Model LR-232-x-x Series

TECHNICAL REFERENCE MANUAL

February 1, 2008

TEL 1-401-270-4445
FAX: 1-401-270-5221

WEB: www.link-research.com

Document Number: 4-903-49

Table of Contents

1. General

This technical reference covers the following product part numbers, collectively refered to as the “Serial Port daughtercard“:

LR-232-2	(2-CHANNELS, RS-232-C INTERFACES)
LR-232-2-S	(2-CHANNELS, RS-232-C INTERFACES, STACKABLE)
LR-232-4	(4-CHANNELS, RS-232-C INTERFACES)
LR-232-4-S	(4-CHANNELS, RS-232-C INTERFACES, STACKABLE)

The Serial Port Daughtercard utilizes the NXP (formally Philips) SC16C2550B (or compatible) dual UART integrated circuit. This IC contains a 16 byte transmit FIFO, and a 16 byte receive FIFO. The 4 channel version of the daughtercard uses 2 dual UART ICs, whereas the 2 channel version uses 1 IC. A datasheet for the SC16C2550B is available from the NXP website at www.nxp.com . The top speed of each UART port is 921.6 kbps.

The UART device(s) interface(s) to the DSK development system via the EMIF (External Memory Inter-Face) in a memory mapped fashion. The method of accessing the UARTs is slightly different, depending on the particular DSK development system that is being used. The following sections detail these differences.

Address Map

Tables 1 and 2 show the relevant information needed to access each UART channel when the daughtercard is attached to each of the supported DSKs. See the Link Research website at www.link-research.com for a current list of the supported DSK’s.

TABLE 1

DSK	BASE MEMORY ADDRESS				Page	CE2 (Chip Enable) control register value
	Channel 1	Channel 2	Channel 3	Channel 4		
5416	0x0800	0x8100	0x8200	0x8300	0x10	N/A
5510	0x500000	0x500200	0x500400	0x500600	N/A	N/A
6416	0xA0200000	0xA0200400	0xA0200800	0xA0200C00	N/A	0x5A35E823
6711	0xA0200000	0xA0200400	0xA0200800	0xA0200C00	N/A	0x5A35E823

6713	0xA0200000	0xA0200400	0xA0200800	0xA0200C00	N/A	0x03D00F21

1. UART Registers

Each serial channel is configured through a set of 12 memory mapped registers that reside inside the UART IC. Table 1 lists the base memory addresses for each UART channel and for each DSK. Table 2 lists the offset values which should be added to the base addresses given in Table 1, to access the individual UART registers.

TABLE 2				
		Offset from Base memory address for each DSK		
REGISTER	REGISTER NAME	6711, 6713, 6416	5510	5416
RBR	Read Buffer Register	0x00	0x00	0x00
THR	Transmit Holding Register	0x00	0x00	0x00
DLL	Divisor (low byte)	0x00	0x00	0x00
DLM	Divisor (high byte)	0x04	0x02	0x01
IER	Interrupt Enable Register	0x04	0x02	0x01
ISR	Interrupt Status Register	0x08	0x04	0x02
FCR	FIFO Control Register	0x08	0x04	0x02
LCR	Line Control Register	0x0C	0x06	0x03
MCR	Modem Control Register	0x10	0x08	0x04
LSR	Line Status Register	0x14	0x0A	0x05
MSR	Modem Status Register	0x18	0x0C	0x06
SPR	Scratch Pad Register	0x1C	0x0E	0x07

For example, to access the Line Control Register (LCR) for the Channel 3 UART on a 6416 DSK,
take the base address from Table 1 and add the register offset value from table 2 as follows:

0xA0200800 + 0x0C = 0xA020080C

Note: The absolute memory addresses of the UART registers for all 4 channels are provided in the demonstration programs. Simply locate the "include file" associated with the demonstration programs for the particular DSK you are using.

2. Supported Data Rates

The LR-232-x-x series of daughtercards support a maximum data rate (baud rate) of 921,600 bps (bits per second). By programming the DLL and DLM divisor registers in the UART, any data rate which is a sub-multiple of 921,600 can be obtained. The general formula relating the DLL/DLM register values to the resulting baud rate is:

Baud Rate = 921,600/(contents of DLM:DLL)

Where DLM:DLL is the 16 bit word obtained by concatenating the two 8 bit divisor registers DLM and DLL.

3. Interrupts

Because the Serial Port Daughtercard has been designed to operate with many TI and Spectrum Digital DSKs, and for future obsolescence-proof, the routing of the interrupt signals from the UART devices to the DSKs peripheral expansion connector has been designed with a jumper block . For example, some DSKs respond to an active-high interrupt signal, while others respond to an active low signal. Also, each DSK has a different number of external interrupt lines made available at the peripheral connector. Additionally, some users may be using some of the external interrupt lines for other peripheral devices. Therefore, the Serial Port Daughtercard has been designed with flexibility in mind to allow the following:

- Any of the interrupt signals generated by the serial channels can be jumpered to any available DSK interrupt line.
- Multiple interrupt signals from any number of serial channels can be combined and jumpered to drive one DSK interrupt line.
- Since interrupt signals generated by the UART are active high signals, inverters are provided to allow interfacing to DSKs with active-low external interrupt input lines.
-

Configuring hardware interrupts on the Serial Port Daughtercard requires that the user install 1 to 4 jumper wires using the following 3 steps:

1) Determine the number of serial channels and the number of interrupt lines desired.
2) Refer to Table 3 to relate the board E-point(s) to the desired external interrupt(s) for the particular DSK in use.
3) Refer to Table 4 to relate the board E-point(s) to the desired UART interrupt source configuration.

TABLE 3

		DSK				
Pin number on Peripheral connector	**E-Point On Board**	**5416**	**5510**	**6416**	**6711**	**6713**
47	E15	N/A	INT0#	N/A	N/A	N/A
48	E16	DC_INT1#	INT2#	EXT_INT5	EXT_INT5	EXT_INT5
53	E17	DC_INT0#	INT1#	EXT_INT4	EXT_INT4	EXT_INT4
60	E18	Provided for backward compatibility with the 5402 DSK				
67	E19	DC_INT2#	INT3#	EXT_INT6	EXT_INT6	EXT_INT6
68	E20	DC_INT3#	N/A	EXT_INT7	EXT_INT7	EXT_INT7

TABLE 4	
E-Point On Board	**Description**
E1	Channel 1, Active High
E2	Channel 1, Active Low
E3	Channel 2, Active High
E4	Channel 2, Active Low
E5	Channels 1 & 2, Active Low
E6	Channels 1 & 2, Active High
E7	Channels 1,2,3 & 4, Active Low
E8	Channels 3 & 4, Active Low
E9	Channels 1,2,3 & 4, Active High
E10	Channels 3 & 4, Active High
E11	Channel 4, Active High
E12	Channel 4, Active Low
E13	Channel 3, Active High
E14	Channel 3, Active Low

Here are some example configurations:

1) Using a 2 channel UART daughtercard, attached to a 6416 DSK, attach the interrupt signal from the channel 1 UART to external interrupt EXT_INT4, and the interrupt signal from the channel 2 UART to external interrupt EXT_INT5.

Solution: Jumper E1 to E17, and E3 to E16

2) Using a 4 channel UART daughtercard, attached to a 5510 DSK, combine the interrupt signals from channels 1 & 2, and route to external interrupt INT0#. Also, combine the interrupt signals from channels 3 & 4, and route to external interrupt INT3#.

Solution: Jumper E5 to E15, and E8 to E19

3) Using a 4 channel UART daughtercard, attached to a 6713 DSK, combine the interrupt signals from all 4 channels , and route to external interrupt EXT_INT7.

Solution: Jumper E9 to E20.

4. LEDs

A pair of LEDs, one green and one red, are provided for each serial channel. The Green LED is labeled “DATA OUT”, and indicates when data is being transmitted by the daughtercard. For DTE configurations, DATA OUT corresponds to the RS-232 “TD” signal which is pin 3 of the male DB-9 connector. For DCE configurations, DATA OUT corresponds to the RS-232 “RD” signal which is pin 2 of the female DB-9 connector. Likewise, the Red LED is labeled “DATA IN”, and indicates when data is being received by the daughtercard. For DTE configurations, DATA IN corresponds to the RS-232 “RD” signal which is pin 2 of the male DB-9 connector. For DCE configurations, DATA IN corresponds to the RS-232 “TD” signal which is pin 3 of the female DB-9 connector.

5. Interface Configuration

For the RS-232 interface, each channel of the Serial Port Daughtercard can be configured as a DTE (Data Terminal Device, male DB-9 connector), or a DCE (Data Communication Device, female DB-9 connector). This configuration is specified at the time of purchase and is implemented by means of a single jumper at the factory. Table 5 shows this in detail.

TABLE 5		
Channel	Configured as DTE	Configured as DCE
1	JP4 not installed	JP4 installed
2	JP8 not installed	JP8 installed
3	JP6 not installed	JP6 installed
4	JP2 not installed	JP2 installed

6. Connector Pinouts

Table 7 shows the DB-9 connector pinouts for the model LR-232-x-x daughtercards.

Table 7

DTE Ports (Male Connector)			DCE Ports (Female Connector)			
Pin	Signal	Signal Direction	Pin	Signal	UART Signal Used	Signal Direction
1	DCD	in	1	DCD	DTR	out
2	RD	in	2	RD	TD	out
3	TD	out	3	TD	RD	in
4	DTR	out	4	DTR	DSR	in
5	Ground	—	5	Ground	Ground	—
6	DSR	in	6	DSR	DTR	out
7	RTS	out	7	RTS	CTS	in
8	CTS	in	8	CTS	RTS	out
9	RI	in	9	Not Used	Not Used	—

7. The Application/Demonstration Software Programs

Four Application/Demonstration programs are provided on the included CD for each of the supported DSK/EVM development systems. These programs are provided as complete Code Composer Studio projects.

All the Application/Demonstration programs utilize a set of 7 high level functions which relieve the user from having to deal with low level UART device accessing, handling of interrupts, and circular data buffering. The 7 high level functions are described in Table 9.

PROGRAM 1
The first program is called "**uart_loopback**". When compiled and loaded into the DSK and executed, it simply monitors all 4 UART channels, taking any incoming data and sending it back out on the same channel. The data rate for all four channels is set to 19.2 kbps with the "**initialize_uart**" function. The user may change the data rates of any channel to suit their needs.

PROGRAM 2
The second project, called "**uart_library**", takes the 7 high level functions and creates a library called **uartlib.lib**, to be used in any Code Composer Studio project.

PROGRAM 3
The third project, called "**library_test**", is simply a test program to check the library created in the previous project. Basically, this project consists of the "**uart_loopback**" project with the 7 functions removed from the source code, and included as a library via "**uartlib.lib**".

PROGRAM 4
The fourth program on the CD is named **"pc_demo"** and is a simple command/response program using the PC running HyperTerminal. To run this program, the user should do the following:

1. Connect a "TO DTE" channel of the Serial Port Daughtercard to an unused serial port on the PC using the supplied DB-9 TO DB-9 cable. If a "TO DCE" channel is used then a Null Modem cable or adapter must be used.

2. Run Windows HyperTerminal and configure it to 19200 bps, 8 data bits, no parity, and 1 stop bit.

3 In Code Composer Studio, load the pc_demo project for your particular DSK from the CD.
  Modify the line "CHANNEL_TO_USE" in the pc_demo.c source file to match the UART channel number in use.
4 Compile and download the program to the DSK.

5 Run the DSP program by clicking on the RUN icon in Code Composer Studio.

At this point the DSK should send the following "welcome" message to the HyperTerminal screen (shown below). Follow the on-screen directions to verify that serial communications between the PC and the DSK is functioning properly.

```
Welcome to the Link Research PC/UART demonstration program

   This demo program verifies the sending and receiving
   of data to and from the DSK. Type a number from 1-3,
   and the DSK will respond with the appropriate message

       ---Menu---
  1:   Send FOX message
  2:   Send string of digits
  3:   Send Character set
```

8. Uart API Function Description

TABLE 9	
Function	Description
void initialize_UART (int, unsigned char);	Initializes the UART, first argument specifies channel number (1,2,3,or 4), second argument specifies the data rate. See the “mainxxxx.h” file for the definitions for the data rate constants.
int UART_send_byte (int, unsigned char);	Sends a single 8 bit character to a UART channel. First argument specifies the channel, second argument specifies the character.
int UART_recv_byte (int, unsigned char*);	Receives a single 8 bit character from a UART channel. First argument specifies the channel, second argument specifies a pointer to the character.
int UART_send_block (int, unsigned char*,unsigned int);	Sends a block of 8 bit characters to a UART channel. First argument specifies the channel, second argument specifies a pointer to the character array, third channel specifies the number of characters to send.
int UART_recv_block (int, unsigned char*,unsigned int);	Reads a block of 8 bit characters from a UART channel. First argument specifies the channel, second argument specifies a pointer to the character array, third channel specifies the number of characters to read.
unsigned int UART_recv_count(int);	Returns the number of characters available to be read. Argument specifies the UART channel to query.

unsigned int UART_xmit_count(int);	Returns the number of characters in the transmit output buffer. Can be used to determine if there is enough room in the transmit buffer for more data. Argument specifies the UART channel to query.