A MSP430 Microcontroller Simulator for Teaching at University during the Covid-19 Pandemic

Septimiu Mischie
Faculty of Electronics, Telecommunications and Information Technologies, Politehnica University
Timisoara, Romania
septimiu.mischie@upt.ro

Gabriel Vasiu
Vitesco Technologies
Timisoara
Romania
gabriel.vasiu@vitesco.com

Robert Pazsitka
Faculty of Electronics, Telecommunications and Information Technologies, Politehnica
University Timisoara, Romania
robert.pazsitka@upt.ro

***Abstract*—Teaching microcontrollers at university during face to face activities implies using development boards such as those from Texas Instruments or Microchip. Changing to online teaching due to the coronavirus pandemic needs free simulators to be used by students to achieve the same level of teaching. The paper proposes an example of a free simulator designed for Texas Instruments MSP430G2553 microcontroller.**

Keywords—microcontroller, simulator, LabVIEW, MathScript Node, timer, LED, oscilloscope

I. INTRODUCTION

Teaching microcontrollers at university [1-3] requires software components such as Integrating Development Environment (IDE) as well as hardware resources such as development boards, LED's, sensors and electronic instruments as oscilloscopes and signal generators. As the coronavirus has started during the first part of the second semester of the academic year 2019-2020 and all the activities were switched to online teaching we needed to make some changes to be able to perform our laboratory classes. Thus, the only solution was using a simulator, so that students could work on their own from home. One of the most used simulators from the microcontroller field is Proteus [4-7]. The demo version that is free has some projects but their hardware structure cannot be changed. Only the corresponding software can be changed. The proposed projects do not contain any example with Analog to Digital Converter (ADC) which is one of the most important module of a microcontroller. The full version of Proteus is expensive and difficult to use by students. So far we have not identified another similar simulator but cheaper or with a student version.

Thus, in order to overcome these disadvantages, our group has implemented our own simulator for Texas Instruments MSP430G2553 microcontroller [8]. This is based on LabVIEW software package and can be used by any student. The rest of the paper is organized as follows. Section II presents the structure and facilities of the proposed simulator. Section III presents some details about the internal structure of the simulator. Section IV presents the most relevant applications and the last section concludes the paper.

II. PRESENTATION OF THE PROPOSED SIMULATOR

The proposed simulator is designed for MSP430G 2553 microcontroller and can be started by running of a suitable application file and its graphical interface is presented in Fig. 1. In order to use this simulator the hex file that corresponds to the desired application must be first generated using IAR Embedded Workbench [9], where the C source file is edited and compiled. This file should then be loaded using the browse instrument from the application- path to HEX file- and then it is displayed in the corresponding window-Content of the HEX file, Fig. 1.

The most important resources of the simulator are the following, Fig.1:

- four LEDs, LED 1 to LED 4 that can be connected to any of the eight pins of the port P1 by the corresponding pop-up selectors;

- the LEDs D1 and D2 that are connected to the P1.0 and, respectively, P1.6 pins of port P1 and the push button S1/P1.3 that is connected to the P1.3 pin of the same port; these three elements are similar to those of MSP-EXP430GET development board [10];

- the push button P1.2 that is connected to the P1.2 pin;

- the content of the CPU registers R0-R15

- a square signal generator having a variable duty cycle that can be connected to the one of the pins having input capture function, that are P1.1, P1.2, P1.5 and P1.6;

- a potentiometer, ADC10 Input Voltage, that can generate a DC voltage between 0 and 3.3 V; this voltage can be applied to one of the pins having the analog input function, that are all the port P1 pins.

- a temperature input field that simulates the temperature corresponding to the internal sensor of ADC10 module of MSP430G2553 microcontroller; thus the conversion results are internally computed depending on this temperature according to the corresponding expression [11].

- the content of the RAM and Flash memories;

- a 2x16 LCD display

- the content of Watchdog Timer (WDT+) Counter

- an oscilloscope that can be connected to any pins of Port P1; it also has some special functions for input capture and output compare capabilities of CCR0 and CCR1 units of Timer_A3 module of MSP430G2553 microcontroller; thus, the interest signals and also some auxiliary waveforms can be displayed to help achieve a better understanding of the respective capabilities.

The microcontroller program can be executed step by step, direct, can be stopped or can be initialized, using the four controls from the top right corner, Fig.1.

III. INTERNAL STRUCTURE OF THE SIMULATOR

The interface of the simulator is implemented in LabVIEW. In order to start it, it is not necessary to have LabVIEW installed on the PC.

2021 International Conference on Applied Electronics (AE) | 978-80-261-0973-0/21/$31.00 ©202110.23919/AE51540.2021.9542897

ISBN 978-80-261-0973-0, © University of West Bohemia, 2021

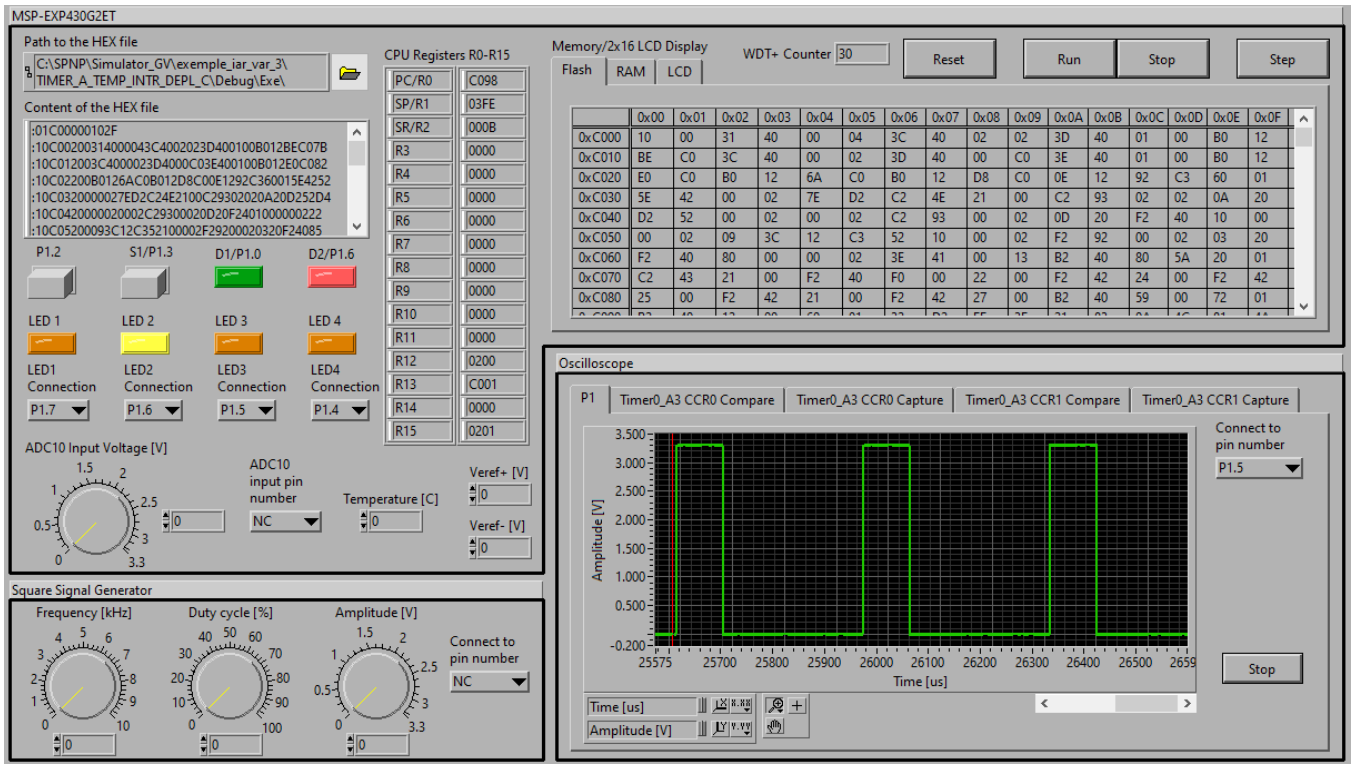

Fig. 1. The graphical interface of MSP430 Simulator

A simplified version of the LabVIEW Block Diagram is presented in Fig.2. Most components of the diagram contain a software part, which is implemented in MathScript Node.

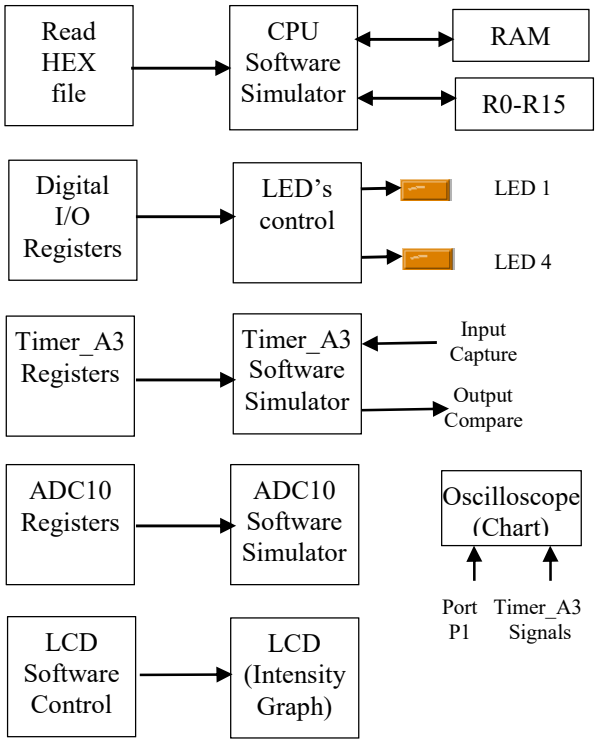

Fig. 2. The simplified LabVIEW Block Diagram of the MSP430 Simulator

The CPU Software Simulator block implements instructions such as decoding and execution. MSP430G2553 microcontroller has an instruction set of 27 instructions. Each instruction has a length of 1 to 3 words. The first word contains the operation code and some information about the source and destination operands and addressing modes. The other two words contain the operands. TABLE I presents two examples. Thus, the most significant four bits of the first word determine the instruction as follows: 5 means ADD and 4 means MOV.

TABLE I. ASSEMBLY INSTRUCTIONS AND THEIR CODE

Assembly instruction	W1	W2	W3
ADD #4D9h, R7	5037h	04D9h	
MOV #4D9h,&214h	40B2h	04D9h	0214h

The CPU block reads the first word W1 of each instruction. Using W1 it allows decoding, that means determines the name of the instruction (MOV, ADD,...). Then CPU block calls a MathScript Node function to implement the execution of the corresponding instruction. The format of the call is as following:

[Flash,RAM,R0_15]=inst_name(W1,Flash,RAM,R0_15),

where Flash, RAM and R0_15 represents the arguments that can be changed by executing the instruction. W1 is used by the instruction to extract the operands.

Execution of the current instruction can change locations in RAM peripherals area where are the peripheral registers are located. These memory locations are inputs to ADC10 or Timer_A Software Simulator blocks. The program that is

running in each of these blocks simulates the functionality of the corresponding peripheral according to the inputs.

In Fig.3 a part of the program that implements the Timer_A is presented.

```
if(mcx==0)                    %Stop?
   dif_tar=0;
  end
  if((mcx==1)||(mcx==2))   %Up or Continuous?
   dif_tar=1;
  end
%-------------------------------------------------------
  ta0r=ta0r+dif_tar;          %change the TAR
%-------------------------------------------------------
  if(mcx==1)                   % Up?
   if(ta0r==(ta0ccr0+1))
    ta0r=0;
   end
   if(ta0r==ta0ccr0)
    cc1ifg=1;
   end
  end
%-------------------------------------------------------
  if(mcx==2)                   % Continuous?
   if(ta0r==65536)
    ta0r=0;
   end
  end
%-------------------------------------------------------
  if(mcx==3)                   %Up-Down?
   if(ta0r==ta0ccr0)
    dif_tar=-1;
   end
   if(ta0r==0)
    dif_tar=1;
   end
   if(ta0r==ta0ccr0)
    cc1ifg=1;
   end
end
```

Fig. 3. A part of the program that implements the Timer

where:

- mcx is a 2-bit field of the TACTL register of Timer_A having the significance as in TABLE II.

TABLE II. THE SIGNIFICANCE OF MCX BITS

mcx	Significance
00	Stop, the timer is halted
01	Up, the timer counts up to TACCR0 register
10	Continuous, the timer counts up to 0xFFFF
11	Up-down, the timer counts up to TAACR0 register and then down to 0

- ta0r represents the timer/counter register of Timer_A

- cc1ifg is an interrupt flag that is set when ta0r equals the content of ta0ccr0 register.

The four LEDs, LED1 to LED4 are controlled depending on Digital I/O registers by means of LED's control block that contains different controls elements.

The oscilloscope is a Chart block while the 2x16 LCD is an Intensity Graph block that is controlled by the LCD Software Control.

IV. APPLICATIONS

In this section, a few representative applications are presented: blinking LEDs, output compare, input capture, ADC with LCD.

The first and the simplest application that can be executed by a microcontroller is the blinking of LEDs, which means switching on or off the LEDs in different ways. This simulator allows this by using the four LEDs as it can be seen in Fig. 1. Moreover, the push button S1/P1.3 can switch between different versions of blinking applications.

The second application presents the output compare function of the Timer_A module. Mainly, this function allows generating square signals having variable duty cycle. The signal that is generated can be seen on the oscilloscope by selecting the suitable pin, P1.2 for instance, but also in the dedicated tab, called Timer0_A3 CCR1 Compare. Thus, in this case, among the signal of the pin P1.2, TA0.1, the following signals are displayed, as in Fig. 4:

- the content of the register TA0CCR0, 249

- the content of the register TA0CCR1, 100

-the content of timer/counter register TA0R. Thus it can be seen that when the TA0R intersects TA0CCR1, the output signal TA0.1 is switched low and then, when the TA0R intersects TA0CCR0, the output signal is switched high. Using the time scale of the oscilloscope, the period of the output signal is computed as (42870-42620) µs = 250 µs and the pulse width is (42720-42620) µs = 100 µs. As expected, these values are proportional to the content of the two registers.

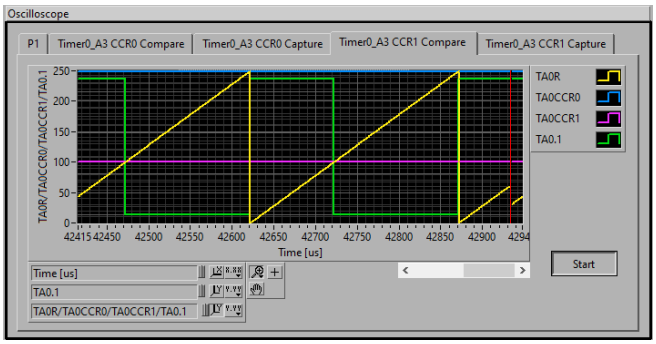

Fig. 4. Output compare function of Timer_A on the oscilloscope

The third application presents the input capture function of the Timer_A module. This function mainly saves the timer value in a register when the selected edge of the input signal occurs. Thus, having two such timer values, the corresponding time interval between the two edges can be computed. To achieve a better understanding of the capture process the oscilloscope displays among the input capture signal, TA0.1 the following, Fig.5:

- the content of the counter register TA0R

- the content of the capture register TA0CCR1

Because the capture mode on both edges is selected, two values of the TA0CCR1 are presented for each pulse, at the intersection between the content of TA0R and the two edges.

Thus, in Fig. 5 the two values that correspond to the middle pulse are about 2088 and 2187, respectively.

To implement this application the signal generator generates a signal of 2 kHz frequency and 20% duty cycle, Fig. 6. Using a suitable software application 20 values are captured and then the corresponding 19 differences are computed. The 20 values, in hex, can be seen in RAM memory, Fig.6, starting with 0x202 address: 58, BB, 24C,…, 828 (2088 decimal), 88B (2187 decimal),..,124F. Then, the differences of the 20 values can be seen, starting with address 0x240, also in hex: 63, 191, 63, 191,…,. These differences in decimal, are 99, 401, 99, 401,…, and they represent the length of the signal pulse and respectively, the pause between pulses, in microseconds. The sum of each two differences, 500 microseconds, represents the period of the signal. The ratio is 99/(99+401)=0.198. These values are as expected according to the frequency and the duty cycle of the input signal.

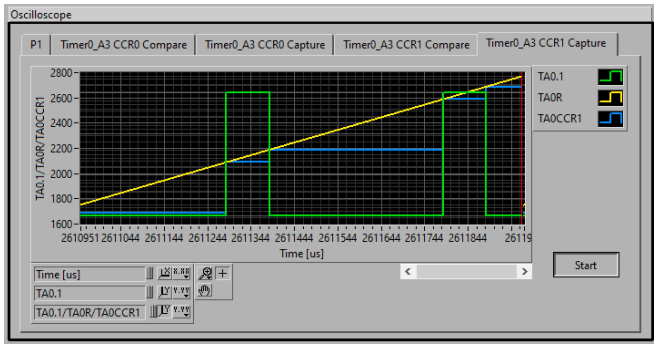

Fig. 5. Input capture function of Timer_A on the oscilloscope

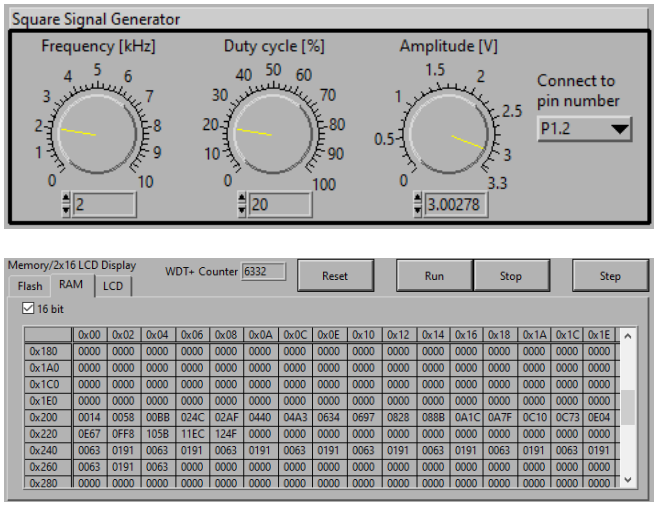

Fig. 6. Parameters of input capture signal (top) and the RAM memory after completion of the capture program (bottom)

The last application presents the use of ADC10 module of MSP430G2553 with the 2x16 LCD display. Fig. 7 presents the LCD after running of the program that measures the voltage from the potentiometer that is connected to the A4 analog input pin. By changing the potentiometer the value on the 2x16 LCD is changed too.

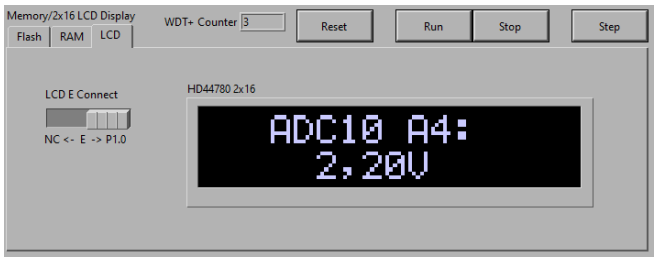

Fig. 7. The 2x16 LCD during use of ADC10 application

V. Conclusions

The paper presented a simulator for MSP430G2553 microcontroller. It can be used for free by students during this pandemic period. This can be an alternative to the demonstrative version of the Proteus environment. It contains the most important resources which are required to develop a basic application such as LEDs, push buttons, signal generator, oscilloscope and LCD display. These devices can be interconnected to different pins of the microcontroller.

As a future work we want to optimize the LabVIEW block diagram and the MathScript functions to increase the execution time of the simulator. Also the proposed simulator can be improved by including a terminal that can simulate the serial asynchronous communication using corresponding peripheral module of the MSP430G2553 microcontroller.

References

[1] C. Unsalan, and H. D. Gurhan, “Programmable microcontrollers with applications. MSP430 LaunchPad with CCS and Grace” McGrawHill Education, 2014.

[2] S. Mischie, “On teaching microcontroller course for undergraduate program of study”, 2015 International Symposium on Signals, Circuits and Systems (ISSCS, pp.1–4, Iasi, Romania, 2015.

[3] J. Kim, “An Ill-Structured PBL-Based Microprocessor Course Without Formal Laboratory” in IEEE Transactions on Education, vol. 55, pp. 145-153, February 2012.

[4] https://www.labcenter.com, PCB Designd and Simulation, accesed on June 30, 2021.

[5] K. Asparuhova, D. Shehova and S. Lyubomirov, “Using Proteus to Support Engineering Student Learning: Microcontroler-Driven Sensors Case Study”, Proc. XXVII International Scientific Conferences Electronics-ET2018, Sozopol, Bulgaria, 2018.

[6] A. Africa, D. Abaluna, A.Abello, J.Lalusin, “Design and Analysis of a Closed-Loop Temperature Enginnering Control System using MikroC and Proteus”, 2020 International Conference on Decision Aid and Application(DASA), pp. 184-189, Sakheer, Bahrain, 2020.

[7] N. Shwetha, L.Niranjan, V. Chidananadan and N. Sangeetha, “Advance System for Driving Assistance Using Arduino and Proteus Design Tool”, Proceedings of the Third International Conference on Inte;;igent Communication Technologies and Virula Mobile Networks (ICICV 2021), pp. 1214-1219, Tirunelveli, India, 2021.

[8] https://www.ti.com/product/MSP430G2553, accesed on June 30, 2021.

[9] https://www.iar.com/ew430, accesed on June 30, 2021.

[10] https://www.ti.com/tool/MSP-EXP430G2ET, accesed on June 30, 2021.

[11] https://www.ti.com/lit/ug/slau144j/slau144j.pdf?ts=1625032060761, , accesed on June 30, 2021.