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

MAX3109

Dual Serial UART with 128-Word FIFOs

General Description

The MAX3109 advanced dual universal asynchronous receiver-transmitter (UART) has 128 words of receive and transmit first-in/first-out (FIFO) and a high-speed SPI or I²C controller interface. The 2x and 4x rate modes allow a maximum of 24Mbps data rates. A phase-locked loop (PLL) and the fractional baud-rate generators allow a high degree of flexibility in baud-rate programming and reference clock selection.

Independent logic-level translation on the transceiver and controller interfaces allows ease of interfacing to microcontrollers, FPGAs, and transceivers that are powered by differing supply voltages. Automatic hardware and software flow control with selectable FIFO interrupt triggering offloads low-level activity from the host controller. Automatic half-duplex transceiver control with programmable setup and hold times allow the MAX3109 to be used in high-speed applications such as PROFIBUS-DP. The 128-word FIFOs have advanced FIFO control, reducing host processor data flow management.

The MAX3109 is available in a 32-pin TQFN (5mm x 5mm) package and is specified over the -40°C to +85°C extended temperature range.

Applications

- Handheld Devices
- Power Meters
- Programmable Logic Controllers (PLCs)
- Medical Systems
- Automotive Infotainment Systems
- Point-of-Sales Systems
- HVAC or Building Control

Features

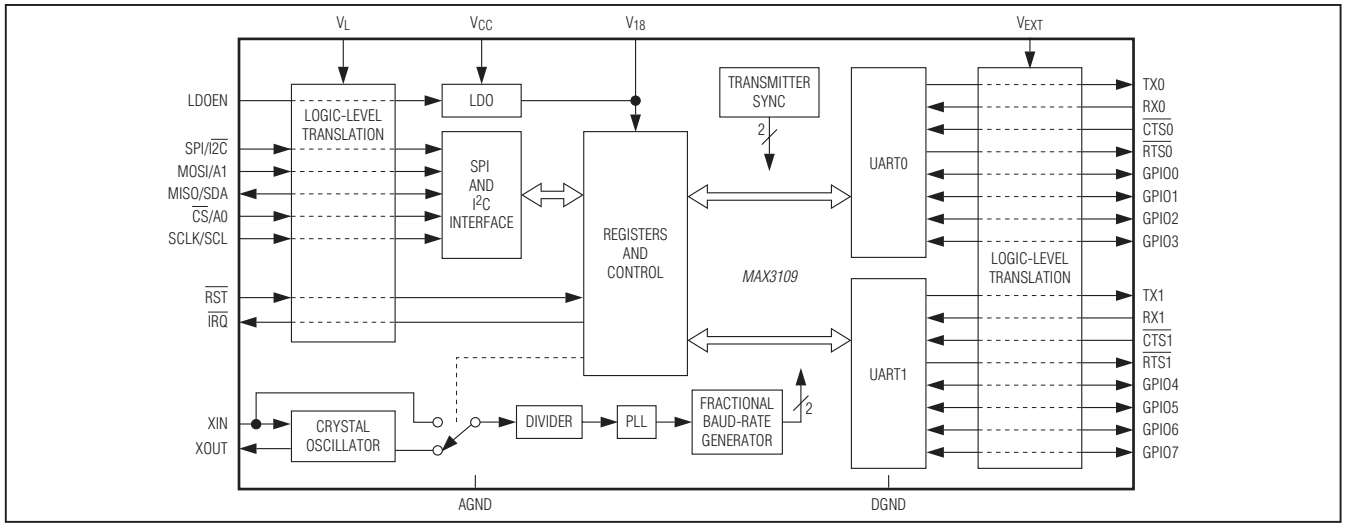
- **24Mbps (max) Baud Rate**
- **Integrated PLL and Divider**
- **1.71V to 3.6V Supply Range**
- **High-Resolution Programmable Baud Rate**
- **SPI Up to 26MHz Clock Rate**
- **Fast Mode Plus I²C Up to 1MHz**
- **Automatic $\overline{RTS}$_ and $\overline{CTS}$_ Flow Control**
- **Automatic XON/XOFF 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**
- **Line Noise Indication**
- **1µA Shutdown Current**
- **Two Timers Routed to GPIOs**
- **8 Flexible GPIOs with 20mA Drive Capability**
- **Register Compatible with MAX3107, MAX3108, MAX14830**
- **Small TQFN (5mm x 5mm) Package**

Ordering Information

PART	TEMP RANGE	PIN-PACKAGE
MAX3109ETJ+	-40°C to +85°C	32 TQFN-EP*

+Denotes a lead(Pb)-free/RoHS-compliant package.
**EP = Exposed pad.*

Functional Diagram



IrDA is a service mark of Infrared Data Association Corporation.

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

19-5806; Rev 2; 10/12

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

LIST OF FIGURES

LIST OF TABLES

LIST OF REGISTERS

LIST OF REGISTERS (continued)

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_, $\overline{CTS\_}$, GPIO_ ..... -0.3V to (V_{EXT} + 0.3V)
DGND ..... -0.3V to +0.3V

Continuous Power Dissipation (T_A = +70°C)
TQFN (derate 34.5mW/°C above +70°C) 2758.6mW
Operating Temperature Range ..... -40°C to +85°C
Maximum Junction Temperature ..... +150°C
Storage Temperature Range ..... -65°C to +150°C
Lead Temperature (soldering, 10s) ..... +300°C
Soldering Temperature (reflow) ..... +260°C

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.

PACKAGE THERMAL CHARACTERISTICS (Note 1)

TQFN
Junction-to-Ambient Thermal Resistance (θ_{JA}) ..... 47°C/W
Junction-to-Case Thermal Resistance (θ_{JC}) ..... 1.7°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**.

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, all UARTs in loopback mode, LDOEN = low			500	
V_{18} Input Power-Supply Current in Shutdown Mode	I_{18SHDN}	$\overline{RST}$ = low, all inputs and outputs are idle			100	µA
V_{CC} + V_L + V_A Shutdown Supply Current	I_{SHDN}	$\overline{RST}$ = low, MISO, SCLK, MOSI, SPI_I2C, $\overline{CS}$, LDOEN = 0/V_L, CTSB0/1 = 0/V_{EXT}, CTSB0/1 = 0/V_{EXT}		0	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
V_{18} Input Power-Supply Current	I_{18}	Baud rate = 1Mbps, 20MHz external clock, PLL disabled, UART in loopback mode, LDOEN = low (Note 4)			4	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 Low Voltage in SPI Mode	V_{OLSPI}	Sink current = 2mA			0.4	V
MISO/SDA Output 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-Z 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} - 0.4			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.) (Notes 2, 3)

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}	(Note 5)			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 + 0.1C_B		1000	ns
		Fast mode (0.3 x V_L to 0.7 x V_L) (Note 6)	20 + 0.1C_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 + 0.1C_B		1000	ns
		Fast mode (0.3 x V_L to 0.7 x V_L) (Note 6)	20 + 0.1C_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.) (Notes 2, 3)

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 negative and currents exiting the IC are positive.
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

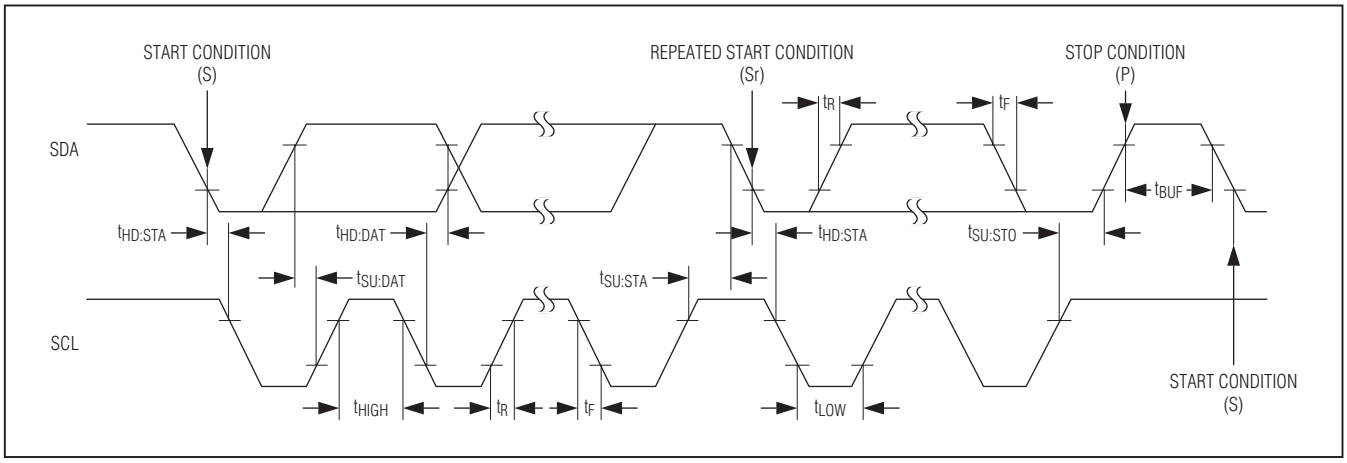


Figure 1. I²C Timing Diagram

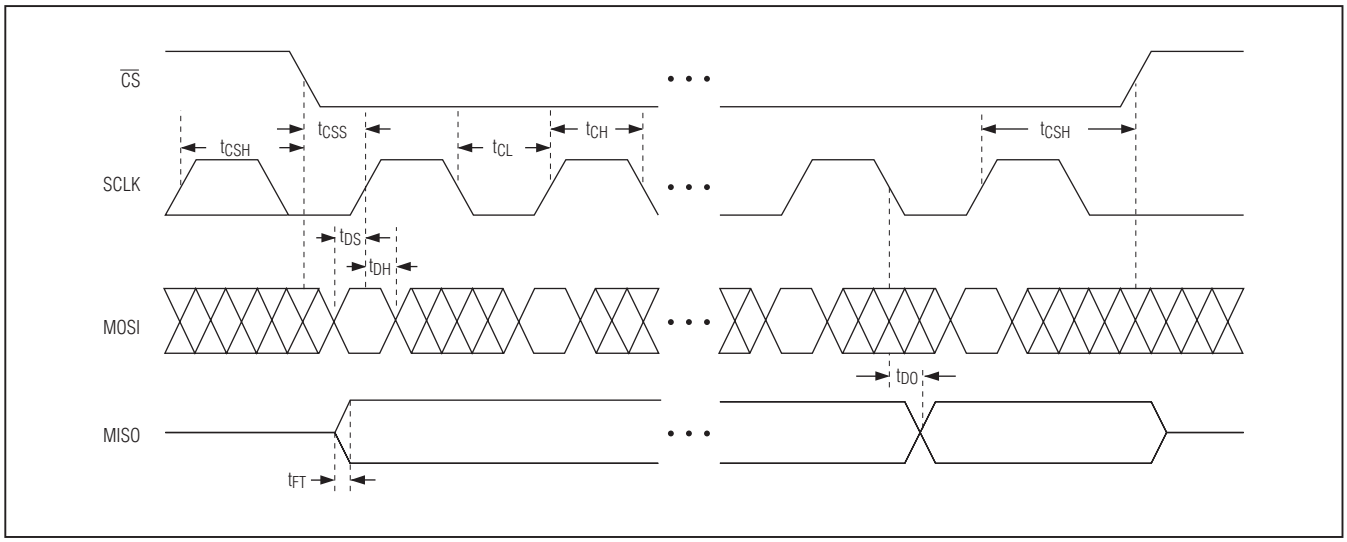
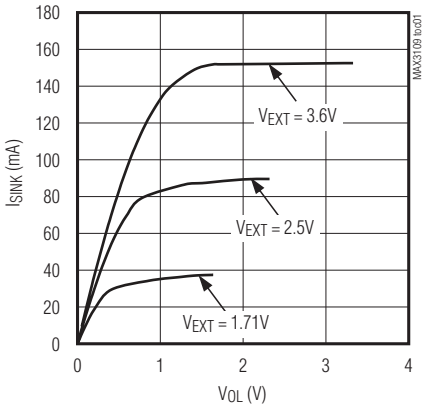


Figure 2. SPI Timing Diagram

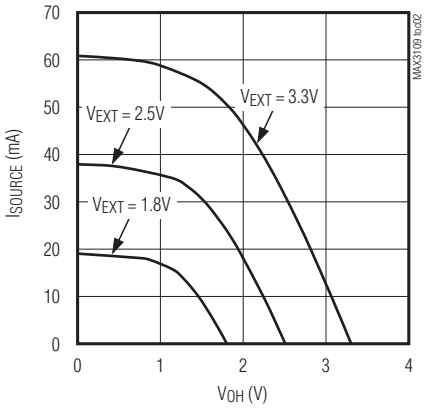
Typical Operating Characteristics

(V_{CC} = 2.5V, V_L = 2.5V, V_{EXT} = 2.5V, V_{LDOEN} = V_L, UART1 in sleep mode, T_A = +25°C unless otherwise noted.)

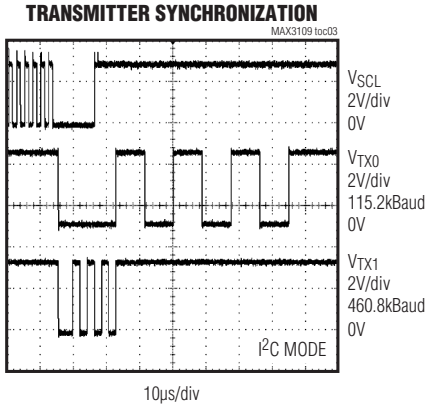
SINK CURRENT (OPEN DRAIN) vs. GPIO_ OUTPUT LOW VOLTAGE



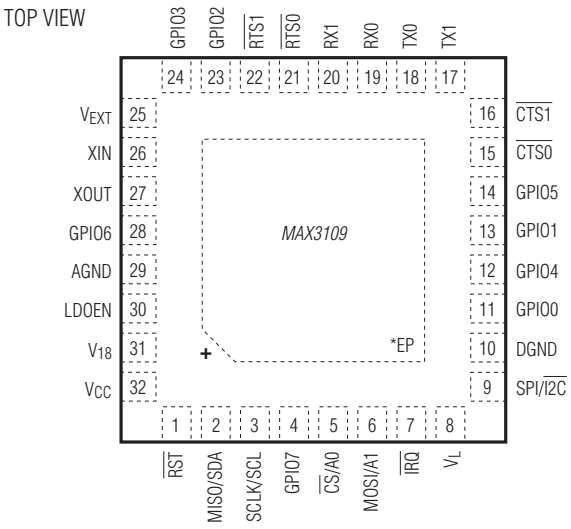
SOURCE CURRENT (PUSH-PULL) vs. GPIO_OUTPUT HIGH VOLTAGE



TRANSMITTER SYNCHRONIZATION



Pin Configuration



TQFN
(5mm × 5mm)

*CONNECT EP TO AGND.

Pin Description

PIN	NAME	FUNCTION
1	$\overline{RST}$	Active-Low Reset Input. Drive $\overline{RST}$ low to force all of the UARTs into hardware reset mode. Driving $\overline{RST}$ low also enables low-power shutdown mode. When $\overline{RST}$ is low, the internal V18 LDO is switched off, even if the LDOEN input is kept high.
2	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, I²C serial-data input/output. MISO/SDA is high impedance when $\overline{RST}$ is driven low or when the externally supplied V18 is powered off.
3	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).
4	GPIO7	General-Purpose Input/Output 7. GPIO7 is user-programmable as an input or output (push-pull or open drain) or an external event-driven interrupt source. GPIO7 has a weak pulldown resistor to DGND when configured as an input.
5	$\overline{CS}$/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 I²C device address programming input. Connect $\overline{CS}$/A0 to DGND, V_L, SCL, or SDA when SPI/$\overline{I2C}$ is low.
6	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.
7	$\overline{IRQ}$	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.

Pin Description (continued)

PIN	NAME	FUNCTION
8	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.
9	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.
10	DGND	Digital Ground
11	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. GPIO0 is the reference clock output when bit 7 of the TxSynch register is set to high (see the *UART Clock to GPIO* section for more information).
12	GPIO4	General-Purpose Input/Output 4. GPIO4 is user-programmable as an input or output (push-pull or open drain) or an external event-driven interrupt source. GPIO4 has a weak pulldown resistor to DGND when configured as an input. GPIO4 is the reference clock output when bit 7 of the TxSynch register is set to high (see the *UART Clock to GPIO* section for more information).
13	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. GPIO1 is the TIMER output when bit 7 of the TIMER2 register is set high.
14	GPIO5	General-Purpose Input/Output 5. GPIO5 is user-programmable as an input or output (push-pull or open drain) or an external event-driven interrupt source. GPIO5 has a weak pulldown resistor to DGND when configured as an input. GPIO5 is the TIMER output when bit 7 of the TIMER2 register is set high.
15	$\overline{CTS0}$	Active-Low Clear-to-Send Input for UART0. $\overline{CTS0}$ is a flow-control status input.
16	$\overline{CTS1}$	Active-Low Clear-to-Send Input for UART1. $\overline{CTS1}$ is a flow-control status input.
17	TX1	Serial Transmitting Data Output for UART1. TX1 is logic-high when $\overline{RST}$ is low or when the externally supplied V18 is not powered.
18	TX0	Serial Transmitting Data Output for UART0. TX0 is logic-high when $\overline{RST}$ is low or when the externally supplied V18 is not powered.
19	RX0	Serial Receiving Data Input for UART0. RX0 has an internal weak pullup resistor to V_{EXT}.
20	RX1	Serial Receiving Data Input for UART1. RX1 has an internal weak pullup resistor to V_{EXT}.
21	$\overline{RTS0}$	Active-Low Request-to-Send Output for UART0. $\overline{RTS0}$ can be set high or low by programming the LCR register. $\overline{RTS0}$ is the UART system clock/fractional divider output when bit 7 of the CLKSource register is set high. $\overline{RTS0}$ is logic-high when $\overline{RST}$ is low or when the externally supplied V18 is not powered.
22	$\overline{RTS1}$	Active-Low Request-to-Send Output for UART1. $\overline{RTS1}$ can be set high or low by programming the LCR register. $\overline{RTS1}$ is the UART system clock/fractional divider output when bit 7 of the CLKSource register is set high. $\overline{RTS1}$ is logic-high when $\overline{RST}$ is low or when the externally supplied V18 is not powered.
23	GPIO2	General-Purpose Input/Output 2. GPIO2 is user-programmable as 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.
24	GPIO3	General-Purpose Input/Output 3. GPIO3 is user-programmable as 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.
25	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.
26	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.

Pin Description (continued)

PIN	NAME	FUNCTION
27	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.
28	GPIO6	General-Purpose Input/Output 6. GPIO6 is user-programmable as input or output (push-pull or open drain) or an external event-driven interrupt source. GPIO6 has a weak pulldown resistor to DGND when configured as an input.
29	AGND	Analog Ground
30	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.
31	V_{18}	Internal 1.8V LDO Output and 1.8V Power-Supply Input. Bypass V_{18} with a 1µF ceramic capacitor to DGND.
32	V_{CC}	Analog Power Supply. V_{CC} powers the PLL and internal LDO. Bypass V_{CC} with a 0.1µF ceramic capacitor to AGND.
—	EP	Exposed Pad. Connect EP to AGND. Do not use EP as the main AGND connection.

Detailed Description

The MAX3109 dual universal asynchronous receiver-transmitter (UART) bridges an SPI/MICROWIRE™ or I²C microprocessor bus to an asynchronous serial-data communication link, such as RS-485, RS-232, or IrDA. The MAX3109 is configured through 8-bit registers, which are accessed through the SPI or I²C interface. These 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 FIFOs, formatted, and sent out at TX_. The MAX3109 adds START, STOP, and parity bits to the data before transmitting the data out at the selected baud rate. The clock configuration registers determine the baud rates, clock source selection, clock frequency prescaling, and fractional baud-rate generator settings for each UART.

The MAX3109 receivers detect 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 FIFOs and can then be read out by the host controller through the Receiver Hold register (**RHR**).

The device features two identical UARTs that are completely independent except for the input clock. Text in this data sheet references individual UART operation, unless otherwise noted.

The MAX3109's register set is compatible with the MAX3107. Refer to Application Note 4938: *Differences Between Maxim's Advanced UART Devices* for information on how to transfer firmware from the MAX3107 to the MAX3109.

Receive and Transmit FIFOs

Each UART's receiver and transmitter has a 128-word-deep FIFOs, 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.

MICROWIRE is a trademark of National Semiconductor Corp.

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 read out by the host controller through the **TxFIFOLvl** register. The transmit FIFO fill level can be programmed to generate an interrupt when greater than or equal to 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 to at least the programmed trigger level, an interrupt is generated in **ISR**[4]: TxTrigInt.

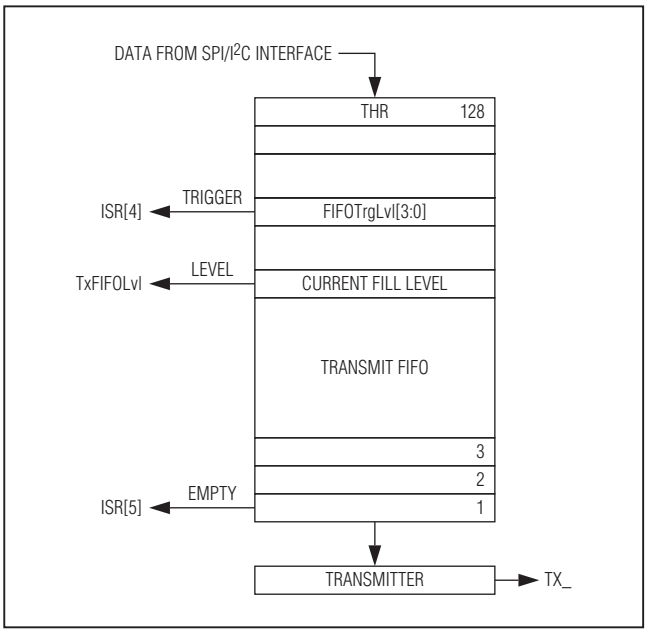


Figure 3. Transmit FIFO Signals

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 (RxInv = 0). The 8-bit data word 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. After a word is read out of **RHR**, **LSR** contains the status information for that word.

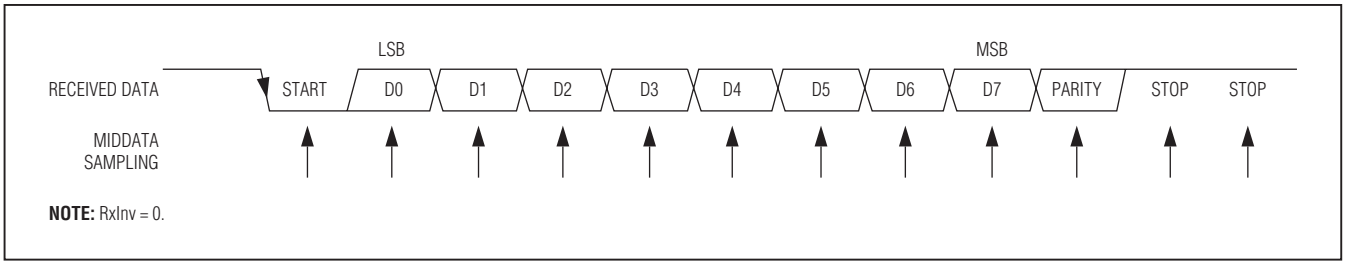


Figure 4. Receive Data Format

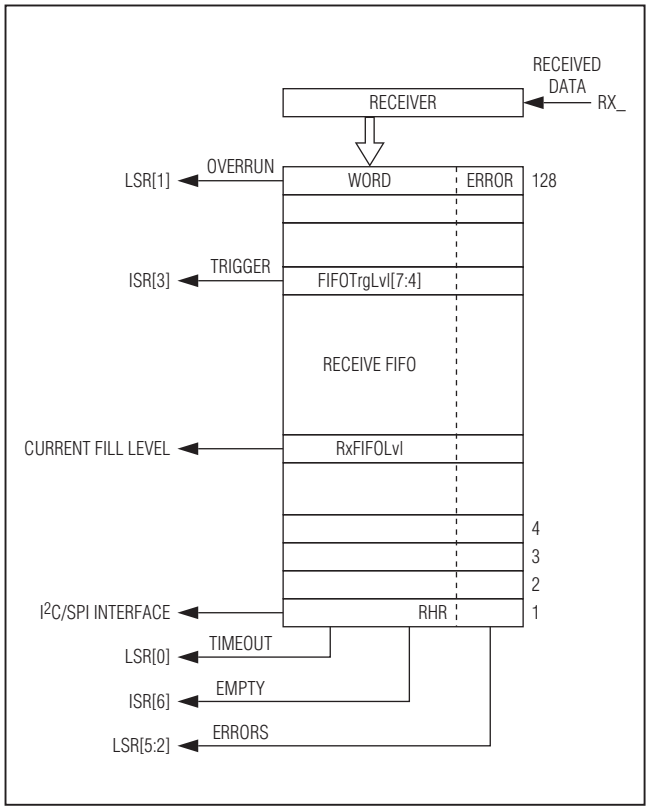


Figure 5. Receive FIFO

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 MAX3109 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 MAX3109 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.

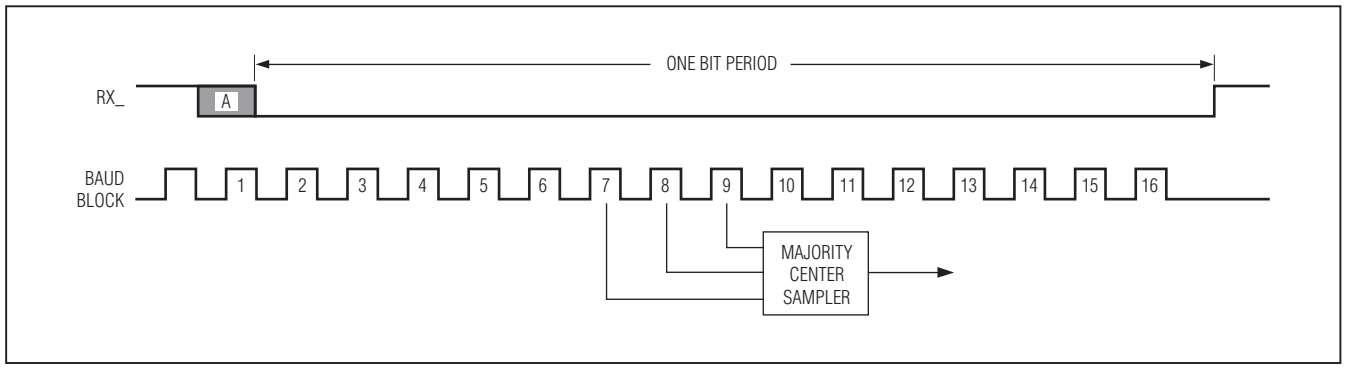


Figure 6. Midbit Sampling

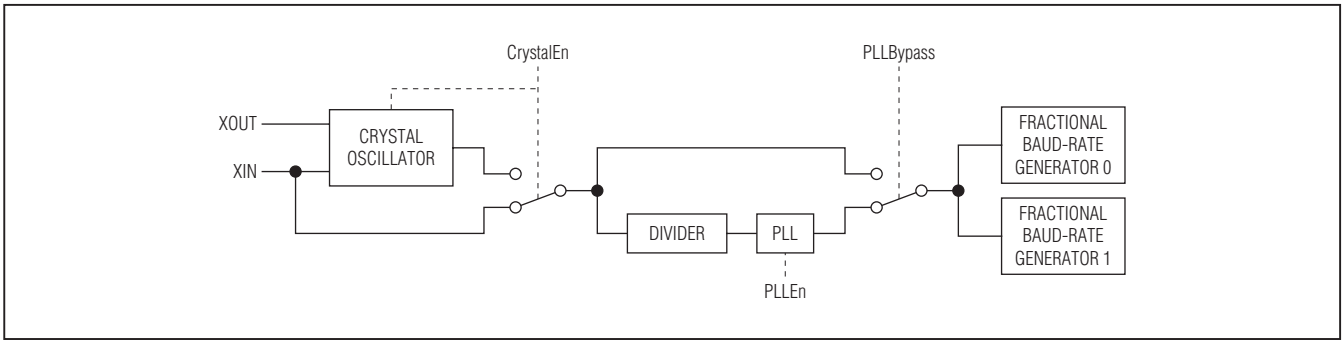


Figure 7. Clock Selection Diagram

Clock Selection

The MAX3109 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 MAX3109 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.

Each UART baud rate can be individually programmed and both share the same reference clock input.

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 MAX3109 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 Generators

Each UART has an internal fractional baud-rate generator that 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:

$$DIV = TRUNC(D)$$

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 reference input frequency of 28.23MHz and 1x (default) rate mode.

The ideal divisor is calculated as:

$$D = 28{,}230{,}000/(16 \times 190{,}000) = 9.286$$

hence DIV = 9.

$$FRACT = ROUND(16 \times 0.286) = 5$$

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) = 189463 baud.

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

2x and 4x Rate Modes

To support higher baud rates than possible with standard operation using 16x sampling, the MAX3109 offers 2x and 4x rate modes. 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).

Low-Frequency Timer

Each UART has a general-purpose timer that can be used to generate a low-frequency clock at a GPIO output and can, for example, be used to drive external LEDs. The low-frequency clock is a divided replica of the given UART baud-rate clock. The timer for each UART is internally routed to the respective GPIO_ output when enabled by the **TIMER2** register as follows:

- UART0: GPIO1
- UART1: GPIO5

The clock pulses at the GPIOs are generated at a rate defined by the baud-rate generator and the timer divider (Figure 9). The baud-rate generator clock frequency is divided by (1024 x Timer[14:0]) to produce the GPIO_ clock, where Timer[14:0] is the 15-bit value programmed into the **TIMER1** and **TIMER2** registers. The timer output is 50% duty cycle clock.

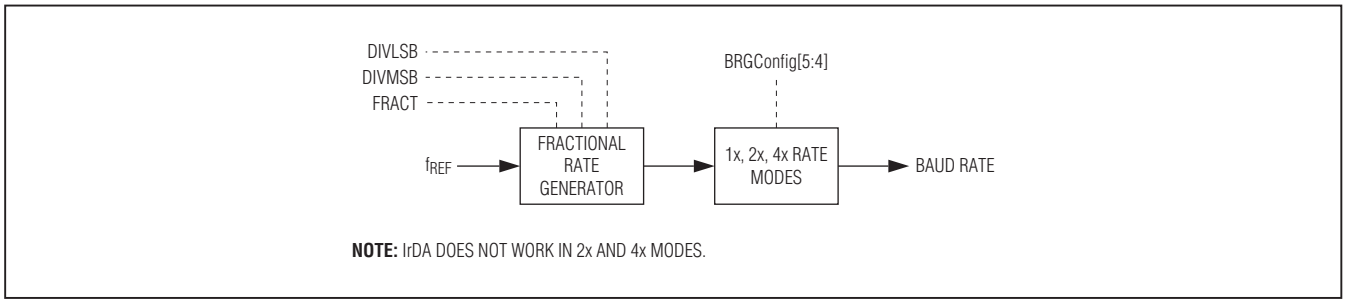


Figure 8. 2x and 4x Baud Rates

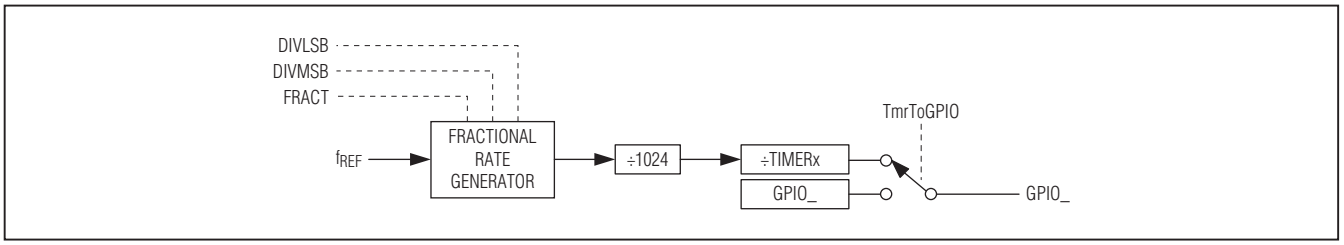


Figure 9. GPIO_ Clock Pulse Generator

UART Clock to GPIO

The MAX3109 reference clock can be routed to the GPIO0 and/or GPIO4 outputs if a synchronous high-frequency clock is needed by another device. Enable routing a UART clock to GPIO0 and/or GPIO4 in the **TxSynch** register. This output clock could, for example, be used to clock another UART device.

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 MAX3109 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 MAX3109 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.

Auto Transceiver Direction Control

In some half-duplex communication systems, the transceiver's transmitter must be turned off when data is being received in order to not load 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 MAX3109 can 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 MAX3109'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 10 shows a typical MAX3109 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 11). 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] bits.

Transmitter Triggering and Synchronization

The MAX3109 allows synchronization of transmitters so that selected UARTs start transmitting data when a trigger command is received. Optional delays can also be programmed that delay the start of transmission after a trigger command is received. A UART's transmitter can be assigned one of 16 possible SPI/I²C trigger commands. A trigger command is defined as any of the 16 special values written into the **GloblComnd** register (see the **GloblComnd** register description for more information). When a byte is written into the **GloblComnd** register, the UART select bit (U) is ignored by the MAX3109 and the **GloblComnd** applies to both UARTs. Transmission is initiated when the MAX3109 receives an assigned SPI/I²C trigger command, the selected transmitter is initially disabled, and data has been loaded into its TxFIFO.

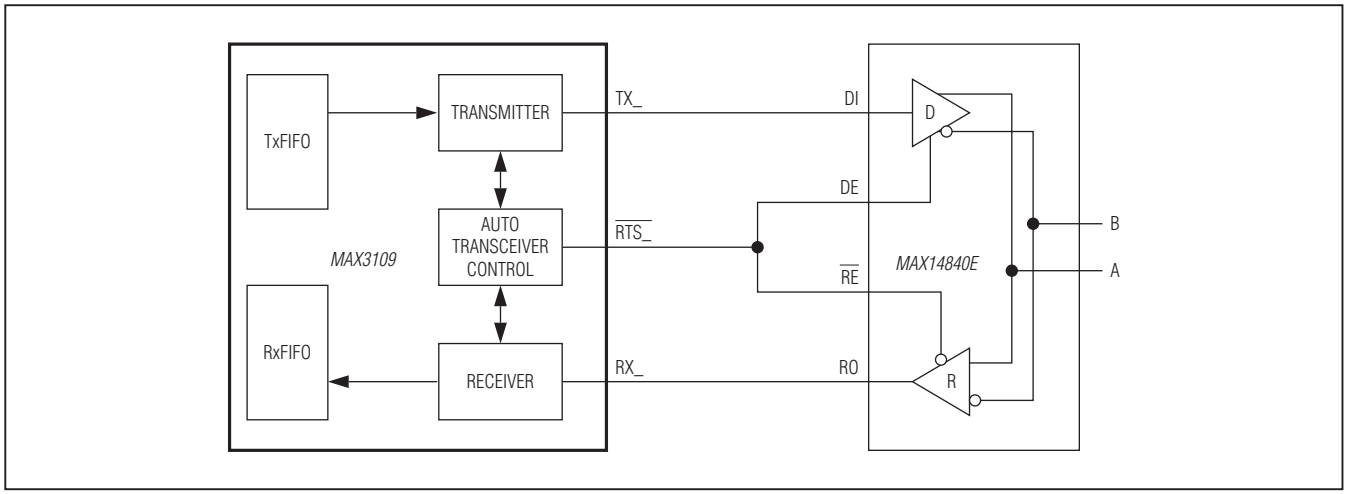


Figure 10. Auto Transceiver Direction Control

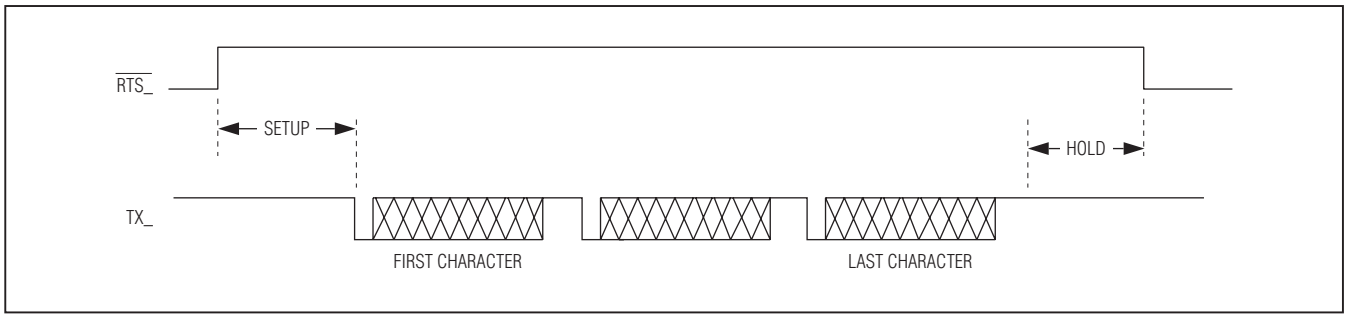


Figure 11. Setup and Hold Times in Auto Transceiver Direction Control

Enable and configure transmitter synchronization with the **TxSynch** register. Triggering and synchronization requires that the transmitters are disabled before the trigger is received. This can be done by setting the **MODE1**[1]: TxDisabl bit high or by using the auto transmitter disable function (**TxSynch**[4] is logic 1).

Transmitter Synchronization

Synchronize multiple UARTs so that their transmitters start transmission simultaneously by assigning a common trigger command to the UARTs that should be synchronized.

Intrachip and Interchip Synchronization

Intrachip transmitter triggering occurs when the two UARTs in a MAX3109 device are triggered by one command. This type of synchronization is supported in both SPI and I²C modes, as the trigger commands are global commands that are received by both UARTs simultaneously.

Interchip transmitter triggering synchronizes UARTs in different MAX3109 devices. This type of synchronization is achievable in SPI mode only. Pull the $\overline{CS}$ input of all the MAX3109 devices on the bus low during the SPI master's write trigger command so that the commands are received by all UARTs on the shared SPI bus.

I²C protocol does not allow simultaneous addressing of multiple devices.

Delayed Triggering

A delay can be programmed to postpone the start of transmission after receiving an assigned trigger command. Set the delay by programming the **SynchDelay1** and **SynchDelay2** registers.

Trigger Accuracy

The delay between the time when the MAX3109 receives a trigger command and the time when the associated transmitter starts transmission is made up of a fixed, deterministic portion, and a variable, random component.

Both portions of the delay are dependent on the UART's clock. When the fractional divider is not used, the intrinsic trigger delay, t_{TRIG}, is bounded by the following limits:

$$\frac{5}{UARTCLK} \le t_{TRIG} \le \frac{6}{UARTCLK}$$

where UARTCLK is the baud-rate divider output. The reference point is the time when the trigger command is received by the MAX3109. This occurs on the final (i.e., the 16th) SPI clock's low-to-high transition (Figure 12). In I^2C mode, this occurs on the final (i.e., the 8th) SCL low-to-high transition.

When the fractional baud-rate generator is used, the random portion is larger than one UART clock period.

Synchronization Accuracy

When synchronizing multiple UART transmitters, the output skew of the TX_ transmitter outputs is based on the triggering delays of each UART (Figure 13). This skew has a baud rate dependent component, similar to the

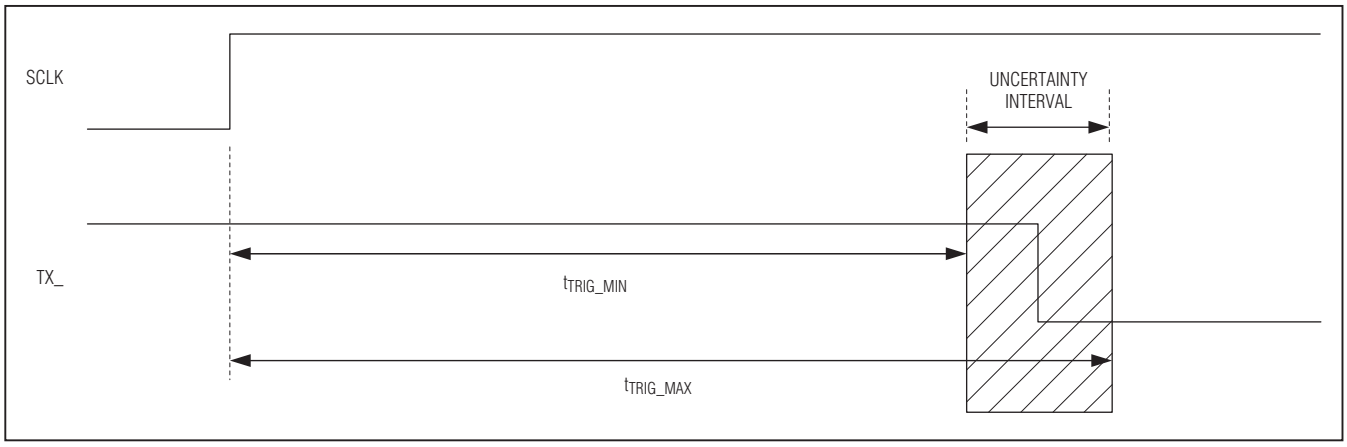


Figure 12. Single Transmitter Trigger Accuracy

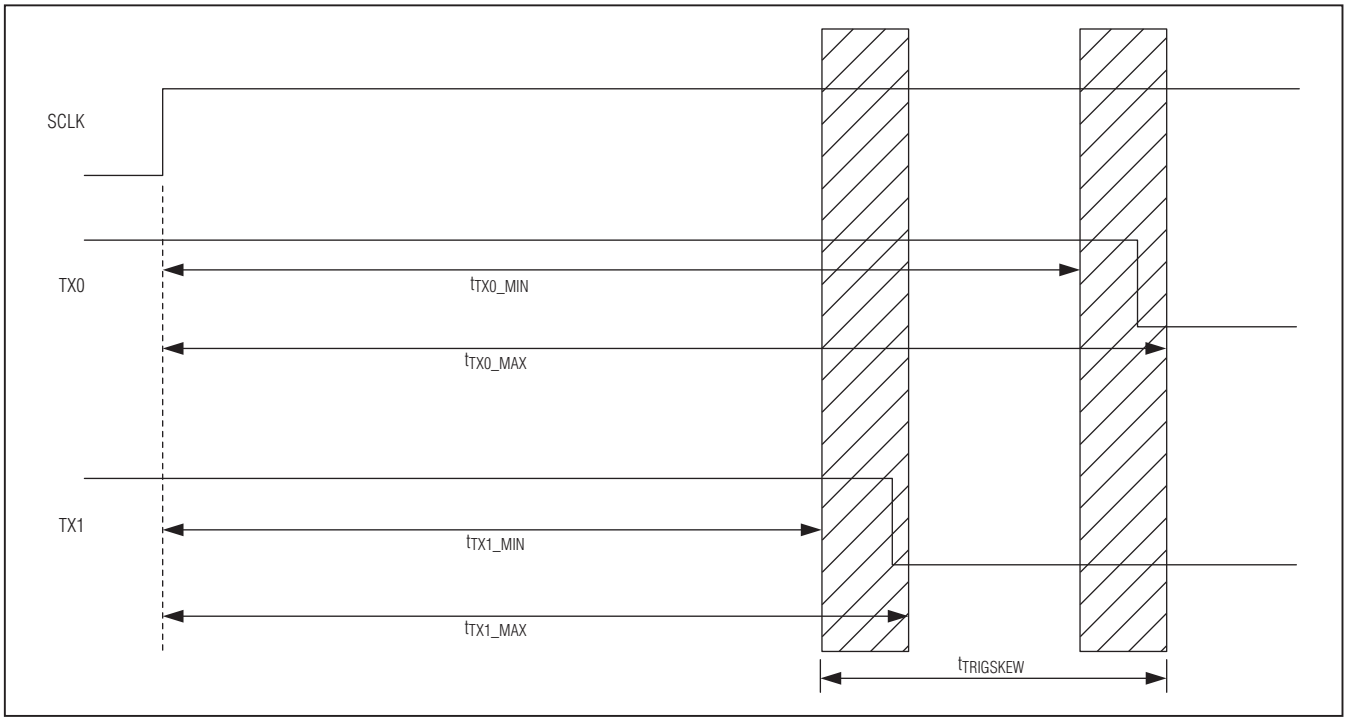


Figure 13. Multiple Transmitter Synchronization Accuracy

trigger accuracy equation for a single transmitter output. Calculate the TX_ transmitter output skew using the following equation:

$$t_{TRIGSKEW} \leq \frac{6}{(UARTCLK)_S} - \frac{5}{(UARTCLK)_F}$$

where $(UARTCLK)_S$ is the fractional divider output clock of the lower/slower baud rate UART, and $(UARTCLK)_F$ is the fractional divider output clock of the higher/faster baud rate UART.

Auto Transmitter Disable

The MAX3109 allows automatic disabling of the transmitter. Enable auto transmitter disabling functionality by setting the **TxSynch**[6]: TxAutoDis bit high. In this mode, the MAX3109 disables the specified transmitter by setting the **MODE1**[1]: TxDisabl bit high after it completes sending all the data in its TxFIFO. New data can then be loaded into the TxFIFO. A disabled transmitter does not send out data on the TX_ output when data is present in its TxFIFO.

To enable transmission after a transmitter has been disabled automatically, either clear the TxAutoDis or toggle the TxDisabl bit.

Echo Suppression

The MAX3109 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 MAX3109's receiver can block the reception of this echoed data by enabling echo suppression. Figure 14 shows a typical RS-485 application using the echo suppression feature. Set the **MODE2**[7]: EchoSuprs bit high to enable echo suppression.

The MAX3109 can also block echoes with a long round trip delay by disabling the transceiver's receiver with the $\overline{RTS\_}$ output while the MAX3109 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 15). The hold time delay is set by the **HDplxDelay**[3:0]: Holdx bits. See the **HDplxDelay** description in the *Detailed Register Descriptions* section for more information.

Echo suppression can operate simultaneously with auto transceiver direction control.

Auto Hardware Flow Control

The MAX3109 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 MAX3109 automatically controls the $\overline{RTS\_}$ handshake without the need for host processor intervention. AutoCTS flow control separately turns the MAX3109'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 MAX3109 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

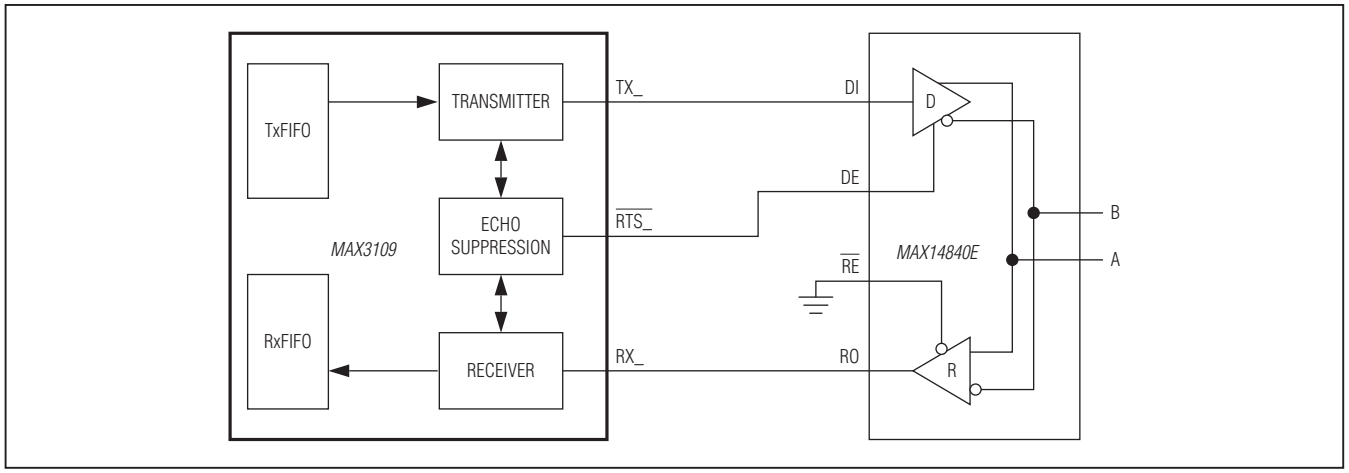


Figure 14. Half-Duplex with Echo Suppression

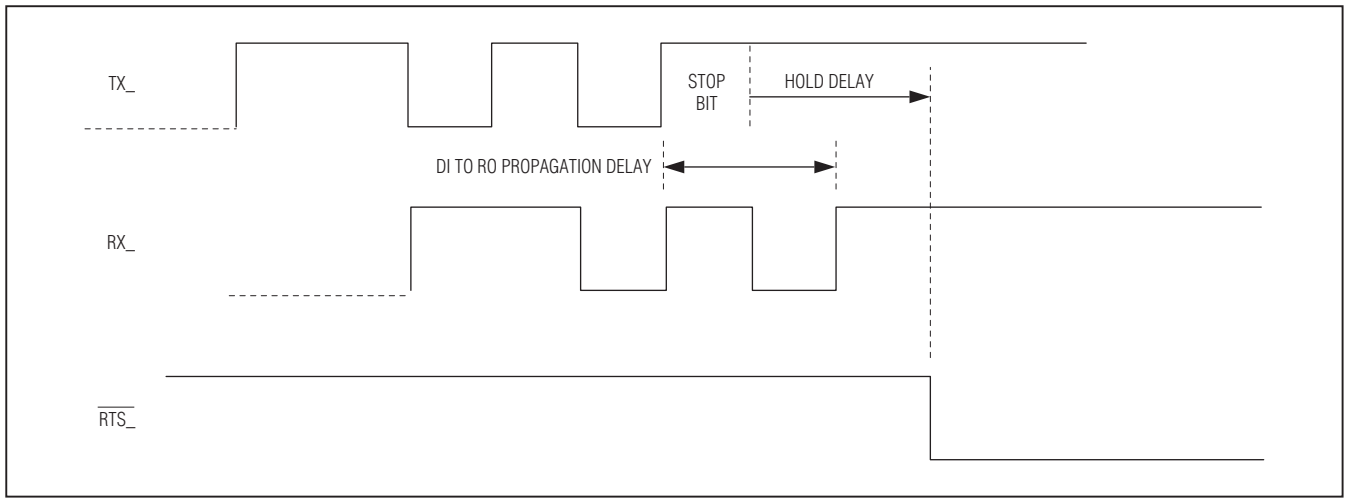


Figure 15. Echo Suppression Timing

levels in the **FlowLvl** register. With differing HALT and RESUME levels, hysteresis of the RxFIFO level can be defined for RTS_ transitions.

When the RxFIFO is filled to a level higher than the HALT level, the MAX3109 deasserts RTS_ and stops the far-end UART from transmitting any additional data. 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 RTS_ flow control management.

AutoCTS Control

When AutoCTS flow control is enabled, the UART automatically starts transmitting data when the CTS_ input is logic-low and stops transmitting data when 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 CTS_ interrupts to IRQ and ensure that the host does not receive interrupts from CTS_ transitions. If CTS_ transitions from low to high during transmission of a data word, the MAX3109 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 MAX3109 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 MAX3109 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 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 MAX3109 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