SPI FLASH Programmer
for
Spartan-3E Starter Kit

Ken Chapman
Xilinx Ltd
November 2004

Rev.1

Limitations

Limited Warranty and Disclaimer. These designs are provided to you “as is”. Xilinx and its licensors make and you receive no warranties or conditions, express, implied, statutory or otherwise, and Xilinx specifically disclaims any implied warranties of merchantability, non-infringement, or fitness for a particular purpose. Xilinx does not warrant that the functions contained in these designs will meet your requirements, or that the operation of these designs will be uninterrupted or error free, or that defects in the Designs will be corrected. Furthermore, Xilinx does not warrant or make any representations regarding use or the results of the use of the designs in terms of correctness, accuracy, reliability, or otherwise.

Limitation of Liability. In no event will Xilinx or its licensors be liable for any loss of data, lost profits, cost or procurement of substitute goods or services, or for any special, incidental, consequential, or indirect damages arising from the use or operation of the designs or accompanying documentation, however caused and on any theory of liability. This limitation will apply even if Xilinx has been advised of the possibility of such damage. This limitation shall apply not-withstanding the failure of the essential purpose of any limited remedies herein.

This design module is **not** 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
Staff Engineer - Applications Specialist
email: chapman@xilinx.com

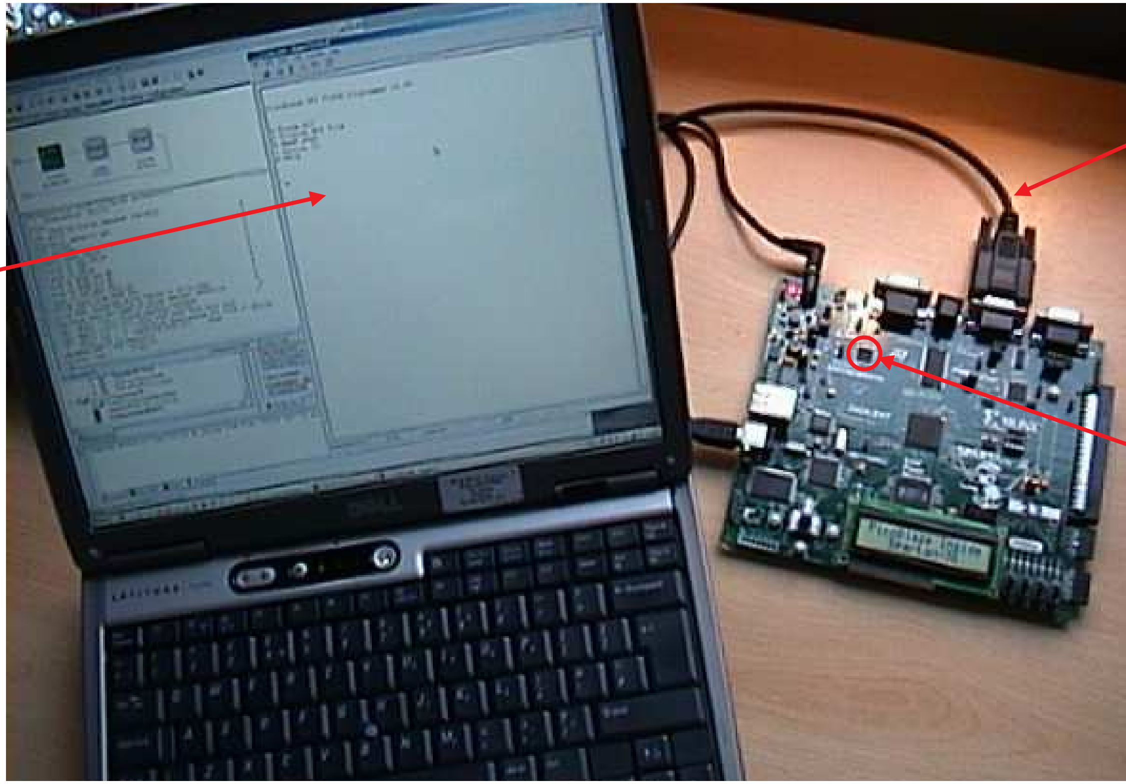
The author would also be pleased to hear from anyone using KCPSM3 or the UART macros with information about your application and how these macros have been useful.

Design Overview

This design will transform the Spartan-3E device on your Spartan-3E Starter Kit into an SPI FLASH programmer. Using a simple terminal program on your PC such as HyperTerminal, you will be able to program the SPI FLASH device with an MCS memory file defining the configuration for the Spartan-3E device as well as perform SPI FLASH memory ID check, bulk erase and read operations.

The design is implemented using a single PicoBlaze processor and UART macros occupying under 5% of the XC3S500E device. It is hoped that the design may be of interest to anyone interested in reading, writing and erasing SPI_FLASH as part of their own applications even if it is not used exactly as provided.

HyperTerminal
(or similar)



RS232
Serial Communication

M25P16
16 Mbit Serial FLASH memory
(STMicroelectronics)

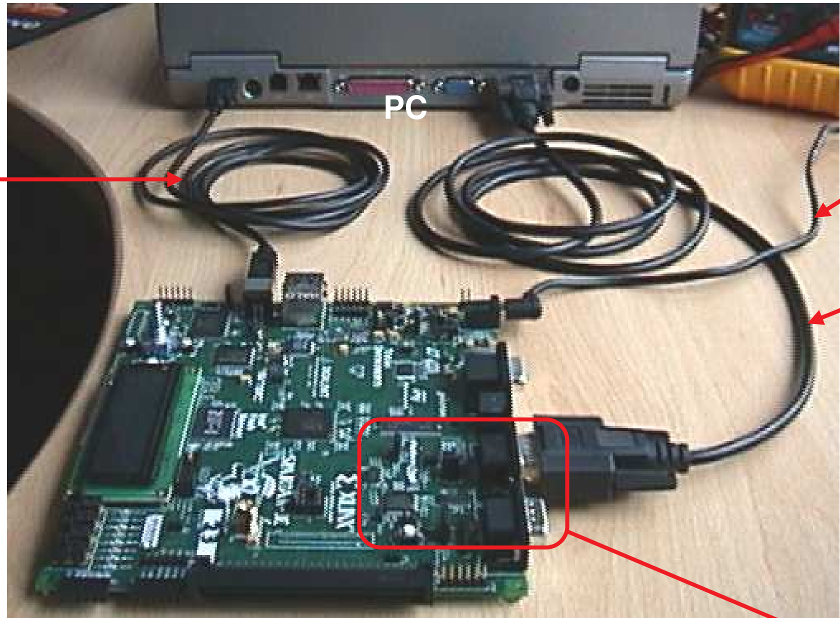
Using the Design

The design is provided as a configuration BIT file for immediate programming of the Spartan XC3S500E provided on the Spartan-3E Starter Kit. Source design files are also provided for those more interested in the intricacies of the design itself. An example MCS programming file is also provided to enable you to verify that your set up is working.

Hardware Setup

USB cable.
Used to configure the Spartan-3E with the PicoBlaze design.

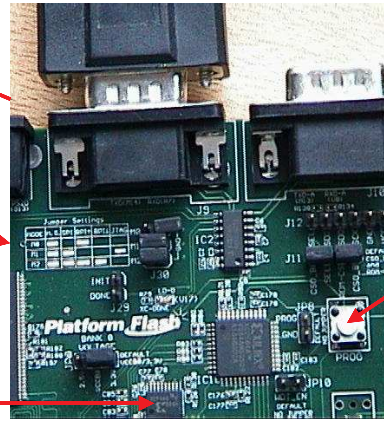
Cable plus devices on board essentially provide the same functionality as a Platform Cable USB to be used in conjunction with iMPACT.



+5v supply
Don't forget to switch on the board too!
(SWP)

RS232 Serial Cable.
Used for programming of the SPI FLASH memory.

Cable connects J9 on the board to your PC serial port. For this you will need a male to female straight through cable (critically pin2-pin2, pin3-pin3 and pin5-pin5).



J30 configuration mode jumpers and selection chart.
It does not matter which settings you have during the JTAG programming of the XC3S500E from via the USB cable but remember to set correctly (M0=open, M1=M2=short) for configuration from the SPI FLASH once it has been programmed (press PROG button or cycle power).

PROG button

Idea – The PicoBlaze SPI programmer design could be programmed into the XCF04S Platform FLASH device so that it can be loaded directly on the board by changing the J30 jumpers.

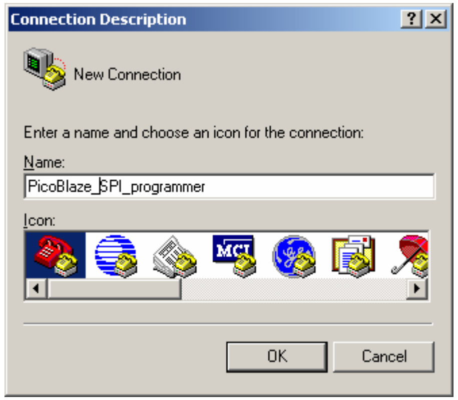
Serial Terminal Setup

Once the design is loaded into the Spartan-3E, you will need to communicate using the RS232 serial link. Any simple terminal program can be used, but HyperTerminal is adequate for the task and available on most PCs.

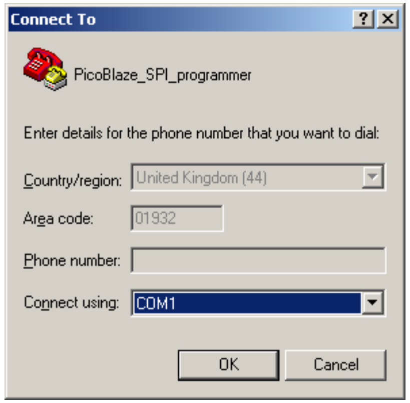
A HyperTerminal configuration file is also provided with this design with the file name ‘PicoBlaze_SPI_programmer.ht’. It should be possible to copy this to a your working directory or to your desktop and then launch HyperTerminal by double clicking on it.

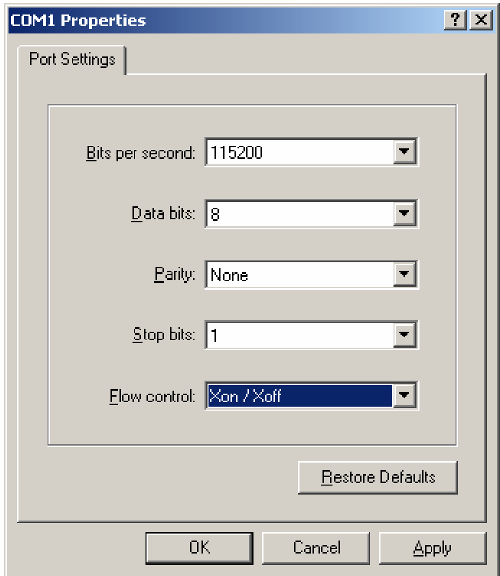
Alternatively a new HyperTerminal session can be started and configured as shown in the following steps. These also indicate the communication settings and protocol required by an alternative terminal utility.

1) Begin a new session with a suitable name.
HyperTerminal can typically be located on your PC at
Programs -> Accessories -> Communications -> HyperTerminal.



2) Select the appropriate COM port (typically COM1 or COM2) from the list of options. Don’t worry if you are not sure exactly which one is correct for your PC because you can change it later.





3) Set serial port settings.

Bits per second : 115200
Data bits: 8
Parity: None
Stop bits: 1
Flow control: **XON/XOFF**

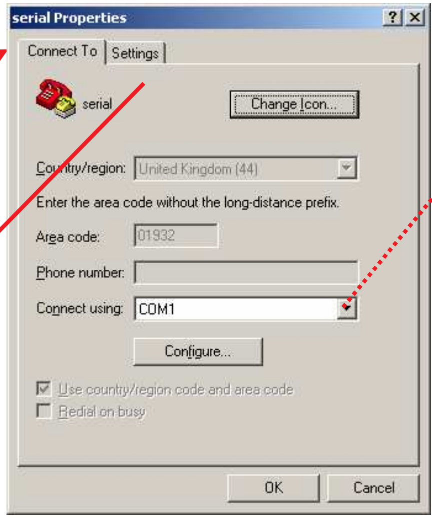
Hint – The design uses XON/XOFF flow control. It may be possible to modify the design and use higher baud rates to reduce SPI programming time .

HyperTerminal Setup

Although steps 1, 2 and 3 will actually create a Hyper terminal session, there are few other protocol settings which need to be set or verified for the PicoBlaze design.

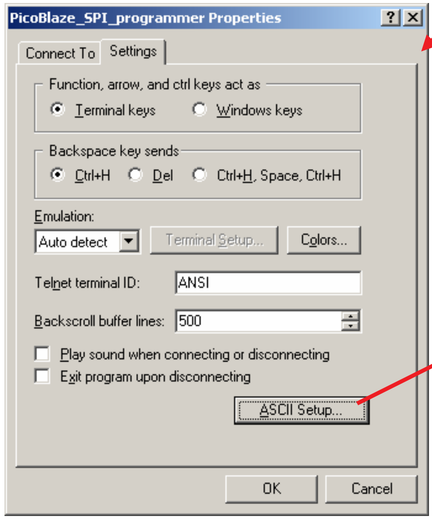
5 - Open the properties dialogue

4 - Disconnect

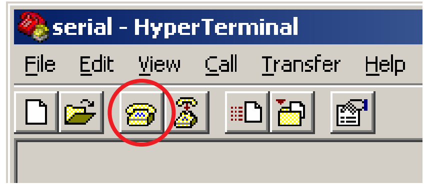


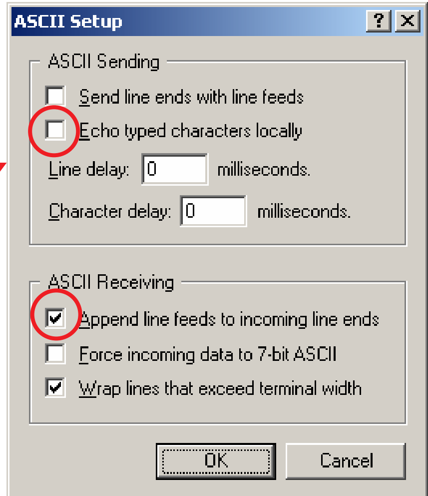
To select a different COM port and change settings (if not correct).

6 - Open Settings



8 - Connect





7 - Open ASCII Setup

Ensure boxes are filled in as shown.

The design will echo characters that you type so you <u>do not</u> need the 'Echo typed characters locally' option.

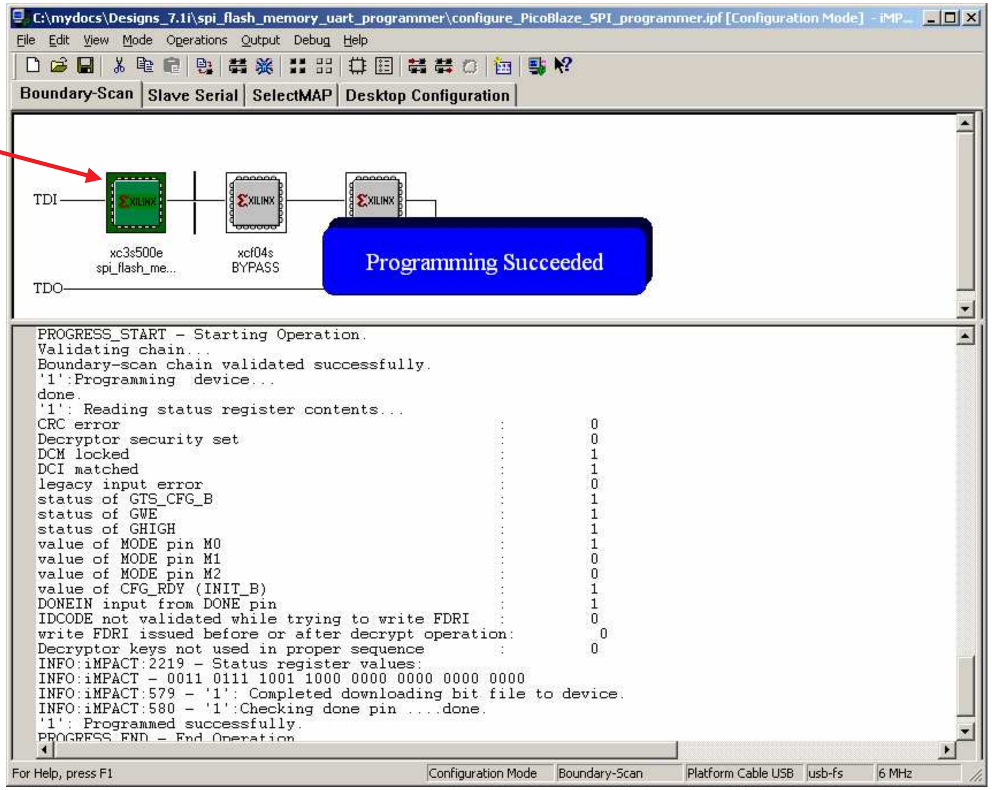
The design transmits carriage return characters (OD_{HEX}) to indicate end of line so you <u>do</u> need the 'Append line feeds to incoming line ends' option to be enabled.

Configure Spartan-3E

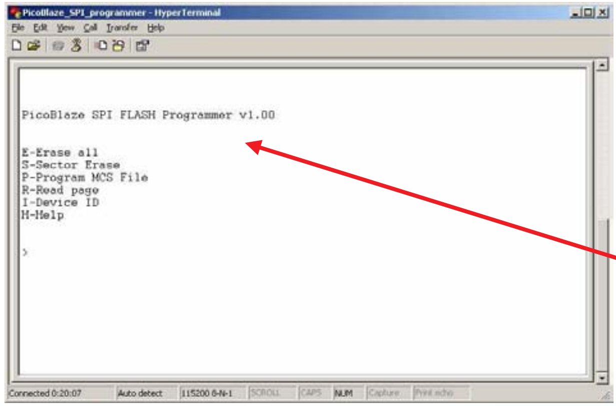
Use iMPACT to configure the XC3S500E device on the Spartan-3E Starter Kit via the USB cable.
An iMPACT project file is provided called ‘configure_PicoBlaze_SPI_programmer.ipf’ or you can set up your own with the BIT file provided.

Configure XC3S500E with provided BIT file ‘spi_flash_memory_uart_programmer.bit’

The other two devices are in BYPASS mode.

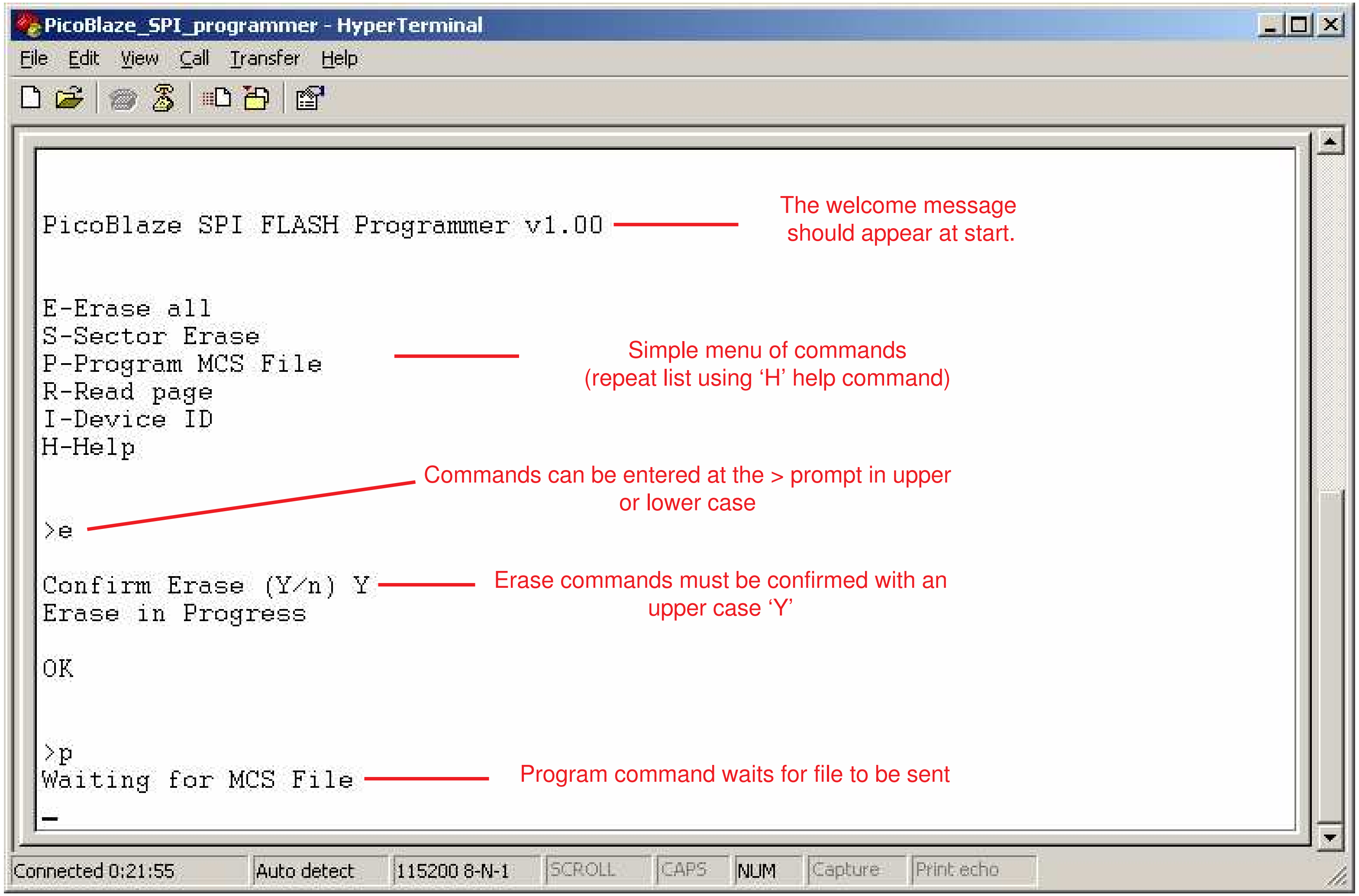


The warning about ‘JtagClk’ can safely be ignored.



Your terminal session should indicate the design is working with a version number and simple menu.

Talking to PicoBlaze



‘H’, ‘I’, ‘E’ and ‘S’ Commands

H – Help command displays the simple menu again.

```
>h

PicoBlaze SPI FLASH Programmer v1.00

E-Erase all
S-Sector Erase
P-Program MCS File
R-Read page
I-Device ID
H-Help
```

I – Read **I**dentification code of the M25P16 ST Microelectronics 16Mbit Serial FLASH memory.

```
>i
ID= 20 20 15
```

This command is a good way to confirm communication with the SPI FLASH is working.
The expected response is 20 20 15 (please see M25P16 data sheet for details)

E – Erase command will perform a bulk erase of the M25P16 device.

```
>e

Confirm Erase (Y/n) Y
Erase in Progress

OK
```

Note that the device will be completely erased using this command and hence you will be asked to confirm the operation with an upper case ‘Y’.

The erase operation can take up to 40 seconds for the SPI FLASH to complete although 20 seconds is more typical (please see M25P16 data sheet for specification and details).

S – Sector Erase command will erase sectors 0 to 5 only. This covers the address range 000000 to 04FFFFFF which is consistent with the storage of a configuration file for the XC3S500E device. This command is faster that the ‘E’ command and will leave the upper memory unchanged

```
>s

Confirm Erase (Y/n) Y
Erase in Progress

OK
```

You will be asked to confirm the operation with an upper case ‘Y’.

The erase operation can take up to 3 seconds per sector (15 seconds total). Typically this command will take 5 seconds to complete.

‘P’ Command

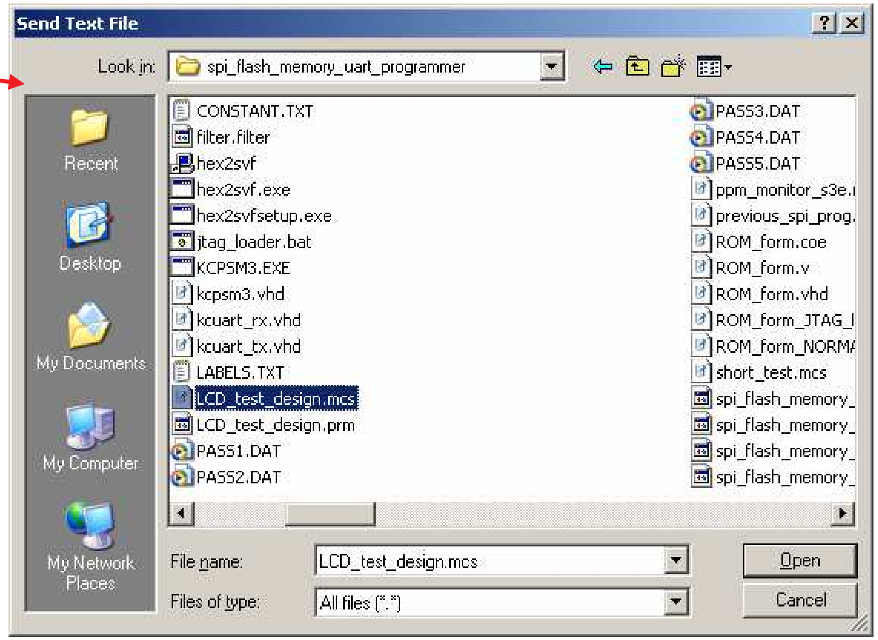
P – Program command.

This is the most important command as it will allow you to program the SPI FLASH device with a configuration bit stream suitable for the XC3S500E to load from at power up or by pressing the PROG button. Later in the documentation we will consider how to prepare an MCS file and what is actually happening, but for now this page shows how to program the provided example file ‘LCD_test_design.mcs’ into the memory.

```
>p
Waiting for MCS File
```

First enter the ‘P’ command and a message prompting you for the MCS file will appear.

In HyperTerminal, select the ‘Transfer’ menu and then select the ‘Send Text File’ option.
(Note: Do not use the ‘Send File’ option)



Navigate to your working directory and select the desired MCS file which in this case is ‘LCD_test_design.mcs’.

You will need to change ‘Files of type’ to ‘All files (*.*)’ to see the MCS files listed.

Once you are happy with your selection click on ‘Open’.

Hint If you accidentally enter the ‘P’ command you can get out by carefully typing the end of file record found in an MCS file which is…..
:00000001FF

‘P’ Command continued

```
045380
045390
0453A0
0453B0
0453C0
0453D0
0453E0
0453F0
045400
045410
045420
045430
045440
045450
045460
045470

OK


>
```

Programming will start immediately and will be indicated by a running display list of hexadecimal numbers.

Each number indicates the address currently being programmed in the SPI FLASH memory as defined in the MCS file. For the XC3S500E the final address displayed is 045470 and hence this can be used to monitor progress.

Programming will typically take **80 seconds** to complete. This time is almost entirely as a result of the RS232 serial interface and why it will be useful to investigate higher baud rates in future.

The programming will complete with ‘OK’ and a return to the > prompt.

It should now be possible to press the PROG button on the board and a simple design (also using a PicoBlaze) will drive the LCD display with some messages and then a free running counter.

Remember, that you will need to reload the SPI Programmer if you want to try any of the other commands.

'R' Command

P – Read page command.

A 'page' is defined as a block of 256 bytes in the SPI FLASH memory (please see M25P16 data sheet for full details).
This command will display any specified block of 256 bytes which can be used (in part) to verify if the device has been programmed or erased correctly.

```
>r
page address=000000

000000  FF FF FF FF AA 99 55 66 30 00 80 01 00 00 00 07
000010  30 01 60 01 00 00 00 60 30 01 20 01 00 00 3F E5
000020  30 01 C0 01 01 C2 20 93 30 00 C0 01 00 00 00 00
000030  30 00 80 01 00 00 00 09 30 00 20 01 00 00 00 00
000040  30 00 80 01 00 00 00 01 30 00 40 00 50 01 14 9A
000050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

OK
```

After entering the 'R' command you will be prompted to enter a page address. You should then enter a 6 digit hexadecimal value.

The M25P16 memory has an address range of 000000 to 1FFFFF.

The display will indicate the address of the first byte on each line followed by 15 bytes from successive memory locations.

Note that the SPI FLASH considers a page to be an address range xxxx00 to xxxxFF. However this read page command is not restricted to these boundaries and will always read sequentially for 256 bytes starting with the address provided.

PicoBlaze rejects incorrect address values but does not support on-line editing in this version. Please just type in a valid address if you are prompted again.

Hint: Data in an erased device will be 'FF' so if you read '00' it has been programmed. It is common for a configuration bit file to contain many '00' bytes especially if the design is relatively small.

MCS files and Device configuration

An MCS file contains additional information to define the storage address which PicoBlaze interprets as well as obtaining the configuration data. How an MCS file defines the addresses is beyond the scope of this document at this time, but in general the first lines of the MCS file defining an FPGA configuration from SPI FLASH will be associated with address zero (000000) and each line contains 16 data bytes to be stored in sequential locations.

If we look at the supplied MCS example file ‘LCD_test_design.mcs’ the first configuration data byes can be identified in each line. Having programmed the SPI FLASH memory, it is possible to read back those same data bytes with the ‘R’ command with page start address ‘000000’.

Start of MCS file with byte data highlighted in blue

```
:020000040000FA
:10000000FFFFFFFF5599AA660C000180000000E089
:100010000C800680000000060C8004800000FCA715
:100020000C800380804304C90C00038000000000A2
:100030000C000180000000900C0004800000000013
:100040000C000180000000800C0002000A8028598A
:1000500000000000000000000000000000000000A0
:100060000000000000000000000000000000000090
:100070000000000000000000000000000000000080
:100080000000000000000000000000000000000070
:100090000000000000000000000000000000000060
:1000A0000000000000000000000000000000000050
:1000B0000000000000000000000000000000000040
:1000C0000000000000000000000000000000000030
:1000D0000000000000000000000000000000000020
:1000E0000000000000000000000000000000000010
:1000F0000000000000000000000000000000000000
etc
```

‘R’ command

```
000000  FF FF FF FF AA 99 55 66 30 00 80 01 00 00 00 07
000010  30 01 60 01 00 00 00 60 30 01 20 01 00 00 3F E5
000020  30 01 C0 01 01 C2 20 93 30 00 C0 01 00 00 00 00
000030  30 00 80 01 00 00 00 09 30 00 20 01 00 00 00 00
000040  30 00 80 01 00 00 00 01 30 00 40 00 50 01 14 9A
000050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

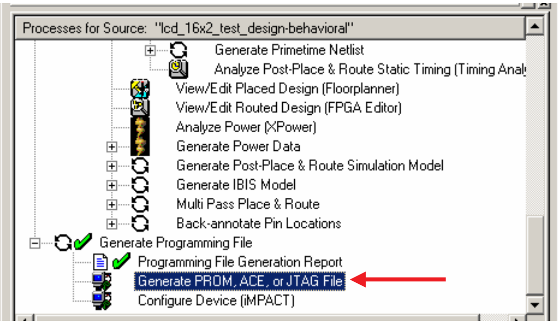
Although the data looks similar, it is not the same. This is because each byte must be bit reversed for configuration of the Spartan-3E device. The MCS file defines the data assuming each byte is serialised LSB first but an SPI FLASH is actually read MSB first. e.g. ‘0C’ becomes ’30’ (‘00001100’ bit reversed is ‘00110000’)

IMPORTANT Note: PicoBlaze is performing the bit reversal operation during programming which allows a standard MCS file to be used. Do not attempt to reverse the bit order in the preparation of the configuration MCS file. Also take care if using this design to store raw data.

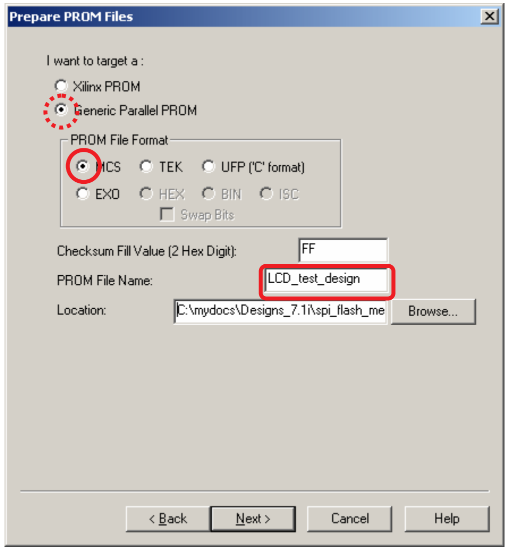
Preparing an MCS file

This design has been provided so that a 'default' MCS programming file generated by the ISE tools can be used. The following indicate how that may be achieved but is not intended to replace existing documentation for PROM generation.

1) Select 'Generate PROM' in Project Manger

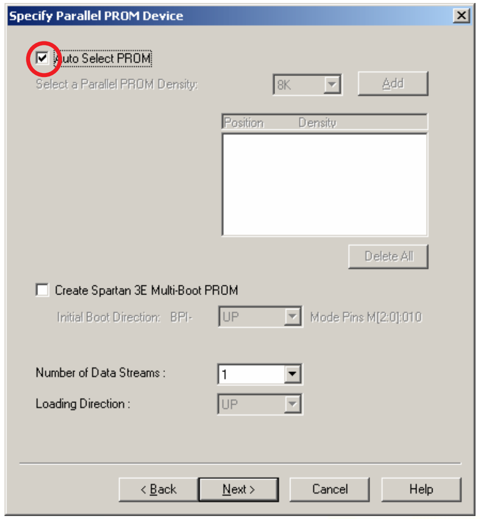


2) Select 'MCS' for the file format and provide a file name.

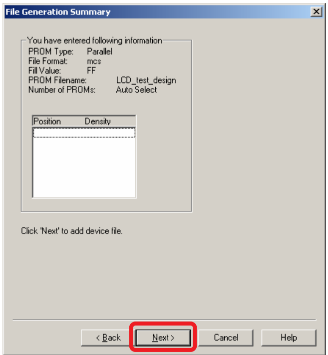


You could generate the file for a 'Xilinx PROM' such as the XCF04S if you like, but this sequence will use the 'Generic Parallel PROM' even though an SPI memory is really serial!

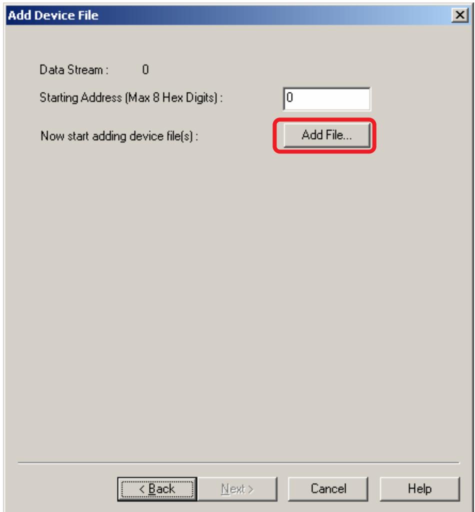
3) 'Auto Select PROM'



4) Confirm file generation

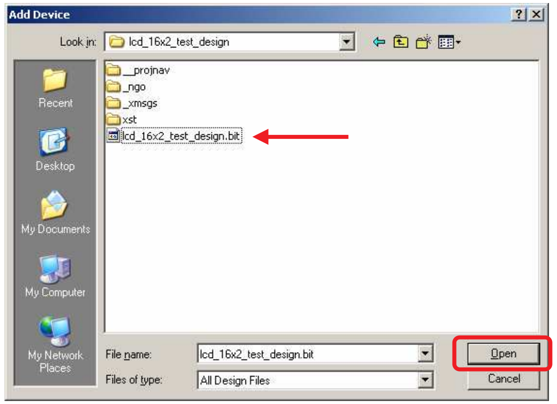


5) Add File….

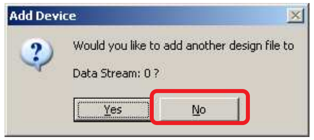


Preparing an MCS file continued

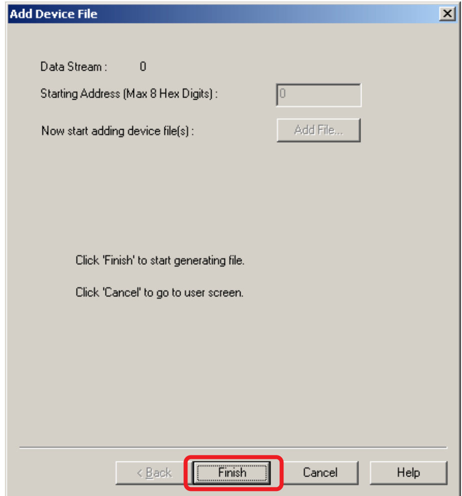
6) Locate the BIT file for your design and 'Open'



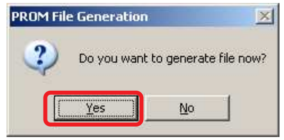
7) No!

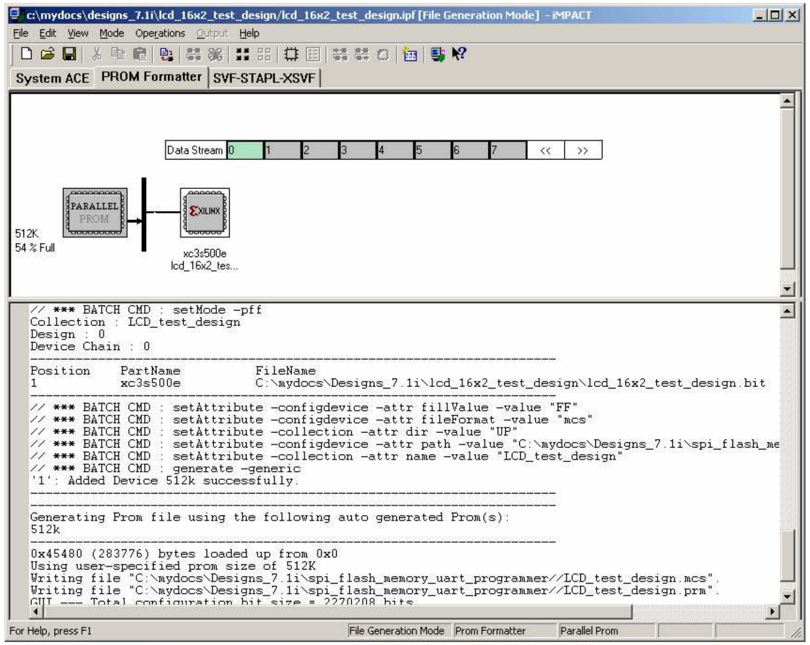


8) Finish



9) Yes!





You now have a suitable MCS file to program into the SPI FLASH memory using PicoBlaze SPI FLASH Programmer.

Design Files

For those interested in the actual design implementation, the following pages provide some details and an introduction to the source files provided. This description may be expanded in future to form a more complete reference design.

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

- **spi_flash_memory_uart_programmer.vhd** — Top level file and main description of hardware. Contains I/O required to disable other device on eth board which may otherwise interfere with the SPI FLASH memory communication.
  - **spi_flash_memory_uart_programmer.ucf** — I/O constraints file for Spartan-3E Starter Kit and timing specifications for 50MHz clock.
  - **kcpsm3.vhd** — PicoBlaze processor for Spartan-3E devices.
  - **spi_prog.vhd** — Assembled program for PicoBlaze (stored in a Block memory)
    - **spi_prog.psm** — PicoBlaze program source assembler code
  - **uart_tx_plus.vhd** — 'uart_tx_plus.vhd' is the same as the 'uart_tx.vhd' supplied with PicoBlaze except that the 'buffer_data_present' signal has also been brought out to provide better support for XON/XOFF flow control.
    - **kc_uart_tx.vhd**
    - **bbfifo_16x8.vhd**
  - **uart_tx.vhd**
    - **kc_uart_tx.vhd**
    - **bbfifo_16x8.vhd**

UART transmitter and receiver with 16-byte FIFO buffers.

Note: Files shown in **green** are not included with the reference design as they are all provided with PicoBlaze download. Please visit the PicoBlaze Web site for your free copy of PicoBlaze, assembler and documentation. www.xilinx.com/picoblaze

PicoBlaze Design Size

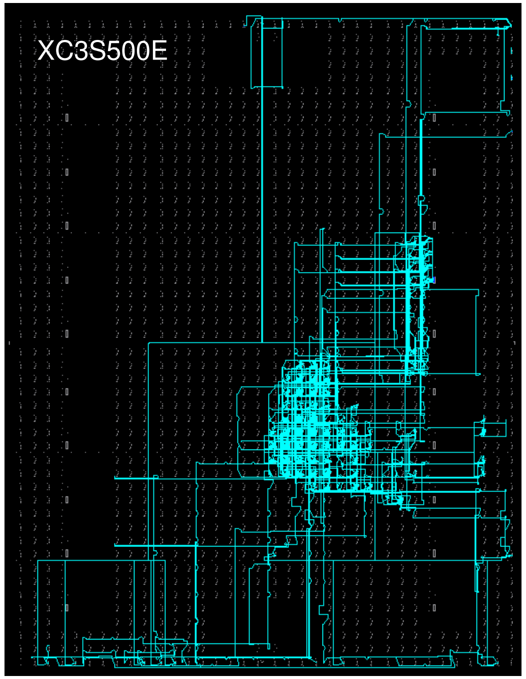
The images and statistics on this page show that the design occupies just 156 slices and 1 BRAM. This is only 3.3% of the slices and 5% of the BRAMs available in an XC3S500E device and would still be less than 17% of the slices in the smallest XC3S100E device.

MAP report

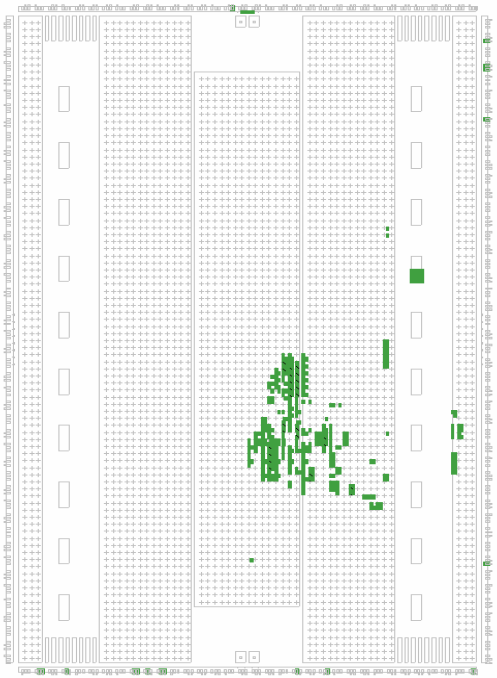
```
Number of occupied Slices:      156 out of    4,656     3%
Number of Block RAMs:             1 out of       20     5%

Total equivalent gate count for design:  78,710
```

PicoBlaze and the UART macros make 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 ☺

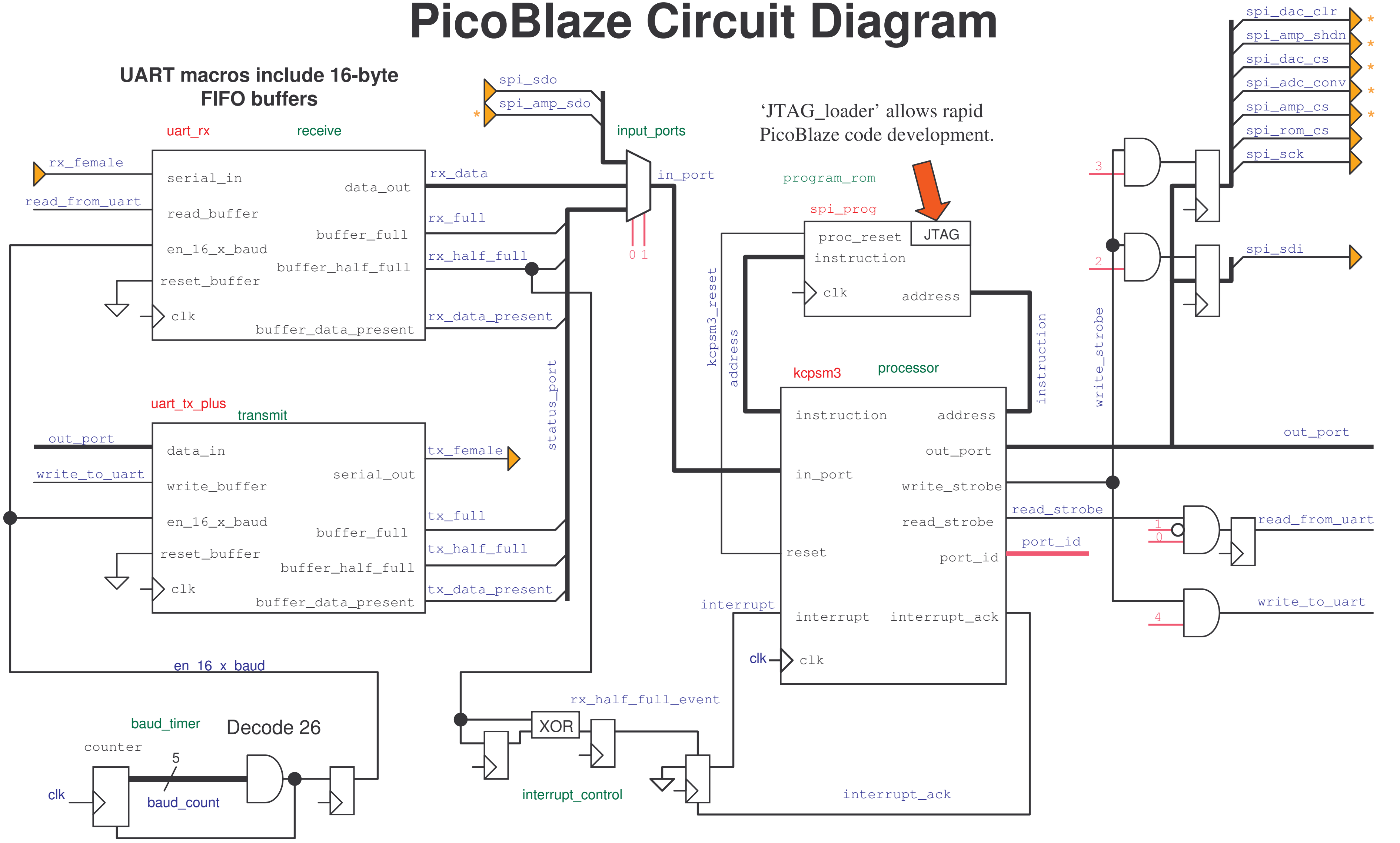
FPGA Editor view

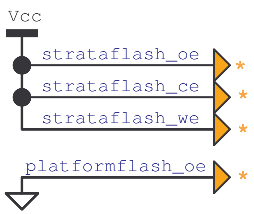


Floorplanner view



PicoBlaze Circuit Diagram

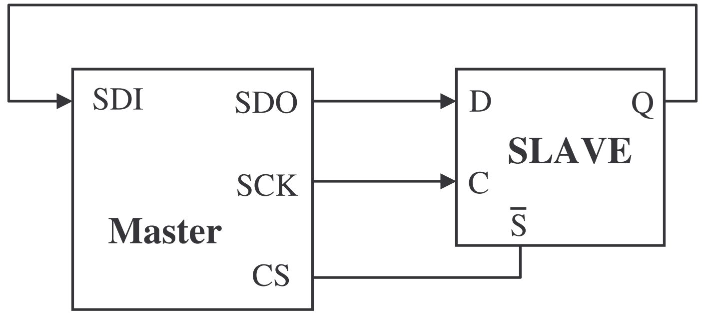


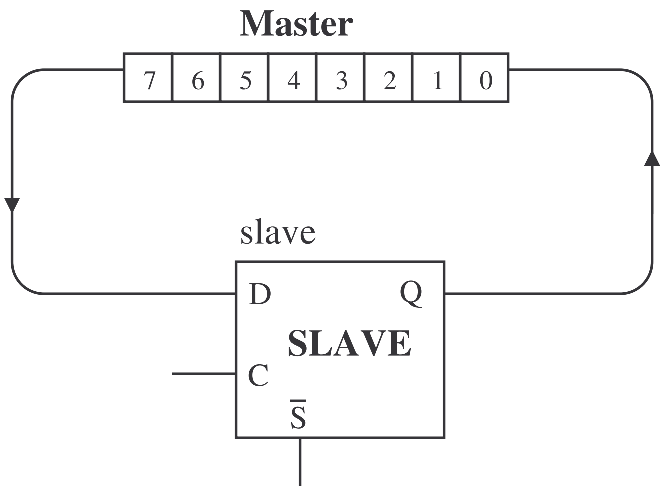


* Other devices on the Starter Kit board must be disabled to prevent interference with SPI FLASH. Some connect to PicoBlaze to enable software to disable or use for future development.

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. In this case the PicoBlaze in eth Spartan-3E is the master and the SPI FLASH memory is the slave.





Bytes describing commands, addresses or data are all transmitted MSB first by the master. As each bit is transmitted, the slave also transmits a bit allowing one byte to be passed in each direction at the same time. In the case of the FLASH memory, this full duplex capability is not used, but it is still necessary to transmit 'dummy' bytes when receiving and ignore received 'dummy' bytes when transmitting.

Each bit is transmitted or received relative to the SCK clock. The system is fully static and any clock rate up to the maximum is possible. The M25P16 captures data (D) on the rising edge of SCK and changes the output data (Q) on the falling edge of SCK.

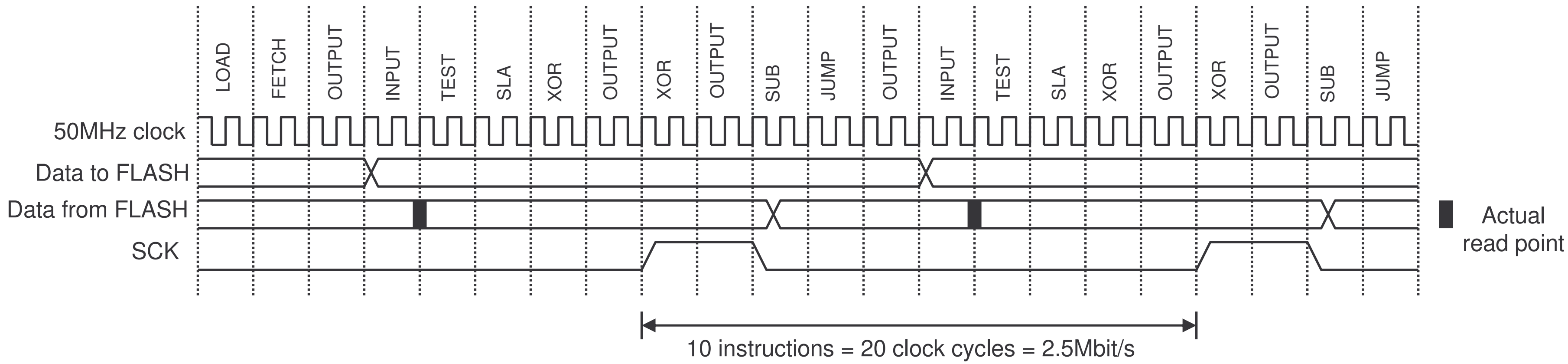
Communication is only possible with the M25P16 device when the select signal (S) is Low. Therefore the PicoBlaze master is responsible for controlling the select signal before transmitting the command byte and then transmitting or receiving any associated bytes. In the cases of writing or erasing the M25P16, it is the release (setting High) of the select line which actually executes the command.

Software SPI Communication

PicoBlaze is used to implement the SPI communication 100% is software. The fundamental byte transfer routine is shown below. The ‘s2’ register is used to supply the data byte (or a dummy byte) for transmission and this will be replaced by the byte data (or a dummy byte) received from the SPI memory.

```
   SPI_FLASH_tx_rx: LOAD s1, 08                     ;8-bits to transmit and receive
                    FETCH s0, SPI_control_status    ;read control status bits
next_SPI_FLASH_bit: OUTPUT s2, SPI_output_port      ;output data bit ready to be used on rising edge
                    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
                    XOR s0, SPI_sck                 ;clock High (bit0)
                    OUTPUT s0, SPI_control_port     ;drive clock High
                    XOR s0, SPI_sck                 ;clock Low (bit0)
                    OUTPUT s0, SPI_control_port     ;drive clock Low
                    SUB s1, 01                      ;count bits
                    JUMP NZ, next_SPI_FLASH_bit     ;repeat until finished
                    RETURN
```

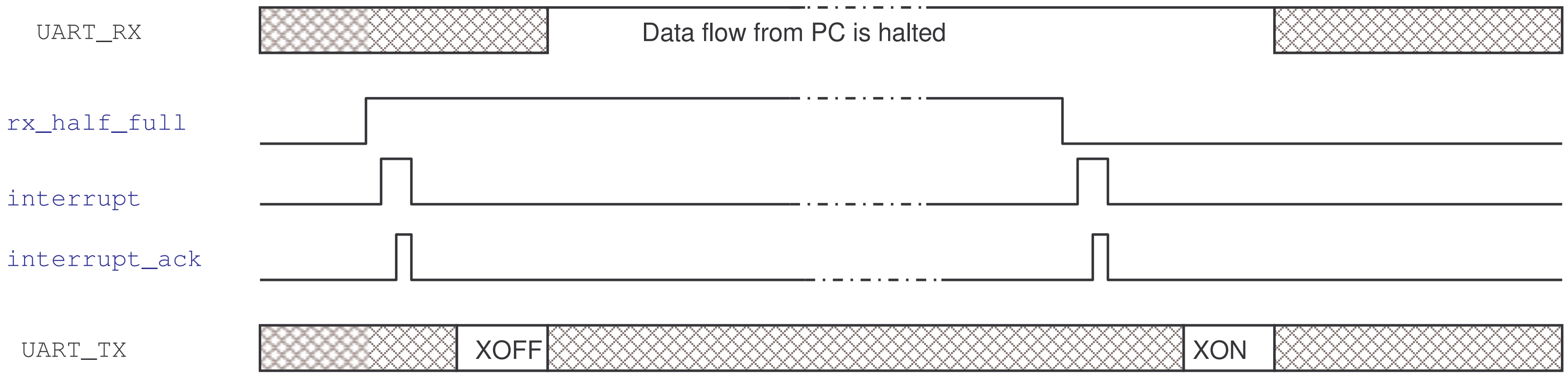
The routine generates SCK. Since every PicoBlaze instruction executes in 2 clock cycle and the design uses the 50MHz clock course on the board, the actual SPI bit rate can be determined. Although this is not as fast as the hardware can support, it is not the weakest link in this system and keeps the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



XON/XOFF Flow Control

When the SPI FLASH device executes a page program (PP) it could take up to 5ms to complete. At the same time the PC will continue to transmit the MCS file at 115200 baud rate. This could mean that 57 characters are transmitted whilst PicoBlaze is waiting for the SPI memory to be free for writing again and the 16 byte FIFO buffer on the UART receiver will overflow. For this reason, the design incorporates a degree of XON/XOFF soft control to enable this design to work at without errors.

The principle requirement of flow control, as explained above, is to limit the flow from the PC to the PicoBlaze design. This is achieved by a combination of hardware and software employing interrupts.



The hardware detects when the ‘half_full’ flag on the receiver buffer changes state and generates an interrupt to the PicoBlaze. When PicoBlaze responds to the interrupt it clears the hardware interrupt automatically with the ‘interrupt_ack’ signal. The interrupt service routine then decides what action to take by reading the status of the ‘half_full’ flag. If the flag is High, then it indicates the buffer has at least 8 characters waiting to be read and so it immediately transmits and XOFF character on the UART transmitter. If the flag is Low, then it indicates the buffer has started to empty and it is able to immediately send an XON character to restore the data flow from the PC.

Note: Although the design includes soft flow control, it is not a comprehensive solution and should only be used as a starting point for other designs. In particular the response to XON/XOFF command characters received from the PC is handled entirely in software and is rather crude at this time.