MCS BASIC–52 V1.3

rejuvenating a popular interpreter

By H.-J. Böhling and D. Wulf

The MCS BASIC–52 interpreter, developed by Intel, is still very popular, but in its 15th year it needs a thoroughgoing facelift.

The MCS BASIC–52 V1.0 interpreter was specially developed by Intel in 1985 for the 8052-AH microcontroller. With its code size of only 8 kB, it fitted into the internal ROM storage of this controller. Unfortunately, the 8052-AH has gone out of production. Nonetheless, the interpreter still enjoys considerable popularity in its final 'official' version (1.1), in spite of a few errors. Nowadays, the 8052 family, with its many derivative versions, has become the most widely used microcontroller family. Regrettably, MCS BASIC 1.1 will not run directly on the fastest members of this family. The authors have made numerous modifications to the source code to arrive at version 1.3, with the following results:

– The interpreter no longer has any known errors.
– BASIC–52 V1.3 can program EEPROMs, even if they are located in external memory.
– The new command 'ERASE' erases an EEPROM.
– The maximum value for 'XTAL' can now be set as high as 78 MHz, so the interpreter can even run on a Dallas 80C320 at 33 MHz!
– A new reset routine recognises the baud rate and various types of controllers. This means that BASIC–52 V1.3 can run on many 8052-family members (80616, 517(A), 528, 535, 537, 575 etc.).
– The new OPCODE 43h reads a value from the argument stack into a variable.
– The size of the interpreter is still only 8 kB, which means that it will fit into an 87C52 with 8 kB of internal EPROM.

Intel has now released the interpreter as freeware. It can thus be legally obtained via the Internet as commented source code. In spite of the comments, the source code is not

easy to read, so the authors were forced to use a simulator to unravel the functions of the interpreter. However, this need not concern us any further here.

Tailor-made improvements

The authors' work started with an extensive Internet search for proposed improvements and error descriptions related to MCS BASIC–52. Unfortunately, the available source code was not relocatable, which meant that its instructions could not be freely moved around. The insertion of even a few extra bytes of code immediately resulted in incorrect operation. This was primarily due to the fact that the source code was contained in two separate files, one for the interpreter itself and another for floating-point calculations. Consequently, the first task was to merge the two files into a single, common file. Labels were also added to all unlabelled absolute and relative jumps and calls. Some parts of the source code were coded as literal data, because the assembler used at the time did not support the new 8052 instructions. The literal data were translated into equivalent mnemonics. One routine in the source code differed from the original Intel ROM code, so it was replaced by the mnemonics corresponding to the ROM code. On completion of these modifications, the code was relocatable, so a number of ORG directives, in particular those linking the interpreter and floating-point code, could be eliminated. However, the primary benefit was that memory space gained by improvements to the code could now be used for further error corrections (see the October 1991 issue of *Elektor Electronics*). Still, the amount of recovered space was insufficient to allow all error corrections and improvements to be fitted into the code. Consequently, the 'ego' message of the Intel programmer J. Katausky had to be deleted (sorry!). Bit 40h in the internal bit-addressable memory, which was made free as a result, was used to detect a multiplication underrun. Previously, this had not been handled properly.

As early as 1993, D. Karmann rewrote the interpreter as version 1.2 for an 8031 controller, and in the process he eliminated two known errors, so that:

– a BASIC Autostart EPROM could work together with command extensions, and
– variable names with 'F' could be used without any restrictions.

Dan Karmann was kind enough to provide the authors with the source code for his improvements.

Running BASIC–52 V1.3 on Elektor Electronics boards

The hardware requirements for running BASIC–52 on the 80C537 or '537-Lite' boards are fully described in the article 'Basic-537', which appeared in the February 2000 issue of *Elektor Electronics*. The information in that article is equally applicable to V1.3, with the following exceptions:

– It is no longer necessary to manually set the value of MTOP, since V1.3 automatically detects the amount of available RAM.
– BASIC–52 V1.3 automatically recognises the baud rate being used, so a space character must be sent to the board after a restart!

The 80C32 Control Computer described in the February and March 1998 issues of *Elektor Electronics* can be used with BASIC–52 V1.3, including using a Dallas 80C320 'Speed it µP'. The Dallas controller is software and hardware compatible with the 8052. However, the processor design is completely new, and it takes only four clock cycles to execute an instruction instead of twelve. It can be run at clock frequencies up to 33 MHz. To revise the board for this controller, you can replace the quartz crystal with a 22.118400 MHz crystal, and the resulting processing speed will be approximately six times greater than that of a standard 80C32. The controller can handle 33 MHz, but this is not feasible here, in part due to the lack of a sufficiently fast EPROM (which would have to have an access time of around 40 ns) and in part because a suitable crystal is not available. For technical reasons, fundamental-mode crystals are only made for frequencies up to around 22 MHz. Above this frequency, overtone crystals are used, with a supplementary resonant network to force the crystal to oscillate at a harmonic frequency. Even at 22 MHz, sufficiently fast program memory (70 ns) must be used for the interpreter. The data memory can be populated with devices a having a standard access time of 150 ns, since the controller automatically inserts wait states when accessing data memory. The two TTL ICs (74HC573 and 74HC00) should be replaced by fast 74AC or 74F types. A 28C64 can be used for the EEPROM (IC5 on the circuit board). However, you must first cut the connections between +5 V and pins 1 and 27 of the socket for the EEPROM. Pin 1 may be left open, but pin 27 must be connected to the /WR signal from the controller (pin 27 of IC3).

Terminal MCS–51

If you order BASIC–52 V1.3 from the Publishers' Readers Services on diskette (order number **000121-11**), you will also receive a full edition of the latest version of Terminal MCS–51. This is a terminal emulator program for DOS computers, which naturally can also be run in a DOS window under Windows. Terminal MCS–51 was written to compensate for the known weaknesses of BASIC–52. For example, BASIC–52 does not have a RENUMBER command for renumbering BASIC command lines, and it has only very limited functions for editing BASIC programs. The functions provided by Terminal MCS–51 include:

– easy loading and storing of BASIC–52 programs,
– renumbering BASIC–52 programs,
– integrated line editor for BASIC–52 command lines,
– any desired ASCII editor can be linked in.

A shareware version of Terminal MCS–51 can also be downloaded from the Free Downloads section on the *Elektor Electronics* website at http://www.elektor-electronics.co.uk. The shareware version is fully functional and not time-limited, but it does not allow BASIC program lines to be renumbered.

```
4   REM ***************************************************
5   REM * Program for controlling A/D converter in 80535/537 *
6   REM ***************************************************
10  RANGE=5 :                  REM Controller reference voltage
20  ADCO=0D8H :                REM Register to check A/D state etc.
30  ADDAT=0D9H :               REM Register for measurement output
40  DAPR=0DAH :                REM Register for meas. range and start
50  RDSFR (ADCO)BUSY :         REM Read other function bits
60  BUSY=BUSY.AND.0C0H :       REM Isolate function bits
70  WRSFR (ADCO)BUSY :         REM Pick measurement input 0 (Port 7.0)
80  WRSFR (DAPR)0 :            REM Start measurement in highest meas. range
90  RDSFR (ADCO)BUSY :         REM Read A/D converter state
10  IF (BUSY.AND.020H)>0 THEN GOTO 90 :        REM wait if busy
110 RDSFR (ADDAT)VOLTAGE :     REM Read measurement value
120 PRINT "Voltage on Input 0 equals:",
130 PRINT VOLTAGE*RANGE/256,"Volt" :   REM Adapt value to meas. range
140 PRINT "Another measurement required in 1-V range (Y/N)?",
150 I=GET.AND.0DFH
160 IF I=ASC(Y) THEN GOTO 190
170 IF I=ASC(N) THEN GOTO 260
180 GOTO 150
190 RANGE=1 :                  REM Measurement range = 1 Volt
200 WRSFR (DAPR)040H :         REM Set range and launch measurement
210 RDSFR (ADCO)BUSY :         REM Read A/D converter state
220 IF (BUSY.AND.020H)>0 THEN GOTO 210 :       REM Wait if busy
230 RDSFR (ADDAT)VOLTAGE :     REM Read measurement value
240 PRINT "Voltage on Input 0 equals exactly:",
250 PRINT VOLTAGE*RANGE/256,"Volt" :   REM Adapt value to meas. range
260 END
```

The A/D converter in the 80517A has a resolution of 10 bits and increased measurement accuracy. However, with this type of controller it is not possible to modify the internal reference voltage via a control register. The following sample BASIC program shows how the input can be selected for the signal to be measured.

```
4   REM ***************************************************
5   REM * Program for controlling A/D converter in 80517A *
6   REM ***************************************************
10  RANGE=5 :          REM Controller reference voltage
20  ADCO=0D8H :        REM Register for measurement input selection
30  ADDATL=0DAH :              REM measurement value, LSB
40  ADDATH=0D9H :              REM measurement value, MSB
50  PRINT "Please enter number of measurement input (0-7):",
60  INPUT CHANNEL
70  CHANNEL=CHANNEL.AND.7 :    REM Mask superfluous bits
80  RDSFR (ADCO)BUSY :         REM Read function bits
90  BUSY=(BUSY.AND.0C0H).OR.CHANNEL :  REM add channel selection
100 NRSFR (ADCO)BUSY :         REM Copy channel selection to controller
110 WRSFR (ADDATL)0 :          REM Launch measurement
120 RDSFR (ADCO)BUSY :         REM Monitor A/D Busy state
130 IF (BUSY.AND.020H)>0 THEN GOTO 120 :       REM Wait if busy
140 RDSFR (ADDATL)UOLTPART :   REM Read least significant part
150 RDSFR (ADDATH)VOLTAGE :    REM Read most significant part
160 VOLTAGE=VOLTAGE+UOLTPART/256 :     REM Compute maesurement values
170 PRINT "Voltage at input:",CHANNEL,"equals:",
180 PRINT VOLTAGE*RANGE/256,"Volt" :   REM Adapt value to meas. range
190 END
```

EPROM programming

Unfortunately, MCS BASIC–52 V1.1 cannot program an EPROM when it is run from external memory. The improvement made to remedy this shortcoming is based on a suggestion made by R. Skowronek in 1996.
BASIC–52 V1.3 can directly program hardware EEPROMs using the 'PROG(1–6)' and 'PGM' commands, even when it is run from external program memory. The EEPROM is written in the same way as RAM, but with a suitably longer cycle time. With this change, port pins 1.3 ($\overline{\text{ALEDIS}}$), 1.4 ($\overline{\text{PRGPLS}}$) and 1.5 ($\overline{\text{PRGEN}}$), as well as bit 15h (INTELB) in the internal memory and memory locations 12Ah and 12Bh in external RAM, are no longer needed and thus can be freely utilised for other purposes. The 'PROG' commands are also redundant, so they have been deleted. This yields additional space savings, and the BASIC token 0F9h that has been made free is used for the new 'ERASE' command. This command erases an EEPROM by overwriting all memory locations between 08000h and 0C000h with 0FFh. 'ERASE' is called in command mode, and it takes around 2.75 minutes to erase this 16-kB region.
A special feature of BASIC–52 is the possibility of extending the command set using up to 16 user-defined commands programmed in assembly language. 'OPCODEs' are provided in order to allow system routines to be used in these extensions. Unfortunately, there was no OPCODE available to assign a value placed on the argument stack to a variable defined in BASIC text. Consequently, it was always necessary resort to awkward routines using the POP command. In BASIC–52 V1.3, OPCODE 43h has been defined for this purpose. It is used, for example, in the command extension 'RDSFR (address) variable'.
Previously, the instruction 'TIME = 0' did not reset the millisecond counter to zero, so that the value of TIME was not reset to exactly zero. This unfortunate error has been corrected in V1.3.
The BASIC–52 instruction 'ASC(x)' always returned an incorrect value for any character position occupied by a BASIC operator (*, +, –, /, <, =, > or

?). The interpreter translated these characters into equivalent tokens during the tokenisation process. For example, a question mark was translated into the token for 'PRINT". Consequently, the value of the token was returned instead of the correctly computed ASCII value of the character. BASIC–52 V1.3 now supplies correct values for all ASCII characters.
In order to allow BASIC–52 V1.3 to run on as many 8052-family derivatives as possible, and in particular on the Dallas Semiconductor 80C320 'Speed it µP' controller, the reset routine has been extensively reworked, as follows:

– A new baud rate detector utilises Timer 2 instead of 'code loop' timing to measure the utilised baud rate. This makes the measurement fully independent of the controller type and clock frequency.
– Timer 0, which is used for time measurements (BASIC Software Clock), is now operated in the 16-bit mode instead of the 13-bit mode that was previously used. This means that the maximum value for XTAL is now 78,627,473 Hz. The power-up default value is still 11,059,200 Hz, as before, but this value can be modified as necessary using 'XTAL = value'. Alternatively, the default value can be changed by patching the following locations in the EPROM with suitable new values:

17F1h	=	11
17F0h	=	05
17EFh	=	92
17EEh	=	00

For baud rate detection, the user still has to send a space character to BASIC–52 via the serial interface. The character length is then measured by the detection routine using Timer 2. In principle, BASIC–52 utilises only Timer 2 as the baud rate generator. This applies to 8052 controllers and the Dallas 80C320. The 80535, 80515 and 80517 controllers cannot activate Timer 2 as a baud rate generator. They have a special baud rate generator that is fully independent of Timer 2, and this allows only two different baud rates to be generated with a specific relationship to the processor clock frequency. If the clock frequency is exactly 12 MHz, the available rates are 4800 and 9600 baud. If the BASIC–52 V1.3 set-up routine detects this type of controller, it activates the baud rate generator with a baud rate that corresponds to that of the measured timing of the space character (4800 or 9600 baud).
The 80C517A controller, which is also frequently used, has an enhanced special baud rate generator. This supports arbitrary data transmission rates, but it still operates independently of Timer 2. BASIC–52 V1.3 also automatically recognises this processor type and activates it with the appropriate baud rate. With all of these special types of controller, Timer 2 is freely available to the user.

Table 1. 80535/537 A/D converter SFRs

SFR	Address	Bit(s)	Name(s)	Function
ADCON	0D8h	2–0	MX2,MX1,MXO	Select analogue input
ADCON	0D8h	3	ADEX	Start via hardware/software
ADCON	0D8h	4	ADM	Single/continuous conversion
ADCON	0D8h	5	BSY	Conversion in progress
DAPR	0DAh	3–0	DAPR.3-.0	Lower reference voltage
DAPR	0DAh	7–4	DAPR.7-.4	Upper reference voltage
ADDAT	0D9h	7–0		8-bit measurement result

I²C and SFR

BASIC–52 can easily be extended in assembly language with additional BASIC commands. Naturally, these extended commands cannot be located within the 8-kB code region of the BASIC interpreter. The authors have implemented six new commands. Four of these are related to I²C communications and have already been described in the article "Implementing the I²C Bus" in the July/August 2000 issue of *Elektor Electronics*, as follows:

I2CSTART
Transmits a Start condition on the I2C bus.
I2CSTOP
Transmits a Stop condition on the I2C bus.
I2CPUT (value)
Transmits a byte on the I2C bus.
I2CGET (value)
Reads a byte from the I2C bus into a variable.

Two additional commands allow Special Function Registers (SFRs) to be read and written. The various derivative versions in the controller family employ supplementary SFRs. The following commands have been implemented to allow these registers to be accessed via BASIC:

WRSFR (address) value
Writes 'value' to the indicated SFR address.
RDSFR (address) variable
Reads the content of the indicated SFR address and writes it to 'variable'.

Using these commands, it is easy to address internal extensions of the various processor derivatives, such as the additional I/O ports of an A/D converter. The two sample BASIC programs in the text boxes (AD-535.LIS and AD-517A.LIS) illustrate how the SFR commands can be used.
Since BASIC–52 V1.3 also runs on the 80535, 537 and 517 controllers, as well as other types, and since the external command extensions allow extended SFRs to be used without any problems to access new hardware features, it is now easy to write programmable control and adjustment programs in BASIC, instead of wrestling with the intricacies of assembly language programming. The sample programs illustrate the use of the A/D converter of an 80535 and an 80517(A). The A/D converter in an 80535/537 controller has a resolution of 8 bits. The external reference voltage for the converter is normally set to +5 V. This yields a minimum resolution of 20 mV. Since the internal reference voltage of the converter can be programmed using the 'DAPR' SFR, it is possible to achieve a higher resolution (for example, 4 mV) by making a second measurement using a smaller measurement range. The SFRs listed in **Table 1** can be used for this purpose.

(000121-1)