

Great Cow BASIC Command Reference and General Information

Last updated: 14/1/2007

Legend to highlighting:

Tested and working

Tested and not working

Not tested

Not implemented

Supported chips:

In theory, all 10/12/14/16/18 series PIC chips. See chipdata.csv for a complete list. Some features may not work on all chips – if you find that this is the case, please do not hesitate to email hconsidine@bigpond.com with the name of the chip and the feature that is causing difficulty.

Commands:

Note: All standard PIC assembly mnemonics can be inserted directly in to the GCBASIC program – anything not recognized by the compiler will be passed straight through to the assembly.

Command	Explanation
Flow Control	
IF condition THEN ... END IF	Execute commands between IF and END IF, if condition is true.
IF condition THEN command	Short form of IF. Does not require END IF. Command must be on same line as IF.
GOTO label	Jump to label
label:	Defines a label
GOSUB label	Call label as a subroutine
RETURN	Return from a GOSUB
SubName [(vars)][#NR]	Call up sub SubName, and pass vars to it. If vars are specified, then brackets must be included. #NR stands for No Return, and will stop the sub from returning values of variables.
SUB Subname [(vars)]	Used to mark the start of a subroutine
EXIT SUB	Exit from a subroutine (Only found in subs) (interchangeable with EXIT FUNCTION)
END SUB	Marks the end of a subroutine (interchangeable with END FUNCTION)
EXIT FUNCTION	Exit from a function (Only found in functions) (interchangeable with EXIT SUB)
FUNCTION functionname [(vars)] ... END FUNCTION	Same as for sub, except a value can be returned automatically ie: Var = FunctionName (var2) (Note: When defining a function, be sure to return a value. To do this, set FunctionName to some value/var.)

FOR var = min TO max STEP step ... NEXT	Loop (max-min) times. Var will start at min and be incremented every loop until it is equal to max. During this time, all code up to NEXT will run. Step is number to add to var each loop. Step can be a negative.
DO WHILE UNTIL condition ... LOOP	Repeat code while/until condition is true
REPEAT count ... END REPEAT	Repeat code count times
SELECT CASE var CASE value1 ... CASE value2 ... CASE ELSE ... END SELECT	Run a different section of code, based on the value of var. The CASE ELSE section runs if none of the others do. Note that CASE ELSE is optional.
CALL sub EVERY ticks	Call up sub every ticks of the interrupt. Set ticks to 0 to stop call for a particular sub. (Interrupt must be defined)

Variable	
[LET] var1=var2	Set var1 to var2. var2 can be given as a function. LET is optional, and exists solely for people familiar with other BASIC dialects. Appending [WORD] to the line will force the PIC to use 16-bit mode when performing the calculation.
ROTATE var {LEFT RIGHT} [SIMPLE]	Rotate var 1 bit left/right
SET var.bit, 0 1	Set bit of var to 0 or 1 (OFF or ON accepted)
DIM ArrayName(size)	Initialize an array.
DIM VarName AS type	Declare a non-byte variable. Only "word" is a valid type at present.

Compiler (*Commands in main file will override those in includes, and commands in includes will override those in nested includes*)

#include "file.h"	Include the subs in file.h into the program <i>Note: using "" will read the file from the same directory as the program. If <> are used, the file will be read from the default include folder.</i>
#define CONST value #define CONST =sum	Create a constant called CONST, and set it to value, or result of sum.
#chip MODEL, MHz	Specifies model and speed of PIC
#osc Type	Type is HS/LP/etc.
#config settinglist	Settinglist is list of config values in .inc file (WDT_OFF, etc.)
#IFDEF constant value[,value2,etc.] ... #ENDIF	If constant != value/value2/etc, then the code between value can also be another constant #IFDEF and #ENDIF will be removed from the compiled code.
#SCRIPT ... #ENDSCRIPT	Used to make a script which GCBASIC runs. Use for creating code that can be run on different PIC chips. See the script commands section for more info
#int frequency	Set up an interrupt of the given frequency. Used in conjunction with the CALL EVERY command. Frequency is Hz (app. Range on 20 MHz: 76-19,000).
#startup subname	Used in include files to define a subroutine that must be called if anything from that file is used.
#RAM ramsize #mem ramsize	Specifies the amount of RAM present on the PIC chip. Not compulsory, but should be used if arrays are in use.

RS 232 Commands (lowlevel\rs232.h)

InitSer (channel, rate, start, data, stop, parity, invert)	Channel = 1,2,3 Rate = r300, r600, r1200, r2400, r4800, r9600, r19200 Start = start bits (usually 1), add +WaitForStart to make the program wait for the start bit when receiving. Data = data bits (usually 8) Stop = stop bits (usually 1) Parity = none, odd, even Invert = normal, invert
SerSend (channel, data)	Data is byte to send using channel
SerReceive (channel, variable)	Received byte from channel will be written to variable
SerPrint (channel, String \$)	Send String\$ over given channel

A/D Commands (lowlevel\ad.h)

ReadAD (port)	Initializes, reads the analog value of port, and restores ports to digital. Port is AN0-ANx, depending on what the selected PIC will allow.
ADFormat (Format_Left Format_Right)	Left ignores highest bits in ADRESH. Right ignores lowest in ADRESL.
ADOff	Disables A/D, and sets all ports to digital

SPI/I2C Commands (lowlevel\ssp.h)

SPIMode (Mode)	Sets the mode of the SPI module. Mode can be MasterFast, Master, MasterSlow, SlaveSS or Slave.
SPITransfer (send, receive)	Sends and receives the given data. If the SPI module is in slave mode, this command will pause the program until data is requested by the master.

EEPROM Commands (*lowlevel\EEPROM.h*)

EPRead (address, var)	Read value from EEPROM. Address is location, var is var to place data in.
EPWrite (address, data)	Write value to EEPROM. Address gives location, data is byte to write.
ProgramWrite (Address, Data)	Write the 14-bit value given by Data to the program memory location given by Address. (<i>Only supported by some chips</i>)
ProgramRead (Address H, Data)	Read the 14-bit value given by Data from the program memory location given by Address. (<i>Only supported by some chips</i>)
ProgramErase (Address)	Erase the 32 bytes of program memory starting at the given address. This must be done before writing to a location. For more details, please see the relevant manual for your PIC chip.

Hardware PWM Commands (*lowlevel\pwm.h*)

PWMOn	Enables PWM by switching the CCP module to PWM mode
PWMOFF	Disables PWM by switching the CCP module off.
HPWM Channel, frequency, duty cycle	Turns on hardware PWM. Channel is 1 for CCP1 and 2 for CCP2 (on chips so equipped). Frequency is measured in KHz, and duty cycle is 0 – 255.

SRF04 Distance Sensor Commands (*srf04.h*)

USDistance (port no.)	Function to read a distance from an SRF-04 distance sensor. The sensor must have been defined correctly – see the include file for more information.
-----------------------	--

Sound Commands (*lowlevel\sound.h*)

Tone (Frequency, Duration)	Frequency is Hz; Duration is in 10 ms units.

LCD Display Commands (*lowlevel\lcd.h*)

PRINT String\$	Write the string to the LCD, starting at the current cursor location.
LOCATE column, line	Set the cursor to the given position
PUT (location, ASCII char)	Write the specified ASCII character to the given location
x = GET (location)	Return the ASCII character at the given position
CLS	Clear the LCD
LCDInt (Value)	Write the given value to the LCD, at the current cursor location.
LCDHex (Value)	Write the given value to the LCD, at the current cursor location. The value will be shown in hexadecimal format.
LCDWord (Value)	Write a 16-bit value to the LCD at the current cursor position. Similar to LCDInt, but can display up to 65535 rather than 255.

7-Segment Display Commands (<i>lowlevel\7segment.h</i>)	
DisplayValue (Display Select, Value)	Display Select is 1, 2, 3 or 4, corresponding to the display that is to be output to. Value is the value between 0 and 9 to be shown on the screen.
DisplayChar (Display Select, ASCII Code)	Display Select is 1, 2, 3 or 4, corresponding to the display that is to be output to. ASCII Code is the character code to be shown. Note: This can be given as a single character, enclosed in quotes)

Keypad Commands (<i>lowlevel\keypad.h</i>)	
Var = KeypadRaw	Returns a 16-bit value, in which each bit corresponds to a key (1 = pressed, 0 = released).
Var = KeypadData	Returns 0-9, KEY_A, KEY_B, KEY_C, KEY_D, KEY_HASH or KEY_STAR, depending on which key is pressed. If no key is pressed, it returns 255.

String handling Functions	
Var = LEN(String\$)	Returns the length of the specified string.
OutString\$ = MID\$ (InString\$, position, length)	Returns the given portion of InString

Timer Functions (<i>lowlevel\timer.h</i>)	
InitTimer0 (Source, Prescaler)	Initializes Timer0. Source is Osc or Ext, Prescaler is PS0_1/2, PS0_1/4, PS0_1/8, ..., PS0_1/256
InitTimer1 (Source, Prescaler)	Initializes Timer1. Source is Osc, Ext or ExtOsc, Prescaler is PS1_1/1, PS1_1/2, PS1_1/4 or PS1_1/8.
InitTimer2 (Prescaler)	
ClearTimer (TimerNo)	Clears selected timer. TimerNo is the number of the timer to clear.
StartTimer (TimerNo)	Starts selected timer. TimerNo is the timer to start. (Note: Timer0 cannot be stopped or started)
StopTimer (TimerNo)	Stops selected timer.

Misc. Commands/Functions	
WAIT length units	Length 0-255, units = us, 10us, ms, 10ms, s, m, hour
WAIT WHILE UNTIL condition	Wait until condition is true
SLEEP time	Equivalent to Wait time s. Included for compatibility with other BASIC dialects
DIR PORTBIT IN OUT	PORTBIT is PORTA.0, PORTA.1, etc. IN/OUT is the direction. On 10/12 series chips, use GPIO.0, etc.
DIR PORT DirByte	PORT is A, B, etc. DirByte is a byte specifying the value for the applicable TRIS register (Can be a var or constant)
PULSEOUT pin, time units	Set pin high for time.
POT pin, output var	Measure the period of an R/C oscillator connected to pin, then write result to output var. <i>This command should be used sparingly, as each use results in 20 assembly instructions. If the same pin is to be read twice, use a function with this command in it.</i>
PWMOut channel, duty, cycles.	Output a software PWM pulse on the given channel. (Channels are defined by setting SoftPWM1, SoftPWM2, etc.) Duty is the duty cycle, and ranges from 0 to 255. Cycles is the number of PWM cycles to output – on a 20 MHz chip, 1 cycle = 28 clock cycles + 10 clock cycles per channel.
` ; REM	Used as comments.
Var = PEEK (location)	Set Var to the contents of the given memory location
POKE (location, value)	Set the given RAM location to the given value
Var = Random	Returns a pseudo-random integer between 0 and 255
Swap (Var1, Var2)	Swaps the values of Var1 and Var2
Swap4(Var)	Swaps the nibbles in Var
ReadTable TableName, TableIndex, OutputVar	Reads the value from location TableIndex of TableName, and store it in OutputVar

Compiler script commands:

Command	Explanation
IF condition THEN ... END IF	Execute the script within if condition is true. Note that constants can be used as vars, and vars must not be used. Otherwise, this is similar to the normal IF.
ERROR message	Add message to the error listing
define = sum	Set the compiler constant to the value of sum
<p><i>Note: These commands are executed by GCBASIC, and are useful for setting constants to a particular value inside include files. They are not downloaded to the PIC.</i></p> <p><i>Constants are treated as variables are normally treated –they can be set. Constants must have been defined before they are used in an IF, either by setting them, or using #define.</i></p>	

Functions:

Function	Explanation
x * y, x / y, x + y, x - y	Self explanatory
x AND y	Logical AND of x and y. Spaces must be included
x OR y	Logical OR of x and y. Spaces must be included

x XOR y	Logical XOR of x and y. Spaces must be included
NOT x	Inverts x. Space must be included
x & y, x y, x # y, ! x	Short form of AND, OR, XOR, and NOT respectively. Spaces are optional with the short form.
x = y	Returns 0 if not equal, 255 if equal
x <> y	Returns 255 if not equal, 0 if equal
x > y	Returns 255 if x is more than y, 0 otherwise
x < y	Returns 255 if x is less than y, 0 otherwise
x >= y	Returns 255 if x is equal to or more than y, otherwise 0.
x <= y	Returns 255 if x is less than or equal to y, otherwise 0.

Short Forms:

x += y is equivalent to x = x + y

x -= y is equivalent to x = x - y

Functions must have “var =” before them (minus quotes), where var is the name of the variable that is to be set.

Order of operations:

- ()
- +, -
- AND, OR, XOR, NOT

Conditions

- = (equal)
- <> or ~ (not equal)
- < (less than)
- > (greater than)
- <= (less than or equal)
- >= (greater than or equal)

Conditions are basically sums, with either zero (false) or non-zero (true) results.

Built-in #defines (used for #IFDEF)

- ChipMHz
- ChipName
- ChipFamily (returns 12, 14 or 16, depending on instruction width)
- OSC
- Var()
- NoVar()
- Bit()
- NoBit()
- AllOf(define1, define2, ...)
- OneOf(define1, define2, ...)

Var(), NoVar(), Bit() and NoBit() are functions that are built in to #IFDEF. They will return true if a variable or bit is declared/not declared in the currently selected PIC chip.

AllOf and OneOf will return true if all of or one of the listed defines is declared.

Built-in #defines (used for IFs, WAITs, etc.)

Name	Value
ON	1
OFF	0
TRUE	255 (or 1 for bit variables)
FALSE	0

NB. Other #defines are set by the built-in include files.

Time Units

Units	GCBASIC units
Microseconds	us
Microseconds * 10	10us
Milliseconds	ms
Milliseconds * 10	10ms
Seconds	s
Minutes	m
Hours	h

Data Types:

Type	Size	Range
Bit	1 bit	0 to 1
Byte	8 bits	0 to 255
Word	16 bits	0 to 65535
Float	24 bits	?

Assembler/Programmer parameters:

These options can be used after the /A: and /P: switches to control the assembler and programming software:

Symbol	Replaced with
%Filename%	The original filename of the BASIC program
%Chipname%	The chip model

Data Tables:

Data tables can be used to store a list of up to 255 numbers in the program memory of the chip, in a way that allows them to be recalled. The format is as follows:

```
Table [Name of Table]
[Value 1]
[Value 2]
[Value 3]
End Table
```

Binary, hexadecimal and decimal numbers can be used in the table.

Chip Configuration:

The following are the default settings used by GCBASIC for chip configuration:

- Oscillator = HS, XT or INTOSC (see help for more detail)
- MCLR = Off
- WDT = Off
- LVP = Off

For 18x chips, type in the config exactly as you would type it in to MPASM. For example:

```
#config OSC = HS_PLL, WDT = ON
```

For 10x/12x/16x chips, you may use either the 18x format as shown above, or the default 10x/12x/16x format as used by MPASM:

```
#config LP_OSC, MCLRE_ON
```

Random Notes:

- When dealing with system registers, the variable name may be omitted and only the bit specified. For example, SET ADCON0.ADON ON is identical to SET ADON ON.
- GCBASIC is not case sensitive – for example, Wait, WAIT, wait and Wait would all be treated identically.

Known Limitations:

- Chip Speed: 0.1-60 Mhz
- Lines of GCBASIC code (including subs): 20000
- Assembly Program: 20000 lines
- Include files: 25 (includes built-in include files)
- Constants: 400
- Variables: approx. 3800 or available RAM on PIC (whichever is less)
- Subs (including in include files): 500
- Array size: 80 bytes on 10/12/16, 255 bytes on 18 (Avoid large arrays wherever possible)
- String length: 20 characters default, 80 if declared using DIM
- Strings: 500

Things which MUST be defined in EVERY program:

```
#chip model, MHz
```

(These can be defined in an include file instead. If something is defined in an include file and the main program, the definition in the main program will be used.)