSound

StopSound;
이 명령은 사운드지원기능이 있는 모델에서만 사용가능합니다.(Cuwin의 iTL모드에서는 지원)
현재 연주 중인 사운드를 중지합니다.
예) StopSound;

GetDate

GetDate

오늘 날짜를 이벤트로 리턴합니다.(년,월,일).

RTC 배터리 기능이 없는 모델에서는 재부팅후 시간 저장이 안되는 제약이 있습니다.

(Cuwin의 iTL모드에서는 지원)

이벤트 설명 부분을 참조하세요.

예) GetDate;

GetTime

GetTime

현재 시각을 이벤트로 리턴합니다(시,분,초)

RTC 배터리 기능이 없는 모델에서는 재부팅후 시간 저장이 안되는 제약이 있습니다.

(Cuwin의 iTL모드에서는 지원)

이벤트 설명 부분을 참조하세요.

예) GetTime;

SetClock

SetClock *year, month, day, hour, minute, second*

현재 시각을 수정합니다.

RTC 배터리 기능이 없는 모델에서는 재부팅후 시간 저장이 안되는 제약이 있습니다.

(Cuwin의 iTL모드에서는 지원)

연, 월, 일, 시, 분, 초 순서로 지정하며 hour는 0~23까지입니다.

예) SetClock 2007,1,1,12,30,33;

DrawImage

DrawImage *file_name [,x ,y] [,transparent_color]*
그림 파일을 화면에 출력합니다. BMP 이미지 포맷을 지원합니다.
*file_name*은 SD카드의 루트 디렉터리(폴더명 'storage card')에 존재하는 파일을 가리킵니다.
*file_name*은 어퍼스트로피(')로 앞뒤를 감싸야 합니다
x,y 좌표를 지정할 경우 그 지점을 좌측 상단으로 위치를 잡아서 출력합니다.
생략할 경우 0,0 지점에다 출력합니다.
특별히 *transparent_color* 값을 지정하면 그 색상으로 표현된 픽셀들은 **투명**하게 보이게 합니다.
참고로, 같은 이미지의 반복 출력을 위해서는 CaptureFile, Paste 커맨드가 속도면에서 유리합니다.
예) DrawImage 'logo.bmp';
DrawImage 'logo.bmp',50,50;
DrawImage '.₩mydir₩logo.bmp',50,50,&HFF00FF;

CaptureScreen

CaptureScreen *[x1, y1, x2, y2, bitmap_id]*
화면 상의 *x1,y1* 지점에서 *x2,y2* 지점까지 대각선으로 하는 사각 영역을 캡쳐하여 가상 공간(메모리)에 저장합니다. *bitmap_id* 는 0~9이며 생략시 0으로 간주합니다. 영역을 생략할 경우 전체 화면 영역으로 간주합니다.
컨트롤들은 캡쳐하지 않고 순수한 그래픽만을 캡쳐합니다.
예) CaptureScreen;
CaptureScreen 0,0,200,200;
CaptureScreen ,,,,9;

CaptureFile

CaptureFile *file_name, [x1, y1, x2, y2, bitmap_id]*
SD카드에 있는 어떤 이미지 파일(현재는 BMP파일만 지원)의 이미지 상의 x1,y1 지점에서 x2,y2 지점까지 대각선으로 하는 사각 영역을 캡쳐하여 가상 공간(메모리)에 저장합니다. *bitmap_id* 는 0~9이며 생략시 0으로 간주합니다. 영역을 생략할 경우 BMP 파일 전체 이미지를 캡쳐합니다. CaptureScreen 명령에 비해 수행 속도가 느립니다.
SaveImage 커맨드와 반대되는 커맨드입니다.
예) CaptureFile 'image.bmp;
CaptureFile 'image.bmp',,,,,9;
CaptureFile 'image.bmp',0,0,199,199,9;

Paste

Paste *[put_x, put_y, bitmap_id, crop_x1, crop_y1, crop_x2, crop_y2] [,transparent_color]*
CaptureScreen이나 CaptureFile 명령에 의해 가상 공간에 저장된 이미지를 put_x, put_y 지점을 좌측 상단으로 하여 빠른 속도로 화면에 출력합니다. *bitmap_id* 는 0~9이며 생략시 0으로 간주합니다. 좌표를 생략한 경우 가장 최근에 CaptureScreen 명령시 사용했던 좌측 상단 지점으로 간주합니다. (처음 상태에선 0,0)
크롭 영역 파라미터(crop_x1, crop_y1, crop_x2, crop_y2)를 생략하면 저장되었던 이미지 전체를 그대로 출력하고, 특별히 지정하면 그 영역만큼 잘라내어 출력합니다.
특별히 transparent_color 값을 지정하면 그 색상으로 표현된 픽셀들은 **투명**하게 보이게 합니다.
예) Paste;
Paste 200,200,0;
Paste ,,0,40,40,300,300;

SaveImage

SaveImage *file_name [,bitmap_id]*
가상 공간(메모리)에 저장되어 있던 이미지를 BMP 파일로 기록합니다.
CaptureFile 커맨드와 반대되는 커맨드입니다.
file_name은 SD카드의 루트 디렉터리(폴더명 'storage card')에 존재하는 파일을 가리킵니다.
file_name은 어퍼스트로피(')로 앞뒤를 감싸야 합니다.
이미 존재하는 파일이 있다면 강제로 교체합니다.
참고로, 가상 메모리 이미지는 CaptureScreen이나 CaptureFile 커맨드에 의해 생성할 수 있습니다.
예) SaveImage 'image1.bmp';
SaveImage 'image1.bmp',3;

GetCtrl

GetCtrl *id, event*
컨트롤의 속성을 조회합니다.(이벤트를 요청합니다.)

event	value – 이벤트로서 얻어짐. <이벤트 각 항목 참조>
$on_check	Checkbox나 Toggle의 체크 상태.
$on_position 또는 $position	XSlider와 YSlider 컨트롤의 위치값.

예) GetCtrl 1,$on_check;
GetCtrl 2,$on_position;

SetCtrl

SetCtrl *id, field, value*

컨트롤의 속성에 변경을 가합니다.

field	value
$caption	Static, Checkbox, Button 컨트롤에 표시될 텍스트로서 어퍼스트로피(‘)로 감싼 문자열.
$position 또는 $on_position	XProgress, YProgress, XSlider, YSlider 컨트롤의 진행값. 실제 픽셀값이 아닌 **가상적인** 값입니다.
$fontcolor	Button, Checkbox, Static 컨트롤의 폰트 색
$backcolor	Button, Checkbox, Static, XSlider, YSlider 컨트롤의 배경색.
$align	Static 컨트롤의 정렬방식. $left, $center, $right 중 하나.
$on_check	Checkbox나 Toggle의 체크 상태. 1이면 체크, 0이면 해제

예) SetCtrl 1,$caption,’Next’;
SetCtrl 1,$position,550;
SetCtrl 3,$backcolor,&H000000;
SetCtrl 5,$align,$left;

MoveCtrl

MoveCtrl *id, x1, y1 [,x2, y2]*

컨트롤의 화면 상의 위치를 조정합니다.

*x2, y2*를 생략했을 경우 *x1, y1* 지점을 좌측 상단이 되게 이동시키고

*x2, y2*가 지정되었을 경우는 *x1, y1* 지점에서 *x2, y2* 지점까지 대각선으로 하는 사각 영역으로 이동시킵니다.

예) MoveCtrl 1,200,200;
MoveCtrl 1,200,200,300,250;

ShowCtrl

ShowCtrl *id, show*

특정 컨트롤을 즉시 보이게 하거나 보이지 않게 합니다.

show값이 1이면 보여주며 0이면 보이지 않게 합니다.

단순히 화면상의 표시 여부만을 결정하므로 감추었다가 다시 나타내도 모든 속성은 보존되며

감추어져 있을 동안에 같은 ID로 다른 새로운 컨트롤을 생성할 수 없습니다.

예) ShowCtrl 5,0;

EnableCtrl

EnableCtrl *id, enable*
특정 컨트롤을 조작 가능하게 하거나 가능하지 않게 합니다.
*enable*값이 1이면 가능이며 0이면 불능입니다..
불능 상태일 때는 옅은 회색으로 바뀝니다.
예) EnableCtrl 5,0;

DelCtrl

DelCtrl *id*
특정 컨트롤을 즉시 완전히 삭제하며 등록된 ID까지 취소합니다.
존재하지 않는 컨트롤을 삭제하려 해도 에러를 발생하지는 않습니다.
예) DelCtrl 5;Button 5,30,30,200,200;

ShowCursor

ShowCursor
마우스 커서를 표시합니다. 단, 마우스(USB 방식)가 연결되어 있지 않으면 보이지 않습니다.
예) ShowCursor;

HideCursor

HideCursor
커서를 보이지 않도록 설정합니다. 전원을 켰을 때는 항상 보이지 않도록 설정되어 있습니다.
예) HideCursor;

EnableDemo

EnableDemo *[enable]*
전원 On시에 데모를 보여줄 것인지 말 것인지를 결정합니다.
enable 값이 1이면 보여주며 0이면 보여주지 않습니다.
즉시 나타나는 화면의 변화는 없습니다.
이 설정은 전원을 꺼도 지속됩니다.
예) EnableDemo;
EnableDemo 0;

SetRecvMode

SetRecvMode *mode*
이벤트를 어떤 형식으로 받을지 결정합니다. 변경 사항은 즉시 적용됩니다.
*mode*가 1일 경우 ASCII 형식이며 2일 경우 Binary 형식으로 수신합니다.
화면에 즉시 나타나는 변화는 없습니다.
출고시 설정은 ASCII 형식이지만 **변경하면 변경된 설정이 전원을 꺼도 지속됩니다**.
예) SetRecvMode 1;
SetRecvMode 2;

ButtonAutoRepeat

ButtonAutoRepeat *id [,enable, wait_time, repeat_time]*
버튼을 일정 시간 누른 상태로 있으면 자동으로 일정 시간 간격으로 누름과 뗌을 반복한 것처럼 효과를 낼 수 있게 해주는 커맨드입니다. enable이 1이면 가능이고 0이면 불가능입니다.
생략하면 1이며 부팅 후 기본 상태는 0입니다.
wait_time은 처음에 얼마나 오래 누르고 있어야 반복 기능을 시작할 지 결정하는 값(1/1000초 단위)이며 생략시는 1200으로 할당됩니다.
repeat_time은 반복 기능이 시작되고 나서 자동으로 눌려지는 주기(1/1000초 단위)를 결정하며 생략시는 300으로 할당됩니다.
enable = 1로 하여 이 함수를 호출하면 Button 생성시에 지정했던 chattering_delay는 효력이 없어집니다.
예) ButtonAutoRepeat 1;
ButtonAutoRepeat 12,1,1000,300;

TouchAutoRepeat

TouchAutoRepeat *[enable, wait_time, repeat_time]*
컨트롤들이 차지하지 않는 일반 터치패드 영역을 일정 시간 누른 상태로 있으면 자동으로 일정 시간 간격으로 누름과 뗌을 반복한 것처럼 효과를 낼 수 있게 해주는 커맨드입니다.
enable이 1이면 가능이고 0이면 불가능입니다. 생략하면 1이며 최초 기본 상태는 0입니다.
wait_time은 처음에 얼마나 오래 누르고 있어야 반복 기능을 시작할 지 결정하는 값(1/1000초 단위)이며 생략시는 1200으로 할당됩니다.
repeat_time은 반복 기능이 시작되고 나서 자동으로 눌려지는 주기(1/1000초 단위)를 결정하며 생략시는 300으로 할당됩니다.
예) TouchAutoRepeat;
TouchAutoRepeat 1,1000,300;
TouchAutoRepeat 0;

Dump

Dump *[enable]*

디버깅을 하기 위해 커맨드 송수신 내용을 파일에 기록할지 결정합니다.

enable 이 1일 경우 <사용함>이며 0일 경우 <사용 안 함>입니다.

Storage Card/ 폴더의 comfile_dump_rx.txt(수신 데이터)와 comfile_dump_tx.txt(송신 데이터) 파일에 기록하므로 SD 카드가 장착되어 있어야 합니다.

최초상태는 기본적으로 항상 <사용 안함> 상태입니다.

Dump를 하면 시스템 속도가 저하되므로, 커맨드 내용을 분석해서 디버깅을 해야 하는 특수한 경우에만 사용하십시오.

예) Dump;

Dump 0;

SetBufferWarning

SetBufferWarning *[bytes]*

LCD쪽으로 들어오는 데이터 버퍼가 정해진 분량 이상으로 쌓였을 경우 $on_buffer_warning이 발생하는데, 그 기준이 되는 용량을 바이트 단위로 설정합니다. 즉 얼마만큼 버퍼가 늘어났을 때 이벤트를 발생시킬 것인지 결정합니다. 최대값은 1M Byte (1048576 byte) 이며 최초상태 즉, 초기값은 0.5M byte (524288 byte) 입니다.

예외적으로 버퍼가 1M Byte를 초과하여 쌓였을 경우 넘치는 패킷은 버리며 화면에 'Buffer Overflow'라는 메시지가 잠시 출력되고 $on_buffer_overflow 이벤트를 발생시킵니다.

예) SetBufferWarning 10000;

FileCmd

FileCmd *file_name [,condition]*
배치실행 커맨드입니다. IntelliLCD 커맨드를 Text 파일로 저장한뒤 이 명령을 실행하면 해당파일의 내용대로 화면에 표시됩니다.
예를들어 graphic1.txt 라는 파일의 내용이 다음과 같은 경우.....

```
TextColor &HFF0000;
RoundBoxFill 30,30,150,150,4,5,&HDFFF00;
Line 30,30,200,200;
```

FileCmd 명령을 수행하면 이 내용이 수행된 화면을 보실 수 있습니다. 파일의 확장자는 상관 없습니다.
예) FileCmd 'graphic1.txt';

만약, 특정한 조건이 충족되었을 때만 파일커맨드가 실행되도록 하려면 두번째 인자로 조건을 지정할 수 있습니다.
조건문 형식 : 변수명 또는 상수 + 조건 연산자 + 변수명 또는 상수
사용할 수 있는 조건 연산자는 다음과 같습니다.

A와 B가 같다 : A=B	A가 B보다 크다 : A>B
A가 B보다 작다 : A<B	A가 B보다 크거나 같다 : A>=B
A가 B보다 작거나 같다 : A<=B	A와 B가 같지 않다 : A<>B

조건 연산자는 and 나 or 문으로 연결할 수 있습니다.
and 나 or 를 혼용할 수 없습니다.
맞음) #sys_touch_x>100
맞음) #a<3 **and** #b=4 **and** #c<>#k
틀림) #a<3 **and** #b=4 **or** #c<>#k

추가설명)
복잡한 배경화면을 화면상에 그리는 경우를 생각해 보겠습니다. 큐블록 (또는 다른 마이컴)에서 RS232 데이터로 대략, 1000 바이트정도의 데이터를 송신해야하는 상황이 생기게 됩니다.
해당 데이터를 큐블록 (또는 마이컴)에서 가지고 있으려면, 메모리 공간도 필요하고, 데이터를 RS232로 송신하는 불편함이 있습니다.
이 때, FileCmd 명령을 사용하면 편리합니다. 배경화면을 그리는 데 필요한 커맨드를 하나의 텍스트파일에 저장한뒤, SD카드에 카피해놓고, FileCmd명령을 실행하면, 해당 내용을 커맨드로 인식해 그대로 화면에 표시됩니다.
이렇게하면 큐블록(또는 마이컴)의 메모리도 절약하게되고, 통신량도 줄어들게 됩니다.

VarCmd

VarCmd *var_key [,condition]*
배치실행 커맨드입니다. 변수에 있는 값을 커맨드로 인식하여 실행합니다.(변수 커맨드)
FileCmd와 유사하나 파일에 있는 내용이 아닌 변수에 담긴 내용을 실행하므로 속도가 빠릅니다.
FileCmd를 사용할 때 반복적이고 속도가 요구되는 작업을 할 경우 FReadToVar와 조합하여 사용하면 유용합니다.
예) SetVar #cmd,'Clear;'; VarCmd #cmd;
FReadToVar 'cmd.txt',#cmd; VarCmd #cmd;
만약, 특정한 조건이 충족되었을 때만 파일커맨드가 실행되도록 하려면 두번째 인자로 조건을 지정할 수 있습니다. 조건을 지정하는 자세한 방법은 FileCmd 항목을 참조하세요.

FileEventOn *event, file_name, [condition]*

(고급 기능입니다)
특정 이벤트가 발생했을 때 파일 커맨드가 자동으로 실행되도록 설정합니다.
(FileCmd를 참고하세요.)
특정한 조건이 충족되었을 때만 파일커맨드가 실행되도록 하려면 세번째 인자로 조건을 지정할 수 있습니다. 조건을 지정하는 자세한 방법은 FileCmd 항목을 참조하세요.

이 커맨드를 실행해도 시리얼 통신 이벤트는 예전대로 발생하므로 시리얼 이벤트가 발생하기를 원치 않으면 LockEvent 커맨드를 사용하면 됩니다.

예) FileEventOn $on_click,'process1.txt';
예) FileEventOn $on_touch,'process2.txt',#sys_down=1;
예) FileEventOn $on_touch,'process2.txt',#sys_down=1 and #sys_touch_x<#var2;

FileEventOff *event*

특정 이벤트가 발생했을 때 파일 커맨드가 실행되도록 했던 것을 해지합니다.
예) FileEventOff $on_click;

FileEventClear

모든 이벤트에 대해 FileEventOff를 수행합니다.
예) FileEventClear;

ButtonStyleNormal

ButtonStyleNormal
Button과 Toggle의 외형적인 스타일을 <기본 상태>로 정의합니다.
기본 상태란 간단한 입체감을 가지며 눌려졌을 때와 떼어졌을 때 서로 다른 입체감을 가진 형태이며, 별도 설정을 안 했을 경우 부팅시 기본 상태입니다..
버튼을 생성하기 전에 해야만 효과가 있습니다. 이미 생성된 버튼의 스타일을 바꿀 수는 없습니다.
예) ButtonStyleNormal;

ButtonStyleSolid

ButtonStyleSolid *[push_font_color] [,push_back_color][,border_color]*
Button과 Toggle의 외형적인 스타일을 <단색 스타일>로 정의합니다.
입체감이 없이 눌려졌을 때와 떼어졌을 때 모두 단색으로 칠합니다.
눌려졌을 때의 폰트 색상과 배경색은 push_font_color와 push_back_color로 지정하며 파라미터 지정을 생략했을 경우는 CtrlColor에서 정의된 색상의 반전된 색상으로 칠합니다.
떼어졌을 때의 색상은 별도로 지정할 필요 없이 CtrlColor 명령의 색상 정의를 그대로 따릅니다.
border_color를 지정하면 테두리 색상이 지정되며 생략했을 경우는 테두리를 그리지 않습니다.
버튼을 생성하기 전에 해야만 효과가 있습니다. 이미 생성된 버튼의 스타일을 바꿀 수는 없습니다.
예) ButtonStyleSolid &H000000,&H00FFFF;
ButtonStyleSolid ,&H00FFFF,&H0000A0;

ButtonStyleImage

ButtonStyleImage *pop_file_name [,push_file_name] [,border_color]*
Button과 Toggle의 외형적인 스타일을 <이미지 스타일>로 정의합니다.
눌려졌을 때와 떼어졌을 때 SD카드에 있는 이미지 파일을 출력합니다.
pop_file_name은 떼었을 때 출력할 이미지 파일 이름이며
push_file_name은 눌렀을 때 출력할 이미지 파일 이름입니다.
push_file_name을 생략하면 pop_file_name의 이미지를 동일하게 가져다 사용합니다.
border_color를 지정하면 테두리 색상이 지정되며 생략했을 경우는 테두리를 그리지 않습니다.
버튼을 생성하기 전에 해야만 효과가 있습니다. 이미 생성된 버튼의 스타일을 바꿀 수는 없습니다.
예) ButtonStyleImage ‘button_image.bmp’;
ButtonStyleImage ‘pop_image.bmp’,’push_image.bmp’;

!@!@;

현재 해석중인 커맨드를 무시하고, 인텔리LCD를 초기화시키는 명령입니다.
인텔리LCD가 켜있는 상태에서 커맨드 송신이 잠시 끊어지거나, 리셋되는 경우, 앞서 해석중인 명령어를 미처 끝맺지 못하고, 새로운 커맨드를 받아들이는 상황이 생기게 됩니다. 이럴때, 인텔리 LCD는 새로운 커맨드를 이어 받게 되어, 화면에는 예상치못한 결과나 에러메세지가 표시될 수 있습니다.
이것을 막기위해, 리셋후 최초에 !@!@;가 오면, 해석중이던 커맨드를 무시하고 상황을 초기화합니다.

```
Const Device = CB280
Dim I As Integer
Opencom 1,9600,3,80,180
@ "!@!@;CLEAR;"
@ "TEXTFONT 64,'HY견고딕';"
@ "PRINT '아름다운 한국',10,50;"
```

#@#@;

!@!@;처럼 현재 커맨드 해석 상황(파싱시스템)을 초기화할 뿐 아니라, 수신 버퍼까지 즉시 제거합니다. 버퍼를 삭제하므로 기존에 쌓여서 아직 처리되지 못하고 실행 대기중이던 모든 커맨드들을 무시하고 없애므로, 이 명령은 즉시 반응합니다. 반면에 !@!@; 명령은 실행 대기중이던 커맨드들을 모두 지우지는 않기 때문에 그 커맨드들이 다 실행되어야 비로소 !@!@; 명령이 실행되므로 느리게 반응할 수도 있습니다.

주의 : Autorun의 내용이 실행중일 때 #@#@; 커맨드가 보내지면 버퍼가 지워지므로 아직 실행되지 않은 autorun의 뒷부분이 잘려서 끝까지 표시되지 않을 수 있습니다. (!@!@;는 그런 현상 없음)

SetDrawingSpeed

SetDrawingSpeed *speed*
그래픽 처리 속도를 0~200의 범위 내에서 지정합니다.
숫자가 높을수록 속도가 빠르지만 부자연스럽게 출력됩니다.
숫자가 낮을수록 속도가 느리지만 부드럽게 출력됩니다.
아무런 지정도 하지 않았을 경우의 기본값은 25이며 권장값입니다.
최고 속도와 최저 속도의 속도 차이는 최고 3배까지 차이 날 수 있습니다.

GetLcdState

GetLcdState
현재 LCD의 상태를 조회합니다. 결과는 $on_lcd_state 이벤트로 바로 응답됩니다.
주의: LCD가 꺼져있거나 부팅중일 때는 이벤트가 발생하지 않으며 1초에 두번 이상 발생하지 않습니다.
예) GetLcdState;

Backlight

Backlight *[enable]*

백라이트를 켜거나 끕니다.

iTL720에서는 지원하지 않습니다. (iTL710과 그 이후의 제품에서는 지원합니다.)

예) Backlight;

Backlight 0;

BacklightMode

BacklightMode *wakeup_mode*

백라이트가 꺼져 있을 때 자동으로 켜지는 옵션을 지정합니다.

wakeup_mode = 0 : 오직 Backlight 커맨드를 시리얼로 보내야만 다시 켜집니다.

wakeup_mode = 1 (디폴트) : LCD 화면에 터치를 누르기만해도 다시 켜집니다. (커맨드를 보내도 켜짐)

iTL720에서는 지원하지 않습니다. (iTL710과 그 이후의 제품에서는 지원합니다.)

예) BacklightMode 1;

LockEvent

LockEvent *[enable],[event],[condition]*

이벤트의 발생을 제한합니다.

enable : 1이면 제한 기능을 사용하며 0이면 제한을 해제시킵니다. 생략하면 1입니다.

event : 제한을 걸거나 해제하고 싶은 대상이 되는 이벤트명을 의미하며, 생략하면 모든 이벤트에 대해 적용합니다.

condition: 0이면 무조건적으로 항상 제한하며, 1이면 LCD측에서 커맨드 처리가 아직 끝나지 않은 'busy' 상태에서만 제한합니다. 생략하면 0으로 간주됩니다.

예) LockEvent; ⇒ 모든 이벤트가 항상 발생하지 않게 합니다.

LockEvent 1,$on_touch; ⇒ 터치 이벤트가 항상 발생하지 않게 합니다.

LockEvent ,$on_click,1 ⇒ LCD가 'busy' 상태일 때만 클릭 이벤트의 발생을 제한합니다.

Cs

Cs *checksum*

바로 뒤에 보내어질 커맨드에 대한 체크섬을 미리 보냅니다.

이 체크섬 커맨드는 의무적으로 보내지 않아도 되는 선택 사항입니다.

후에 보낼 커맨드의 각각의 알파벳(대소문자 구분하며 공백과 세미콜론 포함)의 아스키값들을 모두 더한 값을 256으로 나눈 나머지 값을 10진수 또는 16진수로 표현하여 보냅니다. 16진수로 보내려면 앞에 &H를 붙여야 합니다.

체크섬이 맞지 않으면 해당 커맨드는 실행하지 않습니다.

예) Cs 106; PSet 2,5;

Cs &H6A; PSet 2,5;

설명 : 80('P')+ 83('S')+ 101('e')+ 116('t')+ 32(공백)+ 50('2')+ 44(',')+ 53('5')+ 59(';') = 618

618을 256으로 나눈 나머지는 106

106을 16진수로 표현하면 6A

CsErrNoticeMode

CsErrNoticeMode *[show_msgbx], [send_event]*

체크섬이 맞지 않았을 경우 할 동작을 정의합니다.

show_msgbx가 1일 경우 : LCD 화면 상에 메시지박스를 통해 잠깐 보입니다. 생략시/부팅시 기본값은 1입니다.

send_event가 1일 경우 : 이벤트를 날려서 알립니다. 생략시/부팅시 기본값은 1입니다.

예) CsErrNoticeMode 1;

CsErrNoticeMode ,1;

EnableSysMsg

EnableSysMsg *[enable]*

커맨드 에러나 알림 메시지 등의 시스템 메시지를 메시지박스를 통해 화면에 보여줄지 결정합니다.

이 커맨드는 약간의 시간이 소요될 수 있습니다.

이 설정은 전원을 꺼도 기억됩니다.

enable = 0 : 시스템 메세지를 화면에 나타내지 않습니다.

enable = 1 : 시스템 메세지를 화면에 보여줍니다.(디폴트 값)

예) EnableSysMsg;

EnableSysMsg 0;

SetRtsDelay　[RS485 통신 전용 커맨드]

SetRtsDelay *[delay_time]*

이벤트 발생시 RS485 모드를 지원하기 위한 RTS 관련 지연 시간을 설정합니다(ms단위)

IntelliLCD에서 이벤트를 보낼 때 내부적으로 다음과 같은 순서로 처리합니다.

(일반 RS232 통신시에는 이 커맨드를 호출하지 말거나 SetRtsDelay 0;으로 해야합니다)

1. RTS신호를 올린다.
2. 데이터를 보낸다.
3. 시간 지연시킨다. ⇓
4. RTS신호를 내린다.

여기서 '3.시간 지연' 할 때 지연시킬 값을 ms단위로 지정합니다.

생략시 값은 50이며 사용할 수 있는 값은 0~300입니다.

부팅시에는 0으로 조정되어져 있습니다.

예) SetRtsDelay;

Delay

Delay *[delay_time]*

아무 동작도 하지 않고 시간을 지연시킵니다. 시간값은 millisecond 단위이며 0~5000까지의 값을 가질 수 있으며 생략시 값은 50입니다.

예) Delay 100;

SendByte

SendByte *value*
특정한 값을 RS-232C 통신을 통해 내보냅니다.
출력값(*value*)은 1바이트의 바이너리 값입니다.
예) SendByte 65; ⇓ 'A' 가 출력됨
Set #b,66;SendByte #b; ⇓ 'B'가 출력됨

SendStr

SendStr *string*
특정한 값을 RS-232C 통신을 통해 내보냅니다.
출력값(*value*)은 문자열입니다.
예) SendStr 'Hello'; ⇓ 'Hello' 가 출력됨
Set #str,'Hi!'; SendStr #str; ⇓ 'Hi!'가 출력됨

SendHex

SendHex *value, digits*
특정한 값을 RS-232C 통신을 통해 내보냅니다.
출력값(*value*)은 16진수 형식이며 자리수(*digits*)를 지정할 수 있습니다.
빈자리는 0으로 채웁니다.
예) SendHex 255,3; ⇓ '0FF' 가 출력됨
SendHex 255,4; ⇓ '00FF' 가 출력됨
SendHex 255,5; ⇓ '000FF' 가 출력됨
Set #a,16;SendHex #a,2; ⇓ '10' 이 출력됨

SendDec

SendDec *value [,digits]*
특정한 값을 RS-232C 통신을 통해 내보냅니다.
출력값(*value*)은 10진수 형식이며 자리수(*digits*)를 지정할 수 있습니다.
빈 자리는 0으로 채웁니다.
자릿수를 생략할 수 있습니다.
예) SendDec 255; ⇓ '255' 가 출력됨
SendDec 255,4; ⇓ '0255' 가 출력됨
SendDec 255,5; ⇓ '00255' 가 출력됨
Set #a,16;SendDec #a,2; ⇓ '16' 이 출력됨

* Screen 기능

스크린이란, 단 한 개의 전체화면 사이즈의 BMP 이미지 파일을 활용하여, 부분적인 작은 영역들을 규정하여 컨트롤처럼 사용할 수 있는 화면을 뜻합니다. 기존의 컨트롤과는 다른 개념입니다.

그런 장면들을 미리 구성해 정의해 놓고 실시간으로 원하는 장면을 불러오거나 해제할 수 있습니다.

DefineScreen

DefineScreen *script_file_name [, screen_id]*

스크린을 정의합니다.

script_file_name : Screen script로서 스크린의 구체적인 내용을 스크립트로 작성해 놓은 것입니다. Storage Card 폴더에 있어야 합니다.

screen_id : 스크린 번호(1~255, 생략시 1)로서 나중에 스크린을 구분하는 용도로 쓰입니다.

예) DefineScreen 'script1.txt';

DefineScreen '.₩screen2₩script2.txt',2;

StartScreen

StartScreen *[, screen_id]*

스크린을 동작시킵니다.

스크린이 동작되고 있는 상태에서는 터치를 누를 때 눌려진 그래픽 효과가 표현되고 눌린 영역의 영역ID와 스크린ID가 포함되어져 $on_screen_touch 이벤트가 발생합니다.

(이 때 $on_touch이벤트는 생략됩니다)

DefineScreen으로 미리 정의되어 있어야만 합니다.

screen_id는 구동할 스크린의 ID(1~255)이며 생략시 1입니다.

예) StartScreen 1;

EndScreen

EndScreen

스크린의 동작을 끝냅니다.

스크린의 동작을 끝낸다는 것은 터치를 눌러도 더 이상 그래픽 효과가 나지 않고 이벤트도 발생하지 않는다는 것을 뜻합니다.

동작중인 스크린이 없을 때 EndScreen을 한다 해도 오류가 발생하지 않습니다.

예) EndScreen 1;

* Screen script 파일 작성법

특정 파일명(확장자 무관, 예: script1.txt)을 가진 파일을 PC에서 생성하고 그 내용을 작성하여 ActiveSync를 이용하여 Storage Card 폴더에 복사합니다.

파일 내용 예)

common main_image=full1.bmp

area id=1;rect=30,30,200,150;press_image=button_push1.bmp;

area id=2;rect=100,100,150,150;press_image=full_size_press.bmp;clip_mode=yes;

각 행은 반드시 커맨드와 한칸 띄움으로 시작해야 하며

하나의 영역을 나타내는 커맨드는 'area'입니다.

그 후에 '파라미터=값'이 세미콜론(;)으로 이어붙여져서 파라미터=값; 파라미터=값; 파라미터=값; ...

이런 식으로 표현됩니다. '//'이후는 무시됩니다.

전체 화면은 common 커맨드를 사용하여, common main_image= 이후에 파일명을 지정합니다.

스크립트에서 언급하는 모든 파일은 해당 스크립트 파일이 있는 폴더와 같은 폴더에 있어야 합니다.

<기본 파라미터>

id : 영역 ID로 1~255 범위의 값을 가질 수 있습니다.('영역'이란 마치 컨트롤 같은 개념입니다)

rect : 유저의 터치를 받아드릴 사각 영역을 left,top,right,bottom 의 형식으로 지정합니다.

press_image : 터치가 눌렸을 때 화면에 표시할 이미지의 파일명입니다.
터치를 뗄 경우는 자동으로 다시 원래대로 바뀝니다.(전체 화면 그대로)
별도 지정이 없는 경우, 화면이 변경될 부분은 rect로 지정된 터치 영역입니다.

clip_mode : press_image를 규정짓는 값으로서 두가지 값을 가질 수 있습니다.(yes / no)
no : press_image에 처음부터 딱맞추어진 작은 이미지를 저장합니다.(생략시는 이 설정이 기본값)
yes : press_image에 일단 전체적인 이미지를 저장한 후 추후 표시할 때만 발췌하여 표시

<고급 파라미터>

press_image_rect : 특수하게 만약 '화면 갱신 영역'이 '터치 영역'과 다를 경우 화면 갱신 영역을 강제로 지정합니다. (left,top,right,bottom 형식)

transparent_color : press_image에서 투명색으로 간주할 색상을 지정합니다.
&HFF0000과 같이 IntelliLCD 커맨드에서 컬러 지정하는 방식과 동일하게 사용합니다.

press_sound : 누를 때 낼 소리를 WAV 파일명으로 지정합니다. (사운드가 지원되는 모델에서만 가능)

jump_screen : 누를 때 다른 스크린으로 전환합니다. 스크린 번호가 값이 됩니다.
예) jump_screen=2;

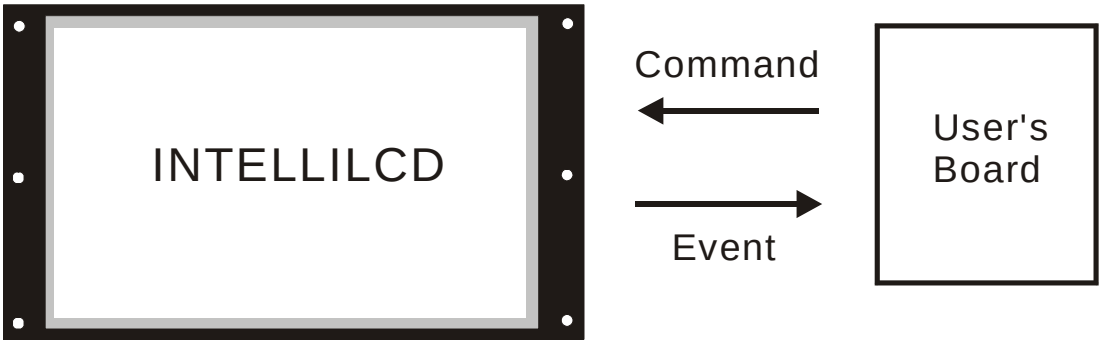
<Tip>

모든 영역에 공통적인 파라미터는 common 커맨드를 사용하면 중복 기입을 피할 수 있습니다.

예) common press_sound=sound1.wav;clip_mode=yes; press_image=fullsize.bmp;
area id=1;rect=30,30,300,200;

3-3. 이벤트

이벤트란 주로 터치입력이 발생했을 경우 인텔리LCD에서 RS232를 송신하는 기능을 말합니다. 터치 입력 이외에도 시스템 상태를 알려 주기 위한 이벤트도 있습니다.



3-3-1. 아스키모드와 바이너리 모드

일단 아스키 모드와 바이너리 모드 중 어떤 것을 사용할지를 결정해야 합니다. 여러분께서 사용하시는 프로세서와 컴파일러의 종류에 따라서 둘중 편리한 것을 선택하시면 됩니다.

큐블록을 사용하시는 분들은 아스키모드를 권장합니다. 큐블록에는 종료코드를 인식해서, 인터럽을 발생시켜주는 기능이 있기 때문입니다. 일반적인 8비트 MCU를 사용하시는 분들은 바이너리 모드를 권장합니다. 바이너리 모드가 어셈블리나 C언어에서는 다루기 편리합니다.

a) 프로토콜 구조

두 가지 모드가 있으며 SetRecvMode 커맨드를 보내서 선택할 수 있습니다.

(1) 아스키 모드 (디폴트이며 권장함)

* 길이 고정형 이벤트: **총 길이는 11 byte로 항상 고정되어 있으며 대부분의 이벤트에서 사용됩니다.**

STX (&H02)	Event Number 16진 문자로 2 byte	7 byte 의 파라미터 (주로 16진 문자로 구성)	ETX (&H03)

* 길이 가변형 이벤트(고급): 총 길이는 가변적입니다.

STX (&H02)	가변형 식별자 'FF' 2 byte	STX~ETX의 길이: 16진 문자로 4 byte	Event Number: 16진 문자로 2 byte	내용: 가변적 길이	ETX (&H03)

(2) 바이너리 모드

* 길이 고정형 이벤트: **총 길이는 8 byte로 항상 고정되어 있으며 대부분의 이벤트에서 사용됩니다.**

STX (&H02)	Event Number: 1 byte	Param : 5 byte	ETX (&H03)

* 길이 가변형 이벤트(고급: 총 길이는 가변적입니다.)

STX (&H02)	가변형 식별자 &HFF 1 byte	STX~ETX의 길이: 상위 바이트	STX~ETX의 길이: 하위 바이트	Event Number: 1 byte	내용 : 가변적 길이	ETX (&H03)

* 반면 송신용 커맨드는 무조건 ASCII 방식입니다. 여기에서 ASCII 모드를 BINARY 모드로 바꾸어도 송신 커맨드는 그대로 ASCII 모드입니다.

3-3-2. 컨트롤 입력 관련 이벤트

LCD화면상에 컨트롤을 클릭할 경우 해당 컨트롤이 조작되었다는 것을 알려주기 위해서 다음과 같은 이벤트가 발생됩니다.

b) 이벤트 목록

(1) $on_click – 버튼을 클릭할 경우 발생.

아스키 :

&H02	'8' (&H38)	'1' (&H31)	컨트롤 ID : 16진 문자로 2 byte	'1'(&H31)이면 눌림, '0'(&H30)이면 뗌 1 byte	'*' (&H2A) 4 byte 연속	&H03

바이너리 :

&H02	&H81	컨트롤 ID : 1 byte	&H01 이면 눌림, &H00 이면 뗌 1 byte	&H00	&H00	&H00	&H03

(2) $on_check – 체크박스나 토글 버튼을 체크/해제할 경우 발생

아스키 :

&H02	'8' (&H38)	'2' (&H32)	컨트롤 ID : 16진 문자로 2 byte	'1'(&H31)이면 체크, '0'(&H30)이면 해제 1 byte	'*' (&H2A) 4 byte 연속	&H03

바이너리 :

&H02	&H82	컨트롤 ID : 1 byte	&H01 이면 체크, &H00 이면 해제 1 byte	&H00	&H00	&H00	&H03

(3) $on_position - 슬라이더 컨트롤의 위치값을 통지 받아야 할 필요가 있을 경우 발생

아스키 :

&H02	'8' (&H38)	'3' (&H33)	컨트롤 ID : 16진 문자로 2 byte	위치값: 16진 문자로 4 byte	'*' (&H2A)	&H03

바이너리 :

&H02	&H83	컨트롤 ID : 1 byte	위치값 상위 byte	위치값 하위 byte	&H00	&H00	&H03

3-3-3. 유저 입력 반환 이벤트

컨트롤이 없는 지역, 즉 빈 화면을 눌렀을 경우에는 해당 지점의 좌표를 송신해줍니다.

(4) $on_touch – 터치(마우스 버튼)가 눌리거나 뗄 경우 발생.

아스키 :

&H02	'8' (&H38)	'4' (&H34)	'1'(&H31)이면 눌림, '0'(&H30)이면 뗌 1 byte	X 좌표값: 16진 문자로 3 byte	Y 좌표값: 16진 문자로 3 byte	&H03

바이너리 :

&H02	&H84	&H01 이면 눌림, &H00 이면 뗌 1 byte	x 좌표 상위 byte	x 좌표 하위byte	y 좌표 상위 byte	y 좌표 하위byte	&H03

인텔리 LCD에는 PC용 키보드를 연결할 수 있습니다. 사용자가 키보드의 키를 입력했을 경우 다음과 같은 이벤트가 발생합니다.

(5) $on_key_press – 키를 누를 경우 발생. (키보드를 연결했을 경우)

아스키 :

&H02	'8' (&H38)	'5' (&H35)	아스키 코드 (대소문자 구분)	'*' (&H2A) 6 byte 연속	&H03

바이너리 :

&H02	&H85	아스키 코드 (대소문자 구분)	&H00	&H00	&H00	&H00	&H03

3-3-4. RTC관련 이벤트

현재 시각을 이벤트로 받을 수 있습니다. iTL720이나 Cuwin iTL 모드에는 RTC 배터리가 내장되어 있어서 시간이 저장되지만, **iTL710/740/840에서는 재부팅 후에 시각이 보존되지 않습니다.**

(6) $on_date – 현재 날짜 조회 요청에 대한 응답입니다.

아스키 :

&H02	'8' (&H38)	'6' (&H36)	year: 16진 문자로 3 byte	month: 16진 문자로 1 byte	day: 16진 문자로 2 byte	'*' (&H2A)	&H03

바이너리 :

&H02	&H86	year 상위 byte	year 하위byte	month 1 byte	day 1 byte	&H00	&H03

(7) $on_time – 현재 시각 조회 요청에 대한 응답입니다.

아스키 :

&H02	'8' (&H38)	'7' (&H37)	hour: 16진 문자로 2 byte	minute: 16진 문자로 2 byte	second: 16진 문자로 2 byte	'*'(&H2A)	&H03

바이너리 :

&H02	&H87	hour: 1 byte	minute: 1 byte	second: 1 byte	&H00	&H00	&H03

3-3-5. Screen 관련 이벤트

인텔리LCD는 풀화면 그래픽 파일을 이용해 장면을 만들고 특정 영역을 컨트롤처럼 사용할 수 있는 Screen 기능이 있습니다.

(8) $on_screen_touch – 스크린상의 영역이 눌려졌을 때 발생합니다. 이 이벤트가 발생할 때는 $on_touch 이벤트가 발생하지 않습니다.

아스키 :

&H02	'8' (&H38)	'C' (&H43)	screen ID: 16진 문자로 2 byte	영역 ID: 16진 문자로 2 byte	'1'(&H31)이면 눌림, '0'(&H30)이면 뗌 1 byte	'*' (&H2A) 2 byte 연속	&H03

바이너리 :

&H02	&H8C	screen ID: 1 byte	영역 ID: 1 byte	&H01 이면 눌림, &H00 이면 뗌 1 byte	&H00 2 byte 연속	&H03

3-3-6. 기타 이벤트

인텔리LCD에는 1M Byte (1048576 byte)의 수신 버퍼가 있습니다. 그래픽 처리 등을 하느라 커맨드 처리 속도가 통신 속도를 따라가지 못하다보면 버퍼가 지속적으로 늘어날 수 있습니다. 그러다가 버퍼가 일정수준 이상 채워진다면 다음과 같은 이벤트가 발생됩니다.

(9) $on_buffer_warning - 버퍼가 경고 수준(사용자가 설정 가능 - SetBufferWarning 커맨드) 이상 채워졌을 경우 발생합니다. 넘치는 패킷을 버리지 않습니다. 클라이언트 측에서 이 이벤트를 받았을 경우 커맨드 전송을 잠시 지연시켰다가 재개하도록 하면 보다 안정적인 동작을 구현할 수 있습니다.

아스키 :

&H02	'8' (&H38)	'8' (&H38)	'*' (&H2A) 7 byte 연속	&H03

바이너리 :

&H02	&H88	&H00	&H00	&H00	&H00	&H00	&H03

(10) $on_buffer_overflow – 버퍼가 1M Byte (1048576 byte) 이상 채워졌을 경우 발생합니다.
넘치는 패킷은 버리므로 커맨드가 부분적으로 깨어지므로 에러가 계속 발생합니다.
실제 필드에서는 어떠한 경우에도 이 이벤트가 날라올 만큼 버퍼가 차지 않도록 송신을 제한해야 합니다.

아스키 :

&H02	'8' (&H38)	'9' (&H39)	'*' (&H2A) 7 byte 연속	&H03

바이너리 :

&H02	&H89	&H00	&H00	&H00	&H00	&H00	&H03

(11) $on_lcd_state – GetLcdState 명령에 대한 응답으로서 LCD의 상태를 알려줍니다.

처음 부팅했을 때는 커맨드 요청이 없이도 1번 자연적으로 발생합니다.

주의: LCD가 부팅 중이거나 전원이 꺼져 있을 때는 이벤트가 발생하지 않으며 이 이벤트는 1초에 두번 이상 발생하지 않습니다.

아스키 :

&H02	'8' (&H38)	'A' (&H41)	LCD 정상 동작 여부 '1'이면 정상, '0'이면 비정상 1byte	'*' (&H2A) 6 byte 연속 (추후 항목 추가 가능)	&H03

바이너리 :

&H02	&H8A	LCD 정상 동작 여부 &H01이면 정상, &H00이면 비정상 1byte	&H00 4 byte 연속 (추후 항목 추가 가능)	&H03

(12) $on_checksum_error – 체크섬 에러가 발생했을 경우 보내지는 이벤트입니다.

아스키 :

&H02	'8' (&H38)	'B' (&H42)	'*' (&H2A) 7 byte 연속	&H03

바이너리 :

&H02	&H8B	&H00 5 byte 연속	&H03

3-3-7. 길이 가변형 이벤트 (고급)

(13) $on_file_read – 파일을 읽어서 그 내용을 알려줄 필요가 있을 경우 발생합니다. 보통 FRead 코맨드에 대한 응답입니다.

아스키 :

STX (&H02)	'F' (&H46)	'F' (&H46)	STX~ETX의 길이: 16진 문자로 4 byte	'4' (&H34)	'1' (&H31)	Query ID: 16진 문자로 2 byte	'1'이면 성공, '0'이면 실패 1 byte	16진 문자의 연속체로 표현된 파일 내용(실패일 경우 없음): 가변 길이	ETX (&H03)

* Query ID :

FRead 코맨드를 보낼 때 파라미터로 보냈던 값으로서 요청과 응답이 일치하는지 확인하기 위한 용도로 쓰입니다.

* 16진 문자의 연속체로 표현된 파일 내용 :

아스키 모드에서는, 만약 &H30,&H2A,&HF7 과 같은 3 byte의 데이터가 있을 경우 '302AF7'(6 byte)로 표현됩니다.

바이너리 :

STX (&H02)	&HFF	'STX~ETX의 길이'의 상위 바이트	'STX~ETX의 길이'의 하위 바이트	&H41	Query ID: 1 byte	&H01 이면 성공, &H00 이면 실패 1 byte	Binary 파일 내용 (실패일 경우 없음): 가변 길이	ETX (&H03)

제4장 부가적 기능

4-1. 스피커 연결

제품 자체 내에는 스피커가 없으므로 일반 스피커(Amp내장)를 연결하시면 오디오 출력을 확인할 수 있습니다. (iTL710/740에서는 지원하지 않습니다.)

4-2. USB Key & Mouse 연결

USB Host와 Device 각각 1개의 Port를 가지고 있으며 USB Ver1.1입니다.
USB Host Port는 마우스 및 키보드를 사용 할 수 있으며 마우스와 키보드를 동시에 사용 하려면 USB Hub를 사용하셔야 하며 USB Ver1.1의 HUB를 사용할 것을 권장 합니다.
USB Device는 Active Sync를 사용하기 위한 USB Port입니다. – ActiveSync 사용법을 참조하세요
[주의] USB Mouse는 PS2호환 되는 것을 사용하시기 바랍니다.

4-3. SD 메모리 카드

SD메모리 카드는 최대 2GB까지의 용량을 지원하며 IntelliLCD에서 다음과 같은 용도로 Memory를 사용합니다. (모듈만 구입시 SD카드가 포함되어 있지 않습니다.)

1. 사용자 Data를 저장할 수 있는 영역입니다.
   - BMP 이미지, WAV 사운드 파일을 미리저장한뒤, 화면에 표시하거나 PLAY할 수 있습니다.
   - 운영중 발생하는 데이터를 파일형태로 저장할 수 있습니다.
2. 사용자 추가할 수 있는 트루 타입 폰트를 저장 할 수 있습니다.
   - 지정된 폴더에 (Storage Card₩Fonts) 폰트를 추가한 후 전원Off후 다시 On하면 폰트를 자동 인식합니다.

(TIP) FAT 파일 시스템을 사용하고 있어 오랜 기간 동안 파일을 쓰고 삭제할 경우 파일의 단편화 문제로 IntelliLCD에서 인식되는 시간이 길어질 수 있습니다. 이럴 경우 PC에서 디스크조각 모음을 해주시면 인식 소요시간이 최적화 됩니다.

4-4. 펌웨어 업그레이드

인텔리LCD의 기능향상 및 버그수정을 위해서 유저가 펌웨어를 업그레이드할 수 있도록 해놓았습니다. 따라서 유저여러분께서는 펌웨어 업그레이드를 위해서, 해당제품을 저희회사로 보내시거나, 방문하실 필요없이, 지금 설명하는 방법대로, 직접 펌웨어 업그레이드를 하실 수 있습니다.
펌웨어 업그레이드는 SD 메모리카드를 통해 이루어 지며 업그레이드 파일은 당사 웹사이트를 통해 최신의 펌웨어 파일을 다운로드 받으실 수 있습니다.
웹사이트 주소 (www.comfile.co.kr)

< 업데이트 방법 >

1. 본사의 사이트를 통해 펌웨어 파일을 다운 받습니다.
2. 받은 파일을 SD 메모리카드에 복사 합니다.
   - PC에 SD Card Reader기가 없을 경우 ActiveSync를 사용합니다. (ActiveSync사용법 참고)
   - 복사 위치는 SD 메모리카드의 Root에 이름은 변경하지 말고 복사합니다.
3. 복사가 완료되면 SD 메모리카드를 IntelliLCD에 장착 후 전원을 Off - On하면 다음과 같은 화면과 같이 업데이트 정보가 나타나고 업데이트를 완료합니다.

[주의1] 업데이트 후 원본파일(SD Card의 파일)은 자동 삭제됩니다.

[주의2] 해당제품은 제품 제조시점에서 최신 펌웨어가 기록되어 있습니다.
여러분이 제품을 구입하신 뒤, 펌웨어 버전을 확인하시는 것이 좋습니다.
제품 제조시점과 구입시점에 시간차로 인해, 이전 버전의 펌웨어가 기록되어 있는 경우도 있기 때문입니다.

4-5. 사용자 폰트 추가

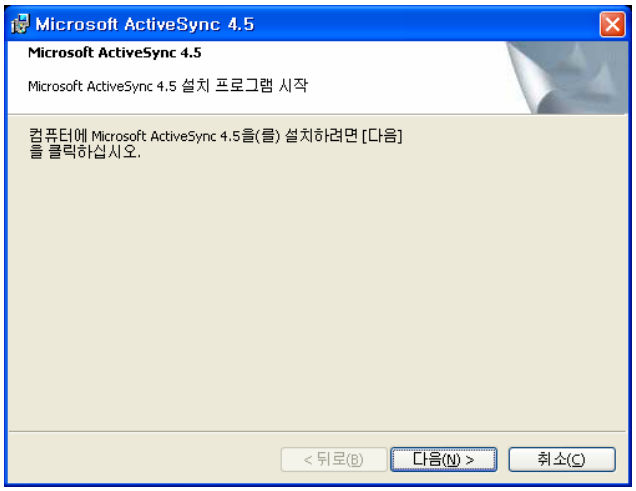
1. 트루타입 폰트를 다음의 위치에 복사합니다. (SD 메모리 카드의 Fonts폴더에 폰트 파일 추가 합니다. 확장자는 .TTF or .TTC) 여러분의 PC에 있는 폰트를 IntelliLCD에서 사용하실 수 있습니다. PC상의 폰트파일은 Windows 폴더아래 Fonts 폴더에 있습니다.
2. 폰트 추가 후, 전원을 Off - On하면 폰트리스트가 출력됩니다.
   - 폰트리스트에 나타나는 이름이 폰트 이름으로 폰트 변경시 Command와 같이 전송할 데이터 입니다. (TextFont 커맨드의 *font_face*인자에 해당됩니다.)
3. 기존 폰트(내장) 리스트
   - 굴림체, 굴림
   - 돋움체, 돋움
   - 바탕체, 바탕
   - 궁서체, 궁서
   - Tahoma
   - Courier New

[주의] 폰트 파일(*.TTF나 *.TTC)이 읽기 전용 속성이 되어 있으면 인식하지 않습니다.

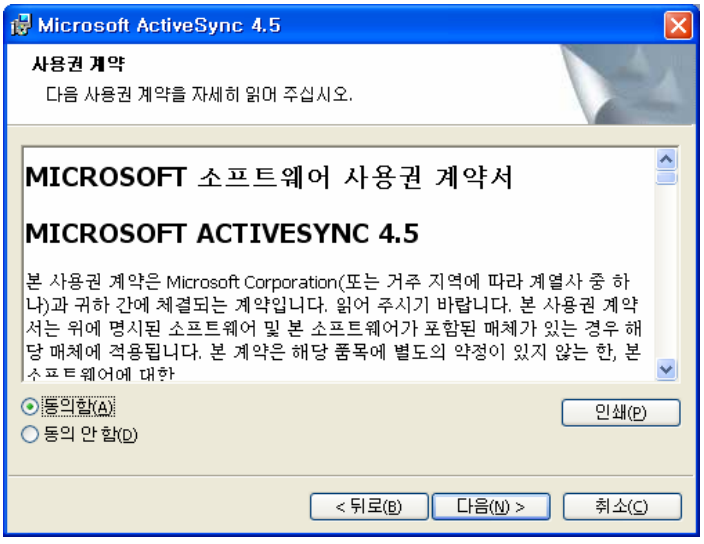
4-6. ActiveSync 사용법

ActiveSync라는 소프트웨어 인텔리 LCD와 PC를 연결하기위한 소프트웨어입니다.
ActiveSync를 다음 사이트에서 다운로드 받으실 수 있습니다. (URL은 변경될 수 있습니다)
“http://www.microsoft.com/korea/windowsmobile/activesync/activesync45.mspx”
만약 마이크로소프트사의 웹사이트에서 다운받는 것이 번거로우시다면, Naver검색 등을 통해서도 쉽게 구하실 수 있습니다. ActiveSync는 무료 소프트웨어 입니다.
ActiveSync 4.5버전을 기본으로 설명해 드리겠습니다.

1. ActiveSync설치 하기 전까지는 IntelliLCD와 PC를 연결하지 않습니다.
2. 다운 받으신 파일을 더블 클릭하시면 아래와 같이 진행 됩니다. “다음”을 클릭합니다.

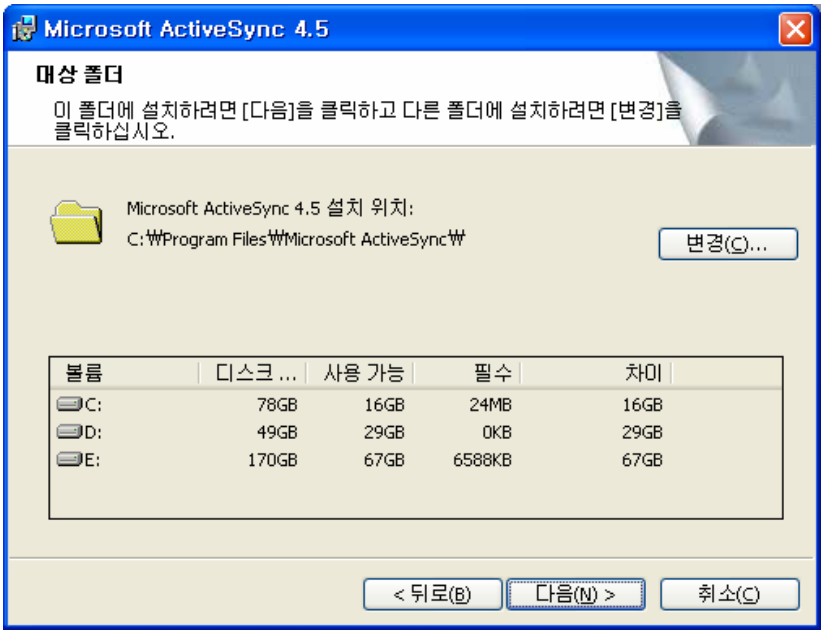


3. 프로그램 설치준비가 끝나면, 아래와 같이 사용권 계약 창이 뜨게 됩니다.
“동의함”을 선택합니다. 다음을 누릅니다.

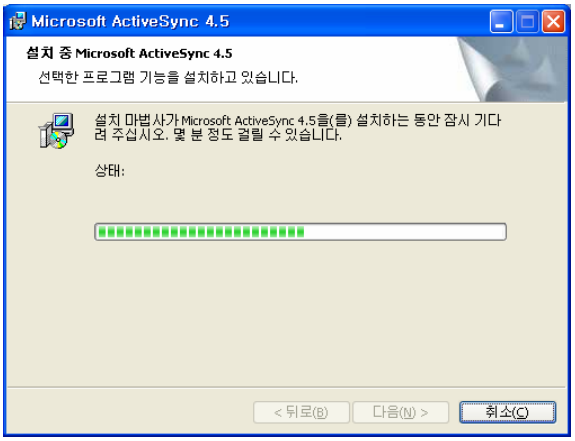


4. 사용자 정보를 입력 합니다. “다음”을 클릭 합니다.

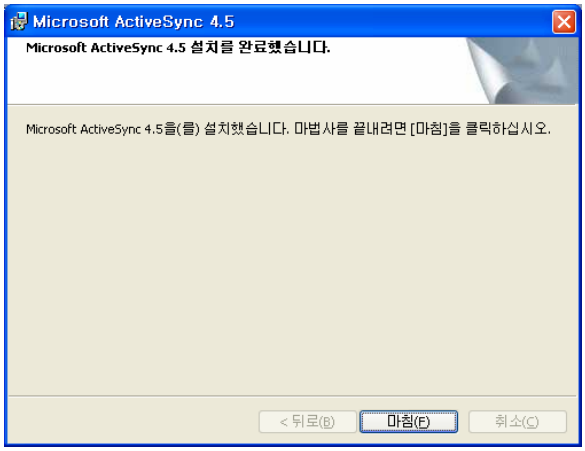
5. “다음”을 누르게 되면 다음과 같이 나타나는데, 설치할 폴더를 결정할 수 있습니다.
(아무 설정 없이 설치할 것을 권장 합니다.)



6. 역시 “다음”을 누르면, 설치되는 진행 상황이 다음과 같이 나타나면서 설치가 진행 됩니다.

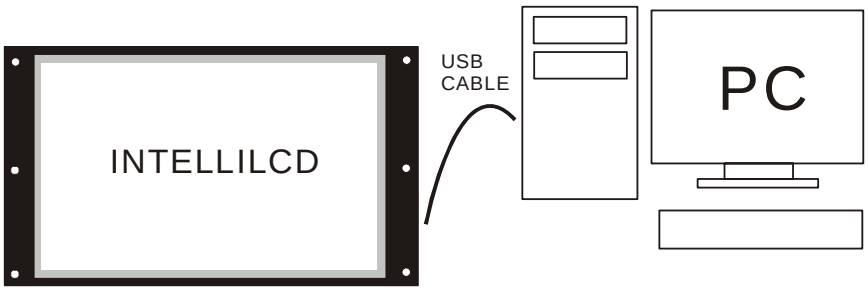


7. 설치가 끝나면 다음 같은 화면을 볼 수 있습니다.

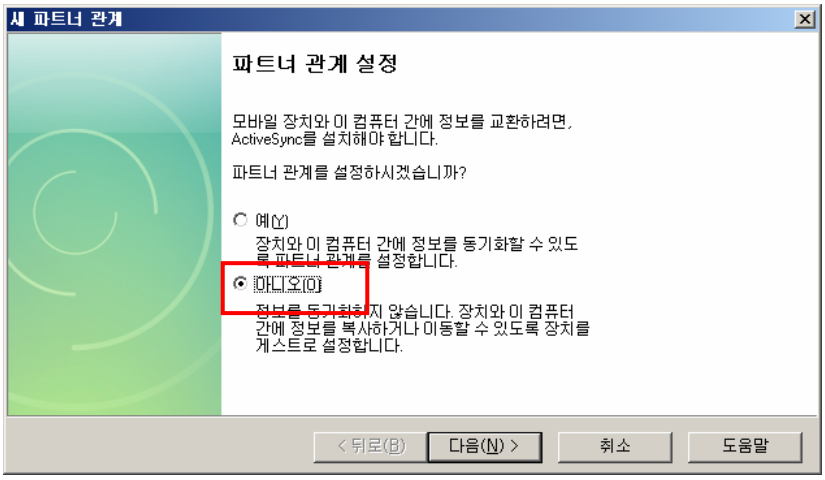


4-7. ActiveSync 실행

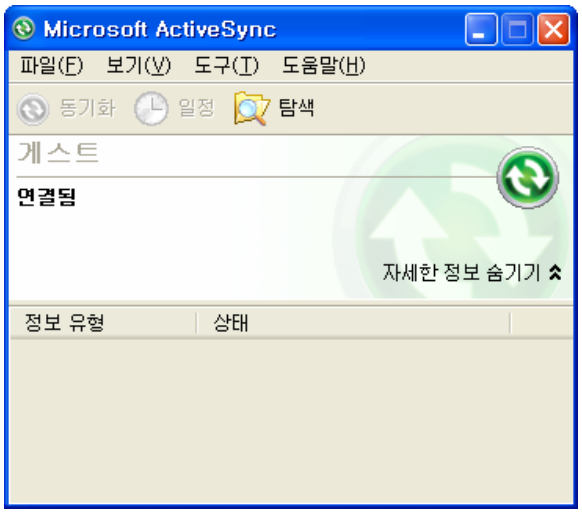
1. IntelliLCD와 PC를 제공된 USB Cable로 서로 연결합니다.



2. 연결이 정상적으로 이루어지면 다음과 같이 새 파트너 관계 창이 뜨게 됩니다. **"아니오"를 선택합니다.**



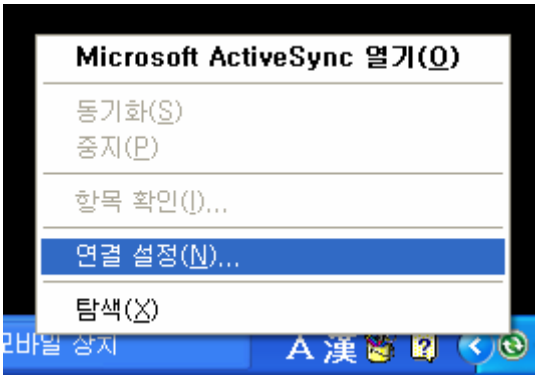
3. 연결이 완료되면 다음과 같은 창이 뜨게 됩니다.



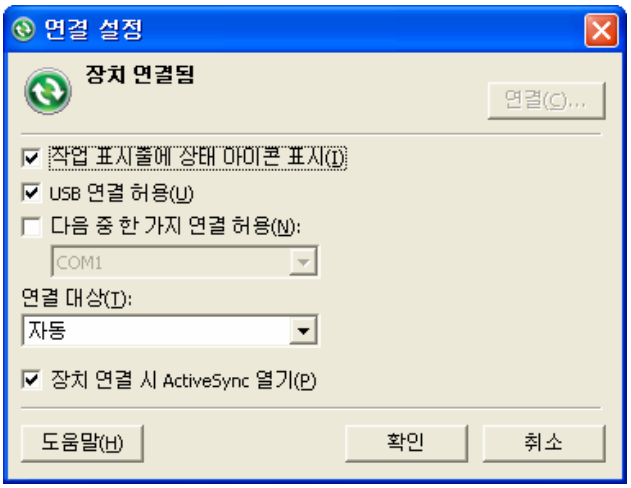
4. 이것으로 IntelliLCD와 PC간의 연결이 완료 된 것입니다. 연결이 된 상태에서는 PC에서 IntelliLCD의 저장장치의 내용을 확인할 수 있습니다.

만약 위와 같이 했는데도 연결이 안될 경우 다음과 같이 확인하세요.

- ActiveSync 트레이 아이콘 위에서 오른쪽 버튼을 클릭합니다. 연결 설정을 선택하세요



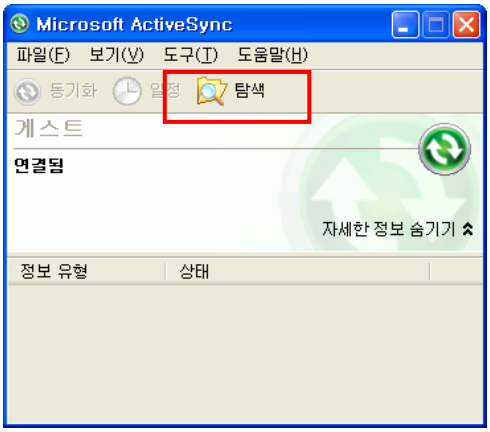
- 아래와 같이 설정되어있는지 확인 하세요.



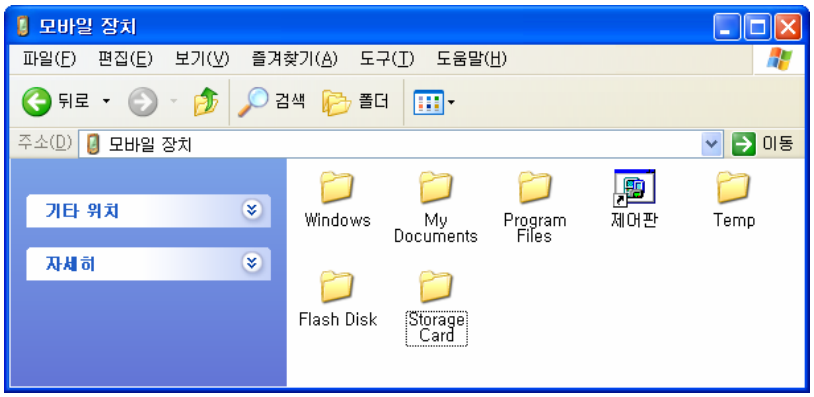
- 또는 USB 케이블을 재 연결 하시기 바랍니다.

4-8. IntelliLCD의 저장 장치 접근하기

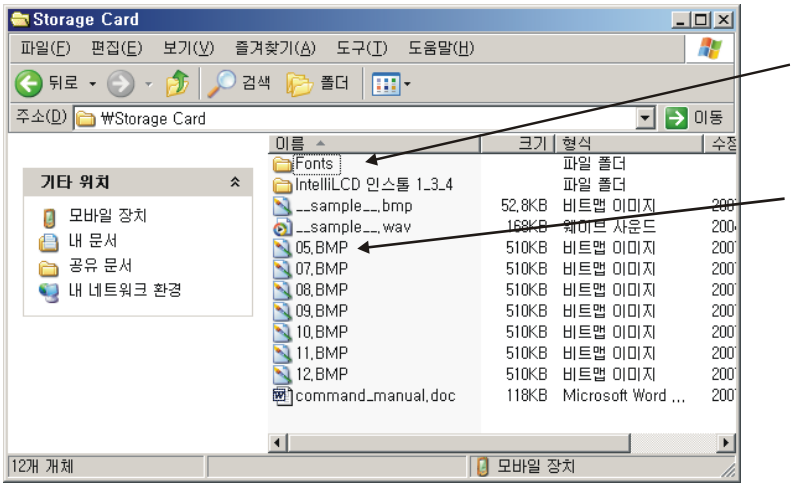
1. 탐색 아이콘을 클릭 합니다.



2. 탐색아이콘을 클릭하면 다음과 같은 탐색창이 뜨게 됩니다.



윈도우의 탐색기와 사용방법이 동일합니다. 여러분의 PC에 있는 파일을 이곳을 끌어다 놓으면 IntelliLCD로 복사됩니다. 반대로 이곳에 있는 파일을 PC상의 탐색기의 특정 폴더에 끌어놓으면 IntelliLCD의 파일을 PC로 복사할 수 있습니다. 주의 하실점은 Storage Card 폴더 이외의 영역은 건드리면 안된다는 것입니다. Storage Card폴더는 SD카드입니다. 이 안에 Font나 BMP파일을 카피해넣으면 됩니다.

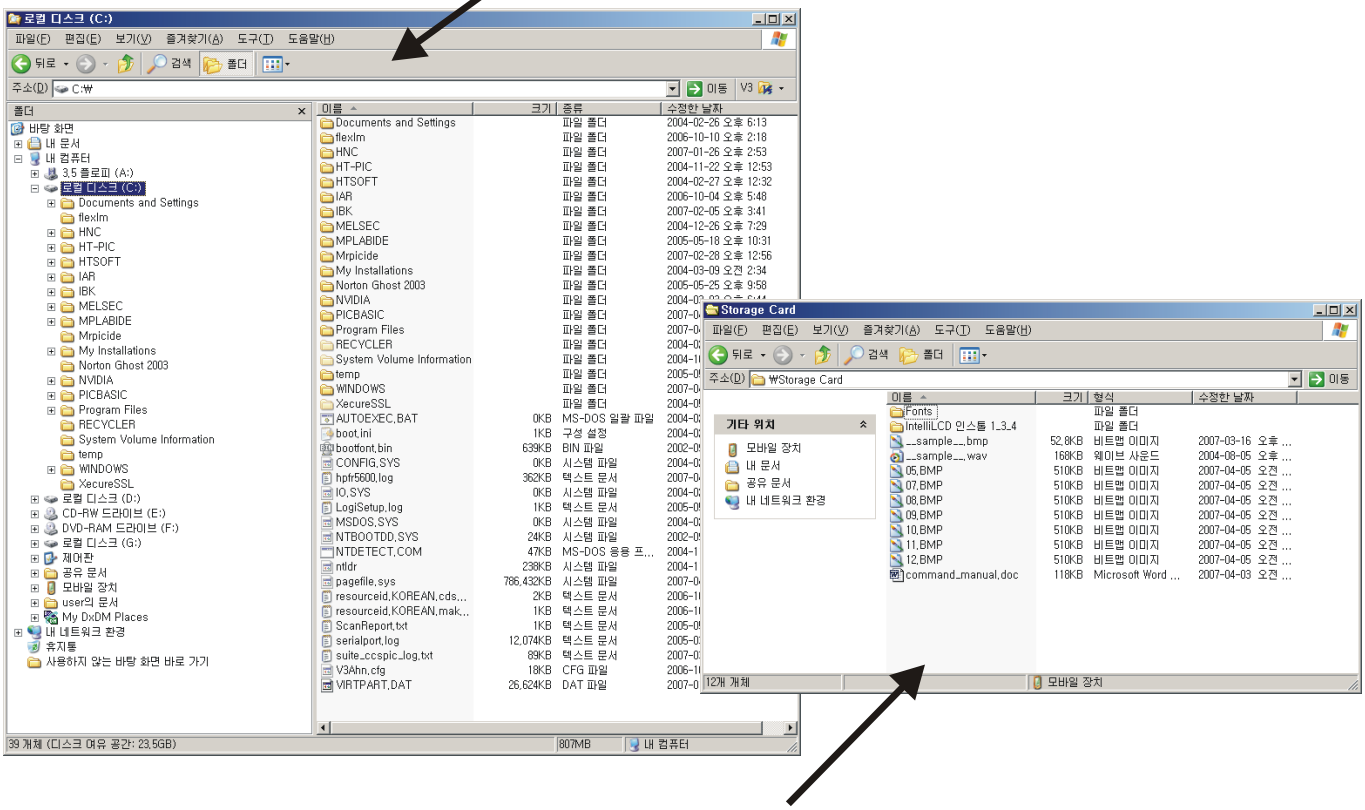


다음은 Storage Card 폴더안의 내용입니다. 폰트를 추가하시려면 Fonts폴더안에 해당파일을 카피하시면 됩니다.

BMP파일은 이곳에 카피하신 뒤, DrawImage 명령어로 해당 파일명과 위치를 지정하시면, 화면상에 표시됩니다.

펌웨어 업그레이드 시에도 이 폴더에 해당파일을 카피하신 뒤, 전원을 Off-On하시면 됩니다.

이 탐색기는 PC상의 탐색기입니다. 이곳에 있는 파일을
드레그해서 오른쪽에 있는 탐색기에 드롭하면, 해당파일이 카피됩니다.



이 탐색기는 IntelliLCD의 탐색기입니다. ActiveSync에 의해서
마치 PC의 드라이브중 하나인 것처럼 연결된 것입니다.

3. 다음은 폴더에 대한 설명입니다.

- Flash Disk : 내부 저장 Flash 메모리로 사용자가 임의로 파일을 쓰거나 삭제하는 것을 권장하지 않습니다. 시스템 예약영역입니다.
- Storage Card : 이 폴더는 SD 카드를 나타내며 사용자가 임의대로 쓰거나 삭제할 수 있는 폴더입니다. (사용자가 사용하는 폴더입니다.) 인텔리LCD 펌웨어 업그레이드 할 경우, 이 폴더안에 펌웨어 파일을 카피해야 합니다.

  Tip : 커맨드 파라미터로 파일 경로를 지정할 때 Storage Card의 하위 디렉터리에 있는 파일을 가리키고자 할 때 '.₩mydir₩image.bmp' 이런 식으로 지정할 수 있습니다.
- 기타 폴더 : 시스템 전용 폴더들로 사용자가 쓰거나 삭제할 수 없으며 만약 데이터를 쓸 경우 IntelliLCD 전원OFF시 모두 사라지게 됩니다.

4-9. Demo 화면 설정, 바꾸기

1. 데모 없애기

IntelliLCD의 초기 상태는 내장된 데모가 전원을 켜자마자 자동으로 반복 재생되게 되어 있습니다. 이 데모를 플레이하고 싶지 않을 경우 EnableDemo 0; 이라는 커맨드를 보내주면 다음부터는 전원을 재부팅해도 데모가 나오지 않습니다. 자세한 내용은 EnableDemo 커맨드의 설명 부분을 참조하세요.

2. 데모 바꾸기

전원을 켜자 마자 데모 대신 직접 작성한 커맨드가 나오게 하려면 다음과 같이 합니다.

1) PC에서 메모장을 이용하여 원하는 커맨드를 작성합니다.
   예) color &H000000;boxfill 0,0,800,480;
2) autorun.txt 라는 파일명으로 저장합니다.
3) IntelliLCD의 SD카드에다 복사합니다. (또는 Flash Disk에 복사해도 됨)

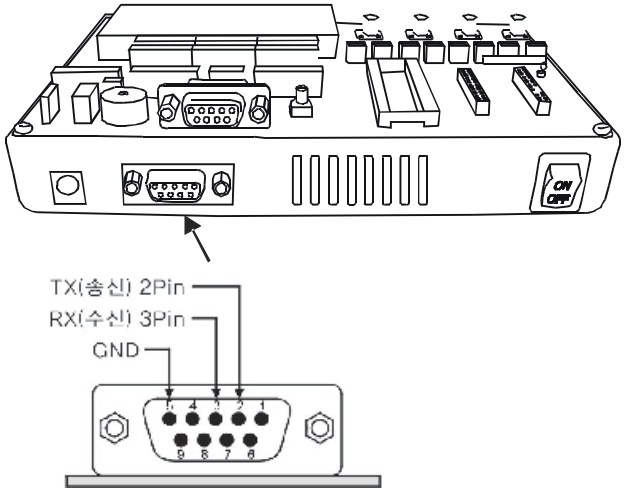
이제 전원을 켜면 데모 대신 사용자가 지정한 커맨드가 수행됩니다.

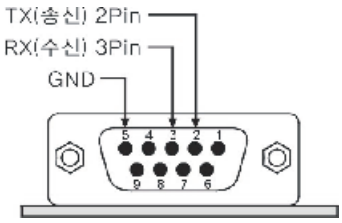
제5장 큐블록에서의 사용법

5-1. 큐블록 스터디보드와 IntelliLCD의 연결방법

큐블록과 IntelliLCD를 서로 연결해주어야, 큐블록에서의 명령어를 IntelliLCD로 전달할 수 있습니다. 앞에서 설명했듯이 IntelliLCD는 RS232로 데이터를 전송받습니다. 따라서 큐블록의 RS232채널 중 하나를 IntelliLCD와 연결해주면 됩니다.

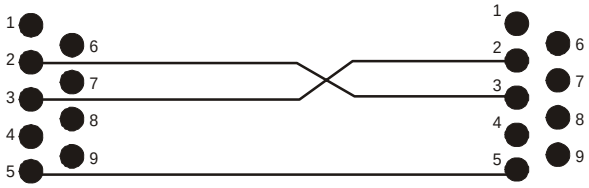
다음은 Cubloc Study Board-1을 이용해서 IntelliLCD와 연결하는 방법입니다. 아래 그림에서 보는 것처럼 Study Board-1 에 있는 RS232연결부와 IntelliLCD의 연결부가 Femail로 된 같은 콘넥터로 되어 있습니다.





IntelliLCD

따라서 1:1 케이블을 사용할 수 없고, 양쪽이 Male 로 되어 있는 크로스 케이블을 사용해야 합니다.



제품 구입시 크로스케이블 (양쪽이 Male로 되어 있는 케이블)이 포함되어 있습니다.

5-2. 큐블록에서 IntelliLCD로 커맨드 송신방법

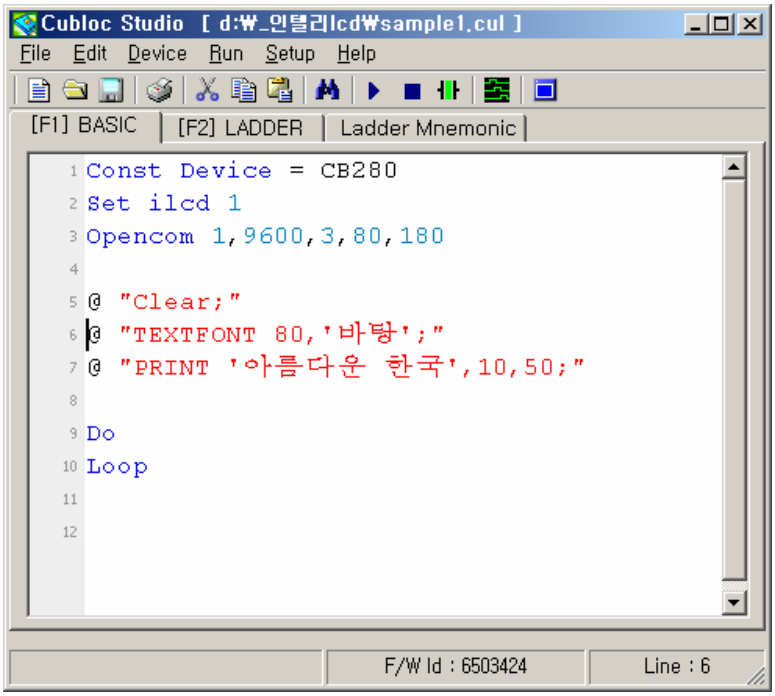
다음은 큐블록을 사용해서 IntelliLCD를 사용하는 예제 프로그램입니다. (큐블록이란, 컴파일 테크놀로지에서 만든 "임베디드컴퓨터"로써, BASIC언어와 Ladder logic으로 프로그램을 작성합니다. 큐블록에 대한 자세한 내용은 www.comfile.co.kr을 참고하시기 바랍니다.)

IntelliLCD를 위해서 Set iLcd라는 명령이 추가되었습니다. (Cubloc Studio V2.1.F 부터)
만약 가지고 계신 Cubloc Studio 가 버전 2.1.F보다 낮은버젼이라면, 컴파일 홈페이지(www.comfile.co.kr)로부터 새로운 버전의 Cubloc Studio를 다운받아 설치하시기 바랍니다.

Set iLcd 포트번호

인텔리 LCD 커맨드를 출력하는 Rs232 채널번호를 설정하는 명령어입니다. 생략시 디폴트로 1이 할당되어 있습니다.
사용 예) Set iLcd 1 ' 인텔리 LCD커맨드를 채널1로 출력합니다.

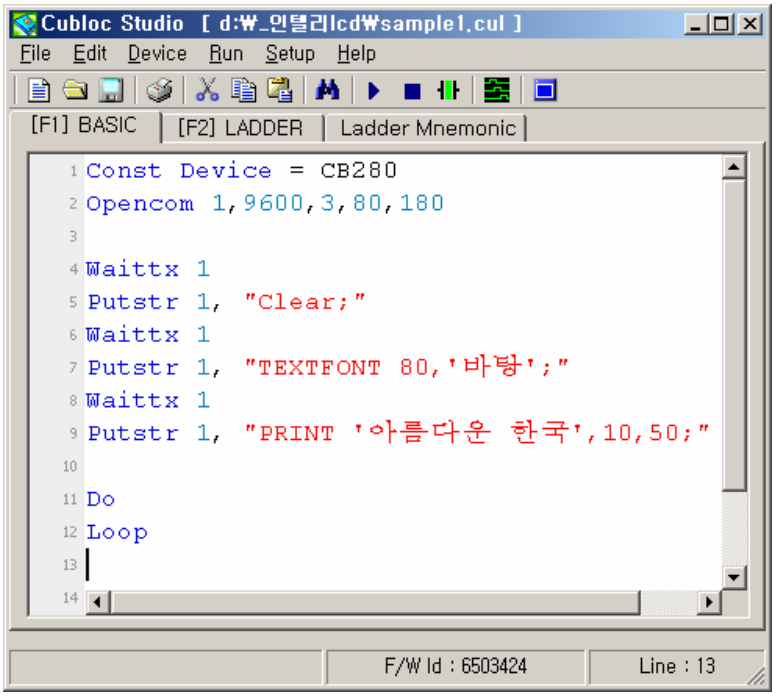
사용하시는 큐블록 코어모델에 따라 RS232채널이 다릅니다. CB280 모델의 경우 RS232채널 0또는 1을 사용할 수 있습니다. CB405의 경우 채널0부터 3까지 사용할 수 있습니다.



이 프로그램은 아주 간단한 예제 프로그램입니다. 인텔리 LCD화면에 "아름다운 한국"이라는 글자를 표시합니다.

이 프로그램을 보면 인텔리LCD용 커맨드 앞에 @ (골뱅이)를 붙여 주는 것을 보실 수 있습니다. 사실 @ (골뱅이)는 Putstr 명령어와 같습니다. 매번 Putstr 명령어를 써주는 불편함을 덜어주기 위해 @(골뱅이)를 사용해 주고 있습니다.

다음은 앞의 프로그램과 같은 동작을 하는 프로그램입니다.



@를 사용하지 않는다면, RS232데이터를 송신해주는 Putstr 명령과, 송신버퍼 데이터가 모두 송신될 때까지 기다려주는 Waittx 명령어를 매번 써주어야하는 불편함이 있습니다.

다음은 인텔리 LCD에 글자를 순차적으로 표시해주는 프로그램입니다.

```
Const Device = CB280
Dim ai As Integer
Dim x1 As Integer, x2 As Integer, y1 As Integer, y2 As Integer
Dim fname As String * 20, Str As String * 90
'Set ilcd 1
Const waittime = 5000
Opencom 1,9600,3,80,180

Do
@ "Clear;"

‘ sd card에 미리 저장해둔 3개의 bmp파일을 표시합니다.

@ "DrawImage '05.bmp';"
Wait waittime
@ "DrawImage '07.bmp';"
Wait waittime
@ "DrawImage '08.bmp';"


Wait waittime
xxx1:
@ "Clear;"
@ "Textpos 10,5;"
```

```
fname = "돋움"
Str = "아름다운 우리나라 Korea 1234"
y1 = 8
x1 = 5
Gosub strprint

y1 = 16
Gosub strprint

y1 = 24
Gosub strprint

Str = "아름다운 Korea 1234"
y1 = 48
Gosub strprint

y1 = 60
Gosub strprint

y1 = 80
Gosub strprint

Str = "아름다운 Korea"
y1 = 100
x1 = x1 + 10
Gosub strprint

Wait waittime
'-------------------------------------------
@ "Clear;"
@ "Textpos 10,5;"
fname = "바탕"
Str = "아름다운 우리나라 Korea 1234"
y1 = 8
x1 = 5
Gosub strprint

y1 = 16
Gosub strprint

y1 = 24
Gosub strprint

Str = "아름다운 Korea 1234"
y1 = 48
Gosub strprint

y1 = 60
Gosub strprint

y1 = 80
Gosub strprint

Str = "아름다운 Korea"
y1 = 100
x1 = x1 + 10
Gosub strprint

Wait waittime
'-------------------------------------------
@ "Clear;"
```

```
@ "Textpos 10,5;"
fname = "HY견고딕"
Str = "아름다운 우리나라 Korea 1234"
y1 = 8
x1 = 5
Gosub strprint

y1 = 16
Gosub strprint

y1 = 24
Gosub strprint

Str = "아름다운 Korea 1234"
y1 = 48
Gosub strprint

y1 = 60
Gosub strprint

y1 = 80
Gosub strprint

Str = "아름다운 Korea"
y1 = 100
x1 = x1 + 10
Gosub strprint

Wait waittime
'-----------------------------------------
@ "Clear;"
@ "Textpos 10,35;"
@ "TEXTFONT 280,'HY견고딕';"
@ "PRINT '한국';"



Wait WAITTIME
Loop
Do
Loop

strprint:

@ "Textpos 10,",Dp(x1),";"
@ "TEXTFONT ",Dp(y1),",'",fname,"';"
x1 = x1 + y1 + 15
@ "Print '",Str,"';"
Return
```

Tips

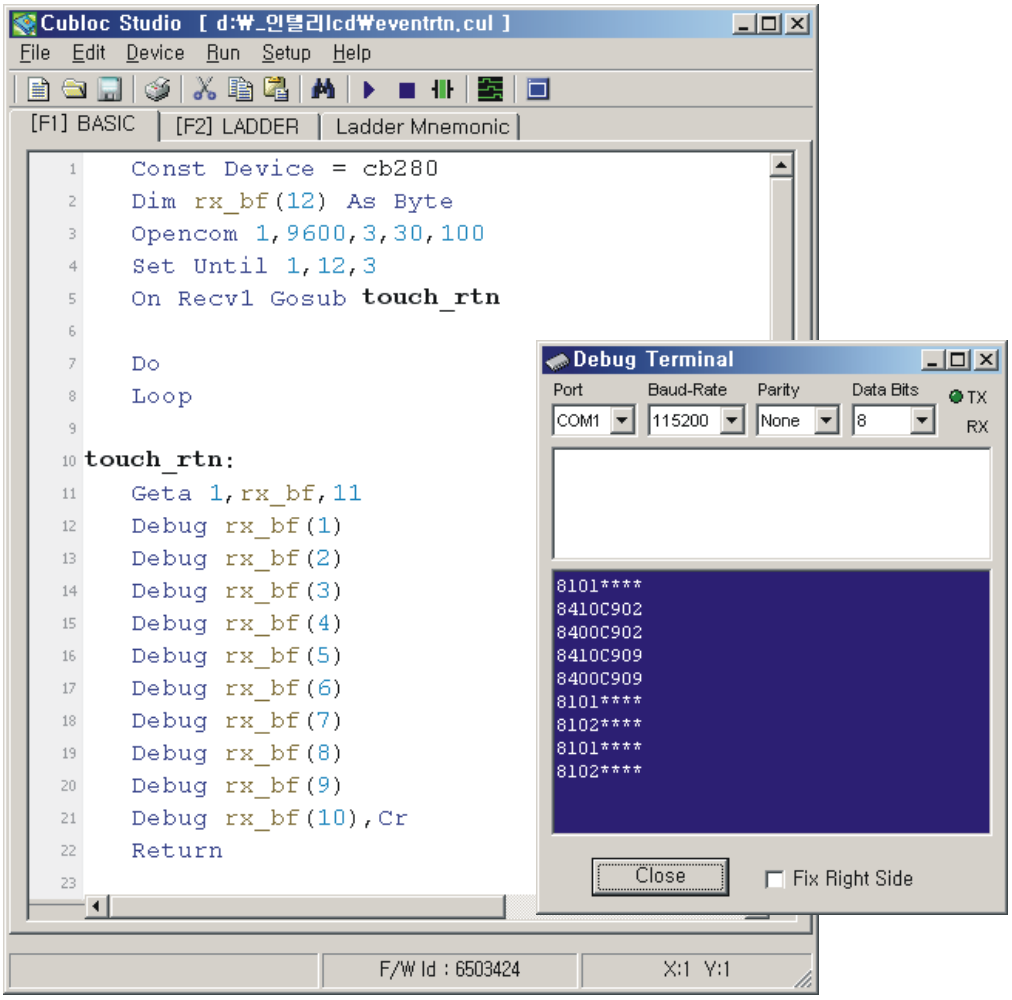
큐블록의 변수값을 IntelliLCD커맨드에 포함시키고 싶을때는 다음과 같이 기술합니다.

```
@ "Textpos 10,",Dp(x1),";"
```

Dp함수를 쓰면, 어떤 변수를 10진 문자열로 바꾸어줍니다.

5-3. 큐블록에서 이벤트수신 방법

큐블록에서는 아스키 수신모드를 권장합니다. IntelliLCD의 초기 디폴트상태가 ASCII상태이므로, 특별히 바꾸실 것은 없습니다. 다음과 같이 코드를 기술해주면, 종료코드 (03h)가 올때마다 인터럽이 발생해서 수신루틴을 호출하게 됩니다. 다음은 이벤트를 수신하여 Debug터미널에 표시해주는 프로그램입니다.



1번째 데이터는 무조건 02h 가 옵니다. 2번째와 3번째 데이터를 읽어서 이벤트의 종류를 판단합니다. (IF문 또는 Select Case문을 사용). 그 뒤에 있는 데이터를 읽어서 이벤트의 상태를 판단합니다.

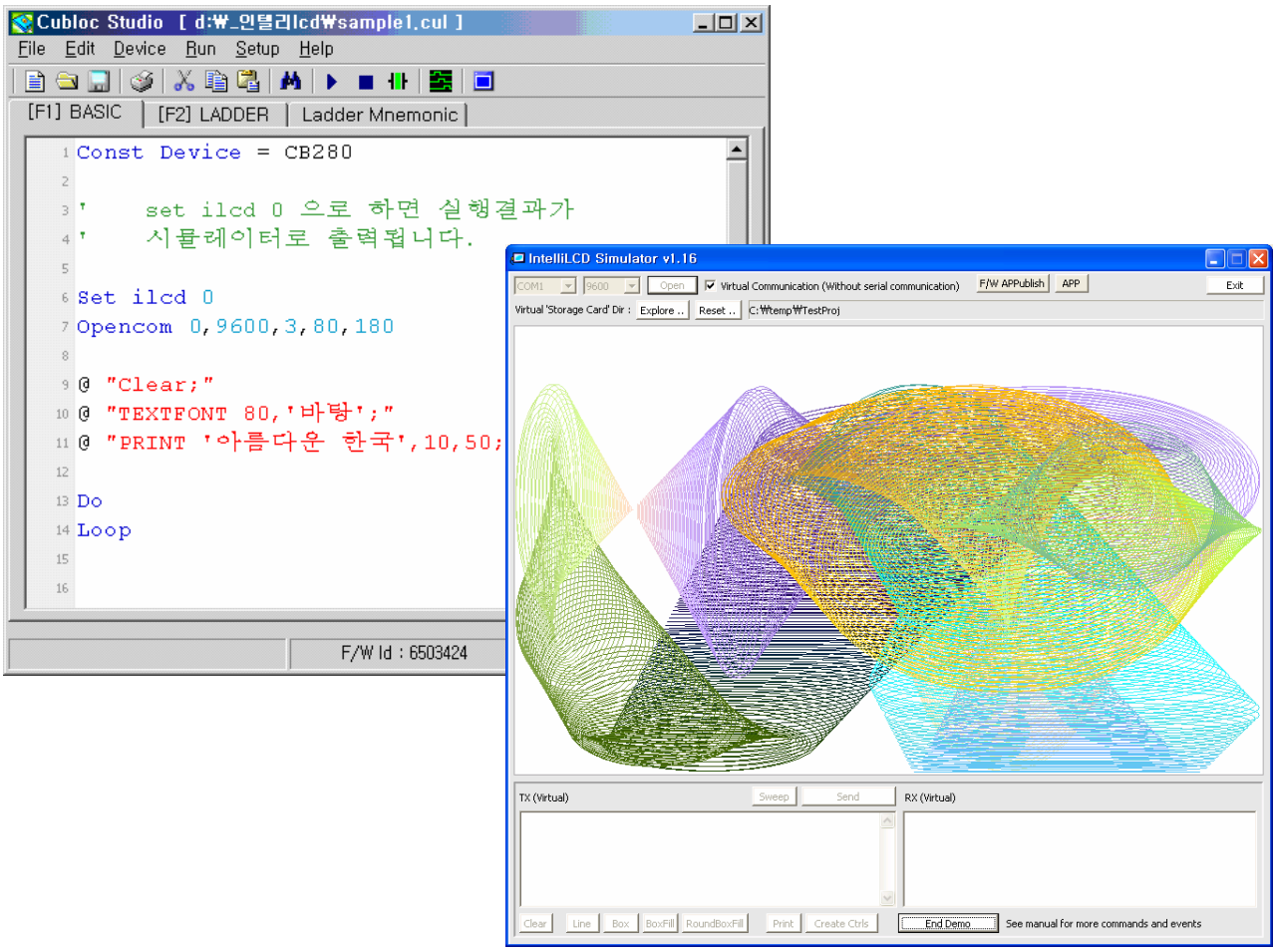
제6장 인텔리 LCD 시뮬레이터

인텔리 LCD 시뮬레이터는 CUBLOC Studio 내에서 실행시킬 수 있는 LCD 에뮬레이터입니다.

Set iLcd 명령은 인텔리 LCD 커맨드를 전송할 RS232채널을 설정하는 명령어라고 설명하였습니다.
채널0은 PC와 연결되어있는 RS232 를 의미합니다. 따라서 Set iLCD 0 이라고 설정한다면, 인텔리 LCD용 커맨드가 PC로 전송 되는 것을 의미합니다.

이렇게 한다면, PC에서 “인텔리LCD 시뮬레이터”가 실행됩니다.

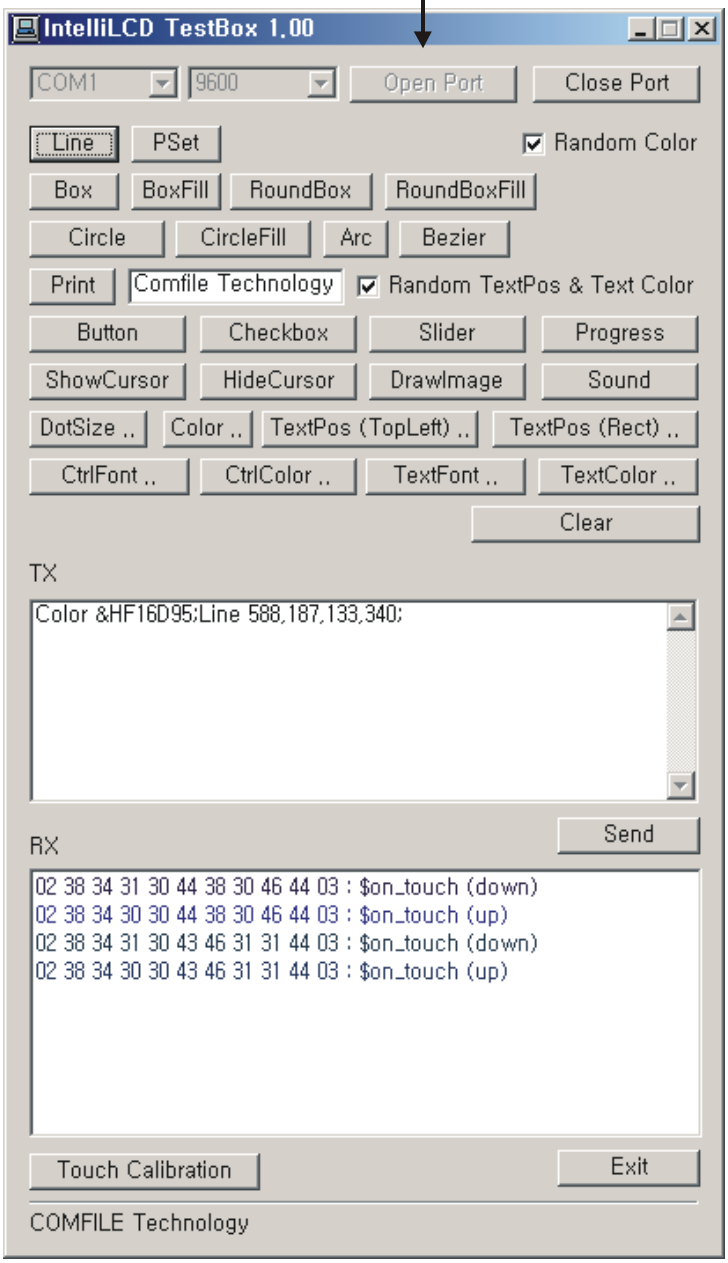
여러분이 인텔리 LCD 제품을 가지고 있지 않아도 인텔리LCD의 사용하는것과 똑 같은 효과를 여러분의 PC에서 미리 볼 수 있습니다.
인텔리 LCD 시뮬레이터를 사용할 때에는 Debug명령어를 사용할 수 없습니다. Set Debug Off 명령을 써서, 모든 Debug명령어를 실행하지 못하도록 해주어야 합니다.



주의: 시뮬레이터의 표시속도는 IntelliLCD 보다 빠릅니다. 일반적인 PC의 하드웨어와 IntelliLCD 의 하드웨어상 차이로 인한 것입니다. 시뮬레이터에서는 표시 속도 이외에도 다른 차이점이 있을 수 있다는 것을 염두하시고 사용하시기 바랍니다.

제7장 인텔리 LCD TestBox

IntelliLCD를 PC와 연결한상태에서 IntelliLCD의 동작상태를 점검하거나, 기능을 테스트할 수 있는 유틸리티 프로그램입니다. www.comfile.co.kr에서 무료로 다운로드 하실 수 있습니다. PC와 IntelliLCD는 1:1 시리얼 케이블로 연결하십시오. PC의 Com포트를 확인하시고 접속하시기 바랍니다. TestBox프로그램을 실행시키면 다음과 같은 화면이 표시됩니다.



PC의 COM PORT와 보레이트를 선택하신 뒤 Open Port 버튼을 누르면 사용가능한 상태가 됩니다.

이곳에 있는 버튼을 누르면, 해당 기능의 실행화면을 IntelliLCD 화면상에서 보실 수 있습니다.

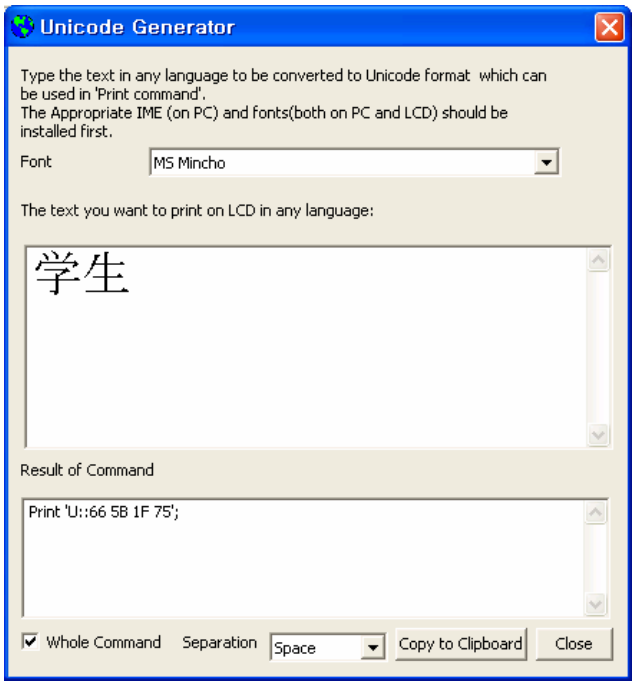
PC에서 IntelliLCD로 전송한 커맨드의 구체적인 내용을 확인할 수 있는 창입니다.

IntelliLCD에서 이벤트가 발생하였을 경우 이벤트의 내용을 표시해주는 창입니다.

TestBox 프로그램은 IntelliLCD의 이상유무를 점검할 때도 유용하게 사용할 수 있습니다.

제8장 인텔리 Unicode Generator

Unicode Generator는 LCD에 다국어를 표현하기 위한 **Print 커맨드**를 작성할 때 도움을 주는 PC용 프로그램입니다. 직접 원하는 각국 언어별 텍스트를 입력해보고 결과 커맨드를 얻을 수 있습니다.



Unicode Generator를 제대로 실행하려면 준비 작업을 해야 합니다.
첫째는 폰트 설치이고 두번째는 IME 설치입니다.

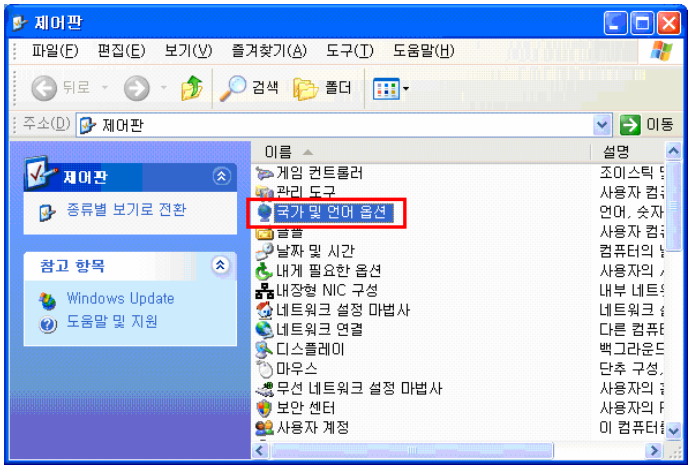
8-1. 폰트 설치

1. PC측 설치
   TTF또는 TTC 확장자를 가진 폰트 파일(예: msmincho.ttf)을 Windows/Fonts 폴더에 복사해 넣으십시오.

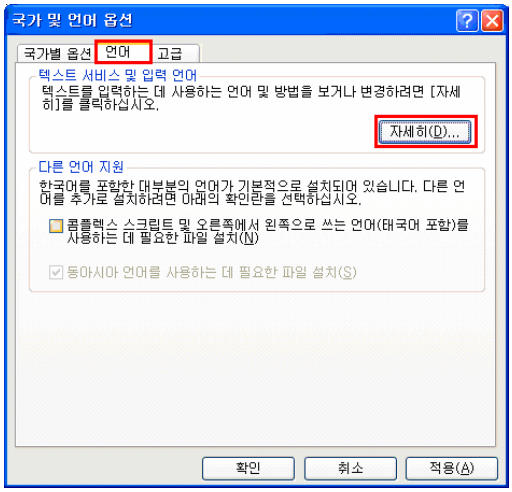
2. IntelliLCD측 설치 (PC상에서 Unicode Generator를 실행해보기 위해서는 불필요)
   PC에서 쓰는 동일한 TTF,TTC 파일을 IntelliLCD의 Storage Card/Fonts 폴더에 복사한 후 재부팅합니다. 자세한 내용은 <5-5. 사용자 폰트 추가> 항목을 참조하세요.

8-2. IME 설치

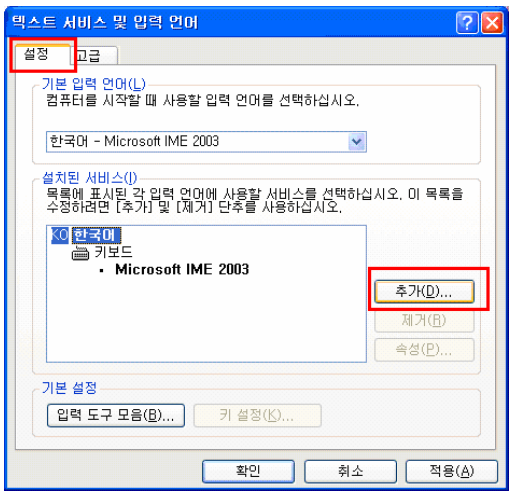
1. 제어판에서 <국가 및 언어 옵션>을 클릭합니다.



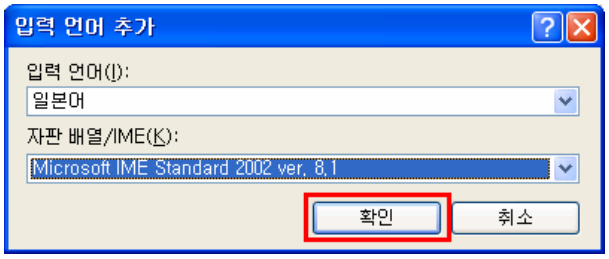
2. [언어] 탭에서 [자세히] 버튼을 누릅니다.



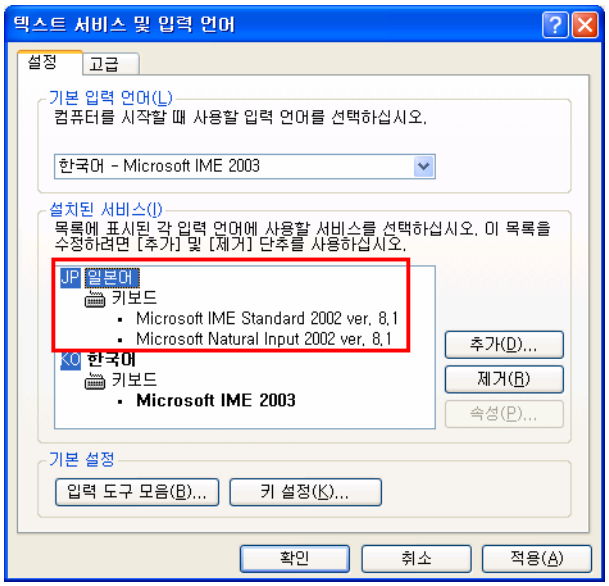
3. [설정] 탭에서 [추가] 버튼을 누릅니다.



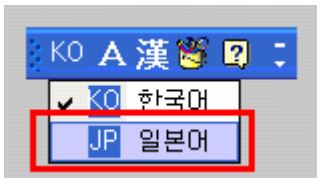
4. 원하는 언어를 선택한 후 [확인] 버튼을 누릅니다.



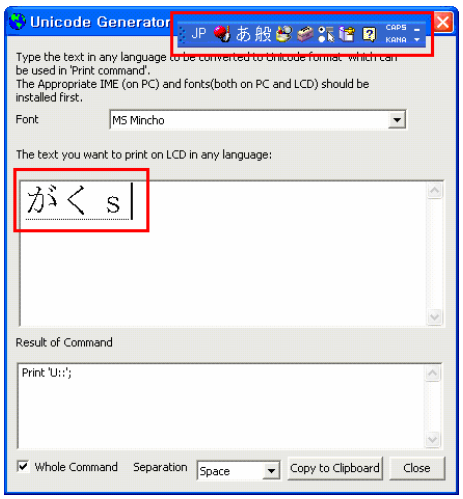
5. 원하는 언어의 입력도구가 추가된 것을 확인합니다.



6. Unicode Generator를 실행시킨 후, 원하는 언어의 입력도구를 선택합니다.

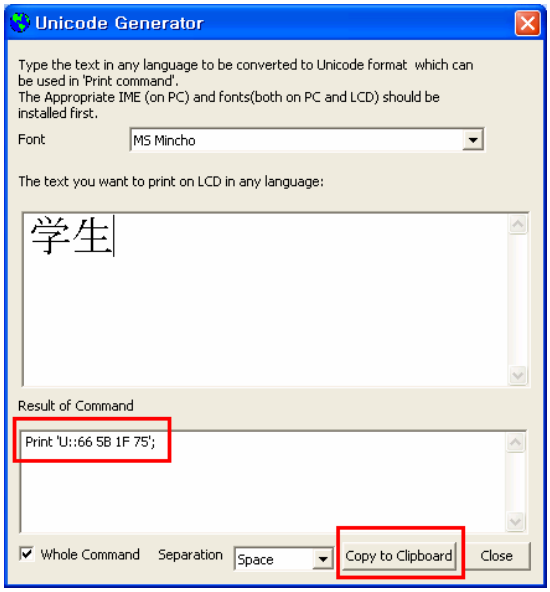


7. 문구를 입력합니다.



8. 결과창에 결과 커맨드 코드가 출력되는 것을 확인합니다.

[Copy to Clipboard] 버튼을 눌러 클립보드에 복사할 수도 있습니다.



제9장 시스템 변수

SetVar 커맨드로 변수를 인위적으로 세팅하는 것과 다르게, 시스템이 자연적으로 세팅하는 변수를 시스템 변수라고 합니다. 예를 들어 화면을 터치하면 #sys_touch_x 와 #sys_touch_y 값에 터치한 부분의 좌표 x, y값이 자연적으로 세팅됩니다.
시스템 변수는 #sys_ 라는 접두어로 시작하며 DelVar나 DelVarAll 로 삭제해도 지워지지 않습니다.

현재 지원되는 시스템 변수는 다음과 같습니다.

9-1. 기본 시스템 변수

변수명	내용
#sys_fw_ver	펌웨어 버전 (예: v1.94일 경우 194, v2.00일 경우 200)
#sys_screen_width	LCD의 화면 가로 픽셀 수 (예: 800)
#sys_screen_height	IntelliLCD의 화면 세로 픽셀 수 (예: 480)
#sys_tick_count	부팅 후 총 경과한 시간(초 단위)
#sys_untreated_buffer	처리되지 못하고 쌓여있는 커맨드들이 차지하는 버퍼 크기(byte)로서 약 1초에 한번씩 값이 자동 갱신 (이 값이 1MByte가 넘으면 커맨드 손실 시작)

9-2. 이벤트 관련 시스템 변수

관련 이벤트	변수명	내용
공통	#sys_ctrl_id	최근 유저가 조작한 컨트롤 ID
	#sys_down	최근 조작한 터치나 버튼, 체크박스 컨트롤, 스크린 영역의 눌림 상태 1이면 눌림, 0이면 뗌
$on_position	#sys_position	최근 유저가 조작한 슬라이더 컨트롤의 위치값
$on_touch	#sys_touch_x	최근 누르거나 뗀 터치 포인트의 x 좌표값
	#sys_touch_y	최근 누르거나 뗀 터치 포인트의 y 좌표값
$on_key_press	#sys_ascii	최근 눌린 키의 아스키 코드
$on_date	#sys_year	최근 GetDate 커맨드가 들어왔을 때의 연도
	#sys_month	최근 GetDate 커맨드가 들어왔을 때의 월
	#sys_day	최근 GetDate 커맨드가 들어왔을 때의 일
$on_time	#sys_hour	최근 GetTime 커맨드가 들어왔을 때의 시각
	#sys_minute	최근 GetTime 커맨드가 들어왔을 때의 분
	#sys_second	최근 GetTime 커맨드가 들어왔을 때의 초
$on_screen_touch	#sys_screen_id	최근 눌려진 스크린 ID
	#sys_area_id	최근 눌려진 스크린의 영역 ID

<끝>