TOWARDS A METHODOLOGY FOR MICROCONTROLLER SOFTWARE ENGINEERING

Antonio Bono-Nuez, Bonifacio Martín-del-Brío, and Carlos Bernal-Ruiz

Abstract

Microcontrollers or embedded processors are the electronic basic components of many control and data acquisition systems. These electronic devices have experienced a great performance increase because of the development of microelectronic technologies in last decades. For instance, the MC68HC908GP32 microcontroller from Freescale Inc. (formerly Motorola Semiconductor Products) integrates 32KB flash memory, 512B RAM, serial communication channels, A/D converter, timer, PWM generator, etc., for about 5 euros, and there are even small microcontrollers for about 1 euro. Thus, hundreds of microcontrollers are used in industrial controls, in the car (for controlling ABS systems, airbag,...), at home (home appliances, audio and video, security,...), etc.

Microcontroller system development includes hardware design and programming; both subjects are thought at microcontroller courses. Traditionally much time has been spent at teaching hardware design, and the limited time available to software development has been dedicated to teaching assembler programming. Nowadays, programming embedded processors in C language is common, and due to the large flash memory blocks integrated the current trend is to increase the software complexity. Therefore, more complex and powerful algorithms for instrumentation and control can be included (from conventional PID controls to most sophisticated algorithms based on fuzzy logic and neural networks).

Microcontroller education should adapt to these technological changes, emphasizing the software development phase, and teaching an efficient programming methodology for dealing with the software complexity. In this paper, we propose a method for embedded software development that can be called Embedded Software Engineering, parallel to programming methodologies used in computers for years.

The methodology we propose includes the following stages:

1) Analysis. It consists of modelling the system behaviour with dataflow and state transition diagrams, and data dictionaries. It can be seen as the generalization of flow diagram used in simple programs.
2) Design. Top-down design of the system based on phase 1.
3) Development. It consists of software programming.
4) Test. Exhaustive test of the software developed.
5) Documentation. Traditionally it has been a tedious and boring phase, but when the engineer has followed the previous phases the documentation has been developed step by step.

These phases are inspired on the ones applied in management of large engineering projects, and in computer software engineering, but incorporating the experience of one of the authors in developing large microcontroller based projects in the industry. In the paper all phases are explained in detail; the software development of the control of a microwave oven is used as illustration.

Expected results of the proposed methodology are software quality, working team management and step by step documentation producing.

References.