80C52 With BASIC-52 INTERPRETER

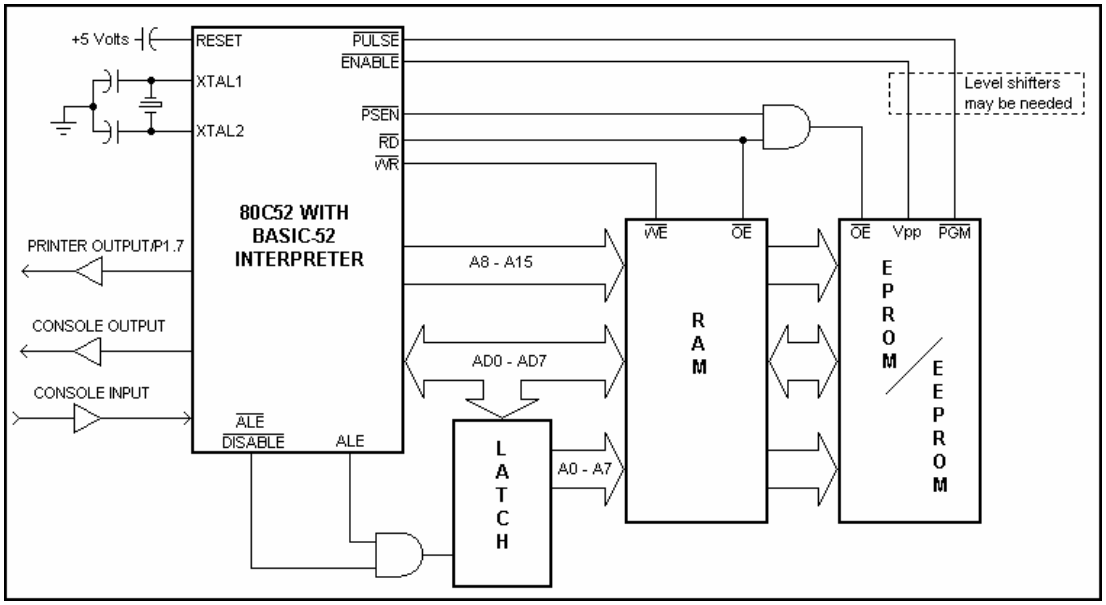
The 80C52-BASIC chip is a custom-masked 80C52 microcontroller with a full-featured 8k-byte ROM-resident BASIC-52 interpreter. The 80C52-BASIC chip is specifically designed to address the needs of process control, measurement, and instrumentation applications. The internal BASIC-52 language routines are easily executed as CALL routines from BASIC.

The fully static design of the 80C52-BASIC chip allows the user to reduce system power by reducing the clock frequency from 12 MHz down to any value, even DX, without loss of data or internal registers (typical operating frequency for BASIC-52 is 11.0592 MHz). In addition, the 80C52 has two software modes for reduced activity: Idle Mode, where the CPU is frozen but the serial port, timers, and interrupt system continue to function; and Power-Down Mode, where the internal RAM is saved but all other functions are ceased.

A minimum amount of hardware is required to support the 80C52-BASIC Interpreter. Small systems can be constructed with only an address latch, 1k byte of external memory, and appropriate serial port drivers. With the addition of a transistor, a gate, and a few passive components, BASIC-52 can program EPROM/EEPROMs directly. Both standard and fast programming algorithms are supported.

Software
- Full BASIC interpreter in ROM on a single chip
- BCD floating-point math
- Fast tokenized interpreter
- "Stand-alone" software development
- Interrupts can be handled by BASIC or Assembly Language
- Built-in real-time clock
- Generates all timing necessary to program EPROM and EEPROMs

Hardware
- Low-power fully CMOS static design
- Operates DC to 12 MHz
- Commercial or Industrial temperature range
- Jumper-selectable ROMless 80C31/80C32 operation
- Full-duplex serial port
- Three 16-bit timers
- 5-source 2-level interrupt structure
- On-chip oscillator and clock circuit
- 256 x 8-bit RAM/ROMs



BASIC-52 INTERPRETER

BASIC-52 is a very complete implimentation of the BASIC language contained in just 8k bytes of ROM. It provides a powerful tool which combines ease of development in a high-level language with speed necessary for the most time-critical process-control applications. Due to the low system overhead and 11-MHz system clock, BASIC-52 is extremly fast and efficient. It actually runs the "Sieve" benchmark program faster than an IBM PC. BASIC-52 offers many unique features. Including an accurate real-time clock, the ability to process interrupts, and the ability to treat EPROM memory as if it were mass storage. In addition, the I/O routines and arithmetic routines in BASIC are callable as assembly language subroutines.

BASIC-52 permits use of both integer and floating-point numbers. Integer numbers range from 0 to 65535, and floating-point numbers range from -1E-127 to 0.999999999E+127 with eight digits of significance. Numbers may be entered in integer, decimal, hexadecimal, or exponential format.

The following is a list of commands, statements, and operators supported by the BASIC-52 Interpreter. Although some are unique to BASIC-52, just a cursory inspection reveals that the full power of structured programming in BASIC for process-control applications is available.

Command Function

RUN - Execute a program
CONT - Continue after a stop or Control-C
LIST - List program to the console device
LIST# - List program to serial printer port (P1.7)
NEW - Erase the program stored in RAM
NULL - Set null count after carriage return/line feed
RAM - Evoke RAM mode, current program in read/write memory
ROM - Evoke ROM mode, current program in ROM/EPROM
XFER - Transfer a program from ROM/EPROM to RAM
PROG – Saves the currently selected program to EPROM
PROG1 – Saves baud rate and sends sign-on message on reset
PROG2 – Saves baud rate and enters run mode upon power up or reset
PROG3 – Saves the baud rate and MTOP clears memory up to MTOP and sends the sign-on message
PROG4 - Saves the baud rate and MTOP clears memory up to MTOP and enters run mode upon power up or reset

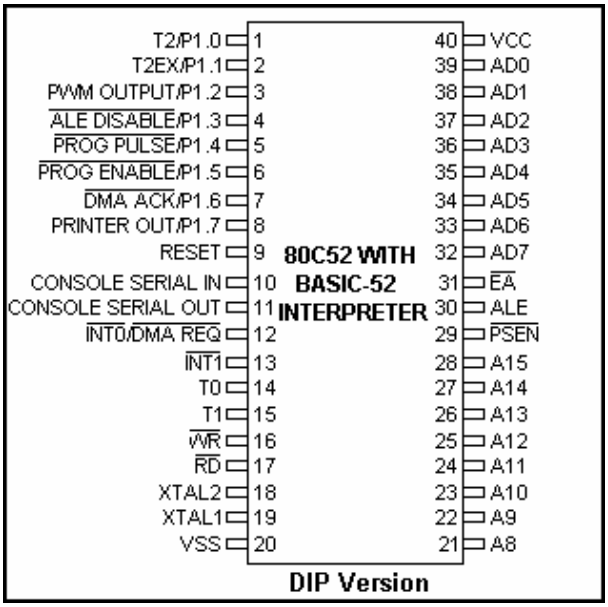
Statement Function

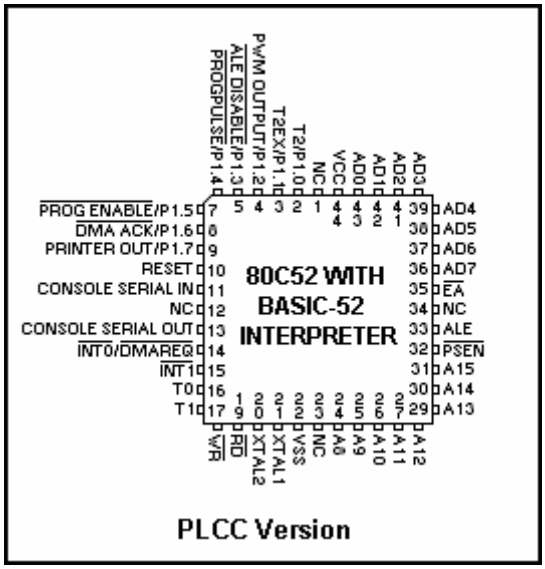
ASC() - Returns integer of ASCII character
BAUD - Set data-transmission rate for line-printer port
CALL - Call assembly-language program
CHR() - Returns ASCII character of integer
CLEAR - Clear variables, interrupts, and strings
CLEARS - Clear stacks
CLEARI - Clear interrupts
CLOCK1 - Enable real-time clock
CLOCK0 - Disable real-time clock
DATA - Data to be read by READ statement
READ - Read data in DATA statement
RESTORE - Restore READ pointer
DIM - Allocate memory for arrayed variables
DO - Set up loop for WHILE or UNTIL
UNTIL - Test DO loop condition (loop if false)
WHILE - Test DO loop condition (loop if true)
END - Terminate program execution
FOR-TO-{STEP} - Set up FOR...NEXT loop
NEXT - Test FOR...NEXT loop condition
GOSUB - Execute subroutine
RETURN - Return from subroutine
GOTO - GOTO program line number
ON GOTO - Conditional GOTO
ON GOSUB - Conditional GOSUB
IF-THEN-{ELSE} - Conditional test
INPUT - Input a string or variable
LET - Assign a variable or string a value (LET is optional)
ONERR - ONERR or GOTO line number
ONTIME - Generate an interrupt when time is equal to or greater than ONTIME argument; line number is after comma
ONEX1 - GOSUB to line number following ONEX1/ when INT1 pin is pulled low
PRINT - Print variables, strings, or literals, P. is shorthand for print
PRINT# - Print to serial printer port (P1.7)
PH0. - Print hexadecimal mode with zero suppression
PH1. - Print hexadecimal mode with no zero suppression
PH0.# - PH0.# to serial printer port (P1.7)
PH1.# - PH1.# to serial printer port (P1.7)
PUSH - Push expressions on argument stack
POP - Pop argument stack to variables
PWM - Pulse-width modulation
REM - Remark
RETI - Return from interrupt
STOP - Break program execution
STRING - Allocate memory for strings
UI1 - Evoke user console input routine
UI0 - Evoke BASIC console input routine
UO1 - Evoke user console output routine
UO0 - Evoke BASIC console output routine

Operator Function
CBY( ) - Read program memory
DBY( ) - Read/assign internal data memory
XBY( ) - Read/assign external data memory
GET - Read console
IE - Read/assign IE register
IP - Read/assign IP register
PORT1 - Read/assign I/O port 1 (P1)
PCON - Read/assign PCON register
RCAP2 - Read/assign RCAP2 (RCAP2H:RCAP2L)
T2CON - Read/assign T2CON register
TCON - Read/assign TCON register
TMOD - Read/assign TMOD register
TIME - Read/assign real-time clock
TIMER0 - Read/assign TIMER0 (TH0:TL0)
TIMER1 - Read/assign TIMER1 (TH1:TL1)
TIMER2 - Read/assign TIMER2 (TH2:TL2)
+ - Addition
/ - Division
** - Exponentiation
* - Multiplication
- - Subtraction
.AND. - Logical AND
.OR. - Logical OR
.XOR. - Logical exclusive OR

Stored Constant
PI - 3.1415926

Operators-Single Operand
ABS( ) - Absolute value
NOT( ) - One's complement
INT( ) - Integer
SGN( ) - Sign
SQR( ) - Square root
RND - Random number
LOG( ) - Natural log
EXP( ) - "e" (2.7182818) to the X
SIN( ) - Returns the sine of argument
COS( ) - Returns the cosine of argument
TAN( ) - Returns the tangent of argument
ATN( ) - Returns the arctangent of argument





Operating Conditions*
Operating Temperature:
Commercial 0°C to 70°C
Industrial -40°C to 85°C

Operating voltage (Vcc) : +5Volts ± 10%

Absolute Maximum Ratings*
Voltage on any pin with respect to ground (Vss): -0.5V to 7.0V

Power dissipation: 200 mW

Maximum Icc at 12 MHz : 24 mA

80C52 Pin Descriptions

Vss - Circuit Ground Potential

Vcc - Circuit Supply Voltage

AD0 - AD7 - The multiplexed low-order address and data bus used during access to external memory. External pull-up resistors (10kΩ) are required on these pins if BASIC-52 EPROM/EEPROM programming feature is used.

A8 - A15 - The high order address bus used during access to external memory.

PORT1 - Port 1 is a quasi-bidirectional 8-bit input/output port. It can be used as a standard parallel I/O port with the PORT1 command in BASIC-52, or the individual pins of Port 1 can have alternative functions as follows.
PORT1.0(T2) - Can be used to trigger input to Timer/Counter #2. A logic 1 must be written to this bit in order for this function to operate.
PORT1.1(T2EX) - Can be used as the external input to Timer /Counter #2. A logic 1 must be written to this bit in order for this function to operate.
PORT1.2(PWM) - This pin is used as the Pulse Width Modulated (PWM) output port when the PWM statement is executed. The PWM statement can generate pulses of varying frequency and duty cycle.

PORT1.3(ALE DISABLE) - This pin is used to disable the ALE signal to the external latch when the EPROM/EEPROM programming feature is being used. In a system, this pin is logically ANDed with ALE.
PORT1.4(PROGRAMMING PULSE) - This pin provides the proper programming pulse when programming EPROM/EEPROMs.
PORT1.5(PROGRAMMING ENABLE) - This pin is used to enable the programming voltage (Vpp) when programming EPROMs and remains active low during programming. On EEPROMs that do not require any programming voltage, this pin is not used.
PORT1.6(DMA ACKNOWLEDGE) - When the pseudo-DMA feature is implemented (as outlined in the BASIC-52 Programmer's Manual), this pin functions as an active-low DMA Acknowledge output.
PORT1.7(LINE PRINTER OUTPUT) - This pin functions as a serial output when the LIST# and the PRINT# commands are used in BASIC. This enables the user to have a hard-copy output during program operation or for program listings.

RESET - A logic 1 (>3.5V) on this pin for more than two machine cycles while the oscillator is running will reste the device. An internal pull-down resistor permits power-on reset using only a capacitor connected betweetn this pin and Vcc.

ALE - (Address Latch Enable) an output pin that is used to latch the low-order address byte during read, write, or program fetch operations to external memory.

PSEN - (Program Store Enable) a signal used to enable external program memory. This pin will remain a logic 1 unless the user is running an assembly language program in external memory.

XTAL1 - Input to the inverting amplifier that forms the oscillator. This input should be left floating when an external oscillator is used.

XTAL2 - Output of the inverting amplifier that forms the oscillator and input to the internal clock generator. Receives the external oscillator signal when an external oscillator is used.

RD - This pin is a control that is used to enable read operations to external data memory.

WR - This pin is a control signal that is used to enable write operations to external data memory.

T1 - This pin can be programmed to be an external input to Timer/Counter #1.

T0 - This pin can be programmed to be an external input to Timer/Counter #0.

INT1 - This is the external interrupt 1 input pin. Interrupts on this pin may be handled in either BASIC-52 or assembly language.

INT0/DMA REQUEST - This is the external interrupt 0 input pin. It may optionally be programmed to function as a DMA request input pin or used by EEPROM devices during programming.

CONSOLE SERIAL OUTPUT - This is the serial output pin that transmits data from the console device. Standard serial ASCII codes consisting of 8-bit data with no parity at standard data rates are assumed.

CONSOLE SERIAL INPUT - This is the serial input pin that receives data from the console device. Standard serial ASCII codes consisting of 8-bit data with no parity at standard data rates are assumed. After RESET in BASIC-52, if desired and if the first character received is a "space", then BASIC-52 will perform an auto-baudrate calculation and automatically set the console serial input to the incoming data rate.

EA - When EA is held high, the CPU functions as an 80C52 with BASIC interpreter executing out of internal memory. (unless the program counter exceeds 0FFFH). When EA is held low, the CPU functions as a generic 80C32 microcontroller chip.

Micromint, Inc Products Using the 80C52 BASIC-52 Interpreter

Domino 1
Domino 2
BCC52
BCC52CX
RTC52
RTC52Plus