Microcontroller-based multiple-platform PWM signal generation procedures for industrial use

Bogdan CROITORU, Adrian TULBURE
Department of "Precise and Engineering Sciences", “1 Decembrie 1918” University of Alba Iulia, Alba Iulia, ALBA County, Romania,
E-mail: cbsm_ab@yahoo.com, tulbureadrian@yahoo.de

Mihail ABRUDEAN
Department of Automation, Technical University of Cluj-Napoca, Cluj-Napoca, CLUJ County, Romania,
E-mail: Mihai.Abrudean@aut.utcluj.ro

Abstract—This contribution describes three procedures and the related software operations which are used to obtain the pulse-width modulation (PWM) signal for industrial use. The hardware platforms used to generate the signals are embedded systems, manufactured in recent years. Three types of microcontrollers based on their register architecture were used: 8-bit ATMEGA 328P, 16-bit SAB80C167 and 32-bit PIC32MX320F128H. PWM is a preferred way for control in modern semiconductor devices. The very short rising and falling time ensure the minimum of switching transition time respectively of the switching losses. In this context, the paper will analyze comparatively a new structure, related on a classical reference structure, both the hardware and the software. At the end of the paper some conclusions might be drawn on the code length, system latency and the technical limitations of their usual or rated performance.

Keywords—PWM; microcontroller; signal; software; generate; timer; embedded; system; register

I. INTRODUCTION

In the context of increasingly industrialization and automation, the need of intelligent devices capable of driving and controlling a wide range of electrical and electro-mechanical devices becomes a great challenge [1], [2].

Any modern factory or facility is full of all types of intelligent machines (ex. mobile robots, robotic arms and so on). These machines or special devices may contain one or more microcontrollers / microprocessors capable of controlling them.

Different types of AC motors, DC motors (servo-motors, stepper-motors and so on), switching-on power supplies, power amplifiers (especially class D power amplifiers) use the PWM signal to be controlled. These PWM signals are usually generated by special microcontrollers [3], [4], [5], [6], [7].

There are many types of microcontrollers with a wide range of features. One of these features is the PWM function on one or more digital pins. The end-user is able to manipulate the characteristics of a PWM signal by accessing the special PWM registers (ex. Timer Registers, Timer Counter / Control Registers, Output Compare Registers) [4], [8], [9], [10].

Modifying the values of the PWM registers is leading to modifications such as:

• Frequency of a PWM signal;
• The duty-cycle value;
• The type of the PWM generation mode (ex. Fast PWM, Phase-Correct PWM);
• The pre-scale factor value (ex. 1:1, 1:8, 1:256 and so on);
• The bit length of the timers (ex. 8 bits timers, 16 bits timers, 32 bits timers);
• The bit length of the PWM frequency range (ex. 10 bit PWM, 11 bit PWM and so on).

In this context the present paper is aimed to describe the process of generating some PWM signals and analyzing few aspects like:

• Differences in using various microcontroller register architectures (8 bits, 16 bits and 32 bits);
• The ease of access for generating the PWM signals by accessing the PWM registers;
• Boot-up / reset timings until the PWM signal generation is active (depends on the values of R and C of the reset mechanism);
• Amplitude spectrum analysis for the PWM signals and highlighting the presence of parasite signals / noise (rectangular FFT generated by a digital phosphor oscilloscope);
• Source-code involved in the process of generating the PWM signals [1], [2], [3], [4].

In the experiments were used three different frequencies and two different duty-cycle values (Frequencies: 500 Hz, 1 KHz, 2 KHz; Duty-cycle values: 50%, 90%) [10].

Three types of microcontrollers have also been used: 8-bit ATMEGA 328P; 16-bit SIEMENS SAB80C167, 32-bit PIC32MX320F128H.
The steps and the results will be presented in the corresponding section bellow.

II. RELATED WORK

An intense work has been conducted in the field of embedded systems, especially regarding the generation and use of PWM signals in industrial and non-industrial environments by using microcontroller systems.

In the field of robot control / robot movement through PWM techniques a suggestive work is described in [3], [5], [6]. Each of the mobile robots presented in the above papers are using DC motors to produce motion.

The speed and steering are PWM controlled by a type of microcontroller. A motor driver is used in [3], [5], [6] and typically is a H-Bridge system.

In [4] is presented a model for controlling an induction motor drive using PWM techniques provided by a PLC system.

In [7] are presented and treated aspects regarding power converters. The aspects consists in: some mathematical analysis of power converters, converter topologies, modern control strategies for power converters, including PWM modulation techniques.

III. IMPLEMENTATION AND RESULTS

In this section of the paper is presented the experimental work for generating and analyzing the PWM signals. All three microcontrollers used are the core of three embedded development boards: ATMEGA328P is used with ARDUINO UNO board; SIEMENS SAB80C167 is used with EVA167 board; PIC32MX320F128H is used with DIGILENT CHIPKIT UNO32 board.

A brief technical description is made bellow about each microcontroller used, especially regarding timers and PWM techniques. More detailed information about hardware, software and other technical aspects of the microcontrollers can be found inside each datasheet of them.

It is also presented and analyzed the source-code involved in the process of custom generation of PWM signals for each microcontroller. An analysis is performed for the time delay occurred when the microcontrollers are rebooted (the rebooting time is influenced by the R and C values of the reset mechanism) [8], [9]. Another analysis is performed on the generated PWM signal and a rectangular FFT is displayed in the pictures bellow were some similarities / differences (presence of parasite noise) can be noticed.

A calibrated “TEKTRONIX DPO4104 Digital Phosphor Oscilloscope (1 GHz, 5 GS/s)” was used to perform the signal analysis.

A. Brief technical description of microcontrollers

1) ATMEGA 328P

The first microcontroller used in the experiments is an 8-bit ATMEGA 328P microcontroller. It runs at 16 MHz clock frequency, 32 KB flash memory, 2 KB SRAM memory.

It has three timers (Timer 0, Timer 1, and Timer 2). Timer 0 and Timer 2 are 8-bit timers. Timer 1 is 16-bit timer.

Digital pins 3 and 11 are assigned to Timer 2. Digital pins 5 and 6 are assigned to Timer 0.

Digital pins 9 and 10 are assigned to Timer 1. These 6 pins have PWM capabilities.

Each of these timers has two Output Compare Registers (OCR) which are used to control the PWM width.

2) SIEMENS SAB80C167

The second microcontroller used is a SIEMENS SAB80C167, which is an 16-bit microcontroller. It runs at 20 MHz frequency.

It is also equipped with 2 kB On-Chip Internal RAM (IRAM) and 2 kB On-Chip Extension RAM (XRAM). It is capable of 100 ns Instruction Cycle Time at 20 MHz CPU Clock.

The on-board integrated PWM unit consists in four PWM channels, which allows the generation of up to 4 independent patterns.

Each PWM-channel possess the following elements: a period register PPx, up/down counter PTx, pulse width register PWx, 16-bit shadow latch, two comparators, and the corresponding control logic /3/, /5/, /6/.

3) MICROCHIP PIC32MX320F128H

The third microcontroller used is a 32-bit PIC32MX320F128H. It features: 80 MHz frequency (this frequency is customizable for different tasks by changing the values of the pre-scale factor), 16KB SRAM, 128KB FLASH and numerous communication protocols and I/O ports. It also features five 16-bit timers.

Timer 1 is a synchronous/asynchronous 16-bit timer that can operate as a free-running interval timer for various timing applications and counting external events.

Timer 2 with Timer 3 and Timer 4 with Timer 5 can be combined to obtain two synchronous 32-bit timers. The 32-bit timers can operate in three modes:

• Synchronous Internal 32-bit Timer
• Synchronous Internal 32-bit Gated Timer
• Synchronous External 32-bit Timer

B. Implementation and results

To generate PWM signals with different characteristics, few software applications (standard C) were built. Each application is customized for each microcontroller [9], [10].

The goal was to generate three PWM signals with three different frequencies and a duty-cycle value of 50% and 90% on each microcontroller [1], [2], [3], [5], [6].
In the beginning, in Fig. 1 is presented the Evaluation Kit “EVA167” which contains the 16-bit SIEMENS SAB80C167 microcontroller. This board is a very simple microcontroller development board which is widely used in various automation processes. It is also a good training embedded platform for learning hardware and software aspects of microcontrollers.

In Fig. 2 and Fig. 3 are presented two fragments of the standard C application which generates the desired PWM signals on PORT 3 of the “EVA167” microcontroller board. The PWM frequencies and duty-cycle values are illustrated in Fig. 2. The GPT1 (General Purpose Timer Module 1) settings are made in Fig. 3, where the PORT 3 is enabled as OUTPUT in order to generate the desired PWM signals. The PWM frequency is 1 KHz and the duty-cycle value is 90% (Fig. 2).

In Fig. 4 is presented the C/C++ source-code for generating a PWM signal with three different frequencies. Only one frequency is active at a time. The source-code is adapted for the PIC32MX320F128H microcontroller.

It is written in the Multi-Platform Integrated Development Environment (MPIDE) which is an open-source embedded software development platform for both PIC and ATMELE microcontrollers (for only a small group of microcontrollers).

Digital pin 5 (Fig. 4) is set as an OUTPUT and the function “analogWrite()” will generate a PWM signal on this pin with a duty-cycle value of 50% (128) and 1 KHz frequency.

In both Fig. 4 and Fig. 5 the unsigned integer variable “PWM_TIMER_PERIOD” is bound to PR2 register (Period Register of Timer 2) and Timer 2 (16-bit timer). This value is used in the OCMP module (in both OC2R and OC2RS registers).

“PWM_TIMER_PERIOD” variable value is responsible for the PWM frequency generation. The OC2RS register is holding the duty-cycle value of the PWM signal.

The pre-scale factor value is set to 1:256 and for generating the other frequencies (500 Hz and 2 KHz) this value is not modified. The only parameter which is changed is the “PWM_TIMER_PERIOD” variable.

In particular, Fig. 5 represents a fragment from “WIRING_ANALOG.C” file where the “analogWrite()” function is defined. The “PWM_TIMER_PERIOD” variable was initially defined as a static value, but for more flexibility and ease of use it was modified to be defined and initialized in the user’s application (“extern unsigned int PWM_TIMER_PERIOD”).

Instead of working directly with Timer 2 and PWM registers, it was preferred a much easier way to achieve the desired frequency and generation mode of the PWM signal [8].
For 1 KHz frequency the "PWM_TIMER_PERIOD" value was set to 311, for 2 KHz frequency the "PWM_TIMER_PERIOD" value was set to 155 and for 500 Hz frequency the value of the "PWM_TIMER_PERIOD" was set to 624. The "analogWrite()" function works as Fast-PWM mode. The second parameter of this function is the value of the OC2RS register which is part of the OCMP module.

In Fig. 6, Fig. 7 and Fig. 8 is presented the C/C++ source-code for generating a PWM signal of 500 Hz, 1 KHz and 2 KHz frequency and 50% duty-cycle using the 8-bit ATMEGA 328P microcontroller [10].

```c
void setup()
{
  pinMode(3, OUTPUT);
  pinMode(9, OUTPUT);
  TCCR1A = 0;
  TCCR1B = 0;
  // The general PWM register settings for 1 KHz and 500 Hz...
  // ATMEGA 328P -> 8-bit microcontroller
  TCCR1A = _BV(COM1A1) | _BV(COM1B1) | _EV(OC1M11) | _EV(OC1M10);
  TCCR1B = _BV(WGM13) | _EV(OC1M1) | _EV(OC1M0);
  OCR1A = 248; // 500 Hz frequency = 1 kHz
  OCR1B = 128; // duty-cycle value = 50%
}
void loop()
{
  ...}
```

Fig. 6 C/C++ Source-code for generating a PWM signal of 1 KHz frequency and 50% duty-cycle using the 8-bit ATMEGA 328P microcontroller

```c
void setup()
{
  pinMode(10, OUTPUT);
  pinMode(5, OUTPUT);
  TCCR1A = 0;
  TCCR1B = 0;
  // The general PWM register settings for 2 KHz...
  // ATMEGA 328P -> 8-bit microcontroller
  TCCR1A = _BV(COM1A1) | _BV(COM1B1) | _EV(OC1M11) | _EV(OC1M10);
  TCCR1B = _BV(WGM13) | _EV(OC1M1) | _EV(OC1M0);
  OCR1A = 999; // 2 KHz frequency = 2 KHz
  OCR1B = 127; // duty-cycle value = 50%
}
void loop()
{
  ...}
```

Fig. 7 C/C++ Source-code for generating a PWM signal of 2 KHz frequency and 50% duty-cycle using the 8-bit ATMEGA 328P microcontroller

```c
void setup()
{
  pinMode(10, OUTPUT);
  pinMode(5, OUTPUT);
  TCCR1A = 0;
  TCCR1B = 0;
  // The general PWM register settings for 500 Hz...
  // ATMEGA 328P -> 8-bit microcontroller
  TCCR1A = _BV(COM1A1) | _BV(COM1B1) | _EV(OC1M11) | _EV(OC1M10);
  TCCR1B = _BV(WGM13) | _EV(OC1M1) | _EV(OC1M0);
  OCR1A = 499; // 500 Hz frequency = 500 Hz
  OCR1B = 127; // duty-cycle value = 50%
}
void loop()
{
  ...}
```

Fig. 8 C/C++ Source-code for generating a PWM signal of 500 Hz frequency and 50% duty-cycle using the 8-bit ATMEGA 328P microcontroller

The chosen approach for generating the desired PWM signal on the 8-bit ATMEGA 328P microcontroller was to directly tune/modify the values of the TIMER 1 registers.

It was a much more convenient method, rather than using the "analogWrite()" function, because of different structure of the microcontroller. Digital pins 9 and 10 are bound to TIMER 1 and used to generate the PWM signals on the ATMEGA 328P microcontroller.

The Waveform Generation Mode bits (WGM) are set for fast PWM with OCR1A register (Output Compare Module has two registers: OCR1A and OCR1B) controlling the top limit. The OCR1B is controlling the duty-cycle value. OCR1A’s mode is set to "Toggle on Compare Match".

OCR1A value is set to 248 (Fig. 6) for a 1 KHz frequency and OCR1B is set to a value of 128 (50% duty-cycle). For 2 KHz frequency the OCR1A value is 999 (Fig. 7) and for 500 Hz frequency the OCR1A value is 499 (Fig. 8).

Because each microcontroller platform is structurally different (regarding many hardware and software aspects), the size (byte length) of the final program ready to be uploaded in its FLASH memory has variable length [8], [9], [10].

In Fig. 9 and Fig. 10 is displayed the final binary sketch size for two microcontrollers (ATMEGA 328P and PIC32MX320F128H) after compilation. The final length of the binary sketch is a sum of variables, which include [4], [5], [6]:

- the architecture, complexity and structure of the microcontroller;
- the structure of the software libraries;
- the complexity of the software libraries used in the programming processes;
- the number of imported libraries and software functions used in the current project;
- the level of source-code optimization.

Even if each microcontroller used in the experiments is generating the same types of PWM signals, the final binary...
sketch files (created in the final step of the UPLOADING process) used have different sizes.

For a simple PWM signal generation the binary sketch size for the ATMEGA 328P is 738 bytes length and for the PIC32MX320F128H is 6772 bytes length. There is a big length difference between the binary sketch files of an 8-bit microcontroller and a 32-bit microcontroller.

The compilation and uploading time is different for each microcontroller. The longest compilation and uploading time (about 25 seconds) was measured for the PIC32MX320F128H microcontroller.

Three oscilloscope screen captures are presented in Fig. 11, Fig. 12 and Fig. 13 for each type of microcontroller used. For the screen capture it was selected a 1 KHz PWM signal with both 50% and 90% duty-cycle.

In Fig. 11 and Fig. 12 there are also presented the amplitude spectrum components of the signal. It is focused on the 1 KHz harmonic (the fundamental harmonic) with an amplitude value of 1.478 V for ATMEGA 328P and 1.442 V for PIC32MX320F128H. The TTL logic level is 3.3 V.

Another step was performed and it was analyzed the reboot / reset time of each microcontroller. The reset time of each microcontroller is influenced by the values of R and C involved in the reset process.

In Fig. 14 and Fig. 15 are displayed two time-delay measurements on the ATMEGA 328P and PIC32MX320F128H regarding the reset / reboot process.

A total time of 4.93 seconds (the reset capacitors have higher values) was measured when a reset command was received by the PIC32MX320F128H microcontroller (Fig. 14).

A total time of 1.99 seconds (values of reset capacitors are low) was measured when a reset command was received by the ATMEGA 328P microcontroller (Fig. 15).

For the SIEMENS SAB80C167 it was measured a reset time of 3.54 seconds (+/- 0.05 seconds accuracy).

The goal of these measurements was to highlight the initializing time of the microcontrollers (influenced by R and C...
values), because in critical processes were it might be involved, this reset time is important.

If a power fluctuation occurs, these microcontrollers might reboot automatically and the reboot time is closely related to the R and C values used in the reset mechanism.

Excluding the manual reboot, another automatic reboot possibility is when a software failure / error occurs, or when a some sort of short-circuit happens.

If a reboot occurs and depending on the industry process architecture / structure, the initializing time (few seconds) of the microcontrollers could destabilize the processes which they are controlling. The values of the reset capacitors should be chosen carefully (lower values were possible) in order to reduce as much as possible all time delays in the reset process. Some special industry processes might request PWM signals with very less noise. Noise-free PWM signals are difficult to be produced [9], [10].

In the present experiments all the generated signals have some noise (parasite frequencies), which in some types of process control might be a problem. Parasite signals (noise) could be introduced by oscilloscope probes, or by cables, or by faulty wire contacts, environmental parameters, etcetera [4], [5], [6], [7], [8].

IV. CONCLUSIONS AND FUTURE WORK

The purpose of the this paper was to present three procedures and the related software operations which are used to obtain the pulse-width modulation (PWM) signal for industrial use by using three different microcontrollers.

Each PWM generation procedure is simple and 100% safe, so the main goal of the paper was achieved. Depending on the industry process where the microcontrollers are used, each of the presented microcontroller architectures can be easily used.

For simple PWM control of some small-sized DC motors, the 8-bit microcontroller has the advantage of quickly programming and fast response.

For some advanced PWM techniques and additional processing functions for a much more complex automation process control, the use of an 16-bit, or 32-bit microcontroller is a good choice. Depending on the industry process type, the presence of parasite signals (noise) could be a problem.

For future research, all three microcontrollers presented in this paper will be interfaced with a MOSFET driver and a thyristor bench test. This will lead to some experimental analysis of total harmonic distortion in the both control and the power signal with interpretation of practical results.

REFERENCES