BASIC 52

8052 MICROCONTROLLER BASIC

QUICK REFERENCE

Via dell’Artigiano, 8/6
40016 San Giorgio di Piano
(Bologna) ITALY
Email: grifo@grifo.it

http://www.grifo.it http://www.grifo.com
Tel. +39 051 892.052 (a.r.) FAX +39 051 893.661

BASIC 52 Edition 3.00 Rel. 11 January 1999

grifo®, GPC®, are trade marks of grifo®
MCS BASIC 52 is a powerful software tool, capable of managing a BASIC high level programmation of all Intel 51 family based cards. It is a "romated" software that generates "romable" software in an easy to use environment; it also reduces the necessity of external hardware (in circuit emulator, EPROM programmer, etc.) and at the same time it speeds up debugging phase of the User application program.

MCS BASIC 52 is referred to generic software tools, but each cards has a specific version of software associated to their hardware features; so for each card the name MCS BASIC 52 become BASIC followed by the card final name.
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.
GENERAL INDEX

QUICK REFERENCE TO MCS BASIC 52 ......................................................................................... 1
CONVENTIONS .................................................................................................................................. 1
OPERATOR LIST .............................................................................................................................. 1
INSTRUCTIONS LIST .................................................................................................................... 2

BASIC 52 MODIFICATIONS FOR GRIFO®'S CARDS ................................................................. 12
REMOVED COMMANDS, INSTRUCTIONS, OPERATORS ......................................................... 12
ADDED COMMANDS ....................................................................................................................... 12
ADDED OPERATORS ....................................................................................................................... 12
ADDED INSTRUCTIONS .................................................................................................................. 12
SECOND SERIAL MANAGEMENT (SOFTWARE SERIAL LINE) ......................................................... 13
A/D CONVERTER MANAGEMENT ................................................................................................. 13
REAL TIME CLOCK INTERRUPT MANAGEMENT ......................................................................... 14
BLOCK READ/WRITE ON SERIAL EEPROM AND RAM RTC ..................................................... 14
BYTE READ/WRITE ON SERIAL EEPROM AND RAM RTC ......................................................... 14
OPERATOR KEYBOARD MANAGEMENT ....................................................................................... 15
OPERATOR DISPLAY SELECTION AND INITIALIZATION .......................................................... 15
82C55 INITIALIZATION FOR CONSOLE REDIRECTION MANAGEMENT ................................. 15
SFR (SPECIAL FUNCTION REGISTER) READ/WRITE ................................................................. 16
REAL TIME CLOCK MANAGEMENT .............................................................................................. 17
PWM LINES MANAGEMENT .......................................................................................................... 17
FIGURE INDEX

FIGURE 1: ADDED INSTRUCTION TABLE ................................................................. 13
FIGURE A-1: KDx x24 ELECTRIC DIAGRAM ......................................................... A-1
FIGURE A-2: QTP 24P ELECTRIC DIAGRAM PART 1 ........................................... A-2
FIGURE A-3: QTP 24P ELECTRIC DIAGRAM PART 2 ........................................... A-3
FIGURE A-4: QTP 16P ELECTRIC DIAGRAM ...................................................... A-4
FIGURE A-5: PPI 82C55 ELECTRIC DIAGRAM ................................................... A-5
This quick reference to the BASIC 52 programming language lists the keywords alphabetically, along with brief descriptions of function and use.

CONVENTIONS

The reference uses the following typographic conventions:

KEYWORDS (boldface uppercase)
BASIC 52 keywords

placeholders (italics)
Variables, expressions, constants, or other information that you must supply

[optional items] (enclosed in square brackets)
Items that are not required

repeating elements... (followed by ellipsis (three dots))
You may add more items with the same form as the preceding item.

OPERATOR LIST

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>variable = expression</td>
<td>Assigns a value to a variable</td>
<td>C,R</td>
</tr>
<tr>
<td>expression = expression</td>
<td>Equivalence test (relational operator)</td>
<td>C,R</td>
</tr>
<tr>
<td>expression + expression</td>
<td>Add</td>
<td>C,R</td>
</tr>
<tr>
<td>expression - expression</td>
<td>Subtract</td>
<td>C,R</td>
</tr>
<tr>
<td>expression * expression</td>
<td>Multiply</td>
<td>C,R</td>
</tr>
<tr>
<td>expression / expression</td>
<td>Divide</td>
<td>C,R</td>
</tr>
<tr>
<td>expression ** expression</td>
<td>Raises first expression to value of second expression (exponent)</td>
<td>C,R</td>
</tr>
<tr>
<td>expression &lt;&gt; expression</td>
<td>Inequality test (relational operator)</td>
<td>C,R</td>
</tr>
</tbody>
</table>
expression < expression  
Less than test (relational operator)  

expression > expression  
Greater than test (relational operator)  

expression <= expression  
Less than or equal test (relational operator)  

expression >= expression  
Greater than or equal test (relational operator)  

**INSTRUCTIONS LIST**

?  
Same as PRINT

**ABS** (expression)  
Returns the absolute value of expression  

expression .AND. expression  
Logical AND  

**ASC**(character)  
Returns the value of ASCII character  

**ATN**(expression)  
Returns the arctangent of expression  

**BAUD** expression  
Sets the baud rate for LPT (pin 8). For proper operation, XTAL must match the system’s crystal frequency.  

**CALL** integer  
Calls an assembly-language routine at the specified address in program memory.  

**CBY**(expression)  
Retrieves the value at expression in program, or code, memory.  

**CHR**(expression)  
Converts expression to its ASCII character.  

**CLEAR**  
Sets all variables to 0, resets all stacks and interrupts evoked by BASIC.  

**CLEARI**  
Clears all interrupts evoked by BASIC. Disables ONTIME, ONEX1.
CLEARS
Resets BASIC 52’s stacks. Sets control stack = 0FEh, argument stack = 1FEh, internal stack = value in 3Eh in internal RAM.

CLOCK0
Disables the real-time clock.

CLOCK1
Enables the real-time clock.

CONT
Continues executing program after STOP or CONTROL+C.

COS(expression)
Returns the cosine of expression

CR
PRINT option. Causes a carriage return, but no line feed, on the host display.

DATA expression [,....,expression]
Specifies expressions to be retrieved by a READ statement.

DBY(expression)
Retrieves or assigns a value at expression in internal data memory.

DIM array name [(size)] [,....array name(size)]
Reserves storage for an array. Default size is 11 (0-10). Size limits are 0-254.

DO: [program statements]: UNTIL relational expression
Executes all statements between DO and UNTIL until relational expression is true.

DO: [program statements]: WHILE relational expression
Executes all statements between DO and WHILE until relational expression is false.

END
Terminates program execution.

EXP (expression)
Raises e (2.7182818) to the power of expression

FOR counter variable = start-count expression
TO end-count expression [STEP count-increment expression]: [program statements]: NEXT [counter variable]
Executes all statements between FOR and NEXT the number of times specified by the counter and step expressions.
**FPROG, FPROG1-FPROG6**  
Like PROG, PROG1-PROG6, but using Intelligent programming algorithm.

**FREE**  
Returns the number of bytes of unused external data RAM.

**GET**  
Contains the ASCII code of a character received from the host computer’s keyboard. After a program reads the value of GET (For example, G=GET), GET returns to 0 until a new character arrives.

**GOSUB line number**  
Causes BASIC 52 to transfer program control to a subroutine beginning at line number. A RETURN statement returns control to the line number following the GOSUB statement.

**GOTO line number**  
Causes BASIC 52 to jump to line number in the current program.

**IDLE**  
Forces BASIC 52 to wait for ONTIME or ONEX1 interrupt.

**IE**  
Retrieves or assigns a value to the 8052’s special function register IE.

**IF relational expression**  
**THEN** program statements  
**[ELSE] [program statements]**  
If relational expression is true, executes program statements following THEN. If relational expression is false, executes program statements following ELSE, if used.

**INPUT [“Prompt message”][,] variable [,variable] [,....variable]**  
Displays a question mark and optional prompt message on the host computer and waits for keyboard input. Stores input in variable(s). A comma before the first variable suppresses the question mark.

**INT(expression)**  
Returns integer portion of expression.

**IP**  
Retrieves or assigns a value to the 8052’s special function register IP.

**LD@ expression**  
Retrieves a 6-byte floating-point number and places it on the argument stack. Expression points to the most significant byte of the number.

**LEN**  
Returns the number of bytes in the current program.
[LET] variable = expression
Assigns a variable to the value of expression. Use of LET is optional.

LIST[line number][-line number]
Displays the current program on the host computer.

LIST# [line number][-line number]
Writes the current program to LPT (pin 8).

LIST@ [line number][-line number]
Writes the current program to a user-written assembly-language output driver at 40C3h. Setting bit 7 of internal data memory location 27H enables the driver.

LOG(expression)
Returns natural logarithm of expression.

MTOP [=highest address in RAM program space]
Assigns or reads the highest address BASIC 52 will use to store variables, strings, and RAM programs. Usually 7FFFh or lower, since EPROM space begins at 8000h.

NEW
Erases current program in RAM; clears all variables.

NOT (expression)
Returns 1’s complement (inverse) of expression.

NULL [integer]
Sets the number (0-255) of NULL characters (ASCII 00) that BASIC 52 sends automatically after a carriage return. Only very slow printers or terminals need these extra nulls.

ON expression GOSUB line number [,line number] [.....line number]
Transfers program control to a subroutine beginning at one of the line numbers in the list. The value of expression matches the position of the line number selected, with the first line number at position 0.

ON expression GOTO line number [,line number] [.....line number]
Transfers program control to one of the line numbers in a list of numbers. The value of expression matches the position of the line number selected, with the first line number at position 0.

ONERR line number
Passes control to line number following an arithmetic error. Arithmetic errors include ARITH. OVERFLOW, ARITH. UNDERFLOW, DIVIDE BY ZERO, and BAD ARGUMENT.
ONEX1 *line number*  
On interrupt 1 (pin 13), BASIC 52 finishes executing the current statement, and then passes control to an interrupt routine beginning at *line number*. The interrupt routine must end with RETI.

ONTIME *number of seconds, line number*  
When TIME = *number of seconds*, BASIC 52 passes control to an interrupt routine beginning at *line number*. The interrupt routine must end with RETI. CLOCK1 starts the timer.

*expression .OR. expression*  
Logical OR

P.  
Same as PRINT

PCON  
Retrieves or assigns a value to the 8052’s special function register PCON.

PGM  
Programs an EPROM, EEPROM, or NV RAM with data from memory. The following data must be stored in internal data memory in the locations listed:
- 1Bh,19h  High byte, low byte of first address of data to program
- 1Ah,18h  High byte, low byte of first address to be programmed - 1
- 1Fh,1Eh  High byte, low byte indicating number of bytes to program
- 40h,41h  High byte, low byte indicating width of programming pulse.  
  High byte = (65536 - pulse width in seconds * XTAL/12) / 256.  
  Low byte = (65536 - pulse width in seconds * XTAL/12) .AND. 0FFh.  
- 26h  For Intelligent programming, set bit 3.  
  For 50-millisecond programming, clear bit 3.

PH0.  
Same as PRINT, but displays values in hexadecimal format. Uses two digits to display values less than 0FFh.

PH0.#  
Same as PRINT#, but displays values in PH0. hexadecimal format

PH0.@  
Same as PRINT@, but outputs values in PH0. hexadecimal format.

PH1.  
Same as PRINT, but displays values in hexadecimal format. Always displays four digits.

PH1.#  
Same as PRINT#, but displays values in PH1. hexadecimal format.
**PH1.@**
Same as PRINT@, but outputs values in PH1. hexadecimal format.

**PI**
Constant equal to 3.1415926.

**POP variable [...variable]**
Assigns the value of the top of the argument stack to variable.

**PORT1**
Retrieves or assigns a value to PORT1 (pins 1-8).

**PRINT [expression] [...expression] []**
Displays the value of expression(s) on the host computer. A comma at the end of the statement suppresses the CARRIAGE RETURN/LINEFEED. Values are separated by two spaces. Additional PRINT options are CR, SPC, TAB, USING.

**PRINT#**
Same as PRINT, but outputs to LPT (pin 8). BAUD and XTAL values affect the PRINT# rate.

**PRINT@**
Same as PRINT, but outputs to a user-defined output driver. Requires an assembly language output routine at 403Ch in external program memory. Setting bit 7 of internal data memory location 24h enables the output routine.

**PROG**
Stores the current RAM program in the EPROM space.

**PROG1**
Saves the serial-port baud rate. On power-up or reset, BASIC 52 boots without having to receive a space character. The terminal’s baud rate must match the stored value.

**PROG2**
Like PROG1, but on power-up or reset, BASIC 52 also begins executing the first program in the EPROM space.

**PROG3**
Like PROG1, but also saves MTOP. On power-up or reset, BASIC 52 clears memory only to MTOP.

**PROG4**
Like PROG2, but also saves MTOP. On power-up or reset, BASIC 52 clears memory only to MTOP.
PROG5
Like PROG3, but also reads 5Fh in external data memory on power-up or reset. If 5Fh contains 0A5h, BASIC 52 doesn’t clear external data memory. If data memory location 5Eh contains 34h, BASIC 52 will automatically begin executing a program in external data memory.

PROG6
Like PROG5, but if external data memory location contains 5Fh, BASIC 52 calls a user-written assembly-language reset routine beginning at program memory 4039h.

PUSH expression [, ..., expression]
Places the values of expression(s) sequentially on BASIC 52’s argument stack.

PWM expression1, expression2, expression3
Outputs a pulse-width modulated (PWM) sequence of pulses on pin 3. Expression1 is the width of each high pulse, expressed in clock cycles. Expression2 is the width of each low pulse, expressed in clock cycles. Expression3 is the number of PWM cycles output. One clock cycle = 12/XTAL. One PWM cycle = one high pulse plus one low pulse. Expression1 and Expression2 must each be at least 25. Maximum for each Expression is 65535.

RAM
Selects the current program in the RAM space.

RCAP2
Retrieves or assigns a value to the 8052’s special function registers RCAP2H and RCAP2L.

READ variable [, ..., variable]
Retrieves the expressions in a DATA statement and assigns each expression to a variable.

REM
Introduces a comment, or remark. BASIC 52 ignores all text after REM in a program line.

RESTORE
Resets READ pointer to the first expression in the DATA statement.

RETI
Returns program control to the line number following the most recently executed ONEX1 or ONTIME statement.

RETURN
Returns program control to the line number following the most recently executed GOSUB statement.
RND
Returns a pseudo-random number between 0 and 1 inclusive.

ROM [program number]
Selects a program in the EPROM space (beginning at 8000h). Default program number is 1.

RROM [program number]
Changes to ROM mode and runs the specified program. Default program number is 1.

RUN
Executes the current program. Clears all variables.

SGN (expression)
Returns +1 if expression >=0, zero if expression = 0, and -1 if expression <0.

SIN(expression)
Returns the sine of expression

SPC (expression)
PRINT option. Causes the display to place expression additional spaces (besides the minimum two) between values in a PRINT statement.

SQR(expression)
Returns square root of expression.

ST@ expression
Copies a 6-byte floating-point number from the argument stack to external data memory. Expression points to the most significant byte of the number.

STOP
Halts program execution.

STRING expressions, expression2
Allocates memory for strings (variables each consisting of a series of text characters).
Expression1 = (Expression2 * number of strings) + 1.
Expression2 = maximum number of bytes (characters) per string + 1. Executing STRING clears all variables. Maximum number of strings is 255.

T2CON
Retrieves or assigns a value to the 8052’s special function register T2CON.

TAB(expression)
PRINT option. Specifies the position (number of spaces) to begin displaying the next value in the PRINT statement.
**TAN(expression)**  
Returns the tangent of *expression*.

**TCON**  
Retrieves or assigns a value to the 8052’s special function register TCON.

**TIME**  
Retrieves or assigns a value, in seconds, to BASIC 52’s real-time clock.

**TIMER0**  
Retrieves or assigns a value to the 8052’s special function registers TH0 and TL0.

**TIMER1**  
Retrieves or assigns a value to the 8052’s special function registers TH1 and TL1.

**TIMER2**  
Retrieves or assigns a value to the 8052’s special function registers TH2 and TL2.

**TMOD**  
Retrieves or assigns a value to the 8052’s special function register TMOD.

**U.**  
PRINT option. Same as USING.

**UI0**  
Restores BASIC 52’s console input driver after using UI1.

**UI1**  
Allows a user-provided assembly-language console (host computer) input routine to replace BASIC 52’s console input driver. External program memory location 4033h must contain a jump to the user’s routine.

**UO0**  
Restores BASIC 52’s console output driver after using UI1.

**UO1**  
Allows a user-provided assembly-language console (host computer) output routine to replace BASIC 52’s console output driver. External program memory location 4030h must contain a jump to the user’s routine.

**USING (FN)**  
PRINT option. Causes BASIC 52 to output numbers in exponential format with N significant digits. BASIC 52 always outputs at least 3 significant digits. Maximum *expression* is 8.

**USING(0)**  
PRINT option. Causes BASIC 52 to output numbers from ±9.99999999 to ±0.1 as decimal fractions. Numbers outside this range display in USING (FN) format. USING(0) is the default format.
USING (#[...#][.]#[..#])
PRINT option. Causes BASIC 52 to output numbers using decimal fractions, with # representing the number of significant digits before and after the decimal point. Up to eight # characters are allowed.

XBY(expression) C,R
Retrieves or assigns a value in external data memory.

XFER C
Copies the current program from the EPROM space (beginning at 8010h for program 1) to RAM (beginning at 200h), and selects RAM mode.

eexpression .XOR. eexpression C,R
Logical exclusive OR

XTAL C,R
Assigns a value equal to the system’s crystal frequency, for use by BASIC 52 in timing calculations.
BASIC 52 MODIFICATIONS FOR GRIFO®'S CARDS

Here follows a brief description of MCS BASIC 52 variation=**BASIC xxx**, performed by grifo® to satisfy all user's requests.

**REMOVED COMMANDS, INSTRUCTIONS, OPERATORS**

<table>
<thead>
<tr>
<th>Removed commands</th>
<th>Removed instruction</th>
<th>Removed operators</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST#</td>
<td>BAUD</td>
<td>None</td>
</tr>
<tr>
<td>FPROG</td>
<td>PRINT#</td>
<td></td>
</tr>
<tr>
<td>FPROG1</td>
<td>PH0.#</td>
<td></td>
</tr>
<tr>
<td>FPROG2</td>
<td>PH1.#</td>
<td></td>
</tr>
<tr>
<td>FPROG3</td>
<td>PWM</td>
<td></td>
</tr>
<tr>
<td>FPROG4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FPROG5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FPROG6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ADDED COMMANDS**

**ERASE**  ->  Deletes EEPROM content removing all the application program saved in with command PROG,PROG1,...,PROG6.

**ADDED OPERATORS**

None.

**ADDED INSTRUCTIONS**

Here a summary of the differences between original MCS BASIC 52 and BASIC for GRIFO's cards. This additions are really interesting to manage on board hardware resources with high level intructions. With these instructions the development of the application program is really faster and easier, even for first time users.
SECOND SERIAL MANAGEMENT (SOFTWARE SERIAL LINE)

**COM2 (expression)**

This procedure manages all the operation on the software serial line. For the transmission on this line, the user must use the PRINT@... instruction, while for the reception, there is a buffer allocated in EXTERNAL RAM. If the software serial line management is active, the user can't use the TIMER 0 instruction because this timer is used as baud rate generator.

(expression) --> 
0 = It disables the software serial line
1 = It disables the software serial line at 1200 BAUD
2 = It disables the software serial line at 2400 BAUD
3 = It disables the software serial line at 4800 BAUD
4 = It reads the characters number already saved in the reception buffer
5 = It resets the reception buffer

A/D CONVERTER MANAGEMENT

**A_D (expression)**

It performs an A/D conversion of an analog input. The conversion is made on the request channel and the result is returned to the main program.

(expression) --> Channel number (0 to 7)
REAL TIME CLOCK INTERRUPT MANAGEMENT

ALARM \((\text{expression1}), (\text{expression2}), (\text{expression3}), (\text{expression4}), (\text{expression5}), (\text{expression6}), (\text{expression7}), (\text{expression8})\)

It enables the interrupt of RTC so it can generate time based and to manage the alarm.

\((\text{expr1})\) \(\rightarrow\) 0 = It enables NO CLOCK ALARM (ALARM MODE)
1 = It enables DAILY ALARM (ALARM MODE)
2 = It enables WEEKDAY ALARM (ALARM MODE)
3 = It enables DATED ALARM (ALARM MODE)
4 = It enables TIMER (TIMER MODE)
5 = Reset flag of ALARM

TIMER MODE

\((\text{expr2})\) \(\rightarrow\) 0 to 99 = Count byte
\((\text{expr3})\) \(\rightarrow\) 0 = No timer
1 = It counts "CENTS OF SECOND"
2 = It counts "SECONDS"
3 = It counts "MINUTES"
4 = It counts "HOURS"
5 = It counts "DAYS"

ALARM MODE

\((\text{expr2})\) \(\rightarrow\) Byte with HOURS value (0 to 23).
\((\text{expr3})\) \(\rightarrow\) Byte with MINUTES value (0 to 59)
\((\text{expr4})\) \(\rightarrow\) Byte with SECONDS value (0 to 59)
\((\text{expr5})\) \(\rightarrow\) Byte with DAY OF WEEK value (0 to 59)
\((\text{expr6})\) \(\rightarrow\) Byte with DAY OF MONTH value (1 to 31)
\((\text{expr7})\) \(\rightarrow\) Byte with MONTH value (1 to 12)
\((\text{expr8})\) \(\rightarrow\) Byte with YEAR value (0 to 3)

BLOCK READ/WRITE ON SERIAL EEPROM AND RAM RTC

\text{BL}_\text{EE} \((\text{expression1}), (\text{expression2}), (\text{expression3})\)

It performs a data block read or write operation at a specified address, on serial EEPROM. The W/R data buffer is located in EXTERNAL RAM address.

\((\text{expression1})\) \(\rightarrow\) 0 = Reading of a data block
1 = Writing of a data block
\((\text{expression2})\) \(\rightarrow\) Initial location address (0 to last device address)
\((\text{expression3})\) \(\rightarrow\) Number of bytes to write or read (1 to 255)

BYTE READ/WRITE ON SERIAL EEPROM AND RAM RTC

\text{BY}_\text{EE} \((\text{expression1}), (\text{expression2}), (\text{expression3})\)

It performs a byte read or write operation at a specified address, on serial EEPROM. The user must remember that in read procedure the \((\text{expression3})\) parameter must be given even if it has no meaning.

\((\text{expression1})\) \(\rightarrow\) 0 = Reading of byte
1 = Writing a byte
\((\text{expression2})\) \(\rightarrow\) Location address (0 to last device address)
\((\text{expression3})\) \(\rightarrow\) Byte to write (0 to 255)
OPERATOR KEYBOARD MANAGEMENT

KEYB \((expression)\)
It enables or disables the matrix keyboard scanning and reads the possible key pressed code. This procedure can start or stop a periodic keyboard scanning, with a debouncing on the pressed key, or it can return the pressed key code (0 if no key is pressed) through the stack.
\((expression)\) --> 0 = Keyboard scanning OFF.
1 = Keyboard scanning ON.
2 = Return the pressed key code (0 if no key is pressed) through the stack. The keyboard scanning is enabled if it was OFF.

OPERATOR DISPLAY SELECTION AND INITIALIZATION

DISPLAY \((expression)\)
It inizializes the selected display. Remember that the user must call this new instruction before using the output ridirection (UO1) instruction.
\((expression)\) --> 0 = FUTABA 20x2
1 = FUTABA 40x1
2 = FUTABA 40x2
3 = FUTABA 40x4
4 = LCD 20x2
5 = LCD 20x4
6 = LCD 40x2
7 = LCD 40x4

82C55 INITIALIZATION FOR CONSOLE REDIRECTION MANAGEMENT

P8255 \((expression)\)
It initializes PPI 82c55 so it can manage a user pannell. It is necessary to call it only once before to use the other user pannell instruction (KEYB,DISPLAY,UO1).
\((expression)\) --> 0 = PORT in INPUT
1 = PORT in OUTPUT
SFR (SPECIAL FUNCTION REGISTER) READ/WRITE

**RW_SFR** *(expression1),(expression2),(expression3)*

It performs a special function register (SFR) read or write operations.

The user must remember that in "read procedure" the *(expression3)* parameter must be given even if it has no meaning. The SFR identification byte is a numeric code, with the following meaning:

<table>
<thead>
<tr>
<th>SFR NAME</th>
<th>SFR CODE</th>
<th>SFR NAME</th>
<th>SFR CODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>for GPC®</td>
<td>552,553,554</td>
<td>for GPC®</td>
<td>323,324</td>
</tr>
<tr>
<td>CTCON</td>
<td>0</td>
<td>DPL1</td>
<td>0</td>
</tr>
<tr>
<td>CTH3</td>
<td>1</td>
<td>DPH1</td>
<td>1</td>
</tr>
<tr>
<td>CTH2</td>
<td>2</td>
<td>DPS</td>
<td>2</td>
</tr>
<tr>
<td>CTH1</td>
<td>3</td>
<td>CKCON</td>
<td>3</td>
</tr>
<tr>
<td>CTH0</td>
<td>4</td>
<td>EXIF</td>
<td>4</td>
</tr>
<tr>
<td>CMH2</td>
<td>5</td>
<td>SCON1</td>
<td>5</td>
</tr>
<tr>
<td>CMH1</td>
<td>6</td>
<td>SBUF1</td>
<td>6</td>
</tr>
<tr>
<td>CMH0</td>
<td>7</td>
<td>TA</td>
<td>7</td>
</tr>
<tr>
<td>CTL3</td>
<td>8</td>
<td>WDCON</td>
<td>8</td>
</tr>
<tr>
<td>CTL2</td>
<td>9</td>
<td>EIE</td>
<td>9</td>
</tr>
<tr>
<td>CTL1</td>
<td>10</td>
<td>EIP</td>
<td>10</td>
</tr>
<tr>
<td>CTL0</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CML2</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CML1</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CML0</td>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IEN1</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IP1</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTE</td>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S1ADR</td>
<td>18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S1DAT</td>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S1STA</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S1CON</td>
<td>21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STE</td>
<td>22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMH2</td>
<td>23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TML2</td>
<td>24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TM2CON</td>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TM2IR</td>
<td>26</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T3</td>
<td>27</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td>28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P5</td>
<td>29</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(expression1) --> 0 to 1 = R/W selection byte (0=Reading; 1=Writing).
(expression2) --> 0 to 29 = SFR identification byte.
(expression3) --> 0 to 255 = Byte to write.
REAL TIME CLOCK MANAGEMENT

GES_RTC (expression1),(expression2),(expression3),(expression4),
(expression5),(expression6),(expression7),(expression8)
It initializes the RTC or return date or its time. The user must remember that in read procedure all
parameters must be given even if they have no meaning.

(expr1) --> 0 = Reading of HOUR, MINUTE, SECOND.
--> 1 = Reading of DAY of WEEK, DAY, MONTH, YEAR.
--> 2 = It initializes of the RTC.

(expr2) --> Byte to write hours (0 to 23)
(expr3) --> Byte to write minutes (0 to 59)
(expr4) --> Byte to write seconds (0 to 59)
(expr5) --> Byte to write the day of week (0 to 6)
(expr6) --> Byte to write the day of month (1 to 31)
(expr7) --> Byte to write month (1 to 12)
(expr8) --> Byte to write year (0 to 3)

PWM LINES MANAGEMENT

SET_PWM (expression1),(expression2),(expression3)
It generates PWM signals on CPU line.

(expression1) --> PWM line selection
(expression2) --> Frequency
(expression3) --> Duty_Cycle (0 to 100%)
If (expression2) and (expression3) are both set to 0 the PWM line is set and maintained at "0" logic
value.
If (expression2) and (expression3) are both set to 1 the PWM line is set and maintained at "1" logic
value.
**APPENDIX A: OPERATOR INTERFACE ELECTRIC DIAGRAM**

**FIGURE A-1: KDx x24 ELECTRIC DIAGRAM**

- **LCD20x2**
  - LCD 20x2 Futaba VFD
- **LCD20x4**
  - LCD 20x4

**Components**
- **R1**: 47Ω
- **R2**: 47Ω
- **R3**: 22KΩ 9+1 SP
- **R4**: 22KΩ 9+1 SP
- **R5**: 10KΩ trimmer
- **C1**: 100nF
- **C2**: 22µF 6.3V Tant anium
- **C3**: 100nF
- **C4**: 100nF
- **C5**: 22µF 6.3V Tant anium
- **CN1**: 2 pins mini male connector
- **CN2**: 10 pins male strip
- **CN3**: 20 pins male low profile c connector
- **CN4**: LCD L214 (20x4)
- **CN5**: Futaba VFD 20x2
- **CN6**: LCD L2012 (20x2)
- **IC1**: 7407

**Title**: KDL/F-2/424  **grifo®**

**Date**: 9-12-1998  **Rel.** 1.2

**Page**: 1  of  1
FIGURE A-2: QTP 24P ELECTRIC DIAGRAM PART 1
**Figure A-4: QTP 16P Electric Diagram**

- **Title:** QTP 16P
- **Date:** 22-07-98
- **Rel.:** 1.2
- **Page:** 1 of 1
Figure A-5: PPI 82C55 Electric Diagram

Title: PPI example
Date: 16/11/1998
Page: 1 of 1

Basic 52-Basic xxx Rel.3.00

Page A-5