

# An Extended Delta Compression Algorithm and the Recovery of Failed Updating in Embedded Systems

Kohei Terazono (terazono@jp.fujitsu.com)  
and Yoshiyuki Okada (okada.yoshiyuki@jp.fujitsu.com)  
FUJITSU LABORATORIES LTD.

10-1 Morinosato-Wakamiya, Atsugi, Kanagawa, 243-0197, Japan

## 1. Introduction

We have developed a new program-updating method for embedded systems adopting the RISC architecture. Devices such as cell phones usually use flash memory to store the programs and if some errors exist in the programs, transmitting delta files to update them can correct them in a short time.

Previously, we developed a delta compression algorithm<sup>[1]</sup> but when the algorithm is actually implemented with devices, the transmission time is longer than the assumed time. In addition, we should assume the possibility that the power supply may be shut off during the updating process. Our new method solves these problems.

## 2. Extended Delta Compression Algorithm

Our target was creating a delta file that is 10% smaller than those currently available. In our previous paper, we noted that the regular change of operation code by correcting a RISC program is described by some commands.

Now, after close analysis, we defined seven command codes. Furthermore, we found that those codes are most likely with "COPY" the next command to appear and consequently we devised to describe in one short code combining two commands suitable for the compression. Figure 1 shows the result of the above delta compression. We successfully achieved our target value.

## 3. Recovery System

A delta file is expanded, referring to the data stored in flash memory. The new data generated as a result is overwritten. If a failure occurs during the restoring process, the sector data of the flash memory being overwritten ends up being unrecognizable data. In such a case, the restoring process is unable to be continued.

To overcome this problem, we developed a new restoring method, as shown in figure 2. A characteristic of this system is that the expanded data are stored in "nonvolatile" memory before being written into the flash memory, and the flash memory is not restored until the data for the next block has been expanded. With this system, the data was protected from the destruction by overwritten and we were able to continue the update process after a reboot.

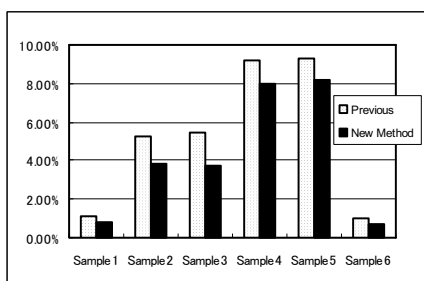


Figure 1. Results of delta compression

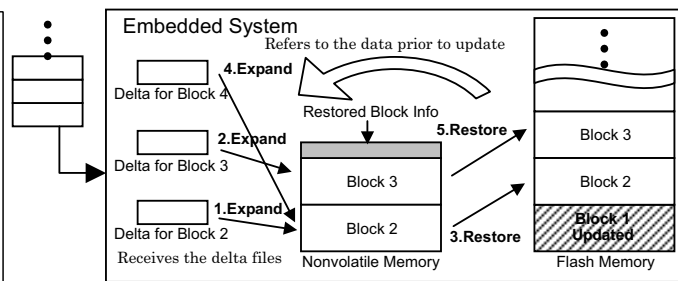


Figure 2. New restoring system

[1] Yoshiyuki Okada, Kohei Terazono; "A New Delta Compression Algorithm suitable for Program Updating in Embedded systems", DCC2003, pp.440, March 2003