ME2700 Firmware Notes

Martin Eberhard
29 January 2016

The ME2700 EPROM Programmer's firmware is partitioned into two components that are loaded separately.

The Kernel

The first portion is the "Kernel", which must be loaded into the PIC microcontroller with a PICKit-III or similar device. The primary function of the Kernel is to allow the rest of the firmware to be loaded via the ME2700 RS232 serial port, so that firmware updates can be distributed electronically and installed without any special equipment.

The architecture of the particular PIC microcontroller used in the ME2700 has a provision to protect a block of its Flash memory from accidental over-writes by firmware. The Kernel resides in this protected block, so that even if a serial port firmware load goes terribly wrong, the kernel itself will remain intact, allowing you to reload the firmware anyway.

The minimum size for this protected block is 2K bytes - much larger than is required for the loader. So as not to waste Flash space, the remainder of this protected block is filled with low-level routines for servicing the serial port interrupt, printing, getting user input, accessing the internal EEPROM and external SRAM, etc. The main firmware accesses these routines through a fixed-location jump table.

It is unlikely that this code will ever get updated. In the unlikely event that a bug is found in any of the low-level routines within the kernel, such a bug will be fixed by replacing the offending routine with one in the main ME2700 firmware.

The ME2700 Firmware

The remainder of the PIC's 32K Flash memory holds the ME2700 firmware, which may be reloaded via the ME2700 serial port. As of version 1.1, this code comprises about 9K of actual assembly-language code, and about 20K of tables and text strings.

The firmware to program a single type of EPROM could be relatively simple. But a general EPROM programmer (both hardware and firmware) gets a little complex:

- The pinouts of the various EPROMs are not consistent
- Most EPROMs are powered by +5V, but some also require +12V and -5V.
- There are about 6 different Vpp programming voltages to support
- Some EPROMs require an elevated Vcc during programming
- Some EPROMs require an elevated voltage on other pins (e.g. +12V on -OE) during programming
- Although there is a "standard" (50 mS pulse) algorithm that works on many EPROMs, this is in fact not universal, and will fail to meet specifications for several EPROMs. Many EPROMs require a shorter programming pulse, and others require (or at least encourage the use of) a "Smart" (or "Fast" or "Express" or "Rapid" or some other name) algorithm - and there is no standard for such algorithms.
- Some EPROMs (and all EEPROMs) cannot be blank-checked prior to programming, because they erase to an indeterminate state (or in the case of EEPROMs, don't get erased)

- Some EEPROMs supported by the ME2700 require a per-byte erase prior to writing, while others require data polling to determine the end of the byte-write cycle. Others require a timer to determine the end of the byte-write cycle.

And so, even with some clever coding techniques, the ME2700 has 2K + 9K = 11K of assembly language code. This is a fair bit of code for assembly language, and provides opportunities for bugs, despite fairly extensive testing.

The most likely place for a bug to materialize is in the table that describes the supported EPROMs - it is possible (despite the testing that I performed) that some parameter or other will be wrong in one (or some) of the EPROMs supported by the ME2700. When such a bug is brought to my attention, I will release a new version of the firmware, and distribute it electronically.

However, if you are desperate to program such an EPROM, there is a relatively easy way to get around such a bug: Use the COPY command of the Custom EPROM Editor to copy the specs of the particular EPROM into one of the four custom EPROM slots. Then use the Custom EPROM Editor to fix whatever spec I got wrong. You can then use the fixed custom EPROM type to replace the broken EPROM type in the firmware.