Microcomputers in Process Control

VLADIMÍR VAŠEK, PETR DOSTÁLEK, JAN DOLINAY, DAGMAR JANÁČOVÁ,
KAREL KOLOMAZNÍK
Department of Automatic Control
Faculty of Applied Informatics, Tomas Bata University in Zlin
Mostní 5139, 760 01 Zlín
CZECH REPUBLIC
vasek@fai.utb.cz, http://web.fai.utb.cz

Abstract: - This work deals with overview of the microcontrollers which are usable for the controllers based on the modern control algorithms. In this paper are described necessities of the self tuning controllers. In our applications, we are focused to the microcontrollers Motorola and National Semiconductor that are commonly used for process control. Microcontrollers were divided into two categories: 8-bit and 16-bit. In the part of the work dedicated to 8-bit microcontrollers, the properties of M68HC08 (Motorola) and COP8 (National Semiconductor) are described. The part, which deals with 16-bit microcontrollers, contains description of M68HC16 (Motorola) and CR16 (National Semiconductor). Finally has been evaluated CPU performance of selected microcontrollers in clock cycles needed for factorial computation.

Key-Words: - Microcontroller, CPU performance, 68HC08, 68HC12, COP8, CR16, adaptive - self tuning control.

1 Introduction

There is wide choice of computer systems which can be utilized for process control and automation of the technological processes ranging from powerful industrial PCs to cheaper microcomputers and programmable logic controllers. The new generation of 8-bit and 16-bit microcomputers is so powerful, that it is possible to use them for most industrial applications in monitoring and control systems. They can be advantageously used in distributed systems to solve partial control loops. In this contribution a program library focused on the monitoring and control systems will be described. The library was written for Motorola (Freescale) 68HC11 which is representative of older microcontroller generation. That is why means for using the program modules on newer microcomputers were developed. This work presents our experience with implementing standard and modern control methods on microcontrollers used in chromium recycling technology and possibilities of progress with new microcontrollers.

2 Program Library

There is necessary to solve harmony between requirements of the modern control algorithms and possibilities of the hardware and software properties microcontrollers.

2.1 Overview of the library

Although higher programming languages (such as C) are generally preferred today, it is sometimes useful or necessary to program in assembler, especially with microcontrollers. The disadvantage of assembler when compared to higher programming languages is troublesome portability of programs and also low productivity of labor. One of the ways of making assembler programming more efficient is to use a library of pre-created modules or subroutines so that the developer can just bring together existing modules instead of writing a new program from scratch.

With the help of this library developer should be able to take required modules and put them together to form a new application with minimal changes and little new code written. There have been several demands determined for the design of the modules for the library, such as easy application of each module, universality and full cooperation with real-time operating system.

The library consists of different modules which may be divided into three main categories:

- Input –output modules
- Simple discrete controllers (three-state controller, PSD controller)
- Adaptive control modules

We will describe these modules in the following sections.

2.2 Input/Output modules

These modules can be considered as independent routines, which realize reading or writing on the

microcomputer's ports. Library contains modules for handling digital input/output ports with possibility to mask unused inputs, negate selected inputs/outputs and others. A/D input module can process up to eight channels of analogue input signals using A/D converter integrated on the microcontroller.
The modules are provided as assembler source code (.ASM) files, which contain not only the code of the module but also definitions of required constants and masks (e.g. port addresses or unused-bit masks).

2.3 Simple discrete controllers

From the aspect of the program logic these controllers appear as independent subroutines called with jump-to-subroutine (JSR) instruction. However, these modules rely on some global variables such as array of measured output signals from the system etc. Required variables are included in the library and it is assumed that they will be used when creating a new application. Even when incorporating controllers into old applications there should be no problems as virtually every application contains desired data, only with different names. In such a case it is sufficient to set the identifiers required by the module to the addresses of appropriate variables in given application.

Three-state controller

The output of this controller can be in one of three working states such as heating, cooling and off. Additionally, it is possible to define dead zone and a zone where penalty is applied. The module requires 364 B of memory and run time is approximately 7800 cycles in worst case.

PSD controller

The PSD controller module exists in two variants – beside the standard version there is also module which implements Takahashi's modification of PSD algorithm. Total size of the module is 226 B and execution takes 3200 cycles.

General discrete linear controller

This controller computes the value of actuating signal u(k) according to the following formula:

$$\begin{aligned} u(k) &= q_0 e(k) + q_1 e(k-1) + \ldots + q_5 e(k-5) \\ &- r_1 u(k-1) - r_2 u(k-2) - \ldots - r_5 u(k-5) \end{aligned} \qquad (1)$$

The module requires about 379 Bytes of memory and run time is approximately 8600 cycles.

2.4 Adaptive control modules

Majority of real processes have stochastic character and their parameters can be considered constant only with some degree of incorrectness. Controllers with fixed parameters cannot respond to changes in the controlled-plant properties and the quality of control degrades. For the above reason adaptive control systems are used to improve the quality of control for hard-to-control systems.
There are several types of adaptive systems such as heuristic adaptive controllers, model reference adaptive systems or self tuning controllers. For our program library the method of self tuning controllers was used. These controllers are based on continuous estimation of controlled plant characteristics and their gradual refinement. Based on this knowledge an optimal controller is then proposed. This procedure makes it possible to react to changes in the controlled plant parameters, improve the regulation process when disturbances are present.
Basic part of a self tuning controller is the block that computes parameters of the control law (parameters of the controller). Generally it is possible to use the same methods for self-tuning controller as for synthesis of conventional controller. The only limiting factor can be the computing power demanded for the algorithm. For our library of program modules two well tried and in conventional controllers widely used methods were chosen – required model method and the pole placement.

Required model method

This method was developed for tuning of conventional controllers [1]. It allows tuning discrete or analog controller so that defined overshoot is achieved for stepwise reference or disturbance on the output of the plant. Compared to well-known Ziegler-Nichols method this method is more accurate and universal while the same simplicity is preserved. The transfer function G_R of a controller, which will ensure desired transfer function G_w of the feedback system

$$G_w = \frac{Y}{W} \qquad (2)$$

Is given as follows

$$G_R = \frac{G_w}{G_s(1-G_w)} \qquad (3)$$

This equation is based on a very general method of required model method, which is also known as compensating method. We suppose system with discrete controller, where the transfer function is as follows:

$$G_w(z) = \frac{k_{od}T}{z-1+k_{od}Tz^{-d}}z^{-d} \qquad (4)$$

$$d = \frac{T_d}{T} \qquad (5)$$

where K_{od} is the gain of open-loop system with discrete controller, T – sample period, d – discrete transfer delay. Using the procedure given in [1] we will obtain the transfer function of the controller:

$$G_R(z) = \frac{aT}{(z-1)G_S(z)}z^{-d} \qquad (6)$$

Pole placement method

Controller based on the placement of the poles of a feedback system is designed so that it stabilizes closed feedback loop whereas the characteristic polynomial has pre-defined poles.

Besides the stability it is quite easy to achieve required course of the output signal (for example maximal overshoot, damping etc.). For FB system the synthesis consists in solving the Diophantine equation

$$AFP + BQ = D \qquad (2.7)$$

Where F is the denominator of the transfer function of reference signal, $\frac{Q}{FP}$ is the transfer function of the controller and $\frac{A}{B}$ is the transfer function of the plant to be identified. D is system characteristic polynomial:

$$D(z^{-1}) = 1 + \sum_{i=1}^{n_d} d_i z^{-i}\ ,\ n_d \le 4\ ,\ nd \le 4 \qquad (2.8)$$

Parameters of the program modules

Besides the two synthesis modules the program library also contains module for recursive identification which allows calculating controlled plant parameters in real time. The recursive identification module employs recursive least squares algorithm with adaptive forgetting, which considerably limits the possibility of short-term instability of the system in the course of adaptive control. The module allows identification of first to third order Z-models.

The total size of the identification module including data is about 2.5 kB and one step of the identification takes about 110000 cycles (less than 100 ms on HC11 with 2 MHz clock rate).

Module for controller synthesis based on Required-model method needs only about 200 Bytes of memory and 11000 cycles.

The module for controller synthesis based on Pole Placement uses 1222 Bytes of memory and approx. 98000 CPU cycles to compute the controller parameters.

3. Possibility of library utilization on M68HC08 and M68HC12 MCUs

Although microcontrollers 68HC08 and 68HC11 are based on the same design as 8-bit microprocessor M6800, they are not object code compatible. This incompatibility is caused by differences in central processing unit. Microcontroller 68HC08 has only one 8-bit accumulator A (68HC11 has two accumulators A and B with possibility to merge them into one 16 bit accumulator D) and two 8-bit registers H and X, which it is possible to merge into one 16-bit index register H:X (68HC11 has two 16-bit index registers IX and IY). Program counter PC and stack pointer SP are equivalent on both microcomputers, quite difference is in location of flags in condition code register [4], [5]. Because program library and other software equipment was created in assembly language, i.e. in language specific for a given microcontroller, it is necessary to rewrite all programs for new microcontroller or find new way how to use existing software equipment. One of the possible solutions is usage of software converter that on the instruction level converts programs from 68HC11 to 68HC08 assembly language.

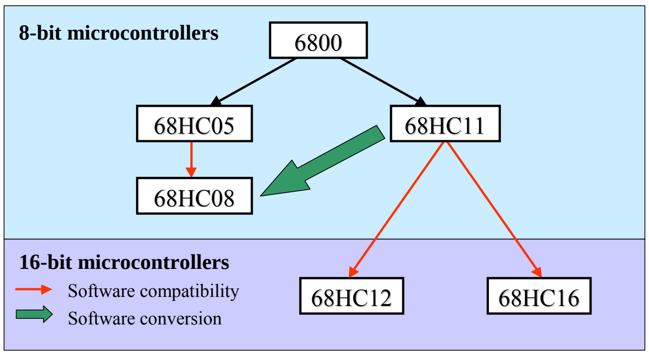


Fig. 1 Freescale 8-bit microcontrollers and their software compatibility

On the other hand core of the microcontroller 68HC12 is the CPU12, a high-speed 16-bit version of the 68HC11 architecture, which is projected to keep the compatibility with the 68HC11 at the level of the source code [3], [4]. 68HC12 fully supports all the internal registers, instructions, addressing modes and operating modes of the 68HC11. So it is possible to utilize developed program library on 68HC12 after minor program adaptation and recompilation.

3.1 Freescale 8-bit microcontrollers

Freescale currently produces 3 basic families of 8-bit microcontrollers – HC05, HC08 and HC11. Although these microcontrollers are based on the same design as 8-bit microprocessor 6800, they are not object code compatible, because their central processing units pass some optimizations. With Freescale 68HC11 are software compatible only 16-bit microcontrollers 68HC12 and 68HC16, because they were designed as a high-speed evolution of the 68HC11 architecture. Microcontroller 68HC08 is fully upward object compatible with 68HC05.

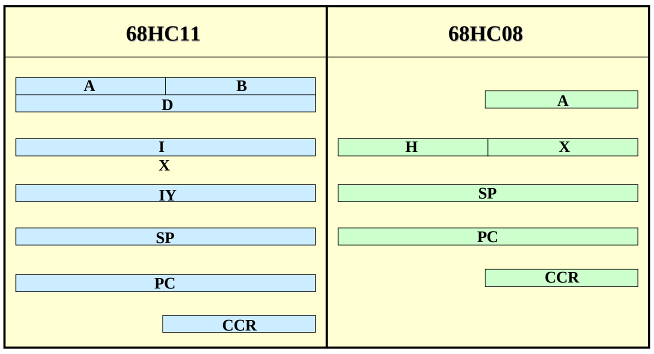


Fig. 2 Register set comparison

Table 3.1.1 compares instruction set complexity of 68HC08 and 68HC11 microcontrollers. Major differences can be seen in arithmetic and logic instructions which cause major problems with software. Programmers' model of the both microcontrollers is depicted in the figure 3.1.1.

Instruction type	68HC11	68HC08
Arithmetic	30	12
Logic	38	13
Load and store	22	10
Branch operations	25	30
Initial state presetting	11	8
Stack manipulations	8	6
CPU control	5	3

Tab. 1 Instruction set complexity comparison

Freescale 68HC08 microcontrollers

The M68HC08 is new Freescale's 8-bit industry standard flash-based microcontroller with Von-Neumann architecture. Central processor unit is fully upward compatible with the 68HC05 family. Their high performance architecture is optimized for C language compilers. On the chip are integrated all basic peripherals such as:

- Timer interface module with input capture and output compare functions
- Serial communication interface (SCI)
- Serial peripheral interface (SPI)
- CAN module
- 8-bit A/D converter with analog multiplexer
- On-chip FLASH memory with in-circuit programming capability
- Byte-erasable EEPROM memory
- Watchdog system
- Clock monitor and low voltage inhibit functions [2], [3].

Freescale 68HC11 microcontrollers

The MC68HC11 are an advanced 8-bit microcontrollers with highly sophisticated, on chip peripheral capabilities. The fully static design allows operation at frequencies down to dc, further reducing power consumption. Central processor unit is fully upward compatible with the microprocessors M6800 and M6801. On the chip are integrated following peripherals:

- Byte Erasable EEPROM Memory
- Expanded Bus Memory Interfaces
- Serial Peripheral Interface (SPI)
- Serial Communications Interface (SCI)
- 8-bit A/D converter with analog multiplexer
- Timer interface module with input capture and output compare functions
- Watchdog system
- Clock monitor and low voltage inhibit functions [4].

3.2 Software conversion process

Conversion is process of instruction set transformation, at which individual instructions of M68HC11 microcontroller are replaced by instructions of M68HC08 microcontroller with condition, that new block of instructions must fully functionally substitute original instruction. The aim is to create software tool that will perform transformation of source code of M68HC11 in the format of compiler AS11 to source code of M68HC08 in the format of compiler CASM08Z, which is part of integrated development environment WinIDE.

The aim of the software tool for program conversion is to transform assembly language source files of

M68HC11 in the format of compiler AS11 to the source files, which can be assembled using the compiler CASM08Z and subsequently executed on M68HC08 microcontrollers with the same results.

Description of operation

Converter reads source file line by line. Every line is analyzed and subsequently partitioned into label, instruction, operand and commentary. If instruction has an operand, converter must determine according to the format of the operand the address mode. Now converter knows all parameters of instruction and starts to search it in an instruction dictionary. If instruction is found, the converter begins to generate its replacement pursuant to the data from the instruction dictionary. Otherwise the converter stops with listing that reports line number, where the error was found. After generating entire alternate code that corresponds to one instruction, converter starts to process next program line. Whole cycle is repeated, until the end of file is identified.
In the conversion stage individual instructions of M68HC11 are converted to the macros or subroutines, which completely substitute their function on M68HC08. Optimization stage on the basis of the converted code analysis performs virtual registers usage a branch optimizations. Principle of the converter operation is obvious from figure 3.2.1.

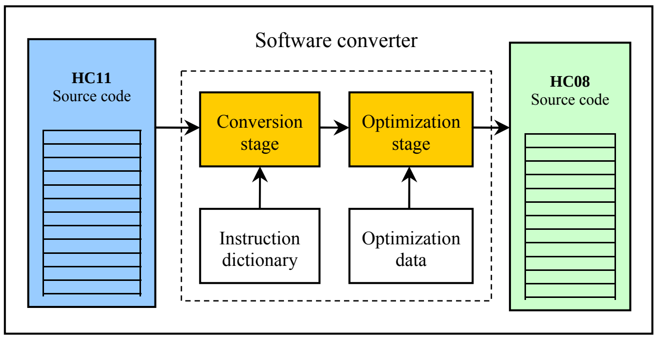
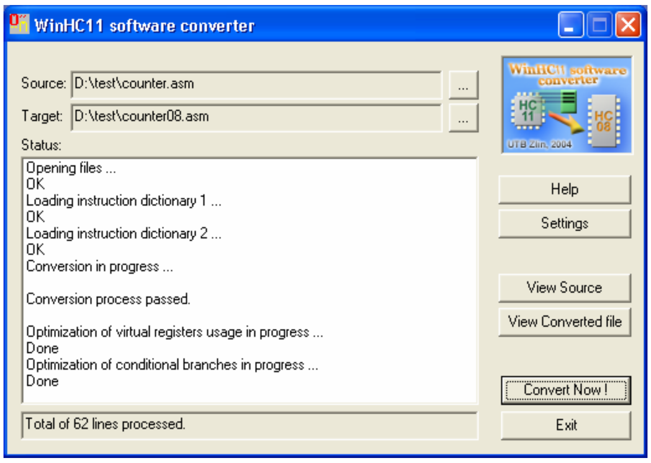


Fig. 3 Principle of operation

Program description

On the basis of instruction set analysis of both microcontrollers software tool WinHC11 was created. In the figure 3.2.2 is main window of the software converter program, which appears after startup. During initialization process default configuration is automatically loaded and program is ready for operation. Conversion process can be initiated, after selection of source (supported are files in the form of AS11 compiler) and target file, by pressing of "Convert Now" button. Actual processed line, status and error messages during software conversion process are displayed in the status window. Source and converted file can be viewed by pressing buttons "View Source" and "View converted file".


WinHC11 software converter

Source: D:\test\counter.asm
Target: D:\test\counter08.asm
Status:

```
Opening files ...
OK
Loading instruction dictionary 1 ...
OK
Loading instruction dictionary 2 ...
OK
Conversion in progress ...

Conversion process passed.

Optimization of virtual registers usage in progress ...
Done
Optimization of conditional branches in progress ...
Done
```

Total of 62 lines processed.

Fig. 4 Software converter – main window

3.3 Experimental verification

Due to high complexity of M68HC11 instruction set there is necessary to thoroughly test correct function of instruction replacements and software converter as well. Verification was performed in two main stages. First stage was focused on correct function of individual instruction replacements. Finally were converted larger programs that produce results that can be easily verified. For this purpose are ideal programs working with math operations. After all tests passed program modules for automatic control for Freescale 68HC11 microcontrollers was converted.
Number of machine cycles necessary for processing appropriate services and correct results were evaluated using the simulator ICS08Z from P&E Microcomputer Systems. Relative speed in table 3.3.1 gives comparison with original speed on M68HC11.

	68HC08 at 8MHz bus clock		
Service	**NT_{08}**	**t_{08} [μs]**	**Rel. speed [%]**
Ident	4157000	519625	+38,4
PSD1	296000	37000	+40,5
PSD2	304200	38025	+40,7
Poleplac	245300	30662,5	+44,8
Robust	395400	49425	+39,2
Invdyn	20500	2562,5	+42,0
Polreg	482800	60350	+44,2
Obecreg	627000	78375	+38,4

Tab. 2 Instruction set complexity comparison

Description of symbols:

- NT_{11} – machine cycles count on M68HC11
- NT_{08} – machine cycles count on M68HC08

- t_{11} – processing time on M68HC11
- t_{08} – processing time on M68HC08

4. Verification of the Control modules

To verify functionality of the library several experimental programs were created and tested on real system in laboratory for recycling chromium from tannery waste. The experimental programs included program for recursive process identification and two adaptive controllers: one based on pole placement and one based on required model method. The result of experiments with these programs proved that the library modules are working properly. The following figures show the results. Figure 4.1 depicts the changes of the parameters of the controlled plant as obtained by the recursive identification module with model of 2nd order system.

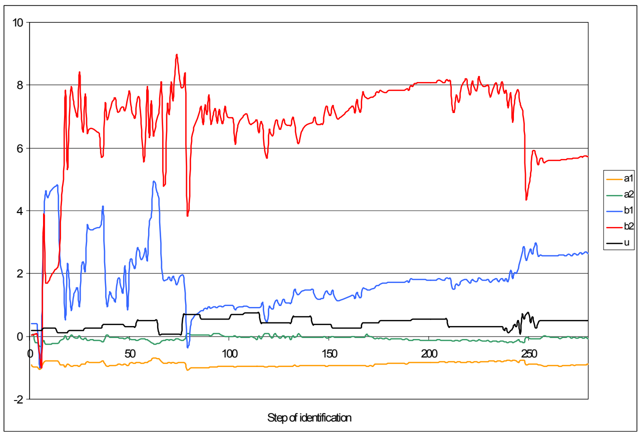


Fig. 5 Identified parameters of the controlled plant. On x-axis is number of steps (sample periods).

In the figure 4.2 result of the adaptive controller module based on required model method can be seen. Figure 4.3 shows the result of the second adaptive controller module, which uses pole placement synthesis.

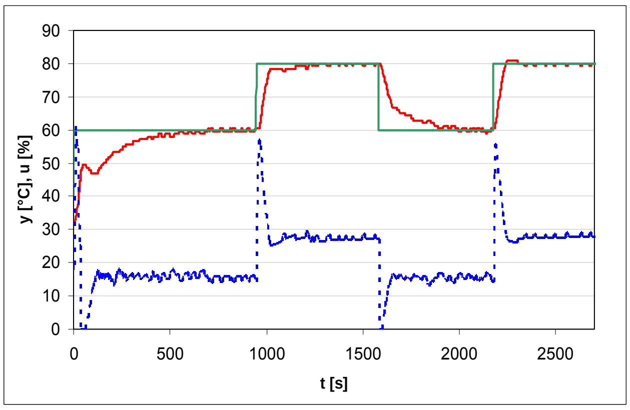


Fig. 6 Control with adaptive controller with required method synthesis, T = 6 s, T_w = 27 s.

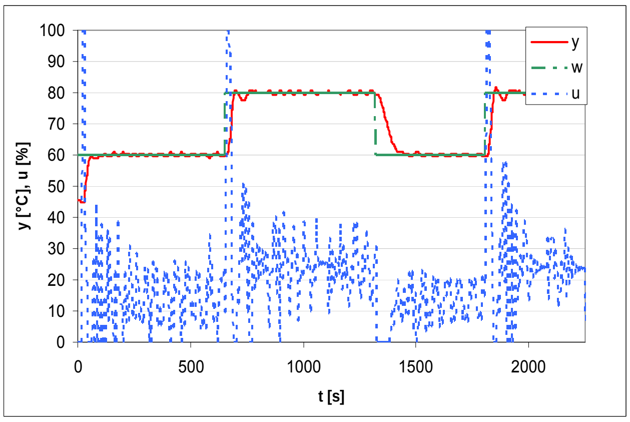


Fig. 7 Control with adaptive controller with pole placement synthesis, T=6 s, poles d_1 = -0,5; d_2 = -0,2.

5. Conclusion

To help with developing applications of Freescale 68HC11 microcontroller a program library has been created. Modules cover the area of automatic control from basic input/outputs through discrete controllers to adaptive control. To verify the functionality of the library several applications were assembled and used for controlling a real system and proved to be functional. Also methodology has been developed to convert program modules made for the HC11 microcomputers to a new hardware platform represented by the modern families of HC08 and HC12 microcomputers. A conversion programs has been compiled and proved for formal adaptations of the source texts.

Acknowledgement

This work was supported by research project MSM 7088352102. This support is very gratefully acknowledged.

References:

[1] M. VITECKOVA, Controller tuning with required model method. Technical University in Ostrava, 2000.

[2] Freescale Semiconductor. CPU08 Central Processor Unit Reference Manual., 2001. Available from WWW: <www.freescale.com>

[3] Freescale Semiconductor. HCS08 Family Reference Manual, Rev.1., 2003. Available from WWW: <www.freescale.com>

[4] Motorola. MC68HC11A8 HCMOS single-chip Microcontroller, 1996. Available from WWW: <www.freescale.com>

[5] Motorola. HC11 Reference Manual, Motorola M68HC11RM/AD, 2000. Available from WWW: <www.freescale.com>