19-5014; Rev 2; 4/10

EVALUATION KIT AVAILABLE

SPI/I²C UART with 128-Word FIFOs and Internal Oscillator

General Description

The MAX3107 is an advanced universal asynchronous receiver-transmitter (UART) with 128 words each of receive and transmit first-in/first-out (FIFO) that can be controlled through I²C or high-speed SPI™. An internal oscillator reduces the need for an external crystal or clock source. The 2x and 4x rate modes allow a maximum of 24Mbps data rates. A phase-locked loop (PLL), prescaler, and fractional baud-rate generator allow for high-resolution baud-rate programming and minimize the dependency of baud rate on reference clock frequency.

Autosleep and shutdown modes help reduce power consumption during periods of inactivity. A low 640µA (typ) supply current and tiny 24-pin TQFN (3.5mm x 3.5mm) package make the MAX3107 ideal for low-power portable devices.

Integrated logic-level translation on the controller and transceiver (RX/TX and $\overline{RTS}/\overline{CTS}$) interfaces enable use with a wide selection of RS-232/RS-485 transceivers.

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 MAX3107 to be used in high-speed applications, for example Profibus-DP.

The MAX3107 is ideal for use in portable devices, industrial applications, and automotive applications. The MAX3107 is available in a 24-pin SSOP package and a 24-pin TQFN package. It is specified over the -40°C to +85°C extended ambient temperature range.

Applications

Portable Devices

Industrial Control Systems

Fieldbus Networks

Automotive Infotainment Systems

Medical Systems

Point-of-Sale Systems

HVAC or Building Control

Features

- **Tiny 24-Pin, Lead-Free TQFN (3.5mm x 3.5mm) and 24-Pin, Lead-Free SSOP Packages**
- **Integrated Internal Oscillator**
- **24Mbps (max) Data Rate**
- **Integrated PLL and Divider**
- **Fractional Baud-Rate Generator**
- **SPI Up to 26MHz Clock Rate**
- **Auto Transceiver Direction Control**
- **Half-Duplex Echo Suppression**
- **Auto RTS/CTS and XON/XOFF Flow Control**
- **Special Character Detection**
- **GPIO-Based Character Detection**
- **9-Bit Multidrop-Mode Data Filtering**
- **SIR- and MIR-Compliant IrDA Encoder/Decoder**
- **+2.35V to +3.6V Supply Range**
- **Logic-Level Translation on the Controller and Transceiver Interfaces (Down to 1.7V)**
- **Four Flexible GPIOs**
- **Line Noise Indication**
- **Shutdown and Autosleep Modes**
- **Low 640µA (typ) Supply Current at 1Mbaud and 20MHz Clock**
- **Low 20µA (typ) Shutdown Power**

Ordering Information

PART	TEMP RANGE	PIN-PACKAGE
MAX3107EAG+	-40°C to +85°C	24 SSOP
MAX3107ETG+	-40°C to +85°C	24 TQFN-EP*

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

Functional Diagram appears at end of data sheet.

SPI is a trademark of Motorola, Inc.

SPI/I²C UART with 128-Word FIFOs and Internal Oscillator

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

SPI/I²C UART with 128-Word FIFOs and Internal Oscillator

LIST OF FIGURES

LIST OF TABLES

LIST OF REGISTERS

SPI/I²C UART with 128-Word FIFOs and Internal Oscillator

ABSOLUTE MAXIMUM RATINGS

(Voltages referenced to AGND.)
V_L, V_A, V_{EXT}, XIN ..... -0.3V to +4.0V
V_{18}, XOUT ..... -0.3V to (V_A + 0.3V)
$\overline{RST}$, $\overline{IRQ}$, DIN/A1, $\overline{CS}$/A0, SCLK/SCL,
DOUT/SDA, LDOEN, $\overline{I^2C}$/SPI ..... -0.3V to (V_L + 0.3V)
TX, RX, $\overline{RTS}$/CLKOUT, $\overline{CTS}$, GPIO_..... -0.3V to (V_{EXT} + 0.3V)
DGND ..... -0.3V to +0.3V
Continuous Power Dissipation (T_A = +70°C)
24-Pin TQFN (derate 15.4mW/°C above +70°C)..... 1229mW
24-Pin SSOP (derate 12.3mW/°C above +70°C)..... 988mW
Junction-to-Ambient Thermal Resistance (θ_{JA}) (Note 1)
24-Pin TQFN..... +65°C/W
24-Pin SSOP..... +81°C/W
Junction-to-Case Thermal Resistance (θ_{JC}) (Note 1)
24-Pin TQFN..... +15°C/W
24-Pin SSOP..... +32°C/W
Operating Temperature Range ..... -40°C to +85°C
Junction Temperature ..... +150°C
Storage Temperature Range..... -65°C to +150°C
Lead Temperature (soldering, 10s) .....+300°C
Soldering Temperature (reflow) .....+260°C

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.maxim-ic.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_A = +2.35V 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_A = +2.8V, V_L = +1.8V, V_{EXT} = +2.5V, T_A = +25°C.) (Note 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Digital Interface Supply Voltage	V_L		1.71		3.6	V
Analog Supply Voltage	V_A		2.35		3.6	V
UART Interface Logic Supply Voltage	V_{EXT}		1.71		3.6	V
Logic Supply Voltage	V_{18}		1.65	1.80	1.95	V
CURRENT CONSUMPTION						
V_A Supply Current	I_A	1.8MHz crystal oscillator active, PLL disabled, V_{LDOEN} = V_L, SPI/I²C interface idle		220	500	µA
		Baud rate = 1Mbps, external clock, SPI frequency is 8MHz, external loopback PLL disabled, V_{LDOEN} = V_L (Note 3)		0.65	1.3	mA
		Internal oscillator enabled, PLL = 6X, TX-RX loopback, continuous data transmission at 115kbps, V_{LDOEN} = V_L (Note 3)		4	8	
V_A Shutdown Supply Current	$I_{A, SHDN}$	Shutdown mode, V_{LDOEN} = 0V, $V_{\overline{RST}}$ = 0V, all inputs and outputs are idle		20	35	µA
V_A Sleep Supply Current	$I_{A, SLEEP}$	Sleep mode, V_{LDOEN} = V_L, $V_{\overline{RST}}$ = V_L, all inputs and outputs are idle		45	100	µA
V_L Supply Current	I_L	All logic inputs are at V_L or V_{EXT} or 0V		4	15	µA
V_{EXT} Supply Current	I_{EXT}	All logic inputs are at V_L or V_{EXT} or 0V		5	10	µA
V_{18} Input Power-Supply Current in Shutdown Mode	I_{18SHDN}	V_{LDOEN} = 0V (V_{18} is powered by an external 1.85V voltage source), static power consumption		7	50	µA

DC ELECTRICAL CHARACTERISTICS (continued)

(V_A = +2.35V 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_A = +2.8V, V_L = +1.8V, V_{EXT} = +2.5V, T_A = +25°C.) (Note 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
SCLK/SCL, DOUT/SDA						
DOUT/SDA Output Low Voltage in I²C Mode	$V_{OL,I2C}$	I_{LOAD} = -3mA, V_L > 2V			0.4	V
		I_{LOAD} = -3mA, V_L < 2V			0.2 x V_L	V
DOUT/SDA Output Low Voltage in SPI Mode	$V_{OL,SPI}$	I_{LOAD} = -2mA			0.4	V
DOUT/SDA Output High Voltage in SPI Mode	$V_{OH,SPI}$	I_{LOAD} = 2mA			V_L - 0.4	V
Input Low Voltage	V_{IL}	SPI and I²C mode			0.3 x V_L	V
Input 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	$C_{IN\_I2C\_SPI}$	SPI and I²C mode		5		pF
$\overline{I2C}$/SPI, $\overline{CS}$/A0, DIN/A1 INPUTS						
Input Low Voltage	V_{IL}	SPI and I²C mode			0.3 x V_L	V
Input 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	$C_{IN\_I2C\_SPI}$	SPI and I²C mode		5		pF
$\overline{IRQ}$ OUTPUT (OPEN DRAIN)						
Output Low Voltage	V_{OL}	I_{LOAD} = -2mA			0.4	V
Output Leakage	I_{LK}	$V_{\overline{IRQ}}$ = 0 to V_L, $\overline{IRQ}$ is not asserted	-1		+1	µA
LDOEN AND $\overline{RST}$ INPUTS						
Input Low Voltage	V_{IL}				0.3 x V_L	V
Input High Voltage	V_{IH}		0.7 x V_L			V
Input Hysteresis	V_{HYST}			50		mV
Input Leakage Current	I_{IN}	V_{IN} = 0 to V_L	-1		+1	µA
$\overline{RTS}$/CLKOUT AND TX OUTPUTS						
Output Low Voltage	V_{OL}	I_{LOAD} = -2mA			0.4	V
Output High Voltage	V_{OH}	I_{LOAD} = 2mA	V_{EXT} - 0.4			V
Input Leakage Current	I_{IN}	Output three-stated, V_{IN} = 0 to V_{EXT}	-1		+1	µA
Input Capacitance	C_{IN_IRSTB}	High-Z mode		5		pF
RX, $\overline{CTS}$ INPUTS						
Input Low Voltage	V_{IL}				0.3 x V_{EXT}	V
Input High Voltage	V_{IH}		0.7 x V_{EXT}			V
Input Hysteresis	V_{HYST}			50		mV
$\overline{CTS}$ Input Leakage Current	$I_{IN\_\overline{CTS}}$	V_{IN} = 0 to V_{EXT}	-1		+1	µA
RX Pullup Current	I_{IN_RX}	V_{IN} = 0V	0.3	1.5	3	µA
Input Capacitance	C_{IN_IUART}			5		pF
GPIO_ OUTPUTS AND INPUTS						
Output Low Voltage	V_{OL}	I_{LOAD} = -2mA, push-pull or open drain			0.4	V
Output High Voltage	V_{OH}	I_{LOAD} = 2mA, push-pull	V_{EXT} - 0.4			V

SPI/I²C UART with 128-Word FIFOs and Internal Oscillator

DC ELECTRICAL CHARACTERISTICS (continued)

(V_A = +2.35V 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_A = +2.8V, V_L = +1.8V, V_{EXT} = +2.5V, T_A = +25°C.) (Note 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Input Low Voltage	V_{IL}	Configured as an input			0.4	V
Input High Voltage	V_{IH}	Configured as an input	2/3 x V_{EXT}			V
Pulldown Current	I_{PD}	GPIO_ = V_{EXT}	0.25	1	2.5	µA
Input Capacitance	C_{IN_IUART}	Configured as an input		5		pF
XIN						
Input Low Voltage	V_{IL}				0.3	V
Input High Voltage	V_{IH}		1.2		V_A	V
Input Capacitance	C_{XI}			16		pF
XOUT						
Input Capacitance	C_{XO}			16		pF

AC ELECTRICAL CHARACTERISTICS

(V_A = +2.35V 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_A = +2.8V, V_L = +1.8V, V_{EXT} = +2.5V, T_A = +25°C.) (Note 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
UART CLOCKING						
Internal Oscillator Frequency	f_{INT}	T_A = 0°C to +70°C	-2%	614.4	+2%	kHz
		T_A = -40°C to +85°C	-3%	614.4	+3%	
External Crystal Frequency	f_{XOSC}		1		4	MHz
External Clock Frequency	f_{CLK}		0.5		35	MHz
External Clock Duty Cycle		(Note 3)	45		55	%
Baud-Rate Generator Clock Input	f_{REF}	(Note 3)			96	MHz
I²C BUS: TIMING CHARACTERISTICS (see Figure 1)						
SCL Clock Frequency	f_{SCL}	Standard mode			100	kHz
		Fast mode			400	
Bus Free Time Between a STOP (P) and START (S) Condition	t_{BUF}	Standard mode	4.7			µs
		Fast mode	1.3			
Hold Time for START (S) Condition and Repeated START (Sr) Condition (Note 3)	$t_{HD:STA}$	Standard mode	4.0			µs
		Fast mode	0.6			
Low Period of the SCL Clock	t_{LOW}	Standard mode	4.7			µs
		Fast mode	1.3			
High Period of the SCL Clock	t_{HIGH}	Standard mode	4.0			µs
		Fast mode	0.6			
Data Hold Time	$t_{HD:DAT}$	Standard mode	0		0.9	µs
		Fast mode	0		0.9	

AC ELECTRICAL CHARACTERISTICS (continued)

(V_A = +2.35V 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_A = +2.8V, V_L = +1.8V, V_{EXT} = +2.5V, T_A = +25°C.) (Note 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Data Setup Time	$t_{SU:DAT}$	Standard mode	250			ns
		Fast mode	100			
Setup Time for Repeated START (Sr) Condition	$t_{SU:STA}$	Standard mode	4.7			µs
		Fast mode	0.6			
Rise Time of SDA and SCL Signals Receiving	t_R	Standard mode (0.3 x V_L to 0.7 x V_L) (Note 5)	20 + 0.1C_B		1000	ns
		Fast mode (0.3 x V_L to 0.7 x V_L) (Note 5)	20 + 0.1C_B		300	
Fall Time of SDA and SCL Signals	t_F	Standard mode (0.7 x V_L to 0.3 x V_L) (Note 5)	20 + 0.1C_B		300	ns
		Fast mode (0.7 x V_L to 0.3 x V_L) (Note 5)	20 + 0.1C_B		300	
Setup Time for STOP (P) Condition	$t_{SU:STO}$	Standard mode	4.7			µs
		Fast mode	0.6			
Capacitive Load for SDA and SCL (Note 3)	C_B	Standard mode			400	pF
		Fast mode			400	
I/O Capacitance (SCL, SDA)	$C_{I/O}$				10	pF
Pulse Width of Spike Suppressed	t_{SP}				50	ns
SPI BUS: TIMING CHARACTERISTICS (see Figure 2)						
SCLK Clock Period	t_{CH+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
DIN Hold Time	t_{DH}		3			ns
DIN Setup Time	t_{DS}		5			ns
Output Data Propagation Delay	t_{DO}				20	ns
DOUT Rise and Fall Times	t_{FT}				10	ns
$\overline{CS}$ Hold Time	t_{CSH}		32			ns

Note 2: All devices are production tested at T_A = +25°C. Specifications over temperature are guaranteed by design.
Note 3: Not production tested. Guaranteed by design.
Note 4: When V_{18} is powered by an external voltage regulator, the external power supply must have current capability above or equal to I_{18}.
Note 5: C_B is the total capacitance of either the clock or data line of the synchronous bus in pF.

SPI/I²C UART with 128-Word FIFOs and Internal Oscillator

Test Circuits/Timing Diagrams

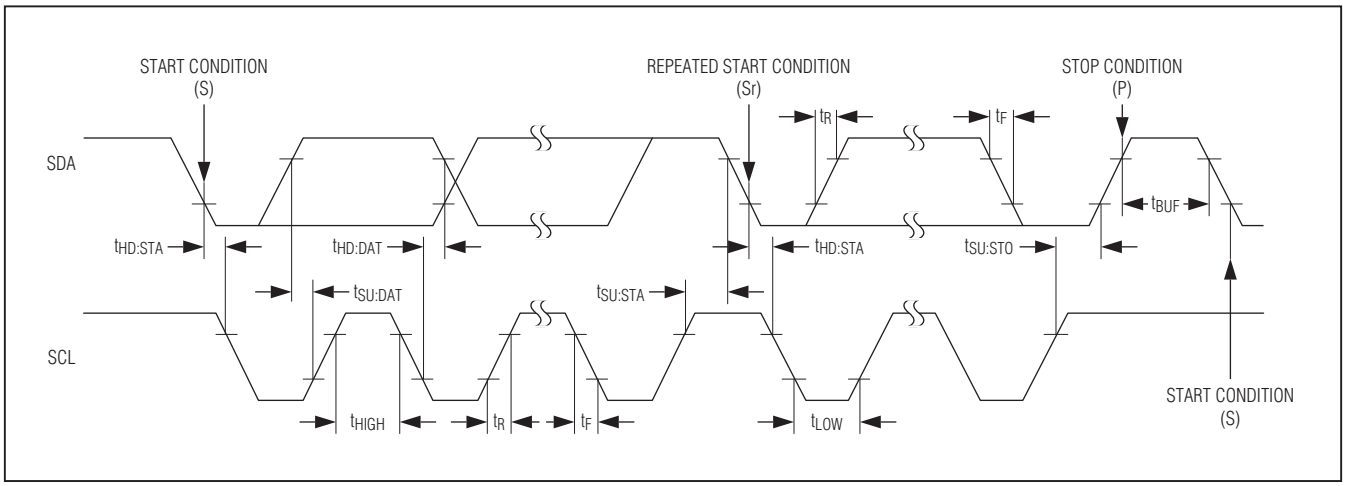


Figure 1. I²C Timing Diagram

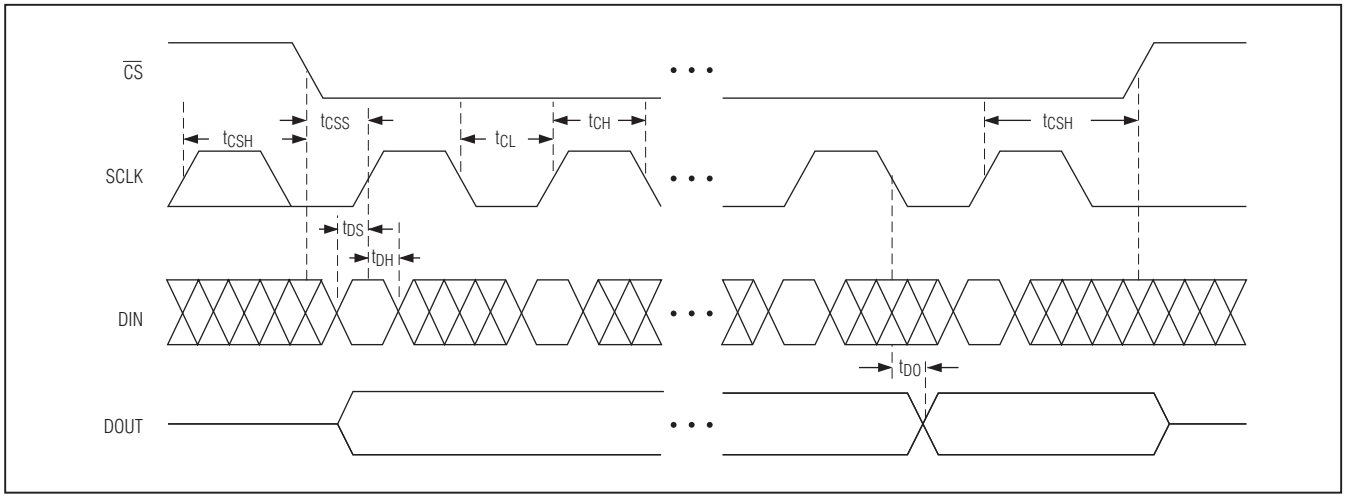
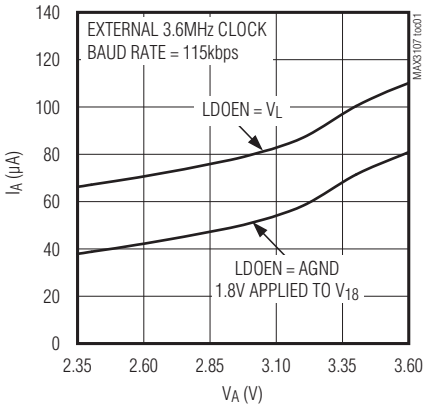


Figure 2. SPI Timing Diagram

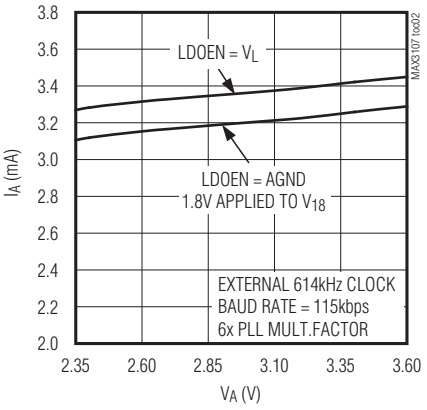
Typical Operating Characteristics

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

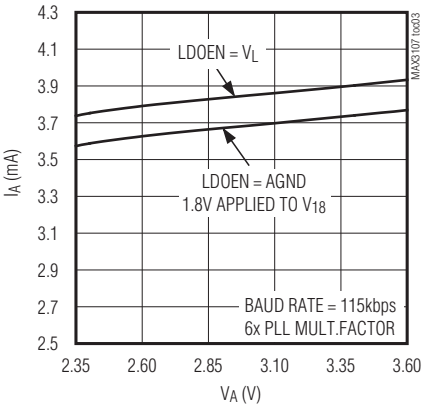
I_A SUPPLY CURRENT vs. V_A VOLTAGE (EXTERNAL CLOCK, PLL DISABLED)



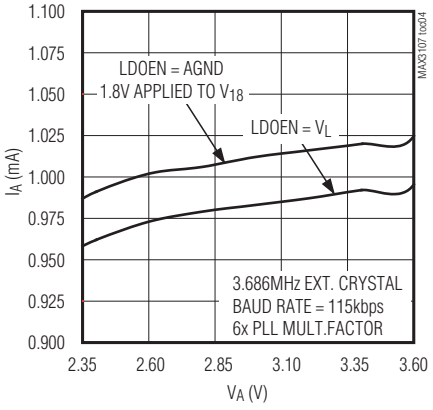
I_A SUPPLY CURRENT vs. V_A VOLTAGE (EXTERNAL CLOCK, PLL ENABLED)



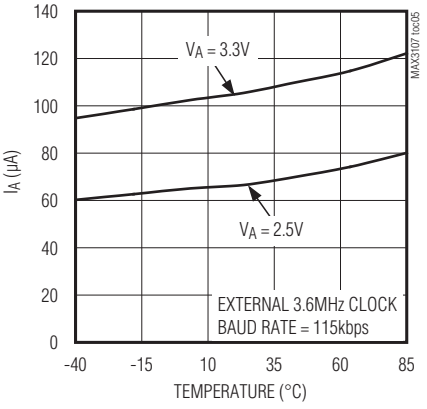
I_A SUPPLY CURRENT vs. V_A VOLTAGE (INTERNAL CLOCK, PLL ENABLED)



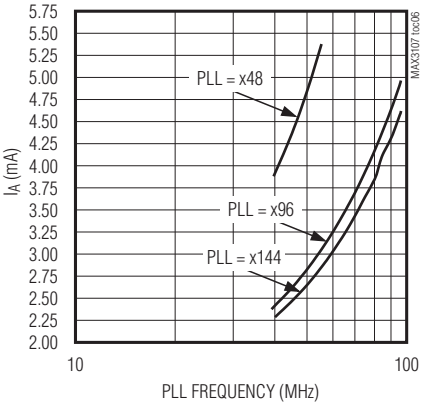
I_A SUPPLY CURRENT vs. V_A VOLTAGE (EXTERNAL CRYSTAL, PLL ENABLED)



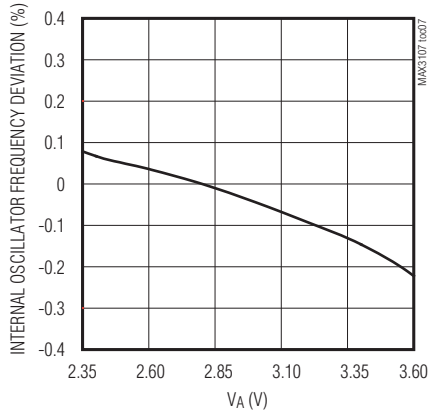
I_A SUPPLY CURRENT vs. TEMPERATURE



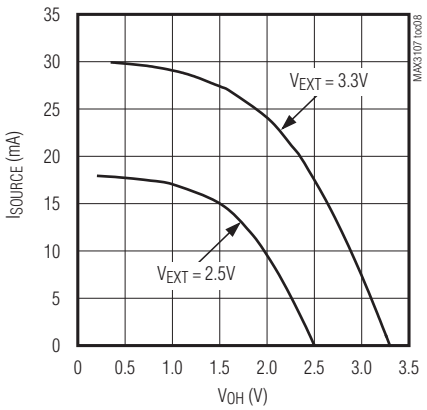
I_A SUPPLY CURRENT vs. PLL FREQUENCY



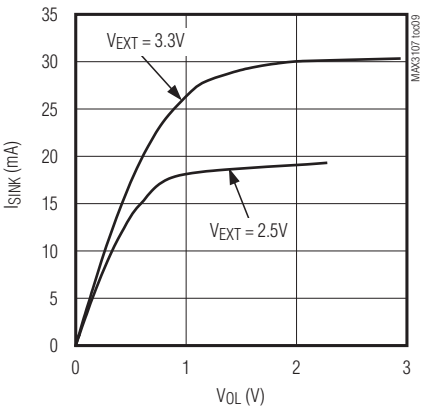
INTERNAL OSCILLATOR FREQUENCY DEVIATION vs. V_A VOLTAGE



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

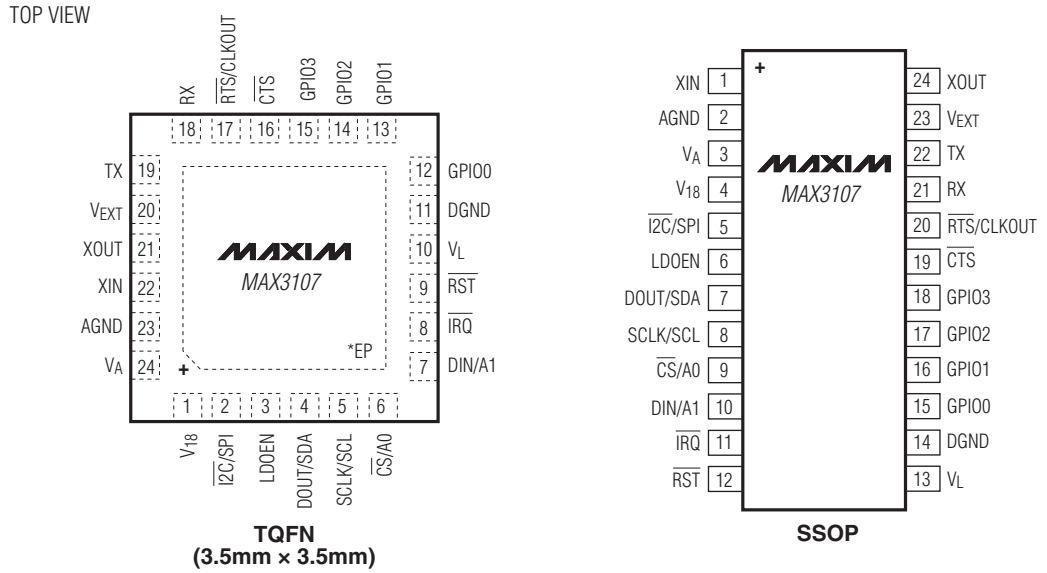


GPIO_ OUTPUT LOW VOLTAGE vs. SINK CURRENT (OPEN DRAIN)



SPI/I²C UART with 128-Word FIFOs and Internal Oscillator

Pin Configurations



Pin Descriptions

PIN		NAME	FUNCTION
TQFN-EP	SSOP		
1	4	V_{18}	Internal 1.8V LDO Output and 1.8V Logic Supply Input. Bypass V_{18} with a 0.1µF ceramic capacitor to DGND.
2	5	$\overline{I2C}$/SPI	SPI or Active-Low I²C Selector Input. Drive $\overline{I2C}$/SPI high to enable SPI. Drive $\overline{I2C}$/SPI low to enable I²C.
3	6	LDOEN	LDO Enable Input. Drive LDOEN high to enable the internal 1.8V LDO. Drive LDOEN low to disable the internal LDO. When LDO is low, V_{18} can be supplied by an external voltage source.
4	7	DOUT/SDA	Serial-Data Output. When $\overline{I2C}$/SPI is high, DOUT/SDA functions as the DOUT SPI serial-data output. When $\overline{I2C}$/SPI is low, DOUT/SDA functions as the SDA I²C serial-data input/output.
5	8	SCLK/SCL	Serial-Clock Input. When $\overline{I2C}$/SPI is high, SCLK/SCL functions as the SCLK SPI serial-clock input (up to 26MHz). When $\overline{I2C}$/SPI is low, SCLK/SCL functions as the SCL I²C serial-clock input (up to 400kHz).
6	9	$\overline{CS}$/A0	Active-Low Chip-Select and Address 0 Input. When $\overline{I2C}$/SPI is high, $\overline{CS}$/A0 functions as the $\overline{CS}$ SPI active-low chip select. When $\overline{I2C}$/SPI is low, $\overline{CS}$/A0 functions as the A0 I²C device address programming input. Connect $\overline{CS}$/A0 to DGND or V_L.

Pin Descriptions (continued)

PIN		NAME	FUNCTION
TQFN-EP	SSOP		
7	10	DIN/A1	Serial-Data and Address 1 Input. When $\overline{I2C}$/SPI is high, DIN/A1 functions as the DIN SPI serial-data input. When $\overline{I2C}$/SPI is low, DIN/A1 functions as the A1 I²C device address programming input and connects to DIN/A1 DGND or V_L.
8	11	$\overline{IRQ}$	Active-Low Interrupt Open-Drain Output. $\overline{IRQ}$ is asserted when an interrupt is pending.
9	12	$\overline{RST}$	Active-Low Reset Input. Drive $\overline{RST}$ low to force the UART into hardware reset mode. In hardware reset mode, the oscillator and the internal PLL are shut down; there is no clock activity.
10	13	V_L	Digital Interface Logic-Level Supply. V_L powers the internal logic-level translators for $\overline{RST}$, $\overline{IRQ}$, DIN/A1, $\overline{CS}$/A0, SCLK/SCL, DOUT/SDA, LDOEN, and $\overline{I2C}$/SPI. Bypass V_L with a 0.1µF ceramic capacitor to DGND.
11	14	DGND	Digital Ground
12	15	GPIO0	General-Purpose Input/Output 0. GPIO0 is user programmable as an input or output (push-pull or open drain). GPIO0 has a weak pulldown resistor to ground.
13	16	GPIO1	General-Purpose Input/Output 1. GPIO1 is user programmable as an input or output (push-pull or open drain). GPIO1 has a weak pulldown resistor to ground.
14	17	GPIO2	General-Purpose Input/Output 2. GPIO2 is user programmable as an input or output (push-pull or open drain). GPIO2 has a weak pulldown resistor to ground.
15	18	GPIO3	General-Purpose Input/Output 3. GPIO3 is user programmable as an input or output (push-pull or open drain). GPIO3 has a weak pulldown resistor to ground.
16	19	$\overline{CTS}$	Active-Low Clear-to-Send Input. $\overline{CTS}$ is a flow-control input.
17	20	$\overline{RTS}$/CLKOUT	Active-Low Request-to-Send Output. $\overline{RTS}$/CLKOUT can be set high or low by programming bit 7 ($\overline{RTS}$) of the LCR register.
18	21	RX	Receive Input. Serial UART data input. RX has an internal weak pullup resistor to V_{EXT}.
19	22	TX	Transmit Output. Serial UART data output.
20	23	V_{EXT}	Transceiver Interface Level 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.
21	24	XOUT	Crystal Output. When using an external crystal, connect one end of the crystal to XOUT and the other to XIN. When using an external clock source or the internal oscillator, leave XOUT unconnected.
22	1	XIN	Crystal/Clock Input. When using an external crystal, connect one end of the crystal to XIN and the other one to XOUT. When using an external clock source, drive XIN with the external clock. When using the internal oscillator, leave XIN unconnected.
23	2	AGND	Analog Ground
24	3	V_A	Analog Supply. V_A powers the internal oscillators, PLL, and internal LDO. Bypass V_A with a 0.1µF ceramic capacitor to AGND.
—	—	EP	Exposed Paddle. Connect EP to AGND. EP is not intended as an electrical connection point. Only for TQFN-EP package.

SPI/I²C UART with 128-Word FIFOs and Internal Oscillator

Register Map

(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†	0x00	RData7	RData6	RData5	RData4	RData3	RData2	RData1	RData0
THR†	0x00	TData7	TData6	TData5	TData4	TData3	TData2	TData1	TData0
INTERRUPTS									
IRQEn	0x01	CTSIEn	RxEmtyIEn	TxEmtyIEn	TxTrgIEn	RxTrgIEn	STSIEn	SpclChrIEn	LSRErrIEn
ISR*†	0x02	CTSInt	RxEmptyInt	TxEmptyInt	TFifoTrigInt	RFifoTrigInt	STSInt	SpCharInt	LSRErrInt
LSRIntEn	0x03	—	—	NoiseIntEn	RBreakIEn	FrameErrIEn	ParityIEn	ROverrIEn	RTimoutIEn
LSR*†	0x04	$\overline{CTS}$bit	—	RxNoise	RxBreak	FrameErr	RxParityErr	RxOverrun	RTimeout
SpclChrIntEn	0x05	—	—	MltDrpIntEn	BREAKIntEn	XOFF2IntEn	XOFF1IntEn	XON2IntEn	XON1IntEn
SpclCharInt †	0x06	—	—	MultiDropInt	BREAKInt	XOFF2Int	XOFF1Int	XON2Int	XON1Int
STSIntEn	0x07	—	SleepIntEn	ClkRdyIntEn	—	GPI3IntEn	GPI2IntEn	GPI1IntEn	GPI0IntEn
STSInt*†	0x08	—	SleepInt	ClockReady	—	GPI3Int	GPI2Int	GPI1Int	GPI0Int
UART MODES									
MODE1	0x09	IRQSel	AutoSleep	ForcedSleep	TrnscvCtrl	RTSHiZ	TXHiZ	TxDisabl	RxDisabl
MODE2	0x0A	EchoSuprs	MultiDrop	Loopback	SpecialChr	RxEmtyInv	RxTrigInv	FIFORst	RST
LCR*	0x0B	$\overline{RTS}$	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	ShortIR	SIR	IrDAEn
FIFO CONTROL									
FlowLvl	0x0F	Resume3	Resume2	Resume1	Resume0	Halt3	Halt2	Halt1	Halt0
FIFOTrgLvl*	0x10	RxTrig3	RxTrig2	RxTrig1	RxTrig0	TxTrig3	TxTrig2	TxTrig1	TxTrig0
TxFIFOLvl†	0x11	TxFL7	TxFL6	TxFL5	TxFL4	TxFL3	TxFL2	TxFL1	TxFL0
RxFIFOLvl†	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									
GPIOConfg	0x18	GP3OD	GP2OD	GP1OD	GP0OD	GP3Out	GP2Out	GP1Out	GP0Out
GPIOData	0x19	GPI3Dat	GPI2Dat	GPI1Dat	GPI0Dat	GPO3Dat	GPO2Dat	GPO1Dat	GPO0Dat
CLOCK CONFIGURATION									
PLLConfig*	0x1A	PLLFactor1	PLLFactor0	PreDiv5	PreDiv4	PreDiv3	PreDiv2	PreDiv1	PreDiv0
BRGConfig	0x1B	—	—	4xMode	2xMode	FRACT3	FRACT2	FRACT1	FRACT0
DIVLSB	0x1C	Div7	Div6	Div5	Div4	Div3	Div2	Div1	Div0
DIVMSB	0x1D	Div15	Div14	Div13	Div12	Div11	Div10	Div9	Div8
CLKSource*	0x1E	CLKtoRTS	—	—-	ExtClock	PLLBypass	PLLEn	CrystalEn	IntOscEn
REVISION									
RevID*†	0x1F	1	0	1	0	0	0	0	1

**Denotes nonzero default reset value: ISR = 0x60, LCR = 0x05, FIFOTrgLvl = 0xFF, PLLConfig = 0x01, DIVLSB = 0x01, CLKSource = 0x08, RevID = 0xA1.*

†Denotes nonread/write value: RHR = R, THR = W, ISR = COR, SpclCharInt = COR, STSInt = R/COR, LSR = R, TxFIFOLvl = R, RxFIFOLvl = R, RevID = R.

Detailed Description

The MAX3107 UART is a bridge between an SPI/MICROWIRE™ or I²C microprocessor bus and an asynchronous serial-data communication link, such as RS-485, RS-232, or IrDA. The MAX3107 contains an advanced UART, a fractional baud-rate generator, and four GPIOs. The MAX3107 is configured and monitored, and data is written and read from 8-bit registers through SPI or I²C. These registers are organized by related function as shown in the *Register Map*.

The host controller loads and transmits data into the Transmit Holding register (THR) through SPI or I²C. This data is automatically pushed into the transmit FIFO and sent out at TX. The MAX3107 adds START, STOP, and parity bits to the data and sends the data out at the selected baud rate. The clock configuration registers determine the baud rate, clock source selection, and clock frequency prescaling.

The receiver in the MAX3107 detects a START bit as a high-to-low RX transition. An internal clock samples this data. The received data is automatically placed in the receive FIFO and can then be read out of the RxFIFO through the RHR.

Register Set

The MAX3107 has a flat register structure without shadow registers. The registers are 8 bits wide. The MAX3107 registers have some similarities to the 16C550 registers.

Receive and Transmit FIFOs

The UART's receiver and the transmitter each have a 128-word deep FIFO, reducing the intervals that the host processor needs to dedicate for high-speed, high-volume data transfer. 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 utilize SPI/I²C burst data block transfers to/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 FIFOTrgLvl with a resolution of eight FIFO locations. When a receive FIFO trigger is generated, the host knows that the receive FIFO has a defined number of words waiting to be read out 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 level is above the programmed trigger level. The host then knows to throttle data writing to the transmit FIFO.

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

Transmitter Operation

Figure 3 shows the structure of the transmitter with the TxFIFO. The transmit FIFO can hold up to 128 words that are written to it through THR.

The current number of words in the TxFIFO can be read out through the TxFIFOLvl register. The transmit FIFO can be programmed to generate an interrupt when a programmed number of words are present in the TxFIFO through the FIFOTrgLvl register. The TxFIFO interrupt trigger level is selectable through FIFOTrgLvl[3:0]. When the transmit FIFO fill level reaches the programmed trigger level, the ISR[4] interrupt is set.

The transmit FIFO is empty when ISR[5]: TxEmtyInt is set. ISR[5] turns high when the transmitter starts transmitting the last word in the TxFIFO. Hence, the transmitter is completely empty after ISR[5] is set with an additional delay equal to the length of a complete character (including START, parity, and STOP bits).

The contents of the TxFIFO and RxFIFOs are both cleared through MODE2[1]: FIFORst.

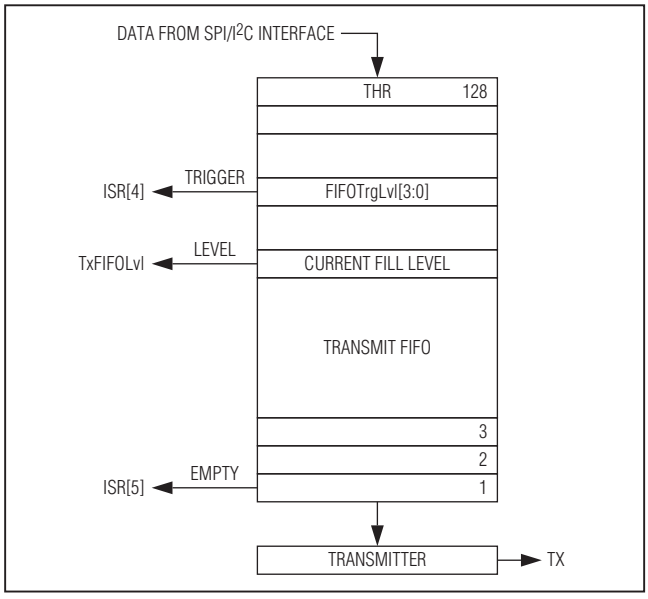


Figure 3. Transmit FIFO Signals

MICROWIRE is a trademark of National Semiconductor Corp.

SPI/I²C UART with 128-Word FIFOs and Internal Oscillator

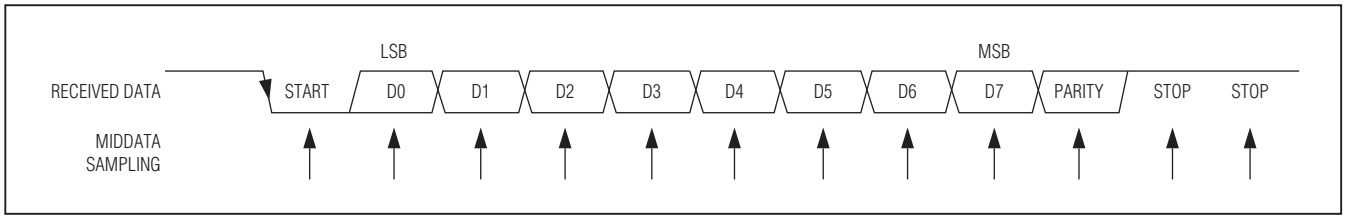


Figure 4. Receive Data Format

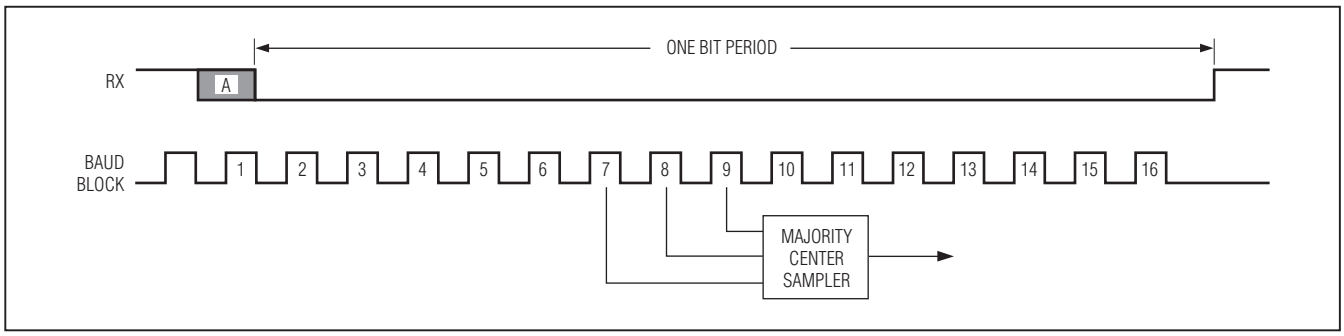


Figure 5. Midbit Sampling

To halt transmission, set MODE1[1]: TxDisabl to 1. After MODE1[1] is set, the transmitter completes transmission of the current character and then ceases transmission.

The TX output logic can be inverted through IrDA[5]: TxInv. If not stated otherwise, all transmitter logic described in this data sheet assumes IrDA[5] is 0.

Receiver Operation

The receiver expects the format of the data at RX to be as shown in Figure 4. The quiescent logic state is a high and the first bit (the START bit) is logic-low. 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 6). The host reads data out of the receive FIFO through the Receive Holding register (RHR), oldest data first. The status information of the current word in the RHR is located in the Line Status register (LSR). After a word is read out of the RHR, the LSR contains the status information for that word.

The following three error conditions are determined for each received word: parity error, framing error, and noise on the line. Line noise is detected by checking the consistency of the logic of the three samples (Figure 5).

The receiver can be turned off through MODE1[0]: RxDisabl. When this bit is set to 1, the MAX3107 turns the receiver off immediately following the current word and does not receive any further data. The RX input logic can be inverted through IrDA[4]: RxInv.

Line Noise Indication

When operating in standard (i.e., not 2x or 4x rate) mode, the MAX3107 checks that the binary logic level of the three samples per received bit are identical. If any of the three samples have differing logic levels, then noise on the transmission line has affected the received data and 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.

Clocking and Baud-Rate Generation

The MAX3107 can be clocked by its internal oscillator, an external crystal, or an external clock source. Figure 7 shows a simplified diagram of the clocking circuitry. When the MAX3107 is clocked by the internal oscillator or a crystal, the STSInt[5]: ClockReady indicates when the clocks have settled and the baud-rate generator is ready for stable operation.

The baud-rate clock can be routed to the $\overline{RTS}$/CLKOUT output. The clock rate is 16x the baud rate in standard operating mode, and 8x the baud rate in 2x rate mode. In 4x rate mode, the CLKOUT frequency is 4x the programmed baud rate. If the fractional portion of the baud-rate generator is used, the clock is not regular and exhibits jitter.

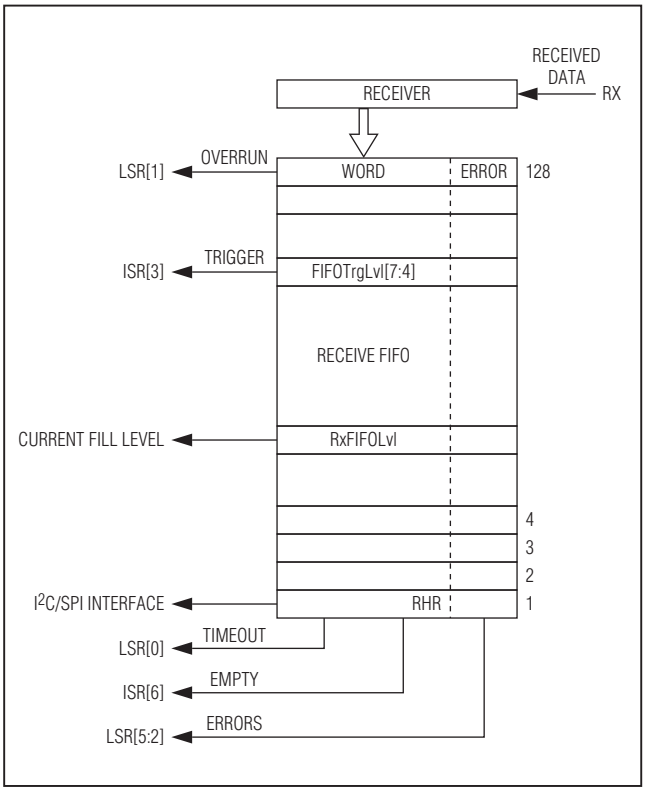


Figure 6. Receive FIFO

Internal Oscillator

The internal 614.4kHz oscillator does not require external components and provides a source for baud-rate generation. The internal oscillator normally requires the use of the internal PLL (see the *PLL and Predivider* section) to achieve common baud rates. Set CLKSource[4]: ExtClock to 0 and CLKSource[0]: IntOscEn to 1 to select and enable the internal oscillator.

Crystal Oscillator

If a higher baud-rate accuracy or low-power consumption is required, the crystal oscillator or an external clock source can be used. Set CLKSource[4]: ExtClock to 1 and CLKSource[1]: CrystalEn to 1 to enable and select the crystal oscillator. The on-chip crystal oscillator has load capacitances of 20pF integrated in both XIN and XOUT. Connect an external crystal or ceramic oscillator between XIN and XOUT.

External Clock Source

When an external clock signal is used, this should be connected to XIN. Leave XOUT unconnected. Set CLKSource[4]: ExtClock to 1 and CLKSource[1]: CrystalEn to 0 to select external clocking.

PLL and Predivider

The internal predivider and PLL allow for 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 through PLLConfig[7:6]. The predivider, located between the input clock and the PLL, allows division of the input clock by a factor between 1 and 63 by writing to PLLConfig[5:0]. See the PLLConfig register description for more information.

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 internal oscillator or with the external crystal or clock source.

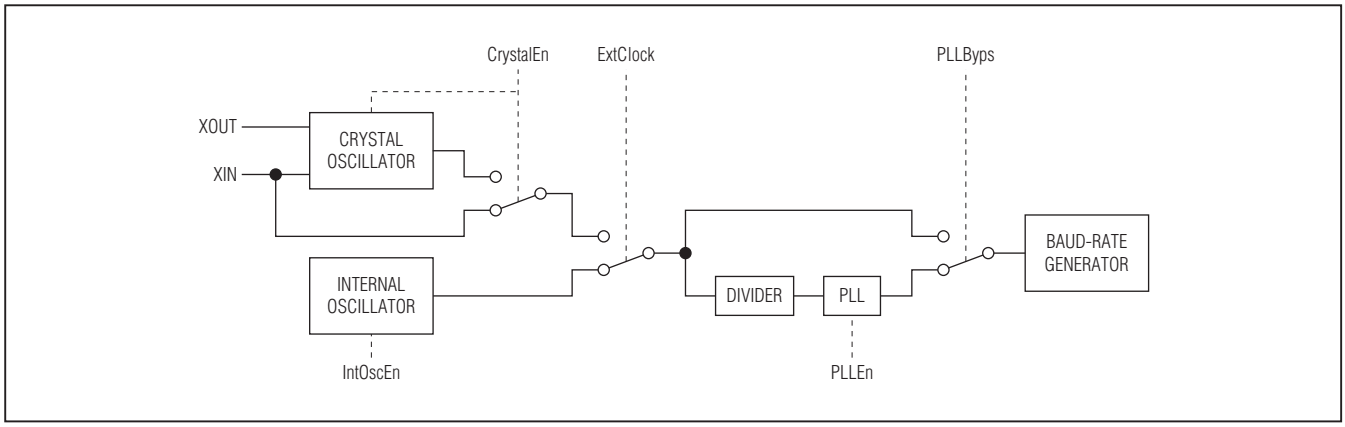


Figure 7. Clock Selection Diagram

SPI/I²C UART with 128-Word FIFOs and Internal Oscillator

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

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

where f_{REF} is the reference frequency input to the baud-rate generator and D is the ideal divisor. f_{REF} must be less than 96MHz. In 2x and 4x rate modes, replace the divisor 16 by 8 or 4, respectively.

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 wide and is programmed into the 2-byte-wide registers DIVMSB and DIVLSB. The minimum allowed for DIVLSB is 1.

The fractional portion of the divisor, FRACT, is a 4-bit nibble, which 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 calculating 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 x 190,000)

= 9.286184210526315789473684210526 3

hence DIV = 9.

FRACT =
ROUND(4.578947368421052631578947368421 1) = 5

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

The resulting (actual) baud rate can be calculated as:

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

For this example: D_{ACTUAL} = 9 + 5/16 = 9.3125

where

$$D_{ACTUAL} = DIV + FRACT/16$$

and

$$BR_{ACTUAL} = 28,230,000/(16 \times 9.3125)$$

= 189463.08724832214765100671140 94 baud

Thus, the baud rate is within 0.000028% of the ideal rate.

2x and 4x Rate Modes

To support higher baud rates than possible with standard (16x sampling) operation, the MAX3107 offers 2x and 4x rate modes. In this case, the reference clock rate only needs to be either 8x or 4x of the baud rate, respectively. The bits are only sampled once at the midbit instant instead of the usual three samples to determine the logic value of the bits. This reduces the tolerance to line noise on the received data. The 2x and 4x modes are selectable through BRGConfig[5:4]. Note that IrDA encoding and decoding does not operate in 2x and 4x 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 programmed baud rate (Figure 8).

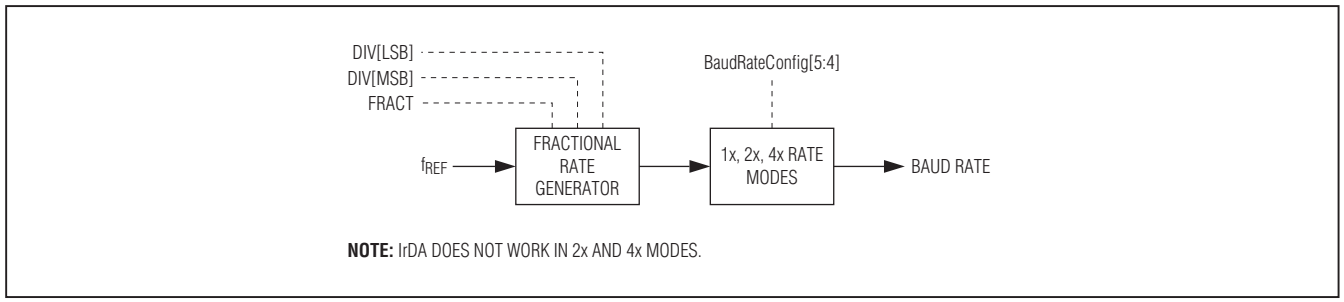


Figure 8. 2x and 4x Baud Rates

Multidrop Mode

In multidrop mode, also known as 9-bit mode, the word length is 8 bits and a 9th bit is used for distinguishing between an address and a data word. Multidrop mode is enabled through MODE2[6]: MultiDrop. Parity checking is disabled and an SpclCharInt[5]: MultiDropInt interrupt is generated when an address (9th bit set) is received.

It is up to the host processor to filter out the data intended for its address. Alternatively, the auto data-filtering mode can be used to automatically filter out the data intended for the station's specific 9-bit mode address.

Auto Data Filtering in Multidrop Mode

In multidrop mode, the MAX3107 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 values and GPIO hardware inputs. Use either XOFF2 or XOFF2[7:4] in combination with GPIO_ to define the address.

Enable multidrop mode by setting MODE2[6]: MultiDrop to 1 and enable auto data filtering by setting MODE2[4]: SpecialChr to 1.

When using register bits in combination with GPIO_ to define the address, the MSB of the address is written to XOFF2[7:4] register bits, while the LSBs of the address are defined through the GPIOs. To enable this mode, set FlowCtrl[2]: GPIAddr, MODE2[4]: SpecialChr, and MODE2[6]: MultiDrop to 1. GPIO_ is automatically read when FlowCtrl[2]: GPIAddr is set to 1, and the address is updated on logic changes at GPIO_.

In the auto data-filtering mode, the MAX3107 automatically accepts data that is meant for its address and places this into the receive FIFO, while it discards data that is not meant for its address. The received address word is not put into the FIFO.

Auto Transceiver Direction Control

In some half-duplex communication systems, the transceiver's transmitter must be turned off when data is being received so as not to load the bus. This is the case in half-duplex RS-485 communication. Similarly in full-duplex multidrop communication, like RS-485 or RS-422/V.11, only one transmitter can be enabled at any one time and the others must be disabled. The MAX3107 can automatically enable/disable a transceiver's transmitter and/or receiver. This relieves the host processor of this time-critical task.

The $\overline{RTS}$/CLKOUT output is used to control the transceivers' transmit enable input and is automatically set high when the MAX3107's transmitter starts transmission. This occurs as soon as data is present in the transmit FIFO. Auto transceiver direction control is enabled through MODE1[4]: TrnscvCtrl. Figure 9 shows a typical MAX3107 connection in a RS-485 application.

The $\overline{RTS}$/CLKOUT 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 through HDplxDelay[7:4]. Similarly, the $\overline{RTS}$/CLKOUT signal can be held high for a programmable period after the transmitter has completed transmission. The hold time is programmed through HDplxDelay[3:0].

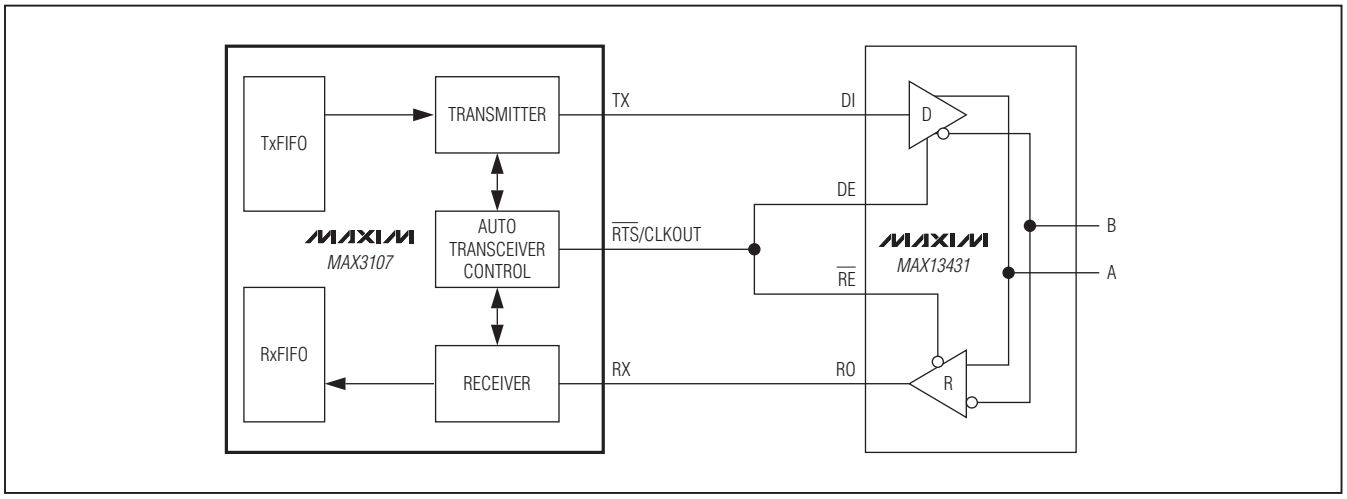


Figure 9. Auto Transceiver Direction Control

SPI/I²C UART with 128-Word FIFOs and Internal Oscillator

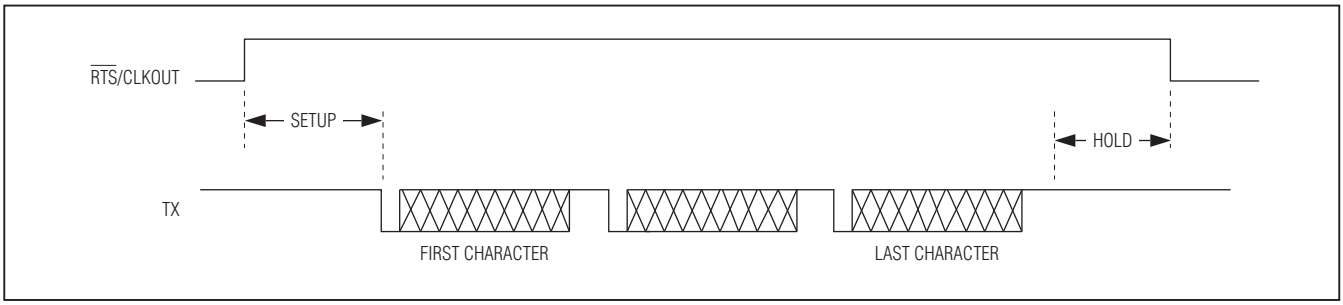


Figure 10. Setup and Hold Times in Auto Transceiver Direction Control

Echo Suppression

The MAX3107 can suppress echoed data, sometimes found in half-duplex communication (e.g., RS-485 and IrDA). If the transceiver's receiver is not turned off while the transceiver is transmitting, copies (echoes) are received by the UART. The MAX3107's receiver can block the reception of this echoed data by enabling echo suppression. Set MODE2[7]: EchoSuprs to 1 to enable echo suppression.

The MAX3107 receiver can block echoes with a long round trip delay. The transmitter can be configured to remain enabled after the end of transmission for a programmable period of time: the hold time delay. The hold time delay is set by the HDplxDelay[3:0] register. See the HDplxDelay description in the *Detailed Register Descriptions* section for more information.

Auto transceiver direction control and echo suppression can operate simultaneously.

Auto Hardware Flow Control

The MAX3107 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 MAX3107 automatically controls the $\overline{RTS}$ handshake without the need for host processor intervention. AutoCTS flow control separately turns the MAX3107's transmitter on and off based on the $\overline{CTS}$ input. AutoRTS and AutoCTS flow control are independently enabled through FlowCtrl[1:0].

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 MAX3107 does this automatically by controlling $\overline{RTS}$/CLKOUT. AutoRTS flow control is enabled through FlowCtrl[0]: AutoRTS. The HALT and RESUME levels determine the threshold levels at which $\overline{RTS}$/CLKOUT is asserted and deasserted. HALT and

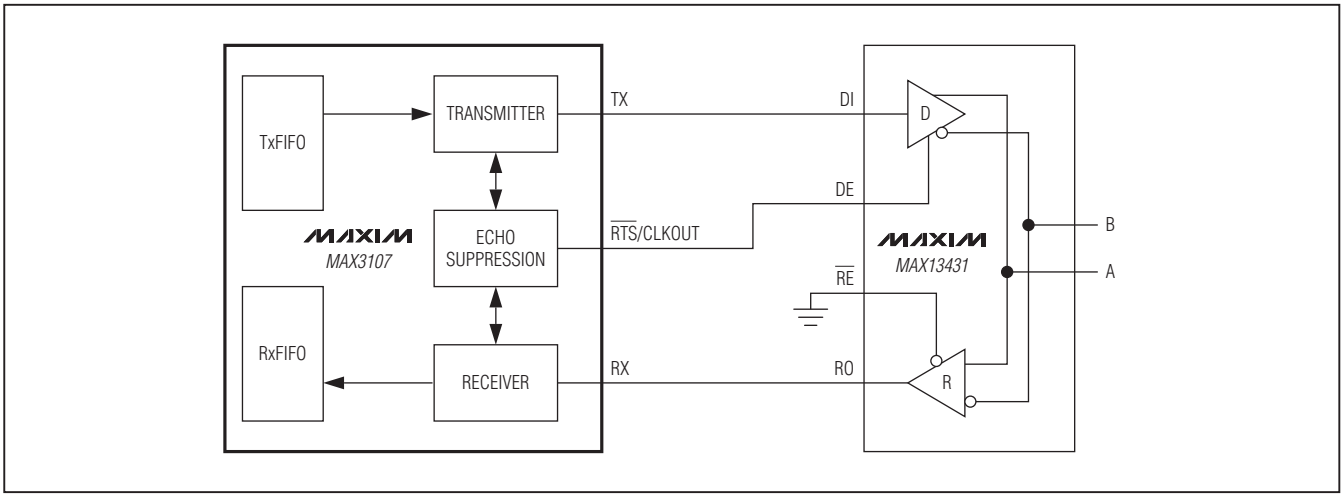


Figure 11. Half-Duplex with Echo Suppression

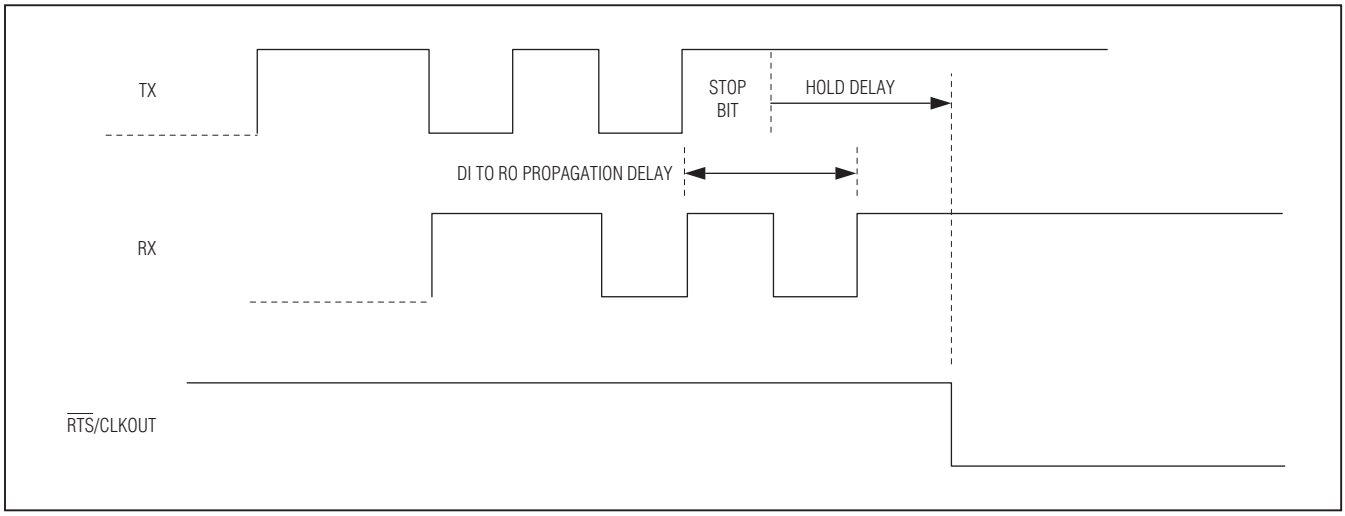


Figure 12. Echo Suppression Timing

RESUME are programmed in FlowLvl. With differing HALT and RESUME levels, hysteresis can be defined for the RTS/CLKOUT transitions.

When the RxFIFO fill level reaches the HALT level (FlowLvl[3:0]), the MAX3107 deasserts RTS/CLKOUT. RTS/CLKOUT remains deasserted until the RxFIFO is emptied and the number of words falls to 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-level low and stops transmitting when CTS is logic-high. This frees the host processor from managing this timing-critical flow-control task. AutoCTS flow control is enabled through FlowCtrl[1]: AutoCTS. During AutoCTS flow control the CTS interrupt works normally. Set the IRQEn[7]: CTSIntEn to 0 to disable CTS interrupts; then ISR[7]: CTSInt is fixed to logic 0 and the host does not receive interrupts from CTS. If CTS is set high during transmission, the MAX3107 completes transmission of the current word and halts transmission afterwards.

Turn the transmitter off by setting MODE1[1] to 1 before enabling AutoCTS control.

Auto Software (XON/XOFF) Flow Control

When auto software flow control is enabled, the MAX3107 recognizes and/or sends predefined XON/XOFF characters to control the flow of data across the asynchronous serial link. Auto flow works autonomously and does not involve 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 MAX3107 allows for double-wide (16-bit) XON/XOFF characters. XON and XOFF are programmed into the XON1, XON2 and XOFF1, XOFF2 registers.

FlowCtrl[7:3] are used for enabling and configuring auto software flow control. An ISR[1] interrupt is generated when XON or XOFF are received and details are found in SpclCharInt. The IRQ can be masked by setting IRQEn[1]: SpclChrIEn to 0.

Software flow control consists of transmitter control and receiver overflow control, which can operate independently of each other.

SPI/I²C UART with 128-Word FIFOs and Internal Oscillator

Transmitter Control

If auto transmitter control (FlowCtrl[5:4]) is enabled, the receiver compares all received words with the XOFF and XON characters. If a XOFF is received, the MAX3107 halts its transmitter from sending further data. The receiver is not affected and continues reception. Upon receiving an XON, the transmitter restarts sending data. The received XON and XOFF characters are filtered out and are not put into the receive FIFO, as they do not have significance to the higher layer protocol.

Turn the transmitter off (MODE1[1]) before enabling transmitter control.

Receiver Overflow Control

If auto receiver overflow control (FlowCtrl[7:6]) is enabled, the MAX3107 automatically sends XOFF and XON control characters to the far-end UART to avoid receiver overflow. XOFF1/XOFF2 are sent when the receive FIFO fill level reaches the HALT value set in the FlowLvl register. When the host controller reads data from the Receive FIFO to a level equal to the RESUME level programmed into the FlowLvl register, XON1/XON2 are automatically sent to the far-end station to signal it to resume data transmission.

If dual-character (XON1 and XON2/XOFF1 and XOFF2) flow control is selected, XON1/XOFF1 are transmitted before XON2/XOFF2.

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, as programmed in FIFOTrgLvl, an ISR[3] or ISR[4] interrupt is generated. There is no relationship between the trigger levels and the HALT or RESUME levels.

The FIFO trigger level can, for example, be used for a block data transfer, since it gives the host an indication when a given block size of data is available for readout in the teceive FIFO or available for transfer to the transmit FIFO.

Low-Power Standby Modes

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

Forced Sleep Mode

In forced sleep mode, all UART-related on-chip clocking is stopped. The following are inactive: the crystal oscillator, the internal oscillator, the PLL, the predivider, the receiver, and the transmitter. The SPI/I²C interface and the registers remain active. Thus, the host controller can access the resisters. To enter sleep mode, set MODE1[5] to 1. To wake up, set MODE1[5] to 0.

Autosleep Mode

The MAX3107 can be configured to operate in autosleep mode by setting MODE1[6] to 1. In autosleep mode, the MAX3107 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 characters lengths.

The MAX3107 exits autosleep mode as soon as activity is detected on any of the GPIO_, RX, or $\overline{CTS}$ inputs.

To manually wake up the MAX3107, set MODE1[6] to 0. After wake-up is initiated, the internal clock starts up and a period of time is needed for clock stabilization. The STSInt[5]: ClockReady bit indicates when the clocks are stable. If an external clock source is used, the STSInt[5] bit does not indicate clock stability.

Shutdown Mode

Shutdown mode is the lowest power consumption mode. In shutdown mode, all the MAX3107 circuitry is off. This includes the I²C/SPI interface, the registers, the FIFOs, and clocking circuitry. The LDO is kept on. To enter shutdown mode, connect $\overline{RST}$ to DGND.

When the $\overline{RST}$ input is toggled high, the MAX3107 exits shutdown mode. When the MAX3107 sets $\overline{IRQ}$ to logic-high, the chip initialization is completed. The MAX3107 needs to be reprogrammed following a shutdown.

Power-Up and $\overline{IRQ}$

$\overline{IRQ}$ has two functions. During normal operation (MODE1[7] is 1), $\overline{IRQ}$ operates as a hardware interrupt output, whereby the $\overline{IRQ}$ is active when an interrupt is pending. An $\overline{IRQ}$ interrupt is only produced during normal operation, if at least one of the IRQEn interrupt enable bits are enabled.

During power-up or following a reset, $\overline{IRQ}$ has a different function. It is held low until the MAX3107 is ready for programming following an initialization delay. Once $\overline{IRQ}$ goes high, the MAX3107 is ready to be programmed. The MODE1[7]: IRQSel bit should then be set in order to enable normal $\overline{IRQ}$ interrupt operation.

In polled mode, the RevID register can be polled to check whether the MAX3107 is ready for operation. If the controller gets a valid response from RevID, then the MAX3107 is ready for operation.

Interrupt Structure

The structure of the interrupt is shown in Figure 13. 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 directly through the ISR. The low-level interrupts typically occur less often and their specific source can be read out through the LSR, STSInt, or SpclChar registers. The three LSBs of the ISR point to the low-level interrupt registers that contain the source detail 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.

Interrupt Clearing

When an ISR interrupt is pending (i.e., any bit in ISR is set) and the ISR is subsequently read, the ISR bits and $\overline{IRQ}$ are cleared. Both the SpclCharInt and the STSInt registers also are clear on read (COR). The LSR bits are only cleared when the source of the interrupt is removed, not when LSR is read.

Detailed Register Descriptions

The MAX3107 has a flat register structure, without shadow registers, that makes programming and code simple and efficient. All registers are 8 bits wide.

RHR—Receiver Hold Register

ADDRESS:	0x00							
MODE:	R							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	RData7	RData6	RData5	RData4	RData3	RData2	RData1	RData0
RESET	0	0	0	0	0	0	0	0

Bits 7–0: RData[7:0]

The RHR is the bottom of the receive FIFO and is the register used for reading data out of the receive FIFO. It contains the oldest (first received) character in the receive FIFO. RHR[0] is the LSB of the character received at the RX input. It is the first data bit of the serial-data word received by the receiver.

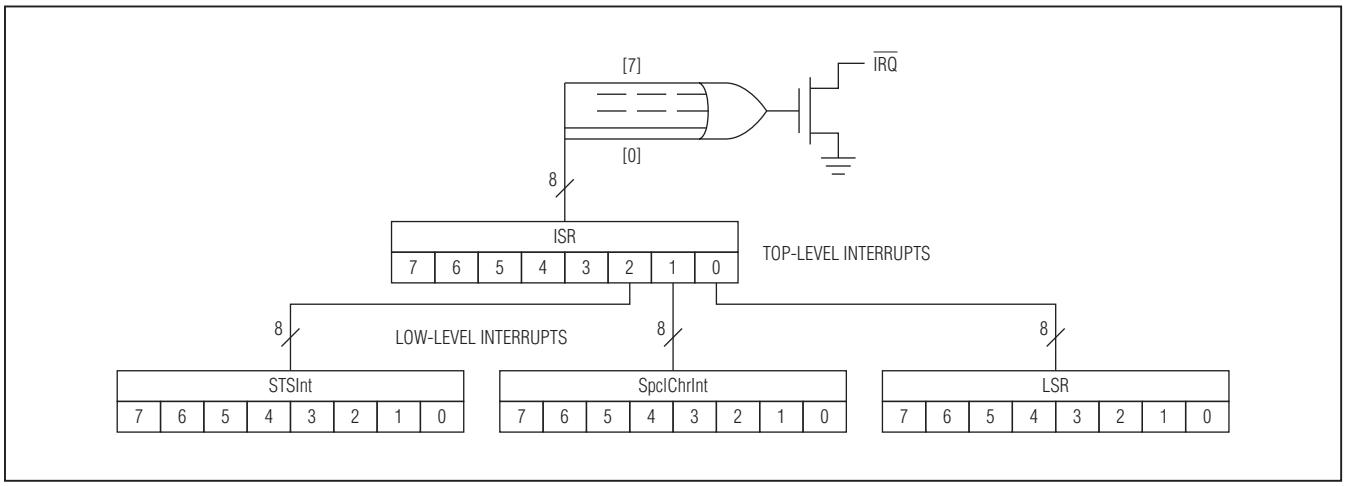


Figure 13. Simplified Interrupt Structure

SPI/I²C UART with 128-Word FIFOs and Internal Oscillator

THR—Transmit Hold Register

ADDRESS:		0x00						
MODE:		W						
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	TData7	TData6	TData5	TData4	TData3	TData2	TData1	TData0
RESET	0	0	0	0	0	0	0	0

Bits 7–0: TData[7:0]

The THR is the register that the host controller writes data to for subsequent UART transmission. This data is deposited in the transmit FIFO. THR[0] is the LSB. It is the first data bit of the serial-data word that the transmitter sends out, right after the START bit.

IRQEn—IRQ Enable Register

ADDRESS:		0x01						
MODE:		R/W						
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	CTSIEn	RxEmtyIEn	TxEmtyIEn	TxTrgIEn	RxTrgIEn	STSIEn	SpclChrIEn	LSRErrIEn
RESET	0	0	0	0	0	0	0	0

The IRQEn is used to enable the $\overline{IRQ}$ physical interrupt. Any of the eight ISR interrupt sources can be enabled to generate an $\overline{IRQ}$. The IRQEn bits only influence the $\overline{IRQ}$ output and do not have any effect on the ISR contents or behavior. Every one of the IRQEn bits operates on an ISR bit.

Bit 7: CTSIEn

The CTSIEn bit enables $\overline{IRQ}$ interrupt generation when the CTSInt interrupt bit is set in the ISR. Set CTSIEn bit low to disable $\overline{IRQ}$ generation from CTSInt.

Bit 6: RxEmtyIEn

The RxEmtyIEn bit enables $\overline{IRQ}$ interrupt generation when the RxEmtyInt interrupt bit is set in the ISR. Set RxEmtyIEn bit low to disable $\overline{IRQ}$ generation from RxEmtyInt.

Bit 5: TxEmtyIEn

The TxEmtyIEn bit enables $\overline{IRQ}$ interrupt generation when the TxEmptyInt interrupt bit is set in the ISR. Set TxEmtyIEn bit low to disable $\overline{IRQ}$ generation from TxEmptyInt.

Bit 4: TxTrgIEn

The TxTrgIEn bit enables $\overline{IRQ}$ interrupt generation when the TFifoTrigInt interrupt bit is set in the ISR. Set TxTrgIEn bit low to disable $\overline{IRQ}$ generation from TFifoTrigInt.

Bit 3: RxTrgIEn

The RxTrgIEn bit enables $\overline{IRQ}$ interrupt generation when the RFifoTrigInt interrupt bit is set in the ISR. Set RxTrgIEn bit low to disable $\overline{IRQ}$ generation from RFifoTrigInt.

Bit 2: STSIEn

The STSIEn bit enables $\overline{IRQ}$ interrupt generation when the STSInt interrupt bit is set in the ISR. Set STSIEn bit low to disable $\overline{IRQ}$ generation from STSInt.

Bit 1: SpclChrIEn

The SpclChrIEn bit enables $\overline{IRQ}$ interrupt generation when the SpCharInt interrupt bit is set in the ISR. Set SpclChrIEn bit low to disable $\overline{IRQ}$ generation from SpCharInt.

Bit 0: LSRErrIEn

The LSRErrIEn bit enables $\overline{IRQ}$ interrupt generation when the LSRErrInt interrupt bit is set in the ISR[0]. Set LSRErrIEn low to disable $\overline{IRQ}$ generation from LSRErrInt.

ISR—Interrupt Status Register

ADDRESS:	0x02							
MODE:		COR						
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	CTSInt	RxEmptyInt	TxEmptyInt	TFifoTrigInt	RFifoTrigInt	STSInt	SpCharInt	LSRErrInt
RESET	0	1	1	0	0	0	0	0

The ISR provides an overview of all interrupts generated in the MAX3107. These interrupts are cleared on reading the ISR. When the MAX3107 is operated in polled mode, the ISR can be polled to establish the UART's status. In interrupt-driven mode, $\overline{IRQ}$ interrupts are enabled through the appropriate IRQEn bits. The ISR contents give direct information on the cause for the interrupt or point to other registers that contain more detailed information.

Bit 7: CTSInt

The CTSInt is set when a logic state transition occurs at the $\overline{CTS}$ input. This bit is cleared after ISR is read. The current logic state of the $\overline{CTS}$ input can be read out through the LSR[7]: $\overline{CTS}$bit.

Bit 6: RxEmptyInt

The RxEmptyInt is set when the receive FIFO is empty. This bit is cleared after ISR is read. Its meaning can be inverted by setting the MODE2[3]: RxEmtyInv bit.

Bit 5: TxEmptyInt

The TxEmptyInt bit is set when the transmit FIFO is empty. This bit is cleared once ISR is read.

Bit 4: TFifoTrigInt

The TFifoTrigInt bit is set when the number of characters in the transmit FIFO is equal to or greater than the transmit FIFO trigger level defined in FIFOTrgLvl[3:0]. TFifoTrigInt is cleared when the transmit FIFO level falls below the trigger level or after the ISR is read. It can be used as a warning that the transmit FIFO is nearing overflow.

Bit 3: RFifoTrigInt

The RFifoTrigInt bit is set when the receive FIFO fill level reaches the receive FIFO trigger level, as defined in the FIFOTrgLvl[7:4]. This can be used as an indication that the receive FIFO is nearing overrun. It can also be used to report that a known number of words are available which can be read out in one block. The meaning of RFifoTrigInt can be inverted through MODE2[2]. RFifoTrigInt is cleared when ISR is read.

Bit 2: STSInt

The STSInt bit is set high when any bit in the STSInt register that is enabled through a STSIntEn bit is high. The STSInt bit is cleared on reading ISR.

Bit 1: SpCharInt

The SpCharInt bit is set high when a special character is received, a line BREAK is detected, or an address character is received in multidrop mode. The cause for the SpCharInt interrupt can be read from the SpclCharInt register, if enabled through the SpclChrIntEn bits. The SpCharInt interrupt is cleared when the ISR is read.

Bit 0: LSRErrInt

The LSRErrInt bit is set high when any LSR bits, which are enabled through the LSRIntEn, are set. This bit is cleared after the ISR is read.

SPI/I²C UART with 128-Word FIFOs and Internal Oscillator

LSRIntEn—Line Status Register Interrupt Enable

ADDRESS:		0x03						
MODE:		R/W						
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	—	—	NoiseIntEn	RBreakIEn	FrameErrIEn	ParityIEn	ROverrIEn	RTimoutIEn
RESET	0	0	0	0	0	0	0	0

The LSRIntEn allows routing of LSR interrupt bits to the ISR[0].

Bits 7 and 6: No Function

Bit 5: NoiseIntEn

Set the NoiseIntEn bit high to enable routing the RxNoise interrupt to LSR[0]. If NoiseIntEn is set low, RxNoise is not routed to LSR[0].

Bit 4: RBreakIEn

Set the RBreakIEn bit high to enable routing the RxBreak interrupt to LSR[0]. If RBreakIEn is set low, RxBreak is not routed to LSR[0].

Bit 3: FrameErrIEn

Set the FrameErrIEn bit high to enable routing the FrameErr interrupt to LSR[0]. If FrameErrIEn is set low, FrameErr is not routed to LSR[0].

Bit 2: ParityIEn

Set the ParityIEn bit high to enable routing the RxParityErr interrupt to LSR[0]. If ParityIEn is set low, RxParityErr is not routed to the LSR[0].

Bit 1: ROverrIEN

Set the ROverrIEn bit high to enable routing the RxOverrun interrupt to LSR[0]. If ROverrIEn is set low, RxOverrun is not routed to LSR[0].

Bit 0: RTimoutIEn

Set the RTimoutIEn bit high to enabled routing the RTimeout interrupt to LSR[0]. If RTimoutIEn is set low, the RTimeout is not routed to LSR[0].

LSR—Line Status Register

ADDRESS:		0x04						
MODE:		R						
BIT	7	6	5	4	3	2	1	0
NAME	$\overline{\text{CTS}}$bit	—	RxNoise	RxBreak	FrameErr	RxParityErr	RxOverrun	RTimeout
RESET	X	0	0	0	0	0	0	0

The LSR shows all errors related to the oldest word in the RxFIFO, waiting to be read out of the RHR. The LSR bits are not cleared upon a read; these bits stay set until the character with errors is read out of the RHR. The LSR also reflects the current state of the $\overline{\text{CTS}}$ input.

Bit 7: $\overline{\text{CTS}}$bit

The $\overline{\text{CTS}}$bit reflects the current logic state of the $\overline{\text{CTS}}$ input. This bit is cleared when the $\overline{\text{CTS}}$ input is low. Following a power-up or reset, the logic state of the $\overline{\text{CTS}}$ bit depends on the $\overline{\text{CTS}}$ input.

Bit 6: No Function

Bit 5: RxNoise

If noise is detected on the RX input during reception of a character, the RxNoise bit is set for that character. The RxNoise bit indicates that there was noise on the line while the current character residing in the RHR was received. RxNoise is cleared when the “noisy character” in the RHR is read out. The RxNoise flag can generate an ISR[0] interrupt, if enabled through LSRIntEn[5].

Bit 4: RxBreak

If a line BREAK (RX input low for a period longer than the programmed character duration) is detected, a BREAK character is put in the RxFIFO and the RxBreak bit is set for this character. A BREAK character is represented by an all-zeros data character. The RxBreak bit distinguishes a regular character with all zeros from a BREAK character. LSR[4] corresponds to the current character in the RHR. RxBreak is cleared when the BREAK character is read out of the RHR. The RxBreak flag can generate an ISR[0] interrupt, if enabled through LSRIntEn[4].

Bit 3: FrameErr

The FrameErr bit is set high when the received data frame does not match the expected frame format in length. FrameErr corresponds to the frame error of the current character in the RHR. A frame error is related to errors in expected STOP bits. This error is cleared when the affected character is read out of the RHR.

The FrameErr flag can generate an ISR[0] interrupt, if enabled, through LSRIntEn[3].

Bit 2: RxParityErr

If the parity computed on the character being received does not match the received character’s parity bit, the RxParityErr bit is set for that character. RxParityErr indicates a parity error for the current word residing in the RHR. The RxParityErr bit is cleared when the affected character is read out of the RHR.

In 9-bit multidrop mode (MODE2[6] = 1) the receiver does not check parity and the RxParityErr represents the 9th (i.e., address or data) bit.

The RxParityErr flag can generate an ISR[0] interrupt, if enabled through LSRIntEn[2].

Bit 1: RxOverrun

If the receive FIFO is full and additional data is received that does not fit into the receive FIFO, the RxOverrun bit is set. The receive FIFO retains the data in it and discards all new data that does not fit into it. The RxOverrun indication is cleared after the LSR is read or the RxFIFO level falls below its maximum. The RxOverrun flag can generate an ISR[0] interrupt, if enabled through LSRIntEn[1].

SPI/I²C UART with 128-Word FIFOs and Internal Oscillator

Bit 0: RTimeout

The RTimeout bit indicates that stale data is present in the receive FIFO. RTimeout is set when the youngest character resides in the RxFIFO for longer than the period programmed into the RxTimeOut register. The timeout counter restarts when at least one character is read out of the RxFIFO or a new character is received by the RxFIFO. If the value in RxTimeOut is zero, RTimeout is disabled. RTimeout is cleared when a word is read out of the RxFIFO or a new word is received. The RTimeout flag can generate an ISR[0] interrupt, if enabled through LSRIntEn[0].

SpclChrIntEn—Special Character Interrupt Enable Register

ADDRESS:		0x05						
MODE:		R/W						
BIT	7	6	5	4	3	2	1	0
NAME	—	—	MltDrpIntEn	BREAKIntEn	XOFF2IntEn	XOFF1IntEn	XON2IntEn	XON1IntEn
RESET	0	0	0	0	0	0	0	0

Bits 7 and 6: No Function

Bit 5: MltDrpIntEn

The MltDrpIntEn bit enables routing the SpclCharInt[5]: MultiDropInt interrupt to ISR[1]. If MltDrpIntEn is set low (default), the MultiDropInt is not routed to the ISR[1].

Bit 4: BREAKIntEn

The BREAKIntEn bit enables routing the SpclCharInt[4]: BREAKInt interrupt to ISR[1]. If BREAKIntEn is set low (default), the BREAKInt is not routed to the ISR[1].

Bit 3: XOFF2IntE

The XOFF2IntEn bit enables routing the SpclCharInt[3]: XOFF2Int interrupt to ISR[1]. If XOFF2IntEn is set low (default), the XOFF2Int is not routed to the ISR[1].

Bit 2: XOFF1IntEn

The XOFF1IntEn bit enables routing the SpclCharInt[2]: XOFF1Int interrupt to ISR[1]. If XOFF1IntEn is set low (default), the XOFF1Int is not routed to the ISR[1].

Bit 1: XON2IntEn

The XON2IntEn bit enables routing the SpclCharInt[1]: XON2Int interrupt to ISR[1]. If XON2IntEn is set low (default), the XON2Int is not routed to the ISR[1].

Bit 0: XON1IntEn

The XON1IntEn bit enables routing the SpclCharInt[0]: XON1Int interrupt to ISR[1]. If XON1IntEn is set low (default), the XON1Int is not routed to the ISR[1].

SpclCharInt—Special Character Interrupt Register

ADDRESS:		0x06						
MODE:		COR						
BIT	7	6	5	4	3	2	1	0
NAME	—	—	MultiDropInt	BREAKInt	XOFF2Int	XOFF1Int	XON2Int	XON1Int
RESET	0	0	0	0	0	0	0	0

Bits 7 and 6: No Function

Bit 5: MultiDropInt

The MultiDropInt interrupt is set when the MAX3107 receives an address character in 9-bit multidrop mode (MODE2[6] is 1). This bit is cleared when SpclCharInt is read. The SpclCharInt bit can be routed to ISR[1] by enabling SpclChrIntEn[5].

Bit 4: BREAKInt

The BreakInt interrupt is set when a line BREAK (RX low for longer than one character length) is detected by the receiver. This bit is cleared after SpclCharInt is read. The BREAKInt interrupt can be routed to ISR[1] by enabling SpclChrIntEn[4].

Bit 3: XOFF2Int

The XOFF2Int interrupt bit is set when an XOFF2 special character is received and special character detection is enabled, through MODE2[4]. This interrupt is cleared upon reading SpclCharInt. The XOFF2Int interrupt can be routed to the ISR[1] interrupt bit, if enabled through SpclChrIntEn[3].

Bit 2: XOFF1Int

The XOFF1Int interrupt bit is set when an XOFF1 special character is received and special character detection is enabled, through MODE2[4]. This interrupt is cleared upon reading SpclCharInt. The XOFF1Int interrupt can be routed to the ISR[1] interrupt bit, if enabled through SpclChrIntEn[2].

Bit 1: XON2Int

The XON2Int interrupt bit is set when an XON2 special character is received and special character detection is enabled, through MODE2[4]. This interrupt is cleared upon reading SpclCharInt. The XON2Int interrupt can be routed to the ISR[1] interrupt bit, if enabled through SpclChrIntEn[1].

Bit 0: XON1Int

The XON1Int interrupt bit is set when an XON1 special character is received and special character detection is enabled, through MODE2[4]. This interrupt is cleared upon reading SpclCharInt. The XON1Int interrupt can be routed to the ISR[1] interrupt bit, if enabled through SpclChrIntEn[0].

SPI/I²C UART with 128-Word FIFOs and Internal Oscillator

STSIntEn—STS Interrupt Enable Register

ADDRESS:		0x07						
MODE:		R/W						
BIT	7	6	5	4	3	2	1	0
NAME	—	SleepIntEn	ClkRdyIntEn	—	GPI3IntEn	GPI2IntEn	GPI1IntEn	GPI0IntEn
RESET	0	0	0	0	0	0	0	0

Bits 7 and 4: No Function

Bit 6: SleepIntEn

Set the SleepIntEn bit high to route the SleepInt status bit to the ISR[2]: STSInt. If set low, the STSIntEn masks the ISR[2] bit from SleepInt.

Bit 5: ClkRdyIntEn

Set the ClkRdyIntEn bit high to route the ClockReady status bit to the ISR[2]: STSInt bit. If set low, the ClkRdyIntEn masks the ISR[2] bit from the ClockReady status.

Bits 3–0: GPI[3:0]IntEn

The GPI[3:0]IntEn bits that are set high route the associated STSInt[3:0]: GPI[3:0]Int bits to the ISR[2] interrupt. GPI[3:0] IntEn bits that are set low, mask the ISR[2] interrupt from the associated GPI[3:0]Int bit.

STSInt—Status Interrupt Register

ADDRESS:		0x08						
MODE:		R/COR						
BIT	7	6	5	4	3	2	1	0
NAME	—	SleepInt	ClockReady	—	GPI3Int	GPI2Int	GPI1Int	GPI0Int
RESET	0	0	0	0	0	0	0	0

Bits 7 and 4: No Function

Bit 6: SleepInt

The SleepInt bit is set when the MAX3107 enters sleep mode. The SleepInt bit is cleared when the MAX3107 exits sleep mode. This status bit is cleared when the clock is disabled and cannot be cleared upon reading. The SleepInt bit can generate an ISR[2]: STSInt interrupt, if enabled through STSIntEn[6].

Bit 5: ClockReady

The ClockReady bit is set high when the clock, the divider, and the PLL have settled, and the MAX3107 is ready for data communication. The ClockReady bit only works with the internal oscillator or the crystal oscillator. It does not work with external clocking through XIN.

The ClockReady status bit is cleared when the clock is disabled and is not cleared upon read. This bit can generate an ISR[2]: STSInt interrupt, if enabled through STSIntEn[5].

Bits 3–0: GPI[3:0]Int

The GPI[3:0]Int interrupts are set high when a change of logic state occurs on the associated GPIO_ input. GPI[3:0]Int is cleared upon reading. These interrupts can be selectively routed to the ISR[2] interrupt bit through the STSIntEn[3:0].

MODE1 Register

ADDRESS:	0x09							
MODE:	R/W							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	IRQSel	AutoSleep	ForcedSleep	TrnscvCtrl	RTSHiZ	TxHiZ	TxDisabl	RxDisabl
RESET	0	0	0	0	0	0	0	0

Bit 7: IRQSel

Depending on the logic level of the IRQSel bit, $\overline{IRQ}$ has different meanings. After a hardware or software (MODE2[0]) reset, the IRQSel bit is set low and after a short delay, the $\overline{IRQ}$ output signals the end of the MAX3107's power-up sequence. The $\overline{IRQ}$ is low during power-up and transitions to high when the MAX3107 is ready to be programmed.

IRQSel can then be set high. In this case, $\overline{IRQ}$ becomes a regular interrupt output that signals pending interrupts, as indicated in the ISR. Details of the IRQSel are described in the *Power-Up and $\overline{IRQ}$* section.

Bit 6: AutoSleep

Set the AutoSleep bit high to set the MAX3107 to automatically enter low-power sleep mode after a period of no activity (see the *Autosleep Mode* section). A STSInt[6]: SleepInt interrupt is generated when the MAX3107 goes to sleep or wakes up.

Bit 5: ForcedSleep

Set the ForcedSleep bit high to force the MAX3107 into low-power sleep mode (see the *Sleep Mode* section). The current sleep or wake state can be read out through this ForcedSleep bit, even when the UART is in sleep mode.

Bit 4: TrnscvCtrl

This bit enables the automatic transceiver direction control. Set TrnscvCtrl high so that $\overline{RTS}$/CLKOUT automatically controls the transceiver's transmit/receive enable/disable inputs. Setting TrnscvCtrl high sets $\overline{RTS}$/CLKOUT low so that the transceiver is in receive mode. When the TxFIFO contains data available for transmission, the auto direction control sets $\overline{RTS}$/CLKOUT high before the transmitter sends out the data. When the transmitter is empty, $\overline{RTS}$/CLKOUT is automatically forced low again.

Setup and hold times of $\overline{RTS}$/CLKOUT with respect to the TX output can be defined through the HDplxDelay register. A transmitter empty interrupt ISR[5] is generated when the transmitter is empty.

Bit 3: RTSHiZ

Set the RTSHiZ bit high to three-state $\overline{RTS}$/CLKOUT.

Bit 2: TxHiZ

Set the TxHiz bit high to three-state the TX output.

Bit 1: TxDisabl

Set the TxDisabl bit high to disable transmission. If the TxDisabl bit is set high during transmission, the transmitter completes sending out the current character and then ceases transmission. Data still present in the transmit FIFO remains in the TxFIFO. The TX output is set to logic-high after transmission.

Bit 0: RxDisabl

Set the RxDisabl bit high to disable the receiver so that the receiver stops receiving data. All data present in the receive FIFO remains in the RxFIFO.

SPI/I²C UART with 128-Word FIFOs and Internal Oscillator

MODE2 Register

ADDRESS:	0x0A							
MODE:	R/W							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	EchoSuprs	MultiDrop	Loopback	SpecialChr	RxEmtyInv	RxTrigInv	FIFORst	RST
RESET	0	0	0	0	0	0	0	0

Bit 7: EchoSuprs

Set the EchoSuprs bit high so that the MAX3107's receiver gates any data it receives when its transmitter is busy transmitting. In half-duplex communication (like IrDA and RS-485) this allows blocking of the locally echoed data. The receiver can block data for an extended time after the transmitter ceases transmission by programming a hold time in HDplxDelay[3:0] bits.

Bit 6: MultiDrop

Set the MultiDrop bit high to enable the 9-bit multidrop mode. If this bit is set, parity checking is not performed by the receiver and parity generation is not done by the transmitter. The parity error bit, LSR[2], has a different meaning in this case. The parity error bit represents the 9th bit (address/data indication) that is received with each 9-bit character.

Bit 5: Loopback

Set the Loopback bit high to enable internal local loopback mode. This internally connects TX to RX and also $\overline{RTS}$/CLKOUT to $\overline{CTS}$. In local loopback mode, the TX output and the RX output are disconnected from the internal transmitter and receiver. The TX output is in three-state. The $\overline{RTS}$ output remains connected to the internal logic and reflects the logic state programmed in LCR[7]. The $\overline{CTS}$ input is disconnected from $\overline{RTS}$ and the internal logic. $\overline{CTS}$ thus remains in a high-impedance state.

Bit 4: SpecialChr

The SpecialChr bit enables special character detection. The receiver can detect up to four special characters, as selected in FlowCtrl:[5:4] and defined in the XON1, XON2, XOFF1 and/or XOFF2 registers, possibly in combination with GPIO_ inputs, enabled through FlowCtrl[2]: GPIAddr. When a special character is received it is put into the RxFIFO and a special character detect interrupt ISR[1] is generated.

Special character detection can be used in addition to auto XON/XOFF flow control, if enabled through FlowCtrl[3]. In this case XON/OFF flow control is then limited to single character XON and XOFF and only two special characters can then be defined (in XON2 and XOFF2).

Bit 3: RxEmtyInv

The RxEmtyInv bit inverts the meaning of the receiver empty interrupt: ISR[6]: RxEmtyInt. If RxEmtyInv is set low (default state), the ISR[6] interrupt is generated when the receive FIFO is empty. If the RxEmtyInv is set high, the ISR[6] interrupt is generated when the receive FIFO contains at least one character (i.e., is not empty).

Bit 2: RxTrigInv

The RxTrigInv bit inverts the meaning of the RxFIFO triggering. When set, an ISR[3]: RFifoTrigInt is generated when the RxFIFO is emptied to the trigger level: FIFOTrgLvl[7:4]. If the RxTrgInv bit is low (default state), the ISR[3] interrupt is generated when the RxFIFO fill level that starts from a level below FIFOTrgLvl[7:4] is filled up to the trigger level programmed into FIFOTrgLvl[7:4].

Bit 1: FIFORst

Set the FIFORst bit high to clear both the receive and transmit FIFOs of all data contents. After the FIFO reset, the FIFORst bit must then be set back to 0 to continue normal operation.

Bit 0: RST

Set the RST bit high to reset the MAX3107. The SPI/I²C bus stays active during this reset, therefore, communication with the MAX3107 is possible. All register bits are reset to their reset state and all FIFOs are cleared.

Once set high, the RST bit must be cleared by writing a 0 to RST.

LCR—Line Control Register

ADDRESS:	0x0B							
MODE:	R/W							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	$\overline{RTS}$	TxBreak	ForceParity	EvenParity	ParityEn	StopBits	Length1	Length0
RESET	0	0	0	0	0	1	0	1

Bit 7: $\overline{RTS}$

The $\overline{RTS}$ bit gives direct control of the $\overline{RTS}$/CLKOUT output logic. If the $\overline{RTS}$ bit is set high, then $\overline{RTS}$/CLKOUT is set to logic-high. The $\overline{RTS}$ bit only works if the CLKSource[7]:CLKtoRTS is not set high.

Bit 6: TxBreak

Set TxBreak to 1 to generate a line break whereby the TX output is held low until TxBreak is set to 0.

Bit 5: ForceParity

ForceParity enables forced parity, as used in 9-bit multidrop communication. Set both LCR[3] and ForceParity to use forced parity. The parity bit is forced high by the transmitter if LCR[4] low. The parity bit is forced low if LCR[4] is high.

Bit 4: EvenParity

Set EvenParity high to enable even parity. If EvenParity is set low odd parity generation/checking is used.

Bit 3: ParityEn

The ParityEn bit enables the use of a parity bit on the TX and RX interfaces. When ParityEn is low, then parity usage is disabled. When ParityEn is set to 1, the transmitter generates the parity bit as defined in LCR[4] and the receiver checks the received parity bit.

Bit 2: StopBits

This defines the number of STOP bits and depends on the length of the word programmed in LCR[1:0] (Table 1). When StopBits is high and the word length is 5, the transmitter generates a word with a STOP bit length equal to 1.5. Under these conditions, the receiver recognizes a STOP bit length greater than a 1-bit duration.

Bits 1 and 0: Length[1:0]

The Length[1:0] bits configure the length of the words that the transmitter generates and the receiver checks for at the asynchronous TX and RX interfaces (Table 2).

Table 1. StopBits Truth Table

LCR[2]	WORD LENGTH	STOP BIT LENGTH
0	5, 6, 7, 8	1
1	5	1–1.5
1	6, 7, 8	2

Table 2. Length[1:0] Truth Table

Length1	Length0	WORD LENGTH
0	0	5
0	1	6
1	0	7
1	1	8

SPI/I²C UART with 128-Word FIFOs and Internal Oscillator

RxTimeOut—Receiver Timeout Register

ADDRESS:		0x0C						
MODE:		R/W						
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	TimOut7	TimOut6	TimOut5	TimOut4	TimOutO3	TimOut2	TimOut1	TimOut0
RESET	0	0	0	0	0	0	0	0

Bits 7–0: TimOut[7:0]

The receive data timeout bits allow programming a time delay after the last (newest) character in the receive FIFO was received until a receive data timeout LSR[0] interrupt is generated. The duration is measured in character intervals and is dependent on the character length, parity, and STOP bit setting and is inversely proportional to the baud rate. If the RxTimeOut value equals zero, a timeout interrupt is not generated.

HDplxDelay Register

ADDRESS:		0x0D						
MODE:		R/W						
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	Setup3	Setup2	Setup1	Setup0	Hold3	Hold2	Hold1	Hold0
RESET	0	0	0	0	0	0	0	0

The HDplxDelay register allows programming setup and hold times between $\overline{RTS}$/CLKOUT and the TX output in auto transceiver direction control mode: MODE1[4] is 1. The Hold[3:0] time can also be used for echo suppression in half-duplex communication. HDplxDelay also functions in the 2x and 4x rate modes.

Bits 7–4: Setup[7:4]

The Setupx bits define a setup time for $\overline{RTS}$/CLKOUT to transition high before the transmitter starts transmission of its first character in auto transceiver direction control mode: MODE1[4]. This allows the MAX3107 to account for skew differences of the external transmitter's enable delay and propagation delays. Setup[7:4] can also be used to fix a stable state on the transmission line prior to start of transmission.

The unit of the HDplxDelay setup time delay is a 1-bit interval, making this delay baud-rate dependent. The maximum delay is 15-bit intervals.

Bits 3–0: Hold[3:0]

The Hold[3:0] bits define a hold time for $\overline{RTS}$/CLKOUT to be held stable (high) after the transmitter ends transmission of its last character in auto transceiver direction control mode: MODE1[4]. $\overline{RTS}$/CLKOUT turns low after the last STOP bit was sent with a Hold[3:0] delay. This keeps the external transmitter enabled during the hold duration.

The second factor that the Hold[3:0] bits define, is a delay in echo suppression mode, MODE2[7]. See the *Echo Suppression* section for more information.

The unit of the HDplxDelay hold time delay is a 1-bit interval, making the delay baud-rate dependent. The maximum delay is 15-bit intervals.

IrDA Register

ADDRESS:		0x0E						
MODE:		R/W						
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	—	—	TxInv	RxInv	MIR	ShortIR	SIR	IrDAEn
RESET	0	0	0	0	0	0	0	0

The IrDA allows selection of IrDA SIR and MIR-compliant pulse shaping at the TX and RX interfaces. It also allows inversion of the TX and RX logic, independently of whether IrDA is enabled or not.

Bits 7 and 6: No Function

Bit 5: TxInv

Set the TxInv bit high to invert the logic at the TX output. This is independent of IrDA operation.

Bit 4: RxInv

Set the RxInv bit high to invert the logic state at the RX input. This is independent of IrDA operation.

Bit 3: MIR

Set the MIR and IrDAEn bits high to select IrDA 1.1 (MIR) with 1/4 period pulse widths.

Bit 2: ShortIR

Set the ShortIR and IrDAEn bits high to select IrDA 1.0 (SIR) with the transmitter producing the minimum allowed pulse widths of 1.63µs.

Bit 1: SIR

Set the SIR bit and the IrDAEn bits high to select IrDA 1.0 pulses (SIR) with 3/16th period pulses.

Bit 0: IrDAEn

Set the IrDAEn bit high so that IrDA-compliant pulses are produced at the TX output and the MAX3107 receiver expects such pulses at its Rx input. If IrDAEn is set to low (default), normal (nonIrDA) pulses are generated and expected at the receiver. IrDAEn must be used in conjunction with the SIR, ShortIR, or MIR select bits.

FlowLvl—Flow Level Register

ADDRESS:		0x0F						
MODE:		R/W						
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	Resume3	Resume2	Resume1	Resume0	Halt3	Halt2	Halt1	Halt0
RESET	0	0	0	0	0	0	0	0

FlowLvl is used for selecting the RxFIFO threshold levels used for software (XON/XOFF) and hardware ($\overline{RTS}$/$\overline{CTS}$) flow control.

Bits 7–4: Resume[7:4]

Resume[7:4] sets the transmit FIFO threshold at which an XON is automatically sent or $\overline{RTS}$/CLKOUT is automatically set low. This signals the far-end station to start transmission. The actual threshold level is calculated as 8 times Resume[7:4]. The resulting level is in the range of 0 to 120.

Bits 3–0: Halt[3:0]

Halt[3:0] sets a receive FIFO threshold level at which an XOFF is automatically sent or $\overline{RTS}$/CLKOUT is automatically set high, depending on whether auto software or hardware flow control is enabled. This signals the far-end station to halt transmission. The actual threshold level is calculated as 8 times Halt[3:0]. Hence, the selectable threshold granularity is eight. The resulting level is in the range of 0 to 120.

SPI/I²C UART with 128-Word FIFOs and Internal Oscillator

FIFOTrgLvl—FIFO Interrupt Trigger Level Register

ADDRESS:		0x10						
MODE:		R/W						
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	RxTrig3	RxTrig2	RxTrig1	RxTrig0	TxTrig3	TxTrig2	TxTrig1	TxTrig0
RESET	1	1	1	1	1	1	1	1

Bits 7–4: RxTrig[3:0]

These 4 bits allow definition of the receive FIFO threshold level at which an ISR[3] interrupt is generated. This can be used to signal that the receive FIFO is nearing overflow or that a predefined number of FIFO locations are available for being read out in one block.

The actual FIFO trigger level is 8 times RxTrig[7:4], hence, the selectable threshold granularity is eight.

Bits 3–0: TxTrig[3:0]

These 4 bits allow definition of the transmit FIFO threshold level at which the MAX3107 generates an ISR[4] interrupt. This can be used to manage data flow to the transmit FIFO. For example, if the trigger level is defined near the bottom of the TxFIFO, the host knows that a predefined number of FIFO locations are available for being written to in one block. Alternatively, if the trigger level is set near the top of the FIFO, the host is warned when the transmit FIFO is nearing overflow, if written to on a word-by-word basis.

The actual FIFO trigger level is 8 times TxTrig[3:0], hence, the selectable threshold granularity is eight.

TxFIFOLvl—Transmit FIFO Level Register

ADDRESS:		0x11						
MODE:		R						
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	TxFL7	TxFL6	TxFL5	TxFL4	TxFL3	TxFL2	TxFL1	TxFL0
RESET	0	0	0	0	0	0	0	0

Bits 7–0: TxFL[7:0]

The TxFIFOLvl register represents the current number of words in the transmit FIFO.

RxFIFOLvl—Receive FIFO Level Register

ADDRESS:		0x12						
MODE:		R						
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	RxFL7	RxFL6	RxFL5	RxFL4	RxFL3	RxFL2	RxFL1	RxFL0
RESET	0	0	0	0	0	0	0	0

Bits 7–0: RxFL[7:0]

The RxFIFOLvl register represents the current number of words in the receive FIFO.

FlowCtrl—Flow Control Register

ADDRESS:	0x13							
MODE:	R/W							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	SwFlow3	SwFlow2	SwFlow1	SwFlow0	SwFlowEn	GPIAddr	AutoCTS	AutoRTS
RESET	0	0	0	0	0	0	0	0

Bits 7–4: SwFlow[3:0]

The SwFlow[3:0] bits configure auto software flow control and/or special character detection in combination with the characters defined in the XON1, XON2, XOFF1 and/or XOFF2 registers. See Table 3.

FlowCtrl[5:4] select which of the XON1, XON2, XOFF1 or/and XOFF2 characters are used for special character detection and/or auto flow control. If auto receiver flow control is enabled through SwFlowEn and FlowCtrl[7:6], the XON and XOFF characters that the MAX3107 receives are filtered out and are not put into the RxFIFO. Set the SwFlowEn bit to 0 and set MODE2[4] to 1 to only enable special character detection. Under these conditions, auto flow transmit flow control is not active.

If both special character detection (MODE2[4]) and auto software flow control (FlowCtrl[3]) are to be enabled, XON1 and XOFF1 define the auto flow control characters, while XON2 and XOFF2 define the special character detection characters.

Bit 3: SwFlowEn

The SwFlowEn bit enables auto software flow control. The characters used for auto software flow control are selected in SwFlow[7:4]. If special character detection (MODE2[4] set to 1) is used in addition to auto software flow control, XON1 and XOFF1 are used for flow control, while XON2 and XOFF2 define the special characters.

Bit 2: GPIAddr

The GPIAddr bit, when set, enables that the four GPIO_ inputs are used in conjunction with XOFF2 for the definition of a special character. This can be used, for example, for defining the address of a RS-485 slave device through hardware. The GPIO_ inputs logic levels, which define the 4 LSBs of the special character, while the 4 MSBs are defined by the XOFF2[7:4] bits. If GPIAddr is set, the contents of the XOFF2[3:0] bits are neglected. In this case, the XOFF2[3:0] bits, when read, also do not reflect the logic on GPIO_.

Bit 1: AutoCTS

The AutoCTS bit enables auto $\overline{CTS}$ flow control by which the transmitter stops and starts sending data depending on the logic state at the $\overline{CTS}$ input. See the *Auto Hardware Flow Control* section for a description of AutoCTS flow control. Logic changes at the $\overline{CTS}$ input result in an ISR[7]: CTSInt interrupt. The transmitter must be turned off by setting MODE1[1] to 1 before AutoCTS is enabled.

Bit 0: AutoRTS

The AutoRTS bit enables auto $\overline{RTS}$ flow control by which the MAX3107 sets its $\overline{RTS}$/CLKOUT output dependent on the receive FIFO fill level. The FIFO thresholds at which $\overline{RTS}$/CLKOUT changes state are set in FlowLvl. See the *Auto Hardware Flow Control* section for more information.

SPI/I²C UART with 128-Word FIFOs and Internal Oscillator

Table 3. SwFlow[3:0] Truth Table

SwFlow3	SwFlow2	SwFlow1	SwFlow0	DESCRIPTION
RECEIVER FLOW CONTROL		TRANSMITTER FLOW CONTROL/SPECIAL CHARACTER DETECTION		
0	0	0	0	No flow control/no character detection.
0	0	X	X	No receiver flow control.
1	0	X	X	Transmitter generates XON1, XOFF1.
0	1	X	X	Transmitter generates XON2, XOFF2.
1	1	X	X	Transmitter generates XON1, XON2, XOFF1, and XOFF2.
X	X	0	0	No transmitter flow control.
X	X	1	0	Receiver compares XON1 and XOFF1 and controls the transmitter accordingly. XON1 and XOFF1 special character detection.
X	X	0	1	Receiver compares XON2 and XOFF2 and controls the transmitter accordingly. XON2 and XOFF2 special character detection.
X	X	1	1	Receiver compares XON1, XON2, XOFF1, and XOFF2 and controls the transmitter accordingly. XON1, XON2, XOFF1, and XOFF2 special character detection.

X = Don't care

XON1 Register

ADDRESS:	0x14							
MODE:	R/W							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RESET	0	0	0	0	0	0	0	0

The XON1 and XON2 register contents define the XON characters used for auto XON/XOFF flow control and/or the special characters used for special character detection. See details in the FlowCtrl register description.

Bits 7–0: Bit[7:0]

These bits define the XON1 character if single-character XON auto software flow control is enabled in FlowCntrl[7:4]. If double-character flow control is selected in FlowCntrl[7:4], these bits constitute the LSB of the XON character. If special character detection is enabled in MODE2[4] and auto flow control is not enabled, these bits define a special character. If special character detection and auto software flow control are enabled, XON1 defines the XON flow control character.

XON2 Register

ADDRESS:		0x15						
MODE:		R/W						
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RESET	0	0	0	0	0	0	0	0

The XON1 and XON2 register contents define the XON characters for auto XON/XOFF flow control and/or the special characters used in special character detection. See details in the FlowCtrl register description.

Bits 7–0: Bit[7:0]

These bits define the XON2 character if single-character auto software flow control is enabled in FlowCntrl[7:4]. If double-character flow control is selected in FlowCntrl[7:4], these bits constitute the MSB of the XON character. If special character detection is enabled in MODE2[4], and auto software flow control is not enabled, these bits define a special character. If both special character detection and auto software flow control are enabled (MODE2[4] and FlowCntrl[3]), these bits define a special character.

XOFF1 Register

ADDRESS:		0x16						
MODE:		R/W						
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RESET	0	0	0	0	0	0	0	0

The XOFF1 and XOFF2 register contents define the XOFF characters for auto XON/XOFF flow control and/or the special characters used in special character detection. See details in the FlowCtrl register description.

Bits 7–0: Bit[7:0]

These bits define the XOFF1 character if single-character XOFF auto software flow control is enabled in FlowCntrl[7:4]. If double character flow control is selected in FlowCntrl[7:4], these bits constitute the LSB of the XOFF character. If special character detection is enabled in MODE2[4] and auto software flow control is not enabled, these bits define a special character. If special character detection and software flow control are both enabled, XOFF1 defines the XOFF flow control character.

XOFF2 Register

ADDRESS:		0x17						
MODE:		R/W						
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RESET	0	0	0	0	0	0	0	0

The XOFF1 and XOFF2 register contents define the XOFF characters for auto XON/XOFF flow control and/or special characters used in special character detection. See details in the FlowCtrl register description.

Bits 7–0: Bit[7:0]

These bits define the XOFF2 character if auto software flow control is enabled in FlowCntrl[7:4]. If double-character flow control is selected in FlowCntrl[7:4], these bits constitute the MSB of the XOFF character. If special character detection is enabled in MODE2[4] and auto flow control is not enabled, these bits define a special character. If both special character detection and auto flow control are enabled (MODE2[4] and FlowCntrl[3]), these bits define a special character.

GPIOConfg—GPIO Configuration Register

ADDRESS:		0x18						
MODE:		R/W						
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	GP3OD	GP2OD	GP1OD	GP0OD	GP3Out	GP2Out	GP1Out	GP0Out
RESET	0	0	0	0	0	0	0	0

The four GPIOs can be configured as inputs or outputs and can be operated in push-pull or open-drain mode. The reference clock has to be active for the GPIOs to work.

Bits 7–4: GP[3:0]OD

Set the GP[3:0]OD bits to 1 to configure open-drain output or input operation. If GP[3:0]OD are 0 (default), the GPIO_are push-pull outputs, if configured as outputs in GPIOConfg[3:0]. If configured as inputs in GPIOConfg[3:0], the GPIO_ are high-impedance inputs with weak pulldowns.

Bits 3–0: GP[3:0]Out

The GP[3:0]Out bits configure the GPIO_ to be inputs or outputs. Set the GP[3:0]Out bits high to configure the associated GPIO_ as outputs. The GP[3:0]Out bits which are set low, are configured to be inputs.

GPIOData—GPIO Data Register

ADDRESS:		0x19						
MODE:		R/W						
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	GPI3Dat	GPI2Dat	GPI1Dat	GPI0Dat	GPO3Dat	GPO2Dat	GPO1Dat	GPO0Dat
RESET	0	0	0	0	0	0	0	0

Bits 7–4: GPI[3:0]Dat

The GPI[3:0]Dat bits reflect the logic on GPIO_ when configured as inputs through GPIOConfg[3:0].

Bits 3–0: GPO[3:0]Dat

The GPO[3:0]Dat bits allows programming the logic state of the GPIO_, when these are configured as outputs through GPIOConfg[3:0]. For open-drain operation, pullup resistors are needed on GPIO_.

PLLConfig—PLL Configuration Register

ADDRESS:	0x1A							
MODE:	R/W							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	PLLFactor1	PLLFactor0	PreDiv5	PreDiv4	PreDiv3	PreDiv2	PreDiv1	PreDiv0
RESET	0	0	0	0	0	0	0	1

Bits 7 and 6: PLLFactor[1:0]

The two PLLFactor[1:0] bits allow programming with select PLL's multiplication factor. The input and output frequencies of the PLL have to be limited to the ranges shown in Table 4. Enable the PLL through CLKSource[2].

Bits 5–0: PreDiv[5:0]

The six PreDiv[5:0] bits allow programming the divisor of the PLL's predivider. The divisor must be chosen such that the output frequency of the predivider, which equals the PLL's input frequency, is limited to the ranges shown in Table 4. The output frequency of the internal oscillator or the input frequency of XIN is f_{CLK}; $f_{PLLIN} = f_{CLK}$/PreDiv (Figure 4). PreDiv is an integer that must be in the range of 1 to 63.

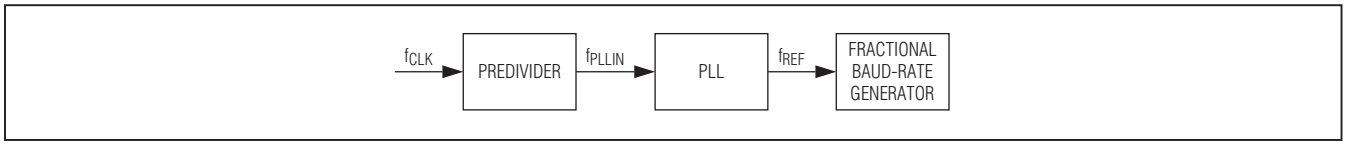


Figure 14. PLL Signal Path

Table 4. PLLFactor[1:0] Selection Guide

PLLFactor1	PLLFactor0	MULTIPLICATION FACTOR	f_{PLLIN}		f_{REF}	
			MIN	MAX	MIN	MAX
0	0	6	500kHz	800kHz	3MHz	4.8MHz
0	1	48	850kHz	1.2MHz	40.8MHz	56MHz
1	0	96	425kHz	1MHz	40.8MHz	96MHz
1	1	144	390kHz	667kHz	56MHz	96MHz

SPI/I²C UART with 128-Word FIFOs and Internal Oscillator

BRGConfig—Baud-Rate Generator Configuration Register

ADDRESS:		0x1B						
MODE:		R/W						
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	—	—	4xMode	2xMode	FRACT3	FRACT2	FRACT1	FRACT0
RESET	0	0	0	0	0	0	0	0

Bits 7 and 6: No Function

Bit 5: 4xMode

When the 4xMode bit is set high, the MAX3107 baud rate is quadruple the regular (16x sampling) baud rate. The 2xMode bit should be set low if 4xMode is enabled. See the *2x and 4x Rate Modes* section for more information.

Bit 4: 2xMode

When the 2xMode bit is set high, the MAX3107 baud rate is double the regular (16x sampling) baud rate. See the *2x and 4x Rate Modes* section for a detailed description.

Bits 3–0: FRACT[3:0]

This is the fractional portion of the baud-rate generator divisor. Set FRACT[3:0] to zero if not used. See the *Fractional Baud-Rate Generator* section for calculations.

DIVLSB—Baud-Rate Generator LSB Divisor Register

ADDRESS:		0x1C						
MODE:		R/W						
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	Div7	Div6	Div5	Div4	Div3	Div2	Div1	Div0
RESET	0	0	0	0	0	0	0	1

DIVLSB and DIVMSB define the baud-rate generator integer divisors. The minimum value is 1. See the *Fractional Baud Rate Generator* section for more information.

Bits 7–0: Div[7:0]

Div[7:0] are the 8 LSBs of the integer divisor portion (DIV) of the baud-rate generator.

DIVMSB—Baud-Rate Generator MSB Divisor Register

ADDRESS:		0x1D						
MODE:		R/W						
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	Div15	Div14	Div13	Div12	Div11	Div10	Div9	Div8
RESET	0	0	0	0	0	0	0	0

Bits 7–0: Div[15:8]

Div[15:8] is the MSB portion of the integer divisor (DIV).

CLKSource—Clock Source Register

ADDRESS:		0x1E						
MODE:		R/W						
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	CLKtoRTS	—	—	ExtClock	PLLBypass	PLLEn	CrystalEn	IntOscEn
RESET	0	0	0	0	1	0	0	0

Bit 7: CLKtoRTS

Set the CLKtoRTS bit to 1 to route the baud-rate generator (16x baud rate) output clock to $\overline{RTS}$/CLKOUT. The clock frequency is a factor of 16x, 8x, or 4x of the baud rate, depending on the BRGConfig[5:4] settings.

Bits 6 and 5: No Function

Bit 4: ExtClock

Set the ExtClock bit high to enable an external clocking source (crystal or clock generator at XIN). Set the ExtClock bit to 0 to select the internal oscillator for clocking.

Bit 3: PLLBypass

Set the PLLBypass bit high to enable bypassing the internal PLL and predivider.

Bit 2: PLLEn

Set the PLLEn bit high to enable the internal PLL. If PLLEn is set low, the internal PLL is disabled.

Bit 1: CrystalEn

Set the CrystalEn bit high to enable the crystal oscillator. When using an external clock source at XIN, CrystalEn must be set low.

Bit 0: IntOscEn

Set the IntOscEn bit high to enable the internal oscillator. If IntOscEn is set low, the internal oscillator is disabled.

RevID—Revision Identification Register

ADDRESS:		0x1F						
MODE:		R						
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RESET	1	0	1	0	0	0	0	1

Bit 7–0: Bit[7:0]

The RevID register indicates the revision number of the MAX3107 silicon, starting with 0xA1. This can be used during software development.

SPI/I²C UART with 128-Word FIFOs and Internal Oscillator

Serial Controller Interface

The MAX3107 can be controlled through SPI or I²C as defined by the logic on I2C/SPI. See the *Pin Configurations* for further details.

SPI Interface

The SPI supports both single-cycle and burst-read/write access. The SPI master must generate clock and data signals in SPI MODE0 (i.e., with clock polarity CPOL = 0 and clock phase CPHA = 0).

SPI Single-Cycle Access

Figure 15 shows a single-cycle read and Figure 16 shows a single-cycle write.

SPI Burst Access

Burst access allows writing and reading in one block by only defining the initial register address in the SPI command byte. Multiple characters can be loaded into the transmit FIFO by using the THR (0x00) as the initial burst read address. Similarly, multiple characters can be read out of the receiver FIFO by using the RHR (0x00) as the SPI's burst read address. If the SPI burst address is different to 0x00, the MAX3107 automatically increments the register address after each SPI data byte. Efficient programming of multiple consecutive registers is thus possible. Chip select, CS/A0, must be kept low during the whole cycle. The SCLK/SCL clock continues clocking throughout the burst access cycle. The burst cycle ends when the SPI master pulls CS/A0 high.

For example, writing 128 bytes into the TxFIFO can be achieved by a burst write access through the following sequence:

- Pull CS/A0 low
- Send SPI write command
- Send 128 byes
- Release CS/A0

This takes a total of (1 + 128) x 8 clock cycles.

I²C Interface

The MAX3107 contains an I²C-compatible interface for data communication with a host processor (SCL and SDA). The interface supports a clock frequency up to 400kHz. SCL and SDA require pullup resistors that are connected to a positive supply.

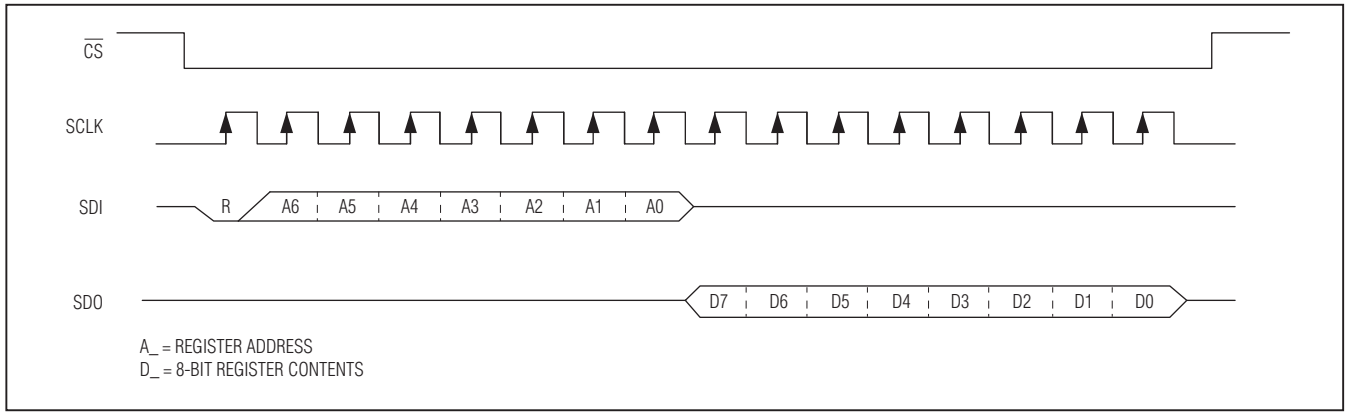


Figure 15. SPI Single-Cycle Read

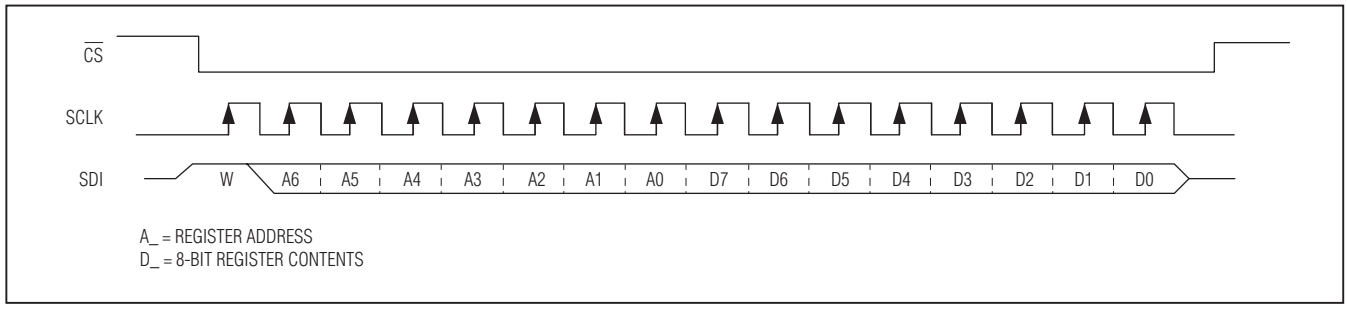


Figure 16. SPI Single-Cycle Write

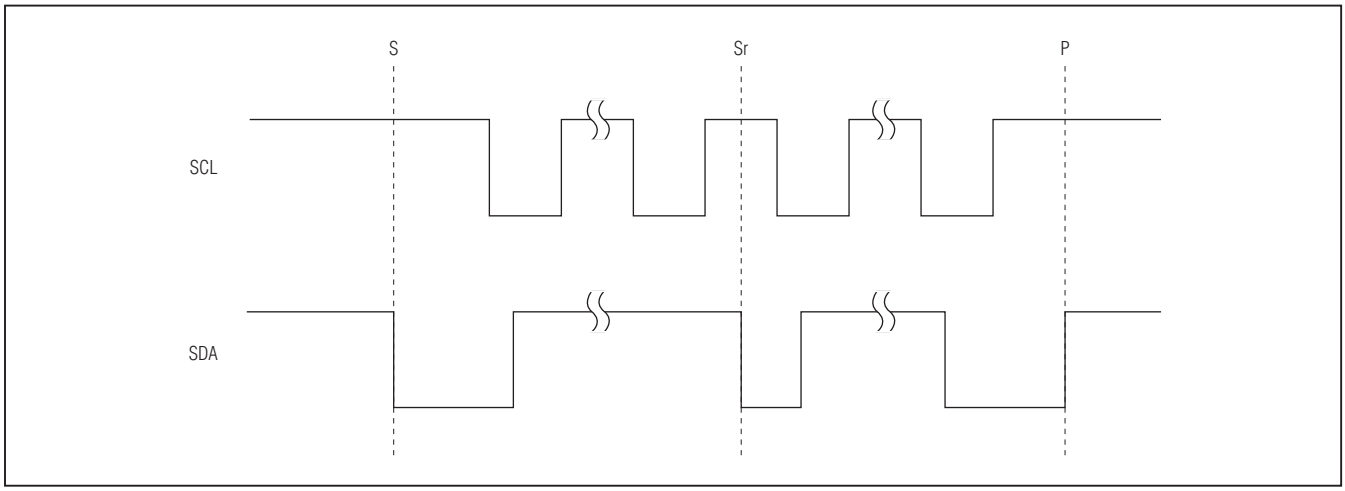


Figure 17. I²C START, STOP, and Repeated START Conditions

Table 5. I²C Address Map

DIN/A1	$\overline{CS}$/A0	READ/ WRITE	I²C ADDRESS
0	0	W	0x58
		R	0x59
0	1	W	0x5A
		R	0x5B
1	0	W	0x5C
		R	0x5D
1	1	W	0x5E
		R	0x5F

START, STOP, and Repeated START Conditions

When writing to the MAX3107 using I²C, the master sends a START condition (S) followed by the MAX3107 I²C address. After the address, the master sends the register address of the register that is to be programmed. The master then ends communication by issuing a STOP condition (P) to relinquish control of the bus, or a repeated START condition (Sr) to communicate to another I²C slave. See Figure 17.

Slave Address

The MAX3107 includes a 7-bit slave address. The first 5 bits (MSBs) of the slave address are factory-programmed and always 01011. These slave addresses are unique device IDs. Connect A1, A0 to ground or V_L to set the I²C slave address (Table 5). The address is defined as the 7 MSBs followed by the read/write bit. Set the read/write bit to 1 to configure the MAX3107 to read mode. Set the read/write bit to 0 to configure the MAX3107 to write mode. The address is the first byte of information sent to the MAX3107 after the START condition.

Bit Transfer

One data bit is transferred during each SCL clock cycle. The data on SDA must remain stable during the high period of the SCL clock pulse. Changes in SDA while SCL is high and stable are considered control signals (see the *START, STOP, and Repeated START Conditions* section). Both SDA and SCL remain high when the bus is not active.

Single-Byte Write

With this operation the master sends an address and 1 or 2 data bytes to the slave device (Figure 18). The write byte procedure is as follows:

1) The master sends a START condition.
2) The master sends the 7-bit slave ID plus a write bit (low).
3) The addressed slave asserts an ACK on the data line.
4) The master sends the 8-bit register address.
5) The active slave asserts an ACK on the data line only if the address is valid (NACK if not).
6) The master sends the 8-bit data byte.
7) The slave asserts an ACK on the data line.
8) The master generates a STOP condition.

Burst Write

With this operation the master sends an address and multiple data bytes to the slave device (Figure 19). The burst write procedure is as follows:

1) The master sends a START condition.

SPI/I²C UART with 128-Word FIFOs and Internal Oscillator

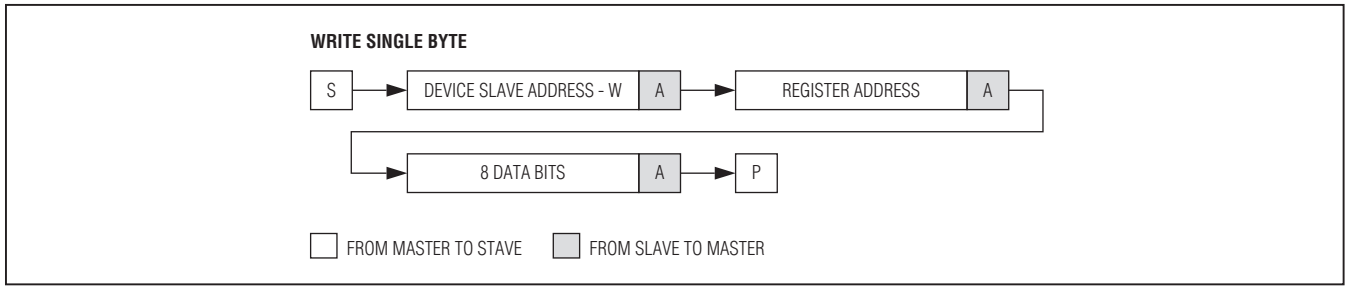


Figure 18. Write Byte Sequence

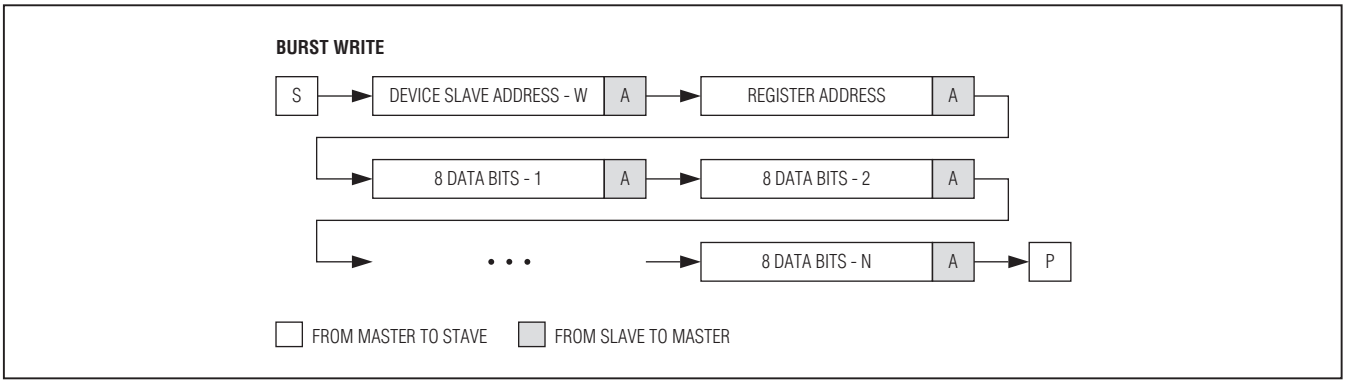


Figure 19. Burst Write Sequence

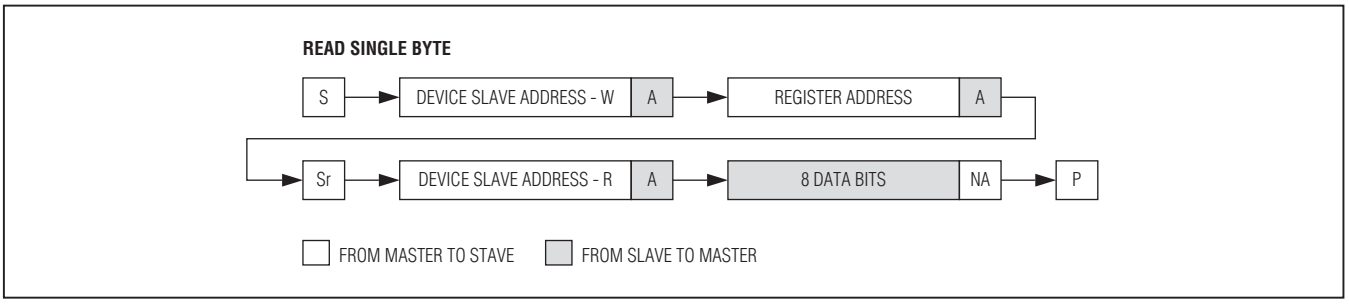


Figure 20. Read Byte Sequence

2) The master sends the 7-bit slave ID plus a write bit (low).
3) The addressed slave asserts an ACK on the data line.
4) The master sends the 8-bit register address.
5) The slave asserts an ACK on the data line only if the address is valid (NACK if not).
6) The master sends 8 bits of data.
7) The slave asserts an ACK on the data line.
8) Repeat steps 6 and 7 N - 1 times.
9) The master generates a STOP condition.

Single-Byte Read

With this operation the master sends an address and receives 1 or 2 data bytes from the slave device (Figure 20). The read byte procedure is as follows:

1) The master sends a START condition.
2) The master sends the 7-bit slave ID plus a write bit (low).
3) The addressed slave asserts an ACK on the data line.
4) The master sends the 8-bit register address.
5) The active slave asserts an ACK on the data line only if the address is valid (NACK if not).
6) The master sends a repeated START (Sr).

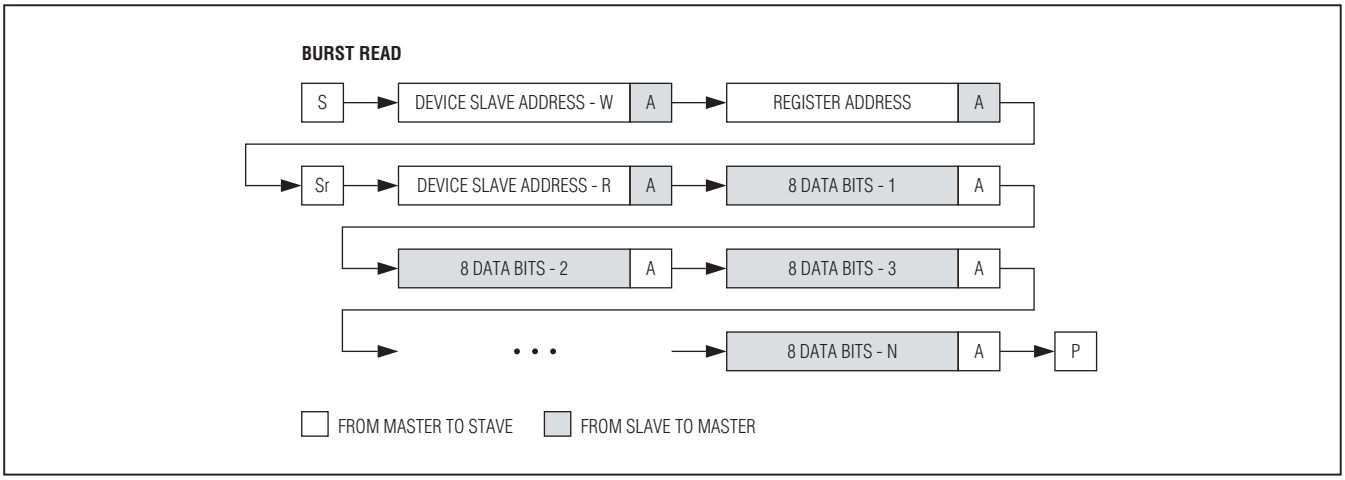


Figure 21. Burst Read Sequence

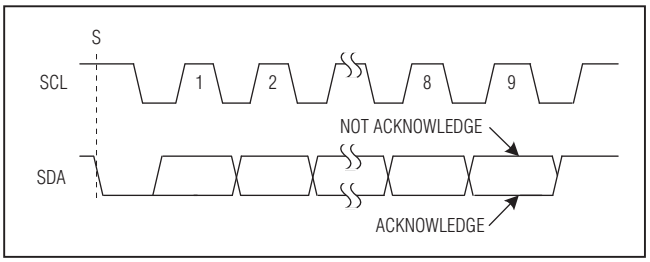


Figure 22. Acknowledge

7) The master sends the 7-bit slave ID plus a read bit (high).
8) The addressed slave asserts an ACK on the data line.
9) The slave sends 8 data bits.
10) The master asserts a NACK on the data line.
11) The master generates a STOP condition.

Burst Read

With this operation the master sends an address and receives multiple data bytes from the slave device (Figure 21). The burst read procedure is as follows:

1) The master sends a START condition.
2) The master sends the 7-bit slave ID plus a write bit (low).
3) The addressed slave asserts an ACK on the data line.
4) The master sends the 8-bit register address.
5) The slave asserts an ACK on the data line only if the address is valid (NACK if not).
6) The master sends a repeated START condition.
7) The master sends the 7-bit slave ID plus a read bit (high). 8) The slave asserts an ACK on the data line.
9) The slave sends 8 bits of data.
10) The master asserts an ACK on the data line.
11) Repeat steps 9 and 10 N - 1 times.
12) The master generates a STOP condition.

Acknowledge

Data transfers are acknowledged with an acknowledge bit (ACK) or a not-acknowledge bit (NACK). Both the master and the MAX3107 generate ACK bits. To generate an ACK, pull SDA low before the rising edge of the 9th clock pulse and keep it low during the high period of the 9th clock pulse (see Figure 22). To generate a NACK, leave SDA high before the rising edge of the 9th clock pulse and keep it high for the duration of the 9th clock pulse. Monitoring for NACK bits allows for detection of unsuccessful data transfers.

Applications Information

Startup and Initialization

The MAX3107 can be initialized following power-up or a hardware or software reset as shown in Figure 23. To verify that the MAX3107 is ready for operation after a power-up or reset, check the $\overline{IRQ}$ output if interrupt driven operation is employed.

In polled mode, repeatedly read a known register until the expected contents are returned. Note that the contents of the RevID change if new revisions of the product are released. If reading RevID, it is recommended to only check for the most significant 4 bits: Ah.

SPI/I²C UART with 128-Word FIFOs and Internal Oscillator

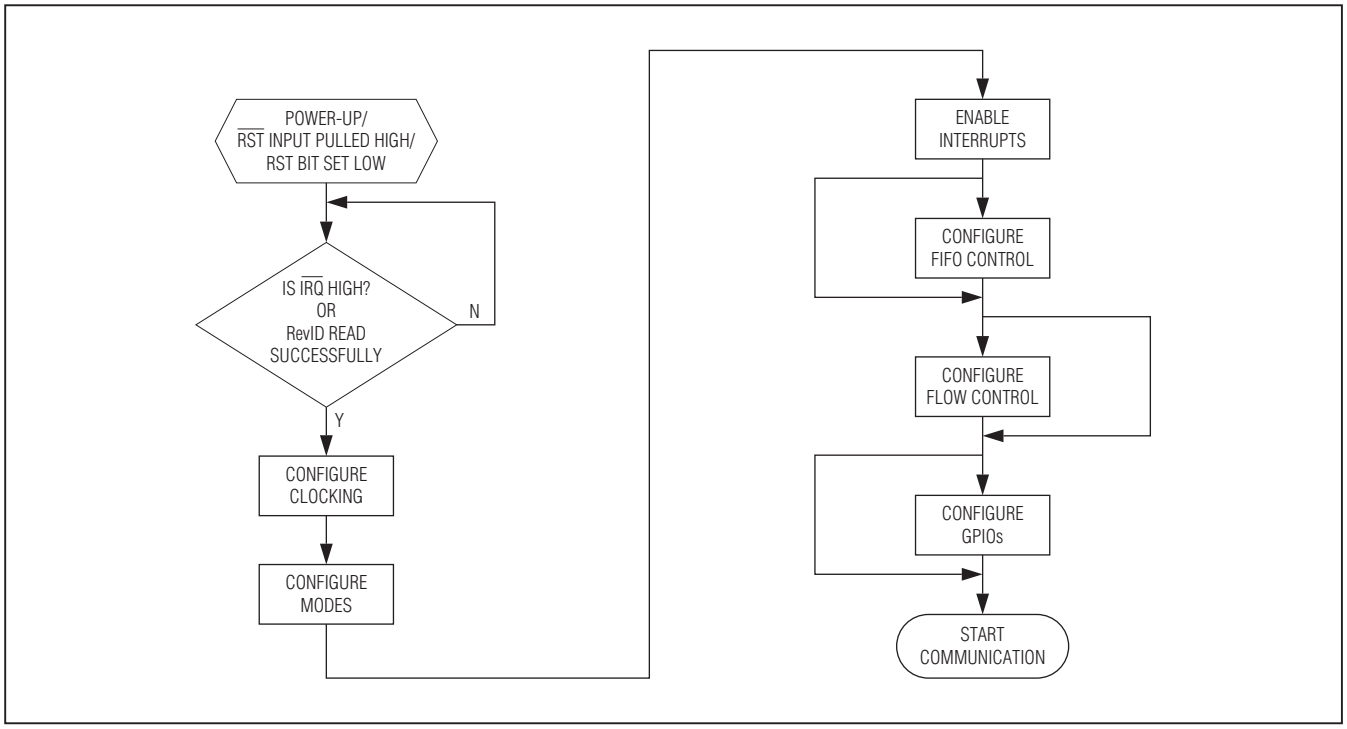


Figure 23. Startup and Initialization Flowchart

Low-Power Operation

To reduce the power consumption during normal operation, the following techniques can be adopted:

- Do not use the internal PLL. This saves the most power of the options listed here. Disable and bypass the PLL. With the PLL enabled, the current to the V_A supply is in the range of a few mA (depending on clock and multiplication factor), while it drops to below 1mA if disabled.
- Use an external clock source. Of the three clocking sources, the internal oscillator consumes the most power (about double that when using an external crystal).
- Keep the internal clock rates as low as possible.
- Use low voltage on the V_A supply.
- Use an external 1.8V supply. This saves the power dissipated in the internal 1.8V linear regulator for the 1.8V logic supply. Connect the external 1.8V supply to V_{18} and disable the internal regulator by connecting LDOEN to DGND.

Interrupts and Polling

The host controller can manage and control the MAX3107 through polling and/or through interrupts. In polled mode, the $\overline{IRQ}$ physical interrupt output is not used and the host controller polls the ISR register at frequent intervals to establish the state of the MAX3107.

Alternatively, the MAX3107's physical $\overline{IRQ}$ interrupt can be used to interrupt the host controller at specified events, making polling unnecessary. The $\overline{IRQ}$ output is an open-drain output that requires a pullup resistor to V_L.

Logic-Level Translation

The MAX3107 can be directly connected to transceivers and controllers that have different supply voltages. The V_L input defines the logic voltage levels of the controller interface while the V_{EXT} voltage defines the logic of the transceiver interface. This ensures flexibility when selecting a controller and transceiver. Figure 24 is an example of a setup when the controller, transceiver, and the MAX3107 are powered by three different supplies.

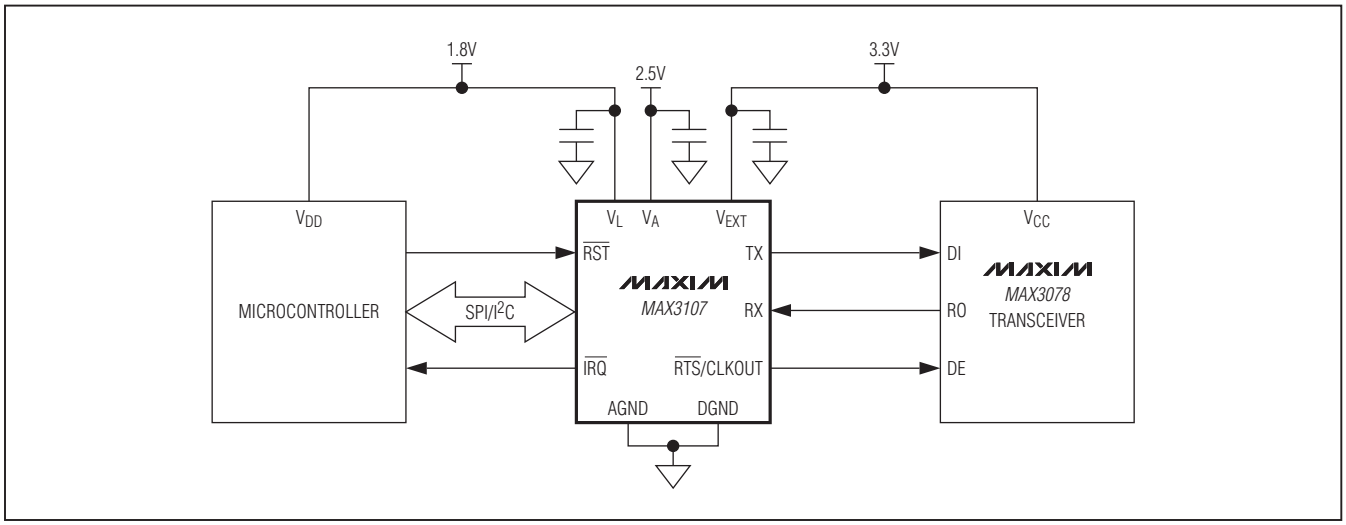


Figure 24. Logic-Level Translation

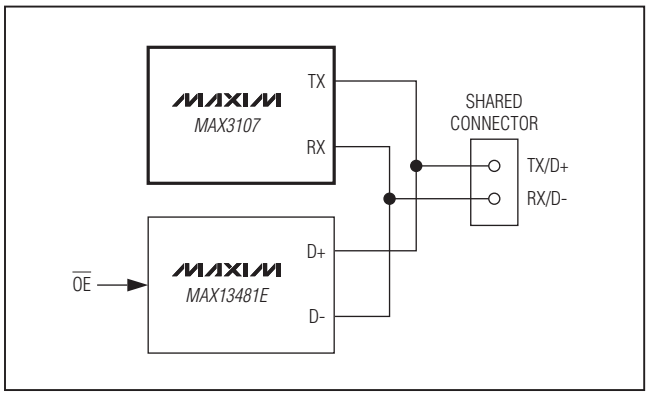


Figure 25. Connector Sharing with a USB Transceiver

Connector Pin Sharing

The TX and $\overline{RTS}$/CLKOUT outputs can be programmed to be high impedance. This can be used in cases where the MAX3107 shares a common connector with other communication devices. Set the output of the MAX3107 to high impedance when the other communication devices are active. Program MODE1[2]: TxHiZ high to set TX to a high-impedance state. Program MODE1[3]: RTSHiZ high to set $\overline{RTS}$/CLKOUT to a high-impedance state. Figure 25 shows an example of connector sharing with a USB transceiver.

RS-232 5x3 Application

The four GPIOs can be used to implement the other flow-control signals defined in ITU V.24. Figure 26 shows how the GPIOs create the DSR, DTR, DCD, and RI signals found on some RS-232/V.28 interfaces.

Set FlowCtrl[1:0] high to enable auto hardware $\overline{RTS}$/$\overline{CTS}$ flow control.

Typical Application Circuit

Figure 27 shows the MAX3107 being used in a half-duplex RS-485 application. The microcontroller, the RS-485 transceiver, and the MAX3107 are powered by 3.3V. SPI is used as the controller's communication interface. The internal oscillator clocks the UART.

The MAX14840 receiver is continually enabled so that echoing occurs. Enable auto echo suppression in the MAX3107 UART by setting MODE2[7]: EchoSuprs to 1.

Set MODE1[4]: TranscvCtrl high to enable auto transceiver direction control to automatically control the DE input of the transceiver.

Chip Information

PROCESS: BiCMOS

SPI/I²C UART with 128-Word FIFOs and Internal Oscillator

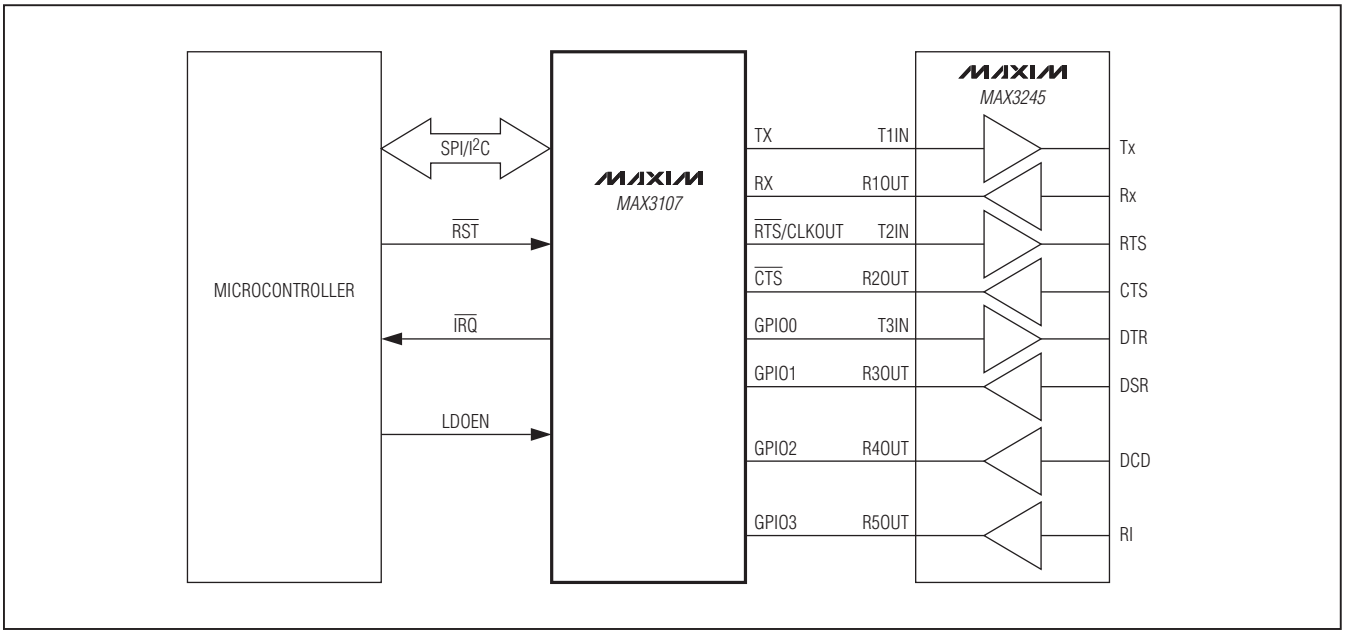


Figure 26. RS-232 Application

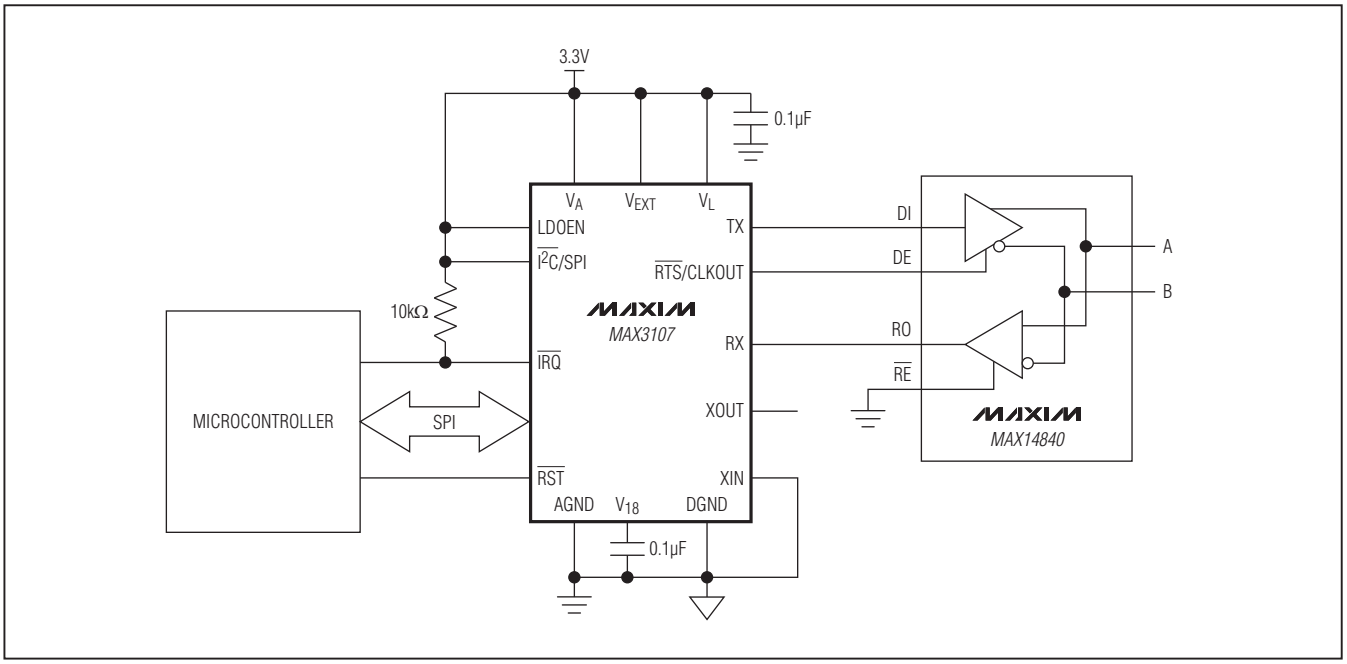
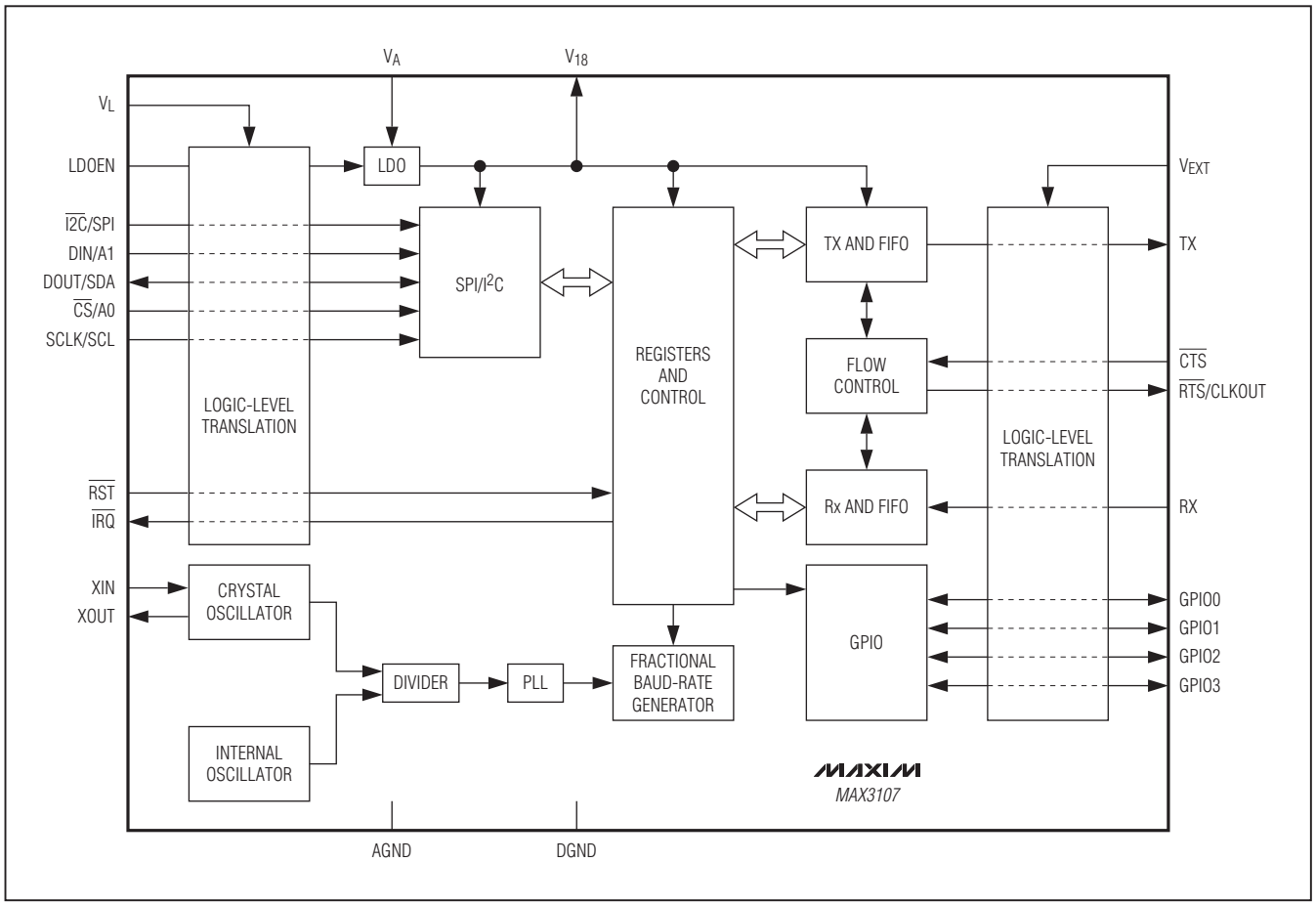


Figure 27. RS-485 Half-Duplex Application

Functional Diagram



Package Information

For the latest package outline information and land patterns, go to **www.maxim-ic.com/packages**. Note that a "+", "#", or "-" in the package code indicates RoHS status only. Package drawings may show a different suffix character, but the drawing pertains to the package regardless of RoHS status.

PACKAGE TYPE	PACKAGE CODE	DOCUMENT NO.
24 SSOP	A24+1	21-0056
24 TQFN-EP	T243A3+1	21-0188

SPI/I²C UART with 128-Word FIFOs and Internal Oscillator

Revision History

REVISION NUMBER	REVISION DATE	DESCRIPTION	PAGES CHANGED
0	10/09	Initial release	—
1	4/10	Changed the maximum number for the “External Clock Frequency” specification from 30MHz to 35MHz in the *AC Electrical Characteristics* table	8
		Replaced the text in the *SPI Burst Access* section	44
2	4/10	Increased the maximum V_{IL} specification for the XIN Clock Input in the *Electrical Characteristics* from 0.2V to 0.3V.	8