A Self-programming Processor can Facilitate Production

Innovative use of device re-programming capability can simplify testing and improve results.

Re-programmable microcontrollers are transforming the nature of embedded applications. Embedded applications in many industrial, automotive, and medical applications implement sensors that require calibration during manufacture to store offset, compensation slope or other configuration data. These systems often have used potentiometers or Serial EEPROM devices to set up and store this calibration information.

In this article we will describe a new way to leverage the self-programming capability of microcontrollers to simplify the process of performing calibration during the production process.

Most popular processors with programmable memory also have the ability to read, write, and erase their own internal memories. In the past, most programmable devices were programmed before placement on the circuit board. With this method, the final application code, and often any test code, would be programmed into the microcontroller even before the product was assembled.

There are drawbacks to preprogramming all code prior to production. First, test routines and other disposable code may have to be run at functional test or in some later process, increasing rework costs when bugs are found. Secondly, the combination of required test or calibration code plus application code requires larger memory, driving the need for a larger (more expensive) processor.

With In-System programming (ISP), where the processor is programmed on the PCB, at manufacturing time, a processor can use the entire code space during the test-phase, and then after test and calibration are complete, the final application can be programmed in the target. Also, if any tables, parameters or other software modifications need to be made based on the test results, these can be downloaded into the final product as well, using in-system programming (ISP) methods.

Let's use as an example application low cost circuit that must accurately read temperature, and accurately check the current flow of a motor that the circuit will eventually be connected to it. The temperature sensor selected for this product is inexpensive, and has a challenging non-linear curve that must be adjusted in software. The current sensor also is very inexpensive, and must rely on calibrating the voltage read back from the voltage drop of current flowing through a piece of wire. This curve must also be adjusted in software to reflect approximate real values.

The product will be built in high volumes, over 100,000 units per month. The manufacturing line beat rate allows no greater than 15 seconds at each processing step. The application code consumes 90 percent of the flash code space in the device, and all calibration values calculated by off target board application need to be stored in the processors EEPROM area.

This circuit also has some other features, some of which can aid us in production:

- TTL serial port making it to an off board connecter, or test points.
- Processor containing 1K of EEPROM and 2K of code space.
- Four 10bit A to D converters built in.

Calibrating the Temperature Sensor

Normally to calibrate a temperature circuit, the product would sit in a thermal chamber, and be exercised through some temperature cycles. Values would be read, and adjustments made either physically (trim adjustments) or though loading calibration data into the processor. All of this calibration time is counter productive to a high speed assembly line. Because of the speed of the line, this approach is not a viable option.

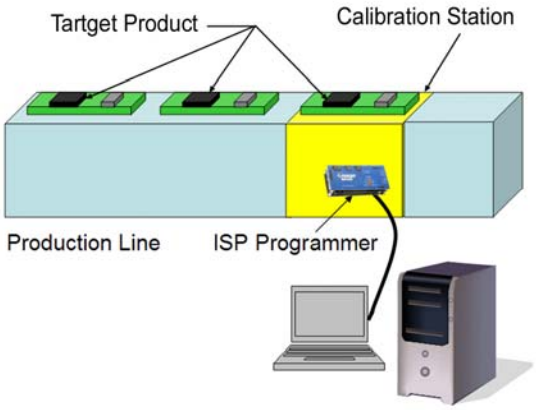
Our goal then is to make the target product gather information from its sensors, store that data, and at a later time use the data to calibrate the target product. The power of a reprogrammable processor allows the product to be turned into a data collection unit during manufacturing. Later in the production line, the data collected will be analyzed externally, and the result will yield calibration values that will be used to make the product more accurate.

In the case of the temperature sensor; the product could be powered up on the assembly line, readings of the target product temperature sensor could be read at intervals and stored in the processor memory, while the product is traveling through a calibrated oven. The oven could contain temperature controlled zones, and as the product goes through the oven, it can record several values in each zone. Once the product is out of the oven, a station could read back the data out of the processor memory. When the data is extracted, since the over temperatures are fixed, any deviation from what the target product recorded is the error or the temperature sensor. Now compensation tables can be generated and stored into the target processor, along with the products final application code. The test-code that originally stored the calibration data in the oven can be discarded to recover precious memory-space in the processor. This only pieces of equipment required to perform this task would be an ISP programmer, a modified oven, and a PC attached to the programmer that has the ability to read and write from the processor memory.

<u>Calibrating the Current Sensor</u>

A similar approach could be used to calibrate the current sensor. But this time instead of having the product exposed to an external stimulus such as the oven, this time we will have the target product control an external stimulus. Since our circuit has an RS232 interface, test code could be written in the target product that would send data out its RS232 port to control external devices. A test fixture could be created that connects the target RS232 port to a PLC that has the ability to set fixed current loads to the target product. When the test code runs, the target board would send out RS232 data that instructs the test fixture to switch in a load. Since the load is precise and constant, the

target product can easily compare the actual value read with the known value, and make adjustments accordingly. Perhaps in this case you do not want to reprogram the entire application (such as with the temperature sensor), therefore; the processor could simply fill its calibration data internally and move on as a finalized product. In this case the application code and test code co-exist in the final product.



Using self-calibration in your next project

Simplify your calibration/test processes in manufacturing with the use of a self-programming microcontroller. Most projects can be accomplished by following these 6 steps:

Load the target processor with test application code

- Power the target board, and start the processor running.
- Either the application itself controls external peripherals or instruments for information or measurements; or the target itself records with its own peripherals events of interest. These events can be stored for later interpretation.
- Exchange pertinent data with systems that calculate calibration values or perform adjustments
- Load final target application into the processor
- Load any other unique data or calibration values to target
- Make any other adjustments to calibrate system

Manufacturers of self-programming microcontrollers, such as Atmel, Microchip Technologies, Renesas, ST Micro, TI, Toshiba and others, offer design help and application notes than can help you get started. The In-System programming can be done by a variety of methods, including the ImageWriter production ISP programmer from Data I/O.

About the Author:

David Beecher has 19 years experience in hardware and software design and development, with a focus on embedded systems solutions for IC device programming. His programming solutions for manufacturing and test in the avionics and consumer products include parallel, serial, JTAG, ISP, ICP, and I2C technologies. Dave is currently directing an applications engineering team at Data I/O Corporation.