James L. Kretzschmar, AE7AX

7420 Huntz Drive, Cheyenne, WY 82009, USA: **jlkwy@bresnan.net**

Programming the DDS AD9850 Signal Generator Module with the Texas Instruments MSP430 LaunchPad

Precision frequency control with assembly code.

The AD9850 Signal Generator module has been available for several years and there are many online sites that provide program code. However, using these pre-packaged programs does not give you the flexibility and control that can be realized by developing your own code. This article describes how to achieve this control by using the Texas Instruments MSP430 LaunchPad to drive the AD9850 Signal Generator module. To accomplish this task, we will be using the Texas Instruments free online IDE (Integrated Development Environment) called Code Composer.[1] To accelerate the process of learning a new software application, a step-by-step guide can be found in *QEXfiles* web page.[2] The inspiration for this project was the need for a stable and precise way to generate a frequency for experimental projects in both the VLF regions of the radio spectrum and the 40 meter band.

First, the features of the MSP430 LaunchPad and AD9850 Signal Generator module are described followed by a discussion on the documentation. Next is a detailed discussion on configuring the AD9850 module and how to calculate the Frequency Number that will be entered into the module. The next section provides a detailed discussion on the mechanics of the Assembly code program. Bringing it all together are the details for building a fixed frequency signal generator for 7.045 MHz. The frequency can easily be modified to fit your needs.

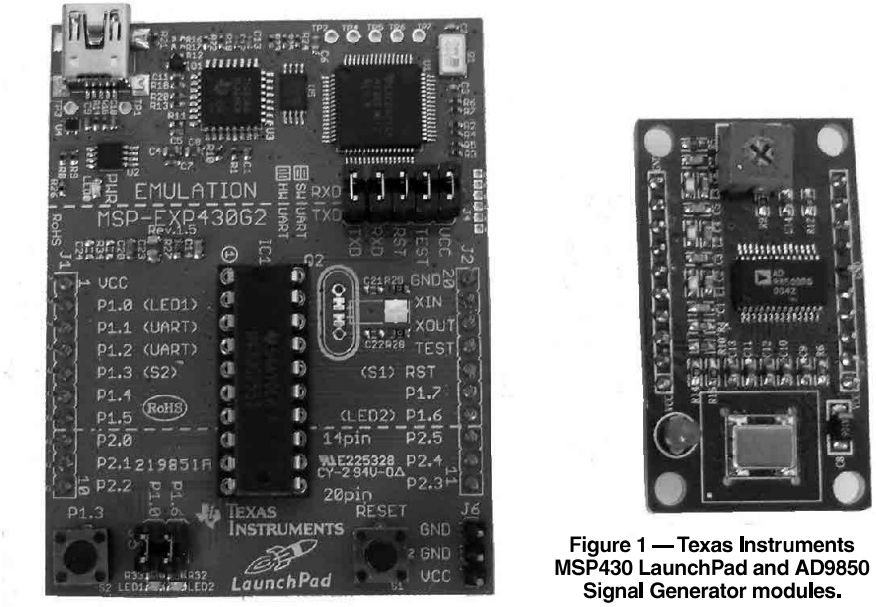

Figure 1 — Texas Instruments MSP430 LaunchPad and AD9850 Signal Generator modules.

Texas Instruments MSP430 LaunchPad

The MSP430 LaunchPad is an easy-to-use development board for the low-power and low-cost MSP430G2x family of microcontrollers and is available from Texas Instruments and other distributors for $10 (see Figure 1). It has on-board emulation for programming and debugging, and includes a 14/20-pin DIP socket, two on-board tactile switches and two LEDs for easy initial

experimentation. The MSP430 LaunchPad comes with two MSP430 microcontrollers, the MSP430G2553 and MSP430G2452. We will use the MSP430G2553 which has 16 kB Flash memory, 512 B RAM, and a CPU speed of 16 MHz. In addition, peripherals incorporated into the microcontroller include an 8 channel 10-bit ADC (Analog-to-Digital Converter), two timers, and serial communication capabilities (UART, I2C and SPI). Useful features of the LaunchPad are that it can program a variety of the MSP430 family of microcontrollers and the chip can be removed from the LaunchPad board and incorporated into a project. Detailed documentation for the microcontrollers included with the MSP430 LaunchPad is available from the Texas Instruments web page.[3]

DDS AD9850 Signal Generator Module

The AD9850 Signal Generator module provides a single frequency output in the range of 0-40 MHz and is available from several online sources for about $16. (see Figure 1). The frequency is defined by a 40-bit Tuning Word that can be sent to the module in either serial or parallel format. The signal phase is controlled by bits 34-39 of the Tuning Word. The module has an onboard 125 MHz reference clock and can be powered by a 5.0 or 3.3 V power supply. For this project, the 3.3 V supply available from the MSP430 LaunchPad will be used, the 40-bit Tuning Word will be entered in serial format, and bits 34-39 will be set to "0". Setting these five bits to "0" produces no change in the phase of the signal. Please note, if two of these modules were to be used for mixing same frequency signal sources that are out of phase, they must both operate from the same reference clock.

Documentation

Documentation for the AD9850 Signal Generator module is available online and does provide a schematic diagram which is useful when configuring the AD9850 module for entry of data in serial mode.[4] The Analog Devices AD9850 datasheet is comprehensive and the figures and diagrams were very helpful in the development of the Assembly program for the MSP430G2553 microcontroller.

After studying the Analog Devices AD9850 datasheet two things became apparent. First, it is necessary to enter a 40-bit Serial-Load Enable Word to establish serial mode operation (this will be discussed in detail in the Assembly Code Program section). Second, in the tables, the terms "W0 - W39" actually refer to "bits" instead of "Words".

AD9850 Configurations

When setting up for serial mode the Analog Devices AD9850 datasheet (page 12, Figure 23) shows that pins 3 and 4 of the AD9850 chip must be tied to the supply voltage, and pin 2 of the AD9850 chip must be grounded. On the AD9850 module pins 3 and 4 are wired to the supply voltage, however, pin 2 is connected to pin D2. Thus, pin D2 of the AD9850 module needs to be grounded for serial operation.

Referring to the Analog Devices AD9850 datasheet (page 12, Figure 22 and Figure 24) there are 3 lines necessary to operate the AD9850. The logic analyzer pictures (see Figures 2, 3, and 4) show these three lines and will be helpful in understanding how data is inputted into the AD9850 module. These three lines are:

1 – DATA line – This is where the 40-bit Serial-Load Enable Word and 40-bit Tuning Word are input into the device.

2 – W_CLK line – (Word Clock line). The W_CLK line is pulsed each time one of the 40 bits of either the Serial-Load Enable Word or the Tuning Word is placed on the DATA line. This line is also pulsed once at the beginning of Serial-Load Enable Word word load sequence.

3 – FQ_UD line – This is the frequency update line. When the 40-bit Serial-Load

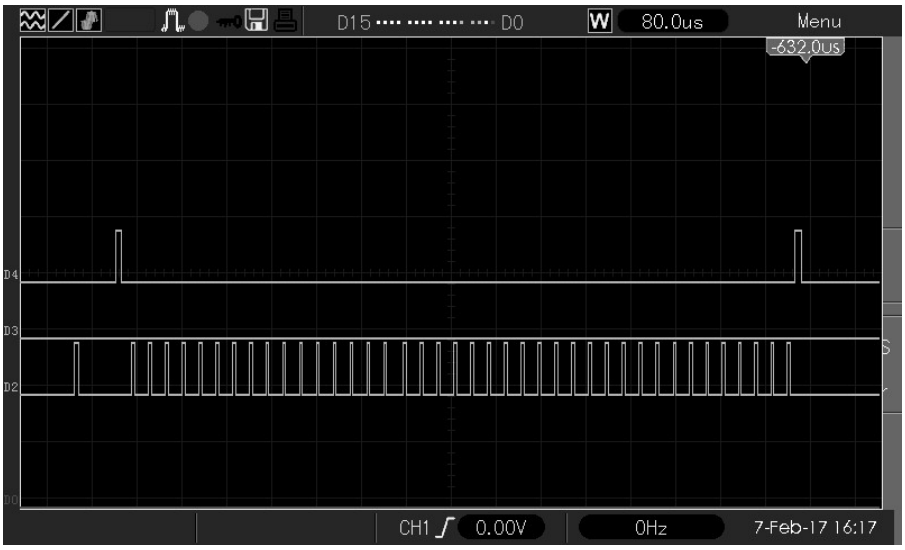

Figure 2 — Logic analyzer picture of 40-bit serial-load enable word, bit-0 on the left and bit-39 on the right. W_CLK line is on the bottom, DATA line in the middle, and FQ_UD line on the top. Note that the DATA line is all "0"s.

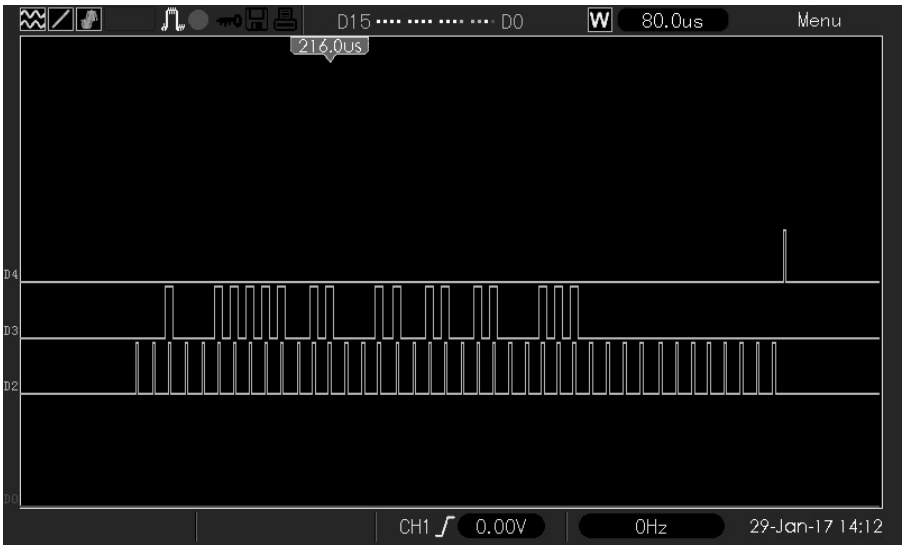

Figure 3 — Logic Analyzer picture of the 40-bit Tuning Word for 7.045 MHz. bit-0 is on the left and bit-39 is on the right. W_CLK line is on the bottom, DATA line in the middle, and FQ_UD line on the top.

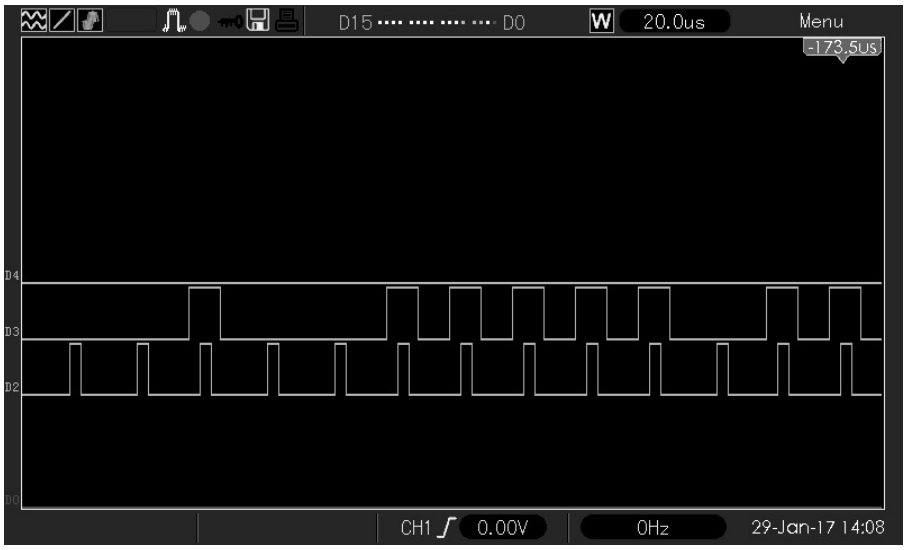

Figure 4 — Enlarged view of the logic analyzer picture of bit-0 (left) through bit-12 (right) of the 40-bit Tuning Word for 7.045 MHz. The timing relationship between the W_CLK line and the bits on the DATA line is clearly seen.

Enable Word is loaded, this line is pulsed at the beginning of the load and at the end of the load.

The Analog Devices AD9850 datasheet (page 3, Timing Characteristics) notes that when any one of these lines is brought "High" it must remain in that state for a minimum of 3.5 ns. In this project we are using the MSP430G2553 microcontroller default CPU clock speed of 1 MHz, which provide pulses well above the minimum requirement.

Table 1
The 40-bit Tuning Word

PHASE + CONTROL	*FREQUENCY NUMBER*
bits (39-35)+(34-32)	bits (31-0)

Calculating the 40-bit Tuning Word

The 40-bit Tuning Word is composed of two parts. The first 32 bits (0-31) are the Frequency Number, and the last 8 bits are the three control bits (32-34) and the five phase selection bits (35-39). The three control bits will be discussed in the next section and all will be "0". The five phase selection bits, as explained earlier, are all set to "0". Table 1 shows a generic pictorial of the 40-bit Tuning Word. The numbers in parenthesis are the bits with the most significant bit (MSB) on the left and the least significant bit (LSB) on the right.

The calculation for the Frequency Number is accomplished outside of the Assembly program and is first calculated in decimal (base 10), then converted into binary (base 2). Calculate the 32-bit Frequency Number according to the AD9850 data sheet.

$$\text{Frequency Number} = \frac{\text{Frequency to be programmed}}{\text{Reference frequency}} \times 2^{32}$$

$$\text{Frequency Number} = \frac{7.045\ \text{MHz}}{125.000\ \text{MHz}} \times 4,294,967,296$$

$$= 242,064,356$$

Convert the 32-bit Frequency Number into a binary number with the help of an online decimal/binary/hexadecimal calculator.[5] The Frequency Number of 242,064,356 is represented by the 28-bit binary number binary sequence,

11100110110110011011111100100

or when parsed into 4-bit segments,

1110 0110 1101 1001 1011 1110 0100.

Since we need a 32 bit Frequency Number and this converted number for 7.045 MHz has only 28 positions filled (bits 0 to bit 27, right to left), we must add four 0 bits in positions 28 to 31 on the left,

0000 1110 0110 1101 1001

1011 1110 0100, is now a 32 bit sequence.

The Frequency Number is now ready for manipulation so it can be inserted into the assembly program.

Table 2
Memory storage for the 40-bit Serial-Load Enable Word

Register 8	*Register 7*	*Register 6*
0000 0000 bits (39-32) (Decimal = 0)	0000 0000 0000 0000 bits (31-16) (Decimal = 0)	0000 0000 0000 0000 bits (15-0) (Decimal = 0)

Table 3
Assembly program lines that store numbers in Registers 6 – 8

```
mov #0, R6      ; Moves "0" into Register 6
                ; bits (0-15) of the 40-bit Serial-Load Enable Word
mov #0, R7      ; Moves "0" into Register 7
                ; bits (16-31) of the 40-bit Serial-Load Enable Word
mov #0, R8      ; Moves "0" into Register 8
                ; bits (32-39) of the 40-bit Serial-Load Enable Word
```

Assembly Code Program

To understand how both the 40-bit Serial-Enable Word and the 40-bit Tuning Word are loaded into the AD9850 it is best to follow along with the assembly code program that can be found in the *QEXfiles* web page. The program is comprised of three parts.

1 – Initialization

(a) The pins of the microcontroller are defined as to which ones will be used for the DATA line, W_CLK line, and the FQ_UD line.

(b) Memory storage for the 40-bit Serial-Load Enable Word. The program code uses three of the twelve 16-bit general purpose registers of the MSP430G2553 microcontroller to store the 40-bit Serial-Load Enable Word. Bits (39-32) are stored in Register 8, bits (31-16) in Register 7, and bits (15-0) in Register 6, as shown Table 2, according to the code shown in Table 3. Following the Analog Devices AD9850 datasheet (see page 12, Figure 22), bits 32, 33, and 34 of the 40-bit Serial-Load Enable Word must all be "0". To keep the rest of the 40-bit Serial-Load Enable Word uncomplicated, the remaining 37 bits were all made "0".

(c) Memory storage for the 40-bit Tuning Word (Table 4). Another 3 registers are used to store the 40-bit Tuning Word. Bits (39-32) are stored in Register 12, bits (31-16) in Register 11, and bits (15-0) in Register 10.

The 32-bit Frequency Number is divided into two 16-bit sections and stored in Register 10 and Register 11. Register 12 is also a 16-bit register, however, the Assembly code (Table 5) is written to use only the first 8 bits that will hold the Phase and Control information.

2 – Development of the "40-bit Serial-Load Enable Word"

See Figure 2.

(a) W_CLK line is pulsed once

(b) FQ_UD line is pulsed once

(c) The process for shifting data out of Registers 6, 7, and 8 onto the AD9850 DATA line for the Serial-Load Enable Word uses the Assembly instruction RRC (Roll Right through the Carry bit). Starting with Register 6, as each bit is shifted out of the register to the right, it is placed in the "Carry bit" and tested as to whether it is a "1" or a "0". The program then branches accordingly to make a "1" (DATA line brought "High"), or a "0" (DATA line brought "Low").

(d) While the "1" or "0" is on the DATA line, the W_CLK line is pulsed

(e) The "1" or "0" is then removed from the DATA line, and the next bit is shifted to the right with the RRC instruction and tested.

(f) After all 16 bits have been shifted out of Register 6, then Registers 7 and 8 are handled in the same way. A counter keeps track of when 40 bits have been placed on the DATA line.

(g) FQ_UD line is pulsed once.

3 – Development of the "40-bit Tuning Word"

See Figure 3 and Figure 4.

(a) The process for shifting data out of Registers 10, 11, and 12 into the AD9850. Starting with Register 10, as each bit is shifted out of the register to the right, it is placed in the "Carry bit" and tested as to whether it is a "1" or "0". The program then branches accordingly to make a "1" (DATA line brought "High"), or a "0" (DATA line brought "Low").

(b) While the "1" or "0" is on the DATA line the W_CLK line is pulsed.

(c) The "1" or "0" is then removed from the DATA line, and the next bit is shifted to the right with the RRC instruction and tested.

(d) After all 16 bits have been shifted out of Register 10, then Registers 11 and 12 are handled in the same way. A counter keeps track of when 40 bits have been placed on the DATA line.

(e) FQ_UD line is pulsed once.

Table 4
Memory storage for the 40-bit Tuning Word

Register 12	*Register 11*	*Register 10*
0000 0000 bits (39-32) (Decimal = 0)	0000 1110 0110 1101 bits (31-16) (Decimal = 3693)	1101 1011 1110 0100 bits (15-0) (Decimal = 39908)

Table 5
Assembly program lines that store numbers in Registers 10 – 12

```
mov #39908, R10     ; Moves "39908" into Register 10
                    ; bits (0-15) of 40-bit Tuning Word
mov #3693, R11      ; Moves "3693" into Register 11
                    ; bits (16-31) of 40-bit Tuning Word
mov #0, R12         ; Moves "0" into Register 12
                    ; bits (32-39) of 40-bit Tuning Word
```

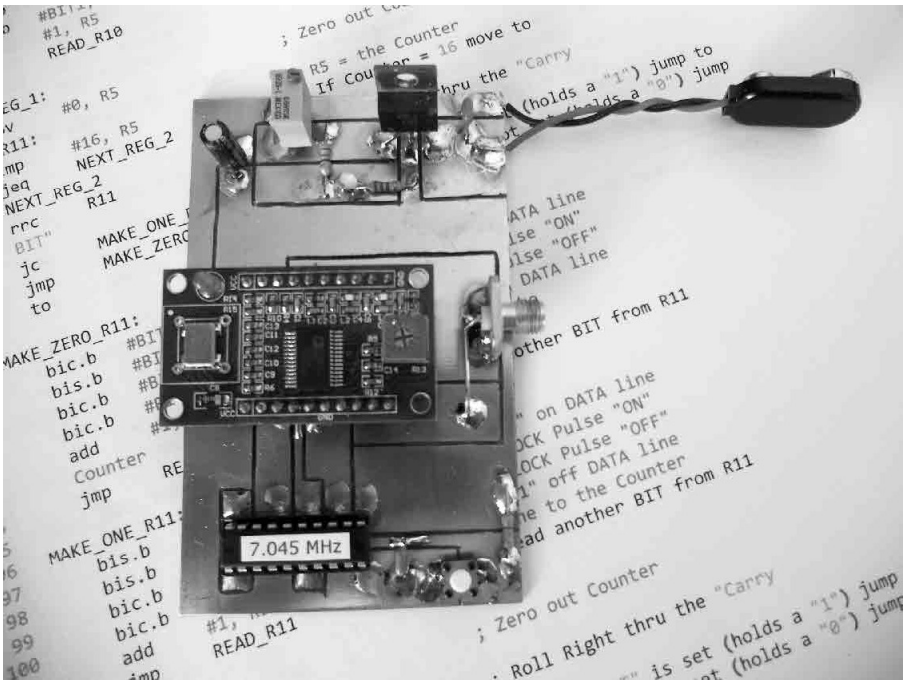

Figure 5 — Fixed frequency signal generator showing the MSP430G2553 microcontroller, AD9850 module,

Fixed Frequency Signal Generator

The Fixed Frequency Signal Generator is comprised of three sections, (1) the microcontroller, (2) the AD9850 module, and (3) a voltage regulator (see Figure 5 and Figure 6). The board template and parts placement diagram can be found in *QEXfiles* web page. Following the schematic diagram of the MSP430 LaunchPad, one tactile switch was connected from pin 16 of the microcontroller to ground for a reset of the MSP430G2553 if needed. To mount the AD9850 module, two 10-pin female header receptacles were used. On both the 20-pin microcontroller socket and the header receptacles, the unused pins were pulled out to make the board design and soldering easier. Pins 10 and 11 of the microcontroller socket have no electrical connections and are used solely to anchor the IC socket. The LM317T variable voltage regulator was chosen so that it could be set to 3.3 V, the same voltage available on the MSP430 LaunchPad.

Once the MSP430G2553 microcontroller is programmed it is removed from the LaunchPad and placed in the IC socket on the board. Applying power initiates the code routine in the microcontroller, and the data is delivered to the AD9850 module, which outputs a very stable and precise sine wave (see Figure 7). In the development of this project two observations were made. First, the frequency observed on the measurement equipment is dependent upon the accuracy of the reference oscillators on the AD9850 module and in the test equipment. This

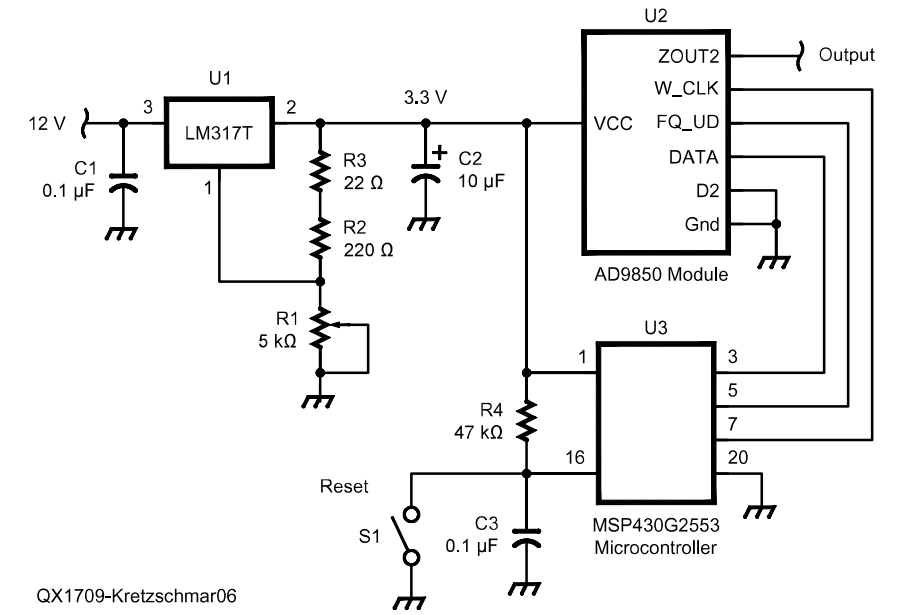

Figure 6 — Schematic diagram of the fixed frequency signal generator.

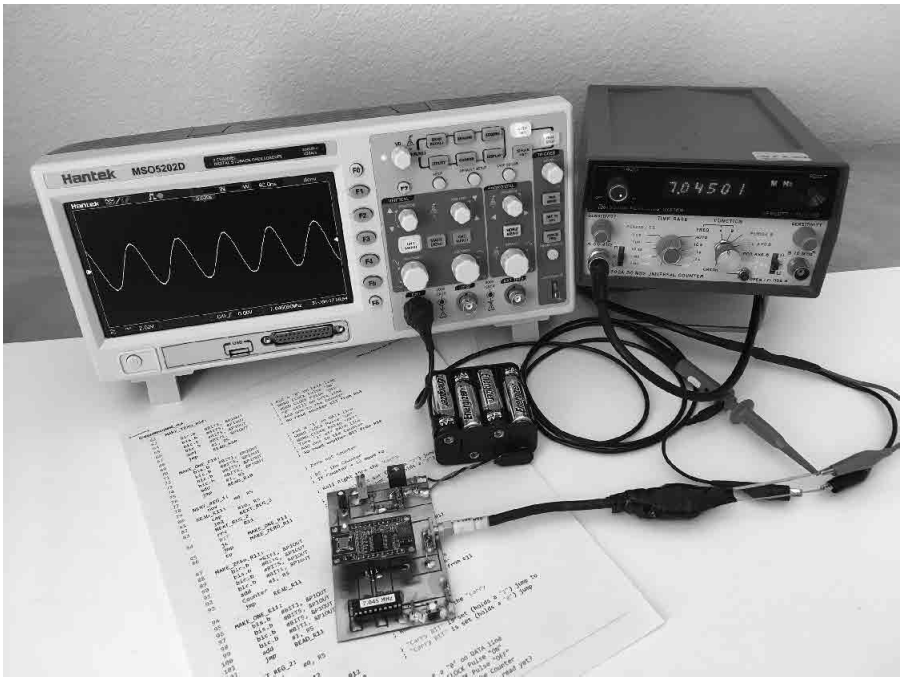

Figure 7 — Output of fixed frequency signal generator at 7.045 MHz.

will most likely result in the observed frequency to be a few hertz different from the calculated frequency. Second, during initial testing, the jumper wires between the MSP430 LaunchPad and the AD9850 module occasionally picked up stray signals (i.e., static) causing the code to not load properly. On the final board stray signal pick up was not a problem.

Summary

Writing your own program in assembly code for the Texas Instruments MSP430G2553 microcontroller allows you to precisely control the frequency output of the AD9850 DDS Signal Generator module. Possible applications include a frequency standard, the first stage of a beacon transmitter, or a way to generate a precise frequency in one of the proposed VLF Amateur Radio bands. Whatever your frequency generation application may require, this project will provide a stable and precise signal. Enjoy experimenting and have fun.

James Kretzschmar, AE7AX, was first licensed in 1972 as a Novice and now holds an Amateur Extra class license. He retired from the US Air Force in 2004 after a 25 year career as a general dentist. He currently works part time in the Oral Surgery department at the Alaska Native Medical Center in Anchorage, Alaska. James received a BA degree in chemistry from Texas Christian University, masters degree in Basic Science from the University of Colorado, and Doctor of Dental Surgery degree from Baylor College of Dentistry. In the past two years he has taken courses in the Electrical and Computer Engineering department at the University of Wyoming on Microcontrollers and Biomedical Instrumentation. He was awarded the "Best Humanitarian Impact" category prize in the 2016 Texas Instruments Innovation Challenge competition for his entry, "Futuristic Energy Saving Lighting System, Color Influenced Temperature Perception". His project involved temperature input to a microcontroller for regulation of RGB LED room lighting. James is an ARRL member and enjoys all aspects of electronics, homebrew projects, 40 meter CW operating, and bicycling.

Notes

[1]Download of Code Composer from Texas Instruments, **www.ti.com/lsds/ti/tools-software/ccs.page.**
[2]**arrl.org/qexfiles.**
[3]**processors.wiki.ti.com/index.php/Getting_Started_with_the_MSP430_LaunchPad_Workshop.**
[4]**www.analog.com/media/en/technical-documentation/data-sheets/AD9850.pdf.**
[5]Online calculator, **https://www.mathisfun/binary-decimal-hexadecimal-converter.**