ENGINEERING MEMORANDUM

To: MICROBRIDGE/MICROGATE DEVELOPMENT TEAM

From: MICHAEL A. MORRIS, VICE-PRESIDENT

SUBJECT: DESCRIPTION OF SSP UART FOR 1700-0403 MICRO-BRIDGE OPTION CARD

Date: 11/2/2013

CC: MicroBridge Engineering Notebook

I have had a discussion with David Walker and Chris Lott regarding the capabilities required of an FPGA-based UART for the 1700-0403 MicroBridge Option Card – Programmable Serial Interface. The 1700-0403 MicroBridge Option Card includes an FPGA which interfaces an expansion connector. The expansion connector provides an SPI-like interface which implements a high-speed serial interface with programmable lengths from 4 bits to 16 bits. In this application, the interface will be configured to operate with a 16-bit frame length.

Typically, SPI uses an external parallel address bus to select internal register in the slave device, which is in this case will located within the 1700-0403 FPGA. Instead of using external address lines to select internal registers in an SPI slave device, I have opted to use an embedded bit field in the serial data stream for this interface. To maintain some semblance of compatibility with the SPI interface used in the 1700-0402A MicroBridge Option Card – Modbus Plus, I have opted to use the same 16-bit frame size and use the most significant 4 bits as an address field. Furthermore, I have reserved the address space used for the Modbus Plus Option Card. Thus, it will be possible to have both the Programmable Serial Interface and the Modbus Plus Option Cards in the system simultaneously.

With the 1700-0380 and 1700-0381 TSXCUSBMBP product, the SPI interface to the FPGA provides direct loopback of these first four bits so that the receive data, MISO data, is tagged with the register select bits from the transmit data, MOSI data, in the same SPI frame. That is, there is no latency in the interface. The same mechanism will be applied in the development of the SPI interface for the 1700-0403.

Since it is desirable to have an FPGA-based UART that provides at least the same level of performance as provided by the internal 16C550-compatible UARTs in the LPC214x ARM microcomputers, it is imperative that the SPI connection to the UART be as efficient as possible. With a 16C550-compatible UART, the parallel bus interface and the Interrupt ID Register (IIR) and the Line Status Register (LSR) are used to provide a fast mechanism for identifying the interrupt sources and status of the transmitter and receiver modules.

In the case of the 1700-0403, the serial expansion connector interface, although it can operate at high serial rates, becomes a bottleneck if register polling is required in order to retrieve data from the Receive Holding Register (RHR). Therefore, I have decided to implement a simplified UART instead of a 16C550-compatible UART. The simplifications are designed to support only the functionality required, and to provide an efficient mechanism for reading and writing to receiver and transmitter functions while simultaneously returning critical status information back to the application.

The following tables define the I/O registers of the SSP UART:

Table 1 SSP UART Registers.

Reg	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDR	1	1	0	0	RFC	TFC	-	TFW	THR[7:0]							
RDR	1	1	0	0	TRDY	RRDY	RTO	RERR	RHR[7:0]							
USR	1	1	0	1	MD[1:0]		RTSi	CTSi	RS [1:0]		TS [1:0]		iRTO	iRDA	iTHE	iTFE
UCR	1	1	1	0	MD[1:0]		RTSo	IE	FMT[3:0]				BR[3:0]			
SPR	1	1	1	1	Scratch Pad Register (Refer to Appendix A for complete description)											

Table 2 SSP UART Register Field Definitions.

Field	Description	Comments
THR[7:0]	Transmit Holding Register	Input to Transmit FIFO
TFW	Transmit FIFO Write	Set to write into THR
TFC	Transmit FIFO Clear	Set to clear/reset transmit FIFO
RFC	Receive FIFO Clear	Set to clear/reset receive FIFO
RHR[7:0]	Receive Holding Register	Output of Receive FIFO
TRDY	Transmit Ready	0 – Tx FIFO Full; 1 – Tx FIFO not Full
RRDY	Receive Data Ready	Set if receive FIFO holds any data
RTO	Receive Time Out	When set, 3 character time out occurred
RERR	Receive Error	When set, indicates RHR error detected
iTFE	Interrupt – Transmit FIFO Empty	Set on Transmit FIFO Empty
iTHE	Interrupt – Transmit FIFO < TFThr	Set when Transmit FIFO < TFThr
iRDA	Interrupt – Receive FIFO ≥ RFThr	Set when Receive FIFO ≥ RFThr
iRTO	Interrupt – Receive Timeout	Set when no character received for 3 character periods
TS[1:0]	Transmit Status	0 – Empty; 1 < TFThr; 2 ≥ TFThr; 3 ≥ Full
RS[1:0]	Receive Status	0 – Empty; 1 < RFThr; 2 ≥ RFThr; 3 ≥ Full
CTSi	Clear To Send input	1 – CTS input asserted
RTSi	Request To Send input	1 – RTS output asserted
MD[1:0]	Operating Mode	0 – RS-232 without flow control 1 – RS-232 with flow control 2 – RS-485 without transmit loopback 3 – RS-485 with transmit loopback
BR[3:0]	Baud Rate Select	Use Table 3 to set desired baud rate
FMT[3:0]	Frame Format Select	Use Table 4 to set desired frame format
IE	Interrupt Enable	0 – disable interrupts; 1 – enable interrupts
RTSo	RTS output	1 – asserts RTS in non-handshake modes

Table 3 SSP UART Baud Rate Select Codes – 48 MHz Reference.

BR[3:0]	Desired Rate (bps)	Prescaler	Divider	Actual Rate (bps)	% error
0	3,000,000	1	1	3,000,000	0.00
1	1,500,000	1	2	1,500,000	0.00
2	500,000	1	6	500,000	0.00
3	187,500	1	16	187,500	0.00
4	230,400	13	1	230,769.2	0.16
5	115,200	13	2	115,384.6	0.16
6	76,800	13	3	76,923.0	0.16
7	57,600	13	4	57,692.3	0.16
8	38,400	13	6	38,461.5	0.16
9	28,800	13	8	28,846.2	0.16
10	19,200	13	12	19,230.8	0.16
11	14,400	13	16	14,423.1	0.16
12	9,600	13	24	9,615.4	0.16
13	4,800	13	48	4,807.7	0.16
14	2,400	13	96	2,403.8	0.16
15	1,200	13	192	1,201.9	0.16

Table 4 SSP UART Serial Frame Formats.

FMT[3:0]	Format	Description
0	-	Reserved; defaults to 8N1
1	8N1	10-bit frame: 8 data bits, no parity, 1 stop bit
2	8O1	11-bit frame: 8 data bits, odd parity, 1 stop bit
3	8E1	11-bit frame: 8 data bits, even parity, 1 stop bit
4	8S1	11-bit frame: 8 data bits, space (0) parity, 1 stop bit
5	8M1	11-bit frame: 8 data bits, mark (1) parity, 1 stop bit
6	-	Reserved; defaults to 8N1
7	8N2	11-bit frame: 8 data bits, no parity, 2 stop bits
8	8O2	12-bit frame: 8 data bits, odd parity, 2 stop bits
9	8E2	12-bit frame: 8 data bits, odd parity, 2 stop bits
10	8S2	12-bit frame: 8 data bits, space (0) parity, 2 stop bits
11	8M2	12-bit frame: 8 data bits, mark (1) parity, 2 stop bits
12	7O1	10-bit frame: 7 data bits, odd parity, 1 stop bit
13	7E1	10-bit frame: 7 data bits, even parity, 1 stop bit
14	7O2	11-bit frame: 7 data bits, odd parity, 2 stop bits
15	7E2	11-bit frame: 7 data bits, even parity, 2 stop bits

As can be seen from the Table 1, the first address (0xC) provides simultaneous access to the Transmit Data Register (TDR) and the Receive Data Register (RDR). In addition, the registers provide functions like those found in the FIFO Control Register (FCR) and the Line Status Register (LSR) of a 16C550-compatible UART. The second address (0xD) provides access to the UART Status Register (USR). The third address (0xE) provides access to the UART Control Register (UCR). The fourth address (0xF) provides access to the Scratch Pad Register (SPR). The USR and UCR provide functionality similar to that found in the Interrupt Enable Register (IER), Divisor Latch (DL), Line Control Register (LCR) of a 16C550-compatible UART. The SPR is provided for test purposes, and will return the value last written to it. Separate addresses have been provided for the USR and the UCR so that a shadow register for the UCR is not required in the software. The USR

can be read by simply writing any value, and the value returned will be as defined in Table 1.

My primary objective for the SSP UART is to provide an optimized interface for the ARM to use to write/read data to/from the FPGA. As a consequence, the data registers, TDR and RDR, need to provide control and status functions as well as access to the THR and RHR FIFOs. Similarly, the UCR and the USR should provide functional equivalents to the 16C550 Interrupt Enable Register (IER), Interrupt ID Register (IIR), Line Control Register (LCR), Modem Control Register (MCR), Line Status Register (LSR), and Modem Control Register (MCR).

After considering the status information provided by the IIR and LSR of a 16C550 UART, I have opted to provide only a few status bits in the RDR, and a few control bits in the TDR. In order to support polled access to the RHR, the TFW bit must be set in order to write the payload data into the transmit FIFO, the Transmit Holding Register. Two other TDR control bits can be used to reset the FIFOs independently.

In the RDR, the TRDY indicates that there is room in the transmit FIFO, i.e. the THR. If the state of the transmit state machine is required, then the USR must be examined. The state of the TRDY bit in the RDR reflects the current state of the THR. The software is responsible for not writing more data to the THR than it can hold.

RRDY is asserted when the receive FIFO has any data. It will remain asserted until the FIFO is empty. RTO indicates that the receive FIFO has not been written to in 3 character periods. When RTO asserts due to a time out, RRDY will also be asserted and will remain asserted until the FIFO is empty. The RRDY flag is not a forward looking status flag, and only reflects the present state of the receive FIFO. Thus, if reading the receive FIFO in a loop, the RRDY flag indicates whether the RHR data read is valid. If RRDY is not set, the data read from the RHR is not valid. (*This means that when reading the RDR and RRDY is being polled, there will always be one read more than required to empty the FIFO.*)

The RERR flag indicates that a receive error occurred. Individual bits for the various receive errors is not required. All receive errors, whether parity errors, framing errors, break, and FIFO overrun errors, must be handled as critical errors. All of the protocols which we implement (and expect to support), use additional error detection measures. Therefore, any receive error is sufficient to invalidate a packet. For example, Profibus which uses an even parity character format, the occurrence of any receive error is sufficient to invalidate the entire packet regardless of the Frame Check Sequence (FCS) results. However, the receive FIFO does track the receive state of each character received instead of for the entire FIFO as in a standard UART, so it is not necessary to reset/clear the receive FIFO when RERR is asserted.

The UCR controls the baud rate, format, interrupt enable, and operating mode of the SSP UART. A four bit field sets the desired baud rate. Presently, the operating frequency of the FPGA and the SSP UART is 48 MHz, which will allow the implementation of Profibus communication rates as well as standard communications rates.

Table 3 defines the sixteen (16) supported baud rates of the SSP UART. Support is provided for the most commonly used baud rates from 1200 baud to 230.4 kbaud. Further, all of the standard Profibus baud rates are supported. For standard baud rates there will be a 0.16% error, but the high-speed Profibus rates will be exact. For most applications, the baud rates required are generally a simple subset of the baud rates that can be programmed. To simplify the implementation, the baud rate is not infinitely programmable in the SSP UART as it is for standard UART.

The next four bits, FMT[3:0], in the UCR set the format of the character frame. Generally, the data field is 8 bits in length. 7-bit data is supported but only with parity included which basically makes it equivalent to 8-bit data. The format control field allows the selection of a large number of data frame formats including formats with two stop bits. For the 7-bit formats, the most significant is cleared by the receiver so that the application software does not need to explicitly mask off the parity bit. Table 4 defines the frame formats supported by the SSP UART; unused entries in the table default to the 8N1 format.

The IE bit enables the generation of an external interrupt when one of the four interrupt flags from the USR is asserted. For the present implementation, the enable interrupt request will be routed to the nINT2 signal on the MicroBridge Expansion Connector. The four interrupt flags will be simply ORed together to form a level sensitive interrupt request. At the present time, no other interrupt sources will be supported. (*RERR conditions will cause a write into the receive FIFO which will subsequently generate RRDY or RTO. Thus, RERR will generate an interrupt indirectly.*)

Unlike a standard 16C550-compatible UART, no interrupts on the change of state of the CTSi signal will be generated. The SSP UART has a hardware handshaking mode that utilizes the CTSi input signal to control the transmitter automatically. Thus, there is no need to use software in an interrupt service routine to implement hardware handshaking. Therefore, there is no need to generate an interrupt on the basis of a change of state of the CTSi signal.

The SSP UART supports four operating modes which automate the use of the serial port. Two modes support 4-wire RS-232 communications, and two modes support RS-485 communications. The operating modes of the SSP UART are defined in Table 2.

Mode 0 (RS-232 without flow control) supports 4-wire RS-232 communications, and requires that the RTS signal level be set using the RTSo bit in the UCR. When RTSo is set in Mode 0, RTS is asserted. In this mode, the transmitter is enabled to transmit whenever there is data in the transmit FIFO.

Mode 1 (RS-232 with flow control) supports 4-wire RS-232 communications but asserts control of the RTS signal automatically. RTS is asserted whenever the receive FIFO is able to accept more data. In Mode 1, the state of the CTSi signal controls the transmitter. While CTSi is not asserted, the transmitter will not transmit any data. If CTSi de-asserts while a character is being transmitted, the transmission of the character will be completed and then the transmitter will wait until the CTS signal is asserted again by the far end re-

ceiver before sending any more data. In Mode 1, the transmitter samples the CTSi signal during the last stop bit. CTSi must be deasserted at this point to prevent the transmitter from sending the next character in the transmit FIFO.

Mode 2 (RS-485 without loopback) represents a half-duplex mode where the receiver's serial input is held in the mark state while the transmitter is sending data. That is, the data from the line side of the RS-485 transceiver is not being looped back into the SSP UART's receiver. In this mode, the transmitter asserts the RS-485 transceiver's DE (Drive Enable) pin when there is data in the transmit FIFO. In this mode, DE is asserted one bit time before the start bit of the first character and deasserted one bit time after the last stop bit of the last character. If the transmit FIFO is loaded before the expiration of the delay bit, then the DE signal will remain asserted and the transmitter will continue transmitting RS-485 data.

Mode 3 (RS-485 with loopback) is the same as Mode 2 except that the receiver's input is allowed to track to RS-485 line output. In this way the software can check for collisions on the line. In all other respects, Mode 3 is the same as Mode 2.

In the RS-232 Modes, RTSi is the state of the RTSo bit. In the RS-485 modes, RTSi is the state RS-485 transceiver's DE signal. Only in Mode 1 does CTSi reflect the state of the external CTS signal; in all other modes, the CTSi bit is read as a logic 1 which reflects the fact that the transmitter is always enabled to transmit. Only in Mode 1 is the transmitter prevented from transmitting the contents for the transmit FIFO on the basis of the state of the CTSi signal. In modes 0, 2, or 3, it is the responsibility of the programmer to insure that the line is idle before filling the transmit FIFO, and to fill (and keep filled,) the transmit FIFO to insure that the line is not released prematurely while attempting to transmit a packet.

The TS[1:0] and RS[1:0] fields of the USR indicate the status of the transmitter and receiver state machines, respectively. The definition of RS in Table 2 provides a good description of the receiver status. It effectively reflects the state of the receive FIFO and of the underlying receive state machine. Similarly, TS is the state of the transmit FIFO and its underlying transmit state machine. TS and RS will be zero only when the respective state machines are idle and the FIFOs empty.

In the USR there are four interrupt flags: iRTO, iRDA, iTHE, and iTFE. These interrupt flags are logically ORed to form the external interrupt request which is enabled by the UCR IE bit. No other interrupts are provided, but these four interrupt sources effectively duplicate the available interrupts for a standard UART and add an additional transmit interrupt source, iTHE, which is not normally available.

The iRDA bit is set on the receive FIFO becoming half full. It is reset by reading the USR, or by reading the RDR and bringing the receive FIFO below the half full threshold level. The iRTO flag is set whenever the last character received is not followed by another in less than 3 character periods. It is cleared by a read of the USR. The receive timeout timer is activated by a write into the receive FIFO, and restarted with every write into that FIFO. The receive timeout timer is stopped either by a timeout, in which case the RTO

status bit and iRTO flag are set, or by reading the receive FIFO. Thus, an iRTO interrupt should not occur unless there are characters in the receive FIFO less than the threshold, and no read of the receive FIFO was performed before the receive timeout timer expired.

The iTHE flag is set by the transmit FIFO becoming less than half full. If the programmer never fills the transmit FIFO above the half full mark, then this interrupt flag will not become set. The iTHE flag will be cleared by reading the USR or by filling the transmit FIFO above the half full threshold. The iTFE flag is set by the transmit FIFO becoming empty. The iTFE flag is cleared by reading the USR or by writing to the transmit FIFO. A single character write to the transmit FIFO will generate an asserted iTFE each time that character is read from the FIFO and loaded into the transmit shift register.

When using interrupts, the programmer should read the USR once and service all of the interrupts indicated before returning. The value read from the USR should be preserved in a working register/variable because the interrupt flags will all be cleared at the completion of the USR read operation. It is possible for some of the interrupt flags to become set while in the interrupt service routine after the USR has been read and reset. I recommend returning from the ISR after servicing the interrupts that were set when the routine first read the USR. The level activated nature of the interrupt request should result in the interrupt routine being immediately re-entered.

The interrupt request is latched and held until the USR is read. The flags should be checked by the ISR to determine the interrupt source. The interrupt flags will assert and deassert as conditions dictate, but the interrupt request will remain asserted until the USR is read. Thus, if a condition occurs in the UART during operation that asserts one of the interrupt flags, then the interrupt request will be asserted. Subsequently, if the condition which caused the interrupt flag to assert goes away, then the interrupt flag will deassert without deasserting the interrupt request. Clearing the Interrupt Enable (IE) bit in the UCR will not deassert the interrupt request either; only reading the USR or reset will deassert the interrupt request once it has been asserted when interrupts are enabled. This was done to prevent the automatic deassertion of the interrupt flags from generating spurious interrupts to the Vectored Interrupt Controller (VIC) of the ARM. (*A spurious interrupt service routine will still be required to handle conditions which occur internal to the ARM when interrupts are being enabled and disabled in the ARM.*)

The present implementation of the SSP UART attempts to utilize the available resources for the 1700-0403 FPGA as much as possible. The other resources on the 1700-0403 card are not being utilized for the first application. Thus, those features/capabilities are loopbacked, or held in a safe, static condition, until they may be needed in the future. As a result, the receive FIFO is implemented as a 128 character FIFO, and the transmit FIFO is implemented as a 128 character FIFO. The half full threshold for the iRDA flag is 64 characters. The half empty threshold for the iTHE is 63 characters, so that there are 65 available locations in the transmit FIFO.

Appendix A – Scratch Pad Register Extensions

The implementation of the Scratch Pad Register (SPR) has been extended from the definition provided in Table 1. Several additional status registers have been added to the UART using the SPR as a window into these additional registers. The following table, Table 5, shows the additional registers added in the latest revision of the SSP UART specification.

Table 5 SPR Status Registers.

Reg	Mode	Reg Sel	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPR	R/W	0	1	1	1	1	0	0	0	SPR[8:0]								
REV	RO	1	1	1	1	1	0	0	1	0	{MajRev[3:0], MinRev[3:0]}							
LEN	RO	2	1	1	1	1	0	1	0	0	{RFLen[3:0], TFLen[3:0]}							
RTFThr	R/W	3	1	1	1	1	0	1	1	WE	{RFThr[3:0], TFThr[3:0]}							
THR	RO	4	1	1	1	1	1	0	0	0	THR[7:0]							
RHR	RO	5	1	1	1	1	1	0	1	RHR[8:0]								
TFCnt	RO	6	1	1	1	1	0	TFCnt[(TFLen+4):0]										
RFCnt	RO	7	1	1	1	1	0	RFCnt[(RFLen+4):0]										

The RegSel field of the SPR register, SPR[11:9], is used to select the auxiliary control/status register. The SPR write operation will set the address of the auxiliary status register, and the subsequent access will return the contents of the addressed auxiliary control/status register. Since the SSP peripheral returns all sixteen bits to the user, the auxiliary control/status register being accessed can be determined from the data read from the SSP peripheral with the exception of the two FIFO Count registers. (*Note that unused bits in the return data are read a logic 0.*)

The SPR register is still a 12-bit R/W register except that some of its bits are used to subaddress the auxiliary control/status registers. Thus, the first 512 values of the SPR retain the original scratch pad feature of the SPR. The software can use this feature to perform a quick check on the SSP interface without affecting other UART registers and internal structures.

The REV register (SPR[11:9] = 0x1) returns the current revision of the FPGA design in a Major.Minor format as two 4-bit fields as indicated above.

The LEN register (SPR[11:9] = 0x2) returns the parameterization settings for the Tx and Rx FIFOs. That is, each 4-bit field indicates the excess size of the FIFO with respect to a standard 16 character FIFO. The values given for each FIFO are expressed as excess powers of two from the base of 4, which is equivalent to 2^4 = 16. Thus, if the field reads 0, then the FIFO depth is (1 << (0 + 4)) = 16. If the value read is 3, then the FIFO depth is (1 << (3 + 4)) = 128. The present implementation of the UART in the XC2S30-5VQ100I FPGA on the 1700-0403 board uses Block RAMs for the FIFOs, and they provide 1024 words of depth. Future implementations of the 1700-0403 FPGA may use smaller Block RAM FIFOs or FIFOs which use the distributed RAM of the FPGA. Therefore, it is important to set up the software driver by reading this register.

The RTFThr register (SPR[11:9] = 0x3) is a read/write auxiliary control/status register that controls the Rx/Tx FIFO threshold settings. The iRDA and the iTHE interrupt flags in the USR are set according to the settings of the Receive FIFO Threshold (RFThr) and Transmit FIFO Threshold (TFThr) settings. The RFThr/TFThr select at which sixteenth of the RFCnt/TFCnt field it is desired to trigger the respective interrupts. The default value of these fields is 0x8, which selects the half full point in the FIFO as the trigger. If a value of 0 is written into these control fields, then both thresholds will be effectively set for 1 character and the UART will operate very much like a standard 16C550 UART. When set for 0xF, the trigger points move to the 15/16 full point, or 960 characters for the current 1024-character implementation of the FIFOs. At a setting of 1, the trigger point is 1/16 full or 64 characters. (*Note: SPR[8] must be a logic 1 in order to modify the RTFThr register.*)

The output of the Tx FIFO (THR, SPR[11:9] = 0x4) and the output of the Rx FIFO (RHR, SPR[11:9] = 0x5), can be examined using the SPR auxiliary register window. Accessing these two registers in this manner allows the value in the FIFO to be checked/examined without advancing the FIFOs, i.e. reading the FIFOs.

The last two auxiliary status registers, TFCnt (SPR[11:9] = 0x6) and RFCnt (SPR[11:9] = 0x7), provide access to the Tx FIFO and Rx FIFO word count registers, respectively. The USR's interrupt flag bits and the transmit/receive state machine status bits provide sufficient status information regarding the state of the transmitter and receiver. However, sometimes it is useful to know how many words to read from the Rx FIFO without polling the RRDY bit. Thus, these two registers may be most useful from the ISR if the SPR is set to point to one or the other. The extra dummy write cycle needed to set up the address can be avoided, and each subsequent write to the SPR will return a new value of either the TFCnt or the RFCnt.