AD9833 DDS レジスタを PIC で書くメモ

2017/6/01 JA5FP

AD9833 DDS は、0〜12.5MHz の出力レンジ・28 ビット分解能・サイン波/三角波/方形波の出力・周波数/位相変調が可能な信号源です。制御は 3 線式 SPI ですので、その機能を有する PIC 12F1822 マイクロプロセッサ等で簡単に行うことができます。それ等のデバイスの詳細は、次の URL からダウンロードできます。

www.analog.com/media/jp/technical-documentation/data-sheets/AD9833_JP.pdf

http://ww1.microchip.com/downloads/jp/DeviceDoc/41413C_JP.pdf

ただし、AD9833 のレジスタは”16 ビット・ワード” として説明されていますが、PIC は”8 ビット・バイト” を取り扱いますので、インターフェースの設定に悩まされます。そこで、その疑問解明を目的にして実回路で確認しましたので、本稿ではその結果をメモとして残します。

図 1 に、インターフェースと書き込みタイミングを示します。

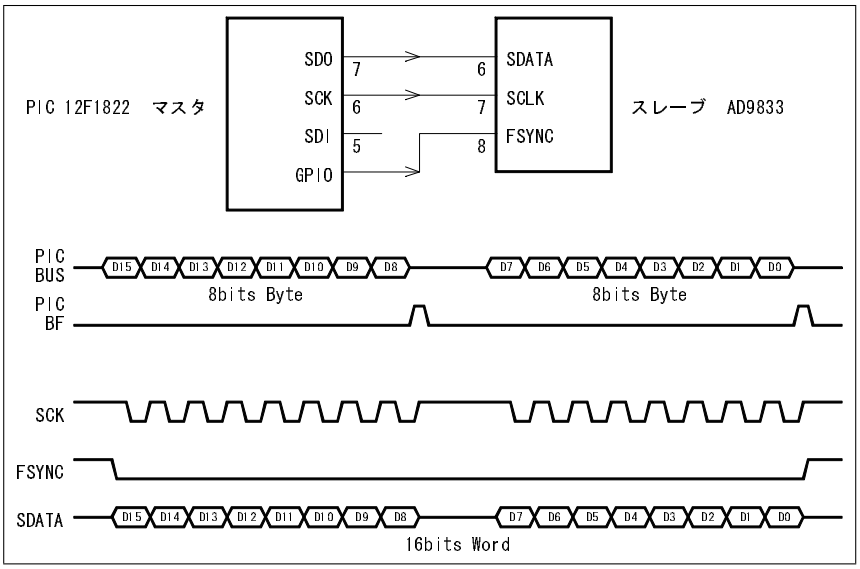

図 1: PIC SPI による AD9833 のレジスタ書き込み

もちろん、PIC 側は SPI マスタとして動作します。データが SSP1BUF にロードされると、SCK からクロックと SDO からデータの送出が行われ、8 ビット送り終えると BF が立ち上がります。これで PIC 内部では 1 バイトの送信が完了しますので、SCK をアイドリング状態に保ちます。ソフトウェアが次のデータである 8 ビット・バイトを SSP1BUF にロードして、SCK と SDO が出力されることになります。

いっぽう、AD9833 側では SCLK に同期してデータ D_{15} から D_8 までをレジスタに収納します。ここで、次のデータが届くまでは SCK がアイドル状態なのでシフトレジスタは停止しており、SCLK の再開とともに D_7 以下のデータを収納します。FSYNC がローレベルの間をデータ長として、16 ビット・ワードを認識します。

次のプログラム例は、単純に 25MHz 基準周波数から 544kHz の方形波クロックを出力するサンプルです。可変周波数・周波数変調/位相変調・出力波形等の応用は各パラメータを変更すれば可能ですが、この段階では省略します。

```
//"ad9833-evl.c" ver.0r00  (c) 2017.05.29 JA5FP
//This program may be compiled with MPLAB XC8 C Compiler on MPLAB X IDE v2.20.
//Target hardware is PIC12F1822 and DDS unit CJMCU-9833(AD9833 with 25MHz TCXO).
//The functions are:
//   (1) to learn how to configure 12F1822‘s SPI
//   (2) to set simplly specific frequency to AD9833 DDS
#include <xc.h>
#include <pic12f1822.h>
#pragma config FCMEN=OFF, IESO=OFF, CLKOUTEN=OFF, BOREN=ON, CPD=ON, CP=ON
#pragma config MCLRE=OFF, PWRTE=ON, WDTE=OFF, FOSC=INTOSC
#pragma config LVP=OFF, BORV=LO, STVREN=OFF, PLLEN=OFF, WRT=OFF
void device1822(void){                  // hardware setting
    APFCON=0b00000000;                  // SDO=RA0,
    ANSELA=0x00;                        // digital portA
    TRISA=0b00001100;                   // RA4=FNC,RA2=SDI,RA1=SCK,RA0=SDO
    OPTION_REG=0b00000000;              // WPU enable
    WPUA=0b00001100;                    // WPU portA
    SSP1CON1=0b00110000;                // SPI master mode,clock=Fosc/4=125kHz
    }// device1822()
void spi(unsigned int variable){
    SSP1BUF=variable; while(!BF);       // transmit 8bits Byte via SPI
    }// spi()
void main(void){
    device1822();                       // interface
    RA4=1;RA4=0;                        // resume 16bits Word transfer
    spi(0b00100000);                    // control register 0x2068
    spi(0b01101000);                    // remained 8bits
    RA4=1;RA4=0;                        // next 16bits Word transfer
    spi(0x61);                          // frequency register with 14LSB
    spi(0x04);                          // remained 8bits
    RA4=1;RA4=0;                        // followed 16bits Word transfer
    spi(0x41);                          // frequency register with 14MSB
    spi(0x64);                          // remained 8bits
// add more Word if needed
    RA4=1;
    while(1){;                          // continue wave
        }// while(1)
    }// main()
```