Self-Programming of NEC Electronics K1 Flash Microcontrollers
A Primer

Introduction

For decades, firmware engineers have had to contend with the problem of firmware upgrades. When microcontrollers (MCUs) were first introduced in the 1970s, one version came with a quartz window over the electrically erasable read-only memory (EEPROM). This expensive quartz window allowed the MCU's memory to be erased when placed under ultraviolet (UV) light, after which the MCU could be reprogrammed.

MCUs soldered to printed circuit boards couldn't be programmed, so they first had to be programmed in a machine (a programmer) and then placed into sockets that were soldered onto the boards. This step allowed developers to change the firmware during the design phase, but it did not address the problem of how to change code once a product went to production. At that time, the only option was to move from UVEPROM directly to mask ROM. In the 1980s, semiconductor companies started making one-time programmable (OTP) MCUs that gave designers the flexibility to change firmware after a product went into production. There was one problem, however. If firmware changes needed to be made to an OTP MCU to a board, the MCU had to be removed. This was not an easy thing to do, and often after doing it a few times, the lands and etch on the board would become damaged and the board would have to be scrapped. Companies using boards with sockets provided customers with an easier method for upgrading, but as a consequence the manufacturing cost was very high.

In the 1990s the advent of flash memory allowed engineers to reprogram MCUs without having to remove them from the circuit boards, which solved some problems, but not the one of how to upgrade firmware in MCUs designed into remote-type systems. The obvious ways of handling the problem—having a customer return the product to the manufacturer to be reprogrammed or having the manufacturer send out entirely new devices—were very expensive. There was also the problem of how to accommodate upgrades to products installed in remote locations, for example, in microcontroller-based pay phones in several thousand telephone booths dispersed throughout the western United States. The notion of sending people out to those various locations to manually reprogram the flash memory in 5000 payphones would be too expensive and time-consuming to consider. The solution was something called "flash self-programming", also known as remote programming because it allowed an MCU to be reprogrammed remotely.

About Self-Programming

A flash-based microcontroller has three distinct code areas: the target application program, the user-written code that manages the self-programming process (known as A1 mode), and the firmware that performs tasks such as erasing a block of flash memory. After A1 mode calls the firmware and the firmware starts performing these tasks, the MCU goes into A2 mode.

Self-programming comprises three steps:

1. Signaling the MCU that it is going to be reprogrammed
2. Sending new data to the MCU
3. Making the MCU reprogram itself

All of the NEC Electronics K1+ MCUs come in flash versions of varying sizes and all can be self-programmed. The most logical way to get new data to an MCU for self-programming is via a serial interface. However, since there are no restrictions, data also can be sent via the port lines or an external memory interface. This flexibility allows designers to choose the method that best suits them.

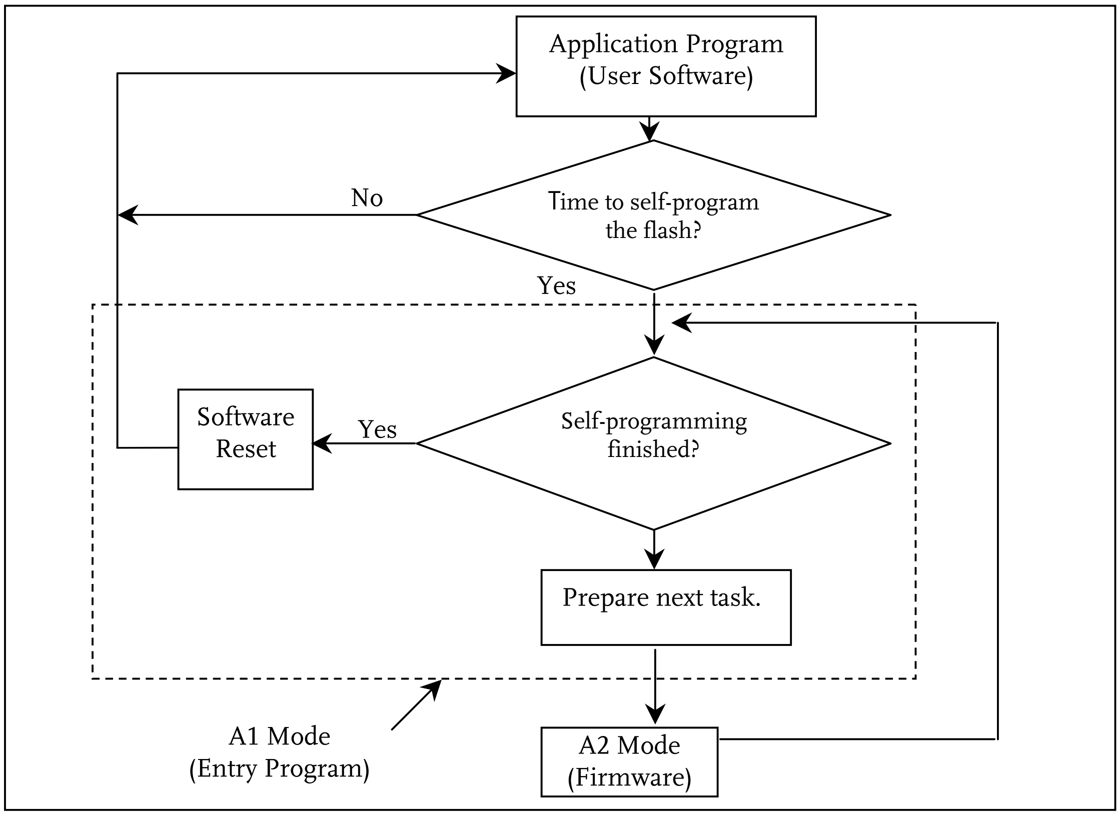
During normal operation while running the application program, an MCU cannot be self-programmed. It first must receive an external signal that causes the application program to abort and the self-programming process to begin. In other words, the MCU goes into A1 mode (shown in Figure 1) and performs the following operations to reprogram itself.

1. Sets up program initialization.
2. Selects and erases a flash block.
3. Programs the flash block with new code.
4. Verifies that the programming is correct.
5. Executes software reset.

These tasks are set up in A1 mode and executed in A2 mode (Figure 1). Once the MCU reprograms a block of flash memory, it selects a new block and repeats the process. When all blocks have been reprogrammed, the MCU initializes a software reset to start program execution using the new the firmware. NEC Electronics' 78K0S/K1+ MCUs have flash blocks of 256 bytes while the 78K0/K1+ and V850/K1+ MCUs have blocks of 2 Kb. Figure 2 shows a detailed diagram of the resources needed to implement self-programming, showing the flow from normal operation to the entry program (A1 mode).

Parameters for reprogramming the flash memory in A2 mode are written by the user and reside in the flash memory that must *not* be overwritten. The MCU then must perform the tasks instigated by CALL instructions to the firmware functions. At this time, the MCU is in A2 mode and the firmware for the A2 mode cannot be altered. Before A1 mode calls functions in A2 mode, it passes arguments and variables via register bank 3, the 48-byte entry RAM area, and 4- to 256-byte data buffer. A stack of 30 bytes is also required. The MCU's on-chip RAM is used for the entry RAM, data buffer and stack. After A2 has finished its CALL, it passes results back to A1 mode via register bank 3, the entry RAM and the data buffer.

Figure 1. Self-Programming Modes

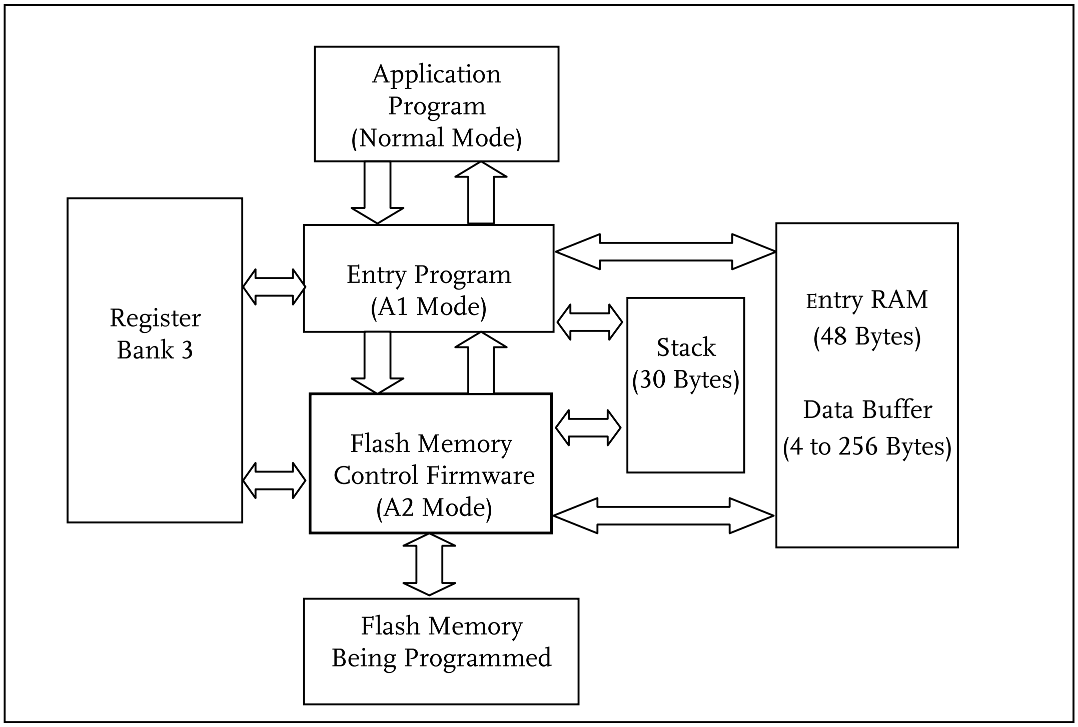


These functions are called in A2 mode:

- Initialization
- Block erase
- Block blank check
- Word programming
- Block internal verify
- Read flash information
- Set flash information
- Mode check

To protect the customer's code, the contents of K1+ flash cannot be read. The write verify operation compares the data that *should be* programmed into the flash block with the data that *is* programmed. The reading of the data can only be done by the firmware in A2 mode, thereby protecting a user's program from being viewed.

Figure 2. Self-Programming Resources



K1+ Flash Sizes

One problem in many microcontroller families is the range of sizes for mask ROM and the limited (usually one or two) sizes for flash. If a designer needed to use flash memory in a product requiring 16 KB of ROM, that designer might be forced to buy a more expensive 60 KB flash device. NEC Electronics' new K1+ MCUs avoid that problem by having multiple flash size options in each family. For example, the KE1+ family (a subfamily of the K1+) has flash versions of 16, 24, 32, 48 and 60 KB, giving designers the option of making products with a less expensive 16 KB flask device instead of a more expensive 60 KB one. The K1+ MCUs require only a single voltage for the flash devices in all 8- and 32-bit families.

Conclusion

The uses for self-programmability in an expanding variety of applications will continue to grow. The 8- and 32-bit K1+ flash microcontrollers provide customers with ease and flexibility in updating firmware in existing products.

Please contact us or visit our web sites for more information.

[Japan]
NEC Electronics Corporation
Kawasaki, Kanagawa, 211-8668 Japan
+044-435-5111
www.necel.com

[North America]
NEC Electronics America, Inc.
Santa Clara, California 95050 U.S.A.
1-800-366-9782
+01-408-588-6000
www.necelam.com

[Europe]
NEC Electronics (Europe) GmbH
Düsseldorf, Germany 40472
0211-650301
www.ee.nec.de

[Asia and Oceania]
NEC Electronics Hong Kong Ltd.
+852-2886-9318
www.necel.com.hk

NEC Electronics Singapore Pte. Ltd.
Singapore 307684
+65-6253-8311
www.necel.com.sg

NEC Electronics Taiwan Ltd.
North Road, Taipei, Taiwan, R.O.C.
02-2719-2377

Document no. U17087EU1V0AN00