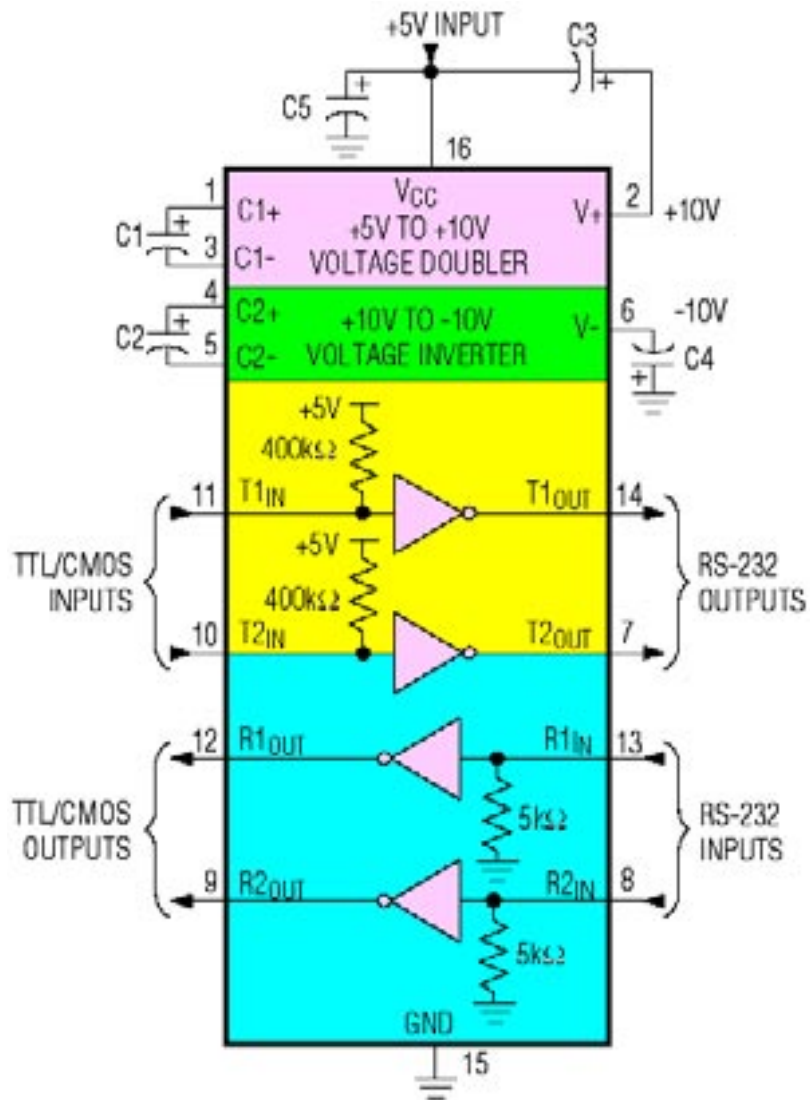


*Theoretic/Practical course on BASCOM AVR Programming.
Author: DAMINO Salvatore.*

SOFTWARE UART MANAGEMENT

Sometimes it happens that an application requires more serial lines than those available on the selected microcontroller. What can be done to solve this problem, in an easy and efficient mode? The experienced programmers will simply think to develop a subroutine, at low level, in order to obtain a proper **Software** communication line. The solution is really perfect but, when **Assembler** language is not sufficiently known, it can be used an interesting feature of **BASCOM** that allows to develop some communication serial lines by using specific **Instructions** of this powerful **Compiler**.

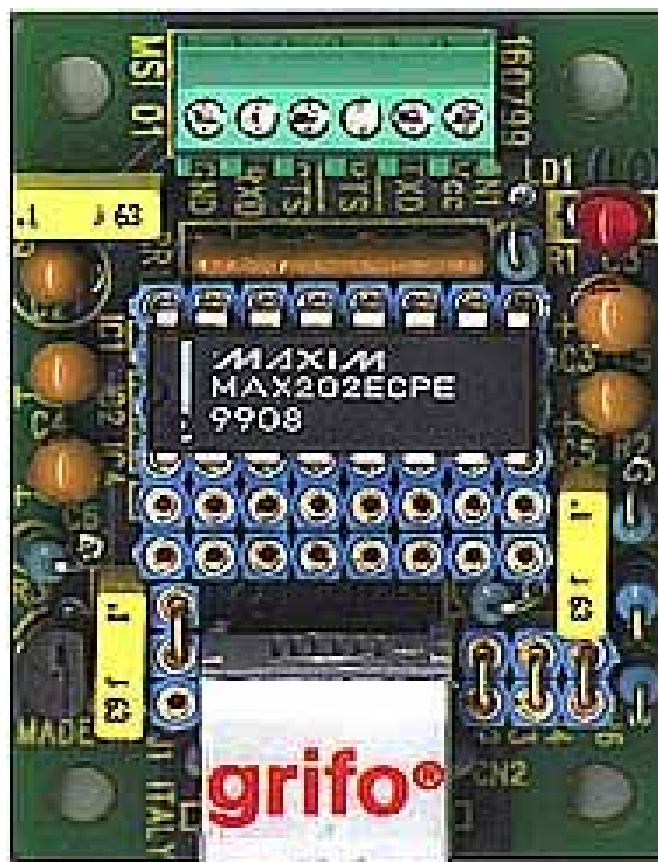


RS232 driver, model MAX202 produced by Maxim.

The instructions supplied by **BASCOM** for serial communication management, are all **Suspensive**. This means that once executed the instruction dedicated to serial reception remains under execution until one character is completely received.

This operational mode, suggested in **Example.053**, can be right for a small educational example but, **certainly.it** can't be used for a real application. Infact when this mode is used then during any communications the program is halted for each character to receive.

In order to overcome this heavy limit it is necessary to appeal to **INTERRUPT** use and features.



MSI 01 Multi Standard Interface configured in RS 232 version.

By using a signal capable to generate interrupt as receive line plus proper service subroutines it can be obtained a really fluid serial communication. The single irreversible limits that remain in software **UART**, even with interrupt management, is the higher **Baud Rate** that can be used and the impossibility to receive and transmit character contemporaneously. The supported speed is anyway acceptable.

Example.053. Development and Management of a Software UART, with Serial RS 232 Driver, through BASCOM AVR. Suspensive Management of Communication.

Added Definitions:

None

Added Declarations:

None

Added Instructions:

OPEN, PUT, GET, CLOSE.

Added Operators:

None

Example program **053** of **BASCOM AVR** course.

It manages an additional **Asynchronous Serial** line.

The **Mini Module GMM AM08** is provided with only one **HW Serial Line**, connected to **UART** peripheral device, and this program adds a second **Software** serial line, connected to two **I/O** lines. The SW serial line is managed through the high level instructions of **BASCOM** that are **Suspensive**, or in other words, they stop program execution during transmission and reception of characters.

By comparing a SW serial line to an HW one, many differences came out.

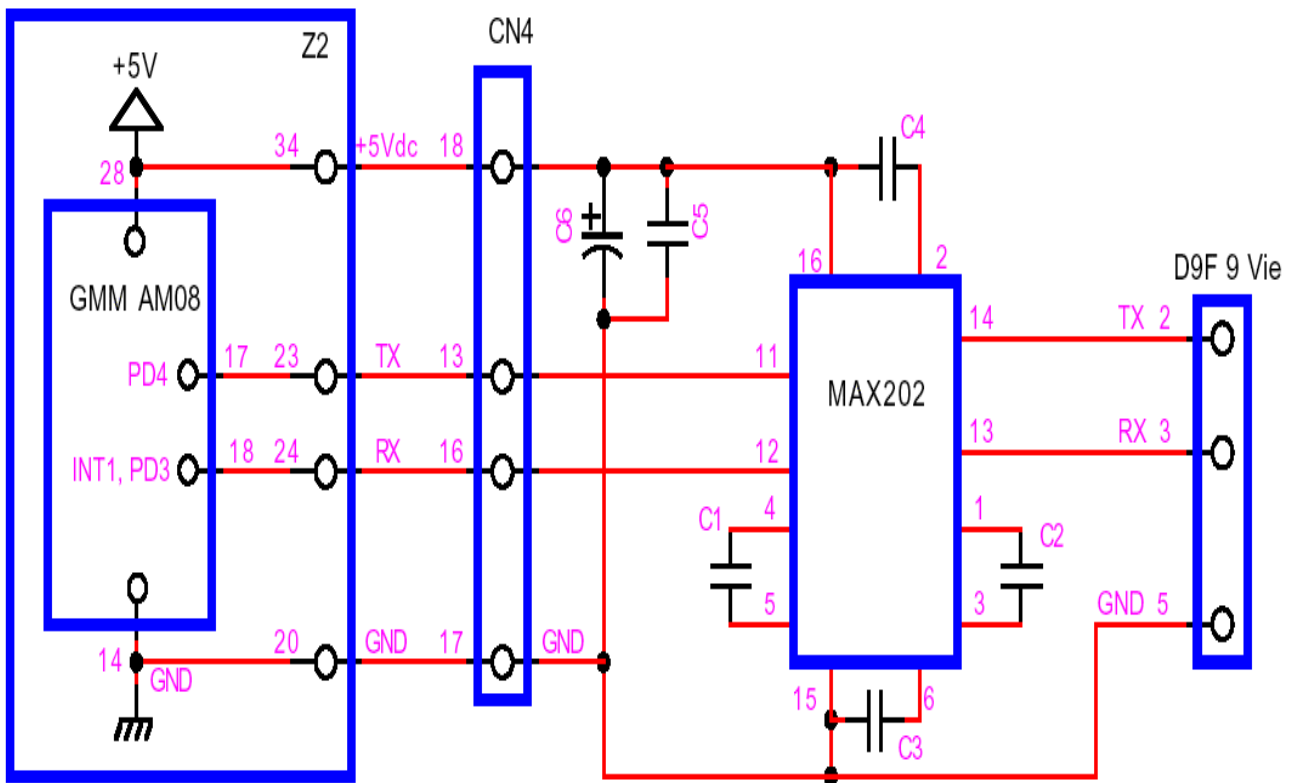
The first is a limits on communication speed (**Baud Rate**), a second difference for a SW serial line is a really higher execution load for the microcontroller.

Finally the third difference is the denied possibility to transmit and receive character contemporaneously on the SW serial line.

The program supports the following operations on SW serial line:

- communicate with **4.800 Baud, 8 Bits x chr, 1 Stop bit, No parity** physical protocol;
- **Receive** characters in **suspensive** mode and shows then on console;
- **Transmit** the characters typed on console.

About electric protocol, the sw serial line of **Mini Module** is at **TTL** level and normally it must be **Buffered** in order to communicate with other external serial devices.



RS 232 Software Serial Interface.

The diagram reported in the figure shows a typical **RS 232** interface based on a classic **MAX202** device.

This type of interface allows to communicate up to **33 m (100 Feet)**.

It is possible to fastly produce this interface by using a comfortable prototype printed circuit board.

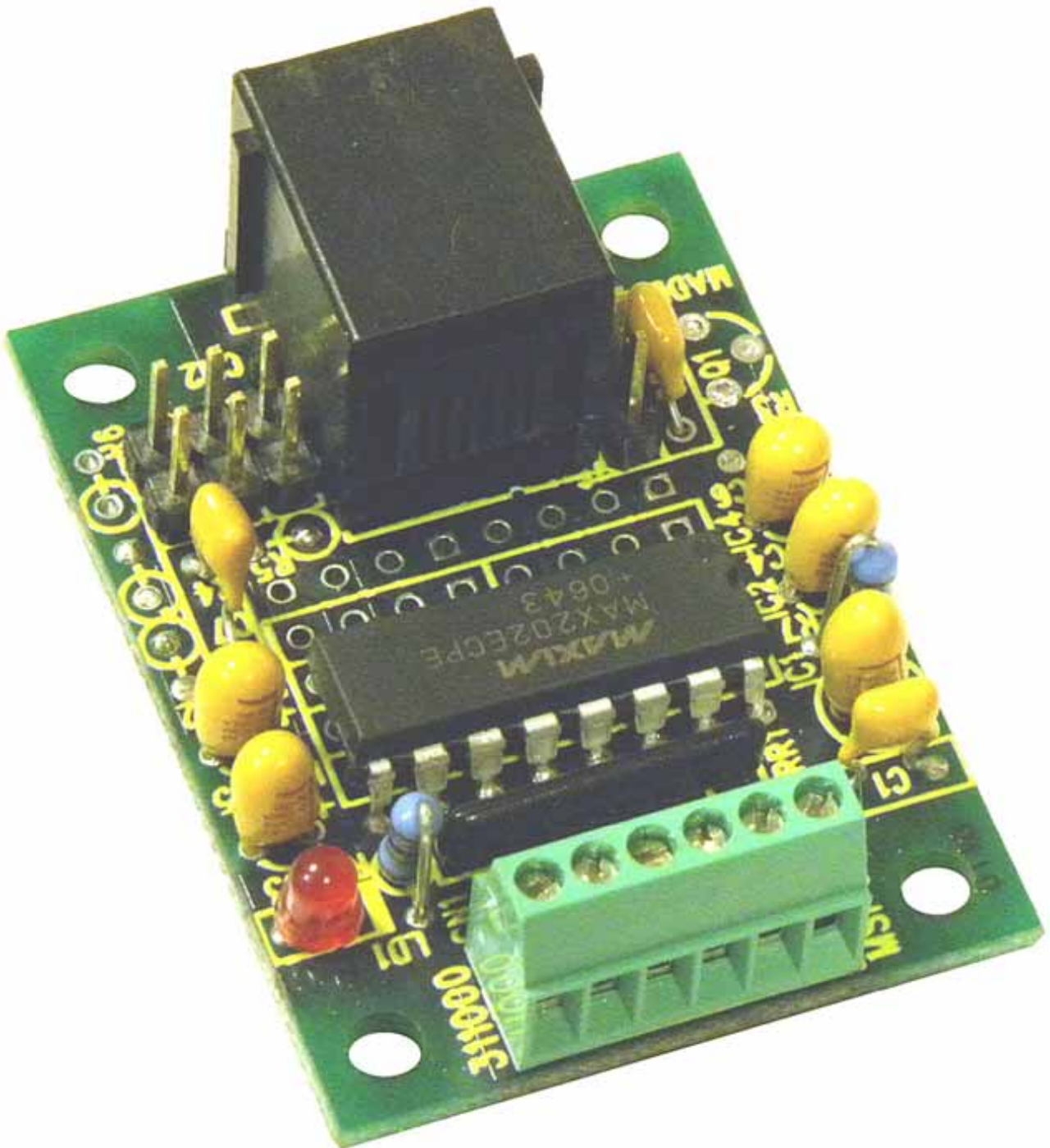
For those users that didn't want to produce the interface we remind the availability of a multi interface card named **MSI 01**. It is available in different version and it converts a **TTL** serial line in **RS 232**, **RS 422**, **RS 485** or **Current Loop** electric standard.

The user can communicate with SW serial line by using a console provided of monitor and keyboard, connected to HW serial line, with a fixed physical protocol at **19.200 Baud**, **8 Bits x chr**, **1 Stop bit**, **No parity**.

This console can be another system capable to support a serial **RS 232** communication.

In order to simplify the use it can be used a **PC** provided of one **COMx** line, that execute a terminal emulation program as **HYPERTERMINAL** or the homonym modality provided by **BASCOM AVR** (see **IDE Configuration**).

The program works only when the **GMM AM08** is mounted on **Z2** socket of **GMM TST3!!**



MSI 01 Small Interface Board, in RS 232 version.

Example.054. Development and Management of a Software UART, with Serial RS 232 Driver, through BASCOM AVR. Not Suspensive Management of Communication.

Added Definitions:

None

Added Declarations:

None

Added Instructions:

ON INT1 ; CONFIG INT1 ; ENABLE INT1.

Added Operators:

None

Example program **054** of **BASCOM AVR** course.

It manages an additional **Asynchronous** serial line, with **Interrupts**.

The **Mini Module GMM AM08** is provided with only one HW serial line, connected to **UART** peripheral device, and this program adds a second SW serial line, connected to two **I/O** lines. The SW serial line is managed through the high level instructions of **BASCOM**, joined with **Interrupt**, in order to become not suspensive, or in other words, they doesn't stop program execution during **Transmission** and **Reception** of characters.

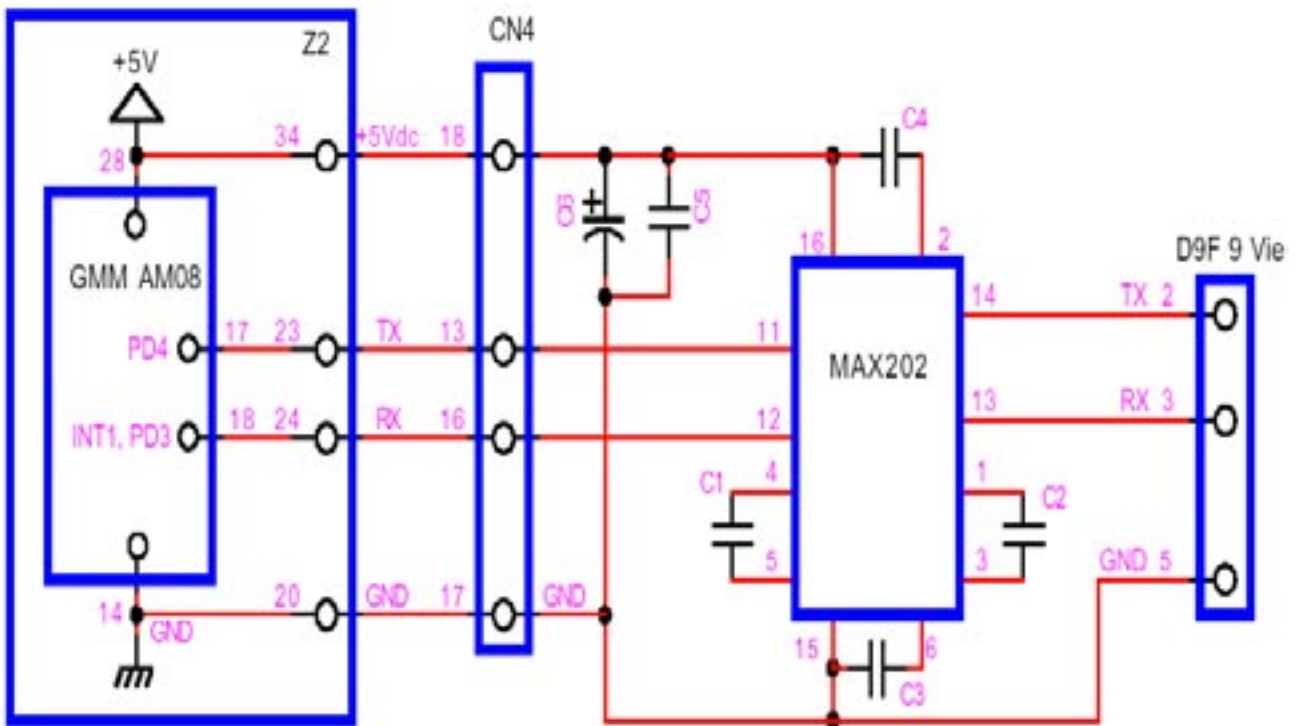
By comparing a SW serial line to an HW one, many differences came out. The first is a limits on communication speed (**Baud Rate**) correctly supported. A second difference for a SW serial line is a really higher execution load for the microcontroller.

Finally the third difference is the denied possibility to transmit and receive character contemporaneously on the SW serial line.

The program supports the following operations on SW serial line:

- communicate with **4.800 Baud, 8 Bits x chr, 1 Stop bit, No parity** physical protocol;
- each received character is recogized by an **Interrupt**, then received and saved inside a circular receive **Buffer**;
- the **Received** characters in a **Not Suspensive** mode are then displayed on console;
- **transmit** the characters typed on console.

About electric protocol, the SW serial line of **Mini Module** is at **TTL** level and normally it must be **Buffered** in order to communicate with other external serial devices



RS 232 Software Serial Interface.

The diagram reported in the figure shows a typical **RS 232** interface based on a classic **MAX202** device.

This type of interface allows to communicate up to **33 m (100 Feet)**.

It is possible to fastly produce this interface by using a comfortable prototype printed circuit board.

The user can communicate with SW serial line by using a console provided of monitor and keyboard, connected to HW serial line, with a fixed physical protocol at **19.200 Baud, 8 Bits x chr, 1 Stop bit, No parity**.

This console can be another system capable to support a serial **RS 232** communication.

In order to simplify the use it can be used a **PC** provided of one **COMx** line, that execute a terminal emulation program as **HYPERTERMINAL** or the homonym modality provided by **BASCOM AVR** (see **IDE Configuration**).

The program works only when the **GMM AM08** is mounted on **Z2** socket of **GMM TST3!!**

Example.055. Development and Management of a Software UART, with Serial RS 232 Driver, through BASCOM AVR. Complete Connection of a QTP 03 Serial Display.

Added Definitions:

\$SERIALINPUT ; \$SERIALOUTPUT.

Added Declarations:

None

Added Instructions:

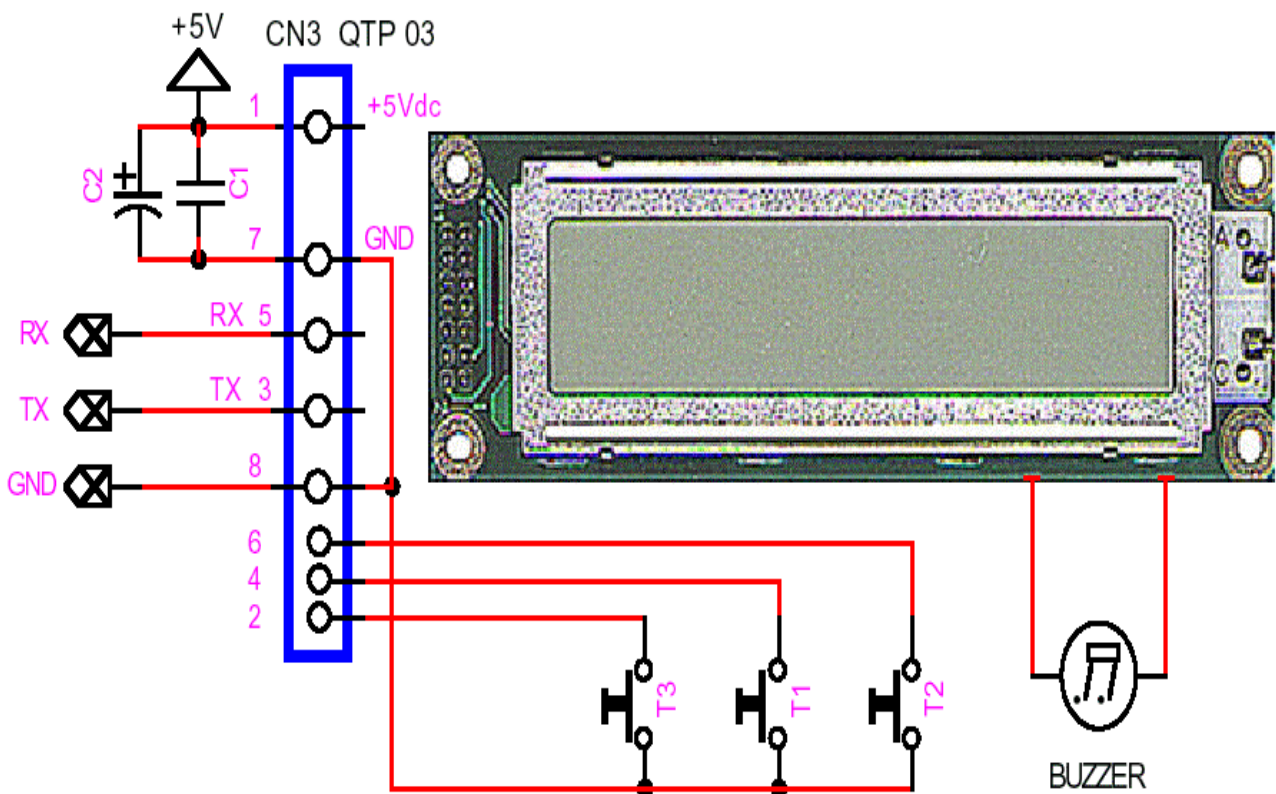
PUSHALL ; POPALL.

Added Operators:

None

Example program 055 of BASCOM AVR course.

It manages an Operator Panel, or Serial Display, QTP 03 through additional Asynchronous Serial Line, with Interrupts.



Quick Terminal Panel type QTP 03 in RS 232.

The program communicate with **QTP 03** by using a physical protocol **4.800 Baud, 8 Bit x chr, 1 Stop bit, No parity** set on SW **Serial Line**.

In order to simplify the program, the SW serial line is managed through the **High Level Instructions** of **BASCOM** dedicated to console; these are **Redirected** on SW serial line in place of the HW one, thanks to a suited selector defined in the source.

About electric protocol the serial line of **QTP 03** can be either **RS 232** or **TTL** and thus it can be connected to **Mini Module** both with an electric driver (i.e. the **MSI 01** interface) or directly.

The program supports a small subset of the numerous functionalities offered by **QTP 03** and they are selectable by user through a console provided of monitor and keyboard, connected to HW serial line.

The used physical protocol is fixed at **19.200 Baud, 8 Bits x chr, 1 Stop bit, No parity**.

This console can be another system capable to support a serial **RS 232** communication.

In order to simplify the use it can be used a **PC** provided of one **COMx** line, that execute a terminal emulation program as **HYPERTERMINAL** or the homonym modality provided by **BASCOM AVR** (see **IDE** Configuration).

The program works only when the **GMM AM08** is mounted on **Z2** socket of **GMM TST3!!**