Chapter 5

Software tools and support

The purpose of this chapter is to present some information that I found useful during the project development stages. This may be helpful at a later stage.

5.1 Software development tools

The compiler, debugger, assembler and linker were purchased from Microtek Research Incorporated (MRI®) [1]. This set of supporting software was purchased as an integrated package titled the "XRAY Integration Kit." Most of the documentation that comes with this package explains in good detail how to use the tools. One of the concerns that I found with this software kit was that it was too complex and advanced to get a really good grip of without a very long "learning curve." At times, it seemed like overkill for the intended application. Most of the "bells and whistles" provided in the package were never utilized. The package was used in our work because it was the package that Grayson used to develop their kernal and we had to be compatible with them.

At times, upon power up, the debugger would load in the source code and crash after executing the first few lines of code. The frequency of this occurrence dampened productivity often times. Unfortunately, the technical support for these tools expired after the first couple of years and I wasn't able to get any support from Microtek when I needed it. Some other times, when the application crashed it would hang the Operating System on the PC. I feel that there are three ways to deal with such occurrences: renew technical support with Microtek, get a simpler set of software tools that were better suited to our needs, or to tolerate the errors and try to minimize the frustration.

5.2 Motorola's support

Most of the information that I needed on the 68306 I found by referring to Motorola's web-sites [9]. Motorola's documentation on the major aspects of the 68306 is very educational and did help me figure out other issues along project development. For example, when browsing through the kernel code, it is a very handy source to understand how the microcontroller is configured and how it can be manipulated based on the project needs.

5.3 Display devices

There are two ways of debugging the application code written on the WMI platform (besides using compiler tools). The outputs of routines can be directed to either the accompanying handheld display screen (LCD) or routed to a PC in terminal mode.

The handheld unit is connected to the WMI through its allotted phone jack. This is a very convenient tool, especially during message processing. The

handheld unit has an in-built keypad that elevates the debugging status [1]. Key-presses can be used to perform a hierarchical debugging function, if necessary. This was used at one point. The kernel code polls the handheld unit using a dedicated 6805. Because of the internal protocol used and the limited 6805 buffer available, the message length of the data routed through this channel is limited. It is recommended in the kernel code to wait until there is at least three times as much buffer space available as there is data to be displayed, before streaming data to this unit. Even when buffering is practiced, the handheld unit routines are at times unpredictable. This unpredictability is characterized by data being partially displayed, extended ASCII characters being displayed instead of character strings, the cursor freezing up (instead of blinking normally), etc. This happens possibly because the processor controlling the handheld unit has slower serial data processing speed.

The PC is convenient when there is sufficient data that needs to be displayed. The PC is connected through the available serial port to the WMI. The Windows ® program "Terminal" is entered with the appropriate baud rates and COM ports being selected. Data transmitted can now be viewed on the open window. Since the "Terminal" program has a faster data processing speed data and has a larger internal buffer than the handheld unit, the visual display on the PC is better.