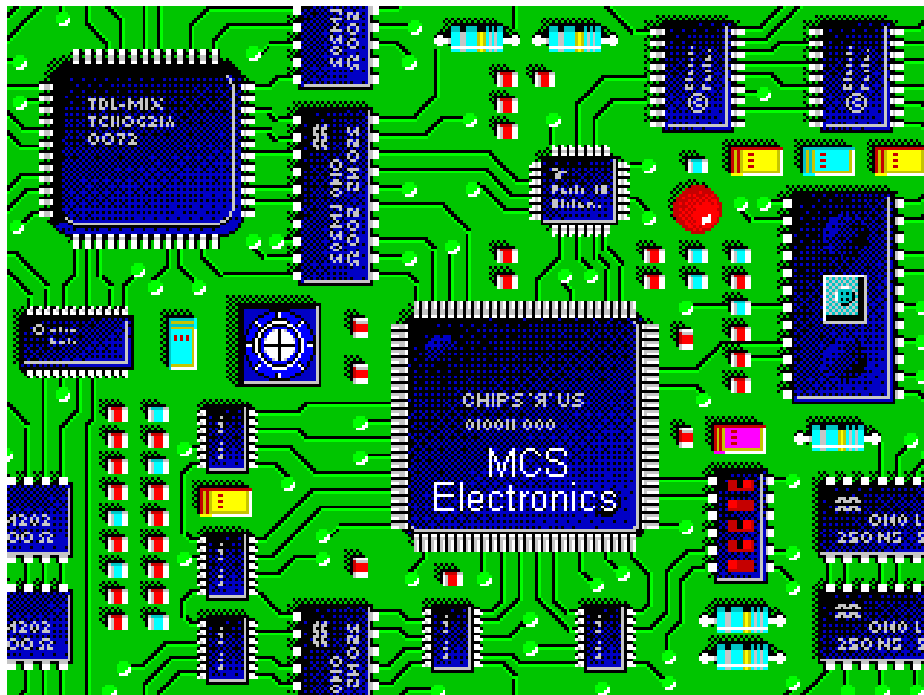


BASCOM 8051

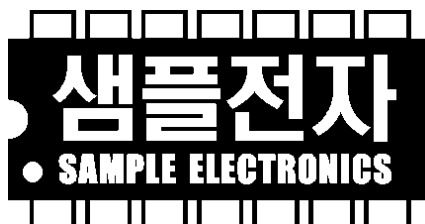
The Windows 8051 BASIC Compiler

2004 Jan 28

V 2.11



SN: _____



서울시 용산구 한강로 3가 16-9
전자랜드 본관 광장층 C-18호
TEL: (02) 701-8051
FAX: (02) 701-8058
sample@korea.com



WWW.SAMPLE.CO.KR

MCS Electronics may update this documentation without notice.

Products specification and usage may change accordingly.

MCS Electronics will not be liable for any mis information or errors found in this document.

All software provided with this product package is provided 'AS IS' without any warranty expressed or implied.

MCS Electronics will not be liable for any damages, costs or loss of profits arising from the usage of this product package.

No part of this document may be reproduced or transmitted in any form or by any means, Electronics or mechanical, including photocopying and recording, for any purpose without written permission of MCS Electronics.

© MCS Electronics. All rights reserved.

Thank you for using BASCOM-8051, the Windows BASIC compiler for the AT89C2051 and other 8051 family members.

This guide will help you to install BASCOM and explains the environment briefly.

If you have questions, remarks or suggestions please let us know.

You can contact us by sending an Email to 8051@mcselec.com

Our website is at <http://www.mcselec.com>

For info on updates : please read the readme.txt file installed into the BASCOM directory !!!

Regards,

Mark Alberts
MCS Electronics

Copyright MCS Electronics®

SAMPLE Electronics co. 샘플전자

WWW <http://www.SAMPLE.co.kr>

E-mail sample@korea.com

TEL 02-701-8051

FAX 02-701-8058

MCS Electronics 는 본 설명서의 내용을 사전 통지 없이 개정할 수 있습니다.
제품의 사양과 용도는 적절하게 수정될 수 있습니다.

MCS Electronics는 본 설명서의 부정확한 정보에 대한 책임을 갖지않습니다.

MCS Electronics는 본 프로그램의 사용 후에 발생하는 재정적손실이나, 위험에 대한 책임을 갖지 않습니다.

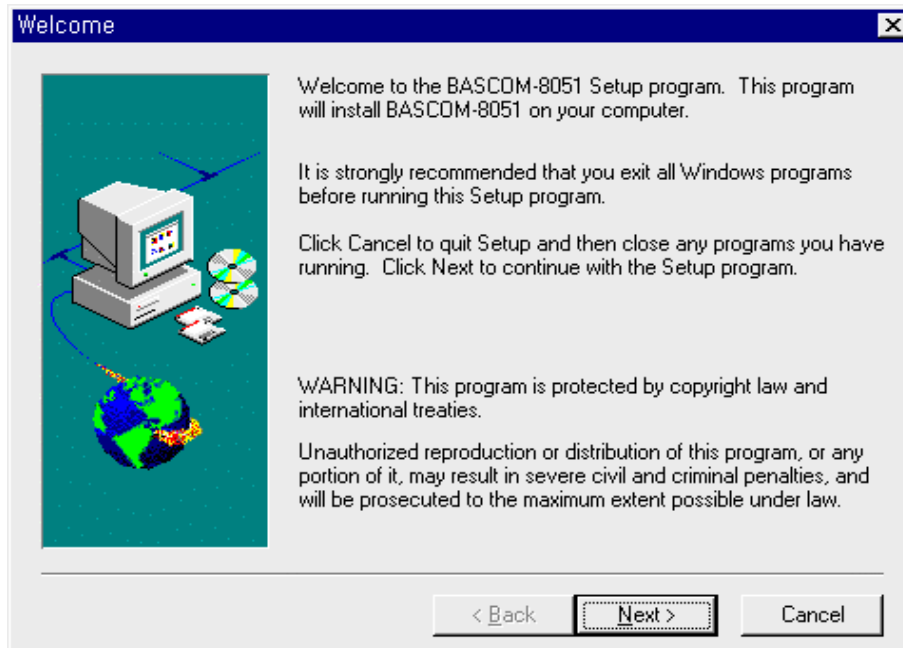
본설명서는 MCS Electronics 의 허가 없는 전자적, 기계적 또는 어떠한 형태로든 복사, 번역을 금합니다.

본 프로그램의 판권은 MCS Electronics 에 있습니다.

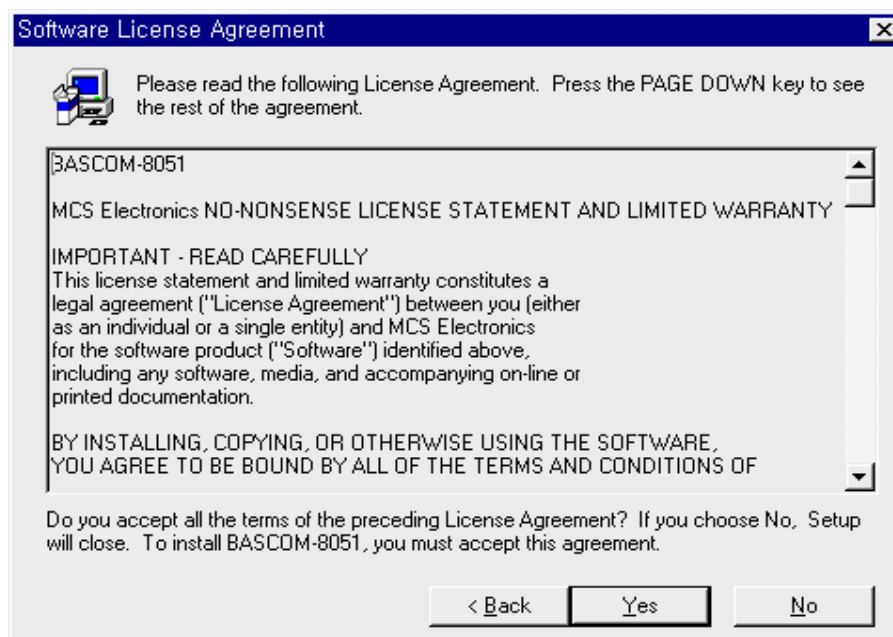
<http://www.mcselec.com>

Installing BASCOM-8051

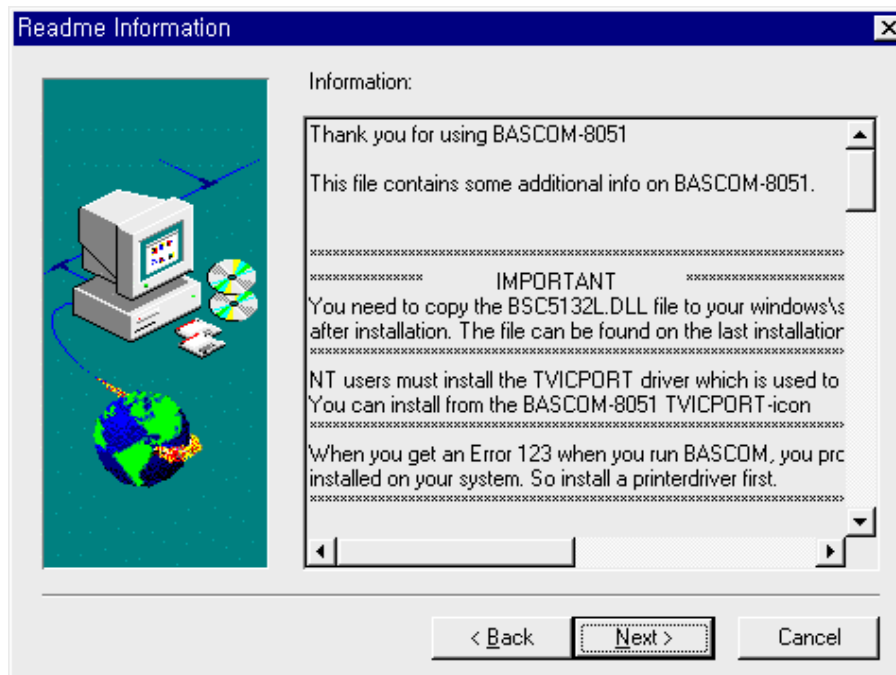
SETUP.EXE를 클릭하여 설치를 시작합니다.



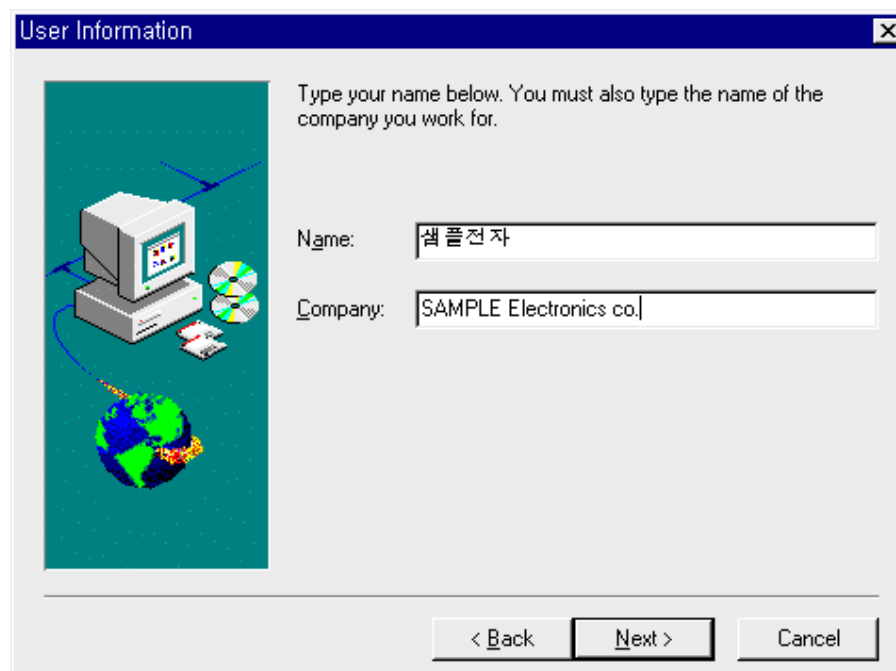
Next-button을 눌러 설치를 계속합니다. 아래의 창이 보일 것입니다 :



라이센스에 동의하면 Yes 버튼을 클릭합니다.

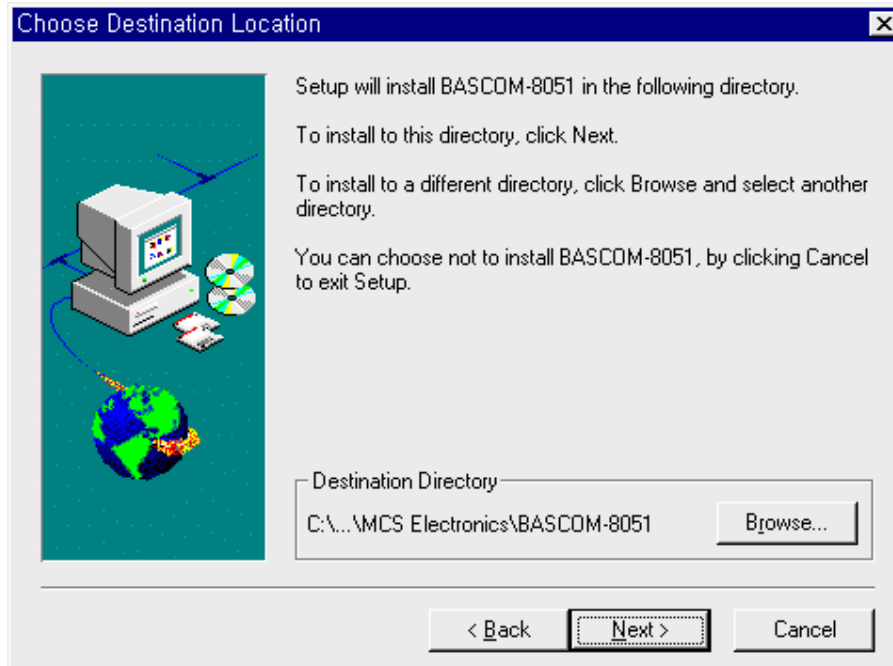


설치시의 참고 사항입니다. NEXT 버튼을 클릭합니다.



사용자 이름과 회사 이름을 입력합니다. Next-button을 눌러 설치를 계속합니다.

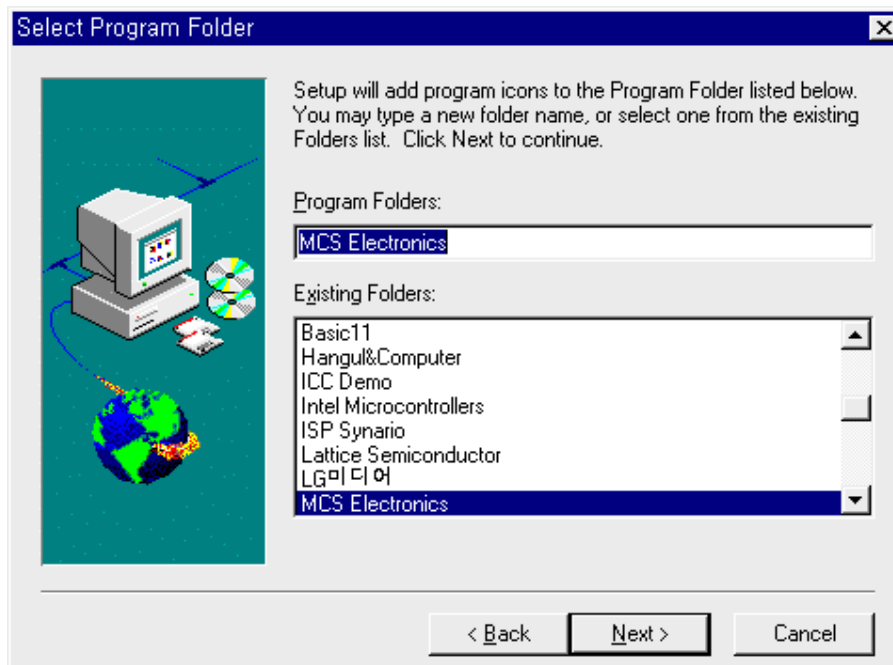
BASCOM 8051 이 설치될 디렉토리를 선택합니다.



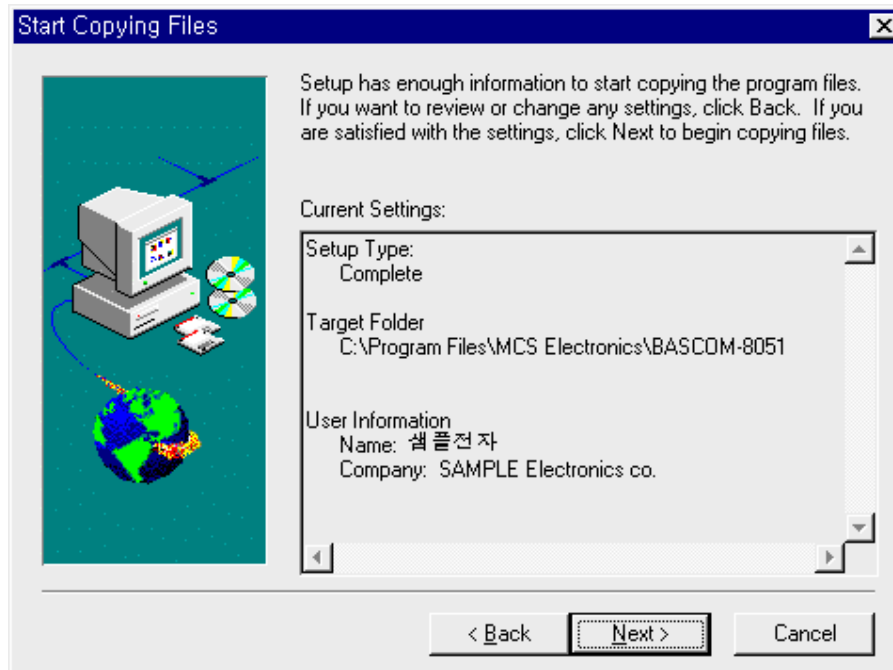
디렉토리 경로를 변경하기 위해서는 Browse 버튼을 눌러 선택합니다.
디폴트로 BASCOM 8051 은

C:\Program File\MCS Electronics\BASCOM-8051 에 인스톨될 것입니다.

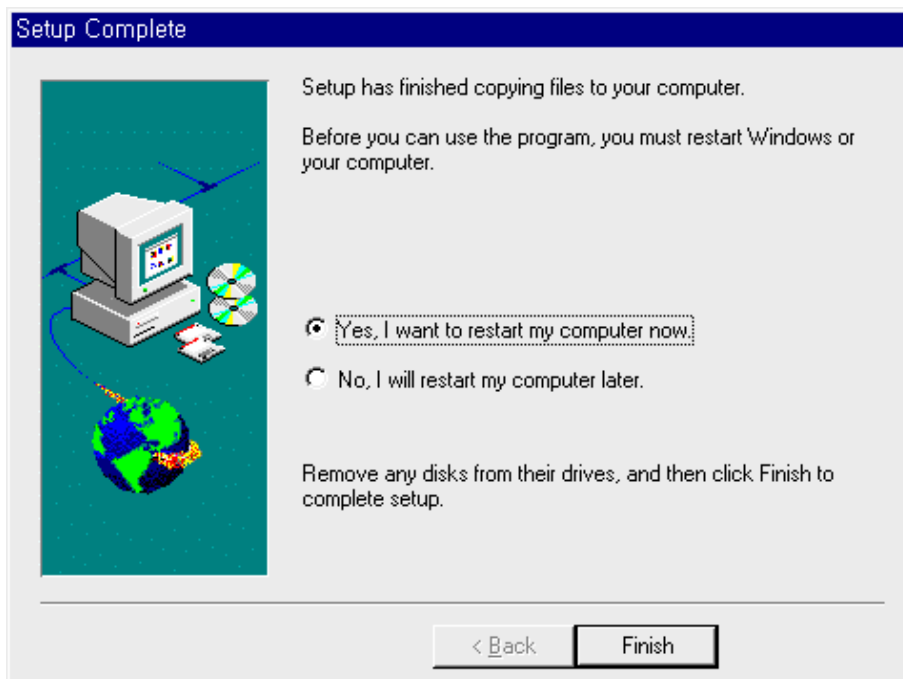
디렉토리를 선택한 후 Next-button을 클릭합니다.



이제 BASCOM 8051 아이콘이 어떤 프로그램 그룹에 위치되어야 하는지를 물어볼 것입니다.
디폴트로 MCS Electronics 라고 명명된 새로운 프로그램 그룹을 만들 것입니다.
그룹을 선택한 후 Next-button 을 눌러 설치를 계속합니다.
간단한 요약문의 나타나면, 변경이 필요한 경우 Back 해서 내용을 수정할 수 있습니다.



BASCOM 8051 를 계속 설치하기 위해 Next-button 을 클릭합니다.



인스톨이 완료되면 Finish-button 을 눌러 설치를 완료합니다.

참고) samples 이라고 명명된 폴더 에는 BASCOM 8051 의 샘플 파일이 모두 들어 있습니다.

***** **IMPORTANT** *****

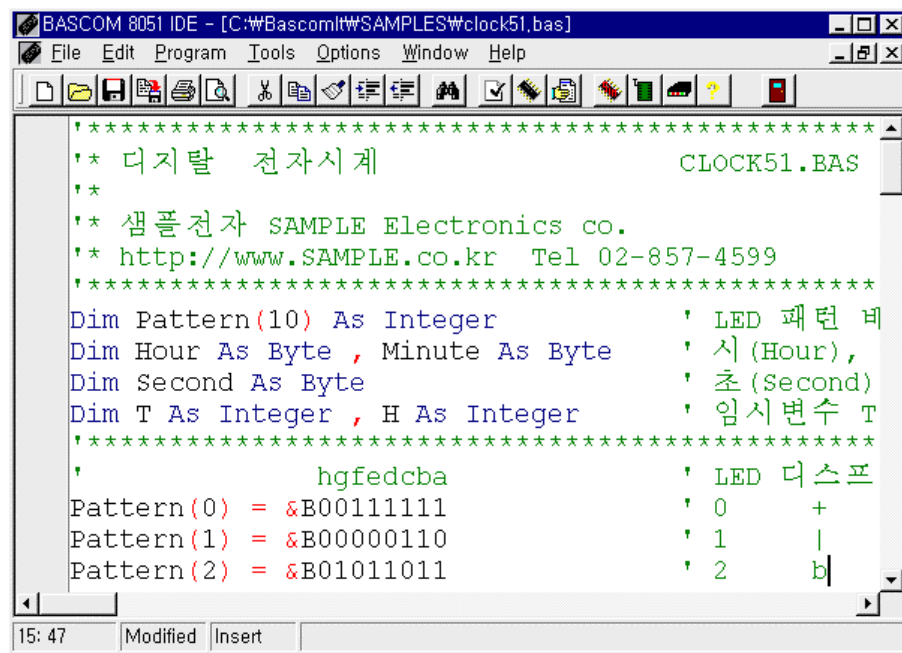
라이센스 파일은 **SETUP.EXE** 와 같은 폴더에 있어야 합니다. (샘플전자에서 제공하는 CD 에는 동일한 폴더에 들어있습니다)

BASCOM 이 설치되는 디폴트 폴더는 아래와 같습니다.

C:\Program Files\MCS Electronics\BASCOM-8051\

BASCOM 8051 실행하기

BASCOM 8051 아이콘을 더블 클릭하면, 다음의 윈도우가 나타날 것입니다.



마지막으로 사용한 파일이 보일 것입니다.

메뉴 옵션 설명

File New (새로운 파일 선택)



새로운 윈도우를 생성하며 이 곳에서 프로그램을 작성할 수 있습니다.

File Open (파일 열기)



디스크로 부터 기존에 제작된 파일을 열어볼 수 있습니다.

BASCOM 8051 은 작성된 파일을 기준 ASCII(아스키) 포맷으로 저장합니다. 그러므로 다른 에디터에서 작성된 프로그램을 불러올 경우 이 파일은 먼저 ASCII TEXT 포맷으로 저장 되어 있어야 합니다.

(참고)

BASCOM 8051 의 Envirenment 옵션에서 reformat 옵션을 선택 하여야 합니다. 다른 에디터에서 작성된 프로그램을 읽을 때 사용됩니다.

File Close (파일의 종료)



현재 프로그램을 종료합니다. 프로그램을 수정 하였거나 다시 포맷한 경우 '프로그램을 먼저 저장하세요 (Save the program first)' 라고 알릴 것입니다.

File Save (파일 저장)



프로그램을 디스크에 빠르게 저장할 수 있습니다. 만일 프로그램이 새로운 파일 옵션에서 작성된 것이면, 파일 이름을 먼저 입력하라고 알릴 것입니다. 다른 이름으로 저장하고자 할 경우는 Save As 옵션을 선택한 후 파일명을 기입하면 다른 이름으로 저장됩니다. 파일은 ASCII 파일로 저장되어야 합니다.

File Save As... (파일을 다른 이름으로 저장)



현재의 파일을 디스크에 저장할 수 있습니다. 프로그램이 저장되기 전에 파일명을 변경할 수 있습니다. 파일을 ASCII 파일로 저장되어야 합니다.

File Print Preview(프린트 프리뷰)



현재의 프로그램을 프린트하기 전에 스크린에 프린트 형태로 디스플레이 합니다.

File Print (프린트)



현재의 프로그램을 프린터에 프린트 합니다.

File Exit (종료)



BASCOM 8051을 종료합니다. 프로그램의 변경사항이 있었다면 저장할 것 인가를 묻는 윈도우가 나타납니다.

Edit Undo (실행 취소)



최종적으로 실행된 것을 취소합니다.
단축키는 CTRL+ Z 입니다.

Edit Redo (취소 재실행)



마지막으로 UNDO 된 것을 재실행 합니다.
단축키는 CTRL+ SHIFT+ Z 입니다.

Edit Cut (잘라내기)



선택영역을 절취하고 클립보드에 저장합니다.
단축키는 CTRL+ X 입니다.

Edit Copy (복사)



선택 영역을 클립보드에 저장합니다.
단축키는 CTRL+ C 입니다.

Edit Past (붙여넣기)



클립보드에 저장된 텍스트를 현재의 커서위치로 붙여넣습니다.
단축키는 CTRL + V 입니다.

Edit Find (찾기)



문자열을 검색합니다.
단축키는 F3 입니다.

Edit Find Next (다음 찾기)



마지막으로 검색하였던 문자열을 계속해서 검색합니다.
단축키는 F3입니다.

Edit Replace (단어교체)



문자열을 지정한 새 문자열로 교환합니다.
단축키는 CTRL+ R 입니다.

Edit Goto (커서 직접 이동)



라인 번호를 입력하여 커서가 직접 이동합니다.
단축키는 CTRL+ G 입니다.

Edit Toggle Bookmark

북마크를 설정하거나 해지합니다. 단축키는 CTRL+ K+ x 이며 x 는 1 부터 8 까지입니다.

Edit Goto Bookmark

설정된 북마크로 커서를 직접 이동합니다.
단축키는 CTRL+ Q+ x 이며 x 는 1 부터 8 까지 입니다.

Edit Indent Block



선택블록을 일정 공백을 두어 오른쪽으로 이동시킵니다.
단축키는 CTRL + SHIFT + I 입니다.

Edit Unindent Block



선택 블록을 왼쪽으로 이동시킵니다.
단축키는 CTRL+ SHIFT+ U 입니다.

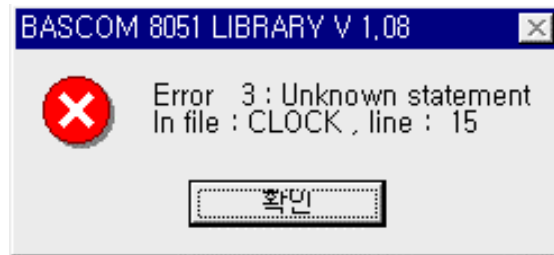
File Compile (Hex 파일 생성)



현재의 프로그램을 컴파일 합니다.
컴파일 되기 전에 현재의 프로그램은 자동적으로 저장됩니다.
컴파일러 설정 선택에 따라서는 아래의 파일이 생성될 것입니다.

File	설명
Xxx.BIN	바이너리 파일이며 마이크로 프로세서에 라이팅 합니다.
Xxx.DBG	시뮬레이터 동작시 필요한 디버그 파일입니다.
Xxx.MAP	시뮬레이터 동작시 필요한 메모리맵 파일입니다.
Xxx.HEX	INTEL 사 방식의 HEX 파일입니다. 프로그램 장비에서 필요합니다.
Xxx.ERR	컴파일시 나타난 에러의 내용을 가지고 있습니다. (에러가 있을 때만 만들어집니다)
Xxx.RPT	컴파일결과로 나타난 각종 정보가 들어있습니다.

에러가 발생되면 에러 경고 메시지가 표시될 것입니다. 예) :



확인 버튼을 누르면 커서(cursor)는 에러가 발생된 곳에 위치할 것입니다.

Program Syntax Check (문법 확인)

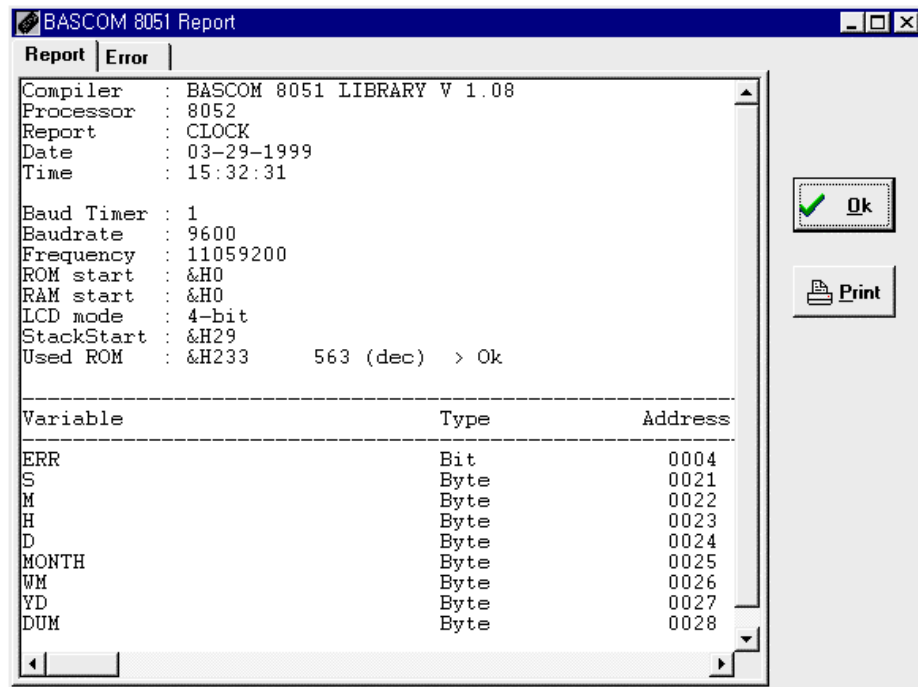


프로그램에서 문법을 체크합니다. 에러가 발견되면 에러 파일만 생성됩니다. 문법확인 단축키는 CTRL+F7 입니다.

Program Show result (결과 확인)



컴파일 결과를 보여 줍니다. Option->Compiler->Output 에서 출력될 파일을 선택합니다. 단축키는 CTRL+W 입니다. 컴파일 결과를 확인할 수 있습니다.



컴파일러 옵션에서 레포트 파일이 생성되도록 선택되어 있어야 합니다.

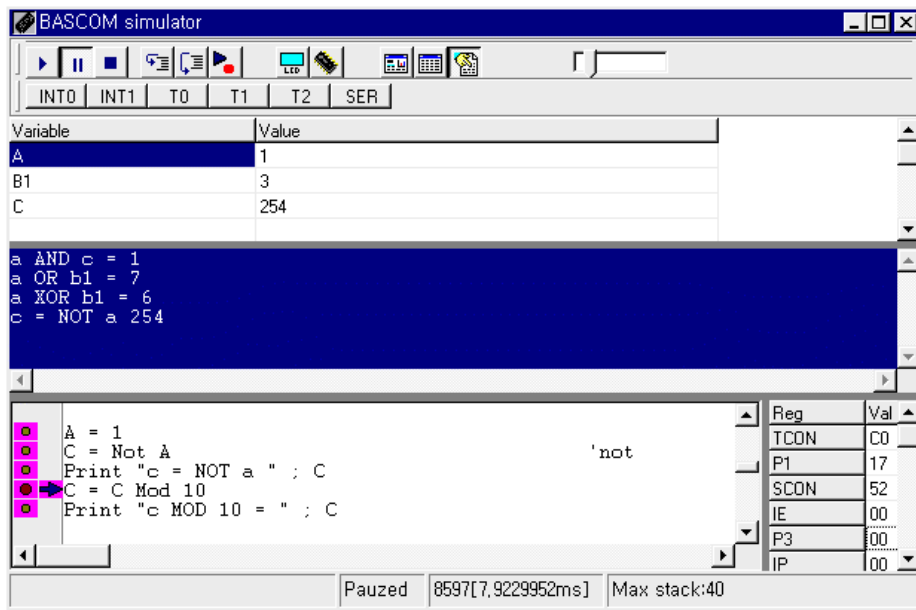
컴파일 결과 보고서와 예러가 표시됩니다.

프린트 아이콘을 누르면 프린트 되며, Ok 아이콘을 누르면 편집화면으로 돌아갑니다.

레포트 파일

Info	Description
Compiler	컴파일러 버전을 표시합니다.
Processor	선택한 타겟 프로세서가 표시됩니다. (.DAT 파일)
Date and time	컴파일 날짜와 시간입니다.
Baud Timer	BAUD RATE를 발생하기 위한 타이머 입니다. 0 은 타이머를 사용하지 않은 경우 표시됩니다.
Baudrate and frequency	마이크로 프로세서에서 사용된 통신속도(Baud Rate)와 크리스탈 주파수 입니다. 이 정보는 RS232 관련 명령에서 중요한 참고사항이 됩니다. 메타커맨드 \$CRYSTAL 과 \$BAUD 를 사용하여 정확한 Baudrate를 표시합니다.
ROM start	ROM 코드의 스타트 어드레스 입니다.(0 이 내정 값입니다.)
RAM start	내부RAM의 시작 어드레스 입니다.(0 이 내정 값입니다.)
LCD Mode	사용된 LCD 의 모드입니다. 4 비트나 8 비트 모드가 표시됩니다.
Stack start	스택의 출발 위치입니다. 스택의 아래쪽은 변수 영역입니다. 스택은 기계어루틴의 호출에 의하여 증가됩니다.
Stack level	가장 깊이 스택이 사용되었을 때의 바이트 수입니다. 이 값은 개략적인 값이며 정확한 것은 시뮬레이터를 동작하여 알 수 있습니다.
Used ROM	바이너리 파일의 길이를 나타냅니다.

Program Simulate (시뮬레이트)



작성된 프로그램을 시뮬레이션합니다.

시뮬레이터의 윈도우의 아이콘

	프로그램 실행 (RUN)		프로그램 일시 정지(PAUSE)		프로그램 정지 (STOP)
	1 라인 실행 (STEP)		1 라인오버 모드 실행(STEP OVER)		선택된 라인까지 실행(RUN)
	LCD 시뮬레이션		하드웨어 시뮬레이션		
	소스파일 변경을 금지하거나 허락		8051 배부 메모리 표시		변수 리프레쉬
	인터럽트 0 (P3.2)		인터럽트 1 (P3.3)		RS-232 인터럽트
	타이머/카운터 0		타이머/카운터 1		타이머/카운터 2

실행 아이콘을 누르면 시뮬레이션을 시작합니다.

PRINT 문에 의한 출력은 윈도우 중간의 청색영역에서 표시됩니다.

INPUT 명령이 실행되면 키보드에 의한 데이터 입력이 필요합니다.

입력하기 전에 청색 윈도우가 활성화 상태로 있는지 확인 하여야 합니다.

커서가 반짝거리고 있는 상태에서 키인 하여야 입력되며 Enter키에 의하여 입력 종료됩니다.

일시정지(Puse) 버튼을 실행하면 시뮬레이션을 멈춥니다.

실행 버튼을 누르면 시뮬레이션이 재개됩니다.(중지한 곳부터 실행)

정지 버튼을 누르면 시뮬레이션을 종료합니다.

스텝 버튼을 누르면 1 라인씩 시뮬레이션 합니다.

소스 윈도우에서 어떤 라인에 커서를 놓고 브레이크 포인트 버튼(F9)을 누르면 프로그램의 실행이 선택한 라인에서 멈춥니다.

관찰하고 싶은 변수가 있으면 변수목록표의 빈칸을 클릭하여 변수이름을 입력합니다.

INT0, INT1, T0, T1 과 SER 버튼을 클릭하면 인터럽트를 걸 수가 있습니다.

이것은 소프트웨어적인 인터럽트 이기 때문에 하드웨어 인터럽트가 발생하는 것은 아닙니다.

이 버튼을 클릭하는 순간에 인터럽트가 발생하는 것 입니다.

칩이 추가적인 인터럽트를 가지면 추가적인 버튼이 디스플레이 될 것입니다.

하드웨어 시뮬레이션 아이콘을 클릭하면 하드웨어 시뮬레이션이 가능합니다만 이 경우 추가적인

외부장치로 하드웨어 시뮬레이터를 부착 하여야 합니다.

하드웨어 시뮬레이터는 Port1 과 Port3 를 시뮬레이션 합니다. 이것은 칩을 프로그램 하지 않고 하드웨어 테스트가 가능하게 합니다.

단순히 Port 에 데이터를 쓰거나 읽는 것만 가능합니다. 인터럽트 처리에 대한 하드웨어 시뮬레이션은 지원하지 않습니다.

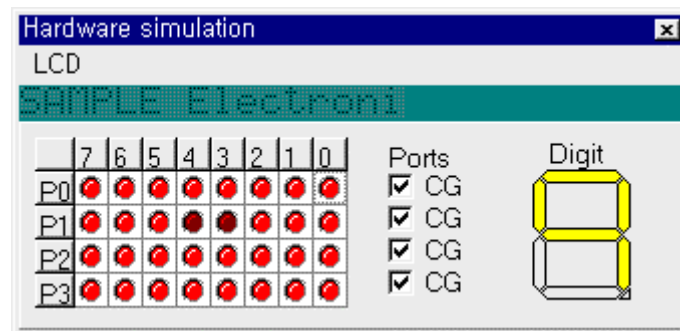
자세한 것은 MCS Electronics 사의 BASCOM 8051 하드웨어 시뮬레이터를 참고하시기 바랍니다.

LCD 버튼을 클릭하면 다음과 같은 윈도우가 생깁니다.

LCD 윈도우는 16 문자 1 라인 구성의 LCD 를 시뮬레이션합니다.

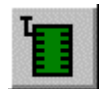
Port0, Port1, Port2, Port3 에 대한 출력상태가 LED 형태로 표시됩니다.

LED 를 클릭하면 On 또는 Off 됩니다.

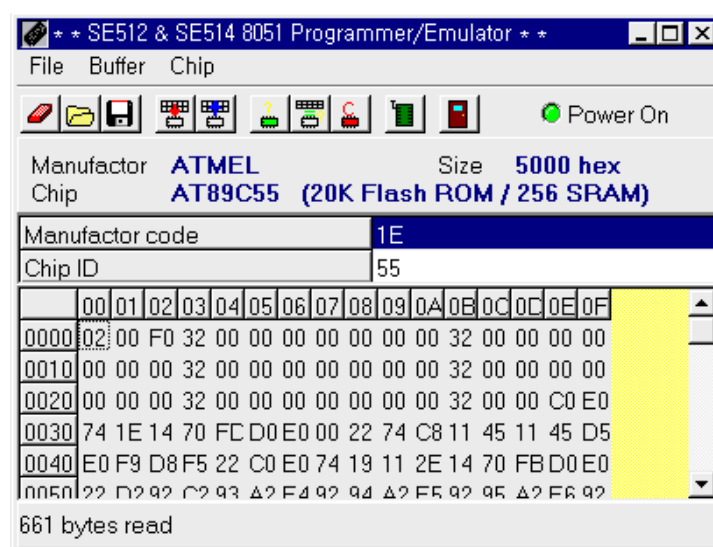


시뮬레이션 실행하는 동안 F9 키에 의하여 브레이크 포인트를 설정할 수 있습니다. 브레이크 포인트가 설정된 라인은 진한 적색 포인트로 표시됩니다. 화살표는 현재 실행된 라인위치를 나타냅니다.

Program Send to chip (칩 프로그램)



컴파일 출력의 BIN/HEX 파일을 8051 칩에 프로그램 합니다. 사용되는 프로그래머의 종류는 Option -> Programmer 에서 선택합니다.

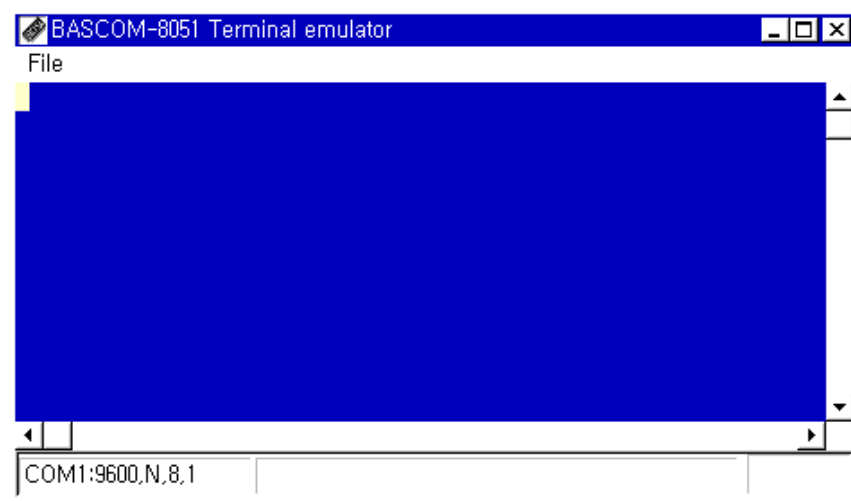


	BASCOM 버퍼를 클리어		HEX / BIN 파일 오른		HEX / BIN 파일 저장
--	-------------------	--	--------------------	--	-----------------

	FLASH 메모리 프로그램		FLASH 메모리 데이타 리드		FLASH 메모리 블랭크 테스트
	버퍼와 FLASH 메모리의 내용비교		FLASH 메모리 이레이즈		오토 프로그램
	프로그래머 사용종료		SE-514/512의 전원 On/Off		

File Terminal Emulator (터미널 에뮬레이터)

에뮬레이터는 마이크로 프로세서와 PC 간의 RS-232 시리얼 인터페이스를 실험하는 것입니다. 다음과 같은 창이 나타납니다.



타이핑 하는 데이터와 마이크로프로세서에서 나오는 데이터가 동일한 윈도우에서 표시됩니다. 마이크로 프로세서와 컴퓨터의 Baud Rate를 같이 설정합니다. 컴파일러 설정 메뉴에서 4800으로 설정되어 있다면 통신속도를 4800 으로 설정 하여야 합니다. 설정된 Baud Rate 는 컴파일 레포트 파일에 나타납니다.

File Send

현재 프로그램의 컴파일 출력된 HEX 포맷 파일 RS-232 시리얼을 통하여 외부로 전송합니다. 이 옵션은 모니터 프로그램으로 전송하는 것이 가능합니다.

File Escape

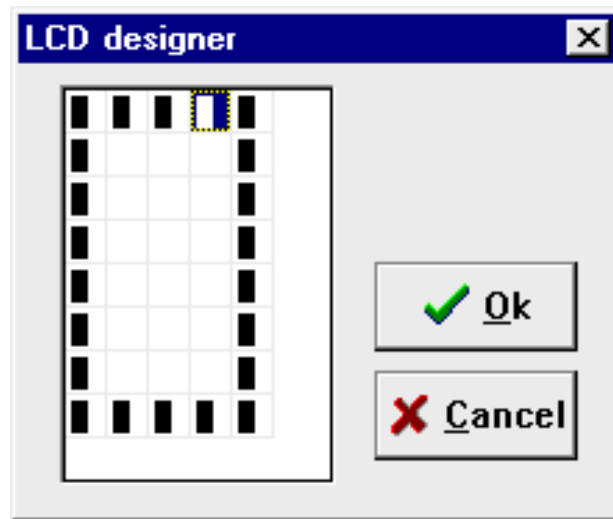
프로그램 전송을 중지 합니다.

File Exit

터미널 에뮬레이터의 사용을 종료합니다.

Edit LCD designer(LCD패턴 설계)

LCD 디스플레이에서 사용자 정의 패턴을 설계합니다.



LCD 매트릭스는 7X5 의 형태입니다. 맨 아래 줄은 커서로 예정되어있습니다.

선택한 위치를 마우스의 왼쪽 버튼으로 더블 클릭합니다.

선택된 곳을 다시 더블클릭하면 선택이 취소됩니다.

OK 버튼을 클릭하면 에디터의 커서위치에 계산된 수치가 "Deflcdchar" 문과 함께 자동으로 입력됩니다.

LCD 문자설계의 명령문은 다음과 같은 형식입니다 ? 는 0 부터 7 까지의 숫자입니다.

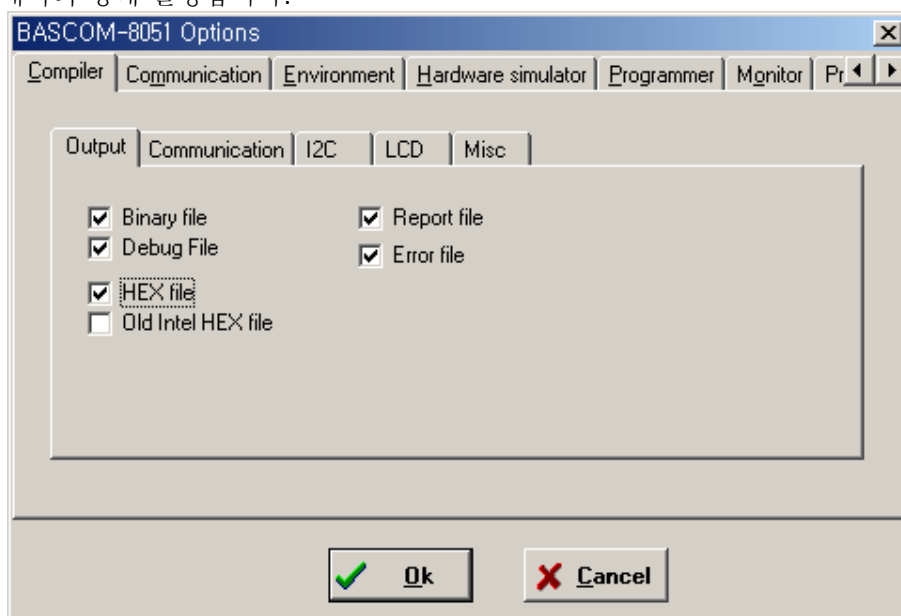
Deflcdchar ?,1,2,3,4,5,6,7,8

You must replace the ?-sign with a character number ranging from 0-7.

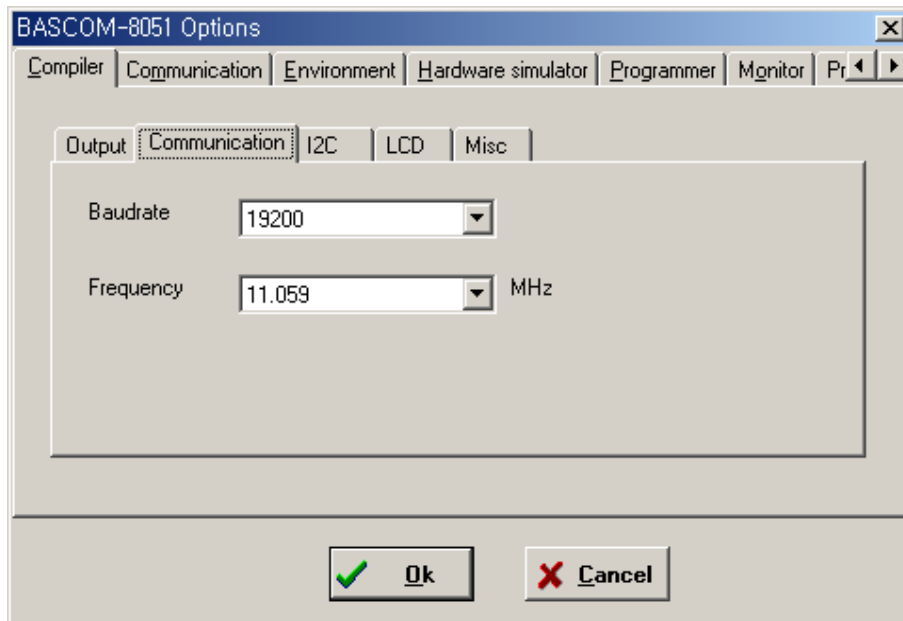
자세한것은 도움말에서 Deflcdchar를 참고하시기 바랍니다.

Options Compiler (컴파일러 옵션)

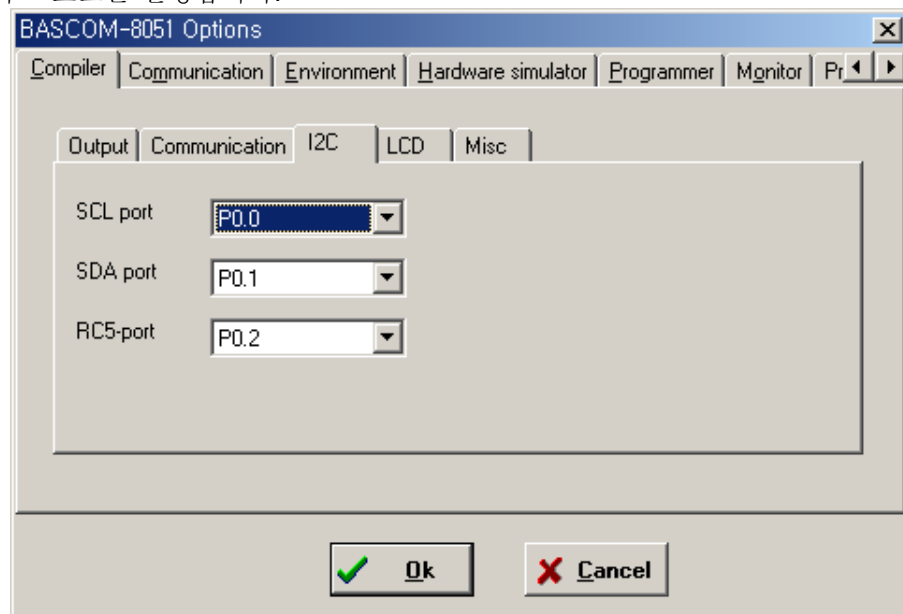
컴파일 출력데이터 형태 설정합니다.



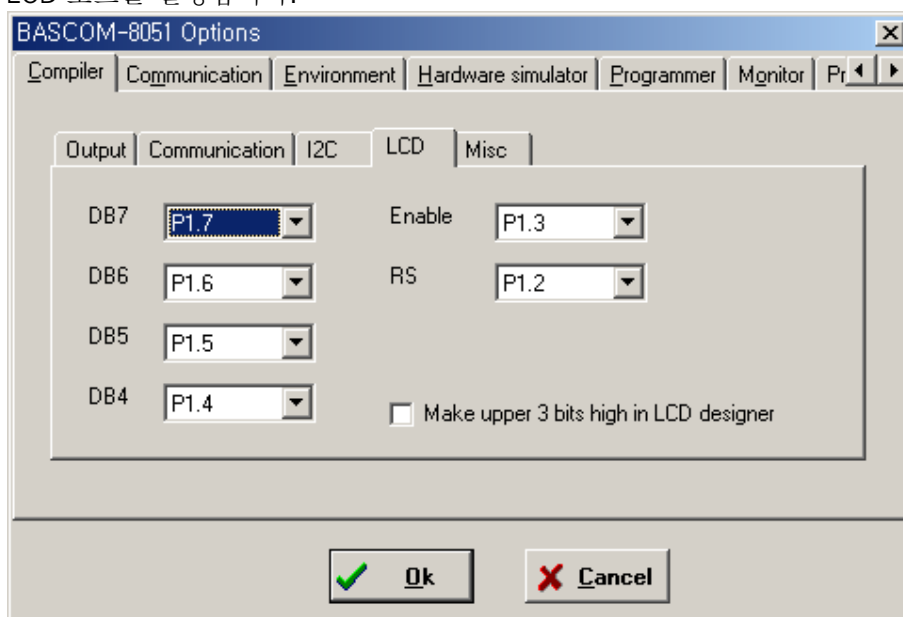
PRINT, GET등 입출력문에 사용하는 시리얼 포트의 통신속도를 설정합니다.



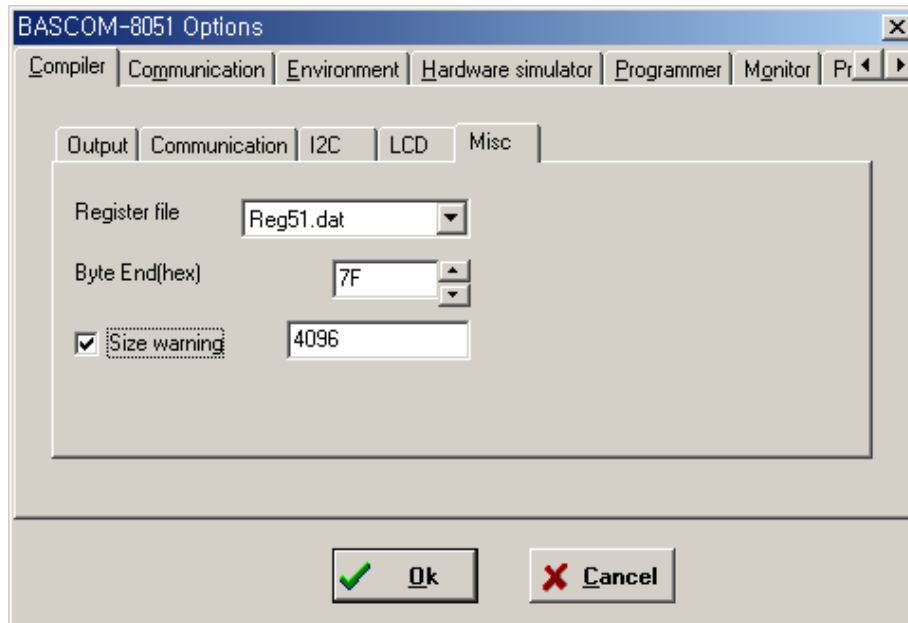
I2C 인터페이스 포트를 설정합니다.



4 비트모드 LCD 포트를 설정합니다.



프로세서 종류와 RAM 상한 어드레스 설정합니다. Register File 에서 사용할 타겟 프로세서의 종류를 선택합니다. Byte End 는 컴파일러에게 이값이상을 사용하게 되면 컴파일 에러를 발생하도록 한것입니다. 따라서 어셈블리 루틴이나 내부메모리를 직접 구동하지 않는다면 큰값으로 설정합니다. AT89C2051/4051과 AT89C51은 7F 가 최대값이며 AT89C52, AT89S8252등은 FF가 최대 값입니다.



TAB	OPTION	DESCRIPTION
Output	Binary file	컴파일 결과로 바이너리 파일을 생성합니다.
	HEX file	인텔(INTEL) 헥사 파일을 출력합니다.
	DEBUG file	시뮬레이터 동작에 필요한 MAP 파일을 생성합니다.
	Report file	레포트 파일을 만듭니다. Generates a report file.
	Old Intel Hex	일부 모니터에서 필요합니다.
	Error file	컴파일시 에러가 발생한 경우 에러내용이 수록된 파일을 만듭니다.
Communication	Baudrate	PRINT 문과 INPUT 문의 RS232 통신속도를 설정합니다. 소스프로그램에서 \$BAUD 메타 커멘드가 사용되지 않았으면 이 값을 사용합니다.
	Frequency	마이크로 프로세서의 크리스탈 주파수를 지정합니다. \$CRYSTAL 메타 커멘드가 사용되지 않았으면 이 값을 기준으로 BAUD를 산출합니다.
I2C	SCL port	I2C 통신에서 SCL 포트를 지정합니다.
	SDA port	I2C 통신에서 SDA 포트를 지정합니다.
	RC5 port	GETRC5 문의 사용 port를 지정합니다.
LCD	DB4-DB7, ENABLE, RS	LCD 디스플레이와 연결되는 포트를 지정합니다.
MISC	Register File	.DAT 파일은 8051의 종류에 따른 특수기능 레지스터(SFR)의 어드레스가 설정되어있습니다. 타겟 시스템에서 사용할 마이크로 프로세서로 설정합니다. 기본적으로 8052.DAT 이 설정되어있습니다.
	Byte End	내부변수로 사용되는 변수의 마지막 위치를 설정합니다. 스택값은 이 값 이후(Byte End + 1)로 설정됩니다. 프로그램 동작시 스택값이 프로세서의 내부메모리를 벗어난다면 이 값을 작은 쪽으로 조정 하여야 합니다. 프로그램의 동작시 스택의 최대값이 어느 정도 될 것인가는 시뮬레이터를 구동하여 확인하여야 합니다.
	Size warning	설정된 값보다 큰 HEX/BIN 코드가 출력되면 경고메시지가 표시됩니다.
	Compiler	BASCOM-8051 또는 BASCOM-LT 컴파일러 모드를 선택합니다.

Options Communication (통신 옵션)

터미널 에뮬레이터의 통신조건을 설정합니다

Item	Description
Comport	PC의 통신포트를 지정합니다.
Baudrate	통신속도를 지정합니다.
Parity	패리티 비트를 설정합니다. 디폴트 None 입니다.
Databits	데이터 비트수를 설정합니다. 디폴트 8 입니다.
Stopbits	스톱비트수를 설정합니다. 디폴트 1 입니다.
Handshake	핸드셰이크 모드를 설정합니다. 디폴트 None입니다.
Emulation	에뮬레이션 방법설정입니다. 디폴트 BBS ANSI 입니다.
Font	에뮬레이터에서 사용하는 Font 종류와 색상을 선택합니다.
Backcolor	터미널 에뮬레이터에서 배경색을 선택합니다.

주의 : 터미널 에뮬레이터에서 설정한 BAUD레이트는 컴파일러 옵션에서 설정한 BAUD 와 같아야 정상적으로 작동됩니다

Options Environment (환경 옵션)

일반적인 환경 설정입니다.

OPTION	DESCRIPTION
Auto Indent	Enter키를 누를 때 다음 라인의 커서 위치가 현재 컬럼위치에 정지합니다.
Don't change case	대소문자의 정리를 원치 않을때 선택합니다.
Reformat BAS files	에디터로 파일을 불러올 때 파일을 리포맷(Reformat) 합니다. 다른 편집기에서 작성된 소스파일을 BASCOM 에서 사용할 때 유용합니다.
Reformat code	에디터 기동시 코드를 리포맷 (Reformat)합니다.
Smart TAB	설정되어 있으면 TAB 은 앞 라인의 문자열 위치에서 TAB 이 걸립니다.
Syntax highlighting	에디터에서 BASCOM 문을 밝게 표시 합니다.
Tooltips	툴팁을 표시합니다.
Show toolbar	단축아이콘과 함께 툴바를 표시합니다.
Size of new editor window	새 윈도우를 선택 할때 윈도우의 크기를 표준으로 할 것인지 최대로 할 것인지 선택합니다.
Editor font	에디터 윈도우에서 표시되는 문자의 폰트종류와 크기를 변경합니다.
Comment	주석문 시작위치를 지정합니다.The position of the comment.
TAB-size	TAB 키를 누를 때 삽입되는 Space 문자 수를 설정합니다.
Background color	편집 윈도우의 색상을 선택합니다.
Keymapping	Default, Classic, Brief, Epsilon 중에서 선택합니다.
No reformat extension	공백 문자에 의한 파일 확장자 지정시 리포맷 하는 것을 금지합니다.

Options Hardware Simulator (시뮬레이터)

하드웨어 시뮬레이터의 연결 프린터포트와 딜레이 타임을 설정합니다.

OPTION	DESCRIPTION
Simulator LPT	시뮬레이터가 연결된 프린터 포트의 어드레스
Simulator delay	시뮬레이터의 밀리초 단위 지연시간. 고속PC 에서 적용

Options (프로그래머 옵션)

OPTION	DESCRIPTION
Programmer	프로그래머의 종류를 선택합니다.
Auto flash	자동 Flash 기능을 지원하는 프로그래머에서 선택합니다. F4를 누르면 프로그래머의 윈도우가 표시되지않은 상태에서도 프로그램 됩니다.
Auto verify	베리파이 기능을 지원하는 프로그래머에서 선택합니다. 프로그램 후에 칩의 내용이 버퍼와 베리파이 됩니다.
LPT address	프로그래머와 연결되는 프린터 포트 어드레스 입니다.
Port delay	딜레이 시간 설정이며 Blow IT 프로그래머에서 적용됩니다.
PCF8574A	MCS Flash programmer 나 시뮬레이터에서 적용됩니다. 사용된 칩이 PCF8574A 인 경우 선택합니다.
Send HEX	EPROM 시뮬레이터가 LPT 의 연결된 경우 BIN 대신에 HEX 파일을 보낼 것을 선택합니다.

Options Monitor (모니터 옵션)

모니터 설정을 변경합니다.

OPTION	DESCRIPTION
Monitor	사용되는 모니터 프로그램을 선택합니다.
Other	외부프로그램을 셸(Shell)가능 합니다. 파라메타로서 파일이름을 지정합니다.

Options Printer (프린터 옵션)

프린터에 필요한 기본값을 설정합니다.

OPTION	DESCRIPTION
Left	프린트 용지의 왼쪽 여유 공백입니다.
Right	프린트 용지의 오른쪽 여유 공백입니다.
Top	프린터 용지의 상단 여유 공백입니다.
Bottom	프린터 용지의 하단 여유 공백입니다.
Font	프린트되는 FONT를 선택합니다.

Options LCD(LCD 옵션)

LCD 디스플레이에 연결되는 포트 핀을 설정합니다.

Window Cascade(캐스케이드 윈도우)

모든 윈도우 화면을 캐스케이드(Cascade)모드로 표시합니다.

Window Tile(타일 윈도우)

모든 윈도우 화면을 타일모드로 배치합니다.

Window Arrange icons(아이콘 정렬)

아이콘을 최소화된 편집윈도우로 재 배열합니다.

Window Minimize all(윈도 축소)

오픈된 모든 편집 윈도우를 최소화 합니다.

Help About(도움말 설명)

BASCOM의 정보를 표시합니다.



시리얼 번호가 표시됩니다. 시리얼 번호의 KR 은 ISO 규정의 대한민국을 나타내는 2 자리의 지역코드입니다.

Help Index(도움말 색인)



BASCOM 의 도움말을 표시합니다. 편집중인 상태에서 현재 사용되는 단어가 키워드가 됩니다.

Help on help(도움말 사용법)

윈도우의 도움말기능 이용 설명입니다.

Editor Keys(편집 키)

Key	Action
LEFT ARROW	왼쪽으로 커서가 한 칸 이동한다.
RIGHT ARROW	오른쪽으로 커서가 한 칸 이동한다.
UP ARROW	커서가 한 줄 위로 이동한다.
DOWN ARROW	커서가 한 줄 아래로 이동한다.
HOME	라인의 처음으로 커서이동
END	라인의 마지막으로 커서이동
PAGE UP	문서위쪽으로 한페이지 이동 표시
PAGE DOWN	문서 아래쪽으로 한 페이지 이동 표시
CTRL+LEFT	왼쪽으로 한 개의 단어 앞으로 커서이동
CTRL+RIGHT	오른쪽으로 한 개의 단어 앞으로 커서이동
CTRL+HOME	문서의 맨 위로 커서 이동
CTRL+END	문서의 마지막으로 커서이동
CTRL+ Y	현재의 라인을 소거
INS	삽입모드와 겹쳐쓰기 모드의 토글
F1	도움말 Help (context sensitive)
F3	다음 위치에서 단어를 검색
CTRL+O	파일을 읽기
CTRL+S	파일저장
F7	소스프로그램의 컴파일
CTRL+P	파일프린트
CTRL+T	터미널 에뮬레이터
CTRL+M	파일 시뮬레이션
CTRL+A	플래시 프로그래머
CTRL+F	단어 검색
CTRL+P	컴파일 옵션
CTRL+W	컴파일 결과 출력 보기
CTRL+X	선택영역의 절단 및 클립보드 저장
CTRL+INS	선택영역의 클립보드로
SHIFT+INS	에디터 화면으로 클립보드 내용의 복사
Select text	쉬프트 키를 누른 상태에서 커서키를 사용하거나 마우스를 이용하여 영역을 선택합니다.

Developing order (개발 순서)

BASCOS 을 기동합니다.
파일을 오픈 하거나 새 파일을 작성합니다.
타겟 시스템의 주파수와 Baud Rate를 확인 합니다.
파일을 컴파일합니다.
에러가 발생하면 수정하고 재 컴파일(F7) 합니다.
Start BASCOS 8051;
정상동작되지 않으면 스택이 오버플로우되지 않는가를 확인합니다.
칩을 프로그램 합니다.

Memory

모든 변수는 메모리를 사용합니다.
가능한 메모리는 128 바이트 입니다.
내부 메모리 128 바이트 중 32 바이트를 컴파일러에서 사용합니다.
약간은 스택으로 사용합니다. 사용된 명령어에 따라 다릅니다.
8 개의 비트는 1 개의 바이트를 점유합니다.
바이트는 1 바이트를 점유합니다.
정수/워드는 2 바이트를 점유합니다.
Long 정수는 4 바이트를 점유합니다.

바이트 변수는 음수(Negative)를 사용할 수 없습니다.
프로그램의 사용되는 바이트수가 1024 이하이면 89C1051을 사용할 수 있습니다. 그러나 UART가 없으므로 PRINT문과 INPUT 문을 사용할 수 없습니다.
다른 모든 8051 칩을 사용할 수 있습니다.

BASCOM-8051

Language Reference

8051 BASIC COMPILER

Bascom 8051 Help File V2.11을 기준으로 작성한 것입니다.

LANGUAGE FUNDAMENTALS

BASCOS 에서 문자셋은 라벨(label), 예약어(keywords), 변수(variable), 연산자(operator)로 사용됩니다.

여기에서 BASCOS 의 프로그램 라인 형식과 문자 셋에 대하여 설명합니다.

- 문자셋에서 특별히 지정된 것은 특별한 의미를 갖고 있습니다.
- BASCOS 프로그램 라인의 형식
- 라인라벨
- 프로그램 라인 길이

Character Set / 문자 셋

BASCOS 에서 문자 셋은 알파벳(A-Z), 숫자(0-9), 특수문자로 이루어 집니다.

BASCOS 에서 영문자는 대문자(A-Z)와 소문자(a-z) 알파벳으로 구성됩니다.

BASCOS 에서 숫자는 0-9 로 구성됩니다.

일부 알파벳 문자(A,B,C,D,E,F)는 숫자(0 – 9)와 함께 헥사 데시말(HEX)로 사용됩니다.

다음의 문자는 BASCOS 의 명령어와 수식에서 특별한 의미를 갖습니다.

Character	Name
ENTER	입력의 종료 / Terminates input of a line
	공백 / Blank (or space)
'	싱글따옴표 / Single quotation mark (apostrophe)
*	곱하기 / Asterisks (multiplication symbol)
+	더하기 / Plus sign
,	컴마 / Comma
-	마이너스 / Minus sign
.	점(십진수에서 소수점) / Period (decimal point)
/	슬래시 (나누기 기호) / Slash (division symbol) will be handled as \
:	콜론 / Colon
“	쌍 따옴표 / Double quotation mark
;	세미 콜론 / Semicolon
<	~보다작다 / Less than
=	같다 / Equal sign (assignment symbol or relational operator)
>	~보다 크다 / Greater than
?	물음표 / Question mark
\	백슬래시 / Backslash (integer/word division symbol)

The BASCOS program line / 프로그램 라인

BASCOS 의 프로그램 라인은 다음과 같은 형식을 갖습니다.

```
[[line-identifier]] [[statement]] [[:statement]] ... [[comment]]
```

Using Line Identifiers / 라인의 구분 선언

BASCOS 는 문 번호나 알파뉴메릭 라인 라벨 중 하나만을 지원합니다.

문자에의한 라인 라벨은 1 부터 32개수의 문자로 구성할 수 있습니다. 반드시 문자로 시작하여 콜론(:)으로 종료합니다.

BASCOS 의 키워드(예약어)는 사용할 수 없습니다. 다음은 사용 가능한 라인 라벨입니다.

Alpha:

ScreenSUB:

Test3A:

대소문자는 구별되지 않습니다. 다음은 모두 같이 취급됩니다.

Alpha:
Alpha:
ALPHA:

라인라벨은 프로그램 라인에서 무엇보다도 먼저 나타날 것입니다. 라벨중간의 공백문자는 허용되지 않습니다. 한 개의 라인은 오직 한 개의 라벨만 가질 수 있습니다.

BASCOS Statements / 실행문

BASCOS 문은 실행 문(executable) 또는 비실행 문(nonexecutable)으로 되어 있습니다. 실행문이란 프로그램의 논리적인 흐름에서 다음에 할 일을 지정하는 것입니다. 비 실행문이란 변수를 저장하기 위한 위치지정문, 변수 형식 선언문 등입니다.

다음의 BASCOS 문은 비 실행 문 입니다.

- **REM** or ' (starts a comment)
- **DIM**

“주석문”은 프로그램 조작과 목적을 분명히 나타내기 위한 비 실행 문 입니다.

주석 문은 REM 문 또는 싱글 따옴표(') 로 시작됩니다.

다음은 모두 같은 것 입니다.

PRINT “Quantity remaining” : REM Print report label.

PRINT “Quantity remaining” ' Print report label.

한 줄에 한 개 이상의 BASCOS 문이 올 수 있습니다만 반드시 콜론(:)으로 구분하여야 합니다.

FOR I = 1 TO 5 : PRINT “G'day, mate.” : NEXT I

BASCOS Line Length / 라인의 길이

에디터에 의하여 프로그램할 경우 BASCOS에서 프로그램 라인의 길이의 제한은 없으나 가독성을 위하여 80 문자 이내일 것을 권장합니다

Data Types / 데이터 형식

BASCOS 에서 모든 변수는 데이터 형식을 갖습니다.

Elementary Data Types / 기본 데이터 형식

- **Bit (1/8 byte)**
- **Byte (1 byte).**
- 바이트는 무 부호 8 비트 이며 0부터 255 까지 입니다.
- **Integer (two bytes).**
- 정수는 부호 확장된 16 비트로 저장되며 -32768부터 +32767까지의 범위입니다.
- **Word (two bytes).**
- 워드는 무부호 16비트 이며 0부터 65536까지의 범위입니다.
- **Long (four bytes)**
- 롱은 부호부 32 비트 이며 -2147483648 에서 2147483647까지입니다.
- **Single**
- 부동소수점 데이터이며 부호가 있는 32 비트 바이너리 데이터입니다.
- **String (up to 254 bytes).**
- 스트링의 크기는 254바이트 까지입니다.
- 스트링은 바이트 단위로 저장되며 0으로 끝납니다.
- 스트링은 길이가 10 바이트일 경우 11 바이트의 외부 메모리를 차지합니다.
-

변수는 내부램(기본)또는 외부로 저장됩니다.

Variables / 변수

변수란 특정 숫자를 나타내는 개체를 참조하는 이름입니다.
변수는 숫자 값(정수, 워드, 바이트 또는 비트)을 저장합니다.
아래의 예는 변수를 지정하는 몇 가지 예입니다.

- 상수지정
A = 5
- 변수의 재정의
abc = def
k = g
- 다른변수, 상수, 연산자의 조합에 의하여 얻어지는 값
Temp = a + 5
Temp = C + 5

Variable Names / 변수이름

BASCOS 에서 변수는 32 문자까지 허용합니다. 변수의 첫번째 문자는 반드시 문자 이어야 합니다.
변수이름은 예약어를 사용하지 못합니다. 그러나 예약어 라도 조합된 것이면 무방합니다.
예를 들어 다음의 변수이름은 허용되지 않습니다.

AND = 8

그러나 다음의 예는 적합합니다.

ToAND = 8

예약어(reserved words)는 모든 BASCOS 의 명령어, 문, 함수이름, 내부 레지스터와 연산자(operator)입니다.

16진수와 2 진수는 &H 와 &B의 접두사를 사용하여 나타낼 수 있습니다.

A = &HA , a = &B1010 and a = 10 는 모두 같은 것입니다.

변수를 사용하기 전에 컴파일러에게 DIM 문을 사용하여 지정할 수 있습니다.

Dim b1 As Bit, l as Integer, k as Byte , s As String * 10 , L as Long

Dim b2 as Xram Byte

DEFINT, DEFBIT, DEFBYTE 와 DEFWORD를 사용할 수 있습니다.

예를 들면 DEFINT c 는 변수의 첫 문자가 c 이면 INT 형임을 알려줍니다.

Expressions and Operators / 수식 표현과 연산자

여기서는 BASCOS에서 사용되는 연산자가 어떻게 연결, 수정, 비교, 또는 정보취득이 이루어지는가에 대하여 설명합니다.

수식과 연산자를 이용하여 언제든지 계산을 할수 있습니다.

어떻게 수식이 구성되며

- 연산자는 계산하기 위하여 사용된다.
- 관계연산자는 숫자의 비교를 위하여 사용된다.
- 논리연산자는 조건의 검사 또는 개별비트의 조작을 위하여 사용된다.
- 기능연산자는 단순연산을 위하여 사용된다.

Expressions and Operators / 수식 연산

수식은 숫자상수, 변수 또는 상수, 변수와 다른 연산자에 의한 수식에 의하여 얻어진 하나의 값이 됩니다.

연산자는 수학적 또는 논리적 기능을 가집니다.

BASCOS에서 지원하는 연산자는 4 개의 영역으로 나누어집니다.

1. Arithmetic /수식 연산

- 2. Relational /관계
- 3. Logical /논리
- 4. Functional / 기능

Arithmetic / 산술연산

수식연산자는 +, -, * 그리고 \ 입니다.

- 정수
정수의 나눗셈은 백슬래시로 나타냅니다.
Example: PRINT X\Y
- 나머지 연산
나머지 연산은 MOD 연산자에 의하여 표시됩니다.
- 나머지 연산은 나눗셈에서 나머지 값을 반환합니다.
Example: X = 10 \ 4 : remainder = 10 MOD 4
- 오버플로우와 0 으로 나누기
- 0 으로 나누면 에라가 발생합니다 이 경우에 아무런 경고도 나타나지 않습니다. 따라서 이러한 일이 발생치 않도록 확실히 해야 합니다.

Relational Operators / 관계연산자

관계연산자는 두개의 값을 비교할 때에 사용합니다.
결과는 프로그램 흐름을 결정하기 위하여 사용됩니다.

Operator	Relation Tested	Expression
=	Equality / 같다	X = Y
<>	Inequality / 다르다	X <> Y
<	Less than / 보다 작다	X < Y
>	Greater than / 보다 크다	X > Y
<=	Less than or equal to /작거나 크다	X <= Y
>=	Greater than or equal to/ 크거나 같다	X >= Y

Logical Operators / 논리 연산자

논리연산자는 관계검사, 비트조작, 또는 BOOLEAN연산을 합니다.
BASCOM 에서 다음의 다섯 종류의 논리연산을 제공합니다.

Operator	Meaning
NOT	Logical complement / 논리반전
AND	Conjunction / 결합
OR	Disjunction / 분리
XOR	Exclusive or / 배타적 화

바이트의 특별한 비트패턴을 검사하기 위하여 논리연산자를 사용하는 것이 가능합니다.
예를들면 **AND** 연산자는 스테터스 (status)바이트의 한 개의 비트 값을 걸러낼 때 사용될 수 있습니다.
반면에 **OR** 연산자는 두개의 바이트 값을 합쳐서 새로운 이진 값을 만들어낼 수 있습니다.

Example

```
A = 63 And 19
PRINT A
A = 10 Or 9
PRINT A
```

Output

16

Floating point / 부동소수점

BASCOS에서 부동소수점 변수가 추가 되었습니다. IEEE 바이너리 부동소수점 형식을 따릅니다. 8 비트의 지수부와 24 비트의 가수부로 구성됩니다.

4 개의 바이트를 사용하며 포맷은 아래와 같습니다.

```

31 30_____23 22_____0
s  exponent      mantissa
  
```

지수부는 128 상향(biased)되어 있습니다. 128 보다 큰수는 양의 지수를 이하는 음의 지수를 표시합니다. 부호비트 s 는 0 일때 양의수를 1 일때 음의수를 나타냅니다. 가수부에는 1 개의 비트가 숨겨져 있습니다. 따라서 항상 24 비트입니다. 모든 연산은 부동소수점 데이터를 처리합니다. Single 변수를 Integer/Word 로 상호 변환할 수 있습니다.

Dim I As Integer, S As Single

S = 100.1 ' assign the single

I = S ' will convert the single to an integer

더 자세한 정보는 single.bas 를 참조하기 바랍니다.

Arrays / 배열

BASCOS 에서 추가된 변수 형식입니다.

배열이란 동일한 형식의 데이터가 참조번호에 의하여 연속적으로 모여진 집합입니다. 각 요소는 고유인 인덱스 번호를 가지고 있습니다.

인덱스 번호는 Byte, Integer 또는 Word 데이터 입니다. 배열의 한 개의 요소가 변경되어도 다른 요소는 영향 받지 않습니다.

BASCOS-8051에서 비트 변수는 Array 요소로 사용할 수 없습니다.

Example:

Dim a(10) As byte ' make a array named a, with 10 elements(1 to 10)

Dim c As Integer

For C = 1 To 10

A(c) = c 'assign array element

Print a(c) 'print it

Next

Strings / 스트링

BASCOS에서 스트링은 254 문자로 구성될 수 있습니다.

메모리를 절약하기 위하여 Dim 문장에서 스트링의 크기를 정합니다.

Dim S As String *10

스트링 변수 S 는 10 바이트 입니다. 실제의 길이는 null(0)코드가 추가되어 11 바이트가 됩니다.

스트링은 + 연산자에 의하여 합쳐질 수 있습니다.

Dim S As String *10, Z As String *10

S = "test"

Z = S + "abc"

Compiler Limits / 컴파일러의 제한

컴파일러에는 몇 가지 제한 사항이 있습니다.
하나의 식에는 단지 한 개의 연산만 허용됩니다.

Good

$a = a * b1$

$a = \text{Asc}(s) + \text{Len}(s)$

False

$a = a * b1 + c$

Limit	Number
Maximum allowed labels / 최대 라벨 수	5000
Maximum allowed variable names / 최대 변수이름 수	1000
Maximum number of INTEGER/WORD variables 최대 INTEGER/WORD 변수 개수	10*
Maximum number of BYTE variables / 최대 BYTE 변수 수	20*
Maximum number of BIT variables 최대 BIT 변수 수	120*
Maximum number of STRING variables/ 최대 스트링 변수 수	up to available external memory
Maximum number of ALIAS statements 최대 ALIAS문	128

* 변수의 사용량과 사용된 실행 문에 따라 다릅니다. AT2051 은 128 바이트의 내부

RAM을 가지고 있습니다.

최대 32 Bytes가 PRINT 와 INPUT 루틴 그리고 SETDATA 와 GETDATA 루틴으로 내부적으로 사용됩니다.

나머지는 응용프로그램으로 사용됩니다.

스택으로 약간의 메모리가 사용됩니다. 따라서 얼마나 많은 변수를 사용할 수 있는가 하는 것은 사용된 실행 문에 다릅니다.

8 비트는 1 바이트입니다.

1 개의 사용된 바이트는 1 byte 입니다.

1 개의 Integer/Word는 2 Bytes 입니다.

1 개의 long 변수는 4 bytes입니다.

10 bytes의 스트링은 11 bytes의 메모리를 차지합니다.

Maximum nesting : / 최대 내포 수

Operation	Max
FOR .. NEXT	50
IF .. THEN	50
DO .. LOOP	50
WHILE .. WEND	50
SELECT CASE	25

The following table shows the reserved BASCOM statements.

Reserved Words / 예약 어

예약 어는 변수이름으로 사용할 수 없습니다.

!	CONST	I2CRECEIVE	OFF	THEN
;	COUNTER	I2CSEND	ON	THIRD
\$INCLUDE	COUNTER0	I2CSTART	OR	THIRDLIN
\$NOINIT	COUNTER1	I2CSTOP	OUT	TIMEOUT
\$NOSP	CPEEK()	I2CRBYTE	P0-P6	TIMER0
\$NOBREAK	CURSOR	I2CWBYTE	PEEK()	TIMER1
\$BAUD	DATA	IDLE	POKE	TO
\$BGF	DEC	IF	POWERDOWN	UNTIL
\$DEFAULT	DECLARE	INC	PSET	UPPER
\$CRYSTAL	DEFBIT	INKEY	PRINT	UPPERLINE
\$LARGE	DEFBYTE	INP()	PRINTEX	VAL()
\$LCD	DEFLCDCHAR	INPUT	PRIORITY	WAIT
\$ROMSTART	DEFINT	INPUTHEX	READ	WAITKEY
\$RAMSIZE	DEFWORD	INT0	READEEPROM	WAITMS
\$RAMSTART	DELAY	INT1	REM	WATCHDOG
\$SERIALINPUT	DIM	INTEGER	RESET	WRITEEEPROM
\$SERIALOUTPUT	DISABLE	INTERNAL	RESTORE	WEND
\$SIM	DISPLAY	IS	RETURN	WHILE
1WRESET	DO	LCD	RIGHT	WORD
1WREAD	DOWNT0	LCDHEX	RIGHT()	XOR
1WRITE	ELSE	LEFT	RND()	XRAM
ACK	ELSEIF	LEFT()	ROTATE	TCON
ALIAS	ENABLE	LOAD	SELECT	P1
ABS()	END	LOCATE	SERIAL	SCON
AND	ERR	LONG	SET	IE
AS	EXIT	LOOKUP	SHIFT	P3
ASC()	EXTERNAL	LOOP	SHIFTLCD	IP
BAUD	FOR	LOW()	SHIFTCURSOR	PSW
BCD()	FOURTH	LOWW()	SHIFTIN	ACC
BIT	FOURTHLINE	LOWER	SHIFTOUT	B
BITWAIT	GATE	LOWERLINE	SHOWPIC	SP
BLINK	GETAD	MAKEBCD()	SOUND	DPL
BOOLEAN	GOSUB	MAKEDEC()	SPACE()	DPH
BREAK	GOTO	MAKEINT()	START	PCON
BYTE	HEXVAL()	MID()	STEP	TMOD
CALL	HIGH()	MOD	STR()	TL0
CASE	HIGHW()	MODE	STRING()	TL1
CLS	HOME	NACK	STOP	TH0
CHR()		NEXT	STOP TIMER	TH1
CONFIG		NOBLINK	SUB	SBUF
		NOSAVE	SWAP	
		NOT		

Option -> Misc 메뉴에서 Register File 을 선택하면 내부 레지스터 가 변경됩니다.

1WRESET,1WREAD,1WWRITE

Action

달라스 세미컨덕터사(Dallas Semiconductor)의 1 선 연결 소자와 통신하기 위한 루틴입니다.

Syntax

1WRESET

1WWRITE *var1*

var2 = **1WREAD()**

Remarks

1WRESET	1 와이어 버스의 리셋. 실패시 에라값으로 1 을 반환합니다.
1WWRITE <i>var1</i>	변수 <i>var1</i> 을 버스에 라이트 합니다.
<i>Var2</i> = 1WREAD()	버스로부터 byte 데이터를 읽어 변수 <i>var2</i> 에 저장합니다.

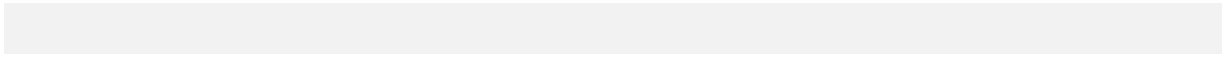
Example

```

'-----
'                               1WIRE.BAS
' demonstrates 1wreset, 1wwrite and 1wread()
' pullup of 4K7 required to VCC from P.1
' DS2401 serial button connected to P1.1
'-----
Config 1wire = P1.1                'use this pin
Dim Ar(8) As Byte , A As Byte , I As Byte

1wreset                            'reset the bus
Print Err                          'print error 1 if error
1wwrite &H33                       'read ROM command
For I = 1 To 8
    A = 1wread()                    'read byte
    Ar(i) = A                       'place into array
Next
For I = 1 To 8
    A = Ar(i) : Printhex A;          'print output
Next
Print                               'linefeed
End

```



#IF, #ELSE, #ENDIF

Action

조건 검사에 따라 컴파일 하도록 합니다.

Syntax

```
#IF test
[#ELSE]
#ENDIF
```

Remarks

test 조건 검사식이며 정의된 상수 값을 사용합니다. 상수 값에 따라 컴파일하면 많은 편리함이 있습니다. **#IF** 문은 **THEN** 키워드가 없습니다. 25 단계까지 내포 가능합니다. **#ELSE** 는 옵션입니다.

See Also

#ELSE , **#ENDIF**

Example

```
CONST DEMO = 1 ' 0 = normal , 1 = demo
#IF Demo
    Print "Demo program"
#ELSE
    Print "Full version"
#ENDIF
```

상수 **DEMO** 에 1 을 지정하면 컴파일러는 "Demo Program"을 출력하는 단지 1 라인만 컴파일합니다. **#ELSE** 와 **#ENDIF** 중간의 문은 컴파일하지 않습니다. **DEMO** 상수에 0 을 지정하면 "Full Version" 문만 컴파일합니다.

\$ASM-\$END ASM

Action

인라인 어셈블리 코드 블록을 시작합니다.

Syntax

\$ASM

Remark

작성 중인 BASIC 코드에 **\$ASM** 과 **\$END ASM** 을 어셈블리 코드와 같이 삽입합니다.

Example

```
Dim c as Byte
$ASM
    Mov r0, #{C} ; address of c
    Mov a, #1
    Mov @r0, a ; store 1 into var c
$END ASM
Print c
End
```

\$INCLUDE

Action

프로그램 현재의 위치에 ASCII 파일을 포함합니다.

Syntax

\$INCLUDE file

Remarks

File	ASCII파일에는 유효한 BASCOM 문을 포함하고 있어야합니다. 이 옵션은 많은 프로그램에서 공통적으로 사용되는 프로그램을 주프로그램에 불러들여 사용하게 합니다. 모듈로 프로그램 작성할 수 있으며 프로그램에 포함할 수 있습니다. 필요시 모듈부분만 수정하면 되므로 BASCOM 프로그램 전체를 수정하지 않아도 됩니다. ASCII 문자로 구성된 파일만 가능합니다.
------	---

Example

```
'-----
' (c) 1997,1998 MCS Electronics
'-----
' file: INCLUDE.BAS
' demo: $INCLUDE
'-----

Print "INCLUDE.BAS"
$include c:\bascom\123.bas 'include file that prints Hello
Print "Back in INCLUDE.BAS"
End
```

\$BAUD

Action

다른 통신속도(Baud Rate)를 사용하기 원할 때 컴파일러에게 지정합니다.

Syntax

\$BAUD = var

Remarks

Var	바우드 레이트(통신속도)
-----	---------------

Var는 상수입니다.

지원되지않는 크리스탈 / 바우드 레이트 를 사용하기 원하면 **\$CRYSTAL** 메타 명령어를 사용합니다. 출력된 레포트에 실제로 만들어진 바우드 레이트가 표시됩니다.

시리얼 통신에서 바우드 드레이트는 표준적인 속도가 정해져 있습니다. (일반적으로 300, 600, 1200, 2400, 4800, 9600, 19200, 38400 etc..) 따라서 8051 프로세서의 크리스탈 주파수는 바이트레이트 발생에 편리한 주파수를 사용합니다. 일반적으로 8051에서 오차를 발생하지 않는 11.0592MHz 또는 22.1184MHz 의 크리스탈을 많이 사용합니다. 다른 주파수의 크리스탈도 사용가능하지만 이때에는 오차가 발생합니다. 시리얼 통신에서의 오차범위는 +3%입니다. 이 오차범위 이내에 들어오면 데이터 송수신에 문제를 발생하지 않습니다.

See also

\$CRYSTAL

Example

```
$BAUD = 2400
$CRYSTAL = 14000000      '14 MHz crystal
PRINT "Hello"
End
```

\$BGF

Action

LCD 디스플레이에 사용할 BASCOM 그래픽 파일(BGF)을 프로그램에 연결합니다.

Syntax

```
$BGF "file"
```

Remarks

"File"은 포함될 BGF 형식의 파일 이름입니다. Tools Graphic Converter를 이용하여 BMP 파일을 BGF 파일로 변환합니다.

See also

SHOWPIC

Example

```
'-----
'
'          (c) 2000 MCS Electronics
'          GLCD.BAS
'
'      Sample to show support for T6963C based graphic display
'      Only 240*64 display is supported with 30 columns(yet)
'      At the moment the display can only be used in PORT mode
' Connection :
' P1.0 - P1.7 to DB0-DB7 of LCD
' P3.2      to FS, font select of LCD can be hard wired too
' P3.5      to CE, chip enable of LCD
' P3.4      to CD, code/data select of LCD
' P3.6      to WR of LCD
' P3.7      to RD of LCD
' A future version will allow external data access too which also uses RD and WR
' The display from www.conrad.com needs a negative voltage for the contrast.
' I used two 9 V batteries
'-----
' configure the LCD display
Config Graphlcd = 240 * 64 , Port = P1 , Ce = P3.5 , Cd = P3.4 , Cols = 30

' dimension some variables used by the DEMO
Dim X As Byte , Y As Byte

'
Reset P3.2                                '8 bit wide char is 30 columns

' The following statements are supported:
Cls                                         'will clear graphic and text
'cls TEXT will clear only the text
'cls GRAPH will clear only the graphic part
```

'To init the display manual you can use:

'Lcdinit

'But this should not be needed as it is initialised at start up.

'Locate is supported and you can use 1-8 for the row and 1-30 for the column

Locate 1 , 1

'cursor control is the same as for normal LCD

Cursor On Blink

'And to show some text you can use LCD

Lcd "Hello world"

'Note that the cursor position is not adjusted. You can set it with locate

'Now comes the fun part for using a graphic LCD

'We can display a BMP file. You may use MSPAINT or any other tool that can create

'a BMP file. With the Graphic converter from the Tools Menu you can convert the file

'into a BGF file. (BASCOS GRAPHICS FILE). The conversion will convert all non white

'pixels to BLACK.

'To display the BGF file you use the SHOWPIC statement that needs an X and Y parameter

'the third param is the label where the data is stored.

'The position must be dividable by 8 because this is the way the display handles the data

Showpic 0 , 0 , Picture1

'And we use the PSET known from QB to set or reset a single pixel

'A value of 0 means clear the pixel and 1 means set the pixel

'create a block

For X = 0 To 10

For Y = 0 To 10

Pset X , Y , 1

Next

Next

'You could remove it too

For X = 0 To 10

For Y = 0 To 10 Step 2

Pset X , Y , 0

Next

Next

'A simple scope or data logger could be made with PSET !

'We hope to get an AN from an inspired user :-)

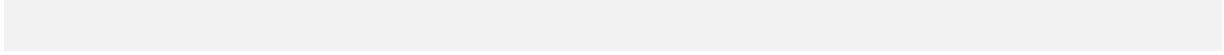
End

'label for the picture

Picture1:

'\$BGF includes the data from the specified file

\$bgf "samples\Wmcs.bgf"



\$CRYSTAL

Action

컴파일러에 크리스탈 주파수를 알려줍니다.

Syntax

\$CRYSTAL = var

Remarks

var	크리스탈 주파수
-----	----------

Var는 상수입니다.

지원되지않는 크리스탈 / 바우드 레이트를 사용하기 원할 경우 사용합니다.

\$BAUDRATE 메타 명령어를 같이 사용하여야 합니다.

See also

[\\$BAUD](#)

Example

```
$BAUD = 2400
$CRYSTAL = 14000000
PRINT "Hello"
End
```

\$IRAMSTART

Action

컴파일러에게 시작 내부 RAM 어드레스를 지정합니다.

Syntax

\$IRAMSTART = constant

Remarks

Address	0-255사이의 상수 값 입니다.
---------	--------------------

Example

```
$NOINIT
$NOSP
$IRAMSTART = &H60    ' first usable memory location
SP = 80
DIM I As Integer
```

\$DEFAULT XRAM

Action

컴파일러에게 선언된 변수의 위치를 XRAM에 배치하도록 지정합니다.

Syntax

\$DEFAULT XRAM

Remarks

XRAM 변수를 많이 사용한다면 변수선언 시 매번 XRAM을 지정하지 않도록 이 옵션을 사용합니다.
변수가 IRAM 에 사용되기 원한다면 IRAM 을 지정합니다.

Example

```
$DEFAULT XRAM
```

```
Dim X As Integer 'will goto XRAM
```

```
Dim Z as IRAM Integer 'will be stored in IRAM
```

\$LARGE

Action

컴파일러에게 어셈블러 서브루틴 호출시 LCALL명령을 사용할 것을 지정합니다.

Syntax

\$LARGE

Remarks

서브루틴 호출시 ACALL 명령을 사용합니다.

ACALL명령은 단지 2 bytes(LCALL명령은 3 bytes)가 필요합니다.

ACALL명령은 어드레스의 오프셋이 2048입니다.

AT89C2051은 Flash 메모리 크기가 2048이므로 문제 없습니다.

64K 어드레스 공간을 전부 호출 가능하기 위하여는 LCALL을 사용 하여야 합니다.

Example

```
$LARGE          'I received an error 148 so I need this option
```

\$LCD

Action

데이터 버스에 8 비트 LCD 디스플레이를 연결하기 위한 코드를 만들어냅니다.

Syntax

\$LCD = [&H]address

Remarks

address	어드레스는 반드시 LCD-디스플레이를 구동 가능한 것이어야 합니다. 데이터 버스 db0-db7은 LCD의 데이터 버스 D0-D7과 연결되어 있어야 합니다. LCD의 RS라인은 반드시 어드레스 A0에 연결하여야 합니다. 외부 ROM/RAM이 연결되어있는 경우 데이터버스에 LCD를 부착 시 주의하여야 합니다. 어드레스 디코더가 LCD를 선택하도록 합니다.
---------	---

Example

```
$LCD = &HA000      'writing to this address will make
                    'the E-line of the LCD high.
LCD "Hello world"
```

\$NOBREAK

Action

컴파일러에게 BREAK문이 컴파일되지 않도록 지시합니다.

Syntax

\$NOBREAK

Remarks

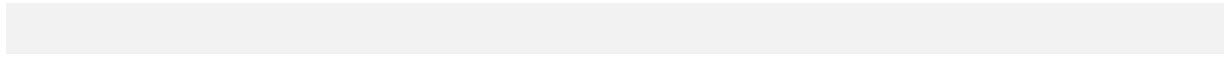
BREAK문은 시뮬레이터의 동작을 멈추기 위한 명령어를 만들어냅니다.

\$NOBREAK 명령어를 사용하면 BREAK 커맨드를 제거하지 않아도 됩니다.

See also

Example

```
$NOBREAK
BREAK      ' this isn't compiled into code
           'so the simulator will not pause
End
```



\$NOINIT

Action

컴파일러에게 프로세서 초기화코드를 만들지 않도록 지시합니다.

Syntax

\$NOINIT

Remarks

BASCOS 는 사용된 문에 따라서 프로세서를 초기화 합니다.

\$NOINIT 메타 컴맨드를 사용하여 초기화코드를 선택합니다. 단지 스택포인트만 초기화되며 LCD명령이 사용된 경우 이에 대한 초기화 코드만 만들어집니다.

See also

Example

```
$NOINIT
..... 'your program goes here
.....
End
```

\$NONAN

Action

Compiler directive for changing nan (not a number) into 0.0

컴파일러에게 단정도 부동 소숫점(Single 형) 연산 시 결과값이 숫자가 아닌 경우 0.0 을 반환하도록 지시합니다. (연산 결과가 정상이 아닌 경우)

Syntax

\$NONAN

Remarks

단 정도 부동소수점 연산 시 결과 값이 숫자가 아니면 NAN을 반환합니다. \$NONAN 지시어를 사용하면 0.0을 반환합니다.

See also

NONE

Example

NONE

\$NONULL

Action

컴파일러에게 DATA 문에서 NULL 대신 다른 값을 반환하도록 합니다.

Compiler directive for changing the behavior of the DATA statements.

Syntax

\$NONULL = value

Remarks

value 0 디폴트 값입니다. 그리고 특별한 경우에 -1을 지정합니다.

Data 문에 스트링이 저장되어 있을 때 스트링을 표시하기 위하여 Null 코드가 첨부되어 있습니다.

어떤 경우 이것을 원치 않을 때가 있습니다. 예를 들어 긴 스트링을 처리하는 전용루틴을 만들어 작업할 때 \$NONULL = -1 을 지정하면 Null 바이트가 첨부되지 않습니다. 0 을 지정하여 다시 실행하면 원래대로 환원됩니다.

See also

NONE

Example

```
$nonnull = -1
```

```
Lbl:
```

```
Data "test" , "this"
```

```
Lbl2:
```

```
$nonnull = 0                                'normal mode
```

```
Data "test" , "this"
```

\$NORAMCLEAR

Action

컴파일러에게 스타트업(리셋)시 내부 RAM을 초기화하지 않도록 지시합니다.

Syntax

\$NORAMCLEAR

Remarks

BASCOS 은 리셋후 내부 RAM 을 클리어 합니다. 이러한 상황을 원치 않으면 \$NORAMCLEAR 지시어를 사용합니다.

See also

-

Example

```
$NORAMCLEAR
```

```
our code goes here
```

```
End
```

\$NOSP

Action

컴파일러에게 스택 초기값을 설정하지 않도록 합니다.

Syntax

\$NOSP

Remarks

BASCOM은 사용된 문에 따라 프로세서를 초기화 합니다.

초기화 작업을 직접 하기 원할 경우 **\$NOINIT** 메타 명령어를 사용합니다.

단지 스택포인터와 **LCD** 디스플레이의 초기화(**LCD**가 사용된 경우만) 코드는 항상 생성됩니다.

\$NOSP 메타 명령어와 같이 사용하여 스택 초기화를 하지 않을 수 있습니다.

See also

\$NOINIT

Example

```
$NOSP
$NOINIT
End
```

\$OBJ

Action

인텔 오브젝트 코드를 삽입합니다.

Syntax

\$OBJ obj

Remarks

Obj는 삽입할 오브젝트 코드입니다.

Example

```
$OBJ D291      ' this is equivalent to SET P1.1
```

\$RAMSIZE

Action

외부RAM의 용량을 지정합니다.

Syntax

\$RAMSIZE = [&H] size

Remarks

size	외부메모리칩의 RAM용량
------	---------------

Example

```
$ROMSTART = 4000
$RAMSTART = 0
$RAMSIZE = $H1000
DIM x AS XRAM Byte           'specify XRAM to store variable in XRAM
```

\$RAMSTART

Action

외부메모리의 위치를 지정합니다.

Syntax

\$RAMSTART = [&H]address

Remarks

Address	데이터가 저장될 어드레스 값 (16진수)입니다. 외부RAM에서 실행프로그램이 동작 되기 원할 때 이 옵션을 사용합니다.
---------	--

Address: 상수 값 입니다.

See also

\$RAMSIZE

Example

```
$ROMSTART = 4000
$RAMSTART = 0
$RAMSIZE = &H1000
```

\$REGFILE

Action

컴파일러에게 레지스터 파일을 지정합니다.

\$REGFILE 이 지정되지 않으면 BASCOM 8051 IDE의 메뉴바의 Options -> Compiler -> Misc 에서 지정한 레지스터파일을 사용합니다.

레지스터 파일은 사용할 프로세서의 종류별로 차이가 있습니다.

Syntax

\$REGFILE = "file"

Remarks

File은 사용할 레지스터 파일 이름입니다.

\$REGFILE은 어떤 실행문이나 컴파일러 지시어 보다도 앞에 있어야 합니다.

REG51.DAT	89C51RC.DAT	80517.DAT	MSC1210.DAT
8052.DAT	89C51CC.DAT	80520.DAT	TRISEND.DAT
89C1051.DAT	89C51RD2.DAT	80552.DAT	8051F000.DAT
89C2051.DAT	89C420.DAT	80535.DAT	8051F020.DAT
89C4051.DAT	89C668.DAT	80573.DAT	8051F300.DAT
89C55WD.DAT		87C520.DAT	80C320.DAT
89S8252.DAT		87DS550.DAT	812.DAT
			871PC764.DAT

See also

NONE

Example

'comment is no problem before the \$REGFILE statement

\$REGFILE = "8052.DAT" 'use the 8052.DAT file

\$ROMSTART

Action

외부ROM의 위치를 지정합니다.

Syntax

\$ROMSTART = [&H] address

Remarks

Address	반드시 실행코드의 시작 어드레스이어야 합니다. \$ROMSTART 문이 없으면 내장 값으로 0 이 사용 됩니다. 실행코드를 RAM 에서 테스트 하기 위하여 사용할 수 있습니다. 코드는 지정된 위치(어드레스)에 다운로드 되어야 하며 모니터 프로그램이 호출 가능 하도록 하여야 합니다. 모니터 프로그램은 인터럽트 어드레스를 재 배치합니다. \$ROMSTART = 4000 이 지정되었다면 모니터 프로그램은 LJMP 명령을 사용하여야 합니다. 어드레스 가 3 일때 4003을 지정하여야 합니다. 그렇지 않으면 인터럽트는 정확하게 동작하지 않습니다. 파일이름이 xxx.INF인 파일을 확인하여야 합니다
---------	---

Example

\$ROMSTART = &H4000 ' ROM enabled at 4000hex

\$SERIALINPUT

Action

시리얼 입력의 방향을 변경 지정 합니다.

Syntax

\$SERIALINPUT = label

Remarks

label	INPUT 루틴으로부터 문자가 필요할 때 호출되는 어셈블러 프로그램의 이름입니다. 반드시 서브루틴 복귀시 ACC에 문자 값이 있어야 합니다.
-------	--

사용자가 직접 만든 서브루틴을 INPUT 명령어로 방향전환합니다.

INPUT 명령어는 RETURN코드(십진수 13 또는 16진수 &H0D)이 입력 되어야 종료됩니다.

See Also

\$SERIALOUTPUT

Example

```
$SERIALINPUT = Myinput
    ' here goes your program
```

END

```
! myinput:
    ;perform the needed actions here
    mov a, sbuf ;serial input buffer to acc
    ret
```

\$SERIALINPUT2LCD

Action

이 컴파일러 지시어는 시리얼포트 로의 에코 대신 모든 시리얼 포트의 입력을 LCD디스플레이로 가도록 방향전환합니다.

Syntax

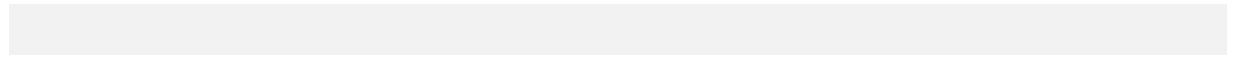
\$SERIALINPUT2LCD

Remarks

\$SERIALINPUT 과 \$SERIALOUT 문을 사용하여 사용자 입력 또는 출력 프로그램을 작성할 수 있습니다. 그러나 LCD를 사용하고 있다면 \$SERIALINPUT2LCD를 사용하는 것이 편리합니다.

EXAMPLE

```
$SERIALINPUT2LCD
Dim v as Byte
CLS
INPUT "Number ", v    ' This will goto the LCD display
```



\$SERIALOUTPUT

Action

시리얼 출력을 방향 전환합니다.

Syntax

\$SERIALOUTPUT = label

Remarks

label	한 개의 문자가 시리얼버퍼(SBUF)로 보낼 때 호출되는 어셈블리 루틴의 이름입니다. 문자는 ACC에 있습니다.
-------	---

PRINT 문의 방향전환과 다른 시리얼출력의 방향전환에 관계되는 명령은 사용자가 작성한 프로그램 루틴이 있어야 합니다.

방향전환은 시리얼 포트 대신 사용자 전용의 출력장치를 사용가능토록 하여줍니다.

Example

```
$SERIALOUTPUT = MyOutput
    'here goes your program
    END

! myoutput:
    ;perform the needed actions here
    mov sbuf, a    ;serial output buffer    (default)
    ret
```

\$SIM

Action

시뮬레이터를 위한 대기(waiting)루프 없는 코드를 생성합니다.

Syntax

\$SIM

Remarks

WAIT 명령을 시뮬레이트할때 많은 시간이 걸리는것을 경험할 것입니다. 사용자는 변수나 소스의 수정없이 \$SIM을 사용하여 WAIT 명령이 빠르게 실행되도록하여 프로그램 개발 시간을 절약할수 있습니다. EPROM에 만든 프로그램을 집어 넣을 때는 반드시 \$SIM 문을 제거 하여야 합니다.

See also

-

Example

```
$SIM                'don't make code for loops
WAIT 2              'the simulator is faster now
```

ABS()

Action

숫자의 절대 값을 반환합니다.

Syntax

var = ABS(var2)

Remarks

Var	Var2의 값을 정수/워드 또는 롱 변수에 지정합니다.
Var2	정수 또는 롱 변수

Var: Byte, Integer, Word, Long

Var2: Integer, Long

절대값은 항상 양의 정수입니다.

See also

-

Example

```
Dim a as Integer, c as Integer
a = -1000
c = Abs(a)
Print c
End
```

Output

1000

ALIAS

Action

변수를 다른 이름으로 지정합니다.

Syntax

newvar ALIAS, oldvar

Remarks

Oldvar	P1.1과 같은 변수 이름
Newvar	지정되는 새로운 변수이름

Port 핀의 이름을 특별한 의미 있는 이름으로 지정합니다.

See also

Example

```
direction ALIAS P1.1 'now you can refer to P1.1 with
                      'the variable direction
SET direction        'has the same effect as SET P1.1
END
```



ASC ()

Action

스트링을 ASCII값으로 변환합니다.

Syntax

var = ASC(stringr)

Remarks

Var	variable.바이트, 정수/워드 변수
Var	스트링 변수 또는 상수

Var: Byte, Integer, Word, Long.

String: String, Constant.

스트링의 첫문자만 사용됩니다.

스트링이 공백이면 0이 반환 됩니다.

See also

CHR()

Example

```
Dim a as byte, s as XRAM String * 10
s = "ABC"
a = Asc(s)
Print a
End
```

Output

65

AVG()

Action

BYTE 형 변수로 구성된 배열의 평균값을 반환합니다.

Syntax

var = AVG(ar(1))

Remarks

Var 평균값을 저장할 변수 입니다.

Ar()

See also

MAX , MIN

Example

```
Dim ar(10) As Byte
Dim bP as Byte
```

```

For bP = 1 to 10
  ar(bP) = bP
Next
bP = Avg(ar(1))
Print bP
End

```

BCD ()

Action

정수/워드 변수 값을 BCD상수 값으로 변환합니다.

Syntax

PRINT BCD(var)
LCD BCD(var)

Remarks

Var	변경하려는 변수
-----	----------

var1: Byte, Integer, Word, Long, Constant.

I2C 클록 장치는 BCD형태로 값을 저장합니다. 이 함수를 사용하여야 정확하게 프린트 됩니다.
 BCD()는 마지막의 0 이 있어야 디스플레이될 것입니다.

BCD()함수는 PRINT/LCD명령을 지원하기 위한것입니다. 변수를 변경하기 위하여는 MAKEBCD를 사용합니다.

See also

MAKEBCD, MAKEDEC

Example

```

Dim a as byte
a = 65
LCD a
Lowerline
LCD BCD(a)
End

```

BITWAIT

Action

비트변수가 셋트 또는 리셋되는 것을 기다립니다.

Syntax

BITWAIT x SET/RESET

Remarks

X	Bit 변수 또는, P1.x 이며 x 는 0부터 7까지
---	--------------------------------

비트변수를 사용할 때 소프트웨어에 의하여 세트/리셋 됩니다.

See also

-

Example

```
Dim a as bit
BITWAIT a , SET      'wait until bit a is set
BITWAIT P1.7, RESET  'wait until bit 7 of Port 1 is 0.
End
```

ASM

```
BITWAIT P1.0 , SET will generate:
Jnb h '91, *+0
```

```
BITWAIT P1.0, RESET will generate :
Jb h'91, *+0
```

BREAK

Action

시뮬레이터를 멈추기 위한 예정된 코드를 만들어 냅니다.

Syntax

BREAK

Remarks

시뮬레이터에서 브레이크 포인터를 설정할 수 있습니다. 그러나 **BREAK** 커맨드에 의해서도 설정할 수 있습니다.

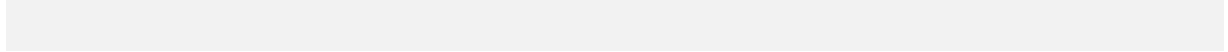
\$NOBREAK 메타 커맨드를 사용하였거나 디버그가 완료되었다면 **BREAK** 문을 제거 하십시오.

See also

\$NOBREAK

Example

```
PRINT "Hello"
BREAK          'the simulator will pause now
.....
.....
End
```



CALL

Action

서브루틴을 호출하고 실행합니다.

Syntax

CALL Test [(var1, var-n)]

Remarks

var1	BASCOS 의 변수 또는 상수 입니다.
Var-n	BASCOS 의 변수 또는 상수 입니다.
Test	서브루틴의 이름입니다. 이 경우 Test 입니다.

Call문에 의하여 프로시저나 서브루틴을 호출할 수 있습니다.

10개의 파라메터를 전달할 수 있지만 파라메타 없는 호출도 가능합니다.

For examples : **Call Test2**

call문은 사용자 전용으로 적용 하는 것이 가능합니다.

CALL문을 사용할 필요가 없다.

Test2 는 서브루틴 test2를 호출 합니다.

CALL 문을 사용하지 않을 경우 괄호를 없애주어야 합니다.

즉 Call Routine(x,y,z)는 반드시 Routine x,y,z 로 해야 합니다.

See also

DECLARE, SUB

Example

```
Dim a as byte, b as byte
```

```
Declare Sub Test(b1 as byte)
```

```
a = 65
```

```
Call test (a)           'call test with parameter A
```

```
test a                  'alternative call without call statement
```

```
End
```

```
SUB Test(b1 as byte)    'use the same variable as the declared one
```

```
    LCD b               'put it on the LCD
```

```
    Lowerline
```

```
    LCD BCD(b1)
```

```
End SUB
```

CHR()

Action

바이트, 정수, 워드 변수나 상수를 문자로 변환 합니다.

Syntax

PRINT CHR(var)
s = CHR(var)

Remarks

var	바이트, 정수, 워드변수 또는 상수
S	스트링 변수

스크린이나 LCD-디스플레이에 문자를 출력할 때 **CHR()**를 사용 하여야 합니다.

See also

-

Example

```
Dim a as byte
a = 65
LCD a
Lowerline
LCDHEX a
LCD Chr(a)
End
```

CLS

Action

LCD-디스플레이를 클리어하고 커서를 홈 위치에 놓습니다.

Syntax

CLS

Remarks

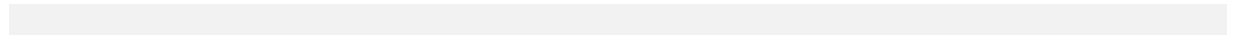
LCD디스플레이를 클리어합니다. 그러나 **CG-RAM**에있는 사용자전용의 문자는 크리어되지 않습니다.

See also

\$LCD, LCD

Example

```
Cls
LCD "Hello"
End
```



CONST

Action

심볼릭 상수를 선언합니다.

Syntax

`DIM symbol AS CONST value`

Remarks

symbol	심볼 이름
Value	심볼에 지정될 수치값

상수의 지정은 프로그램 메모리를 점유하지 않습니다.
컴파일러는 심볼이 나타날 때 값을 지정합니다.

See also

Example

```

'-----
'                (c) 1997-2000 MCS Electronics
'                CONST.BAS
'-----
Dim A As Const 5           'declare a as a constant
Dim B1 As Const &B1001
Waitms A                   'wait for 5 milliseconds
Print A
Print B1
End

```

CONFIG

Config 문은 실행문과 관련된 모든 종류의 하드웨어를 구성합니다.

CONFIG TIMER0, TIMER1
 CONFIG TIMER2 (for 8052 compatible chips)
 CONFIG LCD
 CONFIG LCDBUS
 CONFIG LCDPIN
 CONFIG BAUD
 CONFIG 1WIRE
 CONFIG SDA
 CONFIG SCL
 CONFIG DEBOUNCE
 CONFIG WATCHDOG
 CONFIG SPI

CONFIG TIMER0, TIMER1

Action

TIMER0 또는 TIMER1 을 구성합니다.

Syntax

CONFIG TIMERx = COUNTER/TIMER , GATE=INTERNAL/EXTERNAL , MODE=0/3

Remarks

timerx	TIMER0 or TIMER1. TIMERx 를 COUNTER로 설정하면 COUNTER가 되고 TIMERx 를 TIMER 로 설정하면 TIMER가 됩니다. TIMER는 내부클럭을 사용하며 COUNTER는 외부클럭을 사용합니다.
GATE	INTERNAL or EXTERNAL. INTERNAL과 EXTERNAL 을 지정합니다. EXTERNAL 을 지정하면 INT 입력에 의한 게이트 컨트롤이 가능합니다.
MODE	Timer/Counter 모드 0-3 입니다. 자세한것은 자료를 참고하십시오.

예를들면 **CONFIG TIMER0 = COUNTER, GATE = INTERNAL, MODE=2** 는 TIMER0를 외부게이트
컨트롤을 허가하지 않는 mode2(auto reload) 카운터 모드로 설정합니다.

Timer/Counter 가 정지된 상태에서 Timer/Counter를 설정하는 것이므로 후에는 반드시 **START
TIMERx** 문을 사용 하여야 합니다.

CONFIG문을 사용하기위한 자세한 내용은 프로세서 하드웨어 관계자료를 참고 하십시오.

Example

```
CONFIG TIMER0=COUNTER, MODE=1, GATE=INTERNAL
COUNTER0 = 0           'reset counter 0
START COUNTER0         'enable the counter to run
DELAY                  'wait a while
PRINT COUNTER0         'print it
END
```

CONFIG LCD

Action

Syntax

CONFIG LCD = LCDtype

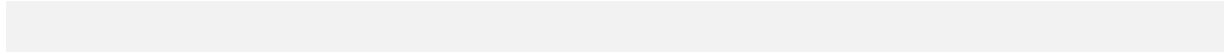
Remarks

LCDtype	LCD 디스플레이의 종류를 선택합니다. 지정되지 않으면 내장 값은 16*2 입니다. 40 * 4, 16 * 1, 16 * 2, 16 * 4, 16 * 4, 20 * 2 or 20 * 4
---------	---

Example

```
CONFIG LCD = 40 * 4
LCD "Hello"           'display on LCD
FOURTHLINE           'select line 4
```

```
LCD "4"          'display 4  
END
```



CONFIG LCDBUS

Action

LCD데이터버스의 폭을 설정합니다.

Syntax

CONFIG LCDBUS = constant

Remarks

Constant	4 비트모드일때 4 , 8 비트모드일때 8(내정치)
----------	------------------------------

이 명령은 \$LCD=address 문과 함께사용합니다. Bus모드로 LCD를 사용하면 모든 데이터 라인을 연결합니다. 4-비트 모드일때 LCD 의 D7-D4라인만 연결합니다.

See Also

CONFIG LCD

Example

```
$LCD = &H8000 ' address of enable signal
Config LCDBUS = 4 '4 bit mode
```

CONFIG BAUD

Action

프로세서의 통신속도를 설정합니다.

이명령어는 단지 80535, 80537 에서 적용됩니다.

Syntax

CONFIG BAUD = baudrate

Remarks

Baudrate	Baudrate to use : 4800 or 9600
----------	--------------------------------

Example

```
CONFIG BAUD = 9600 'use internal baud generator
Print "Hello"
End
```

CONFIG 1WIRE

Action

1 WIRE 문을 사용하기위한 설정입니다.

Syntax

CONFIG 1WIRE = pin

Remarks

pin	P1.0과같이 포트 핀을 지정합니다.
-----	----------------------

See also

1WRESET , 1WREAD , 1WWRITE

Example

```
Config 1WIRE = P1.0    'P1.0 is used for the 1-wire bus
1WRESET                'reset the bus
```

CONFIG SDA

Action

선택 사양설정에서 SDA 핀을 추가합니다.

Syntax

CONFIG SDA = pin

Remarks

Pin	I2C-SDA라인에 연결될 포트핀입니다.
-----	------------------------

다른 작업에서(프로그램 모듈, 또는 서브루틴)새로운 핀을 SDA로 지정할 수 있습니다. 선택사항이 변경되기 전에는 한번 설정한 것이 계속 유효하므로 어떤 핀이 지정 상태인가는 항상 기억하여야 합니다.

See also

CONFIG SCL

Example

```
CONFIG SDA = P3.7      'P3.7 is the SDA line
```

CONFIG SCL

Action

선택사양 설정에서 SCL핀이 추가/변경 되는 것을 지정합니다.

Syntax

CONFIG SCL = pin

Remarks

pin	I2C-SCL 라인에 연결될 포트핀 입니다.
-----	--------------------------

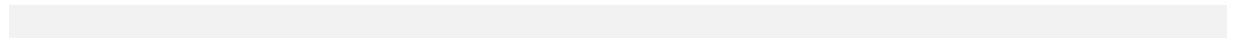
다른 작업에서(프로그램 모듈, 또는 서브루틴)새로운 핀을 SCL로 지정할 수 있습니다. 선택사항이 변경되기 전에는 한번 설정한 것이 계속 유효하므로 어떤 핀이 지정 상태인가는 항상 기억하여야 합니다.

See also

CONFIG SDA

Example

```
CONFIG SCL = P3.5      'P3.5 is the SCL line
```



CONFIG DEBOUNCE

Action

DEBOUNCE 문에서 딜레이 타임을 설정합니다.

Syntax

CONFIG DEBOUNCE = time

Remarks

time	MS 단위의 숫자 상수 입니다.
------	-------------------

Debounce 시간이 설정되지 않으면 내장 값은 25mS입니다.
딜레이 타임은 12MHz 의 클럭 주파수를 기준으로 계산한 것입니다.

See also

DEBOUNCE

Example

```
Config Debounce = 25 mS           '25 mS is the default
```

CONFIG SPI

Action

SPI 문에 대한 핀 설정 합니다.

Syntax

CONFIG SPI = soft , DIN = Pin, DOUT = Pin, CS = Pin, CLK = Pin

Remarks

DIN	데이터 입력핀 이며 P1.0과 같은 포트 핀 이름 입니다.
DOUT	데이터 출력핀이며 P1.1과 같은 포트핀 이름입니다.
CS	칩 셀렉트 핀이며 P1.2 와 같은 포트핀이름입니다.
CLK	클록핀이며 P1.3과 같은 포트핀 이름입니다..

See also

SPIIN SPIOUT

Example

```
Config SPI = SOFT, DIN = P1.0, DOUT = P1.1, CS = P1.2, CLK = P1.3
SPIOUT var1, 1 ' send 1 byte
```

CONFIG LCDPIN

Action

LCD 옵션을 설정 합니다.

Syntax

CONFIG LCDPIN, DB4 = P1.1, DB5= P1.2, DB6 = P1.3, DB7 = P1.4, E = P1.5, RS = P1.6

Remarks

P1.1 등은 단순한 사용예 입니다.

Example

CONFIG LCDPIN, DB4 = P1.1, DB5= P1.2, DB6 = P1.3, DB7 = P1.4, E = P1.5, RS = P1.6

CONFIG WATCHDOG

Action

AT89C8252의 워치독 타이머를 설정합니다.

Syntax

CONFIG WATCHDOG = time

Remarks

time	시간 상수값의 단위는 mS입니다. 다음의 값이 설정 가능 합니다. 16 , 32, 64 , 128 , 256 , 512 , 1024 , 2048.
------	--

워치독 타이머가 동작되면 지정된 시간에 리셋이 발생합니다.

예를 들어 2048로 설정한 경우 2 초 후에 리셋이 발생합니다. 따라서 프로그램에서 주기적으로 워치독 타이머를 리셋해 주어야 합니다.

See also

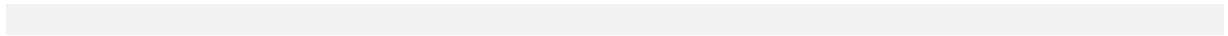
START WATCHDOG, STOP WATCHDOG, RESET

Example

```

'-----
'                               (c) 1998 MCS Electronics
' WATCHD.BAS demonstrates the AT89S8252 watchdog timer
' select 89s8252.dat !!!
'-----
Config Watchdog = 2048                'reset after 2048 mSec
Start Watchdog                        'start the watchdog timer
Dim I As Word
For I = 1 To 10000
    Print I                            'print value
    ' Reset Watchdog
    'you will notice that the for next doesnt finish
    'because of the reset
    'when you unmark the RESET WATCHDOG statement
    'it will finish because the
    'wd-timer is reset before it reaches 2048 msec
Next
End

```



COUNTERX

Action

COUNTER0 또는 COUNTER1 을 설정하거나 , 수정합니다.
8052 TIMER2 호환 칩에서 COUNTER2 를 사용할 수 있습니다.

Syntax

COUNTERx = var or var = **COUNTER**

Remarks

var	바이트, 정수/워드변수 또는 상수이며 카운터에 지정됩니다.
COUNTERx	COUNTER0, COUNTER1 또는 COUNTER2

CounterX= 0 는 카운터를 리셋합니다.

Mode 2 에서 카운터는 0 부터 255 까지 카운팅합니다.(8-비트 오토 리로드 모드)

모드 1(16비트)에서는 65536 입니다.

COUNTERx 변수는 BASCOM에서 TIMER/COUNTER레지스터를 설정하거나 값을 가져 오기위한 명령입니다. COUNTER0 = TLO 와 TH0 이므로 COUNTERx 는 16비트 데이터형입니다.

COUNTERx 변수를 사용한 후에는 START COUNTERx명령을 사용하여 카운터를 시작하여야 합니다.

```
'-----
'
'              (c) 1997-2001 MCS Electronics
'-----
' file: COUNTER.BAS
' demo: COUNTER
'-----
'   Connect the timer input P3.4 to a frequency generator
'   *TIMER/COUNTER 1 is used for RS-232 baud rate generator
'-----
```

Dim A As Byte , C As Integer

Config Timer0 = Counter , Gate = Internal , Mode = 1

'Timer0 = counter : timer0 operates as a counter

'Gate = Internal : no external gate control

'Mode = 1 : 16-bit counter

```
Counter0 = 0           'clear counter
Start Counter0         'enable the counter to count
Do
  A = Inkey            'set up a loop
  C = Counter0         'check for input
  Print C              'get counter value
  Start Counter0       'print it
  Loop Until A = 27    're-start it because it was stopped by accessing the COUNTER
                      'until escape is pressed
```

End

다음의 예는 어셈블러로 변환된 코드를 보여줍니다.

COUNTER0 = 1000

만들어진 코드:

Clr TCON.4

Mov tl0, #232

Mov th0, #3

CPEEK()

Action

코드메모리에 있는 값을 반환합니다.

Syntax

var = **CPEEK**(**address**)

에디타로 프로그램을 편집할 경우 라인의 길이 제한은 없습니다. 그러나 라인당 80 문자를 권장합니다.

Remarks

var	지정된 어드레스의 프로그램 메모리로부터 가져온 데이터를 저장할 변수
address	어드레스 위치를 나타내는 숫자값.

ROM 프로그램 메모리에는 데이터를 써 넣을 수 없기 때문에 CPOKE 명령은 없습니다.

See also

PEEK, POKE, INP, OUT

Example

```
'-----
'
'              (c) 1998 MCS Electronics
'              PEEK.BAS
' demonstrates PEEK, POKE, CPEEK, INP and OUT
'
'-----
Dim I As Integer , B1 As Byte

'dump internal memory
For I = 0 To 127                                'for a 8052 225 could be
used                                              used
'  Break
  B1 = Peek(i)                                    'get byte from internal
memory
  PrintheX B1 ; " ";
  'Poke I , 1                                     'write a value into memory
Next
Print                                             'new line
'be careful when writing into internal memory !!
```

CURSOR

Action

LCD-커서의 상태를 설정합니다.

yntax

CURSOR ON / OFF BLINK / NOBLINK

Remarks

ON 이나 OFF 와 BLINK 또는 NOBLINK를 동시에 지정할 수 있습니다.
초기엔 ON과 NOBLINK로 설정되어 있습니다.

See also

DISPLAY

Example

```
Dim a as byte
a = 255
LCD a
CURSOR OFF      'hide cursor
Wait 1          ' wait 1 second
CURSOR BLINK    'blink cursor
End
```

의하여 읽혀질 값을 지정합니다.

Syntax

DATA var [, varn]

Remarks

Var	숫자 또는 스트링 상수
-----	--------------

정수와 워드값은 반드시 %기호로 끝나야 합니다.

롱형상수는 반드시 &로 끝나야 합니다.

단정도 부동소수점상수(single) 은 반드시 !로 끝나야 합니다.

See also

READ, RESTORE

Example

```
DIM a AS BYTE, I AS BYTE, L as Long S, As XRAM String * 15
RESTORE Dta          'point to data
FOR a = 1 TO 3
  READ a : PRINT a    ' read data and print it
NEXT
RESTORE Dta2          'point to data
READ I : PRINT I
READ I : PRINT I
RESTORE Dta3
READ L : PRINT L
Restore Dta3:
Read S: Print S
END
```

DTA1:

DATA 5, 10, 100

DTA2:

DATA -1%, 1000% 'Integers and Words must end with the %-sign. (Integer : <0 or >255)

DTA3:

DATA 12345678& 'A Long must end with the &-sign.

```
DTA4:  
DATA "Hello Word"  
REM You can also mix different constant types on one line  
DATA "TEST", 5, 1000%, -1&, 1.1!
```



DEBOUNCE

Action

스위치에 연결되는 포트핀의 디바운스 설정합니다.

Syntax

DEBOUNCE Px.y , state , label [, SUB]

Remarks

Px.y	P1.0과같은 포트핀 입니다.
State	0 이면 Px,y 가 로우 상태 일때, 그리고 1 이면 Px,y가 하이 상태 일때 점프합니다
label	지정한 상태가 검출되었을 때 GOTO에 대한 라벨입니다.
SUB	지정한 상태가 검출되었을 때 GOSUB에 대한 라벨입니다.

선택적 파라메타인 SUB를 지정 하였을 경우 GOTO대신에 GOSUB에 대한 라벨(Return으로 종료되는 서브루틴)이 필요합니다.

DEBOUNCE문은 포트핀이 하이(1) 또는 로우(0)될때까지 대기합니다.
25mS 이후에 다시 포트상태를 확인합니다. (스위치의 채터링 노이즈 제거)

DEBOUNCE 문 은 개별적인 1 개 포트에 대하여 1 개 비트 메모리를 사용합니다.

See also

CONFIG DEBOUNCE

Example

```

'-----
'                               DEBOUN.BAS
'                               Demonstrates DEBOUNCE
'-----
Config Debounce = 30           'when the config statement is not
                                'used a default of 25mS will be used

'Debounce P1.1 , 1 , Pr 'try this for branching when high(1)
Debounce P1.0 , 0 , Pr , Sub
'           ^----- label to branch to
'           ^----- Branch when P1.0 goes low(0)
'           ^----- Examine P1.0

'When P1.0 goes low jump to subroutine Pr
'P1.0 must go high again before it jumps again
'to the label Pr when P1.0 is low

Debounce P1.0 , 1              'no branch
Debounce P1.0 , 1 , Pr        'will result in a return without gosub
End

Pr:
  Print "P1.0 was/is low"
Return

```



DEC R

Action

변수의 값을 1 감소합니다.

Syntax

DEC R var

Remarks

Var	정수/워드 또는 바이트변수 입니다.
-----	---------------------

변수값을 1 감소 할 때 사용합니다. **DEC R**문은 `var = var - 1` 보다 빠릅니다.

See also

```
'-----
'          (c) 1997-2001 MCS Electronics
'-----
' file: DEC R.BAS
' Demo: DEC R
'-----
```

Dim A As Byte

```
A = 5                'assign value to a
Decr A               'decrease (by one)
Print A              'print it
End
```

DECLARE SUB

Action

서브루틴을 선언합니다.

Syntax

DECLARE SUB TEST[(var as type)]

Remarks

test	프로시저의 이름입니다.
Var	변수의 이름이며 최대 10 개까지 허용합니다.
Type	변수의 형이며 비트, 바이트, 워드 또는 정수변수 입니다.

서브 프로시저를 기술하기 전에 반드시 선언 하여야 합니다.

See also

,

Example

```
Dim a as byte, b1 as byte, c as byte
Declare Sub Test(a as byte)
a = 1 : b1 = 2: c = 3

Print a ; b1 ; c
```

```

Call Test(b1)
Print a ;b1 ; c
End

Sub Test(a as byte)
    print a ; b1 ; c
End Sub

```

DEFINT, DEFBIT, DEFBYTE, DEFWORD

Action

변수의 형을 선언합니다.

Syntax

```

DEFBIT b
DEFBYTE c
DEFINT I
DEFWORD x

```

Example

```

Defbyte b : DefInt c 'default type for bit and integers
Set b1 'set bit to 1
c = 10 'let c = 10

```

DEFLCDCHAR

Action

사용자 고유의 LCD문자를 정의합니다.

Syntax

```
DEFLCDCHAR char,r1,r2,r3,r4,r5,r6,r7,r8
```

Remarks

char	바이트, 정수/워드 변수 또는 문자를 나타내는 상수(0-7)
r1-r8	문자의 열(row)값

Char: Byte, Integer, Word, Long, Constant

r1-r8: 상수

LCD문자 설계 워저드를 이용하여 LCD문자를 설계할 수 있습니다.

See also

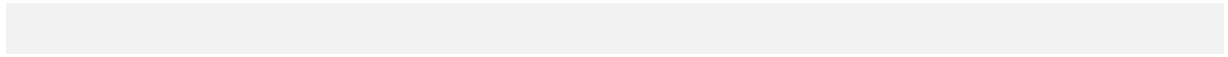
Edit LCD designer

Example

```

DefLCDchar 0,1,2,3,4,5,6,7,8 'define special character
CLS
LCD Chr(0) 'show the character
End

```



DELAY

Action

짧은 시간의 지연프로그램 실행

Syntax

DELAY

Remarks

짧은 시간을 대기하기 위하여 DELAY를 사용합니다.

시스템 주파수가 12M인 경우 딜레이 타임은 100uS가 단위 시간입니다.

See also

WAIT, WAITMS

Example

```
P1 = 5      'write 5 to port 1
DELAY      'wait for hardware to be ready
```

DIM

Action

변수의 차원선언을 선언합니다.

Syntax

DIM var AS [XRAM / IRAM] type

Remarks

var	Longname, j, b1 과 같은 어떤 유효한 변수
Type	비트, 바이트, 워드, 정수 또는 스트링
XRAM	변수를 외부 메모리에 저장할 때 XRAM을 지정한다.
IRAM	변수를 내부메모리에 저장할때 IRAM을 지정합니다.(내정치)

스트링 변수는 길이 파라메타를 요구합니다. Dim s As XRAM String * 10 은 10개의 문자로 스트링이 구성됨을 나타냅니다.

BITS는 내부메모리에만 저장됩니다.

See Also

CONST, ERASE

Example

```
'-----
'                (c) 1997-2001 MCS Electronics
'-----
' file: DIM.BAS
' demo: DIM
'-----
```

```
Dim B1 As Bit           'bit can be 0 or 1
Dim A As Byte           'byte range from 0-255
Dim C As Integer        'integer range from -32767 - +32768
Dim L As Long
Dim S As Single
'Assign bits
```

```

B1 = 1                                'or
Set B1                                'use set

'Assign bytes
A = 12
A = A + 1

'Assign integer
C = -12
C = C + 100
Print C

'Assign long
L = 12345678
Print L

'Assign single
S = 1234.567
Print S

End

```

DISABLE

Action

지정된 인터럽트의 사용을 중지합니다.

Syntax

DISABLE interrupt

Remarks

Interrupt	INT0, INT1, SERIAL, TIMER0, TIMER1 or TIMER2.
-----------	--

초기상태에는 내정 값으로 모든 인터럽트는 금지되어 있습니다.
모든 인터럽트를 금지시키려면 **INTERRUPTS**를 지정합니다.

See also

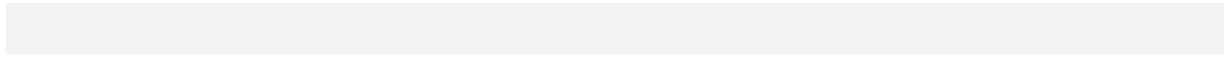
ENABLE

Example

```

ENABLE INTERRUPTS    'enable the setting of interrupts
ENABLE TIMER0        ' enable TIMER0
DISABLE SERIAL        'disables the serial interrupt.
DISABLE INTERRUPTS    'disable all interrupts

```



DISPLAY

Action

LCD-디스플레이를 on 또는 off 합니다.

Syntax

DISPLAY ON | OFF

Remarks

전원 ON시 디스플레이는 ON 됩니다.

See also

-

Example

```
Dim a as byte
a = 255           'assign variable
LCD a             'show on LCD
DISPLAY OFF       'turn display off
Wait 1            'wait 1 second
DISPLAY ON        'turn in back on
End
```

DO..LOOP

Action

조건이 참(TRUE)을 만족 할 때까지 실행문의 블록을 반복실행 합니다.

Syntax

```
DO
  statements
LOOP [ UNTIL expression ]
```

Remarks

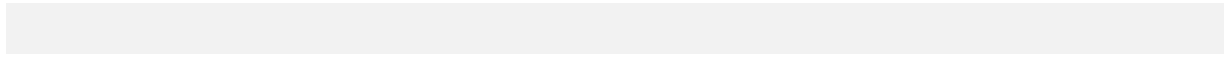
EXIT DO 문에 의하여 DO.. LOOP를 빠져나올 수 있습니다.

See also

’, ’

Example

```
DO                               'start the loop
  A = A + 1                     'increment A
  PRINT A                       'print it
LOOP UNTIL A = 10              'Repeat loop until  A = 10
End
```



ELSE

Action

IF-THEN문이 거짓(FALSE)일때 실행되는 부분을 나타냅니다.

Syntax

ELSE

Remarks

또 다른 조건을 검사할 때 ELSEIF문을 사용합니다.

```
IF a = 1 THEN
...
ELSEIF a = 2 THEN
..
ELSEIF b1 > a THEN
...
ELSE
...
END IF
```

See also

IF, END IF, SELECT CASE

Example

```
A = 10                                'let a = 10
IF A > 10 THEN                         'make a decision
    PRINT " A >10"                    'this will not be printed
ELSE                                   'alternative
    PRINT "A not greater than 10"     'this will be printed
END IF
```

ENABLE

Action

지정한 인터럽트를 가능하게 합니다.

Syntax

ENABLE interrupt

Remarks

Interrupt	INT0, INT1, SERIAL, TIMER0, TIMER1 or TIMER2.
-----------	---

처음에는 모든 인터럽트는 사용불가입니다.

ENABLE INTERRUPTS. 에 의하여 지정한 인터럽트가 가능 상태로 됩니다.

See also

Example

```
ENABLE INTERRUPTS                    ' allow interrupts to be set
ENABLE TIMER1                        'enables the TIMER1 interrupt
```

END

Action

프로그램의 실행을 종료합니다.

Syntax

END

Remarks

STOP 은 **END** 와 같이 프로그램을 종료합니다.

END 또는 **STOP** 문은 모든 인터럽트를 중지하고 무한루프 상태로 됩니다.

See also

Example

```
PRINT "Hello"      'print this
END                'end program execution
```

END IF

Action

IF THEN 구조의 종료를 나타냅니다.

Syntax

END IF or **ENDIF**

Remarks

IF THEN 구조는 반드시 **END IF**로 끝나야 합니다.

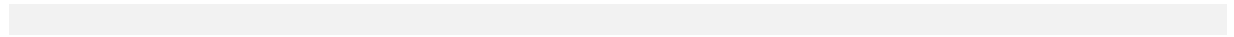
IF THEN문을 내포할 수 있습니다.

ELSE 를 선택적으로 사용할 수 있습니다.

ENDIF 는 **END IF**로 변환됩니다.

Example

```
Dim nmb As Byte
AGAIN:  'label
INPUT "Number " , nmb      'ask for number
IF a = 10 THEN              'compare
    PRINT "Number is 10"    'yes
ELSE                          'no
    IF nmb > 10 THEN         'is it greater
        PRINT "Number > 10" 'yes
    ELSE                    'no
        PRINT "Number < 10" 'print this
    END IF                  'end structure
END IF                      'end structure
END                          'end program
```



ERASE

Action

변수가 소거됩니다. 따라서 메모리가 해지 됩니다. (다른 변수를 사용하기 위한 여유메모리가 증가)

Syntax

ERASE *var*

Remarks

var	소거되는 변수의 이름
------------	-------------

소거되기 전에 변수는 반드시 정의되어 있어야 합니다.

일시적으로 변수를 사용한 경우 소거할 수 있습니다. 이렇게 하면 메모리를 절약할 수 있습니다.

변수의 소거는 항상 마지막으로 정의된 것만 가능합니다. 따라서 새로운 **DIM** 이 정의되면 이변수를 먼저 소거하여야 합니다.

예를 들어:

```
Dim a1 as byte , a2 as byte , a3 as byte , a4 as byte
```

```
'use the vars
```

```
ERASE a3 : ERASE a4          ' 마지막 2 개의 변수를 소거
```

```
Dim a5 as Byte 'Dim new var
```

이제 **a1** 과 **a2**는 소거하지 못합니다.

Example

```
DIM A As Byte          'DIM variable
A = 255                 'assign value
Print A                 'PRINT variable
ERASE A                 'ERASE
DIM A AS INTEGER        'DIM again but now as INT
PRINT A                 'PRINT again
REM Note that A uses the same space a the previous ERASED var A so
REM it still holds the value of the previous assigned variable
```

EXIT

Action

FOR..NEXT, DO..LOOP , WHILE ..WEND 또는 SUB..END SUB 에서 빠져나갑니다.

Syntax

EXIT [**FOR**] [**DO**] [**WHILE**] [**SUB**]

Remarks

EXIT은 어떠한 경우든 구조화 문에서 빠져나갑니다.

Example

```
IF a >= b1 THEN          'some silly code
    DO                    'begin a DO..LOOP
        A = A + 1         'inc a
        IF A = 100 THEN   'test for a = 100
            EXIT DO       'exit the DO..LOOP
        END IF            'end the IF..THEN
    LOOP                  'end the DO
END IF                    'end the IF..THEN
```

FOR

Action

지정된 수만큼 블록의 실행을 반복합니다.

Syntax

FOR var = start **TO** / **DOWNTO** end [**STEP** value]

Remarks

var	계수에 사용될 변수 이름입니다.
Start	변수의 초기 시작 값입니다.
End	변수의 종료 값입니다.
Value	변수는 NEXT 문을 만날 때마다 증가됩니다.

Var: Byte, Integer, Word, Long, Single.

Start: Byte, Integer, Word, Long, Single, Constant

end: Byte, Integer, Word, Long, Single, Constant

step: Byte, Integer, Word, Long, Single, Constant

증가형 루프는 **TO** 를 사용합니다.

감소형 루프는 **DOWNTO**를 사용합니다.

FOR 문은 반드시 **NEXT**문으로 끝나야 합니다.

STEP은 선택사항 입니다. 지정하지 않으면 1이 사용됩니다.

See also

Example

```

y = 10                                'make y 10
FOR a = 1 TO 10                        'do this 10 times
    FOR x = y TO 1                    'this one also
        PRINT x ; a                  'print the values
    NEXT                             'next x (count down)
NEXT                                  'next a (count up)
END

```

FOURTHLINE

Action

LCD의 4 번째 줄에 커서가 오게 합니다.

Syntax

FOURTHLINE

Remarks

LCD 디스플레이가 4 줄인 경우에 해당됩니다.

See also

HOME, UPPERLINE, LOWERLINE, THIRDLINE, LOCATE

Example

```

Dim a as byte
a = 255
LCD a
Fourthline

```

LCD a
 Upperline
 END

FUSING

Action

부동소수점의 값을 형식에 맞춥니다.

Syntax

var = Fusing(source, mask)

Remarks

Var	결과를 저장할 스티링입니다.
Source	포맷할 부동소수점 변수입니다.
Mask	포맷마스크입니다 ###.## # 부호는 디지트 소수점 이전 자리수와 이후 자리수를 표시하는데 사용합니다. Normal 라운딩이 적용됩니다.

See also

STR

Example

\$large

Dim X As Single, Y As Single, Result As Single

Dim I As Integer

Dim Buf As String *16

Input "Enter x ", X

' ask for 2 values

Input "Enter y ", Y

Print "X+Y=" ; : Result = X + Y : Print Result

' calculate

Print "X-Y=" ; : Result = X - Y : Print Result

Print "X/Y=" ; : Result = X / Y : Print Result

Print "X*Y=" ; : Result = X * Y : Print Result

Buf = Fusing(result , #.##)

' format a string

Print Buf

' print it



GET – PUT - TIMEOUT

Action

소프트웨어 UART에서 문자를 가져옵니다.

Syntax

GET #channel , var

Remarks

Channel 오픈 채널을 지시하는 번호(양정수) 입니다.

Var 소프트웨어 UART에서 가져온 문자를 저장할 변수입니다.

OPEN 문에 의하여 채널을 사용가능상태로 설정하여야 합니다. 또한 프로그램의 마지막에 CLOSE 문을 사용하여 채널을 닫아야 합니다. TIMEOUT 문을 사용하여 아무런 문자가 입력되지 않았을 때 빠져 나오도록 할 수 있습니다.

See also

PUT \$TIMEOUT

Example

Dim S As String * 12 , I As Byte , A As Byte , Dum As Byte

Open "com3.1:9600" For Output As #1 'p3.1 is normally used for TX so testing is easy

Open "com3.0:9600" For Input As #2 'p3.0 is normally used for RX so testing is easy

```
S = "test this" 'assign string
Dum = Len(s) 'get length of string
For I = 1 To Dum 'for all characters from left to right
  A = Mid(s , I , 1) 'get character
  Put #1 , A 'write it to comport
Next
```

```
Do
  Get #2 , A 'get character from comport
  Put #1 , A 'write it back
  Print A 'use normal channel
Loop
```

```
Printbin #1, a 'Printbin is also supported
Inputbin #2, a 'Inputbin is also supported
```

```
Close #1                                ' finally close device
Close #2
End
```

'To use the TIMEOUT option include (without the remarks):

'\$TIMEOUT

' Get #2 , A TIMEOUT = 10000 'get character from comport

GETAD2051

Action

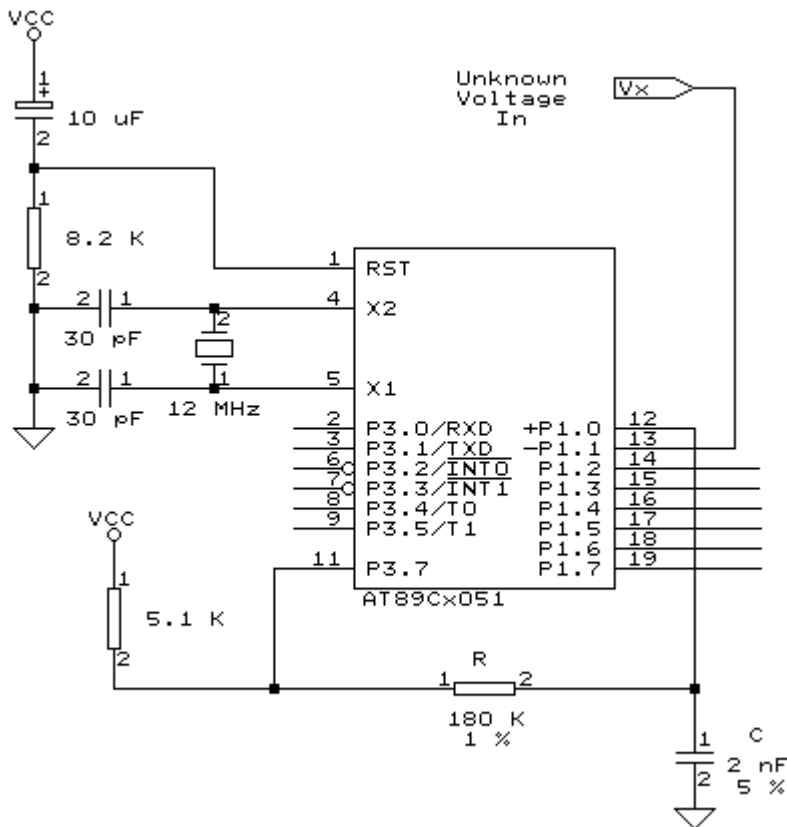
89C2051 또는 89C4051. 프로세서에서 Analog 값을 가져옵니다.

Syntax

```
var = GETAD2051()
```

Remarks

var A/D 값을 저장할 변수입니다.



The GETAD2051() 는 Comparator기능을 가진 89C2051 과 89C4051 프로세서에만 적용됩니다. 그림과 같은 하드웨어 구성을 하여야 합니다.

GETAD2051() 는 저항-콘덴서의 충전곡선을 이용하여 Analog 값을 읽는 방법이므로 예제를 참고하여 직선성이 나오도록 Data 값을 조정하여야 합니다. 6 비트의 AD 컨버터로 사용이 가능합니다.

See also

Example

```
$regfile = "89c2051.dat"
Dim A As Byte
Do
  A = Getad2051()
  A = Lookup(a , Dta)
  Print A
Loop
End
```

'this table converts the value into a packed BCD value

'this value can be used to diaplay the value on 2 7-segment displays

Dta:

Data 0	' 0 0.000
Data 1	' 1 0.047
Data 1	' 2 0.093
Data 2	' 3 0.138
Data 2	' 4 0.184
Data 3	' 5 0.229
Data 3	' 6 0.273
Data 3	' 7 0.317
Data 4	' 8 0.361
Data 4	' 9 0.404
Data 5	' 10 0.447
Data 5	' 11 0.489
Data 6	' 12 0.531
Data 6	' 13 0.573
Data 6	' 14 0.614
Data 7	' 15 0.655
Data 7	' 16 0.696
Data 8	' 17 0.736
Data 8	' 18 0.776
Data 8	' 19 0.815
Data 9	' 20 0.854
Data 9	' 21 0.893
Data &H10	' 22 0.931
Data &H10	' 23 0.969
Data &H10	' 24 1.006
Data &H11	' 25 1.044
Data &H11	' 26 1.080
Data &H11	' 27 1.117
Data &H12	' 28 1.153
Data &H12	' 29 1.189
Data &H12	' 30 1.224
Data &H13	' 31 1.260
Data &H13	' 32 1.295
Data &H13	' 33 1.329
Data &H14	' 34 1.363
Data &H14	' 35 1.397
Data &H14	' 36 1.431
Data &H15	' 37 1.464
Data &H15	' 38 1.497
Data &H15	' 39 1.530
Data &H16	' 40 1.562
Data &H16	' 41 1.594
Data &H16	' 42 1.626
Data &H17	' 43 1.657
Data &H17	' 44 1.688
Data &H17	' 45 1.719
Data &H18	' 46 1.750
Data &H18	' 47 1.780
Data &H18	' 48 1.810
Data &H19	' 49 1.840
Data &H19	' 50 1.869
Data &H19	' 51 1.898
Data &H19	' 52 1.927

Data &H20	' 53 1.956
Data &H20	' 54 1.984
Data &H20	' 55 2.012
Data &H21	' 56 2.040
Data &H21	' 57 2.068
Data &H21	' 58 2.095
Data &H21	' 59 2.122
Data &H22	' 60 2.149
Data &H22	' 61 2.176
Data &H22	' 62 2.202
Data &H22	' 63 2.228
Data &H23	' 64 2.254
Data &H23	' 65 2.279
Data &H23	' 66 2.305
Data &H23	' 67 2.330
Data &H24	' 68 2.355
Data &H24	' 69 2.379
Data &H24	' 70 2.404
Data &H24	' 71 2.428
Data &H25	' 72 2.452
Data &H25	' 73 2.476
Data &H25	' 74 2.499
Data &H25	' 75 2.523
Data &H26	' 76 2.546
Data &H26	' 77 2.569
Data &H26	' 78 2.591
Data &H50	' 79 5.000
Data &H49	' 80 4.953
Data &H49	' 81 4.907
Data &H48	' 82 4.862
Data &H48	' 83 4.816
Data &H47	' 84 4.771
Data &H47	' 85 4.727
Data &H47	' 86 4.683
Data &H46	' 87 4.639
Data &H46	' 88 4.596
Data &H45	' 89 4.553
Data &H45	' 90 4.511
Data &H44	' 91 4.469
Data &H44	' 92 4.427
Data &H44	' 93 4.386
Data &H43	' 94 4.345
Data &H43	' 95 4.304
Data &H42	' 96 4.264
Data &H42	' 97 4.224
Data &H42	' 98 4.185
Data &H41	' 99 4.146
Data &H41	' 100 4.107
Data &H40	' 101 4.069
Data &H40	' 102 4.031
Data &H40	' 103 3.994
Data &H39	' 104 3.956
Data &H39	' 105 3.920
Data &H39	' 106 3.883

Data &H38	' 107 3.847
Data &H38	' 108 3.811
Data &H38	' 109 3.776
Data &H37	' 110 3.740
Data &H37	' 111 3.705
Data &H37	' 112 3.671
Data &H36	' 113 3.637
Data &H36	' 114 3.603
Data &H36	' 115 3.569
Data &H35	' 116 3.536
Data &H35	' 117 3.503
Data &H35	' 118 3.470
Data &H34	' 119 3.438
Data &H34	' 120 3.406
Data &H34	' 121 3.374
Data &H33	' 122 3.343
Data &H33	' 123 3.312
Data &H33	' 124 3.281
Data &H32	' 125 3.250
Data &H32	' 126 3.220
Data &H32	' 127 3.190
Data &H31	' 128 3.160
Data &H31	' 129 3.131
Data &H31	' 130 3.102
Data &H31	' 131 3.073
Data &H30	' 132 3.044
Data &H30	' 133 3.016
Data &H30	' 134 2.988
Data &H29	' 135 2.960
Data &H29	' 136 2.932
Data &H29	' 137 2.905
Data &H29	' 138 2.878
Data &H28	' 139 2.851
Data &H28	' 140 2.824
Data &H28	' 141 2.798
Data &H28	' 142 2.772
Data &H27	' 143 2.746
Data &H27	' 144 2.721
Data &H27	' 145 2.695
Data &H27	' 146 2.670
Data &H26	' 147 2.645
Data &H26	' 148 2.621
Data &H26	' 149 2.596
Data &H26	' 150 2.572
Data &H25	' 151 2.548
Data &H25	' 152 2.524
Data &H25	' 153 2.501
Data &H25	' 154 2.477
Data &H24	' 155 2.454
Data &H24	' 156 2.431
Data &H24	' 157 2.409

GETRC

Action

저항 또는 캐패시터의 값을 가져옵니다.

Syntax

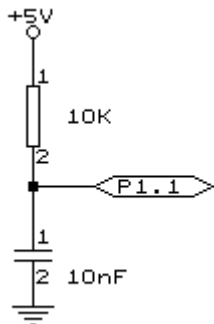
var = **GETRC**(pin)

Remarks

Var	측정값이 저장될 변수입니다.
Channel	RC가 연결된 포트핀입니다.
Range	내부의 측정 범위입니다.

단지 **80515** 와 호환칩에 적용됩니다.

See also



Example

```

'-----
'               GETRC.BAS
' Retrieve resistor value
' Connect 10Kohm variable resistor from +5V to P1.7 for this example
' Connect 10nF capacitor from P1.7 to ground
' The GETRC(pin) function measures the time needed to charge the capacitor
'-----
Config Timer0 = Timer , Gate = Internal , Mode = 1      'the GETRC() functions needs timer 0
Config Getrc = 10                                       '10mS wait for charging the capacitor.
                                                         ' This is the default so for 10 the CONFIG is not needed
                                                         'just my settings
$baud = 9600
$crystal = 11059200
Dim W As Word                                           'allocate space for variable

Do                                                       'forever
  W = Getrc(p1.7)                                       'get RC value
  Print W                                              'print it
  Wait 1                                               'wait a moment
Loop

'return values for cap=10nF .The resistor values where measured with a DVM
'      250 for 10K9
'      198 for 9K02

```

```
'      182 for 8K04
'      166 for 7K
'      154 for 6K02
'      138 for 5K04
'      122 for 4K04
'      106 for 3K06
'      86 for 2K16
'      54 for 1K00
'      22 for 198 ohm
'      18 for 150 ohm
'      10 for 104 ohm
'      6 for 1 ohm (minimum)
```

'As you can see there is a reasonable linearity
 'So you can do some math to get the resistor value
 'But the function is intended to serve as a rough indication for resistor values
 'You can also change the capacitor to get larger values.
 'With 10nF, the return value fits into a byte

GETRC5

Action

RC5 적외선장치로부터 코드와 서브어드레스를 가져옵니다.

Syntax

GETRC5(address, command)

Remarks

address	수신된 RC5 어드레스입니다.
Command	수신된 RC5 커맨드입니다.

SHARP사의 적외선 수신기 SFH506-36을 P3.2에 연결하여 사용합니다.
 이 명령은 INT0를 사용합니다.

```
'-----
'      RC5.BAS (c) 2000 MCS Electronics
' connect SFH506-36 IR-receiver to PORT 3.2 (INT0)
' choose the correct port from the Compiler I2C TAB. Int0 should have P3.2 pin
' On other chips it might be another pin!
'-----
```

Dim New As Bit

Dim Command As Byte , Subaddress As Byte

Reset Tcon.0 'triggered by rising edge

On Int0 Receiverc5

Enable Int0

Enable Interrupts

Do

 If New = 1 Then 'received new code

 Disable Int0

 Print Command ; " " ; Subaddress

 New = 0 'reset new bit

 Enable Int0

 End If

Loop

```
Receiverc5:                                'interrupt routine
'the getrc5 routine uses 30 bytes ! of the stack for measuring
'the interval between the bits

Getrc5(Subaddress,command)
New = 1                                    'set flag
Return
```

GOSUB

Action

서브루틴으로 분기하여 실행합니다.

Syntax

GOSUB label

Remarks

label	분기할 위치의 라벨입니다.
-------	----------------

서브루틴은 반드시 **RETURN**문으로 끝나야 합니다.

See also

Example

```
GOSUB Routine          'branch to routine
Print "Hello"          ' after being at 'routine' print this
END                    'terminate program
Routine:               'this is a subroutine
    x = x + 2           'perform some math
    PRINT X             'print result
RETURN                 'return
```

GOTO

Action

지정된 라벨로 분기합니다.

Syntax

GOTO label

Remarks

라벨은 반드시 32 문자 이내 이어야 합니다.

라벨을 중복 설정하면 컴파일러는 경고 메시지가 나옵니다.

See also

Example

```

Start:          'a label must end with a semicolon
A = A + 1       'increment a
IF A < 10 THEN  'is it less than 10?
    GOTO Start  'do it again
END IF          'close IF
PRINT "Ready"   'that's it

```

HEX()

Action

16진수 값을 스트링으로 변환합니다.

Syntax

var = HEX(x)

Remarks

Var	저장될 스트링 변수 이름
x	Byte, Integer 또는 Word 변수

See also

Example

```

Dim a as Byte, S as String * 10
A = 123
S = Hex(a)
Print s
End

```

HEXVAL()

Action

16진수 헥사 스트링을 숫자 값으로 변환합니다.

Syntax

var = HEXVAL(x)

Remarks

var	저장될 변수 이름
x	변환 가능한 16진 헥사 스트링

See also

, STR

Example

```

Dim a as Integer, s as XRAM String * 15
s = "000A"
a = Hexval(s) : Print a
End

```

HIGH

Action

변수의 상위바이트를 추출합니다.

Syntax

```
var = HIGH( s )
```

Remarks

Var	변수 S 의 상위바이트를 저장할 변수
s	MSB를 가져올 소스변수

See also

Example

```
Dim I As Integer, Z As Byte
I = &H1001
Z = High(I)      ' is 16
```

HIGHW

Action

32비트 정수형 변수에서 상위 16비트 데이터를 가져옵니다.

Syntax

```
var = HIGHW( s )
```

Remarks

Var 16비트 데이터가 저장될 수 있는 WORD 또는 INTEGER 형 변수이어야 합니다.
S 32비트 Long 형 변수입니다.

See also

LOW HIGH LOWW

Example

```
Dim I As Long , Z As Word
I = &H10011001
Z = High(I)
```

HOME

Action

커서를 지정된 라인에 1 번째 위치에 오도록 합니다.

Syntax

HOME UPPER / LOWER / THIRD / FOURTH

Remarks

단지 HOME만 사용한 경우 커서는 맨 윗 줄에 옵니다.

HOME U 와같이 줄을 나타내는 첫번째 문자만 사용할 수도 있습니다.

See also

CLS, LOCATE, LCD

Example

```
Lowerline
LCD "Hello"
Home Upper
LCD "Upper"
```

I2CRECEIVE

Action

I2C 직렬 장치로부터 데이터를 수신 합니다.

Syntax

I2CRECEIVE slave, var

I2CRECEIVE slave, DATA ,b2W, b2R

Remarks

slave	바이트, 워드/정수변수나 숫자 값으로 I2C디바이스로부터의 slave 값
Var	바이트, 워드/정수변수이며 I2C디바이스로부터 오는 정보를 수신합니다.
B2W	써넣기 바이트 개수 너무많은수의 바이트를 지정하면 안됩니다.
B2R	읽기 바이트 갯수 너무많은 수의 바이트를 지정하면 안됩니다.

이 명령은 약간의 부가적인 하드웨어에서 동작합니다.

See also

Example

```
x = 0                                'reset variable
slave = &H40                         'slave address of a PCF 8574 I/O IC
I2CRECEIVE slave, x                  'get the value
PRINT x                              'print it
```

```
Dim buf(10) as String
Buf(1) = 1 : buf(2) = 2
I2CRECEIVE slave, buf(), 2, 1 ' send two bytes and receive one byte
```

```
Print buf(1)           ' print the received byte
```

I2CSEND

Action

I2C소자로 데이터를 송신합니다.

Syntax

I2CSEND slave, var

I2CSEND slave, DATA , bytes

Remarks

slave	I2C디바이스의 슬레이브 어드레스
Var	바이트, 정수/워드 또는 숫자이며 I2C디바이스로 송신될 숫자 값
Bytes	송신할 바이트 개수

이 명령은 약간의 부가적인 하드웨어에서 동작합니다. 부록 D참조

See also

Example

```
X = 5
```

```
Dim ax(10) As Byte
```

```
Slave = &H40 ' slave address of a PCF8574 I/O IC
```

```
Bytes = 1 ' Send 1 byte
```

```
I2CSEND slave, x ' send the value of
```

```
For a = 1 to 10
```

```
    Ax(a) = a
```

```
Next
```

```
Bytes = 10
```

```
I2CSEBD slave, ax(), bytes
```

```
END
```

I2CSTART, I2CSTOP, I2CRBYTE, I2CWBYTE

Action

I2CSTART 는 I2C 스타트 코드를 만들어 냅니다.

I2CSTOP은 I2C스톱 코드를 만들어 냅니다.

I2CRBYTE는 I2C로부터 한 개의 바이트를 읽습니다.

I2CWBYTE는 I2C디바이스로 한 개의 바이트를 보냅니다.

Syntax

I2CSTART

I2CSTOP

I2CRBYTE var, 8/9

I2CWBYTE val

Remarks

var	I2C-디바이스로부터 읽은 데이터를 저장할 바이트, 정수/워드 변수 입니다.
8/9	여러 개의 바이트를 읽을 경우 8 또는 ACK 를 지정합니다. 마지막 바이트를 읽은 경우 9 또는 NACK 를 지정합니다.

Val	I2C디바이스로 보낼 바이트, 정수/워드 또는 상수값
-----	-------------------------------

이 명령은 약간의 부가적인 하드웨어에서 동작합니다.

이 기능은 I2CSEND와 I2CRECEIVE기능에 추가되어 제공됩니다.

See also

I2CRECEIVE I2CSEND

Example

```
'----- Writing and reading a byte to an EEPROM 2404 -----
DIM a AS byte
DIM adresW AS CONST 174      'write of 2404
DIM adresR AS CONST 175      'read adres of 2404
I2CSTART                      'generate start
I2CWBYTE  adresW              'send slaveadres
I2CWBYTE  1                   'send adres of EEPROM
I2CWBYTE  3                   'send a value
I2CSTOP                       'generate stop
WaitMS 10                    'wait 10 mS because that is the time
                              'that the chip needs to write the data

'-----now read the value back into the var a -----
I2CSTART                      'generate start
I2CWBYTE  adresW              'write slaveadres
I2CWBYTE  1                   'write adres of EEPROM to read
I2CSTART                      'generate repeated start
I2CWBYTE  adresR              'write slaveadres of EEPROM
I2CRBYTE  a,9                 'receive value into a. 9 means
                              'last byte to receive
I2CSTOP                       'generate stop
PRINT a                      'print received value
END
```

IDLE

Action

프로세서를 아이들 모드로 만듭니다.

Syntax

IDLE

Remarks

아이들모드에선 CPU에 클럭을 공급하지않습니다. 그러나 인터럽트, 시리얼포트, 타이머/카운터는 유효합니다.

아이들모드는 인터럽트를 만나거나 RESET-핀에 의한 시스템 리셋에 의하여 종료됩니다.

See also

Example

IDLE



IF

Action

부울리안 연산결과에 따른 조건부 실행 또는 분기

Syntax

```
IF expression THEN
[ ELSE expression THEN ]
[ ELSE ]
```

END IF

Remarks

expression	논리적인 참, 거짓을 나타낼 수 있는 어떤 식 또는 표현
------------	---------------------------------

한줄의 IF 문장도 가능합니다.

IF expression THEN statement [ELSE statement]

여기서 [ELSE] 문은 옵션입니다.

비트 변수의 IF 사용도 가능합니다.

IF var.bit = 1 THEN

See also

,

Example

```
DIM A AS Integer
```

```
A = 10
```

```
IF A = 10 THEN                                'test expression
    PRINT " This part is executed."           'this will be printed
```

```
ELSE
    PRINT " This will never be executed."      'this not
```

```
END IF
```

```
IF A = 10 THEN PRINT "New in BASCOM"
```

```
IF A = 10 THEN GOTO LABEL1 ELSE PRINT "A<>10"
```

```
LABEL1:
```

REM The Following example shows enhanced use of IF THEN

```
IF A.15 = 1 THEN                                'test for bit
```

```
    PRINT "BIT 15 IS SET"
```

```
END IF
```

REM the following example shows the 1 line use of IF THEN [ELSE]

```
IF A.15 = 0 THEN PRINT "BIT 15 is cleared" ELSE PRINT "BIT 15 is set"
```

INCR

Action

변수의 값을 하나 증가 시킨다.

Syntax

INCR var

Remarks

Var	정수/워드 또는 바이트 변수
-----	-----------------

숫자를 1 증가 시킬 때 사용합니다.

NCR문은 `var = var + 1` 보다 빠릅니다.

See also

Example

```
DO                'start loop
    INCR a        'increment a by 1
    PRINT a       'print a
LOOP UNTIL a > 10 'repeat until a is greater than 10
```

INKEY

Action

시리얼 입력버퍼로부터 첫번째 ASCII 문자를 반환합니다.

Syntax

var = **INKEY**

Remarks

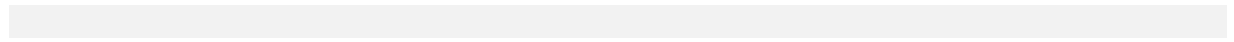
var	Byte, Integer, Word, Long 또는 String 변수
-----	--

아무런 입력이 없으면 0(zero) 을 반환합니다.

INKEY 루틴은 RS232 인터페이스가 컴퓨터에 연결 되어 있어야 사용 가능합니다.

Example

```
DO                'start loop
    A = INKEY      'look for character
    IF A > 0 THEN  'is variable > 0?
        PRINT A    'yes , so print it
    END IF
LOOP              'loop forever
```



INP

Action

하드웨어 포트로부터 값을 읽어 반환합니다.

Syntax

var = **INP**(address)

Remarks

var	바이트 또는 정수/워드 변수
Address	값을 읽을 어드레스

INP 명령은 프로세서의 외부 메모리에서 데이터를 가져 오는 것입니다.

See also

Example

```
Dim a as byte
a = INP(&H8000) 'read value that is placed on databus(d0-d7) at
                'hex address 8000

PRINT a
END
```

INPUTBIN

Action

시리얼 포트로부터 이진수(binary)열을 읽어 들입니다.

Syntax

INPUTBIN , var [, var2]

Remarks

Var1	수신된 데이터를 저장할 변수.
Var2	시리얼 포트로부터 수신된 문자를 저장할 두번째(또는 그이상) 변수

See also

PRINTBIN

Example

```
Dim a as Byte, C as Integer
INPUTBIN a, c 'wait for 3 characters
End
```

INPUTHEX

Action

프로그램 실행 중에 키보드로 부터 문자를 입력합니다.

Syntax

```
INPUTHEX ["prompt"], var [ , varn] [NOECHO ]
```

Remarks

Prompt	프롬프트 문자이전에 프린트 되는 선택적인 스트링 상수
Var, varn	입력값을 저장할 변수
NOECHO	COM-port 에코 기능을 중지 합니다.

INPUTHEX는 마이크로프로세서에 RS-232인터페이스가 있는 경우에 사용할 수 있습니다.

RS-232의 설계는 매뉴얼을 참고 하십시오.

RS-232인터페이스는 컴퓨터와 직렬 통신을 가능하게 합니다..

터미널 에뮬레이션 장치나 키보드를 입력장치로 사용 가능합니다.

Var이 byte 인 경우 입력은 반드시 2 문자 길이어야 합니다.

Var이 정수/워드인 경우 입력은 반드시 4 문자 길이 이어야 합니다.

See also

Example

```
Dim x As Byte
```

```
INPUTHEX "Enter a number ", x          'ask for input
```

INPUT

Action

프로그램 실행중에 키보드로부터 입력합니다.

Syntax

```
INPUT ["prompt"], var [ , varn] [NOECHO ]
```

Remarks

Prompt	프롬프트 문자이전에 프린트 되는 스트링 상수
Var, varn	입력값이나 스트링을 저장할 변수
NOECHO	COM-port 에코 기능을 중지 합니다.

INPUT는 마이크로프로세서에 RS-232인터페이스가 있는 경우에 사용할 수 있습니다.

RS-232의 설계는 매뉴얼을 참고 하십시오.

RS-232인터페이스는 컴퓨터와 직렬 통신을 가능하게 합니다..

터미널 에뮬레이션 장치나 키보드를 입력장치로 사용 가능합니다.

See also

INPUTHEX, PRINT

Example

```

'-----
'          (c) 1997-2001 MCS Electronics
'-----
' file: INPUT.BAS
' demo: INPUT, INPUTHEX
'-----

'To use another baudrate and crystal frequency use the
'metastatements $BAUD = and $CRYSTAL =
$baud = 1200                      'try 1200 baud for example
$crystal = 12000000                '12 MHz

'-----
' When you need that the program times out on waiting for a character
' you need to use the TIMEOUT option.
' When the character is not received within the specified time ERR will be set to 1
' otherwise ERR will be 0.
' IMPORTANT : the TIMEOUT variable will use 4 bytes of internal memory
'-----

Dim V As Byte , B1 As Byte
Dim C As Integer , D As Byte
Dim S As String * 15                'only for uP with XRAM support

Input "Use this to ask a question " , V
Input B1                            'leave out for no question

Input "Enter integer " , C
Print C

Inputhex "Enter hex number (4 bytes) " , C
Print C
Inputhex "Enter hex byte (2 bytes) " , D
Print D

Input "More variables " , C , D
Print C ; " " ; D

Input C Noecho                      'supress echo

Input "Enter your name " , S
Print "Hello " ; S

Input S Noecho                      'without echo
Print S

'unremark next line and remark all lines above for the TIMEOUT option
'this because when you use TIMEOUT once, you need to use it for all INPUT statements
'Input "Name " , S Timeout = 0
'Print Err ; " " ; s
End

```

INSTR

Action

스트링에서 부 스트링과 일치하는 위치 값을 반환합니다.
Returns the position of a sub string in a string.

Syntax

```
var = INSTR( start , string , substr )
var = INSTR( string , substr )
```

Remarks

Var 스트링에서 부 스트링과 일치하는 위치 값을 저장하는 변수입니다. 스트링이 발견되지 않으면 0을 반환합니다. Numeric variable that will be assigned with the position of the sub string in the string. Returns 0 when the sub string is not found.

Start 검색 시작위치 입니다. 지정하지 않으면 스트링 전부를 조사합니다. 지정하지 않으면 처음부터(검색위치 1) 검색을 시작합니다.

String 검사 대상 스트링

Substr 검사 스트링

현재 INSTR() 은 내부 RAM 에 정의된 스트링만 사용가능합니다.

See also

None

Example

```
Dim S As String * 10 , Z As String * 5
Dim Bp As Byte
S = "This is a test"
Z = "is"
Bp = Instr(s , Z) : Print Bp           'should print 3
Bp = Instr(4 , S , Z) : Print Bp      'should print 6
End
```

LCASE

Action

스트링의 대문자를 소문자로 변환합니다.

Syntax

```
dest = LCASE( source )
```

Remarks

dest 소스 스트링을 소문자로 변환하여 저장할 스트링 변수입니다. 소스 스트링은 변경되지 않습니다.

See also

UCASE

Example

```
Dim S As String * 12 , Z As String * 12
Input "Hello " , S           'assign string
S = Lcase(s)                 'convert to lowercase
Print S                      'print string

S = Ucase(s)                 'convert to upper case
Print S                      'print string
```

LCD

Action

LCD디스플레이로 상수나 변수를 보냅니다.

Syntax

LCD x

Remarks

x	디스플레이할 변수나 상수
---	---------------

더많은 변수를 표시할경우 ; 부호를 사용합니다.

LCD a ; b1 ; "constant"

LCD명령은 Print 명령처럼 동작합니다.

See also

LCDHEX, \$LCD CONFIG LCD

Example

```
'-----
'                               (c) 1997-2001 MCS Electronics
'-----
' file: LCD.BAS
' demo: LCD, CLS, LOWERLINE, SHIFTLCD, SHIFTCURSOR, HOME
'       CURSOR, DISPLAY
'-----
$sim
Rem The $sim statement will remove long delays for the simulator
Rem It is important to remove this statement when compiling the final file

'Config Lcdpin = Pin , Db4 = P3.1 , Db5 = P3.2 , Db6 = P3.3 , Db7 = P3.4 ,
E = P3.5 , Rs = P3.6
Rem with the config lcdpin statement you can override the compiler settings

Dim A As Byte
Config Lcd = 16 * 2                               'configure lcd screen
'other options are 16 * 4 and 20 * 4, 20 * 2 , 16 * 1a
'When you dont include this option 16 * 2 is assumed
'16 * 1a is intended for 16 character displays with split addresses over 2
lines
```

```

'$LCD = address will turn LCD into 8-bit databus mode
'      use this with uP with external RAM and/or ROM
'      because it aint need the port pins !

Cls                                'clear the LCD display
Lcd "Hello world."                'display this at the top line
Wait 1
Lowerline                        'select the lower line
Wait 1
Lcd "Shift this."                'display this at the lower
line
Wait 1
For A = 1 To 10
    Shiftlcd Right                'shift the text to the right
    Wait 1                        'wait a moment
Next

For A = 1 To 10
    Shiftlcd Left                 'shift the text to the left
    Wait 1                        'wait a moment
Next

Locate 2 , 1                      'set cursor position
Lcd "*"                          'display this
Wait 1                            'wait a moment

Shiftcursor Right                 'shift the cursor
Lcd "@"                          'display this
Wait 1                            'wait a moment

Home Upper                        'select line 1 and return
home
Lcd "Replaced."                  'replace the text
Wait 1                            'wait a moment

Cursor Off Noblink                'hide cursor
Wait 1                            'wait a moment
Cursor On Blink                   'show cursor
Wait 1                            'wait a moment
Display Off                       'turn display off
Wait 1                            'wait a moment
Display On                        'turn display on

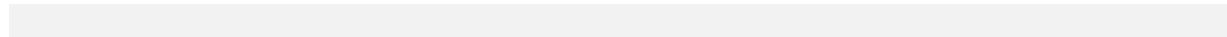
'-----NEW support for 4-line LCD-----
Thirdline
Lcd "Line 3"
Fourthline
Lcd "Line 4"
Home Third                        'goto home on line three
Home Fourth
Home F                            'first letteer also works
Locate 4 , 1 : Lcd "Line 4"
Wait 1

'Now lets build a special character
'the first number is the characternumber (0-7)
'The other numbers are the rowvalues
'Use the LCD tool to insert this line
Deflcdchar 0, 31, 17, 17, 17, 17, 17, 31, 0 ' replace ? with number (0-7)
Deflcdchar 1, 16, 16, 16, 16, 16, 16, 16, 31 ' replace ? with number (0-7)
Cls                                'select data RAM

```

```
Rem it is important that a CLS is following the deflcdchar statements
' because it will set the controller back in datamode
Lcd Chr(0) ; Chr(1)           'print the special character

'----- Now use an internal routine -----
Acc = 1                       'value into ACC
Call Write_lcd                 'put it on LCD
End
```



LCDHEX

Action

LCD-디스플레이로 hexa 변수 값을 보냅니다.

Syntax

LCDHEX var

Remarks

var	표시할 변수
-----	--------

Var1: Bytes, Integer, Word, Long, Single, Constant.

PRINTHEX와 같은 규칙입니다.

See also

Example

```
Dim a as byte
a = 255
LCD a
Lowerline
LCDHEX a
End
```

LCDINIT

Action

LCD 디스플레이를 다시 초기화합니다.

Syntax

LCDINIT

Remarks

사용자 프로그램에서 LCD 디스플레이 루틴을 사용하게 되면 프로세서 기동시에 자동으로 LCD를 초기화 합니다. LCD 루틴은 LCD 의 WR 이 0V 에 연결되어야 합니다. 만약 WR 이 마이크로 프로세서에 PIN에 연결되어 있다면 리셋 과정에서 WR 이 HIGH 상태이므로 정상적으로 초기화 되지 않습니다.

See also

LCDHEX , \$LCD CONFIG LCD

Example

```
'-----
'                                     (c) 1997-2002 MCS Electronics
'-----
'  file: LCD.BAS
'  demo: LCD, CLS, LOWERLINE, SHIFTLCD, SHIFTCURSOR, HOME
'         CURSOR, DISPLAY
'-----
```

```

$sim
Rem The $sim statement will remove long delays for the simulator
Rem It is important to remove this statement when compiling the final file

'Config Lcdpin = Pin , Db4 = P3.1 , Db5 = P3.2 , Db6 = P3.3 , Db7 = P3.4 ,
E = P3.5 , Rs = P3.6
Rem with the config lcdpin statement you can override the compiler settings

Dim A As Byte
Config Lcd = 16 * 2                                'configure lcd screen
'other options are 16 * 4 and 20 * 4, 20 * 2 , 16 * 1a
'When you dont include this option 16 * 2 is assumed
'16 * 1a is intended for 16 character displays with split addresses over 2
lines

'$LCD = address will turn LCD into 8-bit databus mode
'      use this with uP with external RAM and/or ROM
'      because it doesnt need the port pins !

'----- these 2 lines can be used when WR is connected to P1.0
for example ---

P1.0 = 0
INITLCD
'-----
-----
Cls                                'clear the LCD display
Lcd "Hello world."                'display this at the top line
Wait 1
Lowerline                          'select the lower line
Wait 1
Lcd "Shift this."                 'display this at the lower line

```

LEFT()

Action

스트링의 좌측으로부터 지정된 수의 문자를 반환합니다.

Syntax

var = **Left**(var1 , l)

Remarks

var	지정되는 스트링 입니다.
Var1	소우스 스트링 입니다.
N	소우스 스트링 으로부터 얻는 문자수입니다.

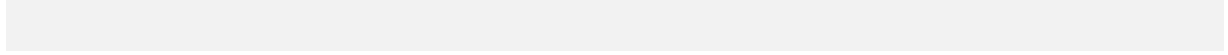
Example

```

Dim s as XRAM String * 15, z as XRAM String * 15
s = "ABCDEFGFG"

```

```
z = Left(s,5)
Print z      'ABCDE
End
```



LEN

Action

스트링의 문자 수를 반환합니다.

Syntax

```
var = LEN(string)
```

Remarks

var	스트링의 길이를 저장할 변수입니다.
String	길이를 계산할 스트링입니다.

Example

```
Dim S As String * 12
Dim A As Byte
s = "test"
A = Len(s)
Print A 'print 4
```

LOAD

Action

오토리로드 모드에서 **TIMER**에 값을 설정합니다.

Syntax

```
LOAD TIMER , value
```

Remarks

TIMER	TIMER0, TIMER1. 또는 TIMER2
Value	변수 또는 값입니다.

TIMER/COUNTER 모드2에서 **ON TIMERx** 문을 사용하면 지정된 시간후 반드시 인터럽트가 일어납니다. 인터럽트가 일어나는 주기를 설정할 수 있습니다.

TIMER0, TIMER1에서 0 부터 255까지의 값이 가능합니다.

TIMER2는 1 부터 65535범위의 값입니다.

LOAD 명령은 다음 계산과 같이 실행됩니다.

The Formula: $TLx = THx = (256 - value)$

For **TIMER2**: $RCAP2L = RCAP2H = (65536 - value)$

이명령은 Timer/Counter에 값을 설정하거나 읽을 목적으로 하지 않습니다. **COUNTERx**명령을 사용하여야 합니다.

Example

```
LOAD TIMER0, 100      ' load TIMER0 with 100
                        ' (256-100)=156 will be stored
                        ' in order to reload after 100 counts.
```

Will generate:

```
Mov tl0, #9Ch
```

```
Mov th0, #9Ch
```



LOCATE

Action

LCD의 커서를 지정된 위치로 이동합니다.

Syntax

LOCATE *y* , *x*

Remarks

<i>x</i>	위치(1-64*)를 나타내는 변수또는 상수입니다.
<i>y</i>	라인(1-4*)을 나타내는 변수 또는 상수 입니다.

* 사용되는 디스플레이에 따라 다릅니다.

See also

CONFIG LCD, LCD, HOME, CLS

Example

```
LCD "Hello"
Locate 1,10
LCD "**"
```

LOOKUP

Action

테이블로부터 바이트 값을 반환합니다.

Syntax

var =LOOKUP(value, label)

Remarks

var	반환되는 바이트 값 입니다.
Value	테이블의 바이트 인덱스 값 입니다.
Label	데이터 시작위치의 라벨입니다.

Var: Byte, Integer, Word, Long, Single

value: Byte, Integer, Word, Long, Constant

See also

LOOKUPSTR

Example

```
DIM b1 As Byte, I As Integer
b1 = Lookup(1, dta)
Print b1           'Prints 2 (zero based)
End
```

```
I = Lookup(0, DTA2)
END
```

DTA:
DATA 1,2,3,4,5

DTA2: 'integer data
1000%, 2000%

LOOKUPSTR

Action

테이블로부터 스트링을 반환합니다.

Syntax

var =LOOKUPSTR(value, label)

Remarks

var	반환되는 스트링입니다.
Value	테이블의 바이트 인덱스 값입니다. 0 이 시작입니다. 따라서 첫번째 값이 됩니다.
Label	데이터 시작위치의 라벨입니다.

Value: Byte, Integer, Word, Long, Constant. Range(0-255)

See also

LOOKUP

Example

```
Dim s as string, idx as Byte
idx = 0 : s = LookupStr(idx,Sdata)
Print s 'will print 'This'
End
```

Sdata:
Data "This" , "is" ,"a test"

LOW

Action

변수의 하위바이트를 가져옵니다.

Syntax

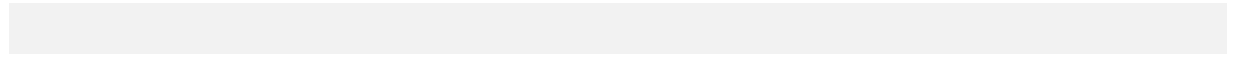
Var = LOW(s)

Var	LSB가 저장될 변수
S	LSB를 가져올 소스변수

See also
HIGH

Example

```
Dim I As Integer, z As Byte
I = &H1001
Z = Low(I) ' is 1
```



LOWERLINE

Action

LCD의 커서를 하위라인으로 이동합니다.

Syntax

LOWERLINE

Remarks

-

See also

UPPERLINE, THIRDLINE, FORTHLINE, HOME

Example

```
LCD "Test"
LOWERLINE
LCD "Hello"
End
```

LOWW

Action

Long 형 변수(32비트)의 하위 워드(16바이트)를 가져옵니다.

Syntax

var = LOWW(s)

Remarks

var 2 개의 바이트가 저장될 수 있는 변수(WORD 또는 INTEGER) 이어야 합니다.
S LONG형 소스변수 입니다.

See also

HIGHW , HIGH , LOW

Example

```
Dim L As Integer , Z As Long
L = &H1001
Z = LowW(L)
```

MAKEBCD()

Action

십진화 바이트, 정수, 워드를 BCD 값으로 변환합니다.

Syntax

var1 = **MAKEBCD**(var2)

Remarks

var1	변환된값이 저장될 바이트, 정수,워드 변수
var2	십진수를 저장하고있는 바이트, 정수, 워드변수나 숫자상수

I2C클록소자로 데이터 입출력 시 사용합니다.

See also

Example

```
Dim a as byte
a = 65
LCD a
Lowerline
LCD BCD(a)
a = MakeBCD(a)
LCD "  " ; a
End
```

MAKEINT

Action

두개의 바이트를 조합하여 워드나 정수로 변환합니다.

Syntax

varn = **MAKEINT**(LSB, MSB)

Remarks

varn	변경된값을 저장할 정수/워드 변수
LSB	LSB에 해당되는 바이트, 숫자 상수 값
MSB	MSB에 해당되는 바이트, 숫자 상수 값

등가 코드는 다음과 같습니다.

$Varn = (256 * MSB) + LSB$

See also

MAKEDEC BCD

Example

```
Dim a As Integer, I as Integer
a = 2
I = MakeInt(a, 1)      ' I = ( 1 * 256) + 2 = 258
End
```

MAKEDEC()

Action

BCD 바이트, 정수, 워드를 DECIMAL 값으로 변경합니다.

Syntax

```
var1 = MAKEDEC(var2)
```

Remarks

var1	변경된 값이 저장될 바이트, 정수, 워드 변수
var2	BCD값이 저장된 바이트, 정수, 워드 변수 또는 숫자상수

I2C 클록 소자로 데이터 입출력 시 사용합니다.

See also

Example

```
Dim a As byte
a = 65
LCD a
Lowerline
LCD BCD(a)
a = MakeDEC(a)
LCD "  " ; a
End
```

MAX()

Action

배열 변수 요소 중 최대 값을 반환합니다.

Syntax

```
var = MAX( ar(1) )
```

Remarks

Var 배열 변수 중 최대 값을 저장할 변수입니다.

Ar() 최대 값을 반환하는 배열 변수이며 지정한 위치 이후부터 조사합니다. 현재 BYTE 변수만 지원하며 다른 데이터 형은 추가될 것입니다.

See also

MIN , AVG

Example

```
Dim ar(10) As Byte
Dim bP as Byte
For bP = 1 to 10
  ar(bP) = bP
Next
bP = Max(ar(1))
Print bP 'should print 10
End
```

MID()

Action

스트링의 중간부분을 리턴합니다.

MID문은 스트링의 일부분을 다른 스트링으로 대체합니다.

Syntax

```
var = MID(var1 ,st [, l] )
```

```
MID(var ,st [, l] ) = var1
```

Remarks

var	목적(destination)스트링입니다.
Var1	소우스 스트링입니다.
St	출발위치 입니다.
L	가져오거나 셋트할 문자수 입니다.

Example

```
Dim s as XRAM String * 15, z as XRAM String * 15
s = "ABCDEFGH"
z = Mid(s,2,3)
Print z                `BCD
z="12345"
Mid(s,2,2) = z
Print s                `A12DEFG
End
```

MIN()

Action

배열 변수 소중 최소값을 반환합니다.

Syntax

```
var = MIN( ar(1) )
```

Remarks

Var 배열 변수 요소중 최소값을 찾아 저장할 변수

ar() 배열 변수이며 찾기 시작하는 위치를 지정합니다. 현재 BYTE 변수만 지원하며 다른 변수는 추가될 예정입니다.

See also

MAX , AVG

Example

```
Dim ar(10) As Byte
Dim bP as Byte
For bP = 1 to 10
    ar(bP) = bP
Next
bP = Min(ar(1))
Print bP 'should print 1
End
```

MOD

Action

나눗셈 결과의 나머지를 반환합니다.

Syntax

```
ret = var1 MOD var2
```

Remarks

ret	나머지가 저장될 변수입니다.
Var1	나누기에서 제수입니다.
Var2	나누기에서 피제수입니다.

Example

```
a = 10 MOD 3      'divide 10 through 3
PRINT a           'print remainder (1)
```

NEXT

Action

FOR..NEXT스트럭처의 끝입니다.

Syntax

```
NEXT [var]
```

Remarks

var	FOR문에서 사용된 인덱스 변수입니다. 사용하지 않아도 됩니다.
-----	-------------------------------------

FOR문은 항상 NEXT문으로 끝나야 합니다.

See also

FOR

Example

```
y = 10                'make y 10
FOR a = 1 TO 10        'do this 10 times
    FOR x = y TO 1     'this one also
        PRINT x ; a    'print the values
    NEXT               'next x (count down)
NEXT a                 'next a (count up) END
```

ON INTERRUPT

Action

인터럽트가 발생하면 서브루틴을 실행합니다.

Syntax

ON interrupt label [NOSAVE]

Remarks

interrupt	INT0, INT1, SERIAL, TIMER0 ,TIMER1 or TIMER2.
Label	인터럽트가 발생하면 라벨위치로 점프합니다.
NOSAVE	NOSAVE를 지정하면 인터럽트 루틴 시작과 종료시 어떤 레지스터도 저장하거나 복구하지 않습니다. 그러므로 이 옵션을 사용한다면 직접 사용한 레지스터를 저장하고 리스토어 하여야 합니다.

인터럽트 루틴에서 **RETURN** 문으로 복귀합니다.

인터럽트 서비스 루틴에는 하나의 **RETURN**문이 있어야 합니다.

TIMER1은 **BAUDRATE**발생기로 사용하고 있으므로 **PRINT** 문과 같은 **SERIAL**에 관련된 명령어를 사용할 경우 **TIMER1**을 지정해선 안됩니다.

INT0와 **INT1**의 인터럽트 트리거 조건을 반드시 지정 하여야 합니다.

TCON레지스터를 조작합니다.

SET TCON.0 : **INT0** 가 H에서 L로 변경되는 순간 인터럽트를 시작합니다.(에지 트리거)

RESET TCON.0 : **INT0** 가 L 레벨인 동안에는 계속해서 인터럽트가 실행됩니다.

SET TCON.2 : **INT1**이 H에서 L로 변경되는 순간 인터럽트를 시작합니다.(에지 트리거)

RESET TCON.2 : **INT1**이 L 레벨인 동안에는 계속해서 인터럽트가 실행됩니다.

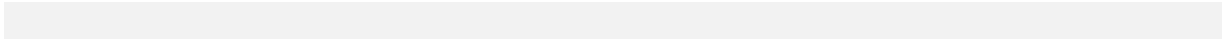
자세한 것은 8051 하드웨어 설명서를 참고 하십시오.

Example

```

ENABLE INTERRUPTS
ENABLE INT0           'enable the interrupt
ON INT0 Label2 nosave 'jump to label2 on INT0
DO                    'endless loop
LOOP
END

Label2:
    PRINT "An hardware interrupt occurred!"    'print message
RETURN
  
```



ON VALUE

Action

변수의 값에 따라 여러 개의 라벨로 점프합니다.

Syntax

ON var [**GOTO**] [**GOSUB**] label1, label2

Remarks

Var	테스트할 변수입니다.(바이트, 정수,워드) P1과 같은 SFR도가능 합니다.
Label1, label2	Var값에 따라 점프할 라벨입니다.

첫번째 라벨의 해당 값은 0 입니다.

Example

```
x = 2                                'assign a variable interrupt
ON x GOSUB lbl1, lbl2, lbl3          'jump to label 'lbl3'
END
lbl3:
    PRINT "lbl3"
RETURN
```

OPEN - CLOSE

Action

디바이스를 오픈, 클로즈합니다.

Syntax

OPEN "device" for **MODE** As #channel
CLOSE #channel

Remarks

device	COM1과 COM2의 2 개의 하드웨어 디바이스가 지원됩니다.: 소프트웨어 UART에서 포트핀과 바우드레이트를 지정하여야 합니다. COM3.0:9600 은 PORT3.0 을 9600 속도로 사용합니다.
MODE	BINARY, INPUT 또는 OUTPUT을 COM1 과 COM2로 지정합니다. 소프트웨어 UART에서 INPUT또는 OUTPUT을 지정하여야 합니다.
Channel	오픈 하는 채널 수 이며 반드시 양 정수 이어야 합니다.

80537프로세서는 2 개의 시리얼 채널을 가지고 있습니다. 컴파일러에게 어떤 포트를 사용할것인지 알려주어야 합니다. 이것이 OPEN 문을 사용하는 이유입니다. 1 개의 시리얼 포트만 가진 프로세서는 관계없습니다.

모든 OPEN 된 디바이스는 CLOSE #channel 명령으로 크로스 되어야 합니다. 채널 번호는 오픈에서 사용한 번호와 같아야 합니다.

소프트웨어 UART는 데이터를 가져오거나 보내는 명령으로 GET과 PUT명령만을 사용할 수 있습니다. COM1: 과 COM2: 는 하드웨어 UART이며 print 명령등을 사용할 수 있습니다.

Example 1

```
'only works with a 80517 or 80537
CONFIG BAUD1 = 9600          'serial 1 baudrate
OPEN "COM2:" FOR BINARY AS #1  'open the port
PRINT #1, "Hello"            'print to serial 1
PRINT "Hello"                'print to serial 0
CLOSE #1                     'close the channel
```

Example 2

```
'works with every port pin
Dim A As Byte , S As String * 16 , I As Byte , Dum As Byte

'a software comport is named after the pin you use
'for example P3.0 will be "COM3.0:" (so there is no P)
'for software comports, you must provide the baudrate
'So for 9600 baud, the devicename is "COM3.0:9600"
'When you want to use the pin for sending, you must open
'the device for OUTPUT
'When you want to use the pin for receiving, you must open
' the device for INPUT
'At this time only variables can be send and received with
' the PUT and GET statements.
'In the feature PRINT etc. will support these software comports.

Open "com3.1:9600" For Output As #1      'p3.1 is normally
                                         'used for tx so testing is easy
Open "com3.0:9600" For Input As #2      'p3.0 is normally used for
                                         'RX so testing is easy

S = "test this"                         'assign string
Dum = Len(s)                           'get length of string
For I = 1 To Dum                        'for all characters from left to right
A = Mid(s , I , 1)                     'get character
Put #1 , A                             'write it to comport
Next

Do
Get #2 , A                             'get character from comport
Put #1 , A                             'write it back
Print A                                'use normal channel
Loop

Close #1                               ' finally close
device
Close #2
End
```

OUT

Action

바이트 값을 하드웨어 포트로 보냅니다.

Syntax

OUT address, value

Remarks

address	어드레스값
value	바이트, 정수, 워드값

OUT 명령은 외부 램에 데이터를 저장하는 것입니다.

See also

Example

```
Dim a as byte
OUT &H8000,1      'send 1 to the databus(d0-d7) at hex address 8000
END
```

Will generate
 Mov A, #1
 Mov dptr, #8000h
 Movx @dptr, a

P1, P3 (P0, P2)

Action

P1 과 P3는 변수로 취급되는 특수기능 레지스터(SFR)입니다.

Syntax

Px = var
 var = **Px**

Remarks

x	포트의 번호입니다(1 또는 3). P3.6은 AT89C2051에서 사용할 수 없습니다!
Var	가져오거나 설정할 값

다른 프로세서에서 P0와 P2를 지정할 수 있습니다.

자세한 것은 하드웨어자료를 참고 하십시오.

Example

```
Dim a as BYTE, b1 as BIT
a = P1      'get value from port 1
a = a OR 2  'manipulate it
P1 = a      'set port 1 with new value
P1 = &B10010101 'use binary notation
P1 = &HAF    'use hex notation
b1 = P1.1    'read pin 1.1
P1.1 = 0     'set it to 0
```

PEEK()

Action

내부 메모리의 값을 반환합니다.

Syntax

var = **PEEK**(**address**)

Remarks

var	메모리의 지정된 어드레스위치의 값을 저장할 변수입니다.
Address	0부터 255까지의 어드레스값이며 변수 또는 상수입니다.

See also

POKE , CPEEK, INP, OUT

Example

```
DIM a As Byte
a = Peek( 0 )      'return the first byte of the internal memory (r0)
End
```

POKE

Action

바이트 값을 메모리에 저장합니다.

Syntax

POKE **address** , **value**

Remarks

address	메모리의 위치(0-255)를 나타내는 어드레스 변수값 메모리의 위치를 나타내는 숫자값입니다.
Value	지정값입니다.(0-255)

POKE 명령문 사용 시 주의 하십시오. 잘못된 사용으로 메모리가 변경될 수도 있습니다.

See also

PEEK

Example

```
POKE 127, 1          'write 1 to address 127
End
```

POWERDOWN

Action

프로세서가 절전모드로 들어가도록 합니다.

Syntax

POWERDOWN

Remarks

정전모드는 시스템 내부의 클럭을 정지합니다.
리셋트에 의하여 절전모드가 해제됩니다.

See also

Example

POWERDOWN



PRINT

Action

RS-232포트에 데이터를 출력합니다.

Syntax

PRINT var ; "constant"

Remarks

var 프린트할 변수나 상수값입니다.

한 개의 라인에서 여러 개를 프린트할 때 (;) 을 사용합니다.

세미콜론을 사용하면 라인피드가 출력되지 않습니다.

PRINT문은 마이크로프로세서에 **RS-232**인터페이스가 연결되어있어야 동작됩니다.

RS-232인터페이스를 설계하는 것은 매뉴얼을 참고 하십시오.

RS-232인터페이스를 컴퓨터의 시리얼 포트에 연결합니다.

이렇게하면 출력장치를 터미널 에뮬레이터로 사용할 수 있습니다.

내장된 터미널 에뮬레이터를 사용할 수 있습니다.

See also

, INPUT, OPEN, CLOSE

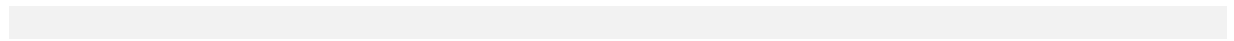
Example

```
'-----
' (c) 1997,1998 MCS Electronics
'-----
' file: PRINT.BAS
' demo: PRINT, PRINTHEX
'-----

Dim A As Byte , B1 As Byte , C As Integer
A = 1
Print "print variable a " ; A
Print                                     'new line
Print "Text to print."                  'constant to print

B1 = 10
Printhex B1                            'print in hexa notation
C = &HA000                              'assign value to c%
Printhex C                             'print in hex notation
Print C                                 'print in decimal notation

C = -32000
Print C
Printhex C
Rem Note That Integers Range From -32767 To 32768
End
```



PRINTBIN

Action

RS-232 포트로 바이너리값을 출력합니다.

Syntax

PRINTBIN var [; varn]

Remarks

var	시리얼 포트로 전송할 변수값입니다.
Varn	추가로 보낼 값입니다.

PRINTBIN은 PRINT CHR(var)과 동일합니다. 그러나 PRINTBIN 은 변수의 전부를 전송할수 있습니다. 예를들어 Long 형 변수를 전송하면 4 개의 바이트가 전송됩니다.

See also

Example

```
Dim a(10) as Byte, c As byte
For c = 1 to 10
  a(c) = a          'fill array
Next
PRINTBIN a(1)      'print content
```

PRINTHEX

Action

시리얼포트로 HEX 값을 출력합니다.

Syntax

PRINTHEX var

Remarks

var	프린트할 변수입니다.
-----	-------------

PRINTHEX는 PRINT문과 동일한 규정이 정해집니다.

PRINT문은 마이크로프로세서에 RS-232인터페이스가 연결되어있어야 동작됩니다.

RS-232인터페이스를 설계하는 것은 매뉴얼을 참고 하십시오.

RS-232인터페이스를 컴퓨터의 시리얼 포트에 연결합니다.

이렇게하면 출력장치를 터미널 에뮬레이터로 사용할 수 있습니다.

내장된 터미널 에뮬레이터를 사용할 수 있습니다.

See also

, INPUTHEX

Example

```
DIM x As Byte
INPUT x          'ask for var
PRINT x          'print it in decimal format
PRINTHEX "Hex " ;x          'print it in hex format
```

PRIORITY

Action

인터럽트의 우선순위를 설정합니다.

Syntax

PRIORITY [SET] [RESET] interrupt

Remarks

SET	인터럽트를 상위레벨로 설정합니다.
RESET	인터럽트를 하위레벨로 설정합니다.
Interrupt	인터럽트를 셋트 하거나 리셋트 합니다.

인터럽트는 **INT0, INT1, SERIAL, TIMER0, TIMER1 and TIMER2** 입니다

INT0는 언제나 최상위 인터럽트 레벨입니다.

동시에 여러 곳의 인터럽트가 발생하면 아래와 같은 순서대로 처리됩니다.

Interrupt	Priority
INT0	1 (highest)
TIMER0	2
INT1	3
TIMER1	4
SERIAL	5 (lowest)

Example

```
PRIORITY SET SERIAL      'serial int highest level
ENABLE SERIAL            'enable serial int
ENABLE TIMER0            'enable timer0 int
ENABLE INTERRUPTS        'activate interrupt handler
ON SERIAL label          'branch to label if serial int occur
DO ..                    'loop for ever
LOOP
```

```
Label:                   'start label
    PRINT "Serial int occurred." 'print message
RETURN                   'return from interrupt
```

READ

Action

변수에 저장된 값을 읽습니다.

Syntax

READ var

Remarks

var	데이터 값이 저장될 변수 입니다.
-----	--------------------

See also

DATA, RESTORE

Example

```

Dim A As Byte, I As Byte, C As Integer, S As XRAM String * 10
RESTORE dta
FOR a = 1 TO 3
    READ i : PRINT i
NEXT
RESTORE DTA2
READ C : PRINT C
READ C : PRINT C
Restore dta3 : Read s : Print s
END

```

dta:

Data 5,10,15

dta2:

Data 1000%, -2000%

dta3:

Data "hello"

READMAGCARD

Action

마그네틱 카드리더로부터 데이터를 읽습니다.

Syntax

READMAGCARD var , bytes , code, timeout

Remarks

Var 마그네틱 카드리더로 부터 읽은 데이터를 저장하기에 충분한 바이트 어레이

bytes 카드로부터 읽은 바이트수The number of bytes read from the card.

Shifts 사용할 코딩번호이며 5 또는 7 입니다. V2.03 은 5 를 지원합니다.

Timeout LONG형 변수 또는 상수이며 TIMEOUT시간 이내에 카드가 감지되지 않으면 Err을 반환합니다.

마그네틱 카드에는 3 개의 트랙이 있습니다.

트랙 1 은 패리티비트 7 번비트를 포함한 데이터가 저장되어 있습니다. 이것은 알파벳과 숫자를 저장하기에 편합니다.

트랙 2 와 트랙 3은 5비트로 코딩된 데이터가 저장됩니다.

ReadMagCard 루틴은 ISO7811-2 규격이며 5 비트와 7 비트 디코딩입니다.

5 비트 코딩의 반환되는 숫자는 아래와 같습니다:

Returned number	ISO characterT
0	0
1	1
2	2
3	3
4	4
5	5
6	6

7	7
8	8
9	9
10	hardware control
11	start byte
12	hardware control
13	separator
14	hardware control
15	stop byte

See also

None

Calls

_Read_Magcard_Code5

Example

```
'[DIM used variables]
Dim X(40) As Byte , I As Byte , Bts As Byte

'[ALIAS the pins used]
_mcs Alias P1.1
_mclock Alias P1.2
_mdata Alias P1.0

Do
    Print "Slide magcard through reader"
    Readmagcard X(1) , Bts , 5, 10000           'call routine
    '          ^ may be 5 or 7. 7 bit coding not implemented yet
    Print "Error " ; Err                        '1 if error occurred
    Print ; " " ; Bts ; " bytes read"           'show number of bytes read

    Print Err
    For I = 1 To Bts
        Print X(i) ; " ";                      'show number
    Next
    Print
Loop
End
```

REM

Action

컴파일러에게 주석문임을 알립니다.

Syntax

REM or ‘

Remarks

프로그램을 분명하게 하기 위하여 주석 문을 사용합니다.

REM 이후에는 BASCOM 의 명령어도 주석문으로 처리되어 프로그램의 아무런 영향을 주지 않습니다.

새로운 주석문 기능이 있습니다.

```
'( start block comment
  print "This will not be compiled"
)' end block command
```

Example

```
REM TEST.BAS version 1.00
PRINT a'      "this is comment      : PRINT "hello"
              ^--- this will not be executed!
```

REPLACE

Action

스트링에서 발견된 문자를 새 문자로 대체합니다.

Syntax

REPLACE string , old , new

Remarks

string The source string to change.

Old A string constant or byte that specifies the character to replace.

New The new character. Also a string constant or a byte.

Example

```
Dim S as String * 12
```

```
s = "Hello"
```

```
REPLACE s , "e" , "a" ' now we got some dutch :-)
```

```
Print s ' should print Hallo
```

RESET

Action

포트(P1.x, P3.x) 또는 비트/바이트 변수를 리셋트 합니다.

Syntax

RESET bit

RESET var.x

Remarks

bit	P1.x , P3.x 또는 0-7 이내의 값인 비트변수
byte	바이트변수
x	리셋할 비트변수 Byte일때 (0-7) 이며 Integer/Word 일때 (0-15)

See also

Example

```
Dim b1 as bit, b2 as byte, I As Integer
RESET P1.3 'reset bit 3 of port 1
RESET b1 'bitvariable
RESET b2.0 'rest bit 0 of bytevariable b2
RESET I.15
```



RESTORE

Action

DATA문에서 READ를 재 사용할 수 있도록 합니다.

Syntax

RESTORE label

Remarks

label	데이터문의 라벨입니다.
-------	--------------

See also

Example

```
DIM a AS BYTE, I AS BYTE
RESTORE dta
FOR a = 1 TO 3
    READ a : PRINT a
NEXT
RESTORE DTA2
READ I : PRINT I
READ I : PRINT I
```

END

DTA1:

Data 5, 10, 100

DTA2:

Data -1%, 1000%

'Integers must end with the %-sign. (Integer : <0 or >255)

RETURN

Action

서브루틴으로 복귀합니다.

Syntax

RETURN

Remarks

서브루틴은 반드시 RETURN문으로 끝나야 합니다.

인터럽트 서브루틴문도 반드시 RETURN문으로 끝나야 합니다.

See also

Example

```
GOSUB Pr          'jump to subroutine
PRINT result      'print result
END               'program ends
```

Pr: 'start subroutine with label

```

    result = 5 * y      'do something stupid
    result = result + 100 'add something to it
RETURN                'return

```

RIGHT()

Action

스트링에서 오른쪽의 지정된 문자를 반환합니다.

Syntax

var = **RIGHT**(var1 ,st [, l])

Remarks

var	목적(destination)스트링입니다.
Var1	소우스 스트링 입니다.
L	가져올 문자 수 입니다.

Example

```

Dim s as XRAM String * 15, z as XRAM String * 15
s = "ABCDEFGH"
z = Right(s,2)
Print z                ' FG
End

```

RND()

Action

난수를 반환합니다.

Returns a random number.

Syntax

var = RND(limit)

Remarks

Limit 반환되는 난수의 최대값을 지정합니다. RND() 함수는 랜덤발생 씨앗 값으로 2 바이트 변수를 사용합니다.

See also

None

Example

```

'-----
'                (c) 2000 MCS Electronics
'                RND.BAS
'-----

Dim W As Word

Do
    'get a random number and limit it to be maximum 100
    W = Rnd(100)

```

Print W
Loop
End



ROTATE

Action

모든 비트를 왼쪽 또는 오른쪽으로 회전합니다.

Syntax

ROTATE *var* , **LEFT/RIGHT** [, *shifts*]

Remarks

var	바이트, 정수, 워드 또는 LONG 형 변수입니다.
Shifts	쉬프트할 수

쉬프트 방향에 따라서 LSB 또는 MSB 가 캐리플래그로 저장됩니다. 이것은 RLC와 RRC 와 같습니다. 이런상황을 원치 않으면 CLC C 명령을 사용하여 캐리를 클리어 합니다.

See also

SHIFTIN, SHIFTOUT

Example

```
Dim a as byte
a = 128
ROTATE a, LEFT, 2
Print a
```

Generated code:

```
Mov R7,#2
Mov R0,#21h
Mov a,@r0
RLC a
Djnz r7, *-1
Mov @r0, a
```

SELECT

Action

표현된 수식 값에 따라 여러 개의 실행 문 블록을 실행합니다.

Syntax

```
SELECT CASE var
  CASE test1 : statements
  CASE test2 : statements
  CASE ELSE : statements
END SELECT
```

Remarks

Var	숫자나 스트링 변수입니다.
Test1	비교 값 입니다.
Test2	비교 값 입니다.

See also

-

Example

```
Dim b2 as byte
SELECT CASE b2      'set bit 1 of port 1
  CASE 2 : PRINT "2"
  CASE 4 : PRINT "4"
  CASE IS >5 : PRINT ">5"  'a test requires the IS keyword
  CASE ELSE
END SELECT
END
```

SET

Action

포트(P1.x P3.x) 또는 비트/바이트 변수를 셋트합니다.

Syntax

SET bit
SET byte.x

Remarks

bit	P1.x, P3.x 또는 비트 변수입니다.
Byte	바이트 변수입니다.
x	셋트할 비트변수입니다.

See also

Example

```
Dim b1 as Bit, b2 as byte, c as Word, L as Long
SET P1.1      'set bit 1 of port 1
SET b1        'bit variable
SET b2.1      'set bit 1 of var b2
SETB b2.1     'use alternative syntax
SET C.15      'set highest bit of Word
SET L.31      'set MS Bit of Long
```

SHIFTCURSOR

Action

LCD 에서 커서를 오른쪽/왼쪽으로 이동합니다.

Syntax

SHIFTCURSOR LEFT / RIGHT

Remarks

-

See also

SHIFTLCD

Example

```
LCD "Hello"
```

```
SHIFTCURSOR LEFT
```

```
End
```



SHIFTIN SHIFTOUT

Action

비트 스트림을 입력하거나 출력합니다.

Syntax

SHIFTIN pin, pclock, var, option

SHIFTOUT pin, pclock, var, option

Remarks

Pin	입력/출력으로 사용할 포트핀
pclock	클럭을 발생할 포트핀
Var	지정할 변수
Option	선택가능한 옵션 0-클럭이 Low로 갈때 MSB부터 쉬프트 In/Out 1-클럭이 High로 갈때 MSB부터 쉬프트 In/Out 2-클럭이 Low로 갈때 LSB부터 쉬프트 In/Out 3-클럭이 High로 갈때 LSB부터 쉬프트 In/Out SHIFTIN명령은 4 를 지정할 수 있으며 이때에는 쉬프트 클럭으로 외부 펄스를 사용합니다.

사용하는 변수에따라서 쉬프트 비트 수가 정해집니다. Byte변수는 8 비트, Integer변수는 16 비트를 쉬프트합니다.

See Also

Example

```
Dim a as byte
SHIFTIN P1.0 , P1.1 , a , 0
SHIFTOUT P1.2 , P1.1 , a , 0
```

For the SHIFTIN example the following code is generated:

```
Setb P1.1
Mov R0,#h'21
Mov r2,#h'01
__UNQLBL1:
Mov r3,#8
__UNQLBL2:
Clr P1.1
Nop
Nop
Mov c,P1.0
Rlc a
Setb P1.1
Nop
Nop
Djnz r3,__UNQLBL2
Mov @r0,a
Dec r0
Djnz r2,__UNQLBL1
```

**Of course it depends on the parameter, which code will be generated.
To shift with an external clock signal:**

```
SHIFTIN P1.0, P1.1 , a , 4 'add 4 for external clock
Generated code:
```

```

Mov R0,#h'21
Mov r2,#h'01
__UNQLBL1:
Mov r3,#8
__UNQLBL2:
Jnb P1.1,*+0
Mov c,P1.0
Rlc a
Jb P1.1,*+0
Djnz r3,__UNQLBL2
Mov @r0,a
Dec r0
Djnz r2,__UNQLBL1

```

SHIFTLCD

Action

LCD디스플레이를 오른쪽/왼쪽으로 이동합니다.

Syntax

SHIFTLCD LEFT / RIGHT

Remarks

-

See also

SHIFT CURSOR

Example

```

LCD "Very long text"
SHIFTLCD LEFT
Wait 1
SHIFTLCD RIGHT
End

```

SHOWPIC

Action

그래픽 디스플레이 장치에 BGF 파일을 표시합니다.

Syntax

SHOWPIC x, y , label

Remarks

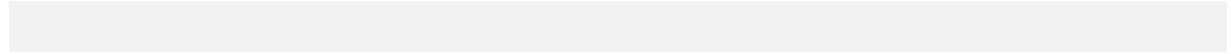
SHOWPIC 명령은 변환된 BMP 파일을 표시합니다. BMP 파일은 반드시 그래픽 컨버터에 의하여 BGF 파일로 변환되어 있어야 합니다. X, Y 라메터는 반드시 0 이거나 8 의 배수이어야 합니다. 그림의 가로와 세로도 반드시 8 의 배수이어야 합니다. 라벨은 컴파일러에게 소스 프로그램 중에 그래픽 데이터 파일 위치를 알려줍니다.

See also

Example

```
CLS GRAPH                'clear graphic part of display
ShowPic 0,0, label        'show picture
End
```

```
Label:
$BGF "mypic.bgf"          'data will be inserted here
```



SOUND

Action

포트핀으로 펄스를 내보냅니다.

Syntax

SOUND pin, duration, frequency

Remarks

pin	P1.0과 같은 모든 I/O 핀번호
duration	보내지는 펄스 갯수이며 바이트, 정수, 워드 또는 상수값 (1 – 32768)
frequency	로우와 하이를 반복하는 시간

포트핀에 버저 또는 스피커를 부착하면 **SOUND** 의하여 소리를 낼 수 있습니다.

See also

-

Example

```
SOUND P1.1 , 10000, 10           'BEEP
End
```

SPACE ()

Action

공백문자로된 스트링을 반환합니다.

Syntax

var = **SPACE**(x)

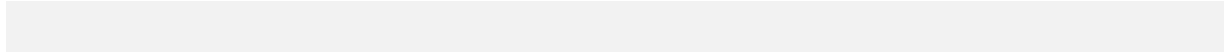
Remarks

x	스페이스의 개수입니다.
Var	지정되는 스트링입니다.

Example

```
Dim s as XRAM String * 15, z as XRAM String * 15
s = Space(5)
Print " { " ; s ; " } "           ' {      }
Dim A as Byte
A = 3
S = Space(a)
Generated code for last 2 lines :
; ----- library routine -----
_sStr_String:
Mov @r1,a
Inc r1
Djnz r2,_sStr_String
Clr a
Mov @r1,a
Ret
;-----
Mov R1,#h'22 ; location of string
Mov R2,h'21 ; number of spaces
```

```
Mov a,#32  
Acall _sStr_String
```



SPC()

Action

시리얼 포트 또는 LCD 로 공백문자를 출력합니다.

Syntax

PRINT SPC(x)

Remarks

x 프린트할 공백문자 수입니다. 범위는 1 - 255입니다. SPACE() 함수는 스트링을 대상으로 한 것입니다. SPC() 는 PRINT , LCD 와 함께 사용합니다.

Example

```
Dim s as XRAM String * 15, z as XRAM String * 15
s = "Hello"
Print " {" ;s ; SPC(3) ; "}"
```

SPIIN

Action

SPI 버스로부터 값을 읽습니다.

Syntax

SPIIN var, bytes

Remarks

var	SPI버스로부터 읽은 값을 저장할 변수
bytes	읽을 바이트수

See also

SPIOUT

Example

```
Dim a(10) as byte
CONFIG SPI = SOFT, DIN = P1.0, DOUT = P1.1, CS=P1.2, CLK = P1.3
SPIIN a(1) , 4 'read 4 bytes
```

SPIOUT

Action

SPI버스로 데이터를 출력합니다.

Syntax

SPIOUT data , bytes

Remarks

data	출력할 데이터의 변수입니다.
Bytes	보내는 바이트수입니다.

See also
SPIIN, CONFIG SPI

Example

```
CONFIG SPI = SOFT, DIN = P1.0, DOUT = P1.1, CS=P1.2, CLK = P1.3
Dim a(10) as Byte , X As Byte
SPIOUT a(1) , 5 'send 5 bytes
SPIOUT X , 1 'send 1 byte
```

START

Action

지정한 타이머/카운터를 시작합니다.

Syntax

START timer

Remarks

<i>timer</i>	TIMER0, TIMER1, COUNTER0 또는 COUNTER1.
--------------	---------------------------------------

인터럽트가 발생 가능한 상태에서 **timer/counter**를 시작 하여야 합니다.

TIMER0 와 COUNTER0 는 동일한 명령어 입니다.

See also

STOP TIMERx

Example

```
ON TIMER0 label2
LOAD TIMER0, 100
START TIMER0
DO 'start loop
LOOP 'loop forever
label2: 'perform an action here
RETURN
```

STOP

Action

프로그램의 실행을 중지합니다.

Syntax

STOP

Remarks

END문도 프로그램의 실행을 종료합니다.

END 나 STOP문을 만나면 모든 인터럽트는 금지상태로 되며 무한 루프상태가 됩니다.

Example

```
PRINT var      'print something  
STOP           'thats it
```

STOP TIMERX

Action

지정된 타이머/카운터의 실행을 중지합니다.

Syntax

STOP timer

Remarks

<i>timer</i>	TIMER0, TIMER1, TIMER2, COUNTER0 또는 COUNTER1.
--------------	---

인터럽트발생을 원치 않을 때에 타이머를 스톱합니다.

TIMER0와 COUNTER0는 동일한 명령어 입니다.

See also

Example

```

'-----
' (c) 1997,1998 MCS Electronics
'-----
' file: TIMER0.BAS
' demo: ON TIMER0
' *TIMER1 is used for RS-232 baudrate generator
'-----
Dim Count As Byte , Gt As Byte
Config Timer0 = Timer , Gate = Internal , Mode = 2
'Timer0 = counter : timer0 operates as a counter
'Gate = Internal : no external gate control
'Mode = 2 : 8-bit auto reload (default)
On Timer0 Timer_0_int
Load Timer0 , 100                'when the timer reaches 100 an interrupt
                                'will occur
Enable Interrupts                'enable the use of interrupts
Enable Timer0                    'enable the timer
Rem Setting Of Priority
Priority Set Timer0               'highest priority
Start Timer0                     'start the timer
Count = 0                       'reset counter
Do
Input "Number " , Gt
Print "You entered : " ; Gt
Loop Until Gt = 1                'loop until users presses ESC key
Stop Timer0
End
Rem The Interrupt Handler For The Timer0 Interrupt
Timer_0_int:
Inc Count
If Count = 2 Then
Print "Timer0 Interrupt occured"
Count = 0
End If
Return

```



STR()

Action

숫자값을 해당되는 스트링으로 변환합니다.

Syntax

var = Str(x)

Remarks

var	스트링 변수
X	바이트, 정수, 워드와 같은 숫자 변수입니다.

See also

Example

```
Dim a as Byte, S as XRAM String * 10
a = 123
s = Str(a)
Print s
End
```

STRING()

Action

일정수의 반복되는 문자열을 반환합니다.

Syntax

var = STRING(char ,x)

Remarks

var	목적(destinaation) 스트링 입니다.
N	스트링에 사용되는 문자입니다.
M	문자의 개수입니다.

스트링의 종료를 표시하는 코드는 0 입니다. 따라서 n 에 0 을 지정할 수 없습니다.

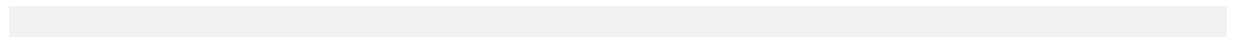
M에 0 을 지정하면 스트링의 길이는 255바이트가 됩니다.

See also

SPACE

Example

```
Dim s as XRAM String * 15, z as XRAM String * 15
s = String(65,5)
Print s                      'AAAAA
End
```



SUB

Action

서브루틴을 선언합니다.

Syntax

SUB Name[(var1)]

Remarks

Name	예약어를 제외한 이름입니다.
Var1	파라미터의 이름입니다.

서브루틴의 종료 시 **END SUB**를 사용하여야 합니다.

SUB문을 사용하기 전에 **SUB**프로시저를 선언 하여야 합니다.
파라미터이름과 형식은 **SUB**프로시저 선언시와 동일 하여야 합니다.

파라미터는 프로그램 전체에 적용되는 글로벌형 변수입니다.
사용되었던 파라미터는 **DIM**문에 의하여 정의 되어야 합니다.
따라서 변수는 프로그램과 서브프로그램에서 사용됩니다.
다음예에서 사용법을 설명합니다.

```
Dim a as byte, b1 as byte, c as byte 'dim used variables
Declare Sub Test(a as byte)         'declare subroutine
a = 1 : b1 = 2: c = 3                'assign variables

Print a ; b1 ; c                     'print them

Call Test(b1)                        'call subroutine
Print a ;b1 ; c                      'print variables again
End

Sub Test(a as byte)                  'begin procedure/subroutine
  print a ; b1 ; c                   'print variables
End Sub
```

See also

Example

-

SWAP

Action

동일한 타입의 2개의 변수내용을 교환합니다.

Syntax

SWAP var1, var2

Remarks

var1	Bit, byte, integer 또는 word 변수
var2	var1과 동일한 형의 변수

swap문이 실행된 후 var1은 var2값을 그리고 var2는 var1값을 가지게 됩니다.

Example

```
Dim a as integer,b1 as integer
a = 1 : b1 = 2      'assign two integers
SWAP a, b1          'swap them
PRINT a ; b1
```

THIRDLINE

Action

LCD커서를 3 라인으로 이동합니다.

Syntax

THIRDLINE

Remarks

-

See also

UPPERLINE, LOWERLINE, FORTHLINE

Example

```
Dim a as byte
a = 255
LCD a
Thirdline
LCD a
Upperline
End
```

UCASE

Action

스트링의 소문자를 대문자로 변환합니다.

Syntax

dest = UCASE(source)

Remarks

dest 소스 스트링의 소문자를 대문자로 변환한 후 저장할 스트링 변수 입니다. 소스 스트링은 변경되지 않습니다.

See also

LCASE

Example

```

Dim S As String * 12 , Z As String * 12
Input "Hello " , S           'assign string
S = Lcase(s)                 'convert to lowercase
Print S                      'print string

S = Ucase(s)                 'convert to upper case
Print S                      'print string

```

UPPERLINE

Action

LCD커서를 한줄 위로 올립니다.

Syntax

UPPERLINE

Remarks

-

See also

LOWERLINE, THIRDLINE, FORTHLINE

Example

```

Dim a as byte
a = 255
LCD a
Lowerline
LCD a
Upperline
End

```

VAL()

Action

스트링으로 표현된 숫자를 숫자로 변환합니다.

Syntax

var = Val(s)

Remarks

var	바이트, 정수 또는 워드 숫자 값
s	스트링 타입의 변수

var: Byte, Integer, Word, Long, Single

See also

Example

```

Dim a as byte, s As XRAM string * 10
s = "123"

```

```
a = Val(s)      'convert string
Print a
End
```



VARPTR()

Action

변수의 메모리-어드레스값을 가져옵니다.

Syntax

```
var = VARPTR(var2)
```

Remarks

var	var2의 어드레스를 지정할 변수
var2	어드레스를 가져올 변수

See also

PEEK, POKE

Example

```
Dim I As Integer, B1 As Byte
B1 = Varptr(I)
```

Generated code:

```
Mov h'23, #h'21
```

WAIT

Action

주어진 시간동안 프로그램의 실행을 멈춥니다.

Syntax

```
WAIT seconds
```

Remarks

seconds	대기할 시간수(초단위)
---------	--------------

딜레이 타임은 12MHz 클럭 주파수를 기준한 것입니다.
이 명령어로는 정확한 타이밍이 얻어지지 않습니다.
인터럽트를 사용할 경우 딜레이를 확장할 수 있습니다.

See also

Example

```
WAIT 3           'wait for three seconds
Print "*"        'Print a star
```

WAITKEY

Action

시리얼 버퍼에 문자가 수신될 때까지 기다립니다.

Syntax

```
variable = WAITKEY
```

Remarks

variable	문자의 ASCII값을 저장할 변수
----------	--------------------

See also

INKEY

Example

```
DIM a as Byte
A = WAITKEY           'wait for a key press
Print A
End
```

WAITMS

Action

MS 단위로 지정한 시간 만큼 프로그램의 실행을 정지합니다.

Syntax

```
WAITMS mS
```

Remarks

mS	밀리세컨드 단위의 시간수(1-255)입니다.
----	--------------------------

딜레이 타임은 12MHz 클럭 주파수를 기준한 것입니다.

이 명령어로는 정확한 타이밍이 얻어지지 않습니다.

인터럽트에 의하여 지연시간이 증가할 수 있습니다.

이 명령어는 I2C 소자의 EEPROM 써넣기 시간을 만들 때 사용할 수 있습니다.

EEPROM에서 데이터를 라이트 한 다음 10mS 대기 하여야 합니다.

See also

Example

```
WAITMS 10           'wait for 10 mS
Print "*"
```

WHILE..WEND

Action

검사 조건이 참이면 루프를 계속실행 합니다.

Syntax

```
WHILE condition
    statements
WEND
```

Remarks

검사조건이 참일 경우 **WEND** 문을 만날 때까지 계속실행 합니다.
WHILE 문으로 복귀한 후에 조건을 검사하여 참이면 계속 실행 합니다.
거짓이면 **WEND** 이후를 실행합니다.

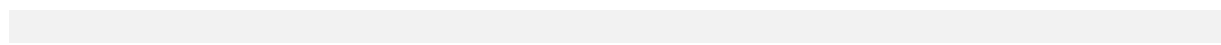
See also

DO .. LOOP

Example

```
WHILE a <= 10
    PRINT a

INC a
WEND
```



HARDWARE - LCD DISPLAY

LCD 디스플레이는 다음과 같이 연결합니다.

LCD-Display	Port	Pin
DB7	P1.7	14
DB6	P1.6	13
DB5	P1.5	12
DB4	P1.4	11
E	P1.3	6
RS	P1.2	4
RW	Ground	5
Vss	Ground	1
Vdd	+5V	2
Vo	0-5 Volts	3

P1.1과 P1.0 그리고 P3은 다른 용도로 사용합니다.

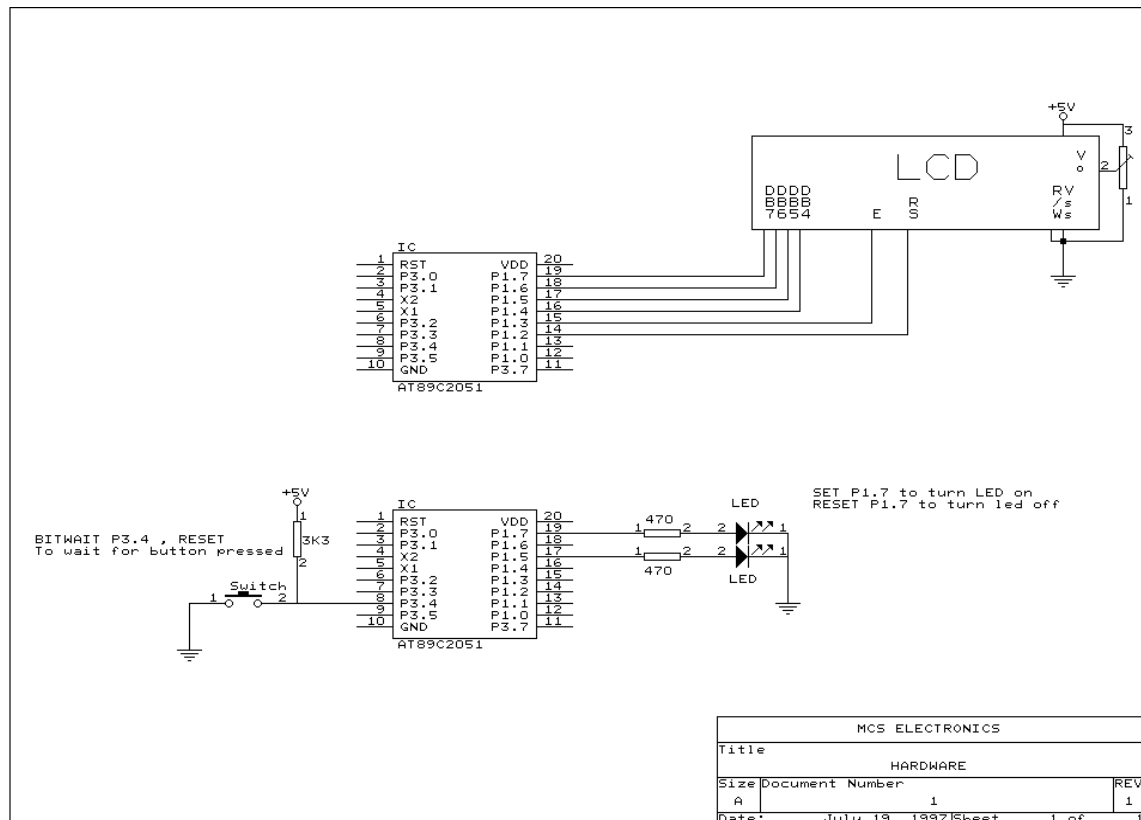
LCD 편 배치는 Option->LCD 에서 변경할 수 있습니다.

LCD 는 4 비트 모드로 동작합니다.

BASCOM 은 많은 LCD 제어명령을 제공합니다. 그러나 자세한 제어를 하기 원한다면 아래의 예를 참고합니다.

Acc = 5	'load register A with value
Call Lcd_control	'it is a control value to control the display
Acc = 65	'load with new value (letter A)
Call Write_lcd	'writes it to the LCD display

lcd_control 과 write_lcd는 BASCOM 에서 호출하는 어셈블리 루틴입니다.



MICROPROCESSOR SUPPORT

일부 마이크로 프로세서는 AT89C2051/8051에 추가적인 사양이 있습니다.

8032/8052/AT89S8252

TIMER2

AT89S8252

WATCHDOG

DATA EEPROM

Alternative port-pin functions

80515,80535,80517,80535

GETAD

WATCHDOG

BAUDRATE GENERATOR

INTERRUPTS and PRIORITY

80517,80537

GETAD

WATCHDOG

BAUDRATE GENERATOR

BAUDRATE GENERATOR1

INTERRUPTS and PRIORITY

AT89S8252 WATCHDOG

AT89S8252 는 워치독 타이머를 가지고 있습니다.

워치독 타이머는 타이머가 일정한 값이 되면 프로세서를 리셋 합니다. 그러므로 프로그램에서 타이머가 최대치가 되기 전 워치독 타이머를 클리어 하여야 합니다.

프로그램이 확실히 동작하기 원할 때 사용합니다. 만약 프로그램이 폭주하여 무한루프에 걸려있을 때 워치독 타이머에 의한 리셋이 발생하여 프로세서가 다시 실행됩니다.

START WATCHDOG 'will start the watchdog timer.

STOP WATCHDOG 'will stop the watchdog timer.

RESET WATCHDOG 'will reset the watchdog timer.

See also

CONFIG WATCHDOG

Example

```
'-----
' (c) 1998 MCS Electronics
' WATCHD.BAS demonstrates the AT89S8252 watchdog timer
' select 89s8252.dat !!!
'-----
Config Watchdog = 2048          'reset after 2048 mSec
Start Watchdog                  'start the watchdog timer
Dim I As Word
For I = 1 To 10000
Print I                        'print value
' Reset Watchdog
'you will notice that the for next doesnt finish because of the reset
'when you unmark the RESET WATCHDOG statement it will finish because the
'wd-timer is reset before it reaches 2048 msec
Next
End
```

DATA EEPROM

AT89C8252는 2K 바이트의 플래시 EEPROM을 가지고 있습니다. 데이터 저장용으로 사용할 수 있습니다. WRITEEEPROM과 READEEPROM의 두개의 명령어가 제공됩니다.

WRITEEEPROM var [, address]

var	BASCOM 변수
address	0 부터 2047 까지의 EEPROM 어드레스 어드레스 지정을 생략하였다면 자동으로 지정되며 report 파일에서 확인할 수 있습니다.

READEEPROM var [,address]

var	BASCOM 변수
address	데이터를 가져올 0 부터 2047 까지의 EEPROM 어드레스

Example

```
Dim S As String * 15 , S2 As String * 10
S = "Hello" : S2 = "test"
Dim L As Long
L = 12345678
Writeeprom S
Writeeprom S2 'write strings
Writeeprom L 'write long
S = "" : S2 = "" : L = 0 'clear variables
Readeeprom L : Print L
Readeeprom S : Print S
Readeeprom S2 : Print S2
End
```

ALTERNATIVE PORT-PIN FUNCTIONS

AT89S8252의 포트는 중복된 기능을 가지고 있습니다.
아래의 표에 중복기능을 표시합니다.

Port pin	Alternate function
P1.0	T2 external count input to timer.counter 2, clock out
P1.1	T2EX timer/counter 2 capture/reload trigger and direction flag
P1.4	/SS Slave port select input
P1.5	MOSI Master data output, slave data input pin for SPI channel
P1.6	MISO Master data input, slave data output pin for SPI channel
P1.7	SCK Master clock output, slave clock input pin for SPI channel
P3.0	RxD serial input port
P3.1	TxD serial output port
P3.2	/INT0 external interrupt 0
P3.3	/INT1 external interrupt 1
P3.4	T0 timer 0 external input
P3.5	T1 timer 1 external input
P3.6	/WR external data memory write strobe
P3.7	/RD external data memory read strobe

/ Means active low

TIMER2 IN 8032 AND COMPATIBLES

일부 마이크로 프로세서는 Timer를 추가로제공합니다. 여기에서 8032 호환 TIMER2에 대하여 설명합니다. TIMER2는 16비트 타이머/카운터이며 이벤트타이머나/이벤트 카운터로 동작됩니다. TIMER 2 는 3 가지의 주요 동작 모드를 가집니다. : 캡처, 오토 리로드(업 또는 다운 카운트), 그리고 바우드 레이트 발생기로 사용할 수 있습니다.

Capture mode

캡처모드에는 두개의 옵션이 있습니다.

* 16-비트 타이머/카운터가 오버 플로우되면 TF2를 셋트합니다. 이 비트는 인터럽트를 발생하는데 사용됩니다.

Counter mode :카운터 모드:

CONFIG TIMER2 = COUNTER, GATE = INTERNAL, MODE = 1

Timer mode:타이머 모드:

CONFIG TIMER2=TIMER, GATE= INTERNAL,MODE =1

* 외부입력핀 T2EX가 1 에서 0으로 변경될 때 TIMER2 의 TL2와 TH2레지스터가 RCAP2L 와 RCAP2H 에 캡춰 됩니다.

Counter mode: 카운터 모드:

CONFIG TIMER2 = COUNTER, GATE = EXTERNAL, MODE = 1

Timer mode: 타이머 모드:

CONFIG TIMER2=TIMER,GATE=EXTERNAL,MODE=1

T2EX 의 변화에 의하여 T2CON레지스터의 EXF2 가 셋트됩니다. TF2 처럼 EXF2가 인터럽트를 발생합니다.

TIMER2인터럽트 루틴에서 TF2의 상태를 확인할수 있으며 EXF2 는 인터럽트를 발생할 것인지를 결정합니다. (이모드에서 리로드 값은 없습니다. T2EX로부터 캡처 이벤트가 발생하여도 T2EX 핀이나 OSC/12 펄스를 카운트합니다.)

Auto reload mode

16비트 오토 리로드 모드에서 TIMER2 는 카운트 업이나 카운트 다운이 가능한 타이머나 카운터를 구성할수 있습니다.

카운팅 방향은 DCEN 에 의하여 결정됩니다.

TIMER2는 초기값으로부터 FFFFH 까지 카운팅 합니다.그리고 TF2 오버플로우 가 발생합니다.

이때 16비트값 RCAP2L과 RCAP2H가 TIMER2로 리로드 됩니다.

RCAP2L 과 RCAP2H는 소프트웨어에 의하여 프리셋트 하여야 합니다.

Counter mode:

CONFIG TIMER2=COUNTER,GATE=INTERNAL,MODE=0

Timer mode:

CONFIG TIMER2=COUNTER,GATE=INTERNAL,MODE=0

EXEN2=1 일때 입력EXEN2 가 1 에서 0 으로 변화하거나 오버플로우 상태가 되면 16비트 리로드값이 트리거됩니다.

이상태변화는 EXF2비트를 셋트합니다.

TIMER2 인터럽트가 가능상태로 되었다면 TF2나 EXF2 가 1 이될 때 발생합니다.

Counter mode: 카운터 모드:

CONFIG TIMER2=COUNTER,GATE=EXTERNAL,MODE=0

Timer mode:타이머 모드:

CONFIG TIMER2=TIMER,GATE=EXTERNAL,MODE=0

TIMER2는 카운트 업이나 다운으로 설정할 수 있습니다. 이 모드에서 T2EX핀에 의하여 카운트 방향을 결정할 수 있습니다. T2EX핀에 로직 1 을 인가하면 TIMER2는 카운트 업합니다. TIMER2 가 &HFFFF에서 오버플로우가 되며 TF2를 셋트합니다. 그리고 인터럽트를 발생립니다. 인터럽트가 가능상태이면 타이머 오버플로우는 16 비트값 RCAP2L 과 RCAP2H를 타이머 레지스터 TL2 와 TH2 로 리로드합니다.

Counter mode:

CONFIG TIMER2=COUNTER,GATE=INTERNAL/EXTERNAL,MODE=0,DIRECTION=UP

Timer mode:

CONFIG TIMER2=COUNTER,GATE=INTERNAL/EXTERNAL,MODE=0,DIRECTION=UP

T2EX 에 로직 0 을 입력하면 TIMER2 는 감소 카운트 합니다.

타이머는 TL2와 TH2가 RCAP2L 과 RCAP2H와 같은 값이 되면 언더 플로우가 발생합니다.

TIMER2 under flows sets the TF2 flag and causes &HFFFF to be reloaded into the timer registers TL2 and TH2.

TIMER2가 언더플로우 발생되면 TF2 플래그가 셋트되며 타이머레지스터 TL2 와 TH2에 &HFFFF가 리로드 됩니다.

Counter mode:

CONFIG TIMER2=COUNTER,GATE=INTERNAL/EXTERNAL,MODE=0,DIRECTION=DOWN

Timer mode:

CONFIG TIMER2=COUNTER,GATE=INTERNAL/EXTERNAL,MODE=0,DIRECTION=DOWN

Baud rate generator

이모드는 시리얼 포트의 바우드레이트 발생기로 사용될 수 있습니다. TIMER1이 이러한 목적을 위하여 사용될수 있습니다.

CONFIG TIMER2=TIMER,GATE=INTERNAL,MODE=2

Receive only

이 모드에서는 단지 데이터 수신을 하기 위하여 바우드 레이트를 사용합니다.

TIMER1 이 데이터 송신을 위한 다른 바우드 레이트로 사용됩니다.

CONFIG TIMER2=TIMER,GATE=INTERNAL,MODE=3

이때 Timer1은 어셈블리루틴으로 설정하여야 합니다.

Transmit only

이모드에서는 단지 데이터 송신을 위하여만 바우드 레이트가 사용됩니다.

TIMER1 이 데이터 수신을 위한 다른 바우드 레이트로 사용됩니다.

CONFIG TIMER2=TIMER,GATE=INTERNAL,MODE=4

이때 Timer1은 어셈블리루틴으로 설정하여야 합니다.

Clock output

일부 8052 프로세서는 P1.0 으로 50% 듀티비의 펄스를 출력할 수 있습니다.

CONFIG TIMER2=TIMER,MODE=5

출력주파수는 = (fOSC / 4) / (65536-CAPTURE)

CAPTURE = value 방법으로 캡처 레지스터를 설정합니다.

How to determine what caused the interrupt

TCON.7 을 오버플로우에 의한 인터럽트가 발생되었는가를 확인 하기 위하여 테스트 할 수 있습니다.

T2EX가 네가티브 상태 변화할 때 리로드 또는 캡춰인가를 T2CON.6으로 확인할 수 있습니다.

Timer2_ISR:

If T2CON.7 = 1 Then

 Print "Timer overflowed"

Else

 If T2CON.6 = 1 Then

 Print "External transition"

 End if

End If

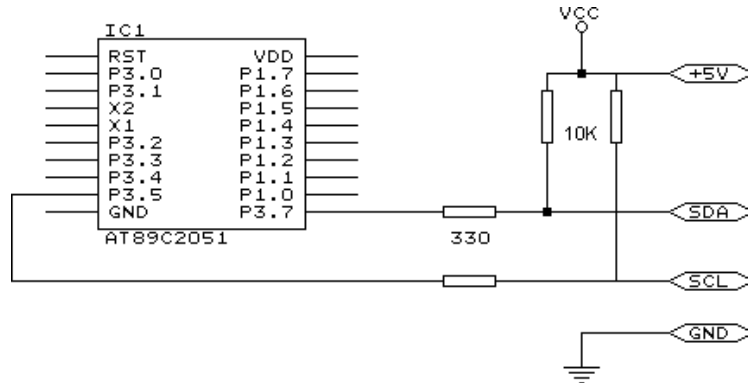
Return

Error codes (에러 코드)

Nr	Error message
1	BASIC 소스파일이 없습니다.
2	FLASHROM 보다는 코드사이즈가 더 큼니다.
3	알지 못하는 명령어가 사용되었습니다.
4	확장인것 같습니다.
5	잘못된 변수 또는 타입설정이 잘못 되었습니다.
6	너무많은 파라메타가 지정되었습니다.
7	BIT 변수의 여유공간이 없습니다.
8	BYTE 변수 여유공간이 없습니다.
9	INTEGER 변수 공간이 없습니다.
10	변수 타입(BIT,BYTE or INTEGER) 설정이 잘못되었습니다.
11	DIM 문에서 AS로 예상됩니다.
12	컴마(,) 가 예상됩니다.
13	알지 못하는 인터럽트입니다.
14	IF THEN 인것 같습니다.
15	FOR,DO 또는 WHILE 인것같습니다.
16	파라메터에서 잘못된 수를 사용한 것 같습니다.
17	잘못된 비교연산자(=,>,<,<=>,>=)입니다.
18	THEN 이 예상됩니다.
19	TIMER0, TIMER1이 예상됩니다.
20	DO 가 예상됩니다.
21	UNTIL 이 예상됩니다.
22	부정확한 연산식입니다.
23	FOR 가 예상됩니다.
24	WHILE 이 예상됩니다.
25	변수의 초기 정의가 되어있지 않습니다.
26	소스파일이 보이지 않습니다.
27	라벨이 보이지 않습니다.
100-134	내부 어셈블리 에러입니다. SAMPLE 전자로 연락하시기 바랍니다.
135	너무많은 RAM 이 사용되었습니다.
136	변수가 이미 정의되어 있습니다.
137	상수값은 반드시 1-8 입니다.
138	지정한 크리스탈 주파수에서 BAUD RATE를 설정할수 없습니다.
139	9 개의 파라메타가 예상됩니다.
140	COUNTER0 와 COUNTR1이 예상됩니다.
141	" = "가 예상됩니다.
142	ALIAS 의 최대개수 는 128입니다.
143	라벨이 중복되어 정의 되었습니다.
144	수치값이 바이트 변수에 적합하지 않습니다.(바이트는 0 부터 255까지이며 음수는 표시할 수 없습니다.)

I2C

아래 도면은 I2C 버스를 구성하는 예입니다. R1 과 R2는 저항 330 OHM 입니다. R3와 R4 는 저항 10K OHM 입니다. 하드웨어 설계시 I2C 인터페이스에서 사용한 Port를 컴파일러에게 알려 주어야 합니다.



Hardware related commands / 하드웨어 관련 명령어

마이크로 프로세서는 0 부터 24MHz 범위의 크리스탈이 사용됩니다. 주파수는 12 분주되므로 12MHz 크리스탈을 사용한 CPU 는 1MHz 의 속도로 동작합니다. 대부분의 명령어가 1 클럭 사이클에 실행되므로 1 MIPS의 속도로 동작합니다.

RS-232 와 관련된 명령으로 INKEY, PRINT와 INPUT문이 있습니다. TIMER1은 시스템 클럭에 연결되어있으므로 ON TIMER 1 거 | 같이 다른 목적으로 사용해서는 안됩니다. RS-232 관련 명령어를 사용하지 않는다면 TIMER1 을 사용하는것이 가능합니다.

BAUD RATE는 시스템 클럭을 분주하여 얻어집니다. 일반적으로 11.0592MHz를 사용합니다. BAUD RATE는 매우 정확하게 발생되어야 합니다. 다른 크리스탈 주파수를 사용하면 2400또는 4800 BAUD RATE가 근사값으로 얻어지며 이보다 빠른 BAUD RATE는 얻어지지 않습니다. 컴파일러 옵션에서 크리스탈 주파수와 BAUD RATE를 설정하거나 프로그램 소스상에서 메타 커멘드 \$BAUD = xx 와 \$CRYSTAL = xx 를 사용하여 지정하는 것이 가능합니다. 정확하게 BAUD RATE 가 만들어지면 레포트 파일에 결과가 나타납니다.

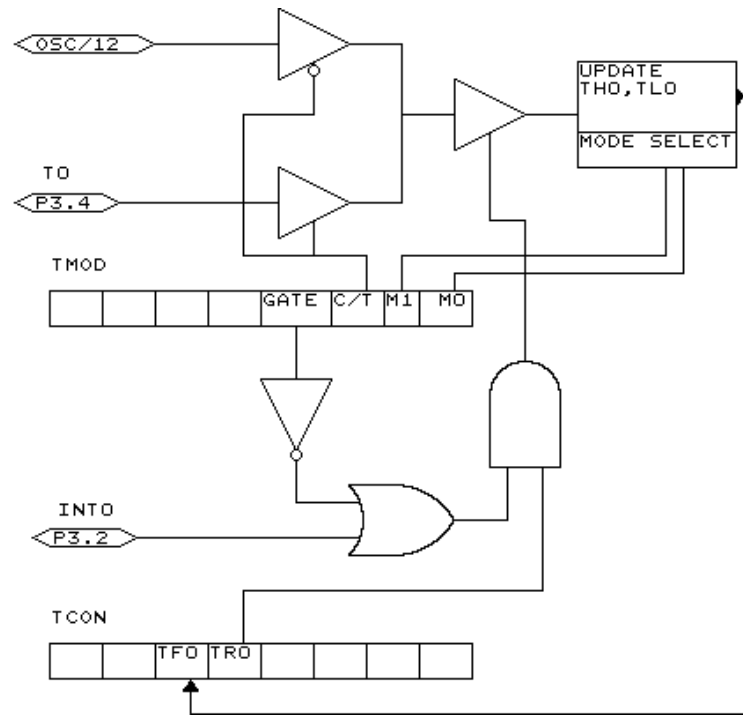
Clock

클럭 주파수는 시스템주파수를 12로 나눈 것입니다. 12MHz 크리스탈은 마이크로 세컨드단위로 레지스터가 증가함을 의미합니다.

Timers and Counters

AT89C2051은 TIMER0, TIMER1 두개의 16비트 타이머를 가지고 있습니다. 아래그림은 timer0의 내부구성을 나타낸 것입니다. TIMER0 와 TIMER1은 거의 동일하므로 TIMER1 은 TIMER0를 참고하면 됩니다.

The AT89C2051 has two 16-bit timers. Named



각 카운터는 두개의 SFR을 가지고 있습니다. **TIMER0**의 SRF은 **TL0**와 **TH0** 입니다.

TL0는 **TIMER0**의 하위 바이트 이며 **TH0** 는 **TIMER0**의 상위 바이트 입니다. 이 두개의 레지스터로 16 비트 카운터를 구성합니다.

타이머는 카운터로서 사용합니다. 타이머는 입력신호로 시스템주파수를 12 분주한 것입니다. 따라서 타이머는 주기적으로 값이 증가됩니다.

카운터는 외부펄스에 의하여 자신의 값이 증가됩니다. 외부펄스는 **TIMER0** 인 경우 **P3.4** **TIMER1** 인 경우 **P3.5** 를 통하여 입력됩니다.

타이머/카운터는 **TR0** 비트에 의하여 동작유무가 결정됩니다.

BASIC 프로그램에서 **STOP TIMER0/COUNTER0** 를 사용하여 정지하는 것이 가능합니다. 마찬가지로 **START TIMER0/TIMER1**으로 동작 시작하는 것이 가능합니다.

Timer/Counter는 **P3.2**에 의하여 제어 하는 것이 가능합니다. 이것은 **INT0** 이지만 타이머를 제어 하는 것이 가능합니다.

GATE은 **INT0** 가 카운터 타이머를 제어하는 것을 허락하거나 금지할 수 있습니다. **GATE** 가 0 이면 **INT0** 의 상태에 따라서 타이머/카운터가 동작상태가 결정됩니다. **GATE** 와 **INT0** 와의 조합 가능한 상태에 따른 카운터/타이머 동작은 다음과 같습니다.

GATE	INT0#	카운터/타이머
L	L	정지
L	H	동작
H	L	정지
H	H	정지

타이머/카운터는 4 가지의 동작모드가 가능합니다.

- **mode 0 : 13-bit counter.**
카운터가 오버플로우 되면 인터럽트를 발생립니다. 그러므로 8192 펄스 주기로 다음인터럽트가 발생립니다.
- **mode 1 : 16-bit counter.**

모드 1 은 모드 0 과 비슷합니다. 16비트 카운터로 동작합니다. 65536 입력펄스에 대하여 다음 번 인터럽트가 발생됩니다.

- **mode 2 : 8-bit auto reload.**

- TL0는 8 비트 타이머/카운터로 작동합니다. 타이머 카운터가 오버 플로우 되면 TH0 값이 TL0 으로 저장됩니다. 그리고 카운트가 계속됩니다. 인터럽트가 오버 플로우시 마다 매번 발생되며 자동으로 리로드가 됩니다.
- mode 3 : TIMER1 is inactive and holds its count.

타이머/카운터는 CONFIG 문에서 설정합니다.

CONFIG TIMER0= COUNTER/TIMER, GATE=INTERNAL/EXTERNAL, MODE=0/3

CONFIG 명령에서 타이머인지 카운터인지 지정하는 것이 첫번째 이어야 합니다.

GATE는 INT0 핀에 의하여 제어되는 것을 허용하거나 금하는 것을 결정합니다.

MODE 는 timer/counter 모드(0 - 3)을 지정합니다.

CONFIG TIMER0 = COUNTER, GATE = INTERNAL, MODE=2 는 TIMER0를 카운터로, INT0에 의한 외부 게이트 콘트롤 금지, 자동 리로드(mode 2) 를 설정한 것입니다.

카운터/타이머가 CONFIG문에 의하여 설정 되었더라도 그냥 동작하지는 않습니다.

이 다음에 START TIMER0 문이 반드시 와야 합니다.

타이머/카운터가 오버 플로우 되어 발생하는 인터럽트를 처리하기 위하여 ON TIMERx 문을 사용하여야 합니다.

타이머 카운터가 mode 2 로 동작된다면 LOAD TIMERx, value 문에 의하여 자동 리로드 값을 지정하여야 합니다. 자동 리로드 값은 8 비트 이므로 255 를 넘을 수 없습니다. 오토 리로드 모드(mode 2)는 8 비트 레지스터를 이용하므로 최장 255uS를 얻는 것이 가능합니다 . 10uS의 주기를 갖도록 하려면 246을 설정하여야 합니다.

COUNTER0 와 COUNTER1는 timer/counter0 과 timer/counter1 의 값을 갖는 변수로 사용됩니다.

COUNTER0 = value 에 의하여timer/counter의 값을 변경하는 것이 가능합니다.

PORT3는 복합기능을 갖습니다.

P3.1 = 1 이나 SET P3.1 처럼 일반적인 포트와 다음 표와 같은 복합 기능을 사용합니다.

Port	Alternative function
P3.0	RxD receive data for RS-232
P3.1	TxD transmit data for RS-232
P3.2	INT0 interrupt 0 input/timer 0 gate control
P3.3	INT1 interrupt 1 input/timer 1 gate control
P3.4	T0 timer 0 input or counter input
P3.5	T1 timer 1 input or counter input
P3.5	-
P3.7	-

RS-232 와 관련된 PRINT, INPUT 명령을 사용하면 P3.0 과 P3.1 은 RS-232 인터페이스로 사용됩니다.

인터럽트INT0/ INT1 을 사용하였으면 인터럽트 소스를 연결하여야 합니다. INTx 인터럽트는 신호의 하강 에지나 LOW 신호에서 발생합니다. 다음은 트리거 조건을 지정하는것 입니다.

SET TCON.0	하강 에지에서 INT0 인터럽트 발생
RESET TCON.0	Low 레벨에서 INT0 인터럽트 발생
SET TCON.2	하강 에지에서 INT1 인터럽트 발생
RESET TCON.2	Low 레벨에서 INT1 인터럽트 발생

인터럽트 입력이 Low 상태가 지속 될 때까지 인터럽트 처리를 원하면 TCON.x를 RESET 하며 하강에지 시점에서만 인터럽트 처리가 되는 것을 원하면 TCON.x 를 SET 합니다.

하드웨어 인터럽트가 발생 하였을 때 TCON.1 과 TCON.3 을 이용하여 테스트할 수 있습니다.

서브루틴이나 인터럽트 서비스루틴에서 복귀하는 RETURN 문이 실행될 때 리셋됩니다. TCON.1 은 INT0의 검사용이며 TCON.3는 INT1 을 검사하기 위한 것입니다.

Internal Registers / 내부 레지스터

BASIC 에 의하여 다음의 8051 내부레지스터를 직접 조작하는 것이 가능합니다. 이 레지스터 이름은 예약어 입니다. 8051의 기종에 따라 차이가 있으며 옵션->컴파일러에서 선택합니다.

BIT addressable registers/ 비트 지정이 가능한 레지스터

TCON	Timer/counter control
P1	Port 0 latch
SCON	Serial port control
IE	Interrupt enable
P3	Port 3 latch
IP	Interrupt priority control
PSW	Program status word
ACC	Accumulator
B	B register

BYTE addressable register

SP	Stack pointer
DPL	Data pointer low word
DPH	Data pointer high word
PCON	Power control
TMOD	Timer/counter mode control
TL0	Timer/counter 0 low byte
TL1	Timer/counter 1 low byte
TH0	Timer/counter 0 high byte
TH1	Timer/counter 1 high byte
SBUF	Serial data port
P1	Port 1 latch
P3	Port 3 latch

레지스터 이름과 어드레스 위치는 *.DAT 에 정의되어있습니다. 레지스터를 직접 조작할 때 세심한 주의가 필요합니다. A와 B는 BASCOM에서 자주 사용되는 것이며 SP 레지스터는 건드리지 않는 것이 좋습니다. 비트지정 가능 변수는 SET/RESET 문을 사용합니다. 그리고 비트 변수로서 사용됩니다. 바이트 지정 가능 레지스터는 바이트 변수로 사용됩니다. 내부레지스터는 예약어 입니다. BASCOM 에서 다른 변수 이름으로 사용하면 안됩니다. 예를들면 DIM B as BYTE 는 안됩니다. 왜냐하면 B 는 내부레지스터의 이름이기 때문입니다. 그러나 B 레지스터에 값을 지정하려면 " B = 숫자 " 와 같이 지정하면 됩니다.

Power Up / 파워 업

8051 에 전원이 공급되면 모든 포트가 High 상태가 됩니다. 어떤 Port이던지 데이터를 읽으려면 Port는 High 상태에 있어야 합니다. 그러므로 데이터를 읽기 전에 Power on 후 처음 한번은 255 (&HFF)을 포트에 써넣어야 합니다. 비트를 개별적으로 입력과 출력의 설정이 가능합니다. 예를들어 P1 = &B00001111 로 하였다면 P1.0, P1.1, P1.2, P1.3은 입력핀으로 사용이 가능합니다.

In line assembly

BASCOM 에서 인라인 어셈블리의 사용이 가능 합니다. 어셈블러에서 주석문은 ";" 기호 입니다. 다음의 두가지 방법으로 어셈블리 프로그램이 가능합니다.

For example :

Dim a as byte

mov {a}, #10 ; variables must be enclosed with { }

Print a

Or you can include an assembler file :

\$INCLUDE myasm.asm

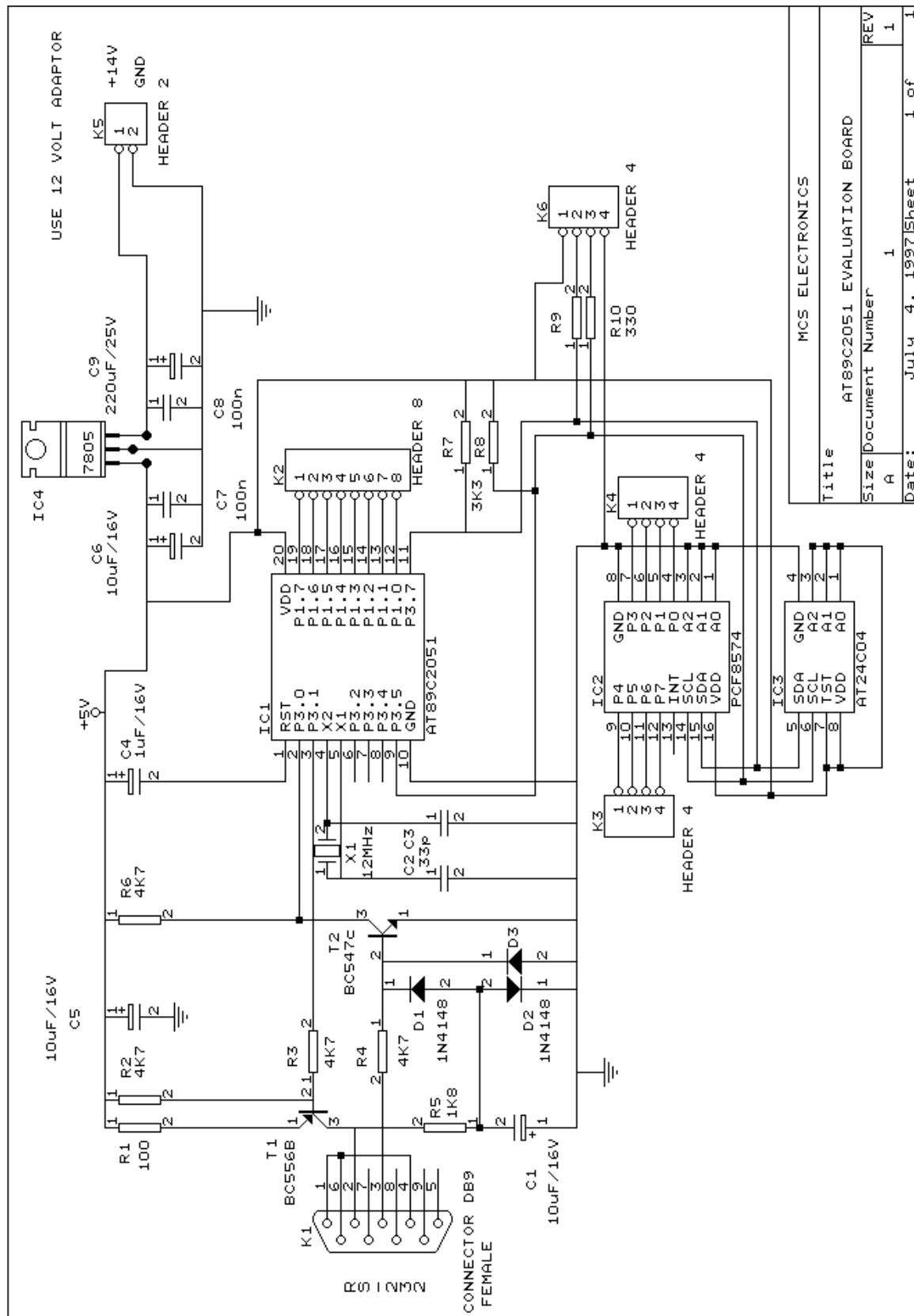
어셈블러 파일의 확장자는 반드시 .ASM 이어야 합니다.

Initialization / 초기화

BASCOM LT/8051 은 사용된 명령어에 따라서 초기화 작업이 실행됩니다. 초기화 루틴을 직접 작성하기 원하면 메타커멘드를 사용합니다.

\$NOINIT

스택포인트 SP와 LCD-디스플레이(사용된 경우에 해당) 초기화 코드만 작성됩니다.



8051 TEST PROGRAM

8051 프로세서를 테스트합니다. 모든 포트에 연결된 LED 를 순차적으로 점등하며 시리얼 포트에 입력된 데이터를 반환합니다. XTAL1 주파수는 11.0592MHz 이며 통신속도는 1200BPS입니다.

```
'-----
'
' SE-516T11M1200.BAS ----- Copy right : SAMPLE Electronics co.
'
```

```
' 설명 :
'
```

```
' SE-516SP 의 I/O 포트와 시리얼 인터페이스를 실험하기 위한 것입니다.
'
```

```
' 시리얼 인터페이스의 통신속도는 오실레이터가 11.0595MHz 일 때
' 1200 BPS 입니다.
'
```

```
' 사용법:
'
```

```
' STEP 1: SE-516P 에 프로세서를 실장하고 이 프로그램을 컴파일하여
'   만들어진 HEX 파일을 프로그램 합니다.
'
```

```
' STEP 2: SE-516P 에 연결된 파워를 분리합니다.
'
```

```
' STEP 3: SE-516P 에 연결된 PC의 시리얼 포트를 SE-516S 오른쪽에 있는
'   DB-9S 커넥터로 연결합니다.
'
```

```
' STEP 4: 다시 SE-516P에 전원 아답터를 연결합니다.
'
```

```
' STEP 5: SE-516S 의 SW2 와 SW3를 조작하여 OSC 와 RS232 로 표시된 LED 가
'   점등(ON)되게 합니다.
'
```

```
' STEP 6: Bascom 8051 IDE 의 윈도우 메뉴바에서 Tools -> Terminal Emulator 를
'   선택합니다. 기본값으로 1200BPS 로 설정되어 있습니다.
'
```

```
' STEP 7: SE-516S 에서 Port 확인용 LED 가 순차로 회전하면 점등하는것이 확인되면
'   PC 의 키보드 아무키나 입력합니다. 이때 터미널 창에는 입력한 키와
'   리턴된 키가 표시됩니다.
'
```

```
' 참고: Option -> Communication 에서 통신포트(Com1 또는 Com2)와
'   통신속도를 변경할 수 있습니다.
'
```

```
'-----
$crystal = 11059200
```

```
' XTAL Frequency 11.0592MHz
```

```
$baud = 1200
```

```
' RS232 Serial Baud Rate 19200 BPS
'
```

```
Dim C As Byte
```

```
Dim A As Byte
```

```
A = &B11111110
```

```
Main:
```

```
C = Inkey
```

```
If C > 0 Then
```

```
    Print Chr(c);
```

```
End If
```

Rotate A , Left

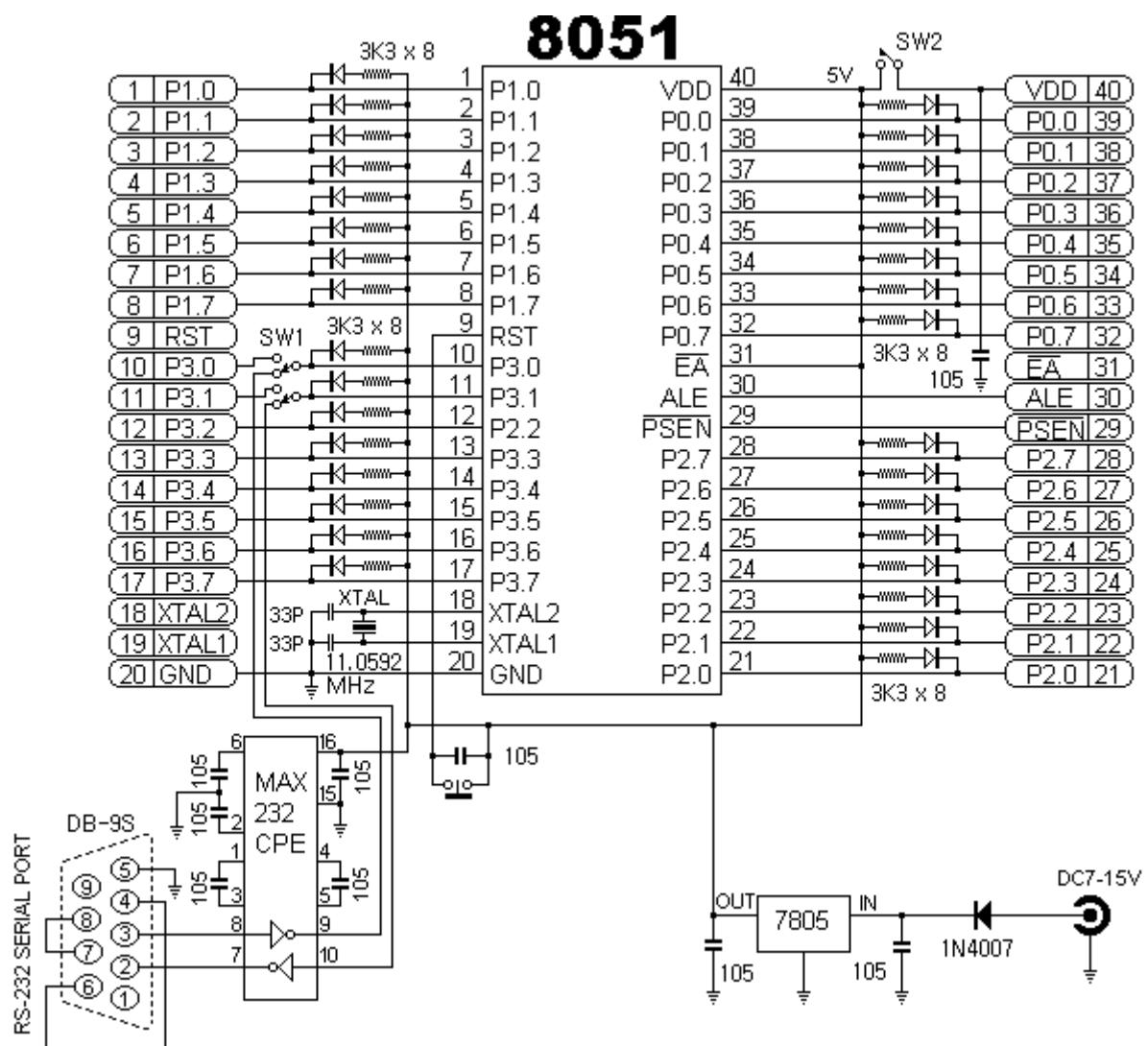
$$P_0 = A$$
$$P1 = A$$
$$P2 = A$$
$$P3 = A$$

Waitms 200

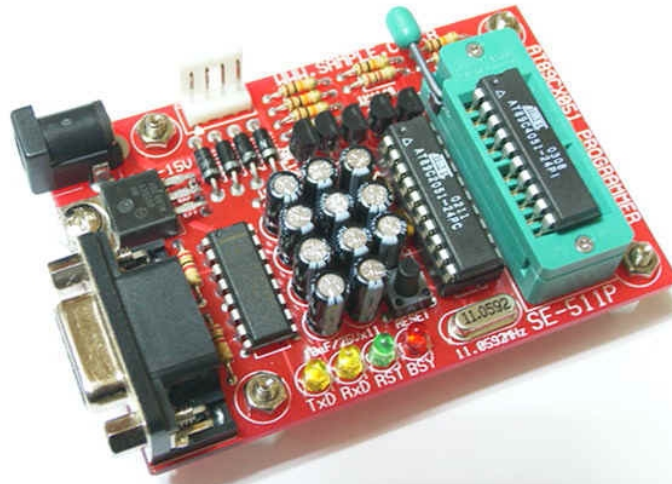
' 200 M SEC Delay Time

[Goto Main](#)

'----- End of FILE -----'



AT89C1051, AT89C2051, AT89C4051 프로그래머 제작



ATMEL사의 AT89C1051, AT89C2051, AT89C4051 프로그래머를 제작합니다.

AT89Cx051의 프로그래밍 모드

라이팅 / Programming :

라이팅 알고리즘은 데이터 쉬트의 상세히 나와있습니다. 그림-1 은 플래쉬 메모리에 코드를 라이팅 하기위한 기본 도면이며 그림-2 는 플래쉬 메모리의 코드를 읽기 위한 도면입니다. 그림 3에 핀에 대한 동작조건이 표시되어 있습니다. AT89Cx051은 공장 출하 시 지워진 상태(전구간의 Flash ROM 이 1)의 Flash ROM 을 가지고 있으며 코드 프로그램이 라이팅 가능하도록 되어있습니다. 코드 프로그램 라이팅이란 Flash ROM 의 비트 상태를 1에서 0으로 변경하는 것입니다. 이레이즈는 모든 Flash ROM의 모든 비트를 1로 초기화하는 것입니다. 프로그램은 바이트 단위로 어떤 위치든지 관계없이 가능하지만 이레이즈는 바이트 단위로 선택할 수 없으며 Flash ROM 전 구간에 대하여 실행됩니다. AT89Cx051은 내부에 어드레스 카운터를 가지고 있으므로 표준 8051과 같이 파라렐 어드레스를 입력하지 않고 XTAL1 핀에 클럭 신호를 공급하면 됩니다. 어드레스 카운터는 RST핀을 0에서 1로 설정할 때 000 이 됩니다. AT89Cx051은 어드레스 카운터가 증가형이므로 사실상 순차적으로 프로그램 하게 됩니다. 어드레스 카운터는 XTAL1핀이 0 에서 1 로 변할 때 1 증가합니다.

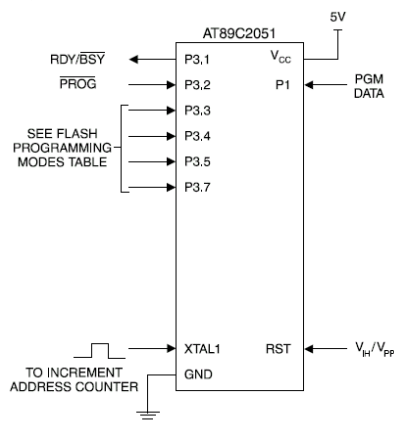


그림-1 : Flash 프로그램

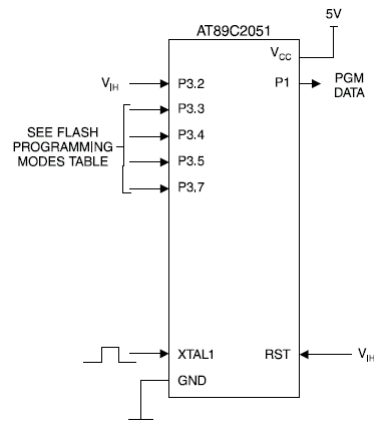


그림-2: Flash 리드

Mode		RST/VPP	P3.2/PROG	P3.3	P3.4	P3.5	P3.7
코드 라이트		12V	- \ / -	L	H	H	H
코드 리드		H	H	L	L	H	H
락 비트	비트1	12V	- \ / -	H	H	H	H
	비트2	12V	- \ / -	H	H	L	L
칩 이레이즈		12V	- \ / -	H	L	L	L
리드 ID 바이트		H	H	L	L	L	L

표-1: Flash ROM 프로그래밍 모드

AT89Cx051 프로그램 알고리즘

1. 전원(+5V)을 공급한다. XTAL1 핀과 RST 핀을 0 으로 한다.
2. RST 핀과 P3.2를 H 로 설정한다.
3. P3.3, P3.4, P3.5, P3.7을 모드에 따라 H 또는 L 로 선택한다.(표 1 참조)
4. P1.0 에서 P1.7 까지 데이터 코드를 입력한다.
5. RST 핀에 12V 를 공급한다.
6. P3.2 에 펄스를 인가한다. 바이트 라이트 사이클은 내부에서 만들어지며 약 1.2mS 이다. 바이트 프로그램이 완료되면 P3.1에서 Low 펄스가 출력된다.
7. RST 핀에 +5V를 인가한다.
8. XTAL1 핀에 High 펄스를 인가한다.
9. 원하는 크기의 라이팅이 완료될 때까지 4,5,6,7,8을 반복한다.

Flash 메모리는 바이트 단위로 프로그램 가능하지만 어드레스 카운터는 1 씩 증가형이므로 0번지 부터 마지막 번지의 데이터를 버퍼에 준비하여두고 순차적으로 프로그램합니다. 데이터 스위트에는 프로그램과 베리파이가 같은 사이클에 있으나 대부분의 프로그래머 제작사들은 프로그램과 베리파이를 나누어 실행합니다. 즉 Flash ROM 전 구간에 대하여 라이팅을 먼저 실시하고 다음 전구간에 대한 베리파이를 실행합니다

프로그램 베리파이 / Program Verify:

Flash ROM에 라이팅 된 메모리가 정상인지를 확인하는 용도와 Flash ROM 에 라이팅 된 데이터를 읽어 PC 에 저장하기 위한 용도로 사용됩니다. LB1과 LB2 가 프로그램 되지 않았다면 AT89Cx051 내부의 코드데이터를 읽어낼 수 있습니다.

이레이즈 / Chip Erase:

Flash ROM의 특정구간이나 바이트 단위로 이레이즈 할 수 없습니다. 이레이즈를 실행하면 Flash ROM 전구간의 데이터는 1로 초기화됩니다. 이레이즈 시에는 복사방지 비트인 LB1과 LB2 도 동시에 지워진다. P3.2 에 10mS 의 Low 펄스를 인가하여야 합니다.

Reading the Signature Bytes:

디바이스의 제작사와 종류를 표시합니다. 3 개의 바이트가 있으며 표2 와 같은 값을 출력한다. 0xIE 제조사가 ATMEL 임을 표시하는 것입니다.

프로세서	Flash ROM	000	001	002
AT89C1051	1 K Bytes	0xIE	0x11	0x00
AT89C2051	2 K Bytes	0xIE	0x21	0x00
AT89C4051	4 K Bytes	0xIE	0x41	0x00

표-2: Signature Bytes

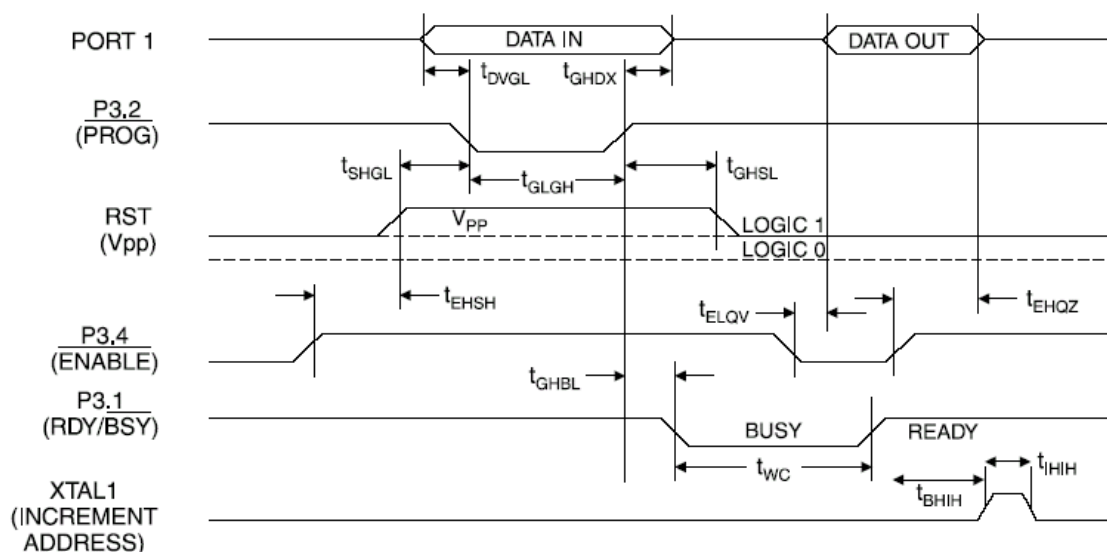


그림-3: 프로그램 타이밍

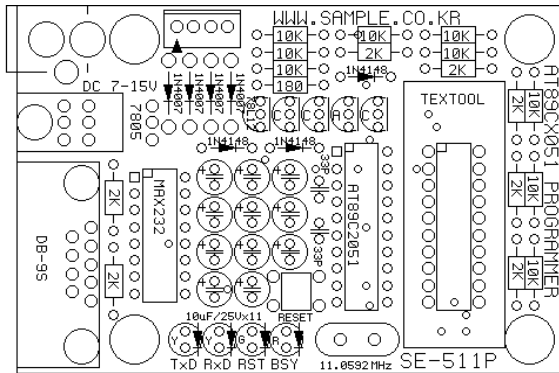
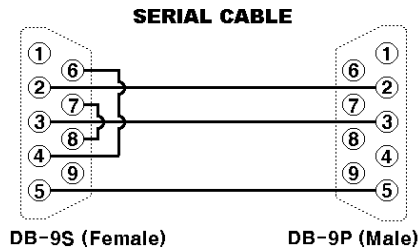


그림-4: 프로그래머 PCB의 실크 인쇄면



DB-9S (Female) DB-9P (Male)

WWW.SAMPLE.CO.KR

그림-5: 시리얼 케이블 결선도

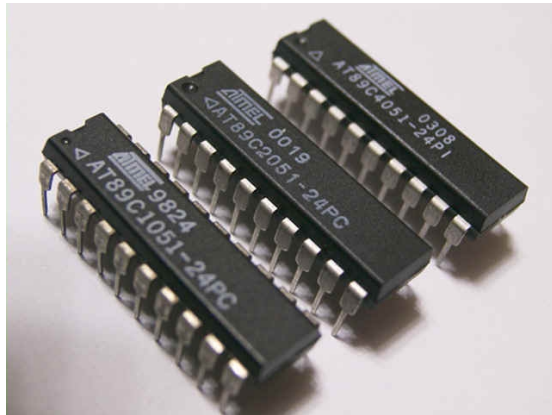


그림-6: 사용 가능한 ATMEL 프로세서

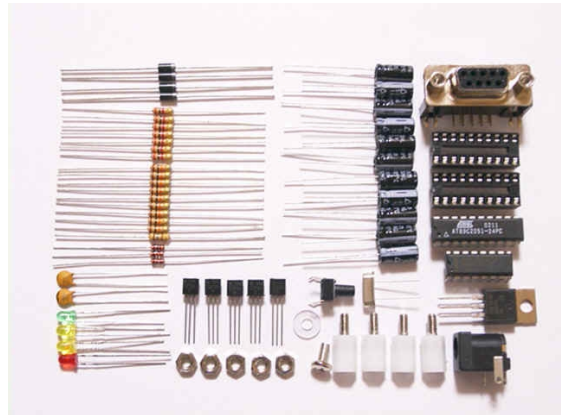


그림-7: 제작에 필요한 부품

PC 의 COM 포트와 SE-511P를 연결하는 시리얼 케이블은 1: 1 로 결선된 9Pin Dsub 연장 케이블을 사용합니다.

시리얼 케이블을 자작할 때에는 3 선만 사용하여 케이블을 제작합니다.

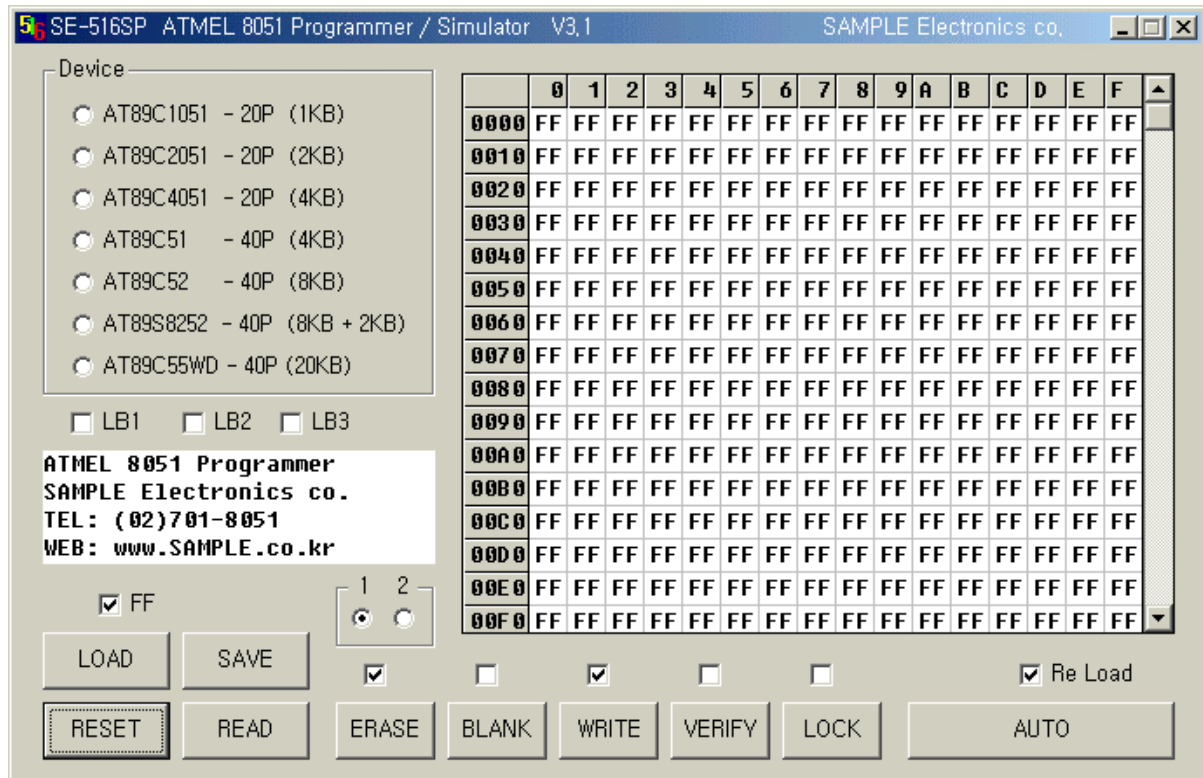


그림-8: PC 에서 드라이버 프로그램 실행

PC 에서 실행된 드라이버 프로그램 윈도우 입니다. Visual BASIC으로 작성하여 사용자가 원하는 기능을 추가하거나 변경할 수 있습니다. 소스 프로그램은 샘플전자 홈페이지에서 다운 받을 수 있습니다.

SE-511P

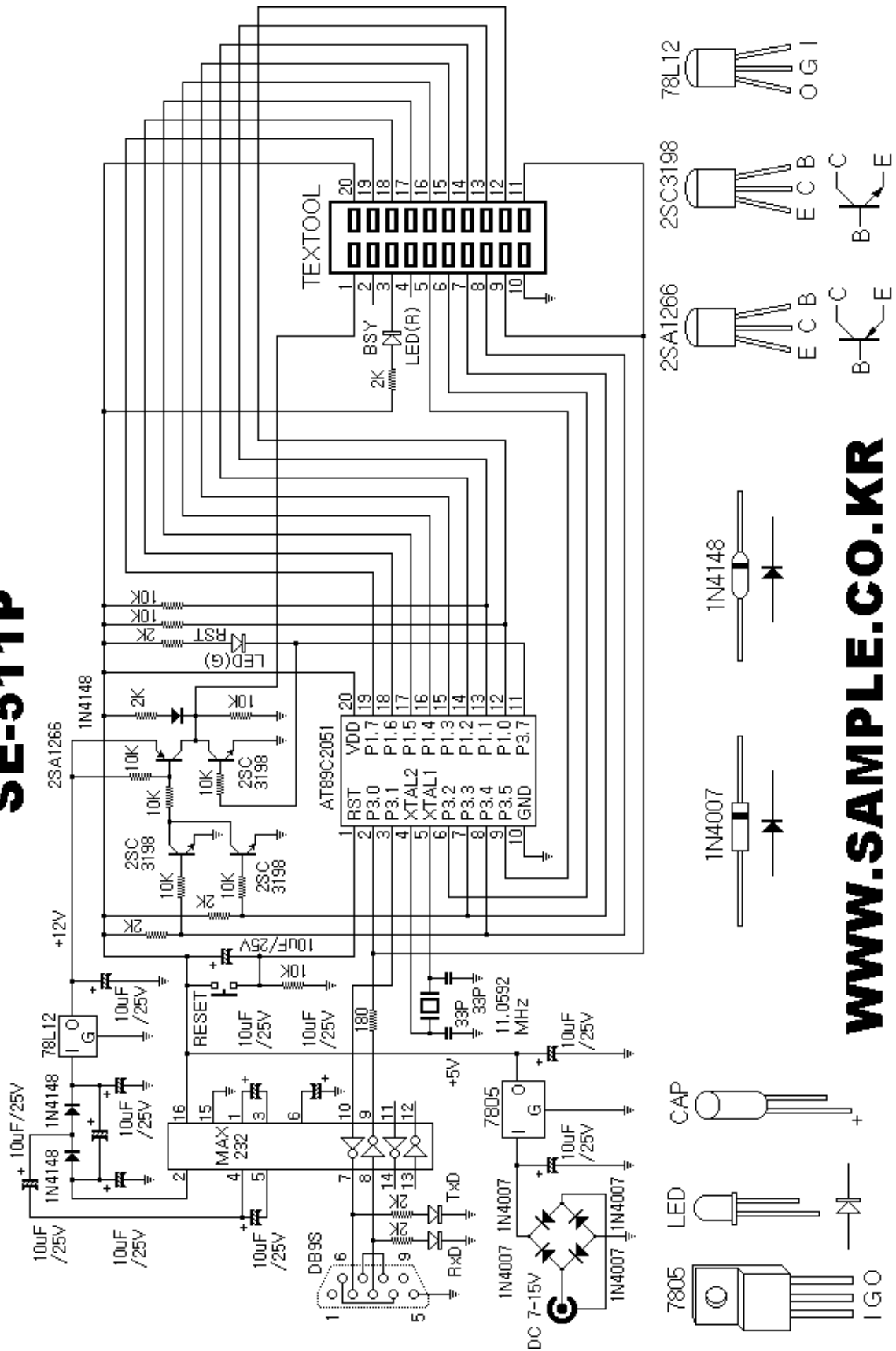


그림-9: 회로도

프로그래머는 가능한 간단한 회로로 구성하는 것을 목표로 하고 설계하였습니다. PC 와의 인터페이스회로, VPP 발생회로, 타겟 프로세서 제어회로로 구성되어있습니다. 모든 제어는 AT89C2051 프로세서에서 실행됩니다. BASIC 언어로 작성하였으며 네델란드 MCS Electronics(2)사의 BASCOM 8051 Demo 버전 컴파일러를 사용하여 HEX 파일을 생성합니다. 소스 코드를 공개하였으므로 기능 변경이나 추가가 가능합니다.

프로세서 제어회로, 전원회로:

프로세서는 AT89C2051을 사용합니다. 2K Bytes의 Flash ROM 과 128 Bytes의 RAM 을 내장하고 있으며 시리얼 통신이 가능합니다. 11.0592MHz의 ATS를 사용하여 동작합니다. 전원은 아답터를 사용하고 7805 레귤레이터 IC를 사용하여 안정된 5V 전압을 공급합니다. 아답터의 전압은 7V-30V 까지 가능하지만 9V를 사용하는 것을 권장합니다. 브릿지 정류기는 아답터의 극성에 관계없이 동작하기 위하여 사용한 것입니다. P3.0 핀은 호스트 컴퓨터로부터 데이터를 받는 RxD와 타겟 CPU 의 P3.5, P3.7 을 제어하는 신호로 검용하여 사용합니다. 이 방법은 8051 프로세서의 구조 때문에 가능한 것이며 포트 핀에 대하여 사용 용도와 입출력 방향을 초기에 지정해야 되는 다른 계열의 프로세서에서 이 방법을 사용하기란 곤란합니다. PC와의 데이터 통신도 P3.5, P3.5 을 제어하는 시간동안은 데이터가 오지 않도록 하여야 합니다. 그러므로 필요한 데이터를 받아서 RAM에 저장하여두고 리이팅 하도록 구성하여야 합니다.

PC 인터페이스 회로:

PC와는 시리얼 인터페이스를 이용하며 RS-232 레벨 컨버터는 MAXIM(3)사의 MAX232 를 사용합니다. 통신속도는 19200 으로 고정되어 있으며 모든 데이터는 ASCII 데이터만 사용합니다. PC와는 TxD, RxD, GND를 사용합니다. PC 쪽의 사용하지 않는 나머지 핀들은 Null 모델 조건으로 합니다.

VPP 발생회로:

AT89C2051은 라이팅 시 12V 의 Vpp 전압이 필요합니다. 본 회로에서는 MAX232 의 10V 구형파를 다이오드 회로에 의하여 2배 승압하여 약 18V 가 나옵니다. 다시 78L12를 사용하여 12V의 정전압을 만들어 Vpp로 사용합니다. 약 3mA 의 전류를 사용할 수 있습니다.

PC 드라이버 프로그램:

Visual BASIC으로 드라이버 프로그램을 구성하였습니다.

드라이버를 설치하면 그림-9 와 같은 윈도우가 표시됩니다. 소스를 공개 하였으므로 사용자가 원하는 기능을 추가하거나 화면 디자인을 변경해 볼 수 있습니다.

LOAD: HEX 파일을 버퍼에 로드합니다. 이때 HEX 파일의 포맷은 INTEL 방식입니다. Binary 코드는 직접 읽을 수는 없으므로 Binary를 Intel HEX로 변경하는 유틸리티(HEX2BIN.EXE)를 이용하여 Intel HEX 포맷으로 변경한 후 Load하여야 합니다. LOAD 버튼에 FF 체크 박스가 체크되어 있으면 HEX파일을 로드하기 전에 버퍼의 모든 내용을 0xFF로 초기화 합니다

SAVE: 버퍼에 저장된 HEX 코드를 Intel HEX 포맷으로 저장합니다. 프로세서에 FLASH 메모리에 저장된 데이터를 읽어 PC 에 저장할 때 사용합니다.

RESET: SE-511P에 장착된 프로세서의 형식을 확인합니다. 확인 가능한 CPU는 ATMEL 사의 8051 프로세서중 DIP 20 핀 계열 AT89C1051, AT89C2051, AT89C4051 입니다.

READ: 프로세서의 FLASH 메모리에 기록된 데이터를 읽어 버퍼에 저장합니다.

ERASE: 프로세서의 FLASH 메모리를 소거(모든 내용을 FF)합니다. 라이팅 하기 전에 항상 FLASH 메모리를 이레이즈 하여야 합니다.

BLANK: SE-511P에 장착된 CPU FLASH 메모리의 내용이 전부 FF인지 확인합니다.

WRITE: 버퍼의 데이터를 FLASH 메모리에 라이팅 합니다.

VERIFY: FLASH 메모리와 BUFFER를 비교하여 동일한지 확인합니다. 1 개의 비트라도 틀리면 Verify ERROR 가 발생합니다.

LOCK: FLASH 메모리에 프로그램된 데이터를 읽지 못하도록 Lock 비트를 프로그램합니다

AUTO: 체크박스에 표시가 된 명령을 순차적으로 실행합니다. 예를 들어 그림에서 [ERASE]와 [WRITE]의 커맨드 버튼의 체크 박스에 체크 표시가 있으므로 [ERASE]실행 후 [WRITE]를 실행합니다. AUTO 버튼의 ReLoad 체크 박스는 AUTO 버튼이 실행되기 전 HEX 파일을 읽을 것을 선택합니다.

Com Port: PC 의 시리얼 포트를 선택할 수 있습니다. (Com1 또는 Com2)

다른 번호의 Com 포트를 원하면 Visual BASIC 으로 작성한 드라이버 프로그램 소스에서 채널 번호만 수정하면 됩니다.

*** 부품 목록 ***

	품목	사양	수량
1	AT89C2051	20 PIN DIP	1
2	MAX232CPE	16 PIN DIP RS-232 LINE DRIVER	1
3	MC7805	TO-220 5V / 1A REGULATOR	1
4	MC78L12	TO-92 12V / 100MA REGULATOR	1
5	1N4007	정류 다이오드	4
6	1N4148	스위칭 다이오드	3
7	RED LED	3 MM	1
8	GREEN LED	3 MM	1
9	YELLOW LED	3 MM	2
10	2SA1266	TO-92 PNP TRANSISTOR	1
11	2SC3198	TO-92 NPN TRANSISTOR	3
12	10uF/25V	전해 컨덴서	11
13	33P	세라믹 컨덴서	2
14	R 10K	1/4W REGISTOR 고동, 검정, 오렌지, 금	9
15	R 2K	1/4W REGISTOR 적, 검정, 적, 금	7
16	R 180 Ohm	1/4W REGISTOR 고동, 회색, 고동, 금	1
17	11.0592MHz	ATS	1
18	DB9S ANGLE	PCB 납땜형	1
19	10MM 서포터	플라스틱 볼트형	4
20	3MM 너트	니켈도금	5
21	3MM 와셔	프라스틱	1
22	3 x 5MM볼트		1
23	20P IC 소켓		2
24	TACT SW		1
25	Power Jack	아답터 잭	1
26	PCB	SE-511 양면 스루홀, 샘플전자 제작	1

본 기사에 관한 모든 자료는 샘플전자 홈페이지에서 다운 받을 수 있습니다.
(<http://www.SAMPLE.CO.KR>)

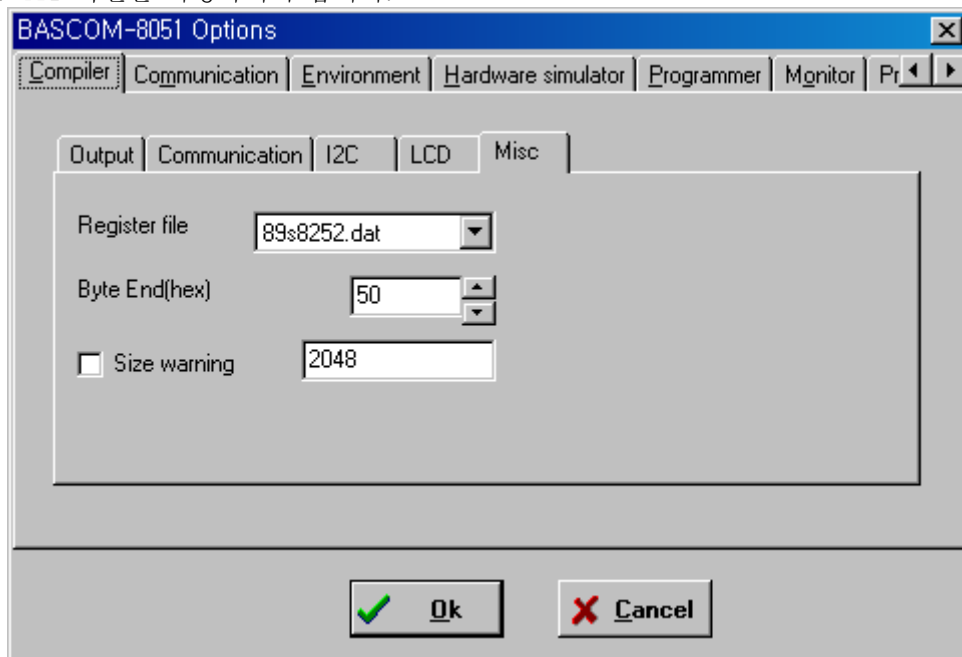
Appendix:

- (1)ATMEL (<http://www.atmel.com>)
- (2)MCS Electronics (<http://www.mcselec.com>)
- (3)MAXIM (<http://www.maxim-ic.com>)

자주 묻는 질문

Q1: 샘플전자 홈페이지에서 예제 소스 프로그램과 BASCOM 8051데모를 다운 받아 컴파일 하였는데 “ Error 8: Out of Internal memory file “ 에라 메시지가 발생하며 HEX 파일이 만들어지지 않습니다. 소스가 잘못되었나요?

소스 프로그램의 오류가 아닙니다. 내부 램 사용 한도 설정을 맞추어야 합니다. BASCOM 윈도우 메뉴에서 Options -> Compiler -> Misc 메뉴를 클릭하면 아래와 같은 윈도우가 표시됩니다. Byte END 의 수치를 큰 값으로 설정하면 에라가 발생하지 않습니다. Byte End란 컴파일러에게 내부 메모리 사용한계를 지정한 것입니다. AT89C51, AT89C1051, AT89C2051, AT89C4051은 7F [Hex]가 내부메모리 최대값이므로 7F 로 변경하면 됩니다. 8052 는 FF [Hex]가 최대입니다. 어셈블리 루틴을 사용하여 내부메모리를 직접 구동할 때에만 Byte End를 적절한 값으로 설정합니다. 그 외에는 프로세서가 허용하는 최대값으로 설정하여 둡니다. **Register File**에서도 사용하는 프로세서의 종류에 맞는 *.DAT 파일을 지정하여야 합니다.



Q2: SE-516SP 와 SP-511P를 BASCOM 8051에서 지원합니까?

아직 안됩니다. 다만 SE-516SP/SE-511P 드라이버 프로그램은 윈도우 버전이므로 BASCOM 8051 프로그램과 동시에 윈도우를 열어놓고 사용하시면 됩니다. SE-516SP/SE-511 드라이버 프로그램에서 한번만 HEX 파일을 읽어들이면 패스를 기억하고 있으므로 Auto 버튼 한번 클릭으로 프로그래밍 됩니다.

Q3: BASCOM 8051 에서 글로벌 , 로컬변수 구분은 어떻게 합니까?

변수는 사용하기전에 반드시 Dim 문으로 선언하여야 합니다. BASCOM 8051은 전부 글로벌 변수만 가능합니다.

Q4: i8255 와 같은 외부 IC 를 부착할 때 어드레스 지정을 어떻게 합니까?

BASCOM 8051 에서 변수선언시 XRAM 을 지정하면 됩니다. 다만 어드레스를 지정할 수는 없으므로 report 파일을 확인해가면서 정확한 어드레스가 어떻게 잡히는지 확인하여야 합니다. 여러 개의 IC 가 연결되는 경우 IC 어드레스가 연속적이지 않을 때가 있으므로 사용하지 않더라도 Dim XRAM Byte를 사용하여 원하는 어드레스에 일치하도록 하여야 합니다.

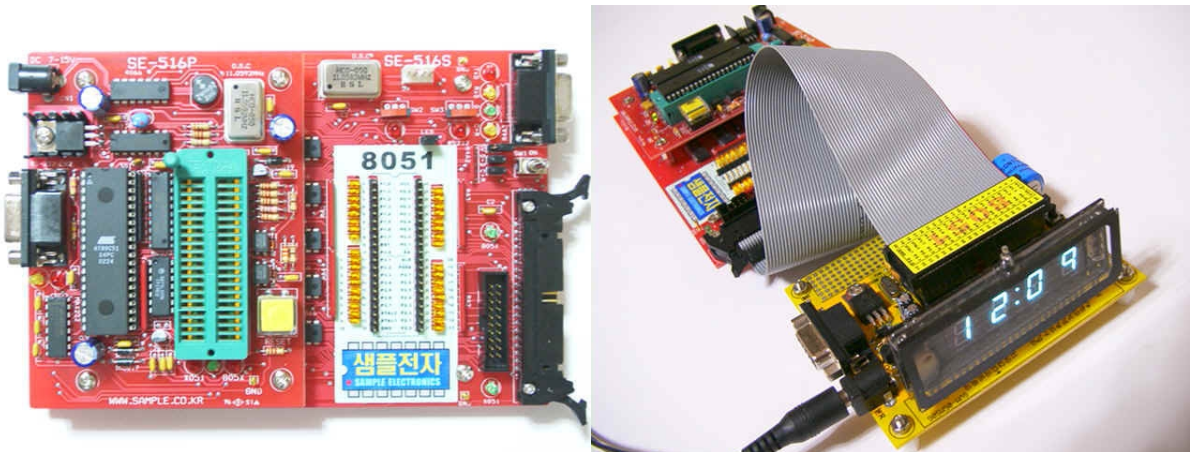
Q5: 부동 소수점 연산이 가능합니까?

BASCOM 8051은 단정도형 부동소수점 변수와 + - / * 의 4 칙 연산을 지원합니다. SINE COSINE 등의 삼각함수는 지원하지 않습니다.

Q6: 프로그램에서 ERR 비트 변수를 선언하지 않았는데 Report 파일에서 ERR 비트 변수가 있는데 작성한 프로그램의 문제가 있습니까?

아닙니다. Report 파일의 ERR 변수는 BASCOM 8051 내부에서 사용합니다. 사용자의 프로그램과는 관계가 없습니다.

SE-516SP Atmel 8051 Programmer / Simulator



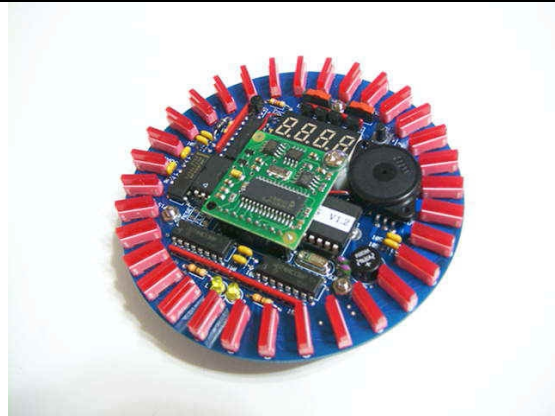
SE-516SP는 호스트 컴퓨터에서 컴파일/어셈블링된 출력파일(HEX 포맷)을 전기적으로 소거/프로그램 가능한 FLASH 메모리 내장형 8051프로세서에 라이팅한 후 타겟 회로가 즉시 동작되는 ATMEL 8051 프로세서 프로그래머/시뮬레이터 입니다. Universal 프로그래머와 EPROM 타입 8051프로세서(Intel사의 P8751)로 8051 제어 시스템을 개발할 경우 프로그램을 변경후 재 라이팅하기 위하여 프로그램 지울 때(ERASE) 30분 이상(EPROM타입) 기다려야 합니다. 또한 라이팅 후 CPU를 ROM라이터에서 뽑아 타겟 회로에 꽂아 주어야 합니다. 프로그램의 디버깅이 한 두 번의 소스 프로그램 수정으로 가능 하다면 이런 방법에 의한 개발도 가능 하지만 대부분의 경우 수십 번의 소스 프로그램 수정 및 이에 따른 ROM 라이팅과 ERASE 그리고 프로그래머와 타겟 회로로부터 CPU의 장착 / 탈착을 하여야 합니다. 이러한 과정은 대단히 번거로우며 8051 프로세서의 장착/탈착 과정시 8051 칩과 타겟회로 하드웨어가 파손되지 않도록 전원을 OFF하여야 하는 세심한 주의가 필요한 작업입니다. SE-516SP는 이러한 번거로움을 해결하기 위하여 개발되었습니다. SE-516SP에 타겟 회로에 사용될 같은 종류의 프로세서를 TEXTTOOL에 장착하고 타겟 회로에는 SE-516SP의 프루브 케이블을 연결합니다. 타겟 회로와 TEXTTOOL에 장착된 8051 프로세서와는 스위치 IC 인 4066 게이트에 의하여 연결되어 있습니다. 라이팅이 완료되면 4066의 트랜스미션 게이트가 ON 되어 타겟 회로와 TEXTTOOL의 8051을 연결하여 타겟 회로에 8051 프로세서가 있는 것과 동일하게 동작합니다. 그러므로 라이팅 완료 후 TEXTTOOL의 8051을 타겟 회로로 이동할 필요 없이 즉시 사용자 프로그램의 정상 동작 상황을 확인할 수 있습니다. 개발자는 소스 프로그램 수정과 라이팅을 무제한으로 반복할 수 있습니다. 최종 프로그램이 완성되면 프루브 케이블을 제거하고 TEXTTOOL의 프로세서를 타겟 회로로 이동하여 장착하면 개발이 완성됩니다. SE-516SP는 ATMEL사의 8051 코어가 내장된 FLASH 마이크로 컨트롤러 9 종류를 지원합니다. 20Pin 시리즈와 40Pin 시리즈가 겸용으로 사용될 수 있습니다. 시뮬레이션이 편리하도록 SE-516S에는 오실레이터와 시리얼 인터페이스 회로(RS-232 Driver)를 내장하고 있습니다. 타겟 실험 회로를 만들 때마다 오실레이터 회로와 시리얼 인터페이스 회로를 구성하지 않아도 되므로 SE-516S에 의하여 빠르고 쉬운 8051 시스템 개발이 가능합니다.

Model	FLASH	RAM	Package	
AT89C1051	1 KB	128 B	20 Dip	UART없음
AT89C2051	2 KB	128 B	20 Dip	
AT89C4051	4 KB	128 B	20 Dip	
AT89C51	4 KB	128 B	40 Dip	
AT89C52	8 KB	256 B	40 Dip	
AT89S8252	8 KB+2 KB	256 B	40 Dip	2K Eeprom
AT89C55WD	20 KB	256 B	40 Dip	
AT89S51	4 KB	128 B	40 Dip	
AT89S52	8 KB	256 B	40 Dip	

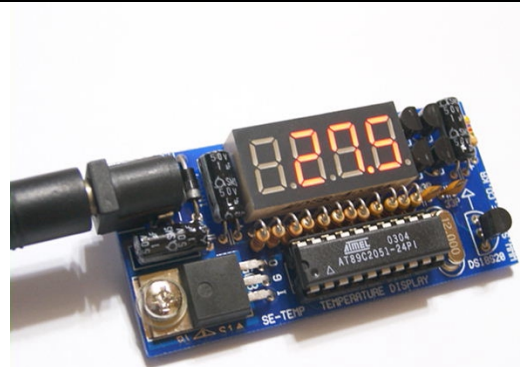
SE-516SP에서 지원하는 8051 프로세서

BASCOM 8051 Application-BASIC Study KIT

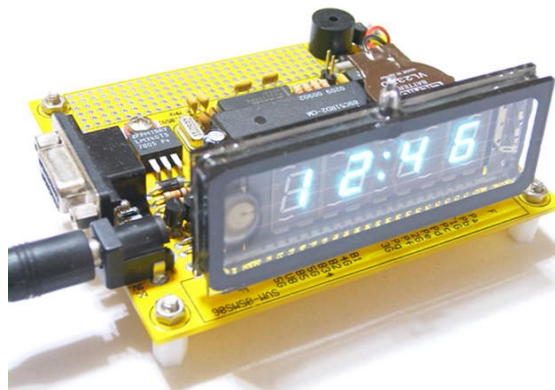
마이크로 프로세서 8051과 BASCOM 8051 을 이용한 키트 / 소스 프로그램 공개 / 회로도 공개



전자 나침반 키트: ROBOT Electronics 사의 나침반 모듈을 이용한 전자나침반 키트입니다. I2C 인터페이스를 이용하여 32개의 대형 LED 바에 방위를 표시합니다.



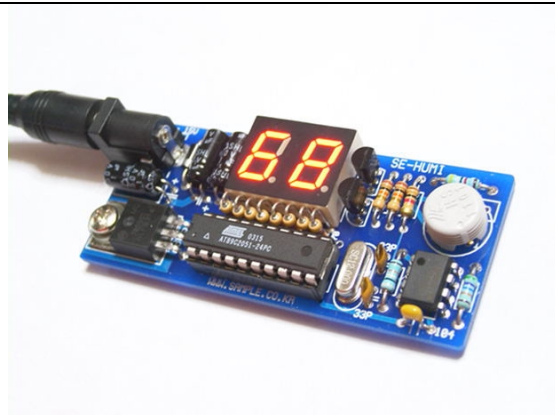
디지털 온도계 키트: MAXIM-사의 DS18S20 온도 센서 IC 를 이용한 온도계 키트 입니다. 1-Wire 통신을 이용하며 0.5 C 의 정밀한 온도를 표시합니다.



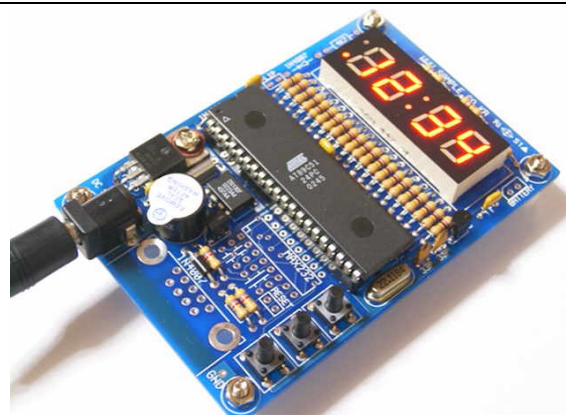
진공 형광관 디지털 시계 키트: MAXIM 사의 DS1307시계 IC 와 진공형관 표시관 (VFD)를 이용한 시계 키트입니다. 클래식한 감각의 시계입니다.



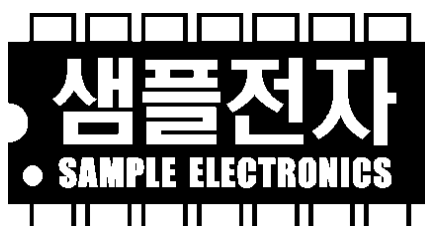
멜로디 크리스마스 트리 키트: 마이크로 프로세서를 처음으로 배우는 초보자에게 적합한 키트입니다. 피에조 를 사용하여 캐롤 송이 연주됩니다.



습도계 키트: HUMIREL사의 정전 용량형 습도센서를 이용한 상대 습도계 키트입니다. TI사의 555타이머 IC 를 사용하여 습도에 따른 주파수를 측정하여 상대 습도를 표시합니다.



FND 디지털 시계키트: 스테틱 방식으로 구성하여 초보자도 쉽게 프로그램을 이해할 수 있습니다. DS1307 을 이용하였습니다.



서울시 용산구 한강로 3가 16-9
전자랜드 본관 광장층 C-18호
TEL: (02) 701-8051
FAX: (02) 701-8058
sample@korea.com
WWW.SAMPLE.CO.KR

