

# MPS 051

Micro Programmer and Simulator  
AT89c1051, AT89c2051 and AT89c4051

MANUALE UTENTE

USER MANUAL



**grifo**<sup>®</sup>

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>

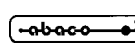
Tel. +39 051 892.052 (a. r.) FAX: +39 051 893.661



MPS 051

Edition 3.00

Rel 20 November 2000

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



# MPS 051

**Micro Programmer and Simulator**  
**AT89c1051, AT89c2051 and AT89c4051**

MANUALE UTENTE

USER MANUAL

L'**MPS 051** è un simulatore/programmatore per i microcontrolloti della serie **MCS 51** ATMEL AT89C4051, AT89C2051 e AT89C1051.

La sezione del **PROGRAMMATORE** fornisce tutte le funzioni correlate al ruolo di un tipico dispositivo di programmazione quali lettura, blank check, programmazione della **FLASH** e dei **LOCK bit**, verifica e cancellazione del circuito. Può programmare i chip ATMEL da AT89C1051 fino all'AT89C4051.

La sezione **SIMULATORE** fornisce quelle funzioni indispensabili per un debugging efficiente dell'applicazione ovvero GOTO, CALL, SINGLE STEP and STEP OVER execution.

**MPS 051** is a simulator/programmer of **ATMEL's** single-chip **MCS 51** microcontrollers, types AT89C4051, AT89C2051 and AT89C1051.

The **PROGRAMMER** part provides all programming-related functions, including reading, blank check, **FLASH** programming, **LOCK bit** programming, program verification, and circuit erasing. It can program chips ATMEL from AT89C1051 to AT89C4051.

The **SIMULATOR** part features all the essential functions for efficiently debugging the application programs like GOTO, CALL, SINGLE STEP and STEP OVER execution.

**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>

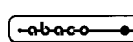
Tel. +39 051 892.052 (a. r.) FAX: +39 051 893.661



**MPS 051**

Edition 3.00

Rel 20 November 2000

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

## DOCUMENTATION COPYRIGHT BY grifo®, 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®**.

### IMPORTANT

Although all the information contained herein have been carefully verified, **grifo®** assumes no responsibility for errors that might appear in this document, or for damage to things or persons resulting from technical errors, omission and improper use of this manual and of the related software and hardware.

**grifo®** reserves the right to change the contents and form of this document, as well as the features and specification of its products at any time, without prior notice, to obtain always the best product.

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

### SYMBOLS DESCRIPTION

In the manual could appear the following symbols:

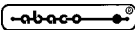


Attention: Generic danger



Attention: High voltage

### Trade Marks

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

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

# INDICE GENERALE

INTRODUZIONE .....	1
INFORMAZIONI PRELIMINARI .....	2
CONVENZIONI E TERMINOLOGIA .....	2
CONVENZIONI .....	2
TERMINOLOGIA .....	2
CARATTERISTICHE GENERALI .....	3
CARATTERISTICHE TECNICHE .....	4
CARATTERISTICHE GENERALI .....	4
CARATTERISTICHE FISICHE .....	4
CARATTERISTICHE ELETTRICHE .....	4
INSTALLAZIONE .....	6
ATTENZIONE AGLI ESD (ELECTROSTATIC SENSITIVE DEVICE) .....	6
CONTENUTO DELLA CONFEZIONE .....	6
CAVO DI CONNESSIONE .....	6
INSTALLAZIONE HARDWARE .....	8
JUMPERS .....	9
LEDS .....	9
AVVIO RAPIDO .....	11
SIMULAZIONE .....	11
PROGRAMMAZIONE .....	11
DESCRIZIONE SOFTWARE .....	12
REQUISITI MINIMI DEL PC .....	12
CONTENUTO DEL DISCO .....	12
LANCIARE IL PROGRAMMA DI CONTROLLO .....	13
AVVIO DEL PROGRAMMA CON PARAMETRI .....	13
SIMULATORE .....	14
NOTE SULLA SIMULAZIONE .....	15
ON LINE HELP .....	15
OPERAZIONI IN SINGLE STEP .....	16
NOTA DI UTILIZZO DELLO STEP OVER .....	17
INSTALLAZIONE DI BREAK POINT NEL PROGRAMMA DEBUGGATO .....	17
LIMITAZIONI DEL PROGRAMMA IN SINGLE STEP .....	17
PROGRAMMATORE .....	18
PROCEDURA CONSIGLIATA .....	19
ESEMPIO OPERATIVO .....	20
DEBUGGING DEL PROGRAMMA .....	21
LIMITAZIONI HARDWARE DEL CIRCUITO AT89C2051 .....	21
LIMITAZIONI SOFTWARE .....	22
COMANDI DEL PROGRAMMA DI CONTROLLO .....	23

<b>MENU SIMULATOR</b> .....	<b>23</b>
<b>MENU PROGRAMMER</b> .....	<b>25</b>
<b>MENU FILE</b> .....	<b>26</b>
<b>MENU BUFFER</b> .....	<b>26</b>
<b>MENU OPTIONS</b> .....	<b>28</b>
<b>MENU QUIT</b> .....	<b>30</b>
<b>MENU ABOUT</b> .....	<b>30</b>
<b>HELP</b> .....	<b>30</b>
<b>TERMINI DELLA GARANZIA</b> .....	<b>31</b>
<b>RISOLUZIONE DEI PROBLEMI</b> .....	<b>32</b>
<b>ERRORI DI COMUNICAZIONE</b> .....	<b>32</b>
<b>PROBLEMI IN LETTURA O SCRITTURA</b> .....	<b>32</b>
<b>STRUMENTI AGGIUNTIVI</b> .....	<b>33</b>
<b>INDICE ANALITICO</b> .....	<b>34</b>

# INDICE DELLE FIGURE

<b>FIGURA 1: FOTO MPS 051 .....</b>	<b>5</b>
<b>FIGURA 2: CONTENUTO DELLA CONFEZIONE .....</b>	<b>7</b>
<b>FIGURA 3: TABELLA RIASSUNTIVA DEI JUMPERS .....</b>	<b>9</b>
<b>FIGURA 4: TABELLA DEI LEDs .....</b>	<b>9</b>
<b>FIGURA 5: POSIZIONE JUMPERS, CONNETTORI E LEDs .....</b>	<b>10</b>
<b>FIGURA 6: POLARITÀ DEL CONNETTORE DI ALIMENTAZIONE .....</b>	<b>10</b>
<b>FIGURA 7: FOTO DELL'ADATTATORE PER T-EMU 52 .....</b>	<b>21</b>
<b>FIGURA 8: TABELLA RITARDI IN ESECUZIONE INTERRUPT .....</b>	<b>22</b>

# GENERAL INDEX

<b>INTRODUCTION .....</b>	<b>37</b>
<b>PRELIMINARY INFORMATION .....</b>	<b>38</b>
<b>CONVENTIONS AND TERMINOLOGY .....</b>	<b>38</b>
<b>CONVENTIONS USED IN THE MANUAL .....</b>	<b>38</b>
<b>TERMINOLOGY USED IN THE MANUAL .....</b>	<b>38</b>
<b>GENERAL FEATURES .....</b>	<b>39</b>
<b>TECHNICAL FEATURES .....</b>	<b>40</b>
<b>GENERAL FEATURES .....</b>	<b>40</b>
<b>PHYSICAL FEATURES .....</b>	<b>40</b>
<b>ELECTRIC FEATURES .....</b>	<b>40</b>
<b>INSTALLATION .....</b>	<b>42</b>
<b>CAUTION FOR ESD (ELECTROSTATIC SENSITIVE DEVICE) .....</b>	<b>42</b>
<b>DELIVERY CONTENTS .....</b>	<b>42</b>
<b>CONNECTION CABLE .....</b>	<b>42</b>
<b>HARDWARE INSTALLATION .....</b>	<b>44</b>
<b>JUMPERS .....</b>	<b>45</b>
<b>LEDS .....</b>	<b>45</b>
<b>QUICKSTART .....</b>	<b>47</b>
<b>SIMULATION .....</b>	<b>47</b>
<b>PROGRAMMING .....</b>	<b>47</b>
<b>SOFTWARE DESCRIPTION .....</b>	<b>48</b>
<b>MINIMAL PC REQUIREMENTS .....</b>	<b>48</b>
<b>DISK CONTENT .....</b>	<b>48</b>
<b>STARTING THE CONTROL PROGRAM .....</b>	<b>49</b>
<b>NOTE ON STARTING THE PROGRAM BY PARAMETERS .....</b>	<b>49</b>
<b>SIMULATOR .....</b>	<b>50</b>
<b>NOTE ON SIMULATION MODE .....</b>	<b>51</b>
<b>ON LINE HELP .....</b>	<b>51</b>
<b>SINGLE STEP OPERATION MODE .....</b>	<b>52</b>
<b>NOTE ON USING THE STEP OVER .....</b>	<b>53</b>
<b>BREAK-POINT INSTALLATION INTO THE DEBUGGED PROGRAM .....</b>	<b>53</b>
<b>DEBUGGED PROGRAM LIMITATIONS IN THE SINGLE STEP MODE .....</b>	<b>53</b>
<b>PROGRAMMER .....</b>	<b>54</b>
<b>RECOMMENDED PROCEDURE .....</b>	<b>55</b>
<b>OPERATION EXAMPLE .....</b>	<b>56</b>
<b>PROGRAM DEBUGGING .....</b>	<b>57</b>
<b>HARDWARE LIMITATIONS ON THE AT89C2051 CIRCUIT .....</b>	<b>57</b>
<b>SOFTWARE LIMITATIONS .....</b>	<b>58</b>
<b>CONTROL PROGRAM COMMANDS .....</b>	<b>59</b>



**SIMULATOR MENU ..... 59**

**PROGRAMMER MENU ..... 61**

**FILE MENU ..... 62**

**OPTIONS MENU ..... 64**

**QUIT MENU ..... 66**

**ABOUT MENU ..... 66**

**HELP SYSTEM ..... 66**

**WARRANTY TERMS ..... 67**

**TROUBLESHOOTING ..... 68**

**COMMUNICATION ERRORS ..... 68**

**READING OR PROGRAMMING PROBLEMS ..... 68**

**ADDITIONAL TOOLS ..... 69**

**ALPHABETICAL INDEX ..... 70**

# FIGURES INDEX

<b>FIGURE 1: MPS 051 PHOTO .....</b>	<b>41</b>
<b>FIGURE 2: DELIVERY CONTENTS .....</b>	<b>43</b>
<b>FIGURE 3: JUMPERS SUMMARIZING TABLE .....</b>	<b>45</b>
<b>FIGURE 4: LEDs TABLE .....</b>	<b>45</b>
<b>FIGURE 5: JUMPERS, CONNECTOR AND SOCKETS LOCATION .....</b>	<b>46</b>
<b>FIGURE 6: SUPPLY CONNECTOR POLARITY .....</b>	<b>46</b>
<b>FIGURE 7: ADAPTOR POD FOR T-EMU 52 PHOTO .....</b>	<b>57</b>
<b>FIGURE 8: DELAYS IN EXECUTION OF INTERRUPTS TABLE .....</b>	<b>58</b>



## INTRODUZIONE

Scopo di questo manuale é la trasmissione delle informazioni necessarie all'uso competente e sicuro dei prodotti. Esse sono il frutto di un'elaborazione continua e sistematica di dati e prove tecniche registrate e validate dal Costruttore, in attuazione alle procedure interne di sicurezza e qualità dell'informazione.

Per un corretto rapporto coi prodotti, é necessario garantire leggibilità e conservazione del manuale, anche per futuri riferimenti. In caso di deterioramento o più semplicemente per ragioni di approfondimento tecnico ed operativo, consultare direttamente l'Assistenza Tecnica autorizzata.

Al fine di non incontrare problemi nell'uso di tali dispositivi, é conveniente che l'utente - **PRIMA DI COMINCIARE AD OPERARE** - legga con attenzione tutte le informazioni contenute in questo manuale. In una seconda fase, per rintracciare più facilmente le informazioni necessarie, si può fare riferimento all'indice generale e all'indice analitico, posti rispettivamente all'inizio ed alla fine del manuale.

Le informazioni fornite in questo manuale sono precise ed affidabili fino alla data di rilascio del manuale stesso, ma l'impegno per migliorare tutti i nostri prodotti non si ferma mai. Siete pregati di consultare i file di documentazione nei floppy per eventuali aggiornamenti dell'ultimo minuto.

Questo programma di controllo è protetto dalle leggi sul diritto d'autore, tutti i diritti sono riservati. Nè il programma di controllo nè alcuna sua parte possono essere disassemblati, analizzati o modificati in alcun modo, o con alcun mezzo, per nessuno scopo.

Questo documento è protetto dalle leggi sul diritto d'autore, tutti i diritti sono riservati, pertanto non può essere copiato, riprodotto o tradotto in alcun modo o con nessun mezzo, nè interamente nè in parte, senza un permesso scritto della **grifo®**.

La **grifo®** non si assume alcuna responsabilità per un uso errato del presente manuale.

La **grifo®** si riserva il diritto di effettuare cambiamenti o miglioramenti al prodotto descritto nel presente manuale in ogni momento senza dover dare alcuna comunicazione preventiva.

Il presente manuale contiene nomi di compagnie, software, prodotti, ecc. che sono registrati dai legittimi proprietari. La **grifo®** rispetta tali diritti di proprietà.

## INFORMAZIONI PRELIMINARI

Questo manuale spiega come installare ed usare il programma di controllo ed il vostro simulatore/programmatore **MPS 051**. Si assume che l'Utente abbia una minima esperienza con PC e con l'installazione di software, comunque il capitolo "AVVIO RAPIDO" vi guiderà passo dopo passo attraverso tutta la procedura di installazione.

Una volta installato il programma di controllo si consiglia di riferirsi sempre all'help sensibile al contesto del programma stesso piuttosto che al presente Manuale Utente. Le revisioni della documentazione sono implementate nell'help prima che nel Manuale Utente.

Il simulatore/programmatore **MPS 051** funziona pressochè con tutti i PC compatibili IBM, dagli XT ai Pentium Pro, sia portatili che desktop. Non si richiede alcuna scheda di interfaccia speciale per collegare il PC poichè viene usata la porta seriale.

Il simulatore/programmatore **MPS 051** funziona impeccabilmente su computers che adottano DOS, Windows 3.x e Windows 95/98 come sistema operativo.

Il simulatore/programmatore è pilotato da un programma di controllo facile da usare con menù a tendina, tasti speciali ed help in linea.

## CONVENZIONI E TERMINOLOGIA

Vengono usati in questo manuale alcune convenzioni e termini speciali:

### CONVENZIONI

I riferimenti alle funzioni del programma di controllo sono in maiuscolo, ad esempio **LOAD FILE**, ecc. I riferimenti ai tasti speciali sono scritti in parentesi angolari <>, ad esempio <F1>.

### TERMINOLOGIA

<b>zoccolo ZIF</b>	Zoccolo senza sforzo d'inserzione ( <b>Z</b> ero <b>I</b> nserzione <b>F</b> orce) usato per ospitare il dispositivo da programmare.
<b>BUFFER</b>	Parte di spazio su disco usato per memorizzazioni temporanee.
<b>linea seriale RS 232</b>	Tipo di porta del PC (seriale), dedicata principalmente alla connessione dei dispositivi seriali (mouse, modem, scanner ecc.).
<b>formato HEX</b>	Formato di file di dati che può essere letto da comuni visualizzatori di testo; esempio il byte 5AH viene rappresentato dai caratter '5' ed 'A', cioè dai bytes 35H e 41H. Una riga di un file HEX (un record) contiene indirizzo iniziale, bytes di dati e checksum.

## CARATTERISTICHE GENERALI

Il simulatore/programmatore **MPS 051** è progettato per emulare e programmare i single chip della **ATMEL** modelli AT89C2051 e AT89C1051, inoltre può programmare gli AT89C4051.

Il simulatore/programmatore **MPS 051** permette di lavorare comodamente con i microcontrollori single chip ATMEL AT89C2051 e AT89C1051. Elimina la necessità di continue rimozioni, riprogrammazioni e reinserimenti dei chip. Inoltre permette all'Utente di programmare il circuito con codice già debuggato. L'**MPS 051** può essere alimentato sia dal circuito applicativo (sufficiente ad alimentare la sezione di simulazione) sia dall'alimentatore fornito. Si collega al PC tramite linea seriale RS 232.

Il progetto della sezione **SIMULAZIONE** impiega i port non utilizzati del processore standard serie 51 a 40 pin per la simulazione circuitale dell'**AT89C2051**. Questa funzione viene effettuata dal processore **AT89C51** fornito di memoria **RAM esterna**. Quest'ultima viene usata per la memorizzazione e l'esecuzione del programma Utente. Il processore contiene un monitor residente che comunica con il programma di controllo, gestisce la RAM interna, quella esterna ed i registri, seleziona una modalità e modifica lo stato dei port. Un vantaggio importante consiste nel poter eseguire a **passi** di singole istruzioni il programma sotto debugging e di poter modificare il contenuto della RAM interna e dei registri del processore. Il monitor non usa interrupts, di conseguenza sono tutti a disposizione dell'Utente sebbene siano rallentati dall'istruzione **LJMP** verso l'area di RAM esterna. Il simulatore si connette al circuito dell'applicazione mediante un flat con uno zoccolo da 20 pin intestato al suo estremo.

La sezione **PROGRAMMAZIONE** fornisce tutte le funzioni correlate, incluse lettura, verifica di cancellazione, programmazione della **FLASH**, programmazione del **LOCK bit**, verifica di programmazione e cancellazione del contenuto della **FLASH**.

Tutte le funzioni dell'**MPS 051** sono controllate da un comodo programma simile a quelli usati per programmatori e simulatori. Il programma è guidato da menu e usa tasti speciali. Gestisce anche un buffer interno che permette la modifica dei dati binari. Inoltre, il buffer può essere caricato o salvato in differenti formati, compreso quello sorgente MCS51.

## CARATTERISTICHE TECNICHE

### CARATTERISTICHE GENERALI

<b>Microcontrollori simulati:</b>	ATMEL AT89c2051 ATMEL AT89c1051
<b>Microcontrollori programmati:</b>	ATMEL AT89c4051 ATMEL AT89c2051 ATMEL AT89c1051
<b>Cavo seriale:</b>	2 m

### CARATTERISTICHE FISICHE

<b>Dimensioni:</b>	132 x 66 x 30 [mm]
<b>Massa:</b>	120 g (il solo <b>MPS 051</b> )
<b>Campo temperatura:</b>	0÷40 °C
<b>Socket ZIF:</b>	24 pins
<b>Socket di emulazione:</b>	20 pins

### CARATTERISTICHE ELETTRICHE

<b>Alimentazione da applicativo:</b>	5V ± 10%
<b>Alimentazione da fonte esterna:</b>	14÷25V/200mA
<b>Alimentatore fornito a corredo:</b>	230 Vac/12 Vdc
<b>Consumo di corrente:</b>	Simulatore - 90 mA max., 50 mA default Programmatore - 140 mA max., 90 mA default
<b>Connettività:</b>	linea seriale RS 232
<b>Velocità di comunicazione con PC:</b>	57600 Baud/11.0592 MHz
<b>Frequenza oscillatore:</b>	24 MHz max.



FIGURA 1: FOTO MPS 051

## INSTALLAZIONE

Questi paragrafi contengono tutte le informazioni essenziali per connettere l'**MPS 051** al PC ed installare il software. **Si prega di leggere completamente questi paragrafi prima di tentare qualunque utilizzo del vostro MPS 051.**

### ATTENZIONE AGLI ESD (ELECTROSTATIC SENSITIVE DEVICE)

**Attenzione!** I danni causati dalla mancata osservazione di queste precauzioni non sono coperti dalla garanzia.

Le precauzioni nel manipolare il simulatore mirano a prevenire danni all'elettronica.

Il simulatore può essere danneggiato da cariche elettrostatiche o correnti transitorie. Osservare le seguenti regole, o regole equivalenti, è indispensabile:

- Prima di usare il simulatore, toccate un oggetto metallico ampio e/o connesso a massa.
- Non toccate mai i piedini del simulatore nè alcun altro contatto metallico dell'**MPS 051** se non siete certi di essere allo stesso potenziale. Ciò vale anche per i suoi circuiti interni.
- Non togliete le protezioni antistatiche dello zoccolo del simulatore a meno di grave necessità.
- Quando collegate il simulatore ad un altro dispositivo, verificate il potenziale zero di quest'ultimo e del simulatore. Idealmente, le masse dovrebbero essere collegate da un conduttore di notevole spessore.
- Il simulatore ed il dispositivo a cui viene collegato devono essere spenti. Questo vale anche quando si connettono il simulatore ed il PC.

### CONTENUTO DELLA CONFEZIONE

La confezione che avete ricevuto deve contenere almeno il seguente materiale:

- Un **MPS 051** comprensivo di cavo di emulazione e zoccolo ZIF
- Un disco da 3.5" contenente utility software
- Un disco da 3.5" contenente questo manual in formato pdf
- Un Alimentatore per la corrente di rete
- Un cavo di collegamento RS 232 lungo 2 metri
- Confezione di cartone

### CAVO DI CONNESSIONE

La velocità di comunicazione relativamente alta tra PC e l'**MPS 051** richiede un cavo di collegamento di alta qualità, come quello fornito in dotazione. Deve essere schermato, e la schermatura collegata a massa, e la sua lunghezza non dovrebbe eccedere i 3 metri.





FIGURA 2: CONTENUTO DELLA CONFEZIONE

## INSTALLAZIONE HARDWARE

Spegnete sia l'**MPS 051** che la scheda target.

Come prima cosa l'Utente deve decidere se preferisce usare i quarzi (XTAL) e controllare il segnale di RESET tramite la scheda target o internamente all'**MPS 051** stesso. Collegate i jumpers JP1, JP2 e JP3 a seconda di ciò che avete deciso, le connessioni sono esplicitate nella figura 3. Per localizzare facilmente i jumper JP1÷3 potete riferirvi alla figura 5. I jumpers sono saldati sul circuito stampato dentro i gusci di plastica, per essere raggiunti i gusci devono essere separati delicatamente. Inserite un capo del cavo RS 232 nell'apposito connettore dell'**MPS 051**, inserite l'altro capo nel connettore di interfaccia seriale RS 232 del PC (COM 1÷4), usate se necessario una riduzione.

Il programmatore/simulatore **MPS 051** viene fornito con uno zoccolo da 20 pin per l'emulazione la cui orientazione è visibile sull'adesivo che vi è attaccato sopra. Il lato colorato del flat corrisponde al pin numero 20. Inserite lo zoccolo del simulatore nello zoccolo della scheda target riservato al microcontrollore AT89C2051/1051 da emulare. Il PC e la scheda target devono avere la massa in comune.

### **NOTA:**

In caso di bisogno, procedete come di seguito: inserite lo zoccolo di simulazione dell'**MPS 051** nello zoccolo per la CPU della scheda target in modo che il pin numero 10 (GND) sia il primo ad entrare in contatto con lo zoccolo target. In alternativa, collegate prima l'**MPS 051** alla scheda target e solo in seguito collegatelo al PC - l'interfaccia RS 232 è più robusta dei port del processore.

*Una connessione errata può provocare danni al processore contenuto nell'**MPS 051** a causa di correnti di equalizzazione.*

Come seconda cosa bisogna selezionare il metodo per alimentare l'**MPS 051**: o da scheda target o dall'alimentatore fornito in dotazione. Il jumper JP4 ha lo scopo di escludere l'**MPS 051** dai pin di alimentazione sul socket del simulatore. Bisogna aprire il jumper JP4 se l'alimentazione sulla scheda target supera i 5V. Si può comunque alimentare l'**MPS 051** (da alimentatore in dotazione) e la scheda target (da una diversa fonte) separatamente mantenendo JP4 connesso, a patto che la condizione sopra indicata sia strettamente rispettata. Per localizzare facilmente il jumper JP4 fate riferimento alla figura 5.

Dopo avere connesso **MPS 051** e scheda target è possibile alimentare i dispositivi. Se state usando l'alimentatore esterno in dotazione, collegatene la spina per la rete in una presa di rete e il connettore coassiale all'**MPS 051**. Il led verde dell'**MPS 051** si deve accendere. Se preferite alimentare dalla scheda target allora alimentate la scheda target. La polarità del connettore esterno di alimentazione è mostrata in figura 6.

Ora potete lanciare il programma S2051.EXE. Per ulteriori informazioni consultate il capitolo "DESCRIZIONE SOFTWARE".

### **NOTE:**

- Lo switch JP5 è preimpostato in fabbrica a seconda della quantità di RAM installata, per cui l'utente non dovrebbe modificarne la posizione. Un cambiamento errato potrebbe danneggiare l'**MPS 051**.
- Il funzionamento interno dell'**MPS 051** è predisposto per accettare un quarzo interno di 11.0592 MHz o un quarzo esterno di una delle frequenze tabulate nel programma di controllo.
- Quando l'alimentazione viene data, il socket del programmatore si trova in uno stato di attesa grazie ad un circuito di reset ed al programma di controllo nel processore.

## JUMPERS

L'**MPS 051** è fornito di 5 jumpers per selezionare la sorgente del clock, la sorgente di alimentazione e la quantità della propria RAM installata. Evitate di toccare JP5 poichè la quantità di memoria propria dell'**MPS 051** è preimpostata alla fabbricazione e l'utente non la può cambiare.

JUMPER	CONNESSIONE	DESCRIPTION
JP1	posizione 1-2	Usa il segnale XTAL1 dalla scheda target.
	posizione 2-3	Usa il segnale XTAL1 interno.
JP2	posizione 1-2	Usa il segnale XTAL2 dalla scheda target.
	posizione 2-3	Usa il segnale XTAL2 interno.
JP3	posizione 1-2	Usa il segnale RESET dalla scheda target.
	posizione 2-3	Usa il segnale RESET interno.
JP4	connesso	L'alimentazione dell' <b>MCS 051</b> è presa dalla scheda target. Scollegare questo jumper se tale alimentazione può superare i 5V.
	non connesso	L'alimentazione dell' <b>MCS 051</b> è presa dall'alimentatore in dotazione.

**FIGURA 3: TABELLA RIASSUNTIVA DEI JUMPERS**

## LEDS

**MPS 051** è fornito di due LEDs per indicare la presenza dell'alimentazione e per visualizzare il proprio stato interno.

LED	COLORE	DESCRIZIONE
LED di alimentazione	Verde	Quando è acceso indica che l' <b>MPS 051</b> è alimentato.
LED di stato	Rosso	Quando è acceso indica che l' <b>MPS 051</b> sta programmando il dispositivo nel socket. Non aprite il socket quando questo LED è acceso.

**FIGURA 4: TABELLA DEI LEDS**

XC1 - connettore RS 232

XC2 - connettore per l'alimentazione esterna

XC3 - connettore del socket di simulazione

S1 - socket di precisione per la programmazione

S2 - socket di precisione per installare il socket ZIF per la programmazione

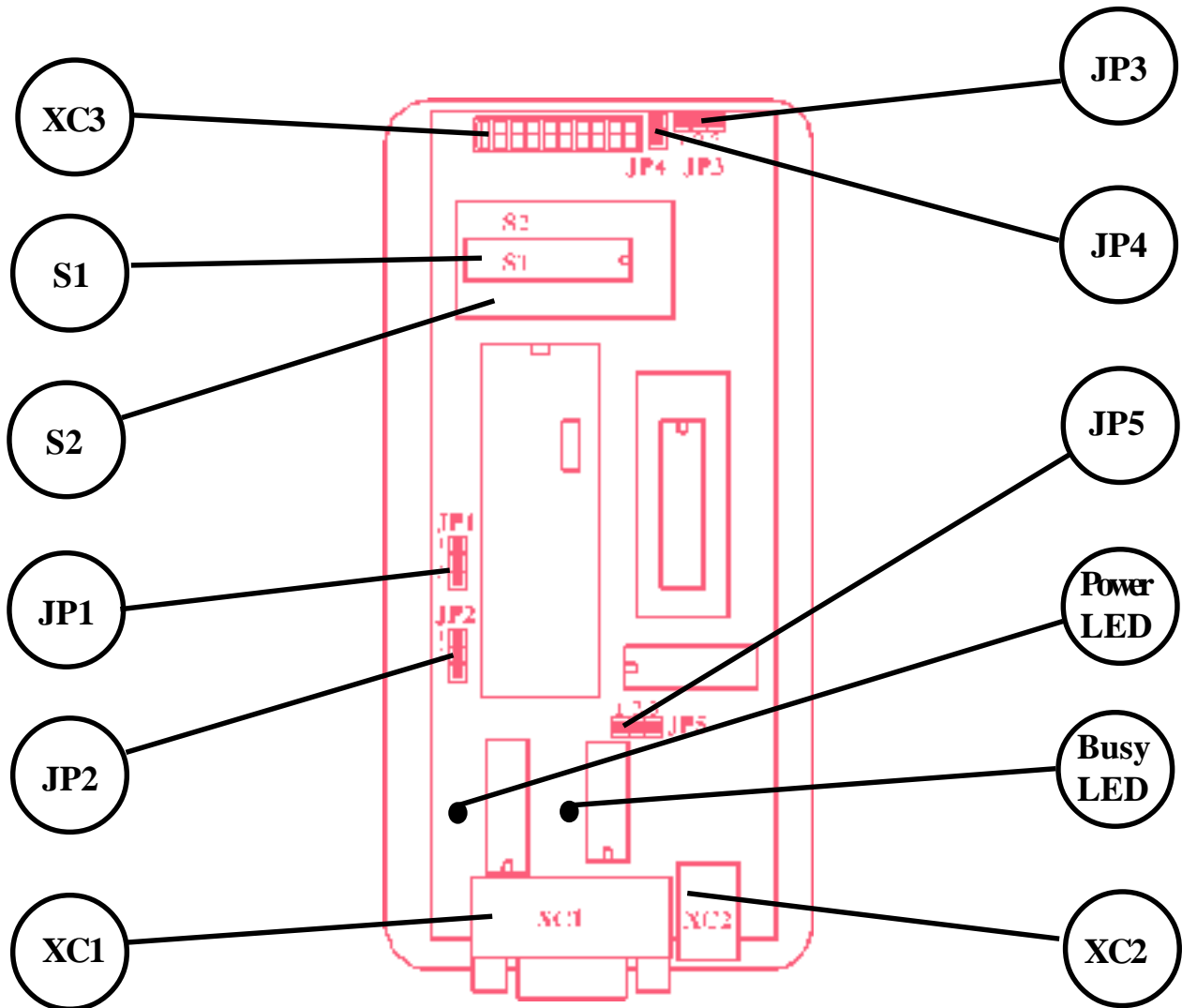


FIGURA 5: POSIZIONE JUMPERS, CONNETTORI E LEDs



FIGURA 6: POLARITÀ DEL CONNETTORE DI ALIMENTAZIONE

## AVVIO RAPIDO

Vengono qui presentate alcune procedure per mettersi rapidamente al lavoro. Ricordate che questo è solamente un sommario delle informazioni fornite nel capitolo “INSTALLAZIONE”, il quale contiene istruzioni dettagliate. Potete anche consultare il capitolo “SOLUZIONE DEI PROBLEMI” se riscontrate degli inconvenienti.

### SIMULAZIONE

- Spegnete il PC, l'**MPS 051** e la scheda target.
- Usate i jumper per scegliere se usare l'oscillatore, il segnale di RESET e l'alimentazione da scheda target o da fonti interne dell'**MPS 051**.
- Inserite il socket di simulazione nel socket della scheda target dedicato alla CPU AT89C2051 o AT89C1051. L'orientamento è indicato sull'adesivo attaccato al socket.
- Collegate l'**MPS 051** al PC.
- Alimentate la scheda target e, se necessario, anche l'**MPS 051**.
- Alimentate il PC.
- Avviate il programma di controllo; se necessario usate il menu Simulator/Find.
- Le tecniche di debugging vengono spiegate nel capitolo “DESCRIZIONE SOFTWARE”.

### PROGRAMMAZIONE

- Se i pin dello zoccolo di simulazione vengono usati come input, rimuovete il socket di simulazione dalla scheda target (né la scheda target né l'**MPS 051** possono essere alimentati durante tale operazione).
- Collegate l'alimentatore fornito al connettore coassiale dell'**MPS 051** e alimentatelo.
- Avviate il programma di controllo; se necessario usate il menu Simulator/Find.
- Inserite l'integrato da programmare nel socket di programmazione con l'orientamento indicato dalla serigrafia accanto al socket stesso.
- Caricate da disco il programma da scrivere usando il menu File/Load o premendo <F3> o, per copiare un dispositivo non protetto, leggetelo con il menu Programmer/Read o premendo <F7>.
- Date il comando di programmazione, il LED rosso deve accendersi. Al termine della programmazione il LED rosso deve spegnersi, i dati scritti possono essere verificati (a patto che i due bit di LOCK non siano stati programmati). L'**MPS 051** cancella automaticamente il circuito prima di programmarlo.
- Rimuovere il circuito programmato solo dopo lo spegnimento del LED rosso.

## DESCRIZIONE SOFTWARE

Questo capitolo contiene una descrizione completa del software di controllo, di come installarlo e di come utilizzarlo.

Il software fornito con il simulatore/programmatore **MPS 051** è stato sviluppato originariamente per il simulatore/programmatore **SIM 2051** dalla **ELNEC**, comunque la **grifo®** assicura una piena compatibilità ed intercambiabilità tra **MPS 051** e **SIM 2051**. In questo modo l'ottimo programma di controllo sviluppato dalla **ELNEC** può essere liberamente fruito anche con il nostro **MPS 051**.

Il file eseguibile del programma di controllo si chiama S2051.EXE ed è progettato per funzionare sotto DOS. Può anche essere eseguito in una finestra DOS sotto WINDOWS 95/98.

### REQUISITI MINIMI DEL PC

L'**MPS 051** può essere collegato ad un PC IBM compatibile. Il software dell'**MPS 051** richiede un sistema operativo MS/PC-DOS versione 3.2 o superiore.

I requisiti minimi dell'hardware sono:

- PC XT
- minimo 512 KBytes di RAM
- Un floppy drive da 3.5"
- Una porta seriale standard minimo da 57600 baud

### CONTENUTO DEL DISCO

Ogni **MPS 051** viene fornito con un dischetto contenente un programma di controllo ed un file di HELP. Si consiglia di fare una copia del contenuto del disco in una cartella specifica sul disco rigido. Il contenuto del disco può essere copiato liberamente. Viene fornito anche un cross-traduttore di tipo SHAREWARE che predetermina le proprie operazioni.

- S2051.EXE - programma di controllo per l'**MPS 051**
- S2051E.HLP - file di HELP
- ASM51\ - cartella contenente la versione shareware del traduttore ASM51
- EXAMPLES\ - cartella contenente gli esempi per il simulatore EXAMPLE1.ASM, EXAMPLE2.ASM

## LANCIARE IL PROGRAMMA DI CONTROLLO

Create una nuova cartella sul vostro disco rigido, copiateci il contenuto del floppy disk, selezionate la cartella come directory corrente poi:

Digitate S2051<**ENTER**> al prompt del DOS

Dopo aver lanciato il programma di controllo, viene verificato il checksum (CRC). Se va tutto bene, il programma inizierà a cercare l'**MPS 051** su tutte le porte seriali fisicamente installate sul computer. Su ogni porta la ricerca viene fatta a 57600 Baud, come viene indicato sul display. L'operazione di ricerca dura finchè non viene trovato un **MPS 051** o se non è connesso alcun **MPS 051**. Se viene trovato un **MPS 051** verrà mostrato un messaggio per l'utente, altrimenti verrà visualizzato un messaggio di errore indicante le possibili cause dell'errore. Dopo avere rimosso una delle possibili cause, premete un tasto, se fallisce di nuovo nel trovare l'**MPS 051** il programma entra in modalità DEMO, ovvero sono bloccati tutti i comandi di controllo nei menu Simulator e Programmer. **Se riuscite a rimuovere la causa dell'errore, potete dare il comando Find (menu Programmer) per forzare una ricerca dell'MPS 051 su tutte le porte seriali.**

Una volta trovato, l'**MPS 051** può essere usato come programmatore o come simulatore.

## AVVIO DEL PROGRAMMA CON PARAMETRI

Una volta avviato il programma cerca automaticamente l'**MPS 051** su tutte le porte seriali, ma questa procedura può essere prevenuta inserendo un parametro in linea di comando che specifica l'indirizzo della porta a cui è collegato l'**MPS 051**. Il parametro è nella forma /Ayyy, ove yyy è l'indirizzo della COM in esadecimale. Se il parametro compare più volte, ogni volta con un indirizzo diverso, il programma ottiene una lista di COMs dove cercare l'**MPS 051**. La lista può essere inserita anche col menu Simulator/Find durante l'esecuzione del programma. Inoltre, il parametro si può usare quando il programma è chiamato in un file BATCH. Ad esempio, una invocazione può essere:

S2051 /A3F8 /A2F8

## SIMULATORE

Il contenuto della RAM nell'**MPS 051** e del buffer del programma di controllo sono “sincronizzati”, ovvero il programma di controllo tenta di renderne i contenuti identici. Quindi, il contenuto della RAM esterna del simulatore viene aggiornato automaticamente prima di ogni comando (CALL, GOTO, SINGLE STEP) caricando i dati dallo spazio di lavoro del buffer riservato alla sezione simulatore (1000H-17FFFH o 13FFFH).

Inoltre, la sezione simulatore permette all'utente di acquisire e modificare i valori dei port P1 e P3 (con la sola eccezione di P3.6 che non viene usato nei processori AT89C2051). La rappresentazione viene data sia in binario sia in esadecimale.

I comandi CALL e GOTO vengono usati per il debugging del programma. Possono eseguire sottoparti specifiche del programma (sottoprogrammi) o il programma completo. Per eseguire sottoprogrammi (CALL) dovete conoscerne l'indirizzo iniziale; per eseguire il programma completo dovete passare il controllo (GOTO) all'indirizzo 1000H (questo indirizzo corrisponde allo 0000H nel circuito standard). Il comando CALL attende il ritorno dal sottoprogramma chiamato o il completamento del programma completo. Il comando GOTO semplicemente passa il controllo all'indirizzo specificato senza attendere il completamento della routine. Se un (sotto)programma debuggato entra in un loop senza fine l'**MPS 051** può venire resettato dando il comando RESET. Si può uscire da un loop a cui si è arrivati con un CALL premendo il tasto <ESC>. Il comando di RESET non modifica il contenuto della RAM esterna a bordo dell'**MPS 051**.

Il comando SINGLE STEP BY INT0 (INT1) viene usato per debuggare un (sotto)programma istruzione per istruzione (single step). Prima di eseguire uno di questi comandi (a seconda che si stia usando P3.2 /INT0 o P3.3 /INT1 per generate interrupts) si presuppone che venga disconnesso il pin che verrà usato per generare internamente l'interrupt. I sottomenu PortIn, PortOut, View/Edit internal RAM e Registers si possono usare durante le operazioni single step. Ulteriori informazioni sulle operazioni single step si possono trovare nei cataloghi dei processori serie MCS-51 o nella letteratura connessa..

Il resto di questo manuale presente la descrizione d'uso di questa modalità dell'**MPS 051**. Gli esempi contenuti nel disco dell'**MPS 051** forniscono informazioni importanti sulle modifiche del programma.

La RAM interna del processore può essere raggiunta sia in lettura che in scrittura tramite il comando View/Edit internal RAM. Qualunque tentativo di scrivere nei registri riservati al monitor interno dell'**MPS 051** viene bloccato. Tali registri comprendono il 20H e gli indirizzi dal 5FH al 7FH (stack).

Il comando BAUD imposta il valore degli opportuni registri dell'**MPS 051** per abilitare l'uso della porta seriale alla velocità di comunicazione selezionata. I dati in input/output possono essere raggiunti tramite il comando Register. Tale comando rende possibile leggere e modificare il registro SBUFF ed altri registri del processore.



Il comando FIND viene usato per cercare l'**MPS 051** presso le porte seriali fisicamente presenti sul PC. Sotto il suo menu si può anche trovare il comando di RESET del simulatore che può essere dato indipendentemente. Comunque, il comando RESET viene eseguito automaticamente dopo il completamento di determinate operazioni se sorge il sospetto che il processore si trovi in uno stato indefinito e se “si rifiuta” di comunicare (il RESET interno del programma di controllo non viene usato se è stato selezionato l'impiego di un RESET esterno; per tale scopo si può usare il meccanismo di RESET della scheda target).

La selezione degli oscillatori da impiegare (se interni o esterni sulla scheda target) si effettua mediante il comando OSCILLATOR. L'oscillatore esterno blocca l'accesso al dispositivo da programmare (non si possono usare PROGAM, READ, VERIFY and BLANK CHECK). L'oscillatore esterno viene selezionato tra quelli della lista che viene presentata a seconda della frequenza, con una tolleranza di  $\pm 2.5\%$ . La tolleranza indica la variazione ammessa sulla velocità di comunicazione PC-**MPS 051** la quale viene dedotta dalla frequenza del quarzo selezionato. Informazioni più dettagliate si possono ottenere dall'help sensibile al contesto premendo il tasto <F1>.

Il programma è preimpostato per l'utilizzo degli oscillatori interni. Supponendo che il vostro **MPS 051** sia configurato per funzionare con un oscillatore esterno, si verifica un errore di comunicazione (se il valore differisce da 11.0592 MHz) e il programma funziona in modo DEMO. Usate il comando OSCILLATOR per impostare il valore corretto dell'oscillatore esterno impiegato. Fatto questo, il programma trova l'**MPS 051**.

## NOTE SULLA SIMULAZIONE

Il programma di controllo può anche essere usato in modalità BATCH, passando il nome del file per il simulatore in linea di comando. In tal caso, il programma carica il contenuto del file (nel formato binario o Intel HEX) nella RAM esterna dell'**MPS 051**, lo esegue, e restituisce il controllo al sistema operativo. L'operazione è equivalente ad eseguire il comando GOTO con l'indirizzo 1000H come parametro. Il nome del file passato deve essere completo del path. Il formato del file viene dedotto dall'estensione; se questa è .HEX il file viene considerato un Intel HEX, altrimenti viene considerato in formato binario. Viene dato per scontato l'utilizzo di un AT89C2051 con oscillatore interno.

## ON LINE HELP

La descrizione individuale dei comandi del programma di controllo può essere visualizzata selezionando l'HELP (tasto <F1>). Premendo <F1> due volte si ottengono tutte le voci dell'Help, comprese quelle sull'HW di **MPS 051**.

## OPERAZIONI IN SINGLE STEP

La modalità single step dell'**MPS 051** è progettata per debuggare il programma utente istruzione per istruzione. Tale modalità è resa possibile dalle proprietà hardware dei processori serie MCS-51, in particolare la risposta del processore agli interrupt collegati agli input /INT0 e /INT1. Anche se generalmente la letteratura tecnica menziona segnali esterni per questi interrupt, possono anche essere generati internamente. Certamente, il programma di controllo dentro il processore dell'**MPS 051** tiene conto di questa possibilità. Comunque, questa tecnica di debugging è limitata dal fatto di non poter usare il piedino esterno associato all'interrupt selezionato (P3.2 o P3.3), il piedino deve essere disconnesso o, al più, usato come uscita.

Sono possibili tre stati nell'esecuzione di operazioni in single step: STEP, RUN e STEP OVER. Lo stato di STEP è usato per eseguire istruzioni singole nel programma di controllo. Lo stato di RUN è simile all'esecuzione normale con la differenza che è possibile inserire i cosiddetti BREAK POINTs (esempio, inserimenti nel codice standard) in corrispondenza dei quali il programma debuggato si ferma e passa in modalità STEP. Lo stato STEP OVER esegue istruzioni ACALL o LCALL in un singolo passo, ovvero senza mostrare l'esecuzione delle singole istruzioni del sottoprogramma.

Procedura:

Supponiamo di avere il compilato di un programma che deve essere debuggato in questa modalità. Caricate il codice buffer del programma di controllo dell'**MPS 051**. A seconda di quale pin avete deciso di disconnettere (P3.2 per /INT0 o P3.3 per /INT1), selezionate il rispettivo comando dal menu:

SIMULATOR/ SINGLE STEP BY INT0, o  
SIMULATOR/ SINGLE STEP BY INT1

Si fa notare che a tale pin non si deve fare riferimento all'interno del programma debuggato. Ci si aspetta che l'utente osservi con attenzione tutte le limitazioni e le raccomandazioni date sopra riguardo all'installazione di BREAK POINTs nel programma debuggato, per non rischiare una errata esecuzione del comando RUN.

La finestra di lavoro in modalità single step consiste di varie parti diverse. La parte superiore mostra i valori attuali dei registri del processore e dei port P1 e P3. Questi valori vengono aggiornati continuamente da letture fatte sul simulatore. Qualunque cambiamento dei registri o dei port (dovuto all'esecuzione di una istruzione o ad un intervento esterno) viene indicato con un diverso colore. La linea inferiore contiene la lista dei comandi utilizzabili in questa modalità. La linea corrente (col cursore al suo estremo) contiene sempre la prossima istruzione che verrà eseguita (premendo il tasto <ENTER> o <F7>). Inoltre in questa modalità è possibile modificare il contenuto della RAM interna e il contenuto di alcuni registri o ports. Premete <F1> se volete ulteriori informazioni.

Quando usate il comando RUN (tasti <ALT+U>) il programma debuggato può "perdere la via". Solitamente in questo caso è necessario ricorrere al reset hardware o usare il tasto <ESC> per ritornare al menu radice del programma di controllo. La stessa cosa si può fare quando è chiaro che il programma, in single step, non eseguirà le istruzioni che ci si aspetta esegua. In ogni caso il ritorno al menu radice viene preceduto da identificazione automatica del simulatore per preservare i parametri di comunicazione corretti verso l'**MPS 051**. Il programma verifica lo stato corretto dello stack, in caso di overflow viene terminato il single step mode, appare sullo schermo un messaggio di allarme e il sistema si pone in attesa di un reset hardware.

L'applicazione della modalità STEP OVER (tasto <F8>) è soggetta ad alcune restrizioni. Essa si basa sull'inserzione forzata di un break point dopo l'istruzione RET di un particolare sottoprogramma (ACALL o LCALL). Questo viene gestito da una parte della RAM esterna sull'MPS 051 compresa tra gli indirizzi 17F6H÷17FFH. Il programma di controllo trasferisce le rispettive istruzioni di CALL a questa area e aggiunge un break point specifico che comprendo un salto verso l'istruzione successiva alla CALL. Il controllo viene rediretto in quest'area sovrascrivendo la ACALL (LCALL) nella RAM esterna dell'MPS 051 con una AJMP (LJMP). Quando sono finite le istruzioni, la RAM esterna dell'MPS 051 in corrispondenza della ACALL (LCALL) viene riportata alle sue condizioni iniziali in modo da poter chiamare il sottoprogramma, ad esempio, in un loop. La modalità STEP OVER può essere usata solo se il programma da debuggare non supera l'indirizzo 17F5H, altrimenti la pressione di <F8> provocherà la sovrascrittura del programma utente agli indirizzi sopra menzionati. Non viene fatto alcun controllo di questa eventualità.

## NOTA DI UTILIZZO DELLO STEP OVER

Può capitare che il sottoprogramma chiamato da ACALL (LCALL) contenga un break point utente. Il programma lo accetta e si ferma. Se in questo momento usate il comando RUN (<Alt+U>) potete ritornare in maniera sicura dal sottoprogramma passando attraverso l'esecuzione del break point forzato nell'area riservata della RAM esterna dell'MPS 051. Il comando RUN può essere rimpiazzato dal single step (<F7>). Si dovrebbe notare comunque che dopo aver lasciato il sottoprogramma invocato con ACALL (LCALL) ci si troverà nell'area riservata al momento dell'esecuzione del break point forzato. Solo dopo il suo completamento il controllo ritorna all'istruzione seguente la ACALL (LCALL).

## INSTALLAZIONE DI BREAK POINT NEL PROGRAMMA DEBUGGATO

Sequenza del BREAK POINT per interrupt INT0\:

```
clr    P3.2
setb   EX0
nop
```

Sequenza del BREAK POINT per interrupt INT1\:

```
clr    P3.3
setb   EX1
nop
```

## LIMITAZIONI DEL PROGRAMMA IN SINGLE STEP

- Il pin P3.2 (per /INT0) o il pin P3.3 (per /INT1) non può essere usato in single step (il pin non utilizzabile dipende dall'interrupt usato).
- Non si possono disabilitare tutti gli interrupt (esempio con l'istruzione CLR EA).
- Non si può disabilitare l'interrupt esterno usato per il single step (esempio, l'istruzione CLR E0 non può essere usata nel programma debuggato se si usa /INT0 per il single step; l'istruzione CLR EX1 non può essere usata nel programma debuggato se si usa /INT1 per il single step).
- Non si può modificare la priorità degli interrupt, più precisamente l'interrupt usato per il single step deve avere priorità massima.

- Non si può cambiare il tipo di interrupt usato per il debugging (il tipo di interrupt viene impostato nel registro TCON).
- 22B dello stack sono disponibili per il debugging in single step mode.
- Se create il vostro stack l'indirizzo di ritorno per il monitor viene perduto dopo aver terminato il modo SINGLE STEP. In tale STEP stato, il programma ne da una segnalazione e RESETta automaticamentel'MPS 051. In modo RUN, normalmente si ha un loop senza fine.
- La routine di utility in debugging modifica i seguenti indirizzi nella RAM interna del processore: 20h, 59h, 5Ah, 5Bh, 5Ch, 5Dh, 5Eh.
- Impstando i BREAK-POINTS, la dimensione del programma debuggato non può eccedere la dimensione della memoria del processore emulato, tenendo conto delle limitazioni dovute al modo STEP OVER.

#### NOTE:

- Si ha un vantaggio utilizzando una macor per introdurre SW BREAK POINT.
- Per evitare frequenti modifiche al sorgente a seconda della tecnica di debugging usata si consiglia di usare la compilazione condizionale, per esempio, with variabili **MPS 051** per cambiare l'indirizzo iniziale del programma, e SSTEP per separare i SW BREAK-POINT.
- Esempi che mostrano le possibilità del single step mode si possono trovare nella cartella EXAMPLES sul dischetto.

### PROGRAMMATORE

La sezione di programmatore contiene tutti i comandi necessari per lavorare con la FLASH EPROM dell'AT89C1051 o dell'AT89C2051.

- Usate il comando SELECT per selezionare il tipo di circuito. Il default è AT89C2051.
- Il comando BLANK CHECK verifica se il circuito sullo zoccolo è cancellato.
- Il comando READ legge il contenuto del circuito nello zoccolo e lo salva nel buffer del programma di controllo entro il campo di indirizzi 0000H÷07FFH (03FFH), per esempio nell'area di lavoro del buffer riservata alla sezione programmatore.
- Il comando VERIFY verifica la programmazione appena effettuata.
- Il comando PROGRAM effettua la programmazione dal buffer del programma di controllo allo zoccolo di programmazione (entro il campo di indirizzi 0000H÷07FFH, o 03FFH). Inoltre, questo comando permette la programamzione dei bit di LOCK. S2051 cancella automaticamente un circuito prima di programmarlo. Il LED rosso sull'**MPS 051** indica che lo zoccolo programmatore è attivo.

#### !!! IMPORTANTE !!!

- I terminali dei port P1 e P3 sono in comune tra lo zoccolo del programmatore e quello del simulatore. Quindi, per evitare interferenze tra il dispositivo simulato e quello che viene programmato, la soluzione migliore è disconnettere il cavo di simulazione dal lato **MPS 051**.
- Il LED rosso (BUSY) dell'**MPS 051** indica che lo zoccolo del programmatore è attivo. Si consiglia di lasciare il circuito nello zoccolo di programmazione finchè il LED non si spegne.
- Si consiglia di non inserire nè togliere circuiti dallo zoccolo di programmazione finchè il LED rosso di BUSY rimane acceso.

La descrizione dei comandi individuali del programma di controllo può essere esaminata mediante l'help (tasto <F1>). Premendo <F1> due volte appare la lista completa delle opzioni dell'help, incluse le voci dell'HW di **MPS 051**.

## PROCEDURA CONSIGLIATA

### Simulazione:

- Usate i jumper per scegliere se usare l'oscillatore, il segnale di RESET e l'alimentazione da scheda target o da fonti interne dell'**MPS 051**.
- Inserite il socket di simulazione nel socket della scheda target dedicato alla CPU AT89C2051 o AT89C1051. L'orientamento è indicato sull'adesivo attaccato al socket.
- Collegate l'**MPS 051** al PC.
- Alimentate la scheda target e, se necessario, anche l'**MPS 051**.
- Alimentate il PC.
- Avviate il programma di controllo; se necessario usate il menu Simulator/Find.
- Le tecniche di debugging vengono spiegate nel capitolo "DESCRIZIONE SOFTWARE".

### Programmazione:

- Se i pin dello zoccolo di simulazione vengono usati come input, rimuovete il socket di simulazione dalla scheda target (né la scheda target né l'**MPS 051** possono essere alimentati durante tale operazione).
- Collegate l'alimentatore fornito al connettore coassiale dell'**MPS 051** e alimentatelo.
- Avviate il programma di controllo; se necessario usate il menu Simulator/Find.
- Inserite l'integrato da programmare nel socket di programmazione con l'orientamento indicato dalla serigrafia accanto al socket stesso.
- Caricate da disco il programma da scrivere usando il menu File/Load o premendo <F3> o, per copiare un dispositivo non protetto, leggetelo con il menu Programmer/Read o premendo <F7>.
- Date il comando di programmazione, il LED rosso deve accendersi. Al termine della programmazione il LED rosso deve spegnersi, i dati scritti possono essere verificati (a patto che i due bit di LOCK non siano stati programmati). L'**MPS 051** cancella automaticamente il circuito prima di programmarlo.
- Rimuovere il circuito programmato solo dopo lo spegnimento del LED rosso.

## ESEMPIO OPERATIVO

La cartella EXAMPLE sul dischetto contiene due esempi. Ne useremo uno, EXAMPLE1.ASM, per dimostrare il funzionamento di **MPS 051**.

Il programma debuggato legge i dati dalla porta seriale, li riporta su P1 e attende 1 ms, dopodichè scrivesul port il negato del valore ricevuto dalla porta seriale. L'overflow del Timer/Counter T0 genera un interrupt il quale attiva una routine che invia il byte 01H sulla porta seriale.

La struttura della RAM interna dell'**MPS 051** è identica alla struttura della memoria programma dei processori AT89C2051/1051, a parte il fatto che inizia dall'indirizzo 1000H. Tale differenza può essere aggirata così:

```

ADR    EQU    1000h    ;linea #1A per debugging
;ADR    EQU    0000h    ;linea #1B programmare il processore
BOC    EQU    002Bh    ;BEGIN_OF_CODE - indirizzo partenza programma
                        ;del processore AT89C2051/1051 (questo
                        ;indirizzo deve essere pari o superiore a
                        ;002Bh)
                        ; questa parte include le definizioni di variabili e costanti

                        ;inizio del codice, tutte le direttive ORG vanno scritte nella forma
                        ;ADR+... form

org    ADR+BOC org    ADR+0000h    ; indirizzo inizio programma dopo reset
ljmp  START    ; salta ad inizio programma
org    ADR+0003h    ; inizio vettore di interrupt da INT0
org    ADR+000Bh    ; inizio vettore di interrupt da T0
org    ADR+0013h    ; inizio vettore di interrupt da INT1
org    ADR+001Bh    ; inizio vettore di interrupt da T0
org    ADR+0023h    ; inizio vettore di interrupt da porta seriale
START:    ; prima istruzione del programma
END    ; fine programma
    
```

La compilazione del programma da debuggare con **MPS 051** deve essere condizionata come descritto nella nota di linea #1A, cioè il codice è compilato per iniziare dall'indirizzo 1000H come richiesto dal simulatore. Un programma così compilato non può funzionare nello spazio di indirizzamento degli AT89C2051/1051 che arriva fino a 07FFH. Quindi, per scrivere il programma sul dispositivo definitivo, bisogna ricompilarlo come da commento della linea #1B, ovvero da indirizzo 0000H.

## DEBUGGING DEL PROGRAMMA

- Compilate il sorgente (con la ORG per il simulatore).
- Usate il comando VIEW/EDIT BUFFER per rivedere e, se necessario, modificare i dati nell'area riservata al simulatore.
- Se tutto va bene, il programma può essere iniziato con una GOTO o una CALL all'indirizzo opportuno entro il range 1000H÷17FFH. Il programma di esempio EXAPMLE1 si aspetta un carattere dalla porta seriale. Dopodichè inizia la sequenza di eventi descritta all'inizio del capitolo. Alla fine dell'esecuzione sul port P1 rimane un valore negato. Lo si può verificare dando il comando INPUT con parametro P1.

## LIMITAZIONI HARDWARE DEL CIRCUITO AT89C2051

- Il simulatore emula un processore AT89C2051-12PC.
- Il simulatore non può funzionare con lo stesso campo di tensioni di alimentazione dei microcontrollori AT89C2051 o 1051. Quando alimentato con il trasformatore in dotazione la scheda target può funzionare con l'alimentazione minima tollerata dagli AT89C2051/1051 (2.7 V) a patto che non insorgano problemi col fatto che i resistori di pull up portano a 5V il livello logico 1 durante il processo di simulazione. Di sicuro è necessaria una alimentazione di almeno 1.9 V per i livelli logici. L'alimentazione del simulatore non influisce su quella della scheda target poichè sono separate da un diodo. Se il jumper JP4 è disconnesso, il valore della Vcc sulla scheda target può eguagliare il massimo valore tollerate dall'AT89C2051, ovvero 6V.
- Corrente massima per livelli logici bassi: 1.6 mA/0.45V solamente.
- Al contrario dell'AT89C2051, il simulatore non è provvisto di comparatore analogico.
- Per assicurare una comunicazione corretta tra **MPS 051** e il PC tramite porta seriale, il valore del cristallo sulla scheda target deve corrispondere a quello fornito nel menu Simulator/Oscillator/External.

Per rimediare alla mancanza del comparatore analogico, è possibile usare un altro emulatore hardware per la famiglia 51, il T-EMU 51, in abbinamento con l'adattatore qui sotto descritto.

- Adattatore per T-EMU 52 ed altri emulatori
- Convertitore 87C51/AT89C2051
- Conversione tra AT89C2051/4051 e 87C51
- Comprende un comparatore analogico
- Possibilità di scollegare P1.0/20-P1.0/40 e P1.1/20-P1.1/40

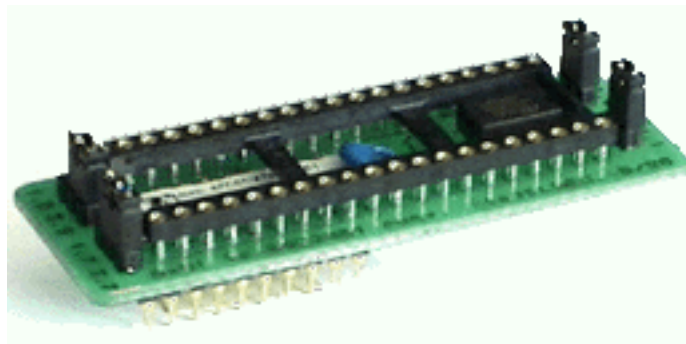


FIGURA 7: FOTO DELL'ADATTATORE PER T-EMU 52

## LIMITAZIONI SOFTWARE

- Lo stack del monitor inizia all'indirizzo 5FH della RAM esterna sull'**MPS 051**. Per il procedimento di debugging mediante istruzione CALL è disponibile a partire dall'indirizzo 63H, quindi può essere lungo 28 bytes. Se invece si usa l'istruzione GOTO, l'indirizzo è 61H, quindi può essere lungo 30 bytes. Se definite il vostro stack, l'indirizzo di ritorno per il monitor viene perso. Si consiglia, in tal caso, di ritornare mediante l'istruzione LJMP 0000 alla fine del programma. Ciò riporterà il monitor al suo stato iniziale.
- L'area RAM usata per memorizzare il programma da debuggare inizia dall'indirizzo 1000H.
- L'indirizzo 20H della RAM interna non può essere cambiato.
- Dato che il buffer contiene due aree di lavoro, e dato che la lettura di un file binario da disco non permette all'utente di distinguere tra area del simulatore e area del programmatore, viene introdotto un compromesso: ogni volta che un file viene caricato con successo, il contenuto dell'area del programmatore viene copiato nell'area del simulatore (la dimensione dipende da quale chip viene simulato). Questo viene fatto anche se l'utente ha cambiato i valori nel riquadro ADDRESS nella finestra di Load File.
- La seguente tabella i ritardi nell'esecuzione degli interrupts dovuti alla diversa modalità di chiamata a secondo della fonte usata. I valori di tempo sono riferiti a  $F_{osc}=11.0592$  MHz.

INTERRUPT	CICLI	CICLI DI CLOCK	DURATA
/INT0	8	96	8.68 ms
T0	2	24	2.71 ms
/INT1	8	96	8.68 ms
T1	2	24	2.71 ms
RI+TI	2	24	2.71 ms

**FIGURA 8: TABELLA RITARDI IN ESECUZIONE INTERRUPT**



## COMANDI DEL PROGRAMMA DI CONTROLLO

La seguente lista mostra tutti i comandi disponibili con il programma di controllo. Il nome del comando è affiancato dalla combinazione di tasti che lo invoca (se esiste) e da una breve descrizione di scopo e modalità di utilizzo. Si ricorda che la descrizione più aggiornata di questi comandi è disponibile tramite l'help in linea, che andrebbe consultato prima di questo manuale.

### MENU SIMULATOR

<i>Nome</i>	<i>Tasti</i>	<i>Funzione</i>
Input	<Alt+N>	Questo comando legge i port P1 e P3 e mostra i valori letti in binario ed esadecimale. P3.6 è sempre ad un valore logico alto.
Output	<Alt+P>	Usate questo comando per scrivere valori nuovi nei port P1 e P3 del simulatore.
Call	<Alt+C>	Questo comando inizia l'esecuzione di un sottoprogramma nella RAM esterna dell' <b>MPS 051</b> . Il comando attende il ritorno del sottoprogramma chiamato (istruzione RET), se il sottoprogramma entra in un loop infinito usate il comando RESET del simulatore o premete <ESC> per un corretto ritorno al monitor. La RAM esterno dell' <b>MPS 051</b> non viene influenzata dal RESET. L'indirizzo specificato deve cadere nel range 1000H÷17FFH per AT89C2051 oppure 1000H÷13FFH per AT89C1051. Prima di eseguire il salto il contenuto del buffer viene copiato nella RAM esterna dell' <b>MPS 051</b> . <b>Nota:</b> L'utente ha 28 bytes di stack a disposizione, se definisce un proprio stack l'indirizzo di ritorno per il monitor viene perduto. In tal caso si consiglia di terminare il programma con LJMP 0000 che riporta il monitor al suo stato iniziale.
Goto	<Alt+G>	Questo comando inizia l'esecuzione del programma contenuto nella RAM esterna dell' <b>MPS 051</b> a partire dall'indirizzo specificato. Il comando non attende il ritorno del programma. Non è possibile comunicare con il simulatore mentre il programma è in esecuzione. Se il programma entra in un loop infinito usate il comando RESET del simulatore o premete <ESC> per un corretto ritorno al monitor. La RAM esterno dell' <b>MPS 051</b> non viene influenzata dal RESET. L'indirizzo specificato deve cadere nel range 1000H - 17FFH per AT89C2051 oppure 1000H - 13FFH per AT89C1051. Prima di eseguire il salto il contenuto del buffer viene copiato nella RAM esterna dell' <b>MPS 051</b> . <b>Nota:</b> L'utente ha 30 bytes di stack a disposizione, se definisce un proprio stack l'indirizzo di ritorno per il monitor viene perduto. In tal caso si consiglia di terminare il programma con LJMP 0000 che riporta il monitor al suo stato iniziale.

<i>Nome</i>	<i>Tasti</i>	<i>Funzione</i>
Single step by int 0	<Alt+0>	Questo comando viene usato per debuggare i (sotto)programmi istruzione per istruzione. Bisogna ricordare che per generare l'interrupt interno viene usato il pin P3.2-/INT0 quindi non sarà disponibile per il programma. In questa modalità è possibile modificare il contenuto dei port P1 e P3 (ad eccezione di P3.2 e P3.6), vedere e modificare la RAM interna ed alcuni registri. Ulteriori informazioni sul single step si possono trovare nella documentazione tecnica dei microcontrollori della famiglia MCS 51. Nei programmi di esempio contenuti nel floppy si trovano informazioni sulle modifiche necessarie al single step. Si consiglia inoltre di leggere i paragrafi di questo manuale dedicati al debugging in modalità STEP.
Single step by int 1	<Alt+1>	Questo comando viene usato per debuggare i (sotto)programmi istruzione per istruzione. Bisogna ricordare che per generare l'interrupt interno viene usato il pin P3.3-/INT1 quindi non sarà disponibile per il programma. In questa modalità è possibile modificare il contenuto dei port P1 e P3 (ad eccezione di P3.2 e P3.6), vedere e modificare la RAM interna ed alcuni registri. Ulteriori informazioni sul single step si possono trovare nella documentazione tecnica dei microcontrollori della famiglia MCS 51. Nei programmi di esempio contenuti nel floppy si trovano informazioni sulle modifiche necessarie al single step. Si consiglia inoltre di leggere i paragrafi di questo manuale dedicati al debugging in modalità STEP.
View Edit internal RAM	<Alt+I>	Questo comando si usa per vedere (modo view) o modificare (modo edit) i dati nella RAM interna al simulatore (visualizzazione solo in modo DUMP) nel campo 00H÷7FH. Usate i tasti freccia per selezionare l'oggetto da modificare, i dati modificati sono visualizzati con un colore diverso. Questo comando non viene usato per mostrare il contenuto degli SFR (Special Function Registers). Cambiare il contenuto di certi indirizzi 20H e da 5FH a 7FH può essere pericoloso, poichè sono usati dall' <b>MPS 051</b> . L'utente viene avvisato mediante un messaggio di avvertimento.
Registers	<Alt+E>	Questo comando permette di visualizzare e/o modificare alcuni registri. I registri modificabili vengono visualizzati con un colore diverso ed è possibile specificare il nuovo valore diverso in esadecimale. Il nuovo valore si può dare in binario premendo <Alt+B>. Alcuni bit nei registri modificabili sono fissi. Ulteriori informazioni si possono trovare nella schermata del programma.
Find	<Alt+F>	Questo comando cerca l' <b>MPS 051</b> su tutte le porte seriali presenti. Si può anche usare per passare da DEMO a READY e viceversa.

<i>Nome</i>	<i>Tasti</i>	<i>Funzione</i>
Reset	<Alt+R>	Questo comando genera il segnale di reset per l' <b>MPS 051</b> . Viene eseguito automaticamente dopo il completamento di Simulator/Call e Simulator/Single Step. <b>Nota:</b> Il comando RESET non è disponibile se la fonte di reset è la scheda target.
Oscillator	<Alt+O>	Questo comando si usa insieme all'impostazione dei jumpers JP1 e JP2, che selezionano se usare gli oscillatori interni o esterni. Dopo avere selezionato con <Enter> una delle voci disponibili viene rifatto il collegamento con il <b>MPS 051</b> alla nuova velocità di comunicazione che dipende dalla frequenza del quarzo selezionato.
Automatic reload		Questo comando abilita o disabilita il caricamento del file debuggato se viene modificato prima dell'esecuzione di qualunque comando di simulazione (Simulator\Call, Simulator\Goto, Simulator\Single step by INT0, Simulator\Single step by INT1). Abilita anche il caricamento automatico del file all'avvio del programma di controllo dopo l'elaborazione del file S2051.SET. I nomi dei file e l'abilitazione del caricamento automatico vengono memorizzati nel file S2051.SET mediante il comando Quit/Yes & Save.

## MENU PROGRAMMER

Select	<F5>	Usate questo comando per selezionare quale dispositivo va programmato nello zoccolo del programmatore tra AT89C4051, AT89C2051 o AT89C1051.
Blank Chk	<F6>	Prima di attivare questo comando assicuratevi che lo zoccolo del simulatore sia scollegato da qualunque dispositivo. Questo comando serve per determinare se il dispositivo nello zoccolo del programmatore è stato cancellato. Se non è stato cancellato il programma di controllo manda una segnalazione acustica e scrive un avviso sullo schermo.
Read	<F7>	Prima di attivare questo comando assicuratevi che lo zoccolo del simulatore sia scollegato da qualunque dispositivo. Questo comando legge i dati dal dispositivo che si trova nello zoccolo del programmatore e li deposita nel buffer del programma di controllo a partire dall'indirizzo 0H, a patto che non siano impostati i LOCK bits. La fine della lettura viene segnalata acusticamente.

<i>Nome</i>	<i>Tasti</i>	<i>Funzione</i>
Verify	<F8>	Prima di attivare questo comando assicuratevi che lo zoccolo del simulatore sia scollegato da qualunque dispositivo. Questo comando verifica che i dati scritti nel dispositivo inserito nello zoccolo del programmatore corrispondano ai dati memorizzati nel buffer del programma di controllo a partire dall'indirizzo 0000H. Alla fine della verifica il programma di controllo manda una segnalazione acustica e se ci sono delle differenze scrive un avviso sullo schermo mostrando la prima differenza.

Program	<F9>	Prima di attivare questo comando assicuratevi che lo zoccolo del simulatore sia scollegato da qualunque dispositivo. Questo comando scrive i dati contenuti nel buffer del programma di controllo a partire dall'indirizzo 0000H nel dispositivo che si trova nello zoccolo di programmazione. Si possono programmare volendo anche i LOCK bits:
---------	------	--

LB1	LB2	
0	0	FLASH EPROM non protetta
1	0	FLASH EPROM protetta dalla programmazione
1	1	FLASH EPROM protetta da programmazione e lettura

Come prima cosa il comando controlla il codice di identificazione del dispositivo, poi lo cancella e lo programma, infine verifica.

## MENU FILE

Load	<F3>	Analizza il formato del file specificato e lo carica nel buffer del programma di controllo. Potete scegliere il formato desiderato (Binary, MOTOROLA, MOS Technology, ASCII space, Tektronix ed Intel (Extended) HEX).
------	------	--

Save	<F2>	Salva il buffer del programma di controllo anche dopo che è stato modificato dall'utente in un file specificato. Potete scegliere il formato desiderato (Binary, MOTOROLA, MOS Technology, ASCII space, Tektronix ed Intel (Extended) HEX).
------	------	---

## MENU BUFFER

View code	<Ctrl+F4>	Questo comando permette di vedere il buffer come istruzioni MCS 51. Usate i tasti freccia per vedere i dati nel campo di indirizzi 1000H÷13FFH (o 17FFH a seconda del dispositivo emulato). Appena attivato il comando mostra i dati a partire dall'indirizzo 0000H.
-----------	-----------	--

<i>Nome</i>	<i>Tasti</i>	<i>Funzione</i>
View/Edit	<F4>	Questo comando visualizza (modo view) o modifica (modo edit) i dati nel buffer del programma di controllo (per visualizzare nel solo modo DUMP). Usate i tasti cursore per selezionare l'oggetto da modificare. I dati modificati vengono mostrati con un colore diverso.
Fill block		Usate questo comando per riempire il blocco selezionato del buffer del programma di controllo con il valore esadecimale (o ASCII) richiesto. Potete impostare il punto d'inizio e quello di fine del blocco.
Copy block		Questo comando permette di copiare blocchi di dati specificati da un indirizzo ad un altro nel buffer del programma di controllo. L'indirizzo di destinazione può ricadere all'interno del blocco che viene copiato. Vedere anche Move block.
Move block		Questo comando sposta blocchi di dati specificati da un indirizzo ad un altro nel buffer del programma di controllo. L'indirizzo di destinazione può ricadere all'interno del blocco che viene spostato. Il blocco di partenza viene riempito con il carattere di blank. Vedere anche Copy block.
Swap block		Questo comando scambia byte alto e byte basso nel blocco selezionate nel buffer del programma di controllo. Il blocco deve iniziare da un indirizzo pari e deve contenere un numero pari di bytes. Se tali condizioni non sono soddisfatte il programma attua automaticamente un compromesso (l'indirizzo iniziale diventa il primo pari precedente a quello dato e/o l'indirizzo finale il primo dispari successivo a quello dato).
Erase buffer	<Ctrl+F2>	Questo comando riempie l'intero buffer del programma di controllo col carattere di blank.
Checksum		<p>La checksum viene calcolata dopo avere specificato gli indirizzi esadecimali del blocco su cui calcolarla:</p> <p>           BYTE           - somma di bytes in "word". CY ignorato.            WORD           - somma di words in "word". CY ignorato.            BYTE (CY)      - somma di bytes in "word". CY sommato.            WORD (CY)      - somma di words in "word". CY sommato.         </p> <p>La colonna NEG contiene la negazione della checksum:            SUM + NEG = FFFFH</p> <p>La colonna SUPPL contiene il complemento della checksum:            SUM + SUPPL = 0000H (+carry)</p>

*Nome*
*Tasti*
*Funzione*
**MENU OPTIONS**

Buffer name	<p>Da questo comando si può impostare il pathname completo del file di buffer su disco in caso non dovesse bastare la memoria centrale del PC o della finestra DOS. Non viene effettuato alcun controllo, se il nome inserito è errato il problema verrà rilevato solo nel momento in cui il programma di controllo tenterà di creare il file. Questa impostazione viene salvata nel file S051.CFG dal comando Options/Save options. Il nome di default è S051.\$0.</p>
Monitor	<p>Questo comando imposta i colori del display VGA usato (monocromatico o a colori). Il cambiamento è immediatamente visibile. Non usate questo comando con schede HERCULES, la scheda viene identificata immediatamente. Questa impostazione viene salvata nel file S051.CFG dal comando Options/Save options. Il default è a colori.</p>
Sound	<p>Questo comando imposta la durata dei segnali audio che accompagnano le varie schermate testuali informative. Questa impostazione viene salvata nel file S051.CFG dal comando Options/Save options. Si può scegliere tra:</p> <ul style="list-style-type: none"> <li>Options\Sound\Long (default)</li> <li>Options\Sound\Short</li> <li>Options\Sound\None</li> </ul>
All Hex loadings	<p>Questo comando imposta varie opzioni per il controllo sul caricamento di file in formato HEX. La prima controlla il cancellamento automatico del buffer del programma di controllo, la seconda imposta l'offset negativo usato per modificare l'indirizzo di caricamento dei dati in formato HEX in modo da poterli caricare sempre negli indirizzi del buffer.</p> <p>Per esempio: un file contiene dati in formato Motorola S con indirizzo iniziale FFFF0H. Il formato S2 usato ha 3 bytes di address array length. Se viene impostato il valore di offset negativo FFFF0H, questo verrà sottratto al valore specificato nel file quindi il caricamento nel buffer del programma di controllo avverrà a partire dall'indirizzo 0000H.</p> <p><b>Attenzione:</b> dato che l'indirizzo viene sottratto dall'indirizzo di caricamento il risultato può essere un numero negativo. Si prega di impostare tale valore con prudenza. Questa impostazione viene salvata nel file S051.CFG dal comando Options/Save options. Come default è disattivato.</p>

<i>Nome</i>	<i>Tasti</i>	<i>Funzione</i>
Intel HEX loading		Questo comando controlla il caricamento dei dati nel buffer del programma di controllo dai files Intel HEX. Quindi all'utente verrà richiesto di inserire un valore per il segmento associato al segmento basso nel file (record di tipo 02) e tutti gli altri segmenti verranno modificati di conseguenza. Per esempio: il file contiene due record di tipo 02 con indirizzi F000H e F800H. Inserendo un nuovo valore per il segmento, ad esempio 0000H, i dati del segmento F000H verranno scritti a partire dall'indirizzo 0000H e i dati del segmento F800H verranno scritti a partire dall'indirizzo 0800H. Questa opzione non si applica ai "semplici" files in formato Intel HEX. Questa impostazione viene salvata nel file S051.CFG dal comando Options/Save options. Di default è inattiva.
Help		Usate questo comando per installare l'help on line sensibile al contesto nella lingua desiderata. Si può scegliere tra inglese, tedesco e lingua slovacca in due diverse translitterazioni. L'help può essere installato in ogni momento e l'impostazione viene salvata nel file S2051.CFG. Si dà per scontata la presenza su disco del file .HLP che viene fornito col disco di distribuzione.
Set Masks		Usate questo comando per impostare le estensioni dei file da associare al tipo del file. La maschera deve contenere uno dei caratteri jolly del DOS (*, ?). Questa impostazione viene salvata nel file S051.CFG dal comando Options/Save options.
Save options		Questo comando salva le impostazioni di tutte le altre voci del menu options. Tutte le opzioni vengono memorizzate nel file di configurazione chiamato S2051.CFG. Potete decidere se il file deve essere creato nella cartella corrente, nella cartella radice di C: o nella cartella da cui S2051.EXE è stato lanciato. Il primo posto dove viene cercato il file di configurazione è la cartella corrente, seguono la cartella radice di C: poi la cartella da cui S2051.EXE è stato lanciato. Se il file non viene trovato vengono usate le impostazioni di default.
Retrieve options		Questo comando permette di ripristinare le impostazioni salvate nel file S2051.CFG con il comando Options/Save options. Potete scegliere da dove verrà letto il file di configurazione (cartella corrente, cartella radice di C: o cartella da cui è stato lanciato S2051.EXE).

*Nome**Tasti**Funzione***MENU QUIT**

No		Cancella la richiesta di uscire dal programma.
Yes		Dealloca lo heap, cancella il buffer su disco (se esiste) e ritorna al sistema operativo.
Yes & Save		Dealloca lo heap, cancella il buffer su disco (se esiste), salva nel file PG4U.SET, che si trova nella cartela corrente, gli ultimi 10 dispositivi selezionati e ritorna al sistema operativo.

**MENU ABOUT**

Scegliere il menu ABOUT fa apparire una finestra che mostra informazioni sul copyright e sulla versione del programma.

**HELP**

Premendo il tasto <F1> si accede all'help in linea sensibile al contesto. Se è in corso una operazione del programmatore la pressione di <F1> non genera una risposta. Premendo <F1> con la finestra di help già aperta verranno visualizzati tutti i messaggi di help disponibili, in modo da essere selezionabili e leggibili. Potrebbero essere evidenziate in grassetto delle parole chiave, selezionandole si possono ottenere ulteriori informazioni (cross references).

Gli elementi evidenziati sono:

- Parole che si riferiscono a comandi descritti nell'help corrente
- Tutte le altre parole significative
- Cross reference corrente; premete <Enter> per ottenere ulteriori informazioni
- Cross-references non selezionati; per selezionarlo usate i tasti freccia e confermatelo con <Enter>

Poichè il sistema di help viene continuamente aggiornato con il programma di controllo, può contenere informazioni non incluse in questo manuale.

Informazioni dettagliate sui comandi individuali dei menu si possono trovare nell'help in linea integrato.



## TERMINI DELLA GARANZIA

Il produttore assicura un periodo di 12 mesi esente da difetti del dispositivo a partire dalla data di consegna evidenziata nella bolla.

Se un **MPS 051** evidenzierà un difetto di fabbricazione, verrà riparato se coperto da garanzia.

Si prega di contattare la **grifo®** per accordarsi sulle modalità di spedizione.

Per accettare l'articolo in riparazione è comunque indispensabile restituirlo con la confezione originale in buono stato e tutto il suo contenuto, ovvero cavo di comunicazione e alimentatore, una descrizione completa ed esauriente del problema riscontrato e delle circostanze in cui si è verificata:

- Versione del programma di controllo
- Possibili interferenze ambientali (uffici, laboratori industriali ed altri tipi di ambiente)
- Descrizione completa della configurazione hardware del PC
- Descrizione completa della configurazione software del PC

Se il materiale giunge senza descrizione del problema la riparazione può essere respinta.

Analogamente, non si può garantire la riparazione se non vengono inviati anche tutti gli accessori esterni usati quando è stata riscontrato il difetto. Richieste di riparazioni non necessarie o fuori garanzia verranno addebitate.

La garanzia non copre danni prodotti da usura, manipolazione da parte di personale non qualificato, danni meccanici, modifiche o riparazioni non autorizzate, danni occorsi durante un trasporto o contatto con sorgenti di alta tensione quali linee di alimentazione o generatori.

Il servizio assistenza clienti è garantito durante ed oltre il periodo di garanzia.

## RISOLUZIONE DEI PROBLEMI

In caso insorgano dei problemi leggete attentamente le operazioni di installazione e di utilizzo di hardware e software un'altra volta, probabilmente la risposta al vostro problema si trova lì. Se il problema dovesse persistere, riferitevi alle indicazioni qui sotto.

### ERRORI DI COMUNICAZIONE

- Il dispositivo deve essere alimentato correttamente, quindi il LED verde di alimentazione deve essere acceso e l'alimentatore usato deve essere quello fornito in dotazione. Per testare l'integrità del cavo torcetelo delicatamente durante il funzionamento.
- Aggiornate spesso il programma di controllo. I miglioramenti apportati includono anche maggiore affidabilità nella comunicazione. Riferitevi al paragrafo "AGGIORNAMENTI SOFTWARE GRATUITI" per ulteriori informazioni.
- Il cavo di comunicazione seriale usato deve essere quello fornito in dotazione o un cavo **con tutti i contatti collegati**. Per testarne l'integrità torcetelo delicatamente mentre il programma di controllo cerca di collegarsi con il dispositivo e vedete se qualcosa cambia.
- Provate ad installare il programma di controllo su un altro computer. Se il sistema funziona correttamente su un altro computer allora il problema risiede nell'altro computer. Confrontate le due macchine.

### PROBLEMI IN LETTURA O SCRITTURA

- Assicuratevi che il dispositivo sia allineato correttamente consultando la serigrafia presso lo zoccolo in cui inserite il dispositivo.
- Aggiornate il programma di controllo.
- Se il target da programmare non è mai stato usato prima o è stato cancellato si trova nello stato di blank. Verificate l'evenienza con il menu Programmer/Blank Chk o premendo il tasto <F6>.
- In alcuni casi il contenuto della FLASH EPROM potrebbe essere protetto. Ciò rende assolutamente impossibile leggerlo, tuttavia non si tratta di un malfunzionamento.

## STRUMENTI AGGIUNTIVI

I programmatori della **grifo**® possono essere affiancati con vari strumenti aggiuntivi per espanderne il campo di applicazione e creare suite complete perfette per l'uso professionale ove sia necessario cancellare, programmare o simulare i più diffusi tipi di EPROM ed i più famosi microcontrollori.

### EP 32

Low cost Eprom/universal Programmer 32 pin devices

EP 32 è un potente programmatore di EPROM, EEPROM, Flash EPROM, RAM tamponate ed EEPROM seriali, progettato per applicazioni di tipo professionale. Inoltre l'EP 32, mediante moduli di supporto, può programmare anche microprocessori (MCS48, MCS51, PIC, AVR), GALs, ecc.

### MP AVR-51

Micro Programmer for families AVR and 51

MP AVR-51 è un piccolo ma potente programmatore per microprocessori della serie MCS51 ed Atmel AVR. MP AVR-51 è anche in grado di programmare EEPROM seriali con interfaccia di tipo IIC (24Cxx), Microwire (93Cxx) ed SPI (25Cxx). Il programmatore è equipaggiato con uno zoccolo ZIF DIP da 40 pin. La qualità del programmatore è integrata da un comodo programma di controllo.

### MP PIK

Micro Programmer per PIC Microchip

MP PIK è un piccolo ma potente programmatore per microcontrollori Microchip PIC. MP PIK è anche in grado di programmare EEPROM seriali con interfaccia di tipo IIC (24Cxx), Microwire (93Cxx) ed SPI (25Cxx). Il programmatore è equipaggiato con uno zoccolo ZIF DIP da 40 pin. La qualità del programmatore è integrata da un comodo programma di controllo.

### UEP 48

Universal Eprom Programmer 48 pin devices

UEP 48 è un programmatore universale capace di supportare ogni genere di tecnologie del silicio e dispositivi programmabili. Il potente pilotaggio dei pin fornisce livelli logici, pull up/pull down, clock, ground, una tensione Vcc e due tensioni di programmazione su ognuno dei 48 pin indipendentemente. Questo concetto di progettazione avanzato permette di programmare quasi ogni dispositivo programmabile in formato DIL fino a 48 pin senza bisogno di adattatori specifici.

### SEEP

Serial EEPROMs Programmer

SEEP è un piccolo ma potente programmatore per EEPROM seriali in formato 8 pin. SEEP mette in grado di programmare EEPROM seriali con interfaccia di tipo IIC (24Cxx), Microwire (93Cxx) ed SPI (25Cxx). Viene supportata la programmazione LV EEPROM (3.3 V). Il programmatore è corredato con zoccolo ZIF. La qualità del programmatore è integrata da un comodo programma di controllo.

### ER 05

Eprom Eraser 05

Cancella tutte le EPROM cancellabili con raggi UV; può cancellare fino a 5 dispositivi; timer preprogrammato con tre diverse durate per risparmiare la lampada; fornito di alimentatore.

## INDICE ANALITICO

**A**

ACALL 17  
ALIMENTAZIONE 4, 8  
AT89C1051 3, 4  
AT89C2051 3, 4  
AT89C4051 3, 4  
AVVIO RAPIDO 11

**B**

BREAK POINT 17

**C**

CALL 14  
CARATTERISTICHE ELETTRICHE 4  
CARATTERISTICHE FISICHE 4  
CARATTERISTICHE GENERALI 3, 4  
CARATTERISTICHE TECNICHE 4  
CAVO DI CONNESSIONE 6  
COMANDI DEL PROGRAMMA DI CONTROLLO 23  
COMPARATORE ANALOGICO 21  
CONSUMO DI CORRENTE 4  
CONTENUTO DEL DISCO 12  
CONTENUTO DELLA CONFEZIONE 6  
CONVENZIONI 2

**D**

DEBUGGING 21  
DESCRIZIONE SOFTWARE 12  
DIMENSIONI 4

**E**

ERRORI DI COMUNICAZIONE 32  
ESD 6  
ESEMPIO OPERATIVO 20

**F**

FREQUENZA OSCILLATORE 4

**G**

GARANZIA 31  
GOTO 14

**H**

HELP 15, 30

HEX 2

**I**

INDICE ANALITICO 34

INFORMAZIONI PRELIMINARI 2

INSTALLAZIONE 6

INSTALLAZIONE HARDWARE 8

INT0 14

INT1 14

INTRODUZIONE 1

**J**

JUMPERS 9

**L**

LCALL 17

LEDS 9

LIMITAZIONI HARDWARE 21

LIMITAZIONI SOFTWARE 22

**M**

MASSA 4

MENU ABOUT 30

MENU BUFFER 26

MENU FILE 26

MENU OPTIONS 28

MENU PROGRAMMER 25

MENU QUIT 30

MENU SIMULATOR 23

MICROCONTROLLORI PROGRAMMATI 4

MICROCONTROLLORI SIMULATI 4

**P**

P3.2 14

P3.3 14

PROBLEMI IN LETTURA O SCRITTURA 32

PROGRAMMATORE 18

PROGRAMMAZIONE 11

**R**

REQUISITI MINIMI DEL PC 12  
RESET 8  
RISOLUZIONE DEI PROBLEMI 32  
RS 232 2, 10

**S**

SIMULATORE 14  
SIMULAZIONE 11  
SINGLE STEP 16  
STEP OVER 17  
STRUMENTI AGGIUNTIVI 33

**T**

TEMPERATURA 4  
TERMINOLOGIA 2

**X**

XTAL 8

**Z**

ZIF 2, 4, 10

## INTRODUCTION

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

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

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

Information provided in this manual is intended to be accurate at the moment of release, but we continuously improve all our products. Please consult the documentation file on the enclosed floppy disk for last minute changes.

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

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

**grifo®** assumes no responsibility for misuse of this manual.

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

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

## PRELIMINARY INFORMATION

This manual explains how to install the control program and how to use Your programmer. It is assumed that the User has some experience with PCs and installation of software, however the “Quick start” chapter will guide You step by step through the complete installation process.

Once you have installed the control program we recommend You consult the context sensitive HELP within the control program rather than the User’s Manual. Revisions are implemented in the context sensitive help before the User Manual.

The programmer works with almost any IBM compatible PC, from XT to Pentium Pro, portable or desktop personal computers. No special interface card is required to connect to the PC since programmers uses the serial port.

All programmers function flawlessly on systems running DOS, Windows 3.x and Windows 95/98. The programmer is driven by an easy-to-use control program with pull-down menus, hot keys and on-line help.

## CONVENTIONS AND TERMINOLOGY

There are some special conventions and terms used in this manual:

### CONVENTIONS USED IN THE MANUAL

References to the control program functions are in capitals, e.g. **LOAD**, **FILE**, etc. References to control keys are written in brackets <>, e.g. <**F1**>.

### TERMINOLOGY USED IN THE MANUAL

<b>ZIF socket</b>	Zero Insertion Force socket used for insertion of target device.
<b>BUFFER</b>	Part of memory or disk, used for temporary data storage.
<b>RS 232 serial line</b>	Kind of PC port (serial), which is primarily dedicated to serial devices connection (mouse, modem, scanner, etc.).
<b>HEX data format</b>	Format of data file, which may be read with standard text viewers; e.g. byte 5AH is stored as characters '5' and 'A', which means bytes 35H and 41H. One line of this HEX-file (one record) contains start address, data bytes and all records are secured with checksum.



## GENERAL FEATURES

**MPS 051** is a simulator/programmer of **ATMEL**'s single-chip **MCS-51** standard microcontrollers, types **AT89C2051** and **AT89C1051**.

**MPS 051** provides fundamental comfort for work with **ATMEL**'s **AT89C2051** and **AT89C1051** single-chip microcontrollers. It eliminates the need for frequent removing, reprogramming and inserting individual circuits. In addition, it enables the user to program the circuit by debugged data after completing his work. The unit is supplied from an application (sufficient for the **SIMULATION** part), or from an external power source of 14..25V/200mA. It is connected to a PC via an **RS232** serial line.

The **SIMULATOR** part design makes use of the unused ports of the standard 40-pin Series 51 processor for the **AT89C2051** circuit simulation. This function is performed by the **AT89C51** processor provided with an **external RAM** memory. The latter is used for storing and executing User's programs. The processor contains a resident monitor that communicates with the control program, operates the internal and the external RAMs and the registers, selects a mode, and modifies port states. An important advantage consists in the possibility to **step** the program being debugged by individual instructions, and to modify the content of the internal RAM of the processor and the registers. The monitor does not use interrupts. Consequently, all of them are available to the user, though slightly decelerated by the **LJMP** instruction to the external RAM area. The simulator is connected to the respective application by a flat cable with a crimp socket.

The **PROGRAMMER** part provides all programming-related functions, including reading, blank check, **FLASH** programming, **LOCK bit** programming, program verification, and circuit erasing.

All **MPS 051** functions are controlled by a comfortable control program similar to that used for programmers and simulators. The control is menu-driven and makes use of hot-keys. The program contains an internal buffer of program allowing for binary form editing. Furthermore, the buffer can be scanned in the **MCS51** source form; files of different formats can be read/written; etc.

## TECHNICAL FEATURES

### GENERAL FEATURES

<b>Microcontrollers simulated:</b>	ATMEL AT89c2051 ATMEL AT89c1051
<b>Microcontrollers programmed:</b>	ATMEL AT89c4051 ATMEL AT89c2051 ATMEL AT89c1051
<b>Serial cable:</b>	2 m

### PHYSICAL FEATURES

<b>Dimensions:</b>	132 x 66 x 30 [mm]
<b>Mass:</b>	120 g (MPS 051 only)
<b>Temperature range:</b>	0÷40 °C
<b>ZIF socket:</b>	24 pins
<b>Emulation socket:</b>	20 pins

### ELECTRIC FEATURES

<b>Power supply from application:</b>	5V ± 10%
<b>Power supply from external source:</b>	14÷25V/200mA
<b>Power supply provided:</b>	230 Vac/12 Vdc
<b>Current consumption:</b>	Simulator - 90 mA max., 50 mA default Programmer - 140 mA max., 90 mA default
<b>Connectability:</b>	RS 232 serial line
<b>Communication rate with PC:</b>	57600 Baud/11.0592 MHz
<b>Oscillator frequency:</b>	24 MHz max.



FIGURE 1: MPS 051 PHOTO

## INSTALLATION

These paragraphs contain all the information essential to connect **MPS 051** to the PC and install the software. **Please read completely these paragraphs before attempting any use of Your programmer.**

### CAUTION FOR ESD (ELECTROSTATIC SENSITIVE DEVICE)

**Caution!** Faults due to a User's failure to observe these precautions are not covered by the guarantee.

Simulator handling safety precautions aimed at preventing it from electric damage.

The simulator may be damaged by electrostatic discharge or transient current. It is advisable to follow the following precautions, or equivalent rules:

- Prior to operating your simulator, touch a larger metal, or a grounded object.
- Never touch simulation base outlets if you are not certain about zero potential relative to the simulator. This also applies to all simulator-internal circuits and the connecting cable outlets.
- Do not remove the antistatic sponge protection from the simulation socket unless necessary.
- When connecting the simulator to another device, check the grounds of both units for zero potential. Ideally, the grounds should be connected by a thicker conductor.
- The simulator and the device to be connected must be OFF. This also holds for interconnecting the simulator and the PC.

### DELIVERY CONTENTS

The package You have been delivered must contain at least the following items:

- 1 **MPS 051** simulator, including an emulation cable (including a ZIF socket)
- 1 diskette 3.5" with utility software
- 1 diskette 3.5" with this manual in pdf format
- mains adapter
- RS 232 connection cable
- transport package

### CONNECTION CABLE

The relatively high communication rate between the PC and the **MPS 051** requires a high-quality connection cable, like the one in the delivery package. It must be shielded (shielding connected to GND), and its length should not exceed 3 m.



FIGURE 2: DELIVERY CONTENTS

## HARDWARE INSTALLATION

At the beginning, the User must decide whether he/she prefers cycling (XTAL) and controlling the RESET signal of the simulator from the debugged application to their generating internally in the **MPS 051**. Connect the JP1, JP2, and JP3 jumpers according to your decision. The respective settings are specified in the table of figure 2. To easily locate jumpers JP1÷3 please refer to figure 4. The jumpers are located on the PCB inside the two plastic shells, so they must be carefully separated. Insert one end of the RS 232 connection cable to the **MPS 051**, and the other end to the RS 232 interface connector (COM 1÷4) of your computer, and use an appropriate reduction if needed. The **MPS 051** simulator/programmer is fitted with a 20-pin simulation socket. Its orientation follows from the sticker. The colour wire corresponds to pin No. 20. Insert the simulation socket into the simulated microcontroller socket as an AT89C2051/1051 circuit. The debugged system and the PC must be commonly grounded. Both the **MPS 051** and the device under development must be OFF during this operation.

### NOTE:

In case of need, proceed as follows: insert the **MPS 051** simulation socket into the device under development in such a way that pin No. 10 (GND) is the first to come into contact with the device socket. Alternatively, first connect your **MPS 051** to the device under development, and only then connect it to the PC - the RS232 interface is more resistant than processor ports.

*An inappropriate connection procedure may result in damaging the processor in the **MPS 051** by equalizing currents between the **MPS 051** and the PC!!!*

It is necessary for you to select the method of **MPS 051** supplying: either from the system being debugged or from an external power supply. The JP4 jumper is designed for cutting off the **MPS 051** supply from the simulation socket. Remove it if the application supply may exceed 5V. Certainly, the **MPS 051** (from an external power supply) and the application (from a different power supply) may be supplied separately, with the JP4 jumper connected, provided that the above given condition is met. To easily locate jumpers JP4 please refer to figure 4.

When the **MPS 051** and the computer are connected, and the simulation socket is inserted in the system being debugged, you can switch ON the power supply. When using an external power supply, insert the power supply cable to a mains socket, and the coaxial connector to the **MPS 051**. The green LED on the **MPS 051** lights up. If you prefer supplying from the application, switch the application power supply ON. The polarity of the external power supply is shown in figure 5.

Now you can start the S2051.EXE control program. Please refer to the chapter “SOFTWARE DESCRIPTION” for more information about software management.

### NOTES:

- The JP5 switch on the **MPS 051** board is factory-preset depending on the RAM used. Therefore, the User should not change its setting. A change in its configuration may damage the **MPS 051**!
- The operation of the **MPS 051** is preconditioned by the use of an internal crystal of 11.0592 MHz, or an external crystal of permitted frequency in accordance with the table in the control program.
- When the power supply is switched ON, the programming socket is in an idle state thanks to the resetting circuit and the control program in the processor.

**JUMPERS**

MPS 051 is provided with 5 jumpers, to select the clock source, the power supply and the amount of memory. Please do not touch JP5, because the amount of memory is a factory preset and the User cannot change it.

JUMPER	CONNECTION	DESCRIPTION
JP1	position 1-2	Use XTAL1 signal from target system.
	position 2-3	Use internally generated XTAL1 signal.
JP2	position 1-2	Use XTAL2 signal from target system.
	position 2-3	Use internally generated XTAL2 signal.
JP3	position 1-2	Use RESET signal from target system.
	position 2-3	Use internally generated RESET signal.
JP4	connected	Power supply for <b>MCS 051</b> is from the target system. Disconnect this jumper if supply from target may exceed 5V.
	not connected	Power supply for <b>MCS 051</b> is from its external power supply.

FIGURE 3: JUMPERS SUMMARIZING TABLE

**LEDS**

MPS 051 is provided with two LEDs to visualize the power supply presence and to indicate the internal status.

LED	COLOUR	DESCRIPTION
Power LED	Green	When on, indicates that the emulator/programmer is supplied and working.
Status LED	Red	When on, indicates that the emulator/programmer is programming the device in the socket. Leave the device in the socket until this LED is off.

FIGURE 4: LEDS TABLE

- XC1 - RS232 connector
- XC2 - external power supply connector
- XC3 - simulation socket connector
- S1 - precision socket for circuit programming
- S2 - precision socket for ZIF programming socket installation

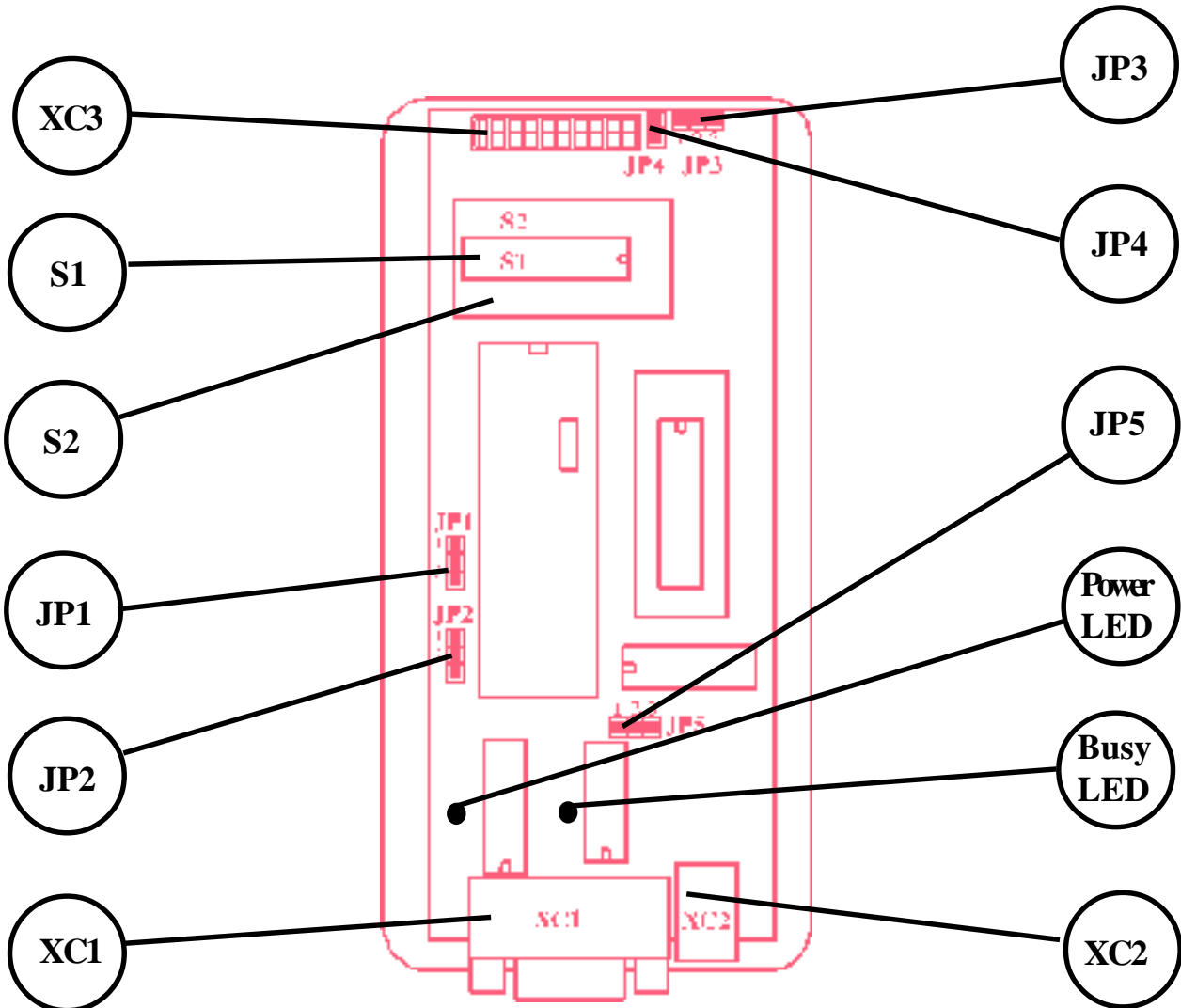


FIGURE 5: JUMPERS, CONNECTOR AND SOCKETS LOCATION



FIGURE 6: SUPPLY CONNECTOR POLARITY



## QUICKSTART

Here are the procedures that allow to get quickly ready to work. Please remark that this is just a summary of the descriptions given in the “INSTALLATION” chapter, which contains detailed instructions and information. Please consult also “TROUBLESHOOTING” chapter if You experience problems.

### SIMULATION

- Turn off PC, **MPS 051** and debugged system.
- Use the jumpers to select the internal oscillator or a debugged system oscillator. The same applies to RESET and power supply source. Please refer to figure 2 and figure 4.
- Insert the simulation socket into the debugged system; the required socket orientation is indicated on the sticker.
- Connect the **MPS 051** and the PC.
- Switch on the debugged system power supply, and possibly also the simulator power supply.
- Start the control program; you may possibly use the Simulator\Find command.
- The program debugging technique is exemplified in the “SOFTWARE DESCRIPTION” chapter.

### PROGRAMMING

- If the simulation socket pins are used as input pins, remove the simulation socket from the debugged system (neither the system nor the **MPS 051** may be supplied during this operation).
- Connect the provided external power supply to the coaxial connector of the **MPS 051** (the power supply must correspond to the values given in the Specifications).
- Start the control program, or use the Simulator\Find command.
- Insert the programmed circuit into the programming socket; its orientation is indicated by a notch in the programming socket.
- Load the program from disk using File\Load or pressing <F3> key or, to copy an unprotected device, read it with Programmer/Read menu or pressing <F7> key.
- Give the PROGRAM command. When the programming is over, the data can be verified (the verification is, however, possible only if the two LOCK bits are not programmed). The programming is indicated by the red LED (the programming LED is live). The control program automatically erases the circuit prior to programming.
- When the red LED (BUSY) is OFF remove the circuit from the programming socket.

## SOFTWARE DESCRIPTION

This chapter contains a complete description of the control software, its installation, features and utilization.

The software provided with the emulator/programmer is written for working with **SIM 2051** of **ELNEC**, but **grifo®** gives warrants a full compatibility and interchangeability between **MPS 051** and **SIM 2051**, so the very good and efficient program developed by **ELNEC** can be freely used also with our programmer.

The executable file of the control program is called **S2051.EXE** and is designed to work under **DOS**. It is possible to execute it in a **DOS** window under **WINDOWS 95/98**

## MINIMAL PC REQUIREMENTS

The **MPS 051** can be connected to **IBM PC** compatible. **MPS 051** software requires an **MS/PC-DOS** operating system, version 3.2 and above.

Minimum hardware requirements are as follows:

- PC XT
- min. 512 KB RAM
- 1 x FDD 3.5"
- 1 x standard serial port - min. 57600 Baud.

## DISK CONTENT

Each **MPS 051** simulator/programmer delivery encompasses a diskette with a control program and a **HELP** file. You are recommended to copy the diskette content into a selected hard disk catalogue. The diskette content may be freely reproduced. The delivery also includes a cross translator of the **SHAREWARE** type, which predetermines its operation.

- **S2051.EXE** - a control program for the **MPS 051** simulator/programmer
- **S2051E.HLP** - a **HELP** file
- **ASM51\** - a directory, including a shareware version of the **ASM51** translator
- **EXAMPLES\** - a directory, including examples for the simulator **EXAMPLE1.ASM**, **EXAMPLE2.ASM**

## STARTING THE CONTROL PROGRAM

Create a new folder on your hard drive, copy in it the content of the floppy disk, select it as current directory, then:

Type S2051<ENTER> at DOS prompt.

After starting the control program, check your checksum (CRC). If it is O.K., it will start searching for the **MPS 051** at individual physically existing serial ports of the computer. Each port is checked at the communication rate of 57600 Baud, this being indicated by listings on the display. The search operation lasts until the **MPS 051** is found, or if no **MPS 051** is connected, until the last serial port is found. If the **MPS 051** is found by the control program, a user display, including a control menu, will appear. Otherwise, the program will display an error message, and indicates possible reasons. When you remove a possible reason, press any key. The control program makes another attempt to find the **MPS 051**. If he fails to find it, the control program switches over to the DEMO mode which blocks the control commands in the Simulator **and the Programmer submenus**. **If you remove the error cause, you can give the FIND command to find the MPS 051 at some of the serial ports.**

When the **MPS 051** is found You can use it as a simulator or as a programmer.

## NOTE ON STARTING THE PROGRAM BY PARAMETERS

When started the program automatically searches for the connected simulator at all existing COMs. This procedure may be suppressed by entering a parameter with the COM address to which the simulator is connected. In this case, the control program finds the simulator directly at the specified COM. The parameter form is /Ayyy, where yyy is the address of the respective COM entered in the hexadecimal form. If this parameter is used several times, each time with a different COM address, the program obtains a list of COMs at which the simulator can be searched for. The list is also accepted by the SIMULATOR\FIND service during program run. In addition, the parameter is used when the program is started as a BATCH. An example of program starting by parameters:  
S2051 /A3F8 /A2F8.

## SIMULATOR

The contents of the external **MPS 051** processor RAM and the control program buffer are “synchronized”, which means that the control program tries to make these contents identical. Therefore, the external simulator RAM content is updated automatically prior to any command (CALL, GOTO, SINGLE STEP) by loading data from the buffer working space reserved for the Simulator part (1000H-17FFH, or 13FFH).

Furthermore, the Simulator part enables the user to scan and modify the P1 and P3 port values (P3 with the exception of P3.6 that is not used in the AT89C2051 processor). The representation is given both in the binary and the HEX formats.

The CALL and GOTO commands are used for program debugging. They start its individual subprograms, or the complete debugged program. When starting subprograms (CALL) you must know their initial address; when starting the complete program (GOTO) pass the control to the 1000h address (this address corresponds to the 0000h address in the standard circuit). The CALL command waits for return from the called subprogram, or for the completion of the debugged program. The GOTO command simply passes the control to the respective address without waiting for routine completion. If a debugged (sub)program gets into an endless loop the **MPS 051** can be reset by the respective pushbutton or by the RESET command whose meaning is the same. You can also return from the endless loop following the CALL command by pressing the ESC key. Depressing the resetting pushbutton, or using the RESET command do not affect the content of the external RAM in the **MPS 051**.

The SINGLE STEP BY INTO (INT1) command is used for debugging an application (sub)program by stepping individual instructions. Prior to calling one of these commands (depending on whether you use the P3.2-INT0 or the P3.3-INT1 pin for generating an interrupts) you are supposed to determine the pin which will be disconnected - such a pin will be used internally for generating an interrupt. The PortIN, PortOut, Registers and View/Edit int. RAM from the Simulator submenu can be used during the single step operation. Further details on the single step mode are given in the MCS-51 series processor catalogues, or in related literature.

The following part of this manual presents the description of using this mode in the **MPS 051**. Examples on the **MPS 051** diskette provide relevant information on program modifications.

The internal RAM of the processor can be accessed for the sake of both reading and writing by the View/Edit internal RAM command. Any attempt to write on the addresses reserved for the internal monitor in the **MPS 051** processor is blocked. This bears on the 20H and 5FH through 7FH addresses (stack).

The BAUD command sets up all necessary processor registers in the **MPS 051** to enable the serial port to be used at a selected communication rate. Output/input data can be accessed by the Registers command. The command makes it possible to scan and modify the SBUFF register and other processor registers.

The FIND command is used for searching for the **MPS 051** at all physically present PC's serialports. This command includes the simulator's RESET that can also be called independently. The effect of this command upon the simulator control processor is the same as that of depressing the resetting pushbutton. In addition, the RESET command is automatically executed after some operations if the processor is suspected to be in an undefined state, and if "rejects" to communicate (the control program RESET is not used if you use an external RESET; use the debugged application reset mechanism for this purpose).

To generate a time-based frequency of the simulator select an internal or an external oscillator by the OSCILLATOR command. The external oscillator blocks the access to the programmed circuit (unpermitted options BLANK CHECK, READ, VERIFY, and PROGRAM). An external oscillator is selected from the option of permitted types according to the frequency, the tolerance being  $\pm 2.5\%$ . The tolerance delimits the range of smooth PC-simulator communication at the transmission rate derived from the applied oscillator value. Details can be obtained by using the contextually sensitive key <F1> in the control program.

The program is preset to the application of an internal oscillator. Provided that your simulator hardware is preset to an external oscillator, a communication error occurs (if its value differs from 11.0592MHz), and the program will run in the DEMO mode. Use an oscillator command, and select the right value of the external oscillator used. Then, the program finds the **MPS 051**.

#### NOTE ON SIMULATION MODE

The control program can also be started as the BATCH version, with the file name parameter. In this case, the program loads the file content (in the binary or the Intel HEX format) into an external RAM in the simulator, starts it, and returns the control to the operation system. The operation corresponds to the GOTO command for the 1000H address. The parameter must contain a complete file name. The format is distinguished by the "extension" in the file name; if the extension is .HEX, it is considered to be a file in the Intel HEX format; in all other cases the file is binary. It is supposed that you use the AT89C2051 processor and an internal oscillator.

#### ON LINE HELP

The description of individual control program commands can be displayed by selecting the HELP (<F1> key). Press <F1> twice to obtain all Help options, including the Help on HW **MPS 051**.

## SINGLE STEP OPERATION MODE

The single step mode in the **MPS 051** is designed for debugging a user's program by individual instructions. This mode is enabled by HW properties of the MCS-51 series processors, notably the processor's response to a interrupt level from the INT0 or INT1 input. While respective catalogues mention an external signal at these inputs, interrupts can be generated internally, too. Certainly, the control program in the **MPS 051** processor has been completed for this purpose. This debugging technique is, however, limited by the fact that the pin used for generating an interrupt cannot be used (P3.2 or P3.3) - the pin must remain disconnected. At best, it may be connected as an output pin.

Three states are possible in the single step operation: STEP, RUN and STEP OVER. The STEP state is used for moving along individual instructions in the control program. The RUN state is similar to the standard program run with the distinction that the debugged program may contain so-called BREAK-POINTS (i.e., insertions in a standard code) at which the debugged program stops and passes to the STEP mode. The STEP OVER state processes ACALL or LCALL instructions as a single step, i.e. without any insertion.

Procedure:

Let us suppose that you have a translated program (or its part) that should be debugged in this mode. Load the code to the **MPS 051** control program buffer. Depending on which of the pins, i.e., P3.2 (INT\0) or P3.3 (INT\1), you decide to "sacrifice" for the single step mode, select the respective command from the control program menu:

SIMULATOR/ SINGLE STEP BY INT0, or  
SIMULATOR/ SINGLE STEP BY INT1

It should be noted that the pin will not be accessed in the user's (debugged) program. The user is expected to carefully observe all above given limitations and recommendations concerning the installation of BREAK-POINTS into the debugged program, this being a precondition of correct RUN command execution.

The working window of the single step mode consists of several parts. The upper part displays current values of the processor registers and the P1 and P3 ports. These values are continuously updated by reading from the simulator. Any changes in the register or the port (by stepping an instruction or by "external" intervention) are indicated by different colours. The bottom line contains a list of commands used for this mode. The current line (with the cursor at its end) always contains a translated instruction to be executed in the next step (by operating the ENTER or the <F7> key). In addition, this mode makes it possible to scan and modify the internal RAM content and the content of some of the registers or ports. Depress the <F1> key to obtain Help for further details.

When you use the RUN command (<Alt+U>) the debugged program may "lose its way". Usually, it is necessary to have recourse to the HW reset, or to use the <ESC> key to return the control program to the ROOT menu. The same applies to stepping by individual instructions in cases when it is clear that the program cannot do what it is expected to. Anyway, the return to the ROOT menu is preceded by automatic simulator identification to preserve correct setting of communication with **MPS 051**. The program checks the stack for its current state. If the stack is overflowed, the single step mode is terminated, an alarm message appears on the display, and the system waits for the HW reset.

The application of the STEP OVER mode (<F8> key) has some restrictions. It is based on the idea of forced insertion of a breakpoint after the RET instruction in the particular subprogram (ACALL or LCALL). This is enabled by a part of the external RAM in the simulator within the address range of 17F6H-17FFH. The control program transfers the respective CALL instruction code to this area, and adds a break-point instruction in a specified syntax, including a jump back to the next instruction. The control is redirected to this area by rewriting the ACALL (LCALL) instruction code in the external RAM to AJMP (LJMP). When these steps are over, the external RAM in the place of the ACALL (LCALL) instruction resumes its original state so that it may be called, for instance, in a loop in this mode. The STEP OVER mode can be only used if the debugged program does not exceed the 17F5H address. Otherwise, the selection of <F8> results in rewriting the user's program at the above mentioned addresses. No check is done for this purpose on the part of the program.

### NOTE ON USING THE STEP OVER

A case may happen that the ACALL (LCALL) routine contains a user's break-point. The program accepts it and stops. If you use the RUN (<Alt+U>) command at this moment you can safely return from the routine by accepting the forced break-point in the reserved area of the external RAM. The RUN command may be replaced by simple single step operation (<F7>). It should be, however, noted that after leaving the ACALL (LCALL) routine the program is in a reserved area, and the forced break-point instructions will follow. It is only after their execution that the control returns to the instruction following the ACALL (LCALL).

### BREAK-POINT INSTALLATION INTO THE DEBUGGED PROGRAM

BREAK-POINT sequence for the INT0\ interrupt:

```
clr    P3.2
setb   EX0
nop
```

BREAK-POINT sequence for the INT1\ interrupt:

```
clr    P3.3
setb   EX1
nop
```

### DEBUGGED PROGRAM LIMITATIONS IN THE SINGLE STEP MODE

- The P3.2 (for INT0\ ) or the P3.3 (for INT1\ ) pin cannot be used in the single-step mode (the pin not to be used is determined by an external interrupt selected).
- You cannot disable all interrupts simultaneously (by means of the CLR EA instruction).
- You cannot disable the external interrupt used to generate a debugging interrupt (i.e., the clr EX0 instruction may not be used in the debugged program if the INT0\ is used to generate a debugging interrupt; if the debugging interrupt is generated by the INT1\ , you are not allowed to use the clr EX1 instruction).
- You are not allowed to change the priority of interrupts, or more precisely, the interrupt used for debugging must maintain the highest priority.

- You may not change the interrupt type used for debugging (the interrupt type is set in the TCON register).
- 22B of the stack are available for debugging in the single step mode.
- If you define your own stack the return address to the monitor gets lost after completing the SINGLE STEP mode. In the STEP state, the program indicates this fact, and RESETs the simulator automatically. In the RUN state, it usually remains in an endless loop.
- The debugging interrupt utility routine changes the following addresses in the internal RAM of the processor: 20h, 59h, 5Ah, 5Bh, 5Ch, 5Dh, 5Eh.
- When entering BREAK-POINTS, the debugged program code may not exceed the program memory capacity for the simulated microprocessor type, including the limitation concerning the use of the STEP OVER state.

#### NOTES:

- It is advantageous to substitute a macro for the instructions of the SW BREAK POINT.
- To avoid various source code modifications for individual debugging techniques you are recommended to use a conditional translation, for example, with **MPS 051** variables to change the initial program address, and SSTEP to separate SW BREAK-POINTS.
- Examples demonstrating the capabilities of the single step mode can be found in the EXAMPLES catalogue on the diskette.

#### PROGRAMMER

The Programmer part contains all commands required for work with the FLASH EPROM of the AT89C1051 or the AT89C2051 microcontroller.

- The SELECT command is used to select a circuit type. The AT89C2051 type is preset.
- The BLANK CHECK command checks whether the programming socket circuit is erased.
- The READ command reads the circuit in programming socket, and saves the data in the control program buffer within the address range of 0H-7FFH (3FFH), i.e., in the working area of the buffer reserved for the Programmer part.
- The VERIFY command checks the circuit for correct programming.
- The PROGRAM command programs the data from the control program buffer to the programming socket microcontroller (within the address range of 0H-7FFH, or 3FFH). In addition, this command allows for programming LOCK bits in the programmed circuit. The S2051 control program automatically erases the circuit prior to programming. The red LED on the simulator indicates the programming socket activity.

#### !!! IMPORTANT !!!

- Individual P1 and P3 port terminals are common to the simulation socket and the programming socket. Therefore, signals from the simulated device are not allowed to affect the operation of the programmed circuit. The best solution is to disconnect the simulation cable on the **MPS 051** side.
- The red LED (BUSY) in the **MPS 051** indicates that the programmer socket is live. It is recommended to leave the programmed circuit in the socket when the LED is lit up.
- It is recommended to insert or remove circuits into/from the programming socket **ONLY** when the **MPS 051** is ON - the BUSY LED is not lit up.



The description of individual control program commands can be displayed by means of Help (<F1> key). When you depress the <F1> key twice the control program provides all Help options, including the Help for HW **MPS 051**.

## **RECOMMENDED PROCEDURE**

### Simulation:

- Use the switches to select the internal oscillator or a debugged system oscillator. The same applies to RESET.
- Insert the simulation socket into the debugged system; the required socket orientation is indicated on the sticker.
- Connect the **MPS 051** and the PC.
- Switch on the debugged system power supply, and possibly also the simulator power supply.
- Start the control program; you may possibly use the Simulator\Find command.
- The program debugging technique is exemplified at the end of this manual.

### Programming:

- If the simulation socket pins are used as input pins, remove the simulation socket from the debugged system (neither the system nor the **MPS 051** may be supplied during this operation).
- Connect an external power supply to the coaxial connector of the **MPS 051** (the power supply must correspond to the values given in the Specifications).
- Start the control program, or use the Simulator\Find command.
- Insert the programmed circuit into the programming socket; its orientation is indicated by a notch in the programming socket.
- Give the PROGRAM command. When the programming is over, the data can be verified (the verification is, however, possible only if the two LOCK bits are not programmed). The programming is indicated by the red LED (the programming LED is live). The control program automatically erases the circuit prior to programming.
- When the red LED (BUSY) is OFF remove the circuit from the programming socket.

## OPERATION EXAMPLE

The EXAMPLES\ folder on the diskette contains two examples. We shall use one of them, EXAMPLE1.ASM, to demonstrate **MPS 051** operation.

The debugged program reads the serial port data, rewrites it to the P1 port, and waits 1 ms. Then, the port is rewritten by a negated data which is sent to the serial port. The overflowing of the TIMER/COUNTER T0 results in an interrupt, and the utility routine of the interrupt sends the 01H byte to the serial port.

The RAM memory structure in the **MPS 051** is identical with the structure of the AT89C2051/1051 processor program memory, however, it starts with the 1000H address. The difference must be eliminated as follows:

```

ADR    EQU    1000h    ;line #1A for debugging
;ADR   EQU    0000h    ;line #1B for programming into the processor
BOC    EQU    002Bh    ;BEGIN_OF_CODE - program start address
                        ;of the AT89C2051/1051 processor (this
                        ;adress must be higher than or equal to the
                        ;002Bh address)
                        ; this part includes the definitions of variables and constants

                        ;the program code beginning, all ORG directives to be written in the
                        ;ADR+... form

org    ADR+BOC org    ADR+0000h    ; program start address following reset
ljmp   START        ; jump to program start
org    ADR+0003h    ; beginning of the interrupt vector from INTO
org    ADR+000Bh    ; beginning of the interrupt vector from T0
org    ADR+0013h    ; beginning of the interrupt vector from INT1
org    ADR+001Bh    ; beginning of the interrupt vector from T1
org    ADR+0023h    ; beginning of the interrupt vector from the serial port
START: ; the first program instruction
      END    ; end of program
    
```

The translation of the program to be debugged in the **MPS 051** contains a note in the form of the #1B line, which means that the program code will be stored in the **MPS 051** RAM starting with the 1000h address as required by the simulator. A program translated in this way would not work in the AT89C2051/1051 whose addresses range from 0000...07FF. Therefore, the note must include a line, labelled as #1A. In addition, it is necessary to make the #1B line valid, and program the translated code into the processor.

## PROGRAM DEBUGGING

- Translate the source file (with ORG modifications for the simulator)
- Use the VIEW/EDIT BUFFER command to review and, if necessary, edit the data in the working area reserved for the simulator
- If everything is O.K. the program may be started by the GOTO or the CALL command to the required address within the range of 1000H...17FFH. The EXAMPLE1 program expects a character from the serial port. Thereafter, the sequence of events described in the beginning of this chapter will be executed. When the program run is over, a negated value remains at the P1 port. You can check it by giving the INPUT command with the P1 parameter.

## HARDWARE LIMITATIONS ON THE AT89C2051 CIRCUIT

- The simulator simulates the AT89C2051-12PC processor.
- The simulator cannot operate in the same supply voltage range as the AT89C2051 or 1051 microcontrollers. When supplied from the internal power source, the debugged application can also operate with the minimum supply voltage for the AT89C2051/1051 (2.7V) provided that no problems arise from the fact that pull-up resistors pull up log 1 to the level of 5V in the simulation process. Certainly, voltage of at least 1.9V must be available to the log 1 input level. Supply voltage of the simulator does not affect that of the application - they are separated by a diode. If the JP4 jumper is disconnected, the VCC value in the application can equal the maximum voltage permissible for the AT89C2051, i.e., 6V.
- Maximum current to L for the SIM2051 is 1.6 mA/0.45V only.
- As opposed to the AT89C2051, the simulator is not provided with any analog comparator. To ensure correct communication between the MPS 051 and the PC through a serial port, the external crystal value must correspond with that given in the SIMULATOR\OSCILLATOR\EXTERNAL menu.

To make up for the lack of the analog comparator, it is possible to use another hardware emulator for family MCS 51 devices, T-EMU52 and an adaptor with the following features:

- Adaptor for T-EMU52 and others MCS51 emulators
- 87C51/AT89C2051 converter
- Converts pins of AT89C2051/4051 to 87C51
- Includes analog comparator
- Availability of disconnecting P1.0/20-P1.0/40 and P1.1/20-P1.1/40

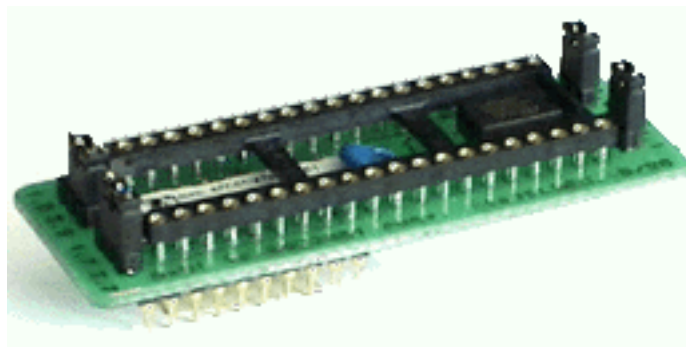


FIGURE 7: ADAPTOR POD FOR T-EMU 52 PHOTO

## SOFTWARE LIMITATIONS

- The monitor stack starts at the 5Fh address of the external RAM of the simulator processor. For the process of debugging by means of the CALL function it is available from the address 63H on, i.e., its length is 28 bytes. In the case of the GOTO function, it is the 61H address, i.e. 30 bytes. If you define a stack of your own, the monitor return address gets lost. It is recommended to return by means of the LJMP 0000 instruction at the end of your routine. This instruction will return the monitor to its initial state.
- The RAM area, which can be used for storing a debugged program, starts at the 1000h address.
- The 20H address byte of the internal RAM may not be changed.
- Since the buffer comprises two working areas, and since the binary reading of the file from the disk does not enable the user to distinguish between the simulator and the programmer data, the binary approach represents a compromise. When the data is successfully loaded, the programmer's working area is automatically copied to the simulator's working area (the area equals the simulated circuit size). This operation takes place irrespective of whether the user changed the preset values in the ADDRESS frame of the LOAD FILE working window.
- The following table gives delays in execution of interrupts due to modified calls of interrupt routines in the MPS 051 monitor for individual interrupt sources. The time values refer to  $f_{OSC} = 11.0592 \text{ MHz}$ .

INTERRUPT	CYCLES	CLOCK CYCLES	TIME
/INT0	8	96	8.68 ms
T0	2	24	2.71 ms
/INT1	8	96	8.68 ms
T1	2	24	2.71 ms
RI+TI	2	24	2.71 ms

**FIGURE 8: DELAYS IN EXECUTION OF INTERRUPTS TABLE**

## CONTROL PROGRAM COMMANDS

The following is a list of all the commands available with control program. The command name is matched with its keyboard shortcut (if present) and a brief description of its purpose and employ modalities. Please remember that the most recently updated instructions list for the control program is always the on line help, which should be consulted before this manual.

### SIMULATOR MENU

<i>Name</i>	<i>Shortcut</i>	<i>Function</i>
Input	<Alt+N>	This command reads P1 and P3 ports permanently and displays its values in hexadecimal and binary form. P3.6 pin is always set to logic H.
Output	<Alt+P>	Use this command for a new value writing to the P1 or P3 port of simulator.
Call	<Alt+C>	<p>The command starts subroutine in external RAM of simulator, which begins on specified address. The command waits , while a called subroutine was beending by a RET command. If run of called subroutine is beending in "endless loop", use a HW RESET of simulator or key &lt;ESC&gt; for a correct return to the Monitor. HW RESET don't changes data in external RAM. Specified address must be in range 1000H - 17FFH for AT89C2051 device respectively 1000H - 13FFH for AT89C1051device. Before every execution of this command are data from buffer copied to an external RAM,sake synchronize data in buffer with the external RAM. Selected address must be from interval 1000H-13FFh respectively 1000H-17FFh by type of selected microcontroller.</p> <p><b>Note:</b> User have free 28B from stack. If ourselves define new stack, return address will be lost. For successfull return we can reference to use instruction LJMP 0000 on the end of routine, which restore monitor into original state.</p>
Goto	<Alt+G>	<p>This command starts a program in external RAM of simulator, which begins on specified address. Command doesn't wait for beending of routine. Communication with simulator isn't possible, while program runs. If program beends in "endless loop" , use a HW RESET of simulator and SIMULATOR/FIND command for communication restoration, or use command SIMULATOR/RESET. RESET don't change data in external RAM. Specified address must be in range 1000H - 17FFH for AT89C2051 device respectively 1000H - 13FFH for AT89C1051 device.</p> <p><b>Note:</b> User have free 30B from stack. If ourselves define new stack, return address will be lost. For successfull return we can reference to use instruction LJMP 0000 on the end of routine, whitch restore monitor into original condition.</p>

<i>Name</i>	<i>Shortcut</i>	<i>Function</i>
Single Step by int 0	<Alt+0>	This command is used for debugging application (sub)program and allows run this program statement by statement. Before every execution of this command is necessary to remember, that for generates interrupt will be used pin P3.2-/INT0 hence in application program will not be apply - it will apply for internal generating of interrupt. In step mode is possible to change actual contents of ports P1 and P3 (bits P3.2 and P3.6 are masked), view and edit internal RAM and some of registers. More information about single step mode are in data book for microcontrollers MCS-51. In examples on distribution floppy disk are informations about adjustments of program for single step mode. Restrictions and recommendations for single step are in STEP MODE.
Single Step by int 1	<Alt+1>	This command is used for debugging application (sub)program and allows run this program statement by statement. Before every execution of this command is necessary to remember, that for generates interrupt will be used pin P3.3-/INT1 hence in application program will not be apply - it will apply for internal generating interrupt. In step mode is possible to change actual contents of ports P1 and P3 (bits P3.3 and P3.6 are masked), view and edit internal RAM and some of registers. More information about single step mode are in data book for microcontrollers MCS-51. In examples on distribution floppy disk are informations about adjustments of program for single step mode. Restrictions and recommendations for single step are in STEP MODE.
View Edit internal RAM	<Alt+I>	This command is used to view (view mode) or edit (edit mode) data in internal RAM of simulator (for viewing in DUMP mode only) in range 00H - 7FH. Use arrow keys for select the object for edit. Edited data are signified by color. There isn't display special function registers (SFR) by this command. The change of contents of some address is dangerous, because the program blocking entry into this address (20H, from 5FH to 7FH). User is for this fact noticed by warning.
Registers	<Alt+E>	This command allows view and edit some of registers. Registers, which maybe edited, are displayed with different color attribute and possibility to change value in hexadecimal. Value can be edited in binary by using <Alt+B> key too. In some registers with possibility to change value are any bits masked. More informations are in Edit contents of registers.
Find	<Alt+F>	The command finds simulator on all present COMs of PC. It is used for switch control program to DEMO / READY mode too.

<i>Name</i>	<i>Shortcut</i>	<i>Function</i>
Reset	< <b>Alt+R</b> >	Command generate reset signal for the <b>MPS 051</b> . Command is executed automatically after finished of commands SIMULATOR\CALL and SIMULATOR\SINGLE STEP. <b>Note:</b> Command RESET is not available, if reset signal is generated from developed equipment.
Oscillator	< <b>Alt+O</b> >	This command correspondes with switchers JP1 and JP2, which are switched by option internal or external oscillator. Select one of items, which are available after < <b>Enter</b> > pressed. Identification of simulator at new communication speed (depending on oscillator frequency) will be executed immediate after choice new type and frequency of oscillator.
Automatic reload		This command enables or disables automatic reload of changed debugged file before executing any simulation command (Simulator\Call, Simulator\Goto, Simulator\Single step by INT0, Simulator\Single step by INT1). This command also enables automatic loading of debugged file at start program after processing S2051.SET file. File name and enabling of automatic load are stored to file S2051.SET by command Quit\Yes & Save.

## PROGRAMMER MENU

Select	< <b>F5</b> >	Use this command for select AT89C1051 or AT89C2051 or AT89C4051 device type before access to device on programming socket. Device AT89C4051 can be selected only if the BIOS version is 3.00 or above.
Blank Chk	< <b>F6</b> >	This command allows to check the all device, its erasing. The program reports the end without error or with the error by sound signal and it writes the warning report on the display. Before select this command be sure, that simulation socket is disconnect from developed equipment.
Read	< <b>F7</b> >	This command allows to read programmed data from the device in the programming socket into the buffer from address 0H (if doesn't set LOCK bits 1 and 2). The end of the reading is indicated by the sound signal. Before select this command be sure, that simulation socket is disconnect from developed equipment.

<i>Name</i>	<i>Shortcut</i>	<i>Function</i>
Verify	<F8>	This command checks the programmed data (if doesn't set LOCK bits 1 and 2). It compares content of the device with data in buffer from address 0H. The program reports the end of verifying without error or with the error by sound signal and it writes the warning report and first error on the display. Before select this command be sure, that simulation socket is disconnect from developed equipment.
Program	<F9>	This command programs the device in the socket of programmer with data from buffer in addresses range 0-3FFH respectively 0-7FFH by type of selected device. You can program all FLASH EPROM with one or both LOCK bits with the following meaning:

LB1	LB2	
0	0	FLASH EPROM not protected
1	0	FLASH EPROM protected against programming
1	1	FLASH EPROM protected against programming and reading

Before select this command be sure, that simulation socket is disconnect from developed equipment. The first the control program checks identification code of device and then erases all device. Programmend data are verifying after the programming.

## FILE MENU

Load	<F3>	Analysis file format and loads the data from specified file to the buffer. You can choose the format desired (Binary, MOTOROLA, MOS Technology, ASCII space, Tektronix and Intel (Extended) HEX). The control program stores a last valid mask for file listing. You can save the mask into the config. file by command Options / Save options.
Save	<F2>	Saves data in the buffer which has been created, modified, or read from a device into a specified file. You can choose the format desired (Binary, MOTOROLA, MOS Technology, Tektronix and Intel (Extended) HEX).

## BUFFER MENU

View code	<Ctrl+F4>	This command is used to view data from buffer in MCS51 instruction format. Use arrow keys for displaying data in range of address 1000H-13FFH (or 17FFH by selected emulated microcontroller). Default data displaying immediate after the choose this command is from address 0000H.
-----------	-----------	---



<i>Name</i>	<i>Shortcut</i>	<i>Function</i>
View/Edit	<F4>	This command is used to view (view mode) or edit (edit mode) data in buffer (for viewing in DUMP mode only). Use arrow keys to select the object to edit. Edited data are signified by color.
Fill block		Selecting this command fills the selected block of buffer with requested hex (or ASCII) string. Set start and end block for filling and requested hex or ASCII string.
Copy block		This command is used to copy specified block of data in current buffer on new address. Target address needn't to be out from source block addresses. See also Move block.
Move block		This command is used to move specified block of data in current buffer to new address. Target address needn't to be out from source block addresses. Source address block (or part) will be filled by topical blank character. See also Copy block.
Swap block		This command swaps a high- and low- order of byte pairs in current buffer block. This block must started on even address and must have an even number of bytes. If this conditions do not fulfil, the program modifies addresses itself (start address is moved on lower even address and/or end address is moved on higher odd address).
Erase buffer	<Ctrl+F2>	If this command is selected, the content of the buffer will be filled with topical blank character.
Checksum		<p>The checksum is calculated after the entering hexadecimal buffer addresses by next way :</p> <p>           BYTE - sum by bytes to "word". CY flag is ignored.            WORD - sum by words to "word". CY flag is ignored.            BYTE (CY) - sum by bytes to "word". CY flag is added to result.            WORD (CY) - sum by words to "word". CY flag is added to result.         </p> <p>Column marked as NEG. is a negation of checksum so that SUM + NEG. = FFFFH.</p> <p>Column marked as SUPPL. is complement of checksum so that SUM + SUPPL. = 0 (+ carry).</p>

*Name*      *Shortcut*      *Function*

## OPTIONS MENU

Buffer name	<p>This option is used to enter full pathname of buffer in case it should be created on disk (buffer on disk is created automatically if the size of selected device is larger than free memory space). It performs no error-checking. If You enter incorrect buffer name, the problem will be raised when creating the file on disk.</p> <p>This setting is saved to file S051.CFG by command Options\Save options. Default name is S051.\$\$0.</p>
Monitor	<p>This command sets the colors for used type of VGA display (monochrome or color). The colors change is visible immediately. Don't use the command for HERCULES card. This card is identified automatically.</p> <p>This setting is saved to file S2051.CFG by command Options\Save options. Default set is color.</p>
Sound	<p>This command sets intensity of sound signals, that accompany the displaying of any text informations. This setting is saved to file PG4U.CFG by command Options\Save options.</p> <p>It's possible to select one of the following options :</p> <ul style="list-style-type: none"><li>Options\Sound\Long (default)</li><li>Options\Sound\Short</li><li>Options\Sound\None</li></ul>
All HEX loadings	<p>This command sets several options for loading control by any of HEX formats. The first option sets erasing buffer automatically before loading by any of HEX formats. The second option sets a negative offset, which is used for data addresses modification by loading from any HEX file so, that data can be written to existing buffer addresses.</p> <p>For example : A file contains data by Motorola S - format. A data block started at address FFFF0H. It is in S2 format with 3 bytes of address array length. For each read data You can set a value of negative offset to FFFF0H. It means, that the offset will be subtracted from current real addresses and so data will be written from buffer address 0.</p> <p><b>Warning:</b> a negative offset is subtracted from real address and therefore a result of subtraction can be a negative number. Please take care of correct setting of this value. This setting is saved to file PG4U.CFG by command Options \ Save options. Default setting means inactive status.</p>

<i>Name</i>	<i>Shortcut</i>	<i>Function</i>
Intel HEX loading		This command is used for data redirecting from file by extended Intel HEX format to physically existing buffer addresses. It means, that the User will be prompted to enter segment, which is associated with the lowest segment in file (record type 02) and the other segments will be modified with this difference. For example: The file contains two records of type 02 with addresses F000H and F800H. When entering a new segment for example 0H, data from segment F000H will be redirected to segment 0H and similar data from segment F800H to segment 800H. This option is not valid for "simple" Intel HEX format. This setting is saved to file PG4U.CFG by command Options \ Save options. Default setting means inactive status.
Help		Use this menu command to install Online context - sensitive help system in the desired language. You can select english, german and slovak (coding of Kamenický or Latin 2 - page 852, and without punctuation) HELP system. You may install the HELP system any time during the run of control program, and save the current settings to .CFG file on disk. Successful installation of a new HELP system supposes .HLP files existence on your disk, which are delivered on distributions file along with the control program.
Set Masks		Use this command to set file-name masks to use as a filter for file listing in save and load file window for all file formats. Mask must contain one of wildcards (*, ?) at least and must be applied correctly by syntax. This setting is saved to CFG file by command Options \ Save.
Save options		This command saves settings You have made in all the settings under the Options menu. All options are stored in configuration file on disk with filename S2051.CFG. You can select, where the configuration file will be created in current directory, root directory on drive C: or directory where S051.EXE is run from. When you start the S051.EXE, it looks in the current directory for the saved configuration file (highest priority). If it does not find the file there, it looks for it in the root directory on drive C, then in the directory where S051.EXE is run from. If the file does not exist, the program sets default options.
Retrieve options		This command allows You retrieve the settings you've saved to a S051.CFG configuration file with the Options / Save options command. You can select where the configuration file will be loaded from (current directory, root directory on drive C: or directory where S051.EXE is run from).

*Name*      *Shortcut*      *Function*

## QUIT MENU

No	This command cancels the request to quit.
Yes	The command deallocates heap, cancels buffer on disk (if exists) and returns back to the operation system.
Yes & Save	The command deallocates heap, cancels buffer on the disk (if exists), saves current setting of last 10 selected devices to file PG4U.SET on the disk in current directory and returns back to the operation system.

## ABOUT MENU

When You choose the Info command from the menu, a window appears, showing copyright and version information.

## HELP SYSTEM

Pressing the <F1> key accesses the context-sensitive Help. If S051.EXE is executing an operation with the programmer <F1> generates no response. When the Help window is open pressing <F1> again causes the program to display the list of all available Help messages. You can select and display any of these. You may see so-called key words (in bold) and some of these may provide more information if selected (cross-references).

The following HELP items are highlighted:

- Words describing the keys referred to by the current Help
- All other significant words
- Current cross-references; press <ENTER> to obtain further information.
- Inactive cross-references; use the cursor keys to select one of them and acknowledge with <ENTER>

Since the HELP system is continuously updated together with the control program, it may contain information not included in this manual.

Detailed information on individual menu commands can be found in the integrated on-line Help.

## WARRANTY TERMS

The manufacturer gives a guarantee on failure-free operation of the device for the period of 12 months from the date of delivery as given in the delivery note.

If a simulator failure results from a defect in workmanship, we shall repair the device as if under guarantee.

The device must be delivered for repair in the original packing, including all accessories, a copy of purchase document, and the description of failure in writing. In reasonable cases, please, supply the description of circumstances accompanying the failure:

- Control program version
- Environment in view of possible interference (office, laboratory, industrial, and other environment types)
- Description of your PC hardware: name, type, PC rate, printer port type (integrated on a HGC card, a multi I/O card, a disk controller card,...), I/O rate
- Description of your PC software: type and version of your operating system, OS system (Shell, Commander, Windows,...), resident programs used.

If your claim is sent without failure description enclosed we shall have to render an account for the time needed for ascertaining these facts. Moreover, we do not guarantee to adjust your claim appropriately if the programmer is not delivered with all the accessories when the failure is outside the device itself. Unjustified claims will be charged.

The guarantee does not apply to common wear, failures due to unqualified handling, and/or through mechanical damaging. It does not cover damage caused by improper modification or repair, shipping, or high voltage surges from external sources such as power line or connected equipment.

Customer service within and after the guarantee period is provided by the manufacturer:

## TROUBLESHOOTING

We really want you to enjoy our product. Nevertheless, problems can occur. In such cases please read carefully all the enclosed documentation again. Probably you will find the needed answer right away. Should the problem persist, please follow the instructions below.

### COMMUNICATION ERRORS

- Programmer must be supplied correctly, so the green supply LED must be ON and the power supply must be the one delivered with the programmer. To test the supply cable's physical integrity twist it delicately and see if something changes.
- Update the control program, often a more recent version includes enhancements also in communication reliability. Please refer to "FREE SOFTWARE UPDATES" paragraph for more information.
- The serial cable must be the one delivered with the programmer or a cable **with all the pins connected**. To test the cable's physical integrity twist it delicately when S051.EXE tries to connect to the emulator/programmer and see if something changes.
- Try to install emulator/programmer and S051.EXE on another computer. If Your system works normally on the other computer you might have a problem with the first one PC. Compare differences between both computers.

### READING OR PROGRAMMING PROBLEMS

- Assure the correct alignment of target device in the socket. Check the serigraph on the emulator/programmer or see the adapter serigraph if using one.
- Update the control program.
- If the target device has never been programmed before or has been deleted it has returned to blank state. Perform "Blank check test" through Programmer/Blanck Chck menu or the keyboard shortcut <F6>.
- Some devices may protect the content of their FLASH EPROM through protection fuses. This makes absolutely impossible to read the device content, but it is not a malfunction.

## ADDITIONAL TOOLS

**grifo®** programmers can be matched with several additional tools that expand their range of applications and make a complete set suitable for professional use whenever it is needed to erase, program or simulate every most diffused kind of EPROM and the most famous microcontrollers.

### EP 32

Low cost Eprom/universal Programmer 32 pin devices

EP-32 is a small and powerful EPROM, EEPROM, Flash EPROM and serial EEPROM programmer and static RAM tester, designed for professional mobile applications. In addition, EP-32 programmer with auxiliary modules support also microprocessors (MCS48, MCS51, PIC, AVR), GALs, etc.

### MP AVR-51

Micro Programmer for families AVR and 51

MP AVR-51 is little and powerful portable programmer for MCS51 series and Atmel AVR microcontrollers. MP AVR-51 enables also programming serial EEPROM with interface types IIC (24Cxx), Microwire (93Cxx) and SPI (25Cxx). The programmer is equipped by ZIF DIP 40pin socket. The quality of programmer is completed by comfortable control program.

### MP PIK

Micro Programmer for Microchip PIC

MP-PIK is little and powerful portable programmer for Microchip PIC series microcontrollers. MP-PIK enables also programming serial EEPROM with interface types IIC (24Cxx), Microwire (93Cxx) and SPI (25Cxx). The programmer is equipped by ZIF DIP 40pin socket. The quality of programmer is completed by comfortable control program.

### UEP 48

Universal Eprom Programmer 48 pin devices

UEP-48 is universal programmer that supports all kinds of types and silicon technologies of programmable devices. Powerful pin-driver provides logic level, pull-up/pull-down, clock, ground, one VCC supply and two programming supply and, certainly read, on each of all 48 pins independently. This advanced design give it the ability to program almost every programmable device in DIL up to 48 pins without adapter or family-specific module.

### SEEP

Serial EEproms Programmer

SEEP is universal programmer of all types serial EEPROM in 8-pin package. SEEP enables programming EEPROM with interface types IIC (24Cxx), Microwire (93Cxx) and SPI (25Cxx). Programmer supports programming LV EEPROM (3.3V). The programmer is equipped by ZIF socket. The quality of programmer is completed by comfortable control program. SEEP is computer peripheral, it is connecting to PC via standard parallel port.

### ER 05

Eprom Eraser 05

Erases all UV erasable EPROMs; holds up to 5 EPROMs; programmable timer with three different exposure durations; saves UV lamp lifetime; provided with power supply.

## ALPHABETICAL INDEX

**A**

ACALL 53  
AT89C2051 39, 40

**B**

BREAK POINT 53  
BUFFER 38

**C**

CALL 50, 57, 58  
COMMUNICATION ERRORS 68  
COMMUNICATION RATE 40  
CONNECTABILITY 40  
CONNECTION CABLE 42  
CONTROL PROGRAM COMMANDS 59  
CONVENTIONS 38  
CURRENT CONSUMPTION 40

**D**

DEBUGGED PROGRAM LIMITATIONS 53  
DEBUGGING 57  
DELIVERY CONTENT 42  
DIMENSIONS 40  
DISK CONTENT 48

**E**

ELECTRIC FEATURES 40  
ELNEC 48  
ESD 42  
EXAMPLE 56

**G**

GENERAL FEATURES 39, 40  
GOTO 50, 57, 58

**H**

HARDWARE LIMITATIONS 57  
HELP 51, 66  
HEX 38



**I**

INSTALLATION 42

**J**

JUMPERS 44, 45

**L**

LCALL 53

LEDS 45

LJMP 39, 58

**M**

MASS 40

MICROCONTROLLERS PROGRAMMED 40

MICROCONTROLLERS SIMULATED 40

**N**

NOTE ON SIMULATION MODE 51

NOTE ON USING THE STEP OVER 53

**O**

OSCILLATOR 51

OSCILLATOR FREQUENCY 40

**P**

PC REQUIREMENTS 48

PHYSICAL FEATURES 40

POWER SUPPLY 40, 44

PRELIMINARY INFORMATION 38

PROCESSOR REGISTERS 50

PROGRAMMER 39, 54

**Q**

QUICKSTART 47

**R**

READING OR PROGRAMMING PROBLEMS 68

RESET 44, 50

RS 232 38

**S**

S2051.EXE 48  
SBUFF 50  
SIM 2051 48  
SIMULATION SOCKET 44  
SIMULATOR 39, 50  
SINGLE STEP 52  
SOFTWARE DESCRIPTION 48  
SOFTWARE LIMITATIONS 58  
STARTING THE CONTROL PROGRAM 49  
STEP OVER 52

**T**

TECHNICAL FEATURES 40  
TEMPERATURE RANGE 40  
TERMINOLOGY 38  
TROUBLESHOOTING 68

**W**

WARRANTY TERMS 67

**X**

XTAL 44

**Z**

ZIF 38