	Top Level (Host PC)	Low Level (Host PC)			Low Level (Target FPGA)		Low Level (Host PC)	
I2C	**Write I2C** Slave Address Sub Address 1 Data Byte	**Build Bitstream** 48 devices x 8 bits Socket Mask Applied	**Transfer Instructions to FPGA** Slave Address Sub Address Wait Ticks (Sets Rate) # of Bytes to Write (1) Socket Mask	**Transfer Write Data to FPGA** Load 64-bit Write FIFO 48 bits used (Devices) 8 Elements (Data)	**Execute Write Transaction** Sequentially send all 8 elements to all 48 devices simultaneously, 1 FIFO eloement (Data Bit) at a time.		**Transfer Status to Host** Transfer Data from Read FIFO to Host 1 FIFO Element per Data Bit	
	Read I2C Slave Address Sub Address # Data Bytes		**Transfer Instructions to FPGA** Slave Address Sub Address Wait Ticks (Sets Rate) # of Bytes to Read (n) Socket Mask		**Execute Read Transaction** Sequentially read all data from all 48 devices. Load 64-bit Read FIFO 48 bits used (Devices) 1 data bit per element	**Transfer Read Data to Host** Transfer Data from Read FIFO to Host 1 FIFO Element per Data Bit	**Transfer Status to Host** Transfer Error Information to Host via Status Array	**Parse Bitstream** Re-Build Data Bytes from Bistream 48 devices x 1 element per data bit
SPI	**Write/Read SPI** Write Data Cycle Mode # Cycle CMDs CMD Lengths CMD Delays	**Build Bitstream** Cycle Mode = False 16 DUTs x n bits Socket Mask Applied	**Transfer Instructions to FPGA** Wait Ticks (Sets Rate) # of Bytes to Write (n) Socket Mask	**Transfer Write Data to FPGA** Load 16-bit Write FIFO 16 bits used (DUTs) nx8 Elements (Data)	**Execute Write/Read Transaction** Sends Data from Write FIFO / Reads data from Read FIFO	**Transfer Read Data to Host** Transfer Data from Read FIFO to Host 1 FIFO Element per Data Bit	**Transfer Status to Host** Transfer Error Information to Host via Status Array	**Parse Bitstream** Re-Build Data Bytes from Bistream 16 DUTs x 1 element per data bit
RFFE	**Write RFFE** [1-8 Bytes]	**Build Bitstream** Cycle Mode = False 16 DUTs x n bits Socket Mask Applied Full Write Data Stream Standard or Extended Format	**Transfer Instructions to FPGA** Wait Ticks (Sets Rate) # of Bytes to Write (n) Socket Mask	**Transfer Write Data to FPGA** Load 16-bit Write FIFO 16 bits used (DUTs) nx8 Elements (Data)	**Execute Write** Sends Data from Write FIFO / Reads data from Read FIFO		**Transfer Status to Host** Transfer Error Information to Host via Status Array	
	Read RFFE [2-8 Bytes]	**Build Bitstream** Cycle Mode = False 16 DUTs x n bits Socket Mask Applied Full Read Data Stream Standard or Extended Format	**Transfer Instructions to FPGA** Wait Ticks (Sets Rate) # of Bytes to Write (n) Socket Mask	**Transfer Write Data to FPGA** Load 16-bit Write FIFO 16 bits used (DUTs) nx8 Elements (Data)	**Execute Read Transaction** Sends Data from Write FIFO / Reads data from Read FIFO	**Transfer Read Data to Host** Transfer Data from Read FIFO to Host 1 FIFO Element per Data Bit	**Transfer Status to Host** Transfer Error Information to Host via Status Array	**Parse Bitstream** Re-Build Data Bytes from Bistream 16 DUTs x 1 element per data bit
	Cycle RFFE [1-8 Bytes]	**Build Bitstream** Cycle Mode = True 16 DUTs x n bits Socket Mask Applied CMD Set (up to 4) Delay Set (ms) Cycle Delay (ms)	**Transfer Instructions to FPGA** Wait Ticks (Sets Rate) CMD Set Delay Set Socket Mask	**Transfer Write Data to FPGA** Load 16-bit Write FIFO 16 bits used (DUTs) nx8 Elements (Data)	**Execute Write** Sends Data from Write FIFO / Reads data from Read FIFO	**Wait Cycle Delay** After Full CMD Set Wait Delay between Cycles	**Transfer Info to Host (On Request)** Cycle Count	

ATEC Matrix Corporation	
Document Title: 1050 LabVIEW Software Design	**Date:** 07/5/2014
Document Number: WS-10024	**Rev:**
Page Title: Top Level Architecture	**Page:** 1 of 8

Initialization:

VIs:
WSVCAP Shell\WSVCAP Shell Host Top Level\WSVCAP SHL Host TL Initialize.vi

Summary:
Upon startup of the Host Application, this VI must be run to initialize the Host and FPGA portions of the application. If Simulation Mode is true (in System Configuration.ini file), the FPGA code will run on the Host PC, and a debug window will launch when this VI is run. This debug window will show graphically the bit stream sent and received to and from the FPGA. If Simulation Mode is False, this VI will launch the FPGA portion on the FPGA Target Resource (in System Configuraiton.ini file).

Details:
A. Read System Configuration.ini file
   1. Default Setup
      a. I2C FPGA Clock Rate
      b. SPI Clock Rate
      c. RFFE Clock Rate
      d. FPGA Target Resource
      e. Simulation Mode
B. Initialize FPGA Core Manager
   1. Load FPGA code onto Target FPGA Resource
   2. Initialize FPGA Top Level Controls Core (Use this to transfer data between Host and FPGA)
C. Load FPGA Wait Tick Values and Actual Clock Rates

Configuration:

VIs:
WSVCAP Shell\WSVCAP Shell Host Top Level\WSVCAP SHL Host TL I2C Configure.vi
WSVCAP Shell\WSVCAP Shell Host Top Level\WSVCAP SHL Host TL RFFE Configure.vi
WSVCAP Shell\WSVCAP Shell Host Top Level\WSVCAP SHL Host TL SPI Configure.vi

Summary:
Any time the Clock Rates, or Slave Addresses of devices are changed, the appropriate Configuration VI must be run. This will set the appropriate Address and Clock Rate in the Host Low Level Core. These values are used with every Read and Write operation.

Details:
A. Compute Wait Ticks
B. Update Actual Clock Cycle
C. Update Slave Address (I2C, RFFE Only)

DUT Mask / Socket Mask:

The DUT Mask and Socket mask are two separate items as a legacy from the previous version of this application. In that version there was not a 1-1 mapping from the DUT Numbering to the Socket numbering. In this version, the DUT Number is the same as the Socket Number.

Write / Read Routines (Non Cycle Mode):

To Write and Read data from the DUT (RFFE) or AD7747 devices (I2C), use the Vis descrbed on the following pages for “Host PC”, according to the interface (I2C, RFFE, SPI) of interest. While this version only supports I2C and RFFE, the SPI code from the previous version is still included available in both the FPGA and Host code. For each interface, there are 2 or 3 write or read options, depending on the scope of the write or read operation. You may access a single device, multiple devices with identical data to each device, or multiple devices with unique data between devices. The DUT mask is used at the low level to disable write/read operations to unused devices.

Cycle Mode:

The SPI and RFFE commands may be sent in a Cycle Mode, where a set of commands may be repeatedly cycled through completely managed by the FPGA, which allows the maximum transfer rates over an extended time to the device. Cycle mode is a Write-Only operation. The Cycle Mode will run until the Stop Cycle routine is executed. In this mode the only interaction between the Host PC and FPGA Target is the Cycle Count Query operation.

FPGA Design

The FPGA uses a nested state machine architecture to manage all operations. The outer while loop (Outer state machine) used to manage instructions coming from the Host. When a transaction is requested, the instructions are passed via the FPGA front panel items from the Host PC to the FPGA. Only one transaction may be processed at a time. The Outer State Machine also loads the Write Data into the appropriate Write FIFO prior to executing a Write transaction, or gets Read Data from the appropriate Read FIFO after a read transaction.

Once the instructions are received, the inner Single-Cycle Timed Loop (inner state machine) is used to manage the sending and receiving of data to and from the devices. This state machine executes one state each clock cycle. Inside this loop is where the data is read from the Write FIFO, sent to the device, and where data is read from the device, and loaded into the Read FIFO. Transactions are built with a series of Operations, some of which are used as components in other transactions. Operations are built with a series of Procedures, which are used as components in other Operations.

Write I2C:

VIs:
WSVCAP Shell\WSVCAP Shell Host Top Level\WSVCAP SHL Host TL I2C Write - Single Device.vi
WSVCAP Shell\WSVCAP Shell Host Top Level\WSVCAP SHL Host TL I2C Write - Multiple Device - Common Data.vi
WSVCAP Shell\WSVCAP Shell Host Top Level\WSVCAP SHL Host TL I2C Write - Multiple Device - Unique Data.vi

Summary:
Write operations will send data from the Host PC to the selected I2C device(s) at the specified sub address. There is no data returned.

Details:
I. Assemble Write Instructions
  A. Build Write Data
    1. Low Level VI takes three separate 2-D arrays, one for each bank of I2C Devices (16 Devices in each bank)
  B. Build Socket Mask
    1. DUT Mask is used to disable write operations to unused devices
    2. DUT Numbering scheme is mapped onto Socket Numbering scheme
II. Build Write Instruction Message
  A. Write Data, Socket Mask, and Subaddres are combined into a Write Instruction Message
  B. Sent Write Instruction Message to Host Low Level Core Manager
III. Write Data to Device
  A. Send Write Instructions to FPGA Core
  B. Send Write Data to FPGA Core through I2C FIFO
  C. Wait for FPGA to finish Write Operation (Monitor Flags from FPGA Front Panel)
  D. Read Status message from FPGA

Read I2C:

VIs:
WSVCAP Shell\WSVCAP Shell Host Top Level\WSVCAP SHL Host TL I2C Read - Single Device.vi
WSVCAP Shell\WSVCAP Shell Host Top Level\WSVCAP SHL Host TL I2C Read - Multiple Device.vi

Summary:
Read operations will send read requests from the Host PC to the selected I2C device(s) at the specified sub address, and the data read will be returned.

Details:
I. Assemble Read Instructions
  A. Build Socket Mask
    1. DUT Mask is used to disable write operations to unused devices
    2. DUT Numbering scheme is mapped onto Socket Numbering scheme
II. Build Read Instruction Message
  A. Socket Mask, and Subaddres, number of bytes to read are combined into a Read Instruction Message
  B. Sent Read Instruction Message to Host Low Level Core Manager
III. Read Data from Device(s)
  A. Send Read Instructions to FPGA Core
  B. Send Read Instructions to FPGA Core through FPGA Front Panel
  C. Wait for FPGA to finish Read Operation (Monitor Flags from FPGA Front Panel)
  D. Read Data from FPGA
  E. Read Status message from FPGA

Transactions:

I2C-Read Data - Inputs[Slave Address, Sub Address, Bytes to Read]; Outputs[Read Data Array, Status Array]
I2C-Write Data - Inputs[Slave Address, Sub Address, Write Data Array]; Outputs[Status Array]

Operations:

I2C-Send Slave Address
I2C-Send Sub Address
I2C-Send Data Bytes
I2C-Read Data Bytes

Procedures:

I2C-Send Start Sequence
I2C-Send Byte
I2C-Read Slave ACK
I2C-Write Master ACK
I2C-Receive Byte
I2C-Send Stop Sequence

Transaction: I2C-Read Data

I. **Operation: I2C-Send Slave Address (W)**
II. **Operation: I2C-Send Sub Address**
III. **Operation: I2C-Send Slave Address (R)**
IV. **Operation: I2C-Read Data Bytes**
V. Transfer Read FIFO to Read Data Array
VI. Transfer Error Info to Status Array

Transaction: I2C-Write Data

I. Transfer Write Data Array to Write FIFO
II. **Operation: I2C-Send Slave Address (W)**
III. **Operation: I2C-Send Sub Address**
IV. **Operation: I2C-Send Data Bytes**
V. Transfer Error Info to Status Array

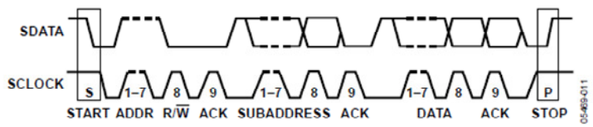


Figure 22. Bus Data Transfer

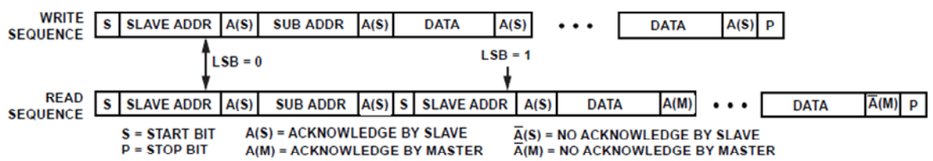


Figure 23. Write and Read Sequences

Operation: I2C-Send Slave Address
A: **Procedure: I2C-Send Start Sequence**
B: Load Write Byte [1-7] with Slave Address
C: Load Write Byte [0] with R/W Bit (0=Write, 1=Read)
D: **Procedure: I2C-Send Addres Byte**
E: **Procedure: I2C-Read Slave ACK**

Operation: I2C-Send Sub Address
A: Load Write Byte with Sub Address
B: **Procedure: I2C-Send Address Byte**
C: **Procedure: I2C-Read Slave ACK**

Operation: I2C-Send Data Bytes
A: Read I2C Write FIFO
If Write FIFO Timed Out (No more Data)
Go to D
If Write FIFO Not Timed Out (Data Available)
Load Write Bits (U32)
Go to B
B: **Procedure: I2C-Send Data Byte**
C: **Procedure: I2C-Read Slave ACK**
Go to A
D: **Procedure: I2C-Send Stop Sequence**

Operation: I2C-Read Data Bytes
A: Load Read Bytes Remaining with Bytes to Read
B: **Procedure: I2C-Receive Byte**
Decrement Read Bytes Remaining
If Read Bytes Remaining = 0
Set Master ACK = High
If Read Bytes Remaining > 0
Decrement Read Bytes Remaining
Set Master ACK = Low
C: **Procedure: I2C-Write Master ACK**
If Master ACK = High
Goto D
If Master ACK = Low
Go to B
D: **Procedure: I2C-Send Stop Sequence**

Procedure: I2C-Send Start Sequence
1. Set SCL = Low
2. Wait Half Cycle
3. Set SDA = High
4. Wait Half Cycle
5. Set SCL = High
6. Wait Half Cycle
7. Set SDA = Low
8. Wait Half Cycle
9. Set SCL = Low
10. Wait Half Cycle

Procedure: I2C-Read Slave ACK
1. Set SCL = High (Start ACK Read)
2. Wait Half Cycle
3. Read SDA = Slave ACK
4. Wait Half Cycle
5. Set SCL = Low (End ACK Read)

Procedure: I2C-Send Data Byte
1. Set Write Bits Remaining = 8
2. Load Write Bit (U32) from Write FIFO
3. Set SDA = Write Bit (U32) [i] (i: 7 >> 0)
   Decrement Write Bits Remaining
4. Wait Half Cycle
5. Set SCL = High (Start Bit Write)
6. Wait Half Cycle
7. Set Clock Low (End Bit Write)
8. Wait Half Cycle
9. If Write Bits Remaining = 0
   Finished
   If Write Bits Remaining > 0
   Go to **2**

Procedure: I2C-Send Address Byte
1. Set Write Bits Remaining = 8
2. Set SDA = Write Byte [i] (i: 7 >> 0)
   Decrement Write Bits Remaining
3. Wait Half Cycle
4. Set SCL = High (Start Bit Write)
5. Wait Half Cycle
6. Set Clock Low (End Bit Write)
7. Wait Half Cycle
8. If Write Bits Remaining = 0
   Finished
   If Write Bits Remaining > 0
   Go to **2**

Procedure: I2C-Send Stop Sequence
1. Set SCL = Low
2. Wait Half Cycle
3. Set SDA = Low
4. Wait Half Cycle
5. Set SCL = High
6. Wait Half Cycle
7. Set SDA = High
8. Wait Half Cycle
9. Set SCL = Low
10. Wait Half Cycle

Procedure: I2C-Write Master ACK
1. Set SDA = Master ACK
2. Wait Half Cycle
3. Set SCL = High (Start ACK Write)
4. Wait Half Cycle
5. Set SCL = Low (End ACK Write)
6. Wait Half Cycle
7. Set SDA = Low

Procedure: I2C-Receive Byte
1. Set Read Bits Remaining to 8
2. Set SCL = High (Start Bit Write)
3. Wait Half Cycle
4. Read SDA = Read Bit
   Load Read Bit into Read FIFO
   Decrement Read Bits Remaining
5. Wait Half Cycle
6. Set SCL = Low (End Bit Write)
7. Wait Half Cycle
8. If Read Bits Remaining = 0
   Finished
   If Read Bits Remaining > 0
   Go to **2**

ATEC Matrix Corporation	
Document Title: 1050 LabVIEW Software Design	**Date:** 07/5/2014
Document Number: WS-10024	**Rev:**
Page Title: I2C - Target FPGA	**Page:** 4 of 8

Write RFFE:

VIs:
WSVCAP Shell\WSVCAP Shell Host Top Level\WSVCAP SHL Host TL RFFE Write - Single DUT.vi
WSVCAP Shell\WSVCAP Shell Host Top Level\WSVCAP SHL Host TL RFFE Write - Multiple Device - Common Data.vi

Summary:
Write operations will send data from the Host PC to the selected RFFE device(s) at the specified sub address. The RFFE commands are specified as a write string that includes both the data and the address information. This is done to allow commonality between Cycle and Non-Cycle modes. There is no data returned.

Details:
I. Assemble Write Instructions
  A. Build Write Data
    1. Low Level VI takes one 2-D array, with a column of data for each DUT (16-DUTs)
  B. Build Socket Mask
    1. DUT Mask is used to disable write operations to unused devices
    2. DUT Numbering scheme is mapped onto Socket Numbering scheme
II. Build Write Instruction Message
  A. Write Data, Socket Mask, are combined into a Write Instruction Message
  B. Sent Write Instruction Message to Host Low Level Core Manager
III. Write Data to Device
  A. Send Write Instructions to FPGA Core
  B. Send Write Data to FPGA Core through FPGA Front Panel
  C. Wait for FPGA to finish Write Operation (Monitor Flags from FPGA Front Panel)
  D. Read Status message from FPGA

Read RFFE:

VIs:
WSVCAP Shell\WSVCAP Shell Host Top Level\WSVCAP SHL Host TL RFFE Read - Single Device.vi
WSVCAP Shell\WSVCAP Shell Host Top Level\WSVCAP SHL Host TL RFFE Read - Multiple Device.vi

Summary:
Read operations will send read requests from the Host PC to the selected RFFE device(s), and the data read will be returned. The RFFE commands are specified as a write string that includes both the data and the address information. This is done to allow commonality between Cycle and Non-Cycle modes.

Details:
I. Assemble Read Instructions
  A. Build Socket Mask
    1. DUT Mask is used to disable write operations to unused devices
    2. DUT Numbering scheme is mapped onto Socket Numbering scheme
II. Build Read Instruction Message
  A. Socket Mask, and data are combined into a Read Instruction Message
  B. Sent Read Instruction Message to Host Low Level Core Manager
III. Read Data from Device(s)
  A. Send Read Instructions to FPGA Core
  B. Send Read Instructions to FPGA Core through FPGA Front Panel
  C. Wait for FPGA to finish Read Operation (Monitor Flags from FPGA Front Panel)
  D. Read Data from FPGA
  E. Read Status message from FPGA

RFFE Cycle Mode:

VIs:
WSVCAP Shell\WSVCAP Shell Host Top Level\WSVCAP SHL Host TL RFFE Write Cycle - Single DUT.vi
WSVCAP Shell\WSVCAP Shell Host Top Level\WSVCAP SHL Host TL RFFE Write Cycle - Multiple DUTs - Common Data.vi

Summary:
When the Cycle Mode is used the FPGA will manage repetitive write operations to the RFFE devices until instructed to stop

Details:
I. Assemble Cycle Instructions
  A. Build Socket Mask
  B. Load Commands
  C. Load Command Delays
II. Build Read Instruction Message
  A. Socket Mask, Commands and Command Delays are combined into a Cycle Instruction Message
  B. Sent Cycle Instruction Message to Host Low Level Core Manager
III. Start Cycling
  A. Send Cycle Instructions to FPGA
  B. Wait for Stop Cycle Command from Host

RFFE

Transaction: RFFE-Write Data

Pre-Transfer: Transfer Write Data Array to Write Memory
Set Write Bit Index = 0
Set Write CMD = 1
Set CMD Delay = CMD 1 Delay
Set Write CMD Length = CMD 1 Length

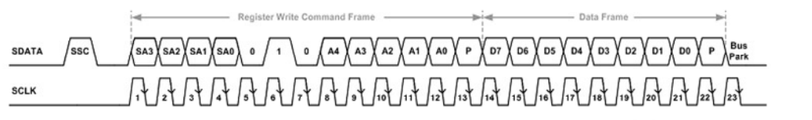
Transfer: :
I. **Operation: RFFE-Send Ext Reg Cmd Frame (W)**
II. **Operation: RFFE-Send Address**
III. **Operation: RFFE-Send Data Bytes**

Post-Transfer: Transfer Error Info to Status Array

Transaction: RFFE-Read Data

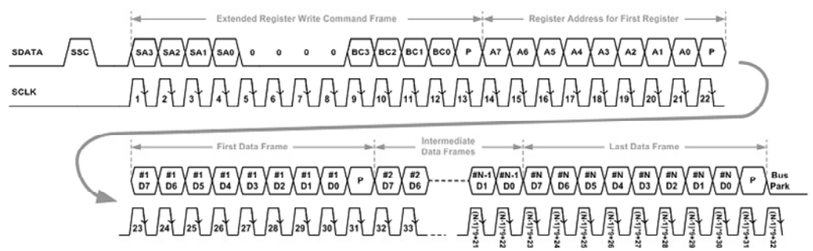
I. **Operation: RFFE-Send Ext Cmd Frame (R)**
II. **Operation: RFFE-Send Address**
II. **Operation: RFFE-Read Data Bytes**
Transfer Read FIFO to Read Data Array
Transfer Error Info to Status Array

RFFE Write



RFFE Basic Write Command Sequence

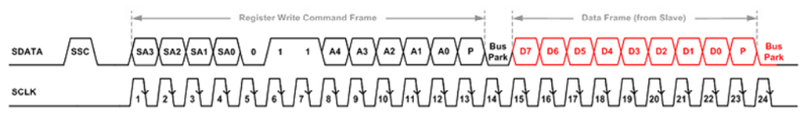
Bus Park makes sure the bus is at logic low before the turn-around:
If the preceding Parity bit is high, SDATA goes low on next rising SCLK
If the preceding Parity bit is low, SDATA remains low on the next rising SCLK



For extended Write/Read, the number of payload bytes is defined by the Byte Count field: BC<3:0> = #bytes to write/read – 1
e.g. BC<3:0>= 0000 write/read 1 byte
BC<3:0>= 0011 write/read 4 bytes

RFFE Basic Extended Command Sequence

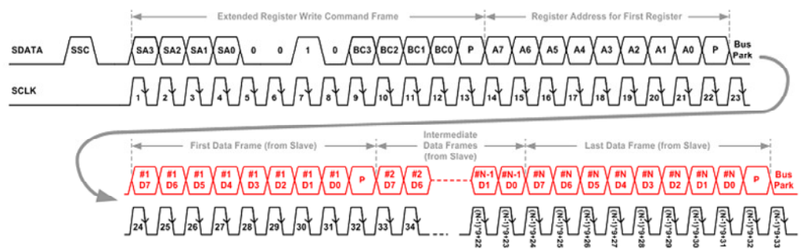
RFFE Read



RFFE Basic Read Command Sequence

Red is data written by the Slave

Bus Park makes sure the bus is at logic low before the turn-around:
If the preceding Parity bit is high, SDATA goes low on next rising SCLK
If the preceding Parity bit is low, SDATA remains low on the next rising SCLK



For extended Write/Read, the number of payload bytes is defined by the Byte Count field: BC<3:0> = #bytes to write/read – 1
e.g. BC<3:0>= 0000 write/read 1 byte
BC<3:0>= 0011 write/read 4 bytes

RFFE Extended Read Command Sequence

RFFE Operations

RFFE-Send Ex Reg Cmd Frame
RFFE-Send Ex Reg Address
RFFE-Write Data
RFFE-Read Data Bytes

RFFE Procedures

RFFE-Send SSC
RFFE-Send Data [DS]
RFFE-Receive Byte
RFFE-Bus Park

Operation: RFFE-Read Data

A: **Procedure: RFFE-Send SSC**
B: Load Write Word [0-3] with Slave Address
Load Write Word [4-7] with R/W Bits (0010=Read)
Load Write Byte [8-11] with Byte Count - 1
Procedure: RFFE-Send Data [12]
C: Load Write Byte (0-7) with Address
Procedure: RFFE-Send Data [8]
D: **Procedure: RFFE-Bus Park**
Load Read Bytes Remaining with Byte Count
E: **Procedure: RFFE-Receive Byte**
Decrement Read Bytes Remaining
If Read Bytes Remaining = 0
Goto
If Read Bytes Remaining > 0
Decrement Read Bytes Remaining
Go to B
F: **Procedure: RFFE Bus Park**
G: **Procedure: SPI/RFFE - Clear Lines**

Operation: RFFE-Write Data

A: **Procedure: RFFE-Send SSC**
B: Load Write Byte [0-3] with Slave Address
Load Write Byte [4-7] with R/W Bits (0000=Write)
Load Write Byte [8-11] with Byte Count - 1
Procedure: RFFE-Send Data [12]
C: Load Write Byte (0-7) with Address
Procedure: RFFE-Send Data [8]
D: Load Write Byte (0-7) with Address
Procedure: RFFE-Send Data [8]Set Wait Ticks Remaining = CMD Delay (Ticks)
If CMD Delay = 0
Load Next Command
Increment Cycle Count
Set CMD 1-4 = Command n
Set CMD Length (Bits) = Command n Length
Set CMD Delay (Ticks) = Command n Delay
Set Write Bit Index [0-48] = 0
Preload Bit 0 (for internally pipelined memory read inside XFR VI)
If Stop Cycle = True
Goto **D:**
If Stop Cycle = False
Goto **A:**
If CMD Delay > 0
Goto **C:**
E: **Procedure: RFFE/SPI-Wait CMD Delay**
F: **Procedure: SPI/RFFE-Clear Lines**

Procedure: RFFE-Clear Lines

1. Set SDAA = Low

Procedure: RFFE-Send SSC

1a. Set SCL = Low
1b. Set SDA = Low
1c. If Wait Ticks > 0
Goto **2**
If Wait Ticks = 0
Goto **3a**
Set Wait Ticks Ramaining = Wait Ticks
2. Wait Half Cycle
3a. Set SDA = High
3b. If Wait Ticks > 0
Goto **4**
If Wait Ticks = 0
Goto **5**
Set Wait Ticks Ramaining = Wait Ticks
4. Wait Half Cycle
5a. Set SDA = Low
5b. If Wait Ticks > 0
Goto **6**
If Wait Ticks = 0
Finished
Set Wait Ticks Ramaining = Wait Ticks
6. Wait Half Cycle

Procedure: RFFE-Receive Data

1a. Set SCL = High (Start Bit Read)
1b. If Start of Procedure = True
Set SDAA Enabled = False
1d. If Wait Ticks > 0
Goto **2**
If Wait Ticks = 0
Goto **3a**
Set Wait Ticks Ramaining = Wait Ticks
2. Wait Half Cycle
3a. Set SCL Low (End Bit Read)
3b. If End of Byte = False
Read SDA into Read FIFO
Decrement Read Bits Remaining
3c. If End of Data = True
Goto **5a**
If End of Data = False
If Wait Ticks > 0
Goto **4**
If Wait Ticks = 0
Goto **1a**
3c. Set Wait Ticks Ramaining = Wait Ticks
4a. Wait Half Cycle
4b. If End of Data = True
Goto **5a**
If End of Data = False
Goto **1a**
5a. Set SCL = High (Start Bus Park)
5b. Set SDAA Enable = True
5c. If Wait Ticks > 0
Goto **6**
If Wait Ticks = 0
Goto **7a**
Set Wait Ticks Ramaining = Wait Ticks
6. Wait Half Cycle
7a. Set SCL Low (End Bus Park)
7b. If Wait Ticks > 0
Goto **8**
If Wait Ticks = 0
Finished
7c. Set Wait Ticks Ramaining = Wait Ticks
8. Wait Half Cycle

Procedure: RFFE-Send Data

1a. Set SCL = High (Start Bit Write)
1b. If RFFE Cycle Mode = True
Set SDAA = RFFE Cycle CMD [Write Bit Index]
If RFFE Cycle Mode = False
Set SDAA = Next Element in DUT Write Data FIFO
1d. If Wait Ticks > 0
Goto **2**
If Wait Ticks = 0
Goto **3a**
Set Wait Ticks Ramaining = Wait Ticks
2. Wait Half Cycle
3a. Set SCL Low (End Bit Write)
3b. If Wait Ticks > 0
Goto **4**
If Wait Ticks = 0
If End of Address/Data Byte = True
Goto **5a**
If End of Address/Data Byte = False
Goto **1a**
Increment Write Bit Index [0-47]
3c. Set Wait Ticks Ramaining = Wait Ticks
4a. Wait Half Cycle
4b. If End of Address/Data Byte = True
Goto **5a**
If End of Address/Data Byte = False
Goto **1a**
5a. Set SCL = High (Start Parity Bit)
5b. Set SDAA = Low (Parity Bit always set to Low)
5c. If Wait Ticks > 0
Goto **6**
If Wait Ticks = 0
Goto **7a**
Set Wait Ticks Ramaining = Wait Ticks
6. Wait Half Cycle
7a. Set SCL Low (End Parity Bit)
7b. If Wait Ticks > 0
Goto **8**
If Wait Ticks = 0
If End of Transfer = True
Goto **9a**
If End of Address/Data Byte = False
Goto **1a**
7c. Set Wait Ticks Ramaining = Wait Ticks
8a. Wait Half Cycle
8b. If End of Transfer = True
Goto **9a**
If End of Address/Data Byte = False
Goto **1a**
9a. Set SCL = High (Start Bus Park)
9b. Set SDAA = Low
9c. If Wait Ticks > 0
Goto **10**
If Wait Ticks = 0
Goto **11a**
Set Wait Ticks Ramaining = Wait Ticks
10. Wait Half Cycle
11a. Set SCL Low (End Bus Park)
11b. If Wait Ticks > 0
Goto **12**
If Wait Ticks = 0
Finished
11c. Set Wait Ticks Ramaining = Wait Ticks
12. Wait Half Cycle

ATEC Matrix Corporation	
Document Title: 1050 LabVIEW Software Design	**Date:** 07/5/2014
Document Number: WS-10024	**Rev:**
Page Title: RFFE - Target FPGA	**Page:** 6 of 8

Write / Read SPI:

VIs:
WSVCAP Shell\WSVCAP Shell Host Top Level\WSVCAP SHL Host TL SPI Write_Read - Single DUT.vi
WSVCAP Shell\WSVCAP Shell Host Top Level\WSVCAP SHL Host TL SPI Write_Read - Multiple DUTs - Common Data.vi
WSVCAP Shell\WSVCAP Shell Host Top Level\WSVCAP SHL Host TL SPI Write_Read - Multiple DUTs - Unique Data.vi

Summary:
Write operations will send data from the Host PC to the selected SPI device(s). The SPI commands are specified as a write string that includes both the data and the address information. This is done to allow commonality between Cycle and Non-Cycle modes. Using the SPI bus means that during each clock cycle a bit is both written out to (falling edge), and read back from (rising edge) the device.

Details:
I. Assemble Write Instructions
  A. Build Write Data
    1. Low Level VI takes one 2-D array, with a column of data for each DUT (16-DUTs)
  B. Build Socket Mask
    1. DUT Mask is used to disable write operations to unused devices
    2. DUT Numbering scheme is mapped onto Socket Numbering scheme
II. Build Write Instruction Message
  A. Write Data, Socket Mask, are combined into a Write Instruction Message
  B. Sent Write Instruction Message to Host Low Level Core Manager
III. Write Data to Device
  A. Send Write Instructions to FPGA Core
  B. Send Write Data to FPGA Core through FPGA Front Panel
  C. Wait for FPGA to finish Write Operation (Monitor Flags from FPGA Front Panel)
  D. Get Read Data from FPGA through FPGA Front Panel
  D. Read Status message from FPGA

SPI Cycle Mode:

VIs:
WSVCAP Shell\WSVCAP Shell Host Top Level\WSVCAP SHL Host TL SPI Write Cycle - Single DUT.vi
WSVCAP Shell\WSVCAP Shell Host Top Level\WSVCAP SHL Host TL SPI Write Cycle - Multiple DUTs - Common Data.vi

Summary:
When the Cycle Mode is used the FPGA will manage repetitive write operations to the SPI devices until instructed to stop

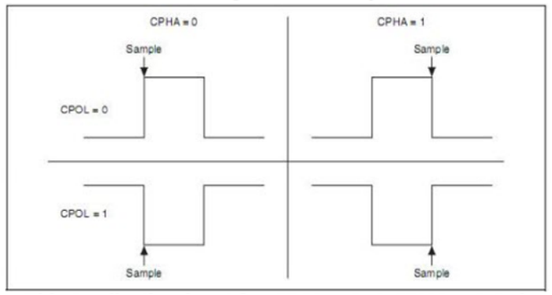
Details:
I. Assemble Cycle Instructions
  A. Build Socket Mask
  B. Load Commands
  C. Load Command Delays
II. Build Read Instruction Message
  A. Socket Mask, Commands and Command Delays are combined into a Cycle Instruction Message
  B. Sent Cycle Instruction Message to Host Low Level Core Manager
III. Start Cycling
  A. Send Cycle Instructions to FPGA
  B. Wait for Stop Cycle Command from Host

SPI

Transaction: SPI-Write/Read Data

Pre-Transfer: Transfer Write Data Array to Write Memory
Set Write Bit Index = 0
Set Write CMD = 1
Set CMD Delay = CMD 1 Delay
Set Write CMD Length = CMD 1 Length
Transfer: : **Operation: SPI-Write/Read Data**
Post-Transfer: Transfer Read FIFO to Read Data Array
Post-Transfer: Transfer Error Info to Status Array

SPI Polarity and Phase Options



This Application is set for CPOL = 0 / CPHA = 0 Only

Operation: SPI-Write/Read Data
A: **Procedure: SPI-Clear Lines**
Set "Enable SPI" = TRUE
B: **Procedure: SPI-Send/Receive Byte**
Set Wait Ticks Remaining = CMD Delay (Ticks)
If CMD Delay = 0
Load Next Command
Increment Cycle Count
Set CMD 1-4 = Command n
Set CMD Length (Bits) = Command n Length
Set CMD Delay (Ticks) = Command n Delay
Set Write Bit Index [0-39] = 0
Preload Bit 0 (for internally pipelined memory read inside XFR VI)
If Stop Cycle = True
Goto **D:**
If Stop Cycle = False
Goto **B:**
If CMD Delay > 0
Goto **C:**
C: **Procedure: SPI-Wait CMD Delay**
D: **Procedure: SPI-Clear Lines**

Procedure: SPI-Clear Lines
1a. If Enable SPI = False
Set SPI_SDEN = Low
If Enable SPI = True
Set SPI_SDEN = High
1b. Set SPI_SCLK = Low

Procedure: SPI-Send/Read Data Byte
1a. Set SPI_SCLK = Low (Bit Write)
1b. Set SPI_SDAA = Write Memory:CMD N:Bit n [N:1-4] [n:0-39]
1c. If Wait Ticks > 0
Goto **2**
If Wait Ticks = 0
Goto **3a**
Set Wait Ticks Ramaining = Wait Ticks
2. Wait Half Cycle
3a. Set SPI_SCLK = High (Bit Read)
3b. If SPI Cycle Mode = False
Read SPI_DATO = Bit n of Read FIFO
3c. Increment Write Bit Index
If Wait Ticks > 0
Goto 4
If Wait Ticks = 0
Goto 1a
4a. Wait Half Cycle
4b. If Write Bit Index = Write CMD Length
Set Finished
If Write Bit Index < Write CMD Length
Go to 1***a***

Procedure: SPI-Wait CMD Delay
1a. At first tick (single cycle)
Run Procedure:SPI-Clear Lines
1b. Wait CMD Delay (ticks)

ATEC Matrix Corporation	
Document Title: 1050 LabVIEW Software Design	**Date:** 07/5/2014
Document Number: WS-10024	**Rev:**
Page Title: SPI - Target FPGA	**Page:** 8 of 8