# D/A Converter Control
for
Spartan-3E Starter Kit

Ken Chapman
Xilinx Ltd
21st February 2006

Rev.2

Limitations

This design module is **<u>not</u>** supported by general Xilinx Technical support as an official Xilinx Product.
Please refer any issues initially to the provider of the module.

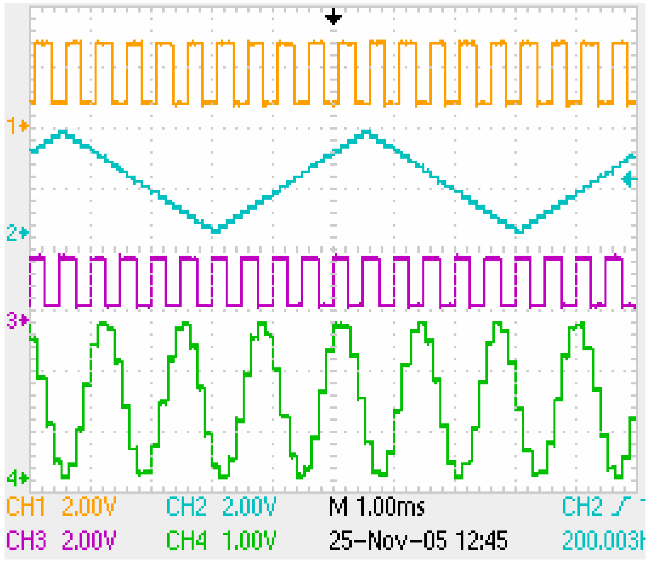
Any problems or items felt of value in the continued improvement of KCPSM3 or this reference design would be gratefully received by the author.

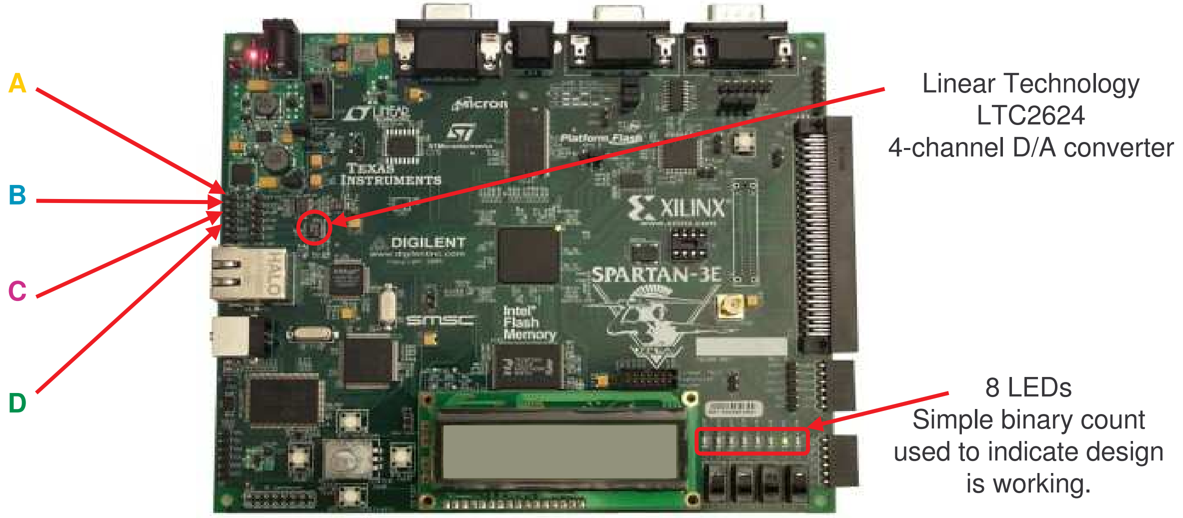
Ken Chapman
Senior Staff Engineer – Spartan Applications Specialist
email: chapman@xilinx.com

The author would also be pleased to hear from anyone using KCPSM3 or the UART macros with information about your application and how these macros have been useful.

Design Overview

This design outlines the fundamental operation of the Linear Technology LTC2624 Digital to Analogue (D/A) converter provided on the Spartan-3E Starter Kit and illustrates how it can be controlled using PicoBlaze. It is hoped that the design may form the basis for future PicoBlaze designs as well as provide a general introduction to the A/D converter requirements for other design implementations. Some exercises are suggested to encourage further self study.





A
B
C
D
Linear Technology LTC2624 4-channel D/A converter
8 LEDs Simple binary count used to indicate design is working.

There are four analogue outputs provided on J5. As well as providing the SPI communication, PicoBlaze is synthesising different waveforms at a sample rate of 8KHz which can be viewed on an oscilloscope.

Channel A = 2KHz square wave (0.66v to 2.64v)

Channel B = 200Hz triangle wave (0v to 3.29v)

Channel C = 2KHz square wave (0.50v to 2.00v)

Channel D = 770Hz sine wave (i.e. one of the DTMF tones)

Try it now – it only takes 30 seconds!

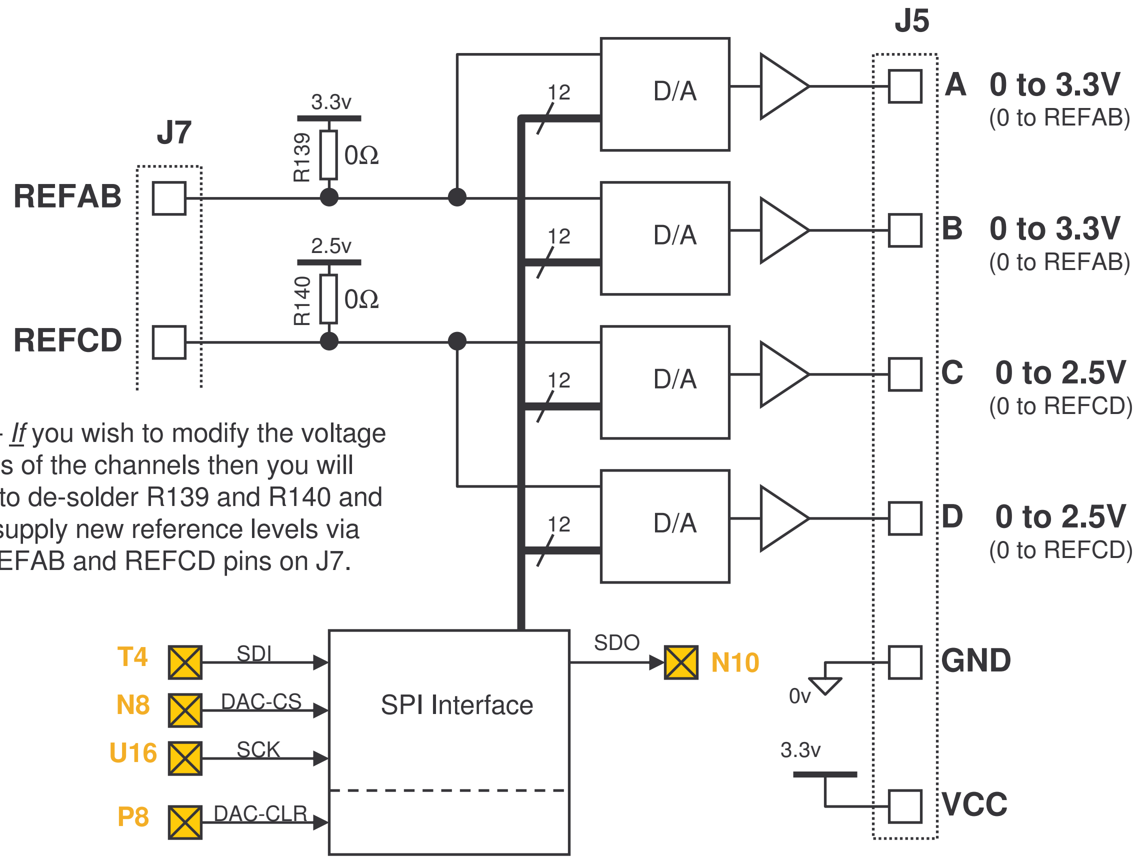
As well as the source design files, a compiled configuration bit file is provided which you can immediately download into the Spartan XC3S500E device on your board. It is recommended that you try this to become familiar with what the design does before continuing to read. To make this task really easy the first time, unzip all the files provided into a directory and then….

double click on 'install_picoblaze_dac_control.bat'.

Assuming you have the Xilinx software installed, your board connected with the USB cable and the board powered (don't forget the switch), then this should open a DOS window and run iMPACT in batch mode to configure the Spartan-3E with the design.

D/A Overview

This overview is not intended to replace the Linear Technology data sheet and you are recommended to consult the LTC2624 data sheet to review the full range of features it offers as well as check the operating specifications particularly in relation to analogue outputs.



Hint – *If* you wish to modify the voltage ranges of the channels then you will need to de-solder R139 and R140 and then supply new reference levels via the REFAB and REFCD pins on J7.

The LTC2624 provides 4 independent analogue outputs which are available on the J5 connector.

Each output is the analogue equivalent of a 12-bit unsigned digital value.

The 'A' and 'B' outputs have the range 0v to 3.3V (on an unmodified board).

The 'C' and 'D' outputs have the range 0v to 2.5V (on an unmodified board).

There are also convenient GND and Vcc (3.3v) pins provided on J5.

The device has a **S**erial **P**eripheral **I**nterface (SPI) to allow the digital values to be supplied and each channel to be controlled. There is also a direct reset control. All of these signals link directly to the Spartan-3E using the pins indicated in yellow.

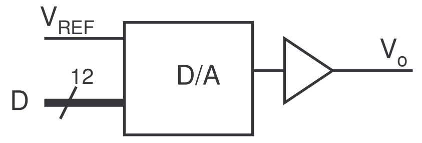
Important hint - The SPI bus signals are shared by other devices on the board.
It is vital that other devices are disabled when communicating with the D/A.

Digital to Analogue

Each output level is the analogue equivalent of a 12-bit unsigned digital value written to the device via the SPI interface.

The ideal conversion is described by the following simple equation….

<u>Hint</u> – 12-bits can represent values 0 to 4095 (000 to FFF hex).



$$Vo = \frac{D}{4096} \times V_{REF}$$

The reference voltage (V_{REF}) is set on the Starter Kit board by REFAB and REFCD connections on J7 but is already provided with a default value (see previous page) by connection to the 3.3v and 2.5v supply rails respectively. Therefore the unmodified board will have the following ideal characteristics…

Channels 'A' and 'B'

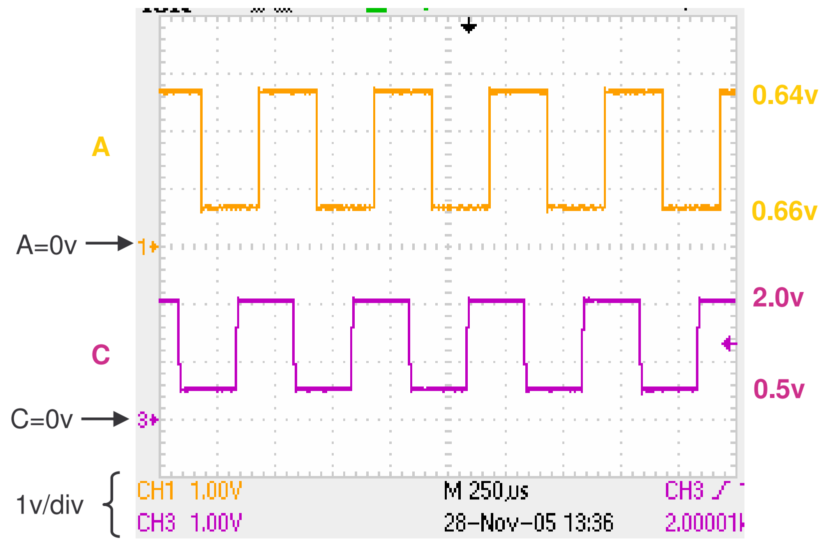
V_{REF} = 3.3v

V_o = 0.000v to 3.299v

Channels 'C' and 'D'

V_{REF} = 2.5v

V_o = 0.000v to 2.499v



<u>Note</u> - In practice, the supply voltage will be slightly different and this will modify the actual range of each output. Also any fluctuation or noise present in the supply rails will be reflected in the analogue levels.

The design provides an example of the different analogue levels using the 'A' and 'C' channels. In both cases, the amplitude description for each waveform is identical, but as this plot shows, the 'A' waveform is of a larger amplitude.

PicoBlaze initially sets 'C' channel to have the levels 0.5v and 2.0v.

$$D = \frac{4096}{V_{REF}} \times Vo$$

$D = (4096/2.5) \times 0.5 = 819$ (333 hex)

$D = (4096/2.5) \times 2.0 = 3277$ (CCD hex)

PicoBlaze then uses these same values to set channel 'A' resulting in different levels just because of the difference in reference voltage.

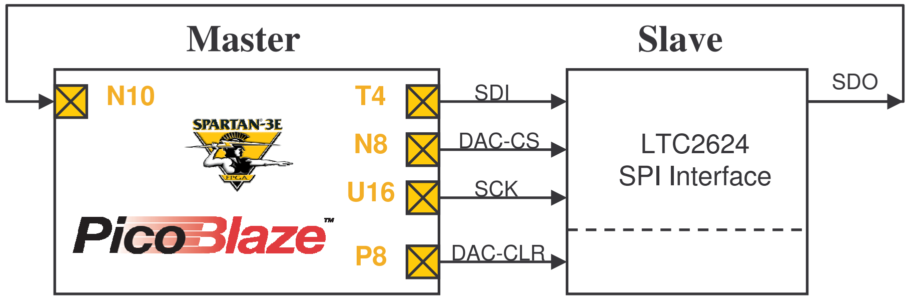
$$Vo = \frac{D}{4096} \times V_{REF}$$

$V_o = (819/4096) \times 3.3 = 0.66$v

$V_o = (3277/4096) \times 3.3 = 2.64$v

SPI Communication

The **S**erial **P**eripheral **I**nterface (SPI) is formally described as being a full-duplex, synchronous, character-oriented channel employing a 4-wire interface. As each bit is transmitted by the master, the slave also transmits a bit allowing one byte to be passed in each direction at the same time. In this case the PicoBlaze in the Spartan-3E is the master and the SPI D/A Converter is the slave.

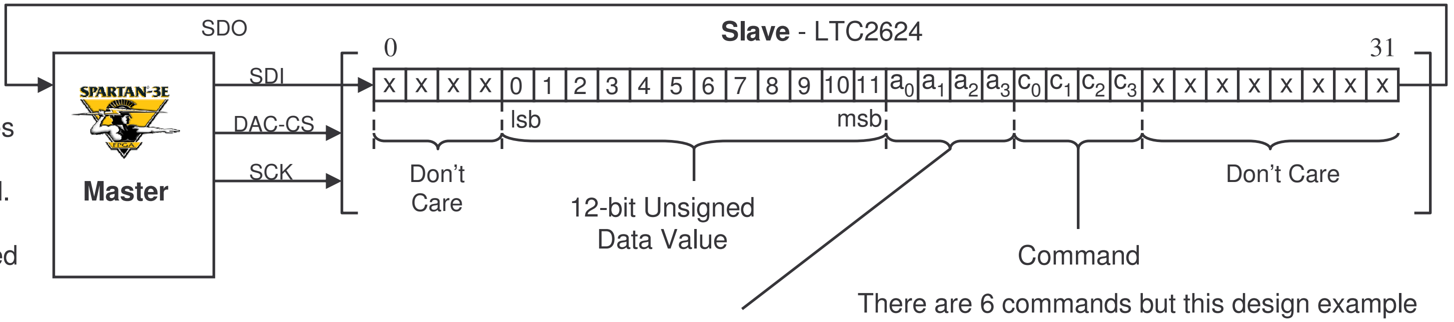


DAC-CS

Communication is only possible with the LTC2624 device when the select signal (DAC-CS) is Low. Therefore the PicoBlaze master is responsible for driving DAC-CS Low before transmitting and receiving a command word and then driving DAC-CS High. It is the act of driving DAC-CS High which actually causes the D/A converter to execute the operation.

Looking specifically at the LTC2624 D/A converter, each communication is formed of 4 bytes or 32-bits. Inside the D/A converter, the SPI interface is formed by a 32-bit shift register. As a new 32-bit command word formed of command, address and data fields is transmitted to it, the 32-bit word previously sent is echoed back to the master. In order to use the D/A converter this response can be ignored, however, it is a useful to confirm correct communication is taking place and it is read back in the supplied PicoBlaze code.

PicoBlaze needs to transmit 4 bytes (each with most significant bit first) to create the 32-bit command word. So bit 31 is transmitted and received first and bit 0 is transmitted and received last.



SDO
Slave - LTC2624
0
31
SPARTAN-3E
Master
SDI
DAC-CS
SCK
x x x x 0 1 2 3 4 5 6 7 8 9 10 11 a0 a1 a2 a3 c0 c1 c2 c3 x x x x x x x x
lsb
msb
Don't Care
12-bit Unsigned Data Value
Don't Care
Command

When all bits have been transmitted and the DAC-CS driven High, the D/A interprets the bits as described here.

Address

a_3	a_2	a_1	a_0	
0	0	0	0	DAC A
0	0	0	1	DAC B
0	0	1	0	DAC C
0	0	1	1	DAC D
1	1	1	1	All

Command

There are 6 commands but this design example uses only the following….

C[3:0]="0011"

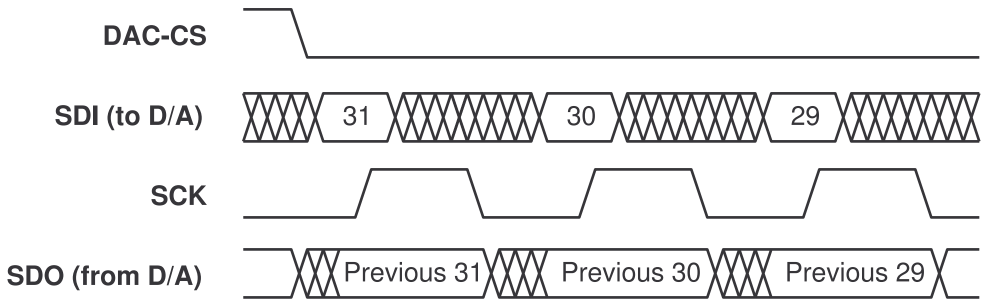
This instructs the D/A to use the 12-bit value supplied to drive the analogue output specified by the address immediately.

SPI Communication Detail

Each bit is transmitted or received relative to the SCK clock. The system is fully static and any clock rate up to the maximum of 50MHz supported by the LTC2624 is possible**. Remember to check all timing parameters in the LTC2624 data sheet if you intend working at or close to the maximum speed.

**Only applies to transmitting (see page9)

The LTC2624 captures data (SDI) on the rising edge of SCK, so the data needs to be valid for at least ±4ns relative to the rising edge.



Note that bit 31, the MSB, is the first to be transmitted after DAC-CS is driven Low.

The LTC2624 changes the output data (SDO) in response to the falling edge of SCK allowing the master to read the value at or near the next rising edge. It is important to notice that SDO must be read on the first clock following the enable being asserted (DAC-CS='0') otherwise bit 31 will be missed.

In theory the SPI interface allows command words to be transmitted at a rate slightly higher than 1.5 M-words/second. Even if this is used to set all four channels individually, this rate would exceed the conversion rate actually supported by the D/A converter and obviously some spacing between commands would be necessary.

Software Exercise

The PicoBlaze code provided (see next page) supports a communication rate of approximately 78 k-words/second. This would be adequate for 8KHz sample rates used in telecommunications but would still occupy over 40% of the processor time just to set 4 channels as well as occupying the SPI bus preventing access to the other devices which share it.

1) Remove the ability to read back the echo response from the LTC2624 and calculate/measure the new maximum word rate.
2) The LTC2624 can be set using only 24-bit command words (although not ideal if wanting to read the echo response). Modify the code to only transmit 24-bit command words without read back and again calculate/measure the new maximum word rate.

Hardware Exercise

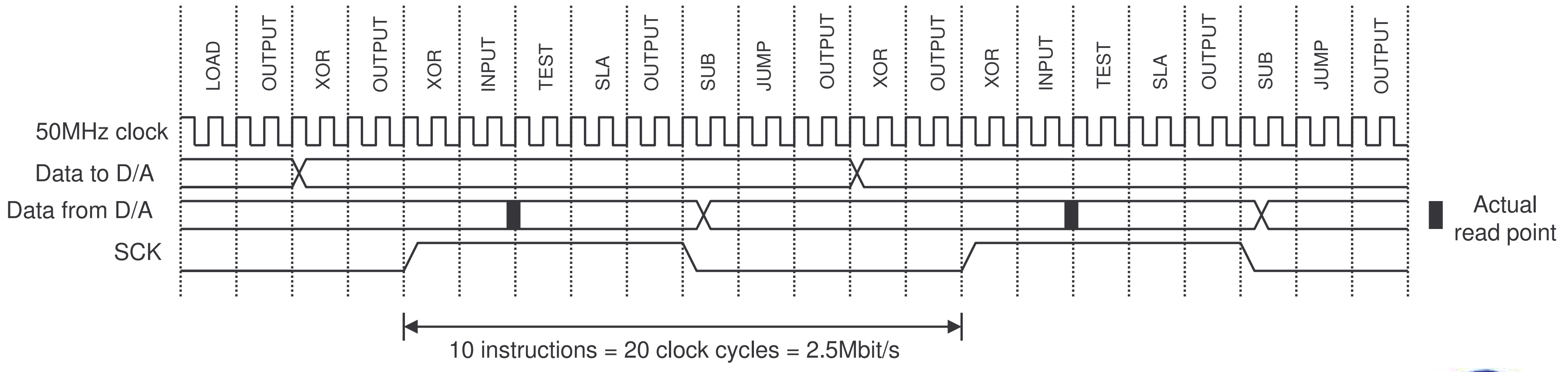
Implement a hardware interface which allows PicoBlaze to OUTPUT and INPUT complete bytes of the command word and accelerates the bit rate of the serial communication. Do NOT implement a 24-bit or 32-bit shift register in hardware which would be unnecessarily large.

Software SPI Communication

PicoBlaze is used to implement the SPI communication 100% in software. The fundamental byte transfer routine is shown below. The ‘s2’ register is used to supply the data for transmission and this will be replaced by the data received from the D/A converter.

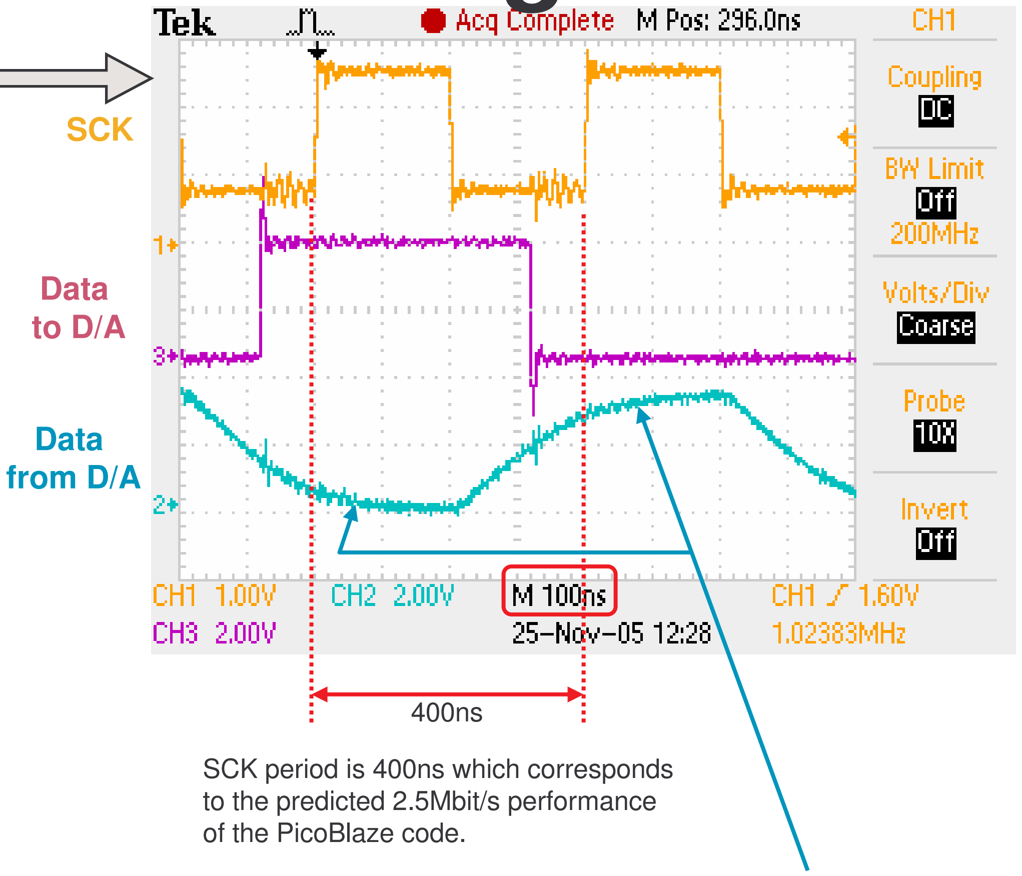
```
   SPI_dac_tx_rx: LOAD s1, 08                      ;8-bits to transmit and receive
next_SPI_dac_bit: OUTPUT s2, SPI_output_port       ;output data bit ready to be used on rising edge
                  XOR s0, SPI_sck                  ;clock High (bit0)
                  OUTPUT s0, SPI_control_port      ;drive clock High
                  XOR s0, SPI_sck                  ;prepare clock Low (bit0)
                  INPUT s3, SPI_input_port         ;read input bit
                  TEST s3, SPI_sdi                 ;detect state of received bit
                  SLA s2                           ;shift new data into result and move to next transmit bit
                  OUTPUT s0, SPI_control_port      ;drive clock Low
                  SUB s1, 01                       ;count bits
                  JUMP NZ, next_SPI_dac_bit        ;repeat until finished
                  RETURN
```

The routine generates SCK. Since every PicoBlaze instruction executes in 2 clock cycles and the design uses the 50MHz clock source on the board, the actual SPI bit rate can be determined. Although this is not as fast as the hardware can support, it keeps the design small and flexible.



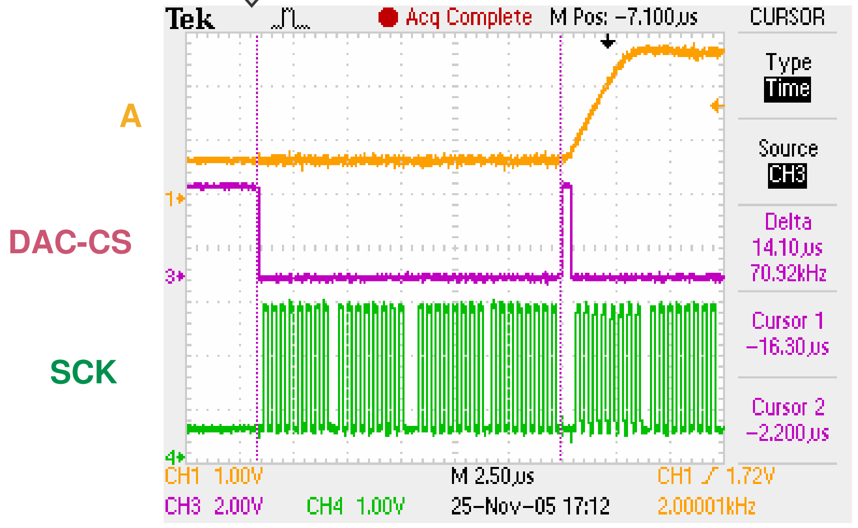
SPI Communication Signals

Oscilloscope trace of the SPI signals observed at the J12 connector shows the timing of the SCK and data signals.



SCK period is 400ns which corresponds to the predicted 2.5Mbit/s performance of the PicoBlaze code.

Oscilloscope trace showing analogue output ‘A’ relative to the SPI signals.
Hint - Always exploit programmable logic during development. In this design, DAC-CS is duplicated to the IO11 pin on J4 to make probe connection easy.



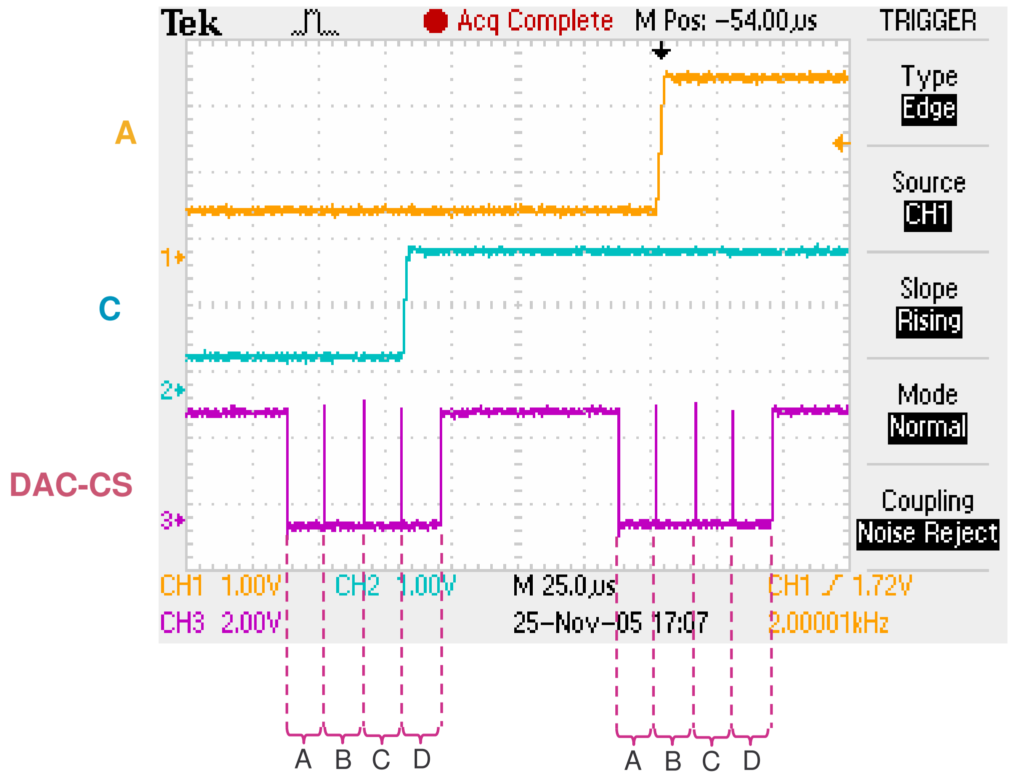
This trace shows the DAC-CS going Low followed by the 32 SCK cycles associated with the command word for the ‘A’ output. When DAC-CS is driven High, the D/A conversion starts and the analogue output responds. Note that it takes approximately 14µs for the SPI communication of each command and the analogue output takes ~3µs to rise (see data sheet for actual specification). Hint – These factors limit the maximum sample rate.

PicoBlaze is allowing 140ns after the falling edge of SCK before sampling SDO. This allows for the slow rise and fall time of the SDO signal on the board which is mainly due to the low drive strength the LTC2624 device (in the region of ±100µA is implied in the data sheet). The principle purpose of the SDO output on the D/A device is to cascade LTC2624 devices and not really for bidirectional communication.
Hint – Use 24-bit commands and do not implement read back when higher sample rates are required.

Sample Rate and Timing

The design contains a simple counter which interrupts PicoBlaze at every 125µs (8KHz). Each time PicoBlaze receives an interrupt, it transmits the command words to set 'A', 'B', 'C' and 'D' on the D/A converter.

Because the SPI communication is predicable, each channel will update at the 8KHz sample rate. However, it can be seen that relative to each other, the channels change at a different time as each 32-bit command transfer completes.



DAC-CS is driven Low for each 32-bit command. As it is driven High at the end of each command the corresponding analogue output changes. The square waves on the 'A' and 'C' outputs clearly show the points of change.

<u>Exercise</u>

Modify the design such that all analogue outputs change simultaneously. This would be vital when phase relationships are important. Demonstrate your design is working by showing that the square waves are exactly 90˚ different in phase.

Hint - This design is using the D/A command C[3:0]="0011" which sets and updates each channel immediately. Investigate the command which allows new digital values to be written without updating the analogue output and the command which updates all outputs simultaneously.

Design Files

The source files provided for the reference design are…..

picoblaze_dac_control.vhd	Top level file and main description of hardware. Contains I/O required to disable other device on the board which may otherwise interfere with the SPI D/A communication.
picoblaze_dac_control.ucf	I/O constraints file for Spartan-3E Starter Kit and timing specifications for 50MHz clock.
kcpsm3.vhd	PicoBlaze processor for Spartan-3E devices.
dac_ctrl.vhd	Assembled program for PicoBlaze (stored in a Block memory)
dac_ctrl.psm	PicoBlaze program source assembler code

Note: The file shown in **green** is not included with the reference design as it is provided with PicoBlaze download. Please visit the PicoBlaze Web site for your free copy of PicoBlaze, assembler, JTAG_loader and documentation.

www.xilinx.com/picoblaze

PicoBlaze Design Size

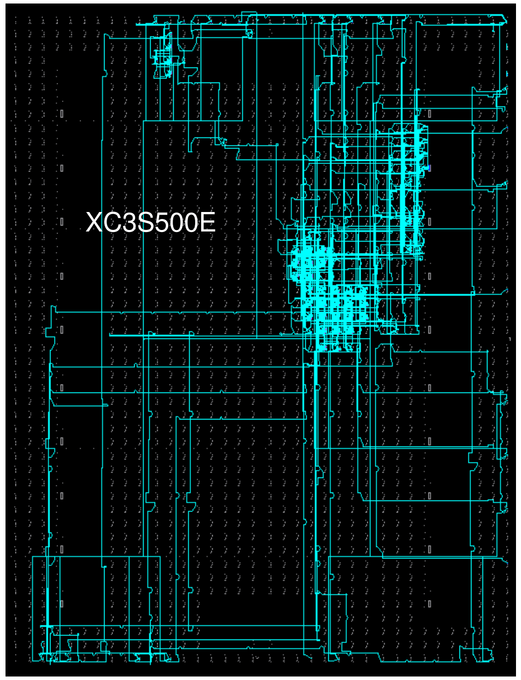
The images and statistics on this page show that the design occupies just 107 slices and 1 BRAM. This is only 2.5% of the slices and 5% of the BRAMs available in an XC3S500E device and would still be less than 12% of the slices in the smallest XC3S100E device.

MAP report

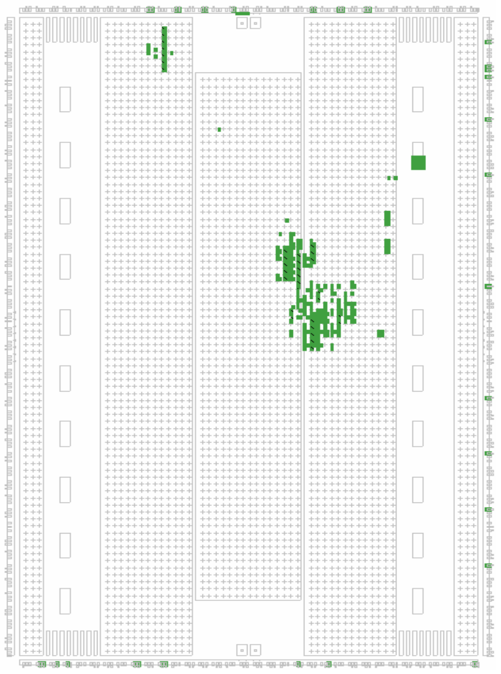
```
Number of occupied Slices:      116 out of    4,656     2%
Number of Block RAMs:             1 out of       20     5%

Total equivalent gate count for design:  75,911
```

PicoBlaze makes extensive use of the distributed memory features of the Spartan-3E device leading to very high design efficiency. If this design was replicated to fill the XC3S500E device, it would represent the equivalent of over 1.5 million gates. Not bad for a device even marketing claims to be 500 thousand gates ☺

FPGA Editor view



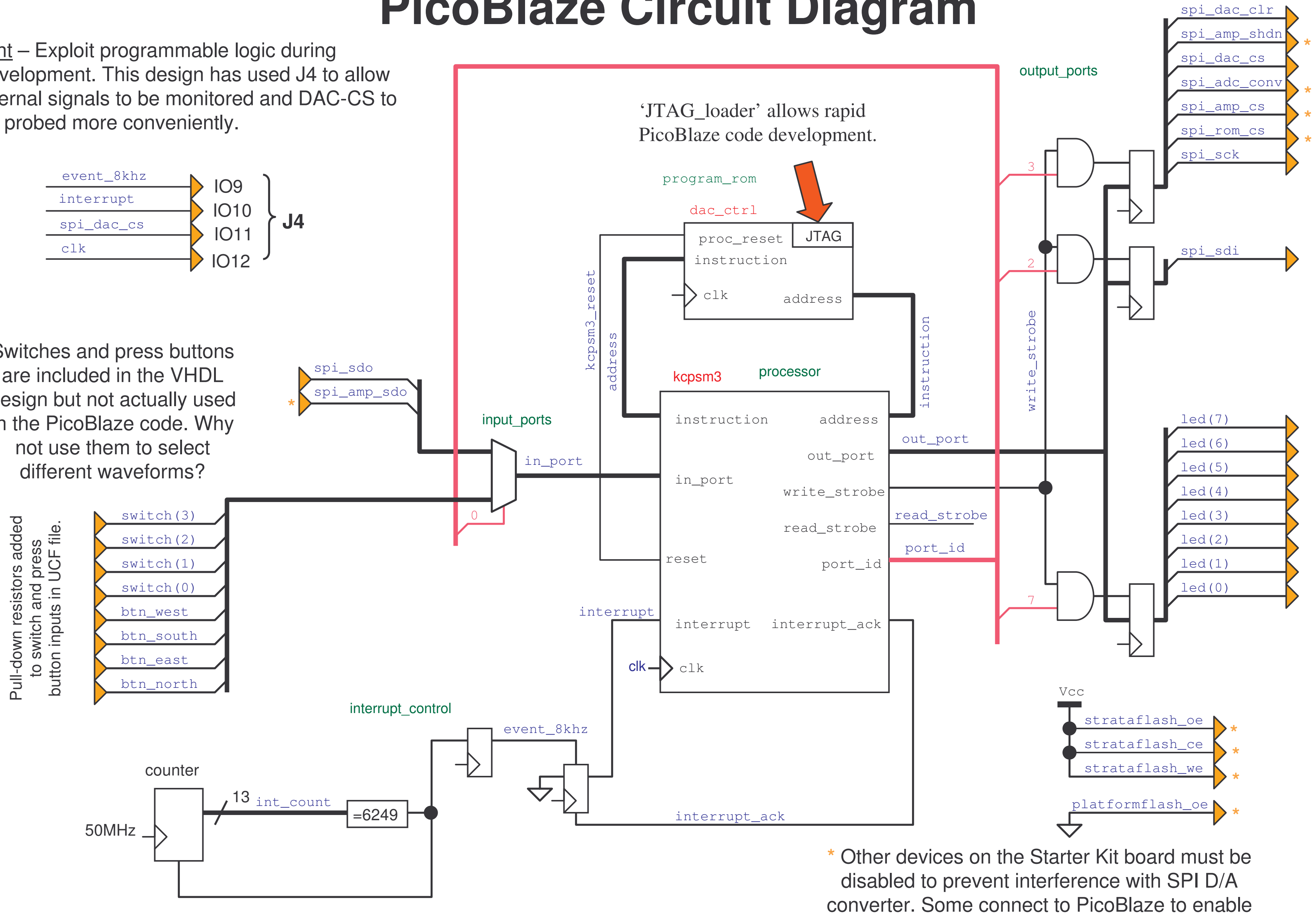
Floorplanner view



PicoBlaze Circuit Diagram

Hint – Exploit programmable logic during development. This design has used J4 to allow internal signals to be monitored and DAC-CS to be probed more conveniently.

Switches and press buttons are included in the VHDL design but not actually used in the PicoBlaze code. Why not use them to select different waveforms?

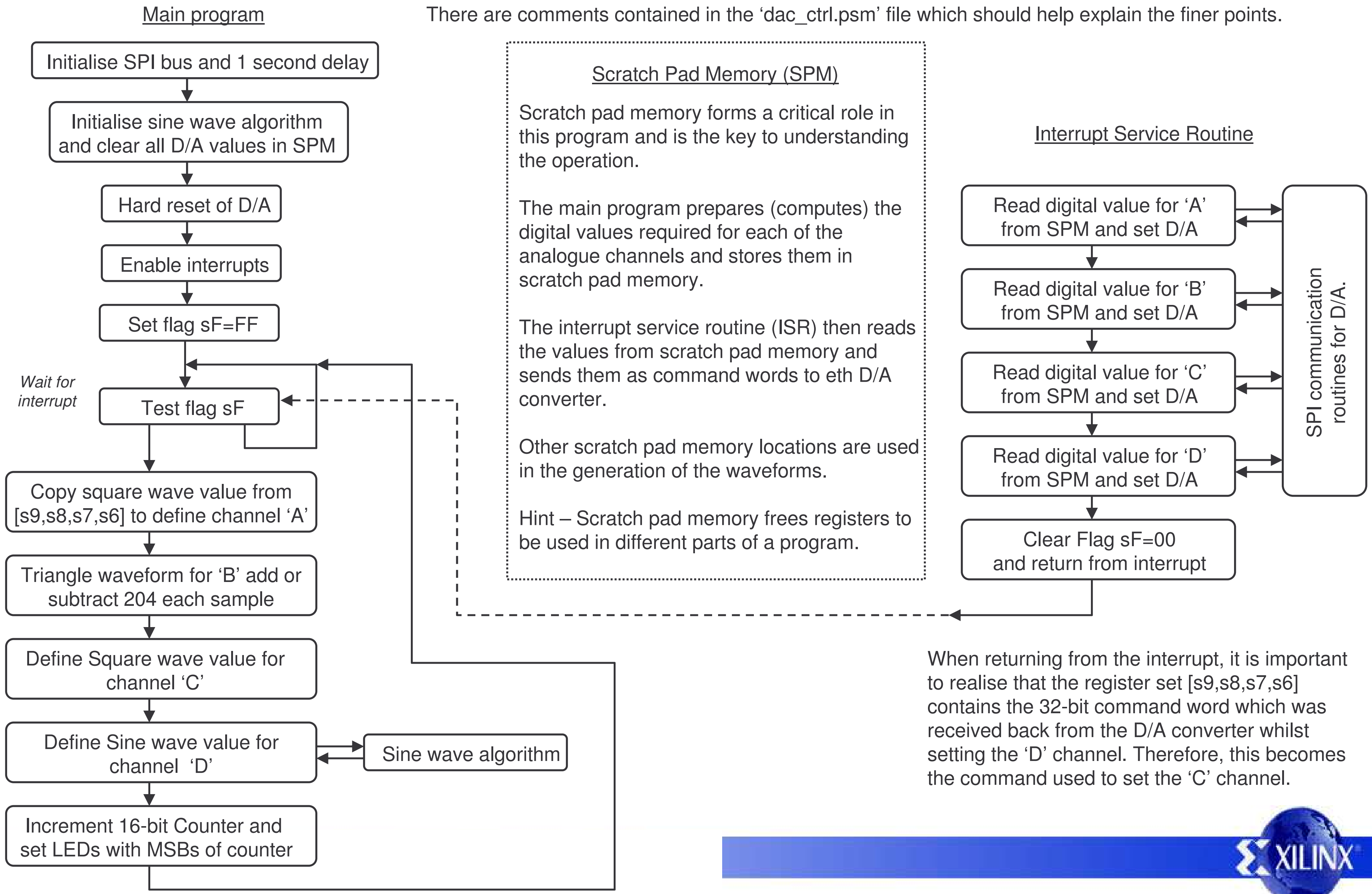


* Other devices on the Starter Kit board must be disabled to prevent interference with SPI D/A converter. Some connect to PicoBlaze to enable software to disable or use for future development.

PicoBlaze Program

This information is intended to give a guide to the way in which the PicoBlaze assembler code is organised. It is not intended to be a lesson in how to write assembler code or explain how PicoBlaze works. Please refer to the documentation for PicoBlaze (KCPSM3).

There are comments contained in the 'dac_ctrl.psm' file which should help explain the finer points.


Main program
Initialise SPI bus and 1 second delay
Initialise sine wave algorithm and clear all D/A values in SPM
Hard reset of D/A
Enable interrupts
Set flag sF=FF
Wait for interrupt
Test flag sF
Copy square wave value from [s9,s8,s7,s6] to define channel 'A'
Triangle waveform for 'B' add or subtract 204 each sample
Define Square wave value for channel 'C'
Define Sine wave value for channel 'D'
Sine wave algorithm
Increment 16-bit Counter and set LEDs with MSBs of counter
Interrupt Service Routine
Read digital value for 'A' from SPM and set D/A
Read digital value for 'B' from SPM and set D/A
Read digital value for 'C' from SPM and set D/A
Read digital value for 'D' from SPM and set D/A
Clear Flag sF=00 and return from interrupt
SPI communication routines for D/A.

Scratch Pad Memory (SPM)

Scratch pad memory forms a critical role in this program and is the key to understanding the operation.

The main program prepares (computes) the digital values required for each of the analogue channels and stores them in scratch pad memory.

The interrupt service routine (ISR) then reads the values from scratch pad memory and sends them as command words to eth D/A converter.

Other scratch pad memory locations are used in the generation of the waveforms.

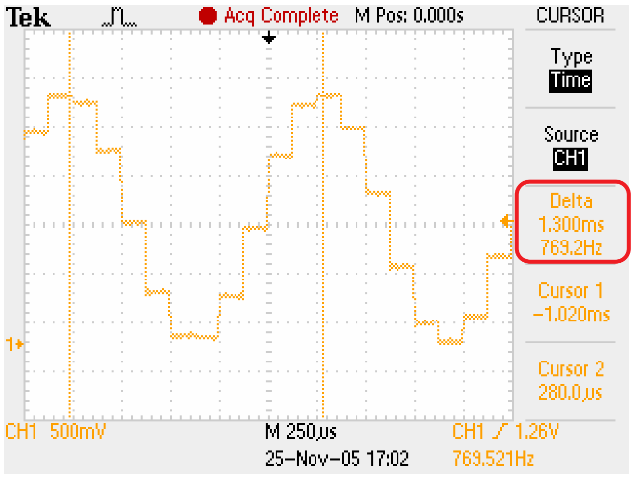
Hint – Scratch pad memory frees registers to be used in different parts of a program.

When returning from the interrupt, it is important to realise that the register set [s9,s8,s7,s6] contains the 32-bit command word which was received back from the D/A converter whilst setting the 'D' channel. Therefore, this becomes the command used to set the 'C' channel.

Sine Wave Generation

PicoBlaze is synthesizing the 770Hz sine wave which is output on channel D using a small algorithm. This is a very efficient way of generating sinusoidal waveforms and avoids the requirement for large sine lookup tables held in memory.

This oscilloscope trace of the 'D' output shows that the fundamental frequency is the specified 770Hz. The waveform looks like 'steps' simply because the 8KHz sample rate means that there are only just over 10 samples per output cycle. Notice how the levels forming are different in each of the cycles.

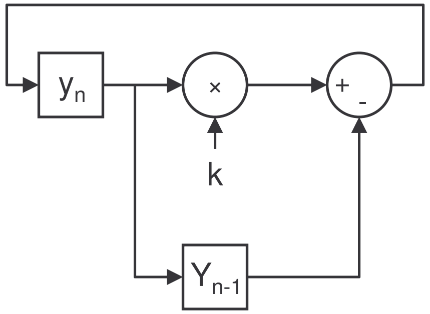


One way to make the output waveform smoother would be to increased the sample rate. However, a more practical solution is to use a simple analogue low pass filter formed by a resistor and capacitor to act as a smoothing circuit. In audio communications systems, it is often the bandwidth limitations of the system itself which naturally performs the smoothing.

Exercise – Design a suitable R-C low pass filter to smooth this waveform.

The sine wave is generated using a recursive algorithm. The art was to quantise the values to 16-bits; the details of which can be seen in the assembly code.

$$k = \frac{2}{\sqrt{1 + \tan^2\left[\frac{2.\pi.f_o}{f_s}\right]}}$$



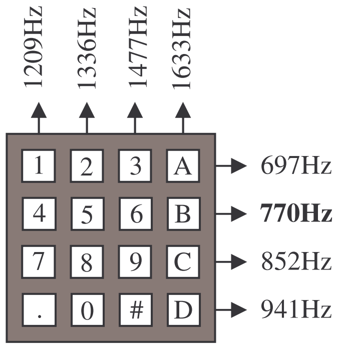
The constant 'k' sets the frequency (f_o) relative to the sample rate (f_s). The current output (y_n) is delayed by one sample (y_{n-1}) before being subtracted from the product of multiplication. Due to the recursive nature, it needs an initial value for y_n to start the oscillation. The magnitude of this initial value will determine the peak amplitude of the waveform (but is not the peak value itself!).

Hint 1 - Use Radians when calculating 'k'.

Hint 2 - Remember that Spartan-3E is the fastest simulator. Use JTAG_loader to reload PicoBlaze and rapidly test different 'k' and initial values.

Exercise
Create a DTMF (Dual Tone Multi Frequency) dialler for your home phone number

Each button corresponds to the addition of two unique tones.


1209Hz
1336Hz
1477Hz
1633Hz
1 2 3 A
4 5 6 B
7 8 9 C
. 0 # D
697Hz
770Hz
852Hz
941Hz