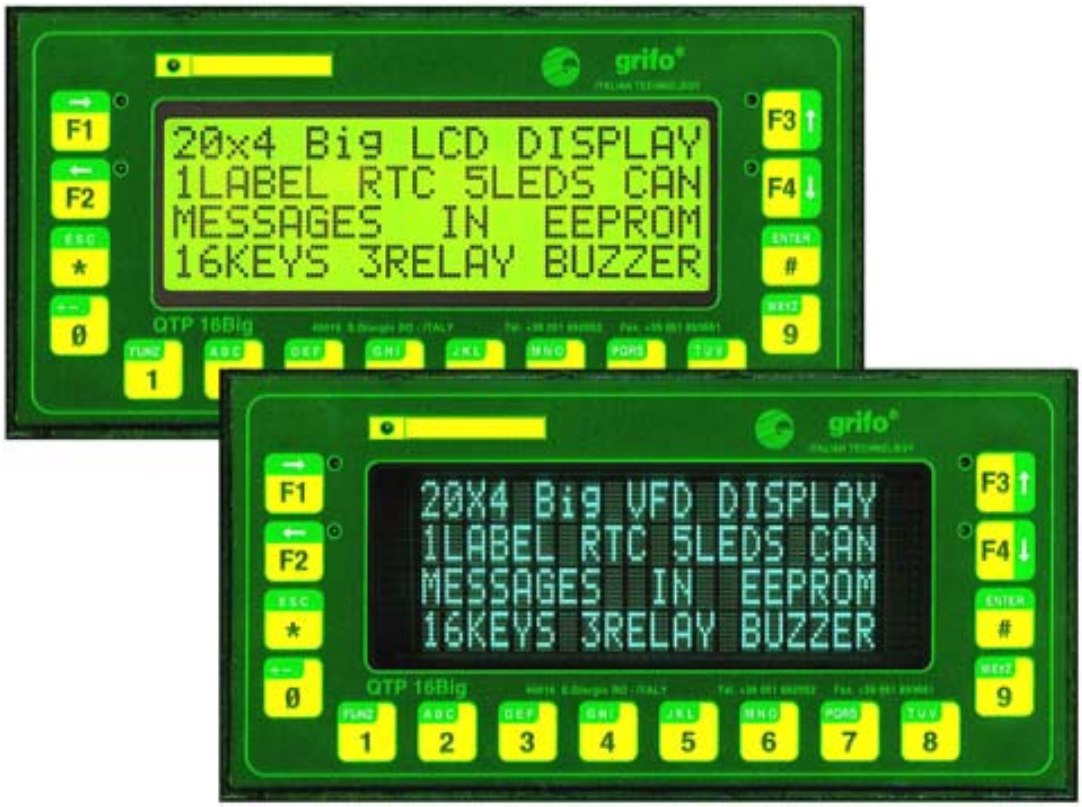
QTP 16Big Library

Library for Quick Terminal Panel 16 keys Big display

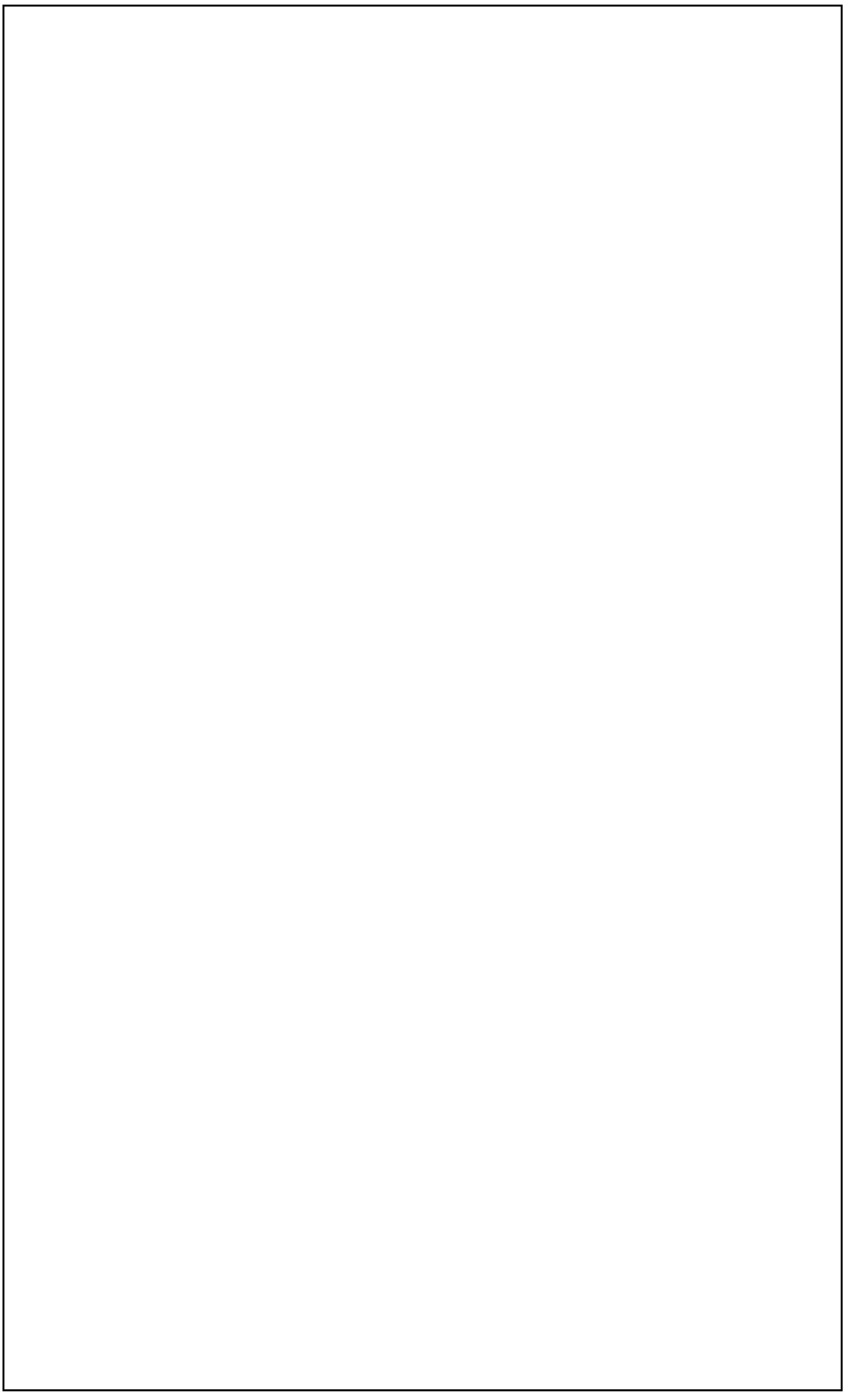
USER MANUAL



grifo®
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

QTP 16Big.LIB Rel. 5.00 *Edition 29 November 2006*



QTP 16Big Library

Library for Quick Terminal Panel 16 keys Big display

USER MANUAL

The **QTP 16Big** library is a firmware properly realized to let any users develop his own **Application Program** in a comfortable, fast and efficacious way. With this library the **QTP 16Big** acts as a powerful **Controller Complete of Operator Interface** and capable to work either alone or joined with other systems.
A rich list of **Commands** , that can be directly called with relative **Parameters** and **Results**, allows the customers to produce the application programs that satisfy all their requirements in the best mode, with a really short **Development time**. These commands cover the normal demands of industrial environment and they are the results of decennial experiences in the sector.
The **QTP 16Big.LIB** is not a finished product ready for installation but it must be previously **Specialized** from the user. This specialization can be done with comfortable and cheap **Development Tools**, either at high and low level; they transform the **QTP** in a very flexible and versatile product. Infact the user application program, that specialize it, allows the solution of every problems even with high complexity, and at the same time it allows the development of different applications, by using the same hardware.
The greater number of the develoment tool for **I51** family of microcontrollers can use the **QTP 16Big** library and they provide comfortable **Debug** modalities of the application program, by reducing again the total preparation time. Among these ones we can remind: **Assembler**; **PASCAL** compilers (SYS51PW); **C Compilers** (HTC 51, SYS51CW, µC/51); **Contacts Logic** (LadderWORK); **BASIC** compilers (BASCOM-8051); etc.

grifo®
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

QTP 16Big.LIB Rel. 5.00 *Edition 29 November 2006*

DOCUMENTATION COPYRIGHT BY grifo®, ALL RIGHTS RESERVED

IMPORTANT

Although all the information contained herein have been carefully verified, grifo® assumes no responsability 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


Attention: ESD sensitive device

Trade Marks

abaco, **GPC®**, grifo® : are trade marks of grifo®.
Other Product and Company names listed, are trade marks of their respective companies.

GENERAL INDEX

INTRODUCTION 1

VERSION 3

GENERAL INFORMATION 4

REQUIREMENTS 6
- QTP 16BIG 6
- PERSONAL COMPUTER 6
- SERIAL COMMUNICATION CABLE 6
- WORKING SOFTWARE 7
  - FLASH EPROM WRITING PROGRAM: FLIP 7
  - SERIAL TERMINAL EMULATION PROGRAM 7
  - DEVELOPMENT TOOLS FOR APPLICATION PROGRAM 8

QTP 16BIG LIBRARY DESCRIPTION 9
- QTP 16BIG CONFIGURATION 9
- COMMUNICATION BUFFERS 9
- INTEGRATION AND USE OF LIBRARY 10
- RESOURCES USED BY LIBRARY FIRMWARE 14
- FLASH EPROM PROGRAMMING 16
- LOCAL SETUP 19
- COMMUNICATION MODALITY 20
- DEMO PROGRAMS FOR LIBRARY 20

QTP 16BIG LIBRARY USED WITH µC/51 22
- LIBRARY INTEGRATION ON µC/51 22
- SUPPLIED FILES WITH µC/51 23
- HOW TO START WITH µC/51 24

QTP 16BIG LIBRARY USED WITH BASCOM 8051 28
- LIBRARY INTEGRATION ON BASCOM 8051 28
- SUPPLIED FILES WITH BASCOM 8051 29
- HOW TO START WITH BASCOM 8051 30

BIBLIOGRAPHY 35

APPENDIX A: ON BOARD DEVICES DESCRIPTION A-1
- T89C5115 OR T89C51CC02 MICROCONTROLLER A-1
- I51 FAMILY A-2
- OPTIONAL EEPROM A-3
- RTC+SRAM PCF8583 ..... A-4

APPENDIX B: ALPHABETICAL INDEX B-1

FIGURES INDEX

FIGURE 1: APPLICATION EXAMPLE WITH QTP 16BIG.LIB 5
FIGURE 2: SERIAL CONNECTION BETWEEN QTP 16BIG AND DEVELOPMENT PC 6
FIGURE 3: CODE AREA ORGANIZATION WITH LIBRARY 11
FIGURE 4: RAM USAGE WITH LIBRARY 15
FIGURE 5: FLIP SETTINGS WINDOW (1 OF 4) 16
FIGURE 6: FLIP SETTINGS WINDOW (2 OF 4) 17
FIGURE 7: FLIP SETTINGS WINDOW (3 OF 4) 17
FIGURE 8: FLIP SETTINGS WINDOW (4 OF 4) 18
FIGURE 9: DEVELOPMENT MODE WITH LIBRARY 19
FIGURE 10: SERIAL COMMUNICATION BLOCK DIAGRAM 21
FIGURE 11: EXECUTION OF µC/51 DEMO PROGRAM 25
FIGURE 12: FLASH EPROM PROGRAMMING WITH µC/51 26
FIGURE 13: PROGRAM COMPILE WITH µC/51 27
FIGURE 14: SERIAL CONFIGURATION WITH BASCOM 8051 30
FIGURE 15: EXECUTION OF BASCOM 8051 DEMO PROGRAM 31
FIGURE 16: PROGRAMMER CONFIGURATION WITH BASCOM 8051 32
FIGURE 17: FLASH EPROM PROGRAMMING WITH BASCOM 8051 32
FIGURE 18: COMPILER CONFIGURATION WITH BASCOM 8051 33
FIGURE 19: PROGRAM COMPILE WITH BASCOM 8051 34

INTRODUCTION

The use of these devices has turned - **IN EXCLUSIVE WAY** - to specialized personnel.

This device is not a **safe component** as defined in directive **98-37/CE**.

Pins of module are not provided with any kind of ESD protection. Many pins of the card are directly connected to their respective pins of on board's components and these last are sensitive to electrostatic noises. So personnel who handles the product/s is invited to take all necessary precautions that avoid possible damages caused by electrostatic discharges.

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

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

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

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

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

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

grifo® provid this documentation "as is" without warranty of any kind. In no event shall **grifo®** be liable for indirect, special, incidental or consequential damages of any kind arising from any error in this documentation, including any loss or interruption of business, profits, use , or data. Moreover is not guaranteed the updating of the product for new computers or new operating systems, that will become available in the future.

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.

VERSION

This handbook make reference to version **2.1** of **QTP 16Big library** and following ones. The validity of the information contained in this manual is subordinated to the version number of the used firmware and the user must always verify the correct correspondence beetween the notations. The user can get the version number in different modes:

- from the card, as described in **QTP 16Big** user manual;
- through a proper command, provided in the library;
- in the name of the received library file.

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

In addition, this manual reports information about other different programs that are integrant parts of the library: each one of these programs has an own version number that is specifically described when it is necessary. Finally also the hardware is provided of his version as indicated in the related manuals.
When the user requires technical assistance it is really important that he provides a description of the problem plus the version numbers of all the used products.

Like any products, also **QTP 16Big** library is continuosly changed and improved to satisfy completely the new requirements of the users and correct the discovered problems and bugs. Here follows a brief description of the changes made to the package according to the version number:

Ver. 1.0 -> First version for internal development and test.

Ver. 1.3 -> Realized management of **QTP 12/R84**.

Ver. 2.1 -> Modified for management of **QTP 16Big** with printed circuit version **110705**.

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

GENERAL INFORMATION

The**QTP 16Big** library coincides with a firmware specifically realized to let any users develop his own **application program** in a comfortable, fast and efficacious way. With this library the **QTP 16Big** acts as a powerful **controller complete of operator interface** and capable to work either alone or joined with other systems.
A rich list of **commands** , that can be directly called with relative **parameters** and **results**, allows the customers to produce the application programs that satisfy all their requirements in the best mode, with a really short **development time**. These commands cover the normal demands of industrial environment and they are the results of decennial experiences in the sector.

In this manual the abbreviation **QTP 16Big.LIB** is used to identify the library in object with a shorter name.

The **QTP 16Big.LIB** is not a finished product ready for installation but it must be previously **specialized** from the user. This specialization can be done with comfortable and cheap **development tools**, either at high and low level; they transform the **QTP** in a very flexible and versatile product. Infact the user application program, that specialize it, allows the solution of every problems even with high complexity, and at the same time it allows the development of different applications, by using the same hardware.
The greater number of the develoment tools for **I51** family of microcontrollers can use the **QTP 16Big** library and they provide comfortable **debug** modalities of the application program, by reducing again the total preparation time. Among these ones we can remind: **Assembler**; **PASCAL** compilers (SYS51PW); **C** compilers (HTC 51, SYS51CW, µC/51); **Contacts logic** (LadderWORK); **BASIC** compilers (BASCOM-8051); etc.

The present manual contains the detailed user information relative only to library while all the other information about connectors, configurations, commands, connections, mounting and installation are available in the **QTP 16Big** user manual. All the indications of this last manual that are modified on **QTP 16Big.LIB**, are reported with the suitable modifications, inside paragraphs that have the same name. So, the user must take the **QTP 16Big** manual and integrate it with the **QTP 16Big Library** manual, by performing a substitution of the homonimous paragraphs of the first manual with those of the second.
The library can be used also for the **QTP 16BigH** variant, that is the version without keyboards and LEDs: in this case the present manual must be integrated with the correspondent user manual. The user must remind that all the **QTP 16Big** indications reported in this manual refer indifferently both to **QTP 16Big** and **QTP 16BigH**.

Main general features of library, are as follows:

- It can be easily used to develop numerous **operator interface** applications.
- **Reduces** the **developing time**, ans consequently **cost**, of the application.
- It uses a very small quantity of **hardware resources**.
- It doesn't use the **asynchronous serial line** that remain available for the connection with other systems, for the **debug** of application and for the **console** commununication.
- It is featured by a **fast execution time** that allows the solution of problems with reduced response time, too.
- It can be **integrated** and/or **linked** at the greater part of the **programming language** and **development tools**, available on the market.

- It includes numerous **commands** equal to as many functionalities.
- It mantains **compatibility** with other firmwares of **QTPs**.
- It provides an **easy call modality** to the commands and a comfortable **exchange** tecnique of **parameters** and **responses**.
- It satisfy the normal requirements of **industrial** environments.
- For the application development it requires only a standard PC without any additional hardware.
- Comfortable selection of working modality (AUTORUN or DEBUG).
- For the development tools proposed by **grifo®** the user is provided of a complete description of both the configurations to perform the integration and the directives that arrange the application program.
- Large documentation and rich list of examples, both in executable and source format.
- No license, nor additional cost. The user is free to develop all the applications that he requires.

In this manual are described all the features of **QTP 16Big** library, that are sufficient for a pratical and complete use of the product.

FIGURE 1: APPLICATION EXAMPLE WITH QTP 16BIG.LIB

REQUIREMENTS

Below is described the list of the fundamental materials (hardware and software) that are necessary to operate with **QTP 16Big** library:

QTP 16BIG

It is the operator panel **QTP 16Big** or **QTP 16BigH** that belong to grifo® industrial cards set. As detailed described in user manual, this product can be supplied with two different display and with some additional options. The library can manage all the display models and any combination of the options.
Anyway the choice of the panel configuration must be done according with the specific requirements of the application to develop.

PERSONAL COMPUTER

The **QTP 16Big** library requires a personal computer, up to now named **development PC**, provided of the following minimum features:

Personal Computer:	IBM or compatible (with CPU ≥ 486)
RAM memory:	≥ 32 MBytes
Operating system:	Windows 98, ME, 2000, NT, XP
Monitor:	Colours
Mass memory disks:	CD-ROM reader
	Hard disk with 30 MBytes free
Mouse:	Microsoft compatible, with own driver installed
Interfaces:	One free COMx serial line, in RS 232, following V24 specifications.

SERIAL COMMUNICATION CABLE

Some of the phases provided for the library require a serial connection between one of the free serial line of development PC and the serial line of **QTP 16Big**. This connection consist only of the traditional communication signals (transmit data, receive data and ground) and it must follow the V24 normative of C.C.I.T.T.

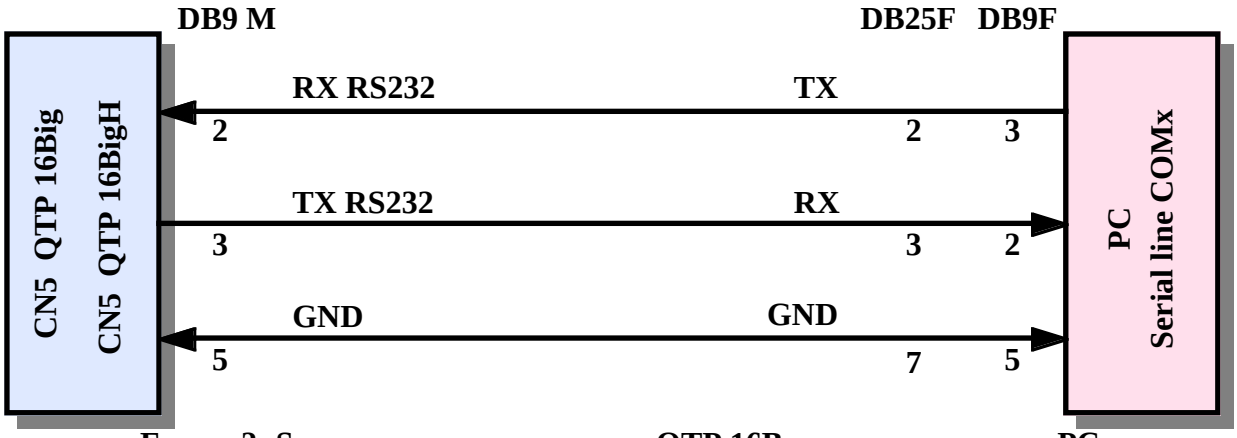


FIGURE 2: SERIAL CONNECTION BETWEEN QTP 16BIG AND DEVELOPMENT PC

On PC side, whenever there aren't any serial lines, it can be used specific converters that, once added to used PC, supply a complete RS 232 serial line. Among these devices we can remind the USB <-> RS 232 converters, the ETHERNET <-> RS 232 converters, the multi I/O cards with additional RS 232 lines, etc. Naturally these added serial lines can be used only if they are correctly installed either for hardware and software aspects, as defined by the manufacturing company.
If the user doesn't want to realize the described communication cable, or he wants to save time, grifo® supplies a ready to use accessory, that can be ordered with the code **CCR 9+9R**.

WORKING SOFTWARE

In addition to the described hardware in order to correctly operate with **QTP 16Big** library are necessary some working sofwares that develop, debug and set up the application program. These softwares are organized under the form of complete packages and they can be divided in some principal groups, as below described.

FLASH EPROM WRITING PROGRAM: FLIP

As described in following paragraphs the **QTP 16Big** library must be saved on the microcontroller FLASH EPROM together with user application program. The saving is performed through an ISP tecnique (In System Programming) that reduces the costs and the times for application development in fact it eliminates the use of external EPROMs, programmers, erasers, simulators, etc.
The ISP programmation requires only the development PC that executes a proper management program, named FLIP (FLexible In system Programming); it interacts with a Boot Loader available on microcontroller side and it is capable to read, erase, verify, program either the FLASH or the EEPROM memory. Everything happen through a simple serial connection between development PC and **QTP 16Big.LIB** normally done with RS 232 serial line (see figure 2) or alternatively with CAN line (for this possibility contact directly grifo®).
The **FLIP** program is supplied in the materials received with the order of QTP 16Big library, but it can be freely downloaded from ATMEL web site, by following the links:

"Products | Microcontrollers | 8051 Architecture | Tools & Software | FLIP"

SERIAL TERMINAL EMULATION PROGRAM

It is a generic communication program capable to manage a classic terminal emulation, with a selectable physical communication protocol. This program is used as a console by the supplied demo programs and it must show on monitor each characters received from serial line and transmit, on the same line, all the keys typed on PC keyboard.
For this purpose we remind the **GET51** program developed by grifo®, the famous **HYPERTERMINAL** of Windows, or the diffused programs **CROSS TALK**, **PROCOMM**, **BITCOMM**, **TERMINAL**, etc. developed by third part companies.

DEVELOPMENT TOOLS FOR APPLICATION PROGRAM

The application program, must be previously generated, befor it can be saved on **QTP 16Big**. Many suitable development tools can be used to generate the program in a comfortable way, in fact they let the user write it on the development PC and then convert it in machine code, used by microcontroller.
Generally all software packages available for the mounted microcontroller (or in other words the numerous tools for the I51 family) can be used, either at high and low level. The software development tools supplied by **grifo®** always include many example programs, libraries with console redirection , header files and accessories that integrate the library and make it ready to use. Among these we remind:

HI TECH C 51: cross compiler for C source programs. It is a powerful software tool that includes editor, C compiler, assembler, optimizer, linker, and remote debugger, in one easy to use Integrated Development Environment. Moreover the libraries sources files are included.

SYS51CW: cross compiler for C source program. It is a powerful software tool that provides editor, C compiler, assembler, optimizer, linker, library and remote symbolic debugger, included in an easy to use IDE for Windows.

SYS51PW: cross compiler for PASCAL source program. It is a powerful software tool that provides editor, PASCAL compiler, assembler, optimizer, linker, library and remote symbolic debugger, included in an easy to use IDE for Windows.

DDS MICRO C 51: low cost cross compiler for C source program. It is a software tool that provides editor, C compiler (integer), assembler, linker, and a remote debugger based on a monitor firmware, in a complete IDE. There are also included the libraries sources and some utilities programs.

BASCOM 8051: low price cross compiler for BASIC source programs. It is based on a comfortable IDE for Windows that provides editor, BASIC compiler and a powerful simulator for source debugging. Many memory models, data types and instructions are available for a direct use of hardware resources.

µC/51: comfortable, low cost, development tools with a colplete IDE that allows to use an editor, an ANSI C compiler, an assembler, a linker and a remote source level debugger user configurable. Souces of main libraries and of remote debugger are included, and so several utilities and demo programs.

LADDER WORK: It is an easy to use tools to generate automation application using the very famouts and diffused contacts logic. It includes a graphic editor to place and connect hardware components of the card (like digital I/O, counters, A/D, etc.) like on an electric diagram and define their properties, an efficent compiler to create the executable code and an utility to download it to card memories. Integrated IDE makes comfortable use of all these components under Windows operating system.

QTP 16BIG LIBRARY DESCRIPTION

In this chapter are described all the features of **QTP 16Big** concerning both its use on the operator panel and the generation of application program. If the user desires to speed up the first use of the library, he can execute direcly the steps of the HOW TO START.... paragraphs that, when required, refer to detailed info of this chapter.

QTP 16BIG CONFIGURATION

The **QTP 16Big** library can be used only if the same product has been correctly configured. This configuration regards mainly the asynchronous line for serial communication that is used for all the download, test and saving operations of the application program.
In detail the required configuration is those that set the serial line in RS 232:

J3, J4	->	not connected
J1	->	it must be connected and not connected during the use
IC11 socket	->	driver MAX 202

While all the other jumpers configurations don't care and they can be set according with working requirements. Whenever the asynchronous serial line in RS 232 is an unallowable configuration for the user application program, some alternative solution can be found: please contact directly grifo®.
It is important to remind that J1 jumper configuration must be frequently changed during the development of the application program; if the use of a normal female jumper link is uncomfortable, as an alternative, it could be used a connector that place the jumpers at a sufficient distance. The **QTP 16Big** user manual describes this specific connector (inside the accessories paragraph) and it includes some figures that report the jumpers and socket locations.

COMMUNICATION BUFFERS

QTP 16Big.LIB is provided of two communication buffers that allow the parameters and results exchange of the commands supplied by user application program.
The first is a receive buffer: it is **24 bytes** long, it memorizes each character received from user program and then it is examined at the end of the currently executed operation. Once the command receive is complete, the library execute it immediately; consequently the possible **QTP 16Big** problems of receive buffer full or in overflow can't happen. So the application program doesn't require any delay to avoid overflow of the receive buffer.
The second is a transmit buffer: it is **20 bytes** long, it memorizes each character the library must return to user application program and it is filled with the keys pressed codes and the executed commands responses. As the data remain in transmit buffer until the application program requires them, this buffer can become full and when this filling occours all following data are no more saved in the transmit buffer, and these are defenitely lost. So the user application program must manage data reception from **QTP 16Big.LIB** at least in two situations: before to send commands provided of responses (to empty the buffer for the same response) and periodically (to get the possible keys pressed).

INTEGRATION AND USE OF LIBRARY

The **QTP 16Big** library has been developed with the following aims:

- linkable with all the available programming languages;
- reduce the used hardware resources;
- mantain compatibility of use with firmwares of other **QTP**s;
- provide easy tecniques for the commands calls and parameters exchange;
- cover the normal and diffused requests of industrial environments;

that have defined the integration and use modalities of the same library, inside the user application program.

It is important to remind that the user of **QTP 16Big** library must have a basic knowledge of the used microcontroller and of embedded software development in fact the following documentation doesn't supply these information but it uses them. If the user have not this know how, he can found it in software development tool documentation and in microcontroller data sheets, reported in APPENDIX A of the manual.

In this paragraph are listed all the general information about integration and use of **QTP 16Big** library suitable for every user, with each development tools supplied by **grifo®** or other companies.

The integration and use of library needs some hardware and software tools properly specified in following descriptions. Their complete documentation is provided inside the same tools and it is not duplicated in this manual.

In conclusion the operations necessary to integate and use the library are:

a) Install the software development tool preselected to realize the application program, on the development PC. Generally all software packages available for the mounted microcontroller, or in other words the numerous tools for the I51 family can be used (assemblers, compilers, interpreters, etc.). All the development tools supplied by **grifo®** always include all the elements that completely integrate the library, by making it ready to use.

b) Install the ISP (In System Programming) tool on the development PC, that is the FLIP program capable to communicate with microcontroller Boot Loader, through a serial line, and that allows to read, erase and program the FLASH EPROM memory.
The ISP programmation reduces the costs and the times for application development in fact it eliminates the use of external EPROMs, programmers, erasers, simulators, etc. For a detailed explanation on ISP and FLIP, please consult the specific documementation released by ATMEL.

c) Prearrange the software development tool to ensure that the generated application program reserves the hardware resource used by library. As illustrated in figures 3, 4 and in next RESOURCES USED BY LIBRARY paragraph, the application program can't use the last area of FLASH, some internal RAM areas, one timer counter, the on board EEPROM, etc.

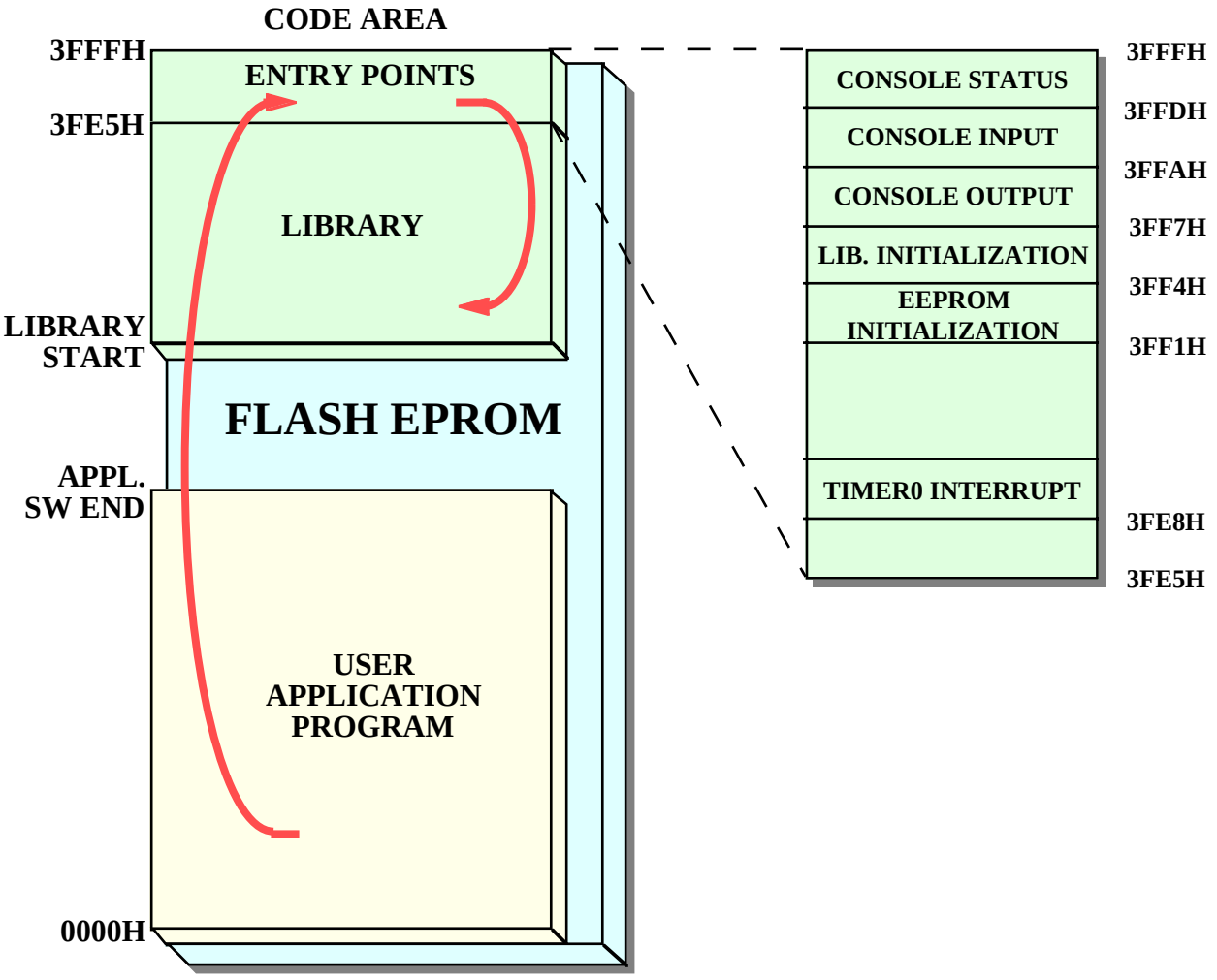


FIGURE 3: CODE AREA ORGANIZATION WITH LIBRARY

d) Physically the library coincides with an executable code that must be saved at the end of the code area of microcontroller, as described on figure 3. This code is supplied in QTP16Bxy.HEX file that, thanks to its HEX format, could be directly used for FLASH burning. On this memory, in addition to described library, it must be saved also the executable code of user application program at the beginning of area code to ensure its immediate execution after a power on or a reset. The transit from application program to library is performed through a proper entry points table, located at fixed addresses, that acts as a "link bridge" between the two codes saved in the single code area.
The choice of allocation addresses of the three areas on FLASH has been carefully made to obtain the maximum free space for the user application program, to have fixed entry points that don't change also when the library is updated and/or expanded and to have the same entry points addresses in all libraries of different **QTP**. With this choice the user can employ a new library version, or a different **QTP**, by simply reprogramming it on FLASH, with no intervents nor modifications, on his application program.

e) The **LIBRARY START** value is established by the same library, so it depends from its version; with the current version 2.1 it is fixed at **2C00H** and anyhow it can be easily obtained by loading the QTP16Bxy.HEX file (where xy corresponds to version number) and cheking its start address.

The user, after each generation of his application program, must verify that the **APPL. SW END** address is lower than **LIBRARY START** address or, in other words, that the two codes are not overlapped. This verification is easily performed in fact normally all the software development tools (assemblers, compilers, languages, etc.) inform about the dimension of the generated code and it will be sufficient to compare these information with the **LIBRARY START** value, before described.

f) Redirect the interrupt service routine for microcontroller TIMER0 to the corresponding library entry points illustrated in figure 3. The redirections must be performed following the rules of the development tool and normally it coincides with an absolute jump instructions (i.e. LJMP 3FE8H) to entry address, placed inside the relative interrupt service routine. Please remind that the TIMER0 interrupt redirection is absolutely necessary to obtain a complete functionality of the library; in detail it manages all the processes based on time as keyboard scanning, messages shift; time based activation of resources, intermittence, real time clock visualizations, keyclick, clock alarm, etc.

g) The commands execution, the parameters and results exchange and the use of the library are simplified by three procedures, with as many entry points, with the following features:

CONSOLE STATUS: returns the status of data presence that the library must send to application program; the data can be either a keypressed code or the answer of a command previously executed. The procedure has no input variables and a single output variable, saved in *Accumulator* register, that coincides with the number of characters ready to be sent to application program. This number of characters has also a status function in fact it is zeroed when there are no data and viceversa.

CONSOLE INPUT: waits the availibility of a data that library must send to application program and returns this data; also for this procedure the data can be either a keypressed code or the answer of a command previously executed. The procedure has no input variables and a single output variable, saved in *Accumulator* register, that coincides with the described data.

CONSOLE OUTPUT: sends a data from application program to library; the data can be a character to show on display or a command to execute or its possible parameters. The procedure has a single input variable, saved in *Accumulator* register, that coincides with the described data and no output variables.

In the application program about all the functionalities of the library are used with the three decribed procedures; it is really profitable that the user redirects the console management procedures of the selected software development tools, that already have a compatible structure, on these procedures. This method is the reason why the prefix CONSOLE has been used in the procedures names and it ensures a remarkable semplification and an incomparable flexibility. In fact the high level instructions PRINT, PRINTF, PUTS, KBHIT, SCANF, INPUT, etc. of a C or BASIC language development tools, automatically call the three entry procedures of the library and all their possibilities can be used. For example the C instruction:

printf("Product pieces=%4d",npcs);

shows on display the *Product pieces=* string followed by the value of variable *npcs*, formatted with 4 decimal digits.

The three described procedures use, and thus modify, the greater number of microcontroller registers. When the development tools doesn't accept these modifications, the user must save and restore all them before and after their call, inside the console redirection procedures.

g) After a reset or a power on the user application program must prearrange the library for next operations. These initializations are performed by two proper procedures, with as many entry points, with the following features:

LIB. INITIALIZATION: executes all the initialization operations as:
- setting of variables;
- clear buffers;
- disable buzzer, status LEDs, digital relays outputs;
- disable and deactive clock alarm;
- setup and clear display;
- setup blinking cursor in Home position;
- load user characters patterns;
- show possible power on automatic visualization;
- setup saved keyclick mode;
- enable keyboard scanning;
- enable time based functions;
- etc.

The procedure has no input nor output variables and it must be always called before to perform any other operations with library.

EEPROM INITIALIZATION: initializes the base EEPROM with default data described in DATA STORED ON EEPROM paragraph of **QTP 16Big** manual and then executes all the initialization operations listed for LIB. INITIALIZATION. The procedure has two input variables and no output variables. The input variables are passed to library with the following tecniques:

Accumulator register -> RL2 relay function equal to digital output NO OUT2
- 0 -> user output
- 1 -> alarm clock output

B register -> optional EEPROM size
- 0 -> no optional EEPROM
- 1 -> 16K Bytes optional EEPROM (**QTP 16Big-xx.LIB.EE128**)
- 2 -> 32K Bytes optional EEPROM (**QTP 16Big-xx.LIB.EE256**)
- 3 -> 64K Bytes optional EEPROM (**QTP 16Big-xx.LIB.EE512**)

Remind that the execution time of this procedures is about 20 seconds. The EEPROM INITIALIZATION procedure must be called only one time to mantain enduarance of the EEPROM that it writes: its typical use are in correspondence of the first installation or when default settings must be restored after wrong and unwanted modifications.

Normally these procedures must be called only one time at the beginning of user application program, by using the rules of the selected development tool; normally they coincides with an absolute call instructions (i.e. LCALL 3FF4H) to entry address, preceded by a possible setting of the input variables = registers.

h) Once developed the user application program that uses the library with all the features described in the previous points, it must be saved on FLASH EPROM of **QTP 16Big.LIB** together with the library, as described in FLASH EPROM PROGRAMMING paragraph.

i) At this point the **QTP 16Big.LIB** is complete and ready to be used and tested inside the real application system. The debug of the obtained application program can be done with the modalities of the used software development tool; the serial line (not used by library) is an excellent candidate for this function.

RESOURCES USED BY LIBRARY FIRMWARE

The library offers commands that allows to easily manage the numerous resorces of the board as display, contrast, keyboard, buzzer, LEDs, digital relays outputs, EEPROM and backed SRAM memories, real time clock, I2C BUS interface, etc. These commands however use an additional list of **QTP 16Big** hardware resources that are briefly described in this paragraph, together with the consequent limitations on the user aplication program side:

Code area in microcontroller FLASH EPROM: coincides with an area at the end of FLASH EPROM, already described in poits (d), (e) of previous paragraph. On this area is saved the code of library and it must not be absolutely used by application program, to avoid malfunctions of the entire system. Some software development tools (as BASCOM 8051) can be configured to autonomously advise the user when the length of generated code is higher than a prefixed limit (**APPL. SW END** >= **LIBRARY FW START**).

Data area in microcontroller internal RAM: coincides with two internal RAM areas (the first located in the direct access area and the second located in the indirect access area of microcontroller) where are saved all the flags, variables, buffers of the library. These areas are located at fixed addresses described on figure 4 and they must not be absolutely used from application program, to avoid malfunctions of the entire system.
The protection and reservation of these memories is obtained following the rules of the used software development tool and normally it is performed with compiler directives, setting of the possible project, dummy variables declaration that are located but never used, setting of user start up code, etc. The software deveopment tools normally shows windows that reports the program generation results: these allow the user to easily check the safeguard of these areas and to previously discover possible malfunctions during test phase.
The choice of library memory usage has been done carefully with the intention to left unused a portion of each type of microcontroller memories, that are: 4 bytes addressable at bit level equal to 32 user bits, 48 bytes with direct access, 63 bytes with indirect access and finally 256 bytes of external RAM. Moreover the completeness of the commands offered from firmware drastically reduces the requirements of memory and data from user application program, up to few work variables and the stack.

Stack area of microcontroller: the library have not its own stack and it uses those of application program. During configuration of the selected software development tool, the user must consider the stack size required from library, that in worst conditions can reach the 27 bytes.

TIMER0 timer counter of microcontroller: all time based process of library are managed through a periodic interrupt generated by TIMER0 of microcontroller. The user application program can not use this resource and it must redirect the microcontroller standard interrupt vector (000BH) to specific entry point (3FE8H) of library. Once initialized the firmware services this interrupt each 2.5 msec and consequently slow down the execution of user application program. The performances reduction depends upon the used commands and upon process in execution: in some circumstances the delay can reach the tens of msec, as properly indicated in the commands descriptions.

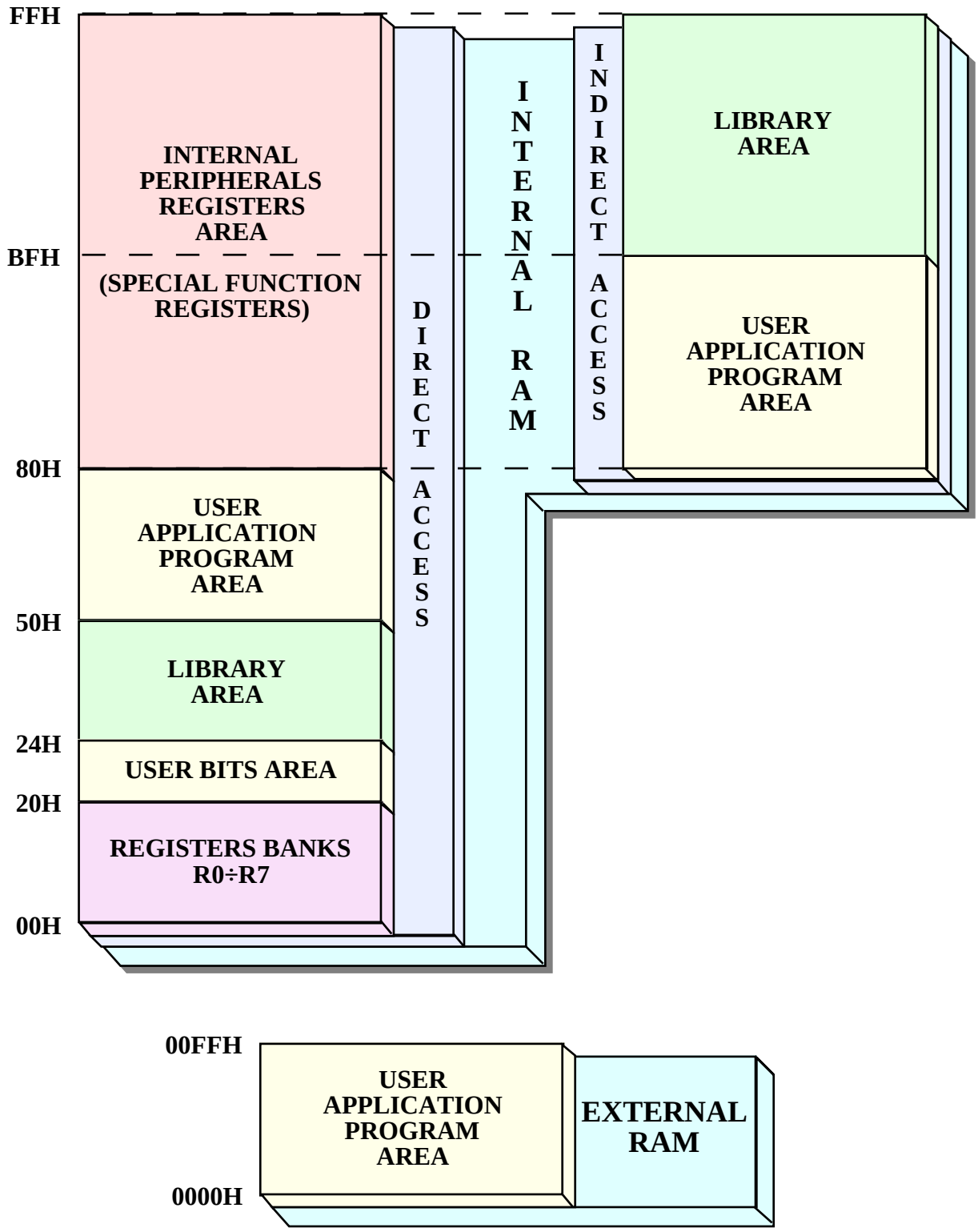


FIGURE 4: RAM USAGE WITH LIBRARY

FLASH EPROM PROGRAMMING

As described in previous paragraphs the library must be saved on the microcontroller FLASH EPROM together with user application program, as illustrated in figure 3. The FLASH management is performed through an ISP tecnique (In System Programming) that reduces the cost and the time for application development.
The ISP requires only the development PC that executes a proper management program named FLIP (FLexible In system Programming); it interacts with a Boot Loader available on microcontroller side and it is capable to read, erase, verify, program either the FLASH or the EEPROM memory. Everything happens through a simple serial connection between development PC and **QTP 16Big.LIB** normally done with RS 232 serial line or alternatively with CAN line (for this last possibility contact directly **grifo®**).
As described in **QTP 16Big** user manual, the jumper J1 selects the operating mode between the two available:

Not connected	->	RUN mode
Connected	->	DEBUG mode

In RUN mode after a power on the application program saved in FLASH is always executed, independently by external conditions, while in DEBUG mode the power on causes the execution of microcontroller Boot Loader and thus allows the ISP programmation. The J1 configuration must be frequently changed during the development of the application program; if the use of a normal female jumper link is uncomfortable, as an alternative, it could be used a connector that place the jumpers at a sufficient distance, in a suitable place. In the **QTP 16Big** manual are described the possible female connectors for J1, in the accessories appendix.
The following points describe all the operations that the user must execute to correctly program the FLASH of **QTP 16Big.LIB**:

01) Get FLIP program from ATMEL site or received **grifo®** CD and install it on development PC.
02) Select DEBUG mode, that is J1 connected.
03) Connect **QTP** to a free COMx of development PC and close all the programs that use this COMx.
04) Supply power to **QTP 16Big**.
05) Run the **FLIP** program, installed at point 01.
06) Select the device to program by clicking the first button on the top left, picking the correct name in the *Device Selection* window and press *OK*. The selection of the device to program must be done according to ordered **QTP**, or in detail T89C5115 in case of **QTP 16Big** and T89C51CC02 in case of **QTP 16Big.CAN**.

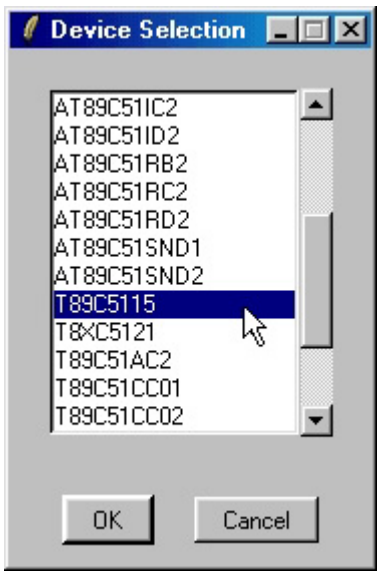


FIGURE 5: FLIP SETTINGS WINDOW (1 OF 4)

07) Select the communication mode for ISP programmation of **QTP** by cliking the second button on the top left, picking in sequence: *RS 232*, the serial *Port* of development PC, *115200 Baud* and then press *Connect*. At this point the FLIP starts communication with microcontroller Boot Loader and fill in a list of data in its main window. If communication fails and after about 20 seconds the window *Timeout error* appears, try in sequence to:check serial connection; reduce communication baud rate to 19200; repeat points 02÷07.

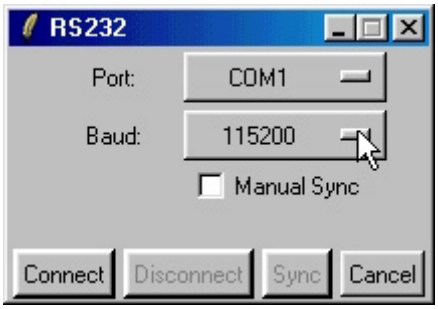


FIGURE 6: FLIP SETTINGS WINDOW (2 OF 4)

08) Open the *Buffer | Options* window, set to *NO* the option *Reset Buffer Before Loading*, select the option *Whole Buffer* in order to ensures the correct loading of the files to program and then confirm all the selections with *OK* push button.

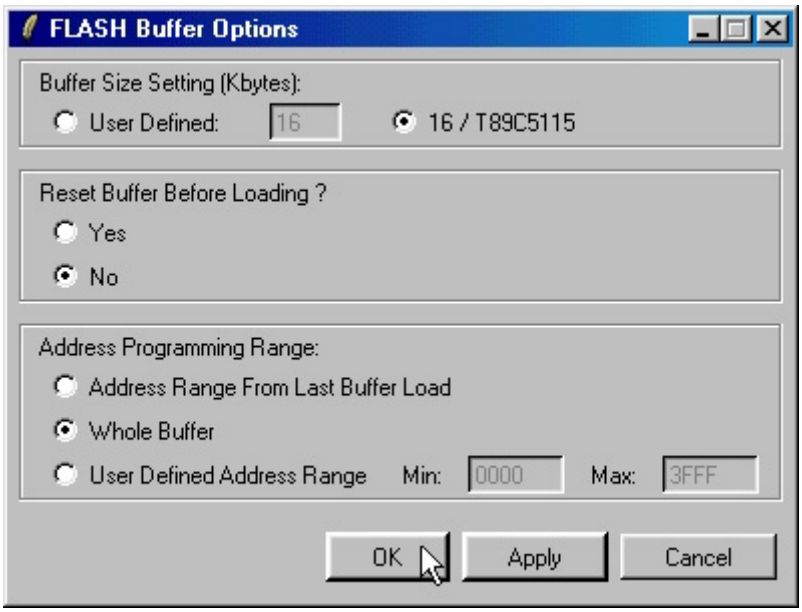


FIGURE 7: FLIP SETTINGS WINDOW (3 OF 4)

09) Load the two files to write in FLASH EPROM that are the library *QTP16Bxy.HEX* and the application program *<user file>.HEX*, by executing two times the following operations: click the nineth button on top left and select the file by using the displayed dialog box.
10) Select all the check boxes in the frame *Operations Flow* as in figure 8, to let FLIP execute in sequence the four operations: erase, blank check, program and verify.
11) At this point make sure that main windows of FLIP looks like figure 8; in details for the boxes: *Size:*, *X2*, *Device SSB* and *BSB / EB / SBV* the data must exactly match.

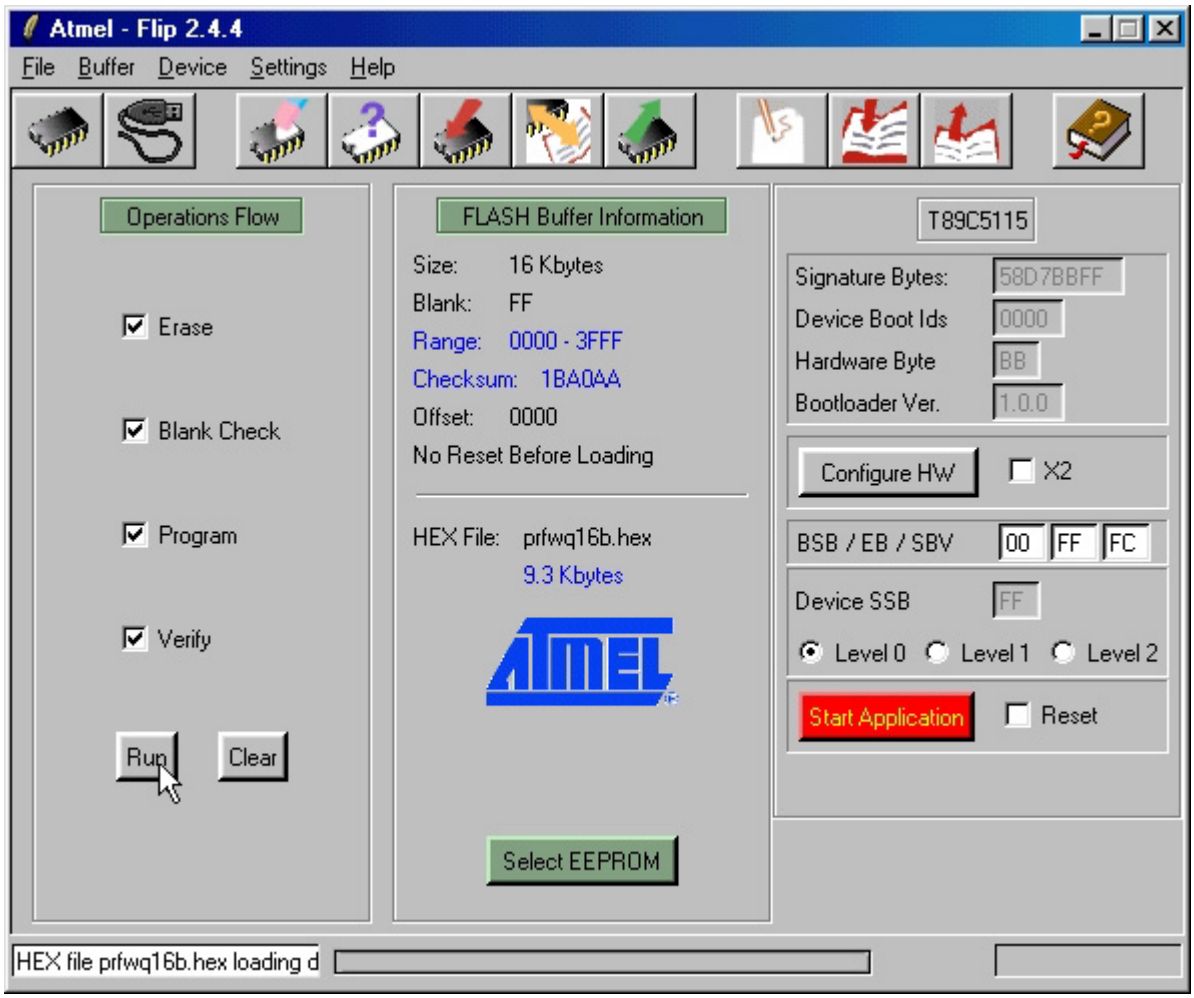


FIGURE 8: FLIP SETTINGS WINDOW (4 OF 4)

12) Press button *Run* in the main window to start the preselected ISP operations.
13) Wait the execution end of ISP operation . The status bar on the bottom reports operation progess and near text box, on the left, reports operation status; the check boxes become red and then green when the respective operation is succesfully completed. Thus wait for *Verify* check box to become green.
14) At thisp point the FLASH is programmed and **FLIP** can be closed.

The steps just described must be repeated for any test of the application program and for this reason it could be preferible to speed up their execution, by using the BatchISP modality of FLIP. This automatically performs the instruction of a proper command file, as the following one:

-device T89C5115
-hardware RS232
-port COM1
-baudrate 115200
-operation

(commands continue on right column)

memory FLASH
erase F
loadbuffer "QTP16B21.HEX"
loadbuffer "<user file>.HEX"
addrange 0x0000 0x3FFF
program
verify

The paragraphs HOW TO START... include complete examples of FLASH EPROM programming, accompanied by explanation photographs; for further information on ISP programmation and FLIP use, please consult the specific technical documentation released by ATMEL.

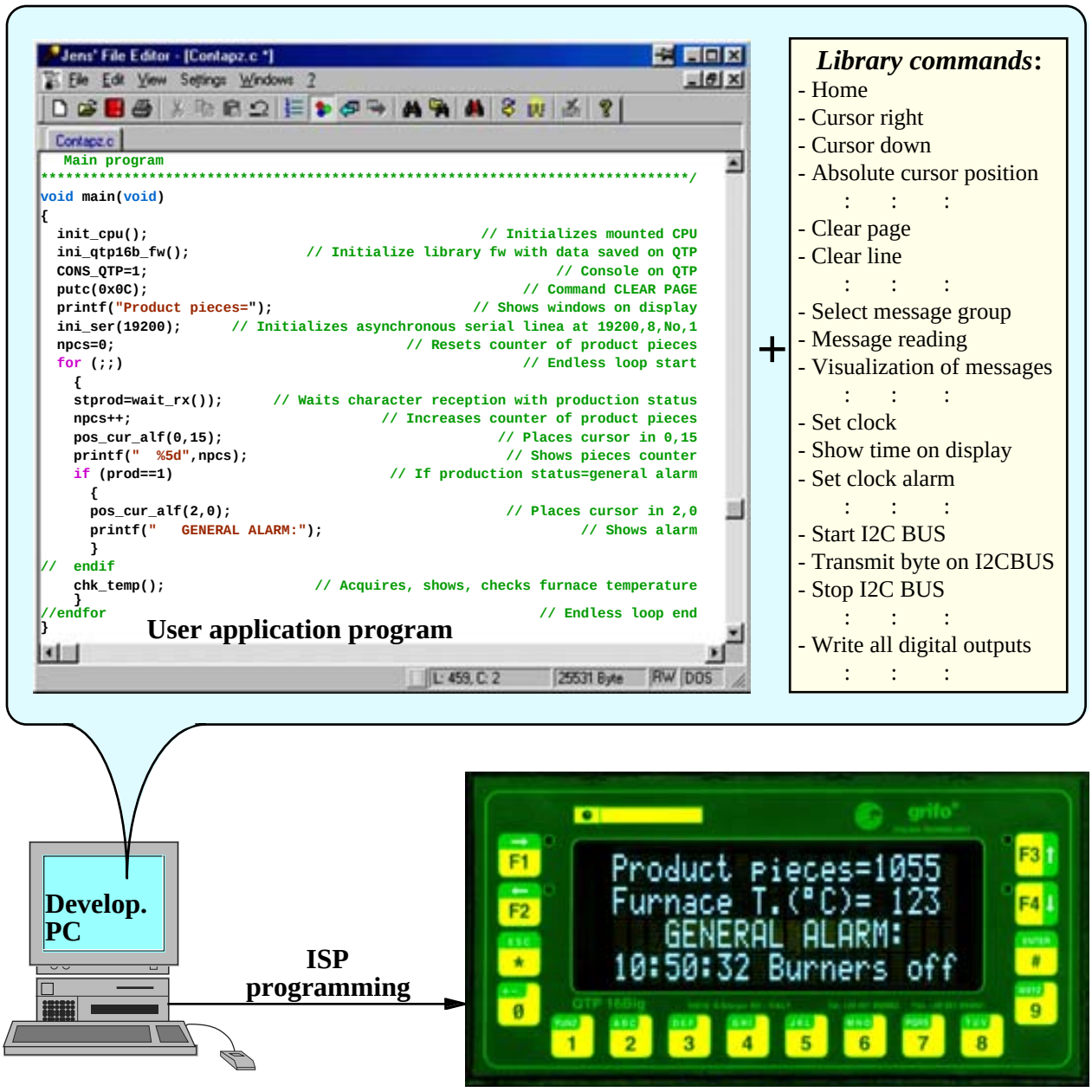


FIGURE 9: DEVELOPMENT MODE WITH LIBRARY

LOCAL SETUP

On **QTP 16Big.LIB** is not available the local setup modality, described in the user manual of the card. Many of the working parameters selectable through this modality are no more necessary because they regards the communication with the command unit, that is no more managed with library. The remaining parameters settings are defined through the EEPROM INITIALIZATION procedure, described in previous paragraphs.

COMMUNICATION MODALITY

On the contrary of normal firmware of **QTP 16Big** the library doesn't provide any communication modality with command units. Thus the asyncronous serial line on CN5 is not used by library, while the I2C BUS line is used by library only in Master mode but not in Slave mode.
So the user application program can communicate with other devices with no restriction both in point to point and network connection, with any logic, physic and electric protocol. Thanks to selected development tools the serial line can be also used as a debug channel that let the user found out the errors and reduces the time necessary to obtain a finished system.
The physic protocol can be defined by user application program, through some directives or by setting some internal registers of microcontroller. In details the lines can be programmed to operate with 8,9 bits per character; even, odd or no parity; 1 or 2 stop bits; standard or no standard baud rates up to 115200 Baud.
The logic protocol is as much free and it is always defined by user application program. It can implement standard protocols (MODBUS, XMODEM, etc.) and user propetary protocols and it can consequently communicate with all the serial devices available on the market as printers, PC, modem, PLC, terminals, other operator panels, etc.
Finally the electric protocol can be selected by hardware as described in **QTP 16Big** user manual, but with library the user application program must also manage a specific signal, named **DIR** and connected to P2.1 signal of microcontroller, as below described:

RS 422 point to point communication: both the transmit and receive drivers can be always enabled.

P2.1 = DIR = 0 -> transmit driver enabled

RS 422 network communication: the receive driver is always enabled while the transmit driver must be enabled only during data transmission.

P2.1 = DIR = 0 -> transmit driver enabled
P2.1 = DIR = 1 -> transmit driver disabled

RS 485 communication: the receive driver is always enabled while the transmit driver must be enabled only during data transmission phase, obtaining the receiving or transmitting functionality on the half duplex line.

P2.1 = DIR = 0 -> line trasmitting
P2.1 = DIR = 1 -> line receiving

All the transmitted characters are at the same time received (echo) when RS 485 communication is used; in this way the line conflicts can be immediately recognized by simply testing the echo received character after each transmission.

During and after a power on phase the P2.1 = DIR signal is forced to high level that mantains the RS 485 driver receiving and that disables the RS 422 transmit driver; this condition eliminates any conflicts on the communication line.

DEMO PROGRAMS FOR LIBRARY

In correspondence of the first purchase included in the received materials (floppy disks or CD) there are numerous demo programs that allow to test and evaluate immediately the received product. These programs are provided both in executable and source format, they are complete of all the surrounding files (headers, projects, libraries, configurations, etc.) and they are realized with all the software development tools suggested by grifo®.

As described in HOW TO START .... paragraphs the programs named *PRFWQ16B.** use all the commands of **QTP** with a simple iteraction with the user; they manage the display in all the visualization modes, the keyboard, the LEDs and buzzer, the EEPROM/s, the messages, the digital outputs, the real time clock and its alarm, some I2C BUS peripherals, etc. The demos have been divided in different programs, each one dedicated to main groups of **QTP 16Big.LIB** features, in order to speed up their use and to obtain programs sizes compatible with the free code area of the library. The user can examine the remarks and descriptions of these demos and decide himself if they are interesting.
All the demo programs can be used directly or modified or partially used, according to applications requirements, without any autorizathion, license or additional cost. Furthermore in case of unusual requirements or combinations, specific new demo programs can be obtained, or demos for different development tools, after proper agreenment with **grifo®**.

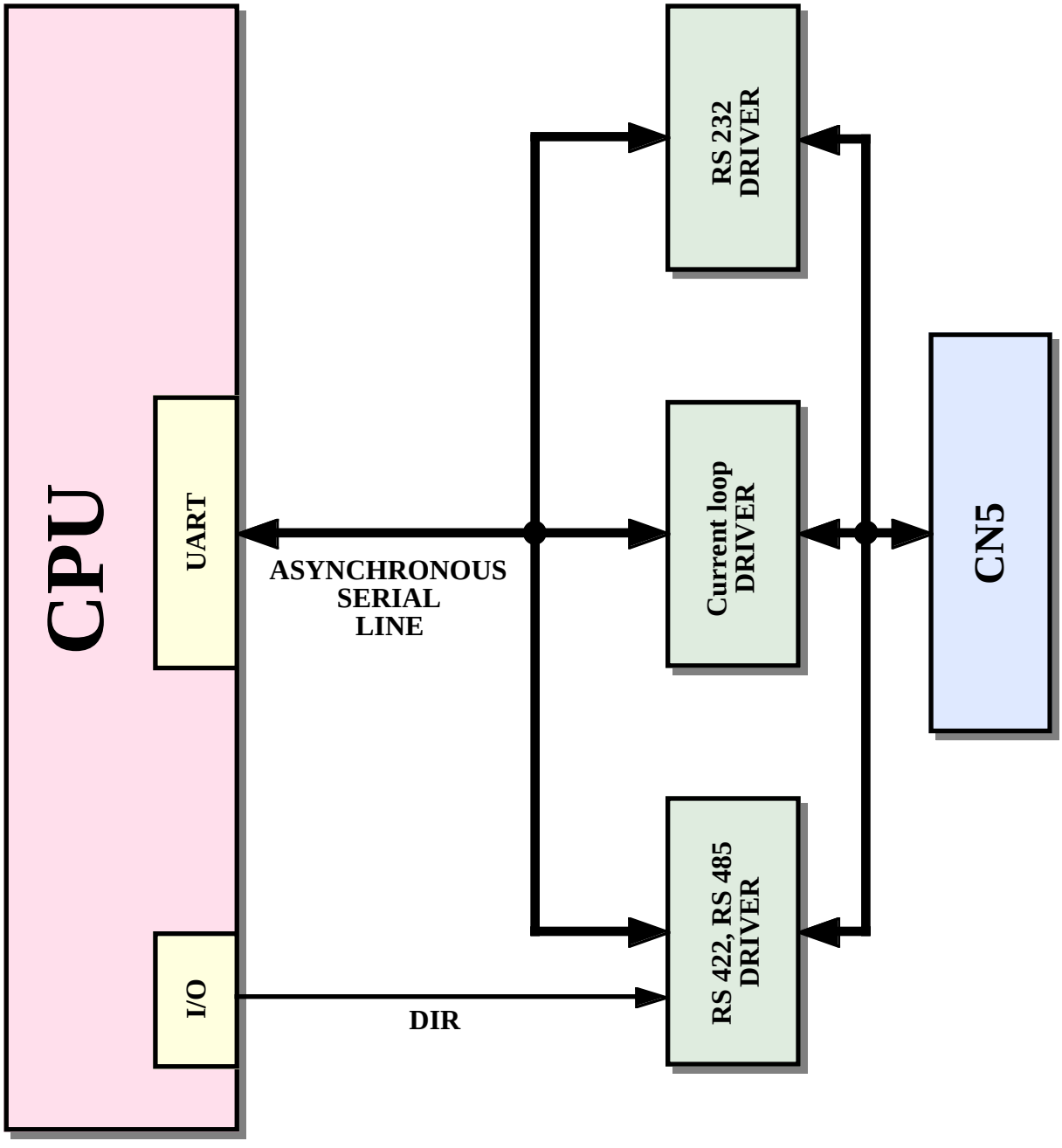


FIGURE 10: SERIAL COMMUNICATION BLOCK DIAGRAM

QTP 16BIG LIBRARY USED WITH µC/51

In this chapter it is described the integration and the use of **QTP 16Big** library with the **µC/51** development tools.
The **µC/51** is a comfortable, low cost, development tools with a colplete IDE that allows to use an editor, an ANSI C compiler, an assembler, a linker and a remote source level debugger user configurable. It includes the souces of main libraries, interesting utility programs, user documentation and severaland demo programs.
Please remind that this chapter reports only the information about the library use but not those on normal use of **µC/51**; so it is suggested first to examine the development tools documentation and then the following paragraphs.

LIBRARY INTEGRATION ON µC/51

As already described in paragraph INTEGRATION AND USE OF LIBRARY, the tecniques used to integrate the library in the development tools are the following ones:

- Some of the compiler libraries have been modified in order to redirect the console functions on library procedures. In details the modification has mantained the original management of microcontroller serial line and it adds the library management. This is obtained through a flag, named *CONS_QTP*, that selects the used console device with the following corrispondence:
  0 -> console associated to serial line
  1 -> console associated to **QTP 16Big** library
  The flag *CONS_QTP* coincides with a bit in internal RAM, located at address **28H**.
  The **µC/51** has three different console libraries with as many management modalities of the serial line (D = interrupt, P = polling, K = fast polling); all them has been modified as described, and the letter Q has been added as suffix to the original file name (see paragraph SUPPLIED FILES WITH µC/51).

- The areas of internal RAM used by library have been reserved by defining three proper segments relative to: direct access area bit addressable, direct access byte addressable and indirect access byte addressable. These segments have been referenced inside an user start up procedure that informs the compiler about the used memories.

- The interrupt service routine for microcontroller TIMER0 has been redirected.

- Two functions have been created to call the relative initialization procedures of **QTP 16Big** library. These functions have the following prototype:
  ini_qtp16b_fw(); -> calls the **LIB. INITIALIZATION** procedure;
  ini_qtp16b_ee(fnrl2,exteesize); -> calls the **EEPROM INITIALIZATION** procedure with the two passed parameters, properly saved in the rispective registers.

All the described tecniques have been grouped in a single file, named *FWQTP16B.H*, that coincides with the header file with all the declaration necessary for **QTP 16Big** library management. This file must be always included in the application program developed by the user, in fact it allows an immediate use of the same library.

SUPPLIED FILES WITH µC/51

Here follows a brief description of the necessary files to develop an user application program with the library, by using the software development tools **µC/51**. Naturally all these files are saved in the folders of the CD or disk received in correspondence of the first order of **QTP 16Big.LIB**.

\LIB\LARGE*.* -> Console library files for *large* memory model of **µC/51**. These files associate C console instructions to library procedures when internal bit addressable RAM 28H is set and viceversa, to hw serial line, when the bit is cleared. As on original **µC/51** library files there are 3 different management modes for console on hw serial line:

SER_IODQ.LIB = interrupt management with debugger;
SER_IOPQ.LIB = polling management with debugger;
SER_IOKQ.LIB = fast polling management without debugger;

These files must be manually copied on hard disk of development PC inside the folder \UC51\LIB\LARGE\ .

\LIB\SMALL*.* -> Console library files for *small* memory model of **µC/51**. These files associate C console instructions to library procedures when internal bit addressable RAM 28H is set and viceversa, to hw serial line, when the bit is cleared. As on original **µC/51** library files there are 3 different management modes for console on hw serial line:

SER_IODQ.LIB = interrupt management with debugger;
SER_IOPQ.LIB = polling management with debugger;
SER_IOKQ.LIB = fast polling management without debugger;

These files must be manually copied on hard disk of development PC inside the folder \UC51\LIB\SMALL\.

QTP16Bxy.HEX -> File with executable code of **QTP 16Big** library, with version x.y, in HEX Intel format.
It must be manually copied on the working folder of development PC.

CANARYE.H -> Header file with special function registers declaration of the CANary microcontroller used on **QTP 16Big.LIB**.
It must be manually copied on the working folder of development PC.

FWQ16B.H -> Header file with integration of **QTP 16Big** library (memory areas reservation, entry points declarations, interrupts redirections, etc.).
It must be manually copied on the working folder of development PC.

DL.BAT -> File that performs the download of the genearted code on the FLASH of **QTP 16Big.LIB** (trough the command file described in following points) and then it runs the terminal emulation program HYPERTERMINAL.
It must be manually copied on the working folder of development PC and then it must be modified by changing the pathnames where are saved the used programs.

TERM19_COM1.HT -> Configuration file for the terminal emulation program HYPERTERMINAL, that sets the physic communication protocol on the development PC; the last can thus operate as a console for the demo programs.
It must be manually copied on the working folder of development PC.

PRFWQ16B.C -> Source file of the first demo program that uses all the standard commands of **QTP 16Big** library.
It must be manually copied on the working folder of development PC.

PRFWQ16B.MAK -> Project file of the homonimous source file.
It must be manually copied on the working folder of development PC.

PRFWQ16B.HEX -> File with executable code of the homonimous sorce file.
It must be manually copied on the working folder of development PC.

PRFWQ16B.WSP -> Workspace file that arranges the JFE editor to be used as a complete IDE that allows the user to modify, compile and program the homonimous demo program.
It must be manually copied on the working folder of development PC and then it must be modified by changing the pathnames where are saved the used folders.

PRFWQ16B.CMD -> Command file for ISP programming of FLASH on **QTP 16Big.LIB**: it performs all the operations that save the demo program and the library, by using the batch version of FLIP.
It must be manually copied on the working folder of development PC and modified according with used microcontroller, if necessary.

PRFWQ16B_OPT.* -> Source file of the second demo program that uses all the commands of **QTP 16Big** library, dedicated to options and external accessories.
They must be manually copied on the working folder of development PC, as described for the similar files *PRFWQ16B.**.

The indication "working folder of development PC" identifies the folder created on development PC where the customer wants to develop his application program. Summarizing, in this folder must be copied all the supplied files, except the library ones. Moreover while time goes by, new programs and/or files, not described in previous list, can be developed: please examine them through their initial descriptions.
Once all the described files has been copied in the right destination folders please ensure that the attribute *Read only* is not active and if it is, disable it by hand. Normally this happens only when files are copied from CD rom, under Windows operating system previous than Windows XP.

HOW TO START WITH µC/51

In this paragraph are listed the operations that must be performed to start using the **QTP 16Big.LIB** in a pratical and fast way, with **µC/51** environment, solving the typical beginners problems. The paragraph contains interesting information even for the users that already know the product, in fact there is the description of a fast and complete functional test.

A) *Serial connection between **QTP 16Big.LIB** and development PC:*

A1) Perform the serial connection described in figure 2 or on the other hand connect the two communication signals (TX RS232, RX RS232) and the reference ground signal (GND), to free COMx serial port of development PC. It can be easily discovered that this connection cable is reversed and it can be conveniently ordered to **grifo®** with the code CCR 9+9R.

A2) Found the HYPERTERMINAL communication program on the development PC, that normally is located on Windows menu: *Start* | *Program* | *Accessories* | *Communication*, execute it and through the HYPERTERMINAL properties windows, setup the communication parameters to:

Connect	directly to COM x (those used at point A1)
Bit rate	19200
Data Bits	8
Parity	No
Stop Bit	1
Flow control	None

Then wait the presentation of communication window.

A3) Select RUN mode, that is jumper J1 not connected.

A4) Supply power voltage on CN4 and check that: buzzer is immediately disabled, on development PC appears the demo program presentation message and after 5 seconds on display is shown the message *Demo program ..... for QTP 16Big.LIB*, with a blinking block cursor in the right down corner. Each **QTP 16Big.LIB**, received for the first time, is delivered with its demo program and library already saved in FLASH and arranged to allow automatic start at power on: if the demo program presentation screen doesn't appear please check again the serial connection and the J1 right configurations.

A5) Follow the instructions of the demo and check execution of all the commands of library: the user must interact with demo either through serial console on development PC and the resources of same **QTP**.

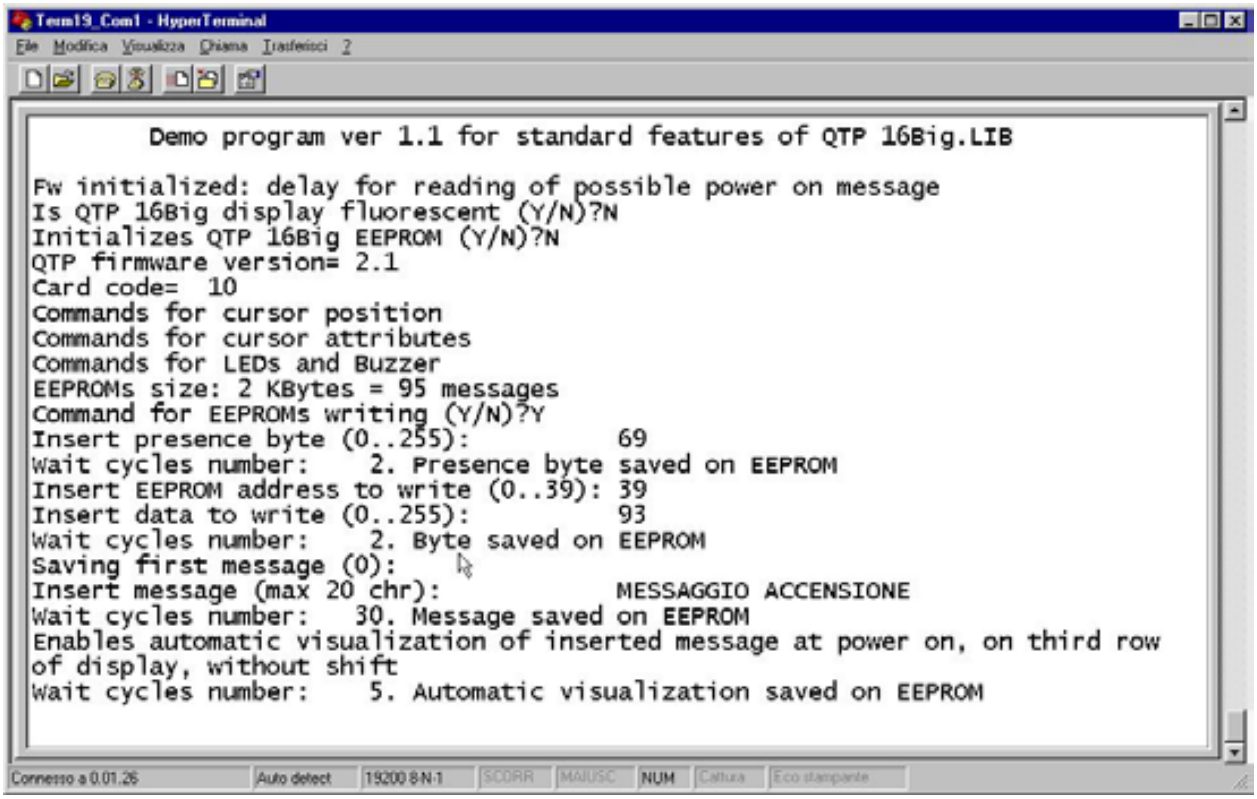


FIGURE 11: EXECUTION OF µC/51 DEMO PROGRAM

A6) When demo execution is terminated turn off **QTP** power supply.

A7) Exit from HYPERTERMINAL program on development PC.

B) *Reprogram of FLASH with demo program:*

B1) On the received **grifo®** disk find and then install the utility program FLIP on a comfortable folder of development PC hard disk. FLIP manages the ISP programming of FLASH EPROM on **QTP 16Big.LIB**, as described in FLASH EPROM PROGRAMMING paragraph.

B2) Install on the hard disk of the development PC the **µC/51** software development tools and performs its registration, to bypass the limits of demo version.

B3) Create a new folder on hard disk of development PC that will be the place where the user saves all his files or, in other words, the working folder described in previous paragraphs.

B4) Copy the library files described in SUPPLIED FILES WITH µC/51 paragraph, in the library folders created at point B2, by removing the possible *Read only* attribute.

B5) Copy the remaining files described in SUPPLIED FILES WITH µC/51 paragraph, in the working folder created at point B3, by removing the possible *Read only* attribute.

B6) Modify the file *DL.BAT* through a simple ASCII text editor (i.e. *NOTEPAD* of Windows), by defining the right pathnames where programs have been saved. In detail these pathnames are those where there is FLIP (installed at point B1) and HYPERTERMINAL (installed by Windows operating system). This step execution is simplified by taking the pathnames from the property windows of the same programs and, as it happens in all batch files, they must be inserted with MS-DOS standard format. This format, when file names are longer than 8 characters, mantains the first 6, adds the ~ character and adds a progressive number that starts from 1 (for example *\Programs\Windows XP* becomes *\Progra~1\Window~1*).

B7) Modify the *PRFWQ16B.WSP* through a simple ASCII text editor (i.e. *NOTEPAD* of Windows), by defining the right pathname where the **µC/51** has been saved. In detail the modifications regard the pathname of the command line compiler *UMAKE.EXE* installed at point B2, that can be easily obtained by property windows of the same program. As the *PRFWQ16B.WSP* file is quite large we suggest to find out all the occourences of *UMAKE.EXE* and then modify all the pathnames.

B8) Verify that inside *PRFWQ16B.CMD* file it is selected the used microcontroller and modify it when necessary through a simple ASCII text editor.

B9) Run the *JFE editor* installed at point B2, through the *Start* menu of Windows.

B10) Open the workspace file *PRFWQ16B.WSP* by using theproper command *File |Open Workspace* and by selecting the working folder created at point B3.

B11) Select DEBUG mode, that is J1 connected.

B12) Supply power to **QTP 16Big.LIB** and check that buzzer remains active.

B13) Press *DL.BAT* on the buttons row of JFE. With this pressure the batch version of FLIP is executed and it performs the operations listed in *PRFWQ16B.CMD* command file, obtaining the FLASH programming.

B14) Wait executions of all the programming steps and verify that no errors occour; at this point press a key to continue.

FIGURE 12: FLASH EPROM PROGRAMMING WITH µC/51

B15) Wait execution of terminal emulation program HYPERTERMINAL.

B16) Select RUN mode, that is J1 not connected.

B17) Turn off and on power supply of **QTP 16Big.LIB**.

B18) At this point the demo program just saved in FLASH, must start and proceed as already happened at point A.
B19) Exit from HYPERTERMINAL and return to JFE.

C) *Creation of executable code of demo program:*
C1) Compile the source file of demo program with the simple pressure of *RE-MAKE* on the buttons row of JFE, and verify that no errors happens. The file *PRFWQ16B.HEX* must be obtained equal to those available on **grifo®** disk and already used at points B.

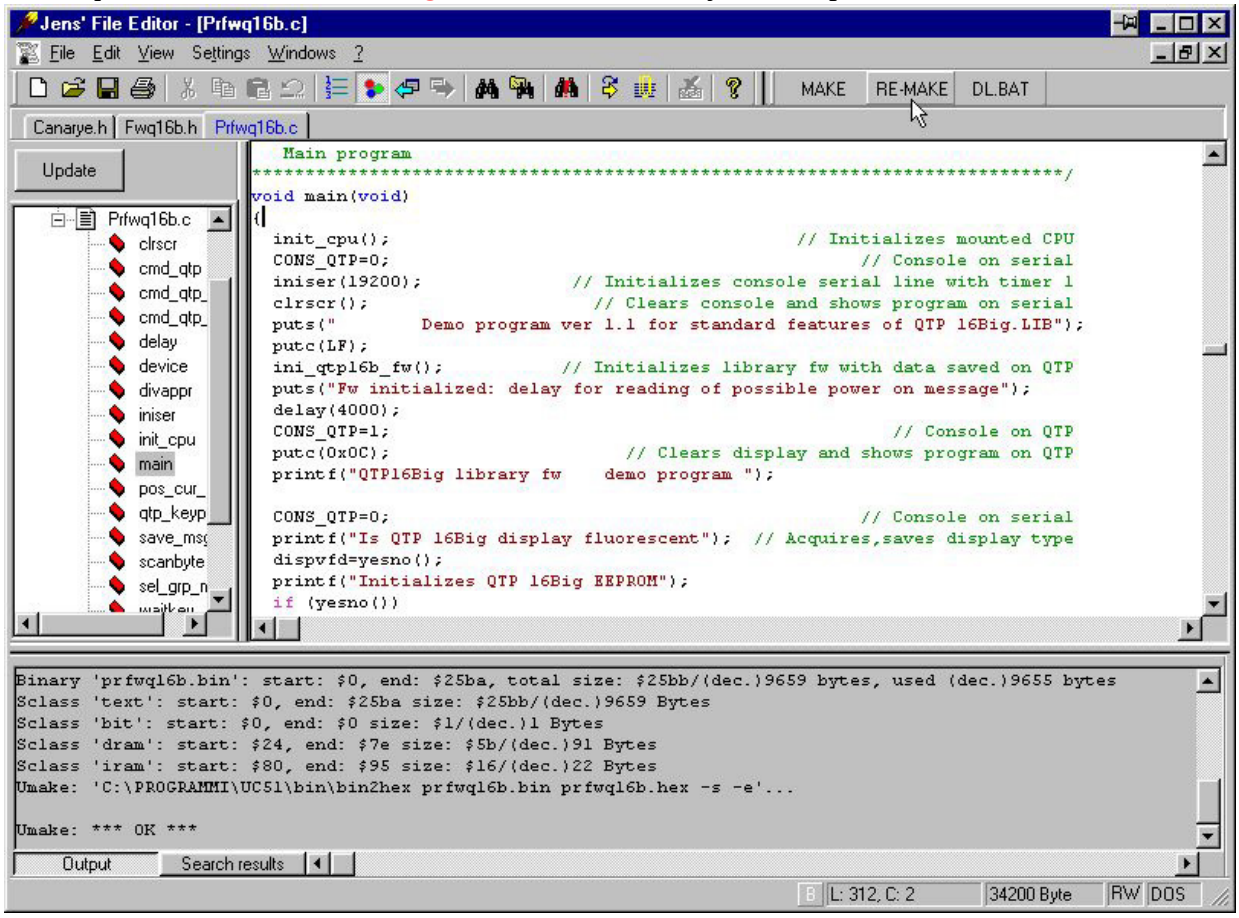


FIGURE 13: PROGRAM COMPILE WITH μC/51

C2) Reprogram the FLASH with the code of the demo program just compiled and retest it , by executing again the points B11÷B19.

When during execution of the steps above described a problem or a malfunction is found, we suggest to read and repeat again all the steps carefully and if malfunction persists please contact directly **grifo®** technicians.
Instead when execution of all the steps is right, the user has realized his first application program that coincides with demo of **QTP 16Big.LIB**. At this point it is possible to modify the source of the demo/s program/s according to application requirements and test the obtained program with the steps above listed (from B11 to C2) in cyclic mode, until the developed application program is completely well running. When this focus is reached the developmnet PC can be removed, by obtaining a self running product, as below described:

D) *Final preparation of application:*
D1) Select the RUN mode (jumper J1 not connected) and disconnect the development PC if it is not required by the same application.

QTP 16BIG LIBRARY USED WITH BASCOM 8051

In this chapter it is described the integration and the use of **QTP 16Big** library with the **BASCOM 8051** development tools.
The **BASCOM 8051** is a low price development tools based on a comfortable and complete IDE that provides editor, BASIC compiler, a simulator, an integrated or external programmer and a full featured terminal emulator. It includes many memory models, data types and specific instructions for industrial automation, interesting utility programs, on line help documentation and a rich list of examples.
Please remind that this chapter reports only the information about the library use but not those on normal use of **BASCOM 8051**; so it is suggested first to examine the development tools documentation and then the following paragraphs.

LIBRARY INTEGRATION ON BASCOM 8051

As already described in paragraph INTEGRATION AND USE OF LIBRARY, the tecniques used to integrate the library in the development tools are the following ones:

- The compiler console instrunctions have been redirected on two dedicated procedures that manages the input and output of a single character. In detail BASCOM 8051 provides these redirections through the directive *$serialinput* and *$serialoutput* that specifies the relative procedure to execute. In details the redirection has mantained the original management of microcontroller serial line and it adds the library management. This is obtained through a flag, named *Cons_qtp*, that selects the used console device with the following corrispondence:
  0 -> console associated to serial line
  1 -> console associated to **QTP 16Big** library
  The flag *Cons_qtp* coincides with a bit in internal RAM, declared in the program and thus directly located by compiler.
  The redirection procedures are written in assembly language in order to: save all the microcontroller registers, call the entry point addresses of **QTP 16Big** library and easily manage the input and output variables saved in *Accumulator*.

- As the **BASCOM 8051** doesn't provide the redirection of console status function (*Inkey*) it has been developed a proper procedure that performs this verification, named *Consolesta*. Naturally the procedure acts on both the console devices (hw serial line and library) according with usual *Cons_qtp* flag setting and it returns both the status of character available and the possible received character.

- The areas of internal RAM used by library have been reserved by using the proper directive *$iramstart* of compiler, that is set at the start address of USER APPLICATION PROGRAM AREA, described on figure 4. Moreover in the window *Option* | *Compiler* | *Misc* of **BASCOM 8051** the *Byte End(Hex)* value is set to a value obtained by the USER APPLICATION PROGRAM AREA end address decreased of the stack area; for example by providing a stack long 40 (=28H) Bytes the value will be BF - 28 = 97.

- The interrupt service routine for microcontroller TIMER0 thas been redirected through the proper directives and high level instructions of **BASCOM 8051** (*On Timer0.....*).

- Two functions have been created to call the relative initialization procedures of **QTP 16Big** library. These functions have the following prototype:

Ini_qtp16b_fw()	->	calls the **LIB. INITIALIZATION** procedure;
Ini_qtp16b_ee(fnrl2,exteesize)	->	calls the **EEPROM INITIALIZATION** procedure with the two passed parameters, properly saved in the rispective registers.

All the described tecniques have been grouped in a single file that contains also the source of the user application program. From pratical point of view the library integration coincides with some rows in the source, properly grouped and identified; this rows must be always copied in the application programs developed by the user, in fact they allow an immediate use of the same library.

SUPPLIED FILES WITH BASCOM 8051

Here follows a brief description of the necessary files to develop an user application program with the library, by using the software development tools **BASCOM 8051**. Naturally all these files are saved in the folders of the CD or disk received in correspondence of the first order of **QTP 16Big.LIB**.

89C5115.DAT -> File with special function registers declaration and other features of homonimous microcontroller used on **QTP 16Big.LIB**.
It must be manually copied on the **BASCOM 8051** installation folder of development PC (*MCS Electronics\BASCOM8051*) .

8951CC02.DAT -> File with special function registers declaration and other features of homonimous microcontroller used on **QTP 16Big.CAN.LIB**.
It must be manually copied on the **BASCOM 8051** installation folder of development PC (*MCS Electronics\BASCOM8051*) .

QTP16Bxy.HEX -> File with executable code of **QTP 16Big** library, with version x.y, in HEX Intel format.
It must be manually copied on the working folder of development PC.

PRFWQ16B.BAS -> Source file of the first demo program that uses all the standard commands of **QTP 16Big** library.
It must be manually copied on the working folder of development PC.

PRFWQ16B.HEX -> File with executable code of the homonimous sorce file.
It must be manually copied on the working folder of development PC.

PRFWQ16B.CMD -> Command file for ISP programming of FLASH on **QTP 16Big.LIB**: it performs all the operations that save the demo program and the library, by using the batch version of FLIP.
It must be manually copied on the working folder of development PC and modified according with used microcontroller, if necessary.

PRFWQ16B_OPT.* -> Source file of the second demo program that uses all the commands of **QTP 16Big** library, dedicated to options and external accessories.
They must be manually copied on the working folder of development PC, as described for the similar files *PRFWQ16B.**.

The indication "working folder of development PC" identifies the folder created on development PC where the customer wants to develop his application program. Summarizing, in this folder must be copied all the supplied files, except the declaration ones, with .DAT extension. Moreover while time goes by, new programs and/or files, not described in previous list, can be developed: please examine them through their initial descriptions.
Once all the described files has been copied in the right destination folders please ensure that the attribute *Read only* is not active and if it is, disable it by hand. Normally this happens only when files are copied from CD rom, under Windows operating system previous than Windows XP.

HOW TO START WITH BASCOM 8051

In this paragraph are listed the operations that must be performed to start using the **QTP 16Big.LIB** in a pratical and fast way, with **BASCOM 8051** environment, by solving the typical beginners problems. The paragraph contains interesting information even for the users that already know the product, in fact there is the description of a fast and complete functional test.

A) *Serial connection between **QTP 16Big.LIB** and development PC:*

A1) Perform the serial connection described in figure 2 or on the other hand connect the two communication signals (TX RS232, RX RS232) and the reference ground signal (GND), to free COMx serial port of development PC. It can be easily discovered that this connection cable is reversed and it can be conveniently ordered to grifo® with the code CCR 9+9R.

A2) Install on the hard disk of the development PC the **BASCOM 8051** software development tools, performs its registration and update the compiler to last version if it is higher than the installed one.

A3) Create a new folder on hard disk of development PC that will be the place where the user saves all his files or, in other words, the working folder described in previous paragraphs.

A4) Copy the declaration files described in SUPPLIED FILES WITH BASCOM 8051 paragraph, in the installation folder selected at point A2, by removing the possible *Read only* attribute.

A5) Copy the remaining files described in SUPPLIED FILES WITH BASCOM 8051 paragraph, in the working folder created at point A3, by removing the possible *Read only* attribute.

A6) Run the ***BASCOM*** installed at point A2, through the *Start* menu of Windows.

A7) Open the source file *PRFWQ16B.BAS* by using proper command *File |Open* and by selecting the working folder created at point A3.

A8) Open the configuration window of **BASCOM 8051** terminal emulator, by selecting the command *Options | Communication*, and then select the serial line (*COM port*) connected at point A1, define the communication physic protocol described in the following figure and finally confirm with *Ok* button.

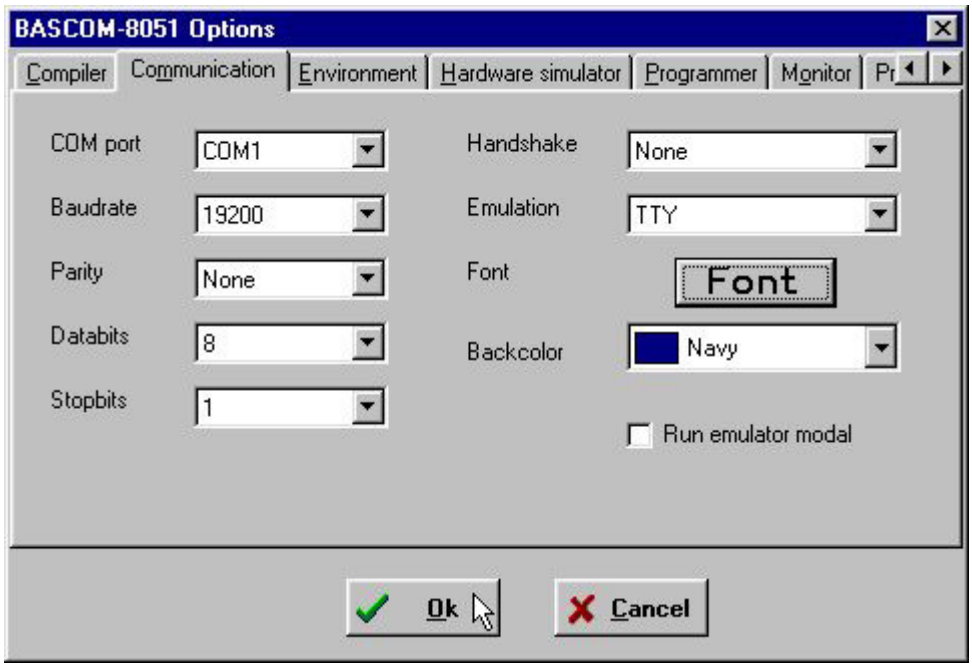


FIGURE 14: SERIAL CONFIGURATION WITH BASCOM 8051

A9) Open the terminal emulation window through the command *Tools | Terminal emulator* and check that it is correctly displayed on development PC.

A10) Select RUN mode, that is jumper J1 not connected.

A11) Supply power voltage on CN4 and check that: buzzer is immediately disabled, on *BASCOM-8051 Terminal emulator* window of development PC appears the demo program presentation message and after 5 seconds on display is shown the message *Demo program ..... for library of QTP 16Big in BASCOM 8051*, with a blinking block cursor in the right down corner. Each **QTP 16Big.LIB**, received for the first time, is delivered with its demo program and library already saved in FLASH and arranged to allow automatic start at power on: if the demo program presentation screen doesn't appear please check again the serial connection and the J1 right configurations.

A12) Follow the instructions of the demo and check execution of all the commands of library: the user must interact with demo either through serial console on development PC and the resources of same **QTP**.

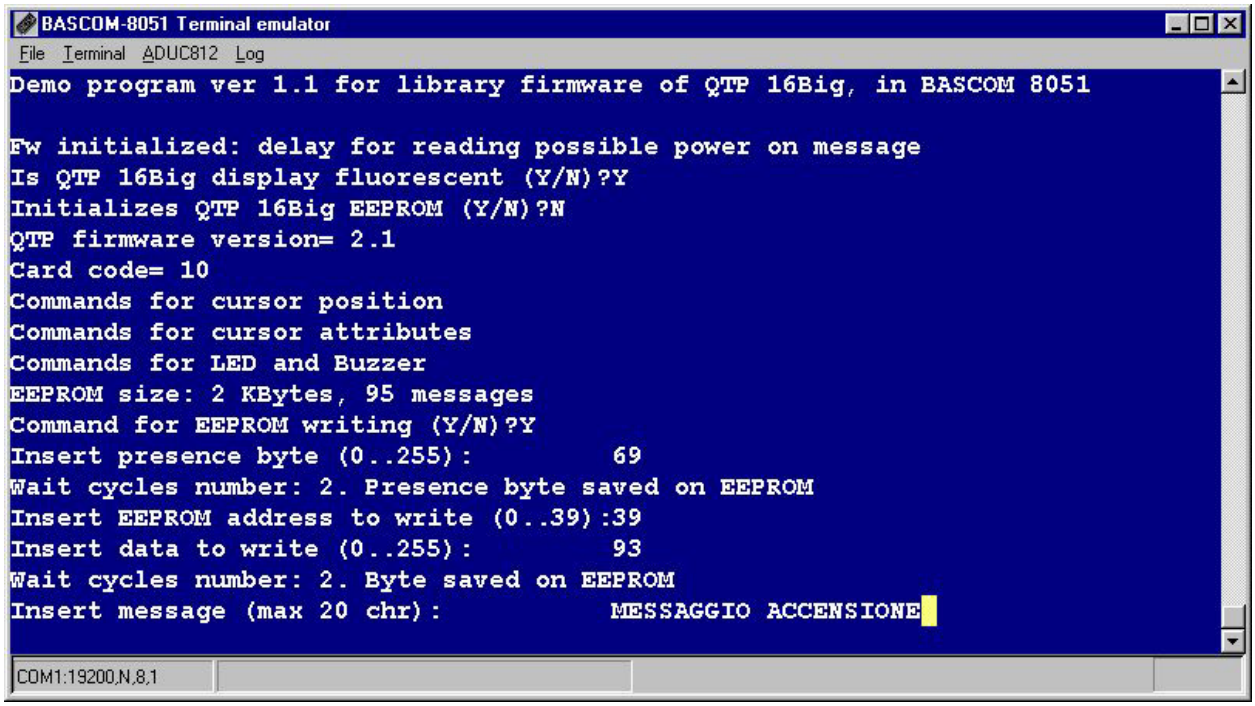


FIGURE 15: EXECUTION OF BASCOM 8051 DEMO PROGRAM

A13) When demo execution is terminated turn off **QTP** power supply.

A14) Close the terminal emulation window through the command *File | Exit*, or the dedicated button *X* in the right top corner of the same window.

B) *Reprogram of FLASH with demo program:*

B1) On the received **grifo®** disk find and then install the utility program FLIP on a comfortable folder of development PC hard disk. FLIP manages the ISP programming of FLASH EPROM on **QTP 16Big.LIB**, as described in FLASH EPROM PROGRAMMING paragraph.

B2) Verify that inside *PRFWQ16B.CMD* file it is selected the used microcontroller and modify it when necessary, through a simple ASCII text editor.

B3) Open the configuration window of **BASCOM 8051** programmer, by selecting the command *Option | Programmer*, and then define the settings described in the following figure and finally confirm with *Ok* button.
It is important underline that the setting for *Program* field coincide with the selection of *BATCHISP.EXE* program of FLIP environment, installed at point B1.

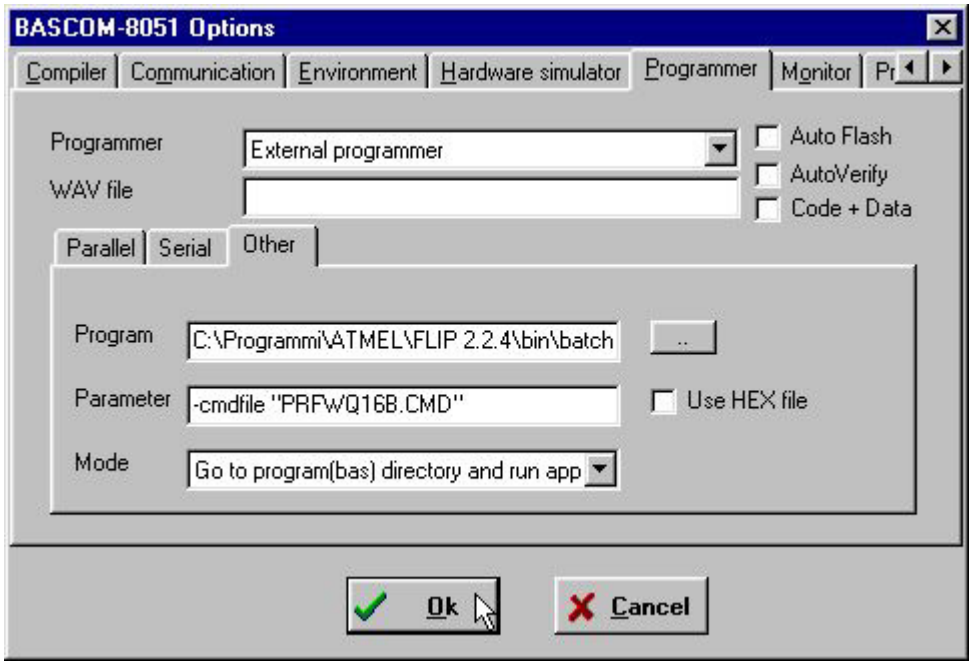


FIGURE 16: PROGRAMMER CONFIGURATION WITH BASCOM 8051

B4) Select DEBUG mode, that is J1 connected.

B5) Supply power to **QTP 16Big.LIB** and check that buzzer remains active.

B6) Press *the* hot keys *F4* or select the command *Program* | *Send to chip*. With this action the batch version of FLIP is executed and it performs the operations listed in *PRFWQ16B.CMD* command file, obtaining the FLASH programming.

B7) Wait executions of all the programming steps and verify that no errors occour; at this point press a key to continue.

FIGURE 17: FLASH EPROM PROGRAMMING WITH BASCOM 8051

B8) Reopen the terminal emulation window through the command *Tools* | *Terminal* emulator or the hot keys *Ctrl+T*.

B9) Select RUN mode, that is J1 not connected.
B10) Turn off and on power supply of **QTP 16Big.LIB**.
B11) At this point the demo program just saved in FLASH, must start and proceed as already happened at point A.
B12) Close the terminal emulation window and so return to IDE of **BASCOM 8051**.

C) *Creation of executable code of demo program:*
C1) Open the configuration window of **BASCOM 8051** compiler, by selecting the command *Option | Compiler | Misc*, then define the settings described in the following figure and finally confirm with *Ok* button.
The setting for *Register file* field must match the used microcontroller and it can be seleted only when the declaration files have been correctly copied, as described at point A4.

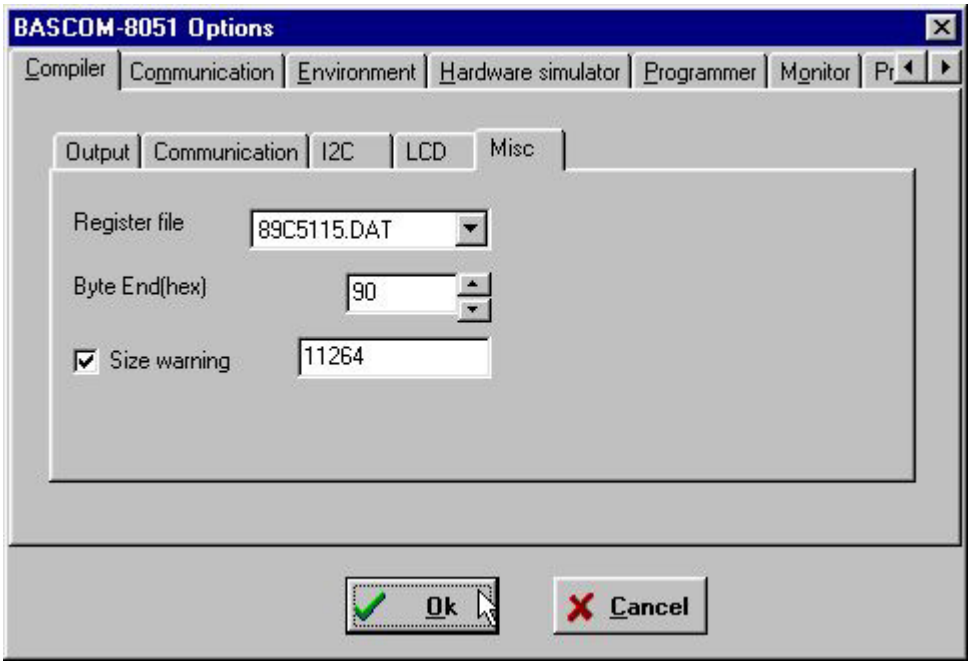


FIGURE 18: COMPILER CONFIGURATION WITH BASCOM 8051

C2) Compile the source file of demo program with the simple pressure of the hot key *F7*, or by selecting the command *Program | Compile*, and verify that no errors happens. The file *PRFWQ16B.HEX* must be obtained equal to those available on **grifo®** disk and already used at points B.
The compilation time change according to used development PC; anyway the user must wait that both the passes are completed, through a specific status window displayed during compilation, and then check that the bottom side of IDE doesn't show errors. In other words at the end of compilation it must be displayed a window similar to those reported in following figure.

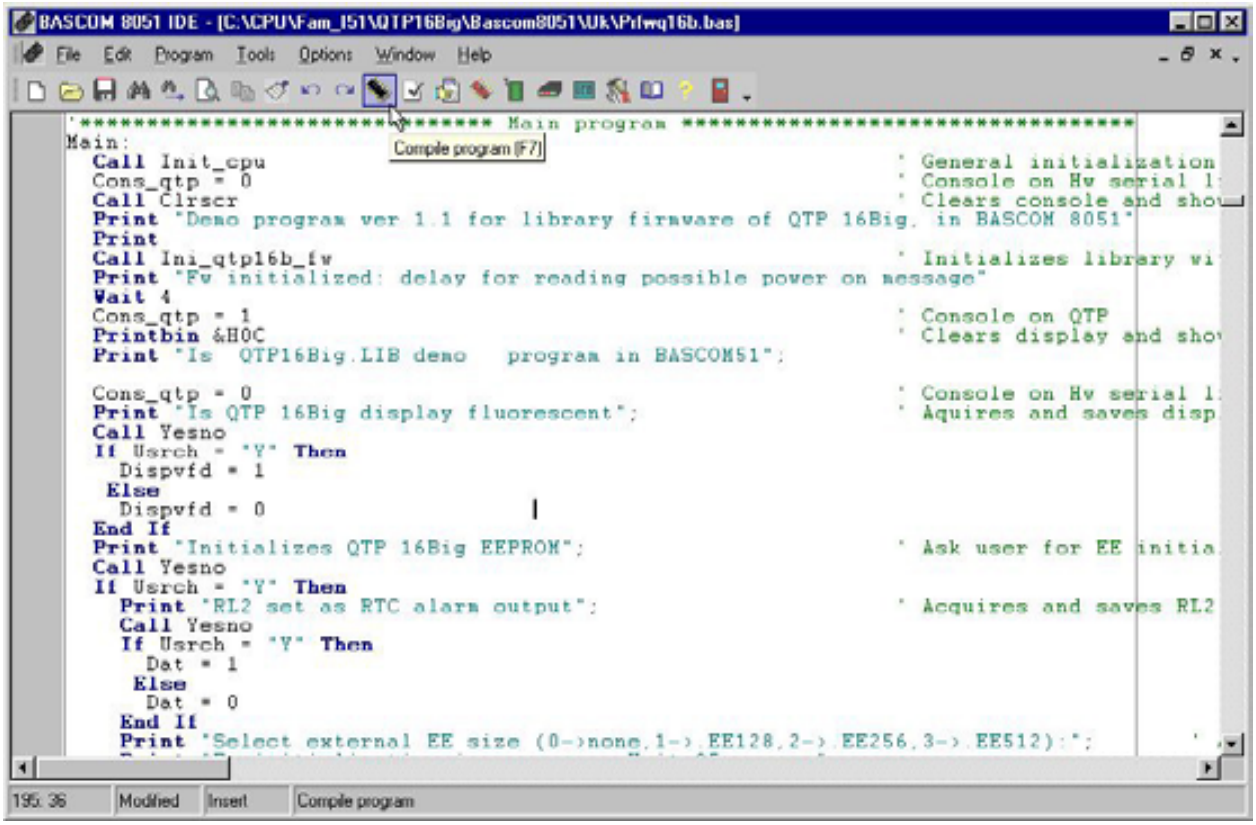


FIGURE 19: PROGRAM COMPILE WITH BASCOM 8051

C3) Reprogram the FLASH with the code of the demo program just compiled and retest it, by executing again the points B4÷B12.

When during execution of the steps above described a problem or a malfunction is found, we suggest to read and repeat again all the steps carefully and if malfunction persists please contact directly **grifo®** technicians.
Instead when execution of all the steps is right, the user has realized his first application program that coincides with demo of **QTP 16Big.LIB**. At this point it is possible to modify the source of the demo/s program/s according to application requirements and test the obtained program with the steps above listed (from B4 to C3) in cyclic mode, until the developed application program is completely well running. When this focus is reached the developmnet PC can be removed, by obtaining a self running product, as below described:

D) *Final preparation of application:*
D1) Select the RUN mode (jumper J1 not connected) and disconnect the development PC if it is not required by the same application.

Please remind that the configuration steps described at points A8, B3 and C1 must be manually executed only the first time, in fact **BASCOM 8051** saves all the configurations of its IDE during the exit and it automatically reloads them during next execution.

BIBLIOGRAPHY

In this chapter there is a complete list of technical data books and sheets, where the user can find all the necessary documentations on the components mounted on **QTP 16Big.LIB** board.

ATMEL manual:	*Microcontroller - AT89 series*
CTC data sheets:	*LCD module specification*
F.T. data sheet:	*Smal Signal Relays*
HEWLETT PACKARD manual:	*Optoelectronics Designer's Catalog*
MAXIM manual:	*New Releases Data Book - Volume IV*
NORITAKE ITRON data sheets:	*Dot VFD Modules*
PHILIPS manual:	*Application notes and development tools for 80C51 microcontrollers*
PHILIPS manual:	*80C51 - Based 8-Bit Microcontrollers*
PHILIPS manual:	*I^2C-bus compatible ICs*
S.E. data sheets:	*SI series - Switching power supply*
SGS-THOMSON manual:	*Small signal transistor - Data Book*
TEXAS INSTRUMENTS manual:	*The TTL Data Book - SN54/74 Families*
TEXAS INSTRUMENTS manual:	*RS-422 and RS-485 Interface Circuits*

The described manuals can be requested directly to manufacturer or local dealers. Alternatively this information and/or their upgrades can be found in specific internet web pages, of the listed companies.

APPENDIX A: ON BOARD DEVICES DESCRIPTION

grifo® provides a completely free technical documentation service to make available the data sheets of on board components, through its web site. This chapter shows only the first page of the data sheets, but the user can dowload the complete documents from the "Technical documentation Service" link on the home page.

T89C5115 OR T89C51CC02 MICROCONTROLLER

ATMEL WIRELESS & µC

T89C51CC02

8-bit MCU with CAN controller and Flash

1. Description

Part of the CANary™ family of microcontrollers dedicated to CAN network applications, the T89C51CC02 is a low pin count 8-bit Flash microcontroller.

While remaining fully compatible with the 80C51 it offers a superset of this standard microcontroller. In X2 mode a maximum external clock rate of 20 MHz reaches a 300 ns cycle time.

Besides the full CAN controller T89C51CC02 provides 16 Kbytes of Flash memory including In-system Programming (ISP), 2-Kbyte Boot Flash Memory, 2-Kbyte EEPROM and 512 bytes RAM.

Special attention is payed to the reduction of the electro-magnetic emission of T89C51CC02.

CANary™

2. Features

- 80C51 core architecture:
  - 256 bytes of on-chip RAM
  - 256 bytes of on-chip ERAM
  - 16 Kbytes of on-chip Flash memory
    Read/Write cycle : 10k
    Data Retention 10 years at 85°C
  - 2 Kbytes of on-chip Flash for Bootloader
  - 2 Kbytes of on-chip EEPROM
    Read/Write cycle : 100k
  - 14-source 4-level interrupt
  - Three 16-bit timer/counter
  - Full duplex UART compatible 80C51
  - maximum crystal frequency 40 MHz. In X2 mode, 20 MHz (CPU core, 40 MHz)
  - three or four ports: 16 or 20 digital I/O lines
  - two-channel 16-bit PCA with:
    - PWM (8-bit)
    - High-speed output
    - Timer and edge capture
  - Double Data Pointer
  - 21-bit watchdog timer (including 7 programmable bits)
- A 10-bit resolution analog to digital converter (ADC) with 8 multiplexed inputs
  - Separate power supply for analog
- Full CAN controller:
  - Fully compliant with CAN standard rev 2.0 A and 2.0 B
  - Optimized structure for communication management (via SFR)
  - 4 independent message objects:
    - Each message object programmable on transmission or reception
    - individual tag and mask filters up to 29-bit identifier/message object
    - 8-byte cyclic data register (FIFO)/message object
    - 16-bit status & control register/message object
    - 16-bit Time-Stamping register/message object
    - CAN specification 2.0 part A or 2.0 part B programmable message objects
    - Access to message object control and data register via SFR
    - Programmable reception buffer lenght up to 4 message objects
    - Priority management of reception of hits on several message objects at the same time (Basic CAN Feature)
    - Priority management for transmission
    - message object overrun interrupt
  - Supports
    - Time Triggered Communication.
    - Autobaud and Listening mode
    - Automatic reply mode programmable
  - 1 Mbit/s maximum transfer rate at 8MHz* Crystal frequency in X2 mode.
  - Readable error counters
  - Programmable link to on-chip Timer for Time Stamping and Network synchronization
  - Independent baud rate prescaler
  - Data, Remote, Error and overload frame handling
- Power saving modes:
  - Idle mode
  - Power down mode
- Power supply: 5V +/- 10% ,3V +/- 10%
- Temperature range: Industrial (-40° to +85°C)
- Packages: PLCC28, SOIC28, (TSSOP28, SOIC24)**

PROGRAMMER'S GUIDE AND INSTRUCTION SET

Memory Organization

Program Memory

The 80C51 has separate address spaces for program and data memory. The Program memory can be up to 64k bytes long. The lower 4k can reside on-chip. Figure 1 shows a map of the 80C51 program memory.

The 80C51 can address up to 64k bytes of data memory to the chip. The MOVX instruction is used to access the external data memory.

The 80C51 has 128 bytes of on-chip RAM, plus a number of Special Function Registers (SFRs). The lower 128 bytes of RAM can be accessed either by direct addressing (MOV data addr) or by indirect addressing (MOV @Ri). Figure 2 shows the Data Memory organization.

Direct and Indirect Address Area

The 128 bytes of RAM which can be accessed by both direct and indirect addressing can be divided into three segments as listed below and shown in Figure 3.

1. Register Banks 0-3: Locations 0 through 1FH (32 bytes). The device after reset defaults to register bank 0. To use the other register banks, the user must select them in software. Each register bank contains eight 1-byte registers 0 through 7. Reset initializes the stack pointer to location 07H, and it is incremented once to start from location 08H, which is the first register (R0) of the second register bank. Thus, in order to use more than one register bank, the SP should be initialized to a different location of the RAM where it is not used for data storage (i.e., the higher part of the RAM).
2. Bit Addressable Area: 16 bytes have been assigned for this segment, 20H-2FH. Each one of the 128 bits of this segment can be directly addressed (0-7FH). The bits can be referred to in two ways, both of which are acceptable by most assemblers. One way is to refer to their address (i.e., 0-7FH). The other way is with reference to bytes 20H to 2FH. Thus, bits 0-7 can also be referred to as bits 20.0-20.7, and bits 8-FH are the same as 21.0-21.7, and so on. Each of the 16 bytes in this segment can also be addressed as a byte.
3. Scratch Pad Area: 30H through 7FH are available to the user as data RAM. However, if the stack pointer has been initialized to this area, enough bytes should be left aside to prevent SP data destruction.

Figure 2 shows the different segments of the on-chip RAM.

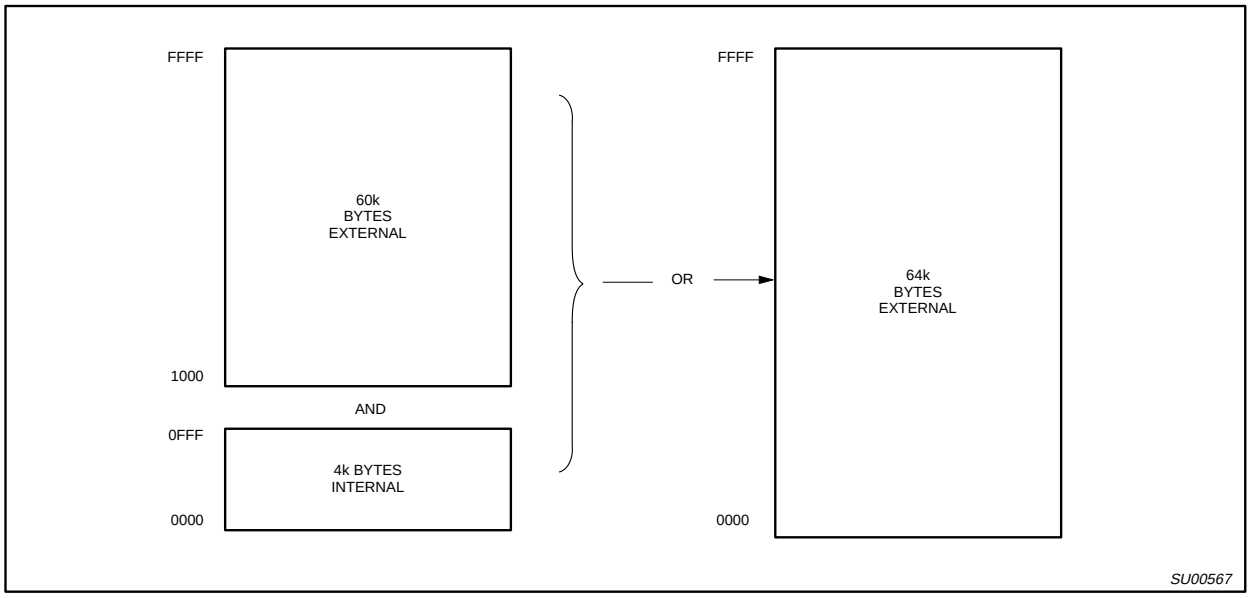


Figure 1. 80C51 Program Memory

OPTIONAL EEPROM

MICROCHIP

24AA512/24LC512/24FC512

512K I²C™ CMOS Serial EEPROM

Device Selection Table

Part Number	Vcc Range	Max. Clock Frequency	Temp. Ranges
24AA512	1.8-5.5V	400 kHz(1)	I
24LC512	2.5-5.5V	400 kHz	I, E
24FC512	2.5-5.5V	1 MHz	I

Note 1: 100 kHz for Vcc < 2.5V

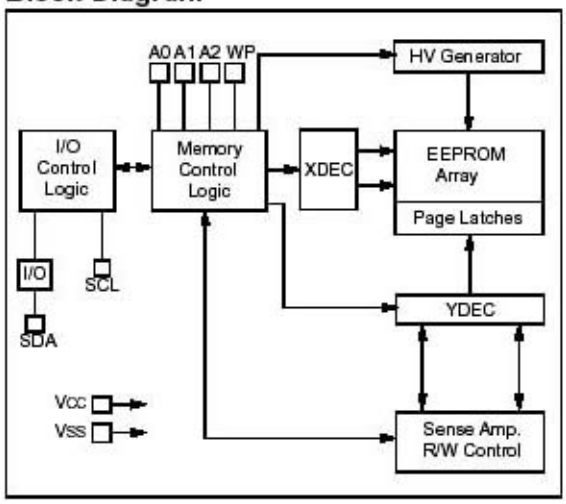
Features:

- Low-power CMOS technology:
  - Maximum write current 5 mA at 5.5V
  - Maximum read current 400 µA at 5.5V
  - Standby current 100 nA, typical at 5.5V
- 2-wire serial interface bus, I²C™ compatible
- Cascadable for up to eight devices
- Self-timed erase/write cycle
- 128-byte Page Write mode available
- 5 ms max. write cycle time
- Hardware write-protect for entire array
- Schmitt Trigger inputs for noise suppression
- 1,000,000 erase/write cycles
- Electrostatic discharge protection > 4000V
- Data retention > 200 years
- 8-pin PDIP, SOIC (208 mil), and DFN packages
- 14-lead TSSOP package
- Pb-free finishes available
- Temperature ranges:
  - Industrial (I): -40°C to +85°C
  - Automotive (E): -40°C to +125°C

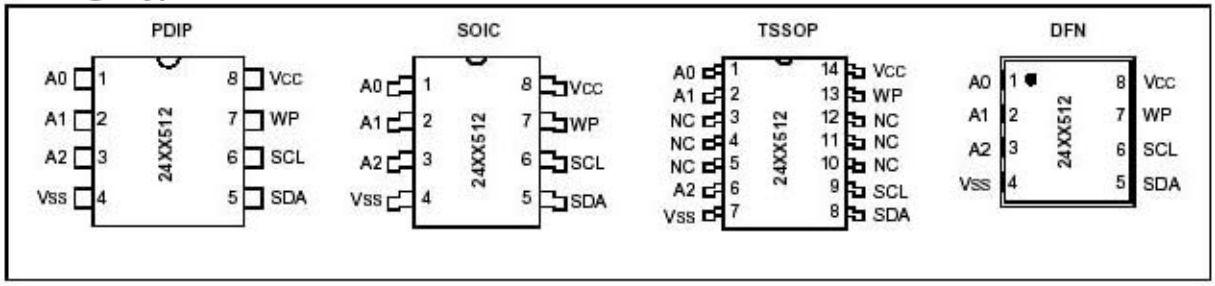
Description:

The Microchip Technology Inc. 24AA512/24LC512/24FC512 (24XX512*) is a 64K x 8 (512 Kbit) Serial Electrically Erasable PROM, capable of operation across a broad voltage range (1.8V to 5.5V). It has been developed for advanced, low-power applications such as personal communications and data acquisition. This device also has a page write capability of up to 128 bytes of data. This device is capable of both random and sequential reads up to the 512K boundary. Functional address lines allow up to eight devices on the same bus, for up to 4 Mbit address space. This device is available in the standard 8-pin plastic DIP, SOIC, DFN and 14-lead TSSOP packages.

Block Diagram



Package Type



* 24XX512 is used in this document as a generic part number for the 24AA512/24LC512/24FC512 devices.

Clock/calendar with 240 × 8-bit RAM PCF8583

1 FEATURES

- I²C-bus interface operating supply voltage: 2.5 V to 6 V
- Clock operating supply voltage (0 to +70 °C): 1.0 V to 6.0 V
- 240 × 8-bit low-voltage RAM
- Data retention voltage: 1.0 V to 6 V
- Operating current (at f_{SCL} = 0 Hz): max. 50 μA
- Clock function with four year calendar
- Universal timer with alarm and overflow indication
- 24 or 12 hour format
- 32.768 kHz or 50 Hz time base
- Serial input/output bus (I²C)
- Automatic word address incrementing
- Programmable alarm, timer and interrupt function
- Slave address:
  - READ: A1 or A3
  - WRITE: A0 or A2.

2 GENERAL DESCRIPTION

The PCF8583 is a clock/calendar circuit based on a 2048-bit static CMOS RAM organized as 256 words by 8 bits. Addresses and data are transferred serially via the two-line bidirectional I²C-bus. The built-in word address register is incremented automatically after each written or read data byte. Address pin A0 is used for programming the hardware address, allowing the connection of two devices to the bus without additional hardware.

The built-in 32.768 kHz oscillator circuit and the first 8 bytes of the RAM are used for the clock/calendar and counter functions. The next 8 bytes may be programmed as alarm registers or used as free RAM space.
The remaining 240 bytes are free RAM locations.

3 QUICK REFERENCE DATA

SYMBOL	PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNIT
V_{DD}	supply voltage operating mode	I²C-bus active	2.5	–	6.0	V
		I²C-bus inactive	1.0	–	6.0	V
I_{DD}	supply current operating mode	f_{SCL} = 100 kHz	–	–	200	μA
I_{DDO}	supply current clock mode	f_{SCL} = 0 Hz; V_{DD} = 5 V	–	10	50	μA
		f_{SCL} = 0 Hz; V_{DD} = 1 V	–	2	10	μA
T_{amb}	operating ambient temperature range		−40	–	+85	°C
T_{stg}	storage temperature range		−65	–	+150	°C

4 ORDERING INFORMATION

TYPE NUMBER	PACKAGE		
	NAME	DESCRIPTION	VERSION
PCF8583P	DIP8	plastic dual in-line package; 8 leads (300 mil)	SOT97-1
PCF8583T	SO8	plastic small outline package; 8 leads; body width 7.5 mm	SOT176-1

APPENDIX B: ALPHABETICAL INDEX

Symbols

µC/51 **8**, **22**

A

Accessories **7**, **24**, **30**
Accumulator **12**
Application program **11**, **27**, **34**
Assistance **1**

B

BASCOM 8051 **8**, **28**
Baud rate **20**, **24**, **30**
Bibliography **35**
Bit access RAM **15**, **22**, **28**
Boot Loader **7**, **16**
Buffer **9**, **13**
Buzzer **13**, **25**, **26**, **31**, **32**

C

Clock alarm **13**
Code **11**
Code area **11**
Communication **6**, **20**, **24**, **30**
Compilers **8**, **22**, **28**
COMx **6**, **24**, **30**
Connection **7**
Console **12**, **22**, **23**, **28**
Console input **12**
Console output **12**
Console status **12**
Contacts logic **8**
Container **1**

D

Data bits **20**, **24**, **30**
DDS MICRO C 51 **8**
DEBUG mode **16**, **26**, **32**
Declaration file **29**, **33**
Delay **9**, **14**
Demo programs **20**, **23**, **25**, **29**, **31**
Development PC **6**, **24**, **30**
Digital output **13**
DIR **20**

Direct acces RAM **15**
Directive **1**
Documentation **1**

E

EEPROM **13**, **19**, **22**, **29**, **A-3**
Electric protocol **20**
Electrostatic noises **1**
Entry points **11**
ESD **1**
ETHERNET **7**
Executable code **27**, **33**

F

First purchase **20**, **23**, **29**
FLASH EPROM **7**, **11**, **14**, **16**
FLASH programming **16**, **25**, **31**
FLIP **7**, **16**, **25**, **31**
Flow control **24**

G

General information **4**
GET51 **7**

H

Handshake **24**, **30**
Header **20**, **23**
HI TECH C 51 **8**
How to start **24**, **30**
HYPERTERMINAL **7**, **24**

I

I2C BUS **20**
In System Programming **7**, **10**, **16**
Initializations **13**, **19**, **28**
Integration of library **10**, **28**
integration of library **22**
Interfaces **6**
Interrupts **12**, **22**, **28**
Introduction **1**
ISP **7**, **10**, **16**, **25**, **31**

J

Jumpers **9**, **16**, **25**, **31**

L

LADDER WORK **8**
LEDs **13**
Libraries **20**, **23**
License **21**
Logic protocol **20**

M

Malfunction **27**, **34**
Memory **6**, **11**, **22**, **28**
Microcontroller **16**, **29**, **35**, **A-1**, **A-2**
Monitor **6**
Mouse **6**

N

Network **20**
NO OUT2 **13**

O

Operating systems **6**
Options **13**, **24**, **29**

P

Parity **20**, **24**, **30**
PC **6**
Performances **14**
Physic protocol **20**, **24**, **30**
Point to point **20**
Power on **13**, **20**
Projects **20**, **23**
Protection **1**

Q

QTP16Bxy.HEX **11**, **17**, **23**, **29**

R

RAM **14**, **15**, **22**, **28**
Receive buffer **9**
Redirection **12**, **22**, **28**
Registers **13**, **15**, **23**, **29**

Requirements **10**, **14**
Resources **10**, **14**
RS 232 **6**, **9**
RS 422 **20**
RS 485 **20**
RTC **A-4**
Rules **1**
RUN mode **16**, **25**, **31**

S

Safety **1**
Serial line **20**, **24**, **30**
Software development tools **10**, **22**, **28**
Special function registers **15**, **23**, **29**
SRAM+RTC **A-4**
Stack **14**, **28**
Stop bits **20**, **24**, **30**
SYS51CW **8**
SYS51PW **8**

T

T89C5115 **16**, **18**, **A-1**
T89C51CC02 **16**, **A-1**
Timeout error **17**
TIMER0 **12**, **14**, **22**, **28**
Timing **14**
Trademarks **2**
Transmit buffer **9**

U

USB **7**
Use of library **10**

V

Vector **14**
Version **3**

W

Warranty **1**, **2**