MAX14830

Quad Serial UART with 128-Word FIFOs

General Description

The MAX14830 is an advanced quad universal asynchronous receiver-transmitter (UART), each UART having 128 words of receive and transmit first-in/first-out (FIFO) and a high-speed serial peripheral interface (SPI) or I²C controller interface. A PLL and fractional baud-rate generators allow a high degree of flexibility in baud-rate programming and reference clock selection.

Each of the four UARTs is selected by in-band SPI/I²C addressing. Logic-level translation on the transceiver and controller interfaces allows ease of interfacing to microcontrollers, FPGAs, and transceivers that are powered by differing supply voltages.

Extensive features simplify transceiver control in half-duplex communication applications. The MAX14830 features the ability to synchronize the start of individual UART's transmission by SPI-based triggering. On-board timers allow programming of delays between transmitters as well as clock generation on GPIOs.

The 128-word FIFOs have advanced FIFO control reducing host processor data flow management.

The MAX14830 is available in a 48-pin TQFN (7mm x 7mm) package and is specified to operate. over the extended -40°C to +85°C temperature range.

Applications

- Industrial Control Systems
- Programmable Logic Controllers (PLC)
- IO-Link Master Controllers
- Medical Systems
- Point-of-Sales Systems
- Airplane Communication Buses

Typical Operating Circuit* and *Ordering Information* appear at end of data sheet.

Benefits and Features

- Bridges an SPI/MICROWIRE or I²C Microprocessor Bus to an Asynchronous Interface like RS-485, RS-232, or IrDA℠
  - SIR- and MIR-Compliant IrDA Encoder/Decoder
  - Line Noise Indication Ensures Data Link Integrity
- Deep, 128-Word Buffer and Automated Control Features Help Offload Activity on the Microcontroller
  - 128-Word Transmit and Receive FIFOs per UART
  - Transmitter Synchronization Through SPI Commands
  - Automatic Hardware Flow Control Using $\overline{RTS}$_ and $\overline{CTS}$_ Outputs and Inputs
  - Automatic Software Flow Control (XON/XOFF)
  - Auto Transceiver Direction Control
  - Programmable Setup and Hold Times for Transceiver Control
  - Auto Transmitter Disable
  - Half-Duplex Echo Suppression
  - 9-Bit Multidrop-Mode Data Filtering
    - Special Character Detection
    - GPIO-Based Character Detection
    - Four Timers Routed to GPIOs
    - 16 Flexible GPIOs with 20mA Drive Capability
- Saves Board Space
  - TQFN (7mm x 7mm) Package
- Fast Data Rates Allow Maximum System Flexibility Across Interface Standards
  - 6Mbaud (max) Data Rate in 16x Sampling Mode
  - 12/24Mbaud (max) Data Rate in 2x/4x Rate Modes
  - High-Resolution Programmable Baud-Rate
  - SPI Up to 26MHz Clock Rate
  - Fast Mode Plus (Pm+) I²C Up to 1MHz
- Integrated Internal Oscillator Eliminates the Need for an External Oscillator and Reduces the BOM Cost
  - Fractional Baud-Rate Generators, Predivider, and Phase-Locked Loop (PLL)
  - Logic-Level Translation Down to 1.61V on the Controller and Transceiver Interfaces Ensures System Compatibility
  - Register Compatible with MAX3107, MAX3108, MAX3109

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

19-5547; Rev 7; 2/16

maxim integrated™

TABLE OF CONTENTS

General Description . . . 1
Applications . . . 1
Benefits and Features . . . 1
Functional Diagram . . . 7
Absolute Maximum Ratings . . . 8
Package Thermal Characteristics . . . 8
DC Electrical Characteristics . . . 8
AC Electrical Characteristics . . . 10
Test Circuits/Timing Diagrams . . . 13
Typical Operating Characteristics . . . 14
Pin Configuration . . . 15
Pin Description . . . 15
Detailed Description . . . 18
- Receive and Transmit FIFOs . . . 18
- Transmitter Operation . . . 18
- Receiver Operation . . . 19
- Line Noise Indication . . . 20
- Clocking and Baud-Rate Generation . . . 20
  - Crystal Oscillator . . . 20
  - External Clock Source . . . 20
- PLL and Predivider . . . 20
- Fractional Baud-Rate Generators . . . 21
- 2x and 4x Rate Modes . . . 21
- Low-Frequency Timer . . . 22
- UART Clock to GPIO . . . 22
- Multidrop Mode . . . 22
  - Auto Data Filtering in Multidrop Mode . . . 22
- Auto Transceiver Direction Control . . . 22
- Transmitter Triggering and Synchronization . . . 23
  - Transmitter Synchronization . . . 23
  - Intrachip and Interchip Synchronization . . . 23
  - Delayed Triggering . . . 23
  - Trigger Accuracy . . . 24
  - Synchronization Accuracy . . . 24
- Auto Transmitter Disable . . . 24
- Echo Suppression . . . 24
- Auto Hardware Flow Control . . . 26
  - AutoRTS Control . . . 26

TABLE OF CONTENTS (continued)

AutoCTS Control . . . 26
FIFO Interrupt Triggering . . . 26
Auto Software (XON/XOFF) Flow Control . . . 26
Transmitter Flow Control . . . 27
Receiver Overflow Control . . . 27
Power-Up and $\overline{IRQ}$ . . . 27
Shutdown Mode . . . 27
Interrupt Structure . . . 27
Interrupt Enabling . . . 28
Interrupt Clearing . . . 28
Register Map . . . 28
Detailed Register Descriptions . . . 30
Serial Controller Interface . . . 58
SPI Interface . . . 58
MISO Operation . . . 58
SPI Burst Access . . . 58
Fast Read Cycle . . . 59
I^2C Interface . . . 59
START, STOP, and Repeated START Conditions . . . 59
Slave Address . . . 60
Bit Transfer . . . 61
Single-Byte Write . . . 61
Burst Write . . . 61
Single-Byte Read . . . 62
Burst Read . . . 62
Acknowledge Bits . . . 63
Applications Information . . . 63
Startup and Initialization . . . 63
Low-Power Operation . . . 63
Interrupts and Polling . . . 63
Logic-Level Translation . . . 63
IO-Link Application . . . 63
Typical Operating Circuit . . . 65
Chip Information . . . 67
Ordering Information . . . 67
Package Information . . . 67
Revision History . . . 68

LIST OF FIGURES

Figure 1. I²C Timing Diagram . . . 13
Figure 2. SPI Timing Diagram . . . 13
Figure 3. Transmit FIFO Signals . . . 18
Figure 4. Receive Data Format . . . 19
Figure 5. Receive FIFO . . . 19
Figure 6. Midbit Sampling . . . 19
Figure 7. Clock Selection Diagram . . . 20
Figure 8. 2x and 4x Baud Rates . . . 21
Figure 9. GPIO_ Clock Pulse Generator . . . 22
Figure 10. Auto Transceiver Direction Control . . . 23
Figure 11. Setup and Hold times in Auto Transceiver Direction Control . . . 23
Figure 12. Single Transmitter Trigger Accuracy . . . 24
Figure 13. Multiple Transmitter Synchronization Accuracy . . . 25
Figure 14. Echo Suppression Timing . . . 25
Figure 15. Half-Duplex with Echo Suppression . . . 26
Figure 16. Simplified Interrupt Structure . . . 27
Figure 17. PLL Signal Path . . . 51
Figure 18. SPI Write Cycle . . . 58
Figure 19. SPI Read Cycle . . . 59
Figure 20. SPI Fast Read Cycle . . . 59
Figure 21. I²C START, STOP, and Repeated START Conditions . . . 60
Figure 22. Write Byte Sequence . . . 61
Figure 23. Burst Write Sequence . . . 61
Figure 24. Read Byte Sequence . . . 62
Figure 25. Burst Read Sequence . . . 62
Figure 26. Acknowledge Bits . . . 63
Figure 27. Startup and Initialization Flow Chart . . . 63
Figure 28. Logic-Level Translation . . . 64
Figure 29. Interchip Synchronization . . . 64

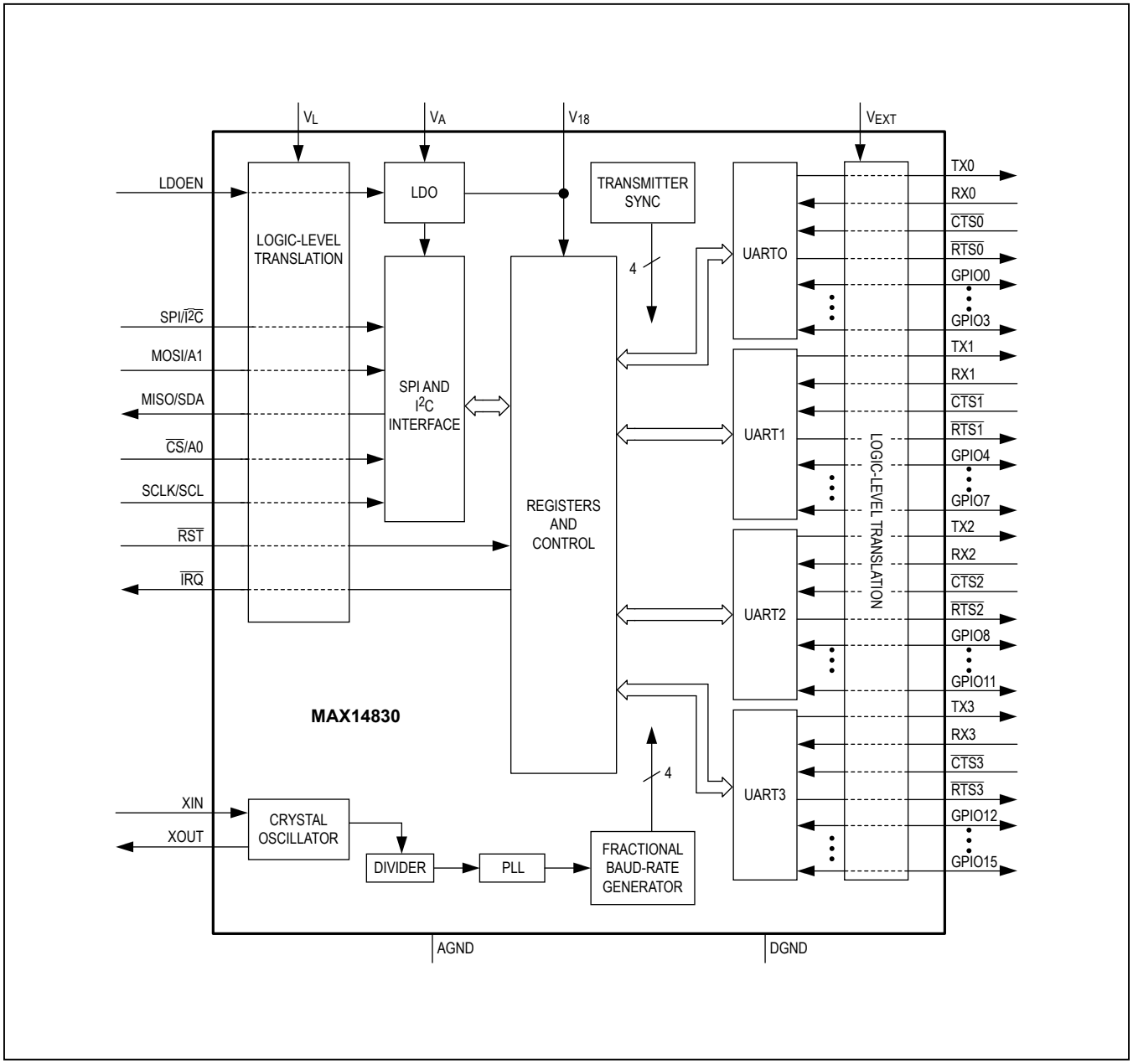
LIST OF TABLES

Table 1. UART GPIO Assignments for GPIO Interrupts 38
Table 2. StopBits Truth Table 41
Table 3. Length_ Truth Table 41
Table 4. SwFlow_ Truth Table 46
Table 5. UART GPIO Assignments for GPIO Configuration 49
Table 6. UART GPIO Assignments for GPIO Input/Output Data. 50
Table 7. PLLFactor_ Selector Guide 51
Table 8. GloblComnd Command Descriptions. 54
Table 9. Extended Mode Addressing (SPI only) 54
Table 10. SPI Command Byte Configuration 58
Table 11. SPI U1, U0 UART Selection 58
Table 12. I^2C Address Map 60

LIST OF REGISTERS

RHR—Receive Hold Register . . . 30
THR—Transmit Hold Register . . . 30
IRQEn—IRQ Enable Register . . . 31
ISR—Interrupt Status Register . . . 32
LSRIntEn—Line Status Interrupt Enable Register . . . 33
LSR—Line Status Register . . . 34
SpclChrIntEn—Special Character Interrupt Enable Register . . . 35
SpclCharInt—Special Character Interrupt Register . . . 36
STSIntEn—STS Interrupt Enable Register . . . 37
STSInt—Status Interrupt Register . . . 38
MODE1 Register . . . 39
MODE2 Register . . . 40
LCR—Line Control Register . . . 41
RxTimeOut—Receiver Timeout Register . . . 42
HDplxDelay Register . . . 42
IrDA Register . . . 43
FlowLvl—Flow Level Register . . . 44
FIFOTrigLvl—FIFO Interrupt Trigger Level Register . . . 44
TxFIFOLvl—Transmit FIFO Level Register . . . 45
RxFIFOLvl—Receive FIFO Level Register . . . 45
FlowCtrl—Flow Control Register . . . 45
XON1 Register . . . 47
XON2 Register . . . 47
XOFF1 Register . . . 48
XOFF2 Register . . . 48
GPIOConfg—GPIO Configuration Register . . . 49
GPIOData—GPIO Data Register . . . 50
PLLConfig—PLL Configuration Register . . . 51
BRGConfig—Baud-Rate Generator Configuration Register . . . 52
DIVLSB—Baud-Rate Generator LSB Divisor Register . . . 52
DIVMSB—Baud-Rate Generator MSB Divisor Register . . . 52
CLKSource—Clock Source Register . . . 53
GlobalIRQ—Global IRQ Register . . . 53
GloblComnd—Global Command Register . . . 54
TxSynch—Transmitter Synchronization Register . . . 55
SynchDelay1—Synchronization Delay Register 1 . . . 56
SynchDelay2—Synchronization Delay Register 2 . . . 56
TIMER1—Timer Register 1 . . . 56
TIMER2—Timer Register 2 . . . 57
RevID—Revision Identification Register . . . 57

Functional Diagram



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 the lesser of (V_A + 0.3V) and +2.0V
$\overline{RST}$, $\overline{IRQ}$, MOSI/A1, $\overline{CS}$/A0, SCLK/SCL,
MISO/SDA, LDOEN, SPI/$\overline{I2C}$.....-0.3V to (V_L + 0.3V)
TX0, RX0, $\overline{CTS0}$, GPIO0, GPIO1,
GPIO2, GPIO3.....-0.3V to (V_{EXT} + 0.3V)
TX1, RX1, $\overline{CTS1}$, GPIO4, GPIO5,
GPIO6, GPIO7.....-0.3V to (V_{EXT} + 0.3V)
TX2, RX2, $\overline{CTS2}$, GPIO8, GPIO9,
GPIO10, GPIO11.....-0.3V to (V_{EXT} + 0.3V)
TX3, RX3, $\overline{CTS3}$, GPIO12, GPIO13,
GPIO14, GPIO15.....-0.3V to (V_{EXT} + 0.3V)
DGND..... -0.3V to +0.3V
Continuous Power Dissipation (T_A = +70°C)
TQFN (derate 38.5mW/°C above +70°C).....3076.9mW
Operating Temperature Range.....-40°C to +85°C
Maximum Junction Temperature.....+150°C
Storage Temperature Range.....-65°C to +150°C
Lead Temperature (soldering, 10s).....300°C
Soldering Temperature (reflow) .....+260°C

Package Thermal Characteristics (Note 1)

TQFN
Junction-to-Ambient Thermal Resistance (θ_{JA}) .....26°C/W
Junction-to-CaseThermal Resistance (θ_{JC}).....1°C/W

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

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

DC Electrical Characteristics

(V_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 = +2.5V, V_L = +1.8V, V_{EXT} = +2.8V, T_A = +25°C.) (Notes 2, 3)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Digital Interface Supply Voltage	V_L		1.71		3.6	V
Analog Supply Voltage	V_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		1.95	V
CURRENT CONSUMPTION						
V_A Supply Current	I_A	1.8MHz crystal oscillator active, PLL disabled, SPI/I²C interface idle, UART interfaces idle, V_{LDOEN} = V_L			400	µA
		Baud rate = 1Mbps, 20MHz external clock, SPI/I²C interface idle, PLL disabled, all UARTs in loopback mode, V_{LDOEN} = 0V			0.5	mA
V_A Shutdown Supply Current	I_{ASHDN}	Shutdown mode, V_{LDOEN} = 0V, $V_{\overline{RST}}$ = 0V, all inputs and outputs are idle			35	µA
V_L Shutdown or Sleep Supply Current	I_L	Shutdown mode, V_{LDOEN} = 0V, $V_{\overline{RST}}$ = 0V, all inputs and outputs are idle			12	µA
V_{EXT} Shutdown Supply Current	I_{EXT}	Shutdown mode, V_{LDOEN} = 0V, $V_{\overline{RST}}$ = 0V, all inputs and outputs are idle			8	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 = +2.5V, V_L = +1.8V, V_{EXT} = +2.8V, T_A = +25°C.) (Notes 2, 3)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
V_{18} Input Power-Supply Current in Shutdown Mode	I_{18SHDN}	Shutdown mode, V_{LDOEN} = 0V, $V_{\overline{RST}}$ = 0V, all inputs and outputs are idle			200	µA
V_{18} Input Power-Supply Current	I_{18}	Baud rate = 1Mbps, 20MHz external clock, PLL disabled, all UARTs in loopback mode, V_{LDOEN} = 0V (Note 4)			5	mA
SCLK/SCL, MISO/SDA						
MISO/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	
MISO/SDA Output Low Voltage in SPI Mode	$V_{OL,SPI}$	I_{LOAD} = -2mA			0.4	V
MISO/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}	SPI and I²C mode		5		pF
SPI/$\overline{I2C}$, $\overline{CS}$/A0, MOSI/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}	SPI and I²C mode		5		pF
$\overline{IRQ}$ OUTPUT (OPEN DRAIN)						
Output Low Voltage	V_{OL}	I_{LOAD} = -2mA			0.4	V
Output Leakage Current	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
UART INTERFACE						
$\overline{RTS0}$, $\overline{RTS1}$, $\overline{RTS2}$, $\overline{RTS3}$, TX0, TX1, TX2, TX3 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 is three-stated, $V_{\overline{RTS_}}$ = 0 to V_{EXT}	-1		+1	µA
Input Capacitance	C_{IN}	High-Z mode		5		pF

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 = +2.5V, V_L = +1.8V, V_{EXT} = +2.8V, T_A = +25°C.) (Notes 2, 3)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
RX0, RX1, RX2, RX3, $\overline{CTS0}$, $\overline{CTS1}$, $\overline{CTS2}$, $\overline{CTS3}$ 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
$\overline{CTS0}$, $\overline{CTS1}$, $\overline{CTS2}$, $\overline{CTS3}$ Input Leakage Current	I_{IN_CTS}	$V_{\overline{CTS_}}$ = 0 to V_{EXT}	-1		+1	µA
RX0, RX1, RX2, RX3 Pullup Current	$I_{IN_RX_}$	$V_{RX_}$ = 0V, V_{EXT} = 3.6V	-7.5	-5.5	-3.5	µA
Input Capacitance	C_{IN_UART}			5		pF
GPIO0–GPIO15 INPUTS/OUTPUTS						
Output Low Voltage	V_{OL}	I_{LOAD} = -20mA, V_{EXT} > 2.3V, push-pull or open drain			0.45	V
		I_{LOAD} = -20mA, V_{EXT} < 2.3V, push-pull or open drain			0.55	
Output High Voltage	V_{OH}	I_{LOAD} = 5mA, push-pull			V_{EXT} - 0.4	V
Input Low Voltage	V_{IL}	GPIO_ is configured as an input			0.4	V
Input High Voltage	V_{IH}	GPIO_ is configured as an input	2/3 x V_{EXT}			V
Pulldown Current	I_{PD}	GPIO_ = V_{EXT} = 3.6V	3.5	5.5	7.5	µA
XIN						
Input Low Voltage	V_{IL}				0.2	V
Input High Voltage	V_{IH}		1.2			V
Input Capacitance	C_{XIN}			16		pF
XOUT						
Input Capacitance	C_{XOUT}			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.) (Notes 2, 3)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
INTERNAL OSCILLATOR						
External Crystal Frequency	f_{XOSC}		1		4	MHz
External Clock Frequency	f_{CLK}		0.5		35	MHz
External Clock Duty Cycle		(Note 5)	45		55	%
Baud-Rate Generator Clock Input	f_{REF}	(Note 5)			96	MHz

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.) (Notes 2, 3)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
I²C BUS: TIMING CHARACTERISTICS (SEE FIGURE 1)						
SCL Clock Frequency	f_{SCL}	Standard mode			100	kHz
		Fast mode			400	
		Fast mode plus			1000	
Bus Free Time Between a STOP and START Condition	t_{BUF}	Standard mode	4.7			µs
		Fast mode	1.3			
		Fast mode plus	0.5			
Hold Time for START Condition and Repeated START Condition	$t_{HD:STA}$	Standard mode	4.0			µs
		Fast mode	0.6			
		Fast mode plus	0.26			
Low Period of the SCL Clock	t_{LOW}	Standard mode	4.7			µs
		Fast mode	1.3			
		Fast mode plus	0.5			
High Period of the SCL Clock	t_{HIGH}	Standard mode	4.0			µs
		Fast mode	0.6			
		Fast mode plus	0.26			
Data Hold Time	$t_{HD:DAT}$	Standard mode	0		0.9	µs
		Fast mode	0		0.9	
		Fast mode plus	0			
Data Setup Time	$t_{SU:DAT}$	Standard mode	250			ns
		Fast mode	100			
		Fast mode plus	50			
Setup Time for Repeated START Condition	$t_{SU:STA}$	Standard mode	4.7			µs
		Fast mode	0.6			
		Fast mode plus	0.26			
Rise Time of SDA and SCL Signals Receiving	t_R	Standard mode (0.3 x V_L to 0.7 x V_L) (Note 6)	20 + 0.1C_b		1000	ns
		Fast mode (0.3 x V_L to 0.7 x V_L) (Note 6)	20 + 0.1C_b		300	
		Fast mode plus			120	
Fall Time of SDA and SCL Signals	t_F	Standard mode (0.7 x V_L to 0.3 x V_L) (Note 6)	20 + 0.1C_b		300	ns
		Fast mode (0.7 x V_L to 0.3 x V_L) (Note 6)	20 + 0.1C_b		300	
		Fast mode plus			120	
Setup Time for STOP Condition	$t_{SU:STO}$	Standard mode	4.7			µs
		Fast mode	0.6			
		Fast mode plus	0.26			
Capacitive Load for SDA and SCL (Note 4)	C_b	Standard mode			400	pF
		Fast mode			400	
		Fast mode plus			550	

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.) (Notes 2, 3)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
SCL and SDA I/O Capacitance	$C_{I/O}$	(Note 5)			10	pF
Pulse Width of Spike Suppressed	t_{SP}				50	ns
SPI BUS: TIMING CHARACTERISTICS (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
CS Fall to SCLK Rise Time	t_{CSS}		0			ns
MOSI Hold Time	t_{DH}		3			ns
MOSI Setup Time	t_{DS}		5			ns
Output Data Propagation Delay	t_{DO}				20	ns
MISO Rise and Fall Times	t_{FT}				10	ns
$\overline{CS}$ Hold Time	t_{CSH}		30			ns

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

Test Circuits/Timing Diagrams

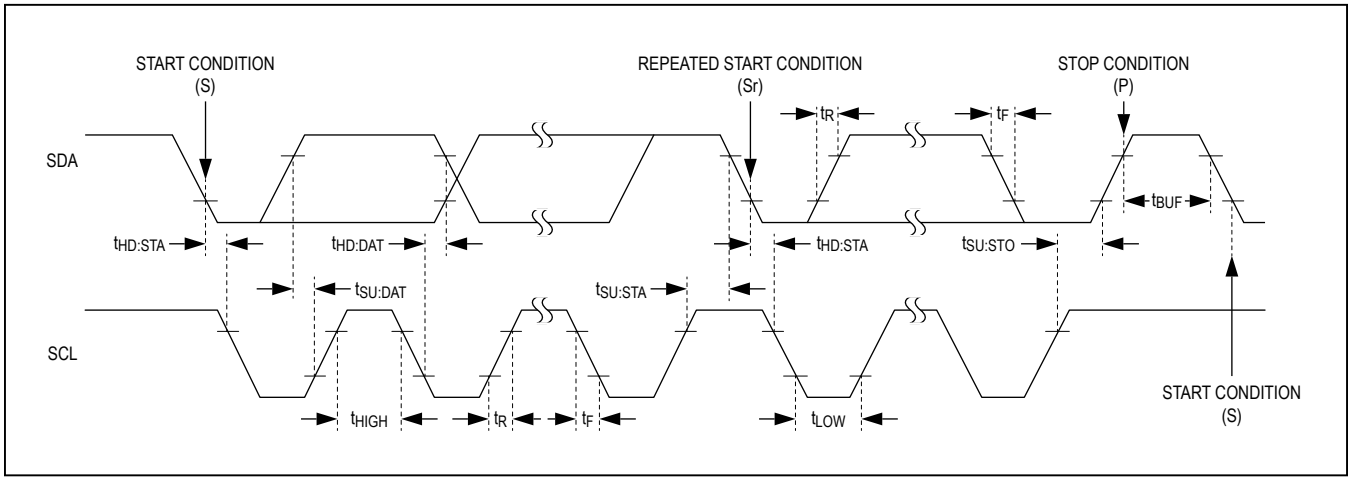


Figure 1. I²C Timing Diagram

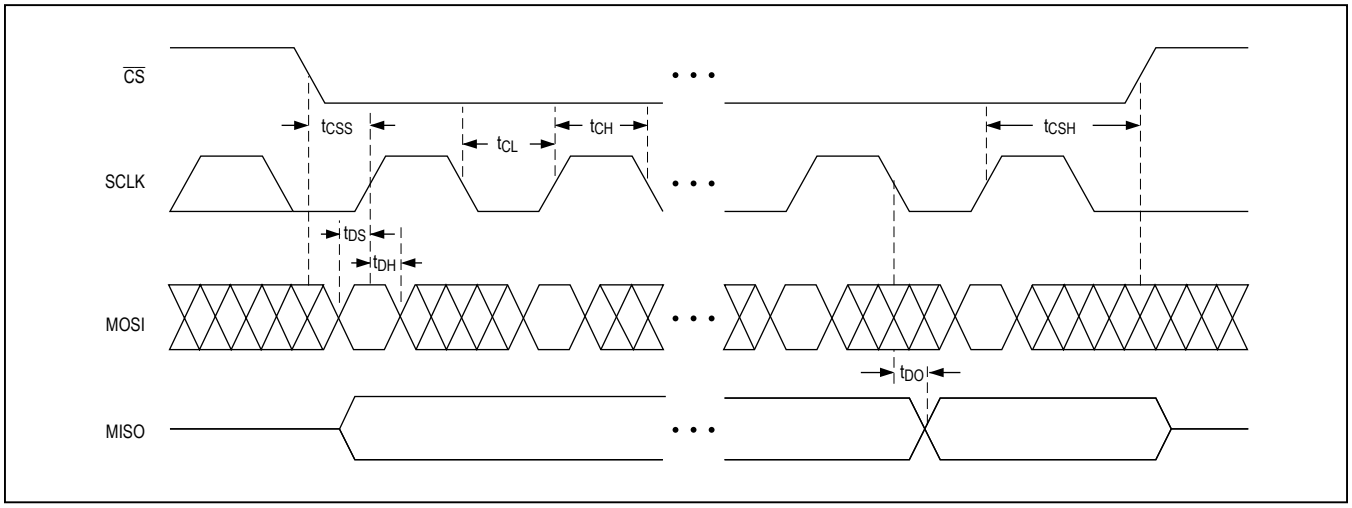
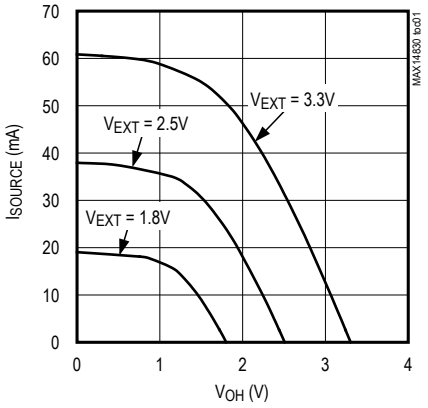


Figure 2. SPI Timing Diagram

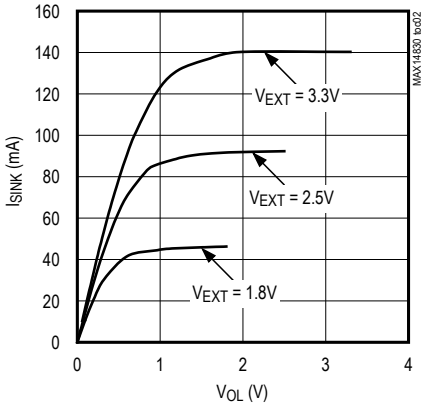
Typical Operating Characteristics

(T_A = +25°C, unless otherwise noted.)

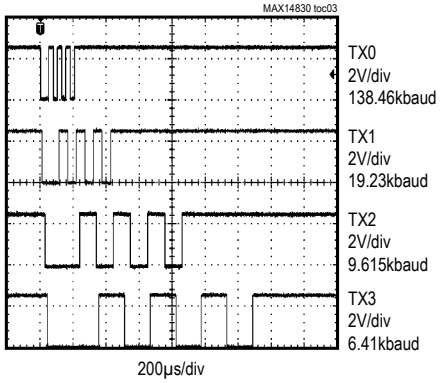
GPIO_ OUTPUT HIGH VOLTAGE vs. SOURCE CURRENT (PUSH-PULL)



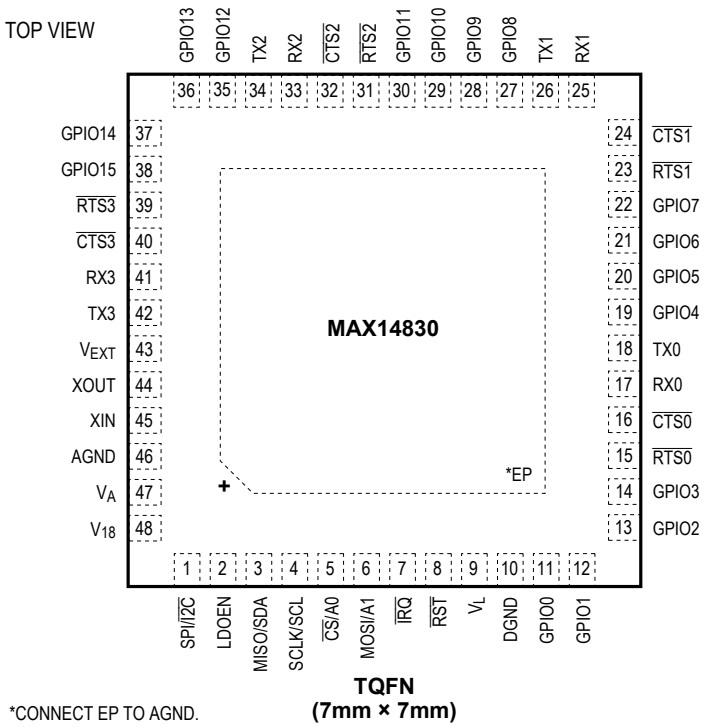
GPIO_ OUTPUT LOW VOLTAGE vs. SINK CURRENT (PUSH-PULL)



TRANSMITTER SYNCHRONIZATION



Pin Configuration



Pin Description

PIN	NAME	FUNCTION
1	SPI/$\overline{I2C}$	SPI or Active-Low I²C Selector Input. Drive SPI/$\overline{I2C}$ high to enable SPI. Drive SPI/$\overline{I2C}$ low to enable I²C.
2	LDOEN	LDO Enable Input. Drive LDOEN high to enable the internal 1.8V LDO. Drive LDOEN low to disable the internal LDO. When LDOEN is low, V_{18} can be supplied by an external voltage source.
3	MISO/SDA	Serial-Data Output. When SPI/$\overline{I2C}$ is high, MISO/SDA functions as the MISO, SPI serial-data output. When SPI/$\overline{I2C}$ is low, MISO/SDA functions as the SDA, I²C serial-data input/output.
4	SCLK/SCL	Serial-Clock Input. When SPI/$\overline{I2C}$ is high, SCLK/SCL functions as the SCLK, SPI serial-clock input (up to 26MHz). When SPI/$\overline{I2C}$ is low, SCLK/SCL functions as the SCL, I²C serial-clock input (up to 1MHz).
5	$\overline{CS}$/A0	Active-Low Chip-Select and Address 0 Input. When SPI/$\overline{I2C}$ is high, $\overline{CS}$/A0 functions as the $\overline{CS}$, SPI active-low chip-select input. When SPI/$\overline{I2C}$ is low, $\overline{CS}$/A0 functions as the A0, I²C device address programming input. Connect $\overline{CS}$/A0 to SDA, SCL, DGND, or V_L when SPI/$\overline{I2C}$ is low.
6	MOSI/A1	Serial-Data and Address 1 Input. When SPI/$\overline{I2C}$ is high, MOSI/A1 functions as the MOSI, SPI serial-data input. When SPI/$\overline{I2C}$ is low, MOSI/A1 functions as the A1, I²C device address programming input. Connect MOSI/A1 to SDA, SCL, DGND, or V_L when SPI/$\overline{I2C}$ is low.
7	$\overline{IRQ}$	Active-Low Interrupt Open-Drain Output. $\overline{IRQ}$ is asserted when an interrupt is pending.
8	$\overline{RST}$	Active-Low Reset Input. Drive $\overline{RST}$ low to force all of the UARTs into hardware reset mode. In hardware reset mode, the oscillator and the internal PLL are shut down and there is no clock activity.

Pin Description (continued)

PIN	NAME	FUNCTION
9	V_L	Digital Interface Logic-Level Supply. V_L powers the internal logic-level translators for $\overline{RST}$, $\overline{IRQ}$, MOSI/A1, $\overline{CS}$/A0, SCLK/SCL, MISO/SDA, LDOEN, and SPI/$\overline{I2C}$. Bypass V_L with a 0.1µF ceramic capacitor to DGND.
10	DGND	Digital Ground
11	GPIO0	General-Purpose Input/Output 0. GPIO0 is user-programmable as an input or output (push-pull or open drain) or external event interrupt source. GPIO0 has a weak pulldown resistor to ground. GPIO0 is the reference clock output when bit 7 of the TxSynch register is set to 1 (see the *UART Clock to GPIO* section for more information).
12	GPIO1	General-Purpose Input/Output 1. GPIO1 is user-programmable as an input or output (push-pull or open drain) or external event interrupt source. GPIO1 has a weak pulldown resistor to ground. GPIO1 is the TIMER output when bit 7 of the TIMER2 register is set to 1.
13	GPIO2	General-Purpose Input/Output 2. GPIO2 is user-programmable as an input or output (push-pull or open drain) or external event interrupt source. GPIO2 has a weak pulldown resistor to ground.
14	GPIO3	General-Purpose Input/Output 3. GPIO3 is user-programmable as an input or output (push-pull or open drain) or external event interrupt source. GPIO3 has a weak pulldown resistor to ground.
15	$\overline{RTS0}$	Active-Low Request-to-Send Output for UART0. $\overline{RTS0}$ can be set high or low by programming the LCR register. $\overline{RTS0}$ is the UART system clock/fractional divider output when bit 7 of the CLKSource register is set to 1.
16	$\overline{CTS0}$	Active-Low Clear-to-Send Input for UART0. $\overline{CTS0}$ is a flow control status input.
17	RX0	Serial Receiving Data Input for UART0. RX0 has a weak pullup to V_{EXT}.
18	TX0	Serial Transmitting Data Output for UART0
19	GPIO4	General-Purpose Input/Output 4. GPIO4 is user-programmable as an input or output (push-pull or open drain) or external event interrupt source. GPIO4 has a weak pulldown resistor to ground. GPIO4 is the reference clock output when bit 7 of the TxSynch register is set to 1 (see the *UART Clock to GPIO* section for more information).
20	GPIO5	General-Purpose Input/Output 5. GPIO5 is user-programmable as an input or output (push-pull or open drain) or external event interrupt source. GPIO5 has a weak pulldown resistor to ground. GPIO5 is the TIMER output when bit 7 of the TIMER2 register is set to 1.
21	GPIO6	General-Purpose Input/Output 6. GPIO6 is user-programmable as an input or output (push-pull or open drain) or external event interrupt source. GPIO6 has a weak pulldown resistor to ground.
22	GPIO7	General-Purpose Input/Output 7. GPIO7 is user-programmable as an input or output (push-pull or open drain) or external event interrupt source. GPIO7 has a weak pulldown resistor to ground.
23	$\overline{RTS1}$	Active-Low Request-to-Send Output for UART1. $\overline{RTS1}$ can be set high or low by programming the LCR register. $\overline{RTS1}$ is the UART system clock/fractional divider output when bit 7 of the CLKSource register is set to 1.
24	$\overline{CTS1}$	Active-Low Clear-to-Send Input for UART1. $\overline{CTS1}$ is a flow control status input.
25	RX1	Serial Receiving Data Input for UART1. RX1 has a weak pullup to V_{EXT}.
26	TX1	Serial Transmitting Data Output for UART1
27	GPIO8	General-Purpose Input/Output 8. GPIO8 is user-programmable as an input or output (push-pull or open drain) or external event interrupt source. GPIO8 has a weak pulldown resistor to ground. GPIO8 is the reference clock output when bit 7 of the TxSynch register is set to 1 (see the *UART Clock to GPIO* section for more information).

Pin Description (continued)

PIN	NAME	FUNCTION
28	GPIO9	General-Purpose Input/Output 9. GPIO9 is user-programmable as an input or output (push-pull or open drain) or external event interrupt source. GPIO9 has a weak pulldown resistor to ground. GPIO9 is the TIMER output when bit 7 of the TIMER2 register is set to 1.
29	GPIO10	General-Purpose Input/Output 10. GPIO10 is user-programmable as an input or output (push-pull or open drain) or external event interrupt source. GPIO10 has a weak pulldown resistor to ground.
30	GPIO11	General-Purpose Input/Output 11. GPIO11 is user-programmable as an input or output (push-pull or open drain) or external event interrupt source. GPIO11 has a weak pulldown resistor to ground.
31	$\overline{RTS2}$	Active-Low Request-to-Send Output for UART2. $\overline{RTS2}$ can be set high or low by programming the LCR register. $\overline{RTS2}$ is the UART system clock/fractional divider output when bit 7 of the CLKSource register is set to 1.
32	$\overline{CTS2}$	Active-Low Clear-to-Send Input for UART2. $\overline{CTS2}$ is a flow control status input.
33	RX2	Serial Receiving Data Input for UART2. RX2 has a weak pullup to V_{EXT}.
34	TX2	Serial Transmitting Data Output for UART2
35	GPIO12	General-Purpose Input/Output 12. GPIO12 is user-programmable as an input or output (push-pull or open drain) or external event interrupt source. GPIO12 has a weak pulldown resistor to ground. GPIO12 is the reference clock output when bit 7 of the TxSynch register is set to 1 (see the *UART Clock to GPIO* section for more information).
36	GPIO13	General-Purpose Input/Output 13. GPIO13 is user-programmable as an input or output (push-pull or open drain) or external event interrupt source. GPIO13 has a weak pulldown resistor to ground. GPIO13 is the TIMER output if bit 7 of the TIMER2 register is set to 1.
37	GPIO14	General-Purpose Input/Output 14. GPIO14 is user-programmable as an input or output (push-pull or open drain) or external event interrupt source. GPIO14 has a weak pulldown resistor to ground.
38	GPIO15	General-Purpose Input/Output 15. GPIO15 is user-programmable as an input or output (push-pull or open drain) or external event interrupt source. GPIO15 has a weak pulldown resistor to ground.
39	$\overline{RTS3}$	Active-Low Request-to-Send Output for UART3. $\overline{RTS3}$ can be set high or low by programming the LCR register. $\overline{RTS3}$ is the UART system clock/fractional divider output when bit 7 of the CLKSource register is set to 1.
40	$\overline{CTS3}$	Active-Low Clear-to-Send Input for UART3. $\overline{CTS3}$ is a flow control status input.
41	RX3	Serial Receiving Data Input for UART3. RX3 has a weak pullup to V_{EXT}.
42	TX3	Serial Transmitting Data Output for UART3
43	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.
44	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, leave XOUT unconnected.
45	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.
46	AGND	Analog Ground
47	V_A	Analog Supply. V_A powers the PLL, and the internal LDO. Bypass V_A with a 0.1µF ceramic capacitor to AGND.
48	V_{18}	Internal 1.8V LDO Output and 1.8V Logic Supply Input. Bypass V_{18} with a 1µF ceramic capacitor to DGND.
—	EP	Exposed Paddle. Connect EP to AGND. Do not use EP as the main AGND connection.

Detailed Description

The MAX14830 quad UART bridges an SPI/MICROWIRE® or I²C microprocessor bus to an asynchronous interface like RS-485, RS-232, or IrDA. The MAX14830 contains advanced UARTs and baud-rate generators with a synchronous serial-data interface and an interrupt generator. The MAX14830 is configured by writing an 8-bit word to the configuration registers through either SPI or I²C. These registers are organized by related function as shown in the *Register Map*.

The host controller loads transmit data into the THR register through SPI or I²C. This data is automatically pushed into the Transmit FIFOs, formatted, and sent out at TX_. The MAX14830 adds START and STOP and parity bits to the data and sends the data out at the selected baud rates. The clock configuration registers determine the baud rates, clock source selection, clock frequency prescaling, and fractional baud-rate generators.

The MAX14830 receiver detects a START bit as a high-to-low RX_ transition. An internal clock samples this data at 16 times the data rate. The received data is automatically placed in the Receive FIFOs and can then be read out of the RxFIFOs through the RHRs.

The MAX14830 features four identical UARTS. Text in this data sheet references individual UART operation, unless otherwise noted.

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_ and 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 and 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 FIFOTrigLvl 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, 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 at any time 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 the Transmit Hold Register (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]:TFifoEmptyInt 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).

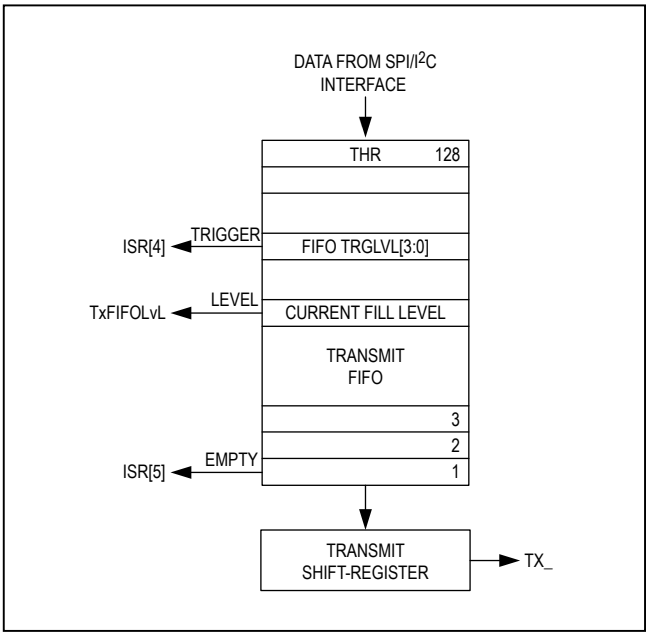


Figure 3. Transmit FIFO Signals

MICROWIRE is a registered trademark of National Semiconductor Corp.

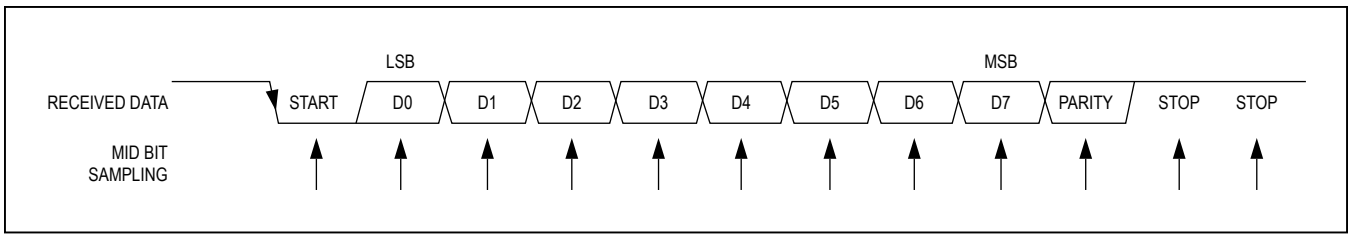


Figure 4. Receive Data Format

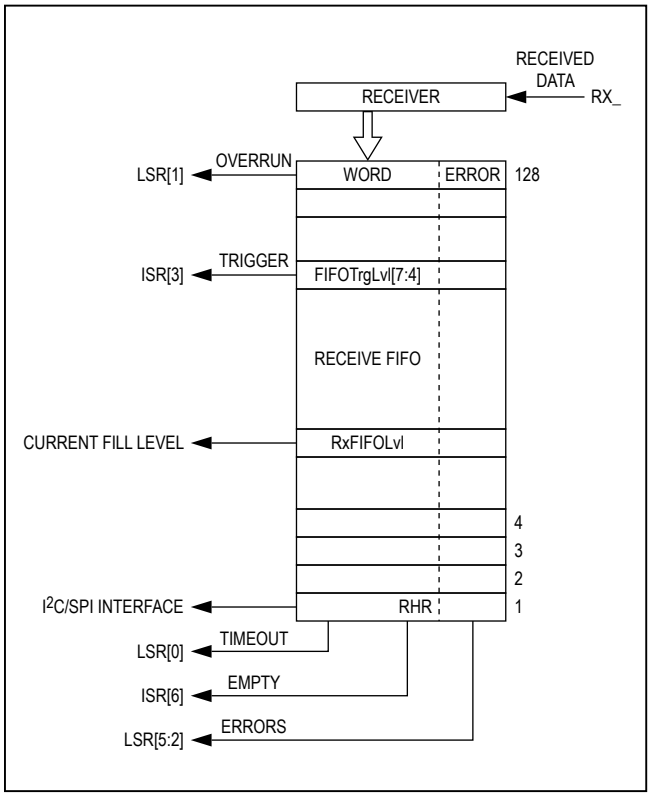


Figure 5. Receive FIFO

The contents of the TxFIFO and RxFIFOs are both cleared through MODE2[1]: FIFORst. 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 that 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 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 5). The host reads the data out of the Receive FIFO through the Receive Hold Register (RHR), oldest data first. The status information of the most recently read 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 6).

The receiver can be turned off through MODE1[0]: RxDisabl. When this bit is set to 1, the MAX14830 turns

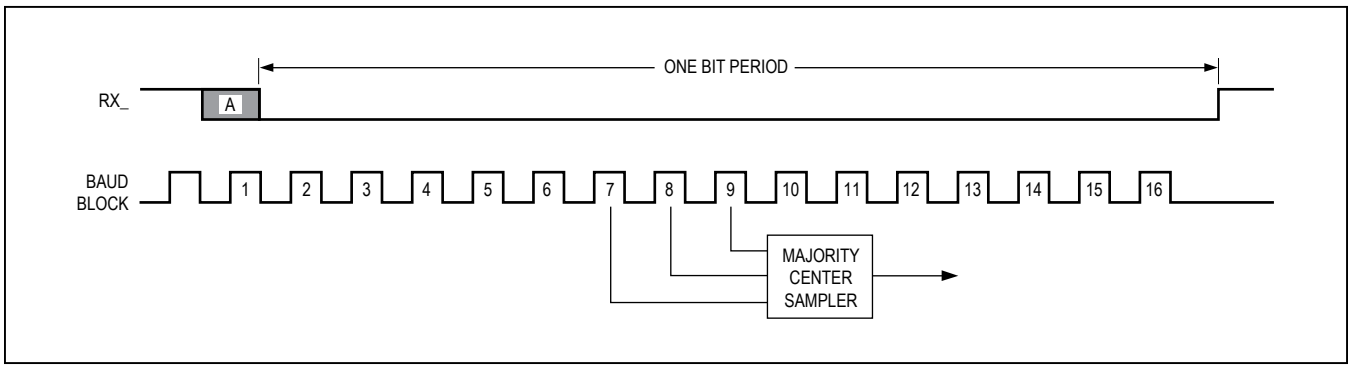


Figure 6. Midbit Sampling

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 or 2x (i.e., not 4x) rate mode, the MAX14830 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 MAX14830 can be clocked by an external crystal, or an external clock source. Figure 7 shows a simplified diagram of the clocking circuitry. When the MAX14830 is clocked by a crystal, the STSInt[5]: ClockReady indicates when the clocks have settled and the baud-rate generator is ready for stable operation.

Each UART baud rate can be individually programmed. To achieve fast baud rate changes, first disable the UART's clock by setting CLKDisabl to 1. Then change the baud rate divisor and subsequently enable the clock via CLKDisabl.

To check that the UART's clocking is programmed as expected, route the baud rate clock to $\overline{RTS}$ using the CLKtoRTS bit. The clock rate of this 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.

Crystal Oscillator

Set BRGConfig[6]: CLKDisabl to 0 and CLKSource[1]: CrystalEn to 1 to enable and select the crystal oscillator. The on-chip crystal oscillator circuit has load capacitances of 16pF (typ) integrated in both XIN and XOUT. Connect an external crystal or ceramic oscillator between XIN and XOUT.

External Clock Source

Connect an external clock source to XIN when not using a crystal oscillator. Leave XOUT unconnected. Set 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 the PLLConfig register. 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.

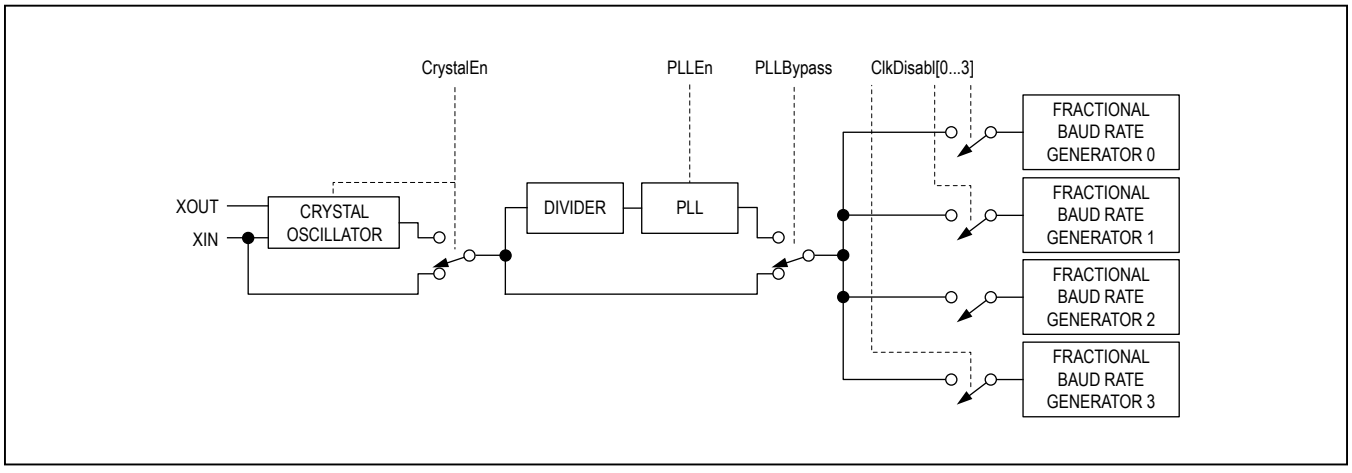


Figure 7. Clock Selection Diagram

Fractional Baud-Rate Generators

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 with the external crystal or clock source.

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. 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 value 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 default rate mode.

The ideal divisor is calculated as:

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

hence DIV = 9.

FRACT = ROUND(4.579) = 0x05

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 = 9.313,

where D_{ACTUAL} = DIV + (FRACT/16) and

BR_{ACTUAL}= 28,230,000/(16 x 9.3125) = 189,463.087 baud

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

2x and 4x Rate Modes

To support higher baud rates than possible with standard (16x sampling) operation, the MAX14830 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. In 4x mode only, 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 the programmed baud rate (Figure 8).

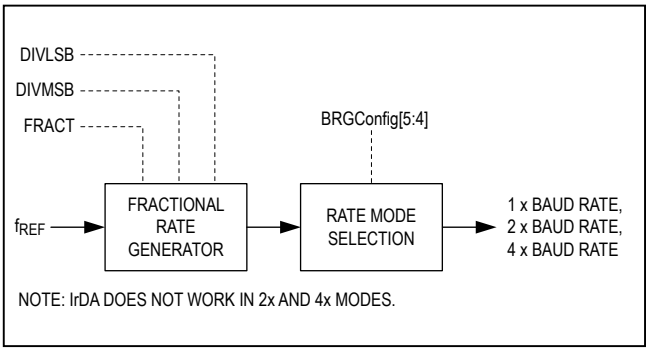


Figure 8. 2x and 4x Baud Rates

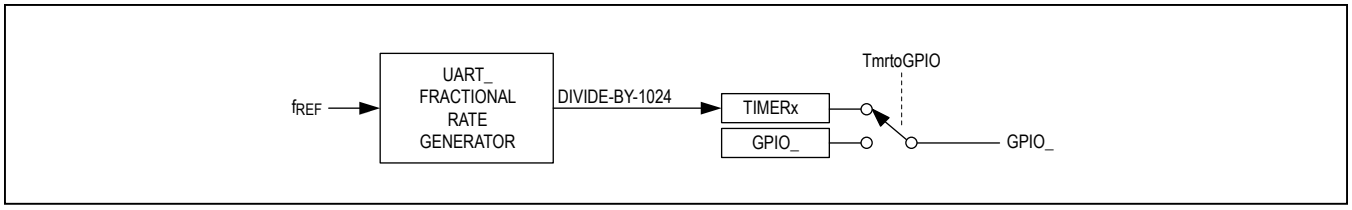


Figure 9. GPIO_ Clock Pulse Generator

Low-Frequency Timer

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

- UART0: GPIO1
- UART1: GPIO5
- UART2: GPIO9
- UART3: GPIO13

The clock pulses at the GPIOs are generated at a rate defined by the baud-rate generator and the timer divider (Figure 9). The baud-rate generator clock is divided by (1024 x TIMERx), where TIMERx is a 15-bit integer programmed into the TIMER1 and TIMER2 registers. The timer output is a 50% duty cycle clock.

UART Clock to GPIO

The MAX14830 reference clock can be routed to the GPIO0, GPIO4, GPIO8, and/or GPIO12 outputs in case a synchronous high-frequency clock is needed by another device. Enable routing a UART clock to GPIO0, GPIO4, GPIO8, and/or GPIO12 in the TxSynch register. This output clock could, for example, be used to clock another UART device (Figure 29).

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 MAX14830 can be configured to automatically filter out data that is not meant for its address. The address is user-definable either by programming a register value or a combination of a register value and GPIO hardware inputs. Use either 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_ are 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 MAX14830 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 MAX14830 can automatically enable/disable a transceiver's transmitter and/or receiver. This relieves the host processor of this time-critical task.

The $\overline{\text{RTS}}$_ output is used to control the transceivers' transmit enable input and is automatically set high when the MAX14830'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 10 shows a typical MAX14830 connection in a RS-485 application.

The $\overline{RTS_}$ output can be set high in advance of TX_ transmission by a programmable time period called the setup time (Figure 11). The setup time is programmed through HDplxDelay[7:4]. Similarly, the $\overline{RTS_}$ signal can be held high for a programmable period after the transmitter has completed transmission. The hold time is programmed through HDplxDelay[3:0].

Transmitter Triggering and Synchronization

The MAX14830 allows synchronization of transmitters so that selected UARTs start transmitting data when a trigger command is received. Optional delays can also be programmed, which delay the start of transmission after a trigger command is received. A UART's transmitter can be assigned one of 16 possible SPI/I2C trigger commands. A trigger command is defined as any of 16 special values written into the GloblComnd register (see the *GloblComnd* section for more information). When a byte is written into the GloblComnd register, UART select bits (U0 and U1) are ignored by the MAX14830, and the GloblComnd applies to all four UARTs. Transmission is initiated when the MAX14830 receives the assigned SPI/I2C trigger command if the selected transmitter is initially disabled and data has been loaded into its TxFIFO.

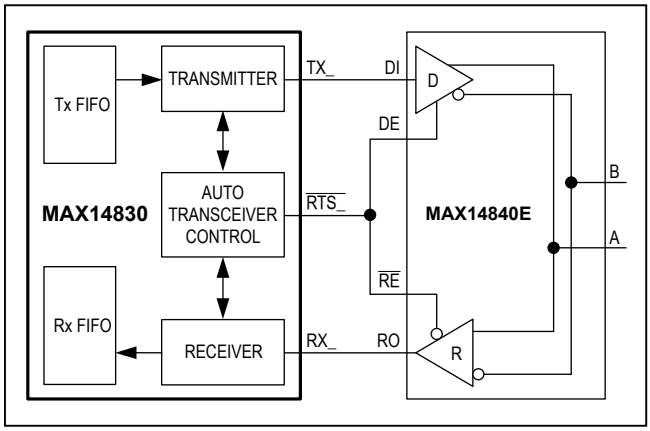


Figure 10. Auto Transceiver Direction Control

Enable and configure transmitter synchronization in the TxSynch register. Triggering and synchronization requires that the TxFIFOs are disabled before the trigger is received. This can be done by setting the MODE1[1] bit to 1 or by utilizing the auto transmitter disable function (TxSynch[4] is 1).

Transmitter Synchronization

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

Intrachip and Interchip Synchronization

Intrachip transmitter triggering occurs when any of the four UARTs in a MAX14830 are triggered by one trigger command. This type of synchronization is supported in both SPI and I2C modes, as the trigger commands are global commands that are received by all four UARTs simultaneously.

Interchip transmitter triggering occurs when the UARTs in different MAX14830 devices are synchronized. This type of synchronization is achievable in SPI mode only. Pull the $\overline{CS}$ of all the MAX14830 devices on the bus low during the SPI master's write trigger command so that the commands are received by all UARTs on the shared SPI bus.

I2C protocol does not allow simultaneous addressing of multiple devices.

Delayed Triggering

A delay can be programmed for delaying the start of transmission after the reception of an assigned trigger command. Set the delay by programming the SynchDelay1 and SynchDelay2 registers.

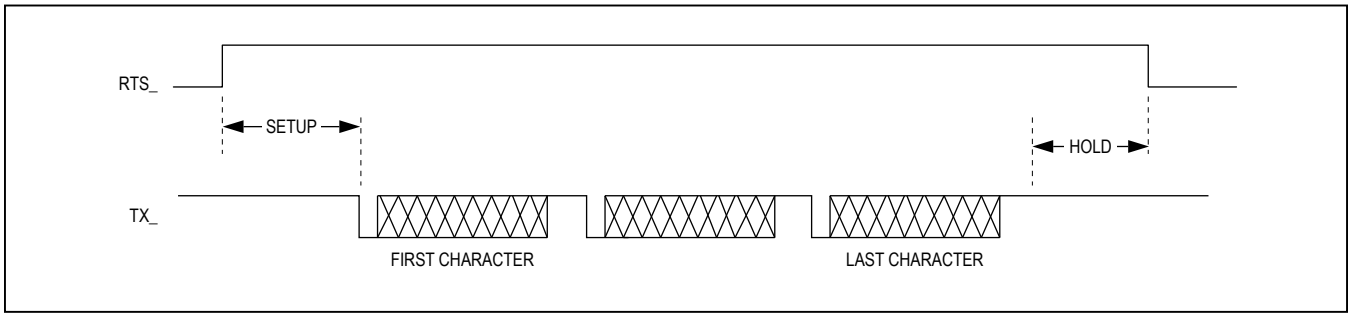


Figure 11. Setup and Hold times in Auto Transceiver Direction Control

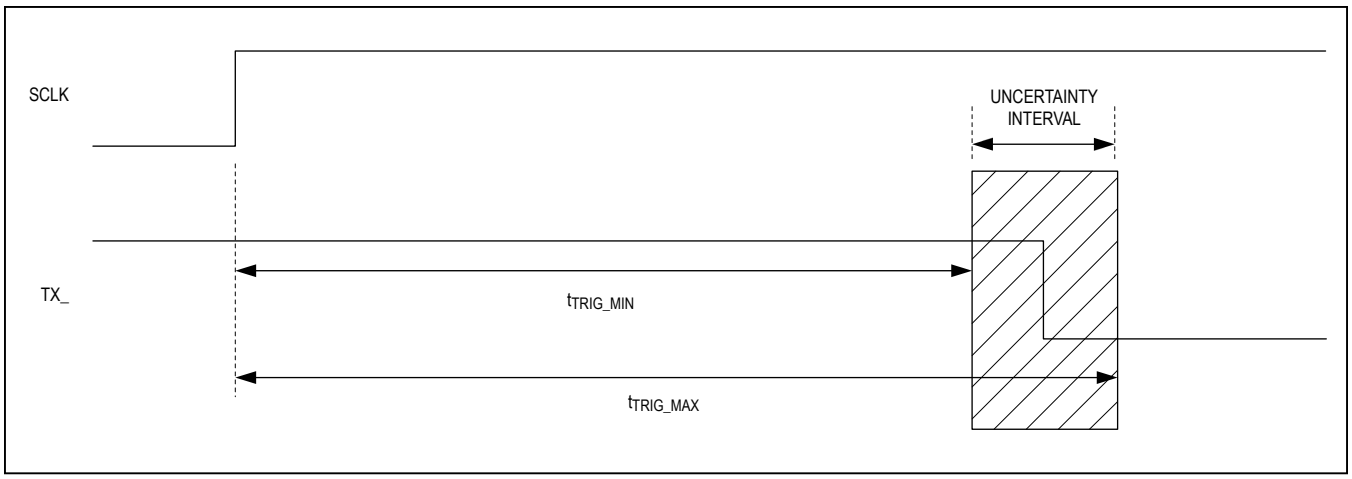


Figure 12. Single Transmitter Trigger Accuracy

Trigger Accuracy

The delay between the time when the MAX14830 receives a trigger command and the time when the associated transmitter starts transmission is made up of a fixed, deterministic portion and a variable, random component. Both portions of the delay are dependent on the UART's clock and baud rates. When the fractional divider is not used, the intrinsic trigger delay, t_{TRIG}, is bounded by the following limits:

$$\frac{5 \times BR}{16} \le t_{TRIG} \le \frac{6 \times BR}{16}$$

where BR is the fractional divider output clock period. This equation is independent on the rate mode. The reference point is the time when the trigger command is received by the MAX14830. This occurs on the final (i.e., the 16th) SPI clock's low-to-high transition (Figure 12).

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

Synchronization Accuracy

When synchronizing multiple UART transmitters, the accuracy of the TX_ transmitter outputs is based on the triggering delays of each UART (Figure 13). This skew has a baud-rate dependent component, similar to the trigger accuracy equation for a single transmitter output. Calculate the TX_ transmitter output skew using the following equation:

$$t_{TRIGSKEW}(max) \le \frac{6 \times BR_S - 5 \times BR_F}{16}$$

where BR_S is the fractional divider output clock of the lower/slower baud-rate UART and BR_F is the fractional divider output clock of the higher/faster baud-rate UART.

Auto Transmitter Disable

The MAX14830 allows automatic disabling of the transmitter. Enable auto transmitter disabling functionality by setting TxSynch[4] to 1. When auto transmitter disabling is activated, the MAX14830 disables the specified transmitter after it completes sending all the data in its TxFIFO. New data can then be loaded into the TxFIFO. A disabled transmitter does not send out data on the TX_output when data is present in its TxFIFO.

To enable transmission, either clear the TxAutoDis bit in the TxSynch register or toggle the TxDisabl bit in the MODE1 register.

Echo Suppression

The MAX14830 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 MAX14830'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 MAX14830 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 (Figure 14). The hold time delay is set by the HDplxDelay[3:0] register. See the *HDplxDelay Register* section for more information.

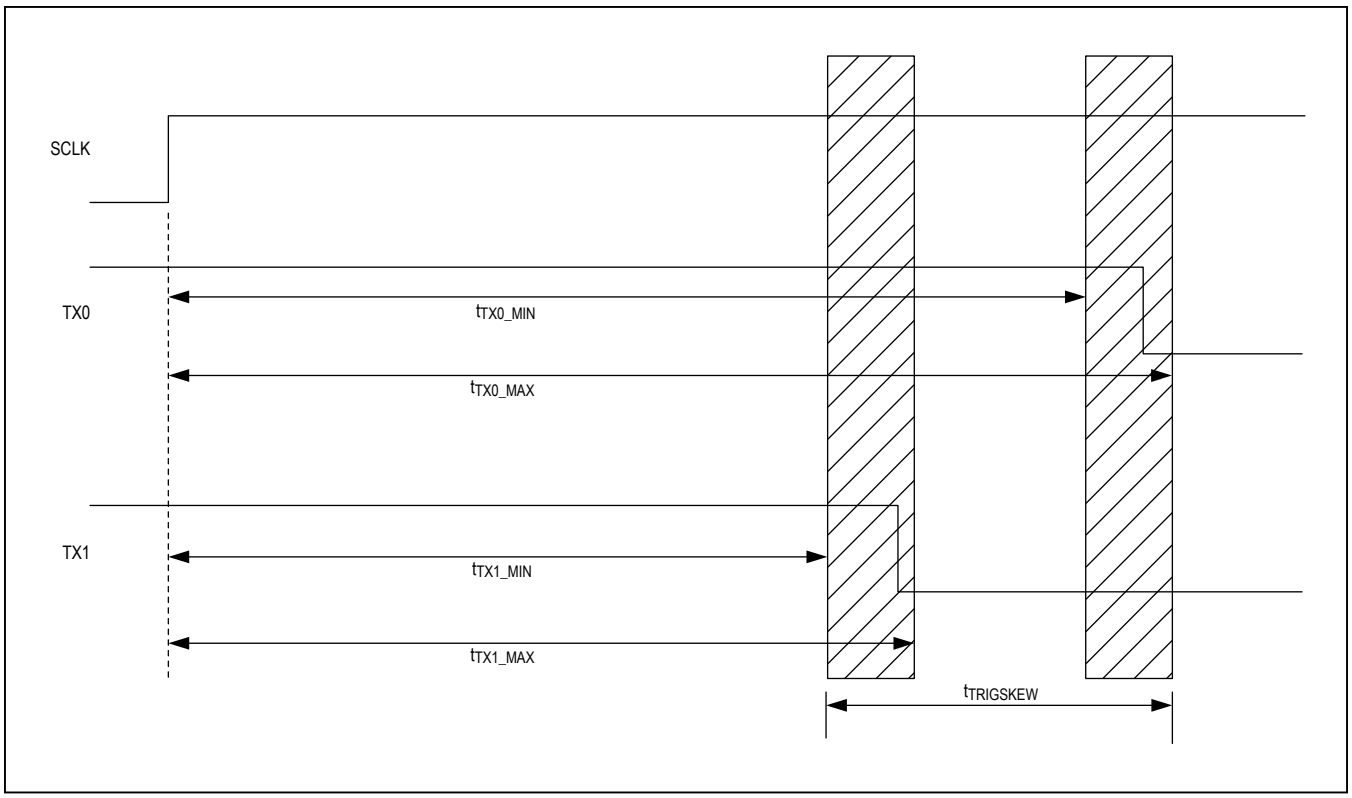


Figure 13. Multiple Transmitter Synchronization Accuracy

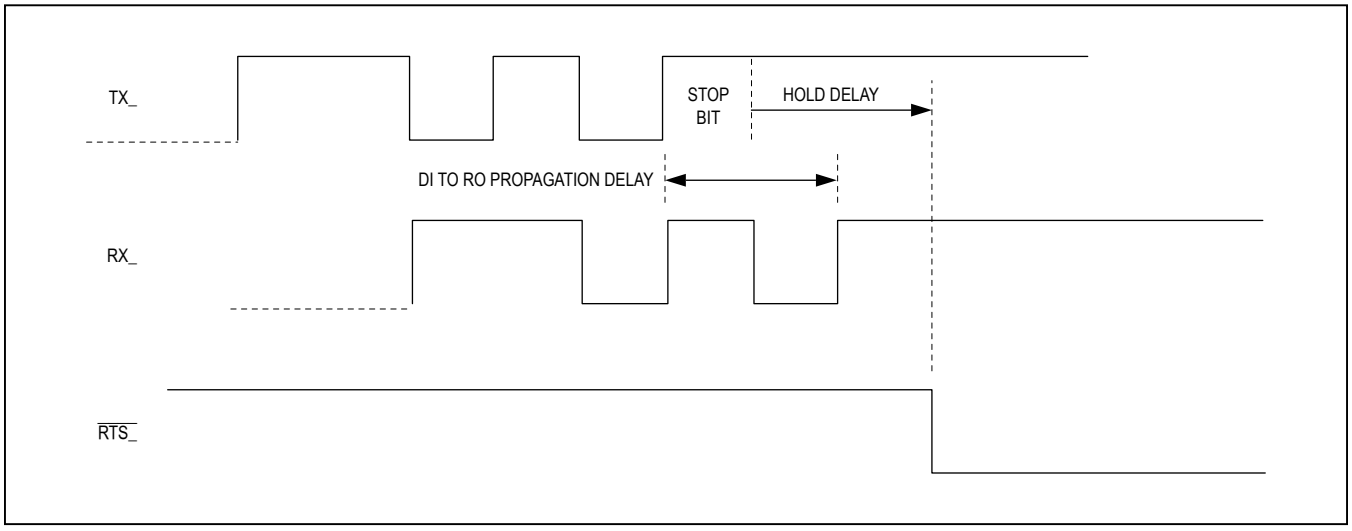


Figure 14. Echo Suppression Timing

Echo suppression can operate simultaneously with auto transceiver direction control (Figure 15).

Auto Hardware Flow Control

The MAX14830 is capable of automatic hardware (RTS and CTS) flow control without the need for host processor intervention. When AutoRTS control is enabled, the MAX14830 automatically controls the RTS handshake without the need for host processor intervention. AutoCTS flow control separately turns the MAX14830'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 MAX14830 does this automatically by controlling $\overline{RTS_}$. AutoRTS flow control is enabled through FlowCtrl[0]: AutoRTS. The HALT and RESUME levels determine the threshold levels at which $\overline{RTS_}$ is asserted and deasserted. HALT and RESUME are programmed in FlowLvl. With differing HALT and RESUME levels, hysteresis can be defined for the $\overline{RTS_}$ transitions.

When the RxFIFO fill level reaches the HALT level (FlowLvl[3:0]), the MAX14830 deasserts $\overline{RTS_}$. $\overline{RTS_}$ 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.

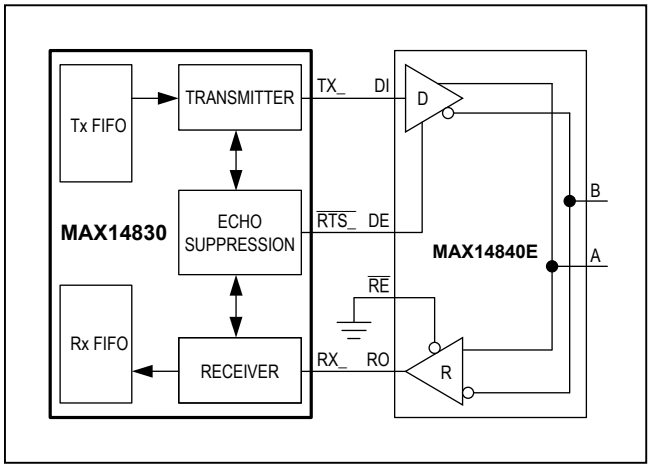


Figure 15. Half-Duplex with Echo Suppression

AutoCTS Control

When AutoCTS flow control is enabled, the UART automatically starts transmitting data when the $\overline{CTS_}$ input is logic-level low and stops transmitting when $\overline{CTS_}$ 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 $\overline{CTS_}$. If $\overline{CTS_}$ is set high during transmission the MAX14830 completes transmission of the current word and halts transmission afterwards.

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

FIFO Interrupt Triggering

Receive and Transmit FIFO fill-dependent interrupts are generated if FIFO trigger levels are defined. When the number of words in the FIFOs reach or exceed a trigger level, 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 reading in the Receive FIFO or available for transfer to the Transmit FIFO.

Auto Software (XON/XOFF) Flow Control

When auto software flow control is enabled, the MAX14830 recognizes and/or sends predefined XON/XOFF characters to control the flow of data across the asynchronous serial link. Automatic 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 MAX14830 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 $\overline{IRQ}$ can be masked by setting IRQEn[1]: SpclChrlEn to 0.

Software flow control consists of transmitter control and receiver overflow control, which can operate independently of one another.

Transmitter Flow Control

When auto transmitter control (FlowCtrl[5:4]) is enabled, the receiver compares all received words with the XOFF and XON characters. If an XOFF character is received, the MAX14830 halts its transmitter from sending further data. The receiver is not affected and continues reception. Upon receiving XON, the transmitter then 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. An interrupt is not generated.

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

Receiver Overflow Control

When auto receiver overflow control (FlowCtrl[7:6]) is enabled, the MAX14830 automatically sends XOFF and XON control characters to the far end UART to avoid receiver overflow. XOFF1/XOFF2 is/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 is/are automatically sent to the far end station to signal it to resume data transmission.

XON1/XOFF1 is transmitted before XON2/XOFF2 when dual character (XON1 and XON2/XOFF1 and XOFF2) flow control is enabled.

Power-Up and $\overline{IRQ}$

$\overline{IRQ}$ has two functions. During normal operation (MODE1[7] = 1), $\overline{IRQ}$ operates as a hardware interrupt output, whereby the $\overline{IRQ}$ is active when an interrupt is pending. An $\overline{IRQ}$ interrupt can only be 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 MAX14830 is ready for programming following an initialization delay. Once $\overline{IRQ}$ goes high, the MAX14830 is ready to be programmed. The MODE1[7]: IRQSel bit should then be set to enable normal $\overline{IRQ}$ interrupt operation.

In polled mode, the DIVLSB register can be polled to check whether the MAX14830 is ready for operation. If the controller gets a valid response from DIVLSB, then the MAX14830 is ready for operation.

Shutdown Mode

Pull $\overline{RST}$ to DGND to enter shutdown mode. Shutdown mode is the lowest power consumption mode. In shutdown mode, all of the MAX14830 circuitry is off. This includes the SPI/I²C interface, the registers, the FIFOs, and clocking circuitry. The LDO is on in shutdown mode.

When the $\overline{RST}$ input is high, the MAX14830 exits shutdown mode. The chip initialization is completed when the MAX14830 sets $\overline{IRQ}$ to logic-high.

The MAX14830 needs to be reprogrammed following a shutdown.

Interrupt Structure

The structure of the interrupt is shown in Figure 16. There are four interrupt source registers for each UART: ISR, LSR, STSInt, and SpclCharInt. Read the GlobalIRQ

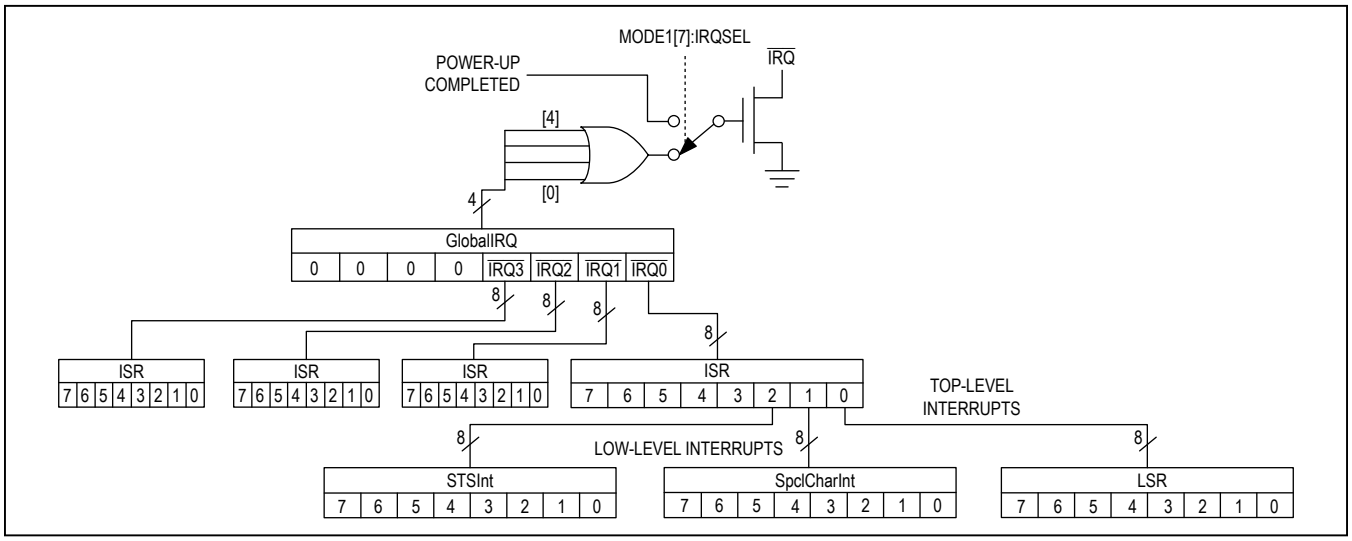


Figure 16. Simplified Interrupt Structure

register to determine which UART is the source of the interrupt. 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 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 are also clear on read (COR). The LSR bits are only cleared when the source of the interrupt is removed, not when LSR is read.

Reading the GlobalIRQ register does not clear the $\overline{IRQ}$ interrupt.

Register Map

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

REGISTER	ADDR	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
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	RFifoEmtyIEn	TFifoEmtyIEn	TFifoTrgIEn	RFifoTrgIEn	STSIEn	SpclChrIEn	LSRErrIEn
ISR*†	0x02	CTSInt	RFifoEmptyInt	TFifoEmptyInt	TFifoTrigInt	RFifoTrigInt	STSInt	SpCharInt	LSRErrInt
LSRIntEn	0x03	—	—	RxNoiseIntEn	RBreakIEn	FrameErrIEn	ParityIEn	ROverrIEn	RTimoutIEn
LSR*†	0x04	CTSbit	—	RxNoise	RxBreak	FrameErr	RxParityErr	RxOverrun	RTimeout
SpclChrIntEn	0x05	—	—	MltDrpIntEn	BREAKIntEn	XOFF2IntEn	XOFF1IntEn	XON2IntEn	XON1IntEn
SpclCharInt†	0x06	—	—	MultiDropInt	BREAKInt	XOFF2Int	XOFF1Int	XON2Int	XON1Int
STSIntEn¥	0x07	—	—	ClockRdyIntEn	—	GPI3IntEn	GPI2IntEn	GPI1IntEn	GPI0IntEn
STSInt†¥	0x08	—	—	ClockReady	—	GPI3Int	GPI2Int	GPI1Int	GPI0Int
UART MODES									
MODE1	0x09	IRQSel	—	—	TrnscvCtrl	RTSHiZ	TXHiZ	TxDisabl	RxDisabl
MODE2	0x0A	EchoSuprs	MultiDrop	LoopBack	SpecialChr	RxEmtyInv	RxTrgInv	FIFORst	RST
LCR*	0x0B	$\overline{RTS}$bit	TxBreak	ForceParity	EvenParity	ParityEn	StopBits	Length1	Length0
RxTimeOut	0x0C	TimOut7	TimOut6	TimOut5	TimOut4	TimOut3	TimOut2	TimOut1	TimOut0
HDplxDelay	0x0D	Setup3	Setup2	Setup1	Setup0	Hold3	Hold2	Hold1	Hold0
IrDA	0x0E	—	—	TxInv	RxInv	MIR	RTSInvert	SIR	IrDAEn
FIFOs 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	Bi6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
XOFF1	0x16	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
XOFF2	0x17	Bit7	Bi6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Register Map (continued)

REGISTER	ADDR	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	—	CLKDisabl	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	—	—	—	PLLBypass	PLLEn	CystalEn	—
GLOBAL REGISTERS									
GlobalRQ	0x1F	0	0	0	0	$\overline{IRQ3}$	$\overline{IRQ2}$	$\overline{IRQ1}$	$\overline{IRQ0}$
GloblComnd	0x1F	GlbCom7	GlbCom6	GlbCom5	GlbCom4	GlbCom3	GlbCom2	GlbCom1	GlbCom0
SYNCHRONIZATION REGISTERS									
TxSynch#	0x20	CLKtoGPIO	TxAutoDis	TrigDelay	SynchEn	TrigSel3	TrigSel2	TrigSel1	TrigSel0
SynchDelay1#	0x21	SDelay7	SDelay6	SDelay5	SDelay4	SDelay3	SDelay2	SDelay1	SDelay0
SynchDelay2#	0x22	SDelay15	SDelay14	SDelay13	SDelay12	SDelay11	SDelay10	SDelay9	SDelay8
TIMER REGISTERS									
TIMER1#	0x23	Timer7	Timer6	Timer5	Timer4	Timer3	Timer2	Timer1	Timer0
TIMER2#	0x24	TmrToGPIO	Timer14	Timer13	Timer12	Timer11	Timer10	Timer9	Timer8
REVISION									
REVID*†#	0x25	1	0	1	1	0	1	0	0

**Denotes nonzero default reset value: ISR = 0x60, LCR = 0x05, FIFOTrgLvl = 0xFF, PLLConfig = 0x01, DIVLSB = 0x01, CLKSource = 0x08, REVID = 0xB4.*

†Denotes nonread/write value: RHR = R, THR = W, ISR = COR, SpclCharInt = COR, STSInt = R/COR,

LSR = R, TxFIFOLvl = R, RxFIFOLvl = R, REVID = R.

¥Each UART has four individually assigned GPIO outputs as follows: UART0: GPIO0–GPIO3, UART1: GPIO4–GPIO7, UART2: GPIO8–GPIO11, UART3: GPIO12–GPIO15.

‡This register can only be programmed by accessing UART0.

#This register can only be directly addressed in I^2C mode. Use extended addressing when operating in SPI mode.

Detailed Register Descriptions

The MAX14830 has registers that are 8 bits wide.

RHR—Receive 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	X	X	X	X	X	X	X	X

Bits 7–0: RData[n]

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 first data bit of the serial-data word received by the receiver at the RX pin.

THR—Transmit Hold Register

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

Bits 7–0: TData[n]

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	RFifoEmtyIEn	TFifoEmtyIEn	TFifoTrgIEn	RFifoTrgIEn	STSIEn	SpclChrIEn	LSRErrIEn
RESET	0	0	0	0	0	0	0	0

The IRQEn register 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.

Note that an error can occur in the TxFIFO when a character is written into THR at the same time as the transmitter is transmitting out data via TX. In the event of this error condition, the result is that a character will not be transmitted.

In order to avoid this, stop the transmitter when writing data to the THR. This can be done via the TxDisable bit in the MODE1 register.

Bit 7: CTSIEn

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

Bit 6: RFifoEmtyIEn

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

Bit 5: TFifoEmtyIEn

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

Bit 4: TFifoTrgIEn

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

Bit 3: RFifoTrgIEn

The RFifoTrgIEn bit enables $\overline{IRQ}$ interrupt generation when the RFifoTrigInt interrupt bit is set in the ISR. Set the RFifoTrgIEn 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 the 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 the 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 the 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	RFifoEmptyInt	TFifoEmptyInt	TFifoTrigInt	RFifoTrigInt	STSInt	SpCharInt	LSRErrInt
RESET	0	1	1	0	0	0	0	0

The Interrupt Status Register provides an overview of all interrupts generated in the MAX14830. These interrupts are cleared upon reading the ISR. When the MAX14830 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 LSR[7]: CTS bit.

Bit 6: RFifoEmptyInt

The RFifoEmptyInt 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]: RxEmtyInt bit.

Bit 5: TFifoEmptyInt

The TFifoEmptyInt 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 FIFOTrigLvl[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 FIFOTrigLvl[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 that 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 upon 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.

LSRIntEn—Line Status Interrupt Enable Register

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 LSR Interrupt Enable register allows routing of LSR interrupt bits to the ISR[0].

Bits 7, 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 enable 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 Line Status Register shows all errors related to the word in the RxFIFO most recently read out of the RHR. The LSR bits are not cleared upon a read; these bits stay set until the next character without 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 $\overline{\text{CTS}}$bit depends on the input of 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 most recently read character residing in the RHR was being received. 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 character most recently read out of the RHR. RxBreak is cleared after the character following 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. LSR[3] corresponds to the frame error of the character most recently read out of the RHR. A frame error is related to errors in expected STOP bits. 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. LSR[2] indicates a parity error for the character most recently read out of the RHR. In 9-bit multidrop mode (MODE2[6] = 1) the receiver does not check parity and the LSR[2] 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 LSR[1] bit is set. The Receive FIFO retains the data in it and discards all new data that does not fit into it. The RxOverrun flag can generate an ISR[0] interrupt, if enabled through LSRIntEn[1].

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 a period longer than the time 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, LSR[0]: RTimeout is disabled. 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, 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: XOFF2IntEn

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, 6: No Function

Bit 5: MultiDropInt

The MultiDropInt interrupt is set when the MAX14830 receives an address character in 9-bit multidrop mode (MODE2[6] = 1). This bit is cleared when SpclCharInt is read. The MultiDropInt 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].

STSIntEn—STS Interrupt Enable Register

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

Bits 7, 6: No Function

Bit 5: ClkRdyIntEn

Set the ClkRdyIntEn bit high to route the ClockReady status bit to the ISR[2]: STSInt bit. If set low, the STSIntEn[5] masks the ISR[2] bit from the ClockReady status.

Bit 4: No Function

Bits 3–0: GPI[n]IntEn

Each UART has four individually assigned GPIO outputs as follows: UART0: GPIO0–GPIO3, UART1: GPIO4–GPIO7, UART2: GPIO8–GPIO11, UART3: GPIO12–GPIO15. For example, for UART0: Bit 0 is GPI0IntEn, Bit 1 is GPI1IntEn, Bit 2 is GPI2IntEn, and Bit 3 is GPI3IntEn. See Table 1.

The GPI[n]IntEn bits that are set high route the associated STSInt[3:0]: GPI[n]Int bits to the ISR[2] interrupt. Set the GPI[n]IntEn bits to 0 to disable the associated GPI[n]Int bits.

STSInt—Status Interrupt Register

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

Bits 7, 6: No Function

Bit 5: ClockReady

The ClockReady bit is set high when the clock, the divider, and PLL have settled and the MAX14830 is ready for data communication. The ClockReady bit only works with 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].

Bit 4: No Function

Bits 3–0: GPI[n]Int

Each UART has four individually assigned GPIO outputs as follows: UART0: GPIO0–GPIO3, UART1: GPIO4–GPIO7, UART2: GPIO8–GPIO11, UART3: GPIO12–GPIO15. For example, for UART0: Bit 0 is GPI0Int, Bit 1 is GPI1Int, Bit 2 is GPI2Int, and Bit 3 is GPI3Int. See Table 1.

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

Table 1. UART GPIO Assignments for GPIO Interrupts

UART	GPI3Int/GPI3IntEn	GPI2Int/GPI2IntEn	GPI1Int/GPI1IntEn	GPI0Int/GPI0IntEn
UART0	GPIO3	GPIO2	GPIO1	GPIO0
UART1	GPIO7	GPIO6	GPIO5	GPIO4
UART2	GPIO11	GPIO10	GPIO9	GPIO8
UART3	GPIO15	GPIO14	GPIO13	GPIO12

MODE1 Register

ADDRESS:	0x09							
MODE:	R/W							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	IRQSel	—	—	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 power-up sequence. The $\overline{IRQ}$ is low during power-up and transitions to high when the MAX14830 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 IRQ* section.

Bits 6, 5: No Function

Bit 4: TrnscvCtrl

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

Setup and hold times of $\overline{RTS_}$ 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 $\overline{RTS_}$.

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.

In auto transmitter disable mode, TxDisabl is high when the transmitter is completely empty.

Bit 0: RxDisabl

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

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 receiver (RX_) 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 $\overline{RTS_}$ to $\overline{CTS_}$. In local loopback mode, the TX_ output and the RX_ input 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/XOFF 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]: RFifoEmptyInt. If RxEmtyInv is set low (default state), the ISR[6] interrupt is generated when the last character residing in the Receive FIFO is read out of the RHR, and the Receive FIFO becomes empty. If the RxEmtyInv is set high, the ISR[6] interrupt is generated when the Receive FIFO is empty, and the UART receives at least one character.

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, which 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 selected UART in the MAX14830. The SPI/I2C bus stays active during this reset and communication with the MAX14830 is possible. All register bits in the selected UART are reset to their reset state and the FIFOs are cleared during a reset.

The global registers are not reset when the RST bit for a given UART is set. 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	$\overline{\text{RTS}}$bit	TxBreak	ForceParity	EvenParity	ParityEn	StopBits	Length1	Length0
RESET	0	0	0	0	0	1	0	1

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

The $\overline{\text{RTS}}$bit provides direct control of the $\overline{\text{RTS_}}$ output logic. If $\overline{\text{RTS}}$bit is set to 1, then $\overline{\text{RTS_}}$ is set to logic-high. The $\overline{\text{RTS}}$bit only works when CLKSource[7]: CLKtoRTS is set to 0.

Bit 6: TxBreak

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

Bit 5: ForceParity

The ForceParity bit enables forced parity, as used in 9-bit multidrop communication. Set both LCR[3]: ParityEn and ForceParity to 1 to use forced parity. The parity bit is forced high by the transmitter if LCR[4]: EvenParity is low. The parity bit is forced low if the EvenParity bit is high.

Bit 4: EvenParity

Set the EvenParity bit to 1 to generate even parity by the transmitter and parity is checked by the receiver. Odd parity generation and checking are used if EvenParity is set low.

Bit 3: ParityEn

The ParityEn bit enables the use of a parity bit on the TX_ and RX_ interfaces. Set the ParityEn bit to 0 to disable parity usage.

When the ParityEn bit is 1, the transmitter generates the parity bit as defined in LCR[4], and the receiver checks the 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 2). When LCR[2] 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, 0: Length[n]

The Length[n] bits configure the length of the words that the transmitter generates and the receiver checks for at the asynchronous TX_ and RX_ interfaces (Table 3).

Table 2. StopBits Truth Table

StopBits BIT	WORD LENGTH	STOP BIT LENGTH
0	5, 6, 7, 8	1
1	5	1–1.5
1	6, 7, 8	2

Table 3. Length_ Truth Table

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

RxTimeOut—Receiver Timeout Register

ADDRESS:	0x0C							
MODE:	R/W							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	TimOut7	TimOut6	TimOut5	TimOut4	TimOut3	TimOut2	TimOut1	TimOut0
RESET	0	0	0	0	0	0	0	0

Bits 7–0: TimOut[n]

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 $\overline{RTS_}$ and the TX_ output in automatic transceiver direction control mode (MODE1[4] = 1). The Hold[3:0] time can also be used for echo suppression in halfduplex communication. HDplxDelay also functions in the 2x and 4x rate modes.

Bits 7–4: Setup[n]

The Setup[n] bits define a setup time for $\overline{RTS_}$ to transition high before the transmitter starts transmission of its first character in auto transceiver direction control mode: MODE1[4]. This allows the MAX14830 to account for skew differences of the external transmitter's enable delay and propagation delays. Setup[n] bits 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 one bit interval, making this delay baud-rate dependent. The maximum delay is 15-bit intervals.

Bits 3–0: Hold[n]

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

The second factor that the Hold[n] 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 one 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	RTSInvert	SIR	IrDAEn
RESET	0	0	0	0	0	0	0	0

The IrDA register 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, 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: RTSInvert

Set the RTSInvert bit high to invert the $\overline{RTS}$ output.

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 MAX14830 receiver expects such pulses at its Rx input. If IrDA[0] is set to low (default), normal (non-IrDA) 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 (RTS/CTS) flow control.

Bits 7–4: Resume[n]

Resume[n] bits set the Transmit FIFO threshold at which an XON is automatically sent or $\overline{RTS_}$ is automatically set low. This signals the far end station to start transmission. The actual threshold level is calculated as 8 x Resume[n]. The resulting level is in the range of 0 to 120.

Bits 3–0: Halt[n]

Halt[n] bits set a Receive FIFO threshold level at which an XOFF is automatically sent or $\overline{RTS_}$ is automatically set high, depending on whether automatic software or hardware flow control is enabled. This signals the far end station to halt transmission. The actual threshold level is calculated as 8 x Halt[n]. Hence the selectable threshold granularity is eight. The resulting level is in the range of 0 to 120.

FIFOTrigLvl—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[n]

The RxTrig[n] 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 x RxTrig[n], hence the selectable threshold granularity is eight.

Bits 3–0: TxTrig[n]

The TxTrig[n] bits allow definition of the Transmit FIFO threshold level at which the MAX14830 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 to be 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 x TxTrig[n], 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[n]

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[n]

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[n]

The SwFlow[n] 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 4.

FlowCtrl[n] selects 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[n], the XON and XOFF characters that the MAX14830 receives are filtered out and are not put into the RxFIFO. Set the SwFlowEn bit to 0 and set MODE2[4] to 1 to enable special character detection. Under these conditions, auto flow transmit flow control is not used.

If both special character detection (MODE2[4]) and automatic 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.

Table 4. SwFlow_ 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, XOFF2 special character detection.

X = Don't care.

Bit 3: SwFlowEn

The SwFlowEn bit enables automatic software flow control. The characters used for automatic software flow control are selected in FlowCtrl[n]. If special character detection (MODE2[4] = 1) is used in addition to automatic 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_ input logic levels define the four LSBs of the special character, while the four 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 automatic 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, (MODE1[1] = 1), before AutoCTS is enabled.

Bit 0: AutoRTS

The AutoRTS bit enables automatic RTS flow control by which the MAX14830 sets its $\overline{RTS_}$ output dependent on the Receive FIFO fill level. The FIFO thresholds at which $\overline{RTS_}$ changes state are set in FlowLvl. See the *Auto Hardware Flow Control* section for more information.

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

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

Bits 7–0: Bit[n]

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.

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

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

Bits 7–0: Bit[n]

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 flow control are enabled (MODE2[4] and FlowCntrl[3]), these bits define a special character.

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

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

Bits 7–0: Bit[n]

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 area both enabled, XOFF1 defines the XOFF flow control character.

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

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

Bits 7–0: Bit[n]

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.

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

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

Bits 7–4: GP[n]OD

Each UART has four individually assigned GPIO outputs as follows: UART0: GPIO0–GPIO3, UART1: GPIO4–GPIO7, UART2: GPIO8–GPIO11, UART3: GPIO12–GPIO15. For example, for UART0: Bit 4 is GP0OD, Bit 5 is GP1OD, Bit 6 is GP2OD, and Bit 7 is GP3OD (see Table 5).

Set GP[n]OD bits to 0 to configure the GPIO_s as push-pull outputs, if configured as outputs in GPIOConfg[3:0].

Set the GP[n]OD bits to 1 to configure to open-drain output operation.

When configured as inputs in GPIOConfg[3:0], the GPIO_s are high-impedance inputs with weak pulldowns.

Bits 3–0: GP[n]Out

Each UART has four individually assigned GPIO outputs as follows: UART0: GPIO0–GPIO3, UART1: GPIO4–GPIO7, UART2: GPIO8–GPIO11, UART3: GPIO12–GPIO15. For example, for UART0: Bit 0 is GP0Out, Bit 1 is GP1Out, Bit 2 is GP2Out, and Bit 3 is GP3Out (see Table 5).

The GP[n]Out bits configure the GPIO_ to be inputs or outputs. Set the GP[n]Out bits to 1 to configure the associated GPIO_s as outputs. Set the GP[n]Out bits to 0 to configure the associated GPIOs as inputs.

Bits 7–4: GPI[n]Dat

Table 5. UART GPIO Assignments for GPIO Configuration

UART	GP3OD/GP3Out	GP2OD/GP2Out	GP1OD/GP1Out	GP0OD/GP0Out
UART0	GPIO3	GPIO2	GPIO1	GPIO0
UART1	GPIO7	GPIO6	GPIO5	GPIO4
UART2	GPIO11	GPIO10	GPIO9	GPIO8
UART3	GPIO15	GPIO14	GPIO13	GPIO12

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

Each UART has four individually assigned GPIO outputs as follows: UART0: GPIO0–GPIO3, UART1: GPIO4–GPIO7, UART2: GPIO8–GPIO11, UART3: GPIO12–GPIO15. For example, for UART0: Bit 4 is GPI0Dat, Bit 5 is GPI1Dat, Bit 6 is GPI2Dat, and Bit 7 is GPI3Dat (see Table 6).

The GPI[n]Dat bits reflect the logic on the GPIO_s.

Bits 3–0: GPO[n]Dat

Each UART has four individually assigned GPIO outputs as follows: UART0: GPIO0–GPIO3, UART1: GPIO4–GPIO7, UART2: GPIO8–GPIO11, UART3: GPIO12–GPIO15. For example, for UART0: Bit 0 is GPO0Dat, Bit 1 is GPO1Dat, Bit 2 is GPO2Dat, and Bit 3 is GPO3Dat (see Table 6).

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

Bits 7, 6: PLLFactor[n]

Table 6. UART GPIO Assignments for GPIO Input/Output Data

UART	GPI3Dat/GPO3Dat	GPI2Dat/GPO2Dat	GPI1Dat/GPO1Dat	GPI0Dat/GPO0Dat
UART0	GPIO3	GPIO2	GPIO1	GPIO0
UART1	GPIO7	GPIO6	GPIO5	GPIO4
UART2	GPIO11	GPIO10	GPIO9	GPIO8
UART3	GPIO15	GPIO14	GPIO13	GPIO12

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

The PLLFactor[n] bits allow programming the PLL multiplication factors. The input and output frequencies of the PLL have to be limited to the ranges shown in Table 7. Enable the PLL through CLKSource[2].

Bits 5–0: PreDiv[n]

The PreDiv[n] bits allow programming the divisor of the PLL's predivider. The divisor must be chosen so that the output frequency of the predivider, which equals the PLL's input frequency, is limited to the ranges shown in Table 4. The input frequency of XIN, is f_{CLK}:

$$f_{PLLIN} = f_{CLK}/PreDiv$$

See Figure 17. PreDiv is an integer that must be in the range of 1 to 63.

Bit 7: No Function

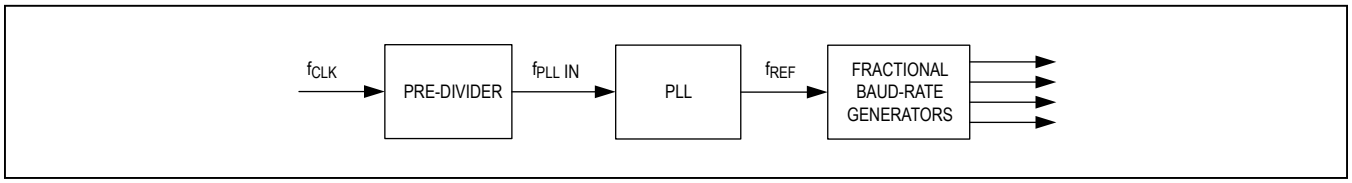


Figure 17. PLL Signal Path

Table 7. PLLFactor_ Selector 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

BRGConfig—Baud-Rate Generator Configuration Register

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

Bit 6: CLKDisabl

Set the CLKDisabl bit high to disable internal clocking of the UART. This is useful to achieve fast baud rate reprogramming or to reduce power dissipation when a specific UART channel is not used. Set CLKDisabl low for normal UART operation.

Bit 5: 4xMode

When the 4xMode bit is set high, the MAX14830 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 MAX14830 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[n]

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

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

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

Bits 7–0: Div[n]

The DIVLSB register is the LSBs of the integer divisor portion (DIV) of the baud-rate generator.

Bits 7–0: Div[n]

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

The DIVMSB register is the MSB portion of the integer divisor (DIV).

Bit 7: CLKtoRTS

CLKSource—Clock Source Register

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

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

Bits 6, 5: No Function

Bit 4:

Bi 4 can be programmed to logic 0 or logic 1.

Bit 3: PLLBypass

Set the PLLBypass bit to 1 to enable bypassing the internal PLL and predivider.

Bit 2: PLLEn

Set the PLLEn bit to 1 to enable the internal PLL. Set PLLEn to 0 to disable the internal PLL.

Bit 1: CrystalEn

Set the CrystalEn bit to 1 to enable the crystal oscillator. When using an external clock source at XIN, set CrystalEn to 0.

Bit 0:

Always keep Bit 0 at logic 0.

Bits 7–4: No Function

GlobalIRQ—Global IRQ Register

ADDRESS:	0x1F							
MODE:	R							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	—	—	—	—	$\overline{\text{IRQ3}}$	$\overline{\text{IRQ2}}$	$\overline{\text{IRQ1}}$	$\overline{\text{IRQ0}}$
RESET	0	0	0	0	1	1	1	1

Bits 3–0: $\overline{\text{IRQ}}$[n]

The MAX14830 has a single $\overline{\text{IRQ}}$ output. The GlobalIRQ register bits report which of the UARTs have an interrupt pending, as enabled in the ISRIntEn registers.

The GlobalIRQ register can be read in two ways: either by reading register 0x1F of any of the four UARTs or by sampling the 4 bits sent to the master on MISO during the command byte of a read cycle (full-duplex SPI) (see the *Fast Read Cycle* section for more information).

$\overline{\text{IRQ}}$[n] is set to 0 when the associated UART's internal IRQ is generated.

$\overline{\text{IRQ}}$_ bits are cleared when the associated UART interrupt is cleared. UART interrupts are cleared by reading the UART ISR register.

Bits 7–0: GlbCom[n]

GloblComnd—Global Command Register

ADDRESS:	0x1F							
MODE:	W							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	GlbCom7	GlbCom6	GlbCom5	GlbCom4	GlbCom3	GlbCom2	GlbCom1	GlbCom0

The GloblComnd register is the only global write register in the MAX14830. Every byte written to GloblComnd is sent simultaneously to all four UARTs. Every byte sent by the SPI/I2C master to location 0x1F is interpreted as a global command by all the four internal UARTs.

The MAX14830 logic supports the following commands (Table 8):

- Global Tx Synchronization
- Extended Addressing Space Enable (to get access to registers beyond address 0x1F)
- Extended Addressing Space Disable (to disable access to registers beyond address 0x1F)

The last two commands (0xCE/0xCD) enable/disable the access to registers in the extended space of the register map when MAX14830 operates in SPI mode. The SPI command byte has only 5 bits to address a given register so that the registers beyond 0x1F could not be addressed using the standard access method.

In I2C mode, there is no need to explicitly enable and disable the extended register map access as I2C allows up to 7 bits for register addressing.

To extend the addressing capability of the SPI command byte, send a 0xCE to location 0x1F. The internal SPI address is generated as 0010 A3A2A1A0, where A3A2A1A0 is the least significant nibble of the command byte. Bit A4 of the command byte is disregarded when the extended space of the register map is enabled and only the least significant nibble is used for addressing purposes (Table 9).

Bits U1 and U0 of the command byte maintain their meaning in the extended mode. See the *SPI Interface* section for more information.

To return to standard addressing mode, the SPI master has to send the 0xCD command. In this case, the internal SPI address is generated as follows (default): 000A4 A3A2A1A0

Table 8. GloblComnd Command Descriptions

GloblComnd[7:0]	COMMAND DESCRIPTION
0xE0	Tx Command 0
0xE1	Tx Command 1
0xE2	Tx Command 2
0xE3	Tx Command 3
0xE4	Tx Command 4
0xE5	Tx Command 5
0xE6	Tx Command 6
0xE7	Tx Command 7
0xE8	Tx Command 8
0xE9	Tx Command 9
0xEA	Tx Command 10
0xEB	Tx Command 11
0xEC	Tx Command 12
0xED	Tx Command 13
0xEE	Tx Command 14
0xEF	Tx Command 15
0xCE	Enable extended register map access
0xCD	Disable extended register map access

Table 9. Extended Mode Addressing (SPI only)

REGISTER	SPI MODE ADDRESS	I2C MODE ADDRESS
TxSynch	0x00	0x20
SynchDelay1	0x01	0x21
SynchDelay2	0x02	0x22
TIMER1	0x03	0x23
TIMER2	0x04	0x24
RevID	0x05	0x25

TxSynch—Transmitter Synchronization Register

ADDRESS:		0x20						
MODE:		R/W						
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	CLKtoGPIO	TxAutoDis	TrigDelay	SynchEn	TrigSel3	TrigSel2	TrigSel1	TrigSel0
RESET	0	0	0	0	0	0	0	0

The TxSynch register is used to configure transmitter synchronization with a global SPI or I2C command. One of 16 trigger commands (Table 5) can be selected to be the synchronization trigger source for every UART. This allows simultaneous start of transmission of multiple UARTs that are associated with the same global trigger command. The synchronized UARTs can be on a single MAX14830 or on multiple devices if they are controlled by a common SPI interface.

UARTs start transmission when a global trigger command is received. Start of transmission is considered to be the falling edge of the START bit at the TX_ output. A delay can optionally be programmed through the SynchDelay1 and SynchDelay2 registers.

Tx synchronization is managed through software by transmitting the broadcast trigger Tx command (Table 5) to the MAX14830 through the SPI or I2C interface. To selectively synchronize ports that are on the same MAX14830 (Intrachip Synchronization) or on different MAX14830 (Interchip Synchronization) devices, up to 16 trigger Tx commands have been defined (see the *GloblComnd* section for more information).

Bit 7: CLKtoGPIO

The CLKtoGPIO bit is used to provide a buffered replica of the UARTs system clock (i.e. the fractional divider input) to a GPIO. The assignment is as follows: UART0's clock is routed to GPIO0, UART1's clock is routed to GPIO4, UART2's clock is routed to GPIO8, and UART3's clock is routed to GPIO12.

Bit 6: TxAutoDis

Set the TxAutoDis bit to 1 to enable automatic transmitter disabling. When TxAutoDis is 1, the transmitter is automatically disabled when all data in the TxFIFO has been transmitted. After the transmitter is disabled, the TxFIFO can then be filled with data that is transmitted when its assigned trigger command, defined by the TrigSelx bits, is received.

Bit 5: TrigDelay

Set TrigDelay to 1 to enable delayed start of transmission. The UART starts transmitting data following a delay programmed in SynchDelay1 and SynchDelay2 after receiving the assigned trigger command.

Bit 4: SynchEn

Set SynchEn to 1 to enable the software Tx synchronization. When SynchEn is high, the UART starts transmitting data after receiving the expected trigger command, if the TxFIFO contains data. Setting SynchEn high forces the TxDisabl bit (MODE1[1]) high and thereby disables the UART's transmitter. This prevents the transmitter from sending data as soon as the TxFIFO contains some. Once the TxFIFO has been loaded, the UART starts transmitting data only upon receiving the assigned trigger command.

Set SynchEn to 0 to disable transmitter synchronization for that UART. When SynchEn is 0, that UART's transmitter does not start transmission through any trigger command.

Bits 3–0: TrigSel[n]

The TrigSel[n] bits select the trigger command for that UART's transmitter synchronization when SynchEn is 1. For example, set TxSynch[3:0] to 0x08 for the UART to be triggered by TX command 8 (0xE8, Table 5).

SynchDelay1—Synchronization Delay Register 1

ADDRESS:	0x21							
MODE:		R/W						
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	SDelay7	SDelay6	SDelay5	SDelay4	SDelay3	SDelay2	SDelay1	SDelay0
RESET	0	0	0	0	0	0	0	0

The SynchDelay1 and SynchDelay2 register contents define the time delay between when the UART receives an assigned transmitter trigger command and when the UART begins transmission.

Bits 7–0: SDelay[n]

SDelay[7:0] are the 8 LSBs of the delay between when the UART receives an assigned transmitter trigger command and when the UART begins transmission. The delay is expressed in number of UART bit intervals (1/BaudRate). The maximum delay is 65,535-bit intervals.

For example, given a baud rate of 230.4kbps and a bit time of 4.34µs, the maximum delay is 284ms.

SynchDelay2—Synchronization Delay Register 2

ADDRESS:	0x22							
MODE:		R/W						
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	SDelay15	SDelay14	SDelay13	SDelay12	SDelay11	SDelay10	SDelay9	SDelay8
RESET	0	0	0	0	0	0	0	0

The SynchDelay1 and SynchDelay2 register contents define the time delay between when the UART receives an assigned transmitter trigger command and when the UART begins transmission.

Bits 7–0: SDelay[n]

SDelay[15:8] are the 8 MSBs of the delay between when the UART receives an assigned transmitter trigger command and when the UART begins transmission. The delay is expressed in number of UART bit intervals (1/BaudRate). The maximum delay is 65,535-bit intervals.

For example, given a baud rate of 230.4kbps and a bit time of 4.34µs, the maximum delay is 284ms.

TIMER1—Timer Register 1

ADDRESS:	0x23							
MODE:		R/W						
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	Timer7	Timer6	Timer5	Timer4	Timer3	Timer2	Timer1	Timer0
RESET	0	0	0	0	0	0	0	0

The TIMER1 and TIMER2 register contents can be used to generate a low-frequency clock signal on a GPIO_ output. The low-frequency clock is a divided replica of the fractional divider output.

Bits 7–0: Timer[n]

Timer[7:0] are the 8 LSBs of the 15-bit timer divisor. See the TIMER2 register description.

If TIMER1 and TIMER2 are both 0x00, the low-frequency clock is off.

TIMER2—Timer Register 2

ADDRESS:	0x24							
MODE:	R/W							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	TmrToGPIO	Timer14	Timer13	Timer12	Timer11	Timer10	Timer9	Timer8
RESET	0	0	0	0	0	0	0	0

The TIMER1 and TIMER2 register contents can be used to generate a low-frequency clock signal on a GPIO_ output. The low-frequency clock is a divided replica of the fractional divider output.

Bit 7: TmrToGPIO

Set TmrToGPIO to 1 to enable clock generation at a GPIO output. The clock signal is routed to a GPIO output as follows: UART0 clock signal to GPIO1, UART1 clock signal to GPIO5, UART2 clock signal to GPIO9, UART3 clock signal to GPIO13. The output clock has a 50% duty cycle.

Bits 6–0: Timer[n]

Timer[14:8] are the 7 MSBs of the 15-bit timer divisor. The clock frequency is calculated using the following formula:

$$f_{TIMER_CLK} = UARTClk/(1024 \times Timerx)$$

where UARTClk is the fractional baud-rate generator output (i.e. 16 x BaudRate). When using 2x or 4x rate modes, UARTClk is 8 x BaudRate or 4 x BaudRate, respectively.

If TIMER1 and TIMER2 are both 0x00, the low-frequency clock is off.

RevID—Revision Identification Register

ADDRESS:	0x25							
MODE:	R							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAME	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RESET	1	0	1	1	0	1	0	0

Bits 7–0: Bit[n]

The RevID register indicates the revision number of the MAX14830 silicon—starting with 0xB1. This can be used during software development as a known reference.

Table 10. SPI Command Byte Configuration

SPI COMMAND BYTE							
BIT 7	**BIT 6**	**BIT 5**	**BIT 4**	**BIT 3**	**BIT 2**	**BIT 1**	**BIT 0**
W/$\overline{R}$	U1	U0	A4	A3	A2	A1	A0

A[4:0] = Register Address

Table 11. SPI U1, U0 UART Selection

U1	U0	UART SELECTED
0	0	UART0
0	1	UART1
1	0	UART2
1	1	UART3

Serial Controller Interface

The MAX14830 can be controlled through SPI or I²C as defined by the logic on SPI/$\overline{I2C}$. See the *Pin Configuration* section for further details.

SPI Interface

The SPI interface 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).

Each of the four UARTs is addressed using 2 bits (U1 and U0) in the command byte (see Table 10 and Table 11).

MISO Operation

Before a specific UART has been addressed, all four UARTs can attempt to drive MISO. To avoid this contention, the MISO line is held in high impedance during a write cycle (Figure 18).

During a read cycle, MISO is high impedance for the first 4 clock cycles of the command byte. Once the SPI address (U1 and U0) has been properly decoded, the addressed SPI drives the MISO line (Figure 19).

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 TxFIFO by using the THR (0x00) as the initial burst write address. Similarly, multiple characters can be read out of the RxFIFO by using the RHR (0x00) as the SPI's burst read address. If the SPI burst address is different to 0x00, the MAX14830 automatically increments the register address after each SPI data byte. Efficient programming of multiple consecutive registers is thus possible. Chip select, $\overline{CS}$/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 $\overline{CS}$/A0 high.

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

1) Pull $\overline{CS}$/A0 low.
2) Send SPI write command.
3) Send 128 bytes.
4) Release $\overline{CS}$/A0.

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

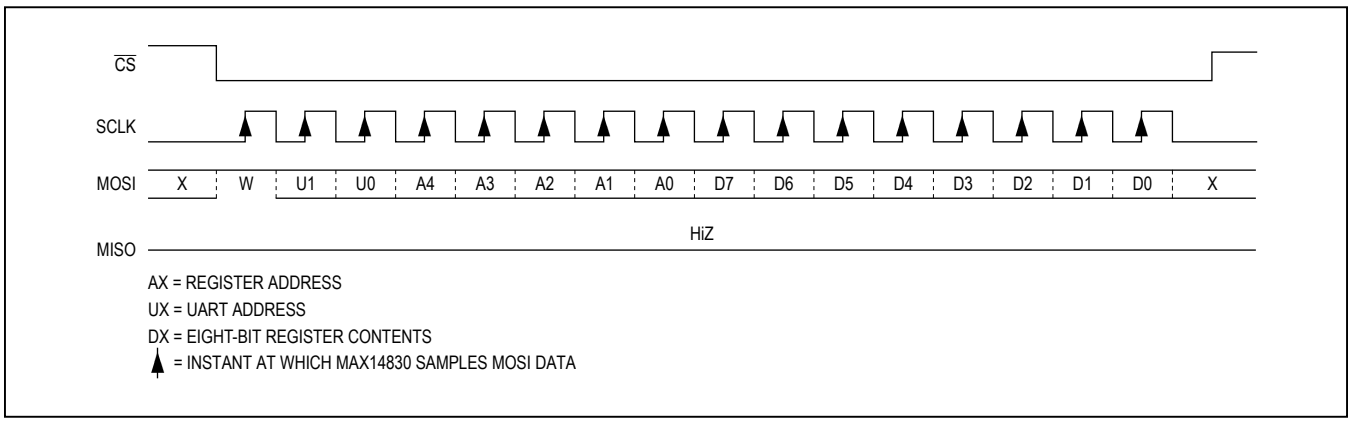


Figure 18. SPI Write Cycle

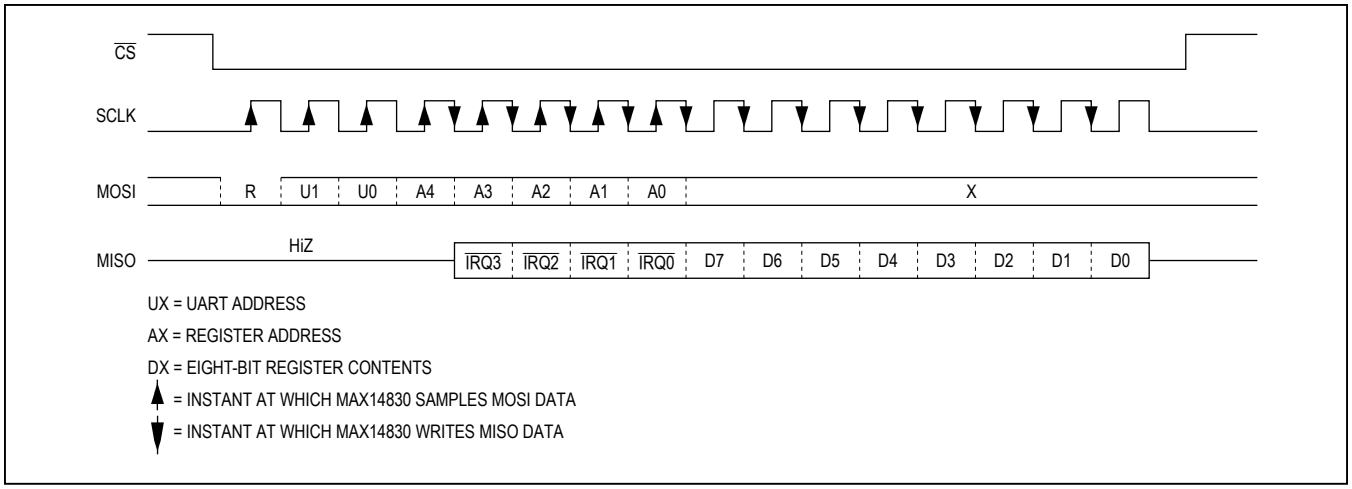


Figure 19. SPI Read Cycle

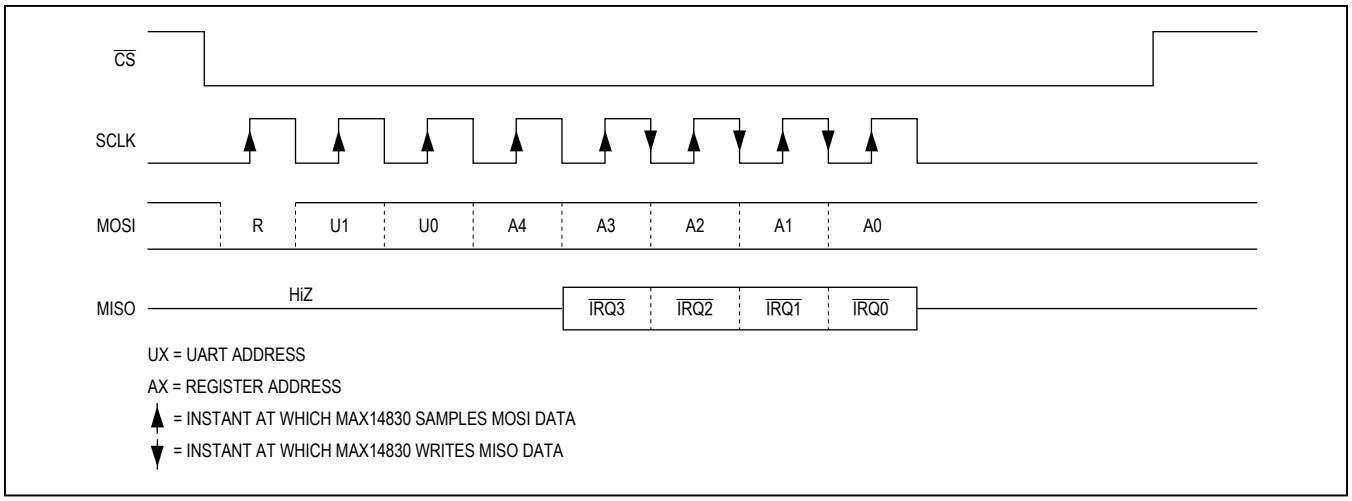


Figure 20. SPI Fast Read Cycle

Fast Read Cycle

On the MAX14830 the four UART interrupts share the single $\overline{IRQ}$ output. When operating in interrupt-based mode, the microcontroller needs to locate the source of the interrupt (i.e. which of the four UARTs generated the interrupt) and clear the interrupt.

To locate the source of an interrupt more quickly, the MAX14830 implements the SPI fast read cycle. This means that the microcontroller can determine which UART is the source of the interrupt (UART0, UART1, UART2, or UART3) using only 8 clock cycles (Figure 20). U1 and U0 bits are ignored during the fast read cycle.

I²C Interface

The MAX14830 contains an I²C-compatible interface for data communication with a host processor (SCL and SDA). The interface supports a clock frequency up to 1MHz. SCL and SDA require pullup resistors that are connected to a positive supply.

START, STOP, and Repeated START Conditions

When writing to the MAX14830 using I²C, the master sends a START condition (S) followed by the MAX14830 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

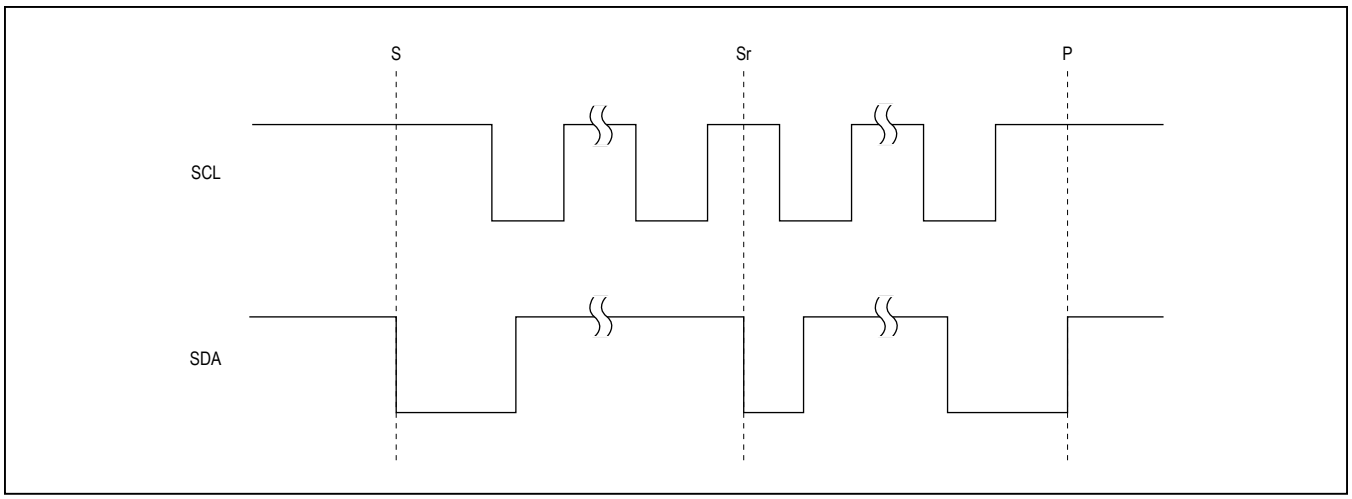


Figure 21. I²C START, STOP, and Repeated START Conditions

Table 12. I²C Address Map

MOSI/A1	$\overline{CS}$/A0	UART0		UART1		UART2		UART3	
		WRITE	READ	WRITE	READ	WRITE	READ	WRITE	READ
DGND	DGND	0xD8	0xD9	0xB8	0xB9	0x58	0x59	0x38	0x39
DGND	V_L	0xC2	0xC3	0xA2	0xA3	0x42	0x43	0x22	0x23
DGND	SCL	0xC4	0xC5	0xA4	0xA5	0x44	0x45	0x24	0x25
DGND	SDA	0xC6	0xC7	0xA6	0xA7	0x46	0x47	0x26	0x27
V_L	DGND	0xC8	0xC9	0xA8	0xA9	0x48	0x49	0x28	0x29
V_L	V_L	0xCA	0xCB	0xAA	0xAB	0x4A	0x4B	0x2A	0x2B
V_L	SCL	0xCC	0xCD	0xAC	0xAD	0x4C	0x4D	0x2C	0x2D
V_L	SDA	0xCE	0xCF	0xAE	0xAF	0x4E	0x4F	0x2E	0x2F
SCL	DGND	0xD0	0xD1	0xB0	0xB1	0x50	0x51	0x30	0x31
SCL	V_L	0xD2	0xD3	0xB2	0xB3	0x52	0x53	0x32	0x33
SCL	SCL	0xD4	0xD5	0xB4	0xB5	0x54	0x55	0x34	0x35
SCL	SDA	0xD6	0xD7	0xB6	0xB7	0x56	0x57	0x36	0x37
SDA	DGND	0xC0	0xC1	0xA0	0xA1	0x40	0x41	0x20	0x21
SDA	V_L	0xDA	0xDB	0xBA	0xBB	0x5A	0x5B	0x3A	0x3B
SDA	SCL	0xDC	0xDD	0xBC	0xBD	0x5C	0x5D	0x3C	0x3D
SDA	SDA	0xDE	0xDF	0xBE	0xBF	0x5E	0x5F	0x3E	0x3F

condition (P), to relinquish control of the bus, or a Repeated START condition (Sr) to communicate to another I²C slave. See Figure 21.

Slave Address

The MAX14830 includes a 7-bit I²C slave address, allowing up to 16 MAX14830 devices to share the same I²C bus. The address is defined by connecting the MOSI/A1 and $\overline{CS}$/A0 inputs to ground, V_L, SDA or to SCL (Table 12). Set the read/write bit to 1 to configure the MAX14830 to read mode. Set the read/write bit to 0 to configure the MAX14830 to write mode. The address is the first byte of information sent to the MAX14830 after the START condition.

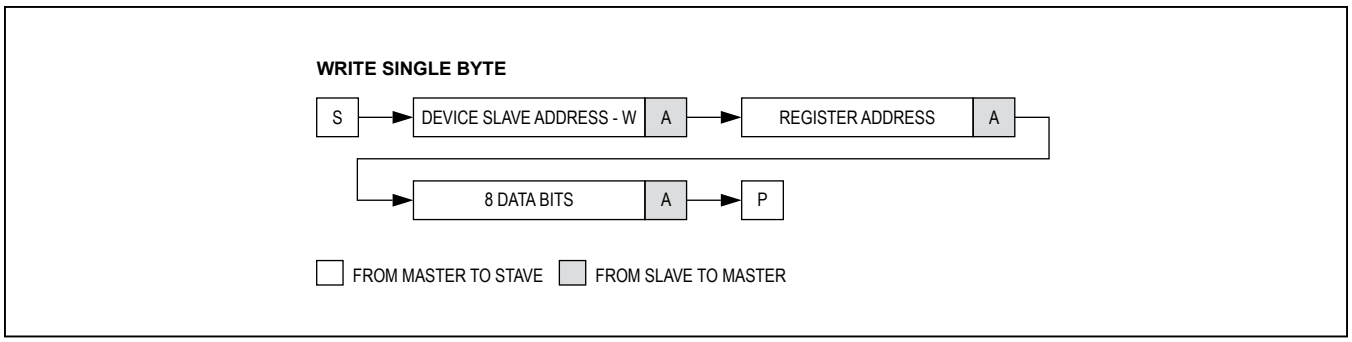


Figure 22. Write Byte Sequence

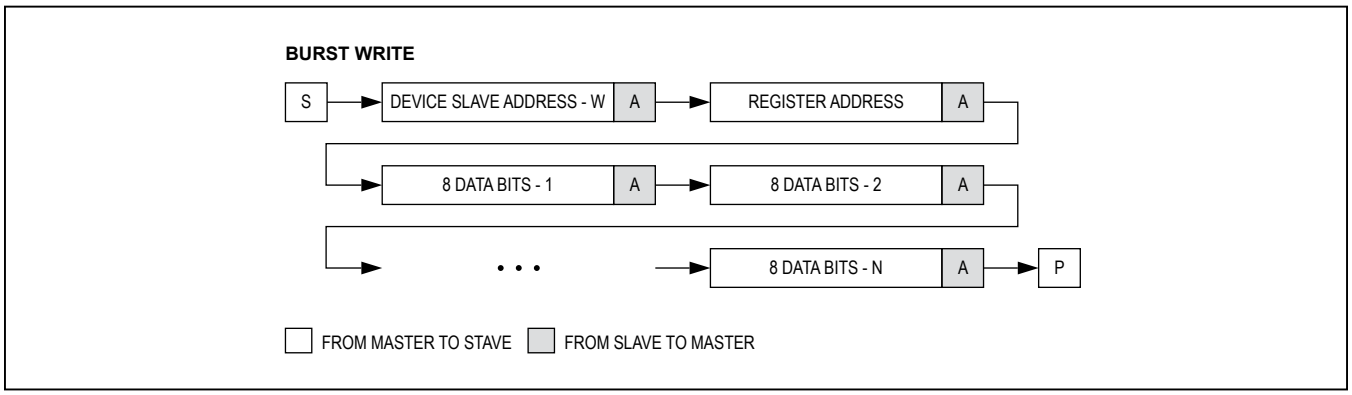


Figure 23. Burst Write Sequence

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 one or two data bytes to the slave device (Figure 22). The write byte procedure is the following:

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 (NAK if not).
6) The master sends an 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 23). The burst write 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 (NAK 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 as needed.
9) The master generates a STOP condition.

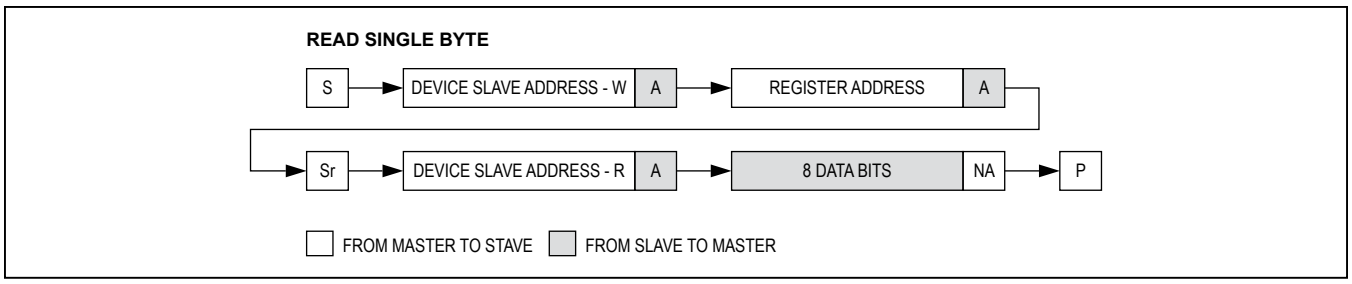


Figure 24. Read Byte Sequence

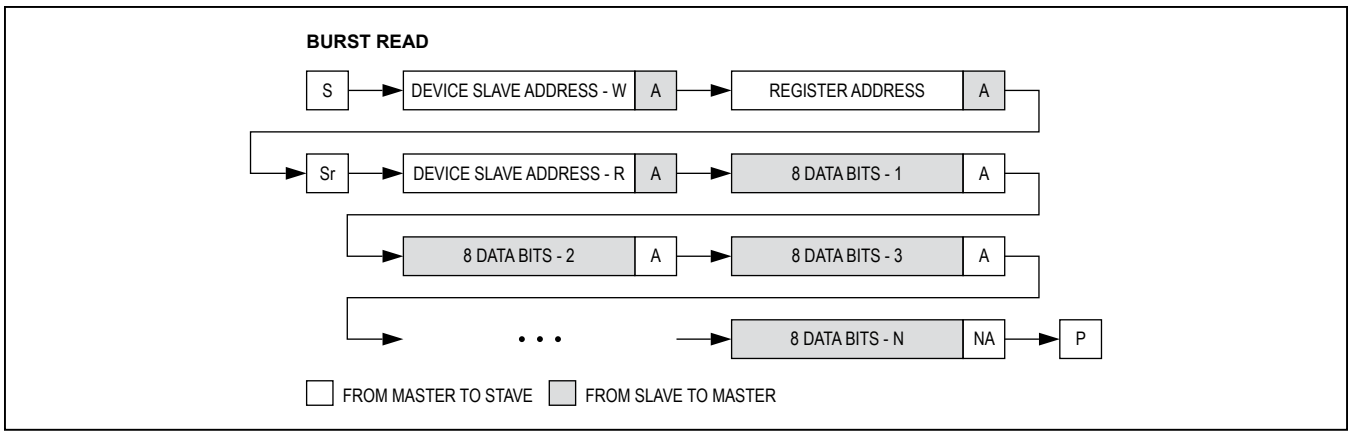


Figure 25. Burst Read Sequence

Single-Byte Read

With this operation the master sends an address and receives 1 or 2 data bytes from the slave device (Figure 24). 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 (NAK if not).
6) The master sends a repeated START (Sr).
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 25). 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 (NAK 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 9 and 10 (N-2) times.
12) The slave sends the last 8 data bits.
13) The master asserts a NACK on the data line.
14) The master generates a STOP condition.

Acknowledge Bits

Data transfers are acknowledged with an acknowledge bit (ACK) or a not-acknowledge bit (NACK). Both the master and the MAX14830 generate ACK bits. To generate an ACK, pull SDA low before the rising edge of the ninth clock pulse and keep it low during the high period of the ninth clock pulse (Figure 26). To generate a NACK, leave SDA high before the rising edge of the ninth clock pulse and keep it high for the duration of the ninth clock pulse. Monitoring for NACK bits allows for detection of unsuccessful data transfers.

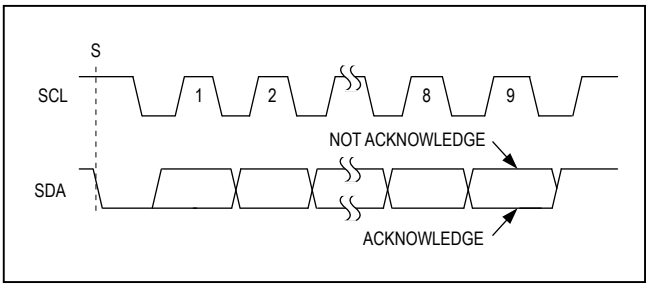


Figure 26. Acknowledge Bits

Applications Information

Startup and Initialization

The MAX14830 is initialized following power-up or a hardware or software reset (Figure 27). Check that the MAX14830 is ready for operation after a power-up or reset by monitoring the $\overline{IRQ}$ output, if interrupt driven operation is employed.

In polled mode, repeatedly read a known register until the expected contents are returned.

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.
- When any of the four UARTs are not being used, sop clicking via CLKDisabl.
- Use an external 1.8V supply at V_{18}. This saves the power dissipated in the internal 1.8V linear regulator for the 1.8V core supply. Disable the internal regulator by connecting LDOEN to DGND.
- Keep internal clock rates as low as possible.
- Use a low voltage on the V_A supply.

Interrupts and Polling

Monitor the MAX14830 by polling the ISR register or by monitoring the $\overline{IRQ}$ output. 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 MAX14830.

Alternatively, the physical interrupt, $\overline{IRQ}$, of the MAX14830 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.

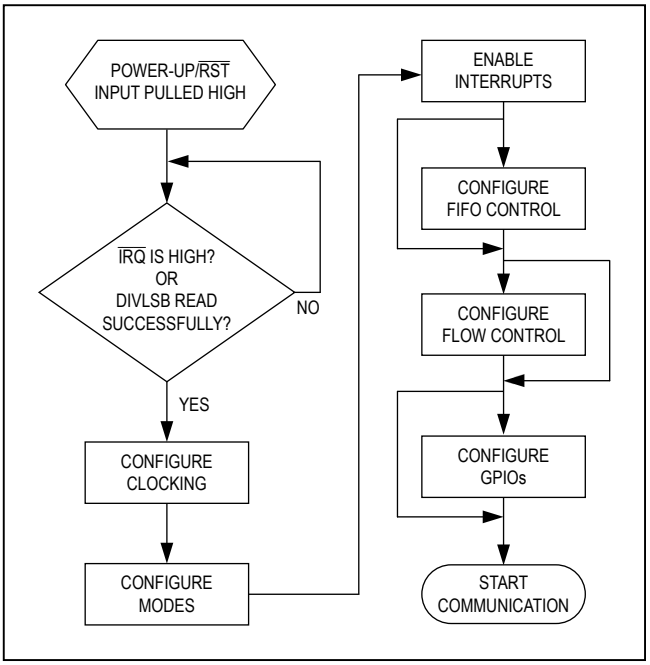


Figure 27. Startup and Initialization Flow Chart

Logic-Level Translation

The MAX14830 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 28 is an example of a setup when the controller, transceiver, and the MAX14830 are powered by three different supplies.

IO-Link Application

The *Typical Operating Circuit* shows a four-part IO-link master circuit with SPI control on the MAX14830 and the IO-link transceivers.

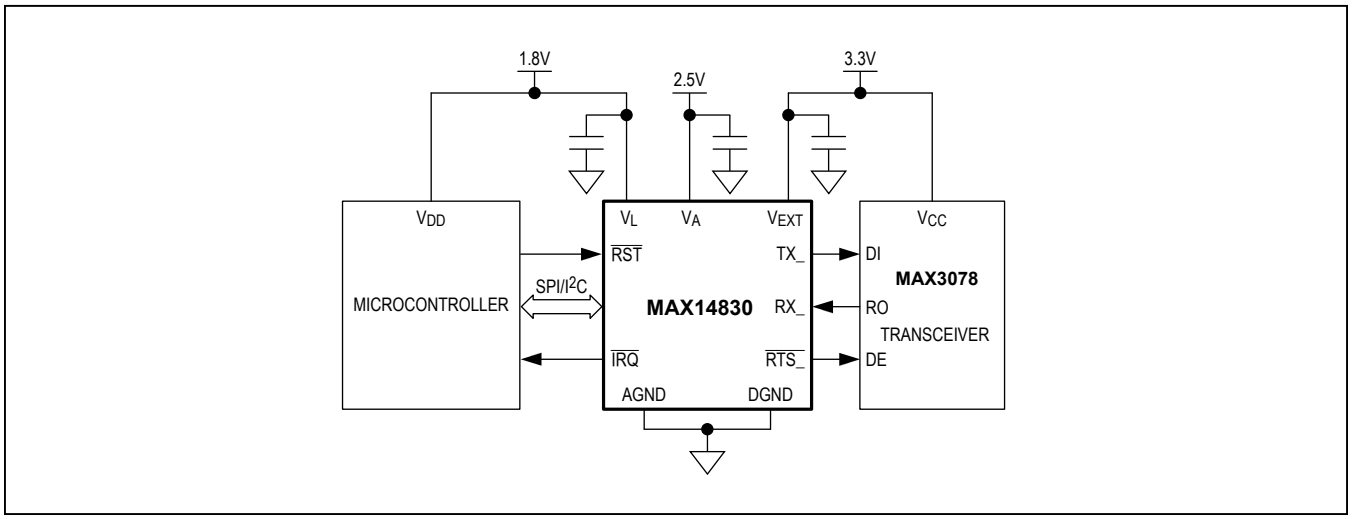


Figure 28. Logic-Level Translation

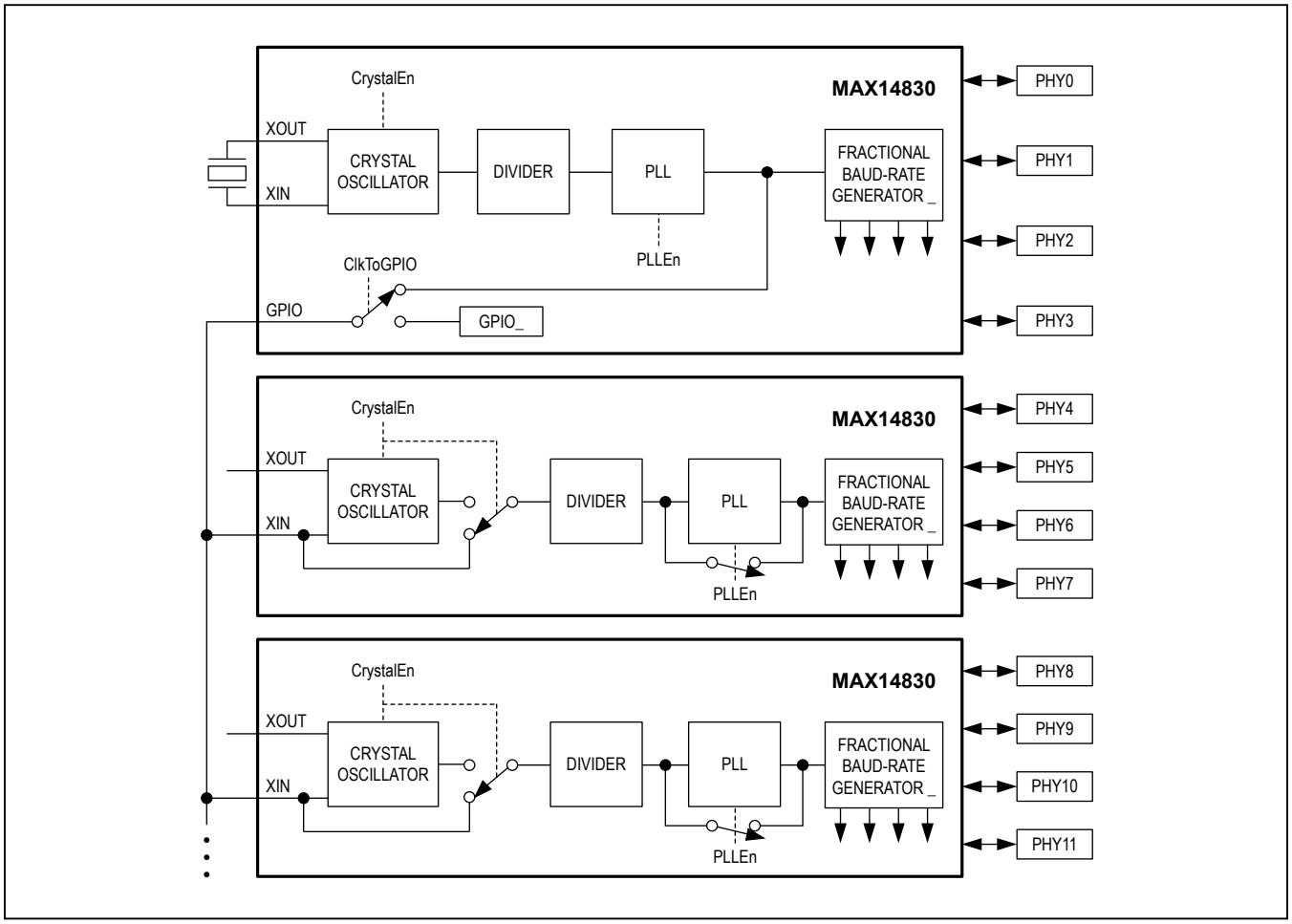
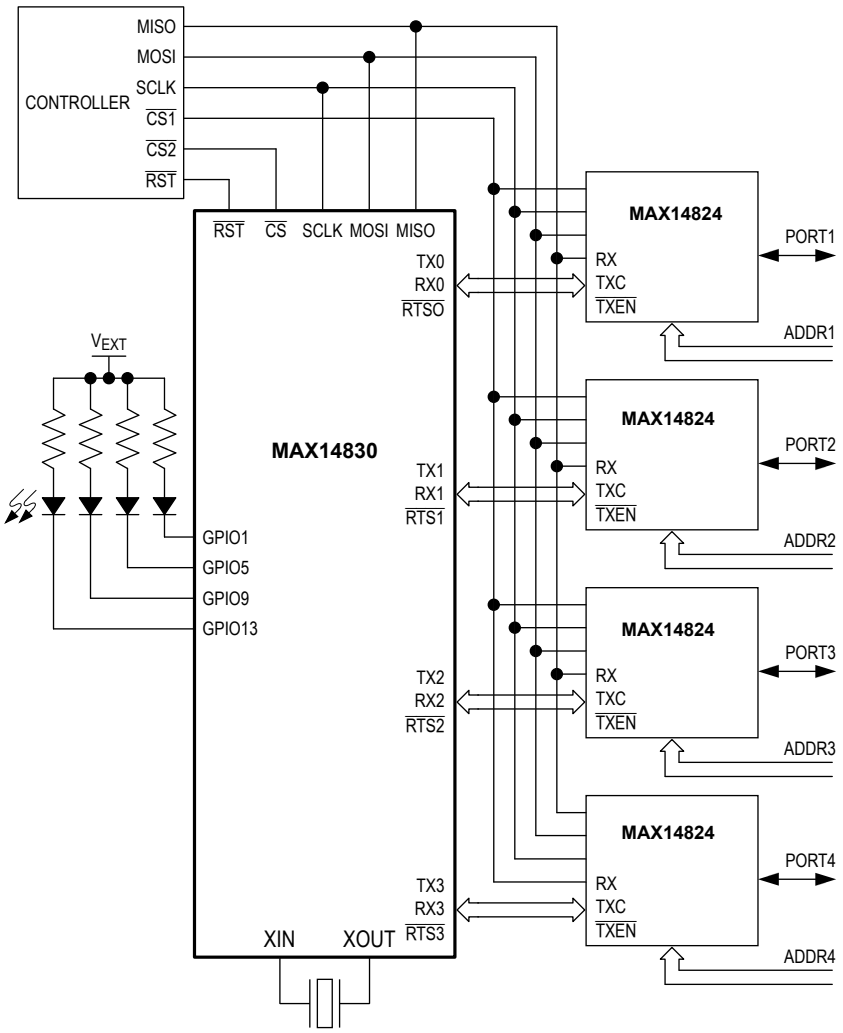


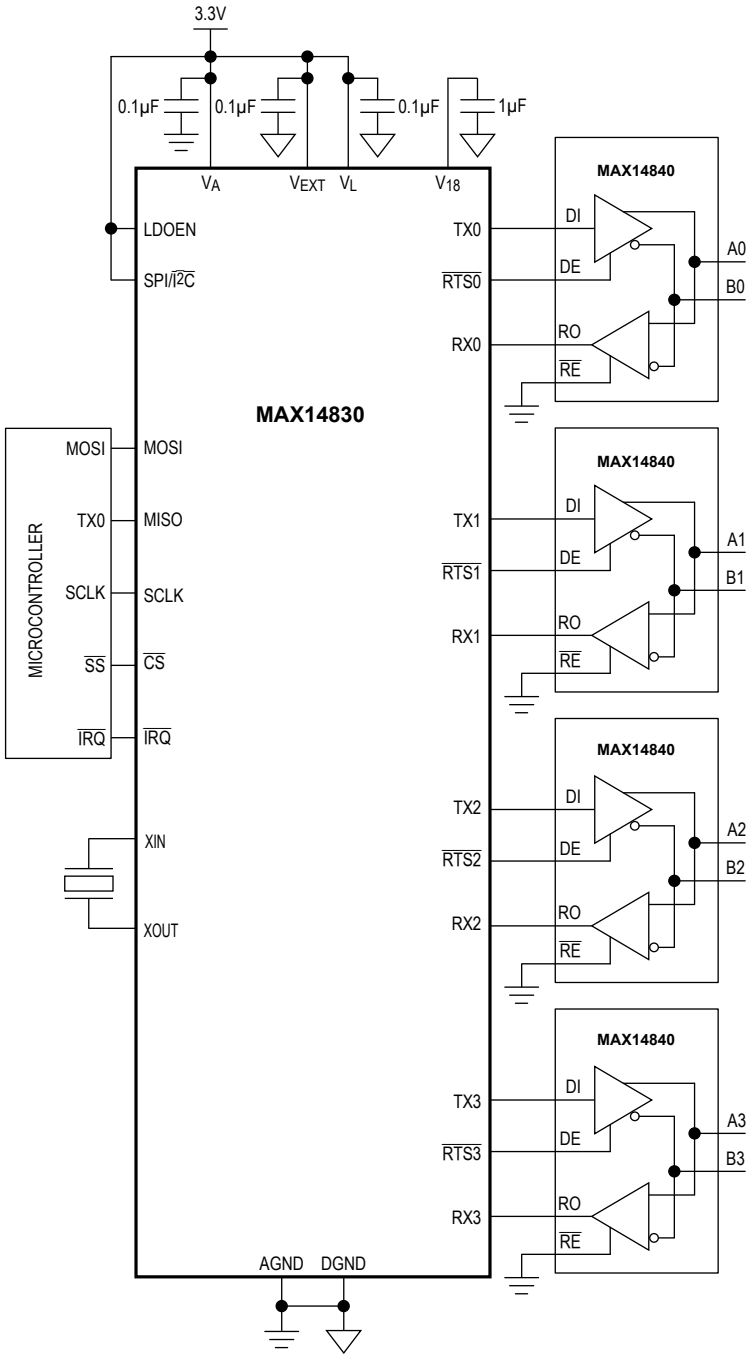
Figure 29. Interchip Synchronization

Typical Operating Circuit



IO-LINK QUAD MASTER APPLICATION

Typical Operating Circuit (continued)



QUAD RS-485 INTERFACE CONTROLLED THROUGH SPI

Chip Information

PROCESS: BiCMOS

Ordering Information

PART	TEMP RANGE	PIN-PACKAGE
MAX14830ETM+	-40°C to +85°C	48 TQFN-EP*

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

**EP = Exposed paddle.*

Package Information

For the latest package outline information and land patterns (footprints), go to **www.maximintegrated.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	OUTLINE NO.	LAND PATTERN NO.
48 TQFN	T4877+3	21-0144	90-0129

Revision History

REVISION NUMBER	REVISION DATE	DESCRIPTION	PAGES CHANGED
0	9/10	Initial release	—
1	12/10	Corrected specifications in the *Absolute Maximum Ratings* and *DC Electrical Characteristics*, updated the *Register Map*, corrected Table 12	8, 9, 29, 34, 37, 38, 40, 57, 60
2	9/11	Removed internal oscillator description throughout data sheet; deleted TOCs 1 and 2; corrected Figure 7; changed V_{18} capacitor to 1µF; corrected I^2C burst read sequence; corrected ISR description; added RTSInvert bit; added CLKDisabl bit	1, 2, 7, 8,10, 14,17,19, 20, 21, 27, 28, 29, 30, 34, 35, 40, 43, 52, 53, 57, 62, 63, 66
3	1/13	Updated *DC Electrical Characteristics* table; corrected *Typical Operating Circuit*	10, 53, 66
4	11/14	Removed automotive reference in *Applications* section	1
5	2/15	Added to the *Receive and Transmit FIFOs* section a note about how the TxFIFOLvl and RxFIFOLvl values can be in error; added a note to the *Transmitter Operation* and *Receiver Operation* sections about how errors can occur; updated the RHR, THR, TxFIFOLvl, and RxFIFOLvl register bit descriptions	18, 19, 30, 45
6	5/15	Revised *Benefits and Features* section	1
7	2/16	Updated TxFIFO and RxFIFO errata, REVID was changed from 0XB3 to 0XB4, and updated read/write while receiving/transmitting errata	19, 20, 29, 30, 31, 46, 58

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

Maxim Integrated cannot assume responsibility for use of any circuitry other than circuitry entirely embodied in a Maxim Integrated product. No circuit patent licenses are implied. Maxim Integrated reserves the right to change the circuitry and specifications without notice at any time. The parametric values (min and max limits) shown in the Electrical Characteristics table are guaranteed. Other parametric values quoted in this data sheet are provided for guidance.