



**DOS based hyper-terminal
emulation program for
RoboteQ controllers**

Summary

One of the most effective ways of programming a computer to drive a RoboteQ controller is to use a PC like SPC (Single Board Computer) in the same fashion as an Embedded Microcontroller. The most effective OS (Operating System) is DOS, since, unlike Windows or Linux, it allows to reach any hardware section of the SPC.

This Application Note presents:

- a free OS cloning DOS
- a method to utilize a fully static SPC (no hard disk and no CPU fan) for enhanced mechanical reliability
- the emulation of Hyper-Terminal, a commonly used Communication program running under Windows via the RS232, by the program "hyperterm_dos.exe."

We will refer to a Microcontroller SPC to stress the fact that the SPC is used with full access to its hardware, namely the RS232 Port, the Parallel Port, the Timer and various services accessible through BIOS calls.

The program "hyperterm_dos.exe" is implemented in C, with few lines implemented in "in line" assembly code in those situations where C functions are not readily available.

Using a mechanically static SPC

The most important requirement of a Microcontroller SPC is to be mechanically static, with no moving parts.

The Via Mini ITX comes with a fan-less CPU eliminating the fan; as an additional step the hard disk is replaced by a Flash Memory Card, therefore achieving a fully static design.

The size of the Flash Memory Card is inessential, since the overall software footprint is below 10 Mbyte. This means that the cheapest Flash Memory Card can be used. We have found that the smallest one is 256 Mbyte, very large compared to what needed.

The need for a Real Time OS

Microcontroller SPC applications require an Operating System which leaves the application in full control of the hardware.

Operating systems like Windows XP or Linux perform the so called "Hardware Visualization"; they prevent the application from driving the hardware. The OS has exclusive control of the hardware and, should an application try to reach the hardware, they block the attempt and usually show on the screen a message informing the user that the application has performed an illegal operation and has been shut down.

This means that any application request for hardware control is intercepted by the OS and performed by the OS; this introduces delays, forces the programmer to use languages like Visual C++ or Visual Basic which incorporate "objects" which handle the hardware. All this adds to the cost and degrades performances.

For this reason we have opted for FreeDOS, a DOS clone freely available from Internet. It behaves like Microsoft DOS and it is robust and bug-free. There is no learning curve, since all the commands are identical to the ones in DOS.

And once the application is launched, the OS gets out of the way allowing real time performance.

Formatting the Flash Memory Card

The first step consists of formatting the Flash Memory Card.

The SPC needs to be setup to boot from a USB floppy disk; this is a BIOS option very simple to activate.

A bootable floppy disk containing a minimum DOS and the utility FDISK will boot the SPC; after booting FDISK is used to setup an active DOS primary partition. Once this is done, the SPC is setup to boot from Hard Disk 0 (BIOS option) and the SPC will boot normally.

Loading the software

The next step is to copy the software (OS and Hyterm_dos.exe) from floppy disk to disk C. Once this is done, the SPC will boot and run the program Hyterm_dos.exe.

Once this is done, the floppy drive is no longer needed.

It could become necessary from time to time to load new software; this will be easily accomplished by using again the floppy drive.

The program Hyterm_dos.exe

Implementing C programs under DOS is simple; there are free compilers available which will do the job starting from a C source file created with a text editor.

The source file for Hyterm_dos.exe is Hyterm_dos.c

The program uses:

- The system timer to trigger repetitive or non repetitive events at precise time intervals.
- The RS232 port to communicate with the RoboteQ controller.
- The Parallel Port for additional functions.

The timer is programmed directly at the relevant chip set registers. The system date and time registers are used to generate an absolute time reference in milliseconds which never rolls over (actually it does every century) so all uncertainty relative to a timer reaching maximum count and resetting to zero is eliminated.

Program timers are written as C functions, which allow the creation of one-shot or repetitive timers, with programmable delay, ON time and act as trigger timers, becoming TRUE for a very short trigger-like period of time. These timers are poll based; the CPU is so fast that there is no real need for an interrupt based timer, although such interrupt is available, if the programmer chooses this way.

The Serial Port COMM1 is setup using BIOS services; it is a simple Int call and BIOS takes care of all the details. Also in this case we read the port by polling, although an interrupt based RS232 function could be created, if so desired.

The Parallel Port is programmed by the simple C instructions INP/OUTP which allow reading or writing a byte or a word to the port.

The program source code is heavily commented clarifying step by step the meaning of the code.

This code has the following advantages:

- Shows the inner workings of the RS232 Port at the lowest hardware level.
- Shows the inner workings of the system timer at the lowest hardware level.
- It is real time. The application controls the hardware and the OS is out of the way.
- Shows at hardware level how the SPC and the RoboteQ controller communicate.