The International Journal of Engineering and Science (IJES)
|| Volume || 12 || Issue || 6 || June || Pages || PP 01-08 || 2023 ||
ISSN (e): 2319-1813 ISSN (p): 20-24-1805

Using the MSP430 Microcontroller as a Signals Generator to drive a single-phase power inverter

M. Garduño Aparicio[1*], J.L. Avendaño[2]
[1]*Ph. D. Electronics, Universidad Autónoma de Querétaro, Querétaro, México*
[2]*M. C. Electronics, Universidad Autónoma de Querétaro, Querétaro, México*
**Corresponding Author*

---***ABSTRACT***---
This paper shows an MSP430G2 microcontroller application like a generator of control signals for a typical single-phase power inverter. For this case, the microcontroller is used as an alternative to generate high frequencies with low noise and short dead times, which is usually a problem with other microcontrollers, besides if the power inverter is for high voltage applications. The output signals are complementary, including a dead time between the active states with a wide range of operation output frequencies 0 – 200 kHz. Constant dead times of 200 ns were obtained in all of the range of frequencies; the noise is reduced, including a stable behavior between the output signals, with the resulting high frequencies that could be possible to design high-frequency power inverters. The assembler code also explains the mathematical behavior of the cycles and time consumption to make this kind of application.

KEYWORDS;*-MSP430; single-phase inverter; MSP assembler; MSP430G2.*

Date of Submission: 01-06-2023 Date of Acceptance: 10-06-2023

I. INTRODUCTION

A microcontroller is one of the last solutions for technology implementation in the automation field for industrial and domestic products; in the previous decades, it has been used in many different applications. Automation aims to be one of their most important objectives. The computational needs were the motivation for developing these computing machines and small devices in which day-per-day innovation is essential. Also, speed, better communications, ease to use, realistic environment, fast implementation, and smaller size with practical computation are some of the issues which could further increase expectations from the computing machines now called microcontrollers, launchpads, development kits, prototyping platforms, shields, embedded systems, etc. **[1-3]**.

An embedded system is a combination of computer hardware and software that does a specific job; these embedded systems are used in consumer electronics, the food processing industry, power sources, chemical plants, cement plants, biomedical equipment, telecommunication and security, and research applications **[4]**.

The difference between an interpreter and a compiler lies in generating machine code. The compiler reads the entire program, translates it into object code, and then executes it by the processor. On the other hand, the interpreter takes one statement of a high-level language program as input, translates it into object code, and executes the instructions. One of the significant drawbacks of high-level languages is that the machine code compiled from high-level language programscan run slower thanthe machine codes of the assembly language program. For this reason, most real-time application programs are written in assembly language **[4, 5]**.

The MSP430 microcontrollers from Texas Instruments are 16-bit mixed-signal processors designed for ultra-low power, direct memory access, autonomous peripherals for different consumption configurations, and many software tools to program their flash memory **[6]**.

A. Power Inverter Circuit

An inverter is a circuit that converts direct voltage (DC) to alternating current (AC). The voltage changes of polarity, frequency, and magnitude have specific values that could be selected from the kind of inverter circuit or operation parameters designed or specified from the final application of their energy obtained **[7]**. This polarity change is made by synchronized switching of power semiconductors BJTs, MOSFETs, or IGBTs. They are driven with gating signals; in the case of MOSFETs and IGBTs,these signals should be ~15V of amplitude and provided with currents with more than 200 mA to get effective and stable switching of power semiconductors **[2, 8, 9]**.

Fig. 1 shows a single-phase inverter in a half-bridge topology. It works with two IGBTs, which will be switched according to two signals generated by a microcontroller. For this case, it is an MSP430G2353 that is studied and analyzed. The produced signals need to be strengthened using driver circuits to get a signal with enough current to switch the power semiconductors selected from the final application of the inverter circuit **[10, 11]**.

The circuit shown in Fig 1 uses one direct current source with a value of E Volts, and the energy is obtained from the power load. During a time interval, Q1 is switched (turned on), so the voltage from the CD source goes directly to the power load. Meanwhile, the next time interval, Q1 is turned off, and Q2 is switched to on state. At this time, the load voltage is 0 Volts. If the load were inductive, the induced current would return to the ground connection. After this time interval, Q2 is going to switch off, and Q1 turned on, forming a sequence that makes an alternating current in the power load, including the square waveform in this output; also, the waveform depends on the designed times of gating signals, and these signals are generated from a microcontroller.

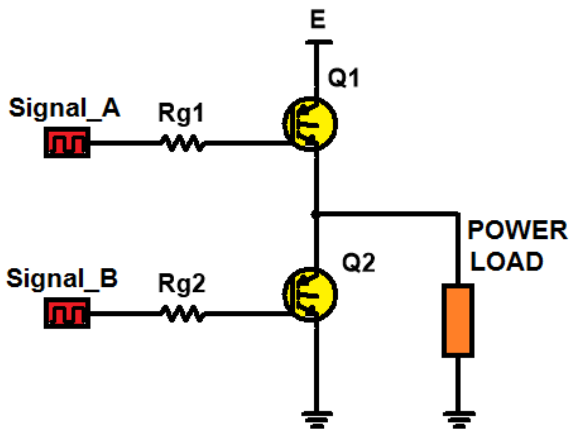

Fig. 1. Single-Phase Half-Bridge Inverter.

There are many different topologies in inverter circuits; there are three-phase or single-phase inverters, their connections with half-bridge or full-bridge, and their use depends on the user application and their specific characteristics **[2, 12, 13]**.

For this case, in Fig. 1, the output voltage in the power load has only one polarity, which could be considered just the pulsed direct current; instead of that, this power signal could be applied to a brushless motor to modify the speed and torque in robotics applications for example. So before to made power circuits, one crucial step is how to generate the gating signals for controlling the period of the power signal.

Another situation to consider was when Q1 and Q2 turned on simultaneously, a short circuit is produced. That is forbidden in power electronics (undesirable condition) even though the time of this occurrence is short; that could happen due to delay time or time response of power semiconductors even if the signals A and B were fully inverted between them. For this reason,creating a dead time (DT) between the changes in high levels of the gating signals is highly suggested. That is, a time-lapse in whichsignals A and B have a 0 Voltsto prevent one IGBT from continuing in on state when the other IGBT switches to on simultaneously, preventing the short circuit, which causes problems.

B. Gating signals of a Single-Phase Inverter Circuit

Considering the characteristics, inverter behavior, and a dead time for gating signals, signals A and B could be like the waveforms shown in Fig. 2. The signals are fully inverted, including a time interval when both signals have 0 Volts.

Fig. 2 shows the times t1, t2, t3, and t4,which are the active time when a power semiconductor is switched to the on state, in t1 the Q1 is turned on; after this is possible to notice that the deadtime appears before the time t2. That could be enough to wait for Q2 switches to off state; this dead time (DT) depends on the power semiconductors used, and it is constant in all of the cycles during the operation of the inverter circuit; typical values are 250 ns for almost high-frequency power semiconductors.

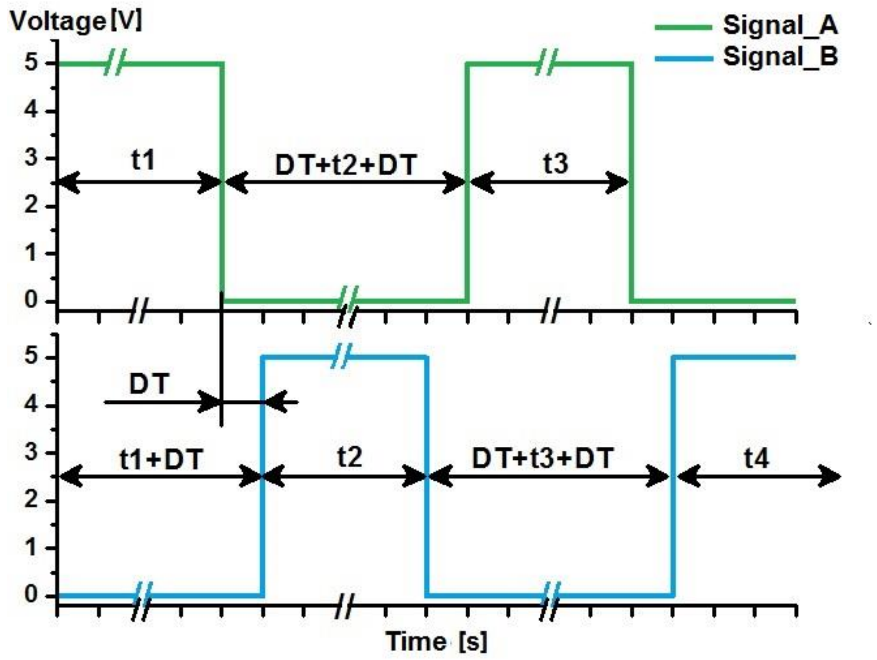

Fig. 2. Gating signals for a single-phase half-bridge inverter.

Because the dead time is small about the active times, the inverter circuit's operation frequency depends mainly on the times when Q1 and Q2 are in on state (t1 and t2 for this case). Considering the dead times, it is possible to write equation (1) for the operation frequency of these inverter circuits.

$$f_{op} = \frac{1}{2\mathrm{DT} + \mathrm{t1} + \mathrm{t2}} \tag{1}$$

II. METHODOLOGY

A gating signals generator can be implemented with a microcontroller, like the signals shown in Fig. 2 **[11, 14]**. The operation frequency could be adjustable with a constant dead time between the active times and symmetrical alternating signals A and B. For this case, an MSP430G2353 microcontroller is used under assembler programming, which allows continuous operation when the program is running and enough machine instructions to work the application.

C.Implementation Setup

To program MSP430G2553, a LaunchPad MSP430 (development tool) is used and connected with pin 2 (P1.0) and pin 3 (P1.1) from port 1 of the microcontroller GPIO (General Purpose Input/Output); this is shown in the schematic connection diagram (Fig. 3).

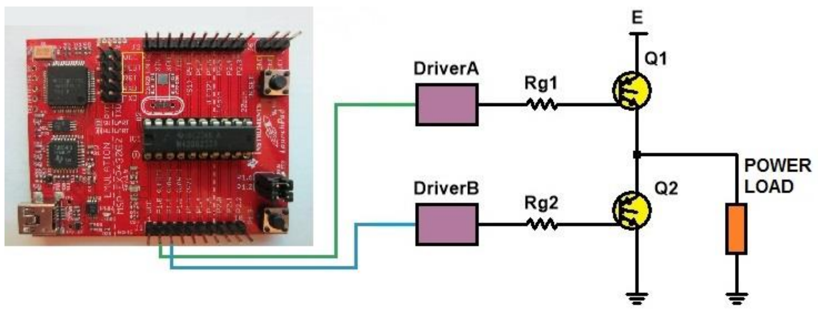

Fig. 3. Schematic diagram for implementation setup.

In Fig. 3, there are two circuit drivers. They are used to improve the current of signals generated by the microcontroller because more current is necessary to get the power semiconductors to on state; for this case, a TLP250 photocoupler is used; with this circuit, the maximum output current could be 1.5 A, more than enough to get switching a lot of commercial IGBTs.

D.Assembler program

In Table 1, it is possible to see one assembler program example used to generate both gating signals described in Fig. 2. A dead time is considered, and the active time for signals isentirely symmetrical.

In line 1 of the code example1, the “include” permits using the standard tags for specific registers of MSP430. From lines 2 to 7,the initial configuration for the assembler. Line 9 is the initialization of the stack pointer; line 10 stops the watchdog. Line 13 selects the beginning of the stack pointer for this program. Line 14 sets up pins 2 and 3 like digital outputs. The line 16 y 17 configure the main clock frequency for this case is 20 MHz; from line 19 to 26 is an infinite cycle where the first line (20) clears the digital outputs (pins 2 and 3). This creates a dead time that occurs during four clock cycles before the instruction in line 21. This instruction in line 21 sends a high state on pin 2 (P1.0); in lines 22 and 23, there are twono-operation instructions to make a symmetrical output between the high state of pin 2 and the high state of pin 3. Then, the *jmp* instruction consumes two clock cycles more than the instruction in line 21; also, in line 24, a dead time is created again before of change the high state now for pin 3 (P1.1) when the instruction in line 25 occurs. Finally, instruction 26 makes the infinite loop, in which the signals are made again, forming both gating signals that can be used to control a power inverter.

In lines 28 to 32, the program's end and the stack pointer's end are written; the reset vector for this case is not used.

Table I. Assembler Code of example1

1	.cdecls C,LIST,"msp430.h"	; Include device header file
2	.def RESET	; Export program entry-point to
3		; make it known to the linker.
4	.text	; Assemble into program memory.
5	.retain	; Override ELF conditional linking
6		; and retain the current section.
7	.retainrefs	; And retain any sections that have
8		; References to the current section.
9	RESET **mov.w** #__STACK_END,SP	; Initialize stackpointer
10	StopWDT**mov.w** #WDTPW|WDTHOLD,&WDTCTL	; Stop watchdog timer
11	.text	; program start
12	.global _main	; define an entry point
13	_main **mov.w** #0280h,SP	; initialize direction of stack pointer
14	**bis.b** #00000011b,&P1DIR	; make P1.0 and P1.1 output
15		; all others are inputs by default
16	**mov.w** #0xFF,&DCOCTL	; maximum DCO value
17	**mov.w** #0x8F,&BCSCTL1	; maximum main clock frequency
18	**nop**	
19	Mainloop	
20	**bic.b** #00000011b,&P1OUT	; clear P1.0 , P1.1 ; 4 cycles
21	**bis.b** #00000001b,&P1OUT	; set P1.0 ; 5 cycles
22	**nop**	; 1 cycle
23	**nop**	; 1 cycle
24	**bic.b** #00000011b,&P1OUT	; clear P1.0 , P1.1 ; 4 cycles
25	**bis.b** #00000010b,&P1OUT	; set P1.1 ; 5 cycles
26	**jmp**Mainloop	; infinite loop ;2 cycles
27		
28	**.global** __STACK_END	; end of stack pointer
29	**.sect** .stack	
30	**.sect** ".reset"	; MSP430 RESET Vector
31	**.short** RESET	; .short _main
32	.end	; end of the program

E. Assembler program times

The time used by the processor after the time when the two digital outputs (pin 2 and 3) are cleared in the instruction bic.b in line 20, there arefour main clock cycles (200 ns) due to that four cycles are the time used for this instruction and the main clock frequency is 20 MHz, pretty similar to bis.b who uses five main clock cycles (line 21), the no operation instruction (*nop*) uses only one cycle, so the time used for the processor when the pin 2 is in high level (line 21-23) is the same when the pin 3 is in high level (line 25 and 26) because the instruction *jmp* uses two cycles. With this theoretical measurement, there are seven cycles in on state for the digital outputs and four cycles for dead time.

The example1 could get the maximum frequency for the generation of 2 gating signals for a single-phase half-bridge inverter when the time in on state is the same for both signals; for this case, this time is 350 ns, but it is possible to write or modify this time adding more *nop* cycles, just after the line 21 and 25 with the same number of *nop* (no operation instructions), with all of this information is possible to write a numerical relation for the assembler code of example1.

$$f_{op} = \frac{20\times10^{6}}{8+7+7} \tag{2}$$

In equation (2), the value of 20×10^{6} is for the main clock frequency, in this case 20 MHz, the value 8 is for 2 dead times with 4 clock cycles each one, the first 7 is for the 5 cycles of instruction in line 21 including the 2 cycles of *nop* in line 22 and 23, the second 7 is for the 5 cycles of instruction in line 25 and 2 cycles added for the *jmp* instruction in line 26.

Therefore is possible to write equation (3) where f_{ck} is the main clock frequency, and the n value is the number of *nop* cycles added for the *on state* in both signals generated (signalsA and B), depending on the application of the single-phase inverter.

$$f_{op} = \frac{f_{ck}}{22+2n}; \quad n = 0, 1, 2, 3, \ldots \infty \tag{3}$$

III. RESULTS

In Fig. 4, it is possible to observe the generated signals with the assembler code example. The values of equation (2) could be corroborated with the real signals obtained when an MSP430G2353 microcontroller is used in MSP430 LaunchPad; herein, pins 2 and 3 are the digitaloutputs, and the $f_{op} = 909\text{ kHz}$. As a result, it is an excellent high frequency for a resonant single-phase inverter with research applications.

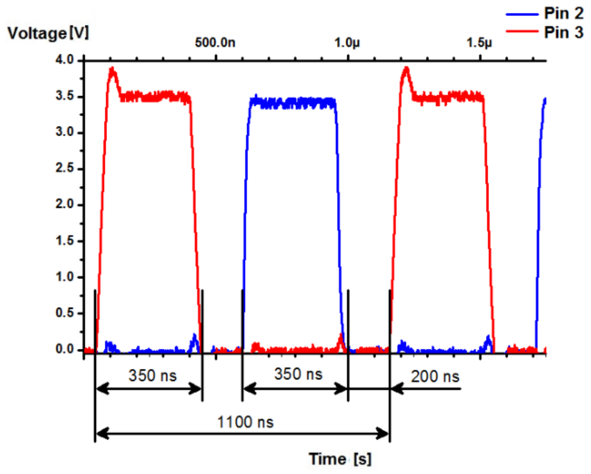

Fig. 4. Obtained signals from implementing assembler code example1 in MSP430G2353 microcontroller.

Using the assembler code example1 and adding cycles *nop* after lines 21 and 25 is possible to generate the operation frequency for inverters from 0 - 909 kHz, just like the equation (3), using delay routines, or they could be controlled by potentiometer leads to analog input. So, the number of *nop* cycles will depend on analogic value outside of the microcontroller; for this case, little changes in assembler code could be made, like the flowchart in Fig. 5.

The flowchart of Fig. 5 is similar to the assembler code of example1. A delay routine after a change to on state in digital outputs is controlled by an amount of no-operation cycles about the analog input received, depending on final operation frequencies. If the work to read the analog input is made for each loop or this work could be controlled for a selected time to read a new analog value, the only thing to consider about this situation is to add the same times of no operation cycles for the other complementary output signal. That is shown in Fig. 5, like *nop compensation* to get equilibrated gating signals.

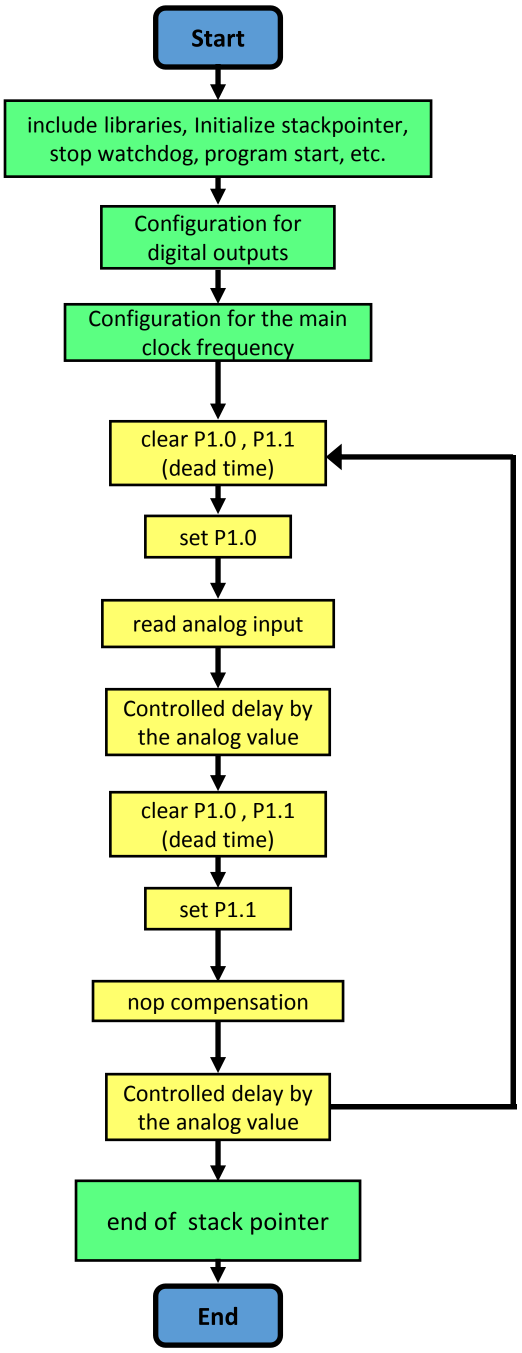
Start
include libraries, Initialize stackpointer, stop watchdog, program start, etc.
Configuration for digital outputs
Configuration for the main clock frequency
clear P1.0 , P1.1 (dead time)
set P1.0
read analog input
Controlled delay by the analog value
clear P1.0 , P1.1 (dead time)
set P1.1
nop compensation
Controlled delay by the analog value
end of stack pointer
End

Fig. 5. Flowchart for gating signals controlled by analog input.

In Fig. 6 is possible to see the results from a program made considering the flowchart in Fig. 5. The time in on state for both signals is 5 ms; during this process is possible to read the analog input each period of the gating signals generated, and the operating frequency for this example is 100 Hz, which could be applied for a single-phase inverter to control a brushless universal motor or similar power load, and the possibility to change the operating frequency with this assembler program is a range of 0 – 200 kHz, The operationalrange could be selected considering the final application of the inverter circuit.

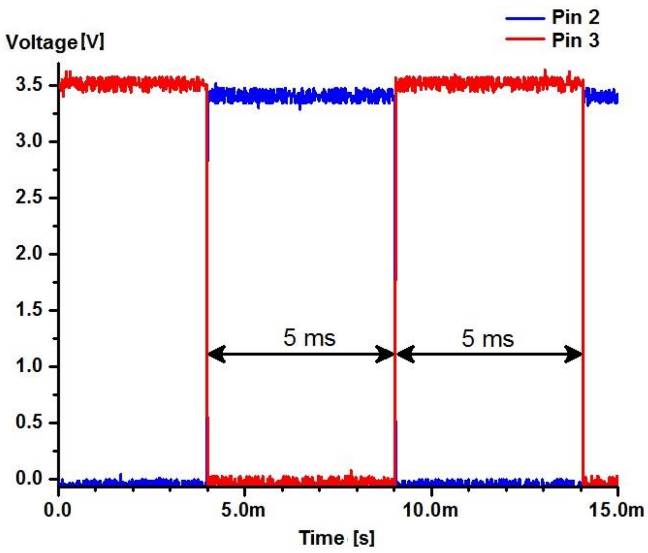

Fig. 6. Generated signals from the implementation of the flowchart of Fig. 5 using the MSP430G2353 controlled by a simple analog input.

According to the time responses from the power semiconductors selected is possible to use the same dead times for all of the generated cycles; this value is a constant during the microcontroller's operation time. For this case, this time has four cycles of the main clock frequency (microcontroller clock), but it is possible to increase with the addition of no operation cycles in the assembler code. After lines 20 and 24 of code example1, this dead time is an easy way to prevent a short circuit in the power semiconductors.

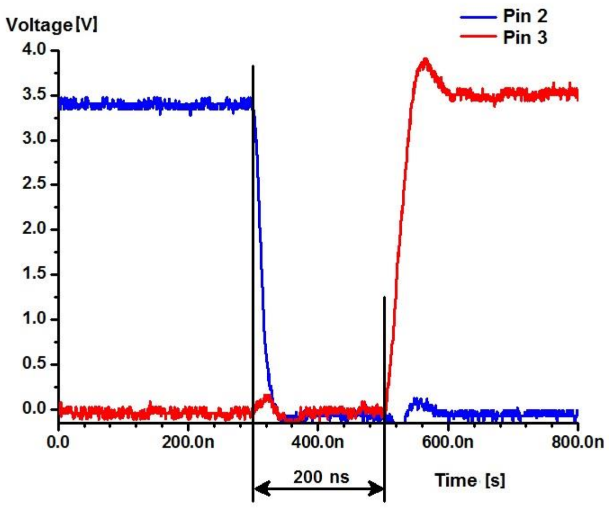

Fig. 7. Dead time obtained from the MSP430G2353 used for gating signals generator for a single-phase inverter controlled by analog input.

In Fig. 7, the practical dead time produced during all of the operation time when the microcontroller MSP430G2353 is used to generate gating signals when the frequency control is by analog input (outside of the microcontroller), this time, is a constant of 200 ns, instead of the output signals frequency has 100 Hz.

There are many different applications of this kind of gating signals generator; when it is used to drive a power inverter with the frequency of 100 Hz is possible to drive a brushless motor for a robot, just an example, because the operation frequency of the motor could be from 0 Hz to 300 Hz, an inductive oven works with the range 1 - 20 kHz, so with the MSP430G2 is possible to manipulate manufacturing procedures, using digital control also. For research applications, many of the circuits of high voltage generators are based on power inverters, and the frequency is a fundamental fact, so to work with frequencies >50 kHz with low noise, stable signals, and the digital control system is an important opportunity to develop a different kind of experiments or solutions in plasma physics.

IV. CONCLUSIONS

Using the MSP430G2 microcontrollers and their LaunchPad development tools is possible to make applications for power electronics, for this case applying the microcontroller to generate control signals that could drive single-phase inverters with a wide range of operating frequencies. 20 MHz of the main clock frequency in a microcontroller could get a 909 kHz in maximum output frequency, and use another kind of control; for example, with the use of one analog input was possible to get a range between 0 – 200 kHz for the operating frequency that can be used for a single-phase inverter circuit. The programming of this task was made in assembler code.

The output signals from the microcontroller were stables with less noise and easy to handle for power applications; the assembler code is flexible for normal purposes of single-phase inverters and research applications where the frequency is an important fact, science topics like surface treatment or high voltage technology.

The mathematical equations (2, 3) are in accordance with the time consumptions from the assembler code, the cycles and times were corroborated with the practical results, and they could be useful for power inverters designers. With this kind of gating signals, in the generator designed in this work using an MSP microcontroller was possible to add a dead time between the on states for these output signals to improve better switching in power semiconductors and prevent short circuits in these power circuits, including dead time as a constant with programmable period by the assembler code.

REFERENCES

[1] A. V. Deshmukh, Microcontrollers: Theory and Applications: McGraw-Hill, 2005.

[2] R. H. Chu, D. D. C. Lu, and S. Sathiakumar, “Project-Based Lab Teaching for Power Electronics and Drives,” IEEE Transactions on Education, vol. 51, no. 1, pp. 108-113, 2008.

[3] C. S. Lee, J. H. Su, K. E. Lin, J. H. Chang, and G. H. Lin, “A Project-Based Laboratory for Learning Embedded System Design With Industry Support,” IEEE Transactions on Education, vol. 53, no. 2, pp. 173-181, 2010.

[4] V. Udayashankara, Microcontroller: McGraw-Hill Education (India) Pvt Limited, 2009.

[5] A. Maly, and V. Krischuk, "The Dynamic Description of System of Instructions of Microcontrollers." pp. 264-265.

[6] Texas_Instruments, MSP Low-Power Microcontrollers, ti.com/msp, 2015.

[7] M. H. Rashid, Electrónica de potencia: circuitos, dispositivos y aplicaciones, p.^pp. 878: Pearson Educación, 2004.

[8] N. Mohan, Electric Drives: An Integrative Approach: MNPERE, 2003.

[9] E. B. Portillo, and R. P. López, Electrónica de Potencia, p.^pp. 680 Marcombo, 2012.

[10] N. Mohan, Electric Machines and Drives: Wiley, 2012.

[11] M. Gunasekaran, and R. Potluri, “Low-Cost Undergraduate Control Systems Experiments Using Microcontroller-Based Control of a DC Motor,” IEEE Transactions on Education, vol. 55, no. 4, pp. 508-516, 2012.

[12] J. de Dios Sánchez López, Dispositivos electrónicos de potencia: Universidad Autónoma de Baja California, 2002.

[13] D. W. Hart, Electrónica de potencia, p.^pp. 451: Pearson Educación, 2001.

[14] M. Hedley, and S. Barrie, “An undergraduate microcontroller systems laboratory,” IEEE Transactions on Education, vol. 41, no. 4, pp. 345-345, 1998.

M. Garduño Aparicio, et. al. "Using the MSP430 Microcontroller as a Signals Generator to drive a single-phase power inverter." *The International Journal of Engineering and Science (IJES)*, 12(6), (2023): pp. 01-08.