

# 11 ASYNCHRONOUS SERIAL PORTS

## 11.1 General

The ETRAX 100LX contains four complete asynchronous serial receivers/transmitters with full buffering and parity control. Each asynchronous serial port has one handshake signal in each direction.

The receivers/transmitters support baud rates from 48 up to 1,843,200 baud, plus a non-standard baud rate of 6,250,000 baud.

## 11.2 Connection to Input/Output Pins

The pins of Asynchronous Serial Port p0 are available in all configurations. The other asynchronous serial port pins are multiplexed as shown in table 11-1 below:

Asynchronous Serial Port Pins	Multiplexed Pins
Asynchronous Serial Port p1	Synchronous Serial Port p1 USB Port p1 General I/O Port
Asynchronous Serial Port p2	SCSI-8 Port p0 SCSI-W Port ATA Port General I/O Port
Asynchronous Serial Port p3	Synchronous Serial Port p3 SCSI-8 Port p1 ATA Port General I/O Port

*Table 11-1 Asynchronous serial port pin multiplexing*

Pin configuration is done in registers R\_GEN\_CONFIG and R\_GEN\_CONFIG\_II.

In order to set the correct default value of Asynchronous Serial ports p2 and p3, the **hcfg** pin should be tied to 0:

if (hcfg = 0), then {s0sel = 1, s1sel = 1}

For more information see section 19.11 *I/O Pin Default Values*.

### 11.3 Asynchronous Serial Port Registers

Table 11-2 below provides a summary of the asynchronous serial port registers.

For more detailed information see chapter 18.8 *Serial Port Registers*.

Register	Function
R_SERIAL0_CTRL R_SERIAL1_CTRL R_SERIAL2_CTRL R_SERIAL3_CTRL	A 32-bit wide write only register for baud rate selection, serial transmitter/receiver control, and serial data out.
R_SERIAL0_BAUD R_SERIAL1_BAUD R_SERIAL2_BAUD R_SERIAL3_BAUD	A byte wide write only register for transmitter/receiver baud rate selection.
R_SERIAL0_REC_CTRL R_SERIAL1_REC_CTRL R_SERIAL2_REC_CTRL R_SERIAL3_REC_CTRL	A byte wide write only register for serial receiver control.
R_SERIAL0_TR_CTRL R_SERIAL1_TR_CTRL R_SERIAL2_TR_CTRL R_SERIAL3_TR_CTRL	A byte wide write only register for serial transmitter control.
R_SERIAL0_TR_DATA R_SERIAL1_TR_DATA R_SERIAL2_TR_DATA R_SERIAL3_TR_DATA	A byte wide write only register for serial data out.
R_SERIAL0_READ R_SERIAL1_READ R_SERIAL2_READ R_SERIAL3_READ	A byte wide read only register for serial transmitter/receiver status, and serial data in.
R_SERIAL0_STATUS R_SERIAL1_STATUS R_SERIAL2_STATUS R_SERIAL3_STATUS	A byte wide read only register for serial transmitter/receiver status.
R_SERIAL0_REC_DATA R_SERIAL1_REC_DATA R_SERIAL2_REC_DATA R_SERIAL3_REC_DATA	A byte wide read only register for data in from the serial receiver.
R_SERIAL0_XOFF R_SERIAL1_XOFF R_SERIAL2_XOFF R_SERIAL3_XOFF	A 32-bit wide write only register for serial transmitter control, and <b>xoff</b> handling.
R_ALT_SER_BAUDRATE	A 32-bit wide write only register for alternative baud rate selection.
R_SERIAL_PRESCALE	A 16-bit write only register for the divide factor for serial clock prescaling.
R_SER_PRESC_STATUS	A 16-bit read only register for the current count value of the serial divide factor.

Table 11-2 *Asynchronous serial port registers*

## 11.4 Operation Modes

The serial port operation modes are configured in R\_SERIAL $_X$ \_CTRL. The receiver and transmitter can also be configured separately by using the registers R\_SERIAL $_X$ \_REC\_CTRL and R\_SERIAL $_X$ \_TR\_CTRL respectively. The following modes can be configured:

- Receiver and transmitter baud rate
- Odd, even, fixed or no parity
- 7 or 8 data bits

The transmitter can also be configured to send 1 or 2 stop bits, and to handle  $\overline{\text{cts}}$  automatically or to ignore  $\overline{\text{cts}}$ . When automatic  $\overline{\text{cts}}$  handling is selected, a high on the  $\overline{\text{cts}}$  input will halt the transmitter after the ongoing byte, and keep it halted until  $\overline{\text{cts}}$  becomes low again.

The  $\overline{\text{rts}}$  output is also controlled by a bit in the R\_SERIAL $_X$ \_CTRL register.

In the R\_SERIAL $_X$ \_XOFF register, the serial port can be configured to stop transmission when the xoff character is received. The character code for xoff is also configurable.

Receiver and transmitter baud rate configuration can be done in either R\_SERIAL $_X$ \_REC\_CTRL or R\_SERIAL $_X$ \_BAUD.

Asynchronous Serial Port p0 can be used when bootstrapping the ETRAX 100LX. See chapter 6 *Bootstrap Methods*.

**Note 1:** In order to avoid unnecessary repetition of port numbers, the  $_X$  in the interrupt and register names stands for either 0\_, 1\_, 2\_ or 3\_, representing Asynchronous Serial Ports p0, p1, p2 and p3.

### 11.5 Baud Rate Selection

There are four different ways to set the baud rate, which is configured in R\_ALT\_SER\_BAUDRATE:

- The default setting for R\_ALT\_SER\_BAUDRATE is to choose one of the available predefined baud rates shown below in table 11-3. The predefined baud rate is set in R\_SERIAL<sub>x</sub>\_CTRL or R\_SERIAL<sub>x</sub>\_BAUD. The baud rate can be set individually for each port, and separately for input and output:

Predefined Baud rates		
300	9,600	230,400
600	19,200	460,800
1,200	38,400	921,600
2,400	57,600	1,843,200
4,800	115,200	6,250,000

Table 11-3 Predefined baud rates

- Use a scalable baud rate. The scalable baud rate can be set between 1,562,500 baud and 48 baud through a 16 bit divide factor loaded in R\_SERIAL\_PRESCALE. The resulting baud rate will be  $(3,125,000/\text{ser\_presc}$  baud), where **ser\_presc** is the divide factor in R\_SERIAL\_PRESCALE.
- Use an external baud rate clock which is enabled in R\_GEN\_CONFIG\_II. The clock must be less than 20 MHz. The baud rate achieved from the external clock is the external clock divided by eight, so the maximum baud rate is 2,500,000 baud. The General I/O Port **pb6** is assigned to receive the external baud rate clock.
- Use **timer0** as the baud rate generator. The frequency generated with **timer0**, is configured in the R\_TIMER\_CTRL register. For transmission the timer output clock is used directly resulting in a maximum baud rate of 12.5 MHz. For receiving the output is divided by eight resulting in a maximum baud rate of 1,562,500 baud.

For more detailed information see chapter 15 *Timers*.

## 11.6 CPU Controlled Operation

When the CPU controls serial port operation, the port can be polled or interrupt driven.

When the transmitter is ready to accept new data, the **tr\_ready** field in R\_SERIAL<sub>X</sub>\_READ and R\_SERIAL<sub>X</sub>\_STATUS is set to **ready**, and the **ser<sub>X</sub>\_ready** interrupt is generated. The **tr\_ready** bit and the **ser<sub>X</sub>\_ready** interrupt are cleared when data is written to the R\_SERIAL<sub>X</sub>\_TR\_DATA register or to the **data\_out** field of R\_SERIAL<sub>X</sub>\_CTRL.

When data is available from the serial receiver, the **data\_avail** field in R\_SERIAL<sub>X</sub>\_READ and R\_SERIAL<sub>X</sub>\_STATUS is set to **yes**, and the **ser<sub>X</sub>\_data** interrupt is generated. The **data\_avail** bit and the **ser<sub>X</sub>\_data** interrupt are cleared when data is read from the R\_SERIAL<sub>X</sub>\_REC\_DATA register or from the **data\_in** field of R\_SERIAL<sub>X</sub>\_READ.

If the serial receiver encounters a parity error, framing error or overrun error, the errors are indicated in the R\_SERIAL<sub>X</sub>\_READ and R\_SERIAL<sub>X</sub>\_STATUS registers. The error status is valid whenever the **data\_avail** bit is set, and is cleared when data is read from the R\_SERIAL<sub>X</sub>\_REC\_DATA register or from the **data\_in** field of R\_SERIAL<sub>X</sub>\_READ.

## 11.7 DMA Controlled Operation

In R\_GEN\_CONFIG, the asynchronous serial ports can be connected to the internal DMA channels as shown in the table below:

Async Serial Port	DMA Channel	
	In	Out
Async Serial Port p0	Channel 7	Channel 6
Async Serial Port p1	Channel 9	Channel 8
Async Serial Port p2	Channel 3	Channel 2
Async Serial Port p3	Channel 5	Channel 4

In a DMA controlled operation, received data is entered into the FIFO of the connected internal DMA channel. The data in the FIFO will only be written out when it reaches the configured FIFO trip point (i.e. 16 or 32 bytes, which is configured in R\_BUS\_CONFIG). Therefore, after the last received byte, there may be up to 31 bytes left in the FIFO. The remaining FIFO contents can be written out to the memory by issuing an **eop** (end of packet) command to DMA. This is done by writing to the R\_SET\_EOP register.

In addition, the **data\_avail** field in R\_SERIAL<sub>X</sub>\_READ and R\_SERIAL<sub>X</sub>\_STATUS is set when a data byte is received, and is cleared by a CPU read from the R\_SERIAL<sub>X</sub>\_REC\_DATA register or from the **data\_in** field of R\_SERIAL<sub>X</sub>\_READ. The same is also true for the **ser<sub>X</sub>\_data** interrupt. Therefore, the **data\_avail** field and/or the **ser<sub>X</sub>\_data** interrupt can be used to detect when new data has been entered into the FIFO.

To handle receive errors in a DMA controlled operation, there are two different modes:

- 1 If the **dma\_err** field in R\_SERIAL $x$ \_CTRL is set to **stop**, a receive error generates an **eop** to DMA and stops the receiver so the erroneous byte is not entered into the FIFO. The error condition is cleared by a CPU read from the R\_SERIAL $x$ \_REC\_DATA register, or from the **data\_in** field of R\_SERIAL $x$ \_READ.
- 2 If the **dma\_err** field is set to **ignore**, receive errors are ignored and the erroneous byte is entered in the FIFO.

During a DMA controlled transmission, DMA transfers may be temporarily stopped and CPU mode entered by disconnecting DMA in R\_GEN\_CONFIG. This can be used, for example, for inserting a CPU controlled xoff transmission into a DMA controlled data stream.

### 11.8 Asynchronous Serial Port Interrupts

Each asynchronous serial port has two interrupts, one for the receiver and one for the transmitter:

#### **ser $x$ \_ready**

This interrupt is set when the asynchronous serial port is ready to acquire new data for transmission. This interrupt is cleared when new data is written to the R\_SERIAL $x$ \_TR\_DATA register, or to the **data\_out** field of R\_SERIAL3\_CTRL. The **ser $x$ \_ready** interrupt should be masked when DMA is used for data transfers.

#### **ser $x$ \_data**

This interrupt is set when input data is available on the port. It is cleared when data is read from the R\_SERIAL $x$ \_REC\_DATA register, or from the **data\_in** field of R\_SERIAL $x$ \_READ. When DMA is used for the data transfer, this interrupt indicates that at least one byte was received since the interrupt was last cleared.

For more detailed information see chapter 17 *Interrupts*.