

**CECS 460 – UART Chip Specification with TSI**

Benjamin Santos

May 1, 2018

­­­­­

*This page intentionally left blank*

**Table of Contents**

I. Introduction……………………………………………………………………………… 5  
II. Documents………………………………………………………………………………. 5  
III. Requirements …………………………………………………………………………. 6  
IV. Top Level ……………………………………………………………………………… 7  
 a. Description ……………………………………………………………………… 7  
 b. Block Diagram …………………………………………………………………. 8  
 c. Data Flow Description .…………………………………………………………. 10  
 d. I/O ………………………………………………………………………..……... 10  
 e. Clocks ………………………………………………………………………….. 10  
 f. Resets …………………………………………………………………………… 10  
 g. Software ………………………………………………………………………… 10  
V. Externally Developed Blocks ………………………………………………………….. 11  
 a. Description ……………………………………………………………….……... 11  
 b. Block Diagram …………………………………………………………………. 12  
 c. I/O ………………………………………………………………………………. 13  
 d. Register Map ……………………………………………………………..……... 13  
VI. Internally Developed Blocks ………………………………………………………….. 14  
 1. UART  
 a. Description ……………………………………………………………… 14  
 b. Block Diagram ………………………………………………………….. 15  
 c. I/O ………………………………………………………………………. 16  
 d. State Machines …………………………………………………………. 16  
 e. Register Map ……………………………………………………………. 17  
 f. Verification ……………………………………………………………… 17  
 2. TX  
 a. Description ……………………………………………………………… 19  
 b. Block Diagram ………………………………………………………….. 19  
 c. I/O ………………………………………………………………………. 20  
 d. State Machines …………………………………………………………. 20  
 e. Register Map ……………………………………………………………. 21  
 f. Verification ……………………………………………………………… 22  
 3. RX  
 a. Description ……………………………………………………………… 23  
 b. Block Diagram ………………………………………………………….. 24  
 c. I/O ………………………………………………………………………. 25  
 d. State Machines …………………………………………………………. 25  
 e. Register Map ……………………………………………………………. 26  
 f. Verification ……………………………………………………………… 27  
VII. Chip Level Verification ……………………………………………………………… 27  
VIII. Chip Level Test ……………………………………………………………………… 28  
VIV. Tramelblaze Instruction Set …………………………………………………………. 31  
X. All Verilog Files ………………………………………………………………………. 39

**1 Introduction**

This project creates a UART chip has the RX and TX engine that receives and transmits serial data, respectively, alongside the 16-bit Tramelblaze Microcontroller written by John Tramel. The Transmit Engine provides the bit to be output onto the serial capture program according to the instruction set from the assembly code programmed into the Tramelblaze.

**II Documents**

1. External Documents
2. Customer Specifications
3. Industry Standards
4. References
5. Internal Documents
6. Internally generated specifications
7. Applicable methodology documents
8. Chip Test Plan
9. Analyses or trade studies

**III Requirements**

* The Chip should not start transmitting data when the TXRDY signal is not active and the TXRDY should stay active while it is in the middle of transmitting data.
* The Chip should not receive any data while the RXRDY signal is not active and the RXRDY should stay active while it is in the middle of receiving data.
* All modules in the program should use Asynchronous reset.
* The Receiver (Serial Capture Program) and the UART should be in the same BAUD rate in order to have the same data transmitted and received. For each BAUD rate it counts to the following integer count:

|  |  |  |  |
| --- | --- | --- | --- |
| **Baud[3:0] (switches)** | **Rate** | **Bit Time** | **Count** |
| 0000 | 300 | 0.003333333 | 333,333 |
| 0001 | 1200 | 0.000833333 | 83,333 |
| 0010 | 2400 | 0.000416667 | 41,667 |
| 0011 | 4800 | 0.000208333 | 20,833 |
| 0100 | 9600 | 0.000104167 | 10,417 |
| 0101 | 19200 | 5.20833E-05 | 5,208 |
| 0110 | 38400 | 2.60417E-05 | 2,604 |
| 0111 | 57600 | 1.73611E-05 | 1,736 |
| 1000 | 115200 | 8.68056E-06 | 868 |
| 1001 | 230400 | 4.34028E-06 | 434 |
| 1010 | 460800 | 2.17014E-06 | 217 |
| 1011 | 921600 | 1.08507E-06 | 109 |

**IV Top Level**

A. Description

The Top Level module of this project connects the CORE logic module and the Technology Specific Instantiations module to the Nexys4DDR board in order to communicate with the serial capture program (i.e Realterm). The CORE logic module has the UART chip and the Tramelblaze, while the Technology Specific Instantiations (TSI) module allocates each I/O of the chip to meet the electrical and timing requirements of the external interface to avoid any shorting on the hardware aspect of the chip.

Moreover, the CORE logic of this project echoes the user’s input from the keyboard onto the serial capture program with some preset functions such as:

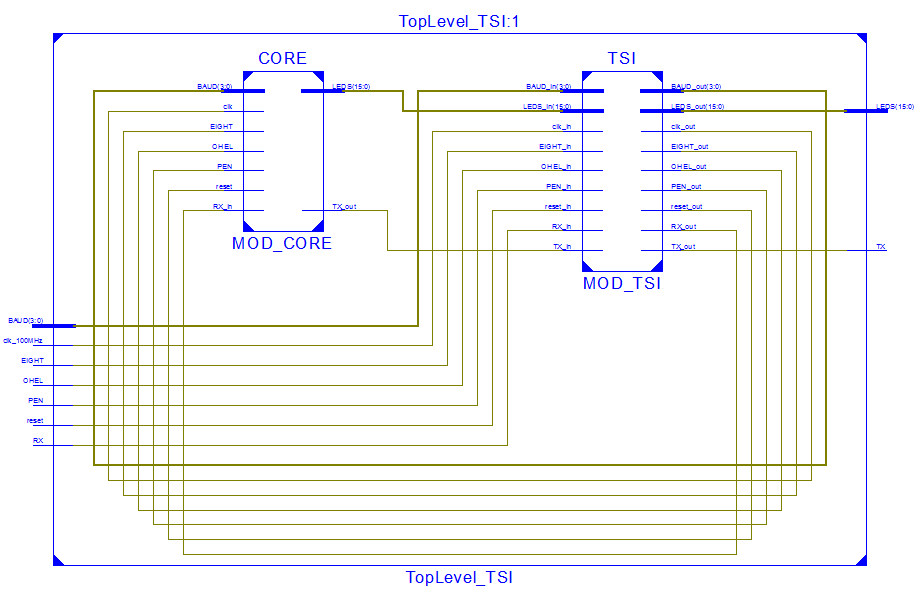
@ - Prints out the current character written on screen.

\* - Prints out my Hometown which is “Naga City”

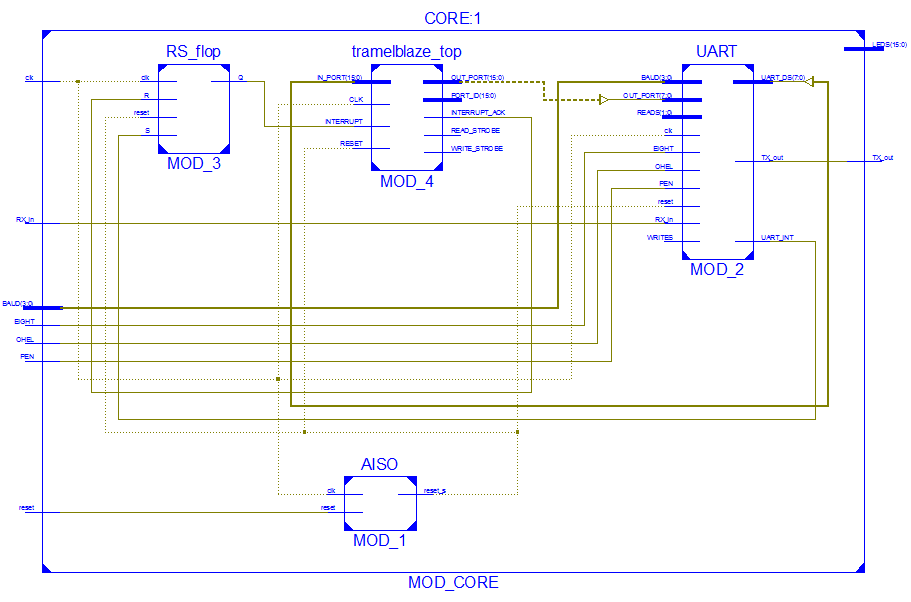
BACKSPACE – Removes the current character the pointer is pointing to.

B. Block Diagram

1. Connection of TSI and Core



2. CORE



C. Data Flow Description

- The UART chip accepts the RX signal and leads it to the RX engine and generates the   
 8 bit data with the TX engine to be sent to the Tramelblaze.

- The TX engine also generates the TX signal to be sent to the serial capture program (Realterm) in accordance to the WRITES and OUT\_PORT signal coming from the Tramelblaze.

D. I/O

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Bits** | **Type** | **Description** |
| clk\_100MHz | 1 | Input | 100MHz clock provided by the Nexys4 board |
| reset | 1 | Input | Main reset of the program set to a button |
| BAUD | 4 | Input | Determines Baud rate of the Transmit Engine using 4 switches |
| EIGHT | 1 | Input | EIGHT flag if data being transmitted or received  is 8 bits. |
| PEN | 1 | Input | Parity Enable flag |
| OHEL | 1 | Input | Odd high or Even Low parity bit flag |
| RX\_in | 1 | Input | The 1 bit serial input to the UART generated by the serial capture program along with the Tramelblaze |
| TX\_out | 1 | Output | The 1 bit serial output of the UART to be received by the serial capture program (i.e Realterm) |
| LEDS | 16 | Output | Running LEDs on the Nexys4DDR board |
| anode | 8 | Output | Anodes on the Nexys4DDR board |

E. Clocks

The Top Level uses the default 100MHz clock of the Nexys4DDR board.

F. Resets

The Reset for the Top Level is provided by a user input from the Nexys4DDR board (btnu). The Top Level generates an Asynchronous Reset with the AISO.v and uses it for all other modules.

G. Software

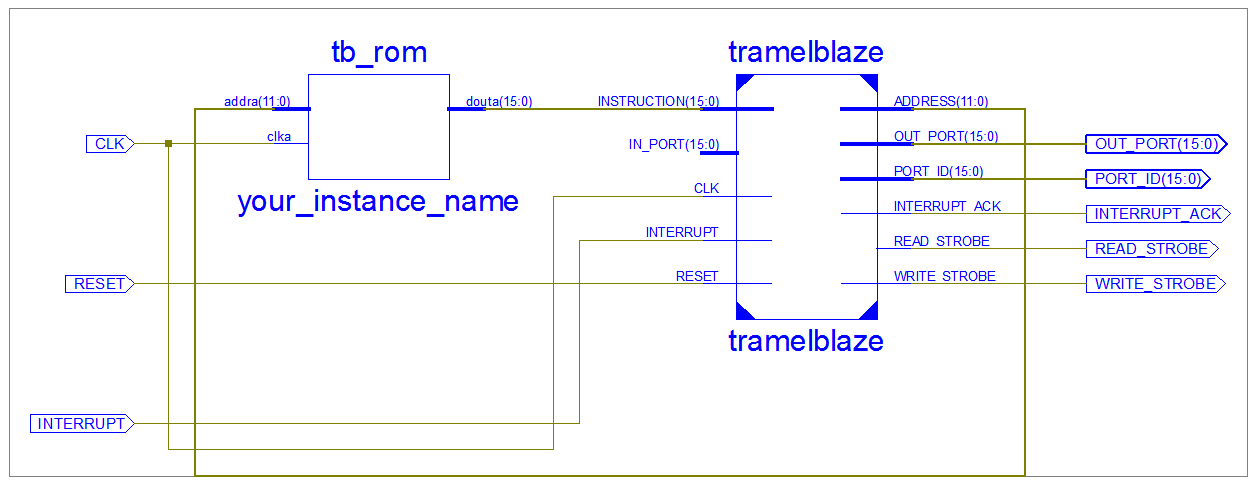
The project was written in Verilog using the XILINX ISE

**V. Externally Developed Blocks**

A. Description

Tramelblaze is the 16-bit Picoblaze microcontroller written by John Tramel. Tramelblaze utilizes two volatile memories, Stack (128 x 16) and Scratch (512 x 16) RAMs and one Instruction ROM (4096 x 16) where the user can generate the Block Memory for the instructions from an assembly code.

B. Block Diagram



C. I/O

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Bits** | **Type** | **Description** |
| Clock | 1 | Input | Clock of the Tramelblaze |
| Reset | 1 | Input | Reset of the Tramelblaze |
| IN\_PORT | 16 | Input | Data coming from the UART |
| INTERRUPT | 1 | Input | 1 clock cycle Interrupt flag |
| OUT\_PORT | 16 | Output | Data to be sent to the UART |
| PORT\_ID | 16 | Output | Address to be used by the Address Decoder |
| READ\_STROBE | 1 | Output | Flag generated when the Chip is currently reading data in. |
| WRITE\_SROBE | 1 | Output | Flag generated when the Chip is currently writing data out. |
| INTERRUPT\_ACK | 1 | Output | Flag that is high for 1 clock cycle when it just received an Interrupt. |

D. Register Map

|  |  |
| --- | --- |
| **Type** | **Definition** |
| R | Read only |
| W | Write only |
| RW | Read and Write |
| RC | Read and Clear |

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Bits** | **Description** |
| INSTRUCTION | RW | 16 | Instruction Set traversing through the Tramelblaze |
| ADDRESS | RW | 12 | Address of where the Instruction Set is being written or read |

**VI. Internally Developed Blocks**

**1. Universal Asynchronous Receiver-Transmitter (UART)**

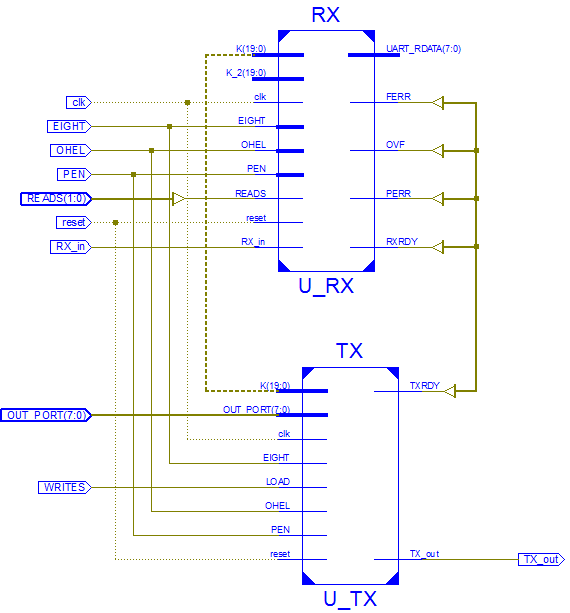
1. A. Description

The UART chip communicates with the Tramelblaze, the Nexys4DDR board and the serial capture program (Realterm).

- The UART chip accepts the RX signal and leads it to the RX engine and generates the   
 8 bit data with the TX engine to be sent to the Tramelblaze.

- The TX engine also generates the TX signal to be sent to the serial capture program (Realterm) in accordance to the WRITES and OUT\_PORT signal coming from the Tramelblaze.

1. B. Block Diagram



1. C. I/O

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Bits** | **Type** | **Description** |
| Clk | 1 | Input | Clock of the UART |
| Reset | 1 | Input | Reset of the UART |
| BAUD | 20 | Input | BAUD rate according to the Nexys4DDR board. |
| WRITES | 1 | Input | WRITES[0] flag from the Address Decoder |
| READS | 2 | Input | READS from the Address Decoder to determine whether the data being output (UART\_DS) is the status or the data. |
| OUT\_PORT | 8 | Input | OUT\_PORT from the Tramelblaze |
| EIGHT | 1 | Input | EIGHT flag if data being transmitted or received  is 8 bits. |
| PEN | 1 | Input | Parity Enable flag |
| OHEL | 1 | Input | Odd high or Even Low parity bit flag |
| RX\_in | 1 | Input | The 1 bit serial input to the UART generated by the serial capture program along with the Tramelblaze |
| TX\_out | 1 | Output | 1 bit serial output of the TX engine to be received by the Tramelblaze |
| UART\_DS | 8 | Output | Data to be sent to the Tramelblaze |
| UART\_INT | 1 | Output | Interrupt generated if either TXRDY or RXRDY is active for 1 clock cycle |

1. D State Machines

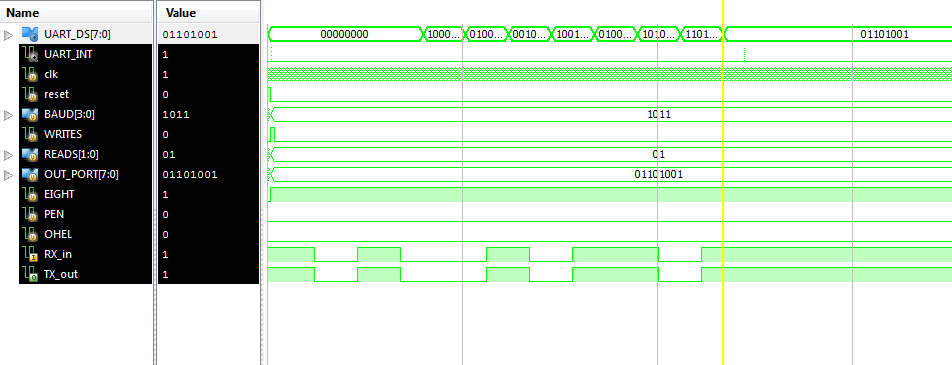
1. E Register Map

|  |  |
| --- | --- |
| **Type** | **Definition** |
| R | Read only |
| W | Write only |
| RW | Read and Write |
| RC | Read and Clear |

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Bits** | **Description** |
| K | RW | 1 | BAUD rate in integer |
| K\_2 | RW | 1 | Half of the BAUD rate in integer |
| TXRDY | RW | 1 | 1 Clock delay of input LOAD for the shift register |
| RXRDY | RW | 1 | Flag produced by Bit Time Counter that is active for one clock when the counter counts up to 1 bit time according to the BAUD rate. |
| UART\_STATUS | RW | 8 | Output of the loadable register from the input. |
| UART\_RDATA | RW | 1 | 10th bit to be input to the shift register |
| UART\_RXRDY | RW | 1 | 9th bit to be input to the shift register |
| UART\_TXRDY | RW | 1 | BAUD rate in integer |

1. F Verification

The following waveform has the RX\_out connected to the TX\_in in order to simulate an ECHO command from the Tramelblaze. Moreover the UART is given the input OUT\_PORT = 0x65 or 0110\_0101 while the BAUD rate is set on the fastest speed



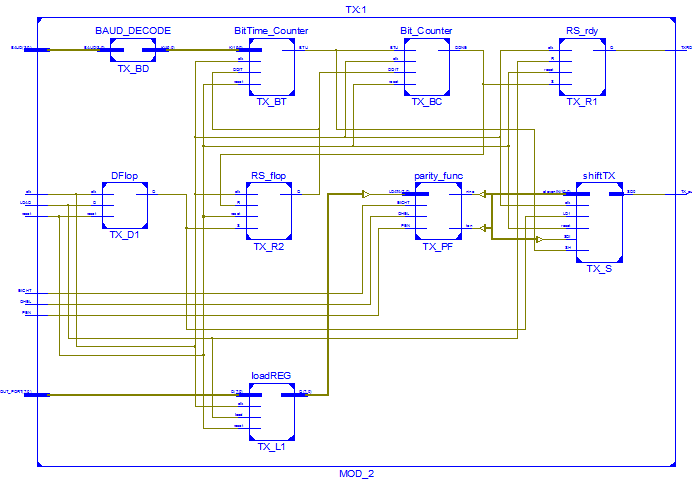
As what can be seen, the TX\_out along with the RX\_in, has the start bit(0) and starts changing the data from the Least Significant Bit to the Most Significant Bit (1010\_0110) and finally the stop bit(1).

**2. Transmit Engine (TX)**

2. A. Description

The Transmit Engine generates a 1 bit output for the serial capture device to display on the screen from the instruction set from the Tramelblaze

2. B. Block Diagram



2. C. I/O

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Bits** | **Type** | **Description** |
| Clk | 1 | Input | Clock of the TX engine |
| Reset | 1 | Input | Reset of the TX engine |
| K | 20 | Input | BAUD rate in integer form. |
| LOAD | 1 | Input | LOAD flag that is high when WRITES[0] from the Address Decoder is high |
| OUT\_PORT | 8 | Input | Data received by the Shift Register of the TX when LOAD is high. |
| EIGHT | 1 | Input | EIGHT flag if data being transmitted or received  is 8 bits. |
| PEN | 1 | Input | Parity Enable flag |
| OHEL | 1 | Input | Odd high or Even Low parity bit flag |
| TXRDY | 1 | Output | TXRDY flag if the TX engine is ready to transmit data. |
| TX\_out | 1 | Output | 1 bit serial output of the TX engine to be received by the Tramelblaze |

2. D. State Machines

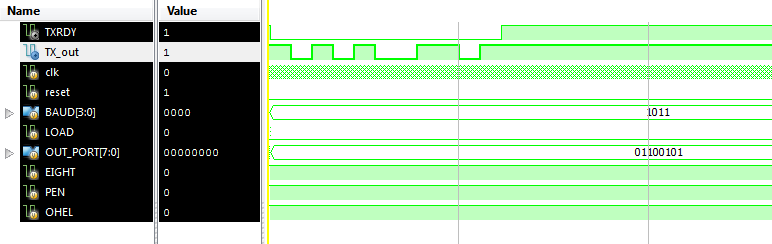
2. E. Register Map

|  |  |
| --- | --- |
| **Type** | **Definition** |
| R | Read only |
| W | Write only |
| RW | Read and Write |
| RC | Read and Clear |

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Bits** | **Description** |
| DOIT | RW | 1 | Flag used for the Bit Time Counter and Bit Counter |
| DONE | RW | 1 | Flag produced by Bit Counter that is active for one clock when the TX is done transmitting data |
| LOADD1 | RW | 1 | 1 Clock delay of input LOAD for the shift register |
| BTU | RW | 1 | Flag produced by Bit Time Counter that is active for one clock when the counter counts up to 1 bit time according to the BAUD rate. |
| LDATA | RW | 8 | Output of the loadable register from the input. |
| Ten | RW | 1 | 10th bit to be input to the shift register |
| Nine | RW | 1 | 9th bit to be input to the shift register |
| K | RW | 20 | BAUD rate in integer |

2. F. Verification

The following waveform is given the input OUT\_PORT = 0x65 or 0110\_0101 while the BAUD rate is set on the fastest speed



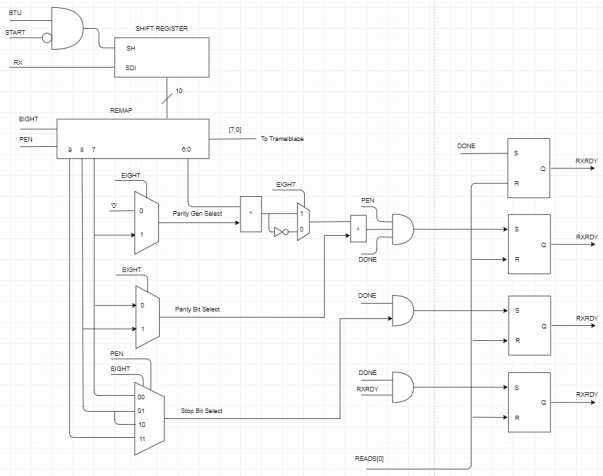
As what can be seen, the TX\_out outputs the start bit(0) and starts transmitting the data from the Least Significant Bit to the Most Significant Bit (1010\_0110) and finally the stop bit(1).

**3. Receive Engine (RX)**

3. A. Description

The Receive Engine receives a 1 bit input from the serial capture device and synchronizes the data collection with the TX communication. The Receive Engine is always polling the RX line looking for a high to low transition indicating the arrival of the START bit and a simple Moore state machine ensures that the START bit remains active until the Bit Time Counter is done counting (BTU is up) and it goes to the data collection part. Finally, the state machine lets the receive engine continue to receive data until the DONE flag from the Bit Counter is active.

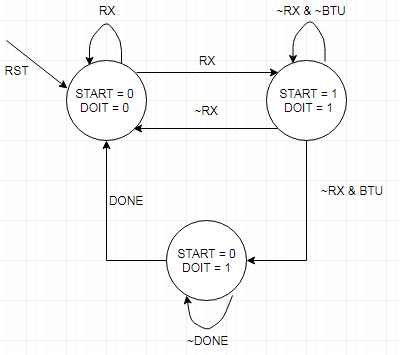
3. B. Block Diagram



3. C. I/O

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Bits** | **Type** | **Description** |
| Clk | 1 | Input | Clock of the RX |
| Reset | 1 | Input | Reset of the RX |
| K | 20 | Input | BAUD rate in integer form |
| K\_2 | 20 | Input | Half of the BAUD rate in integer form |
| RX\_in | 1 | Input | The 1 bit serial input to the UART generated by the serial capture program along with the Tramelblaze |
| EIGHT | 1 | Input | EIGHT flag if data being transmitted or received  is 8 bits. |
| PEN | 1 | Input | Parity Enable flag |
| OHEL | 1 | Input | Odd high or Even Low parity bit flag |
| READS | 1 | Input | The READS[0] bit from the Address Decoder and Tramelblaze for the UART\_STATUS flags |
| UART\_RDATA | 8 | Output | Data to be sent to the Tramelblaze directly from the Remapped Shift Register |
| OVF | 1 | Output | Overflow Error flag |
| FERR | 1 | Output | Framing Error flag |
| PERR | 1 | Output | Parity Error flag |
| RXRDY | 1 | Output | RXRDY flag |

3. D. State Machines



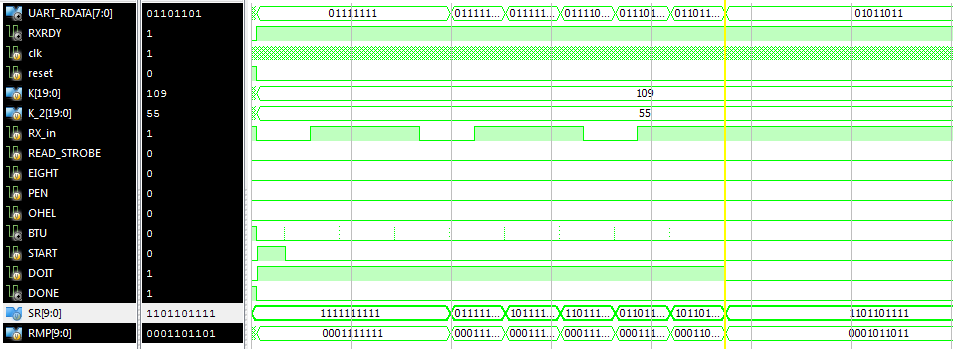
3. E. Register Map

|  |  |
| --- | --- |
| **Type** | **Definition** |
| R | Read only |
| W | Write only |
| RW | Read and Write |
| RC | Read and Clear |

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Bits** | **Description** |
| START | RW | 1 | Flag is high when RX is on the START bit |
| DOIT | RW | 1 | Flag used for the Bit Time Counter and Bit Counter |
| DONE | RW | 1 | Flag produced by Bit Counter that is active for one clock when the RX is done receiving data |
| BTU | RW | 1 | Flag produced by Bit Time Counter that is active for one clock when the counter counts up to 1 bit time according to the BAUD rate. |
| K\_in | RW | 20 | BAUD rate in integer according to the START bit |
| SH | RW | 1 | Flag for the Shift Register (BTU && ~START) |
| SR | RW | 10 | Shift Register |
| RMP | RW | 10 | Remapped Shift Register to send [6:0] to the Tramelblaze |
| parr2 | RW | 1 | Higher priority of the PERR RS flop. |
| SBS | RW | 1 | Stop Bit Select used for the FERR (DONE && ~SBS) |

3. F. Verification

The RX engine is fed the signals 0x5B = 0101\_1011 and as what can be seen, the UART\_RDATA signal gets the same bits sent from the testbench.



**VII. Chip Level Verification**

UART Verification shown above.

**VIII. Chip Level Test**

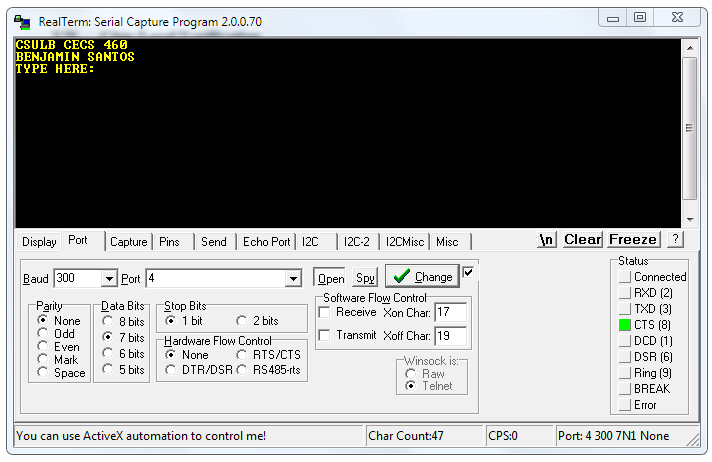
The CORE logic of this project echoes the user’s input from the keyboard onto the serial capture program with some preset functions such as:

@ - Prints out the current character written on screen.

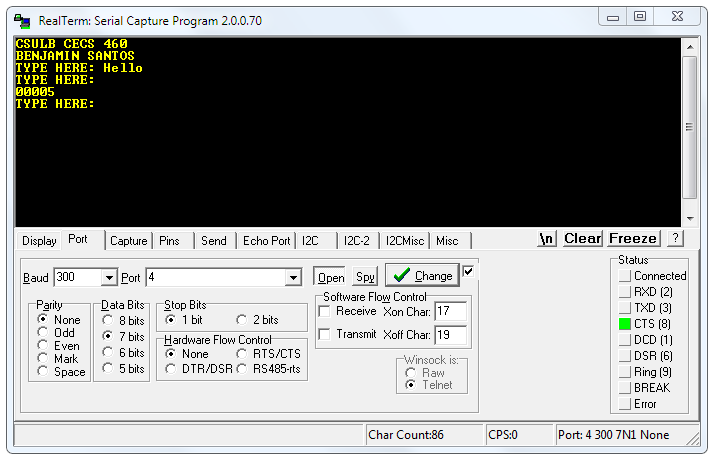
\* - Prints out my Hometown which is “Naga City”

BACKSPACE – Removes the current character the pointer is pointing to.

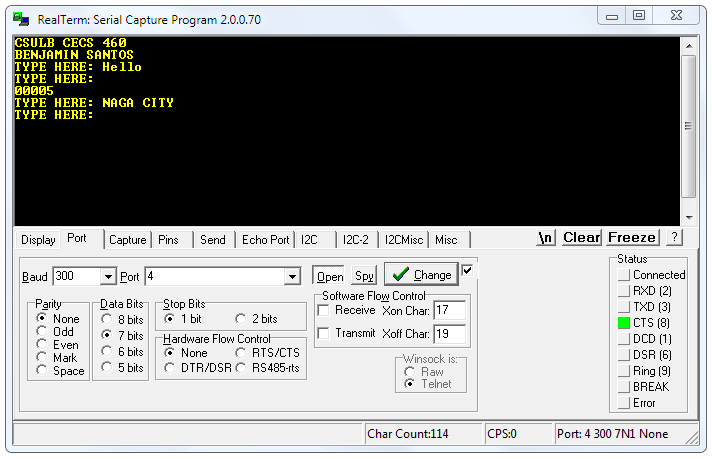
On launch:



Verification of @ function:



Verification of \* function:



Tramelblaze Instruction Set (Using Assembly Language) :

; Benjamin Santos

; CECS 460 - UART

; April 28, 2018

; declare constants for coding

ONE EQU 1000

ZERO EQU 0000

asc\_0 EQU 0030

asc\_1 EQU 0031

asc\_2 EQU 0032

asc\_3 EQU 0033

asc\_4 EQU 0034

asc\_5 EQU 0035

asc\_6 EQU 0036

asc\_7 EQU 0037

asc\_8 EQU 0038

asc\_9 EQU 0039

asc\_A EQU 0041

asc\_B EQU 0042

asc\_C EQU 0043

asc\_D EQU 0044

asc\_E EQU 0045

asc\_F EQU 0046

asc\_G EQU 0047

asc\_H EQU 0048

asc\_I EQU 0049

asc\_J EQU 004A

asc\_K EQU 004B

asc\_L EQU 004C

asc\_M EQU 004D

asc\_N EQU 004E

asc\_O EQU 004F

asc\_P EQU 0050

asc\_Q EQU 0051

asc\_R EQU 0052

asc\_S EQU 0053

asc\_T EQU 0054

asc\_U EQU 0055

asc\_V EQU 0056

asc\_W EQU 0057

asc\_X EQU 0058

asc\_Y EQU 0059

asc\_Z EQU 005A

asc\_Sp EQU 0020

asc\_CR EQU 000D

asc\_CR\_7 EQU 008D

asc\_LF EQU 000A

asc\_BS EQU 0008

asc\_BS\_7 EQU 0088

asc\_ST EQU 002A

asc\_ST\_7 EQU 00AA

asc\_AT EQU 0040

asc\_AT\_7 EQU 00C0

asc\_PS EQU 0025

asc\_AR EQU 003E

asc\_FF EQU 000C

asc\_NULL EQU 0000

TENTHOUSANDS EQU 0103

THOUSANDS EQU 0104

HUNDREDS EQU 0105

TENS EQU 0106

ONES EQU 0107

; WALKING ONES

WALKING EQU R0

WALK\_COUNT EQU R1

; ISR

TEMP EQU R2

COUNTER EQU R3

DATA EQU R4

CHAR\_FLAG EQU R5

NEW\_FLAG EQU R6

BACK\_FLAG EQU R7

HOME\_FLAG EQU R8

NEW\_POINTER EQU R9

BACK\_POINTER EQU RA

HOME\_POINTER EQU RB

CHAR\_COUNT EQU RC

BORDER\_COUNT EQU RD

TXOK EQU RE

CHAR\_POINTER EQU RF

; begin

START ENINT

; banner

LOAD TEMP, asc\_FF

STORE TEMP, 0000

LOAD TEMP, asc\_C

STORE TEMP, 0001

LOAD TEMP, asc\_S

STORE TEMP, 0002

LOAD TEMP, asc\_U

STORE TEMP, 0003

LOAD TEMP, asc\_L

STORE TEMP, 0004

LOAD TEMP, asc\_B

STORE TEMP, 0005

LOAD TEMP, asc\_Sp

STORE TEMP, 0006

LOAD TEMP, asc\_C

STORE TEMP, 0007

LOAD TEMP, asc\_E

STORE TEMP, 0008

LOAD TEMP, asc\_C

STORE TEMP, 0009

LOAD TEMP, asc\_S

STORE TEMP, 000A

LOAD TEMP, asc\_Sp

STORE TEMP, 000B

LOAD TEMP, asc\_4

STORE TEMP, 000C

LOAD TEMP, asc\_6

STORE TEMP, 000D

LOAD TEMP, asc\_0

STORE TEMP, 000E

LOAD TEMP, asc\_Sp

STORE TEMP, 000F

LOAD TEMP, asc\_CR

STORE TEMP, 0010

LOAD TEMP, asc\_LF

STORE TEMP, 0011

; Name

; 0012

LOAD TEMP, asc\_B ;B

STORE TEMP, 0012

LOAD TEMP, asc\_E ;E

STORE TEMP, 0013

LOAD TEMP, asc\_N ;N

STORE TEMP, 0014

LOAD TEMP, asc\_J ;J

STORE TEMP, 0015

LOAD TEMP, asc\_A ;A

STORE TEMP, 0016

LOAD TEMP, asc\_M ;M

STORE TEMP, 0017

LOAD TEMP, asc\_I ;I

STORE TEMP, 0018

LOAD TEMP, asc\_N ;N

STORE TEMP, 0019

LOAD TEMP, asc\_Sp ;

STORE TEMP, 001A

LOAD TEMP, asc\_S ;S

STORE TEMP, 001B

LOAD TEMP, asc\_A ;A

STORE TEMP, 001C

LOAD TEMP, asc\_N ;N

STORE TEMP, 001D

LOAD TEMP, asc\_T ;T

STORE TEMP, 001E

LOAD TEMP, asc\_O ;O

STORE TEMP, 001F

LOAD TEMP, asc\_S ;S

STORE TEMP, 0020

; prompt and new line

; 0021

LOAD TEMP, asc\_CR

STORE TEMP, 0021

STORE TEMP, 0101

LOAD TEMP, asc\_LF

STORE TEMP, 0022

STORE TEMP, 0102

LOAD TEMP, asc\_T

STORE TEMP, 0023

LOAD TEMP, asc\_Y

STORE TEMP, 0024

LOAD TEMP, asc\_P

STORE TEMP, 0025

LOAD TEMP, asc\_E

STORE TEMP, 0026

LOAD TEMP, asc\_Sp

STORE TEMP, 0027

LOAD TEMP, asc\_H

STORE TEMP, 0028

LOAD TEMP, asc\_E

STORE TEMP, 0029

LOAD TEMP, asc\_R

STORE TEMP, 002A

LOAD TEMP, asc\_E

STORE TEMP, 002B

LOAD TEMP, 003A ;:

STORE TEMP, 002C

LOAD TEMP, asc\_Sp

STORE TEMP, 002D

; backspace

; 002E

LOAD TEMP, asc\_BS

STORE TEMP, 002E

LOAD TEMP, asc\_Sp

STORE TEMP, 002F

LOAD TEMP, asc\_BS

STORE TEMP, 0030

; hometown

; 0031

LOAD TEMP, asc\_N

STORE TEMP, 0031

LOAD TEMP, asc\_A

STORE TEMP, 0032

LOAD TEMP, asc\_G

STORE TEMP, 0033

LOAD TEMP, asc\_A

STORE TEMP, 0034

LOAD TEMP, asc\_Sp

STORE TEMP, 0035

LOAD TEMP, asc\_C

STORE TEMP, 0036

LOAD TEMP, asc\_I

STORE TEMP, 0037

LOAD TEMP, asc\_T

STORE TEMP, 0038

LOAD TEMP, asc\_Y

STORE TEMP, 0039

; initializing registers

LOAD COUNTER, ZERO

LOAD WALKING, ONE

LOAD NEW\_POINTER, 0021

LOAD BACK\_POINTER, 002E

LOAD HOME\_POINTER, 0031

LOAD CHAR\_COUNT, 0000

LOAD NEW\_FLAG, 0000

LOAD BACK\_FLAG, 0000

LOAD HOME\_FLAG, 0000

LOAD CHAR\_POINTER, 0101

; main loop, walking ones

FIRST LOAD WALK\_COUNT, FFFF

OUTPUT WALKING, 0001

RR WALKING

JUMP DELAY

DELAY SUB WALK\_COUNT, 0001

COMP WALK\_COUNT, 0000

JUMPZ FIRST

JUMP DELAY

; address for interrupt service routine

ADDRESS 0500

; set up for in isr

ISR COMP NEW\_FLAG, FFFF ;right

JUMPZ NEWLINE

COMP BACK\_FLAG, FFFF

JUMPZ BACKSPACE

COMP HOME\_FLAG, FFFF

JUMPZ HOMETOWN

COMP CHAR\_FLAG, FFFF

JUMPZ CHARACTERS

COMP COUNTER, 002E

JUMPZ USERINPUT

FETCH TEMP, COUNTER

OUTPUT TEMP, 0000

ADD COUNTER, 0001

RETEN

USERINPUT INPUT TEMP, 0001

STORE TEMP, 0100

AND TEMP, 0001

COMP TEMP, 0001

JUMPZ RECEIVE

FETCH TEMP, 0100

AND TEMP, 0002

COMP TEMP, 0002

JUMPZ TXBLOCK

RETEN

TXBLOCK LOAD TXOK, FFFF

RETEN

RECEIVE INPUT DATA, 0000

COMP TXOK, FFFF

JUMPZ TRANSMIT

RETEN

TRANSMIT LOAD TXOK, 0000

COMP DATA, asc\_CR

JUMPZ NEWLINE

COMP DATA, asc\_CR\_7

JUMPZ NEWLINE

COMP DATA, asc\_BS

JUMPZ BACKCHECK

COMP DATA, asc\_BS\_7

JUMPZ BACKCHECK

COMP DATA, asc\_ST

JUMPZ HOMETOWN

COMP DATA, asc\_ST\_7

JUMPZ HOMETOWN

COMP DATA, asc\_AT

JUMPZ PRETENTHOUSAND

COMP DATA, asc\_AT\_7

JUMPZ PRETENTHOUSAND

ADD CHAR\_COUNT, 0001

STORE CHAR\_COUNT, 0200

OUTPUT CHAR\_COUNT, 0003

ADD BORDER\_COUNT, 0001

OUTPUT DATA, 0000

COMP BORDER\_COUNT, 0021

JUMPZ PRENEWLINE

RETEN

PRENEWLINE LOAD NEW\_FLAG, FFFF

RETEN

NEWLINE LOAD NEW\_FLAG, FFFF

FETCH TEMP, NEW\_POINTER

OUTPUT TEMP, 0000

ADD NEW\_POINTER, 0001

COMP NEW\_POINTER, 002E

JUMPZ NEWRESET

RETEN

NEWRESET LOAD NEW\_FLAG, 0000

LOAD NEW\_POINTER, 0021

LOAD BORDER\_COUNT, 0000

RETEN

BACKCHECK COMP BORDER\_COUNT, 0000

JUMPNZ BACKSPACE

LOAD DATA, asc\_NULL

OUTPUT DATA, 0000

RETEN

BACKSPACE LOAD BACK\_FLAG, FFFF

FETCH TEMP, BACK\_POINTER

OUTPUT TEMP, 0000

ADD BACK\_POINTER, 0001

COMP BACK\_POINTER, 0031

JUMPZ BACKRESET

RETEN

BACKRESET LOAD BACK\_FLAG, 0000

LOAD BACK\_POINTER, 002E

SUB BORDER\_COUNT, 0001

RETEN

HOMETOWN LOAD HOME\_FLAG, FFFF

FETCH TEMP, HOME\_POINTER

OUTPUT TEMP, 0000

ADD HOME\_POINTER, 0001

COMP HOME\_POINTER, 0040

JUMPZ HOMERESET

RETEN

HOMERESET LOAD HOME\_FLAG, 0000

LOAD HOME\_POINTER, 0031

JUMP PRENEWLINE

PRETENTHOUSAND LOAD TEMP, 0000

COMP CHAR\_COUNT, 0000

JUMPNZ TENTHOUSAND

LOAD DATA, asc\_NULL

OUTPUT DATA, 0000

RETEN

TENTHOUSAND SUB CHAR\_COUNT, 2710

JUMPC PRETHOUSAND

ADD TEMP, 0001

JUMP TENTHOUSAND

PRETHOUSAND STORE TEMP, TENTHOUSANDS

LOAD TEMP, 0000

ADD CHAR\_COUNT, 2710

COMP CHAR\_COUNT, 2710

JUMPZ CHARACTERS

JUMP THOUSAND

THOUSAND SUB CHAR\_COUNT, 03E8

JUMPC PREHUNDRED

ADD TEMP, 0001

JUMP THOUSAND

PREHUNDRED STORE TEMP, THOUSANDS

LOAD TEMP, 0000

ADD CHAR\_COUNT, 03E8

COMP CHAR\_COUNT, 03E8

JUMPZ CHARACTERS

JUMP HUNDRED

HUNDRED SUB CHAR\_COUNT, 0064

JUMPC PRETEN

ADD TEMP, 0001

JUMP HUNDRED

PRETEN STORE TEMP, HUNDREDS

LOAD TEMP, 0000

ADD CHAR\_COUNT, 0064

COMP CHAR\_COUNT, 0064

JUMPZ CHARACTERS

JUMP TEN

TEN SUB CHAR\_COUNT, 000A

JUMPC PREONE

ADD TEMP, 0001

JUMP TEN

PREONE STORE TEMP, TENS

LOAD TEMP, 0000

ADD CHAR\_COUNT, 000A

COMP CHAR\_COUNT, 000A

JUMPZ CHARACTERS

JUMP ONE\_1

ONE\_1 STORE CHAR\_COUNT, ONES

JUMP CHARACTERS

CHARACTERS LOAD CHAR\_FLAG, FFFF

COMP CHAR\_POINTER, 0108

JUMPZ CHARRESET

FETCH TEMP, CHAR\_POINTER

ADD CHAR\_POINTER, 0001

COMP TEMP, 0000

JUMPZ LOADZERO

COMP TEMP, 0001

JUMPZ LOADONE

COMP TEMP, 0002

JUMPZ LOADTWO

COMP TEMP, 0003

JUMPZ LOADTHREE

COMP TEMP, 0004

JUMPZ LOADFOUR

COMP TEMP, 0005

JUMPZ LOADFIVE

COMP TEMP, 0006

JUMPZ LOADSIX

COMP TEMP, 0007

JUMPZ LOADSEVEN

COMP TEMP, 0008

JUMPZ LOADEIGHT

COMP TEMP, 0009

JUMPZ LOADNINE

OUTPUT TEMP, 0000

RETEN

LOADZERO LOAD DATA, asc\_0

OUTPUT DATA, 0000

RETEN

LOADONE LOAD DATA, asc\_1

OUTPUT DATA, 0000

RETEN

LOADTWO LOAD DATA, asc\_2

OUTPUT DATA, 0000

RETEN

LOADTHREE LOAD DATA, asc\_3

OUTPUT DATA, 0000

RETEN

LOADFOUR LOAD DATA, asc\_4

OUTPUT DATA, 0000

RETEN

LOADFIVE LOAD DATA, asc\_5

OUTPUT DATA, 0000

RETEN

LOADSIX LOAD DATA, asc\_6

OUTPUT DATA, 0000

RETEN

LOADSEVEN LOAD DATA, asc\_7

OUTPUT DATA, 0000

RETEN

LOADEIGHT LOAD DATA, asc\_8

OUTPUT DATA, 0000

RETEN

LOADNINE LOAD DATA, asc\_9

OUTPUT DATA, 0000

RETEN

CHARRESET LOAD CHAR\_FLAG, 0000

LOAD CHAR\_POINTER, 0101

FETCH CHAR\_COUNT, 0200

LOAD DATA, asc\_NULL

OUTPUT DATA, 0000

JUMP PRENEWLINE

; ISR vectored through 0FFE

ADDRESS 0FFE

ENDIT JUMP ISR

END

**X. All Verilog Files**

The following are all the Verilog files used for this project.