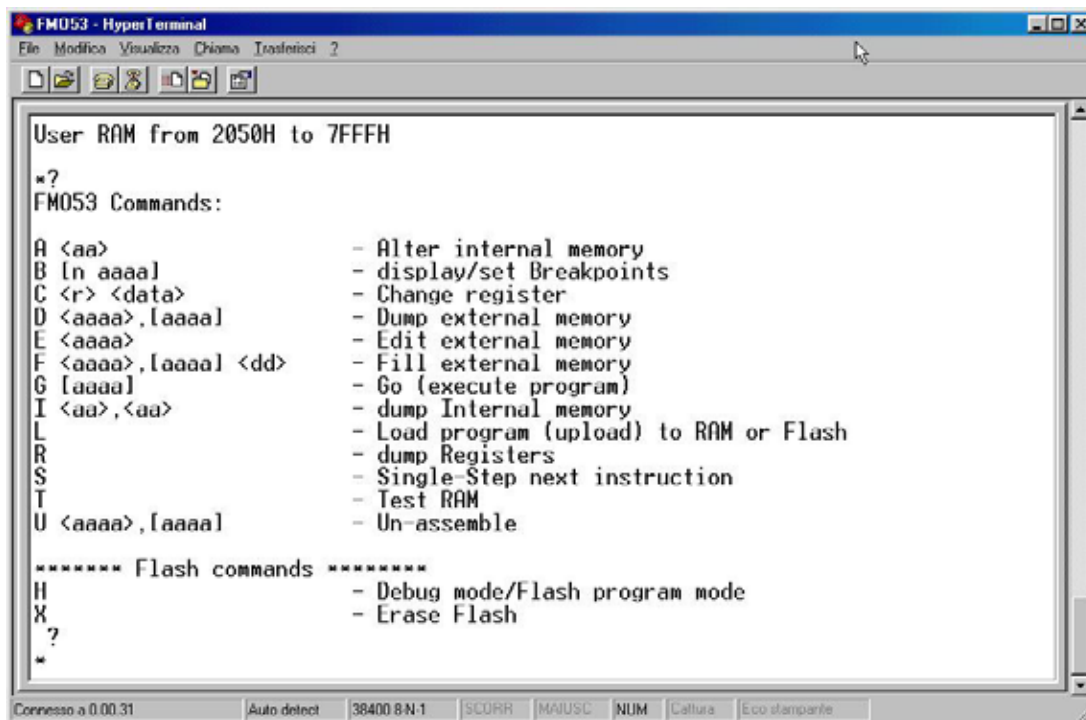


# FMO 53

Flash Monitor Debugger for 51 family

## USER MANUAL



```
FM053 - HyperTerminal
File Modifica Visualizza Chiama Istruzioni 2
User RAM from 2050H to 7FFFH
*?
FM053 Commands:
A <aa> - Alter internal memory
B ln 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 <aa>,<aa> - dump Internal memory
L - Load program (upload) to RAM or Flash
R - dump Registers
S - Single-Step next instruction
T - Test RAM
U <aaaa>,[aaaa] - Un-assemble

***** Flash commands *****
H - Debug mode/Flash program mode
X - Erase Flash
?
*
```

**grifo**<sup>®</sup>

ITALIAN TECHNOLOGY

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>

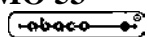
Tel. +39 051 892.052 (a.r.) FAX: +39 051 893.661



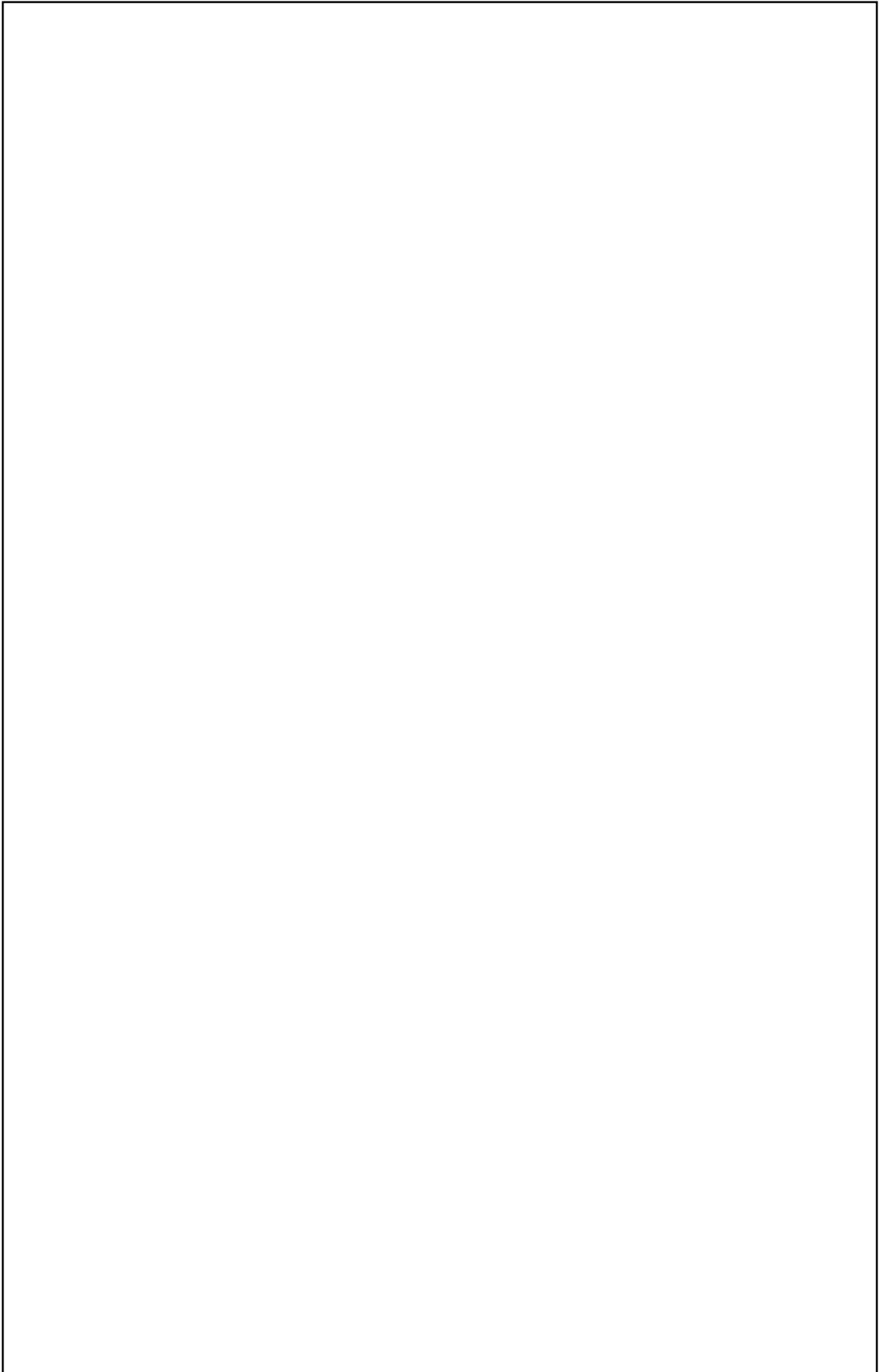
FMO 53

Rel. 5.00

Edition 11 February 2009



, GPC<sup>®</sup>, grifo<sup>®</sup>, are trade marks of grifo<sup>®</sup>



# FMO 53

Flash Monitor Debugger for 51 family

## USER MANUAL

**FMO 53** is an interactive software tools, composed by a monitor debugger and a FLASH EPROM manager, that allows to develop the user application in a fast and comfortable way. It is available for many **GPC**® cards included in **grifo**® industrial boards group, based on Intel 51 microprocessor family. Thanks to **FMO 53** each user can prepare the firmware for the used control card, with no requirements of complicated and expensive external development systems, obtaining a considerable reduction of time and costs. The inexpensive use of **FMO 53** is confirmed also by the slight list of required items: it can be used only with a standard PC connected to preselected card, through RS 232 serial line.

The **FMO 53** provides numerous functions as dump and change of memories content, breakpoints management, single step code execution, real time code execution, inspection and modification of microcontroller's registers, a complete disassembler, test of RAM, FLASH erasure, FLASH programming with user program and automatic execution of the program saved on FLASH. These functions are the typical features of an hardware In Circuit Emulator and they are well replaced by **FMO 53**, that is completely realized by software.

**grifo**®

ITALIAN TECHNOLOGY

Via dell' Artigiano, 8/6  
40016 San Giorgio di Piano  
(Bologna) ITALY

E-mail: [grifo@grifo.it](mailto:grifo@grifo.it)

<http://www.grifo.it>

<http://www.grifo.com>

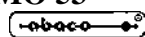
Tel. +39 051 892.052 (a.r.) FAX: +39 051 893.661



**FMO 53**

Rel. 5.00

Edition 11 February 2009



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

## DOCUMENTATION COPYRIGHT BY **grifo**<sup>®</sup>, ALL RIGHTS RESERVED

No part of this document may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means, either electronic, mechanical, magnetic, optical, chemical, manual, or otherwise, without the prior written consent of **grifo**<sup>®</sup>.

### IMPORTANT

Although all the information contained herein have been carefully verified, **grifo**<sup>®</sup> 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**<sup>®</sup> 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

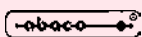


Attention: High voltage



Attention: ESD sensitive device

### Trade Marks



**GPC**<sup>®</sup>, **grifo**<sup>®</sup> : are trade marks of **grifo**<sup>®</sup>.

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

# GENERAL INDEX

INTRODUCTION .....	1
VERSIONS .....	3
GENERAL INFORMATION .....	4
REQUIREMENTS .....	7
CONTROL BOARD .....	7
PERSONAL COMPUTER .....	7
SERIAL COMMUNICATION CABLE .....	8
SOFTWARE OF WORK .....	9
APPLICATION PROGRAM DEVELOPMENT TOOL .....	10
SERIAL TERMINAL EMULATION PROGRAM .....	12
EPROM PROGRAMMER .....	12
FMO 53 DESCRIPTION .....	13
CONFIGURATION BOARD .....	13
USED RESOURCES .....	18
SERIAL COMMUNICATION .....	19
USER APPLICATION PROGRAM CONFIGURATION .....	20
FLASH EPROM MANAGEMENT .....	22
CODE ON FLASH .....	22
DATA ON FLASH .....	23
OPERATING MODE SELECTION .....	24
INTERRUPTS .....	24
INTEGRATION WITH DEVELOPING ENVIRONMENT .....	25
USE WITH BASCOM 8051 .....	25
CONFIGURATIONS FOR SERIAL COMMUNICATION .....	25
APPLICATION PROGRAM INSTRUCTION .....	27
USING FMO 53 WITH GET 51 .....	29
CONFIGURATION FOR SERIAL COMMUNICATION .....	29
USING WITH HYPERTERMINAL .....	31
CONFIGURATIO FOR SERIAL COMMUNICATION .....	31
USING WITH $\mu$ C/51 .....	34
CONFIGURATIONS FOR SERIAL COMMUNICATION .....	35
APPLICATION PROGRAM INSTRUCTION .....	35
COMMANDS .....	37
ALTER INTERNAL MEMORY .....	38
SET AND SHOW BREAKPOINTS .....	38
MODIFY REGISTERS .....	39
SHOW EXTERNAL DATA MEMORY .....	39
MODIFY EXTERNAL DATA MEMORY .....	39
FILL EXTERNAL DATA MEMORY .....	40

PERFORM REAL SPEED .....	40
ENABLE COMMANDS FOR FLASH .....	40
SHOW INTERNAL MEMORY .....	40
LOAD FILE .....	41
SHOW REGISTERS .....	42
EXECUTE STEP TO STEP .....	42
BOARD SRAM TEST .....	42
CODE UNASSEMBLE .....	43
ERASE FLASH .....	43
SHOW COMMANDS .....	43
HOW TO START .....	44
APPENDIX A: ALPHABETICAL INDEX .....	49

# FIGURE INDEX

<b>FIGURE 1: SERIAL COMMUNICATION BETWEEN DEVELOPMENT PC AND CONTROL BOARD .....</b>	<b>8</b>
<b>FIGURE 2: CONNECTOR AND ACCESSORIES FOR SERIAL COMMUNICATION .....</b>	<b>9</b>
<b>FIGURE 3: WORKING MODALITY .....</b>	<b>11</b>
<b>FIGURE 4: BOARD CONFIGURATION (1 OF 2).....</b>	<b>14</b>
<b>FIGURE 5: BOARD CONFIGURATION (2 OF 2).....</b>	<b>15</b>
<b>FIGURE 6: MEMORY ORGANIZATION IN DEBUG MODE .....</b>	<b>16</b>
<b>FIGURE 7: MEMORY ORGANIZATION IN AUTORUN MODE .....</b>	<b>17</b>
<b>FIGURE 8: USED SERIAL LINE SELECTION .....</b>	<b>19</b>
<b>FIGURE 9: MEMORY AREAS ADDRESSES IN DEBUG MODE .....</b>	<b>20</b>
<b>FIGURE 10: MEMORY AREAS ADDRESSES IN AUTORUN MODE .....</b>	<b>21</b>
<b>FIGURE 11: MEMORY AREAS MAXIMUM SIZE .....</b>	<b>21</b>
<b>FIGURE 12: BASCOM 8051 EMULATION TERMINAL CONFIGURATION .....</b>	<b>26</b>
<b>FIGURE 13: DOWNLOADING FILE CONFIGURATION WITH BASCOM 8051 .....</b>	<b>26</b>
<b>FIGURE 14: UPLOADING FILE WITH BASCOM 8051 .....</b>	<b>27</b>
<b>FIGURE 15: CONFIGURATION OF SERIAL PORT WITH GET 51 .....</b>	<b>29</b>
<b>FIGURE 16: PARAMETER TO COMMUNICATE WITH FMO 52 .....</b>	<b>30</b>
<b>FIGURE 17: INTEL HEX FILE DOWNLOAD USING GET 51 .....</b>	<b>30</b>
<b>FIGURE 18: HYPERTERMINAL COMMUNICATIO CONFIGURATION ( 1 OF 4) .....</b>	<b>31</b>
<b>FIGURE 19: HYPERTERMINAL COMMUNICATIO CONFIGURATION ( 2 OF 4) .....</b>	<b>32</b>
<b>FIGURE 20: HYPERTERMINAL COMMUNICATIO CONFIGURATION ( 3 OF 4) .....</b>	<b>32</b>
<b>FIGURE 21: HYPERTERMINAL COMMUNICATIO CONFIGURATION ( 4 OF 4) .....</b>	<b>33</b>
<b>FIGURE 22: DOWNLOADING FILE WITH HYPERTERMINAL (1 OF 2) .....</b>	<b>33</b>
<b>FIGURE 23: DOWNLOADING FILE WITH HYPERTERMINAL .....</b>	<b>34</b>
<b>FIGURE 24: COMMAND LIST .....</b>	<b>37</b>
<b>FIGURE 25: ENABLING COMMANDS FOR FLASH .....</b>	<b>41</b>
<b>FIGURA 26: FMO53 POWER ON MESSAGE ON SERIAL A .....</b>	<b>45</b>
<b>FIGURE 27: FMO53 POWER ON MESSAGE ON SERIAL B .....</b>	<b>45</b>
<b>FIGURE 28: DOWNLOADING AND EXECUTION OF PROGRAM IN SRAM.....</b>	<b>46</b>
<b>FIGURE 29: DOWNLOADING AND EXECUTION OF PROGRAM IN FLASH .....</b>	<b>47</b>
<b>FIGURE 30: AUTOMATIC PROGRAM STARTING IN AUTORUN .....</b>	<b>48</b>

## INTRODUCTION

The use of these devices has turned - IN EXCLUSIVE WAY - to specialized personnel. This device is not a **safe component** as defined in directive **98-37/CE**.



Pins of Mini Module are not provided with any kind of ESD protection. They are connected directly to their respective pins of microcontroller. Mini Module is affected by electrostatic discharges. Personnel who handles Mini Modules is invited to take all necessary precautions to avoid possible damages caused by electrostatic discharges.

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.

The reported data are destined- IN EXCLUSIVE WAY- to specialized users, that can interact with the devices in safety conditions for the persons, for the machine and for the environment, impersonating an elementary diagnostic of breakdowns and of malfunction conditions by performing simple functional verify operations , in the height respect of the actual safety and health norms.

The informations for the installation, the assemblage, the dismantlement, the handling, the adjustment, the reparation and the contingent accessories, devices etc. installation are destined - and then executable - always and in exclusive way from specialized warned and educated personnel, or directly from the TECHNICAL AUTHORIZED ASSISTANCE, in the height respect of the manufacturer recommendations and the actual safety and health norms.

The devices can't be used outside a box. The user must always insert the cards in a container that respect the actual safety normative. The protection of this container is not threshold to the only atmospheric agents, but specially to mechanic, electric, magnetic, etc. ones.

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 AUTHORIZED TECHNICAL ASSISTANCE directly.

To prevent problems during card 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, respectly at the begining and at the end of the manual, to find information in a faster and more easy way.



## VERSIONS

This handbook makes reference to version **2.0** of the **FMO53** product and following ones. The validity of the information contained in this manual is subordinated to the version number of the used firmware and the user must always verify the correct correspondence between the notations. The version number is reported on the received EPROM label and it is also displayed by the device during the configuration.

Normally the **FMO53** is always supplied with the latest firmware version that is available but, for specific requirements, the user can receive also a different version; he must carefully specify this particular condition in the order phase.

In addition, this manual reports information about other different programs that are integrant parts of **FMO53** each one of these programs has an own version number that is specifically described when it is necessary. Finally also the hardware is provided of his version as indicated in the related technical manuals.

When the user requires technical assistance it is really important that he provides a description of the problem plus the version numbers of the used components.

Like any products, also **FMO53** is continuously changed and improved to satisfy completely the new requirements of the users and correct the discovered problems and bugs. Here follows a brief description of the changes made to the package according to version number:

- Ver. 1.0 -> First version for internal development and test.
- Ver. 1.1 -> First realeased version.
- Ver. 1.2 -> Add the execution on boards with DALLAS microprocessor, with its short execution time.
- Ver. 1.3 -> Add the execution with **GPC® 550**.
- Ver. 2.0 -> Add the used serial selection on board with two lines; increased the serial communication baud rate, delated the serial initialization in case of program in autorun; increased the maximum size of user program memorizable on FLASH; improved the FLASH management; improved the SRAM test; corrected the initial setting of program counter; changed the pause, re-take and stop mode of long viewing.

Any eventual improvement or addition the user thinks may be interesting, can be suggested by contacting directly **grifo®**.

## GENERAL INFORMATION

**FMO53** is an interactive software tools, composed by a monitor debugger and a FLASH EPROM manager, that allows to develop the user application in a fast and comfortable way. It is available for many **GPC®** cards included in **grifo®** industrial boards group, based on Intel 51 microprocessor family. Thanks to **FMO53** each user can prepare the firmware for the used control card, with no requirements of complicated and expensive external development systems, obtaining a considerable reduction of time and costs. The inexpensive use of **FMO53** is confirmed also by the slight list of required items: it can be used only with a standard PC connected to preselected card, through RS 232 serial line.

The monitor debugger operates in machine language and therefore it can be joined with any programming language, always for I51 family; among these we can remind the numerous assemblers, C compilers, BASIC compilers, PASCAL compilers, etc.

The **FMO53** provides numerous functions as dump and change of memories content, breakpoints management, single step code execution, real time code execution, inspection and modification of microcontroller's registers, a complete disassembler, test of RAM, FLASH erasure, FLASH programming with user program and automatic execution of the program saved on FLASH. These functions are the typical features of an hardware In Circuit Emulator and they are well replaced by **FMO53**, that is completely realized by software.

About operations to perform, the user must only write the application program for the used card, through the selected programming language; at this point he must convert the source in executable code (compile and/or assemble it) and then upload it to card through **FMO53**. The program, once uploaded, can be executed on the board by obtaining the possibility to test his functionality directly on the real system; if the functionality has some bugs the user must solve them by repeating the steps described up to now. When a functionality completely right is obtained the user can proceed by saving the tested application program on FLASH EPROM, always through **FMO53**. At this point the card is ready to be installed on the final plant infact in the following power ons the application program will start automatically. Whenever, after a variable time period, the user must attend on application program to verify or update it, the **FMO53** can stop the execution of saved program and re-allow all the operations previously described, with a simple displacement of a proper on board selector.

It is important to remind that **FMO53** uses very few resourcers of the used card and it is not intrusive for the user application program. However these resources (described in detail in following descriptions) are completely released when the program saved on FLASH is executed; the user can develop his program as if it were running standalone, without fear of any conflicts.

The most important features of **FMO53** are below summarized:

- Monitor debugger for **GPC®** cards of **grifo®**, provided of Intel 51 microcontrollers (see list on following paragraphs).
- Communication on asynchronous serial line, in **RS 232**.
- Physical communication protocol:
  - 38400** Baud;
  - 8 bits** for character;
  - No Parity**;
  - 1 Stop Bit**.

- Communication **serial line selectable** (between **A** and **B**) on those cards provided of two independent lines.
- It can be used with numerous communication program for **PC**. Basically it is sufficient a terminal emulation program capable to use the described physic protocol. Among these programs we can remind those specially developed by **grifo®** (as **GET51**) or those prepared by other companies (i.e. **HYPERTERMINAL** of Windows).
- Numerous available **commands**, equivalent to as many functionalities (see following paragraphs for details).
- Possibility to upload application program to the card, either in **HEX Intel** or **S Motorola** format.
- Fast upload** of the program (normally in **1 second** it is transferred **1K** Bytes of code, equal to about **3K** Bytes of HEX file).
- Application program can be executed in **real time** or **single step mode**.
- Management of **four different breakpoints**, that can be freely inserted by the user to check the program execution flow.
- The microcontroller's **data area** and **code area** are addressed in overlapped mode, to allows either read and write access to both type of memories.
- Different types of memories are addressed, with maximum dimension below reported:
  - up to **55,75K** Bytes of **SRAM**;
  - up to **31,75K** Bytes of **FLASH**.
- The user application program can be **saved** on **FLASH** non volatile memory.
- Management of **AUTORUN** mode where the application program saved on FLASH is **automatically** executed after a reset or power on.
- Management of **DEBUG** mode where after a reset or power on, the **FMO53** is always executed.
- Selector of start modality (**AUTORUN** or **DEBUG**) through on board **DIP-SWITCH** or **Jumper**.
- It uses **very few resources** of selected card and these are released when user application program is executed:
  - one serial line for the communication with development PC;
    - up to 208 Bytes** of external **RAM**;
    - 8K** Bytes of **EPROM** for ots code;
    - TIMER1** as baud rate generator for communication and for sinle step execution;
    - one configuration input as selector of start modality;
    - from **1 to 3 interrupts**, according to used serial line
- It **preserves/restore** the entire contents of registers and internal RAM when it take/pass the control from/to user program.
- Optimized visualization** of information on a PC screen, that allow to mantain a complete vision of the card status, during the debug phase.
- For the commands that show large quantity of information, are provided the **suspend**, **restart** and **abort** actions on visualization.
- Complete management of microcontroller's **interrupts**, that are redirected in proper areas of code memory; the user have not to modify the interrupt service routines but he must simply relocate them in the foregoing areas. Normally the **relocation** of the code start is sufficient to relocate all the **interrupts vectors**, too.
- It includes **utility** procedures that can be used to write data areas of the FLASH , directly by user program, up to a maximum size of **7,75K** Bytes.
- Protections** and **redundant controls** on the FLASH content that ensure the validity of the saved information (either as code and data) in any working conditions.

- Integrated **test** of the **SRAM** memory always installed, easily recallable by proper command.
- Perfectly **integrable** inside the development tools for I51 family, as the **BASCOM 8051** (BASIC compiler), **µC/51** (C compiler), **SYS51PW** (PASCAL compiler), etc.
- The user application program that is developed in conjunction with **FMO53**, requires very few interventions:

- define the **start address** of the **code** and adjacent interrupt vectors;

- define the **start address** of the **data** used by the program (variables).

By using the development tools described in previous point, these settings coincide with two compiler directive, that must be added at the **source** or **project** file of the same program.

- For the development tools suggested by **grifo®** the user found a complete and detailed description of both the **configurations** required for the **integration** and the **directive** to prepare the application program.

- Wide **documentation** and rich list of **examples** both in **source** and **executable** format. These examples coincide with those developed for the control cards, are coded in different programming language and they are ready to be uploaded and executed.

- The **FMO53** is composed by a **CD** that contains the software, user manual and examples, and by an **EPROM** mounted on the used card.

- It has **No License** nor additional costs. The user is free to develop all the applications that he requires.

## REQUIREMENTS

Following is reported a short description of the necessary material (hardware and software), to operate with **FMO53**:

### CONTROL BOARD

This one meet the control board concerned in the industrial **grifo®** cards, based on family 51 microprocessors with an external memory as: **GPC® R63; GPC® T63; GPC® R63D; GPC® T63D; GPC® 323; GPC® 323D; GPC® 324; GPC® 324D; GPC® 550; GPC® 552; GPC® 553; GPC® 554;** etc.

The control board, independently from the application to realize, has to be composed by:

- at least 32K Byte of SRAM
- one asynchronous serial line in RS 232
- one EPROM with the following items:

**FMO53**  
**Ver. ??**  
**GPC xxxx**

where:           xxx =    board code  
                  ??.? =   program version

The features indicated above are the minimal structure for work, in fact the same system can be enlarged adding some potentiality. The selection of the configuration of the control board has to be started in according with specific applications to be developed. Among these options we can remind:

- FLASH EPROM Atmel:                   29c256 of 32K Bytes (**.32KF**)  
  29c010 of 128K Bytes (**.128KF**)

for save the program on board.

- SRAM:                                    from 32K Bytes (**.32K**)  
  to 128K Bytes (**.128K**)

to develop very long programs or with big data areas.

In case of purchase of control board, **FMO53** and potential options, the memory devices are provided already installed on the board.

### PERSONAL COMPUTER

The **FMO53** bundle need a un personal computer, that can be called **development PC**, with the following features:

*Personal Computer:*   IBM compatible (with CPU ≥ 486).  
*RAM Memory:*           Minimum 640K Bytes.  
*Operating System:*    MS-DOS o WINDOWS 95, 98, ME, 2000, NT, XP  
*Mass memories:*       Lettore CD-ROM  
                                  Hard Disk with 2 MByte free.  
*Seriale:*                 RS 232

## SERIAL COMMUNICATION CABLE

For all **FMO53** phases is necessary to perform a serial connection among one of the serial lines of the development PC and one of the serial lines of the control board. That connection need only the reception, transmission and ground signals (RxD, TxD and GND) following the V24 rules of C.C.I.T.T.

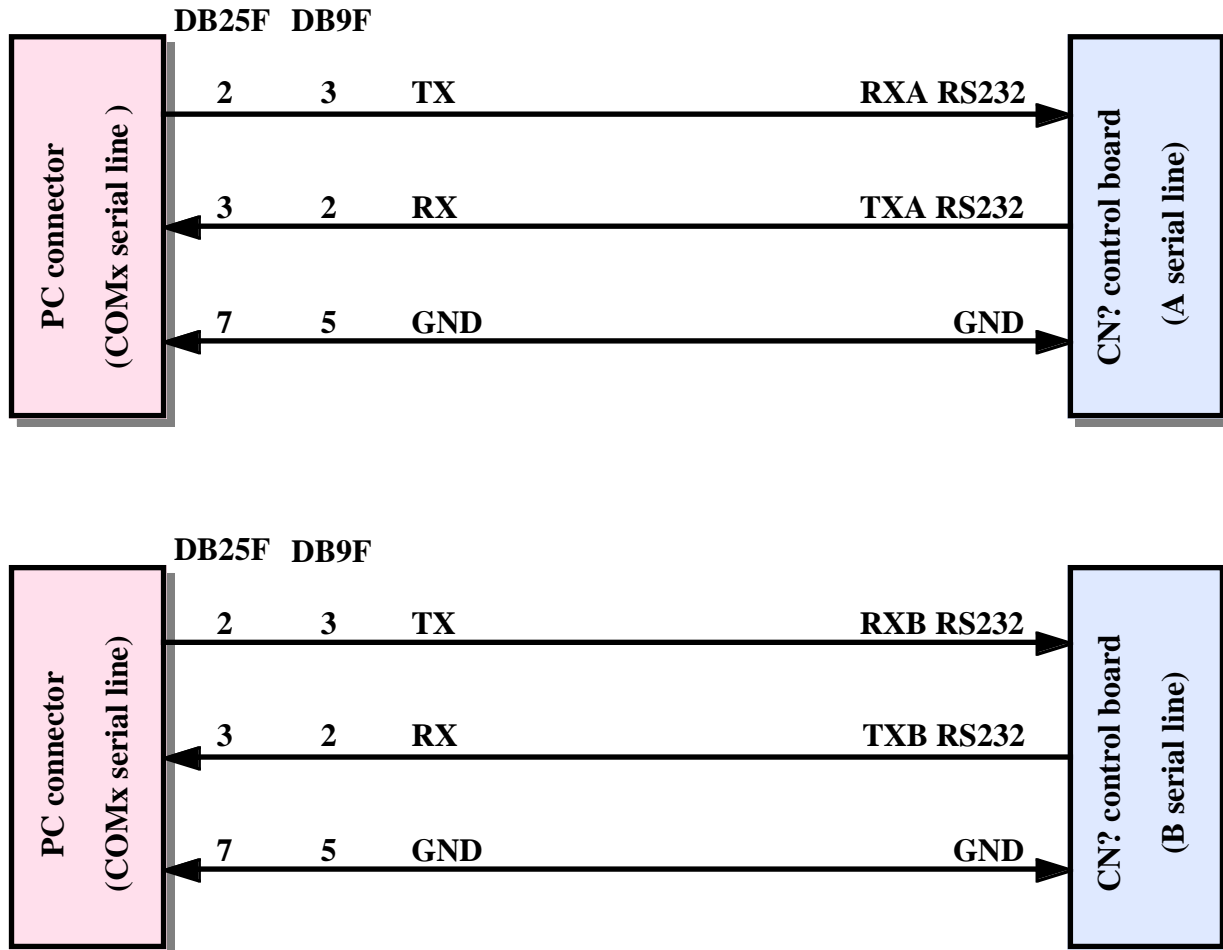


FIGURE 1: SERIAL COMMUNICATION BETWEEN DEVELOPMENT PC AND CONTROL BOARD

Figure 1 shows the connection between serial connectors of a PC and present connector on control board: the contact numbers are signalled in the connector because the numbers can change when changing the connector.

In the side of PC, when no serial communication line are available, we can use proper converter that once added to PC can communicate with the RS 232 serial reached. Among them we can find the USB<->RS232 converter, ETHERNET<->RS232 converter, multi I/O boards with RS232 adding, etc. Naturally **FMO53** can use these devices if they are correctly installed by hardware and by software.

In the control board side, as shown in the previous figure, user can configure which line would like to use when the board has got at least two serial. This feature can develop the application when the serial is busy too (for example for network connection, with modems, with printers, or other devices, etc) and is described SERIAL LINE SELECTION.

To make easier the connection phase and to not has to build a connection cable, **grifo**® can offer to You serial communication cables and accessories ready to use for any choice and any PC;

Control board	Serial	CN?	accessory code	Serial leads code
GPC® R63 GPC® T63 GPC® R63D GPC® T63D	A	CN3	MSI 01; AMP8.Cable	CCR.PLUG25F or CCR.PLUG9F
GPC® 323 GPC® 323D	A	CN3A	-	CCR.PLUG25F or CCR.PLUG9F
	B	CN3B	-	CCR.PLUG25F or CCR.PLUG9F
GPC® 324	A	CN3A	-	CCR.PLUG25F or CCR.PLUG9F
GPC® 324D	A	CN3A	-	CCR.PLUG25F or CCR.PLUG9F
	B	CN3B	-	CCR.PLUG25F o CCR.PLUG9F
GPC® 550	A	CN4A	-	CCR.PLUG25F or CCR.PLUG9F
	B	CN4B	-	CCR.PLUG25F or CCR.PLUG9F
GPC® 552	A	CN7	-	CCR.9+9R
	B	CN5	-	-
GPC® 553	A	CN3A	-	CCR.PLUG25F or CCR.PLUG9F
	B	CN3B	-	CCR.PLUG25F or CCR.PLUG9F
GPC® 554	A	CN3A	-	CCR.PLUG25F or CCR.PLUG9F
	B	CN3B	-	CCR.PLUG25F or CCR.PLUG9F

FIGURE 2: CONNECTOR AND ACCESSORIES FOR SERIAL COMMUNICATION

The table describes for all control board, the connectors with thier available serial line, the potential accessories and the serial cable that perform the connection directly or indirectly through accessory. The final numbers of the cable indicate if the connector female for PC is 25 or 9 ways.

**SOFTWARE OF WORK**

In order to work with **FMO53** and the hardware described before, is necessary a software for develop the application program. This software is divided into proper bundles and can be described as follow.



## APPLICATION PROGRAM DEVELOPMENT TOOL

Before the transfer and then saved the application program to the control board, has to be generated. For this purpose we can use the proper development tools that can allow to write the program on the development PC and to convert it in machine code used from **FMO53**.

So the board can use all the software resources of the mounted microcontroller, that are several bundles for I51 family, both in high and in low level. All these bundles are provided from **grifo®** and there are examples, in source and executable format, ready to use with **FMO53**. Among them we remind:

**HI-TECH C**: cross compiler for C source program. It is a powerful software tool that includes editor, C compiler, assembler, optimizer, linker, library, and remote symbolic debugger, in one easy to use integrated development environment.

**SYS51CW**: cross compiler for C source program, executable on P.C. with WINDOWS and with a comfortable IDE You can use: editor, C compiler, assembler, optimizer, linker, library of a remote symbolic debugger.

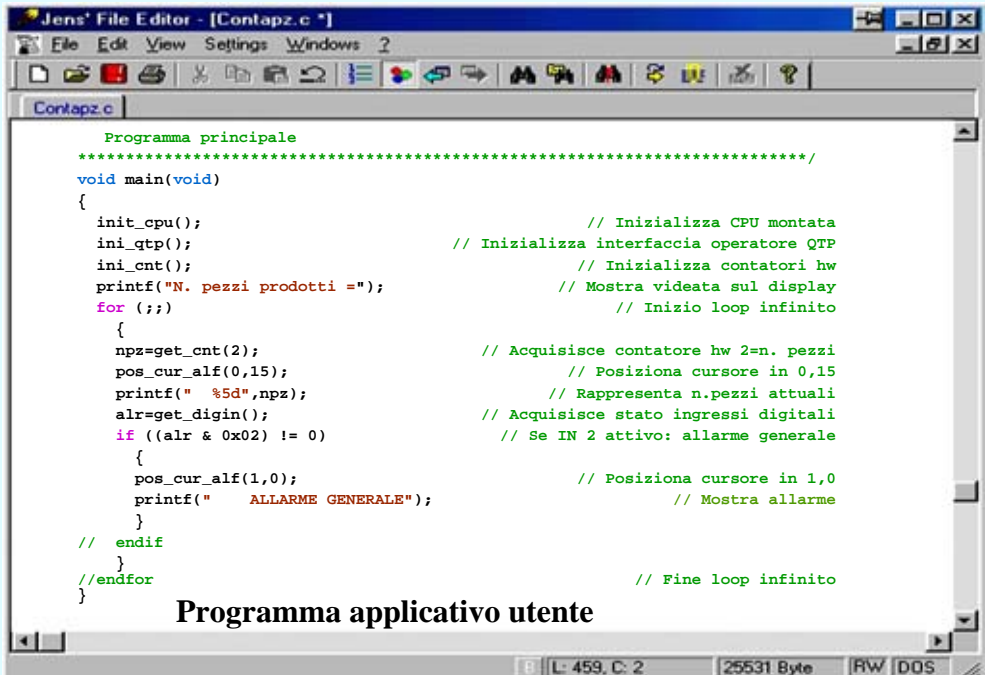
**SYS51PW**: cross compiler for C source program, executable on P.C. with WINDOWS and with a comfortable IDE You can use: editor, PASCAL compiler, assembler, optimizer, linker, library of a remote symbolic debugger.

**DDS MICRO C 51**: low cost cross compiler for C source program. It is a powerful software tool that includes editor, C compiler (integer), assembler, optimizer, linker, library, and remote symbolic debugger, in one easy to use integrated development environment. There are also included the library sources and many utilities.

**BASCOM 8051**: cross compiler for BASIC source program. It is a powerful software tool that includes editor, BASIC compiler and simulator included in an easy to use integrated development environment for Windows. Many memory models, data types and direct use of hardware resource instructions are available.

**μC/51**: it is a comfortable, low cost, software package with a complete IDE that allows to use an editor, and ANSI C compiler, and assembler, a linker and a remote source level debugger user configurable. Sources of main libraries and remote debugger are included, and so several utility and demo programs.

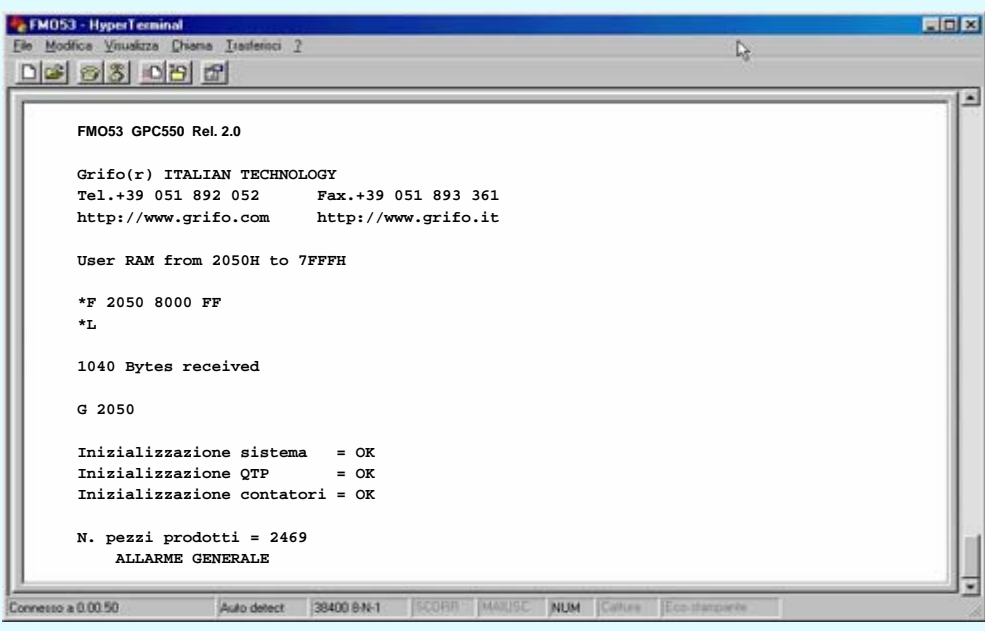
**LADDERWORK**: it is a easy to use system to generate automation application using the very famous contact logic. It features a graphic editor to place and connect components that refer to hardware resources (like digital I.O, counters, A/D, etc.) like on an electric diagram and define their properties, and efficient compiler to create the executable code and an utility to download it. Integrated IDE makes comfortable use of these tools. Delivered in a CD for Windows with user manual and hardware key.



**Programma applicativo utente**

**Development tool**

+



**Serial emulation terminal**

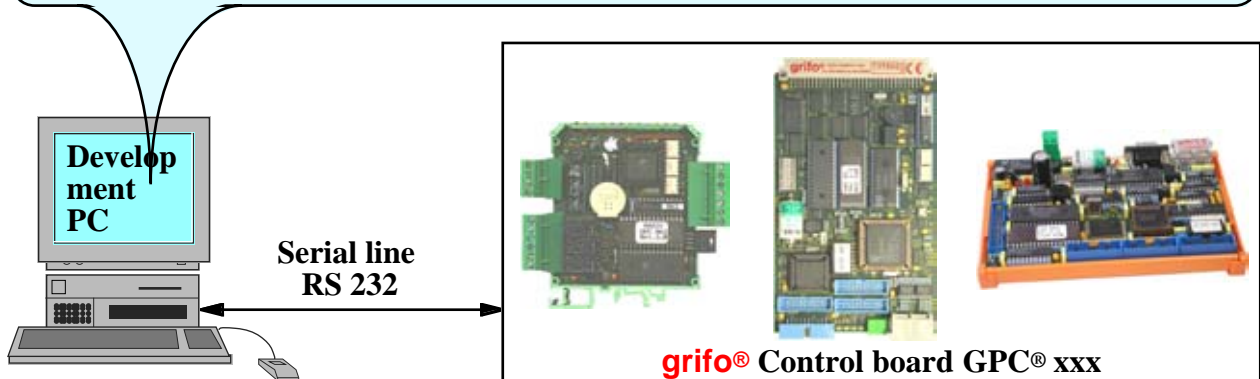


FIGURE 3: WORKING MODALITY



## SERIAL TERMINAL EMULATION PROGRAM

This program meet a generic communication program able to manage a classical terminal emulation, with a settable communication physical protocol. That program is a part of the **FMO53** and has to show on monitor everything is received in the serial line, transmit on serial the buttons pressed on the keyboard and at the end to transmit one file saved on the development PC.

In other words the development PC works as the console for the control board, so the usec can use the all properties of **FMO53**.

Remind that the **GET51** assembled in **grifo®**, **HYPERTERMINAL** of Windows operating system, or other programs as **CROSS TALK**, **PROCOMM**, **BITCOMM**, **TERMINAL**, etc, created by third party.

## EPROM PROGRAMMER

An EPROM programmer able to program the present files on the development PC is needed to complete the created application. In fact the generated code, once debugged and completely tested, has to be permanetely saved EPROM installed on the control board.

Remind that the EPROM programmer only if the used board is not working with FLASH EPROM (.32KF or .128KF). In fact in this last case the saving on the same FLASH is performed by the control board, through the development PC and a proper command of **FMO53**.

## FMO 53 DESCRIPTION

In this chapter are described all **FMO53** properties for using it on the control board. If the user wants to be faster, we recommend to follow the steps of HOW TO START chapter.

### CONFIGURATION BOARD

To use all the **FMO53** properties, the control board has to be properly configured. That configuration is referred to the board memory mapping, for the all downloading, testing and saving operations of the application program.

Generally **FMO53** has to be able to read and write both the code area and data area addressed by microcontroller, because of the 8051 structure do not allow to write the code area, the board has to put on the two areas. In other words, **FMO53** can download the code, manage the stop points, execute the application program, test the memory, etc, only if the SRAM of the board is addressed both as data and as code (/PSEN and /RD signals of microcontoller added).

In details this operating condition is obtained in a different mode depending on the used control board, and physically meets the installation of the proper memory components and the proper configuration of jumpers and/or dip switch. As described in the CONTROL BOARD paragraph of the REQUIREMENTS chapter if **FMO53** is ordered with the board, the user has to receive the hardware already configured and ready to use. To obtain that condition it has to specify in the order the **.FMO53** code, as board name extension, potentially followed from the memory option codes. For example, to receive a **GPC 324@D** with **FMO53** and 32K of FLASH ready to use, it has to specify the code:

**GPC® 324D.FMO53.32KF**

Viceversa in case of separated order, the tables of figure 4 and 5 show how to install the memories and configurate the board, in order to use correctly the FMO53. In these tables are reported several information, described below:

- Board	indicates the board involved in the instruction.
- Memory Mapping	indicates the name of memory mapping, as reported in the technical manual of the board.
- Mapping selection	indicates which jumpers and/or dip switch has to be moved to obtain the mapping.
- AUTORUN	indicates the setting in order to automatically start at the beginning the user program, saved in FLASH
- EPROM FMO53	indicates the socket for FMO53 EPROM installation
- SRAM default	indicates the socket where to insert SRAM (if not present)
- Multimemory socket	indicates the socket for the optional or adding memory installation
- Jumper memory	reports the configuration of the jumpers for described memories

To identify the location of jumpers and sockets described in these tables, please, see the proper figures of the technical manual of the board.

Board	Memories Mapping	Mapping Selection	AUTO RUN	EPROM FMO53	SRAM base	Multim. socket	Memories Jumpers
<b>GPC® R63</b> <b>GPC® T63</b> <b>GPC® R63D</b> <b>GPC® T63D</b> Rel 101197	MODE 3	J2 closed J3 in 2-3	J1 closed	IC 7	IC12 installed	SRAM (.32K) IC 10	J4 in 2-3 J5 in 2-3
						FLASH (.32KF) IC 10	J4 in 1-2 pin 3 of J4 with pin 2 of J5
<b>GPC® R63</b> <b>GPC® R63D</b> Rel ≥ 110901	MODE 3	J2 closed J3 in 2-3	J1 closed	IC 7	IC12 installed	SRAM (.32K) IC 10	J5 in 2-3 and 4-5
						FLASH (.32KF) IC 10	J5 in 1-2 and 3-4
<b>GPC® 323</b> <b>GPC® 323D</b> Rel 110197	MODE 1	DIP 5 ON DIP 6 ON DIP 7 OFF	DIP 8 ON	IC 5	IC 4	SRAM (.32K) IC 3	J2 in 2-3 J3 in 2-3 J4 and J5 in 1-2
						FLASH (.32KF) IC 3	J2 in 1-2 pin 3 of J2 with pin 2 of J3 J4 and J5 in 1-2
<b>GPC® 323</b> <b>GPC® 323D</b> Rel ≥ 250601	MODE 3	DIP 5 ON DIP 6 ON DIP 7 OFF	DIP 8 ON	IC 5	IC 4	SRAM (.32K) IC 3	J2 in 2-3 and 4-5
						FLASH (.32KF) IC 3	J2 in 1-2 and 3-4
<b>GPC® 324</b> <b>GPC® 324D</b> Rel 100997	MODE 3	J2 closed J3 in 2-3	J1 closed	IC 8	IC7 installed	SRAM (.32K) IC 5	J4 in 2-3 J5 in 2-3 J6 in 1-2
						FLASH (.32KF) IC 5	J4 in 1-2 pin 3 of J4 with pin 2 of J5 J6 in 1-2
<b>GPC® 324</b> <b>GPC® 324D</b> Rel ≥ 110400	MODE 3	J2 closed J3 in 2-3	J1 closed	IC 8	IC7 installed	SRAM (.32K) IC 5	J4 in 2-3 and 4-5
						FLASH (.32KF) IC 5	J4 in 1-2 and 3-4

FIGURE 4: BOARD CONFIGURATION (1 OF 2)

Board	Memories Mapping	Mapping selection	AUTO RUN	EPROM FMO53	SRAM default	Multim. socket	Memories Jumpers
<b>GPC® 550</b> Rel 200702	MODE 3	J4 in 2-3 J5 closed	DIP 8 ON	U1	U2 installed	SRAM (.32K) U3	J1 in 2-3 and 4-5
						FLASH (.32KF) U3	J1 in 1-2 and 3-4
<b>GPC® 550</b> Rel $\geq 120605$	MODE 3	J1 closed J2 closed J12 open	DIP 8 ON	IC19	IC7 installed	SRAM (.32K) IC 18	J3 in 1-2 J7 in 2-4 e 5-6
						SRAM (.128K) IC 18	J3 in 1-2 J7 in 2-4;5-6;7-8
						FLASH (.32KF) IC 18	J3 in 1-2 J7 in 1-2;4-5;6-8
						FLASH (.128KF) IC 18	J3 in 1-2 J7 in 2-3;4-5;6-8
<b>GPC® 552</b> Rel $\geq 180796$	MODE 1	J1 in 3 J2 in 3 J14 (1,2,3) in ASM	DIP 8 ON	IC 15	IC 13	SRAM (.32K) IC 12	J3 and J4 in 1-2 J17 in 2-3 J18 in 2-3
						FLASH (.32KF) IC 12	J3 and J4 in 1-2 pin 2 di J17 with pin 3 of J18 J18 in 1-2
<b>GPC® 553</b> Rel $\geq 070198$	MODE 3	DIP 5 ON DIP 6 ON J7 open	DIP 8 ON	IC 1	IC 2	SRAM (.32K) IC 3	J2 in 2-3 J3 in 2-3 J4 and J5 in 1-2
						FLASH (.32KF) IC 3	J2 in 1-2 pin 2 of J3 with pin 3 of J2 J4 and J5 in 1-2
<b>GPC® 554</b> Rel $\geq 100997$	MODE 3	J1 closed J6 in 2-3	J2 chiuso	IC 4	IC8 installed	SRAM (.32K) IC 6	J3 in 2-3 J4 in 2-3 J5 in 1-2
						FLASH (.32KF) IC 6	J3 in 1-2 pin 3 of J3 with pin 2 of J4 J5 in 1-2

FIGURE 5: BOARD CONFIGURATION (2 OF 2)



Once the board is configured, as previous indications, **FMO53** manages the available memory resources in two different modalities, as shown in the following figures.

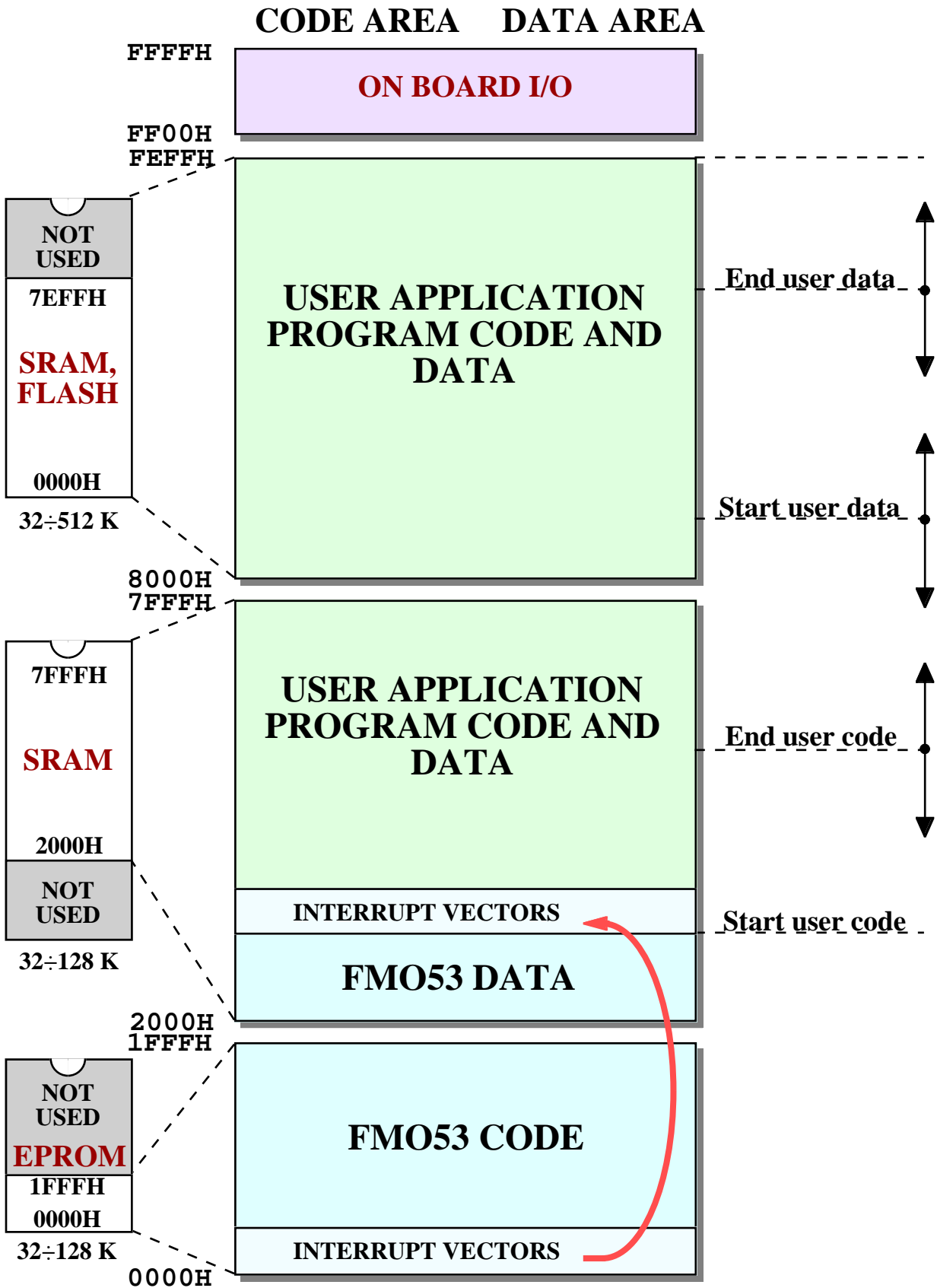


FIGURE 6: MEMORY ORGANIZATION IN DEBUG MODE

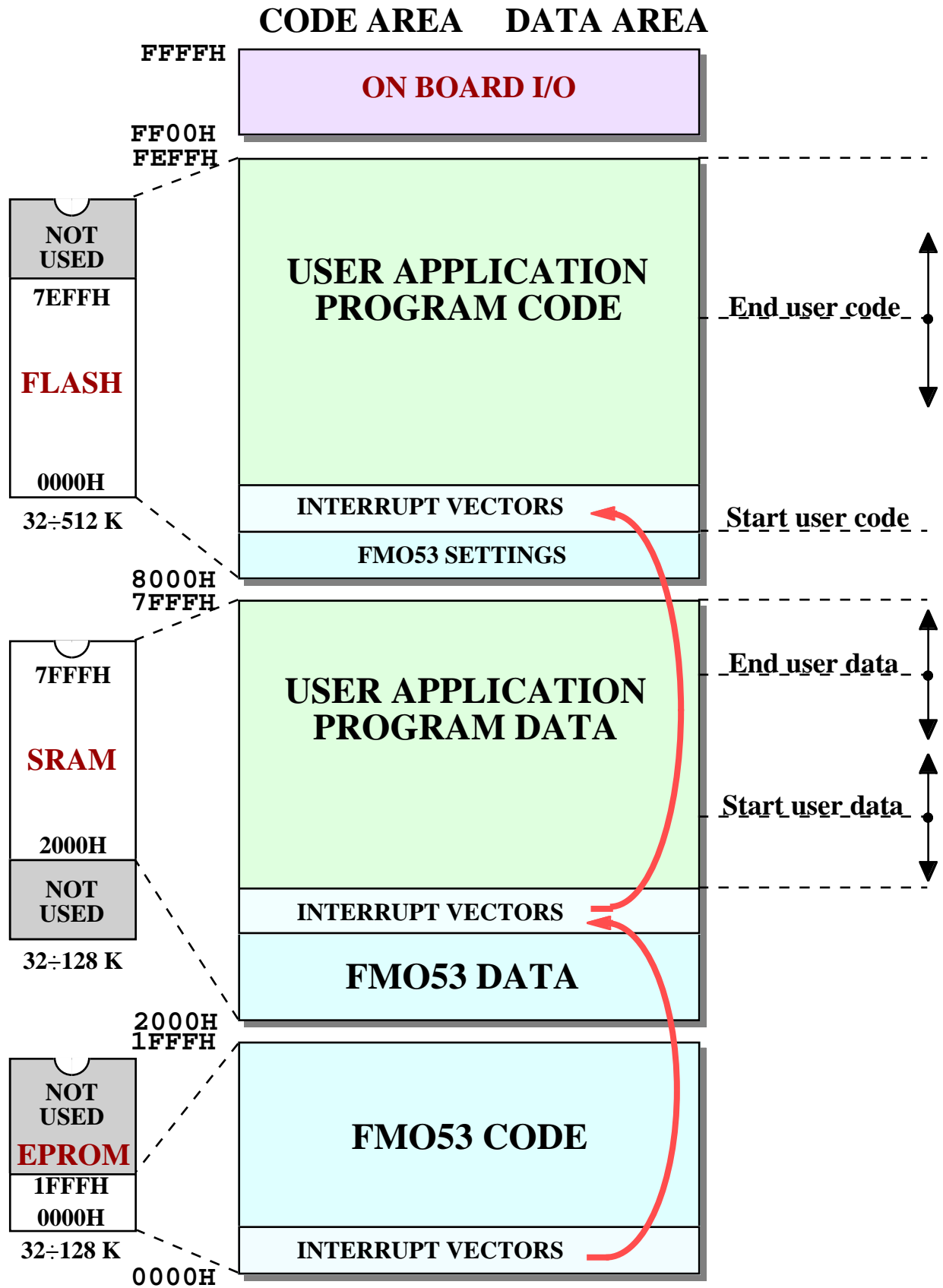


FIGURE 7: MEMORY ORGANIZATION IN AUTORUN MODE

The modalities DEBUG and AUTORUN are the operative conditions that **FMO53** selects after each start, as described in the following OPERATIVE MODE SELECTION paragraph.

The figures 6 and 7 report all the needed information to develop the proper application program in according with **FMO53**, in fact show the address of start and end of the code and data available from user.

In the same figures we can do the following affirmations:

- a) Several indications are generic because they change themselves depending on operative conditions; to have real values for addresses we recommend to read the USER APPLICATION PROGRAM CONFIGURATION paragraph and the figures 9 and 10.
- b) The double vertical indicators, indicates that the same address can change from the other inferior up to the next superior extreme.
- c) The EPROM memory addressed from 0000÷1FFFH meets the SRAM column of EPROM FMO53 of the tables 4 and 5.
- d) The SRAM memory addressed from 2000÷7FFFH meets the SRAM default column of tables of figure 4 and 5.
- e) The SRAM or FLASH memory addressed from 8000÷FEFFH meets the multifunction socket column of tables of figure 4 and 5.
- f) The FLASH memory can be written only through the proper command or procedure of FMO53, but is not managed neither from microcontroller or the user program.
- g) The user application program can be executed and debugged only if saved on SRAM memory.

## USED RESOURCES

To develop the proper work, the **FMO53** uses a few amount of resources of the board that execute it, as listed below:

- **6 Bytes** of **internal RAM** (08H÷0DH) for its variables of work;
- **208 Bytes** of **external SRAM** to save the microcontroller status, stop points, etc.;
- **8K Bytes** of **EPROM** for its code;
- **20 Bytes** of **stack**;
- **80 Bytes** of **FLASH** for AUTORUN setting and validity check of user program;
- one **serial line RS232** for communication to development PC;
- **Timer1** for the serial communication and the step to step execution;
- **interrupt Timer1** for step to step execution;
- **interrupt Timer1** and **/INT0** for the communication with software B serial;
- an hardware **selector** for selecting start modo;
- a **configuration** for the used serial selection.

During the FLASH programming through the user program, are used more resources of internal RAM, external SRAM and the interrupt of used hardware serial. That programming operation meets the final phase of the development process of user program, so the user can be don't care of this resources using.

Remind that all resources are completely free when the program saved in FLASH is in execution; so the user can work as if his program is alone.

**SERIAL COMMUNICATION**

The relationship between **FMO53** and user is completely based on asynchronous serial communication that perform between the control board and the development PC. In the SERIAL COMMUNICATION CABLE paragraph is described the electrical connection between the two systems when in this system are reported the rest of other information, referred to serial communication.

The physical protocol used by **FMO53**, that has to be set in the PC too, is the following:

- Baud Rate = 38400
- Bit per character = 8
- Parity = None
- Stop Bit = 1

In the control board side, the user can select which line want to use when the board has at least two serials. This features allows to develop the application too that has the serial busy ( for example for network connections, to modems, to printers, to other devices, etc.). Moreover allows to use the development PC connected in RS232 when the remaining board line too is configured in RS422, RS485 or Current Loop, without using expensive converters or perform changing in configuration. The choosing of the serial line to use is performed by a proper hardware condition on control board, that are verified in corrispondence to the **FMO53** start, that is after one reset or one power on. That condition change at board changing, as described below:

Control board	Serial	CN?	accessory code	Serial leads code
<b>GPC® R63</b> <b>GPC® T63</b> <b>GPC® R63D</b> <b>GPC® T63D</b>	A	CN3	MSI 01; AMP8.Cable	CCR.PLUG25F or CCR.PLUG9F
<b>GPC® 323</b> <b>GPC® 323D</b>	A	CN3A	-	CCR.PLUG25F or CCR.PLUG9F
	B	CN3B	-	CCR.PLUG25F or CCR.PLUG9F
<b>GPC® 324</b>	A	CN3A	-	CCR.PLUG25F or CCR.PLUG9F
<b>GPC® 324D</b>	A	CN3A	-	CCR.PLUG25F or CCR.PLUG9F
	B	CN3B	-	CCR.PLUG25F o CCR.PLUG9F
<b>GPC® 550</b>	A	CN4A	-	CCR.PLUG25F or CCR.PLUG9F
	B	CN4B	-	CCR.PLUG25F or CCR.PLUG9F
<b>GPC® 552</b>	A	CN7	-	CCR.9+9R
	B	CN5	-	-
<b>GPC® 553</b>	A	CN3A	-	CCR.PLUG25F or CCR.PLUG9F
	B	CN3B	-	CCR.PLUG25F or CCR.PLUG9F
<b>GPC® 554</b>	A	CN3A	-	CCR.PLUG25F or CCR.PLUG9F
	B	CN3B	-	CCR.PLUG25F or CCR.PLUG9F

**FIGURE 8: USED SERIAL LINE SELECTION**

On the board which is not reported the condition, the **FMO53** communicate always and only with serial line A. The table reports the configurations too that allows to work the selected serial line: the user has to perform these configurations with the technical manual of the board.



## USER APPLICATION PROGRAM CONFIGURATION

In any application program we can always identify two separated memory spaces that are defined:

**code area** -> to program instruction code, to constants, messages, library, run time, etc. The microcontroller manages that area in read only.

**data area** -> to all program data so to variables, strings, buffers, etc. The microcontroller manages that area both in reading operation and in writing operation.

Beginning from figures 6 and 7, in this paragraph are described the configuration that the user has to perform on the proper application program, to make it compatible with **FMO53**.

Generally these actions are always needed:

- Define the address of start user code and for interrupt vectors.
- Verify that the code area size are not too large, that is the **End user code** address has a real value.
- Define the address of **Start user data**.
- Verify that the data area size are not too large, that is the **End user data** address has a real value.

While these expedients are recommended:

- If the application program preview a serial console on the same line connected to development PC, is recommended to set in the program the same physical protocol of **FMO53** (38400, 8, No, 1).
- During the development phase, if debug commands are used ( stop points, single step executions, memory test, etc.), is not recommended to use the resources used by **FMO53**.

The value of start and end addresses reported in previous description has different values depending on the selected operating mode, the selected communication line and the kind of memory installed on multimemory socket, as indicated into two following tables:

Used serial	Multim. socket	start user code	End user code	Start user data	End user data
A	SRAM (.32K)	2050H	< Start user data and $\leq$ FEFFH	End user code $\div$ FEFFH	$\leq$ FEFFH
	FLASH (.32KF)	2050H	< Start user data and $\leq$ 7FFFH	End user code $\div$ 7FFFH	$\leq$ 7FFFH
B	SRAM (.32K)	2100H	< Start user data and $\leq$ FEFFH	End user code $\div$ FEFFH	$\leq$ FEFFH
	FLASH (.32KF)	2100H	< Start user data and $\leq$ 7FFFH	End user code $\div$ 7FFFH	$\leq$ 7FFFH

**FIGURE 9: MEMORY AREAS ADDRESSES IN DEBUG MODE**

Used serial	Multim. socket	Start user code	End user code	Start user data	End user data
A	FLASH (.32KF)	8050H	≤ FEFFH	20C6H ÷ 7FFFH	≤ 7FFFH
B	FLASH (.32KF)	8050H	≤ FEFFH	2176H ÷ 7FFFH	≤ 7FFFH

FIGURE 10: MEMORY AREAS ADDRESSES IN AUTORUN MODE

As You can to note from both the tables, several addresses has not a defined value but a changing range.

In that case the user can define the favourit value to has a best way. For example with an application program that has a code length of 15K Bytes and a data length of 2K Bytes, is convenient to set the **User start data** = 7000Hin order to keep fix both in DEBUG and in AUTORUN mode and in order to leave free space both for code and for data.

Always examining the table of figure 9 and 10 we can obtain the maximum length size of code and data area of the application program, in according to **FMO53**.

Area	Maximum size in DEBUG mode	Maximum size in AUTORUN mode
User code	55,6K Bytes - User data (2050H ÷ FEFFH)	31,6K Bytes (8050H ÷ FEFFH)
User data	55,6K Bytes - User code (2050H ÷ FEFFH)	23,8K Bytes (20C6H ÷ 7FFFH)

FIGURE 11: MEMORY AREAS MAXIMUM SIZE

If the maximum size reported above are insufficient for the application to develop, we remind that all control boards have other maps that provide more memory. In these cases the user can continues to use **FMO53** in DEBUG mode to test the proper program and at the end to save it on EPROM or FLASH EPROM with an external programmer, without AUTORUN mode. For the details relative to these possibility we recommend to contact directly **grifo®** company.

The settings of addresses of code area start and end and data normally meet the **directives** to add into the **source** or the of the application program. Naturally those directives are different among them depending on the used development tool and often are indentified by name for the ROM and RAM start. For the proposed development tools by the user can find a complete description of all settings that configurates the user application program, in the INTEGRATION WITH DEVELOPMENT TOOL following chapter. Moreover these inputs are present in a lot of demo programs, provided both in source format and executable format, in order to immediately test it anf than re-used to realize the proper program.

## FLASH EPROM MANAGEMENT

One of the principal features of **FMO53** is its possibility to manage the potential FLASH EPROM present on the control board. In the previous parameters are described the devices marks, the order codes, the mapping and the location addresses, while in this one are described the memory management modalities.

First of all we remind that FLASH is divided in three regions:

- 8000H÷804FH -> reserved for AUTORUN settings and validity check space
- 8050H÷FEFFH -> user code space
- E000H÷FEFFH -> on FLASH data space

which the first two issues are directly managed by the commands of saving of application program, while the last one is managed by proper procedures of general purpose. On all parts there are protections and some redundant check in order to guarantee any operative conditions intent.

### CODE ON FLASH

To save the application program of the user in FLASH EPROM it has to follow these steps:

- C1) Generate the program with configuration for AUTORUN mode (please see the previous paragraph), so assign the 8050H value to **start user code** address.
- C2) Verify that the hardware configuration of the board is ok ( please see figures 4 and 5).
- C3) Enable the proper commands to FLASH with the command H.
- C4) Erase the previous content of FLASH with the command X.
- C5) Download the HEX Intel file or S Motorola obtained at the step C1 to the control board, through The command L.
- C6) Verify that **FMO53** shows a correct programming message for the activating of AUTORUN setting.

For further information we recommend to examine the commands description, in the proper chapter. If the application program to save in FLASH uses the data space, it has to verify that first one don't extend itself on the second one; in fact the two areas use a common addressing space.

## DATA ON FLASH

**FMO53** includes the procedure directly called from the user application program, that allows to save the values for data on FLASH. The typical addresses of this features are the permanent saving of parameters, work settings, production data, etc., that can be drawn and used from the same application program or from other programs.

The space for data on FLASH is limited to **7,75K Bytes**, is organized in 124 pages of 64 bytes and is available to saved code size on the same FLASH. From the reported addresses at the beginning of the page we can obtain the space for data on FLASH that can be used for the user code too.

To save data on FLASH the user application program it has to execute the following steps:

- D1) Verify that the hardware configuration of the board is correct ( please, see figures 4 and 5).
- D2) Set the input parameters of data writing procedure on FLASH, setting some proper locations in external SRAM, organized with the following correspondence:
- |             |    |   |
|-------------|----|---|
| 2001H÷2041H | -> | 64bytes that were memorized on FLASH  |
| 2041H       | -> | number of page to write (00H÷7BH)   |
| 2042H       | -> | check sum of all input parameters, obtained by the sum of all set parameters in the from 2001H÷2041H (64 bytes and page number) without curry, <u>increased of 1.</u> |
- For example, to reset all the 64 location in the page 8 of the data spece on FLASH, the parameters have to be set with the following values:
- |             |    |             |       |
|-------------|----|-------------|-------|
| 2001H÷2041H | -> | bytes       | = 00H |
| 2041H       | -> | page number | = 08H |
| 2042H       | -> | check sum   | = 09H |
- D3) Execute the procedure performing a call (LCALL) to address 1A00H and remembering that are used up to 15 bytes of stack.
- D4) Examine the result of the call to writing procedure, verifying the content of location 2041H, with the following correspondence:
- |     |    |                                 |
|-----|----|---------------------------------|
| 80H | -> | writing correctly executed      |
| AAH | -> | invalid page number             |
| BBH | -> | wrong checksum input parameters |
| CCH | -> | invalid FLASH size              |
| EEH | -> | FLASH malfunction               |

### Notes:

- The command of FLASH erasing, erase the whole device, so all data saved on FLASH were definitively lost
- The data writing procedure on FLASH can be used only with a 32K Bytes device ( .32K option), while with all the remaining size the procedure reports the error code CCH.
- Remind that the FLASH memory has a close number of writing; so the data to save have to be properly selected in order to not exceeded with the writing procedure.

## OPERATING MODE SELECTION

The **FMO53** foresees two operative modes, on purpose projected in order to cover every user need. The operative mode check to use is executed only at the start ( when you switch the card on or when you reset it), and at the end of the checking the **FMO53** enters the identified mode without any chance that the user can change it, for example with a command.

The operative mode features have been described several times in the previous paragraphes and figures, so in this paragraph we intend to describe the conditions which select them:

**DEBUG** This mode is chosen when in the FLASH EPROM it hasn't been saved an user application programme, or when the proper hardware selector is set on the homonym DEBUG condition.

In this operative mode the **FMO53** is always executed and the user has the chance to use all its functions, allowing it to develop a new programme, update an already installed one and see the functioning result , correct the possible errors, etc.

**AUTORUN** This mode is chosen when in the FLASH EPROM has been saved an user application programme and the proper hardware selector is set on the RUN homonym condition.

In this operative mode is automatically executed the application program previously saved in the FLASH, allowing the user to eliminate the developing pc and install the checking card on the system that you want to create.

The hardware selector, which has been previously described, changes when the used card changes as well, as it has been described in the AUTORUN paragraph, on the table number 4 and 5. On every card it has always chosen an easy to use selector ( Dip switch or jumper) in order to easily allow the user to change the operative mode.

## INTERRUPTS

The **FMO53** redirects all the interrupt vectors of the microcontoller from the address **Start user code** which changes when the selected operative mode changes, as it has been described in the figures 6,7,9,10. The redirection is necessary because the original vectors of the microcontroller are allocated, starting from 0000H where the EPROM of the **FMO53** resides and the choice to redirect it to this address, reflects the standard organization of the 8051, where the interrupt vectors follow the reset vector from where the user code starts.

Just to make an example the vector fr the Timer0 interrupt, originally allocated at the 000BH reset address, it is redirected to 205BH in case of DEBUG mode on the A serial, to 210BH in case of Debug mode on B serial and to 805BH in case of AUTORUN mode.

When the application user programme uses the interrupt it is enough allocate the enters to the answer modalities to the redirected vectors, while the answer modalities don't suffer any changing. In case of lack of time, it is important to remember that the redirection that it has been done can cause a delay during the esecution of the answer modality which is quantifiable in 2,2,  $\mu$ sec.

Normally, by using an high level developing environment ( C compilers, BASIC, etc.) the interrupt vector allocation is automatically done by the compiler and the programmer has no reason to be worried.

We also want to remember that in DEBUG modality some interrupts are used by the FMO53, as it has been described in the paragraph USED SOURCES.

## INTEGRATION WITH DEVELOPING ENVIRONMENT

In order to simplify and to speed up the use of the **FMO53**, you can normally integrate the same programme with the used developing environment, in order to obtain only a working structure. The integration can be easily made with most of the developing environments available on sale and when it is not possible you can always use the two structures separately, with a little complication during the use without harming the use benefits.

In general the **FMO53** can be integrated with all the pc programmes which are able to manage a serial terminal emulation with the physical protocol pointed out in the paragraph SERIAL COMMUNICATION, and download a file saved on the same pc or on the same serial line. In the following paragraphs is described the integration with some of these programmes, introducing the serial communication setting and the configurations for the generating the applicative programme for the developing environments provided with a compiler and only the setting of the communication for the programmes with an easy terminal emulation.

### USE WITH BASCOM 8051

**Bascom 8051** is a Basic cross compiler, cheap, efficient, easy to use, provided with a complete IDE (Integrated Development Environment). Which is available for every version of Windows and generate an executable code for microcontrollers of the "I51" family.

For further information about the general features of this product it is suggested to consult the technical paper, the handbook or the complete help on line. The info related to the integration with **FMO53** are reported instead in the following paragraphs.

### CONFIGURATIONS FOR SERIAL COMMUNICATION

One of the several functionality of the IDE **BASCOM 8051** coincides with a serial terminal emulator that turns in a window of Windows and it is totally configurable and adaptable to the **FMO53** specifics of functioning. Plus the terminal emulator is able to automatically download to the card the applicative programme code, which it has been obtained from the last compilation. In this way the **FMO53** can receive from the **BASCOM 8051** the downloading command of a HEX Intel file, the same file (or the last compiled programme) and the end downloading characters, simply by activating the voice of a proper menu.

In order to obtain this result you need to use the IDE of the **BASCOM 8051** in the following way:

- 1) Through the *Options / Communications* menu, select the COMx serial port in use on the pc, configure the parameters of the communication physical protocol of the **FMO53**, disable the handshake hardware, as it is illustrated in the figure 12 and push OK.

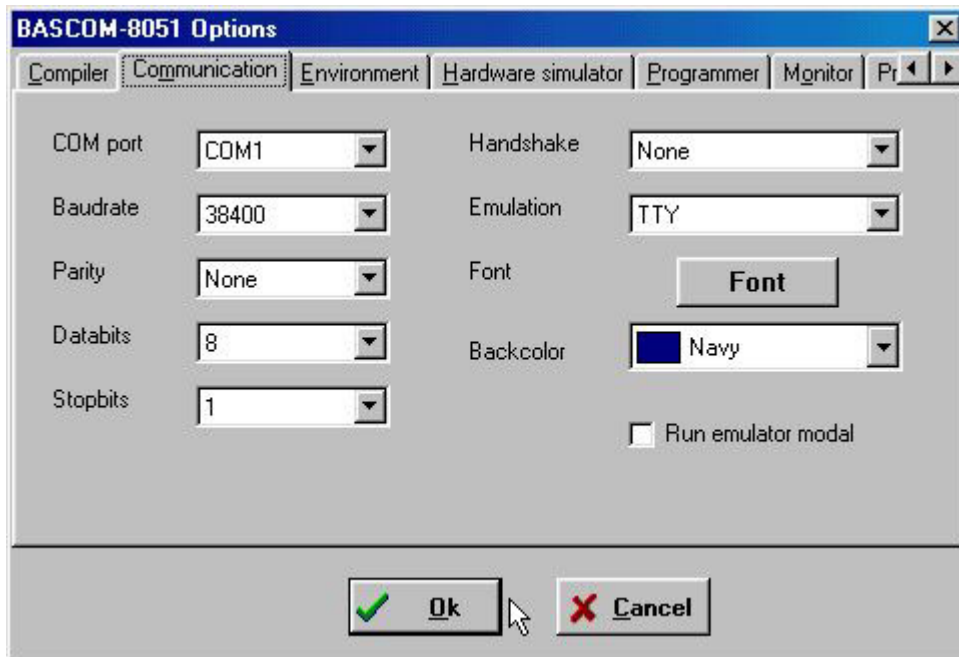


FIGURE 12: BASCOM 8051 EMULATION TERMINAL CONFIGURATION

2) Through the Options/Monitor menu, select the Hex Mon window and set on this last one the settings which are essential to the downloading of a HEX file to the **FMO53**, the relative communications delays, as it is illustrated in the figure 13, and push OK.

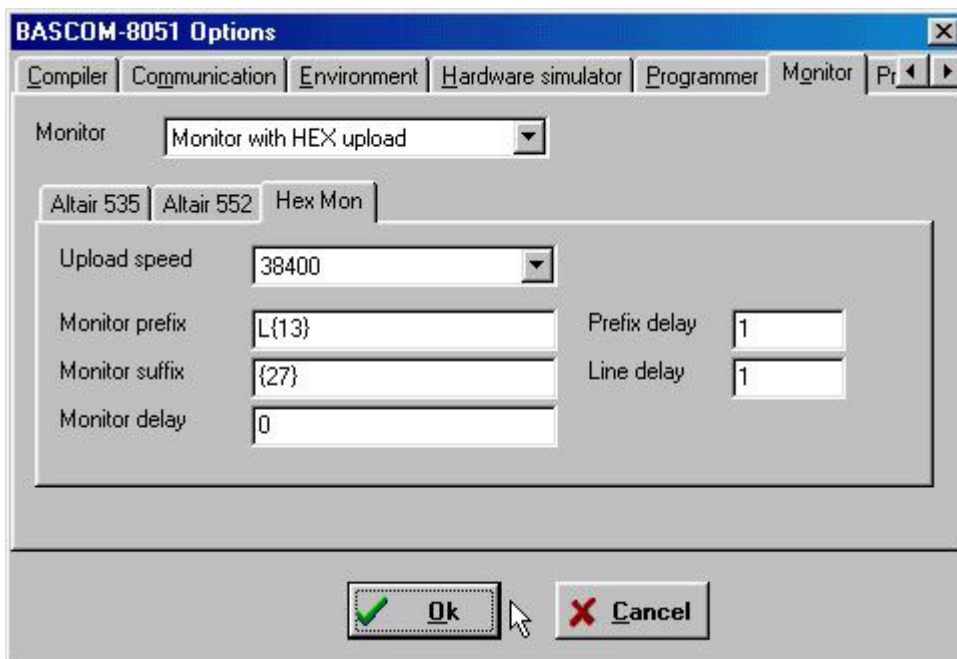
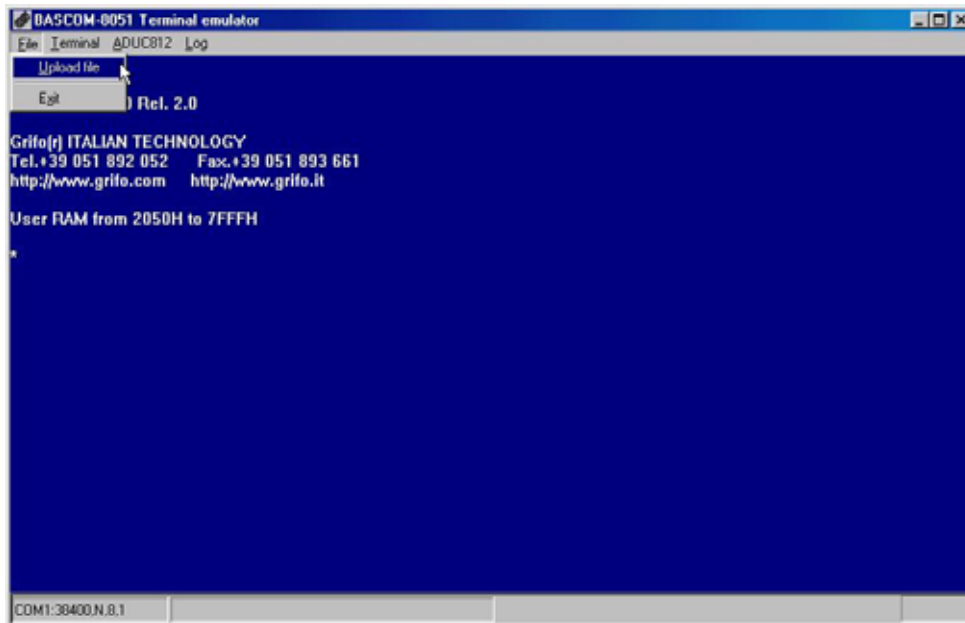


FIGURE 13: DOWNLOADING FILE CONFIGURATION WITH BASCOM 8051

3) Write the source program, or load it from the file, and compile it. For further information about how to compile a **BASCOM 8051** program, please read the following paragraph. We also want to remind you that every card by **grifo®** is supplied with a demo program which is arranged to be compiled with the **BASCOM 8051** that can be handy used in this phase.

- 4) Enter the terminal emulator by pushing the proper button on the buttons bar or through the Tools/ Terminal emulator menu or by pushing Ctrl t.
- 5) Feed or reset the control card: the main window of the figure 26 or 27 with the prompt\*. Repeat the first two points of that list if that window does not appear.
- 6) Activate the downloading of the HEX Intel file which is generated to the point 3, with the voice File /Upload file of the BASCOM-8051 Terminal Emulator (see figure 14).



**FIGURE 14: UPLOADING FILE WITH BASCOM 8051**

- 7) Wait for the end of the download that is pointed out by a proper scroll bar in the low corner on the right and at the end of the operation please verify that the esadecimal indication with the calculation of the number of downloaded bytes appears (it coincides with the length of the generated code, rounded up to a multiple of 16=10H).
- 8) Execute the downloaded program by digiting the command G < Start user code >, where **Start user code** correspond to the starting address of the downloaded applicative program, in esadecimal way (es.2050).

## APPLICATION PROGRAM INSTRUCTION

When a compilation of a **BASCOM 8051** source is executed, a HEX Intel file is generated; it contains the machine code of the microconroller that it has been used. Through some guidelines of the compiler, the user can configure the program so that it can be downloaded and executed with the **FMO53**, as we have described in the **APPLICATIVE USER CONFIGURATION PROGRAM** paragraph.

The **BASCOM 8051** applicative program instructions can be defined in proper IDE menu or directly added to the source. Between these two techniques the second one is for sure advantageous because it has priority on the first one and at the same time makes it soon available to every programmer and it works in every operative condition.

<i>Directive</i>	<i>Description</i>
\$baud	-> Indicates the rate baud of the serial communication of the program. If the program foresees a serial console it is suggested to set this console with the same physical protocol of the <b>FMO53</b> , in other words by using this directive to set a rate baud = 38400.
\$romstart	-> Indicates the address of the first code byte of the program, in other words it coincides with the direction <b>Start user code</b> , which is used in this manual. This directive also relocates the answer vectors to the interrupts.
\$ramstart	-> Indicates the address of the first byte of external SRAM available as data area of the program, that meet the <b>Start user data</b> indication, used in this manual.
\$ramsize	-> Indicates the size of the external SRAM which is usable as a data area and it coincides with the value of ( <b>End user code -Start user code</b> )
\$large	-> Indicates the compiler to generate a code that can be bigger than 2 k, in other words to address all the available memory.

These and other directives which are not useful for the **FMO53**, are described in the documentation of the **BASCOM 8051**.

As it is described in the previous chapter, the values of the directives change according to the operative conditions and here we want to make some example of assignation:

- Program in DEBUG mode on A serial:

```
$baud= 38400           'Serial Rate Buad hw
$romstart=&H2050       'Starting code and interrupt vectors in SRAM
$ramstart=&H7000       'It foresees 4k of data area by starting from 7000H
$ramsize=&H0FFF
$large                 'Code size >2k
```

- Program in DEBUG mode on B serial:

```
$romstart=&H2100       'Starting code and interrupt vectors in SRAM
$ramstart=&H7000       'It foresees 4k of data area by starting from 7000H
$ramsize=H0FFF
$large                 'Code size >2k
```

Program in AUTORUN mode:

```
$baud=38400           'Serial Rate Buad hw
$romstart=&H8050       'Starting code and interrupt vectors in FLASH
$ramstart=&H7000       'It foresees 4k of data area by starting from 7000H
$ramsize=&H0FFF
$large                 'Code size >2k
```

## USING FMO 53 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 53**, realized for the MS DOS but that can be used with Windows too up to the ME version.

This manual will report only the information useful for communication with **FMO 53**, for further information about **GET 51** please refer to its manual on **grifo®** CD or on **grifo®** Web Site at [www.grifo.com](http://www.grifo.com).

### CONFIGURATION FOR SERIAL COMMUNICATION

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 53**. This way, **FMO 53** 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:

- 1) Through the menu Options | Serial Port, selecting under the label 'COM port' the name of the PC serial port, and select the communication baud rate of **FMO53**, as shown in figure 15 then press OK.

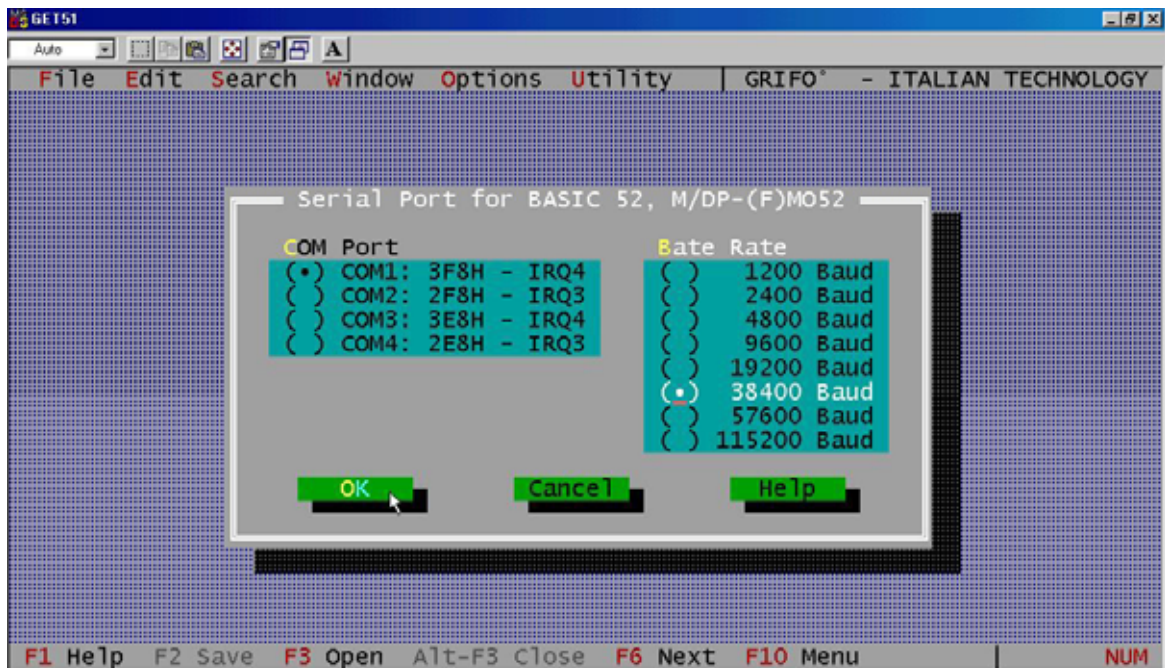


FIGURE 15: CONFIGURATION OF SERIAL PORT WITH GET 51

- 2) Configure the parameters for communication with **FMO 53** using the menu Options | Set terminal as shown in figure 16, then press OK.

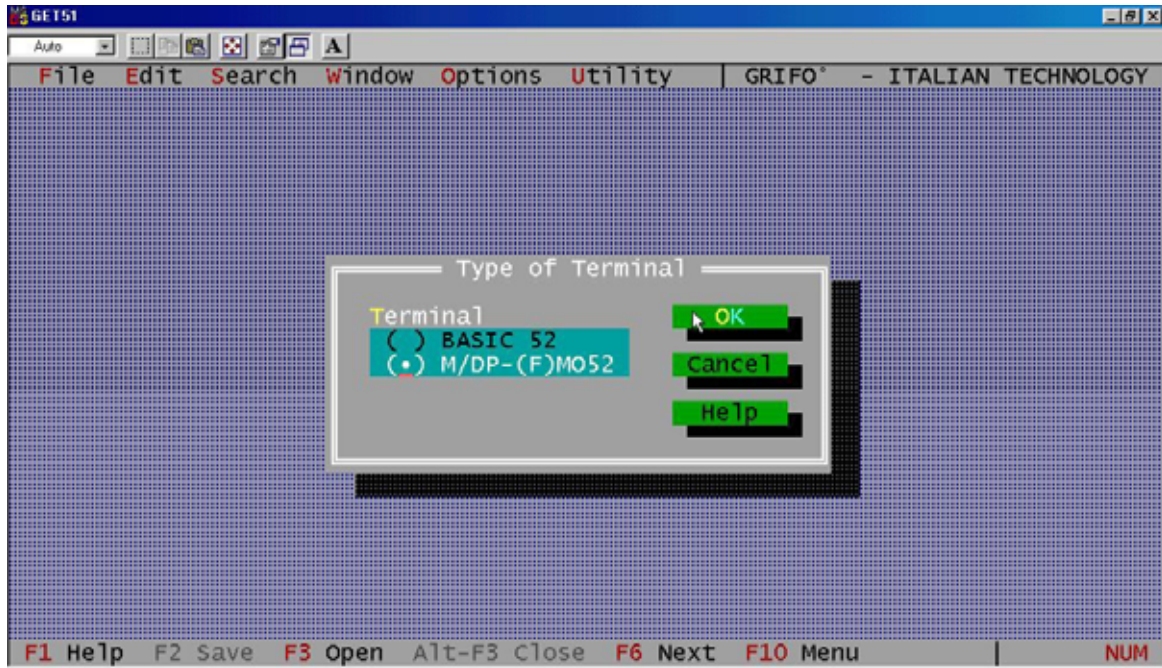


FIGURE 16: PARAMETER TO COMMUNICATE WITH FMO 52

- 3) Enter the terminal emulator by the menu Options | Terminal or pressing the combination of keys Alt+t.
- 4) Supply or reset the control board. You must see the starting screen shown on figure 26 or 27 with prompt '\*'. Repeat the first two steps of this list if you encounter problems.
- 5) To select the file to download with the issue Utility | Transmit file, or pressing F8. The window that appears ( see figure 17) 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 53.

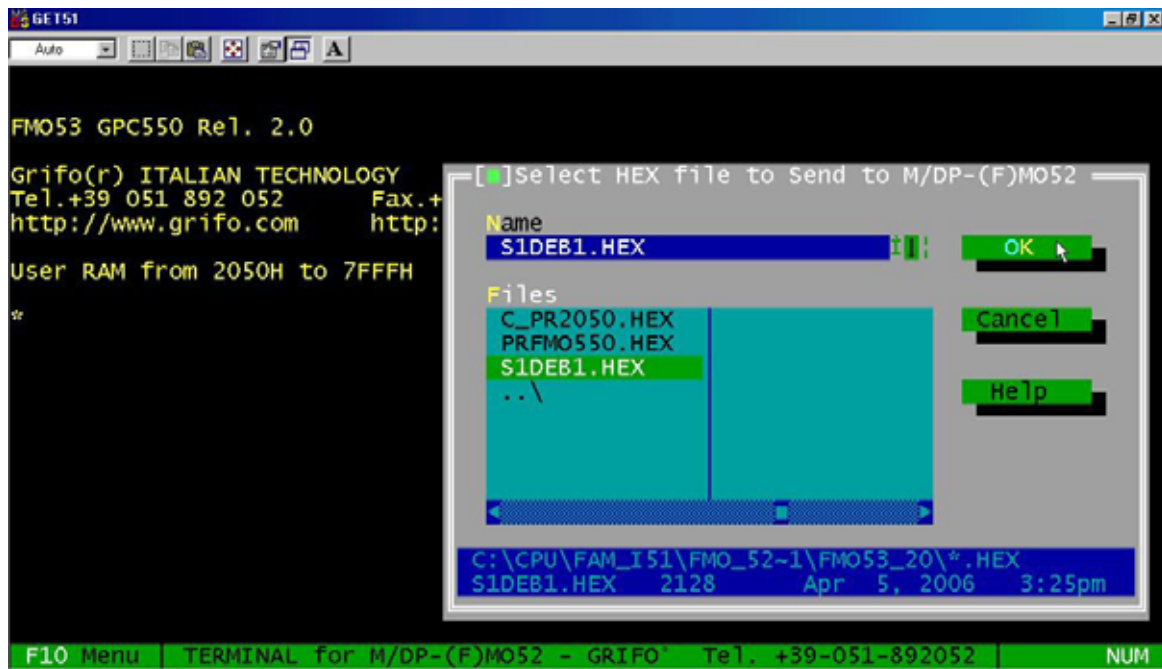


FIGURE 17: INTEL HEX FILE DOWNLOAD USING GET 51

- 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.

## USING WITH HYPERTERMINAL

**HYPERTERMINAL** is a famous communication program that supports the emulation terminal function too, compatible with **FMO53**. It is provided by Windows operating system, so available in all its version. For further information of general features of this product, we recommend to examine the relative help on line.

## CONFIGURATIO FOR SERIAL COMMUNICATION

Among the several functionality of **HYPERTERMINAL** we remind the serial terminal emulator that is completely configurable to the basic function of **FMO53**.

Moreover the program is able to download to the board, a code of a generic application program. In this way the **FMO53** can receive from **HYPERTERMINAL** the downloading command of Intel HEX file, the same file ( generated by another development tool) and the end downloading characters, in a semiautomatic modality.

To obtain this, **HYPERTERMINAL** must be configured as follows:

- 1) Start the program through the installation folder as: Start | Programs | Accessory | Communication | Hyperterminal. At this point is shown a window called connection description where we insert the **FMO53** name, as described in figure 18, and press OK.



FIGURE 18: **HYPERTERMINAL** COMMUNICATIO CONFIGURATION ( 1 OF 4)

2) In the following window *Connect* to select the option for the direct connection to used COM on PC, in according with the *connect* field, as described in figure 19, and confirm with *OK* button.



FIGURE 19: HYPERTERMINAL COMMUNICATIO CONFIGURATION ( 2 OF 4)

3) In the window *Property COMx*, configure the parameters of the communication physical protocol of FMO53, disable the hardware handshake, as shown in figure 20, and press *OK*.

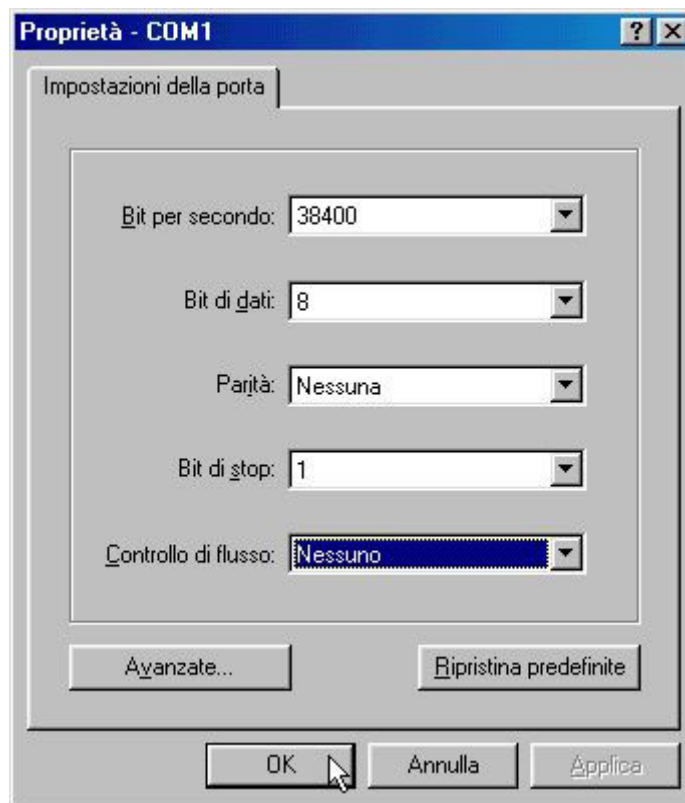


FIGURE 20: HYPERTERMINAL COMMUNICATIO CONFIGURATION ( 3 OF 4)

- 4) Save the performed configurations in a file in order to quickly recall during all the future using of **FMO53** with **HYPERTERMINAL** through the option *File | Save as...* . For example we recommend to call FMO53.HT the configuration file ( as shown in figure 21), and to use it for any execution, with a simple double click in its icon.

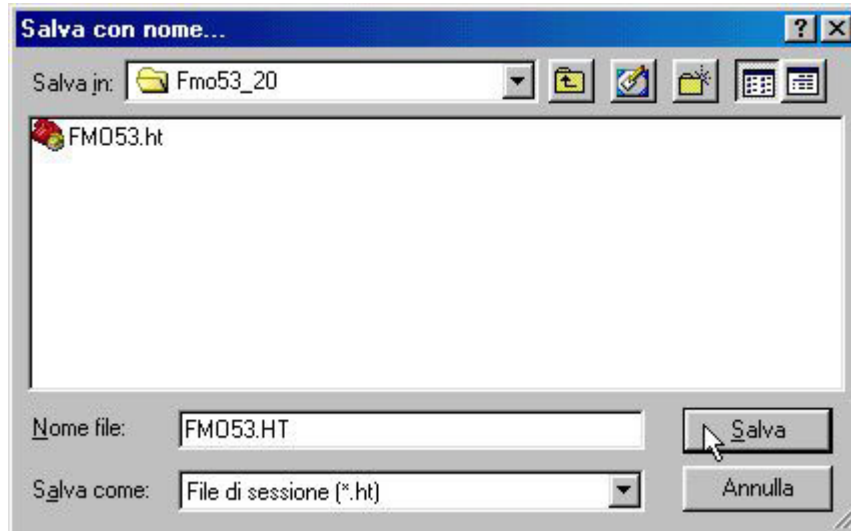


FIGURE 21: **HYPERTERMINAL** COMMUNICATIO CONFIGURATION ( 4 OF 4)

- 5) Switch on or reset the control board: in the window it has to show the initial list of figure 26 or 27 with the prompt\*. Repeat the first three point of this list if this one not appear.
- 6) Perform the downloading command pressing the sequence of keys **L** and **ENTER**.
- 7) Activate the downloading of Intel HEX file that can be saved on one of the disks of development PC, through the issue *Transfer | Send text file...* .The window that appear ( see figure 23) allow to search the wanted file on all disk of PC, to select it and confirm it pressing ENTER or OK.

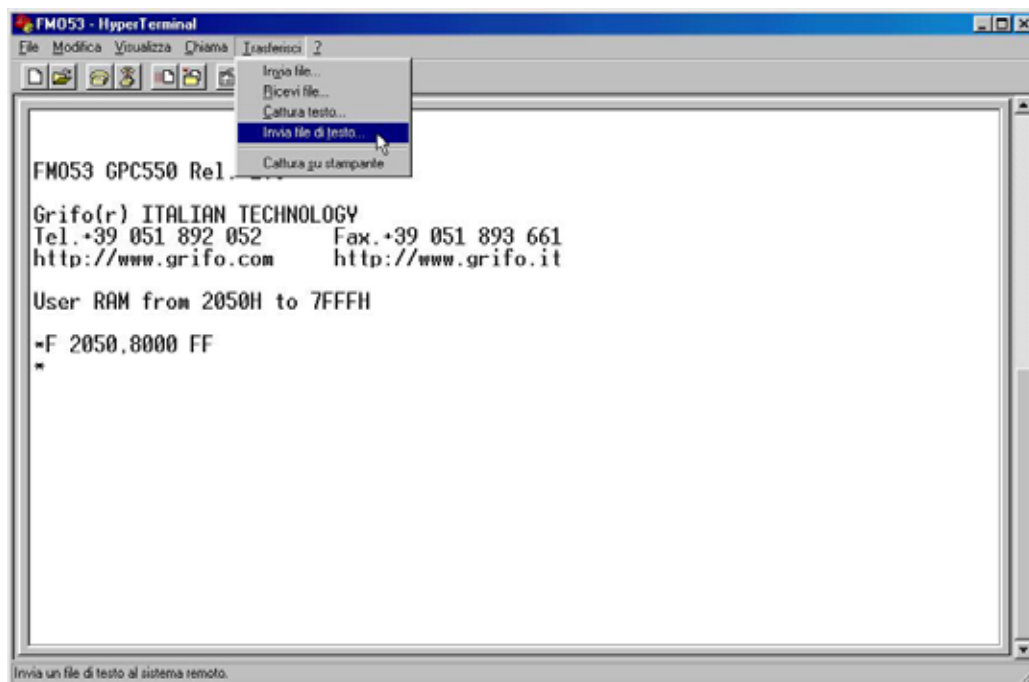


FIGURE 22: **DOWNLOADING** FILE WITH **HYPERTERMINAL** (1 OF 2)

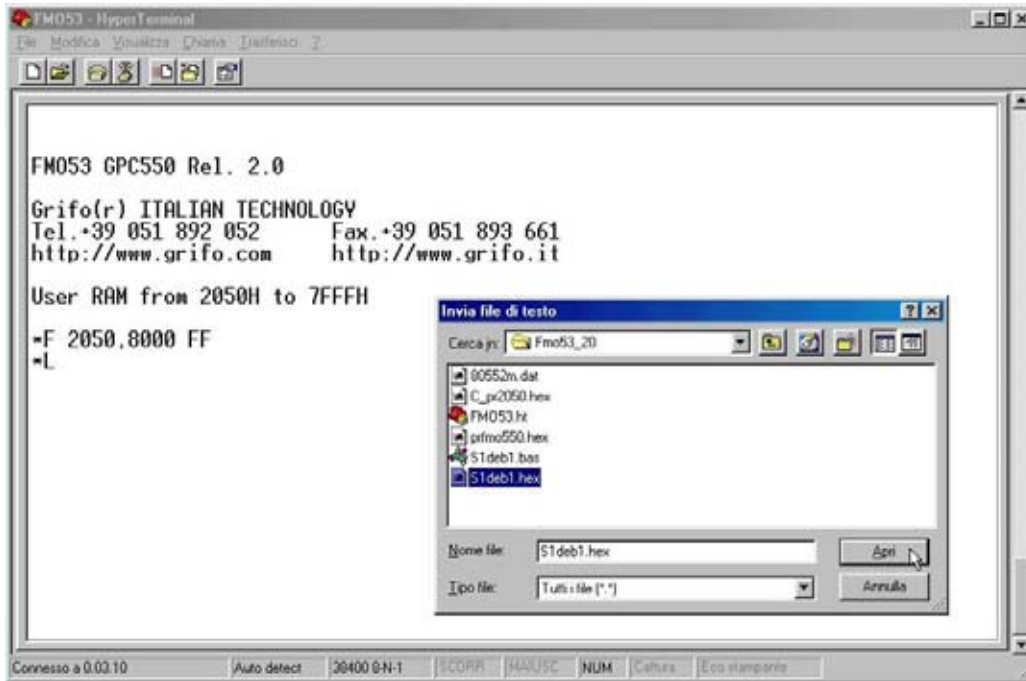


FIGURE 23: DOWNLOADING FILE WITH HYPERTERMINAL (2 OF 2)

- 8) Waiting for the end of the downloading selected with the occur number of downloaded bytes (that meet the length of the generated code) . During this time the user has to care of don't touch any button on the keyboard, in fact this in transmitted on the serial line and modify the contemporary file tranmitted and generating an error.
- 9) Execute the downloaded program typing the command G <start user code>, where start user code meet the beginning application program address downloaded in hexadecimal format (i.e. 2050).

### USING WITH $\mu$ C/51

$\mu$ C/51 is a C cross compiler, cheap,efficient, easy to use, provided with a complete IDE (Integrated Development Environment). Which is avalaible for every version of Windows and generate an executable code for microcontrollers of the "I51" family.

For further information about the general features of this product it is suggested to consult the technical paper, the handbook or the complete help on line. The info related to the integration with **FM053** are reported instead in the following paragraphs.

## CONFIGURATION FOR SERIAL COMMUNICATION

Among the installed programs with  $\mu\text{C}/51$  none manage the serial communication with the control board that execute the FMO53. For this reason we recommend to use one of the programs for the emulation terminal ( as GET51 or HYPERTERMINAL) described in the previous paragraphs. Potentially is possible to integrate the start of the communication program selected on IDE of Jen's file Editor (JFE), when workspace files are used generated from the utility program MakeWiz. In order to obtain this mix it is sufficient to read the user manual of  $\mu\text{C}/51$  and modify the file DL.BAT, to make the start of the communication program to use.

For example if You want to use HYPERTERMINAL You have to use the describer operations in the USING WITH HYPERTERMINAL paragraph and than to insert the following command lines in the file DL.BAT:

```
@ECHO OFF
REM Enable emulatio terminal with configured HYPERTERMINAL
REM from file FMO53.HT
C:\PROGRA~1\ACCESS~1\HYPER~1\HYPERTERM.EXE FMO53.HT
ECHO ON
```

## APPLICATION PROGRAM INSTRUCTION

When a compilation of a  $\mu\text{C}/51$  source is executed, a HEX Intel file is generated; it contains the machine code of the microconroller that it has been used. Through some guidelines of the compiler, the user can configure the program so that it can be downloaded and executed with the **FMO53**, as we have described in the APPLICATIVE USER CONFIGURATION PROGRAM paragraph.

The  $\mu\text{C}/51$  compilation instructions hare to be defined in proper project files .MAK that inform the compiler in its transformation process. Those files can be realized with the proper utility program *Make Wiz*, that allow to define all requirement necessary as source files name, memory model, obtimizations, generated files, allocation addresses,used libraries, etc.

From this list, only three items, are connected to **FMO53**, that are:

Directives	Description
<i>Linker   Rom Start (Hex)</i>	-> Indicates the first byte of the program code address, that meets with the indication Start user code, used in this manual. This directive re-locate the response vectors too.
<i>Linker   Gen. HEX-File</i>	-> Indicates the address of the first byte of the external SRAM available as data area of program, that meets with the indication Start user data, used in this manual.
<i>Misc   Gen. Hex-file</i>	-> Indicates to compiler to generate the application program code in intel Hex format.

These, and other directives for **FMO53**, are described in the documentation of  $\mu\text{C}/51$ .

As shown in the previous chapter, the directives values change depending on the operative conditions and following we reports sme typical examples:

- Program in DEBUG mode on serial A:

Rom Start (Hex) =2050

External Ram Start (Hex) =7000

- Program in DEBUG mode on serial B:

Rom Start (Hex) =2100

External Ram Start (Hex) =7000

- Program in AUTORUN mode:

Rom Start (Hex) =8050

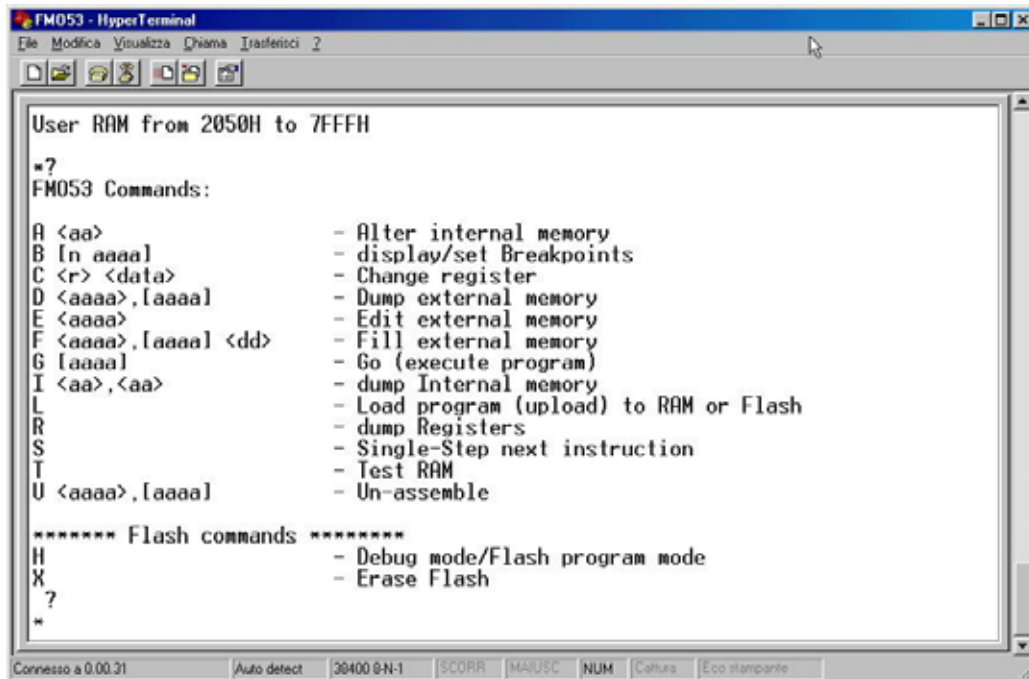
External Ram Start (Hex) =7000

## COMMANDS

All the functionalities performed by **FMO53** are enabled through a command series, that allow to communicate with the control board and obtain the wanted targets. After the start in **DEBUG** mode, the **FMO53**, shows the character of ready \* return line, where are passed and is waiting for the other command from the user. Several commands are signed from a command line, where are passed the required parameters, to perform the target action.

The whole command line has to be typed on development PC keyboard and on the monitor are shown the the potential results of the executed command.

In order to have the maximum amount of information in one screen of PC, the **FMO53** is projected to reduce at the minimum level the representation lines. All command that show one information line only (as **B** and **S**) make it on same line. For commands that show big data quantity (as **D**, **U**, etc.) pressing a key during the representation, it stop all. At this point each next **SPACE** pressure will show a new line, when the **ENTER** key pressure is used to definitively stop the command showing.



```

User RAM from 2050H to 7FFFH

*?
FMO53 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 <aa>, <aa>         - dump Internal memory
L                   - Load program (upload) to RAM or Flash
R                   - dump Registers
S                   - Single-Step next instruction
T                   - Test RAM
U <aaaa>, [aaaa]     - Un-assemble

***** Flash commands *****
H                   - Debug mode/Flash program mode
X                   - Erase Flash
?
*

```

FIGURE 24: COMMAND LIST

In the following paragraphs are described all commands of **FMO53** with a lot information for their input parameters and shown result. In that parameters we put that user know the user microcontroller and its interesting sections.

## ALTER INTERNAL MEMORY

A <address>

Alter INTERNAL memory. **FMO 53** 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 ENTER to terminate the Alter command.

## SET AND SHOW BREAKPOINTS

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).

## **MODIFY REGISTERS**

**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).

The *value* to assign has to be inserted in hexadecimal format with 2 digits when the register is 8 bit, or 4 digits if the register is 16 bit.

## **SHOW EXTERNAL DATA MEMORY**

**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.

## **MODIFY EXTERNAL DATA MEMORY**

**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.

## **FILL EXTERNAL DATA MEMORY**

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

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

## **PERFORM REAL SPEED**

**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.

## **ENABLE COMMANDS FOR FLASH**

**H**

Enables the use of the FLASH programming related commands, such modality is indicated by the prompt shape, shown at the begin line that is as follows:

\* -> debug mode -> commands for FLASH not enabled  
&' -> FLASH program mode -> commands for FLASH enabled

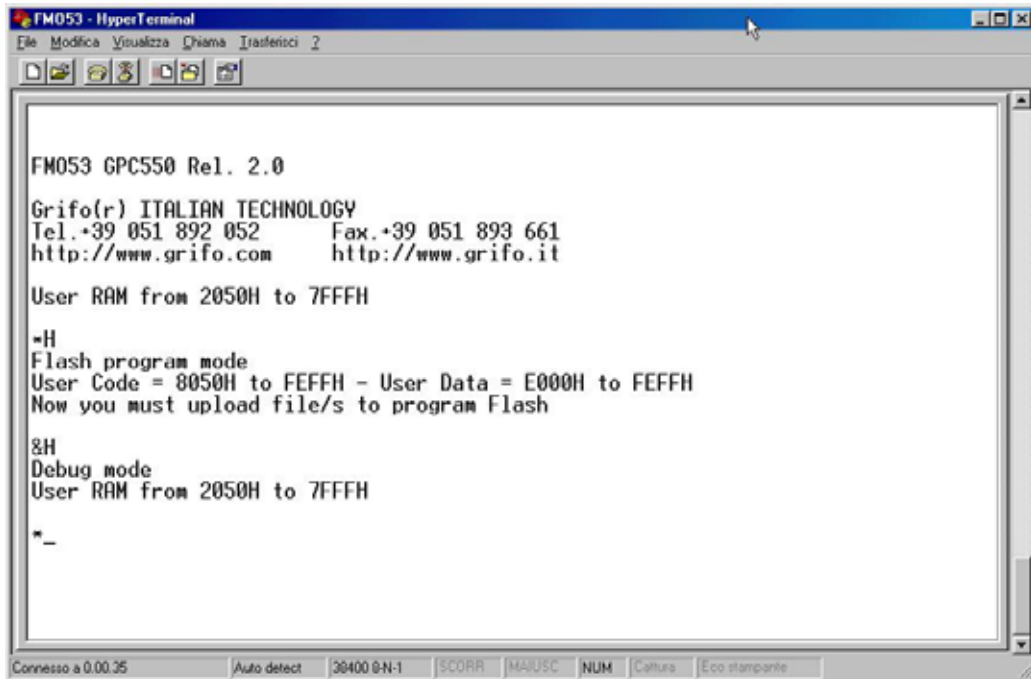
To simply this concept see figure 25.

We remind that enabling the commands for FLASH for the different situation influences the command L(UPLOAD), as indicated in the proper paragraph.

## **SHOW INTERNAL MEMORY**

**I** <start>,<end>

Displays the contents of INTERNAL RAM memory between the specified addresses in hexadecimal format.



```
FM053 GPC550 Rel. 2.0
Grifo(r) ITALIAN TECHNOLOGY
Tel. +39 051 892 052 Fax. +39 051 893 661
http://www.grifo.com http://www.grifo.it

User RAM from 2050H to 7FFFH

-H
Flash program mode
User Code = 8050H to FEFFH - User Data = E000H to FEFFH
Now you must upload file/s to program Flash

&H
Debug mode
User RAM from 2050H to 7FFFH

*
_
```

FIGURE 25: ENABLING COMMANDS FOR FLASH

## LOAD FILE

### L

Downloads data from the console port, which may be in either MOTOROLA or Intel Hex format. If you accidentally 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 **debug mode**, while if the prompt is '&' then the code will be used to **program the Flash memory**. In the last case if the downloading of the file and the FLASH programming are obtained with succesful, at the end the AUTORUN condition is set always in FLASH. So in the following power on, the user application program downloaded is automatically executed.

At the end of the succesful downloading, the FMO53 shows the number of saved Bytes in the board memory, rounded in excess. Viceversa in case of problems in downloading file or in FLASH programming are shown a message error. For more details for the L command, see figures 28 and 29 of this manual.

## SHOW REGISTERS

### R

Displays the current values of the 8051 registers (A, B, DPTR, SP, PC, PSW and R0-R7). This command, as all showing commands, is very comfortable in searching problem phase for the application program, in fact it allow to verify the execution status and find potential different or wrong values.

## EXECUTE STEP TO STEP

### 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.

## BOARD SRAM TEST

### 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).

The main property of this command is to verify the correct working of base SRAM and in the same time to check if the control board is correctly configured.

## **CODE UNASSEMBLE**

**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 neumonic, and operands to instruction.

## **ERASE FLASH**

**X**

Perform the total erasing of the FLASH content. Once recognized the command, the FMO53 ask a confirm to user in order to avoid erasing not wanted. Moreover this command can be performed only if before the command for FLASH were enabled, with the command H.

We underline that the FLASH erasing is total, so once executed the command, both the application program and data on FLASH are definitively lost; for further information, please see the FLASH EPROM MANAGEMENT paragraph.

## **SHOW COMMANDS**

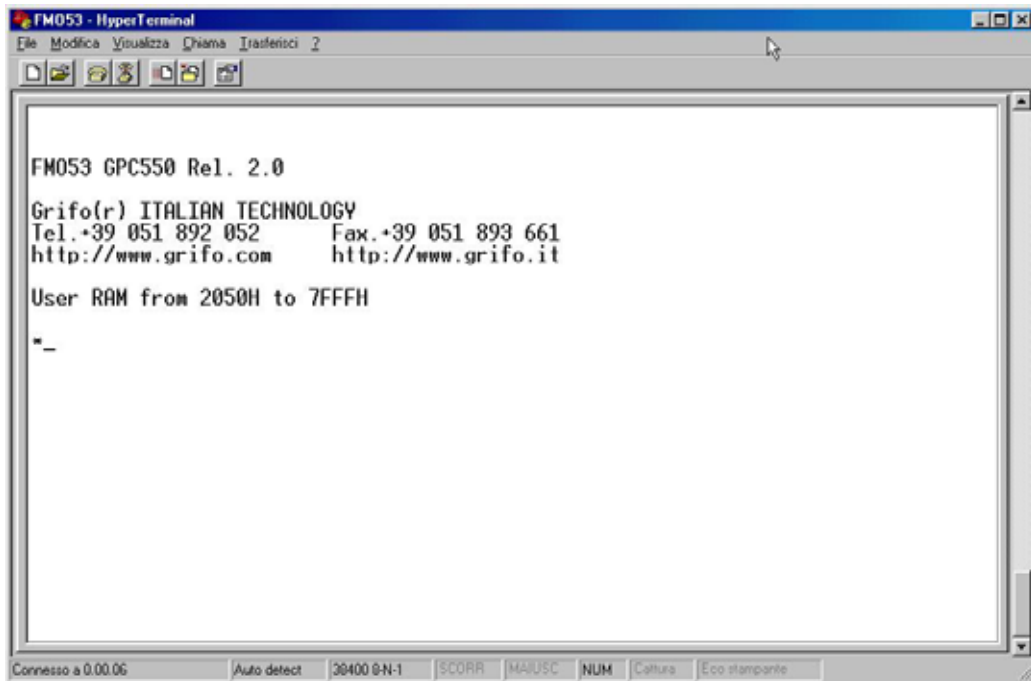
**?**

Displays a short help summary of the commands.

## HOW TO START

This chapter describes the operations to perform in order to begin to use **FMO53** package. In detail here is reported the correct sequence of operation with **grifo®** boards. For further information, please refer to the previous chapters, where each operation here explained is described with many more details.

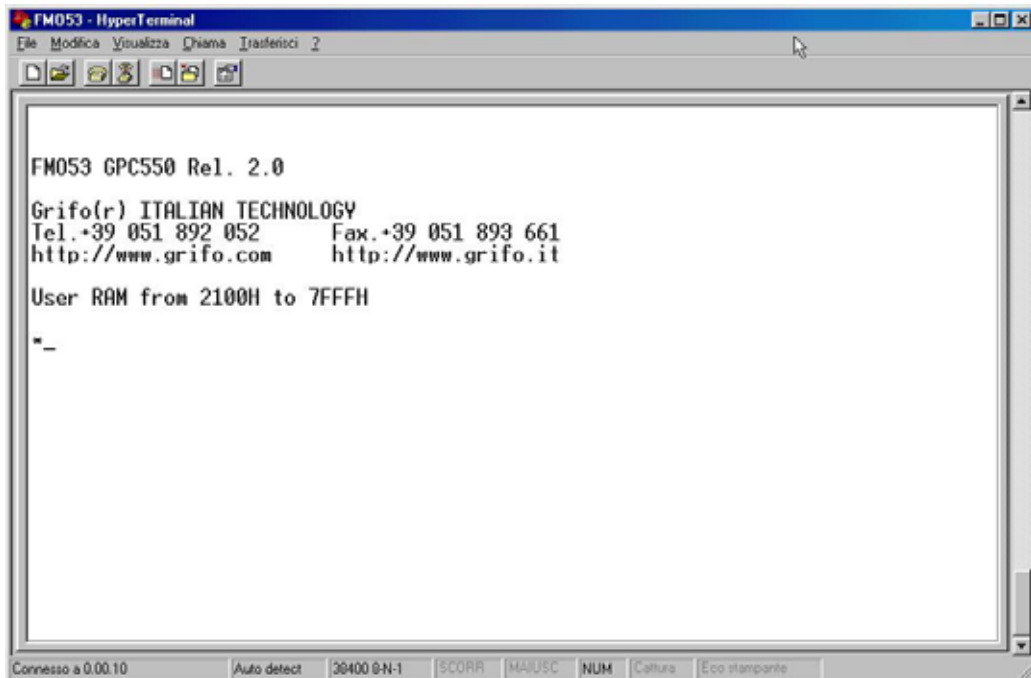
- 1) Read all the documentation included in the software package.
- 2) Select the the serial line to use on the control board as described in SERIAL COMMUNICATION paragraph.
- 3) Configure the board in according with the installed memories, following the proper descriptions.
- 4) Set the selector of the control board in order to select DEBUG mode, as described in the OPERATING MODE SELECTION paragraph.
- 5) Turn on the development PC and select the serial line to use. If this serial came from a converter (for exemple USB <-> RS 232 interfaces), perform the proper operation indicated in the product documentation, and verify the correct working.
- 6) Perform the serial connection following the indication of the SERIAL COMMUNICATION CABLE paragraph.
- 7) Install on PC the selected development tool and, if this one include the terminal emulation properties, configure it for the communication with **FMO53**. Otherwise a external program is needed, as **HYPERTERMINAL** of Windows. In the configuration naturally is to select the communication line of the PC, connected to point 6. For more details, please, see the proper paragraph as SERIAL COMMUNICATION and CONFIGURATION FOR SERIAL COMMUNICATION.
- 8) Activates the terminal emulation in the program configured at point 7.
- 9) Switch on the control board and verify that on the monitor of the development PC is shown the presentation messages of **FMO53**. This one is composed by: the name of the used board, the **FMO53** version, **grifo®** generality, the start and stop addresses of base SRAM for users and at the end the ready character \*. °To note that the start address of user SRAM, depending on the serial used on board, as described in figures 26 and 27.
- 10) Provide some command and verify their working with the results. Among them we recommend the command T (tests the memory of the board), F (fills the external data memory), D ( shows the external data memory), R ( shows registers), etc.



```
FM053 GPC550 Rel. 2.0
Grifo(r) ITALIAN TECHNOLOGY
Tel.+39 051 892 052      Fax.+39 051 893 661
http://www.grifo.com   http://www.grifo.it

User RAM from 2050H to 7FFFH
*_
```

FIGURA 26: FMO53 POWER ON MESSAGE ON SERIAL A

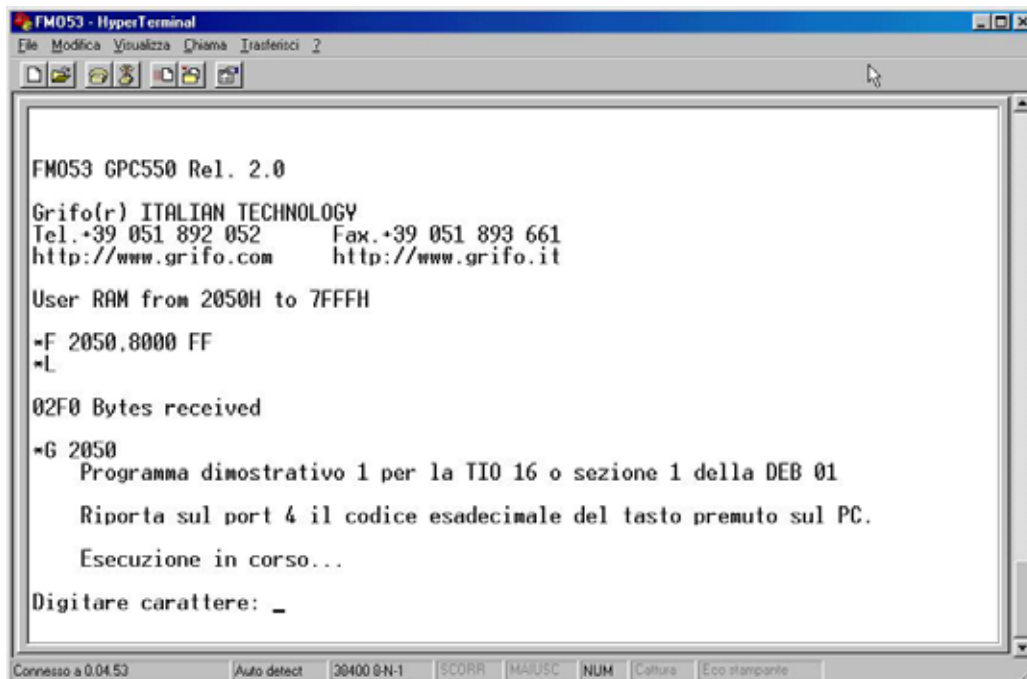


```
FM053 GPC550 Rel. 2.0
Grifo(r) ITALIAN TECHNOLOGY
Tel.+39 051 892 052      Fax.+39 051 893 661
http://www.grifo.com   http://www.grifo.it

User RAM from 2100H to 7FFFH
*_
```

FIGURE 27: FMO53 POWER ON MESSAGE ON SERIAL B

- 11) Looking for one of the example programs for the used board on the received CD , coded with the selected development tool, and copy it in a folder on the development PC. If the example are using other files too, copy all file.
- 12) Open the demo program described at point 11 with the selected development tool and configure it for DEBUG mode on the serial line selected at point 2. This configuration meet the setting of both the addresses, the **start user code** and **start user data** address in SRAM, described in CONFIGURATION USER APPLICATION PROGRAM and APPLICATION PROGRAM INSTRUCTION paragraphs.
- 13) Compile the demo program opened, verify there are no errors and that the HEX Intel file was created, with the executable code of the demo.
- 14) Download the HEX Intel program obtained at point 13 into SRAM of the control board and execute it, following the instruction of CONFIGURATION FOR SERIAL COMMUNICATION paragraph. The picture 28 show the execution of this step.



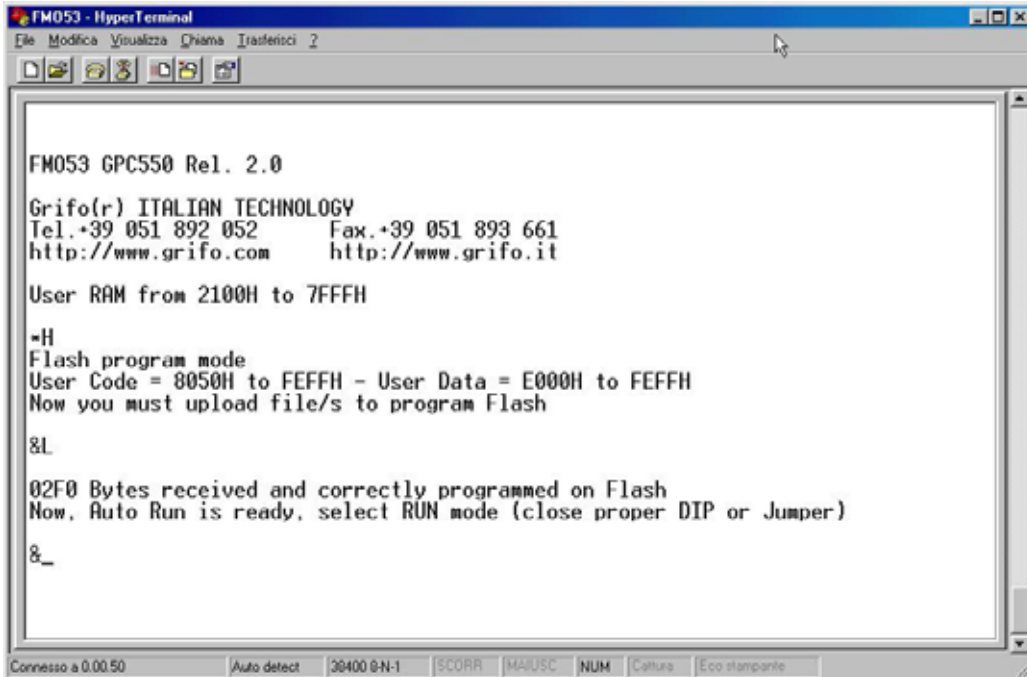
```

FM053 - HyperTerminal
File Modifica Visualizza Chiama Istruzioni ?
FM053 GPC550 Rel. 2.0
Grifo(r) ITALIAN TECHNOLOGY
Tel. +39 051 892 052 Fax. +39 051 893 661
http://www.grifo.com http://www.grifo.it
User RAM from 2050H to 7FFFH
*F 2050,8000 FF
*L
02F0 Bytes received
*G 2050
Programma dimostrativo 1 per la TIO 16 o sezione 1 della DEB 01
Riporta sul port 4 il codice esadecimale del tasto premuto sul PC.
Esecuzione in corso...
Digitare carattere: _
Connesso a 0.04.53 | Auto detect | 38400 8-N-1 | SCORR | MAIUSC | NUM | Callus | Eco stampante
    
```

**FIGURE 28: DOWNLOADING AND EXECUTION OF PROGRAM IN SRAM**

- 15) Verify that the program executed works completely.
- 16) Re-configure the development tool in AUTORUN mode. This configuration sets the **start user code** address on FLASH and **start user data** address in SRAM too, described in CONFIGURATION USER APPLICATION PROGRAM and APPLICATION PROGRAM INSTRUCTION paragraphs.
- 17) Compile the demo program opened, verify there are no errors and that the HEX Intel file was created, with the executable code of the demo.

- 18) Download the HEX Intel program obtained at point 13 into FLASH of the control board and execute it, following the instruction of CONFIGURATION FOR SERIAL COMMUNICATION paragraph. The picture 29 show the execution of this step.



```
FM053 - HyperTerminal
File Modifica Visualizza Chiama Trasferisci 2

FM053 GPC550 Rel. 2.0
Grifo(r) ITALIAN TECHNOLOGY
Tel.+39 051 892 052 Fax.+39 051 893 661
http://www.grifo.com http://www.grifo.it

User RAM from 2100H to 7FFFH
-H
Flash program mode
User Code = 8050H to FEFFH - User Data = E000H to FEFFH
Now you must upload file/s to program Flash

&L

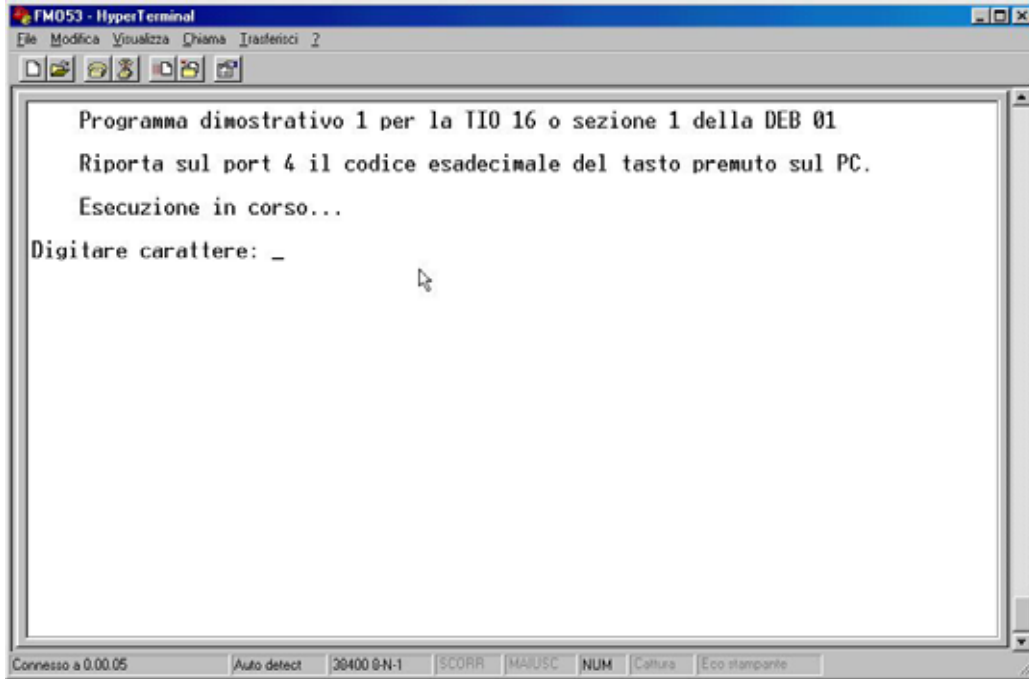
02F0 Bytes received and correctly programmed on Flash
Now, Auto Run is ready, select RUN mode (close proper DIP or Jumper)

&_

Connesso a 0.00.50 Auto detect 38400 8-N-1 SCORR MAIUSC NUM Cultura Eco stampante
```

FIGURE 29: DOWNLOADING AND EXECUTION OF PROGRAM IN FLASH

- 19) Verify that the downloading and the programming are performed correctly. At this point in FLASH is programmed the demo program and the settings that make start it in AUTORUN.
- 20) Set the selector of the control board in order to select the AUTORUN mode, so as described in the OPERATING MODE SELECTION paragraph.
- 21) In this phase reset or re-power on the board to obtain the automatic start of the demo program saved in FLASH, as shown in figure 30.
- 22) Re-set the selector of the control board in order to select the DEBUG mode, so as described in the OPERATING MODE SELECTION.
- 23) Reset or re-power on the board to obtain again the **FM053** start , already described to point 9 and in figures 26 and 27.



**FIGURE 30: AUTOMATIC PROGRAM STARTING IN AUTORUN**

## APPENDIX A: ALPHABETICAL INDEX

**Simboli**

.128K 7  
.128KF 7  
.32K 7  
.32KF 7  
? 43  
 $\mu$ C/51 10, 34

**A**

ADDS-VIEWPOINT 29  
Alter INTERNAL memory 38  
Assistance 1  
AUTORUN 17, 24, 46

**B**

BASCOM 8051 10  
Bascom 8051 25  
Baud Rate 19  
breakpoint 38  
Buffer 20

**C**

Command of FMO 53  
T 42  
Command of FMO 53  
? 43  
A 38  
B 38  
C 39  
D 39  
E 39  
F 40  
G 40  
H 40  
I 40  
L 41  
R 42  
S 42  
U 42  
Communication 25  
Communcation 19  
Computer 7  
Current Loop 19

**D**

DATA 23  
DEBUG 16, 24, 46  
Display help 43  
Download data 41  
dump 39

**E**

Edit memory 39  
Electrostatic 1  
Emulation 12  
EPROM 7  
ESD 1

**F**

Fill memory 40  
FLASH 7, 22, 40

**G**

GET 51 29  
GET51 12  
GND 8  
GPC® 7

**H**

Handshake 25  
HYPERTERMINAL 12, 31, 44

**I**

INTRODUCTION 1

**M**

Mapping 13

**N**

Norms 1

**P**

Parity 19

**R**

Realse 3  
RS232 8, 19  
RS422 19  
RS485 19  
RxD 8

**S**

Safety 1  
Serial 8  
Single-Step 42  
Software 9  
SRAM 7  
SRAM test 42  
Stop Bit 19

**T**

TxD 8

**U**

Un-assemble 43

**V**

Version 2, 3  
Versione 7