**FMO 52**

Flash Monitor Debugger for family 51

**USER MANUAL**

FMO 52 is an interactive software package for microprocessors of family 51 including a debugger, a monitor in machine language that allows to debug the user program.

FMO 52 requires the presence of an ATMEAL AT29c256 FLASH memory only to manage the programming and execution of final user program and uses very low amount of resources from the hardware where it is installed.

It provide features like:
- visualization and alteration of memory content
- breakpoint management
- single step execution
- real speed code execution
- a complete disassembler
- test of SRAM
- erasing of Flash
- programming of Flash with user program
- automatic execution of user program by AUTO RUN mode

Via dell' Artigiano, 8/6
40016 San Giorgio di Piano
(Bologna) ITALY
E-mail: grifo@grifo.it
http://www.grifo.it http://www.grifo.com
Tel. +39 051 892.052 (a. r.) FAX: +39 051 893.661

FMO 52 Rel. 5.00 Rel. 07 March 2003

GPC®, grifo®, are trade marks of grifo®
IMPORTANT

Although all the information contained herein have been carefully verified, grifo® assumes no responsibility for errors that might appear in this document, or for damage to things or persons resulting from technical errors, omission and improper use of this manual and of the related software and hardware. grifo® reserves the right to change the contents and form of this document, as well as the features and specification of its products at any time, without prior notice, to obtain always the best product.

For specific informations on the components mounted on the card, please refer to the Data Book of the builder or second sources.

SYMBOLS DESCRIPTION

In the manual could appear the following symbols:

- Attention: Generic danger
- Attention: High voltage

Trade Marks

GPC®, grifo®: are trade marks of grifo®.

Other Product and Company names listed, are trade marks of their respective companies.
GENERAL INDEX

INTRODUCTION ................................................................................................................... 1

PRELIMINARY INFORMATION ............................................................................................. 2

INSTALLATION .......................................................................................................................... 4

USING FMO 52 WITH BASCOM 8051 .................................................................................... 7

USING FMO 52 WITH GET 51 .............................................................................................. 10

COMPILING BASCOM 8051 PROGRAMS FOR FMO 52 ................................................. 14
   GENERATING TEST PROGRAMS .................................................................................. 14
   FINAL PROGRAMS GENERATION ................................................................................. 16

FLASH MANAGEMENT FROM USER PROGRAM ............................................................... 18

INTERRUPT ............................................................................................................................... 19

COMMANDS OF FMO 52 ........................................................................................................ 20
   DISPLAY OUTPUT ........................................................................................................... 20
   LIST ............................................................................................................................... 20

APPENDIX A: ALPHABETICAL INDEX ............................................................................. A-1
FIGURE INDEX

FIGURE 1: STARTING SCREEN ............................................................................................................ 3
FIGURE 2: MEMORY MAP FOR FMO 52 ........................................................................................... 4
FIGURE 3: GPC® CARDS CONFIGURATION FOR FMO 52 ............................................................. 5
FIGURE 4: BASCOM 8051 SERIAL PORT CONFIGURATION ........................................................... 7
FIGURE 5: PARAMETERS FOR COMMUNICATION WITH FMO 52 .................................................... 8
FIGURE 6: INTEL HEX FILE DOWNLOAD WITH BASCOM 8051 ................................................... 9
FIGURE 7: CONFIGURATION OF SERIAL PORT WITH GET 51 ...................................................... 10
FIGURE 8: PARAMETER TO COMMUNICATE WITH FMO 52 ........................................................ 11
FIGURE 9: INTEL HEX FILE DOWNLOAD USING GET 51 .............................................................. 11
FIGURE 10: LIST OF FMO 52 COMMANDS .................................................................................. 13
FIGURE 11: ENTER FLASH PROGRAMMING MODE ........................................................................ 16
FIGURE 12: PROGRAMMING FLASH MEMORY ........................................................................... 17
INTRODUCTION

The purpose of this handbook is to give the necessary information to the cognizant and sure use of the products. They are the result of a continual and systematic elaboration of data and technical tests saved and validated from the manufacturer, related to the inside modes of certainty and quality of the information.

To be on good terms with the products, is necessary guarantee legibility and conservation of the manual, also for future references. In case of deterioration or more easily for technical updates, consult the Web site www.grifo.com or the AUTHORIZED TECHNICAL ASSISTANCE directly.

To prevent problems during product utilization, it is a good practice to read carefully all the informations of this manual. After this reading, the User can use the general index and the alphabetical index, respectively at the begining and at the end of the manual, to find information in a faster and more easy way.

This program is copyrighted, all rights reserved. The control program or any part of it may not be analyzed, disassembled or modified in any form, on any medium, for any purpose.

This document is copyrighted, all rights reserved. This document or any part of it may not be copied, reproduced or translated in any form or in any way without the prior written permission of grifo®.

grifo® assumes no responsibility for misuse of this manual.

grifo® reserves the right to make changes or improvements to the product described in this manual at any time without notice.

This manual contains names of companies, software products, etc., which may be trademarks of their respective owners. grifo® respects those trademarks.
**PRELIMINARY INFORMATION**

**FMO 52** is an interactive software package for microprocessors of family 51 including a debugger, a monitor in machine language that allows to debug the user program downloaded in SRAM from address 2050H to 7FFFH, where data and code share the same space.

If code is compiled for execution in AUTO RUN mode in FLASH memory from 8050H to DFFFH, SRAM area is free and available for user data from address 20C6H to 7FFFH.

**FMO 52** requires the presence of an **ATMEL AT29c256** FLASH memory only to manage the programming and execution of final user program and uses very low amount of resources from the hardware where it is installed.

**FMO 52** uses one serial line for communication with development PC, 80 bytes of external SRAM (from 2000H to 204FH) for its variables, 6631 bytes of ROM for its code and **TIMER 1** for **baud rate generation and single step execution**.

By means to load an application program from development PC and set the requested breakpoints, **FMO 52** must be able to write into code area.

Harvard architecture of microcontroller 8051 does not allow this kind of operations because it separates data and code memory, so it is essential to address memory where the program is downloaded both as code and as data (**/PSEN and /RD or'ed**).

**FMO 52** is an **option orderable explicitly** for **grifo** cards that can support it (please see the list in the table of figure 3).

To order it the card name must be extended using the order code:

`.FMO52`

while to order also the FLASH the order code extention is:

`.FMO52.32KF`

It provides features like:

- visualization and alteration of memory content
- breakpoint management
- single step execution
- real speed code execution
- a complete disassembler
- test of SRAM
- erasing of Flash
- programming of Flash with user program
- automatic execution of user program by **AUTO RUN** mode

This manual is referred to **FMO 52 version 1.2 or greater** for all the **GPC** cards supported.
**Figure 1: Starting Screen**
FMO 52 is a downloader, monitor and debugger for microcontrollers of family 51. For its correct working the memory mapping of hardware where it runs must be like in the picture:

![Figure 2: Memory map for FMO 52](image)

**FMO 52** can support this memory mapping, amongst the others. Most of them are suitable for (and provided with) a specific version of **FMO 52**. Table of figure 3 lists all these cards and the information to configure memory mapping, memory type, their installation and how to switch between AUTO RUN and DEBUG mode. To know the address of last FLASH memory location please refer to card manual.

**FMO 52** is usually provided as an option orderable explicitly already installed and configured on the **GPC®** card that has been delivered. In this case, on the cards there must be:

- An EPROM labelled “**FMO 52 Rel. x.x GPC® yyyy**” where x.x is the **FMO 52** version and yyyy is the name of **GPC®** card
- A 32 KByte SRAM
- Optionally, a FLASH Atmel 29c256 size 256 KBytes

In case **FMO 52** has been ordered apart or in a second time, table of figure 3 explains how to install the EPROM and configure the **GPC®** to execute **FMO 52**. To identify sockets and jumpers please refer to **GPC®** card manual. In detail:

- **Card** indicates the card for which instructions are valid
- **Memory Mapping** indicates the mapping name as from the card manual
- **Mapping Selection** indicates jumpers/and dip switch to act on to access the mapping
- **EPROM** indicates the socket where to insert the EPROM labelled **FMO 52**
- **SRAM** indicates the socket where to insert SRAM (if not present)
<table>
<thead>
<tr>
<th>Card</th>
<th>Memory Mapping</th>
<th>Mapping Selection</th>
<th>EPROM</th>
<th>SRAM</th>
<th>FLASH</th>
<th>AUTO RUN</th>
<th>Jumpers Memories</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPC® 323/D</td>
<td>MODE 1</td>
<td>DIP 5 ON DIP 6 ON</td>
<td>IC 5</td>
<td>IC 4</td>
<td>IC 3</td>
<td>DIP 8 ON</td>
<td>J4 and J5 in 1-2</td>
</tr>
<tr>
<td>Rel. 110197</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>J2 in 1-2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>J3 pin 2 connected to J2 pin 3</td>
</tr>
<tr>
<td>GPC® 323/D</td>
<td>MODE 3</td>
<td>DIP 5 ON DIP 6 ON</td>
<td>IC 5</td>
<td>IC 4</td>
<td>IC 3</td>
<td>DIP 8 ON</td>
<td>J2 in 1-2 and 3-4</td>
</tr>
<tr>
<td>Rel. 250601</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GPC® 324/D</td>
<td>MODE 3</td>
<td>J2 connected J3 in 2-3</td>
<td>IC 8</td>
<td>soldered</td>
<td>IC 5</td>
<td></td>
<td>J4 in 1-2</td>
</tr>
<tr>
<td>Rel. 100997</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>J5 pin 2 connected to J4 pin 3</td>
</tr>
<tr>
<td>GPC® 324/D</td>
<td>MODE 3</td>
<td>J2 connected J3 in 2-3</td>
<td>IC 8</td>
<td>soldered</td>
<td>IC 5</td>
<td></td>
<td>J4 in 1-2</td>
</tr>
<tr>
<td>Rel. 110400</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>J2 and 3-4</td>
</tr>
<tr>
<td>GPC® 552</td>
<td>MODE 1</td>
<td>J1 in 3 J2 in 3 J14 in 1,2,3 ASM</td>
<td>IC 15</td>
<td>IC 13</td>
<td>IC 12</td>
<td>DIP 8 ON</td>
<td>J18 in 1-2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>J17 pin 2 connected to J18 pin 3</td>
</tr>
<tr>
<td>GPC® 553</td>
<td>MODE 3</td>
<td>DIP 5 ON DIP 6 ON</td>
<td>IC 1</td>
<td>IC 2</td>
<td>IC 3</td>
<td>DIP 8 ON</td>
<td>J4 and J5 in 1-2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>J2 in 1-2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>J3 pin 2 connected to J2 pin 3</td>
</tr>
<tr>
<td>GPC® 554</td>
<td>MODE 3</td>
<td>J1 connected J6 in 2-3</td>
<td>IC 4</td>
<td>soldered</td>
<td>IC 6</td>
<td></td>
<td>J3 in 1-2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>J4 pin 2 connected to J3 pin 3</td>
</tr>
<tr>
<td>GPC® R/T63</td>
<td>MODE 3</td>
<td>J2 connected J3 in 2-3</td>
<td>IC 7</td>
<td>soldered</td>
<td>IC 10</td>
<td></td>
<td>J4 in 1-2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>J3 pin 2 connected to J4 pin 3</td>
</tr>
<tr>
<td>GPC® 550</td>
<td>MODE 3</td>
<td>J4 in 2-3 J5 connected</td>
<td>U1</td>
<td>soldered</td>
<td>U3</td>
<td>DIP 8 ON</td>
<td>J1 in 1-2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>and 3-4</td>
</tr>
</tbody>
</table>

**Figure 3: GPC® cards configuration for FMO 52**
After having installed the memories and configured jumpers and/or dip switch, connect main serial port of GPC® card to a serial port of a PC or to a terminal emulator.
To know which is the main serial port and to know which cable to use please refer to GPC® manual.

**FMO 52** can work fine also with compilers featuring IDE's which embedd also a terminal emulator and allow explicitely file downloading modalities of the last compiled program in Intel Hex or S Motorola format.
For example **BASCOM 8051**, a cheap, efficent and easy to use basic compiler.
In addition, a grifo® product, terminal emulator **GET 51**, is created with a specific modality to manage the communication with **FMO 52**.

Specific instructions to configure the above mentioned products can be found in following chapters, instructions to configure a generic terminal emulator are:

- Baud Rate 19200
- Data bits 8
- Parity None
- Stop bits 1

After having configured the terminal emulator, supply or reset the GPC® card.
At power on or after a reset, **FMO 52** tests to see it AUTO RUN or DEBUG condition are set.

- if AUTO RUN is set, the user program stored in FLASH memory is immediatly executed, if present, or **FMO 52** will start normally
- if DEBUG is set **FMO 52** will start normally, this is useful to download user program to SRAM and debug it

Figure 1 shows the starting screen that you must see if GPC® is connected properly to terminal emulator and has been configured correctly.
In first row, name of GPC® will change according to which one you are using.
If you should not see it, please check all the settings explained in figure 3.

When prompt '*' is on the screen you can download user program in **Intel HEX** format directly after aving given the download command 'l', or start a debuggin/dumping session of data already present in the card's memories taking advantage of the powerful commands of **FMO 52** that will be explained in detail further.
USING FMO 52 WITH BASCOM 8051

BASCOM 8051 is a low cost, efficient and easy to use basic cross-compiler, provided with a complete IDE that runs under every version of Windows and can generate executable code for microcontrollers of family 51.

One of its IDE features is a terminal emulator running in a window and configurable both in classic serial port parameters and in behaviour.
In fact it is possible to activate through a comfortable menu the automatic send of a string sequence configurable by the user and of the last compiled file.

This way, FMO 52 can receive from BASCOM 8051 directly the download command, the Intel Hex format file resulting from last compile and the 'download complete' command, simply activating a menu voice.

To obtain this, BASCOM 8051 IDE must be configured as follows:

- Configure the serial port parameters like in figure 4 using the menu Options | Communications, selecting in the list box labelled 'COM port' the name of the PC serial port you have connected to GPC® main serial port, then press OK.

![Figure 4: BASCOM 8051 Serial Port Configuration](image-url)
- Configure the parameters for communication with **FMO 52** using the menu Options | Compiler | Monitor as shown in figure 5, then press OK.

**Figure 5: Parameters for communication with FMO 52**

- Write the source, or load it from a file, and compile it. For further information about how to compile a **BASCOM 8051** program suitable for **FMO 52**, please refer to specific paragraph of the manual. Every **GPC®** card of family 51 is provided with a demo complete program ready to be compiled with **BASCOM 8051** and designed to easy application development.

- Enter the terminal emulator by pressing the specific button on the button toolbar or by the menu Tools | Terminal emulator or pressing the combination of keys Ctrl t.

- Supply or reset **GPC®** card. You must see the starting screen shown on figure 1 with prompt "*". Repeat the first two steps of this list if you encounter problems.

- Download the Intel Hex file of last compiled program with menu File | Upload (see figure 6).

- After downloading, the hexadecimal figures that appears is a downloaded byte number count, it corresponds to the generated code size rounded to the next multiple of 16 (10 hexadecimal). The program is executed by typing the command G 2050. Please see the specific paragraph about commands for further information.
**Figure 6:** Intel Hex file download with BASCOM 8051
USING FMO 52 WITH GET 51

GET 51 is an intelligent terminal emulator compatible with ADDS-VIEWPOINT protocol and capable to interface directly with famous MCS BASIC and FMO 52. This manual will report only the information useful for communication with FMO 52, for further information about GET 51 please refer to its manual on grifo® CD or on grifo® Web Site at www.grifo.com.

One of GET 51 features is terminal emulation configurable both in classic serial port parameters and in behaviour. In fact it is possible to activate through a comfortable menu the send of a Intel Hex file to FMO 52.

This way, FMO 52 can receive from GET 51 directly the download command, the Intel Hex format file and the 'download complete' command, simply activating a menu voice.

To obtain this, GET 51 must be configured as follows:

- Configure the serial port parameters like in figure 7 using the menu Options | Serial Port, selecting under the label 'COM port' the name of the PC serial port you have connected to GPC® main serial port, then press OK.

![Figure 7: Configuration of serial port with GET 51](image-url)
**Figure 8:** Parameter to communicate with FMO 52

**Figure 9:** Intel Hex file download using GET 51
- Configure the parameters for communication with **FMO 52** using the menu Options | Set terminal as shown in figure 8, then press OK.

- Enter the terminal emulator by the menu Options | Terminal or pressing the combination of keys Alt t.

- Supply or reset **GPC** card. You must see the starting screen shown on figure 1 with prompt '*'. Repeat the first two steps of this list if you encounter problems.

- To select the file to download, press F8 (see figure 9). The window that appears allows to search for the file to download across all the mass memory storage devices available and to select it by pressing Enter, then pressing Enter again sends the selected file to **FMO 52**.

- After downloading, the hexadecimal figures that appears is a downloaded byte number count, it corresponds to the generated code size rounded to the next multiple of 16 (10 hexadecimal). The program is executed by typing the command G 2050. Please see the specific paragraph about commands for further information.
User RAM begin at 2050H, end at 7FFFH

*?

FMO52 Commands:

A <aa> - Alter internal memory
B [n aaaa] - display/set Breakpoints
C <r> <data> - Change register
D <aaaa>[aaaa] - Dump external memory
E <aaaa> - Edit external memory
F <aaaa>[aaaa] <dd> - Fill external memory
G [aaaa] - Go (execute program)
I <a>,<a> - dump Internal memory
L - Load program (download)
R - dump Registers
S - Single-Step next instruction
T - Test RAM
U <aaaa>[aaaa] - Un-assemble

******* Flash commands *******
H - Debug mode/Flash program mode
W - Copy all the RAM from 2050H into Flash from 8050H
X - Erase Flash

**Figure 10: List of FMO 52 commands**
When a **BASCOM 8051** source program is compiled, it is produced an Intel Hex file that contains the machine code for the used microcontroller. This file must be burned into an EPROM or into microcontroller internal ROM, but if the source program is still in the debug phase and it is not completed, it becomes necessary to repeat the burn operation for many times. This involves a big quantity of used time and uncomfortable component replacement.

To simplify the described debug phase it is sufficient to install the **FMO52** monitor on **GPC®** card, if it is not already installed, following the instructions in chapter “INSTALLATION” and take advantage of the possibility to download the program to debug in SRAM installed on **GPC®**.

This allows to test the program an unlimited number of times directly on the hardware structure that will be installed in the final application, without ever having to erase and reprogram the EPROM.

### GENERATING TEST PROGRAMS

**BASCOM 8051** compiler can generate executable binary files without constraints on clock frequency, memory structure and CPU installed on the card where the program is going to run. To modify each of these parameters it is possible to use the IDE menus, or (recomended) to write compile directives on the source code.

These directives must be written at the beginning of the source and inform the compiler about the hardware where the program will run, in addition they document the information in a human readable way to help the programmer.

In detail:

- Directive $regfile specifies a file that describes SFR of the CPU used
- Directive $crystal indicates the quartz frequency in Hertz
- Directive $baud indicates baud rate of user program serial communication (not baud rate of FMO 52 serial communication, which is always 19200 baud)
- Directive $romstart indicates the address of the first byte of program code
- Directive $ramstart indicated the address of the first byte of external memory
- Directive $ramsize indicates external SRAM size

The are several other directives not related to **FMO 52** but useful for the programmer, for further information please refer to **BASCOM 8051** on-line manual.

As already said in first chapter of this manual, **FMO 52** requires a particular structure of memory, shown in figure 2, that can be obtained on GPC® cards with settings described in figure 3. In detail, the values for $romstart, $ramstart and $ramsize are different for the test program and for the final program.

Settings for the test program are:
The value of $romstart is the address of the first SRAM location for the test program, because locations from 2000H to 204FH are used by FMO 52 variables.
Value of $ramstat is assigned temporarily because in the SRAM locations from 2050H to 7FFFH program code and external variables must live together.
Value of $ramsize is the SRAM not occupied by the program code, so it is temporary too.

Directive $regfile tells the compiler which SFR description file must be used for the next compilation. These are text files, human readable and writeable, and instruct BASCOM 8051 about SFR different from the standard ones of 8051 architecture.
Essentially, each one of these files describes a different family 51 microcontroller.
For further information, please refer to BASCOM 8051 documentation.
Current correspondence between grifo® GPC® cards and SFR descriptor files is:

<table>
<thead>
<tr>
<th>Card</th>
<th>File</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPC® R/T63</td>
<td>8052.DAT</td>
</tr>
<tr>
<td>GPC® 550</td>
<td>80552m.DAT</td>
</tr>
<tr>
<td>GPC® 552</td>
<td>80552m.DAT</td>
</tr>
<tr>
<td>GPC® 553</td>
<td>80552m.DAT</td>
</tr>
<tr>
<td>GPC® 554</td>
<td>80552m.DAT</td>
</tr>
<tr>
<td>GPC® 323/D</td>
<td>87C520.DAT</td>
</tr>
<tr>
<td>GPC® 324/D</td>
<td>87C520.DAT</td>
</tr>
</tbody>
</table>

As an example, the beginning of a test program source code is reported here:

```
$regfile = "80552m.dat"
$crystal = 22118400   ' microcontroller clock frequency
$sbaud = 19200       ' RS 232 communication speed
$romstart = &H2050    ' machine code start address
$ramstart = &H6000    ' external data start address
$ramsize = &H1FFF     ' external data area size is 8K
$large                ' 16 bit addressing
'************************** Program Start **************************
Waitms 1              ' delay for signals settling
.....                 ' main body of program
.....                 ' optional procedures
End                   ' end of program
.....                 ' end of program
```

The instruction 'Waitms 1' must be present. If you don't see the program running please check for the presence of this instruction.
FINAL PROGRAMS GENERATION

All the considerations written for test program generation are valid also for final program generation. The destination memory bank changes, that is the program will have to be downloaded and written to the **optional** FLASH memory, if present, so compilation directives and downloading procedure must change too.

As can be seen from the memory map of figure 1, FLASH memory addresses start form 8000H and reach a final address that depends on the target board, so please refer to the card manual to know it. Directives $ramstart, $romstart and $ramsize must be changed like this:

\[
\begin{align*}
$romstart &= &H8050 \quad \text{'machine code start address} \\
$ramstart &= &H20C6 \quad \text{'external data start address} \\
$ramsize &= &H5EFF \quad \text{'external data area size is 24K}
\end{align*}
\]

In this case the values of $ramstart and $ramsize are exact and must copied as written here.

After doing this modification it is possible to recompile the program, the operations are the same indicated in the chapters “USING FMO 52 WITH...” except for the last two ones:

**FIGURE 11: ENTER FLASH PROGRAMMING MODE**
- At the prompt `*`, to enter the **FMO 52** FLASH programming mode type `h`. The message on the first lines of figure 11 should appear.

- At prompt `&`, download the program as indicated in the chapters “USING FMO 52 WITH...” (please refer also to figures 6 or 9 according to the terminal emulator used).

- Type `w`. This command writes into the FLASH memory the program just downloaded and sets the **FMO 52** flag that indicates the presence of a finale user program in the non volatile memory.

- To obtain the automatic execution of the program stored in FLASH you have to activate the **AUTO RUN** mode (performing the corresponding setting shown in the table of figure 3) and reset or turn off and on the card.

For further information about **FMO 52** commands, please refer to the specific chapter.

---

**FIGURE 12: PROGRAMMING FLASH MEMORY**
FLASH MANAGEMENT FROM USER PROGRAM

FLASH memory area is divided in three parts:

- from 8000H to 804FH   Reserved
- from 8050H to DFFFH   User program
- from E000H to FEFFH   Free memory, divided in 124 blocks of 64 bytes

The last part of FLASH memory can be managed by the user during the AUTO RUN mode in block of 64 bytes using a CALL to address 1A00H which takes 2 bytes of stack. The very last byte of memory depends on the card used.

To use properly the above mentioned procedure it is essential to initialize some external memory locations:

- from 2001H to 2042H   64 bytes that will be written to the Flash
- 2043H                  indicates the block number, from 0 to 7BH
- 2044H                  Check sum of all data, which means to sum the values of all the memory locations from 2001H to 2043H without carry, and adding 1.

After using this procedure, the content of location 2043H reports the result of last programming operation according to its value:

- AAH                Invalid block number
- BBH                Wrong check-sum
- EEH                Flash malfunction
- 80H                OK, operation correctly performed

For example, to write data in block 1 of FLASH memory, filling it with zero’s (0), these are the values to write in the above mentioned locations:

- Data: da 2001H a 2042H 00H .... 00H
- Block: 2043H 01H
- Check-sum: 2044H 02H
**INTERRUPT**

**FMO52** re-vectors all interrupts (except for **TIMER 1 OVERFLOW** which is used internally for single-stepping and takes three more SRAM memory locations, see command 'S') to the corresponding locations in the first page of user SRAM.

When a user program happens to use interrupts it will just have to:

- allocate them in the user SRAM, if **AUTO RUN** mode is not selected, that is to sum to each of them the starting address of such SRAM (2050H)

- allocate them in the FLASH user area, if **AUTO RUN** mode is selected, that is to sum to each of them the starting address of such FLASH (8050H)

Interrupt response procedures should not be changed.

Using **BASCOM 8051** of course, interrupt vectors allocation is made automatically by the compiler so the programmer does not have to worry about it.

When the user program is executed in **AUTO RUN** mode, **FMO 52** copies the interrupt vectors from Flash to SRAM before running the program. In detail, the vectors are copied from 8050H - 80C5H to 2050H - 20C5H, to have the same configuration as in debugging mode.
COMMANDS OF FMO 52

DISPLAY OUTPUT

In order to maintain the maximum amount of data on the terminal screen during the debugging session, **FMO 52** has been designed to minimize the number of screen lines in any display. All commands which output only one line of data (such as 'B' and 'S'), will do so on the same line on which the command is issued (To the right of the command).

For commands which output larger amounts of data (such as 'D', 'U'), pressing the SPACE key during the display will halt the display on the next line. Each subsequent SPACE entered will cause one more line of data be displayed. Pressing CARRIAGE RETURN will allow the display to continue at full speed.

The escape key may be used to cancel command output, and return to the '*' or '&prompt.

LIST

A <address>

Alter INTERNAL memory. **FMO 52** prompts with the specified address and its current contents. You may enter TWO hex digits to change its value, SPACE to advance to the next location, BACKSPACE to backup to the previous location, or CARRIAGE RETURN to terminate the Alter command.

B <breakpoint#> <address>

Set breakpoint at specified address. Breakpoint is removed if address is 0000. There can be up to four breakpoints, which are referenced by the numbers 0-3.

If SPACE is entered instead of a breakpoint#, the currently set breakpoint addresses are displayed.

**NOTE:** The 8051 family of processors does not have a single byte transfer instruction such as is normally used to implement breakpoints. Breakpoints are handled by inserting 'LCALL' instructions into your code during the processing of a 'G' command and restoring the code in the breakpoint handler. Each 'LCALL' occupies three bytes of memory, which causes the following restrictions when using breakpoints:

You MUST be careful to place breakpoints in locations where there will NOT be any JUMPs or CALLs to the addresses containing the second and third bytes of the breakpoint. For example, if you set a breakpoint at address 2234, there should NOT be a label in your program occurring at address 2235 or 2236 (Note, a label at 2234 is OK).
You may not set breakpoints that are within three bytes of each other. The message 'Breakpoint conflict' results if you attempt to do so.

Attempt to 'G'o at an address containing a breakpoint will also result in the 'Breakpoint conflict' message. This will most commonly occur when you wish to resume execution following a breakpoint. In this case, you must either remove the breakpoint, or use the 'S'tep command to advance the program counter until it is NOT positioned over any part of a breakpoint (Remember, breakpoints are three bytes long).

**C** <register> <value>

Changes 8051 registers values. Register is a single character, which may be as follows:

- **A** - Set Accumulator (8 bit value).
- **B** - Set B register (8 bit value).
- **D** - Set DPTR (16 bit value).
- **S** - Set stack pointer (8 bit value).
- **P** - Set program counter (16 bit value).
- **W** - Set PSW (8 bit value).
- **0-7** - Set R0-R7 in current register bank (8 bit value).

**D** <start>,<end>

Displays EXTERNAL DATA memory, in HEX/ASCII dump format, starting at the indicated address. If a SPACE is entered for <end> address, assumes FFFF.

**E** <address>

Edit's EXTERNAL DATA memory. Address and contents are displayed. Enter TWO hex digits to change value. Entering SPACE skips to the next location, BACKSPACE backups to the previous location. CARRIAGE RETURN terminates the edit command.

**F** <start>,<end> <value>

Fill's external memory from <start> to <end> with the byte <value>.
G `<address>`

Begins execution at the indicated address. If a SPACE is entered instead of an address, begins execution at the address in the 8051 program counter.

H

Enables the use of the FLASH programming related commands, such modality is indicated by the prompt shape, if '*' is printed then you are working in debugging mode, while if the '&' character is printed then the Flash related command are enabled (the different situation influences the command L):

- **W** Flash writing, copy of the SRAM content from 2050H to 7FFFH, into the Flash memory from 8050H to DFFFH.
- **X** Erasing of the whole content of FLASH memory.

I `<start>,<end>`

Displays the contents of INTERNAL memory between the specified addresses.

L

Downloads data from the console port, which may be in either MOTOROLA or Intel Hex format. If you accidently enter this command, you may enter either 'S9' or ':00' to signify a null download file and return to the command prompt.

If the prompt shown is '*' then the code downloaded will be used in debugging mode, while if the prompt is '&' then the code will be used to program the Flash memory, in fact during the download the file will be reallocated 6000H bytes backward (6000H is the offset between RAM and Flash), this will make possible the Flash programming.

On startup the user program will be automatically executed from Flash memory if AUTO RUN mode is enabled (see table of figure 3).

R

Displays the current values of the 8051 registers (A, B, DPTR, SP, PC, PSW and R0-R7).
S

Single-Steps one instruction from the current 8051 Program Counter address. Disassembly of the instruction stepped is displayed on the console. The use this command involves the loss of TIMER1 overflow interrupt, as this is redirected to the SRAM memory 206BH, 206CH and 206DH locations. **NOTE:** Rarely, with microcontrollers Dallas 80c320 and 80c552, this command executes two instructions instead of only one, stopping at the instruction next to the one where the command has been invoked.

T

A complete SRAM test is performed from 2050H to 7FFFH. This command alters definitely the SRAM content, after having used it is suggested to reformat the SRAM content with value 00H by a F command. (Example: F 2050 7FFF 00)

U <start>,<end>

Un-assembles PROGRAM memory, starting at indicated address. If SPACE is entered for <end> address, assumes FFFF. Disassembler output contains address, opcodes bytes, instruction mnemonic, and operands to instruction.

? 

Displays a short help summary of the commands.
APPENDIX A: ALPHABETICAL INDEX

SYMBOLS

.FMO52  2
.FMO52.32KF  2
?  23

A

ADDS-VIEWPOINT  10
ALTER INTERNAL MEMORY  20
AT29C256  2
AUTO RUN  5, 6, 17, 19

B

BASCOM 8051  6, 7, 19
BREAKPOINT  20

C

CHANGE REGISTERS  21
COMMANDS OF FMO 52  20
?  23
A  20
B  20
C  21
D  21
DISPLAY OUTPUT  20
E  21
F  21
G  22
H  22
I  22
L  22
R  22
S  23
T  23
U  23

COMPILING BASCOM 8051 PROGRAMS FOR FMO 52  14

D

DEBUG  5, 14
DISPLAY 8051 REGISTERS  22
DISPLAY HELP  23
DOWNLOAD DATA  22
DUMP  21
E
EDITOR MEMOY 21
EPROM 4

F
FILL MEMORY 21
FLASH 2, 4, 16, 18, 22
FLASH MANAGEMENT FROM USER PROGRAM 18

G
GET 51 6, 10
GPC® 5, 15
grifo® 2

I
INTEL HEX 6
INTERRUPT 19
INTRODUCTION 1

M
MEMORY MAPPING 4

S
SINGLE-STEP 23
SRAM TEST 23

T
TEST PROGRAMS 14
TIMER 1 19

U
UN-ASSEMBLE 23

W
WAITMS 15