PicoBlaze™

DS2432 Communicator

Ken Chapman
Xilinx Ltd
6th April 2006

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.

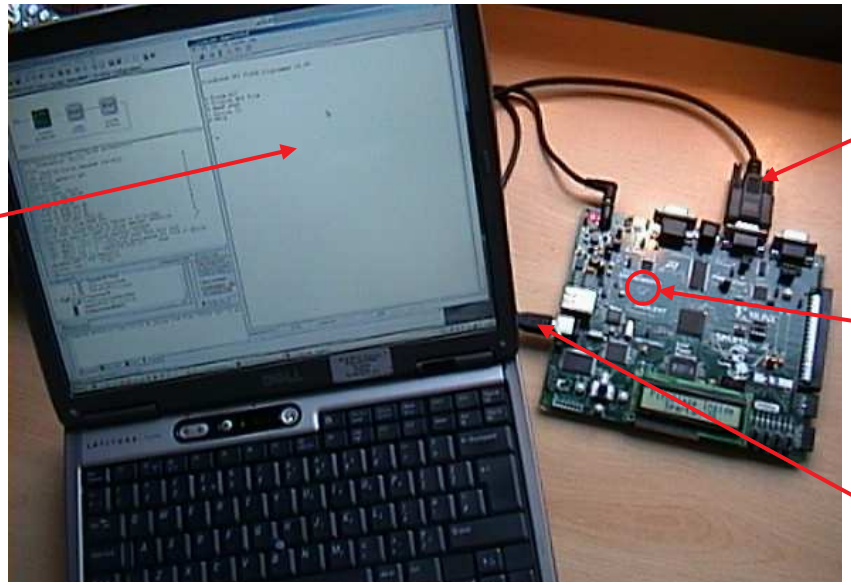
Ken Chapman
Senior Staff Engineer – Spartan Applications Specialist
email: chapman@xilinx.com

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 will allow you to investigate the Dallas Semiconductor DS2432 device which is a 1k-Bit Protected EEPROM with internal SHA-1 Engine. This device has an interesting 1-Wire interface which is used to provide both power and bidirectional communication. The design employs PicoBlaze to implement all the 1-wire communication protocol and provide a simple user interface on your PC via the RS232 serial port (use HyperTerminal or similar). Some of the DS2432 commands are fully supported whilst others can be investigated using simple byte write and byte read options.

This design occupies under 5% of the XC3S500E device. It is hoped that the design may be of interest to anyone interested in using the DS2432 or other 1-wire devices in their own designs. PicoBlaze can easily be reprogrammed in this design using the JTAG_loader supplied with PicoBlaze.



Hint – It is recommended that you obtain a copy of the DS2432 data sheet. Ideally print this document to refer to whist using this design and reading this description. It is particularly useful to have the flow charts available.

Hint – XAPP780 provides a design which can be used to provide copy protection for your own designs by exploiting the special properties of the DS2432.

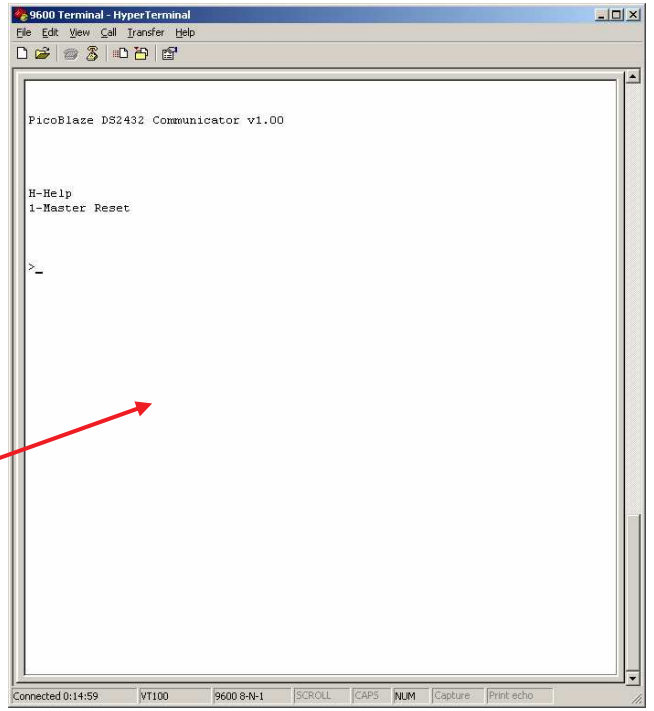
Using the Reference Design

This document is really in two sections. The first covers how to use the design 'AS IS' and in the process provides an introduction to the features and operation of the DS2432 device. The second section covers in some detail the actual design implementation from both the hardware and PicoBlaze perspectives. It is recommended that you use the design first to be come familiar with what it offers to make the second section easier to understand.

Configuring the Spartan-3E 'The Quick Way'!

Unzip all the files provided into a directory.
Connect a suitable serial cable (see previous page).
Start a HyperTerminal (or similar) session using 9600 baud, 1 stop and no parity (see following pages).
Check you have the USB cable connected and the board is turned on.
Double click on the file **'install_PicoBlaze_DS2432_communicator.bat'**.
This should open a DOS window and run iMPACT in batch mode to configure the Spartan device.

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



Alternatively use iMPACT manually to configure the XC3S500E device on the Spartan-3E Starter Kit via the USB cable with the BIT file provided.

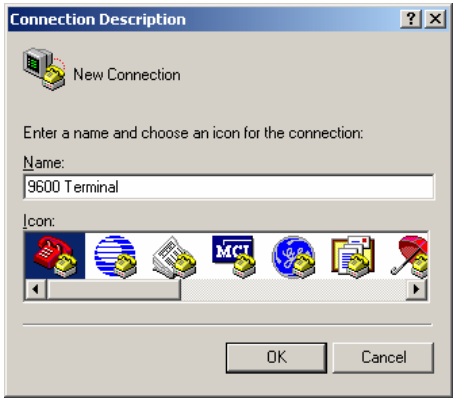
Hint – Stop now and take time to read of the DS2432 data sheet and grasp the fundamentals of the device. It is hoped that this design will help bring the data sheet to life and that you should have many "oh, that's what that means" moments as a result ☺

Serial Terminal Setup

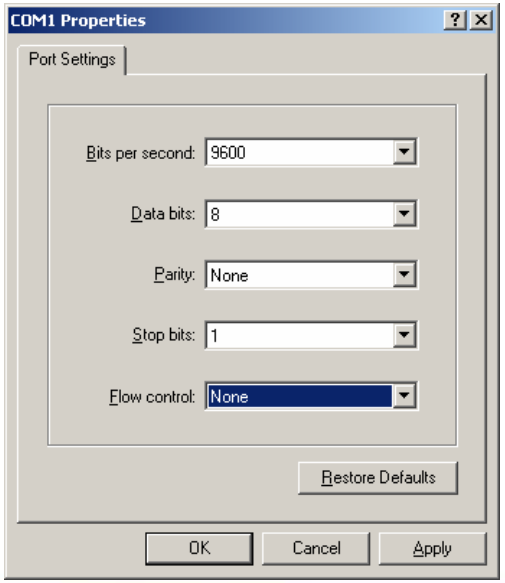
An RS232 serial link is used to communicate with the design. Any simple terminal program can be used, but HyperTerminal is adequate for the task and available on most PCs.

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.

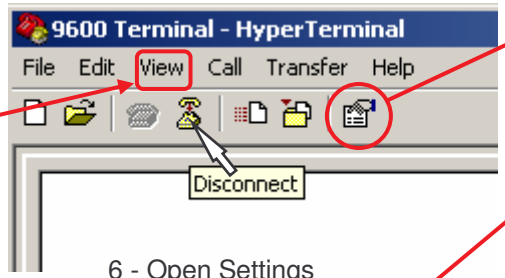
Bits per second : 9600
Data bits: 8
Parity: None
Stop bits: 1
Flow control: **None**

Go to next page to complete set up…

HyperTerminal Setup

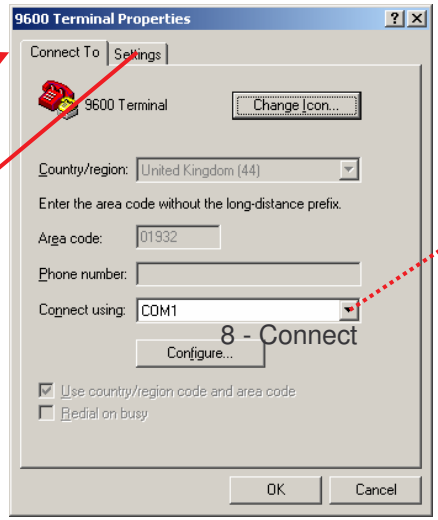
Although steps 1, 2 and 3 will actually create a Hyper terminal session, there are few other protocol settings which need to be set or verified for the PicoBlaze design to work as expected.

4 - Disconnect



Optional step…..
Set Font to
Courier New,
Regular, 10

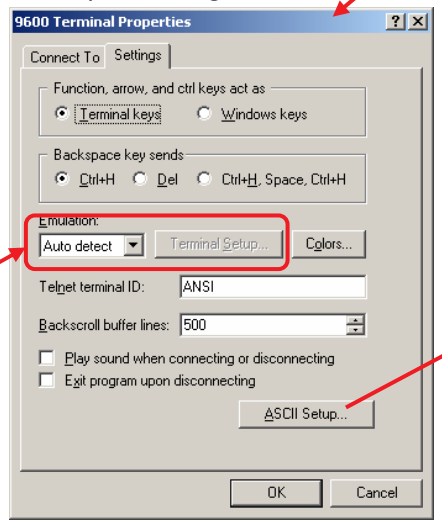
5 - Open the properties dialogue



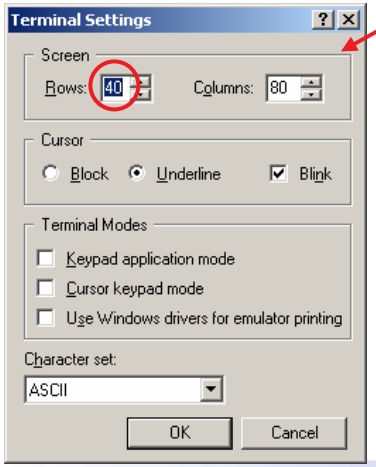
To select a different COM port and change settings (if not correct).

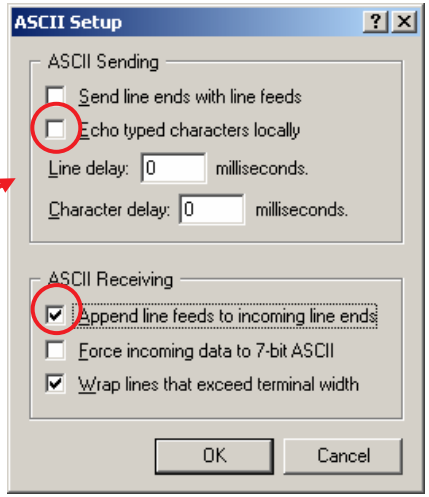
8 - Connect

6 - Open Settings



Optional steps…..
Select VT100 and then click ‘Terminal Setup’
Set ‘Rows’ to 40.
(May require you to stretch main screen later to fit).





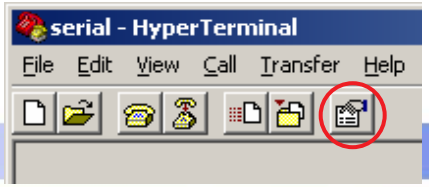
7 - Open ASCII Setup

Ensure boxes are filled in as shown.

The design will echo characters that you type so you do not need the ‘Echo typed characters locally’ option.

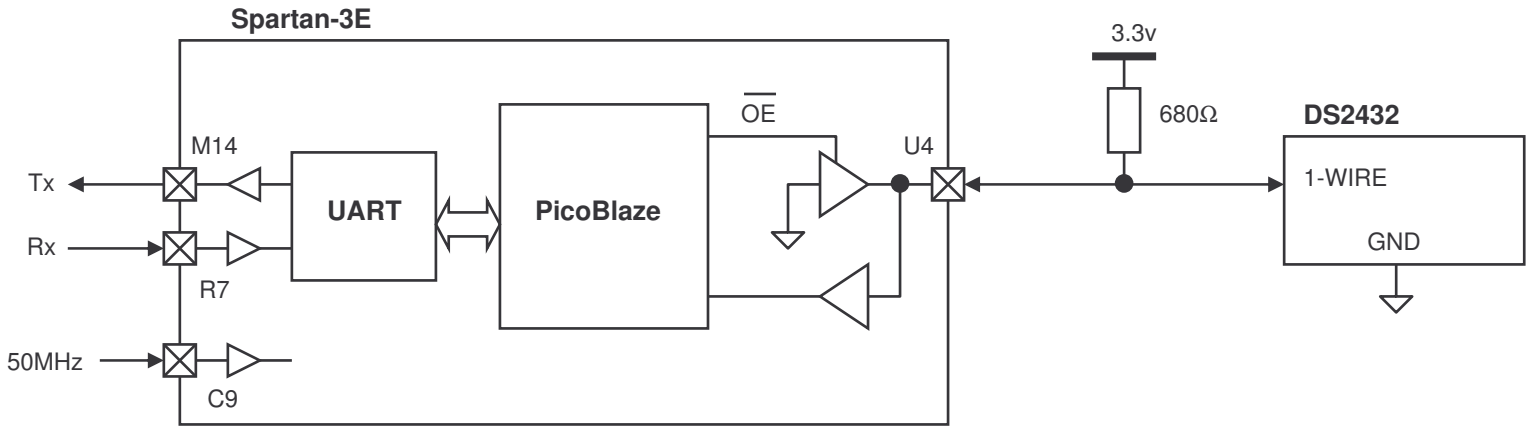
The design transmits carriage return characters (OD_{HEX}) to indicate end of line so you do need the ‘Append line feeds to incoming line ends’ option to be enabled.

8 - ‘OK’ the boxes to get back to main screen and then Connect.



System Overview

The DS2432 has a 1-wire interface. Not only does this single wire provide bidirectional serial communications, it is also the only way in which the device is powered. Therefore this wire is generally held High by an external pull-up resistor of 680Ω.



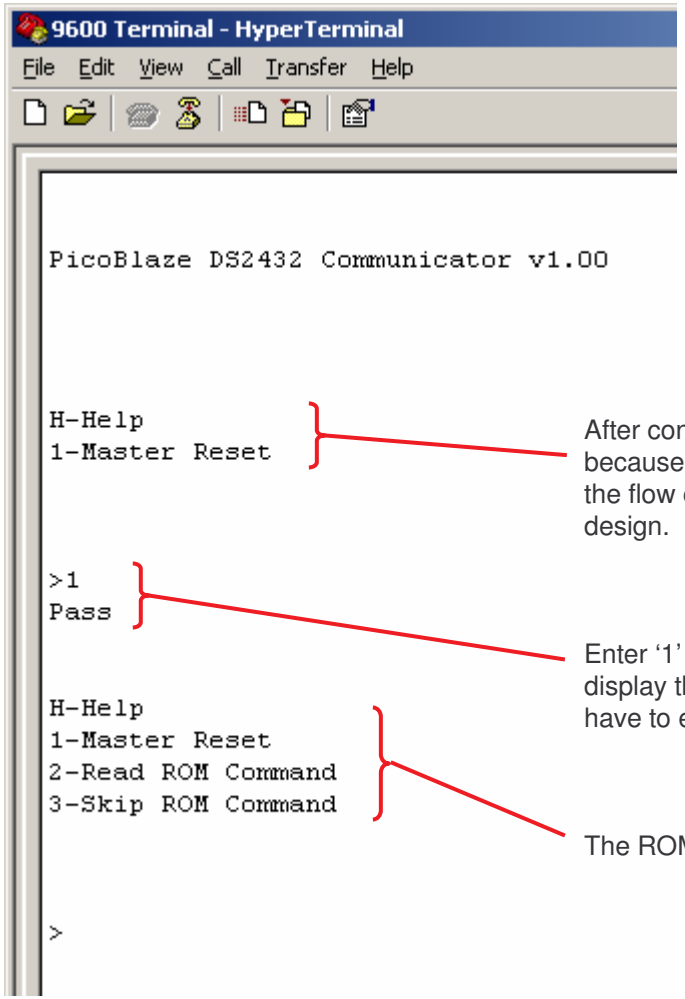
The 1-wire interface is an open collector 'bus' which allows either the master (PicoBlaze inside the Spartan-3E in this case) or the DS2432 to pull the signal Low. If both devices have released the 'bus' then the pull-up provides the 'High'. To maintain power, the High level is maintained for the majority of time and signal is only pulsed Low for a few micro-seconds at a time.

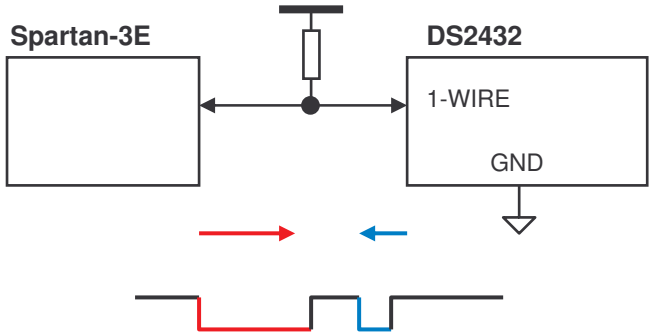
Obviously, all communication with the DS2432 must be performed serially using specific timing. PicoBlaze implements all this in software using the 50MHz clock as the reference for all timing. After initialisation of the DS2432, all serial communication is performed using serial bytes (8-bits) which are transmitted and received least significant bit (LSB) first.

The RS232 serial communications are implemented by some simple UART macros including 16-byte FIFO buffers (supplied with PicoBlaze). These isolate PicoBlaze from the intricacies of the UART signalling and timing although PicoBlaze is responsible for all characters transmitted and interpreting all characters received. In fact, the user interface is a large part of the program.

Master Reset

Before any communication with the DS2432 can take place it must be initialised. This is achieved by an exceptionally long duration Low pulse on DS-wire by PicoBlaze. If the DS2432 is functional, it responds with a short Low pulse of its own (the 'presence pulse'). This master reset sequence must also be repeated after command sequences have been performed so that a new sequence can begin (Hint – this is to support a true bus system with multiple devices).





After configuring Spartan-3E with the reference design you are guided to issue a master reset pulse because this is the only valid option. In other words, the PicoBlaze program attempts to guide you thorough the flow chart show in the DS2432 data sheet and this is why it is useful to have available when using this design.

Hint – 'H' will simply display the available menu options again.

Enter '1' to issue the master reset. If the DS2342 responds with its 'presence pulse' then PicoBlaze will display the 'Pass' message and proceed to the next menu. Otherwise it will report 'Fail' and only you will have to enter another master reset.

The ROM function commands become available after the presence pulse has been received.

ROM Commands

After a presence pulse has been received, the DS2432 state machine progresses to the next stage in which one of seven ROM commands is expected. This design only supports two commands called ‘Read ROM’ and ‘Skip ROM’ either of which must be executed to reach the next and final level of commands.

```
H-Help
1-Master Reset
2-Read ROM Command
3-Skip ROM Command



>2
code=33
serial=000000CAAC92
CR=BC
Pass


H-Help
1-Master Reset
2-Read Memory Command
3-Write Scratchpad Memory Command
4-Read Scratchpad Memory Command
5-Write Byte
6-Read Byte



>
```

Entering ‘2’ executes the read ROM command sequence. First PicoBlaze transmits the command byte 33 Hex and then it reads back 8 bytes from the DS2432 which are displayed….

First byte is the device code which should be 33 hex.

The next 6 bytes are the unique 48-bit serial number (registration number) of your device. These are received LS-Byte first (in this case 92 followed by AC etc.)

The last byte is an 8-bit Cyclic Redundancy Check (CRC) used to confirm the previous 7 bytes. PicoBlaze also calculates the CRC and displays a ‘Pass’ or Fail’ as required.

If a ROM command has been executed successfully the you reach the memory and SHA Functions level of the DS2432 state machine and PicoBlaze provides a new selection of options.

Other ROM Commands

Skip ROM (supported) is the same as Read ROM but you just do not get the device code and unique serial number returned or displayed. This would speed up communication in a real application.

Match ROM, Search ROM and Resume commands are used in situations where there are multiple devices connected to the same 1-wire bus. The Starter Kit has only one device.

Overdrive Skip ROM and Overdrive Match ROM commands have the same general use as Skip ROM and Match ROM commands but have the additional effect of increasing the speed of eth serial communications. Although PicoBlaze is more than capable of supporting the higher speed, he regular speed was adequate for this design and avoids the complication of supporting both rates.

Read Memory Command

```
H-Help
1-Master Reset
2-Read Memory Command
3-Write Scratchpad Memory Command
4-Read Scratchpad Memory Command
5-Write Byte
6-Read Byte


>2
0000    00 00 00 00 00 00 00 00
0008    00 00 00 00 00 00 00 00
0010    00 00 00 00 00 00 00 00
0018    00 00 00 00 00 00 00 00
0020    00 00 00 00 00 00 00 00
0028    00 00 00 00 00 00 00 00
0030    00 00 00 00 00 00 00 00
0038    00 00 00 00 00 00 00 00
0040    00 00 00 00 00 00 00 00
0048    00 00 00 00 00 00 00 00
0050    00 00 00 00 00 00 00 00
0058    00 00 00 00 00 00 00 00
0060    00 00 00 00 00 00 00 00
0068    00 00 00 00 00 00 00 00
0070    00 00 00 00 00 00 00 00
0078    00 00 00 00 00 00 00 00
0080    FF FF FF FF FF FF FF FF
0088    00 00 00 55 00 00 00 00
0090    33 92 AC CA 00 00 00 BC
OK

>

H-Help
1-Master Reset


>_
```

The read memory command allows the complete memory contents to be observed plus some other information.

Entering '2' executes the read memory command sequence. First PicoBlaze transmits the command byte F0 Hex. Then it transmits a 16-bit address LS-Byte first which indicates the start address in memory for the read. In this design, PicoBlaze always transmits 0000 hex so that it starts at the beginning.

PicoBlaze is then able to read bytes back from the DS2432 until it reaches address 0097. These are displayed in a tabular style with the address for the first byte of each line on the left side.

Hint - The read memory command always reads until the end of memory unless a master reset is issued to abort the process.

Addresses 0000 to 007F cover the 128 bytes forming the 1K-bit memory. This can always be read, but requires knowledge of the secret to write.

Addresses 0080 to 0087 are the locations holding the 64-bit secret which for obvious reasons can not be read directly and is masked to 'FF'.

Addresses 0088 to 008F indicate write protection settings and the '55' shown at address 008B is a read only 'Factory Byte'

Addresses 0090 to 0097 provide the same device code, 48-bit unique serial number and 8-bit CRC code as the read ROM command.

After **any** memory of SHA function command the DS2432 state machine expects a master reset followed by a ROM command. Therefore PicoBlaze limits the options to guide you through the required sequence.

Write Scratchpad Command

```
H-Help
1-Master Reset
2-Read Memory Command
3-Write Scratchpad Memory Command
4-Read Scratchpad Memory Command
5-Write Byte
6-Read Byte



>3
address=0080
data0=01
data1=23
data2=45
data3=67
data4=89
data5=ab
data6=cd
data7=ef
CR=F06E
Pass


H-Help
1-Master Reset



>_
```

The EEPROM array of the DS2432 is not written to directly. Instead a RAM based scratch pad memory is provided to which you write both the target address and data. This can then be verified before writing it into the EERPROM array. This is particularly useful when writing a secret since it is impossible to very it by conventional means. The scratch pad is also used in the generation of Message Authentication Codes (MACs).

Entering '3' executes the write scratchpad command sequence. First PicoBlaze transmits the command byte 0F Hex.

Next, PicoBlaze must transmit a 16-bit address LS-Byte first. This will be the target address if the data is finally written to the EEPROM array and is loaded into the TA1 and TA2 registers of the DS2432. which indicates the start address in memory for the read. You are prompted with the 'address=' to enter the 4 digit hexadecimal address. Illegal characters will result in the prompt being repeated.

Hint – Only addresses in the range 0000 to 008F are valid for the DS2432 device. Any address above this range will cause the command sequency to terminate at the DS2432 and in PicoBlaze.
Hint – Internally to the DS2432 the least significant 3 bits of the address are always reset to zero such that the data always falls onto 8-byte boundaries.

The DS2432 then expects PicoBlaze to write 8 bytes of data into the scratch pad. This time PicoBlaze prompts you to enter each byte in turn rejecting any illegal characters by repeating a particular data prompt.

After writing all data, PicoBlaze read back a CRC formed of all 11 bytes of this command sequence. This is a 16-bit CRC (not the 8-bit CRC used in the read ROM command) and the value has been 1's complemented. PicoBlaze also calculates this CRC and reports a 'Pass' or 'Fail' as appropriate.

After **any** memory of SHA function command the DS2432 state machine expects a master reset followed by a ROM command. Therefore PicoBlaze limits the options to guide you through the required sequence.

Read Scratchpad Command

The read scratch pad command is the complement to the write scratchpad command and can be used to verify that the data is good before proceeding with other commands.

```
H-Help
1-Master Reset
2-Read Memory Command
3-Write Scratchpad Memory Command
4-Read Scratchpad Memory Command
5-Write Byte
6-Read Byte


>4
address=0080
E/S=5F
data= 01 23 45 67 89 AB CD EF
CR=E4D6
Pass


H-Help
1-Master Reset


>
```

Entering '4' executes the read scratchpad command sequence. First PicoBlaze transmits the command byte AA Hex.

Then PicoBlaze is able to read 13 bytes as follows…

The first 2 bytes read (LS-byte first) are the contents of the target address registers TA1 and TA2.

Hint – The DS2432 internally resets the least significant 3 bits of the address to zero and therefore the address read may not match that which was set during the write scratchpad command.

The next byte is the ending offset/data status byte (E/S). This should be '5F' hex if all has been successful. A value of 'DF' (MSB is set) would indicate that the scratchpad contents have also been successfully copied into the EEPROM array using another command. Any other values are bad news!

Then the 8-bytes of data can be read and are displayed.

Finally, PicoBlaze can read back a 16-bit CRC formed of all 12 bytes of this command sequence. This complimented 16-bit CRC value is also calculated by PicoBlaze which reports a 'Pass' or 'Fail' as appropriate.

After **any** memory of SHA function command the DS2432 state machine expects a master reset followed by a ROM command. Therefore PicoBlaze limits the options to guide you through the required sequence.

Write Byte and Read Byte

```
>5
Byte=5a
OK
```
Write 5A

```
H-Help
1-Master Reset
2-Read Memory Command
3-Write Scratchpad Memory Command
4-Read Scratchpad Memory Command
5-Write Byte
6-Read Byte
```

```
>5
Byte=80
OK
```
Write 80

```
H-Help
1-Master Reset
2-Read Memory Command
3-Write Scratchpad Memory Command
4-Read Scratchpad Memory Command
5-Write Byte
6-Read Byte
```

Not shown
Write 00
Write 5F

```
>
```

All the DS2432 commands supported by this design allow you to investigate the basic communication with the device but do not allow you to actually modify the EEPROM array contents or use the SHA-1 functions. These will be covered in a separate reference design in which the aspects covered in this design are taken for granted. However, this design does provide two simple menu options which allow you to manually write and read bytes.

These simple options enable you to manually execute any of the commands following the flow charts in the DS2432 data sheet. Option '1' enable you to issue a master reset at the end of any command sequence or to abort if you get confused!

Hint – Try manually entering a write scratchpad command to ensure that you understand that command fully and can follow the flow chart.

To write a byte enter the '5' option and then provide a 2 digit hexadecimal value to the 'Byte=' prompt. Illegal characters will result in the prompt being repeated.

The example shown here is the start of a manual attempt to execute the load first secret command which requires the following sequence…

1) Write the secret to scratch pad with address 0080 (see write scratchpad example).
2) Verify the secret and note address and E/S values (see read scratchpad example).
3) Write the load first secret command 5A hex.
4) Write the values of TA1, TA2 and E/S. These are in order 80, 00 and 5F.
5) Read the DS2432 as many times as you like and confirm response is AA (see below).
6) Master reset to finish sequence.

```
>6
Byte=AA
OK
```
Read AA

Not shown
Master Reset
Read ROM

```
>4
address=0080
E/S=DF
data= 01 23 45 67 89 AB CD EF
CR=22B7
Pass
```
Read scratchpad command confirms write of secret was successful with E/S now 'DF'.

PicoBlaze Design Size

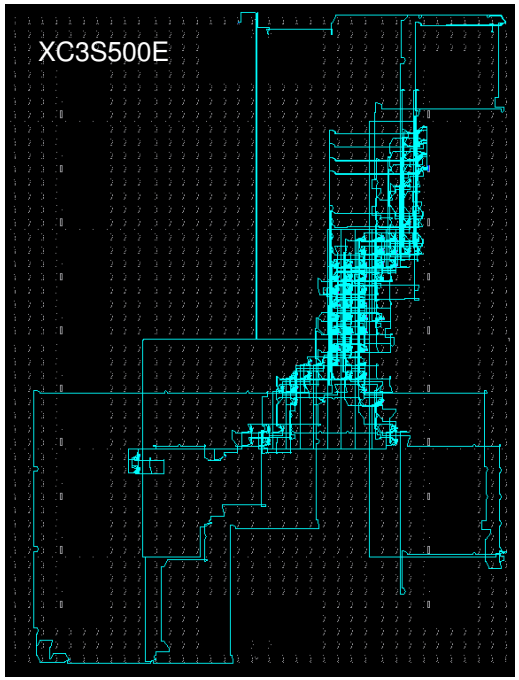
The images and statistics on this page show that the design occupies just 156 slices and 1 BRAM. This is only 3.4% of the slices and 5% of the BRAMs available in an XC3S500E device and would still be less than 17% of the slices in the smallest XC3S100E device.

MAP report

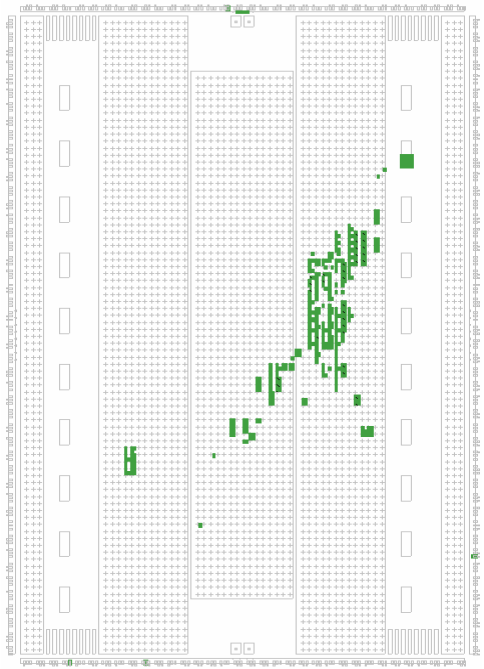
```
Number of occupied Slices:     156 out of    4,656     3%
Number of Block RAMs:            1 out of       20     5%

Total equivalent gate count for design:  78,719
```

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



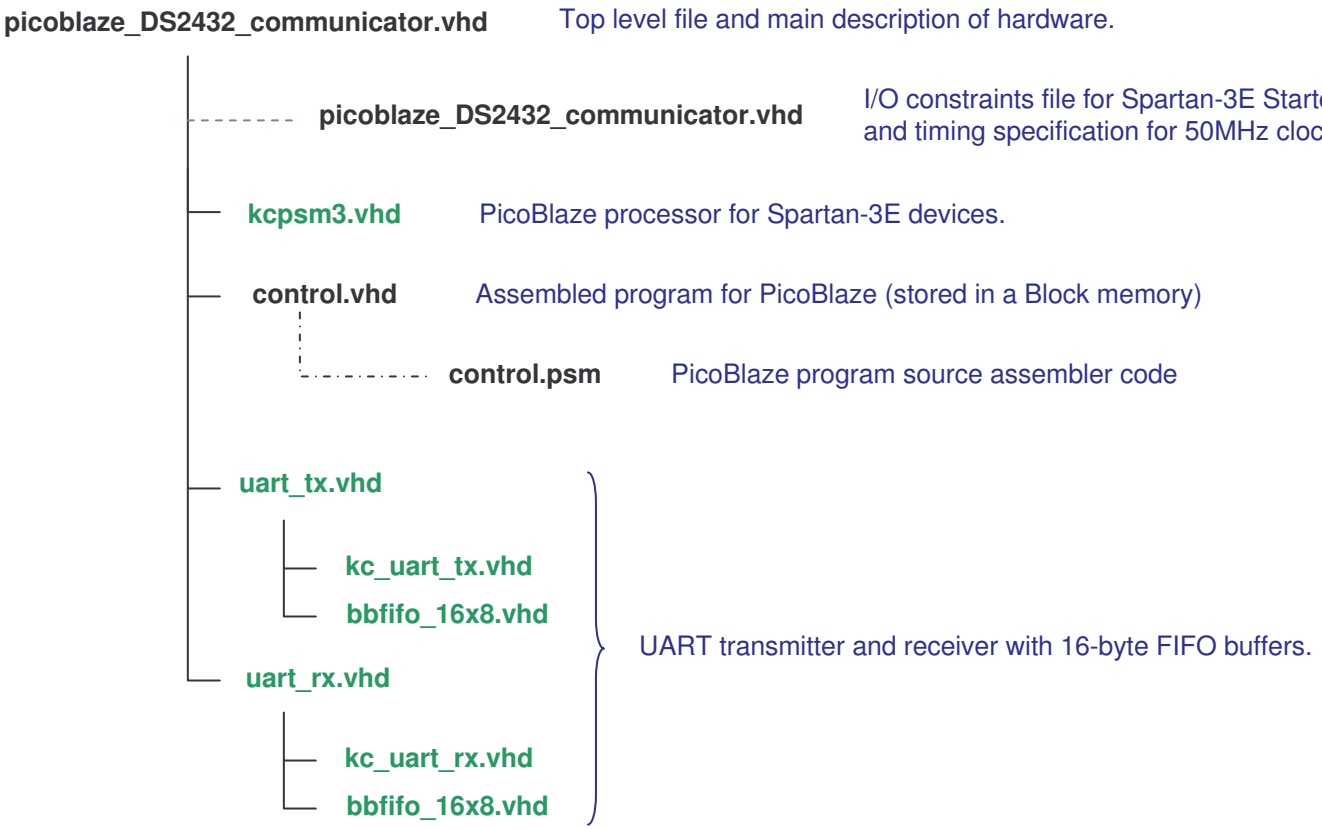
Floorplanner view



Design Files

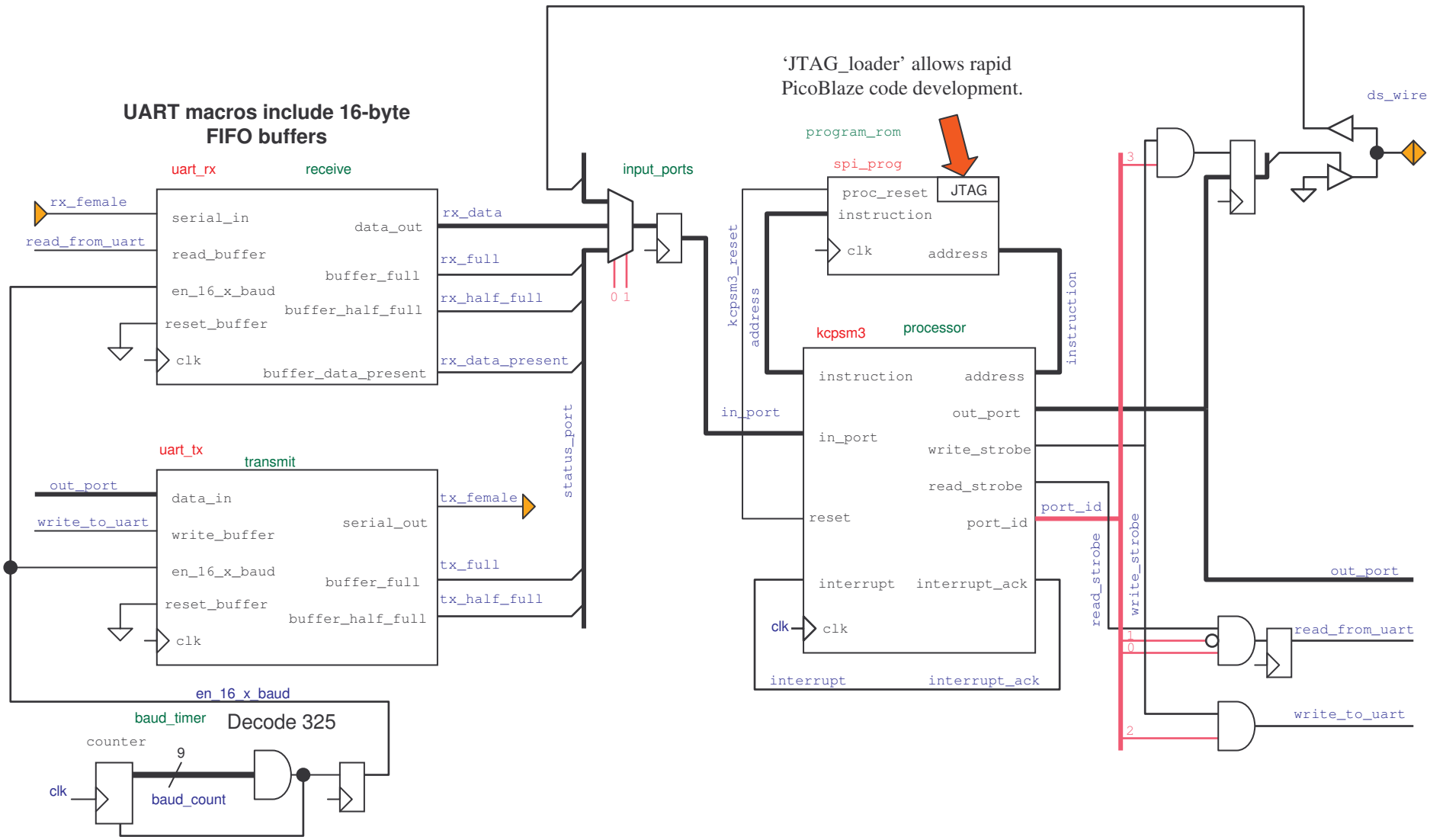
For those interested in the actual design implementation, the following pages provide some details and an introduction to the source files provided. This description may be expanded in future to form a more complete reference design. As well as these notes, the VHDL and PicoBlaze PSM files contain many comments and descriptions describing the functionality.

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



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 Circuit Diagram



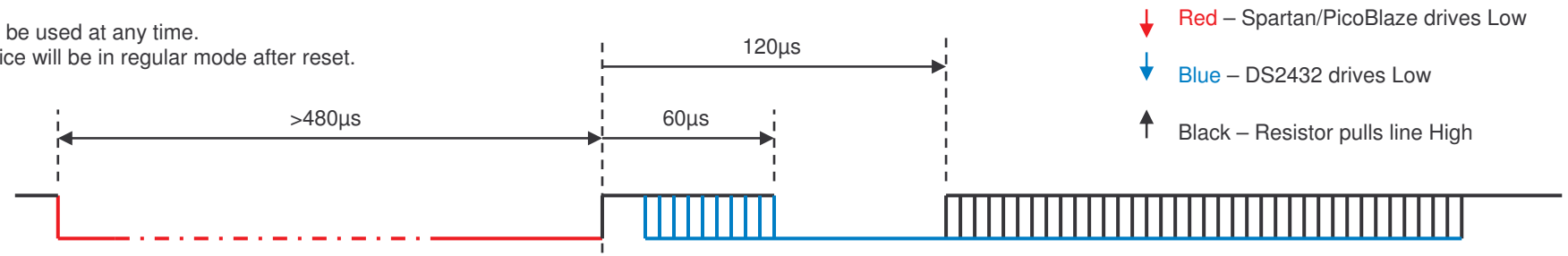
DS2432 Initialisation

The DS2432 is initialised by an active Low master reset pulse. A reset pulse >480µ will force the DS2432 to initialise in 'regular speed' mode. Only if the DS2432 has previously been placed in the faster 'overdrive speed' mode (using an appropriate command) will the faster timing of the overdrive reset pulse be valid and initialise the device whilst remaining in overdrive speed mode.

Following the master reset pulse, the DS2432 acknowledges with an active Low 'presence pulse'. The relative time of which confirms the speed mode. The diagrams below are shown reasonably to scale to show how an active presence pulse between 60 and 120µs is unique to the regular mode.

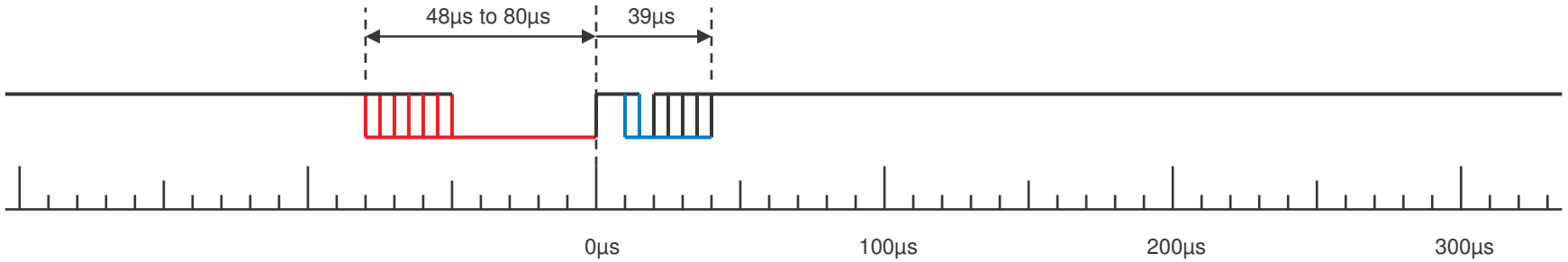
Regular Mode

Can be used at any time.
Device will be in regular mode after reset.



Overdrive Mode

Can only be used if device us already in overdrive mode (set using an Overdrive Skip ROM or Overdrive Match ROM command).
Device will remain in overdrive mode mode after reset.



The PicoBlaze code provided generates a master reset pulse of 500µs to initiate regular mode and a pulse of 64µs to initiate overdrive mode. It then checks for a presence pulse occurring only after 60µs and then waits the full 300µs before returning to ensure the DS2432 has any activity that may happen as completed.

Initialisation Code

Hint – The 'control.psm' file contains comprehensive notes and comments (more than shown in the boxed below).

In this design, PicoBlaze is used to implement the 1-wire communication 100% in software. The fact that a processor is sequential in nature means that the required delays can be formed simply by executing the appropriate number of instructions. PicoBlaze simplifies the task of writing code because all instructions execute in two clock cycles under all conditions. At the clock rate of 50MHz, this mean that all instructions take 40ns to execute.

```
            CONSTANT delay_1us_constant, 0B
delay_1us:  LOAD s0, delay_1us_constant
 wait_1us:  SUB s0, 01
            JUMP NZ, wait_1us
            RETURN
```

The PicoBlaze program supplied implements a 1µs delay in software which it then uses as the base for many of the 1-wire operations. This subroutine is invoked with a 'CALL delay_1us' which then LOADs register s0 with 11 (0B hex). This in turn causes the SUB and JUMP NZ instructions to execute 11 times before RETURN completes the routine. This means that a delay of exactly 1µs is formed by the 25 instructions each taking two clock cycles at 50MHz.

The master reset routine uses this 1µs delay routine many times. To be more precise (although not really required in this application) the delays are further refined by calculating the total number of instructions which will be executed for each delay. This prevents an accumulated error caused by the act of calling the basic 1µs delay many times.

```
DS_init_regular_mode: LOAD s0, 00
                      OUTPUT s0, DS_wire_out_port
                      LOAD s2, 01
                      LOAD s1, BD
      rm_wait_500us:  CALL delay_1us
                      SUB s1, 01
                      SUBCY s2, 00
                      JUMP NC, rm_wait_500us
                      LOAD s0, 01
                      OUTPUT s0, DS_wire_out_port
                      LOAD s1, 38
       rm_wait_60us:  CALL delay_1us
                      SUB s1, 01
                      JUMP NZ, rm_wait_60us
                      LOAD s2, 01
                      LOAD s1, B6
      rm_poll_240us:  CALL delay_1us
                      CALL read_DS_wire
                      AND s2, s0
                      SUB s1, 01
                      JUMP NZ, rm_poll_240us
                      TEST s2, 01
                      RETURN
```

Drive DS_wire Low

A delay of 500us is equivalent to 12500 instructions at 50MHz. Because the loop calling the 1µs delay routine requires 3 instructions, there are actually 28 instructions being executed per iteration. This means that only 446 (01BD hex) iterations are required and not the obvious 500 which would produce a delay of 560µs.

Release DS_wire to be pulled High by external resistor

Wait 60us to miss any overdrive mode response. This time there are 27 instructions per iteration requiring 56 repetitions (56×27×40ns=60.48µs)

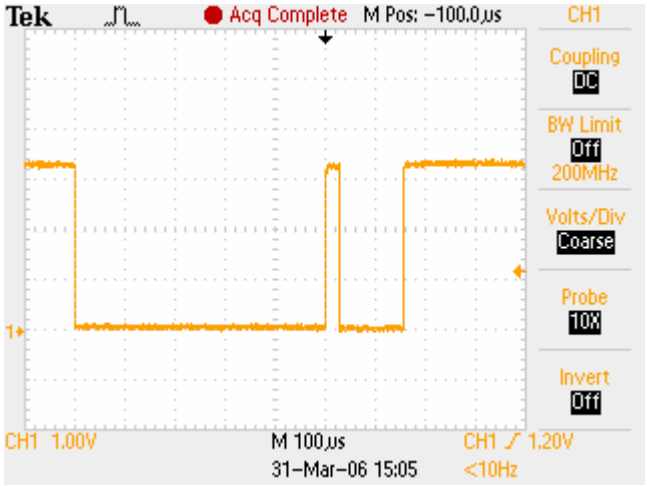
The final delay loop is of 240us. This loop is formed by 33 instructions requiring 182 repetitions. There are more instructions in this loop because PicoBlaze polls the DS_wire (read_DS_wire subroutine) at approximately 1us intervals looking to detect an active Low presence pulse. If a Low is detected a flag in register 's2' is cleared.

Set CARRY flag if no presence pulse detected.

Initialisation and Speeds

Regular Mode

100µs/division



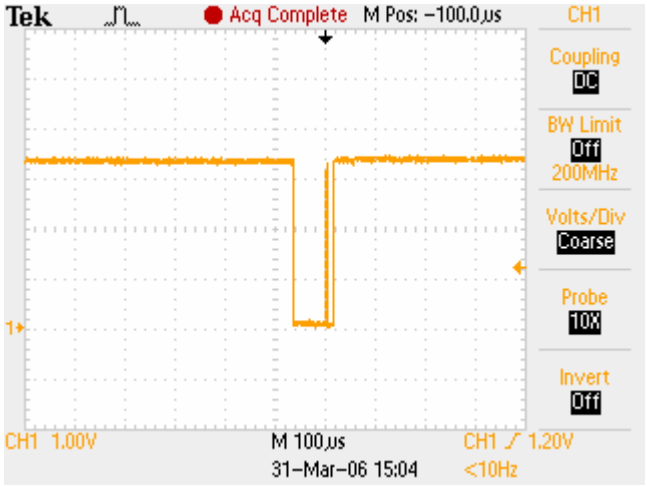
These oscilloscope screen shots show the reset pulses generated by PicoBlaze and the presence pulses generated by the DS2432.

On a scale of 100µs/division, the it can be seen that the master reset pulse generated by PicoBlaze is 500µs. This is then followed by a presence pulse with a duration of just over 100µs.

Although not directly supported by the design provided, experiments were conducted using overdrive mode and the screen shots below clearly show how a master reset pulse of 64µs appears to be almost immediately followed by a much shorter presence pulse. Sampling the presence pulse only after 60µs will clearly prevent detection of the presence pulse for overdrive mode.

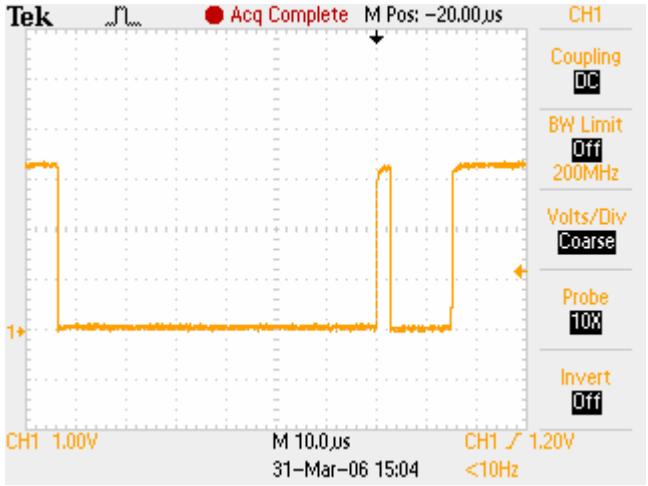
Overdrive Mode

100µs/division



Zoom in

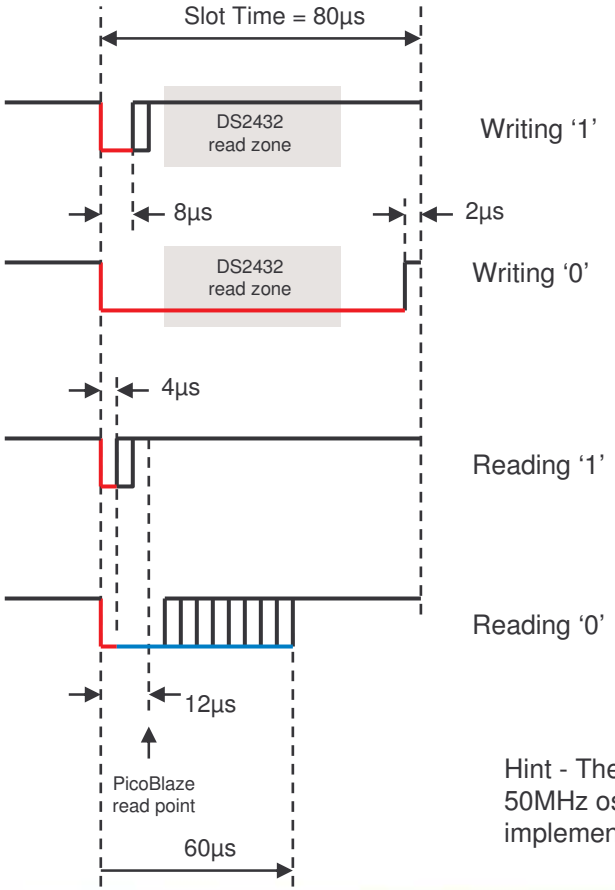
10µs/division.



1-Wire Writing and Reading

Data is written to the DS2432 using the 1-wire interface and requires some careful timing. Although PicoBlaze is more than capable of implementing the higher speed ‘overdrive mode’, the code provided in this reference design implements only the slower ‘regular speed’ mode. This simplifies the operation when first understanding this device because the ‘overdrive’ mode is only achieved by first writing an Overdrive ROM command at regular speed!

The following diagrams show the approximate timing of the routines provided with this reference design. These timing meet the limits specified in the DS2432 data sheet for **regular speed mode** which should be consulted for more detail.



All the write and read routines have been implemented to fit an 80µs slot time which covers the 60µ minimum and allows more than adequate recovery time between successive reads/writes.

A write is initiated by the PicoBlaze driving the wire Low for 1 to 15µs (8µs implemented). PicoBlaze then continues to drive the wire Low to write a ‘0’ or releases the wire to write a ‘1’ which is sampled by the DS242 at some point between 15 and 60us. The wire must be released and High for at least 1µs to complete the write operation (2µs implemented)..

Red – Spartan/PicoBlaze drives Low

Blue – DS2432 drives Low

Black – Resistor pulls line High

A read is initiated by the PicoBlaze driving the wire Low for at least 1µs (4µs implemented). PicoBlaze then releases the wire which allows the DS242 to either continue to hold the wire Low or release the wire for a High. Since the DS2432 may only hold the wire low for up to 15us, PicoBlaze must read the line before this time expires (12µs implemented). After reading the bit value, enough time must be allowed for a slow DS2432 to release the wire and recover for which the 80µs time slot is more than adequate.

Hint - The PicoBlaze code provided implements all timing using software delay loops based on the 50MHz oscillator provided on the Starter kit board. If a different clock rate is used, the delays implemented in the routines ‘write_Low_slow’, ’write_High_slow’ and ‘read_bit_slow’ should be revised.

1-Wire Code to Write

Separate routines provide the ability to transmit a Low or a High with the correct timing. These again use the fundamental 1µs delay routine.

Writing '0'

```
 write_Low_slow: LOAD s0, 00
                 OUTPUT s0, DS_wire_out_port
                 LOAD s1, 48
  wls_wait_78us: CALL delay_1us
                 SUB s1, 01
                 JUMP NZ, wls_wait_78us
                 LOAD s0, 01
                 OUTPUT s0, DS_wire_out_port
                 CALL delay_1us
                 CALL delay_1us
                 RETURN
```

- Drive DS_wire Low to initiate write and following on to Low data value
- Delay 78us (72 iterations × 27 instructions × 40ns = 77.68µs)
- Release DS_wire to be pulled High by external resistor
- Delay 2us

Writing '1'

```
write_High_slow: LOAD s0, 00
                 OUTPUT s0, DS_wire_out_port
                 LOAD s1, 08
   whs_wait_8us: CALL delay_1us
                 SUB s1, 01
                 JUMP NZ, whs_wait_8us
                 LOAD s0, 01
                 OUTPUT s0, DS_wire_out_port
                 LOAD s1, 43
  whs_wait_72us: CALL delay_1us
                 SUB s1, 01
                 JUMP NZ, whs_wait_72us
                 RETURN
```

- Drive DS_wire Low to initiate write
- Delay 8us (8 iterations × 27 instructions × 40ns = 8.64µs)
- Release DS_wire to be pulled High by external resistor to represent High data value
- Delay 72us (67 iterations × 27 instructions × 40ns = 72.36µs)

1-Wire Code to Read

Separate routines provide the ability to transmit a Low or a High with the correct timing. These again use the fundamental 1µs delay routine.

Reading '0' or '1'

```
read_bit_slow: LOAD s0, 00
               OUTPUT s0, DS_wire_out_port
               LOAD s1, 04
 rbs_wait_4us: CALL delay_1us
               SUB s1, 01
               JUMP NZ, rbs_wait_4us
               LOAD s0, 01
               OUTPUT s0, DS_wire_out_port
               LOAD s1, 08
 rbs_wait_8us: CALL delay_1us
               SUB s1, 01
               JUMP NZ, rbs_wait_8us
               CALL read_DS_wire
               SRA s3
               LOAD s1, 3F
rbs_wait_68us: CALL delay_1us
               SUB s1, 01
               JUMP NZ, rbs_wait_68us
               RETURN
```

Drive DS_wire Low to initiate write

Delay 4us (4 iterations × 27 instructions × 40ns = 4.32µs)

Release DS_wire to be pulled High by external resistor or held Low by DS2432.

Delay 8us (8 iterations × 27 instructions × 40ns = 8.64µs)

DS_wire actually sampled 13µs after start of initiation pulse which is before the 15µs required. Read bit is shifted into register 's3' (used when reading bytes).

Delay 68us (63 iterations × 27 instructions × 40ns=68.04µs)

```
 read_DS_wire: INPUT s0, DS_wire_in_port
               AND s0, DS_wire
               TEST s0, DS_wire
               RETURN
```

Subroutine reads the DS_wire and tests it. This means that the value of DS_wire is available in the LSB of register 's0' and also reflected in the status of the CARRY flag.

Reading and Writing Bytes

All data values and commands are communicated as bytes transmitted and received **least significant bit first**. To achieve this simply requires 8 repetitions of the single bit read and write operations.

It is interesting to observe that these byte writing and reading routines are now free of any timing and physical input and output detail. So it is from this level and above that the code becomes one of pure state machine, protocol and application.

Writing a byte

```
write_byte_slow: LOAD s2, 08
       wbs_loop: RR s3
                 JUMP C, wbs1
                 CALL write_Low_slow
                 JUMP next_slow_bit
           wbs1: CALL write_High_slow
  next_slow_bit: SUB s2, 01
                 JUMP NZ, wbs_loop
                 RETURN
```

The byte to be transmitted must be provided in register ‘s3’.

‘s2’ acts as a bit counter.

The input byte ‘s3’ is rotated right 8 times. Each rotation allows the next least significant bit to be observed via the CARRY flag and for the appropriate write High or write Low routine to be executed. Since ‘s3’ is rotated 8 times the value is returned unchanged.

Reading a byte

```
 read_byte_slow: LOAD s2, 08
       rbs_loop: CALL read_bit_slow
                 SUB s2, 01
                 JUMP NZ, rbs_loop
                 RETURN
```

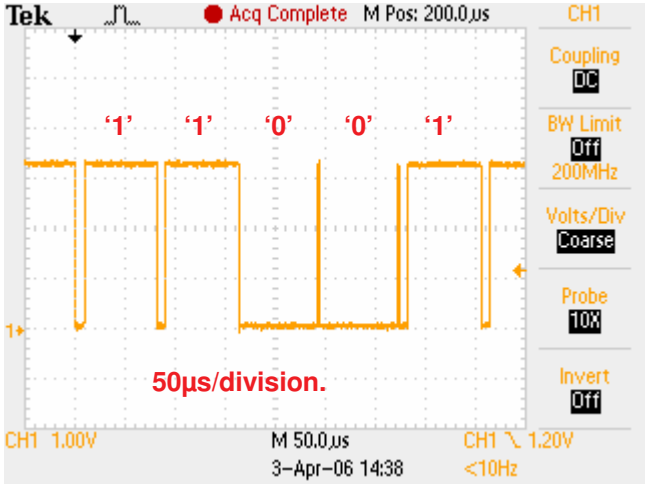
The byte read is returned in register ‘s3’.

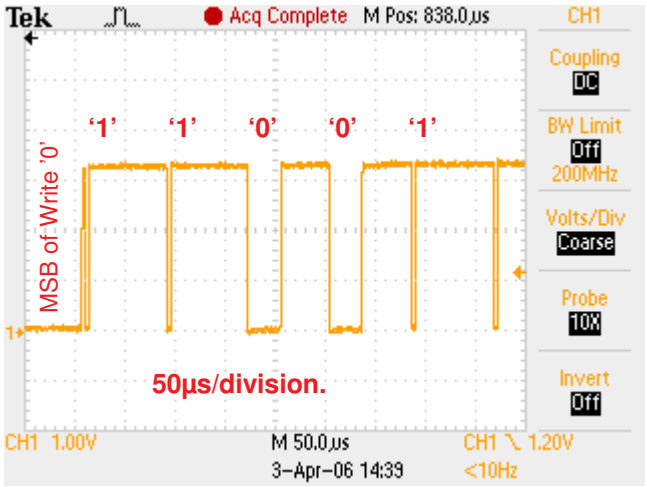
‘s2’ acts as a bit counter.

The read_bit_slow routine shifts each received bit into the MSB of ‘s3’. Therefore after reading all 8 bits the fist received has been shifted down to the LSB position.

1-Wire Signals

These oscilloscope screen shots were captured at pin 2 of the DS2432 device when performing the ‘read ROM’ command. Initially the command byte 33 hex is written which allows the write operations to be seen. PicoBlaze then reads 8 bytes of which the first is the family code (which is also 33 hex). This allows the read process to be observed and shows how the DS2432 drives the wire Low for approximately 30µs on my board.





Writing

The writing of read ROM command.
33 hex = 00110011 binary.

Note the 8µs Low preceding the write of a ‘1’ and the 2µs High completing the writing of a ‘0’.

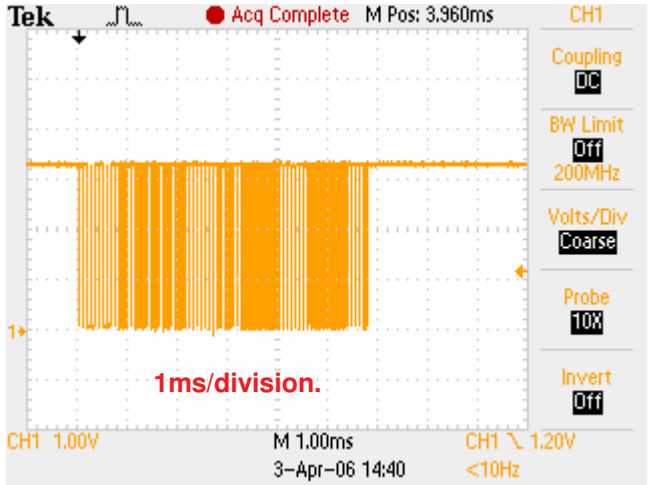
Reading

The reading of the family code.
33 hex = 00110011 binary.

Note the 4µs Low preceding the read of a ‘1’ and how the DS2432 is driving Low for ~30µs in this case.

Scale of 50µs/division.

The complete read ROM command and response (below).
1+8 bytes = 72 bits
72 bits @ 80µs/bit = 5.76ms



Command Code

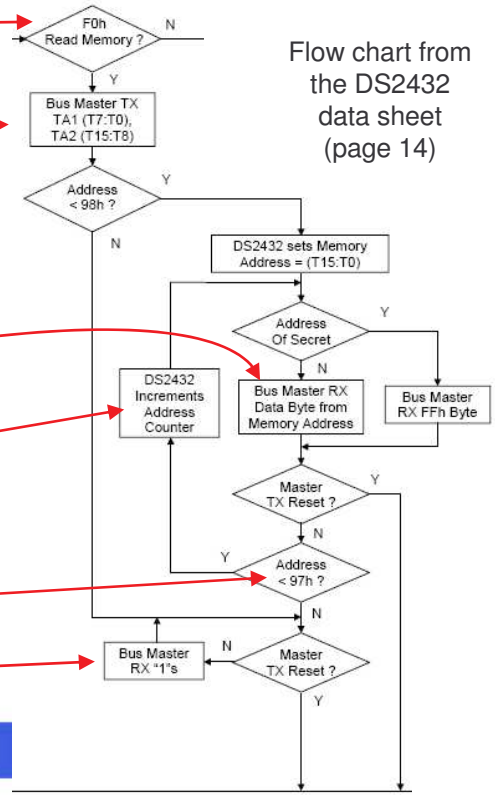
All higher level code is used to directly to trace the path of the flow charts in the DS2432 data sheet. Obviously some interaction is required with the UART to obtain and display results but otherwise the correlation should be easy to follow. This example shows the read memory command.

```
                    ;*************************************************************************
                    ; DS2432 Read Memory Command.
                    ;*************************************************************************
                    ;
                    ; The read memory command (F0 hex) allows the entire memory contents to be read
                    ; except for the secret. This routine displays the address followed by 8 bytes
                    ; of data on each line until the address 0097 is reached.
                    ;
                    ; The initial 'F0' command must be followed by the 16-bit start address transmitted
                    ; LS-byte first. Then reads must continue until address 0097 has been read for the
                    ; command to complete naturally (otherwise a master reset is required).
                    ;
read_memory_command: LOAD s3, F0             ;read memory Command
                     CALL write_byte_slow    ;transmit command
                     LOAD s5, 00             ;initial address in [s5,s4]=0000
                     LOAD s4, 00
                     LOAD s3, s4             ;transmit address
                     CALL write_byte_slow
                     LOAD s3, s5
                     CALL write_byte_slow
          rmc_line_loop: CALL send_CR
                     LOAD s0, s5             ;display 16-bit address
                     CALL send_hex_byte
                     LOAD s0, s4
                     CALL send_hex_byte
                     CALL send_space
                     CALL send_space
          rmc_data_loop: CALL send_space
                     CALL read_byte_slow     ;read data into s3
                     LOAD s0, s3             ;display byte
                     CALL send_hex_byte
                     ADD s4, 01              ;increment address
                     ADDCY s5, 00
                     TEST s4, 07             ;test for 8-byte boundary
                     JUMP NZ, rmc_data_loop
                     COMPARE s4, 98          ;test for last address
                     JUMP NZ, rmc_line_loop
                     CALL send_OK
                     JUMP reset_menu         ;needs master reset next
```

UART display stuff!

Set address to 0000 hex

Hint
This code is shown with all the comments and is typical of all the code provided



Flow chart from the DS2432 data sheet (page 14)

8-bit CRC Code

The 8-bit and 16-bit CRC functions are also implemented in the supplied code and illustrate some true computation can be performed by PicoBlaze.

```
compute_CRC8: FETCH s3, family_code
              FETCH s4, serial_number0
              FETCH s5, serial_number1
              FETCH s6, serial_number2
              FETCH s7, serial_number3
              FETCH s8, serial_number4
              FETCH s9, serial_number5
              LOAD s2, 38
              LOAD s0, 00
   crc8_loop: LOAD s1, s0
              XOR s1, s3
              TEST s1, 01
              JUMP NC, crc8_shift
              XOR s0, 18
  crc8_shift: SR0 s1
              SRA s0
              SR0 s9
              SRA s8
              SRA s7
              SRA s6
              SRA s5
              SRA s4
              SRA s3
              SUB s2, 01
              JUMP NZ, crc8_loop
              RETURN
```

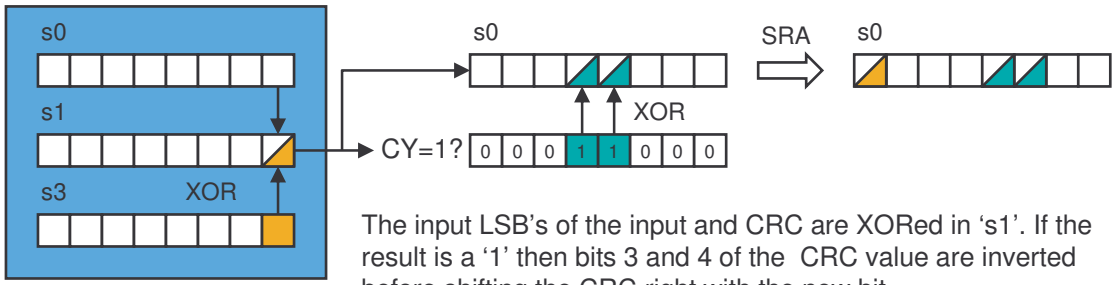
Shift input data

56 times?

The 8-bit CRC routine (shown left) really handles everything directly in local registers having copied all relevant data from scratch pad memory. The 16-bit CRC routine shows how less registers can be used by working with scratch pad memory more interactively.

There are 56-bits(38 hex) to be processed LSB first. This is achieved by loading 7 registers with the data and shifting the whole chain right by one position each iteration.

's0' is used to represent the 8-bit CRC register exactly the same as in the diagram



The input LSB's of the input and CRC are XORed in 's1'. If the result is a '1' then bits 3 and 4 of the CRC value are inverted before shifting the CRC right with the new bit.

Diagram from the DS2432 data sheet (page 4)