Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!


## Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

# MAX3108A

# SPI/I²C UART with 128-Word FIFOs in WLP

## General Description

The MAX3108A small form factor universal asynchronous receiver-transmitter (UART) with 128 words each of receive and transmit FIFOs is controlled through a serial I²C or SPI controller interface. Auto-sleep and shutdown modes help reduce power consumption during periods of inactivity. A low 500µA (max) supply current, 1µA shutdown current, and tiny 25-bump WLP (2.1mm x 2.1mm) package make the device ideal for low-power portable devices. The device operates from a 1.71V to 3.6V supply voltage.

Baud rates up to 24Mbps make the MAX3108A suitable for today's high data rate applications. A phase-locked loop (PLL), predivider, and fractional baud-rate generator allow high-resolution baud-rate programming and minimize the dependency of baud rate on reference clock frequency.

Four GPIOs can be used as inputs, outputs, or interrupt inputs. When configured as outputs, they can be programmed to be open-drain outputs and sink up to 20mA of current.

The device is ideal for portable and handheld devices, is available in a 25-bump (2.1mm x 2.1mm) 0.4mm pitch WLP package, and is specified over the -40°C to +85°C extended temperature range.

## Applications

Portable Communication Devices
Mobile Internet Devices
Low-Power Handheld Devices
Medical Systems
Point-of-Sale Systems

*IrDA is a service mark of Infrared Data Association Corporation.*

## Features

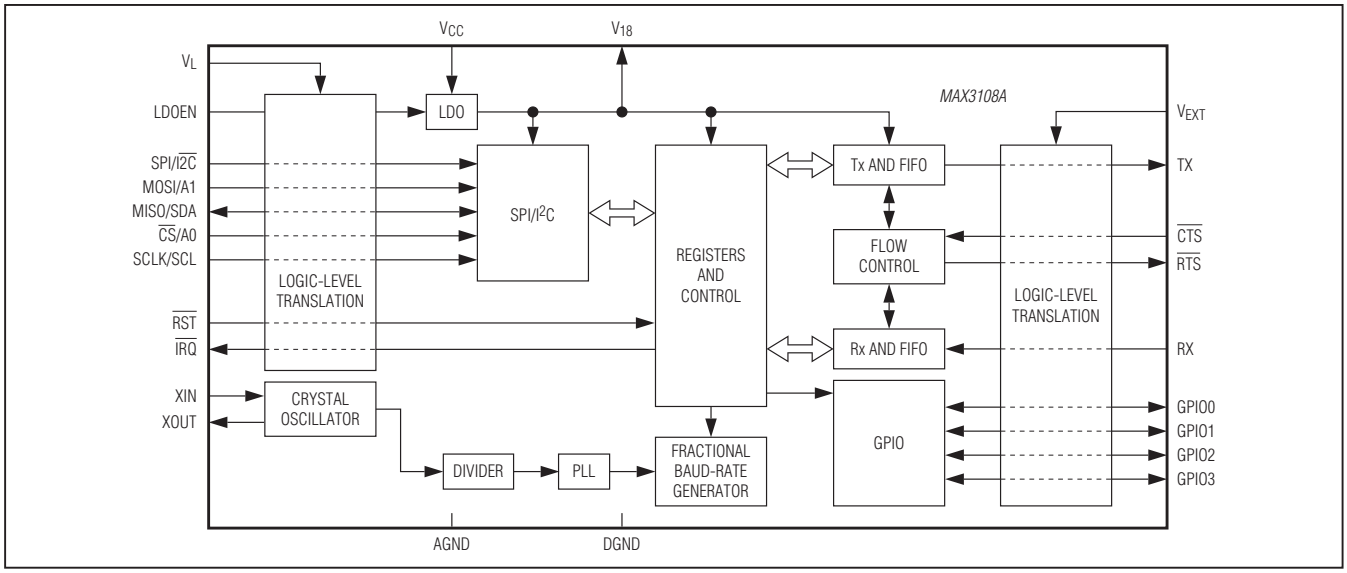
- **24Mbps (max) Baud Rate**
- **1µA Shutdown Current**
- **1.71V to 3.6V Supply Range**
- **High-Resolution Programmable Baud Rate**
- **SPI Up to 26MHz Clock Rate**
- **Fast Mode Plus I²C Up to 1MHz**
- **Automatic $\overline{RTS}$ and $\overline{CTS}$ Hardware Flow Control**
- **Automatic XON/XOFF Software Flow Control**
- **Special-Character Detection**
- **9-Bit Multidrop Mode Data Filtering**
- **SIR- and MIR-Compliant IrDA℠ Encoder/Decoder**
- **Flexible Logic Levels on the Controller and Transceiver Interfaces**
- **Four Flexible GPIOs**
- **Line Noise Indication**
- **Shutdown and Auto-Sleep Modes**
- **Integrated PLL and Divider**
- **Register Compatible with the MAX3107, MAX3109, and MAX14830**
- **25-Bump WLP Package (2.1mm x 2.1mm)**

## Ordering Information

| PART | TEMP RANGE | PIN-PACKAGE |
|---|---|---|
| MAX3108AEWA+T | -40°C to +85°C | 25 WLP |

*+Denotes a lead(Pb)-free/RoHS-compliant package.*

## Functional Diagram




**For pricing, delivery, and ordering information, please contact Maxim Direct at 1-888-629-4642, or visit Maxim's website at www.maximintegrated.com.**

*19-6449; Rev 0; 9/12*

## TABLE OF CONTENTS

## TABLE OF CONTENTS (continued)

## LIST OF FIGURES



## LIST OF TABLES

## LIST OF REGISTERS

## ABSOLUTE MAXIMUM RATINGS

(Voltages referenced to AGND.)

$V_L$, $V_{CC}$, $V_{EXT}$, XIN ..... -0.3V to +4.0V
XOUT ..... -0.3V to ($V_{CC}$ + 0.3V)
$V_{18}$ ..... -0.3V to the lesser of ($V_{CC}$ + 0.3V) and 2.0V
$\overline{RST}$, $\overline{IRQ}$, MOSI/A1, $\overline{CS}$/A0, SCLK/SCL, MISO/SDA, LDOEN, SPI/$\overline{I2C}$ ..... -0.3V to ($V_L$ + 0.3V)
TX, RX, $\overline{RTS}$, $\overline{CTS}$, GPIO_ ..... -0.3V to ($V_{EXT}$ + 0.3V)
DGND ..... -0.3V to +0.3V
Continuous Power Dissipation ($T_A$ = +70°C)
WLP (derate 19.2mW/°C above +70°C) ..... 1536mW
Operating Temperature Range ..... -40°C to +85°C
Maximum Junction Temperature ..... +150°C
Storage Temperature Range ..... -65°C to +150°C
Soldering Temperature (reflow) ..... +260°C

## PACKAGE THERMAL CHARACTERISTICS (Note 1)

WLP
Junction-to-Ambient Thermal Resistance ($\theta_{JA}$) ..... 52°C/W
Junction-to-Case Thermal Resistance ($\theta_{JC}$) ..... 11°C/W

**Note 1:** Package thermal resistances were obtained using the method described in JEDEC specification JESD51-7, using a four-layer board. For detailed information on package thermal considerations, refer to **www.maximintegrated.com/thermal-tutorial**.

*Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

## DC ELECTRICAL CHARACTERISTICS

($V_{CC}$ = 1.71V to 3.6V, $V_L$ = 1.71V to 3.6V, $V_{EXT}$ = 1.71V to 3.6V, $T_A$ = -40°C to +85°C, unless otherwise noted. Typical values are at $V_{CC}$ = 2.8V, $V_L$ = 1.8V, $V_{EXT}$ = 2.5V, $T_A$ = +25°C.) (Notes 2, 3)

| PARAMETER | SYMBOL | CONDITIONS | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|---|
| Digital Interface Supply Voltage | $V_L$ | | 1.71 | | 3.6 | V |
| Analog Supply Voltage | $V_{CC}$ | Internal PLL disabled and bypassed | 1.71 | | 3.6 | V |
| | | Internal PLL enabled | 2.35 | | 3.6 | |
| UART Interface Logic Supply Voltage | $V_{EXT}$ | | 1.71 | | 3.6 | V |
| Logic Supply Voltage | $V_{18}$ | | 1.65 | | 1.95 | V |
| **CURRENT CONSUMPTION** | | | | | | |
| $V_{CC}$ Supply Current | $I_{CC}$ | 1.8MHz crystal oscillator active, PLL disabled, SPI/I²C interface idle, UART interfaces idle, LDOEN = high | | | 500 | µA |
| | | Baud rate = 1Mbps, 20MHz external clock, SPI/I²C interface idle, PLL disabled, UART in loopback mode, LDOEN = low | | | 500 | |
| $V_{CC}$ + $V_L$ + VA Shutdown Supply Current | $I_{SHDN}$ | $\overline{RST}$ = low, MISO, SCLK, MOSI, SPI/$\overline{I2C}$, $\overline{CS}$ = low; LDOEN = low/high, $\overline{CTS}$ = low/high, RX = high. | | 0 | 1 | µA |
| $V_{18}$ Input Power-Supply Current in Shutdown Mode | $I_{18SHDN}$ | Shutdown mode, LDOEN = low, $\overline{RST}$ = low, all inputs and outputs are idle | | | 50 | µA |

## DC ELECTRICAL CHARACTERISTICS (continued)

($V_{CC}$ = 1.71V to 3.6V, $V_L$ = 1.71V to 3.6V, $V_{EXT}$ = 1.71V to 3.6V, $T_A$ = -40°C to +85°C, unless otherwise noted. Typical values are at $V_{CC}$ = 2.8V, $V_L$ = 1.8V, $V_{EXT}$ = 2.5V, $T_A$ = +25°C.) (Notes 2, 3)

| PARAMETER | SYMBOL | CONDITIONS | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|---|
| $V_{18}$ Input Power-Supply Current | $I_{18}$ | Baud rate = 1Mbps, 20MHz external clock, PLL disabled, UART in loopback mode, LDOEN = low (Note 4) | | | 2 | mA |
| **SCLK/SCL, MISO/SDA** | | | | | | |
| MISO/SDA Output Logic-Low Voltage in I²C Mode | $V_{OLI2C}$ | Sink current = 3mA, $V_L$ > 2V | | | 0.4 | V |
| | | Sink current = 3mA, $V_L$ < 2V | | | 0.2 x $V_L$ | |
| MISO/SDA Output Logic-Low Voltage in SPI Mode | $V_{OLSPI}$ | Sink current = 2mA | | | 0.4 | V |
| MISO/SDA Output Logic-High Voltage in SPI Mode | $V_{OHSPI}$ | Source current = 2mA | $V_L$ - 0.4 | | | V |
| Input Logic-Low Voltage | $V_{IL}$ | SPI and I²C mode | | | 0.3 x $V_L$ | V |
| Input Logic-High Voltage | $V_{IH}$ | SPI and I²C mode | 0.7 x $V_L$ | | | V |
| Input Hysteresis | $V_{HYST}$ | SPI and I²C mode | | 0.05 x $V_L$ | | V |
| Input Leakage Current | $I_{IL}$ | $V_{IN}$ = 0 to $V_L$, SPI and I²C mode | -1 | | +1 | µA |
| Input Capacitance | $C_{IN}$ | SPI and I²C mode | | 5 | | pF |
| **SPI/$\overline{I2C}$, $\overline{CS}$/A0, MOSI/A1 INPUTS** | | | | | | |
| Input Logic-Low Voltage | $V_{IL}$ | SPI and I²C mode | | | 0.3 x $V_L$ | V |
| Input Logic-High Voltage | $V_{IH}$ | SPI and I²C mode | 0.7 x $V_L$ | | | V |
| Input Hysteresis | $V_{HYST}$ | SPI and I²C mode | | 50 | | mV |
| Input Leakage Current | $I_{IL}$ | $V_{IN}$ = 0 to $V_L$, SPI and I²C mode | -1 | | +1 | µA |
| Input Capacitance | $C_{IN}$ | SPI and I²C mode | | 5 | | pF |
| **$\overline{IRQ}$ OUTPUT (OPEN DRAIN)** | | | | | | |
| Output Logic-Low Voltage | $V_{OL}$ | Sink current = 2mA | | | 0.4 | V |
| Output Leakage Current | $I_{OL}$ | $V_{\overline{IRQ}}$ = 0 to $V_L$, $\overline{IRQ}$ is not asserted | -1 | | +1 | µA |
| **LDOEN AND $\overline{RST}$ INPUTS** | | | | | | |
| Input Logic-Low Voltage | $V_{IL}$ | | | | 0.3 x $V_L$ | V |
| Input Logic-High Voltage | $V_{IH}$ | | 0.7 x $V_L$ | | | V |
| Input Hysteresis | $V_{HYST}$ | | | 50 | | mV |
| Input Leakage Current | $I_{IL}$ | $V_{IN}$ = 0 to $V_L$ | -1 | | +1 | µA |

## DC ELECTRICAL CHARACTERISTICS (continued)

($V_{CC}$ = 1.71V to 3.6V, $V_L$ = 1.71V to 3.6V, $V_{EXT}$ = 1.71V to 3.6V, $T_A$ = -40°C to +85°C, unless otherwise noted. Typical values are at $V_{CC}$ = 2.8V, $V_L$ = 1.8V, $V_{EXT}$ = 2.5V, $T_A$ = +25°C.) (Notes 2, 3)

| PARAMETER | SYMBOL | CONDITIONS | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|---|
| **UART INTERFACE** | | | | | | |
| **$\overline{RTS}$, TX OUTPUTS** | | | | | | |
| Output Logic-Low Voltage | $V_{OL}$ | Sink current = 2mA | | | 0.4 | V |
| Output Logic-High Voltage | $V_{OH}$ | Source current = 2mA | 0.7 x $V_{EXT}$ | | | V |
| Input Leakage Current | $I_{IL}$ | Output is three-stated, $V_{\overline{RTS}}$ = 0 to $V_{EXT}$ | -1 | | +1 | µA |
| Input Capacitance | $C_{IN}$ | High-impedance mode | | 5 | | pF |
| **$\overline{CTS}$, RX INPUTS** | | | | | | |
| Input Logic-Low Voltage | $V_{IL}$ | | | | 0.3 x $V_{EXT}$ | V |
| Input Logic-High Voltage | $V_{IH}$ | | 0.7 x $V_{EXT}$ | | | V |
| Input Hysteresis | $V_{HYST}$ | | | 50 | | mV |
| $\overline{CTS}$ Input Leakage Current | $I_{IL}$ | $V_{\overline{CTS}}$ = 0 to $V_{EXT}$ | -1 | | +1 | µA |
| RX Pullup Current | $I_{PU}$ | $V_{RX}$ = 0V, $V_{EXT}$ = 3.6V | -7.5 | -5.5 | -3.5 | µA |
| Input Capacitance | $C_{IN}$ | | | 5 | | pF |
| **GPIO_ INPUTS/OUTPUTS** | | | | | | |
| Output Logic-Low Voltage | $V_{OL}$ | Sink current = 20mA, push-pull or open-drain output type, $V_{EXT}$ > 2.3V | | | 0.45 | V |
| | | Sink current = 20mA, push-pull or open-drain output type, $V_{EXT}$ < 2.3V | | | 0.55 | |
| Output Logic-High Voltage | $V_{OH}$ | Source current = 5mA, push-pull output type | $V_{EXT}$ - 0.4V | | | V |
| Input Logic-Low Voltage | $V_{IL}$ | GPIO_ is configured as an input | | | 0.4 | V |
| Input Logic-High Voltage | $V_{IH}$ | GPIO_ is configured as an input | 2/3 x $V_{EXT}$ | | | V |
| Pulldown Current | $I_{PD}$ | $V_{GPIO\_}$ = $V_{EXT}$ = 3.6V, GPIO_ is configured as an input | 3.5 | 5.5 | 7.5 | µA |
| **XIN** | | | | | | |
| Input Logic-Low Voltage | $V_{IL}$ | | | | 0.6 | V |
| Input Logic-High Voltage | $V_{IH}$ | | 1.2 | | | V |
| Input Capacitance | $C_{XIN}$ | | | 16 | | pF |
| **XOUT** | | | | | | |
| Input Capacitance | $C_{XOUT}$ | | | 16 | | pF |

## AC ELECTRICAL CHARACTERISTICS

($V_{CC}$ = 1.71V to 3.6V, $V_L$ = 1.71V to 3.6V, $V_{EXT}$ = 1.71V to 3.6V, $T_A$ = -40°C to +85°C, unless otherwise noted. Typical values are at $V_{CC}$ = 2.8V, $V_L$ = 1.8V, $V_{EXT}$ = 2.5V, $T_A$ = +25°C.) (Note 2)

| PARAMETER | SYMBOL | CONDITIONS | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|---|
| External Cystal Frequency | $f_{XOSC}$ | | 1 | | 4 | MHz |
| External Clock Frequency | $f_{CLK}$ | | 0.5 | | 35 | MHz |
| External Clock Duty Cycle | | (Note 5) | 45 | | 55 | % |
| Baud-Rate Generator Clock Input Frequency | $f_{REF}$ | | | | 96 | MHz |
| **I²C BUS: TIMING CHARACTERISTICS (Figure 1)** | | | | | | |
| SCL Clock Frequency | $f_{SCL}$ | Standard mode | | | 100 | kHz |
| | | Fast mode | | | 400 | |
| | | Fast mode plus | | | 1000 | |
| Bus Free Time Between a STOP and START Condition | $t_{BUF}$ | Standard mode | 4.7 | | | µs |
| | | Fast mode | 1.3 | | | |
| | | Fast mode plus | 0.5 | | | |
| Hold Time for START Condition and Repeated START Condition | $t_{HD:STA}$ | Standard mode | 4.0 | | | µs |
| | | Fast mode | 0.6 | | | |
| | | Fast mode plus | 0.26 | | | |
| Low Period of the SCL Clock | $t_{LOW}$ | Standard mode | 4.7 | | | µs |
| | | Fast mode | 1.3 | | | |
| | | Fast mode plus | 0.5 | | | |
| High Period of the SCL Clock | $t_{HIGH}$ | Standard mode | 4.0 | | | µs |
| | | Fast mode | 0.6 | | | |
| | | Fast mode plus | 0.26 | | | |
| Data Hold Time | $t_{HD:DAT}$ | Standard mode | 0 | | 0.9 | µs |
| | | Fast mode | 0 | | 0.9 | |
| | | Fast mode plus | 0 | | | |
| Data Setup Time | $t_{SU:DAT}$ | Standard mode | 250 | | | ns |
| | | Fast mode | 100 | | | |
| | | Fast mode plus | 50 | | | |
| Setup Time for Repeated START Condition | $t_{SU:STA}$ | Standard mode | 4.7 | | | µs |
| | | Fast mode | 0.2 | | | |
| | | Fast mode plus | 0.26 | | | |
| Rise Time of Incoming SDA and SCL Signals | $t_R$ | Standard mode (0.3 x $V_L$ to 0.7 x $V_L$) (Note 6) | 20 + 0.1$C_B$ | | 1000 | ns |
| | | Fast mode (0.3 x $V_L$ to 0.7 x $V_L$) (Note 6) | 20 + 0.1$C_B$ | | 300 | |
| | | Fast mode plus | | | 120 | |
| Fall Time of SDA and SCL Signals | $t_F$ | Standard mode (0.3 x $V_L$ to 0.7 x $V_L$) (Note 6) | 20 + 0.1$C_B$ | | 1000 | ns |
| | | Fast mode (0.3 x $V_L$ to 0.7 x $V_L$) (Note 6) | 20 + 0.1$C_B$ | | 300 | |
| | | Fast mode plus | | | 120 | |

## AC ELECTRICAL CHARACTERISTICS (continued)

($V_{CC}$ = 1.71V to 3.6V, $V_L$ = 1.71V to 3.6V, $V_{EXT}$ = 1.71V to 3.6V, $T_A$ = -40°C to +85°C, unless otherwise noted. Typical values are at $V_{CC}$ = 2.8V, $V_L$ = 1.8V, $V_{EXT}$ = 2.5V, $T_A$ = +25°C.) (Note 2)

| PARAMETER | SYMBOL | CONDITIONS | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|---|
| Setup Time for STOP Condition | $t_{SU:STO}$ | Standard mode | 4.7 | | | µs |
| | | Fast mode | 0.6 | | | |
| | | Fast mode plus | 0.26 | | | |
| Capacitive Load for SDA and SCL | $C_B$ | Standard mode (Note 5) | | | 400 | pF |
| | | Fast mode (Note 5) | | | 400 | |
| | | Fast mode plus (Note 5) | | | 550 | |
| SCL and SDA I/O Capacitance | $C_{I/O}$ | (Note 5) | | | 10 | pF |
| Pulse Width of Spike Suppressed | $t_{SP}$ | | | | 50 | ns |
| **SPI BUS: TIMING CHARACTERISTICS (Figure 2)** | | | | | | |
| SCLK Clock Period | $t_{CH}+t_{CL}$ | | 38.4 | | | ns |
| SCLK Pulse Width High | $t_{CH}$ | | 16 | | | ns |
| SCLK Pulse Width Low | $t_{CL}$ | | 16 | | | ns |
| $\overline{CS}$ Fall to SCLK Rise Time | $t_{CSS}$ | | 0 | | | ns |
| MOSI Hold Time | $t_{DH}$ | | 3 | | | ns |
| MOSI Setup Time | $t_{DS}$ | | 5 | | | ns |
| Output Data Propagation Delay | $t_{DO}$ | | | | 20 | ns |
| MISO Rise and Fall Times | $t_{FT}$ | | | | 10 | ns |
| $\overline{CS}$ Hold Time | $t_{CSH}$ | | 30 | | | ns |

**Note 2:** All units are production tested at $T_A$ = +25°C. Specifications over temperature are guaranteed by design.
**Note 3:** Currents entering the IC are positive and currents exiting the IC are negative.
**Note 4:** When $V_{18}$ is powered by an external voltage supply, it must have current capability above or equal to $I_{18}$.
**Note 5:** Guaranteed by design; not production tested.
**Note 6:** $C_B$ is the total capacitance of either the clock or data line of the synchronous bus in pF.

## Timing Diagrams

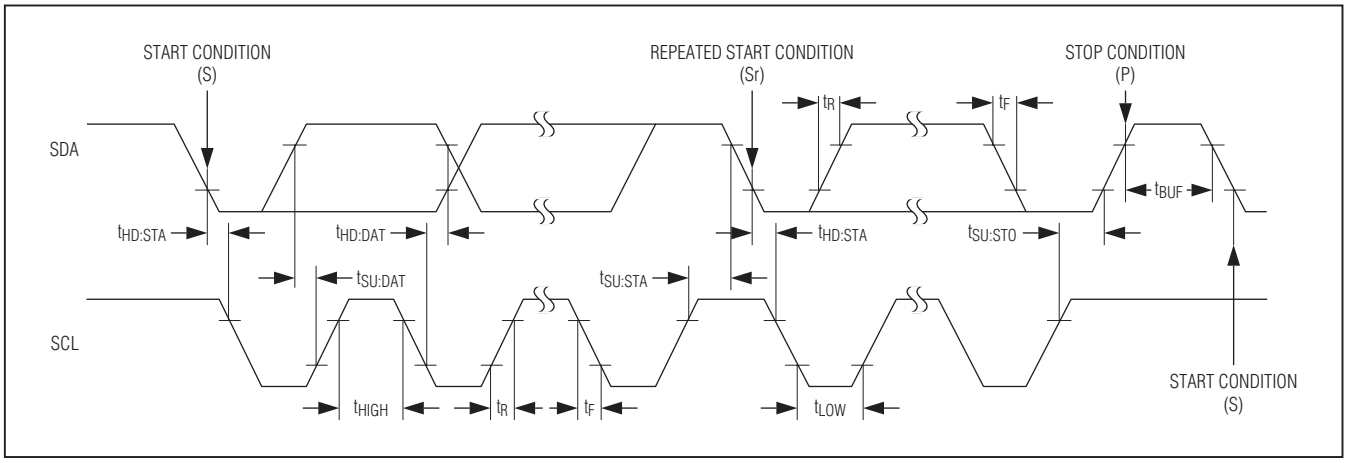



*Figure 1. I2C Timing Diagram*

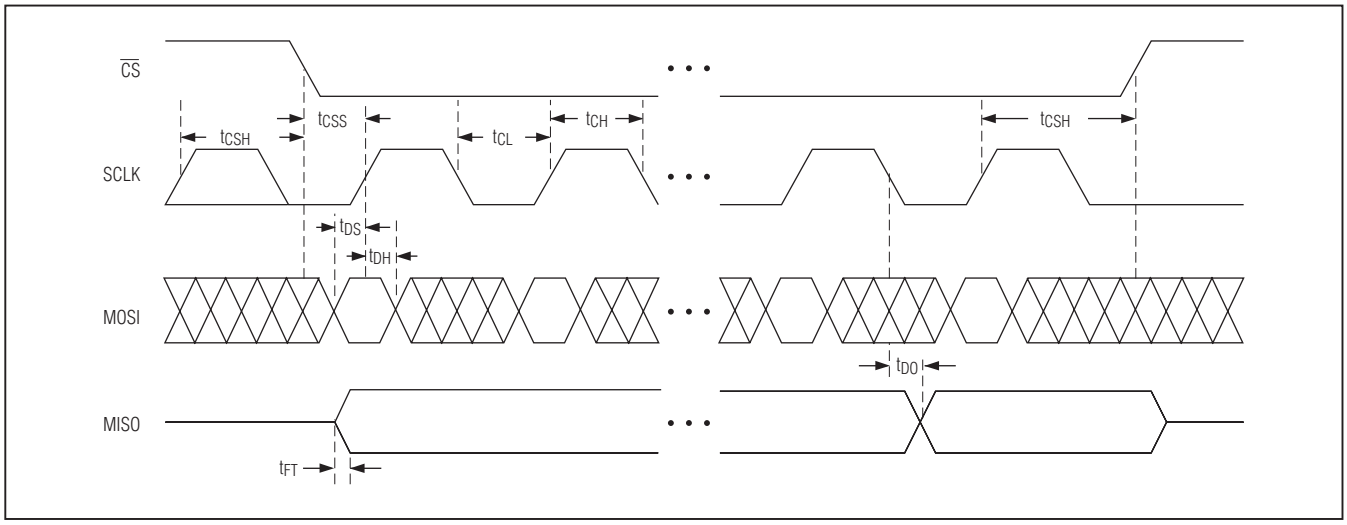
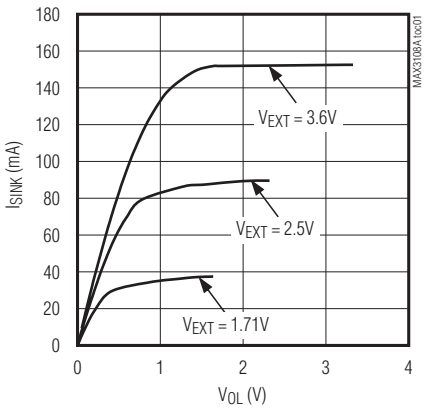



*Figure 2. SPI Timing Diagram*

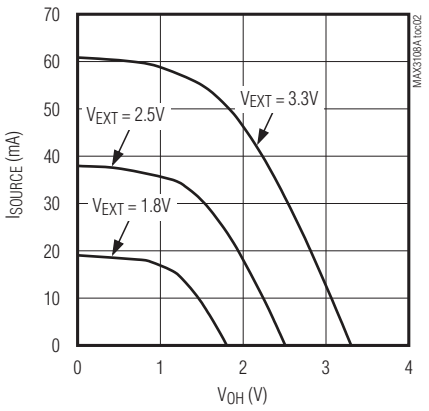
## Typical Operating Characteristics

($V_{CC}$ = 2.5V, $V_L$ = 2.5V, $V_{EXT}$ = 2.5V, $V_{LDOEN}$ = $V_L$, $T_A$ = +25°C, unless otherwise noted.)

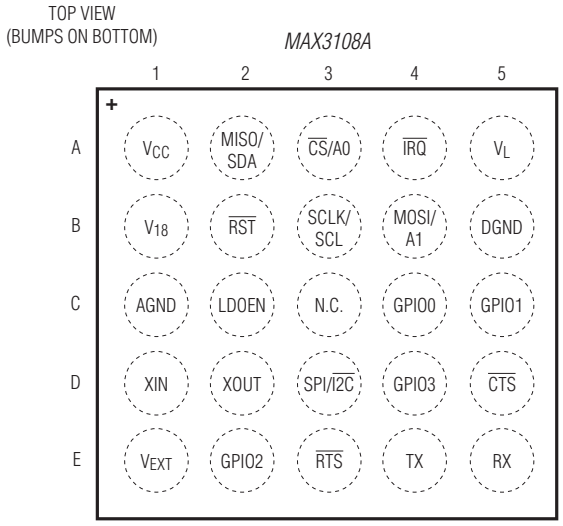
**SINK CURRENT (OPEN DRAIN) vs. GPIO_ OUTPUT LOW VOLTAGE**




**SOURCE CURRENT (PUSH-PULL) vs. GPIO_ OUTPUT HIGH VOLTAGE**



## Bump Configuration




**WLP**
**(2.1mm × 2.1mm)**

## Bump Description

| BUMP | NAME | FUNCTION |
|---|---|---|
| A1 | $V_{CC}$ | Analog Power Supply. $V_{CC}$ powers the PLL and internal LDO. Bypass $V_{CC}$ with a 0.1µF ceramic capacitor to AGND. |
| A2 | MISO/SDA | Serial-Data Output. When SPI/$\overline{I2C}$ is high, MISO/SDA functions as the SPI master-input, slave-output (MISO). When SPI/$\overline{I2C}$ is low, MISO/SDA functions as the SDA, $I^2C$ serial-data input/output. MISO/SDA is high impedence when $\overline{RST}$ is driven low, or when the externally supplied 18V is powered off. |
| A3 | $\overline{CS}$/A0 | Active-Low Chip-Select and Address 0 Input. When SPI/$\overline{I2C}$ is high, $\overline{CS}$/A0 functions as the $\overline{CS}$, SPI active-low chip-select. When SPI/$\overline{I2C}$ is low, $\overline{CS}$/A0 functions as the A0 $I^2C$ device address programming input. Connect $\overline{CS}$/A0 to DGND, $V_L$, SCL, or SDA when SPI/$\overline{I2C}$ is low. |
| A4 | $\overline{IRQ}$ | Active-Low Interrupt Open-Drain Output. $\overline{IRQ}$ is asserted when an interrupt is pending. $\overline{IRQ}$ is high impedance when $\overline{RST}$ is driven low. |
| A5 | $V_L$ | Digital Interface Power Supply. $V_L$ powers the internal logic-level translators for $\overline{RST}$, $\overline{IRQ}$, MOSI/A1, $\overline{CS}$/A0, SCLK/SCL, MISO/SDA, LDOEN, and SPI/$\overline{I2C}$. Bypass $V_L$ with a 0.1µF ceramic capacitor to DGND. |
| B1 | $V_{18}$ | Internal 1.8V LDO Output and 1.8V Power-Supply Input. Bypass $V_{18}$ with a 1µF ceramic capacitor to DGND. |
| B2 | $\overline{RST}$ | Active-Low Reset Input. Drive $\overline{RST}$ low to force the UART into hardware reset mode. Driving $\overline{RST}$ low also enables low-power shutdown mode. When $\overline{RST}$ is low, the internal 18V LDO is switched off, even if LDOEN input is kept high. |

## Bump Description (continued)

| BUMP | NAME | FUNCTION |
|---|---|---|
| B3 | SCLK/SCL | Serial-Clock Input. When SPI/$\overline{I2C}$ is high, SCLK/SCL functions as the SCLK SPI serial-clock input (up to 26MHz). When SPI/$\overline{I2C}$ is low, SCLK/SCL functions as the SCL, I²C serial-clock input (up to 1MHz in fast mode plus). |
| B4 | MOSI/A1 | Serial-Data Input and Address 1 Input. When SPI/$\overline{I2C}$ is high, MOSI/A1 functions as the SPI master-output, slave-input (MOSI). When SPI/$\overline{I2C}$ is low, MOSI/A1 functions as the A1 I²C device address programming input. Connect MOSI/A1 to DGND, $V_L$, SCL, or SDA when SPI/$\overline{I2C}$ is low. |
| B5 | DGND | Digital Ground |
| C1 | AGND | Analog Ground |
| C2 | LDOEN | LDO Enable Input. Drive LDOEN high to enable the internal 1.8V LDO. Drive LDOEN low to disable the internal LDO. Supply $V_{18}$ with an external voltage source when LDOEN is low. |
| C3 | N.C. | Not Connected. Internally not connected. |
| C4 | GPIO0 | General-Purpose Input/Output 0. GPIO0 is user-programmable as an input or output (push-pull or open drain) or an external event-driven interrupt source. GPIO0 has a weak pulldown resistor to DGND when configured as an input. |
| C5 | GPIO1 | General-Purpose Input/Output 1. GPIO1 is user programmable as an input or output (push-pull or open drain) or an external event-driven interrupt source. GPIO1 has a weak pulldown resistor to DGND when configured as an input. |
| D1 | XIN | Crystal/Clock Input. When using an external crystal, connect one end of the crystal to XIN and the other end to XOUT. When using an external clock source, drive XIN with the single-ended external clock. |
| D2 | XOUT | Crystal Output. When using an external crystal, connect one end of the crystal to XOUT and the other end to XIN. When using an external clock source, leave XOUT unconnected. |
| D3 | SPI/$\overline{I2C}$ | SPI Selector Input or Active-Low I²C. Drive SPI/$\overline{I2C}$ low to enable I²C. Drive SPI/$\overline{I2C}$ high to enable SPI. |
| D4 | GPIO3 | General-Purpose Input/Output 3. GPIO3 is user-programmable as an input or output (push-pull or open drain) or an external event-driven interrupt source. GPIO3 has a weak pulldown resistor to DGND when configured as an input. |
| D5 | $\overline{CTS}$ | Active-Low Clear-to-Send Input. $\overline{CTS}$ is a flow-control status input. |
| E1 | $V_{EXT}$ | Transceiver Interface Power Supply. $V_{EXT}$ powers the internal logic-level translators for RX, TX, $\overline{RTS}$, $\overline{CTS}$, and GPIO_. Bypass $V_{EXT}$ with a 0.1µF ceramic capacitor to DGND. |
| E2 | GPIO2 | General-Purpose Input/Output 2. GPIO2 is user programmable as an input or output (push-pull or open drain) or an external event-driven interrupt source. GPIO2 has a weak pulldown resistor to DGND when configured as an input. |
| E3 | $\overline{RTS}$ | Active-Low Request-to-Send Output. $\overline{RTS}$ can be set high or low by programming the LCR register. $\overline{RTS}$ is logic-high when $\overline{RST}$ is low when the externally supplied 18V is not powered. |
| E4 | TX | Serial Transmitting Data Output. TX is logic-high when $\overline{RST}$ is low or when the externally supplied 18V is not powered. |
| E5 | RX | Serial Receiving Data Input. RX has an internal weak pullup resistor to $V_{EXT}$. |

## Detailed Description

The MAX3108A universal asynchronous receiver-transmitter (UART) bridges an SPI/MICROWIRE™ or I2C microprocessor bus to an asynchronous serial-data communication link. The device contains an advanced UART, a fractional baud-rate generator, and four GPIOs. Eight-bit registers configure and monitor the device and are accessed through SPI or I2C, selectable by an external pin. The registers are organized by related function as shown in the *Register Map* section.

The host controller loads data into the Transmit Hold register (**THR**) through the SPI or I2C interface. This data is automatically pushed into the transmit first-in/first-out (FIFO), formatted, and sent out at TX. The device adds START, STOP, and parity bits to the data before transmitting at the selected baud rate. The clock configuration registers determine the baud rate, clock source, and clock frequency prescaling.

The MAX3108A receiver detects a START bit as a high-to-low transition on RX. An internal clock samples this data at 16 times the baud rate. The received data is automatically placed in the receive FIFO and can be read out by the host microcontroller through the Receive Hold register (**RHR**).

The MAX3108A's register set is compatible with the MAX3107.

### Register Set

The device has a flat register structure without shadow registers. The registers are 8 bits wide. The registers have some similarities to the 16C550 registers.

### Receive and Transmit FIFOs

The UART's receiver and transmitter each have a 128-word-deep FIFO, reducing the number of intervals that the host processor needs to dedicate for high-speed, high-volume data transfer to and from the device. As the data rates of the asynchronous Rx/Tx interfaces increase and get closer to those of the host controller's SPI/I2C data rates, UART management and flow-control can make up a significant portion of the host's activity. By increasing FIFO size, the host is interrupted less often and can use data block transfers to and from the FIFOs.

FIFO trigger levels can generate interrupts to the host controller, signaling that programmed FIFO fill levels have been reached. The transmitter and receiver trigger levels are programmed through the **FIFOTrgLvl** register with a resolution of eight FIFO locations. The receive FIFO trigger signals to the host either that the receive FIFO has a defined number of words waiting to be read out in a block or that a known number of vacant FIFO locations are available and ready to be filled. The transmit FIFO trigger generates an interrupt when the transmit FIFO fill level is above the programmed trigger level. The host then knows to throttle data writing to the transmit FIFO through **THR**.

The host can read out the number of words present in each of the FIFOs through the **TxFIFOLvl** and **RxFIFOLvl** registers.

The contents of the TxFIFO and RxFIFO are both cleared when the **MODE2**[1]: FIFORst bit is set high.

### Transmitter Operation

Figure 3 shows the structure of the transmitter with the TxFIFO. The transmit FIFO can hold up to 128 words of data that are added by writing to the **THR** register.

The current number of words in the TxFIFO can be manually read out by the host controller through the **TxFIFOLvl** register. The transmit FIFO fill level can be

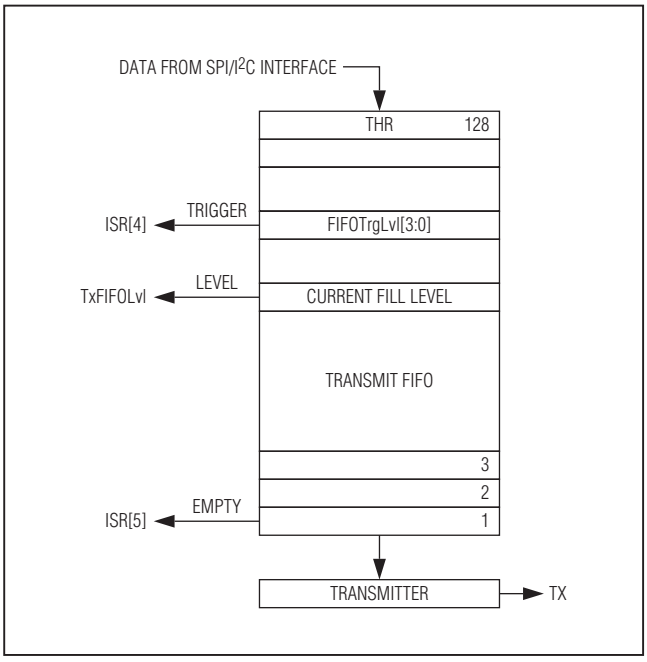



*Figure 3. Transmit FIFO Signals*

*MICROWIRE is a trademark of National Semiconductor Corp.*

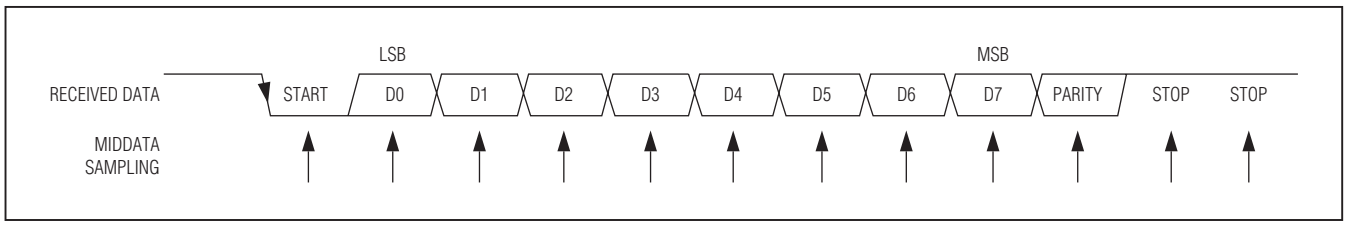



*Figure 4. Receive Data Format*

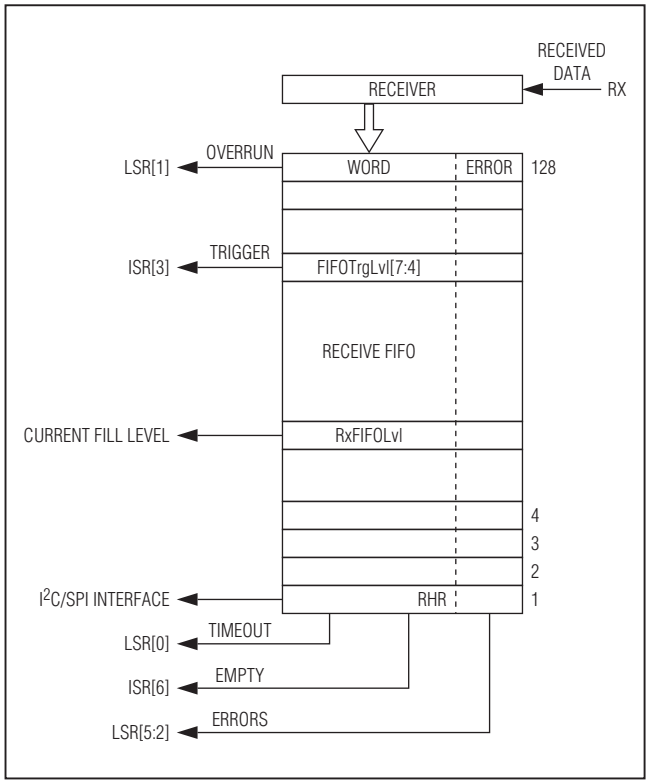



*Figure 5. Receive FIFO*

programmed to generate an interrupt when more than a programmed number of words are present in the TxFIFO through the **FIFOTrgLvl** register. This TxFIFO interrupt trigger level is selectable by the **FIFOTrgLvl**[3:0] bits. When the transmit FIFO fill level increases beyond the programmed trigger level, an interrupt is generated in **ISR**[4]: TxTrgInt.

An interrupt is generated in **ISR**[5]: TFifoEmptyInt when the transmit FIFO is empty. **ISR**[5] goes high when the transmitter starts transmitting the last word in the TxFIFO. An additional interrupt is generated in **STSInt**[7]: TxEmptyInt when the transmitter completes transmitting the last word.

To halt transmission, set the **MODE1**[1]: TxDisabl bit high. After TxDisabl is set, the transmitter completes the transmission of the current character and then ceases transmission. Turn the transmitter off prior to enabling auto software flow control and AutoRTS flow control.

The TX output logic can be inverted through the **IrDA**[5]: TxInv bit. Unless otherwise noted, all transmitter logic described in this data sheet assumes that TxInv is set low.

### *Receiver Operation*

The receiver expects the format of the data at RX to be as shown in Figure 4. The quiescent logic state is logic-high and the first bit (the START bit) is logic-low. The 8-bit data word is expected to be received LSB first. The receiver samples the data near the midbit instant (Figure 4). The received words and their associated errors are deposited into the receive FIFO. Errors and status information are stored for every received word (Figure 5). The host reads the data out of the receive FIFO by reading **RHR**, which comes out oldest data first. The status and error information for the word most recently read from **RHR** is located in the Line Status register (**LSR**). After a word is read out of **RHR**, **LSR** contains the status information for that word.

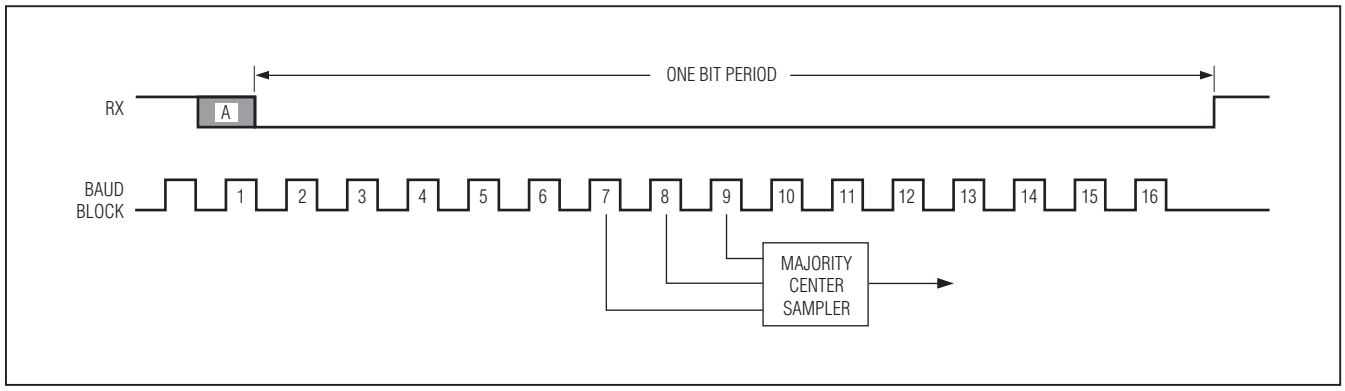



*Figure 6. Midbit Sampling*

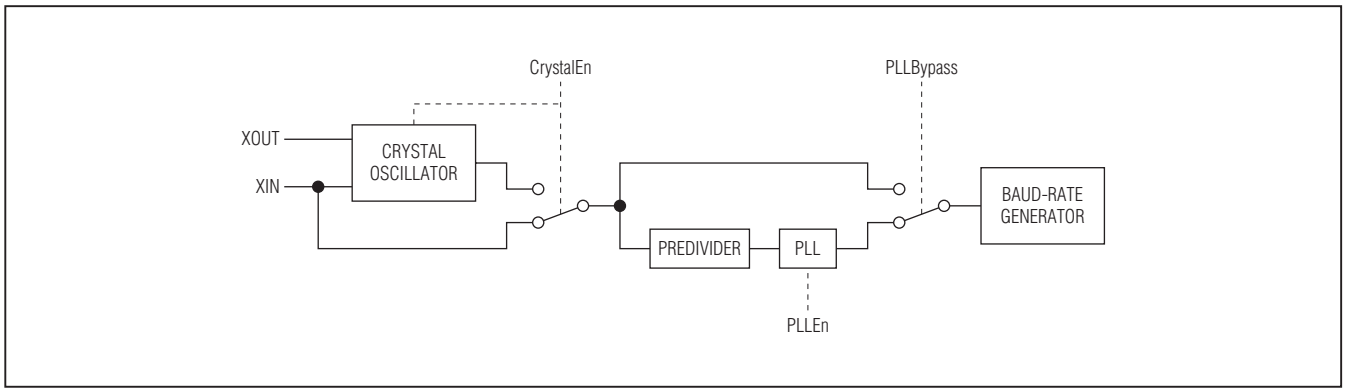



*Figure 7. Clock Selection Diagram*

The following three error conditions are checked for each received word: parity error, frame error, and noise on the line. Parity errors are detected by calculating either even or odd parity of the received word as programmed by register settings. Framing errors are detected when the received data frame does not match the expected frame format in length. Line noise is detected by checking the logical congruency of the three samples taken of each bit (Figure 6).

The receiver can be turned off by setting the **MODE1**[0]: RxDisabl bit high. After this bit is set high, the device turns the receiver off immediately following the current word and does not receive any further data.

The RX input logic can be inverted by setting the **IrDA**[4]: RxInv bit high. Unless otherwise noted, all receiver logic described in this data sheet assumes that RxInv is set low.

### Line Noise Indication

When operating in standard or 2x (i.e., not 4x) rate mode, the device checks that the binary logic level of the three samples per received bit are identical. If any of the three samples per received bit have differing logic levels, then noise on the transmission line has affected the received data and it is considered to be noisy. This noise indication is reflected in the **LSR**[5]: RxNoise bit for each received byte. Parity errors are another indication of noise, but are not as sensitive.

### Clock Selection

The device can be clocked by either an external crystal or an external clock source. Figure 7 shows a simplified diagram of the clock selection circuitry. When the device is clocked by a crystal, the **STSInt**[5]: ClkReady bit indicates when the crystal oscillator has reached steady state and the baud-rate generator is ready for stable operation.

The baud-rate clock can be routed to the $\overline{RTS}$ output by setting the **CLKSource**[7]: CLKtoRTS bit high. The clock rate is 16x the baud rate in standard operating mode, 8x the baud rate in 2x rate mode, and 4x the baud rate in 4x rate mode. If the fractional portion of the baud-rate generator is used, the clock is not regular and exhibits jitter.

#### Crystal Oscillator

The device is equipped with a crystal oscillator to provide high baud-rate accuracy and low power consumption. Set the **CLKSource**[1]: CrystalEn bit high to enable and select the crystal oscillator. The on-chip crystal oscillator has integrated load capacitances of 16pF in both the XIN and XOUT pins. Connect only an external crystal or ceramic oscillator between XIN and XOUT.

#### External Clock Source

Connect an external single-ended clock source to XIN when not using the crystal oscillator. Leave XOUT unconnected. Set the **CLKSource**[1]: CrystalEn bit low to select external clocking.

### PLL and Predivider

The internal predivider and PLL allow for compatibility with a wide range of external clock frequencies and baud rates. The PLL can be configured to multiply the input clock rate by a factor of 6, 48, 96, or 144 by the **PLLConfig**[7:6] bits. The predivider is located between the input clock and the PLL and allows division of the input clock by an integer factor between 1 and 63. This value is defined by the **PLLConfig**[5:0] bits. See the **PLLConfig** register description for more information. Use of the PLL requires $V_{CC}$ to be higher than 2.35V.

### Fractional Baud-Rate Generator

The internal fractional baud-rate generator provides a high degree of flexibility and high resolution in baud-rate programming. The baud-rate generator has a 16-bit integer divisor and a 4-bit word for the fractional divisor. The fractional baud-rate generator can be used either with the crystal oscillator or external clock source.

The integer and fractional divisors are calculated by the divisor, D:

$$D = \frac{f_{REF} \times RateMode}{16 \times BaudRate}$$

where $f_{REF}$ is the reference frequency input to the baud-rate generator, RateMode is the rate mode multiplier (1x default), BaudRate is the desired baud rate, and D is the ideal divisor. $f_{REF}$ must be less than 96MHz. RateMode is 1 in 1x rate mode, 2 in 2x rate mode, and 4 in 4x rate mode.

The integer divisor portion, DIV, of the divisor, D, is obtained by truncating D:

DIV(decimal) = TRUNC(D)

DIV can be a maximum of 16 bits (65,535) wide and is programmed into the two single-byte-wide registers **DIVMSB** and **DIVLSB**. The minimum allowed value for **DIVLSB** is 1.

The fractional portion of the divisor, FRACT, is a 4-bit nibble that is programmed into **BRGConfig**[3:0]. The maximum value is 15, allowing the divisor to be programmed with a resolution of 0.0625. FRACT is calculated as: FRACT = ROUND(16 x (D - DIV)).

The following is an example of how to calculate the divisor. It is based on a required baud rate of 190kbaud and a 28.23MHz reference input frequency and 1x (default) rate mode.

The ideal divisor is calculated as:

D = 28,230,000/(16 x 190,000) = 9.286

hence DIV = 9.

FRACT = ROUND(16 x 0.286) = 5

so **DIVMSB** = 0x00, **DIVLSB** = 0x09, and **BRGConfig**[3:0] = 0x05.

The resulting actual baud rate can be calculated as:

$$BR_{ACTUAL} = \frac{f_{REF} \times RateMode}{16 \times D_{ACTUAL}}$$

For this example: $D_{ACTUAL}$ = 9 + 5/16 = 9.3125, RateMode = 1, and

$BR_{ACTUAL}$= 28,230,000/(16 x 9.3125) = 189,463 baud.

Thus, the actual baud rate is within 0.28% of the ideal rate.

### 2x and 4x Rate Modes

The device offers 2x and 4x rate modes to support higher baud rates than possible with standard operation using 16x sampling. In these modes, the reference clock rate only needs to be either 8x or 4x higher than the baud rate, respectively. In 4x rate mode, each received bit is only sampled once at the midbit instant instead of the usual three samples to determine the logic value of the received bit. This reduces the ability to detect line noise on the received data in 4x rate mode. The 2x and 4x rate modes are selectable through **BRGConfig**[5:4]. Note that IrDA encoding and decoding does not operate in 2x and 4x rate modes.

When 2x rate mode is selected, the actual baud rate is twice the rate programmed into the baud-rate generator. If 4x rate mode is enabled, the actual baud rate on the line is quadruple that of the programmed baud rate (Figure 8).

### Multidrop Mode

In multidrop mode, also known as 9-bit mode, the data word length is 8 bits and a 9th bit is used for distinguishing between an address word and a data word. Multidrop mode is enabled by the **MODE2**[6]: MultiDrop bit. The MultiDrop bit takes the place of the parity bit in the data word structure. Parity checking is disabled and an interrupt is generated in **SpclCharInt**[5]: MultiDropInt when an address (9th bit is 1) is received while in multidrop mode.

It is up to the host processor to filter out the data intended for its address. Alternatively, the auto data-filtering feature can be used to automatically filter out the data not intended for the station's specific 9-bit mode address.

### Auto Data Filtering in Multidrop Mode

In multidrop mode, the device can be configured to automatically filter out data that is not meant for its address. The address is user-definable either by programming a register value or a combination of a register value and GPIO hardware inputs. Use either the entire **XOFF2** register or the **XOFF2**[7:4] bits in combination with GPIO_ inputs to define the address.

Enable multidrop mode by setting the **MODE2**[6]: MultiDrop bit high and enable auto data filtering by setting the **MODE2**[4]: SpecialChr bit high.

When using register bits in combination with GPIO_ inputs to define the address, the MSB of the address is written to the **XOFF2**[7:4] bits, while the LSBs of the address are defined by the GPIOs. To enable this address-definition method along with auto data filtering, set the **FlowCtrl**[2]: GPIAddr bit high in addition to the **MODE2**[4]: SpecialChr and **MODE2**[6]: MultiDrop bits. The GPIO_ inputs are automatically read when the **FlowCtrl**[2]: GPIAddr bit is set high, and the address is automatically updated on logic changes to any GPIO pin.

When using auto data filtering, the device checks each received address against the programmed station address. When an address is received that matches the station's address, received data is stored in the RxFIFO. When an address is received that does not match the station's address, received data is discarded. Addresses are not stored into the FIFO but an interrupt is still generated in **SpclCharInt**[5]: MultiDropInt upon receiving an address. An additional interrupt is generated in **SpclCharInt**[3]: XOFF2Int when the station address is received.

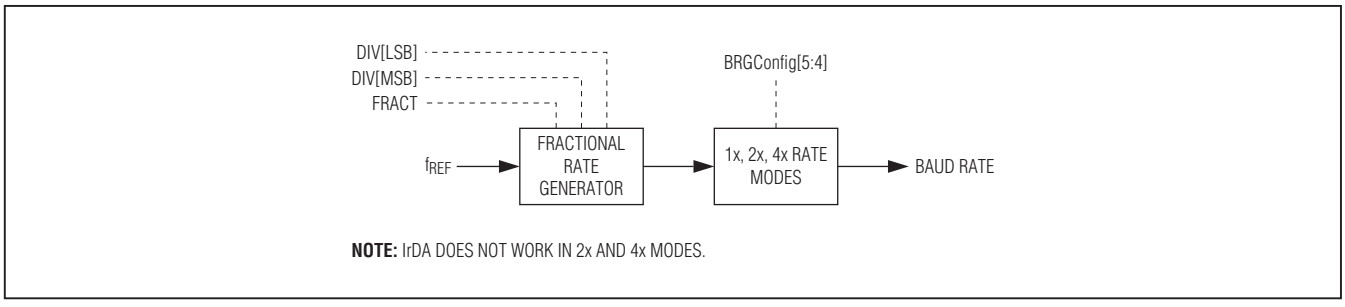



*Figure 8. 2x and 4x Baud Rates*

### Auto Transceiver Direction Control

In some half-duplex communication systems, the transceiver's transmitter must be turned off when data is being received to avoid loading the bus. This is the case in half-duplex RS-485 communication. Similarly, in full-duplex multidrop communication such as RS-485 or RS-422 V.11, only one transmitter can be enabled at any one time while the others must be disabled. The device can be configured to automatically enable/disable a transceiver's transmitter and/or receiver at the hardware level by controlling its DE and $\overline{RE}$ pins. This feature relieves the host processor of this time-critical task.

The $\overline{RTS}$ output is used to control the transceivers' transmit-enable input and is automatically set high when the device's transmitter starts transmission. This occurs as soon as data is present in the transmit FIFO. Auto transceiver direction control is enabled by the **MODE1**[4]: TrnscvCtrl bit. Figure 9 shows a typical connection in an RS-485 application using the auto transceiver direction control feature.

The $\overline{RTS}$ output can be set high in advance of Tx transmission by a programmable time period called the setup time (Figure 10). The setup time is programmed by the **HDplxDelay**[7:4]: Setupx bits. Similarly, the $\overline{RTS}$ output can be held high for a programmable period after the transmitter has completed transmission called the hold time. The hold time is programmed by the **HDplxDelay**[3:0]: Holdx bits.

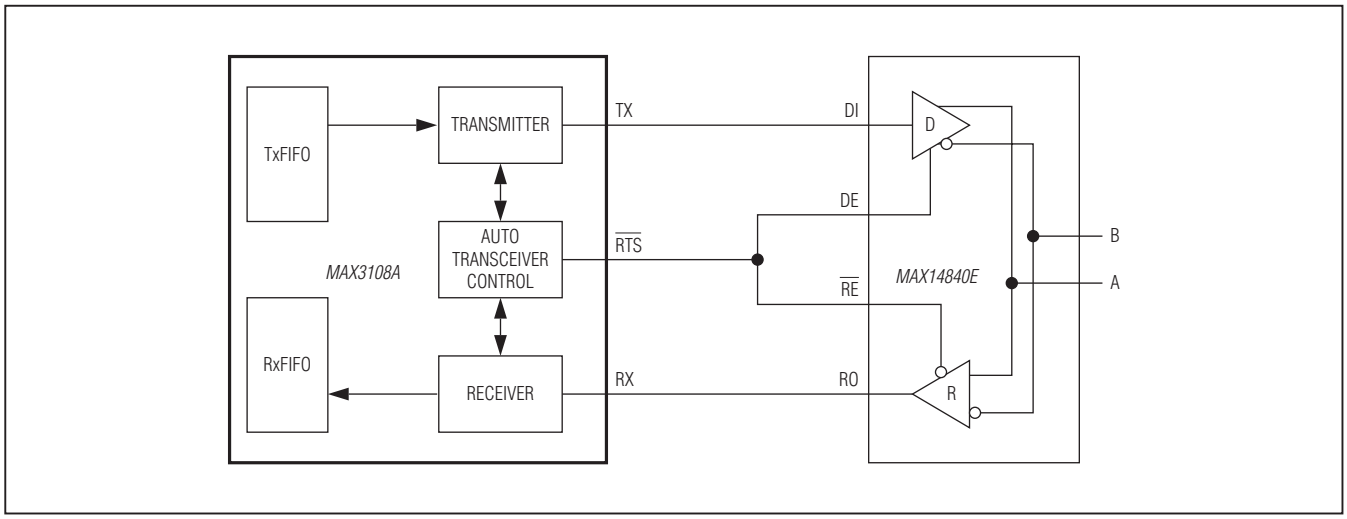



*Figure 9. Auto Transceiver Direction Control*

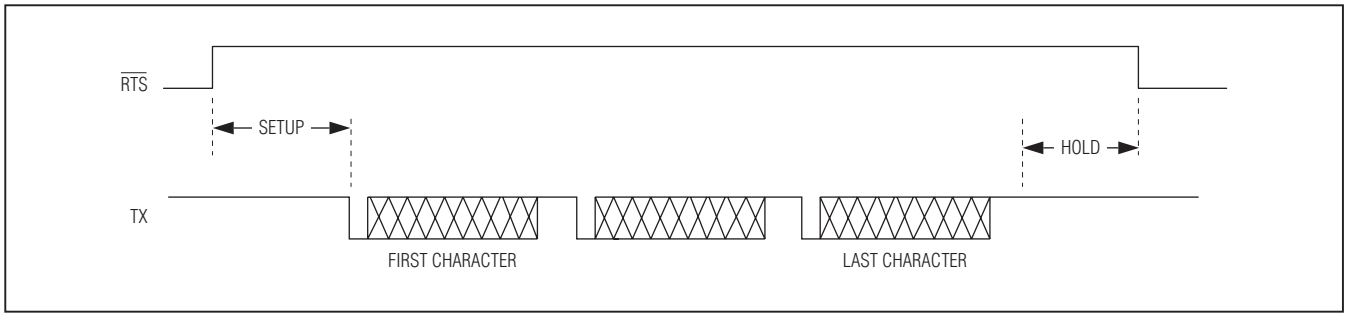



*Figure 10. Setup and Hold times in Auto Transceiver Direction Control*

### Echo Suppression

The device can suppress echoed data that is sometimes found in half-duplex communication networks, such as RS-485 and IrDA. If the transceiver's receiver is not turned off while the transceiver is transmitting, copies (echoes) of the transmitted data are received by the UART. The device's receiver can block the reception of this echoed data by enabling echo suppression. Figure 11 shows a typical RS-485 application using the echo suppression feature. Set the **MODE2**[7]: EchoSuprs bit high to enable echo suppression.

The device can also block echoes with a long round trip delay by disabling the transceiver's receiver with the $\overline{RTS}$ output while the device is transmitting. The transmitter can be configured to remain enabled after the end of the transmission for a programmable period of time called the hold time delay (Figure 12). The hold time delay is set by the **HDplxDelay**[3:0] bits. See the **HDplxDelay** description in the *Detailed Register Descriptions* section for more information.

Echo suppression can operate simultaneously with auto transceiver direction control.

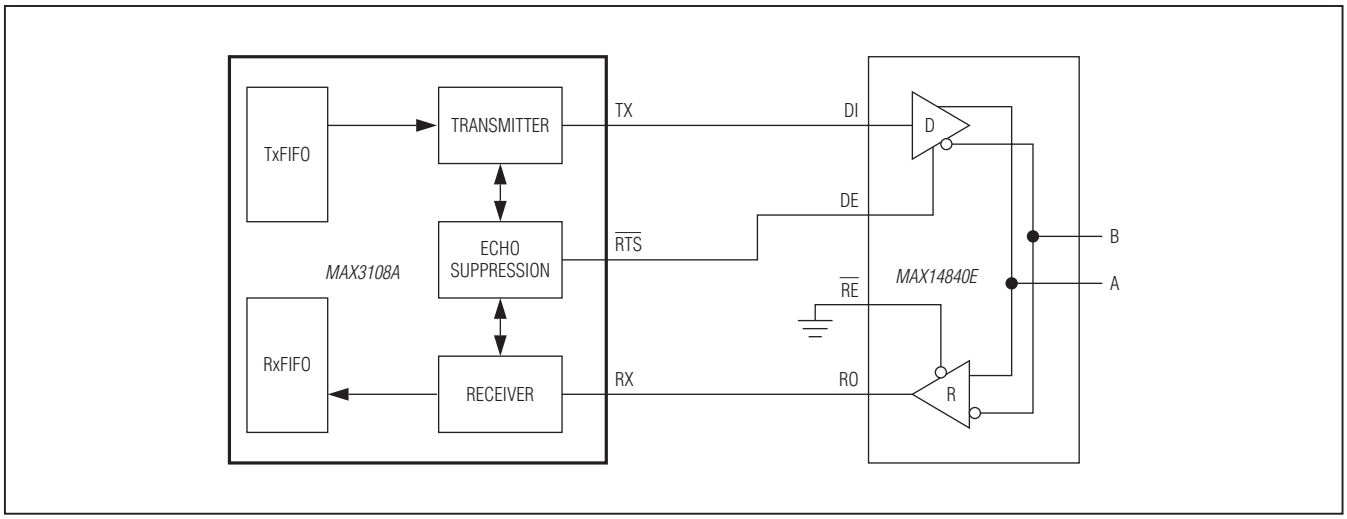



*Figure 11. Half-Duplex with Echo Suppression*

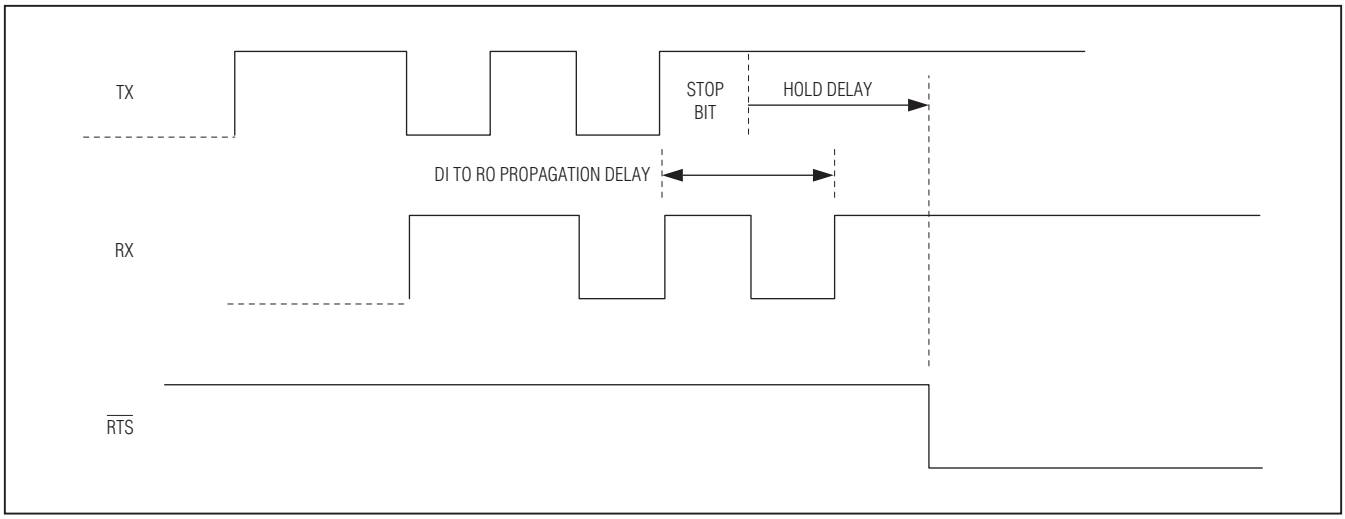



*Figure 12. Echo Suppression Timing*

### Auto Hardware Flow Control

The device is capable of auto hardware ($\overline{RTS}$ and $\overline{CTS}$) flow control without the need for host processor intervention. When AutoRTS control is enabled, the device automatically controls the $\overline{RTS}$ handshake without the need for host processor intervention. AutoCTS flow control separately turns the device's transmitter on and off based on the $\overline{CTS}$ input. AutoRTS and AutoCTS flow control modes are independently enabled by the **FlowCtrl**[1:0] bits.

#### AutoRTS Control

AutoRTS flow control ensures that the receive FIFO does not overflow by signaling to the far-end UART to stop data transmission. The device does this automatically by controlling the $\overline{RTS}$ output. AutoRTS flow control is enabled by setting the **FlowCtrl**[0]: AutoRTS bit high. The HALT and RESUME programmable values determine the threshold RxFIFO fill levels at which $\overline{RTS}$ is asserted and deasserted. Set the HALT and RESUME levels in the **FlowLvl** register. With differing HALT and RESUME levels, hysteresis of the RxFIFO level can be defined for $\overline{RTS}$ transitions.

When the RxFIFO is filled to a level higher than the HALT level, the device deasserts $\overline{RTS}$ and stops the far-end UART from transmitting any additional data. $\overline{RTS}$ remains deasserted until the RxFIFO is emptied enough so that the number of words falls to below the RESUME level.

Interrupts are not generated when the HALT and RESUME levels are reached. This allows the host controller to be completely disengaged from $\overline{RTS}$ flow control management.

#### AutoCTS Control

When AutoCTS flow control is enabled, the UART automatically starts transmitting data when the $\overline{CTS}$ input is logic-low and stops transmitting data when $\overline{CTS}$ is logic-high. This frees the host processor from managing this time-critical flow-control task. AutoCTS flow control is enabled by setting the **FlowCtrl**[1]: AutoCTS bit high. The **ISR**[7]: CTSInt interrupt works normally during AutoCTS flow control. Set the **IRQEn**[7]: CTSIntEn bit low to disable routing of $\overline{CTS}$ interrupts to $\overline{IRQ}$ and ensure that the host does not receive interrupts from $\overline{CTS}$ transitions. If $\overline{CTS}$ transitions from low to high during transmission of a data word, the device completes the transmission of the current word and halts transmission afterwards.

Turn the transmitter off by setting the **MODE1**[1]: TxDisabl bit high before enabling AutoCTS control.

### Auto Software (XON/XOFF) Flow Control

When auto software flow control is enabled, the device recognizes and/or sends predefined XON/XOFF characters to control the flow of data across the asynchronous serial link. The XON character signifies that there is enough room in the receive FIFO and transmission of data should continue. The XOFF character signifies that the receive FIFO is nearing overflow and that the transmission of data should stop. Auto software flow control works autonomously and does not require host intervention, similar to auto hardware flow control. To reduce the chance of receiving corrupted data that equals a single-byte XON or XOFF character, the device allows for double-wide (16-bit) XON/XOFF characters. The XON and XOFF characters are programmed into the **XON1**, **XON2** and **XOFF1**, **XOFF2** registers.

The **FlowCtrl**[7:3] bits are used for enabling and configuring auto software flow control. An interrupt is generated in **ISR**[1]: SpCharInt whenever an XON or XOFF character is received and details are stored in the **SpclCharInt** register. Set the **IRQEn**[1]: SpclChrIEn bit low to disable routing of the interrupt to $\overline{IRQ}$.

Software flow control consists of transmit flow control and receive flow control, which operate independently of each other.

#### Receiver Flow Control

When auto receive flow control is enabled by the **FlowCtrl**[7:6] bits, the device automatically controls the transmission of data by the far-end UART by sending XOFF and XON control characters. The HALT and RESUME levels determine the threshold RxFIFO fill levels at which the XOFF and XON characters are sent. HALT and RESUME are programmed in the **FlowLvl** register. With differing HALT and RESUME levels, hysteresis can be defined in the RxFIFO fill level for the receiver flow control activity.

When the RxFIFO is filled to a level higher than the HALT level, the device sends an XOFF character to stop data transmission. An XON character is sent when the RxFIFO is emptied enough so that the number of words falls to below the RESUME level.

If double-wide (16-bit) XON/XOFF characters are selected by setting the **FlowCtrl**[7:6] bits to 11, then **XON1**/**XOFF1** are transmitted before **XON2**/**XOFF2** whenever a control character is transmitted.

#### Transmitter Flow Control

If auto transmit control is enabled by the **FlowCtrl**[5:4] bits, the receiver compares all received words with the

XOFF and XON characters. When an XOFF character is received, the device halts the transmitter from sending further data following any currently transmitting word. The receiver is not affected and continues receiving. Upon receiving an XON character, the transmitter restarts sending data. The received XON and XOFF characters are filtered out and are not stored into the receive FIFO. An interrupt is not generated.

If double-wide (16-bit) XON/XOFF characters are selected by setting the **FlowCtrl**[5:4] bits to 11, then a character matching **XON1**/**XOFF1** must be received before a character matching **XON2**/**XOFF2** to be interpreted as a control character.

Turn the transmitter off by setting the **MODE1**[1]: TxDisabl bit high before enabling software transmitter flow control.

### FIFO Interrupt Triggering

Receive and transmit FIFO fill-dependent interrupts are generated if FIFO trigger levels are defined. When the number of words in the FIFOs reach or exceed a trigger level programmed in the **FIFOTrgLvl** register, an interrupt is generated in **ISR**[3] or **ISR**[4]. The interrupt trigger levels operate independently from the HALT and RESUME flow control levels in AutoRTS or auto software flow control modes.

The FIFO interrupt triggering can be used, for example, for a block data transfer. The trigger level interrupt gives the host an indication that a given block size of data is available for reading in the receive FIFO or available for transfer to the transmit FIFO. If the HALT and RESUME levels are outside of this range, then the UART continues to transmit or receive data during the block read/write operations for uninterrupted data transmission on the bus.

### Low-Power Standby Modes

The device has sleep and shutdown modes that reduce power consumption during periods of inactivity. In both sleep and shutdown modes, the UART disables specific functional blocks to reduce power consumption.

After sleep or shutdown mode is exited, the internal clock starts up and a period of time is needed for clock stabilization. The **STSInt**[5]: ClkReady bit indicates when the clocks are stable. When an external clock source is used, the ClkReady bit does not indicate clock stability.

#### Forced-Sleep Mode

In forced-sleep mode, all UART-related on-chip clocking is stopped. The following blocks are inactive: the crystal oscillator, the PLL, the predivider, the receiver, and the transmitter. The I2C/SPI interface and the registers remain active and the host controller can access them. To force the device to enter sleep mode, set the **MODE1**[5]: ForcedSleep bit high. To exit forced-sleep mode, set the ForcedSleep bit low.

#### Auto-Sleep Mode

The device can be configured to operate in auto-sleep mode by setting the **MODE1**[6]: AutoSleep bit high. In auto-sleep mode, the device automatically enters sleep mode when all the following conditions are met:

- Both FIFOs are empty.
- There are no pending $\overline{IRQ}$ interrupts.
- There is no activity on any input pins for a period equal to 65,536 UART character lengths.

The same blocks are inactive when the UART is in auto-sleep mode as in forced-sleep mode.

The device exits auto-sleep mode as soon as activity is detected on any of the GPIO_, RX, or $\overline{CTS}$ inputs.

To manually exit auto-sleep mode, set the **MODE1**[6]: AutoSleep bit low.

#### Shutdown Mode

Drive the $\overline{RST}$ input to logic-low to enter shutdown mode. Shutdown mode consumes less than 1µA. In shutdown mode, all the device's circuitry is completely off. This includes the I2C/SPI interface, the registers, the FIFOs, and the clocking circuitry.

When the $\overline{RST}$ input transitions from low to high, the device exits shutdown mode and a hardware reset is initiated. The chip initialization is complete when the controller is able to read out known register contents from the device. This could, for example, be the DIVLSB register.

The device needs to be reprogrammed following a shutdown.

### Power-Up and $\overline{IRQ}$

The $\overline{IRQ}$ output only operates when all supplies are active. $\overline{IRQ}$ operates as a hardware active-low interrupt output; $\overline{IRQ}$ is asserted when an interrupt is pending. An $\overline{IRQ}$ interrupt is only possible during normal operation if at least one of the interrupt enable bits in the IRQEn register is set.

In polled mode, any register with a known reset value can be polled to check whether the device is ready for operation. If the controller gets a valid response from the polled register, then the device is ready for operation.

### Interrupt Structure

Figure 13 shows the structure of the interrupt. There are four interrupt source registers: **ISR**, **LSR**, **STSInt**, and **SpclCharInt**. The interrupt sources are divided into top-level and low-level interrupts. The top-level interrupts typically occur more often and can be read out by the host controller directly through **ISR**. The low-level interrupts typically occur less often and their specific source can be read out by the host controller through **LSR**, **STSInt**, or **SpclCharInt**. The three LSBs of **ISR** point to the low-level interrupt registers that contain the details of the interrupt source.

### Interrupt Enabling

Every interrupt bit of the four interrupt registers can be enabled or masked through an associated interrupt enable register bit. These are the **IRQEn**, **LSRIntEn**, **SpclChrIntEn**, and **STSIntEn** registers. By default, all interrupts are masked.

### Interrupt Clearing

When an interrupt is pending (i.e., $\overline{IRQ}$ is asserted) and **ISR** is read, both the **ISR** bits are cleared and the $\overline{IRQ}$ output is deasserted. Low-level interrupt information does not reassert $\overline{IRQ}$ for the same interrupt, but remains stored in the low-level interrupt registers until each is separately cleared. **SpclCharInt** and **STSInt** are clear-on-read (COR). The **LSR** bits are only cleared when the source of the interrupt is removed, not when **LSR** is read.

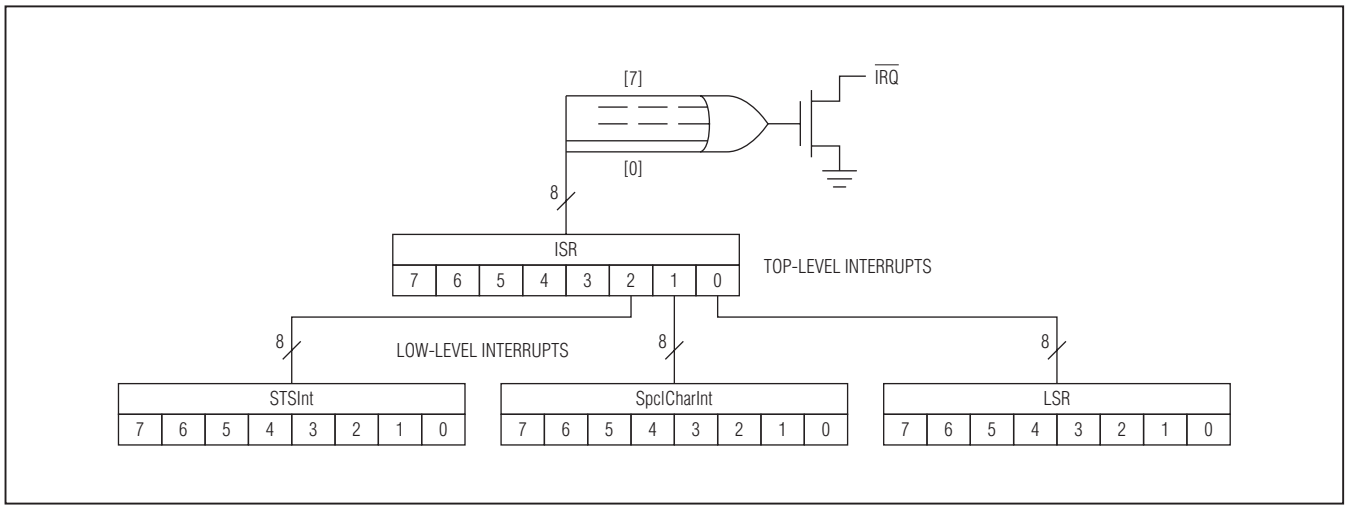



*Figure 13. Simplified Interrupt Structure*

## Register Map

(**Note:** All default reset values are 0x00, unless otherwise noted. All registers are R/W, unless otherwise noted.)

| REGISTER | ADDR | BIT 7 | BIT 6 | BIT 5 | BIT 4 | BIT 3 | BIT 2 | BIT 1 | BIT 0 |
|---|---|---|---|---|---|---|---|---|---|
| **FIFO DATA** | | | | | | | | | |
| RHR[1] | 0x00 | RData7 | RData6 | RData5 | RData4 | RData3 | RData2 | RData1 | RData0 |
| THR[1] | 0x00 | TData7 | TData6 | TData5 | TData4 | TData3 | TData2 | TData1 | TData0 |
| **INTERRUPTS** | | | | | | | | | |
| IRQEn | 0x01 | CTSIEn | RxEmtyIEn | TFifoEmtyIEn | TxTrgIEn | RxTrgIEn | STSIEn | SpChrIEn | LSRErrIEn |
| ISR[1, 2] | 0x02 | CTSInt | RFifoEmptyInt | TFifoEmptyInt | TxTrgInt | RxTrigInt | STSInt | SpCharInt | LSRErrInt |
| LSRIntEn | 0x03 | — | — | NoiseIntEn | RBreakIEn | FrameErrIEn | ParityIEn | ROverrIEn | RTimoutIEn |
| LSR[1] | 0x04 | $\overline{\text{CTS}}$bit | — | RxNoise | RxBreak | FrameErr | RxParityErr | RxOverrun | RTimeout |
| SpclChrIntEn | 0x05 | — | — | MltDrpIntEn | BREAKIntEn | XOFF2IntEn | XOFF1IntEn | XON2IntEn | XON1IntEn |
| SpclCharInt[1] | 0x06 | — | — | MultiDropInt | BREAKInt | XOFF2Int | XOFF1Int | XON2Int | XON1Int |
| STSIntEn | 0x07 | TxEmptyIntEn | SleepIntEn | ClkRdyIntEn | — | GPI3IntEn | GPI2IntEn | GPI1IntEn | GPI0IntEn |
| STSInt[1] | 0x08 | TxEmptyInt | SleepInt | ClkReady | — | GPI3Int | GPI2Int | GPI1Int | GPI0Int |
| **UART MODES** | | | | | | | | | |
| MODE1 | 0x09 | — | AutoSleep | ForcedSleep | TrnscvCtrl | RTSHiZ | TXHiZ | TxDisabl | RxDisabl |
| MODE2 | 0x0A | EchoSuprs | MultiDrop | Loopback | SpecialChr | RFifoEmptyInv | RxTrgInv | FIFORst | RST |
| LCR[2] | 0x0B | $\overline{\text{RTS}}$bit | TxBreak | ForceParity | EvenParity | ParityEn | StopBits | Length1 | Length0 |
| RxTimeOut | 0x0C | TimOut7 | TimOut6 | TimOut5 | TimOut4 | TimOut3 | TimOut2 | TimOut1 | TimOut0 |
| HDplxDelay | 0x0D | Setup3 | Setup2 | Setup1 | Setup0 | Hold3 | Hold2 | Hold1 | Hold0 |
| IrDA | 0x0E | — | — | TxInv | RxInv | MIR | — | SIR | IrDAEn |
| **FIFOs CONTROL** | | | | | | | | | |
| FlowLvl | 0x0F | Resume3 | Resume2 | Resume1 | Resume0 | Halt3 | Halt2 | Halt1 | Halt0 |
| FIFOTrgLvl[2] | 0x10 | RxTrig3 | RxTrig2 | RxTrig1 | RxTrig0 | TxTrig3 | TxTrig2 | TxTrig1 | TxTrig0 |
| TxFIFOLvl[1] | 0x11 | TxFL7 | TxFL6 | TxFL5 | TxFL4 | TxFL3 | TxFL2 | TxFL1 | TxFL0 |
| RxFIFOLvl[1] | 0x12 | RxFL7 | RxFL6 | RxFL5 | RxFL4 | RxFL3 | RxFL2 | RxFL1 | RxFL0 |
| **FLOW CONTROL** | | | | | | | | | |
| FlowCtrl | 0x13 | SwFlow3 | SwFlow2 | SwFlow1 | SwFlow0 | SwFlowEn | GPIAddr | AutoCTS | AutoRTS |
| XON1 | 0x14 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
| XON2 | 0x15 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
| XOFF1 | 0x16 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
| XOFF2 | 0x17 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
| **GPIOs** | | | | | | | | | |
| GPIOConfig | 0x18 | GP3OD | GP2OD | GP1OD | GP0OD | GP3Out | GP2Out | GP1Out | GP0Out |
| GPIOData | 0x19 | GPI3Dat | GPI2Dat | GPI1Dat | GPI0Dat | GPO3Dat | GPO2Dat | GPO1Dat | GPO0Dat |
| **CLOCK CONFIGURATION** | | | | | | | | | |
| PLLConfig[2] | 0x1A | PLLFactor1 | PLLFactor0 | PreDiv5 | PreDiv4 | PreDiv3 | PreDiv2 | PreDiv1 | PreDiv0 |
| BRGConfig | 0x1B | — | — | 4xMode | 2xMode | FRACT3 | FRACT2 | FRACT1 | FRACT0 |
| DIVLSB[2] | 0x1C | Div7 | Div6 | Div5 | Div4 | Div3 | Div2 | Div1 | Div0 |
| DIVMSB | 0x1D | Div15 | Div14 | Div13 | Div12 | Div11 | Div10 | Div9 | Div8 |
| CLKSource[2] | 0x1E | CLKtoRTS | — | — | — | PLLBypass | PLLEn | CystalEn | — |

[1] *Denotes nonread/write mode: RHR = R, THR = W, ISR = COR, LSR = R, SpclCharInt = COR, STSInt = R/COR, TxFIFOLvl = R, RxFIFOLvl = R.*

[2] *Denotes nonzero default reset value: ISR = 0x60, LCR = 0x05, FIFOTrgLvl = 0xFF, PLLConfig = 0x01, DIVLSB = 0x01, CLKSource = 0x18.*