MAX17851

SPI to UART Safety Monitoring Bridge

General Description

The MAX17851 SPI to UART safety monitoring bridge translates communication from SPI format to universal asynchronous receiver transmitter (UART) format specially designed to interface with Maxim's Battery Management Data Acquisition System. The safety monitoring bridge enables robust communication with baud rates extending up to 4Mbps in both single and dual daisy-chain system architectures. High throughput, dual UART systems can read 96 cells of battery voltages within 1173us.

The UART is ISO26262 compliant for automotive safety integrity level (ASIL) D systems providing detection of message corruption, delay, loss, and insertion through the embedded UART communication lock step safety measure state machine (LSSM). The MAX17851 self verifies all communication within the message payload to simplify the software development on the host microcontroller.

Additionally, the MAX17851 monitors the host microcontroller communication to detect any failures. In the event of a communication failure with the host, the MAX17851 first tries to automatically recover communication. If unsuccessful, it assumes control. The MAX17851 can then optionally re-configure the daisy chain network that allows battery conditions to continue to be monitored. This allows for continued safe system operation. Lastly, the MAX17851 provides contactor signalling in the event that the battery conditions fall outside of the programmed range, putting the complete system into a failsafe state.

Applications

- Battery Management Systems (BMS)
- Electric and Hybrid Electric Vehicles (EV/HEV)
- Energy Storage Systems (ESS)

Benefits and Features

- Supports Maxim's Battery Management UART Protocol
- Supports ASIL D Functional Safety Requirements
  - Automated On-Chip Communication Verification with Simplified Host Reporting
  - Always-On Daisy-Chain Fault Polling with Fault Type and Fault Location
- Integrated System Watchdog with Redundant Daisy-Chain Controller
  - Programmable System Recovery
  - Programmable Memory for Daisy-Chain Configuration
  - Programmable Contactor Control
- Integrated Cell Balancing while Microcontroller is in Sleep Mode
  - Safe Long Term Balancing Monitoring with Programmable Interrupts
- UART Baud Rate Programmable Up to 4Mbps
- SPI Interface Up to 10MHz
- 1.71V to 5.5V Operation
- Ultra-Low Quiescent Current
- Operating Temperature Range from -40°C to +125°C (AEC-Q100)

Ordering Information appears at end of data sheet.

19-101190; Rev 0; 8/21

maxim integrated™

Simplified Application Diagram

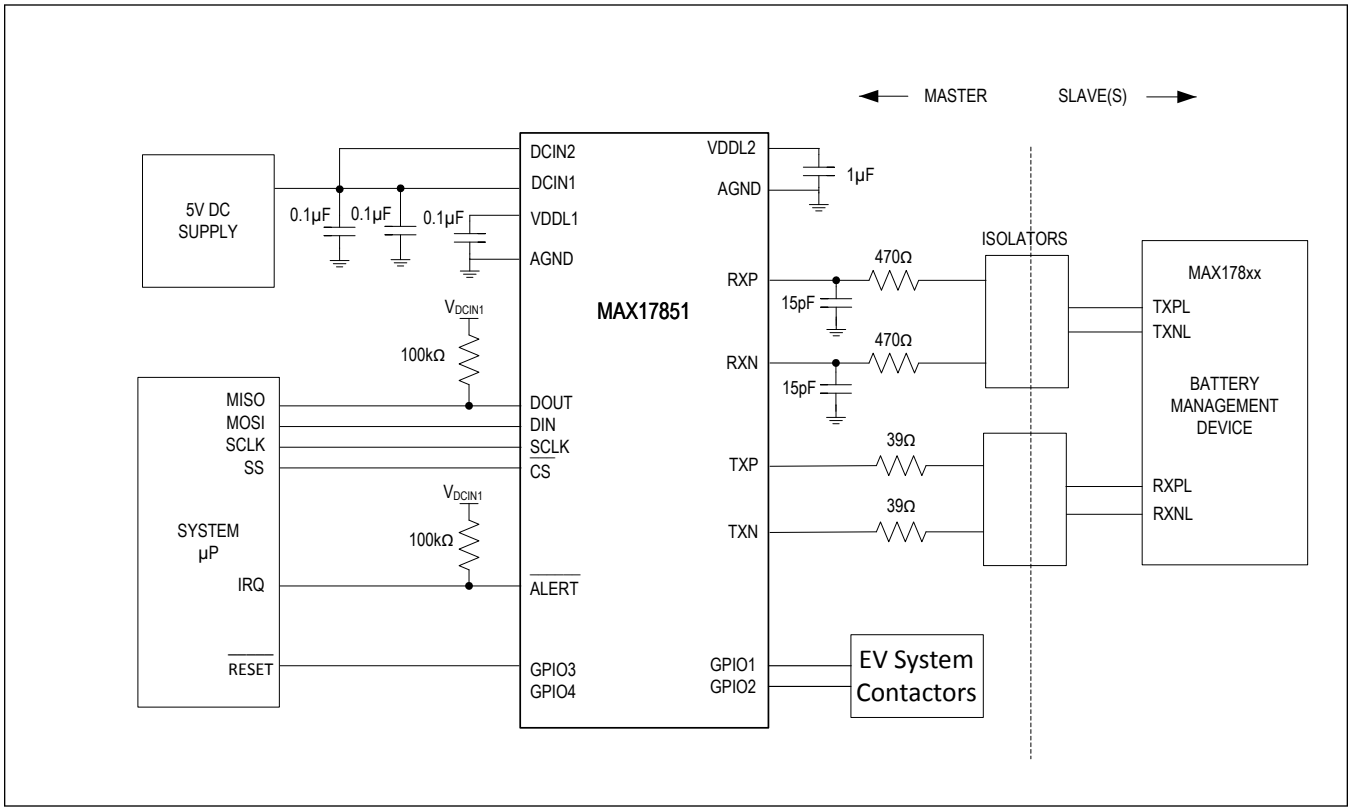


TABLE OF CONTENTS

TABLE OF CONTENTS (CONTINUED)

TABLE OF CONTENTS (CONTINUED)

LIST OF FIGURES

LIST OF TABLES

Absolute Maximum Ratings

DCIN1 to AGND ..... -0.3V to +6V
DCIN2 to AGND ..... -0.3V to +6V
V_{DDL1} to AGND..... -0.3V to +2.2V
V_{DDL2} to AGND..... -0.3V to +6V
TXP, TXN to AGND .....-0.3V to V_{DDL2} + 0.3V
DOUT to AGND ..... -0.3V to V_{DCIN1} + 0.3V
GPIO1,GPIO2,GPIO3,GPIO4 to AGND .. -0.3V to V_{DCIN1} + 0.3V
CS, DIN, SCLK, ALERT/ to AGND ..... -0.3V to V_{DCIN1} + 0.3V
RXP, RXN to AGND.....-32V to +32V
Maximum Continuous Current into Any Pin 20mA
Continuous Power Dissipation (Multilayer Board) (T_A = +70°C)
20 TSSOP (derate 11.1mW/ºC above +70ºC) 887mW
Operating Temperature Range .....-40°C to +125°C
Storage Temperature Range .....-55°C to +150°C
Junction Temperature (Continous) .....+150°C
Soldering Lead Temperature (10s).....+300°C

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

Package Information

TSSOP

Package Code	U20+7C
Outline Number	21-0066
Land Pattern Number	90-0116
Thermal Resistance, Single-Layer Board:	
Junction to Ambient (θ_{JA})	91°C/W
Junction to Case (θ_{JC})	20°C/W
Thermal Resistance, Four-Layer Board:	
Junction to Ambient (θ_{JA})	73.8 °C/W
Junction to Case (θ_{JC})	20°C/W

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 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*.

Electrical Characteristics

(V_{DCIN1} > = V_{DDL1}, V_{DCIN1} < 5.5V, V_{DCIN2} > = V_{DDL2}, V_{DCIN2} < 5.5V, V_{DDL1} = 1.8V, V_{DDL2} = 3.3V, T_A = T_{MIN} to T_{MAX}, unless otherwise noted, where T_{MIN} = -40°C and T_{MAX} = +125°C., Typical values are at T_A = +25°C. Operation is with the recommended application circuit.)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
POWER REQUIREMENTS						
DCIN1 Supply Voltage	V_{DCIN1}	V_{DDL1} = 1.8V	2.375		5.5	V
		V_{DCIN1} = V_{DDL1}	1.71	1.8	1.89	
DCIN2 Supply Voltage	V_{DCIN2}	V_{DDL2} = 3.3V nominal	4.5		5.5	V
		V_{DCIN2} = V_{DDL2}	3.2		5.5	
DCIN1 Static Supply Current	I_{DCIN1_STATIC}	f_{SCLK} = 0, SDO not loaded		920		uA
Incremental DCIN1 Communication Supply Current	I_{DCIN1_COMM}	Continuous SPI Reads, f_{SCLK} = 10MHz, SDO loaded 20pF, *Note: DCIN1 Current*		310		uA
DCIN2 Static Supply Current	I_{DCIN2_STATIC}	UART Baud Rate = 2MHz, TXIDLEHIZ, TX not active with 200pF load		10	20	uA

Electrical Characteristics (continued)

(V_{DCIN1} > = V_{DDL1}, V_{DCIN1} < 5.5V, V_{DCIN2} > = V_{DDL2}, V_{DCIN2} < 5.5V, V_{DDL1} = 1.8V, V_{DDL2} = 3.3V, T_A = T_{MIN} to T_{MAX}, unless otherwise noted, where T_{MIN} = -40°C and T_{MAX} = +125°C., Typical values are at T_A = +25°C. Operation is with the recommended application circuit.)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Incremental DCIN2 Communication Supply Current	I_{DCIN2_COMM}	UART Baud Rate = 2MHz, TXIDLEHIZ, TXL not active with 200pF		880	910	uA
V_{DDL1} REGULATOR						
Output Voltage	V_{DDL1}	0mA < I_{VDDL1} < 10mA, 2.375V < V_{DCIN1} < 5.5V	1.71	1.8	1.89	V
Short-Circuit Current	I_{DDL1_SC}	V_{DDL1} = AGND	5		50	mA
POR Threshold	$V_{DDL1\_PORRISE}$	V_{DDL1} rising	1.3	1.5	1.65	V
POR Hysteresis	V_{DDL1_PORHYS}		20	50		mV
V_{DDL2} REGULATOR						
Output Voltage	V_{DDL2}	0mA < I_{VDDL2} < 10mA, 4.5V < V_{DCIN2} < 5.5V	3.2	3.3	3.4	V
Short-Circuit Current	I_{DDL2_SC}	V_{DDL2} = AGND	13			mA
LOGIC INPUTS ($\overline{CS}$, DIN, SCLK)						
Input High Voltage	V_{IH}	V_{DCIN1} > 2.375V	0.7 x V_{DCIN1}			V
		1.71V < V_{DCIN1} < 1.89V	0.8 x V_{DCIN1}			
Input Low Voltage	V_{IL}	V_{DCIN1} > 2.375V			0.3 x V_{DCIN1}	V
		1.71V < V_{DCIN1} < 1.89V			0.2 x V_{DCIN1}	
Input Leakage Current	I_{IN}	Vin = 0V or V_{DCIN1} (Note 12)			±1	µA
Internal Impedance	R_{PD}	DIN, SCLK pull down to GND (Notes 13, 14)	40	100	160	kΩ
	R_{PU}	$\overline{CS}$ pull up to V_{DCIN1} (Notes 13, 14)	40	100	160	
Input Capacitance	C_{IN}			10		pF
Hysteresis Voltage	V_H			250		mV
LOGIC OUTPUTS (DOUT, $\overline{ALERT}$)						
Output High Voltage	V_{OH}	V_{DCIN1} >2.375V, I_{SOURCE} = 4mA, Note 10	V_{DCIN1} -0.2			V
		1.71V < V_{DCIN1} < 1.89V, I_{SOURCE} = 1.5mA, Note 10	V_{DCIN1} -0.2			
Output Low Voltage	V_{OL}	V_{DCIN1} >2.375V, I_{SINK} = 4mA			0.2	V
		1.71V < V_{DCIN1} < 1.89V, I_{SINK} = 1.5mA			0.2	
Output Tristate Leakage	I_{OZ}	V_{DOUT} = 0 and 5V, $V_{\overline{ALERT}}$ = 5V			±1	µA
Output Tristate Capacitance	C_{OZ}			10		pF

Electrical Characteristics (continued)

(V_{DCIN1} > = V_{DDL1}, V_{DCIN1} < 5.5V, V_{DCIN2} > = V_{DDL2}, V_{DCIN2} < 5.5V, V_{DDL1} = 1.8V, V_{DDL2} = 3.3V, T_A = T_{MIN} to T_{MAX}, unless otherwise noted, where T_{MIN} = -40°C and T_{MAX} = +125°C., Typical values are at T_A = +25°C. Operation is with the recommended application circuit.)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
GENERAL PURPOSE INPUTS/OUTPUTS (GPIO1, GPIO2, GPIO3, GPIO4)						
Output Leakage Current	I_{GPO_LKG}	V_{GPIOn} = 0 and 5V, GPIO disabled	-1		1	µA
Output High Voltage	V_{OH}	V_{DCIN1} > 2.375V, I_{SOURCE} = 4mA	V_{DCIN1} - 0.2			V
		1.71V < V_{DCIN1} < 1.89V, I_{SOURCE} = 1.5mA	V_{DCIN1} - 0.2			
Output Low Voltage	V_{OL}	V_{DCIN1} > 2.375V, I_{SINK} = 4mA			0.2	V
		1.71V < V_{DCIN1} < 1.89V, I_{SINK} = 1.5mA			0.2	
Input High Voltage	V_{IH}	V_{DCIN1} > 2.375V	0.7 x V_{DCIN1}			V
		1.71V < V_{DCIN1} < 1.89V	0.8 x V_{DCIN1}			
Input Low Voltage	V_{IL}	V_{DCIN1} > 2.375V			0.3 x V_{DCIN1}	V
		1.71V < V_{DCIN1} < 1.89V			0.2 x V_{DCIN1}	
Pulldown Resistance	R_{GPIO}	GPIOn_CFG configured as input		2M		Ω
Output Short Circuit Current	I_{OSS}	I_{SINK}, V_{GPIOn} = V_{DCIN1} = 5V			50	mA
POWER AND GROUND FAULT DETECTION						
Open Detection Voltage (V_{DDL1})	$V_{ALRTVDDL1}$		1.62	1.65	1.68	V
Open Detection Voltage (V_{DDL2})	$V_{ALRTVDDL2}$		2.90	3.00	3.10	V
Over Voltage Detection (VDCIN1)	V_{ALRTOV_DCIN1}		5.6	5.7	5.8	V
Overvoltage Detection Hysteresis (VDCIN1)	$V_{DCIN1OV\_HYS}$		100		150	mV
Over Voltage Detection (VDCIN2)	V_{ALRTOV_DCIN2}		5.6	5.7	5.8	V
Overvoltage Detection Hysteresis (VDCIN2)	$V_{DCIN2OV\_HYS}$		100		150	mV
OSCILLATORS						
LF Oscillator Frequency	f_{LFOSC}		32.11	32.768	33.42	kHz
HF Oscillator Frequency	f_{HFOSC}		62.72	64	65.28	MHz
UART INPUTS (RXP, RXN)						
Differential Input High Threshold	V_{TH}	*Note 1*	$V_{DDL1}/2$ - 400mV	$V_{DDL1}/2$	$V_{DDL1}/2$ + 400mV	V
Differential Input Zero-Crossing Threshold	V_{ZC}	*Note 1*	-400	0	400	mV
Differential Input Low Threshold	V_{TL}	*Note 1*	$-V_{DDL1}/2$ - 400mV	$-V_{DDL1}/2$	$-V_{DDL1}/2$ + 400mV	V

Electrical Characteristics (continued)

(V_{DCIN1} > = V_{DDL1}, V_{DCIN1} < 5.5V, V_{DCIN2} > = V_{DDL2}, V_{DCIN2} < 5.5V, V_{DDL1} = 1.8V, V_{DDL2} = 3.3V, T_A = T_{MIN} to T_{MAX}, unless otherwise noted, where T_{MIN} = -40°C and T_{MAX} = +125°C., Typical values are at T_A = +25°C. Operation is with the recommended application circuit.)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Differential Input Hysteresis	V_{HYST}	Note 1	30	80	160	mV
Common-Mode Voltage Bias	V_{CM}			$V_{DDL1}/2$		V
Input Capacitance	C_{IN}			2		pF
Leakage Current	I_{LKG_RX}	V_{RX} = 0.9V		±1.0		µA
Input Resistance to Common Mode Voltage	R_{RXIN}			0.8		MΩ
UART OUTPUTS (TXP, TXN)						
Output Low Voltage	V_{OL}	I_{OL} = -20mA			V_{AGND} + 0.4	V
Output High Voltage	V_{OH}	I_{OH} = 20mA	V_{DDL2} - 0.4			V
UART TIMING						
Bit Period Except for Second STOP Bit	t_{BIT}	f_{UART} = 4Mbps, (Note 2, Note 3)		16		$1/f_{HFOSC}$
Bit Period Except for Second STOP Bit	t_{BIT}	f_{UART} = 2Mbps (Note 3, Note 4)		32		$1/f_{HFOSC}$
		f_{UART} = 1Mbps (Note 3, Note 4)		64		
		f_{UART} = 0.5Mbps (Note 3, Note 4)		128		
Second STOP Bit Period	$t_{STOPBIT}$	Note 2, Note 3		1.125		t_{BIT}
Tx Idle to START Setup Time	t_{TXSTSU}	Note 2, Note 3, Note 4		0.5		t_{BIT}
STOP Hold Time to Idle	t_{SPHD}	Note 2, Note 3			0.5	t_{BIT}
Rx Minimum Idle Time (STOP Bit to START Bit)	$t_{RXIDLESPST}$	(Note 2, Note 3)	1			t_{BIT}
Tx Idle Time	$t_{TXIDLESPST}$	(Note 1, Note 2)		0.5		t_{BIT}
Rx Fall Time	t_{FALL}	Note 2, Note 3, Note 4			0.5	t_{BIT}
Rx Rise Time	t_{RISE}	(Note 2, Note 3, Note 4)			0.5	t_{BIT}
UART MESSAGE TIMING						
SPI Command to Tx Valid Propagation Delay	t_{TX}	(Note 5)	125		250	ns
Tx Valid to Rx Valid Up Stack Delay	t_{RXUP}	(Note 6)			n x t_{PROP}	
Tx Valid to Rx Valid Down Stack Delay	t_{RXDN}	(Note 6)			n x t_{PROP}	
End of STOP Character to RX_STOP_ALRT Flag True	t_{ALERT}	(Note 7)			2	t_{BIT}
SPI START to UART Slave Device Register Write Delay	t_{REGWR}	(Note 8, Note 9)			8 / f_{SCLK} + 130 x t_{BIT} + n x t_{PROP}	

Electrical Characteristics (continued)

(V_{DCIN1} > = V_{DDL1}, V_{DCIN1} < 5.5V, V_{DCIN2} > = V_{DDL2}, V_{DCIN2} < 5.5V, V_{DDL1} = 1.8V, V_{DDL2} = 3.3V, T_A = T_{MIN} to T_{MAX}, unless otherwise noted, where T_{MIN} = -40°C and T_{MAX} = +125°C., Typical values are at T_A = +25°C. Operation is with the recommended application circuit.)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
SPI TIMING						
SCLK Frequency	f_{SCLK}	V_{DCIN1} > 2.375V (*Note 15*)			10	MHz
SCLK Period	t_{CP}		100			ns
SCLK Pulse Width High	t_{CH}		40			ns
SCLK Pulse Width Low	t_{CL}		40			ns
CSB Fall to SCLK Rise Setup Time	t_{CSS0}	To first SCLK rising edge (RE).	40			ns
CSB Fall to SCLK Rise Hold Time	t_{CSH0}	Applies to inactive RE preceding first RE.	25			ns
SCLK Rise to CSB Rise Hold Time	t_{CSH1}	Applies to 8+8*n RE.	25			ns
CSB Pulse Width High	t_{CSPW}		400			ns
CSB Pulse Width High After SWPOR	t_{CSPWSP}	Applies after an accepted/executed SWPOR command.	100			us
SDI to SCLK Rise Setup Time	t_{DS}		10			ns
SDI to SCLK Rise Hold Time	t_{DH}		10			ns
SCLK Fall to SDO Transition	t_{DOT}	V_{DCIN1} > 2.375V, C_{LOAD} = 20pf			40	ns
SCLK Fall to SDO Hold	t_{DOH}	C_{LOAD} = 0pf	2			ns
CSB Fall to SDO Transition	t_{DOE}	V_{DCIN1} > 2.375V, C_{LOAD} = 20pf			40	ns
CSB Rise to SDO Hi-Z	t_{DOZ}	VDCIN1 > 2.375V, Output disable time			40	ns

Note 1: Differential signal (V_{RXP} - V_{RXN}) where V_{RXP}, V_{RXN} do not exceed a common-mode voltage range of ±25V.

Note 2: All parameters measured based on differential signal.

Note 3: Guaranteed by design and not production tested.

Note 4: Fall time measured 90% to 10%, rise time measured 10% to 90%.

Note 5: Measured from falling edge of 8th SCLK cycle of the WR_NXT_LD_Q SPI command byte (B0h).

Note 6: t_{PROP} is the maximum propagation delay through a slave device in a given direction. Refer to the UART slave device data sheet for the actual delay. The number of UART slave devices is denoted by n.

Note 7: Measured from end of 10th bit of stop character.

Note 8: Parameter t_{REGWR} is the minimum amount of time needed to write a register in the nth slave device of the daisy-chain. It is measured from the start of the SPI transaction WR_NXT_LD_Q (B0h) that initiates transmission of a WRITEALL message to when the nth device receives a valid WRITEALL message. For example, for 4MHz SPI frequency, 2Mbps UART baud rate, n = 10 and t_{PROP} = 3 x t_{BIT}, t_{REGWR} = 2µs + 65µs + 15µs = 82µs.

Note 9: Computation of t_{REGWR} consists of three terms: 1) duration of the SPI transaction, 2) partial duration of the UART message, and 3) propagation delay of the UART message. The first term equals the number of bits in the SPI transaction (8) x the SPI bit time (1 / f_{SCLK}). The second term equals the time from the start of the WRITEALL message to the first STOP bit of the last PEC nibble. The last PEC nibble is the 11th character in the message. With each character lasting 12 UART bit times, there are 11 x 12 = 132 bit times from the start of the message to the end of the last PEC nibble. Since the write occurs just before the two STOP bits of the 11th character, the term is actually 130 x t_{BIT}. The third term is the propagation delay required for the WRITEALL message to get to the nth device.

Note 10: V_{OH} specification for $\overline{ALERT}$ is determined by the external pullup resistor and leakage current of the network.

Note 11: Static logic inputs with V_{IL} = AGND and V_{IH} = V_{DCIN1}. CSB = V_{IH} (if pullup is active).

Note 12: No internal safety pullup/pulldown impedances active. Input buffers only.

Note 13: Internal safety pullup/pulldown impedances available with enable function.

Note 14: If pullup is supported, note CSB connection and diode to V_{DDL1}. This diode is present regardless of enable mode.

Note 15: Applications must afford time for the device to drive data on the SDO bus and meet the µC setup time prior to the µC latching in the result on the following SCLK rising edge. In practice, this can be determined by loading and µC characteristics, and the relevant t_{DOT}/t_{DOE}.

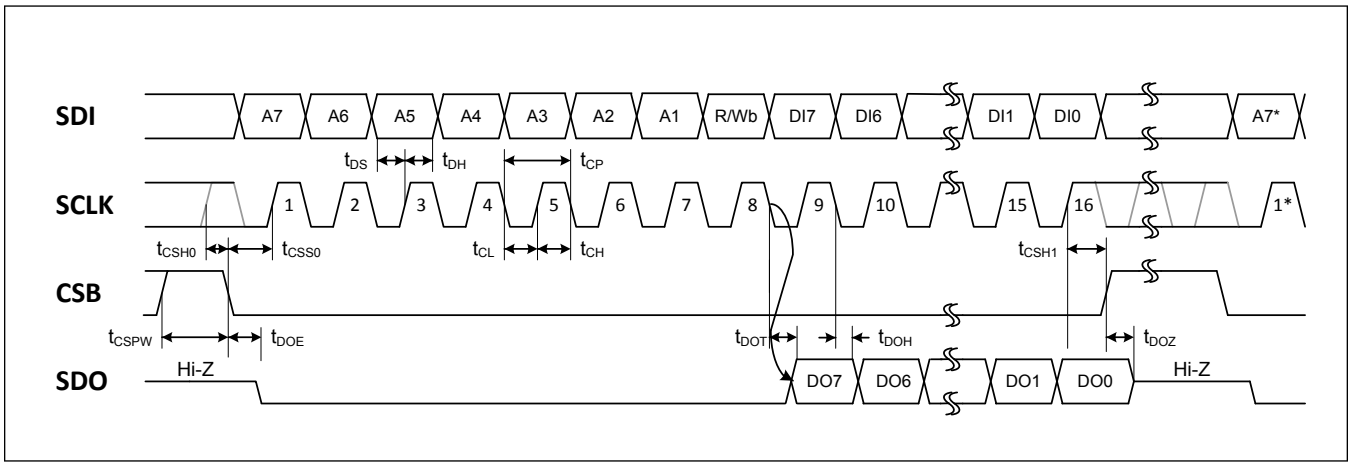


Figure 1. SPI Timing Diagram

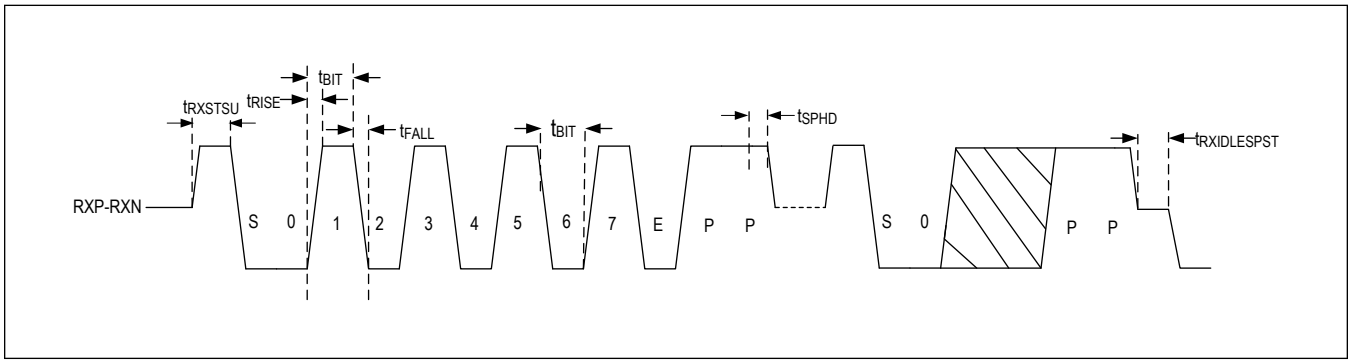


Figure 2. Receive UART Timing

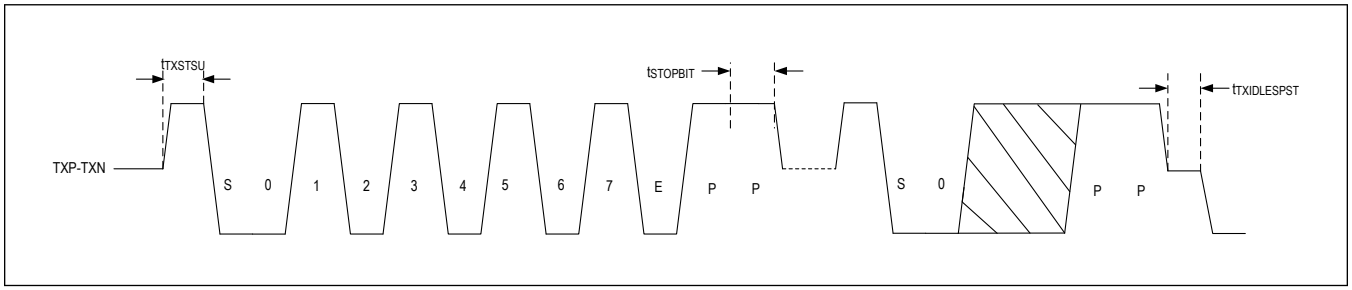


Figure 3. Transmit UART Timing

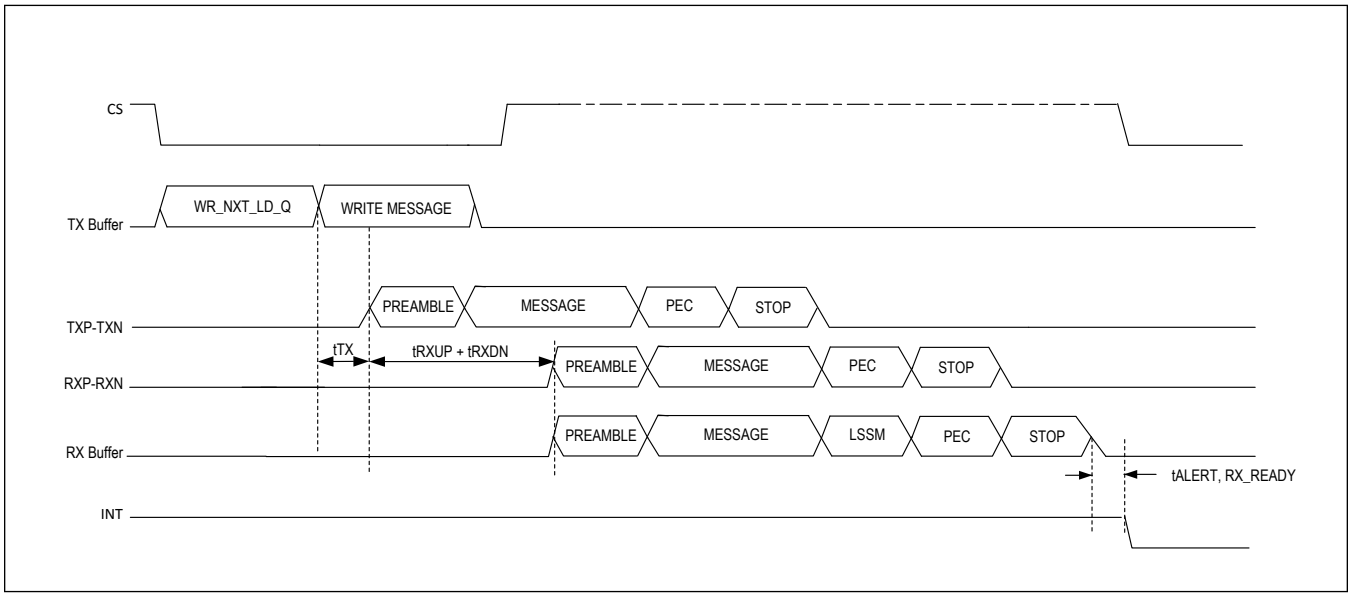


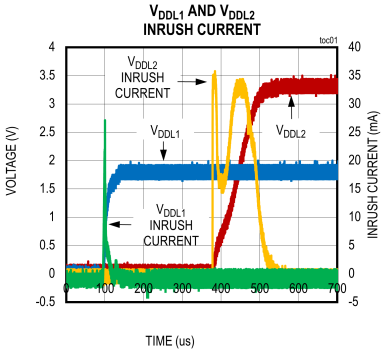
Figure 4. TX Buffer Write and Read

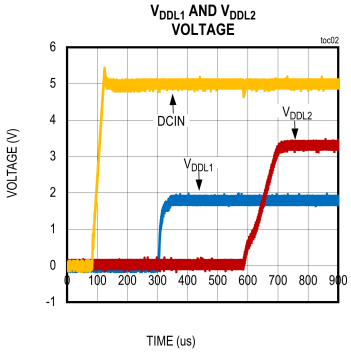
Note: DCIN1 Current: The total DCIN1 current is the summation of IDCIN1_STATIC and IDCIN1_COMM. IDCIN1_COMM is dependent of SPI operation frequency, SDO output data, DCIN1 voltage, and the total capacitance on each of the communication lines. The data patterned use assumed and average 50% transition of the data pattern with a 5V DCIN1.

Note: DCIN2 Current: The total DCIN2 current is the summation of IDCIN2_STATIC and IDCIN2_COMM. IDCIN2_COMM is dependent on UART baud rate, UART output data, DCIN2 voltage, and the total capacitance on each of the communication lines. The data pattern used assumes an average of 67% transition of the data pattern with a 3.3V DCIN2.

Typical Operating Characteristics

(DCIN1 = 3.3V, DCIN2 = 5.0V, T_A = +25°C unless otherwise noted)

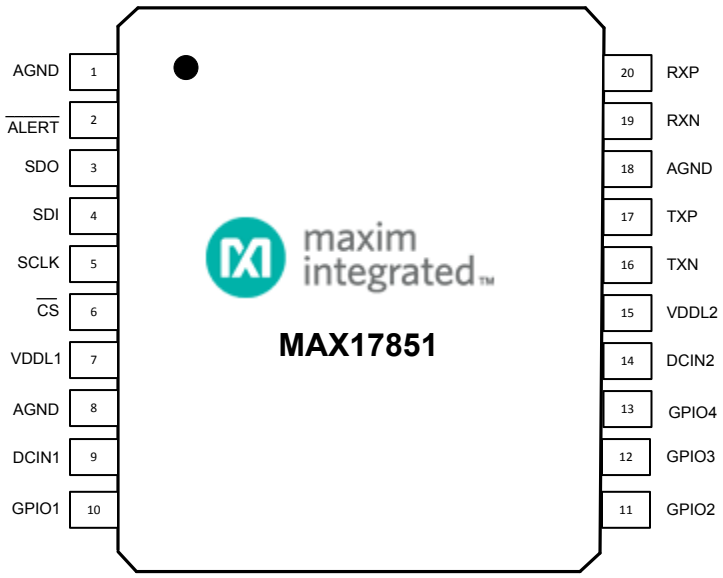




Pin Configuration

MAX17851A

Top View



Pin Description

PIN	NAME	FUNCTION	TYPE
1	AGND	Analog Ground. Connect to the power supply ground.	Ground
2	$\overline{ALERT}$	Active-Low, Open-Drain Interrupt Output. Connect a pullup resistor to this pin per application requirements. This pin is asserted if any interrupt flag is set.	Output
3	SDO	SPI Data Output. Connect to SDI/MISO input of SPI master. This output is tri-stated when $\overline{CS}$ is de-asserted. When $\overline{CS}$ is asserted, this pin is driven between DCIN1 and AGND supplies.	Output
4	SDI	SPI Data Input. Connect to SDO/MOSI output of SPI master. 5V tolerant.	Input
5	SCLK	SPI Clock Input. Connect to SCLK output of SPI master. 5V tolerant.	Input
6	$\overline{CS}$	Active-Low SPI Chip-Select Input. Connect to the Slave Select output of the SPI master. Assert to enable the SPI port. 5V tolerant.	Input
7	V_{DDL1}	Power output of 1.8V LDO and supply for internal oscillators and logic. To supply power directly, connect to DCIN1. Decouple per application circuit.	Power
8	AGND	Analog Ground. Connect to the power supply ground.	Ground
9	DCIN1	Power Input for 1.8V LDO Regulator. Decouple per application circuit.	Power
10	GPIO1	General Purpose I/O. Default configuration is General Purpose Input. Pin can be paralleled with GPIO2 for hardware redundancy. If not used, pin can remain unconnected. See section *GPIO Control* for more information.	Input/Output
11	GPIO2	General Purpose I/O. Default configuration is General Purpose Input. Pin can be paralleled with GPIO1 for hardware redundancy. If not used, pin can remain unconnected. See section *GPIO Control* for more information.	Input/Output
12	GPIO3	General Purpose I/O. Default configuration is General Purpose Input . If not used, pin can remain unconnected. See section *GPIO Control* for more information.	Input/Output
13	GPIO4	General Purpose I/O. Default configuration is General Purpose Input . If not used, pin can remain unconnected. See section *GPIO Control* for more information.	Input/Output
14	DCIN2	Power input for 3.3V LDO regulator. Decouple per application circuit.	Power
15	V_{DDL2}	Power output of 3.3V LDO and supply for UART. To supply power directly, connect to DCIN2. Decouple per application circuit.	Power
16	TXN	UART Transmitter Negative Output. Connect to RXN port of UART slave device per application circuit. This pin is driven between the V_{DDL2} and AGND supplies.	Output
17	TXP	UART Transmitter Positive Output. Connect to RXP port of UART slave device per application circuit. This pin is driven between the V_{DDL2} and AGND supplies.	Output
18	AGND	Analog Ground. Connect to the power supply ground.	Ground
19	RXN	UART Receiver Negative Input. Connect to TXN port of UART slave device per application circuit.	Input
20	RXP	UART Receiver Positive Input. Connect to TXP port of UART slave device per application circuit.	Input

Functional Diagrams

MAX17851 Functional Block Diagram

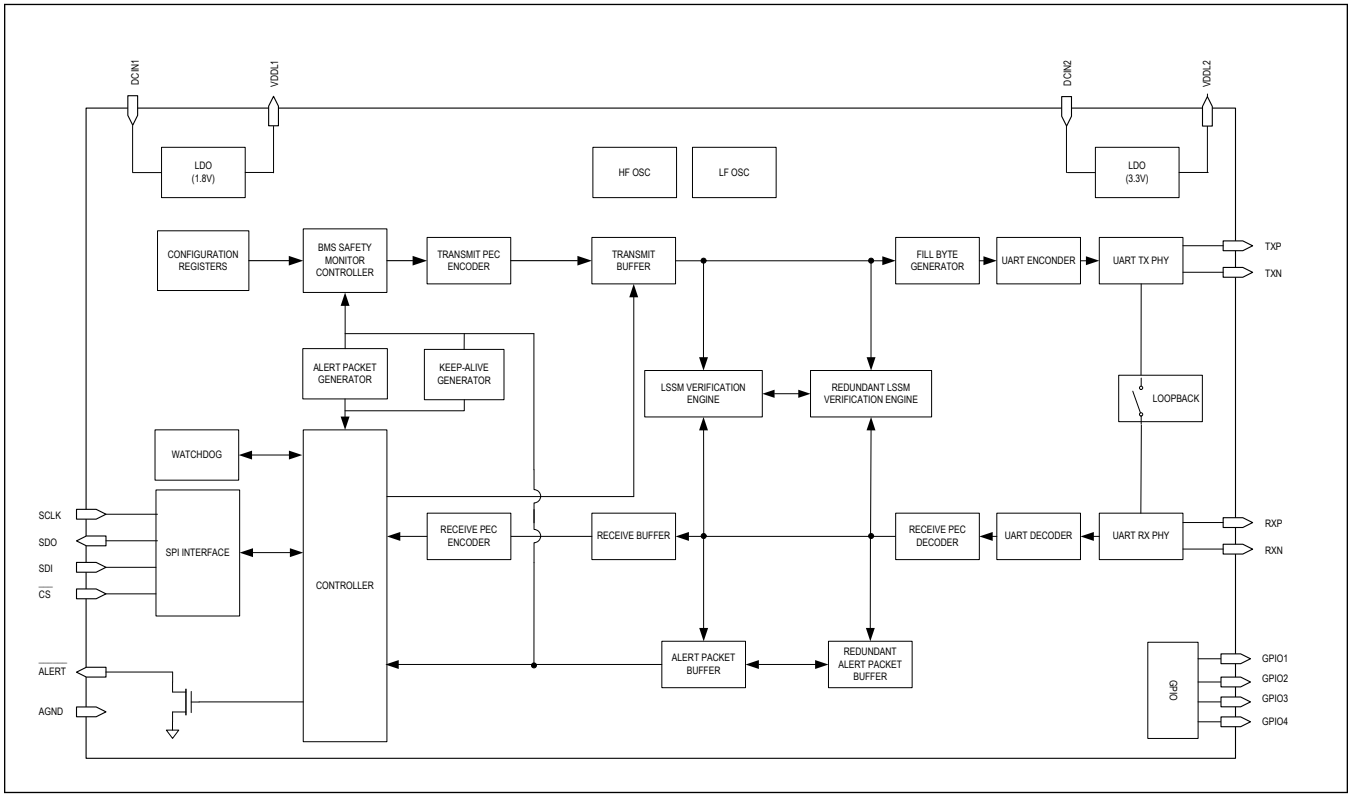


Table 1. MAX17851 Functional Blocks Power Distribution

BLOCK	SUPPLY
Oscillators	V_{DDL1}
Configuration Registers	
BMS Safety Monitor Controller	
Transmit PEC Encoder	
Alert Packet Generator	
Keep Alive Generator	
Watchdog	
Digital Controller	
Transmit Buffer	
Fill Byte Generator	
UART Encoder	
LSSM/Redundant LSSM	
Alert Packet Buffer/Redundant Alert Packet Buffer	
UART Decoder	
Receive PEC Decoder	
Receive Buffer	

Table 1. MAX17851 Functional Blocks Power Distribution (continued)

BLOCK	SUPPLY
Receive PEC Encoder	
UART RX Phy	
UART TX Phy	V_{DDL2}
SPI Interface	
$\overline{ALERT}$ Output	DCIN1
GPIO	
V_{DDL1} LDO Regulator	
V_{DDL2} LDO Regulator	DCIN2

Detailed Description

The MAX17851 allows a host controller with a standard SPI port to reliably communicate with one or more battery management monitors that use Maxim's battery management UART protocol. Together with the host controller, the MAX17851 is the master for communications with the slave devices.

There are several distinguishing features that make the MAX17851 suited for high automotive safety integrity:

- Dual automated lockstep UART processing cores (lock step safety measure (LSSM)) to ensure UART data integrity for simplified host verification. See *Lockstep Safety Measure Verification* for more detail.
- Independent embedded controller to verify the integrity of the host communication with the ability to recover the host microcontroller and, if unsuccessful, reconfigure and poll the daisy-chain for errors. See *BMS Safe Monitoring Mode* for more detail.
- GPIO programmability for system contactor control.
- Embedded user accessible test modes for accelerated safety verification.
- Automated polling of the diasy chain battery management monitors using the ALERTPACKET.
- Dual oscillators for verification of digital operation.
- Multiple LDO regulators with over/under voltage detection.

Operational Modes

There are three modes of device operation:

- Commanded Operation Mode
- Automated BMS Safe Monitoring Mode
- Sleep Mode

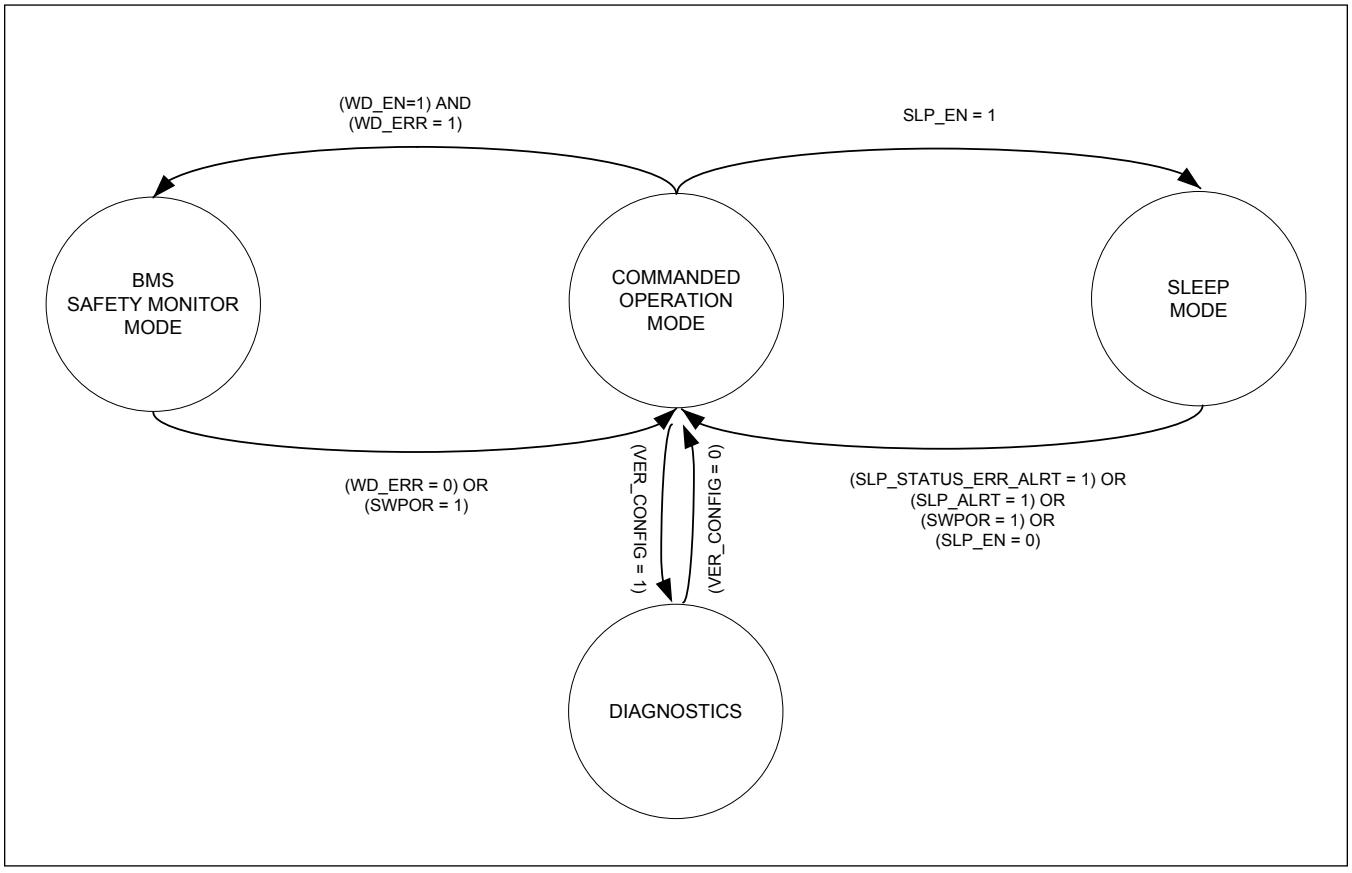


Figure 5. Operational Modes

Commanded Operation Mode

The default state upon power up is the Commanded Operation mode with the watchdog functionality disabled. In Commanded Operation mode, the MAX17851 is under full control of the host microcontroller.

In order to allow entry into the BMS Safe Monitoring mode, the user must enable the watchdog state machine by setting WD_EN. Invalid watchdog response(s) exceeding WD_DBNC will cause the MAX17851 to enter BMS Safe Monitoring mode. The duration of the watchdog timeout is dependent upon WD_OPEN, WD_CLOSED, WD_DIV, and WD_1UD register settings.

Commanded Operational mode may only be re-entered by refreshing the watchdog key register with a valid response (WD_KEY) or issuing a software POR (SWPOR=1).

Sleep Mode

Sleep mode is designed to support long term cell balancing in which the host microcontroller can enter a low power state. The MAX17851 provides periodic wake events for host action independent of controller area network (CAN) bus activity. Additionally, Sleep mode provides safety monitoring of safety critical battery parameters (OV, UV, OT, UT) from the daisy-chain devices with or without cell balancing being enabled.

Sleep mode is entered by the assertion of SLP_EN, which starts an internal timer controlled by SLP_CBNTFY. Entry into Sleep mode disables the watchdog, which prevents the device from transitioning to BMS Safe Monitoring mode while in Sleep mode. Therefore, the watchdog does not need to be refreshed in Sleep mode.

Sleep mode is exited by writing SLP_EN to a 0, the expiration of the SLP_CBNTFY timer, exceeding the

ALRTPCKT_DBNC counter threshold, or through sending the SWPOR command. See Figure 6.

Automated Keep-Alives and Alert Packet communication as programmed in the ALRTPCKT_TIMING register is disabled in this mode, requiring the daisy chain devices to programmatically force the SHDNL pin high (i.e. force the Standby state).

Configuring Sleep mode ALERTPACKETs (SLP_ALRTPCKTEN = 1) allows for safety critical battery monitoring through transmission of the ALERTPACKET command. The frequency of the ALERTPACKET is set by the SLP_SCAN_DLY register. If a persistent ALERTPACKET error is qualified according to the ALRTPCKT_DBNC register, the SLP_STATUS_ERR_ALRT status register is asserted, and Sleep mode is exited. This alert can be propagated to the host controller through the $\overline{\text{ALERT}}$ to a microcontroller interrupt pin if the associated SLP_STATUS_ERR_ALRTEN is enabled.

The use of an interrupt pin is required to wake the microcontroller independent of the CAN bus.

Note: Attempting to write to the Configuration Register space or TX Command Register space in Sleep mode will be ignored, cause the SPI_ERR status bit to be set, and assert the $\overline{\text{ALERT}}$ pin.

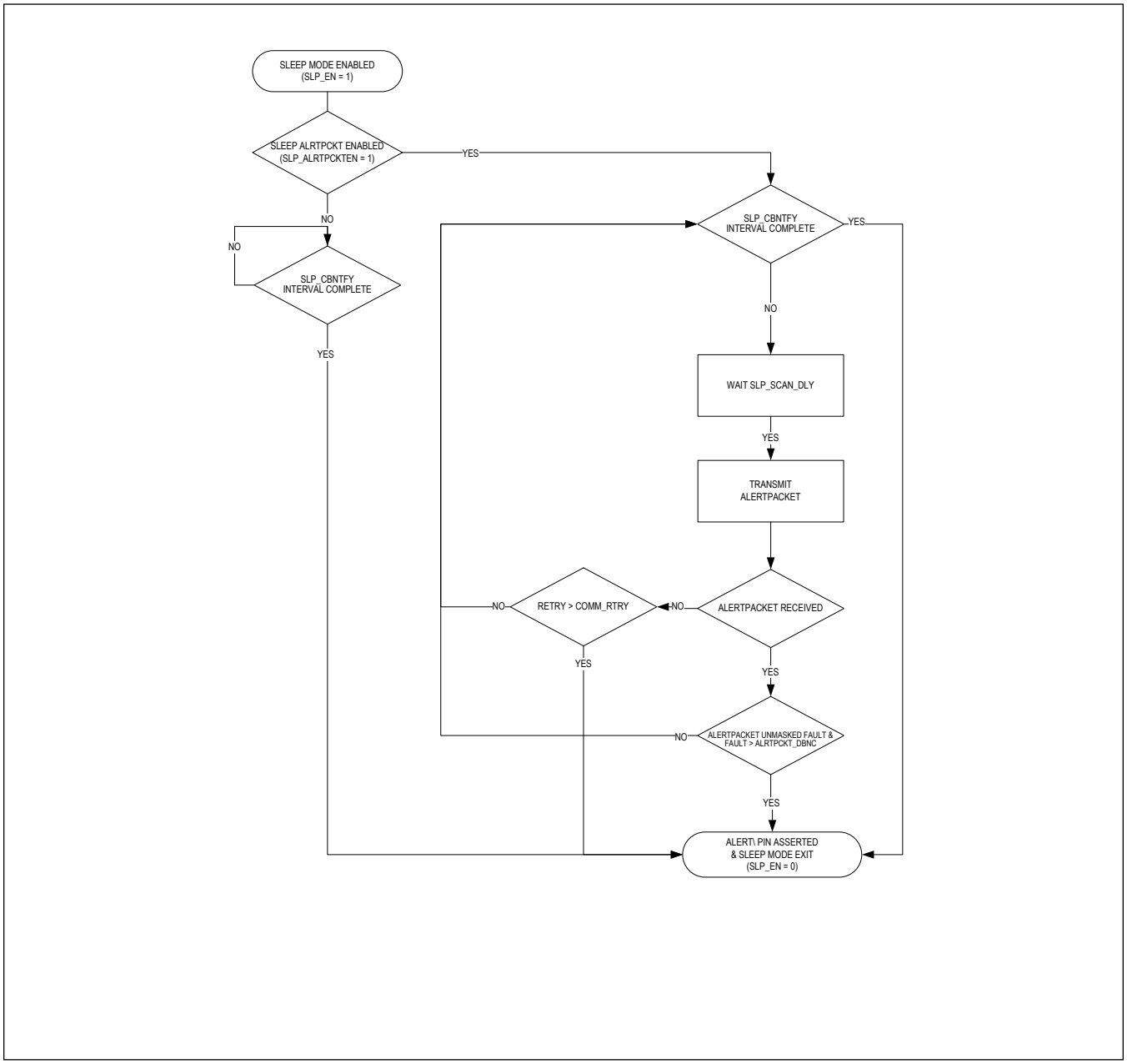


Figure 6. Sleep Mode State Diagram

BMS Safe Monitoring Mode

BMS Safe Monitoring mode provides the following capabilities:

1. Constant BMS Data Acquisition System monitoring for all safety critical events (OV, UV, OT, UT, etc) in case the system microcontroller becomes unresponsive.
2. Recovery/reset signaling to the system via GPIO signals, which can be used to reset the microcontroller and/or power supply network.

3. Vehicle safe-state signaling to the battery contactors via redundant GPIO signals in the event that a BMS Data Acquisition System exceeds user-defined, safety-critical threshold(s).

Continued BMS Data Acquisition Monitoring occurs through the automatic generation and parsing of ALERTPACKET commands. These commands provide real-time status of the BMS Data Acquisition Systems, which is reflected in the STATUS bits of the ALERTPACKET buffer. Any of the individual STATUS bits can be masked using the STATUS_DBNC_MASK[15:0] register, providing flexibility within the application to select any system critical event or combination of events, such as cell over-voltage, cell under-voltage, cell over-temperature, cell under-temperature, over-current, BMS module internal failures, cell balancing failures, etc. In BMS Safe Monitoring mode, the BMS Data Acquisition Systems shall be configured for auto-polling of critical system parameters. This is achieved through the automatic execution of a MAX17851 load configuration memory operation in BMS Safe Monitoring mode. See the *Configuration Memory Transmission* section for details.

Note: It is recommended to mask any STATUS bit that requires a write to logic 0 to clear the fault; otherwise, a single error will falsely indicate a persistent error (e.g., BMS Data Acquisition System faults, such as PEC and ALRTRST). Communication errors (i.e., PEC) are independently verified by the MAX17851 LSSM in all modes of operation.

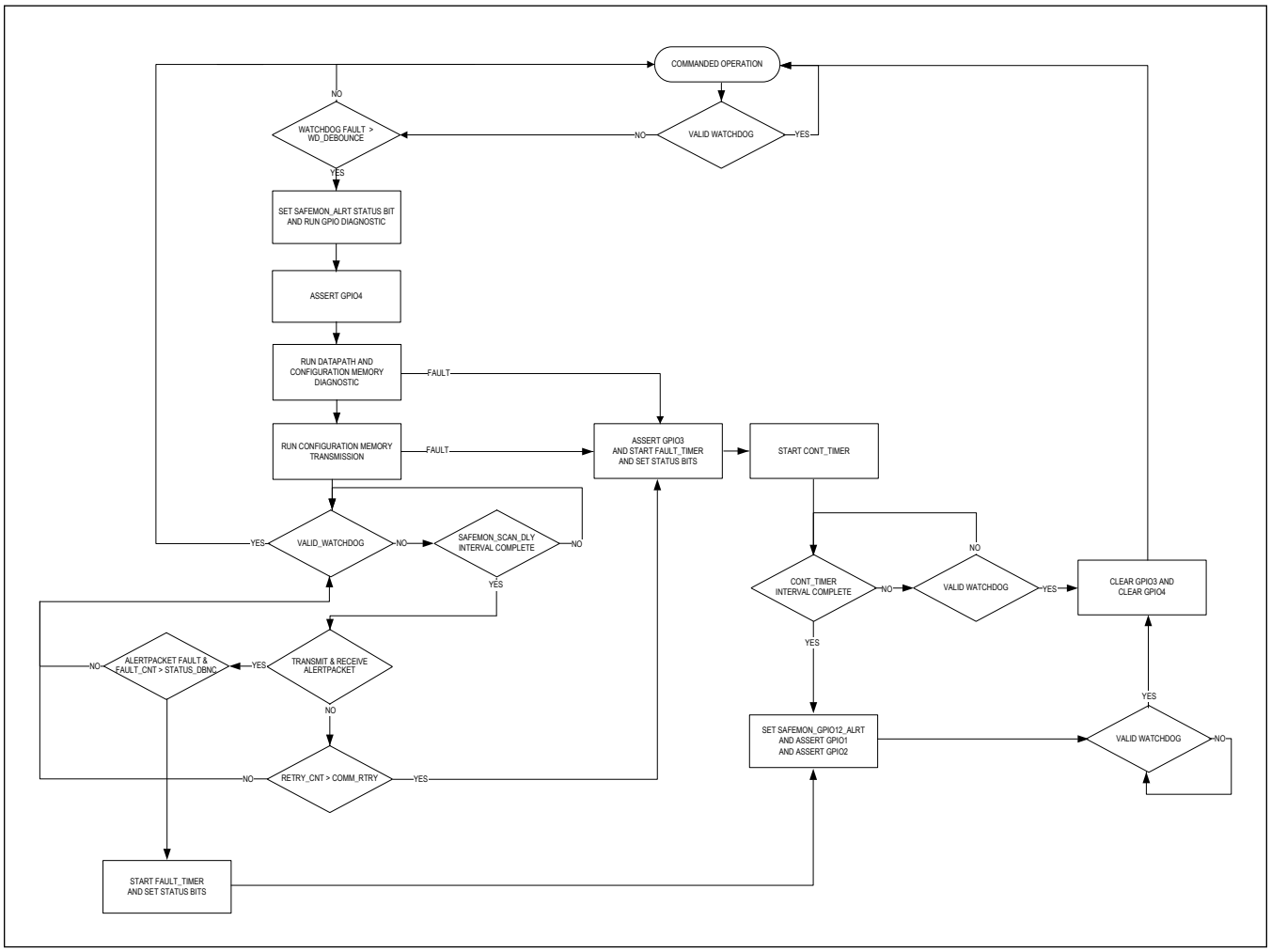


Figure 7. BMS Safe Monitoring Mode Flow Chart

Entering BMS Safe Monitoring Mode and Safety Measure Diagnostic State

When configured as a master (MS_EN = 1x), the MAX17851 enters BMS Safe Monitoring mode only after the number

of consecutive invalid watchdog responses exceeds the watchdog debouncing threshold (WD_DBNC). If WD_EN is not enabled, a configured master MAX17851 will never enter BMS Safe Monitoring mode.

When configured as a slave (MS_EN = 0x), the MAX17851 only enters BMS Safe Monitoring mode in response to a logic high assertion on GPIO3.

On entry, the SAFEMON_ALRT status bit immediately asserts, and the $\overline{\text{ALERT}}$ pin is asserted by default. If different behavior is desired, $\overline{\text{ALERT}}$ pin assertion can be disabled via the SAFEMON_ALRTEN register bit.

Following the assertion of the SAFEMON status bit, the MAX17851 enters a Safety Measure Diagnostic state. The Safety Measure Diagnostic state allows the host to verify the correct functionality of the BMS Safe Monitoring Control logic, watchdog logic, and the GPIO outside of the autonomous MAX17851 BMS Safe Monitoring mode operation. The MAX17851 remains in the diagnostic state according to the programmed value of GPIOREC_DLY.

During the diagnostic state, all GPIO outputs are simultaneously asserted by the BMS Safe Monitoring controller. Although asserted by the controller, the driven pin output is separately masked via the SM_GPIO[x]_MASK register bits. When masked, the GPIO pins idle in their inactive state, while the internal GPIO drive level reads via the GPIO[x]_RD registers. Reading the GPIO[x]_RD registers allows the host microcontroller to assert the correct GPIO operation.

As an example, if GPIO1_CFG = 110 (SAFEMON HI output, idles LO) and SM_GPIO1_MASK =1, GPIO1_RD is asserted high (active), but the GPIO1 pin output is driven low (inactive).

Note: If GPIO masking is disabled (SM_GPIO[x]_MASK = 0), GPIO signals can be sensed by the host microcontroller to verify proper functionality at the board level. To prevent unwanted disruption of vehicle operation, SM_GPIO[x]_MASK register bits are enabled by default.

It is recommended that GPIO1 and GPIO2 always be masked to avoid unintended signaling to the contactor and prevent unwanted disruption of vehicle operation. GPIO3 and GPIO4 must be evaluated in terms of the application to determine if they can be unmasked for system diagnostics. Unmasking GPIO3 and GPIO4 without such consideration may result in undesired operation of the MAX17851.

Upon successful reading of the GPIO[x]_RD registers, the watchdog may be refreshed and normal Commanded Operation mode resumes. If the watchdog is not refreshed by the end of GPIOREC_DLY, BMS Safe Monitoring mode continues.

Microcontroller and Power Supply Recovery

GPIO3 and GPIO4 are designed to interface with the microcontroller and power supply and attempt recovery in BMS Safe Monitoring mode.

For a single UART application, this is achieved by connecting GPIO3 and GPIO4 to the enable or reset pins of the microcontroller and power supply. If the watchdog is not refreshed in BMS Safe Monitoring mode before the expiration of GPIOREC_DLY, the MAX17851 asserts GPIO4 at the pin output, attempting recovery of the microcontroller. If the MAX17851 encounters a communication fault during BMS Safe Monitoring mode execution, it asserts GPIO3 at the pin output, which attempts recovery of the power supply. See the applications section *Single UART Operation* for diagrams and additional detail.

For a dual UART application, this is achieved by connecting GPIO4 of the master to the enable or reset pins of the microcontroller, and GPIO3 of the slave MAX17851 to the power supply. If the watchdog is not refreshed in BMS Safe Monitoring mode before the expiration of GPIOREC_DLY, the MAX17851 asserts GPIO4 at the pin output attempting recovery of the microcontroller. If the master encounters a communication fault during BMS Safe Monitoring mode execution, it asserts GPIO3 at the pin output, which provides a logic indicator to the slave to initiate failsafe monitoring on a redundant communication path in response to the master fault. If the slave encounters a communication fault during BMS Safe Monitoring mode, it asserts GPIO3 at the pin output, which attempts recovery of the power supply. See *Dual UART Operation* in the Applications section for diagrams and additional detail.

All GPIO pins are configured as general purpose inputs by default to prevent potential board-level contention on power up. GPIO pins attached to a microcontroller or power supply should typically be configured as SAFEMON 1-Shot LO output, which pulls the GPIO pin low for 100ms when asserted by the BMS Safe Monitoring controller. For dual UART architectures with two MAX17851 devices present, GPIO3 of the master should be configured as a SAFEMON active HI output, and GPIO4 of the slave MAX17851 should be configured as a SAFEMON input. The host microcontroller should program all GPIO according to the application. See *GPIO Control* for additional detail.

Note: The SM_GPIO[x]_MASK registers are not applicable after the expiration of GPIOREC_DLY.

If NOMON = 1, all GPIOs are simultaneously asserted at the pin output after the expiration of GPIOREC_DLY. This attempts to recover the power supply and microcontroller as discussed above, but also asserts GPIO1 and GPIO2, which are designed to open the battery contactors.

Note: For immediate recovery and contactor assertion, NOMON must be set for all MAX17851 devices in a given architecture.

BMS Safe Monitoring Operation

When NOMON = 0, the MAX17851 begins initialization of the BMS Safe Monitoring mode after the GPIO validation and system recovery attempt. This is done by:

1) Performing self-diagnostics to confirm proper internal operation of the UART datapath and the configuration memory.

2) Transmitting the contents of the configuration memory to the the daisy-chain devices.

The self-diagnostics and programming of the daisy-chain are constructed to meet ISO26262 requirements for ASIL-D systems.

Transmitting the contents of the configuration memory to the daisy-chain devices allows the user to reprogram safety thresholds to new requirements defined by operation during a system fault. Note that in this condition, the state of the battery cells should still qualify as being in normal operating boundaries.

If an error is detected during either of the prior procedures, a status fault is set indicating the error description in the STATUS_SAFEMON register. The CONT_TIMER then starts and increments to the programmed value (the timer has an infinite time setting by default, effectively disabling the timer), giving flexibility to when the GPIO1 and GPIO2 pins are driven upon an initialization fault, thereby opening the battery contactors in a standard application.

After completing the above procedures, the MAX17851 continuously uses ALERTPACKETs to poll the BMS Data Acquisition System for faults using the thresholds defined in the configuration memory. This operation occurs until a valid exit criteria is observed as depicted by Figure 7.

When NOMON register bit is 1, the MAX17851 does not monitor the BMS Data Acquisition System devices after the GPIO validation and system recovery attempt. Instead, it starts the FAULT_TIMER, asserts all GPIO pins according to their configuration, and waits for a valid watchdog refresh as depicted by Figure 8.

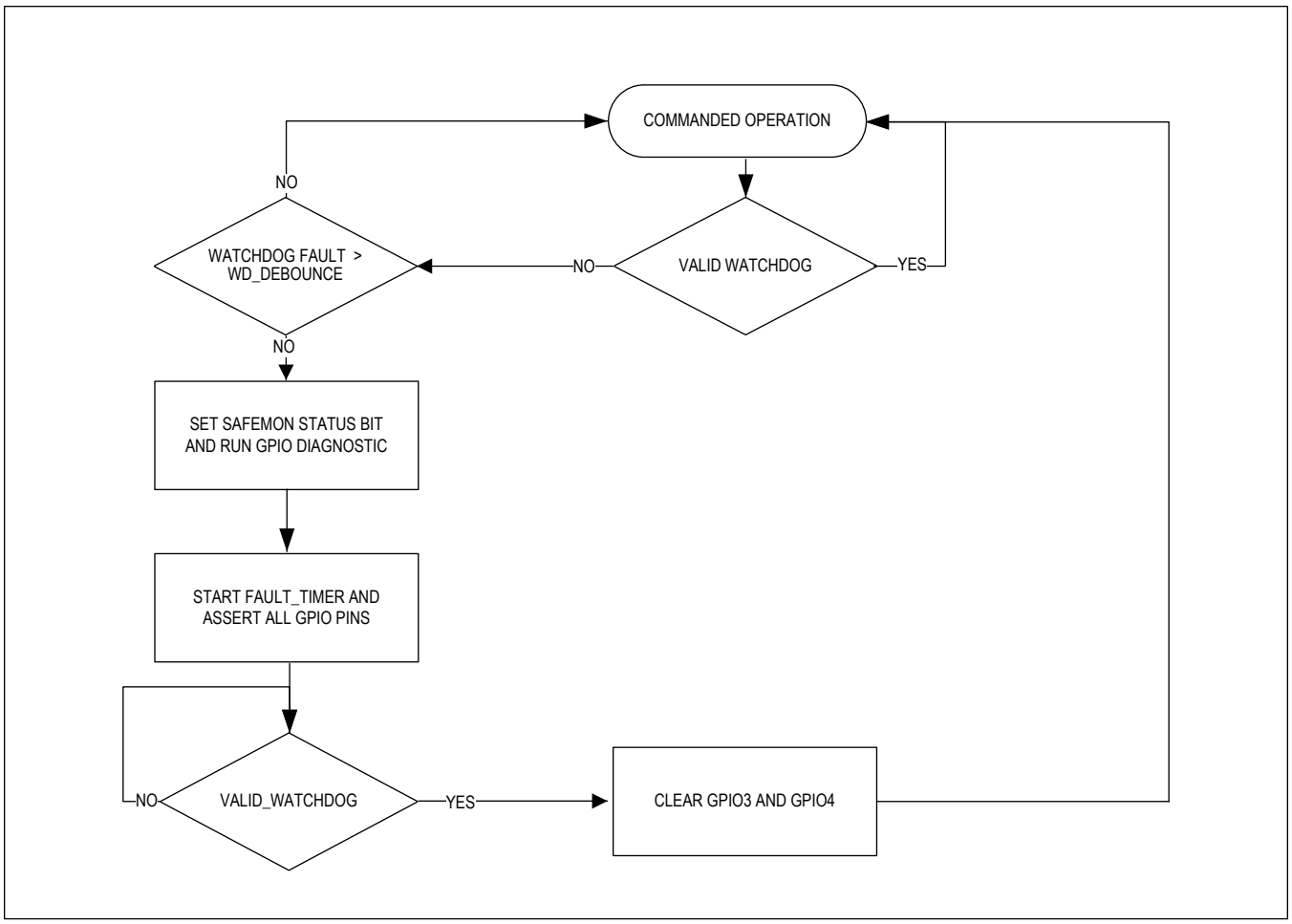


Figure 8. BMS Safe Monitoring (NOMON) Mode Flow Chart

Fault Polling With ALERTPACKET

After successful diagnostics and daisy-chain configuration, ALERTPACKETs are generated based on the SAFEMON_SCAN_DLY interval. The ALERTPACKET data payload is stored in the Alert Packet buffer, then masked and parsed to identify any user specified fault. Only if an unmasked fault persists for the number of times programmed into the SAFEMON_ALRT_DBNC register does the SAFEMON_ERR status bit set and the GPIO1 and GPIO2 pins assert.

SAFEMON_ERR = |(ALERTPACKET.STATUS[n] & ~ALRT_STATUS_MASK[n]) (Debounced SAFEMON_ALRT_DBNC Times)

Note: The BMS daisy-chain monitoring devices may have polling intervals at different rates than the MAX17851 (as set by SAFEMON_SCANDLY). This may result in repetitive fault conditions based on the same prior sample. The application should ensure that the SAFEMON_SCANDLY value is greater than the measurement delay of the BMS monitoring device to ensure unique samples.

If the ALERTPACKET command is not received during the fault polling process, a COMM_TO_ERR fault is issued and continual retries persist until the COMM_RTRY count is exceeded. If the Alert Packet buffer hardware fails integrity checks at any time during BMS Safe Monitoring mode, the ALRTPCKTBUF_HW_ERR fault is asserted. If either an ALRTPCKTBUF_HW_ERR or communications fault occurs during BMS Safe Monitoring mode, the GPIO3 pin asserts, the CONT_TIMER starts, and, upon expiration, the GPIO1 and GPIO2 pins assert.

Exiting BMS Safe Monitoring Mode

Exiting BMS Safe Monitoring mode is achieved by writing a valid watchdog response to WD_KEY. Exit occurs once a safe monitoring cycle is complete (i.e. after the response to a generated ALERTPACKET is received).

Exiting does not clear SAFEMON_ALRT bit, and the $\overline{\text{ALERT}}$ pin remains asserted until the the SAFEMON_ALRT is cleared. If the $\overline{\text{ALERT}}$ pin persists, then the alert status must be queried to determine the status of the MAX17851 and the BMS daisy-chain. If SAFEMON_ALRT is cleared while the device is still in BMS Safe Monitoring operation, the SAFEMON_ALRT instantly re-asserted. Only after successfully exiting the BMS Safe Monitoring mode can SAFEMON_ALRT be cleared.

If GPIO1/2 are actively driven when BMS Safe Monitoring is exited, the pins remain asserted until the SAFEMON_GPIO12_ERR bit is cleared.

If GPIO3/4 are actively driven when BMS Safe Monitoring is exited, the pins are deasserted.

If the master dual UART MAX17851 is unresponsive due to a broken SPI trace, BMS Safe Monitoring mode must be exited on the slave dual UART MAX17851 by programming its MS_EN bit to master single UART and refreshing its watchdog. In this case, the previously programmed slave dual UART becomes the master single UART. In this scenario, the uC should re-initialize the BMS devices to their normal operating state using the newly configured master single UART MAX17851. Note that RX_SWAP_EN should remain unchanged to reflect the daisy-chain path (see *Slave in Commanded Operation Mode*).

Note: The MAX17851 may also exit BMS Safe Monitoring mode by resetting the device (SWPOR = 1). If SWPOR is used as an exit condition, the MAX17851 assumes its default state, and the register programming resets and must be re-initialized via SPI for the application.

Configuration Memory

The status of the BMS daisy-chain can potentially be in a different configuration state than desired (dependent upon when in the BMS software implementation the microcontroller becomes unresponsive). Additionally, the system may desire to change safety critical thresholds due to an unknown microcontroller fault. The configuration memory stores the desired UART command set to be executed in BMS Safety Monitoring mode ensuring all daisy-chain devices are in a known state. After the BMS Data Acquisition System is configured with these settings, it auto polls these parameters periodically for system safety as the host microcontroller is unresponsive. In case of a system fault and depending on the user defined critical threshold/debouncing, the MAX17851 places the system in safe state through GPIO signaling to the host microcontroller and/or contactors.

The configuration memory consists of three queues of 30 data bytes each and a fourth queue of 6 data bytes. The first three queues are sub-divided into 7 data blocks, a PEC byte that protects the queue, and a reserved byte. The fourth queue consists of a single data block with the same structure. Each of the configuration memory data blocks consist of 2 separate register address bytes and 2 data bytes which are used to write a 16-bit data payload to the BMS daisy chain units. The data must first be stored as LSB and subsequently, MSB.

An address value of 0xFF is a null address and signifies no operation for the load configuration memory and configuration memory verification state machines. Two consecutive null addresses in the same block signify the end of device configuration for that data queue. By default, all bytes are reset to 0xFF.

The PEC byte protects all bytes of a queue prior to two consecutive null addresses. If two consecutive null addresses are not encountered within the queue, the PEC byte protects the entire queue. Single null addresses are skipped over and not included in the PEC byte calculation. Queues that begin double null addresses are not processed by the load configuration memory or verify configuration memory processes.

The CONFIG memory can be accessed by using the CONFIGQ, CONFIG_QUEUE_PTR, and CONFIG_BYTE_PTR registers. See the *MAX17851 User Register Map* for details.

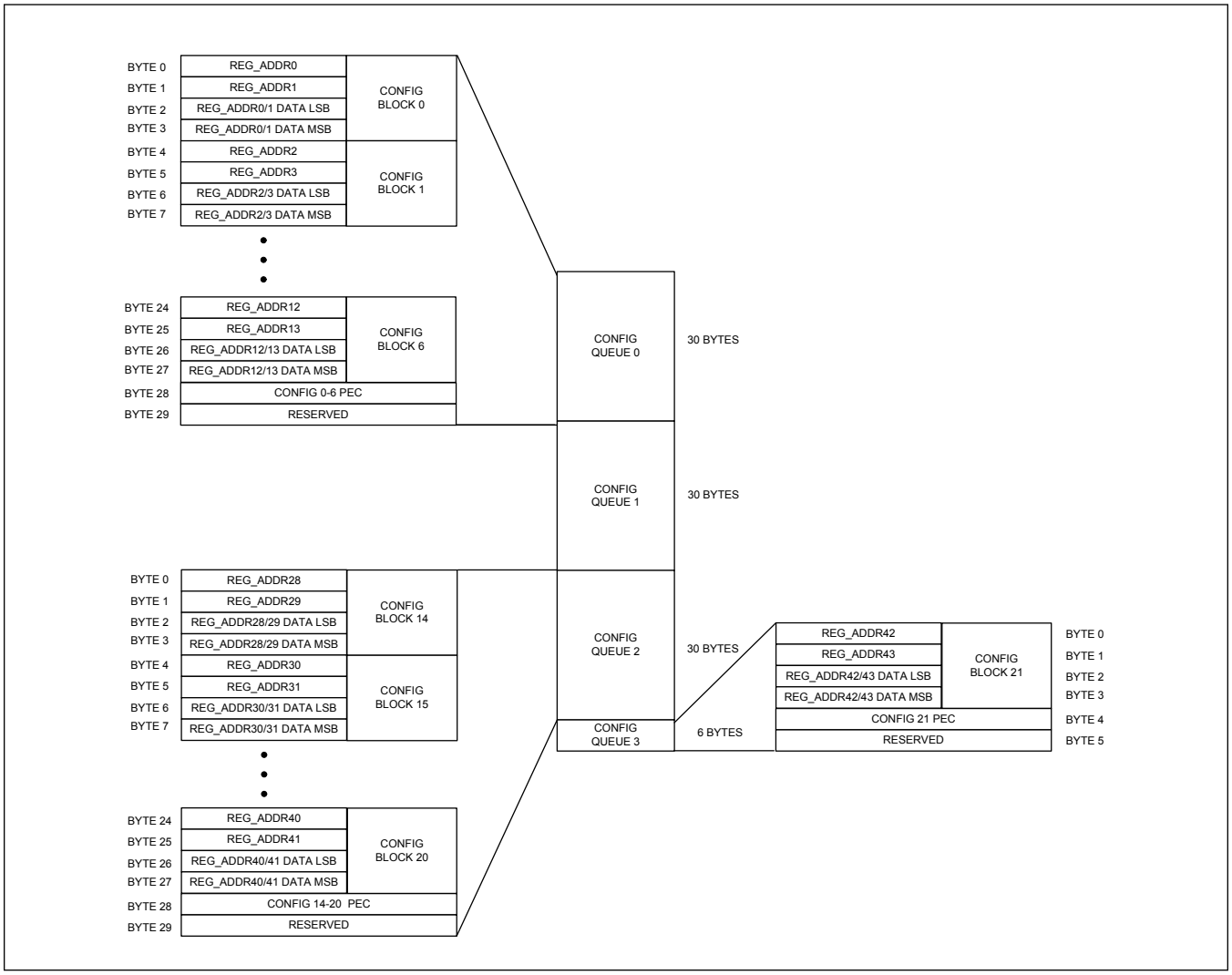


Figure 9. Configuration Memory

Datapath and Configuration Memory Verification

Prior to configuration memory formatting and transmission as UART WRITEALL packets, the content of the configuration memory, as well as the functionality of the UART datapath, PEC encoders, and PEC decoders are verified using an internal loopback circuit. During this process, Alert Packet generation (if enabled) is suspended and Keep-Alive generation is enabled. Keep-Alive characters are generated between the verification process of each Configuration Memory byte. This ensures that the BMS daisy-chain does not inadvertently shutdown during this verification process.

A fault condition during this process is indicated by the DATAPATH_ERR and SAFEMON_CONFIG_ERR_ALRT bits. For master only or slave configurations, the MAX17851 asserts GPIO3, initiates the FAULT_TIMER, initiates the CONT_TIMER, and then asserts GPIO1 and GPIO2 as defined by the GPIO_CFG bitfields. For master with slave configurations, the MAX17851 asserts GPIO3, initiates the FAULT_TIMER, and waits for a watchdog refresh.

Verification of the of the configuration memory and UART datapath may be performed outside of BMS Safe Monitoring by issuing a VER_CONFIG command. The bit auto-clears when the operation is complete, and the results are indicated by the status bit described above. See the *Hardware-In-The-Loop (HIL) Testing Using the TX_AUTO Feature* section for

more information on testing using the internal loopback to test the internal hardware of the MAX17851.

The execution time of the datapath and configuration memory verification diagnostic (t_{VER_CONFIG}) varies with the configured baud rate and the amount of data stored in the configuration memory.

Table 2. Datapath and Configuration Memory Execution Time

Configuration Memory Banks Populated	t_{VER_CONFIG}
1	30us + 72*t_{BIT}
2	60us+ 72*t_{BIT}
3	90us+ 72*t_{BIT}
4	120us+ 72*t_{BIT}

Configuration Memory Transmission

Each CONFIG data block is sequentially formatted into a WRITEALL message, loaded into the Transmit buffer, and transmitted to the UART daisy-chain devices. Refer to Figure 10 for detailed diagram of the data transmission flow chart.

The data transmission occurs twice for each of the register addresses (REG_ADDR1 and REG_ADDR2) within the CONFIG data block. This allows for the same data bytes to be sent to two different register addresses, or the same data to be sent to the same register address for transmission redundancy.

Table 3. WRITEALL Command Sequencing From Configuration Memory

MAX17851 TX	MAX17851 RX
Preamble	Preamble
02h	02h
[REG_ADDR1/2]	[REG_ADDR1/2]
[DATA LSB]	[DATA LSB]
[DATA MSB]	[DATA MSB]
[PEC]	[PEC]
[ALIVE]*	[ALIVE]*
Stop	Stop

* If the automated alive-counter is enabled

The PEC for the issued Configuration Memory command is automatically calculated by the MAX17851. If enabled, an Alive Counter is also issued according to the default alive counter seed value or the rolling alive counter.

Before the next register address (REG_ADDR2) or next CONFIG data block can be transmitted, the communication packet is verified to be properly executed. If an error exists within the data packet integrity or within data packet timing, a fault is stored in the COMM_ERR or COMM_MSMTCH_ERR status bits. Alternatively, if the configuration memory data transmission is not received after a duration programmed by COMM_TO_DLY, a fault is stored in the SAFEMON_CONFIG_ERR bit. Failing data packets are re-transmitted up to the value programmed in the COMM_RTRY register. If the COMM_RTRY count is exceeded, the SAFEMON_CONFIG_ERR bit asserts. For master only or slave configurations, the MAX17851 asserts GPIO3, initiates the FAULT_TIMER, initiates the CONT_TIMER, and then asserts GPIO1 and GPIO2 as defined by the GPIO_CFG bit fields. For master with slave configurations, the MAX17851 asserts GPIO3, initiates the FAULT_TIMER, and wait for a watchdog refresh.

For further details on the communication safety validation, please refer to the *Lockstep Safety Measure Verification* section.

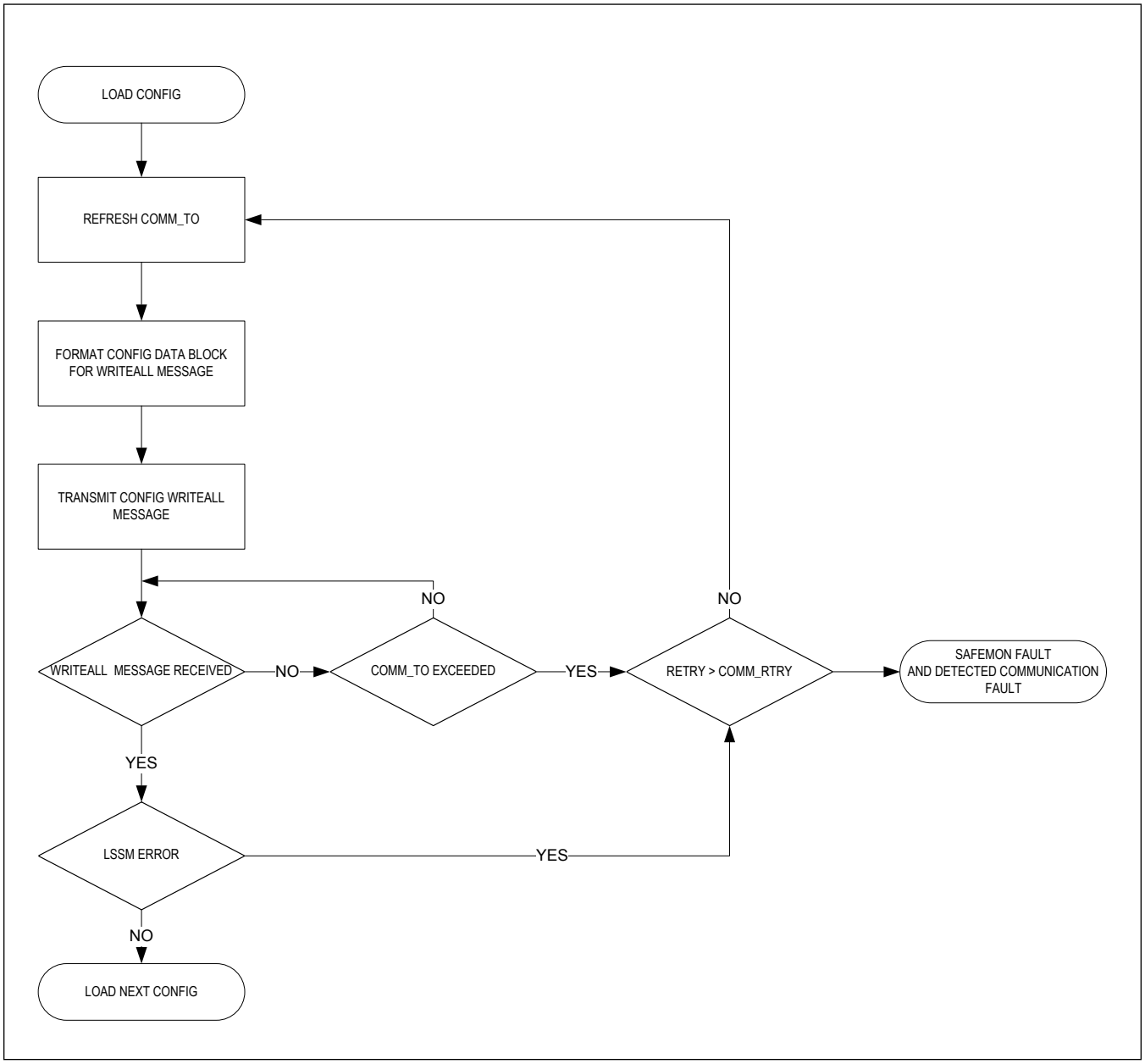


Figure 10. Configuration Memory Data Transmission

Configuration Memory Setup

The configuration memory must be programmed such that the MAX17851 can:

1. Re-initialize daisy-chain device configuration.
2. Initiate auto-polling for all daisy chain devices.

Daisy-chain device configuration is required since a failure causing a watchdog alert may occur during any operational cycle of the BMS operation. Thus, the daisy-chain may be in an undesired configuration, such as a diagnostic cycle or partial measurement cycle. Leaving the daisy-chain in a potentially undesired condition may result in the MAX17851

inadvertently opening the system contactors.

The re-initialization of the daisy-chain devices should include modifying cell measurement configurations (enabled cell, polarity), temperature measurement configurations (enabled auxiliary inputs, auxiliary scan timing, auxiliary ratio-metric sources), diagnostic configuration and measurements (enabled current sources, current source levels, embedded scan diagnostic), as well as scan control (scan type, digital noise filter settings and control).

See *Configuration Memory Sequence for Daisy Chain Configuration* for examples on configuration.

Master/Slave Device Configuration

The MS_EN bit dictates the behavior of the MAX17851 watchdog and BMS Safe Monitoring operation as a master only, master with slave, or slave device in the application circuit.

Master in BMS Safe Monitoring Mode

MAX17851 master configuration (MS_EN = 1X) requires that the watchdog be serviced to prevent BMS Safe Monitoring mode entry.

GPIO4 is asserted upon exiting the safety measure state associated with GPIOREC_DLY timeout. See the *GPIO Control* section for additional details. The assertion of GPIO4 can be used to attempt recovery of the microcontroller or power supply.

GPIO3 will be asserted upon identification of data corruption, internal processing error, or a fault preventing UART communication. The assertion of GPIO3 can be used to transfer control of BMS Safe Monitoring operation to the slave device in a dual UART application.

Additional information and application diagrams are available in the Applications Information sections, *Single UART Operation* and *Dual UART Operation*

Slave in BMS Safe Monitoring Mode

When a MAX17851 device is configured as a slave (MS_EN = 0x), watchdog functionality is disabled regardless of WD_EN configuration, and all watchdog alerts are masked. As such, the slave cannot enter BMS Safe Monitoring mode as a result of loss of SPI communication or watchdog timeout.

The slave device only enters BMS Safe Monitoring mode through the assertion of a logic high to its GPIO4 pin (assuming it is pre-configured as a SAFEMON slave input). Once the slave device has entered BMS Safe Monitoring mode, it issues a DOWNHOST command and begin BMS Safe Monitoring mode. This provides redundancy in the event of a master device communications failure.

GPIO3 is asserted upon identification of data corruption, internal processing error, or a fault preventing UART communication. The assertion of GPIO3 can be used to attempt recovery of the microcontroller or power supply.

The slave device can exit BMS Safe Monitoring mode in two ways; first, by refreshing the watchdog of the master device, which causes the master GPIO3 output to de-assert. Secondly, it can exit by issuing a SWPOR command to the slave (this defaults to master single UART) and re-configuring the daisy-chain. The second scenario is only required if SPI communication to the master is non-operational and its watchdog cannot be refreshed.

A slave configuration is intended only for a dual UART configuration. See the applications *Dual UART Operation* section for additional detail.

Slave in Commanded Operation Mode

The data entering the RX buffer from the UART down path is in reverse order from the UART up path. RXSWAP_EN allows for the down path data to be ordered the same as the up path data to ease processing by the system microcontroller.

Note: In Commanded Operation mode, the MAX17851 device connected to the down path should be a slave device issuing only read commands.

Note: RXSWAP_EN should be programmed to 1 for the slave device connected to the physical down path. However, if the read/write functionality of the up path and the down path needs to be changed, the host microcontroller can issue a DOWNHOST or UPHOST, but, RXSWAP_EN should remain unchanged, since it is configured to the physical path. See [[READALL Command Sequencing In Dual-UART down path (z = Number of Devices)]] for an example.

Table 4. READALL Command Sequencing In Dual-UART Down Path (z = Number of Devices)

MAX17851 UART TX	MAX17851 UART RX	MAX17851 BUFFER STORAGE (RXSWAP = 0)	MAX17851 BUFFER STORAGE (RXSWAP = 1)
Preamble	Preamble		
03h	03h	03h	03h
[REG ADDR]	[REG ADDR]	[REG ADDR]	[REG ADDR]
[DC] = 0x00	[DATA LSB(z)] = [DATA LSB(BA)]	[DATA LSB(z)] = [DATA LSB(BA)]	[DATA LSB(1)] = [DATA LSB(TA)]
[PEC]	[DATA MSB(z)] = [DATA MSB(BA)]	[DATA MSB(z)] = [DATA MSB(BA)]	[DATA MSB(1)] =[DATA MSB(TA)]
[ALIVE]*	[DATA LSB(z-1)] = [DATA LSB(BA +1)]	[DATA LSB(z-1)] = [DATA LSB(BA +1)]	[DATA LSB(2)] = [DATA LSB(TA-1)]
[FD(1) C2h]	[DATA MSB(z-1)] = [DATA MSB(BA +1)]	[DATA MSB(z-1)] = [DATA MSB(BA +1)]	[DATA MSB(2)] =[DATA MSB(TA-1)]
[FD(1) D3h]	...	...	...
[FD(2) C2h]	...	...	...
[FD(2) D3h]	...	...	...
...	...	...	...
...	...	...	[DATA LSB(z-1)] = [DATA LSB(BA +1)]
...	...	...	[DATA MSB(z-1)] = [DATA MSB(BA +1)]
...	[DATA LSB(1)] = [DATA LSB(TA)]	[DATA LSB(1)] = [DATA LSB(TA)]	[DATA LSB(z)] = [DATA LSB(BA)]
...	[DATA MSB(1)] =[DATA MSB(TA)]	[DATA MSB(1)] =[DATA MSB(TA)]	[DATA MSB(z)] = [DATA MSB(BA)]
...	[DC]	[DC]	[DC]
[FD(z) C2h]	[PEC]	[PEC]	[PEC]
[FD(z) D3h]	[ALIVE]*	[ALIVE]*	[ALIVE]*
Stop	Stop	Stop	Stop
12+(4 x z) characters	12+(4 x z) characters	6+(2 x z) bytes	6+(6 x z) bytes

*If Alive Counter mode is enabled

The fill byte values transmitted by the MAX17851 interface alternate between C2h and D3h as shown.

Slave in Sleep Mode

A slave MAX17851 should be commanded to enter Sleep mode while the master MAX17851 is in Sleep mode to prevent competing system interrupts. However, this is not prohibited.

For additional master/slave configuration information, see the Applications Information section *Dual UART Operation*.

Diagnostic and Operational Verification

The MAX17851 performs two diagnostics to ensure the correct operational state of the MAX17851:

- Datapath and configuration memory diagnostic
- WD/GPIO diagnostic

The datapath and configuration memory diagnostic performs a complete configuration memory and UART data path check to verify the UART encoders and decoders, receive PEC decoder, transmit PEC encoder, and UART TX/RX PHY

prior to the pins. In case of a hardware or configuration memory/PEC error, the MAX17851 flags a DATAPATH_ERR bit in the STATUS_GEN register, indicating a UART datapath error. This diagnostic can be entered manually in Commanded Operation mode by writing the VER_CONFIG bit to 1. The diagnostic cannot be manually run when the device is in Sleep mode or BMS Safe Monitoring mode.

The WD/GPIO diagnostic verifies the correct functionality of the watchdog and the GPIO. This diagnostic occurs when BMS Safe Monitoring mode is entered due to a watchdog timeout. The host should perform a GPIO status register read at this time. The diagnostic is exited when a valid watchdog command is written. If the diagnostic is not exited prior to the expiration of GPIOREC_DLY, the MAX17851 proceeds according to the state of the NOMON bit. See the *Entering BMS Safe Monitoring Mode* section for further details.

Watchdog

The MAX17851 watchdog feature is used to verify host microcontroller operation by communication via the SPI interface at a fixed interval. If the host microcontroller becomes unresponsive (i.e., the watchdog times out), a counter is incremented (*WD_DBNC bits, CONFIG_WD2[3:0]*). This counter is programmable from 1 to 128. When the WD_DBNC value is met, the MAX17851 sets the WD_TO_ERR bit in the STATUS_WD register and enters BMS Safe Monitoring mode.

Note: Entry into Sleep mode disables the watchdog, which prevents the device from transitioning to BMS Safe Monitoring mode while in Sleep mode. Therefore, the watchdog does not need to be refreshed in Sleep mode.

The MAX17851 watchdog functionality is enabled using the WD_EN bit in the WD_CONFIG2 register. If the WD_EN bit is de-asserted (or the SWPOR bit is asserted), the watchdog is disabled, and the device is unable to enter BMS Safe Monitoring mode. If the MAX17851 is in BMS Safe Monitoring mode and the SWPOR bit is asserted, the device exits BMS Safe Monitoring mode and enters Commanded Operation mode with default register states.

The watchdog time base is configured using the WD_DIV bits in the *CONFIG_WD1* register. The watchdog clock time is equal to the WD_DIV value, plus 1, multiplied by 256µs (t_{WDCLK} = (WD_DIV[4:0]+1) * 256µs).

The watchdog requires that the host microcontroller send a valid WD_KEY register write periodically. The response must be written after the t_{WD1} time expires and before the t_{WD2} time expires.

Calculation of the t_{WD1} and t_{WD2} times is as follows:

$t_{WD1} = t_{WDCLK} * (WD\_OPN[3:0]+1)*8$

$t_{WD2} = t_{WD1} + t_{WDCLK} * (WD\_CLO[3:0]+1)*8$

where t_{WD1} is the closed window period, t_{WD2} is the closed window plus open window period (max), and t_{WDCLK} *is the watchdog clock period

When initially enabled, the watchdog enters the extended open window. This first open window is extended to allow the host microcontroller enough time to initialize. The extended time is programmed by the WD_1UD bitfield in the CONFIG_WD1 register and is defined as:

t_{WD2} * (WD_1UD[2:0]+1). This time is added to t_{WD2} only when the watchdog is initially enabled.

The response must be written after t_{WD1} time expires and before t_{WD2} time expires. Any valid watchdog refresh terminates the open window, initializes the closed window, and updates the WD_KEY register.

Note: Programming watchdog configuration registers while the watchdog is enabled is not recommended and may cause unexpected operation.

When the WD_FAULT_CNT is equal to the value programmed in WD_DBNC, the MAX17851 will enter BMS Safe Monitoring mode.

If a valid WD_KEY write is sent while in BMS Safe Monitoring mode, the MAX17851 returns to Commanded Operation mode, and the watchdog enters the extended open window and resumes normal operation.

The number of watchdog violations that accumulate in the WD_FAULT_CNT register is reset when any valid watchdog refresh is received.

The WD_LFSR_ERR bit in the STATUS_WD register is set when an invalid watchdog key is written while the watchdog is in Challenge Response mode. A WD_RJCT_ERR (STATUS_WD register) is set when a watchdog key is written during a closed window period while in Challenge Response mode. These bits are cleared when WD_EN is set to 0.

Failure to write a valid WD_KEY within the open window causes the watchdog to timeout and WD_FAULT_CNT to be incremented. The WD_EXP_ERR bit (STATUS_WD register) is set each time the watchdog times out. This bit is cleared when WD_EN is set to 0.

The WD_TO_ERR bit (STATUS_WD register) is set when the WD_FAULT_CNT is equal to the programmed WD_DBNC value. This bit is cleared when WD_EN is set to 0.

If WD_ERR_ALRTEN bit is set (default), the WD_ERR_ALRT bit (ALERT_GEN register) bit is set, and the $\overline{\text{ALERT}}$ pin is asserted in the event of a watchdog error. The WD_ERR_ALRT bit needs to be cleared before the $\overline{\text{ALERT}}$ pin is de-asserted. Watchdog errors that can cause the WD_ERR_ALRT bit to be asserted are WD_TO_ERR, WD_LFSR_ERR, WD_RJCT_ERR, and WD_EXP_ERR.

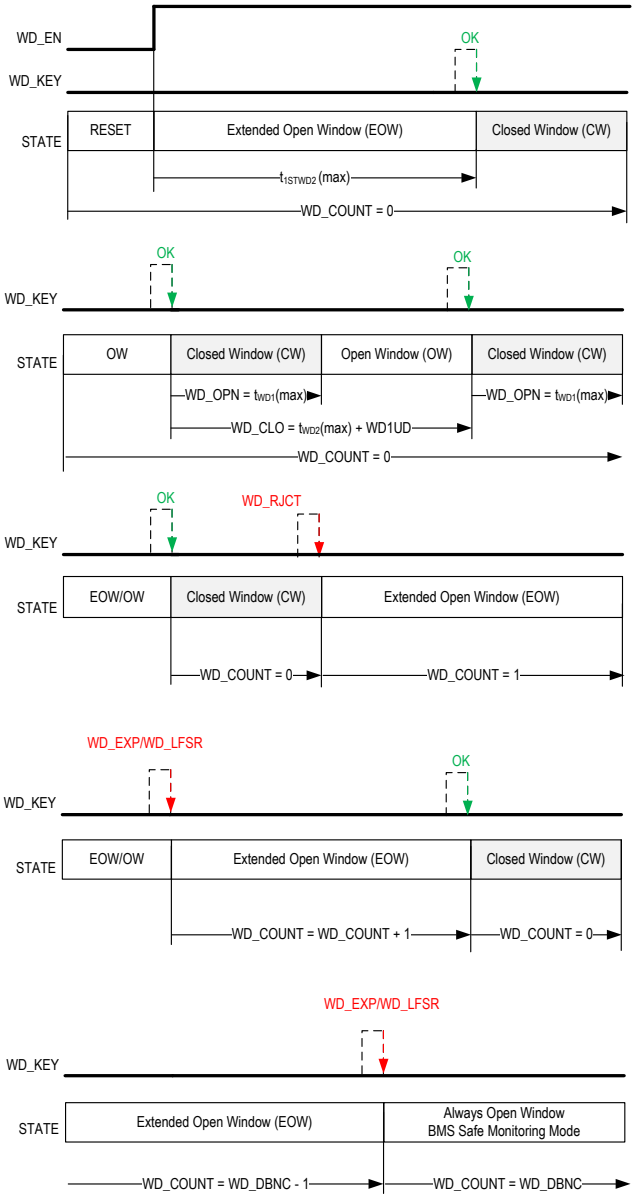


Figure 11. Watchdog Operation

Challenge/Response Mode

The challenge response configuration of the watchdog (WD_SWW =0) requires that the data written to the WD_KEY register is calculated using a linear feedback shift register (also referred to as LFSR) algorithm to calculate the required write response to the current key (see *Sample LFSR/CRC code* section for more information).

The host controller can read the current key at any time from the WD_KEY register.

Simple Windowed Mode

The simple windowed watchdog (WD_SWW = 1) allows for any value written to the WD_KEY to provide a valid watchdog

refresh signal. The value written to the register is ignored.

Sample LFSR/CRC Code

```
// feedback polynomial: x^8 + x^6 + x^3 + x^2 + 1
uint8 lfsr(uint8 iKey)
{
   lfsr = iKey;
   uint8 bit = ((lfsr >> 7) ^ (lfsr >> 5) ^ (lfsr >> 2) ^ (lfsr >> 1) ) & 1;
   lfsr = (lfsr << 1) | bit;
   return lfsr;
}
```

GPIO Control

The functionality of the GPIO is determined by the programming of the individual GPIO[n]_CFG registers.

GPIO pins may be configured as a General Purpose I/O, commanded by explicit register writes, but can also be individually configured to provide specific functionality within BMS Safe Monitoring mode (SAFEMON).

For a single UART configuration (MS_EN = 0b10), GPIO1 and GPIO2 are intended to be configured as a static output (HI or LO) for logic signaling to the battery contactor relay driver interface. GPIO3 and GPIO4 are intended to be configured to reset the microcontroller or microcontroller power supply network in case of an unresponsive condition. This reset function may be configured differently due to the specific device requirements, but in general, it is in the form of a one-shot negative pulse (1-Shot LO, default). The one-shot occurs for a duration of 100ms such that a reset may be applied and released.

For a dual UART configuration (MS_EN = 0b11), the master device passes control to the slave device as described in *Entering BMS Safe Monitoring Mode*. In this configuration, the master should configure GPIO3 as SAFEMON Active HI and GPIO4 may be optionally used to drive the microcontroller or microcontroller power supply network. The slave should configure GPIO4 as SAFEMON Slave Input and GPIO3 may optionally be used to drive the microcontroller or microcontroller power supply network, which is not controlled by the master. This allows for handshaking between the GPIO3 master and GPIO4 slave.

If the general purpose inputs, general purpose HI, or general purpose LO is selected in either master/single UART or master/slave dual UART, these will remain as programmed when the device enters Safe Monitoring mode or Sleep mode.

If GPIO3 and GPIO4 are configured as a SAFEMON output, they idle in their inactive state when in Commanded Operation or Sleep mode.

If GPIO1 and GPIO2 are configured as a SAFEMON output, they idle in their inactive state when in Commanded Operation or Sleep mode unless the SAFEMON_GPIO12_ALRT bit is set.

Reserved GPIO pin configurations idle as high impedance inputs.

Note: When idling as a high impedance input, GPIO pins have a pulldown resistance $R_{GPIO,}$ which causes the pins to present as ground.

Table 5. GPIO Control

	INPUT		OUTPUT					
GPIO1	General Purpose Input **(Default)**		General Purpose HI	General Purpose LO			SAFEMON Active HI	SAFEMON Active LO
GPIO2	General Purpose Input **(Default)**		General Purpose HI	General Purpose LO			SAFEMON Active HI	SAFEMON Active LO
GPIO3	General Purpose Input **(Default)**		General Purpose HI	General Purpose LO	SAFEMON 1-Shot HI	SAFEMON 1-Shot LO	SAFEMON Active HI	SAFEMON Active LO

Table 5. GPIO Control (continued)

GPIO4	General Purpose Input **(Default)**	SAFEMON Slave Input	General Purpose HI	General Purpose LO	SAFEMON 1-Shot HI	SAFEMON 1-Shot LO	SAFEMON Active HI	SAFEMON Active LO

The GPIO[n]_RD bits monitor the pin logic level, regardless of whether the port is defined as an input or an output. If contention exits on the GPIO where the read level is not the same as the programmed output then the GPIO[n]_ERR status bit is asserted.

Note: GPIO[n]_ERR only applies to GPIO output configurations.

Serial Peripheral Interface (SPI)

The SPI port is a synchronous data link that the host uses to read and write the MAX17851 registers and the UART communication buffers.

SPI Transactions

A SPI transaction is initiated when the host drives the $\overline{CS}$ pin low. The host always transmits data's most-significant bit (MSB) first to the MAX17851. After the first byte, it can terminate the transaction (single-byte command transaction), continue to clock data out (write transaction), or start clocking data in (read transaction). However, it does not send and receive data at the same time (half-duplex operation).

For all transactions, the host first sends a 7-bit register address. Reads and writes are delineated by the 8th bit (R/Wb bit). For a read transaction, the second byte is the read data sent by the MAX17851 to the host. For a write transaction, the second byte is the write data sent by the host to the MAX17851.

Reading or writing multiple data bytes (burst mode operation) is allowed as long as $\overline{CS}$ remains active-low. The MAX17851 automatically increments the next read-only register address or the next write-only address. Automatic address increment is disabled for specific addresses to prevent incoherent burst mode access.

Automatic address increment is disabled for the following UART buffer register addresses:

- RX_RD_MSG
- RX_RD_NXT_MSG
- ALRTPCKTBUF_RD_MSG
- NXT_LDQ
- LDQ
- CONFIGQ

When these registers are accessed with a burst mode transaction, the internal SPI address remains static, and the associated buffer pointer for that address is incremented instead.

Automatic address increment is also disabled for the following command register addresses:

- CLR_TXBUF
- CLR_RXBUF
- CLR_LSSM
- CLR_ALIVECOUNT_SEED
- SWPOR
- SLP_EN
- VER_CONFIG
- LOAD_CONFIG

The SPI transaction is terminated when the host drives CS high.

Note: When a burst mode transaction includes a register whose automatic address increment operation is disabled, the automatic address increment stops at the address prior to that register.

See the *Transmit Buffer* section for more detailed information about the transmit queues.

See the *Receive Buffer* section for more detailed information about the receive queues.

Table 6. SPI Register Summary

ADDRESS	NAME	DESCRIPTION
0x00 to 0x0F	Status Registers (See Register Table)	Current status of MAX17851. Registers are read only.
0x10 to 0x1F	Alert Registers (See Register Table)	Current status of MAX17851 Alert Registers. Status can be read or cleared (write 0) only. A write of 1 to the register has no effect.
0x20 to 0x2F	Alert Enable Registers (See Register Table)	Current status of MAX17851 Alert Enable Registers. Status can be read or write.
TX BUFFER REGISTERS		
0x40	CLR_TXBUF	Command: Resets the Transmit buffer to its default state and clears TX_Q and LD_Q.
0xB0	NXT_LDQ	Command: Increments the LDQ and then writes the Transmit buffer load queue.
0xC0	LDQ	Command: Reads/Writes the Transmit buffer load queue.
0xC2	LDQ_PTR	The location in the Transmit LDQ buffer that the host reads or writes.
0xD0	CONFIGQ	Command: Reads/writes the Configuration Data Queue byte as pointed to by the CONFIG_BYTE_PTR and CONFIG_QUEUE_PTR registers.
0xD2	CONFIG_PTR	The location of the CONFIG_QUEUE_PTR and the CONFIG_BYTE_PTR.
RX BUFFER REGISTERS		
0x42	CLR_RXBUF	Command: Resets the Receive buffer and the Receive buffer pointers to their default state.
0x8C	ALRTPCKTBUF_RD_PTR	The location of the Read Pointer for the Alert Packet buffer.
0x8E	ALRTPCKTBUF_RD_PTR	Command: Reads the Receive buffer starting at the address ALRTPCKTBUF_RD_PTR. Automatically increments the read pointer after the byte is read but does not increment the read pointer into the next message.
0x90	RX_RD_MSG	Command: Reads the Receive buffer starting at the address RX_RD_ PTR. Automatically increments the read pointer after the byte is read but does not increment the read pointer into the next message.
0x92	RX_RD_NXT_MSG	Command: Reads the Receive buffer starting at the address RX_NXT_MSG_ PTR (oldest unread message). Automatically increments the read pointer after the byte is read but does not increment the read pointer into the next message.
0x96	RX_RD_PTR	The location in the Receive buffer that the host is to read. The UART automatically increments this pointer.
0x98	RX_WR_PTR	The location in the Receive buffer that is written by the UART as it receives data.
0x9A	RX_NXT_MSG_PTR	The start of the next unread message in the Receive buffer. The RX_RD_Pointer is loaded with this value by the RD_NXT_MSG SPI transaction.
0x9C	RX_SPACE	Number of available bytes in the Receive buffer.
COMMAND REGISTERS		
0x44	CLR_LSSM	Command: Resets the LSSM.
0x48	CLR_ALIVECOUNT_SEED	Command: Clears the ALIVECOUNT_SEED register.
0x4A	SWPOR	Command: Software Power On Reset.
0x4C	SLP_EN	Command: Enable Sleep mode.
0x4E	VER_CONFIG	Command: Verify Configuration memory.
0x50	LOAD_CONFIG	Command: Load Configuration memory.
0x52	WD_KEY	Command: Reads/writes the value of the watchdog key.

SPI Timing

The MAX17851 is only compatible with SPI mode 0 (CPOL = 0/CPHA = 0). In this mode, data is always driven out on the falling edge of SCLK and is sampled in on the rising edge of SCLK. For reads, the MAX17851 starts driving digital output (DOUT) on the first falling edge of SCLK following the R/Wb bit. Data input (DIN) has no effect while driving DOUT data

during a read. Reads attempted beyond the address space return zero.

For writes, registers are written on the falling edge of SCLK after the last bit is sampled. However, if $\overline{CS}$ goes high before the last bit's falling edge of SCLK, that register is not written.

Table 7. SPI Communication Summary

PARAMETER	VALUE
Communication Mode	Half-duplex
Maximum Clock Frequency	10 MHz
Bit Order	Most-significant bit first
Clock Polarity (CPOL)	0 (leading clock edge is rising edge)
Clock Phase (CPHA)	0 (data sampled on leading clock edge)

SPI Read and Write Restrictions

SPI reads and writes are unrestricted in Commanded Operation mode. However, during specific modes of operation, writes and reads are restricted on some registers to prevent conflicting states within the MAX17851.

In the specified modes, disabled register writes are ignored and cause the SPI_ERR_ALRT bit to be set. Disabled register reads return zeros and cause the SPI_ERR_ALRT bit to be set.

Table 8. SPI Read and Write Restrictions

REGISTER	COMMANDED OPERATION	SLEEP MODE	SAFETY MONITORING MODE	LOAD CONFIG	VERIFY CONFIG
STATUS Registers	Unrestricted	Unrestricted	Unrestricted	Unrestricted	Unrestricted
ALERT Registers	Unrestricted	As Defined	As Defined	As Defined	As Defined
ALERTEN Registers	Unrestricted	WR Disabled	WR Disabled	WR Disabled	WR Disabled
COMMAND CLR_TXBUF	Unrestricted	WR Disabled	WR Disabled	WR Disabled	WR Disabled
COMMAND CLR_RXBUF	Unrestricted	WR Disabled	WR Disabled	WR Disabled	WR Disabled
COMMAND CLR_LSSM	Unrestricted	WR Disabled	WR Disabled	WR Disabled	WR Disabled
COMMAND CLR_ALIVECOUNT_SEED	Unrestricted	WR Disabled	WR Disabled	WR Disabled	WR Disabled
COMMAND SWPOR	Unrestricted	Unrestricted	Unrestricted	Unrestricted	Unrestricted
COMMAND SLP_EN	Unrestricted	Unrestricted	WR Disabled	WR Disabled	WR Disabled
COMMAND VER_CONFIG	Unrestricted	WR Disabled	WR Disabled	WR Disabled	Unrestricted
COMMAND LOAD_CONFIG	Unrestricted	WR Disabled	WR Disabled	Unrestricted	WR Disabled
COMMAND WD_KEY	Unrestricted	Unrestricted	Unrestricted	Unrestricted	Unrestricted
CONFIG Registers	Unrestricted	WR Disabled	WR Disabled	WR Disabled	WR Disabled
RX_COMMAND Registers	Unrestricted	Unrestricted	Unrestricted	Unrestricted	Unrestricted
TX_COMMAND Registers	Unrestricted	RD/WR Disabled	RD/WR Disabled	RD/WR Disabled	RD/WR Disabled
INFO Registers	Unrestricted	Unrestricted	Unrestricted	Unrestricted	Unrestricted

UART Interface

Slave devices that use Maxim's battery management UART protocol can be connected in daisy-chain fashion to manage a multiple battery-cell stack. In a BMS, the controller is the host for all slave devices and initiates all communication. The data flow always starts from the host, moves up the daisy-chain and back down to the host, as represented in Figure 12.

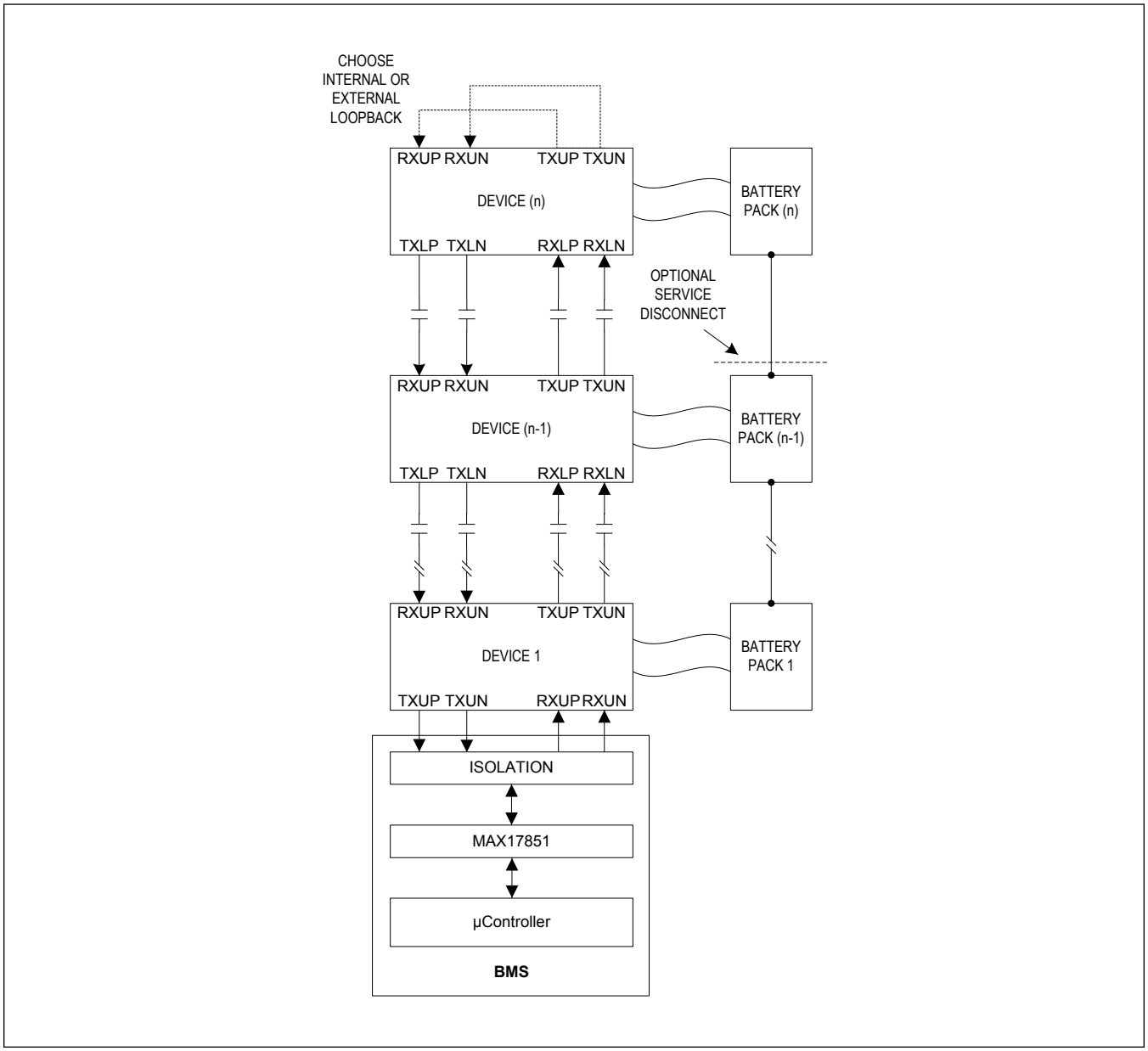


Figure 12. System Data Flow

Battery Management UART Protocol

The MAX17851 uses a UART protocol specifically designed for Maxim battery management devices. This protocol uses the following features to maximize the integrity of the communications:

- All transmitted data are Manchester-encoded (G.E. Thomas convention) where each data bit is transmitted twice with the second bit inverted.
- Every transmitted character contains 12 bits that include a START bit, a parity bit, and two STOP bits.
- Each message is framed by a preamble character and an optional stop character.
- Each received read command message (READALL, READBLOCK, READDEVICE) may optionally contain a data-

check byte for verifying the integrity of the transmission.
- Each READALL, READBLOCK, READDEVICE, WRITEALL, WRITEDEVICE, ALERTPACKET message is protected by a CRC-8 packet error-checking PEC byte.
- Each READALL, READBLOCK, READDEVICE, WRITEALL, WRITEDEVICE, ALERTPACKET message is optionally followed by an Alive Count byte.

The protocol is also designed to minimize power consumption by allowing slave devices to shut down if the data link is idle for a specified period of time. To prevent the unintentional shutdown of slave devices, the host should enable the MAX17851's Keep-Alive mode to periodically transmit Keep-Alive characters. The time period between Keep-Alive characters is configurable by the host.

UART Messages

A message is defined as a sequence of UART characters. The message starts with a preamble character, followed by data characters, and ending with a stop character. Each character consists of the following 12 bits:

- One START bit
- Eight data bits (LSB first)
- One parity bit (even)
- Two STOP bits

Each data byte is transmitted and received as two separate characters with one 12-bit character for each 4-bit data nibble. Each Manchester-encoded nibble requires 8 data bits: 4 true bits and 4 inverted bits. In its default configuration, when the MAX17851 transmits a message, it automatically performs the following functions:

- Frames the message with the required preamble character at the beginning of the message.
- Manchester encodes each data nibble and transmits each encoded nibble with the required START, parity, and STOP bits.
- Transmits the message at the configured baud rate of 0.5Mbps, 1Mbps, 2Mbps, or 4Mbps.
- Frames the message with the required stop character at the end of the message.

These automatic functions can be disabled by enabling the following special transmit modes:

- Transmit No Preamble mode (eliminates preamble characters)
- Transmit No Stop mode (eliminates stop characters)
- Transmit Raw Data mode (transmits data with no Manchester encoding)
- Receive Raw Data mode (receives data as not Manchester-encoded)

Preamble Character

The preamble is a framing character that the UART generates to signal the beginning of a message. It is transmitted as an unencoded 15h, but is still a DC-balanced character. If any bit(s) other than the STOP bits deviate from the unique preamble sequence, the character is not interpreted as a valid preamble, but rather as a data character.

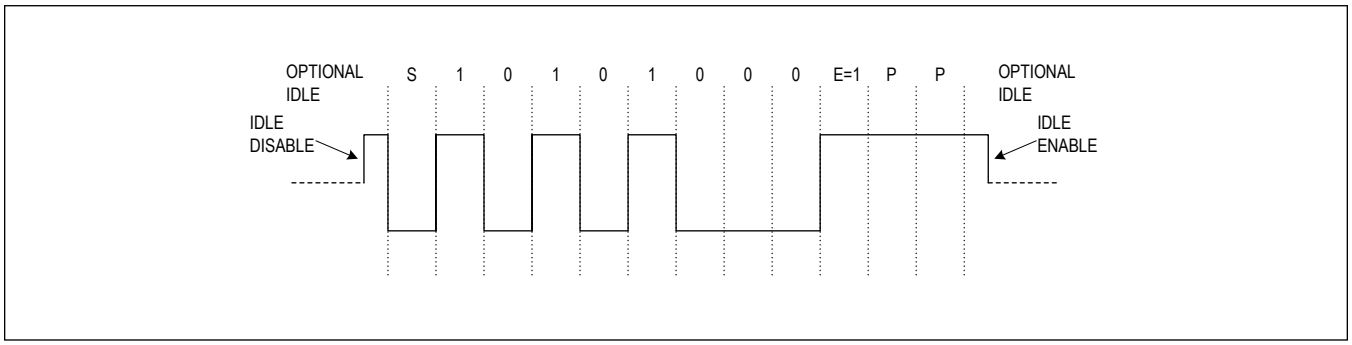


Figure 13. UART Timing for a Preamble

Stop Character

The stop character is a framing character that the UART generates to signal the end of a message. It is transmitted as an unencoded 54h, but it is still a DC-balanced character.

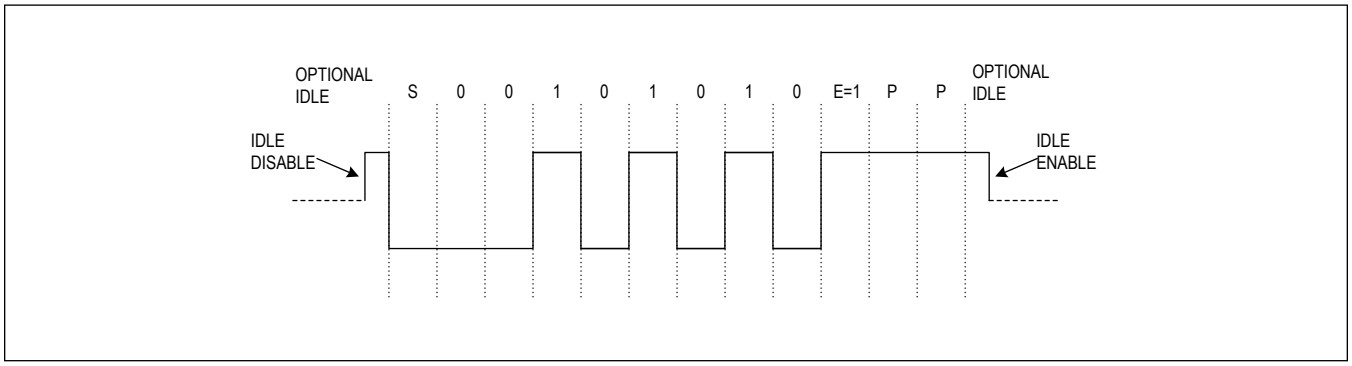


Figure 14. UART Timing for a Stop Character

Manchester Encoding

Each data byte is transmitted as two separate nibbles (four bits) that are Manchester-encoded. For each data bit, the first bit represents the information, and the second bit is its complement. The parity is even so its value should always result in an even number of high bits. Since the data is Manchester-encoded and there are two STOP bits, the parity bit for data characters (but not framing characters) should always be zero.

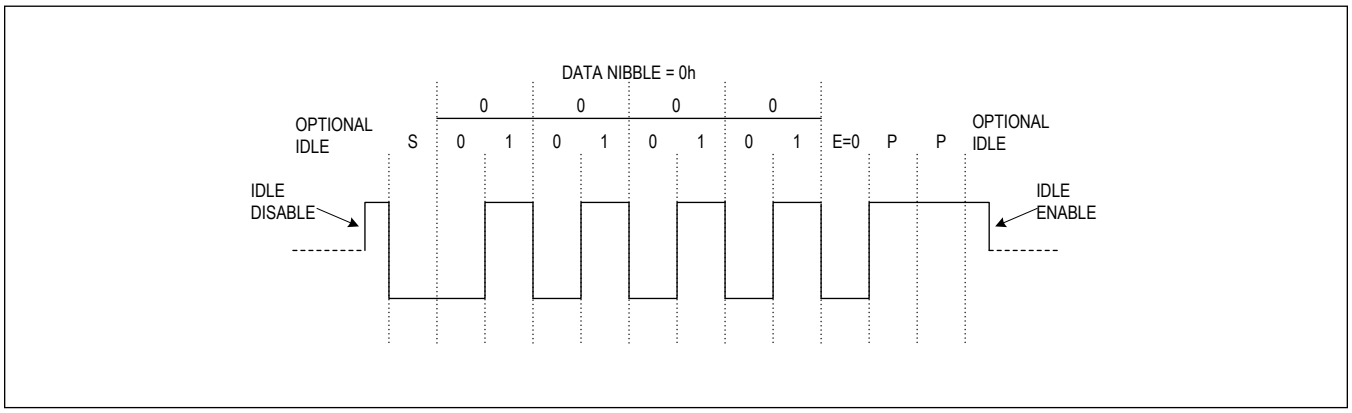


Figure 15. UART Timing for a Manchester-Encoded Data Nibble 0h

Data Types

Maxim's battery management UART protocol supports several different data types as described in Table 9. It is up to the host to both compose the data being transmitted and interpret the data being received. For example, the host must compute the proper PEC value for each transmitted message and must verify the PEC value on each received message.

Assigning Slave Device Addresses

The battery management UART protocol requires the host to assign a unique and contiguous address between 0 and 31 to each UART slave device, as desired. The host performs this assignment by specifying a seed address in the HELLOALL command sequence. As the command propagates up the daisy-chain, each slave device assigns its own address. The HELLOALL sequence returns a value from which the host can determine the number of devices in the daisy-chain, as well as the device addresses.

UART Message Data Types

Table 9. UART Message Data Types

DATA TYPE	DESCRIPTION
Command	Defines the type of message, either a write command or a read command.
Address	Register address to be read or written.
Data	Register data being read or written.
Fill	Bytes with values C2h or D3h transmitted as a part of read commands so that the total number of bytes sent equals the number of bytes received. However, these bytes are not returned to the receiver with their original values; instead, each slave device replaces the fill bytes with the register data requested by the host.
Data-Check	Error status provided by the slave devices. Returned only on reads.
PEC	CRC-8 packet error-checking byte. Sent and returned with every message.
Alive-Counter	Used to verify the number of devices responding to a transmitted message. This byte is optional but is recommended for error-checking purposes.

Common UART Commands

Table 10. Common UART Commands

COMMAND BYTE	VALUE	DESCRIPTION
HELLOALL	57h	Writes a unique device address to each device in the daisy-chain. Required for system initialization.
ALERTPACKET	21h	Reads the alert packet from all devices on the daisy-chain.
WRITEDEVICE	{(DA[4:0]),0b100}	Writes a specified register to a single-device.
WRITEALL	02h	Writes a specified register in all devices.
READDEVICE	{DA[4:0], 0b101}	Reads a specified register from a single device.
READALL	03h	Reads a specific register from all devices.
READBLOCK	{BS[4:0], 0b110}	Reads a set of registers from a single device.
DOWNHOST	09h	Configures devices on the daisy-chain for write access on the DOWN path.
UPHOST	08h	Configures devices on the daisy-chain for write access on the UP path.
Reserved	{_,0b0000}, {_,0b1010}, {_,0b1011}, {_,0b1111}	Reserved. **Note:** Reserved command bytes are recognized as valid by the Lockstep Safety Measure feature and is treated similar to a READ operation.

UART Operation

The UART is the subsystem that transmits messages to the UART slave devices and receives them back. The host uses SPI buffer transactions to store unencoded outgoing messages in the Transmit buffer and to read decoded incoming messages out of the Receive buffer as shown in Figure 16. UART Buffers shows the size and organization of the UART buffers.

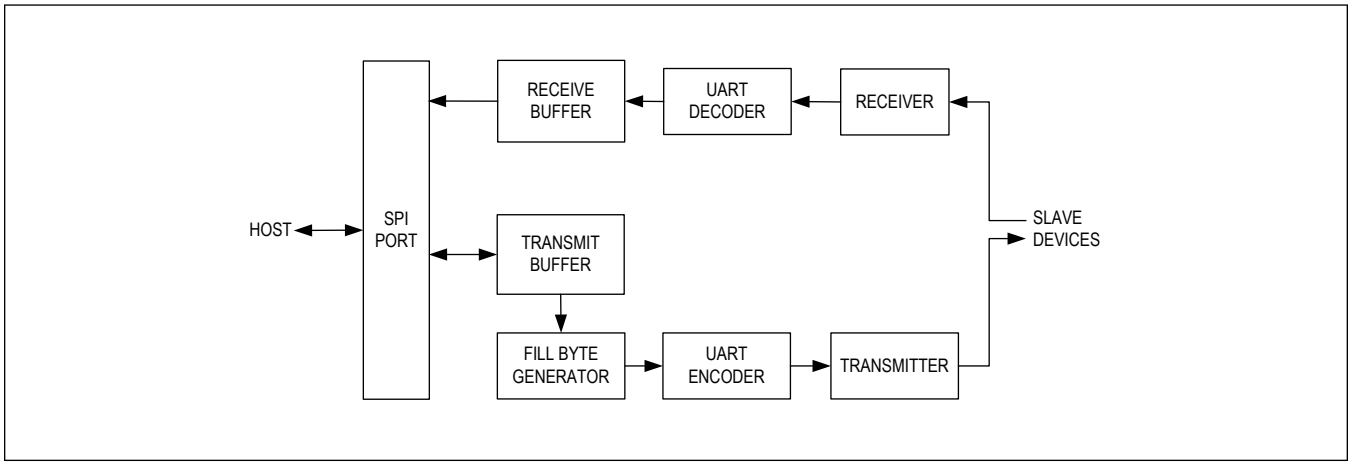


Figure 16. UART Data Flow

UART Operational Modes

Table 11. UART Operational Modes

MODE	CONFIG_GEN2 OR CONFIG_GEN3 REGISTER BIT	DESCRIPTION
Transmit Preambles	TX_PREAMBLES (CONFIG_GEN2)	Transmits preambles continuously (no idle state). Used to wake up the UART slave devices and initialize the UART baud rate of each slave device. This mode takes precedence over all transmit modes except Transmit Pause mode.
Keep-Alive	ALRTPCKT_TIMING (CONFIG_GEN3)	Periodically sends a stop character to prevent UART slave devices from shutting down during periods of no communication (idle state). The idle time in between the periodic stop characters is programmable from zero to 10.24ms through the ALRTPCKT_TIMING[3:0] configuration. The default setting is infinite (mode disabled). The Transmit Pause, Transmit Preambles, and the Transmit Queue modes take precedence over this mode. Note that Keep-Alive (stop characters) are sent when CO_ALRTPCKTEN = 0. When CO_ALRTPCKTEN = 1, alert packets are sent. See the *Alert Packet Buffer* section for more information.
Transmit Queue (default mode)	TX_QUEUE (CONFIG_GEN2)	Starts transmission of the message loaded in the transmit queue if 1) there is sufficient space in the Receive buffer for the message (RX_FULL status is false) or 2) the limitations on message length are removed (TX_Unlimited is set). Default is enabled.
Transmit Unlimited	TX_UNLIMITED (CONFIG_GEN3)	In this mode, the transmit queue automatically limits the message length to 255 bytes instead of the default 86-byte limit. The message transmission is permitted even if the message length is greater than the available write space in the Receive buffer. Note that LSSM errors may occur in this mode since the LSSM buffer may be overflowed. The LSSM should be cleared after the message(s) are sent and regular messages are resumed.
Transmit Pause	TX_PAUSE (CONFIG_GEN2)	Places the transmitter into idle state once the UART is finished transmitting the current byte; however, the TX_BUSY and TX_IDLE status bits remain unchanged. Transmission resumes when this bit is cleared. This mode takes precedence over all other transmit modes.
Transmit Odd Parity	TX_ODDPARITY (CONFIG_GEN2)	Transmits characters with odd parity. Since the battery management UART protocol uses even parity, this mode can be used to test the system's ability to detect parity errors. Even parity is default.
Transmit No Stop	TX_NOSTOP (CONFIG_GEN2)	Transmits messages without a stop character.

Table 11. UART Operational Modes (continued)

MODE	CONFIG_GEN2 OR CONFIG_GEN3 REGISTER BIT	DESCRIPTION
Transmit No Preamble	TX_NOPREAMBLE (CONFIG_GEN2)	Transmits messages without a preamble.
Transmit Raw Data	TX_RAW_DATA (CONFIG_GEN2)	Disables Manchester encoding of transmitted data. In this mode, each data byte is transmitted as one character instead of two characters.
Receive Raw Data	RX_RAW_DATA (CONFIG_GEN2)	Disables Manchester decoding of the received data. In this mode, there is one data byte stored for every character received instead of every two received.

Transmit Buffer

Figure 17 shows the Transmit buffer memory map. It consists of four fixed-length queues, which the host uses to store outgoing messages. At any time, one of the queues is designated as the load queue (i.e., the queue being loaded) and one as the transmit queue (i.e., the queue being unloaded). The load queue is selected by the two-bit register LD_Q, and the transmit queue is selected by the two-bit register TX_Q. Each queue consists of thirty-one bytes.

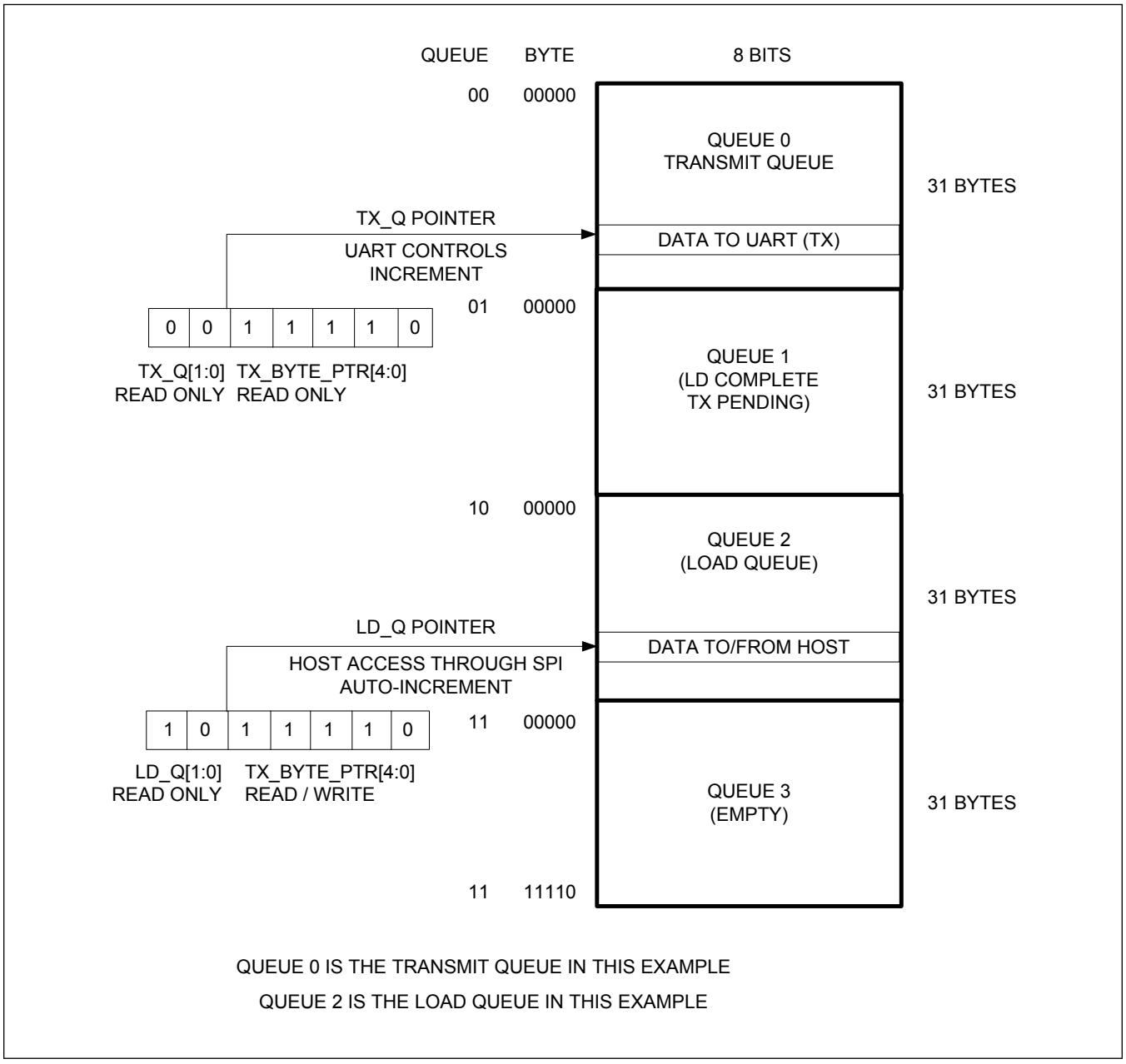


Figure 17. Transmit Buffer Memory Map

Transmit Buffer Queues

In each queue, location zero is reserved for the message length, and the remaining thirty-one locations are for specific message data. The default state of each queue is as shown in Table 12.

Clearing the Transmit Buffer

During UART initialization, it is recommended that the host reset the Transmit buffer by issuing the CLR_TXBUF SPI command. This resets the Transmit buffer as follows:

- TX_Q [1:0] = 00b
- LD_Q [1:0] = 00b
- All data in Transmit buffer is reset to default state per Table 12.

Message Length

Before composing any message, the host should compute the message's length (in bytes, not characters) based on both the type of command (read or write) and the device count. The message length should include any required fill bytes, but not preamble and stop characters. The host writes the message length into location 0 of the load queue, but if the specified message length is greater than 86d, only 86d (56h) is written. If the TX_UNLIMTED = 1, then the maximum message length written is increased to 255d (FFh), but the host must service the Receive buffer accordingly to avoid any possible overflow.

If the specified message length is greater than 30 bytes of user data, the UART automatically appends alternating fill bytes (D3h, C2h) as required by the battery management UART protocol during the latter portion of the message transmission.

If the message length does not correspond to the correct length as dictated by the daisy-chain device count and transaction type, then the LSSM indicates multiple status errors. This requirement ensures that any errors propagated from the host, within the device, or within the daisy-chain are detected according to ISO-26262 ASIL-D criteria.

Table 12. Queue Memory Map

LOCATION	DESCRIPTION	DEFAULT VALUE	MAXIMUM DEFAULT PERMITTED	
			TX_UNLIMITED = 0	TX_UNLIMITED = 1
0	Message length	00h	56h	FFh
1	Data bytes and/or fill bytes	D3h	FFh	FFh
2		C2h		
3		D3h		
4		C2h		
5		D3h		
6		C2h		
7		D3h		
8		C2h		
9..29		C2h..D3h		
30		C2h		

Writing the Load Queue

A message consists of 3 to 31 bytes, not including fill bytes. The HELLOALL sequence, for example, is three bytes: 57h, 00h, 00h (first address set to zero). Since no fill bytes are required, the total message length is 3 bytes. Therefore, the host should write the load queue with the data listed in Table 13.

The host can write the load queue starting at any location within the queue by using appropriate SPI commands listed in the *SPI Transactions* section. However, if the host attempts to write beyond location 31 of the queue, the additional data is ignored.

The UART never attempts to transmit the queue selected by LD_Q, because the host may be in the process of loading it or may need to verify (read) the contents of the load queue, even if it has finished loading. The host can then select the next queue in sequence for loading by writing the NXT_LD_Q command which increments the LD_Q value. It is only when this increment occurs that the UART starts transmitting the data in the previously loaded queue. For both LD_Q[1:0] and TX_Q[1:0], values of 3h increment to 0h.

Table 13. Example of Queue Loaded with Message HELLOALL

LOCATION	VALUE	DESCRIPTION
0	03h	Message length
1	57h	Command byte
2	00h	Address byte
3	00h	Data byte
4	C2h	Not written
5	D3h	Not written
6..29	...	Not written
30	C2h	Not written
31	D3h	Not written

Filling the Transmit Buffer

The host can load all available queues until LD_Q = TX_Q - 1. In this state, the Transmit buffer is full (TX_FULL is true). In this condition, the host cannot start loading the transmit queue because the UART may still be unloading/ transmitting data. If the Transmit buffer is full and the host attempts to write a NXT_LD_Q command and attempts to load the transmit queue, the increment does not occur, and an overflow condition is indicated (TX_OVERFLOW status true). The host cannot write the transmit queue is when the Transmit buffer is full.

Message Transmission

Whenever there are no unsent transactions, the Transmit buffer is considered empty (TX_EMPTY is true). Once the host is finished servicing the queue, it performs a WR_NXT_LD_Q transaction to send the previously loaded message and select the next queue.

The UART unloads/transmits the transmit queue via the WR_NXT_LD_Q command if the following conditions are met:

- The UART is in Transmit_Queue mode (TX_QUEUE bit is set)
- The Transmit buffer has at least one loaded queue (TX_EMPTY is false)
- There is sufficient space in the Receive buffer for the message (RX_SPACE ≥ Message Length)
- There is sufficient space in the LSSM buffer for the message (LSSM_FULL != 1)

Note: The limitation on available space in the Receive buffer can be removed by setting the TX_UNLIMITED bit.

Note: In the event that there is conflict between the Transmit buffer and auto-generated ALERTPACKET commands, the Transmit buffer is serviced first.

Once the transmit conditions are met, the UART automatically starts unloading the transmit queue until the entire message, including any required fill bytes, is transmitted. After the transmission is complete, the contents of the transmit queue are reset to their default values, and the queue is once again available to the host for loading.

Receive Buffer

The Receive buffer is an 86-byte circular buffer that the host can read with the SPI but can only be loaded by the UART as it receives data. It utilizes three pointers, as shown in the Receive buffer memory map (Figure 18).

- RX_RD_PTR: Read pointer or buffer location to be read by host (default 00h, read-only)
- RX_WR_PTR: Write pointer or buffer location to be written by UART (default 01h, read-only)
- RX_NXT_MSG_PTR: Buffer location that is the start of the next unread message (default 00h, read-only)

In the default state, where the read pointer is one less than the write pointer, the Receive buffer is considered empty (RX_EMPTY is true). Any Receive buffer data read in this condition is zero.

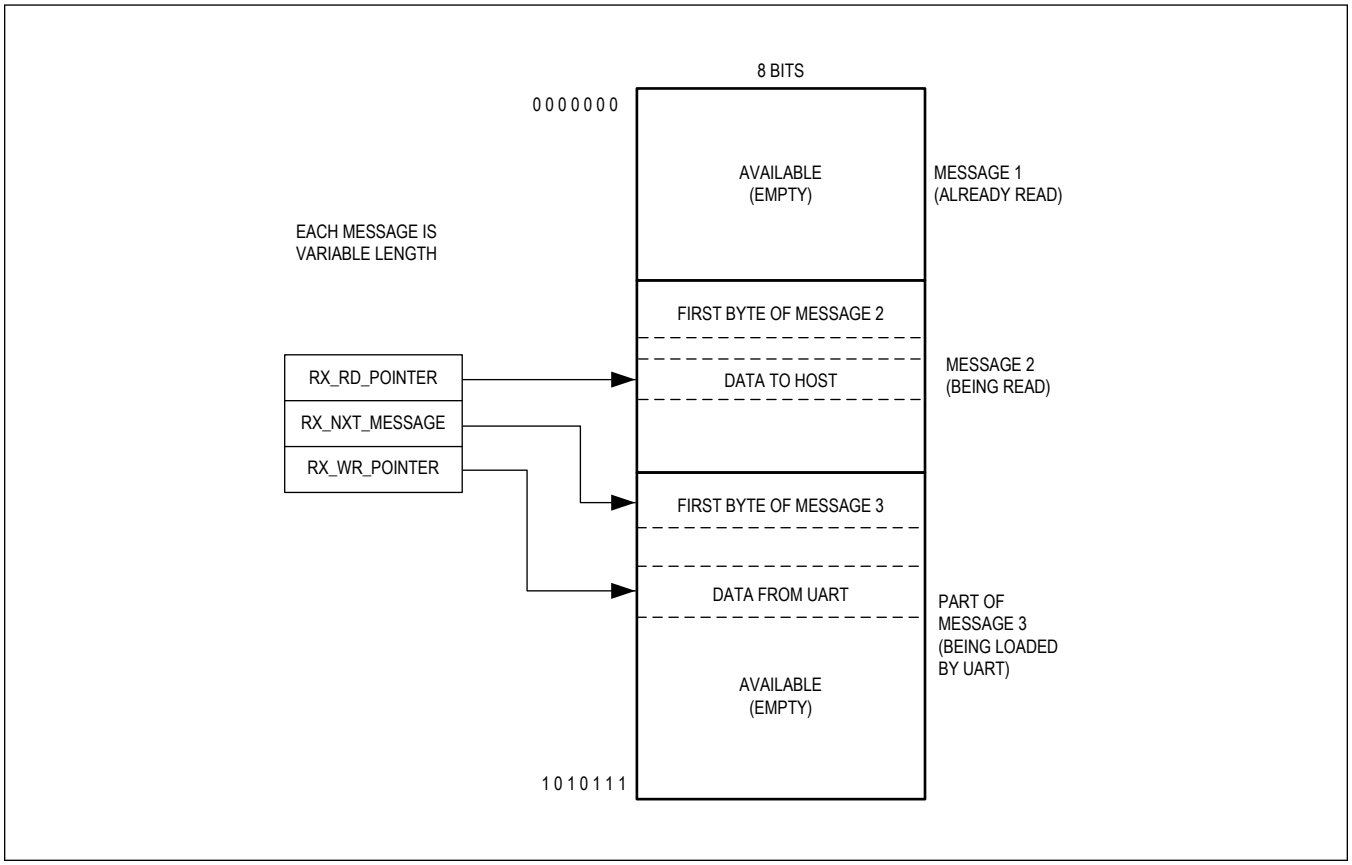


Figure 18. Receive Buffer Memory Map

Clearing the Receive Buffer

During UART initialization, it is recommended that the host clear the Receive buffer by issuing the CLR_RXBUF SPI command. This resets the Receive buffer as follows:

- RX_RD_Pointer: 00h
- RX_WR_Pointer: 01h
- RX_NXT_MSG_POINTER: 00h
- Data in Receive Buffer (86 bytes) is cleared to 00h

If the Receive buffer is cleared during Transmit Preambles mode, the state of the buffer cannot be guaranteed. Therefore, after disabling the Transmit Preambles mode, the host should wait until all transmitted preambles are received before clearing the buffer.

Since the first keep-alive stop character received after the last preamble results in a null message, the host can simply wait until the buffer is no longer empty (RX_EMPTY is false) before clearing the buffer. The RX Clear buffer command acts an asynchronous reset, not only for the RX buffer, but also for the UART receiver logic. When the UART receiver logic is reset, it must resynchronize to the incoming UART signal before bytes can be properly processed. This is accomplished by receiving either a preamble byte or an idle state lasting at least one UART byte period.

The preamble used to resynchronize the data stream should not be the same preamble that is at the beginning of the next transmitted message. The application should make sure that one of these conditions is met following an RX buffer clear prior to sending the next message from the MAX17851.

Receiving Messages

UART messages are framed with the preamble and stop characters. If the UART receiver decodes a valid preamble, it prepares to receive a message, but it does not store the preamble in the Receive buffer. Once data is received, the buffer is no longer empty (RX_EMPTY = 0), and the UART sequentially stores decoded data bytes in the Receive buffer until either a stop character is received, message frame timeout is exceeded (t_{FRMTO}), or another preamble is received. When the end of a message is detected, the UART stores a null byte (00h) in the Receive buffer and sets the RX_READY bit. The RX_READY bit is subsequently cleared when all unread messages have been read (buffer empty) or the next preamble is detected.

Note: RX_READY indicates that a message has been received and is ready to be read back from the RX buffer. When Stop characters are enabled (TX_STOP = 1), the RX_STOP status bit indicates a properly framed message has been received. If RX_READY is set and RX_STOP is not, this indicates a corruption has occurred within the Stop character. A corrupted Stop character does not necessarily indicate corruption within the data payload. In this scenario, the integrity of the data payload should be verified using the LSSM_BYTE.

It is recommended that the host configures the MAX17851 so that the ALERTB pin (RX_READY_ALRTEN = 1) is asserted when a message is received. This provides a clear indication of when the Receive buffer can be serviced. Do not service the buffer prior to the receipt of RX_READY_ALRT.

Multiple messages may be stored in the Receive buffer to improve the processing efficiency of the host microcontroller. Clear the RX_READY_ALRT prior to receiving the next message to ensure successful communication of the completed message through the ALERTB pin message. If the RX_READY_ALRT is not cleared, a persistent alert is communicated.

When the host services the Receive buffer, three bits in the RX_BYTE register indicate specific information about the byte being read (the byte addressed by RX_RD_PTR), which is useful for error checking:

- RX_FIRST_BYTE bit: Indicates the byte is the first data byte in a message. The corresponding character is preceded by preamble character.
- RX_BYTE_ERROR bit: Indicates the byte may contain an error. The corresponding character contains a Manchester and/or parity error.
- RX_LAST_BYTE bit: Indicates the byte is the last byte in a message. The corresponding character is a stop character and was stored as a null byte

Message Exceptions

If a message is not received with a valid preamble, the UART ignores the data and does not store it.

If a message is not terminated with a stop character, then the preamble of the next message (or a receive data timeout) delineates the two messages, resulting in the first message being complete.

A receive data timeout occurs when the LSSM has received no data within two UART frame lengths of the last data byte. This results in the LSSM status byte being written, the RX_READY bit being asserted, and the message being completed.

If the UART receives a preamble followed by a stop character, it stores a null message in the Receive buffer consisting of a single null byte (00h). This occurs when a keep-alive stop character is received after Transmit Preamble mode is disabled. In this use case, the Receive buffer is not empty. The host should dispense the null message by either clearing the Receive buffer or reading the null message and discarding it.

A communication timeout error (COMM_TO_ERR) is defined as the time from when it is sent from the TX port to when a preamble is received by the RX port.

A Receive buffer overflow occurs when the UART receives data but there is no more space to store it. This can potentially occur if TX_UNLIMITED is set or if there is sufficient latency in the daisy-chain. The UART cannot overtake the read pointer and overwrite the data being read; therefore, the last address that can be written is the one following the read pointer. If more data is received after the last address is written, the UART overwrites the last address and sets the RX_OVERFLOW bit, which is cleared when the Receive buffer is read. This creates more write space. To detect any overflow, the status must be checked before servicing the Receive buffer. Afterwards, the status should be checked again for data errors (e.g., parity errors) prior to initiating transmission of the next message.

If multiple messages are received without being read, then an overflow can occur, and the UART sets the RX_OVERFLOW bit. This occurs when the write pointer has incremented until it is one less than the read pointer, at

which point the UART no longer increments it. In this case, the last data byte is overwritten.

Reading Messages

The host can use two different SPI transactions to read the Receive (RX) buffer:

- RX_RD_NXT_MSG: Starts reading at the RX_RD_PTR location (the next unread message).
- RX_RD_MSG: Starts reading at the RX_RD_PTR location.

During any read transaction, the host may continue reading data until the end of the message, after which the data read is 0x00. The host cannot continue reading into the next message, if there is one.

During a RX_RD_NXT_MSG or RX_RD_MSG transaction, the device automatically increments the associated pointers; therefore, multiple SPI transactions can be used to read a given message. For example, if RX_RD_NXT_MSG is used to read back a message, and the transaction is terminated prior to the end of the message. Then, a subsequent RD_RX_MSG can be used to resume readback of that same message. This allows the host to stop a read and restart readback without losing data. After a byte in the buffer is read, it becomes available to the microcontroller unit (MCU) for storing new incoming data.

Note: If the SWAP_EN bit set is of command type READALL, then the MAX17851 automatically reorders the message data using the DEVCOUNT register. In such a scenario, the original PEC received is replaced with a recalculated packet-error checking (PEC) to account for the message data reordering.

Note: Partial-byte SPI access is not supported and may result in unexpected behavior. However, if this occurs during a RD_RX_NXT_MSG or RD_RX_MSG, the MAX17851 attempts to restore the pointer of the partially read byte to prevent data loss. This allows the message readback to be resumed with a subsequent RD_RX_MSG SPI transaction.

Alert Packet Buffer

ALERTPACKET commands are parsed from the received UART data and stored in the two (main and redundant) 6-byte ALERTPACKET buffers. These buffers store the BMS daisy-chain module fault location, and BMS daisy-chain STATUS1 register indication. ALERTPACKET data can only be read from the main ALERTPACKET buffer.

A PEC error within the ALERTPACKET data results in the data being discarded and not stored in the ALRTPCKTBUF. This prevents the host microcontroller from being consistently interrupted within noisy environments. If the ALERTPACKET is unsuccessful for a number of instances greater than COMM_RTRY, an ALRTPCKT_PEC_ERR is asserted. In this instance, it is recommended that the host perform a READALL to query the status of all devices within the daisy-chain.

To ensure data integrity, main and redundant ALERTPACKET buffers are compared after receipt of a new preamble or the ALERTPACKET stop character. A failure of ALERTPACKET data integrity between the two buffers results in an ALRTPCKBUF_HW_ERR status bit assertion. In the case of an ALRTPCKTBUF_HW_ERR bit assertion, the ALERTPACKET data should be ignored, and the STATUS1 register for each of the daisy-chain devices should be monitored through a user-directed READDEVICE and READALL.

The ALRTPCKTBUF_FULL status bit is set when a new ALERTPACKET is received with no PEC error. The bytes in the ALRTPCKTBUF are cleared to zero after the byte is read. If any bytes are read/clear, and a new ALERTPACKET is not received, the ALRTPCKTBUF_FULL register is cleared. In BMS Safe Monitoring and Sleep modes, the ALRTPCKTBUF is cleared automatically, and ALRTPCKTBUF_FULL toggles autonomously.

During Commanded Operation mode, ALERTPACKET STATUS bits are masked by the STATUS_ERR_MASK register and are logically OR'ed together and reported in the ALRTPCKT_STATUS_ERR status bit. It is recommend that the ALRTPACKET_STATUS[5] be masked to avoid persistent reporting of a PEC that may not be cleared. By default, the ALRTPACKET_STATUS[5] (ALRTPEC from the BMS daisy-chain) is masked via the STATUS_ERR_MASK and the STATUS_DBNC_MASK.

Like other MAX17851 status bits, the ALRTPCKT_STATUS_ERR_ALRT bit may be disabled using the ALRTPCKT_STATUS_ERR_$\overline{\text{ALRTE}}$N bit. Disabling this bit prevents non-user directed communication from interrupting microcontroller tasks via the $\overline{\text{ALERT}}$ pin. The ALERTPACKET status can still be verified through intermittent polling of the ALERTPACKET buffer or the ALRTPCKT_STATUS_ERR status bit.

The ALRTPCKT_STATUS_ERR bit is presented in the LSSM Status byte.

During BMS Safe Monitoring or Sleep modes, ALERTPACKET STATUS bits are masked/unmasked by the

STATUS_DBNC_MASK register. Unmasked STATUS bits are logically OR'ed together and debounced by the BMS Safe Monitoring or Sleep logic. During BMS Safe Monitoring or Sleep modes, if an unmasked STATUS bit is asserted in the ALERTPACKET data and persists a greater number of times than the corresponding mode DBNC register, the FAULT_TIMER starts.

Upon receipt of new ALERTPACKET data, the ALRTPCKTBUF always overwrites previous values.

Like all other transactions, the ALERTPACKET data is monitored by the LSSM for overflow, underflow, PEC, timeout, and other data integrity issues.

When the automated ALERTPACKET data is stored in the ALERTPACKET buffer, it is not PEC protected from the MAX17851 to the host microcontroller. However, it is PEC protected when received by the RX port from the TX port transmission, guaranteeing data integrity. If the user issues an ALERTPACKET by writing this to the command to the TX buffer then it is processed as normal UART data and it will be stored in the RX buffer. It is PEC protected from the MAX17851 to the host controller. If an error is observed in the ALERTPACKET buffer, the user should issue an ALERTPACKET to ensure proper communication back to the microcontroller.

Automated and User Specified Alive Counter

The automated Alive Counter provides an additional layer of communication safety by explicitly appending a frame counter at the end of a UART transaction. This allows for a one-to-one comparison of each transaction to ensure no packets have been inserted, deleted, re-ordered, or non-refreshed.

When ALIVECOUNT_EN = 0x11, the MAX17851 automated Alive Counter automatically appends a rolling counter to the end of the outgoing read and write transactions. The first transaction transmitted uses a default ALIVECOUNT_SEED of 0x00, whereas all subsequent transactions increment this seed value by one. If the ALIVECOUNT_SEED reaches 0xFF it rolls over to 0x00 and proceeds.

The transmitted ALIVECOUNT_SEED propagates through the daisy-chain and is incremented by each unit addressed by the command. The returned alive count value is stored in the ALIVECOUNT_RET register and is compared against the expected value for every transaction. In the event the ALIVECOUNT_RET does not match the expected value, the ALIVECOUNT_ERR bit is asserted. The expected ALIVECOUNT_RET value is computed as follows:

Expected ALIVECOUNT_RET = ALIVECOUNT_SEED + DEV_COUNT (WRITEALL/READALL)

Expected ALIVECOUNT_RET = ALIVECOUNT_SEED + 1 (WRITEDEVICE/READDEVICE/READBLOCK)

Note: In the event of an ALIVECOUNT_ERR, verify the accuracy of the DEV_COUNT register content. If a valid error is found, query the ALIVECOUNT_Q until it reads a value of 0x0. Clear the RX, TX, and LSSM buffers (CLR_RXBUF, CLR_TXBUF, CLR_LSSM respectively) to re-align the transactions.

When the automatic Alive Counter is enabled, the returned ALIVECOUNT data is not stored in the RX buffer, since it is internally stored as ALIVECOUNT_RET. Every transaction is verified in the LSSM..

For system architectures that utilize long daisy-chains, multiple transactions may be transmitted before a given transaction is received. The number of these in-flight transactions are stored in the ALIVECOUNT_Q register. For a command/response transaction implementation, where no new commands are sent before the successful verification of the transaction, the ALIVECOUNT_Q register will typically read 0x0.

Note: ALIVECOUNT_Q provides the number of messages in the LSSM buffer that have not yet been received by the RX buffer.

If the CLR_ALIVECOUNT_SEED command is issued while the ALIVECOUNT_Q register is non-zero, the command is ignored, and the ALIVECOUNT_ERR status bit is set.

When enabled, the automated Alive Counter is active in all modes of operation (Commanded Operation, Safe Monitoring, Sleep).

When ALIVECOUNT_EN = 0x10, the MAX17851 is in User Specified Alive Counter mode. In this mode, the host must manually append an alive count to the end of supported transactions when writing UART transmit data to the MAX17851 via SPI. The returned alive count byte is stored in the RX buffer in this mode, and an error will not result in the assertion of ALIVECOUNT_ERR.

Examples of the READALL command propagation in a two module, single UART daisy-chain are shown below:

Table 14. Command Propagation with User Specified Alive Counter (ALIVECOUNT_EN = 10)

TX BUFFER		RECEIVED UART PACKET		RX BUFFER	
Message Length	0x09	Preamble		Command Byte	0x03
Command Byte	0x03	Command Byte	0x03	Register Address	0x12
Register Address	0x12	Register Address	0x12	LSB Data Device Address 1	0xB1
Data Check	0x00	LSB Data Device Address 1	0xB1	MSB Data Device Address 1	0xB2
PEC	0xCB	MSB Data Device Address 1	0xB2	LSB Data Device Address 0	0xB1
Alive Counter	0x00	LSB Data Device Address 0	0xB1	MSB Data Device Address 0	0xB2
		MSB Data Device Address 0	0xB2	Data Check	0x00
		Data Check	0x00	Alive Counter	0x02
		PEC	0x67	LSSM Status Byte	0x00
		Alive Counter	0x02	PEC	0x3E
		Stop			

In the example above (ALIVECOUNT_EN = 10), the user appends an Alive Counter byte to the TX buffer, and the daisy-chain returns a value of 0x02 indicating the number of devices.

Table 15. Command Propagation with Automatic Alive Counter (ALIVECOUNT_EN = 11)

TX BUFFER		RECEIVED UART PACKET		RX BUFFER	
Message Length	0x08	Preamble		Command Byte	0x03
Command Byte	0x03	Command Byte	0x03	Register Address	0x12
Register Address	0x12	Register Address	0x12	LSB Data Device Address 1	0xB1
Data Check	0x00	LSB Data Device Address 1	0xB1	MSB Data Device Address 1	0xB2
PEC	0xCB	MSB Data Device Address 1	0xB2	LSB Data Device Address 0	0xB1
		LSB Data Device Address 0	0xB1	MSB Data Device Address 0	0xB2
		MSB Data Device Address 0	0xB2	Data Check	0x00
		Data Check	0x00	LSSM Status Byte	0x00
		PEC	0x67	PEC	0x7D
		Alive Counter Return (TX Count =5, ALIVECOUNT_SEED = 4)	0x06		
		Stop			

In the example above (ALIVECOUNT_EN = 11), the automated Alive Counter is enabled and is the fifth transaction in the sequence (ALIVECOUNT_SEED = 4).

If the LSSM status byte reads back a RX_READY value of 0, then the read data in the RX buffer is read without proper termination. In this scenario, the RX buffer data is considered to be invalid, and the message should be re-transmitted.

Note: If TX_NOSTOP is enabled, the RX_STOP status bit cannot be used to delineate transaction completion. The RX_READY status bit is valid for all transaction completion use cases.

Data-Check Parser

When enabled via the DC_EN register, the Data-Check byte must be generated by the host. The MAX17851

automatically parses incoming Data-Check bytes and reports the returned daisy-chain DC_PECERR via the COMM_ERR status bit. No other bits from the Data-Check byte are reported. All other Data-Check byte errors are reported in ALRTPCKTBUF data. When the Data-Check byte is enabled, the user has the option to either store or not store it in the RX buffer (10b = data check enabled and stored in the RX buffer, 11b = data check enabled but not stored in the RX buffer).

Lockstep Safety Measure Verification

UART communication is validated for integrity through the comparison of the transmitted UART packet to the received one. This verification occurs on all UART transactions for supported command bytes as well as reserved command bytes (see Table 10).

The lockstep safety measure verification engine contains 2 individual cores that analyze both the data and message properties to determine if a corruption exists within the data payload. These faults are stored as the LSSM Status byte, inserted to each received communication message. The LSSM Status byte is appended at the end of the RX buffer payload and before the PEC byte. The PEC is re-calculated by the MAX17851 to include and protect the LSSM Status byte. This ensures fault protection of the entire RX payload. The LSSM Status byte can be optionally read via the STATUS_LSSM_BYTE register and asserted through the $\overline{ALERT}$ pin if the associated ALRTEN register bits are set.

The following table is a list of errors represented for each of the LSSM status byte bits.

Table 16. LSSM Status Byte Error Mapping

BIT	LSSM STATUS REGISTER BIT	ERRORS ASSOCIATED WITH REGISTER BIT	DESCRIPTION
7	RX_READY	N/A	The UART has finished receiving a properly framed message (stop character, preamble, or frame timeout) and is ready to be read.
6	ALRTPCKT_STATUS_ERR	Alert Packet Buffer Status Error	Logical OR of all non-masked STATUS bits in the automated ALERTPACKET.
5	COMM_ERR	Communication Timeout Error	A transmitted command is not received before the expiration of COMM_TO_DLY,
		Command Byte Invalid	Command byte invalid. Any command that is outside of the defined UART commands is invalid.
		PEC Received Error	PEC RX Error: Decoded PEC does not match received PEC. PEC RX Error: Decoded PEC does not match received PEC. This error does not include PEC RX errors generated by autogenerated ALERTPACKET commands.
		Data Check Error	PEC error indicated by the data-check parser (if enabled).
		Register Address Invalid	Register Address byte invalid.
4	ALRTPCKT_ERR	Alert Packet Communications Error	A response to an automated ALERTPACKET has timed out, or the PEC in the received automated ALERTPACKET is incorrect (ALRTPCKT_COMM_ERR).
		Alert Packet Hardware Error	The redundant Alert Packet buffer and main Alert Packet buffer do not match. (ALRTPCKT_HW_ERR).
3	COMM_MSMTCH_ERR	Command Byte Mismatch	The command byte received differs from the command that was sent.
		Register Address Mismatch	Register Address byte mismatch between received and transmitted transaction.
		Write Message Mismatch	WRITE_ALL, WRITE_DEVICE Message Mismatch between received and transmitted transaction

Table 16.
LSSM Status Byte Error Mapping (continued)

BIT	LSSM STATUS REGISTER BIT	ERRORS ASSOCIATED WITH REGISTER BIT	DESCRIPTION
		LSSM Hardware Error	The redundant LSSM buffer and main LSSM buffer do not match.
		Message Length Mismatch	Message Length Mismatch error between received and transmitted transaction.
		LSSM Over/ Underflow	The LSSM buffer has overflowed or underflowed.
2	COMMAND_OP	N/A	This bit is set when the MAX17851 is in normal operation.
1	ALIVECOUNT_ERR	Incorrect Alive Count	The Alive Count returned is not = ALIVECOUNT_SEED - ALIVECOUNT_Q + DEV_COUNT (WRITEALL/READALL) The Alive Count returned is not = ALIVECOUNT_SEED - ALIVECOUNT_Q + 1 (WRITEDEVICE/READDEVICE/READBLOCK)
0	HW_ERR	Oscillator HW Error	Low or High Frequency oscillator is out of range or has failed.
		VDDL1 HW Error	VDDL1 Undervoltage or Overvoltage condition.
		VDDL2 HW Error	VDDL2 Undervoltage or Overvoltage condition.
		Register HW Error	Internal register has failed test check.
		OTP HW Error	One Time Programmable memory has failed test check.
		VDCIN1 HW Error	VDCIN1 Undervoltage or Overvoltage condition.
		VDCIN2 HW Error	VDCIN2 Undervoltage or Overvoltage condition.

The LSSM_UNDRFLW_ERR bit is asserted when the number of received transactions has exceeded the number of expected transactions (overflow).

To prevent overflow conditions, the LSSM prevents the transmission of data by the Transmit buffer when the LSSM is full (LSSM_FULL is true).

Note: When an LSSM underflow condition occurs, the user must re-initialize the LSSM with the CLR_LSSM bit. If there are transactions in flight during this time (ALIVECOUNT_Q is not 0), the condition may persist, and the operation may need to be repeated.

See *Fault Handling Guidelines* for additional information on handling faults and message exceptions.

Dual Lockstep Datapath Processing

The lockstep safety measure verification engine (LSSM) provides validation of all UART traffic that is sent and received by the MAX17851.

The message from the Transmit buffer is checked and stored by the LSSM (1). After a short delay (2), which is added to detect any noise induced into the signal, the redundant LSSM core (3) also checks and stores the message. The two LSSM cores are compared against each other to detect hardware errors within the MAX17851. Transmitted messages are checked for a valid command, valid length, valid device count, and correct PEC.

Received messages are handled similarly, with redundant checking by the two LSSM cores. Received messages are checked against the outgoing transmitted message to make sure that a valid transaction has occurred. Improperly terminated messages are detected and flagged.

If a daisy-chain device inserts an unexpected message or unexpected data, the LSSM detects and flags the error, since there is not a matching outgoing message in the LSSM. Similarly, a message that is deleted or lost in the daisy-chain is detected and flagged by the LSSM.

The Alert Packet buffer (5) and redundant Alert Packet buffer (4) process and store alert packets and are available to be read directly by the host microcontroller.

Figure 19 shows a diagram of the MAX17851 transmit and receive data processing.

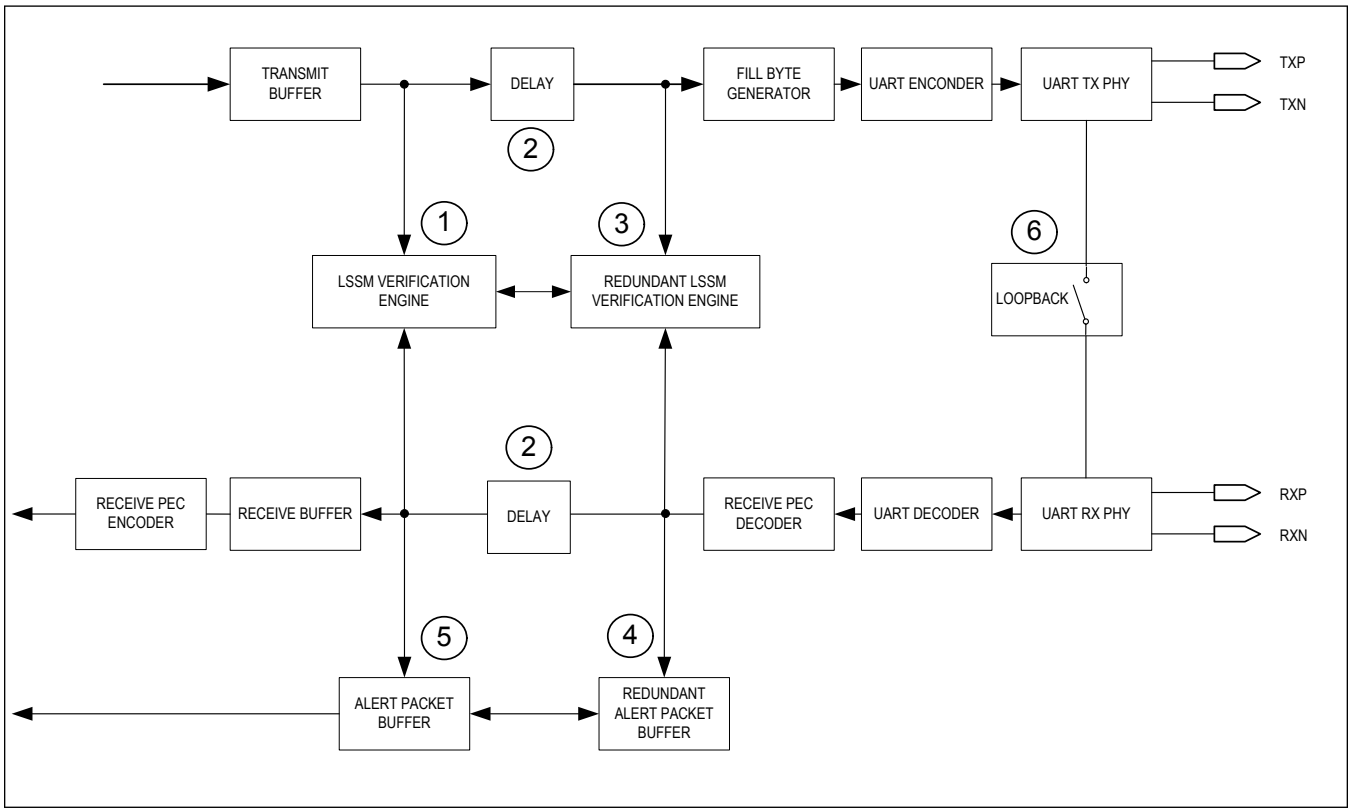


Figure 19. LSSM Datapath

Alerts and Faults

$\overline{\text{ALERT}}$ Pin, Status, and Alert Flag Behavior

The $\overline{\text{ALERT}}$ pin (active low) is asserted when any of the alert flag registers in the ALERT register space are asserted, notifying the host that an event has occurred. An event may indicate a MAX17851 fault, daisy-chain fault, or normal UART transaction event. See the *Register Table* for details.

For applications that utilize a command/response protocol with only one transaction sent to the daisy-chain and immediately read, a direct correlation of events and transactions is provided.

For applications that pipeline multiple transactions through the MAX17851 (multiple transactions are in-flight at the same time within the daisy-chain), the $\overline{\text{ALERT}}$ pin cannot be used to identify which specific transaction caused an alert flag. In this scenario, the host should flush the buffers and re-transmit all data that was in-flight.

There are multiple events that can cause status register bits to be set (see the *Register Table* for details). For each event, there is a status bit, alert enable bit, and alert flag bit. The status bit is the real-time status of the event and is set or cleared autonomously. The alert enable bit corresponding to the status bit determines whether or not the event sets the alert flag, and therefore asserts the $\overline{\text{ALERT}}$ pin.

Regardless of alert enable bit programming, all alert flag bit assertions are gated by ALRTRST. Therefore, the host micro should clear the ALRTRST bit after any POR or SWPOR event. This deasserts the $\overline{ALERT}$ pin and enable alert flag assertion. The ALRTRST bit does not have a corresponding alert enable bit and cannot be disabled.

Alert flag bit assertions are edge-triggered (i.e., the alert flag is set when the alert enable bit is true, and the corresponding status bit transitions from a logic-zero state to logic-one state). The one exception is that alert flag bits (SAFEMON_GPIO12_ALRT) can only be cleared by the host. When enabled, alert flags are not set until the corresponding status bit transitions from a logic-zero state to a logic-one state. If the alert flag bit is cleared when the corresponding status bit is true, the alert flag bit does not set again until the status bit transitions from a logic-zero state to a logic-one state. When any alert flag bit is set, the MAX17851 asserts the $\overline{ALERT}$ pin. All alert flag bits must be cleared for the $\overline{ALERT}$ pin to be deasserted.

Alert flag bits are either event driven or persistent. Persistent alert flags cannot be cleared if the corresponding status bit remains true. Event driven alert flags may be cleared at any time. See the *Register Table* for details.

Note: The LSSM byte contains specific data correlated to the transaction. Thus, it is not recommended to use the $\overline{ALERT}$ pin for LSSM_BYTE notifications when the application utilizes multiple/pipelined transactions to the daisy-chain.

Fault Timer

The fault timer indicates the amount of time in seconds that has expired since a communication fault or daisy-chain fault (as indicated by an ALERTPACKET) has occurred in Sleep or BMS Safe Monitoring mode.

The fault timer starts once the fault is debounced and stops once the SLP_ALRT or SAFEMON_ALRT bits are cleared.

The fault timer is reset at the start of Sleep or BMS Safe Monitoring mode.

The fault timer is read only and two bytes long. The FAULT_TIMER0 register holds the lower byte, while the FAULT_TIMER1 holds the upper byte.

Hardware Fault Detection

The MAX17851 continuously monitors itself for hardware faults that may affect device operation. Faults include supply voltages that exceed a specified operating range, oscillator drift errors that may effect communication timing, corrupted device trim, corrupted memory (stuck at or transient), and internal register errors. A detected fault is reported in the HW_ERR status bit.

Register Fault Detection

The MAX17851 features continual fault detection for the CONFIG and ALRTEN registers. This protects against transient faults and hardware errors that may cause unintended device operation.

SPI input error detection is accomplished with a comparison of SPI input data against the post-write register data. This comparison is executed on every write to the CONFIG or ALRTEN register blocks. A discrepancy between the input SPI data and the register data result in the HW_ERR status bit being set.

Additional error detection in the CONFIG and ALRTEN registers detects a difference between the current state of the registers and that which was originally programmed. As such, this bit may be asserted at any time, even in the absence of SPI communication. A fault results in the HW_ERR status bit being set.

MAX17851 User Register Map

READ ADDRESS	WRITE ADDRESS	NAME	MSB							LSB
STATUS Registers										
0x01	0x00	STATUS_RX[7:0]	RX_ERR	–	RX_BUSY	RX_IDLE	RX_OVRFLW_ERR	RX_FULL	RX_STOP	RX_EMPTY
0x03	0x02	STATUS_TX[7:0]	TX_HELD	–	TX_BUSY	TX_IDLE	TX_OVRFLW_ERR	TX_FULL	TX_AVAIL	TX_EMPTY

READ ADDRESS	WRITE ADDRESS	NAME	MSB							LSB
0x05	0x04	STATUS_LSSM_BYTE[7:0]	RX_READY	ALRTPCKT_STATUS_ERR	COMM_ERR	ALRTPCKT_ERR	COMM_MSMTCH_ERR	COMMAND_OP	ALIVECOUNT_ERR	HW_ERR
0x07	0x06	STATUS_GEN[7:0]	HFOSC_HW_ERRB	DEV_COUNT_ERR	WD_ERR	GPIO_ERR	DATAPATH_ERR	–	ALRTPCKT_COMM_ERR	ALRTPCKTBUF_HW_ERR
0x09	0x08	STATUS_OPSTATE[7:0]	–	–	–	–	SAFEMON	–	–	–
0x0B	0x0A	STATUS_BUF[7:0]	ALRTPCKTBUF_FULL	–	–	–	LSSM_FULL	–	–	–
0x0D	0x0C	STATUS_WD[7:0]	–	–	–	WD_TO_ERR	WD_OPEN	WD_LFSR_ERR	WD_RJCT_ERR	WD_EXP_ERR
0x0F	0x0E	STATUS_GPIO[7:0]	GPIO4_RD	GPIO3_RD	GPIO2_RD	GPIO1_RD	GPIO4_ERR	GPIO3_ERR	GPIO2_ERR	GPIO1_ERR
ALERT Registers										
0x11	0x10	ALERT_RX[7:0]	RX_ERR_ALRT	–	RX_BUSY_ALRT	RX_IDLE_ALRT	RX_OVRFLW_ERR_ALRT	RX_FULL_ALRT	RX_STOP_ALRT	RX_EMPTY_ALRT
0x13	0x12	ALERT_TX[7:0]	TX_HELD_ALRT	–	TX_BUSY_ALRT	TX_IDLE_ALRT	TX_OVRFLW_ERR_ALRT	TX_FULL_ALRT	TX_AVAIL_ALRT	TX_EMPTY_ALRT
0x15	0x14	ALERT_LSSM_BYTE[7:0]	RX_READY_ALRT	ALRTPCKT_STATUS_ERR_ALRT	COMM_ERR_ALRT	ALRTPCKT_ERR_ALRT	COMM_MSMTCH_ERR_ALRT	COMMAND_OP_ALRT	ALIVECOUNT_ERR_ALRT	HW_ERR_ALRT
0x17	0x16	ALERT_GEN[7:0]	–	DEV_COUNT_ERR_ALRT	WD_ERR_ALRT	GPIO_ERR_ALRT	DATAPATH_ERR_ALRT	SPI_ERR_ALRT	ALRTPCKT_COMM_ERR_ALRT	ALRTPCKTBUF_HW_ERR_ALRT
0x19	0x18	ALERT_OPSTATE[7:0]	ALRTRST	–	SLP_ALRT	SLP_STATUS_ERR_ALRT	SAFEMON_ALRT	SAFEMON_GPIO12_ALRT	SAFEMON_STATUS_ERR_ALRT	SAFEMON_CONFIG_ERR_ALRT
0x1B	0x1A	ALERT_BUF[7:0]	ALRTPCKTBUF_FULL_ALRT	–	–	–	LSSM_FULL_ALRT	–	–	–
0x1D	0x1C	ALERT_WD[7:0]	–	–	–	WD_TO_ERR_ALRT	WD_OPEN_ALRT	WD_LFSR_ERR_ALRT	WD_RJCT_ERR_ALRT	WD_EXP_ERR_ALRT

READ ADDRESS	WRITE ADDRESS	NAME	MSB							LSB
0x1F	0x1E	ALERT_GPIO[7:0]	–	–	–	–	GPIO4_ERR_ALRT	GPIO3_ERR_ALRT	GPIO2_ERR_ALRT	GPIO1_ERR_ALRT
ALRTEN Registers										
0x21	0x20	ALRTEN_RX[7:0]	RX_ERR_ALRTEN	–	RX_BUSY_ALRTEN	RX_IDLE_ALRTEN	RX_OVRFLW_ERR_ALRTEN	RX_FULL_ALRTEN	RX_STOP_ALRTEN	RX_EMPTY_ALRTEN
0x23	0x22	ALRTEN_TX[7:0]	TX_HELD_ALRTEN	–	TX_BUSY_ALRTEN	TX_IDLE_ALRTEN	TX_OVRFLW_ERR_ALRTEN	TX_FULL_ALRTEN	TX_AVAIL_ALRTEN	TX_EMPTY_ALRTEN
0x25	0x24	ALRTEN_LSSM_BYTE[7:0]	RX_READY_ALRTEN	ALRTPCKT_STATUS_ERR_ALRTEN	COMM_ERR_ALRTEN	ALRTPCKT_ERR_ALRTEN	COMM_MSMTCH_ERR_ALRTEN	COMMAND_OP_ALRTEN	ALIVECOUNT_ERR_ALRTEN	HW_ERR_ALRTEN
0x27	0x26	ALRTEN_GEN[7:0]	–	DEV_COUNT_ERR_ALRTEN	WD_ERR_ALRTEN	GPIO_ERR_ALRTEN	DATAPATH_ERR_ALRTEN	SPI_ERR_ALRTEN	ALRTPCKT_COMM_ERR_ALRTEN	ALRTPCKTBUF_HW_ERR_ALRTEN
0x29	0x28	ALRTEN_OPSTATE[7:0]	–	–	SLP_ALRTEN	SLP_STATUS_ERR_ALRTEN	SAFEMON_ALRTEN	SAFEMON_GPIO12_ALRTEN	SAFEMON_STATUS_ERR_ALRTEN	SAFEMON_CONFIG_ERR_ALRTEN
0x2B	0x2A	ALRTEN_BUF[7:0]	ALRTPCKTBUF_FULL_ALRTEN	–	–	–	LSSM_FULL_ALRTEN	–	–	–
0x2D	0x2C	ALRTEN_WD[7:0]	–	–	–	WD_TO_ERR_ALRTEN	WD_OPEN_ALRTEN	WD_LFSR_ERR_ALRTEN	WD_RJCT_ERR_ALRTEN	WD_EXP_ERR_ALRTEN
0x2F	0x2E	ALRTEN_GPIO[7:0]	–	–	–	–	GPIO4_ERR_ALRTEN	GPIO3_ERR_ALRTEN	GPIO2_ERR_ALRTEN	GPIO1_ERR_ALRTEN
COMMAND Registers										
0x41	0x40	CLR_TXBUF[7:0]	–	–	–	–	–	–	–	–
0x43	0x42	CLR_RXBUF[7:0]	–	–	–	–	–	–	–	–
0x45	0x44	CLR_LSSM[7:0]	–	–	–	–	–	–	–	–
0x49	0x48	CLR_ALIVECOUNT_SEED[7:0]	–	–	–	–	–	–	–	–
0x4B	0x4A	SWPOR[7:0]	–	–	–	–	–	–	–	SWPOR
0x4D	0x4C	SLP_EN[7:0]	–	–	–	–	–	–	–	SLP_EN

READ ADDRESS	WRITE ADDRESS	NAME	MSB							LSB
0x4F	0x4E	VER_CONFIG[7:0]	–	–	–	–	–	–	–	VER_CONFIG
0x51	0x50	LOAD_CONFIG[7:0]	–	–	–	–	–	–	–	LOAD_CONFIG
0x53	0x52	WD_KEY[7:0]	WD_KEY[7:0]							
CONFIG Registers										
0x61	0x60	CONFIG_GEN0[7:0]	–	–	DEV_COUNT[5:0]					
0x63	0x62	CONFIG_GEN1[7:0]	SINGLE_ENDED	BAUD_RATE[2:0]			–	–	–	–
0x65	0x64	CONFIG_GEN2[7:0]	RX_RAW_DATA	TX_RAW_DATA	TX_PREAMBLES	TX_QUEUE	TX_ODDPARITY	TX_PAUSE	TX_NOSTOP	TX_NOPREAMBLE
0x67	0x66	CONFIG_GEN3[7:0]	–	TX_AUTO	TX_UNLIMITED	SPI_DOUT_EN	ALRTPCKT_TIMING[3:0]			
0x69	0x68	CONFIG_GEN4[7:0]	CO_ALRTPCKTEN	RXSWAP_EN	MS_EN[1:0]		DC_EN[1:0]		ALIVECOUNT_EN[1:0]	
0x6B	0x6A	CONFIG_GEN5[7:0]	TX_START_SETUP	TX_HI_Z	ALRTPCKT_DBNC[2:0]			SPI_SFTYSCLK	SPI_SFTYSDI	SPI_SFTYCSB
0x6D	0x6C	CONFIG_SAFEMON0[7:0]	GPIOREC_DLY[7:0]							
0x6F	0x6E	CONFIG_SAFEMON1[7:0]	CONT_TIMER_DLY[7:0]							
0x71	0x70	CONFIG_SAFEMON2[7:0]	–	–	–	–	–	–	SAFEMON_SCAN_DLY[1:0]	
0x73	0x72	CONFIG_SAFEMON3[7:0]	NOMON	–	–	–	SM_GPIO4_MASK	SM_GPIO3_MASK	SM_GPIO2_MASK	SM_GPIO1_MASK
0x75	0x74	CONFIG_SLP[7:0]	–	–	SLP_SCAN_DLY[1:0]		SLP_ALRTPCKTEN	SLP_CBNTFY[2:0]		
0x77	0x76	CONFIG_COMM[7:0]	–	–	COMM_RTRY[1:0]		–	COMM_TO_DLY[2:0]		
0x79	0x78	STATUS_DBNC_MASK0[7:0]	STATUS_DBNC_MASK[7:0]							
0x7B	0x7A	STATUS_DBNC_MASK1[7:0]	STATUS_DBNC_MASK[15:8]							
0x7D	0x7C	STATUS_ERR_MASK0[7:0]	STATUS_ERR_MASK[7:0]							
0x7F	0x7E	STATUS_ERR_MASK1[7:0]	STATUS_ERR_MASK[15:8]							
0x81	0x80	CONFIG_GPIO12[7:0]	–	GPIO2_CFG[2:0]			–	GPIO1_CFG[2:0]		
0x83	0x82	CONFIG_GPIO34[7:0]	–	GPIO4_CFG[2:0]			–	GPIO3_CFG[2:0]		
0x85	0x84	CONFIG_WD0[7:0]	WD_OPN[3:0]				WD_CLO[3:0]			
0x87	0x86	CONFIG_WD1[7:0]	WD_1UD[2:0]				WD_DIV[4:0]			

READ ADDRESS	WRITE ADDRESS	NAME	MSB							LSB
0x89	0x88	CONFIG_WD2[7:0]	WD_EN	–	–	–	WD_SWW	WD_DBNC[2:0]		
RX COMMAND Registers										
0x8D	0x8C	ALRTPCKTBUF_RD_PTR[7:0]	–	–	–	–	–	ALRTPCKTBUF_RD_PTR[2:0]		
0x8F	0x8E	ALRTPCKTBUF_RD_MSG[7:0]	ALRTPCKTBUF_RD_MSG[7:0]							
0x91	0x90	RX_RD_MSG[7:0]	RX_RD_MSG[7:0]							
0x93	0x92	RX_RD_NXT_MSG[7:0]	RX_RD_NXT_MSG[7:0]							
0x95	0x94	TX_QUEUE_SEL[7:0]	–	–	–	–	TX_Q[1:0]		LD_Q[1:0]	
0x97	0x96	RX_RD_PTR[7:0]	–	RX_RD_PTR[6:0]						
0x99	0x98	RX_WR_PTR[7:0]	–	RX_WR_PTR[6:0]						
0x9B	0x9A	RX_NXT_MSG_PTR[7:0]	–	RX_NXT_MSG_PTR[6:0]						
0x9D	0x9C	RX_SPACE[7:0]	–	RX_SPACE[6:0]						
0x9F	0x9E	RX_BYTE[7:0]	–	–	–	–	–	RX_FIRST_BYTE	RX_BYTE_ERR	RX_LAST_BYTE
TX COMMAND Registers										
0xB1	0xB0	NXT_LDQ[7:0]	NXT_LDQ[7:0]							
0xC1	0xC0	LDQ[7:0]	LDQ[7:0]							
0xC3	0xC2	LDQ_PTR[7:0]	–	–	–	LDQ_PTR[4:0]				
0xD1	0xD0	CONFIGQ[7:0]	CONFIGQ[7:0]							
0xD3	0xD2	CONFIG_PTR[7:0]	–	CONFIG_QUEUE_PTR[1:0]	CONFIG_BYTE_PTR[4:0]					
INFO Registers										
0xDD	0xDC	STATE[7:0]	–	–	–	–	–	STATE[2:0]		
0xDF	0xDE	COMM_RTRY_CNT[7:0]	–	–	–	COMM_RTRY_CNT[4:0]				
0xE1	0xE0	ALRTPCKT_ERR_CNT[7:0]	ALRTPCKT_ERR_CNT[7:0]							
0xE3	0xE2	WD_FAULT_CNT[7:0]	WD_FAULT_CNT[7:0]							
0xE5	0xE4	ALIVECOUNT_SEED[7:0]	ALIVECOUNT_SEED[7:0]							
0xE7	0xE6	ALIVECOUNT_RET[7:0]	ALIVECOUNT_RET[7:0]							
0xE9	0xE8	ALIVECOUNT_Q[7:0]	–	–	–	–	–	ALIVECOUNT_Q[2:0]		
0xEB	0xEA	FAULT_TIMER0[7:0]	FAULT_TIMER[7:0]							
0xED	0xEC	FAULT_TIMER1[7:0]	FAULT_TIMER[15:8]							
0xEF	0xEE	SLP_CBTIMER0[7:0]	SLP_CBTIMER[7:0]							
0xF1	0xF0	SLP_CBTIMER1[7:0]	SLP_CBTIMER[15:8]							
0xF3	0xF2	VERSION[7:0]	MODEL[3:0]				VERSION[3:0]			
0xF5	0xF4	MODEL[7:0]	MODEL[11:4]							

Register Details

STATUS_RX (R/W = 0x01/0x00)

BIT	7	6	5	4	3	2	1	0
Field	RX_ERR	–	RX_BUSY	RX_IDLE	RX_OVRFLW_ERR	RX_FULL	RX_STOP	RX_EMPTY
Reset	0x0	–	0x0	0x1	0x0	0x0	0x0	0x1
Access Type	Read Only	–	Read Only	Read Only	Read Only	Read Only	Read Only	Read Only

BITFIELD	BITS	DESCRIPTION
RX_ERR	7	Receive Error The data byte at location RX_RD_PTR may contain an error (the corresponding character contained a Manchester and/or parity error). This bit is set when the byte is read, not when the byte is received or written.
RX_BUSY	5	Receive Busy The UART is busy receiving data.
RX_IDLE	4	The UART is not receiving data.
RX_OVRFLW_ERR	3	The data byte at location RX_WR_POINTER in the Receive buffer is overwritten because the Receive buffer is full. Cleared when the Receive buffer is not full (when the buffer is read).
RX_FULL	2	The number of empty bytes in the Receive buffer is zero.
RX_STOP	1	RX Stop Received The UART is finished receiving a properly framed message (stop character) and it is ready to be read. The UART clears this bit automatically after all unread messages are read or if the UART detects a new preamble character. The UART does not set this bit if the buffer is empty and receives a stop character. This bit is not be set for automated ALERTPACKET transactions.
RX_EMPTY	0	The Receive buffer is cleared and contains no unread data (RX_RD_PTR = RX_WR_PTR - 1).

STATUS_TX (R/W = 0x03/0x02)

BIT	7	6	5	4	3	2	1	0
Field	TX_HELD	–	TX_BUSY	TX_IDLE	TX_OVRFLW_ERR	TX_FULL	TX_AVAIL	TX_EMPTY
Reset	0x0	–	0x0	0x1	0x0	0x0	0x1	0x1
Access Type	Read Only	–	Read Only	Read Only	Read Only	Read Only	Read Only	Read Only

BITFIELD	BITS	DESCRIPTION
TX_HELD	7	This bit is set if the following condition is true: Number of empty bytes in Receive buffer < (Message length in transmit queue + 1) When this status bit is true and TX_UNLIMITED = 1'b0, the Transmit buffer does not transmit the data in the current LD_Q until the RX Buffer has enough available space. When TX_UNLIMITED = 1'b1, the TX_HELD bit is never set, and data is transmitted regardless of available RX buffer space. The RX_OVRFLW_ERR should be monitored to ensure data integrity in this condition.
TX_BUSY	5	The UART is busy transmitting data.
TX_IDLE	4	The UART is not transmitting data.
TX_OVRFLW_ERR	3	LD_Q could not be incremented, because the next queue contained untransmitted data. Any writes in this state overwrite the load queue.
TX_FULL	2	All queues in the Transmit buffer are full except the load queue (LD_Q = TX_Q - 1).
TX_AVAIL	1	One or more queues in the Transmit buffer are available for loading (TX_FULL is false).
TX_EMPTY	0	All the queues in the Transmit buffer are cleared and available for loading (LD_Q = TX_Q).

STATUS_LSSM_BYTE (R/W = 0x05/0x04)

BIT	7	6	5	4	3	2	1	0
Field	RX_READY	ALRTPCKT_STATUS_ERR	COMM_ERR	ALRTPCKT_ERR	COMM_MSMTCH_ERR	COMMAND_OP	ALIVECOUNT_ERR	HW_ERR
Reset	0x0	0x0	0x0	0x0	0x0	0x1	0x0	0x0
Access Type	Read Only	Read Only	Read Only	Read Only	Read Only	Read Only	Read Only	Read Only

BITFIELD	BITS	DESCRIPTION
RX_READY	7	The UART is finished receiving a properly framed message (stop character, preamble, or frame time_out) and it is ready to be read. The UART clears this bit automatically after all unread messages are read. The UART does not set this bit if the buffer is empty and it receives a stop character This bit is not set for automated ALERTPACKET transactions.
ALRTPCKT_STATUS_ERR	6	ALERTPACKET Status Error An unmasked Alert Packet Status bit is received in the Alert Packet buffer. In Commanded Operation mode, masking of the received ALERTPACKET status bits is a function of the STATUS_ERR_MASK registers. In Sleep or BMS Safe Monitoring mode, masking of the received ALERTPACKET status bits is a function of the STATUS_DBNC_MASK registers and will be auto-cleared as the status is debounced.
COMM_ERR	5	Communications Error A communication error is observed that has corrupted, inserted, or omitted data within the UART message. **Note:** This bit is cleared on receipt of a preamble and set at the completion of a transaction containing an error.
ALRTPCKT_ERR	4	ALERTPACKET Error This status bit is asserted with any of the following ALERTPACKET status errors: ALRTPCKT_COMM_ERR ALRTPCKT_HW_ERR
COMM_MSMTCH_ERR	3	Communication Mismatch Error The LSSM detects a communication problem where the received data does not match the transmitted data.
COMMAND_OP	2	Commanded Operation Mode Status This bit represents Commanded Operation mode (device is not in Sleep or BMS Safe Monitoring mode). If this bit is not a logical 1 then the device has entered into another operational mode.

BITFIELD	BITS	DESCRIPTION
ALIVECOUNT_ERR	1	Automatic Alive Counter Error When the Automatic Alive Counter mode is enabled, this bit is asserted in two scenarios: 1) The received ALIVECOUNT byte (ALIVECOUNT_RET) is not as expected ALIVECOUNT_RET = ALIVECOUNT_SEED - ALIVECOUNT_Q + DEV_COUNT (WRITEALL/READALL) ALIVECOUNT_RET = ALIVECOUNT_SEED - ALIVECOUNT_Q + 1 (WRITEDEVICE/READDEVICE/ READBLOCK) 2) The CLR_ALIVECOUNT_SEED command is executed when the ALIVECOUNT_Q register is non-zero. This bit is not set when the Alive Counter is disabled or set to manual mode.
HW_ERR	0	Hardware Error A hardware error is reported which may affect device operation. Conditions that may cause the assertion of this status bit include: ● Supply voltage or oscillators are detected to be outside of specification. ● Transient logic fault.

STATUS_GEN (R/W = 0x07/0x06)

BIT	7	6	5	4	3	2	1	0
Field	HFOSC_HW_ERRB	DEV_COUNT_ERR	WD_ERR	GPIO_ERR	DATAPATH_ERR	–	ALRTPCKT_COMM_ERR	ALRTPCKTBUF_HW_ERR
Reset	0x1	0x0	0x0	0x0	0x0	–	0x0	0x0
Access Type	Read Only	Read Only	Read Only	Read Only	Read Only	–	Read Only	Read Only

BITFIELD	BITS	DESCRIPTION
HFOSC_HW_ERRB	7	Oscillator Hardware Error Indicates that the high frequency oscillator regularity is not within +/-5% of its expected value when measured against the low frequency oscillator. The status is updated every four low frequency oscillator cycles (64kHz). While it is possible for the SPI interface to continue to function under drift alert conditions, it does not function if the high frequency oscillator is dead or extremely fast/slow. **Note:** This bit can only be cleared by a POR event. **Note:** This status bit is active low and is read back as a logic high when no error is encountered.
DEV_COUNT_ERR	6	Device Count Error This error is asserted when the returned device count from a HELLOALL, UPHOST, or DOWNHOST command does not match the user programmed DEV_COUNT register. This error is cleared if a HELLOALL, UPHOST, or DOWNHOST returns a value that matches the programmed DEV_COUNT register.
WD_ERR	5	Watchdog Error Summary This status bit is asserted simultaneously with any of the following watchdog status errors: WD_TO_ERR WD_LFSR_ERR WD_RJCT_ERR WD_EXP_ERR
GPIO_ERR	4	GPIO HW Error Summary This status bit is asserted simultaneously with any of the following GPIO status errors: GPIO1_ERR GPIO2_ERR GPIO3_ERR GPIO4_ERR

BITFIELD	BITS	DESCRIPTION
DATAPATH_ERR	3	Datapath Error This bit is set when the Verify Config operation encounters an internal UART hardware error or configuration memory hardware/PEC error. This bit is cleared when the Verify Config operation is re-initialized.
ALRTPCKT_COMM_ERR	1	Alert Packet Communication Error This bit is a summary of two conditions that both signify an ALERTPACKET communication error. Condition One = An autogenerated ALERTPACKET decoded PEC does not match the received PEC for a number of times, as programmed by the COMM_RTRY register. In this condition, this bit is cleared when a new autogenerated ALERTPACKET is received with no PEC error. **Note:** When the decoded PEC does not match Received PEC, the ALRTPCKTBUF does not store the received ALERTPACKET data, even if this bit is not set. Condition Two = In BMS Safe Monitoring or Sleep mode, an autogenerated ALERTPACKET encounters a communications timeout as programmed by the COMM_TO_DLY register. In this condition, this bit is cleared when BMS Safe Monitoring or Sleep mode is exited.
ALRTPCKTBUF_HW_ERR	0	The Redundant Alert Packet buffer and Main Alert Packet buffer do not match.

STATUS_OPSTATE (R/W = 0x09/0x08)

BIT	7	6	5	4	3	2	1	0
Field	–	–	–	–	SAFEMON	–	–	–
Reset	–	–	–	–	0x0	–	–	–
Access Type	–	–	–	–	Read Only	–	–	–

BITFIELD	BITS	DESCRIPTION
SAFEMON	3	Safe Monitoring Mode The device has exceeded the number of watchdog response failures and entered BMS Safe Monitoring mode.

STATUS_BUF (R/W = 0x0B/0x0A)

BIT	7	6	5	4	3	2	1	0
Field	ALRTPCKTBUF_FULL	–	–	–	LSSM_FULL	–	–	–
Reset	0x0	–	–	–	0x0	–	–	–
Access Type	Read Only	–	–	–	Read Only	–	–	–

BITFIELD	BITS	DESCRIPTION	DECODE
ALRTPCKTBUF_FULL	7	Alert Packet Buffer is Full The ALRTPCKTBUF_FULL status bit is set when a new ALERTPACKET is received with no PEC error. The data in the ALRTPCKTBUF are cleared to zero after the byte is read. If any given ALERTPACKET byte is read/clear, and a new ALERTPACKET is not received, the ALRTPCKTBUF_FULL register is cleared. In BMS Safe Monitoring and Sleep modes, the ALRTPCKTBUF is cleared automatically, and ALRTPCKTBUF_FULL toggles autonomously.	0 = ALRTPCKTBUF is empty 1 = ALRTPCKTBUF is full
LSSM_FULL	3	LSSM is Full. The LSSM_FULL status bit is set when there are four outstanding messages in the LSSM queue. When the LSSM is full, the Transmit buffer gates additional transactions from being sent to prevent an overflow condition.	0 = LSSM is not full 1 = LSSM is full

STATUS_WD (R/W = 0x0D/0x0C)

BIT	7	6	5	4	3	2	1	0
Field	–	–	–	WD_TO_ERR	WD_OPEN	WD_LFSR_ERR	WD_RJCT_ERR	WD_EXP_ERR
Reset	–	–	–	0x0	0x0	0x0	0x0	0x0
Access Type	–	–	–	Read Only	Read Only	Read Only	Read Only	Read Only

BITFIELD	BITS	DESCRIPTION	DECODE
WD_TO_ERR	4	Watchdog Time Out The number of invalid responses is equal to the number of times programmed into the WD_DBNC register. **Note:** If set, this bit is cleared when WD_EN = 0	
WD_OPEN	3	Watchdog Window is Open	0 = Watchdog window not open. Do not update. 1 = Watchdog window open. Update.
WD_LFSR_ERR	2	Watchdog Invalid Key An invalid key was written to the WD_KEY register during an open window (Extended Open, Open, or Always Open) Cleared on a valid refresh. **Note:** If set, this bit is cleared when WD_EN = 0	0 = LFSR key match 1 = LFSR key mismatch since last read
WD_RJCT_ERR	1	Watchdog Write During closed window Cleared on a valid refresh. **Note:** If set, this bit is cleared when WD_EN = 0	
WD_EXP_ERR	0	Watchdog Expired Cleared on a valid refresh. **Note:** If set, this bit is cleared when WD_EN = 0	0 = Watchdog timer not expired 1 = Watchdog timer expired

STATUS_GPIO (R/W = 0x0F/0x0E)

BIT	7	6	5	4	3	2	1	0
Field	GPIO4_RD	GPIO3_RD	GPIO2_RD	GPIO1_RD	GPIO4_ERR	GPIO3_ERR	GPIO2_ERR	GPIO1_ERR
Reset	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
Access Type	Read Only	Read Only	Read Only	Read Only	Read Only	Read Only	Read Only	Read Only

BITFIELD	BITS	DESCRIPTION
GPIO4_RD	7	Indicates the logic state of the GPIO4 pin. **Note:** During safety measure states of the BMS Safe Monitoring mode, GPIO pin outputs are masked according to the GPIO[n]_MASK register bits, but this register reports the intended output drive value.

BITFIELD	BITS	DESCRIPTION
GPIO3_RD	6	Indicates the logic state of the GPIO3 pin. **Note:** During safety measure states of the BMS Safe Monitoring mode, GPIO pin outputs are masked according to the GPIO[n]_MASK register bits, but this register reports the intended output drive value.
GPIO2_RD	5	Indicates the logic state of the GPIO2 pin. **Note:** During safety measure states of the BMS Safe Monitoring mode, GPIO pin outputs are masked according to the GPIO[n]_MASK register bits, but this register reports the intended output drive value
GPIO1_RD	4	Indicates the logic state of the GPIO1 Pin **Note:** During safety measure states of the BMS Safe Monitoring mode, GPIO pin outputs are masked according to the GPIO[n]_MASK register bits, but this register reports the intended output drive value.
GPIO4_ERR	3	GPIO4 Pin Continuity Error The driven output value does not match the input value.
GPIO3_ERR	2	GPIO3 Pin Continuity Error The driven output value does not match the input value.
GPIO2_ERR	1	GPIO2 Pin Continuity Error The driven output value does not match the input value.
GPIO1_ERR	0	GPIO1 Pin Continuity Error The driven output value does not match the input value.

ALERT_RX (R/W = 0x11/0x10)

BIT	7	6	5	4	3	2	1	0
Field	RX_ERR_ALRT	–	RX_BUSY_ALRT	RX_IDLE_ALRT	RX_OVRFLW_ERR_ALRT	RX_FULL_ALRT	RX_STOP_ALRT	RX_EMPTY_ALRT
Reset	0x0	–	0x0	0x0	0x0	0x0	0x0	0x0
Access Type	Write 0 to Clear, Read	–	Write 0 to Clear, Read	Write 0 to Clear, Read	Write 0 to Clear, Read	Write 0 to Clear, Read	Write 0 to Clear, Read	Write 0 to Clear, Read

BITFIELD	BITS	DESCRIPTION
RX_ERR_ALRT	7	The data byte at location RX_RD_PTR may contain an error (the corresponding character contained a Manchester and/or parity error). Write 0 = Clear Write 1 = No effect **Note:** This register is event driven. If the user attempts a write clear, the register will clear even if the condition persists.
RX_BUSY_ALRT	5	The UART is busy receiving data. Write 0 = Clear Write 1 = No effect **Note:** This register is event driven. If the user attempts a write clear, the register will clear even if the condition persists.
RX_IDLE_ALRT	4	The UART is not receiving data. Write 0 = Clear Write 1 = No effect **Note:** This register is event driven. If the user attempts a write clear, the register will clear even if the condition persists.
RX_OVRFLW_ERR_ALRT	3	The data byte at location RX_WR_POINTER in the Receive buffer is overwritten because it is full. Can be cleared when the Receive buffer is not full (when the buffer is read). Write 0 = Clear Write 1 = No effect **Note:** This register is event driven. If the user attempts a write clear, the register will clear even if the condition persists.
RX_FULL_ALRT	2	The number of empty bytes in the Receive buffer is zero. Write 0 = Clear Write 1 = No effect **Note:** This register is event driven. If the user attempts a write clear, the register will clear even if the condition persists.

BITFIELD	BITS	DESCRIPTION
RX_STOP_ALRT	1	RX Stop Received The UART has finished receiving a properly framed message (stop character) and is ready to be read. The UART clears this bit automatically after all unread messages are read or if the UART detects a new preamble character. The UART does not set this bit if the buffer is empty and it receives a stop character. This bit is not set for automated ALERTPACKET transactions. Write 0 = Clear Write 1 = No effect **Note:** This register is event driven. If the user attempts a write clear, the register will clear even if the condition persists.
RX_EMPTY_ALRT	0	The Receive buffer is cleared and contains no unread data (RX_RD_PTR = RX_WR_PTR - 1). Write 0 = Clear Write 1 = No effect **Note:** This register is event driven. If the user attempts a write clear, the register will clear even if the condition persists.

ALERT_TX (R/W = 0x13/0x12)

BIT	7	6	5	4	3	2	1	0
Field	TX_HELD_ALRT	–	TX_BUSY_ALRT	TX_IDLE_ALRT	TX_OVRFLW_ERR_ALRT	TX_FULL_ALRT	TX_AVAIL_ALRT	TX_EMPTY_ALRT
Reset	0x0	–	0x0	0x0	0x0	0x0	0x0	0x0
Access Type	Write 0 to Clear, Read	–	Write 0 to Clear, Read	Write 0 to Clear, Read	Write 0 to Clear, Read	Write 0 to Clear, Read	Write 0 to Clear, Read	Write 0 to Clear, Read

BITFIELD	BITS	DESCRIPTION
TX_HELD_ALRT	7	The number of empty bytes in the Receive buffer is less than the length of the message in the transmit queue. When this status bit is true and TX_UNLIMITED = 1'b0, the Transmit buffer does not transmit the data in the current LD_Q until the RX buffer has enough available space. When TX_UNLIMITED = 1'b1, the TX_HELD bit is never set, and data is transmitted regardless of available RX buffer space. The RX_OVRFLW_ERR should be monitored to ensure data integrity in this condition. Write 0 = Clear Write 1 = No effect **Note:** This register is event driven. If the user attempts a write clear, the register will clear even if the condition persists.
TX_BUSY_ALRT	5	The UART is busy transmitting data. Write 0 = Clear Write 1 = No effect **Note:** This register is event driven. If the user attempts a write clear, the register will clear even if the condition persists.
TX_IDLE_ALRT	4	The UART is not transmitting data. Write 0 = Clear Write 1 = No effect **Note:** This register is event driven. If the user attempts a write clear, the register will clear even if the condition persists.
TX_OVRFLW_ERR_ALRT	3	LD_Q cannot be incremented because the next queue contained untransmitted data. Any writes in this state overwrite the load queue. Write 0 = Clear Write 1 = No effect **Note:** This register is event driven. If the user attempts a write clear, the register will clear even if the condition persists.

BITFIELD	BITS	DESCRIPTION
TX_FULL_ALRT	2	All queues in the Transmit buffer are full except the load queue (LD_Q = TX_Q - 1). Write 0 = Clear Write 1 = No effect **Note:** This register is event driven. If the user attempts a write clear, the register will clear even if the condition persists.
TX_AVAIL_ALRT	1	One or more queues in the Transmit buffer are available for loading (TX_FULL is false). Write 0 = Clear Write 1 = No effect **Note:** This register is event driven. If the user attempts a write clear, the register will clear even if the condition persists.
TX_EMPTY_ALRT	0	All the queues in the Transmit buffer are cleared and available for loading (LD_Q = TX_Q). Write 0 = Clear Write 1 = No effect **Note:** This register is event driven. If the user attempts a write clear, the register will clear even if the condition persists.

ALERT_LSSM_BYTE (R/W = 0x15/0x14)

BIT	7	6	5	4	3	2	1	0
Field	RX_READY_ALRT	ALRTPCKT_STATUS_ERR_ALRT	COMM_ERR_ALRT	ALRTPCKT_ERR_ALRT	COMM_MSMTCH_ERR_ALRT	COMMAND_OP_ALRT	ALIVECOUNT_ERR_ALRT	HW_ERR_ALRT
Reset	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
Access Type	Write 0 to Clear, Read	Write 0 to Clear, Read	Write 0 to Clear, Read	Write 0 to Clear, Read	Write 0 to Clear, Read	Write 0 to Clear, Read	Write 0 to Clear, Read	Write 0 to Clear, Read

BITFIELD	BITS	DESCRIPTION
RX_READY_ALRT	7	The UART is finished receiving a properly framed message (stop character, preamble, or frame time_out) and it is ready to be read. The UART clears this bit automatically after all unread messages are read. The UART does not set this bit if the buffer is empty and receives a stop character. This bit is not set for automated ALERTPACKET transactions. Write 0 = Clear Write 1 = No effect **Note:** This register is event driven. If the user attempts a write clear, the register will clear even if the condition persists.
ALRTPCKT_STATUS_ERR_ALRT	6	ALERTPACKET Status Error An unmasked Alert Packet status bit is received in the Alert Packet buffer. In Commanded Operation mode, masking of the received ALERTPACKET status bits is a function of the STATUS_ERR_MASK registers. In Sleep or BMS Safe Monitoring mode, masking of the received ALERTPACKET status bits is a function of the STATUS_DBNC_MASK registers and is auto-cleared as the status is debounced. Write 0 = Clear Write 1 = No effect **Note:** This register is event driven. If the user attempts a write clear, the register will clear even if the condition persists.
COMM_ERR_ALRT	5	Communications Error A communication error is observed that has corrupted, inserted, or omitted data within the UART message. Write 0 = Clear Write 1 = No effect **Note:** This register is event driven. If the user attempts a write clear, the register will clear even if the condition persists.

BITFIELD	BITS	DESCRIPTION
ALRTPCKT_ERR_ALRT	4	ALERTPACKET Error This status bit is asserted with any of the following ALERTPACKET status errors: ALRTPCKT_COMM_ERR ALRTPCKT_HW_ERR Write 0 = Clear Write 1 = No effect **Note:** This register is event driven. If the user attempts a write clear, the register will clear even if the condition persists.
COMM_MSMTCH_ERR_ALRT	3	Communication Mismatch Error The LSSM is detected a communication problem where the received data does not match the transmitted data. Write 0 = Clear Write 1 = No effect **Note:** This register is event driven. If the user attempts a write clear, the register will clear even if the condition persists.
COMMAND_OP_ALRT	2	Commanded Operation Mode Alert This bit represents Commanded Operation mode (device has not entered Sleep or BMS Safe Monitoring mode). If this bit is not a logical 1, then the device has entered into another operational mode. This status bit is asserted as follows = !(SLP_EN || SAFEMON_ALRT) Write 0 = Clear Write 1 = No effect **Note:** This register is event driven. If the user attempts a write clear, the register clears even if the condition persists.

BITFIELD	BITS	DESCRIPTION
ALIVECOUNT_ERR_ALRT	1	Automatic Alive Counter Error When the Automatic Alive Counter mode is enabled, this bit is asserted in two scenarios. 1) The received ALIVECOUNT byte (ALIVECOUNT_RET) is not as expected ALIVECOUNT_RET = ALIVECOUNT_SEED - ALIVECOUNT_Q + DEV_COUNT (WRITEALL/READALL) ALIVECOUNT_RET = ALIVECOUNT_SEED - ALIVECOUNT_Q + 1 (WRITEDEVICE/READDEVICE/ READBLOCK) 2) The CLR_ALIVECOUNT_SEED is set when the ALIVECOUNT_Q register is non-zero. This bit is not set when the Alive Counter is disabled or set to manual mode. Write 0 = Clear Write 1 = No effect **Note:** This register is event driven. If the user attempts a write clear, the register will clear even if the condition persists.
HW_ERR_ALRT	0	Hardware Error A hardware error is reported that may affect device operation. Conditions that may cause the assertion of this status bit include: ● Supply voltage or oscillators are detected to be outside of specification. ● Transient logic fault. Write 0 = Clear Write 1 = No effect **Note:** This register is persistent. If the user attempts a write to clear, and the corresponding STATUS register remains set, this bit remains set.

ALERT_GEN (0xB)

(R/W = 0x17/0x16)

BIT	7	6	5	4	3	2	1	0
Field	–	DEV_COUNT_ERR_ALRT	WD_ERR_ALRT	GPIO_ERR_ALRT	DATAPATH_ERR_ALRT	SPI_ERR_ALRT	ALRTPCKT_COMM_ERR_ALRT	ALRTPCKTBUF_HW_ERR_ALRT
Reset	–	0x0	0x0	0x0	0x0	0x0	0x0	0x0
Access Type	–	Write 0 to Clear, Read	Write 0 to Clear, Read	Write 0 to Clear, Read	Write 0 to Clear, Read	Write 0 to Clear, Read	Write 0 to Clear, Read	Write 0 to Clear, Read

BITFIELD	BITS	DESCRIPTION
DEV_COUNT_ERR_ALRT	6	Device Count Error This error is asserted when the returned device count from a HELLOALL, UPHOST, or DOWNHOST command does not match the user programmed DEV_COUNT register. Write 0 = Clear Write 1 = No effect **Note:** This register is persistent. If the user attempts a write to clear, and the corresponding STATUS register remains set, then this bit remains set.
WD_ERR_ALRT	5	Watchdog Error Summary This status bit is asserted with any of the following watchdog status errors: WD_TO_ERR WD_LFSR_ERR WD_RJCT_ERR WD_EXP_ERR Write 0 = Clear Write 1 = No effect **Note:** This register is persistent. If the user attempts a write to clear, and the corresponding STATUS register remains set, then this bit remains set.
GPIO_ERR_ALRT	4	GPIO HW Error Summary This status bit is asserted simultaneously with any of the following GPIO status errors: GPIO1_ERR GPIO2_ERR GPIO3_ERR GPIO4_ERR Write 0 = Clear Write 1 = No effect **Note:** This register is persistent. If the user attempts a write to clear, and the corresponding STATUS register remains set, then this bit remains set.

BITFIELD	BITS	DESCRIPTION
DATAPATH_ERR_ALRT	3	Datapath Error Write 0 = Clear Write 1 = No effect **Note:** This register is persistent. If the user attempts a write to clear, and the corresponding STATUS register remains set, then this bit remains set.
SPI_ERR_ALRT	2	The user attempts to read or write a restricted register while the Sleep, Verify Config Memory, or Load Config Memory processes are active, or while in BMS Safe Monitoring mode. Attempting such an action results in SPI_ERR being set. Write 0 = Clear Write 1 = No effect **Note:** This register is event driven. If the user attempts a write clear, the register clears even if the condition persists.
ALRTPCKT_COMM_ERR_ALRT	1	Alert Packet Communication Error Write 0 = Clear Write 1 = No effect **Note:** This register is persistent. If the user attempts a write to clear, and the corresponding STATUS register remains set, then this bit remains set.
ALRTPCKTBUF_HW_ERR_ALRT	0	The Redundant Alert Packet buffer and Main Alert Packet buffer do not match. Write 0 = Clear Write 1 = No effect **Note:** This register is persistent. If the user attempts a write to clear, and the corresponding STATUS register remains set, then this bit remains set.

ALERT_OPSTATE (R/W = 0x19/0x18)

BIT	7	6	5	4	3	2	1	0
Field	ALRTRST	–	SLP_ALRT	SLP_STATUS_ERR_ALRT	SAFEMON_ALRT	SAFEMON_GPIO12_ALRT	SAFEMON_STATUS_ERR_ALRT	SAFEMON_CONFIG_ERR_ALRT
Reset	0x1	–	0x0	0x0	0x0	0x0	0x0	0x0
Access Type	Write 0 to Clear, Read	–	Write 0 to Clear, Read	Write 0 to Clear, Read	Write 0 to Clear, Read	Write 0 to Clear, Read	Write 0 to Clear, Read	Write 0 to Clear, Read

BITFIELD	BITS	DESCRIPTION
ALRTRST	7	Interrupt flag for POR/Reset Alert Indicates a power-on-reset event occurred. Users should clear this alert immediately after power-on in order to detect future resets. This bit is non-maskable. Write 0 = Clear Write 1 = No effect **Note:** This register is not maskable and has no corresponding ALRTEN bit.
SLP_ALRT	5	Device has woken up from sleep. Write 0 = Clear Write 1 = No effect **Note:** This register is event driven. If the user attempts a write clear, the register clears even if the condition persists.
SLP_STATUS_ERR_ALRT	4	Sleep Mode Status Error During Sleep mode, an unmasked STATUS bit was indicated in the Alert Packet buffer equal to the number of times programmed into the STATUS_DBNC register, causing the device to exit Sleep mode. Masking of the received ALERTPACKET status bits is a function of the STATUS_DBNC_MASK registers. Clearing this register resets the FAULT timer. This bit is unaffected by the ALRTRST register. Write 0 = Clear Write 1 = No effect **Note:** This register is event driven. If the user attempts a write clear, the register clears even if the condition persists.

BITFIELD	BITS	DESCRIPTION
SAFEMON_ALRT	3	Safe Monitoring Mode Alert The device has exceeded the number of watchdog response failures and entered BMS Safe Monitoring mode. Write 0 = Clear Write 1 = No effect Clearing this register resets the FAULT timer. This bit is unaffected by the ALRTRST register. **Note:** This bit cannot be cleared if the device is in BMS Safe Monitoring mode.
SAFEMON_GPIO12_ALRT	2	SAFEMON_GPIO12_ALRT The GPIO1/2 pins are asserted in Safe Monitoring mode and remain asserted upon exit into Commanded Operation mode until this bit is cleared. Write 0 = Clear Write 1 = No effect **Note:** This bit is unaffected by the ALRTRST register. **Note:** This bit toggles autonomously during the safety measure states of BMS Safe Monitoring mode. **Note:** This bit cannot be cleared by the user if the device is in BMS Safe Monitoring mode.
SAFEMON_STATUS_ERR_ALRT	1	Safe Monitoring Mode Status Error During Safe Monitoring mode, an unmasked STATUS bit is indicated in the Alert Packet buffer equal to the number of times programmed into the STATUS_DBNC register. Masking of the received ALERTPACKET status bits is a function of the STATUS_DBNC_MASK registers. Write 0 = Clear Write 1 = No effect **Note:** This bit cannot be cleared if the device is in BMS Safe Monitoring mode.

BITFIELD	BITS	DESCRIPTION
SAFEMON_CONFIG_ERR_ALRT	0	Safe Monitoring Mode Configuration Error This status bit is set when the Verify Configuration Memory process or Load Configuration Memory process encounters an error during execution. Write 0 = Clear Write 1 = No effect **Note:** This bit cannot be cleared if the device is in BMS Safe Monitoring mode.

ALERT_BUF (R/W = 0x1B/0x1A)

BIT	7	6	5	4	3	2	1	0
Field	ALRTPCKTBUF_FULL_ALRT	–	–	–	LSSM_FULL_ALRT	–	–	–
Reset	0x0	–	–	–	0x0	–	–	–
Access Type	Write 0 to Clear, Read	–	–	–	Write 0 to Clear, Read	–	–	–

BITFIELD	BITS	DESCRIPTION	DECODE
ALRTPCKTBUF_FULL_ALRT	7	Alert Packet Buffer is Full The ALRTPCKTBUF_FULL status bit is set when a new ALERTPACKET is received with no PEC error. The data in the ALRTPCKTBUF are cleared to zero after the byte is read. Write 0 = Clear Write 1 = No effect **Note:** This register is persistent. If the user attempts a write to clear, and the corresponding STATUS register remains set, then this bit remains set.	0 = ALRTPCKTBUF is empty 1 = ALRTPCKTBUF is full

BITFIELD	BITS	DESCRIPTION	DECODE
LSSM_FULL_ALRT	3	LSSM is Full. The LSSM_FULL status bit is set when there are four outstanding messages in the LSSM Queue. When the LSSM is full, the Transmit buffer gates additional transactions from being sent to prevent an overflow condition. Write 0 = Clear Write 1 = No effect **Note:** This register is persistent. If the user attempts a write to clear, and the corresponding STATUS register remains set, then this bit remains set.	0 = LSSM is not full 1 = LSSM is full

ALERT_WD (R/W = 0x1D/0x1C)

BIT	7	6	5	4	3	2	1	0
Field	–	–	–	WD_TO_ERR_ALRT	WD_OPEN_ALRT	WD_LFSR_ERR_ALRT	WD_RJCT_ERR_ALRT	WD_EXP_ERR_ALRT
Reset	–	–	–	0x0	0x0	0x0	0x0	0x0
Access Type	–	–	–	Write 0 to Clear, Read	Write 0 to Clear, Read	Write 0 to Clear, Read	Write 0 to Clear, Read	Write 0 to Clear, Read

BITFIELD	BITS	DESCRIPTION	DECODE
WD_TO_ERR_ALRT	4	Watchdog Time Out The number of invalid responses is equal to the number of times programmed into the WD_DBNC register. Write 0 = Clear Write 1 = No effect **Note:** This register is persistent. If the user attempts a write clear, and the condition persists, this bit remains set.	
WD_OPEN_ALRT	3	Watchdog Window is Open Write 0 = Clear Write 1 = No effect **Note:** This register is event driven. If the user attempts a write clear, the register will clear, even if the condition persists.	0 = Watchdog window not open. Do not update. 1 = Watchdog window open. Update.

BITFIELD	BITS	DESCRIPTION	DECODE
WD_LFSR_ERR_ALRT	2	Watchdog Invalid Key An invalid key is written to the WD_KEY register during an open window (Extended Open, Open, or Always Open) Write 0 = Clear Write 1 = No effect **Note:** This register is persistent. If the user attempts a write to clear, and the corresponding STATUS register remains set, then this bit remains set.	0 = LFSR key match. 1 = LFSR key mismatch since last read.
WD_RJCT_ERR_ALRT	1	Watchdog Write During closed window Cleared when a valid key is written during an open window **Note:** This register is persistent. If the user attempts a write to clear, and the corresponding STATUS register remains set, then this bit remains set.	
WD_EXP_ERR_ALRT	0	Watchdog Expired Write 0 = Clear Write 1 = No effect **Note:** This register is event driven. If the user attempts a write clear, the register will clear, even if the condition persists.	0 = Watchdog timer not expired. 1 = Watchdog timer expired.

ALERT_GPIO (R/W = 0x1F/0x1E)

BIT	7	6	5	4	3	2	1	0
Field	–	–	–	–	GPIO4_ERR_ALRT	GPIO3_ERR_ALRT	GPIO2_ERR_ALRT	GPIO1_ERR_ALRT
Reset	–	–	–	–	0x0	0x0	0x0	0x0
Access Type	–	–	–	–	Write 0 to Clear, Read	Write 0 to Clear, Read	Write 0 to Clear, Read	Write 0 to Clear, Read

BITFIELD	BITS	DESCRIPTION
GPIO4_ERR_ALRT	3	GPIO4 Pin Continuity Error The driven output value does not match the input value. Write 0 = Clear Write 1 = No effect **Note:** This register is persistent. If the user attempts a write to clear, and the corresponding STATUS register remains set, then this bit remains set.
GPIO3_ERR_ALRT	2	GPIO3 Pin Continuity Error The driven output value does not match the input value. Write 0 = Clear Write 1 = No effect **Note:** This register is persistent. If the user attempts a write to clear, and the corresponding STATUS register remains set, then this bit remains set.
GPIO2_ERR_ALRT	1	GPIO2 Pin Continuity Error The driven output value does not match the input value. Write 0 = Clear Write 1 = No effect **Note:** This register is persistent. If the user attempts a write to clear, and the corresponding STATUS register remains set, then this bit remains set.
GPIO1_ERR_ALRT	0	GPIO1 Pin Continuity Error The driven output value does not match the input value. Write 0 = Clear Write 1 = No effect **Note:** This register is persistent. If the user attempts a write to clear, and the corresponding STATUS register remains set, then this bit remains set.

ALRTEN_RX (R/W = 0x21/0x20)

BIT	7	6	5	4	3	2	1	0
Field	RX_ERR_ALRTEN	–	RX_BUSY_ALRTEN	RX_IDLE_ALRTEN	RX_OVRFLW_ERR_ALRTEN	RX_FULL_ALRTEN	RX_STOP_ALRTEN	RX_EMPTY_ALRTEN
Reset	0x0	–	0x0	0x0	0x0	0x0	0x0	0x0
Access Type	Write, Read	–	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read

BITFIELD	BITS	DESCRIPTION
RX_ERR_ALRTEN	7	$\overline{\text{ALERT}}$ Pin Enable
RX_BUSY_ALRTEN	5	$\overline{\text{ALERT}}$ Pin Enable
RX_IDLE_ALRTEN	4	$\overline{\text{ALERT}}$ Pin Enable
RX_OVRFLW_ERR_ALRTEN	3	$\overline{\text{ALERT}}$ Pin Enable
RX_FULL_ALRTEN	2	$\overline{\text{ALERT}}$ Pin Enable
RX_STOP_ALRTEN	1	$\overline{\text{ALERT}}$ Pin Enable
RX_EMPTY_ALRTEN	0	$\overline{\text{ALERT}}$ Pin Enable

ALRTEN_TX (R/W = 0x1/0x0)

BIT	7	6	5	4	3	2	1	0
Field	TX_HELD_ALRTEN	–	TX_BUSY_ALRTEN	TX_IDLE_ALRTEN	TX_OVRFLW_ERR_ALRTEN	TX_FULL_ALRTEN	TX_AVAIL_ALRTEN	TX_EMPTY_ALRTEN
Reset	0x0	–	0x0	0x0	0x0	0x0	0x0	0x0
Access Type	Write, Read	–	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read

BITFIELD	BITS	DESCRIPTION
TX_HELD_ALRTEN	7	$\overline{\text{ALERT}}$ Pin Enable
TX_BUSY_ALRTEN	5	$\overline{\text{ALERT}}$ Pin Enable
TX_IDLE_ALRTEN	4	$\overline{\text{ALERT}}$ Pin Enable
TX_OVRFLW_ERR_ALRTEN	3	$\overline{\text{ALERT}}$ Pin Enable
TX_FULL_ALRTEN	2	$\overline{\text{ALERT}}$ Pin Enable
TX_AVAIL_ALRTEN	1	$\overline{\text{ALERT}}$ Pin Enable
TX_EMPTY_ALRTEN	0	$\overline{\text{ALERT}}$ Pin Enable

ALRTEN_LSSM_BYTE (R/W = 0x25/0x24)

BIT	7	6	5	4	3	2	1	0
Field	RX_READY_ALRTEN	ALRTPCKT_STATUS_ERR_ALRTEN	COMM_ERR_ALRTEN	ALRTPCKT_ERR_ALRTEN	COMM_MSMTCH_ERR_ALRTEN	COMMAND_OP_ALRTEN	ALIVECOUNT_ERR_ALRTEN	HW_ERR_ALRTEN
Reset	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x1
Access Type	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read

BITFIELD	BITS	DESCRIPTION
RX_READY_ALRTEN	7	$\overline{\text{ALERT}}$ Pin Enable
ALRTPCKT_STATUS_ERR_ALRTEN	6	$\overline{\text{ALERT}}$ Pin Enable

BITFIELD	BITS	DESCRIPTION
COMM_ERR_ALRTEN	5	$\overline{\text{ALERT}}$ Pin Enable
ALRTPCKT_ERR_ALRTEN	4	$\overline{\text{ALERT}}$ Pin Enable
COMM_MSMTCH_ERR_ALRTEN	3	$\overline{\text{ALERT}}$ Pin Enable
COMMAND_OP_ALRTEN	2	$\overline{\text{ALERT}}$ Pin Enable
ALIVECOUNT_ERR_ALRTEN	1	$\overline{\text{ALERT}}$ Pin Enable
HW_ERR_ALRTEN	0	$\overline{\text{ALERT}}$ Pin Enable

ALRTEN_GEN (R/W = 0x27/0x26)

BIT	7	6	5	4	3	2	1	0
Field	–	DEV_COUNT_ERR_ALRTEN	WD_ERR_ALRTEN	GPIO_ERR_ALRTEN	DATAPATH_ERR_ALRTEN	SPI_ERR_ALRTEN	ALRTPCKT_COMM_ERR_ALRTEN	ALRTPCKTBUF_HW_ERR_ALRTEN
Reset	–	0x0	0x1	0x1	0x0	0x1	0x0	0x0
Access Type	–	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read

BITFIELD	BITS	DESCRIPTION
DEV_COUNT_ERR_ALRTEN	6	$\overline{\text{ALERT}}$ Pin Enable
WD_ERR_ALRTEN	5	$\overline{\text{ALERT}}$ Pin Enable
GPIO_ERR_ALRTEN	4	$\overline{\text{ALERT}}$ Pin Enable
DATAPATH_ERR_ALRTEN	3	$\overline{\text{ALERT}}$ Pin Enable
SPI_ERR_ALRTEN	2	
ALRTPCKT_COMM_ERR_ALRTEN	1	
ALRTPCKTBUF_HW_ERR_ALRTEN	0	

ALRTEN_OPSTATE (R/W =0x29/0x28)

BIT	7	6	5	4	3	2	1	0
Field	–	–	SLP_ALRTEN	SLP_STATUS_ERR_ALRTEN	SAFEMON_ALRTEN	SAFEMON_GPIO12_ALRTEN	SAFEMON_STATUS_ERR_ALRTEN	SAFEMON_CONFIG_ERR_ALRTEN
Reset	–	–	0x1	0x1	0x1	0x1	0x1	0x1
Access Type	–	–	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read

BITFIELD	BITS	DESCRIPTION
SLP_ALRTEN	5	$\overline{\text{ALERT}}$ Pin Enable
SLP_STATUS_ERR_ALRTEN	4	$\overline{\text{ALERT}}$ Pin Enable
SAFEMON_ALRTEN	3	$\overline{\text{ALERT}}$ Pin Enable
SAFEMON_GPIO12_ALRTEN	2	$\overline{\text{ALERT}}$ Pin Enable
SAFEMON_STATUS_ERR_ALRTEN	1	$\overline{\text{ALERT}}$ Pin Enable
SAFEMON_CONFIG_ERR_ALRTEN	0	$\overline{\text{ALERT}}$ Pin Enable

ALRTEN_BUF (R/W = 0x2B/0x2A)

BIT	7	6	5	4	3	2	1	0
Field	ALRTPCKTBUF_FULL_ALRTEN	–	–	–	LSSM_FULL_ALRTEN	–	–	–
Reset	0x0	–	–	–	0x0	–	–	–
Access Type	Write, Read	–	–	–	Write, Read	–	–	–

BITFIELD	BITS	DESCRIPTION
ALRTPCKTBUF_FULL_ALRTEN	7	$\overline{\text{ALERT}}$ Pin Enable
LSSM_FULL_ALRTEN	3	$\overline{\text{ALERT}}$ Pin Enable

ALRTEN_WD (R/W = 0x2D/0x2C)

BIT	7	6	5	4	3	2	1	0
Field	–	–	–	WD_TO_ERR_ALRTEN	WD_OPEN_ALRTEN	WD_LFSR_ERR_ALRTEN	WD_RJCT_ERR_ALRTEN	WD_EXP_ERR_ALRTEN
Reset	–	–	–	0x0	0x0	0x0	0x0	0x0
Access Type	–	–	–	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read

BITFIELD	BITS	DESCRIPTION
WD_TO_ERR_ALRTEN	4	$\overline{\text{ALERT}}$ Pin Enable
WD_OPEN_ALRTEN	3	$\overline{\text{ALERT}}$ Pin Enable
WD_LFSR_ERR_ALRTEN	2	$\overline{\text{ALERT}}$ Pin Enable
WD_RJCT_ERR_ALRTEN	1	$\overline{\text{ALERT}}$ Pin Enable
WD_EXP_ERR_ALRTEN	0	$\overline{\text{ALERT}}$ Pin Enable

ALRTEN_GPIO (R/W = 0x2F/0x2E)

BIT	7	6	5	4	3	2	1	0
Field	–	–	–	–	GPIO4_ERR_ALRTEN	GPIO3_ERR_ALRTEN	GPIO2_ERR_ALRTEN	GPIO1_ERR_ALRTEN
Reset	–	–	–	–	0x0	0x0	0x0	0x0
Access Type	–	–	–	–	Write, Read	Write, Read	Write, Read	Write, Read

BITFIELD	BITS	DESCRIPTION
GPIO4_ERR_ALRTEN	3	$\overline{\text{ALERT}}$ Pin Enable
GPIO3_ERR_ALRTEN	2	$\overline{\text{ALERT}}$ Pin Enable
GPIO2_ERR_ALRTEN	1	$\overline{\text{ALERT}}$ Pin Enable
GPIO1_ERR_ALRTEN	0	$\overline{\text{ALERT}}$ Pin Enable

CLR_TXBUF (R/W = 0x41/0x40)

CLR_TXBUF Command

Resets the Transmit buffer to its default state and clears TX_Q and LD_Q.

Note: This command cannot be written while the Sleep, Verify Config Memory, or Load Config Memory processes are active or while in BMS Safe Monitoring mode. Attempting such an action results in the SPI_ERR_ALRT bit being set.

CLR_RXBUF (R/W = 0x43/0x42)

CLR_RXBUF Command

Resets the Receive buffer to its default state.

Note: This command cannot be written while the Sleep, Verify Config Memory, or Load Config Memory processes are active or while in BMS Safe Monitoring mode. Attempting such an action results in the SPI_ERR_ALRT bit being set.

CLR_LSSM (R/W = 0x45/0x44)

CLR_LSSM Command

Resets the LSSM.

Note: This command cannot be written while the Sleep, Verify Config Memory, or Load Config Memory processes are active or while in BMS Safe Monitoring mode. Attempting such an action results in the SPI_ERR_ALRT bit being set.

CLR_ALIVECOUNT_SEED (R/W = 0x49/0x48)

Clears the ALIVECOUNT_SEED register.

Ignored if the ALIVECOUNT_Q register is non-zero and flags the ALIVECOUNT_ERR status bit.

Note: This command cannot be written while the Sleep, Verify Config Memory, or Load Config Memory processes are active or while in BMS Safe Monitoring mode. Attempting such an action results in the SPI_ERR_ALRT bit being set.

SWPOR (R/W = 0x4B/0x4A)

BIT	7	6	5	4	3	2	1	0
Field	–	–	–	–	–	–	–	SWPOR
Reset	–	–	–	–	–	–	–	0b0
Access Type	–	–	–	–	–	–	–	Write, Read, Pulse

BITFIELD	BITS	DESCRIPTION	DECODE
SWPOR	0	Software POR Command This bit completely resets the device, emulating a POR condition.	0 = Normal operation (default, no effect) 1 = Initiates software POR event

SLP_EN (R/W = 0x4D/0x4C)

BIT	7	6	5	4	3	2	1	0
Field	–	–	–	–	–	–	–	SLP_EN
Reset	–	–	–	–	–	–	–	0b0
Access Type	–	–	–	–	–	–	–	Write, Read, Ext

BITFIELD	BITS	DESCRIPTION	DECODE
SLP_EN	0	Sleep Enable Command This bit is auto-cleared on exit from Sleep mode. The host can also exit Sleep mode by manually clearing this bit. **Note:** This command cannot be written while the Verify Config Memory or Load Config Memory processes are active or while in BMS Safe Monitoring mode. Attempting such an action results in the SPI_ERR_ALRT bit being set.	0 = Normal operation 1 = Sleep mode

VER_CONFIG (R/W = 0x4F/0x4E)

BIT	7	6	5	4	3	2	1	0
Field	–	–	–	–	–	–	–	VER_CONFIG
Reset	–	–	–	–	–	–	–	0b0
Access Type	–	–	–	–	–	–	–	Write, Read, Ext

BITFIELD	BITS	DESCRIPTION	DECODE
VER_CONFIG	0	Verify Configuration Memory Command Failure of this operation will be indicated by the DATAPATH_ERR or SAFEMON_CONFIG_ERR_ALRT bits. **Note:** This command cannot be written while the Sleep or Load Config Memory processes are active or while in BMS Safe Monitoring mode. Attempting such an action results in the SPI_ERR_ALRT bit being set.	0 = Normal operation (default, no effect) 1 = Executes the Verify Configuration Memory command

LOAD_CONFIG (R/W = 0x51/0x50)

BIT	7	6	5	4	3	2	1	0
Field	–	–	–	–	–	–	–	LOAD_CONFIG
Reset	–	–	–	–	–	–	–	0b0
Access Type	–	–	–	–	–	–	–	Write, Read, Ext

BITFIELD	BITS	DESCRIPTION	DECODE
LOAD_CONFIG	0	Load Configuration Memory Command Failure of this operation is indicated by the SAFEMON_CONFIG_ERR_ALRT bit. **Note:** This command cannot be written while the Verify Config Memory or Sleep processes are active or while in BMS Safe Monitoring mode. Attempting such an action results in the SPI_ERR_ALRT bit being set.	0 = Normal operation (default, no effect) 1 = Executes the Load Configuration Memory command using Configuration Memory Queue 0-2

WD_KEY (R/W = 0x53/0x52)

Watchdog Key Register (Read/Write)

BIT	7	6	5	4	3	2	1	0
Field	WD_KEY[7:0]							
Reset	0xAA							
Access Type	Write, Read, Ext							

BITFIELD	BITS	DESCRIPTION
WD_KEY	7:0	The watchdog key. The current key can be read from this register. To update the watchdog, the next value in the sequence must be written to this register. If configured as a simple windowed watchdog, writing any value to the WD_KEY register refreshes the watchdog, and the value written is ignored. If configured as a challenge/response watchdog, writing the incorrect response to the WD_KEY register will result in the value written being ignored and a WD_LFSR violation. Writing the correct response in challenge/response mode during an open window results in a refresh, and the WD_KEY register is updated. Writing the correct response in challenge/response mode during a closed window will result in the write being ignored and a WD_UV violation. LFSR polynomial = $x^8 + x^6 + x^3 + x^2 + 1$

CONFIG_GEN0 (R/W = 0x61/0x60)

General Configuration Register (Read/Write)

BIT	7	6	5	4	3	2	1	0
Field	–	–	DEV_COUNT[5:0]					
Reset	–	–	0x0					
Access Type	–	–	Write, Read					

BITFIELD	BITS	DESCRIPTION
DEV_COUNT	5:0	Device Count register This register should be programmed by the user with the number of devices in the daisy-chain. This register is used with the automated Alive Count and data check parser features to ensure daisy-chain integrity. **Note:** This register should not be confused with the device address of the daisy-chain devices.

CONFIG_GEN1 (R/W = 0x63/0x62)

General Configuration Register (Read/Write)

BIT	7	6	5	4	3	2	1	0
Field	SINGLE_ENDED	BAUD_RATE[2:0]			–	–	–	–
Reset	0x0	0b011			–	–	–	–
Access Type	Write, Read	Write, Read			–	–	–	–

BITFIELD	BITS	DESCRIPTION	DECODE
SINGLE_ENDED	7	Enables the UART to receive a single-ended signal by shifting the input threshold negative (zero differential voltage is a logic one). In this mode, the RXP input should be connected to ground and receive the inverted signal, which is the same when in Differential mode. In this mode, the TX port operates the same as it does in Differential mode also. Default is Differential mode.	0 = Normal operation 1 = Enables single-ended signal operation
BAUD_RATE	6:4	UART Baud Rate **Note:** Programming this register while the UART is transmitting or receiving data can cause unexpected operation.	000 = 500kbps 001 = 500kbps 010 = 1Mbps 011 = 2Mbps (default) 100 = 4Mbps 101-111 = Reserved Writes to reserved values default to 500kbps.

CONFIG_GEN2 (R/W = 0x65/0x64)

General Configuration Register (Read/Write)

BIT	7	6	5	4	3	2	1	0
Field	RX_RAW_DATA	TX_RAW_DATA	TX_PREAMBLES	TX_QUEUE	TX_ODDPARITY	TX_PAUSE	TX_NOSTOP	TX_NOPREAMBLE
Reset	0x0	0x0	0x0	0x1	0x0	0x0	0x0	0x0
Access Type	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read

BITFIELD	BITS	DESCRIPTION	DECODE
RX_RAW_DATA	7	Receive Raw Data Mode	0 = Normal operation 1 = Disables Manchester decoding of the received data. In this mode, there is one data byte stored for every one character received (instead of every two received).
TX_RAW_DATA	6	Transmit Raw Data Mode	0 = Normal operation 1 = Disables Manchester encoding of transmitted data. In this mode, each data byte is transmitted as one character (instead of two characters).
TX_PREAMBLES	5	Transmit Preambles Mode	0 = Normal operation 1 = Transmits preambles continuously. This mode takes precedence over all transmit modes except Transmit Pause mode.
TX_QUEUE	4	Transmit Queue Mode **Note:** TX_UNLIMITED applies only to TX_QUEUE Mode	0 = TX Queue message not transmitted but UART could transmit based on TX_PREAMBLES and TX_PAUSE 1 = Enables transmission of the message loaded in the Transmit queue if 1) there is sufficient space in the Receive buffer for the message (RX_FULL is false) OR 2) the limitations on message length are removed (TX_UNLIMITED is set).
TX_ODDPARITY	3	Transmit Odd Parity Mode Transmits characters with odd parity. Since the UART protocol uses even parity, this mode can be used to test the system's ability to detect parity errors. Even parity is default.	0 = Normal operation 1 = Enables Odd Parity mode

BITFIELD	BITS	DESCRIPTION	DECODE
TX_PAUSE	2	Transmit Pause Mode **Note:** This mode takes precedence over all other transmit modes (Transmit Preambles, Transmit Queue, and Keep-Alive modes).	0 = Normal operation 1 = Places the transmitter into idle state once the UART is finished transmitting the current byte; however, the TX_BUSY and TX_IDLE bits remain unchanged. Transmission resumes when this bit is cleared.
TX_NOSTOP	1	Transmit No Stop Mode Transmits messages without a stop character. By sending subsequent messages with the TX_NOPREAMBLE bit set, a framed message of indefinite length can be constructed. The TX_UNLIMITED bit must be set for messages greater than 86 bytes.	0 = Stop character enabled 1 = Stop character disabled
TX_NOPREAMBLE	0	Transmit No Preamble Mode Transmits messages without a preamble. By first sending a message in which the TX_NOSTOP bit is set, then sending messages with this bit set, a framed message of indefinite length can be constructed. However, if the preceding message is terminated with a stop character (end of frame), then the data sent in this mode is unframed (no preamble) and not stored in the Receive buffer.	0 = Normal operation 1 = Enables Preamble mode

CONFIG_GEN3 (R/W = 0x67/0x66)

General Configuration Register (Read/Write)

BIT	7	6	5	4	3	2	1	0
Field	–	TX_AUTO	TX_UNLIMITED	SPI_DOUT_EN	ALRTPCKT_TIMING[3:0]			
Reset	–	0x0	0x0	0x0	0b1111			
Access Type	–	Write, Read	Write, Read	Write, Read	Write, Read			

BITFIELD	BITS	DESCRIPTION	DECODE
TX_AUTO	6	Transmit Automatic Mode In this mode, the UART automatically executes a NXT_LDQ command upon receipt of a preamble. This mode is intended for in the loop testing and is not operational during Sleep, Safemon, Load Config, or Verify Config processes.	0 = Transmit Automatic mode disabled 1 = Transmit Automatic mode enabled
TX_UNLIMITED	5	Transmit Unlimited Mode In this mode, the transmit queue automatically limits the message length to 255 bytes instead of the default 86-byte limit, and the message transmission is permitted, even if the message length is greater than the available write space in the Receive buffer.	0 = Normal operation 1 = Enables Transmit Unlimited mode
SPI_DOUT_EN	4	SPI Output Enable	0 = DOUT pin is tri-stated when CSB is deasserted (default) 1 = DOUT pin is always driven

BITFIELD	BITS	DESCRIPTION	DECODE
ALRTPCKT_TIMING	3:0	Alert Packet/Keep-Alive Timing This register controls the timing of the Keep-Alive (stop) character generation or Alert Packet generation in Commanded Operation mode. When CO_ALRTPCKTEN = 1, Alert Packets are generated. When CO_ALRTPCKTEN = 0, keep alive characters are generated. Keep-Alive prevents slave devices from shutting down during periods of no communication (idle state). The idle time in between the periodic stop characters is based on the 4-bit value below. The default setting is infinite (mode disabled). Note: The Transmit Pause, Transmit Preambles, and the Transmit Queue modes take precedence over this mode. **Note:** In Sleep or BMS Safe Monitoring mode, Alert Packets are generated according to SLP_SCAN_DLY or SAFEMON_SCAN_DLY respectively. **Note:** The recommended Alert Packet timing rate is 1.28m (default). **Note:** Keep-Alive settings greater than 5.12ms may not prevent the devices in the daisy-chain from shutting down. **Note:** When configured for Alert Packet generation, timing settings lower than 320us default to 320us.	0000 = 0µs (Continuous) 0001 = 10µs 0010 = 20µs 0011 = 40µs 0100 = 80µs 0101 = 160µs 0110 = 320µs (Alert Packet minimum Timing) 0111 = 640µs 1000 = 1.28ms 1001 = 2.56ms 1010 = 5.12ms 1011 = 10.24ms 1111 = Infinite delay/disabled (default)

CONFIG_GEN4 (R/W = 0x69/0x68)

General Configuration Register (Read/Write)

BIT	7	6	5	4	3	2	1	0
Field	CO_ALRTPCKTEN	RXSWAP_EN	MS_EN[1:0]		DC_EN[1:0]		ALIVECOUNT_EN[1:0]	
Reset	0x0	0b0	0b10		0b10		0x0	
Access Type	Write, Read	Write, Read	Write, Read		Write, Read		Write, Read	

BITFIELD	BITS	DESCRIPTION	DECODE
CO_ALRTPCKTEN	7	Commanded Operation Alert Packet Enable	0 = Stop characters are generated according to the ALRTPCKT_TIMING register setting. 1 = Alert Packets are generated according to the ALRTPCKT_TIMING register setting.
RXSWAP_EN	6	Receive Buffer Data Swap Enable This bit reverses the order of RX buffer data readback by the MAX17851.	0 = Receive data is reported normally. 1 = Receive data is reversed.
MS_EN	5:4	Master Enable	0x = Slave, dual UART 10 = Master, single UART (default) 11 = Master, dual UART
DC_EN	3:2	Data Check Byte Enable When enabled, the Data Check byte must be generated by the host. The MAX17851 automatically parses incoming Data Check bytes and reports the returned daisy-chain $\overline{\text{DC\_PECERR}}$ via the $\overline{\text{ALERT}}$ pin and LSSM byte. No other bits from the Data Check byte are reported. All other Data Check byte errors are reported in ALRTPCKTBUF data. When the Data Check byte is enabled, the user has the option to either store it or not in the RX buffer.	0x = Data Check byte is disabled. 10 = Data Check byte enabled and stored in the RX buffer (default). 11 = Data Check byte is enabled and not stored in the RX buffer.

BITFIELD	BITS	DESCRIPTION	DECODE
ALIVECOUNT_EN	1:0	Alive Counter Enable This register selects the Alive Count mode. In user specified Alive Counter mode, the host appends an alive counter byte to the end of a UART transaction. The byte is stored in the RX buffer. In automatic Alive Counter mode, the MAX17851 automatically appends a rolling Alive Counter byte to the UART transaction. The received alive counter byte is stored in ALIVECOUNT_RET register and is not stored in the RX buffer.	0x = Alive Counter disabled (default). 10 = User specified Alive Counter enabled. 11 = Automated Alive Counter enabled. **Note:** This setting should match downstream daisy-chain devices; otherwise, the MAX17851 LSSM reports PEC errors.

CONFIG_GEN5 (R/W = 0x6B/0x6A)

General Configuration Register (Read/Write)

BIT	7	6	5	4	3	2	1	0
Field	TX_START_SETUP	TX_HI_Z	ALRTPCKT_DBNC[2:0]			SPI_SFTYSCLK	SPI_SFTYSDI	SPI_SFTYCSB
Reset	0x1	0x0	0x0			0x0	0x0	0x0
Access Type	Write, Read	Write, Read	Write, Read			Write, Read	Write, Read	Write, Read

BITFIELD	BITS	DESCRIPTION	DECODE
TX_START_SETUP	7	UART TX START Setup Enable This bit enables the optional high set-up time of transmitted UART start bits.	0 = Not enabled. 1 = Enabled (default).
TX_HI_Z	6	UART TX Idle State	0 = TX pins idle at ground. 1 = TX pins idle at Hi-Z.
ALRTPCKT_DBNC	5:3	Status Debounce This register controls the number of unmasked consecutive ALERTPACKET status alerts, which must happen prior to taking action in Safe Monitoring or Sleep mode.	000 = 1(default) 001 = 2 010 = 4 011 = 8 100 = 16 101 = 32 110 = 64 111 = 128
SPI_SFTYSCLK	2		0 = Disables internal 100k pulldown on SCLK. 1 = Enables internal 100k pulldown on SCLK.

BITFIELD	BITS	DESCRIPTION	DECODE
SPI_SFTYSDI	1		0 = Disables internal 100k pulldown on SDI. 1 = Enables internal 100k pulldown on SDI.
SPI_SFTYCSB	0		0 = Disables internal 100k pullup on CSB. 1 = Enables internal 100k pullup on CSB.

CONFIG_SAFEMON0 (R/W = 0x6D/0x6C)

Safe Monitoring Mode Configuration Register 0

BIT	7	6	5	4	3	2	1	0
Field	GPIOREC_DLY[7:0]							
Reset	0x0							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION	DECODE
GPIOREC_DLY	7:0	GPIO Recovery Delay This register controls the GPIO Recovery Delay timer, which allows the uC the opportunity to read a GPIO pin state and recover. Unless set to 0xFF (Disabled), the timer delay is calculated as follows: Delay = (GPIOREC_DLY * 10) + 100 (mSec)	0x00 = 100 mSec 0x01 = 110 mSec ..... 0xFE = 2640 mSec 0xFF = disabled

CONFIG_SAFEMON1 (R/W = 0x6F/0x6E)

Safe Monitoring Mode Configuration Register 1

BIT	7	6	5	4	3	2	1	0
Field	CONT_TIMER_DLY[7:0]							
Reset	0xFF							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION	DECODE
CONT_TIMER_DLY	7:0	SAFEMON Contact Timer Delay This register controls the Safe Monitoring mode contact timer delay. At the expiration of this timer, GPIO 1 and 2 will assert (if configured appropriately for Safe Monitoring Output operation). Unless set to 0xFF (Infinite), the timer delay is calculated as follows: Delay = (CONT_TIMER_DLY * 4) + 16 (min)	0x00 = 16 min 0x01 = 20 min ..... 0xFD = 1028 min 0xFE = 1032 min 0xFF = Infinite (default)

CONFIG_SAFEMON2 (R/W = 0x71/0x70)

Safe Monitoring Mode Configuration Register 2

BIT	7	6	5	4	3	2	1	0
Field	–	–	–	–	–	–	SAFEMON_SCAN_DLY[1:0]	
Reset	–	–	–	–	–	–	0x0	
Access Type	–	–	–	–	–	–	Write, Read	

BITFIELD	BITS	DESCRIPTION	DECODE
SAFEMON_SCAN_DLY	1:0	Safe Monitoring Mode Scan Delay This register controls the timing of the Alert Packet generation in Safe Monitoring mode.	00 = 500mS (default) 01 = 1S 10 = 1.5S 11 = Reserved

CONFIG_SAFEMON3 (R/W = 0x73/0x72)

Safe Monitoring Mode Configuration Register 3

BIT	7	6	5	4	3	2	1	0
Field	NOMON	–	–	–	SM_GPIO4_MASK	SM_GPIO3_MASK	SM_GPIO2_MASK	SM_GPIO1_MASK
Reset	0x0	–	–	–	0x1	0x1	0x1	0x1
Access Type	Write, Read	–	–	–	Write, Read	Write, Read	Write, Read	Write, Read

BITFIELD	BITS	DESCRIPTION	DECODE
NOMON	7	No Monitoring/No ALERTPACKET Communication in BMS Safe Monitoring mode.	0 = Device begins monitoring the daisy-chain BMS Safe Monitoring states after the expiration of GPIOREC_DLY. 1 = Device immediately asserts all GPIO pins if the watchdog is not refreshed prior to the expiration of GPIOREC_DLY.
SM_GPIO4_MASK	3	When the GPIO pin is configured as a SAFEMON output, this bit masks the output pin drive during the Safety Measure states of the SAFEMON FSM.	
SM_GPIO3_MASK	2	When the GPIO pin is configured as a SAFEMON output, this bit masks the output pin drive during the Safety Measure states of the SAFEMON FSM.	
SM_GPIO2_MASK	1	When the GPIO pin is configured as a SAFEMON output, this bit masks the output pin drive during the Safety Measure states of the SAFEMON FSM.	
SM_GPIO1_MASK	0	When the GPIO pin is configured as a SAFEMON output, this bit masks the output pin drive during the Safety Measure states of the SAFEMON FSM.	

CONFIG_SLP (R/W = 0x75/0x74)

BIT	7	6	5	4	3	2	1	0
Field	–	–	SLP_SCAN_DLY[1:0]		SLP_ALRT PCKTEN	SLP_CBNTFY[2:0]		
Reset	–	–	0x0		0x0	0x0		
Access Type	–	–	Write, Read		Write, Read	Write, Read		

BITFIELD	BITS	DESCRIPTION	DECODE
SLP_SCAN_DLY	5:4	Sleep Mode Scan Delay This register controls the timing of the Alert Packet generation in Sleep mode.	00 = 500mS (default) 01 = 1S 10 = 1.5S 11 = Reserved
SLP_ALRTPCKTEN	3	Sleep Mode Alert Packet Enable	0 = Alert Packets are not generated in Sleep mode. 1 = Alert Packets are generated according to the SLP_SCAN_DLY register setting in Sleep mode.

BITFIELD	BITS	DESCRIPTION	DECODE
SLP_CBNTFY	2:0	Sleep Mode Cell Balancing Notify When the Cell Balancing timer exceeds the value programmed into the CBNTFY register, the device exits Sleep mode.	000 = 30 min (default) 001 = 60 min 010 = 120 min 011 = 240 min 1XX = OFF (0 min)

CONFIG_COMM (R/W = 0x77/0x76)

BIT	7	6	5	4	3	2	1	0
Field	–	–	COMM_RTRY[1:0]		–	COMM_TO_DLY[2:0]		
Reset	–	–	0x0		–	0x001		
Access Type	–	–	Write, Read		–	Write, Read		

BITFIELD	BITS	DESCRIPTION	DECODE
COMM_RTRY	5:4	Communication Retry Count Threshold This register controls the Communication Retry Count threshold in BMS Safe Monitoring and Sleep mode.	00 = 2 01 = 4 10 = 8 11 = 16

BITFIELD	BITS	DESCRIPTION	DECODE
COMM_TO_DLY	2:0	Communication Timeout Delay This register controls the Communication Timeout delay in all modes of operation. The Communication Timeout timer starts on the beginning of the transaction. Expiration of the Communication Timeout results in the COMM_ERR status bit being set. In Sleep or in BMS Safe Monitoring mode, ALERTPACKT commands are generated automatically according to the associated SCAN_DLY register. A communications timeout in Sleep or in BMS Safe Monitoring mode also results in an ALRTPCKT_TO_ERR. The Communication Timeout Delay is a function of this register setting divided by the BAUD_RATE register setting. For example, if COMM_TO_DLY = 516 bits (43 UART Frames), and BAUD_RATE = 2Mbps, then the Communication Timeout Delay is 516 bits/2Mpbs = 258 us.	000 = 276 bits (23 frames) 001 = 516 bits (43 frames, default) 010 = 996 bits (83 frames) 011 = 1956 bits (163 frames) 1XX = Disabled

STATUS_DBNC_MASK0 (R/W = 0x79/0x78)

BIT	7	6	5	4	3	2	1	0
Field	STATUS_DBNC_MASK[7:0]							
Reset	0x20							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
STATUS_DBNC_MASK	7:0	Masks the STATUS bits in the Alert Packet buffer to generate the debounced SAFEMON_STATUS_ERR_ALRT and SLEEP_STATUS_ERR_ALRT bits. **Note**: The reset value of 0x20 is intended to mask the PEC status bit of the daisy-chain devices by default.

STATUS_DBNC_MASK1 (R/W = 0x7B/0x7A)

BIT	7	6	5	4	3	2	1	0
Field	STATUS_DBNC_MASK[15:8]							
Reset	0x40							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
STATUS_DBNC_MASK	7:0	Masks the STATUS bits in the Alert Packet buffer to generate the debounced SAFEMON_STATUS_ERR_ALRT and SLEEP_STATUS_ERR_ALRT bits. **Note**: The reset value of 0x40 is intended to mask the ALRTRST status bit of the daisy-chain devices by default.

STATUS_ERR_MASK0 (R/W = 0x7D/0x7C)

BIT	7	6	5	4	3	2	1	0
Field	STATUS_ERR_MASK[7:0]							
Reset	0x20							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
STATUS_ERR_MASK	7:0	Masks the STATUS bits in the Alert Packet buffer to generate the non-debounced ALRTPCKT_STATUS_ERR bit. **Note**: The reset value of 0x20 is intended to mask the PEC status bit of the daisy-chain devices by default.

STATUS_ERR_MASK1 (R/W = 0x7F/0x7E)

BIT	7	6	5	4	3	2	1	0
Field	STATUS_ERR_MASK[15:8]							
Reset	0x40							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
STATUS_ERR_MASK	7:0	Masks the STATUS bits in the Alert Packet buffer to generate the non-debounced ALRTPCKT_STATUS_ERR bit. **Note**: The reset value of 0x40 is intended to mask the ALRTRST status bit of the daisy-chain devices by default.

CONFIG_GPIO12 (R/W = 0x81/0x80)

BIT	7	6	5	4	3	2	1	0
Field	–	GPIO2_CFG[2:0]			–	GPIO1_CFG[2:0]		
Reset	–	0b110			–	0b110		
Access Type	–	Write, Read			–	Write, Read		

BITFIELD	BITS	DESCRIPTION	DECODE
GPIO2_CFG	6:4	Programs the operation of the GPIO2 pin	000 = General Purpose input 001 = Reserved (idles as a high impedance input with 2MΩ impedance to ground) 010 = General Purpose HI output 011 = General Purpose LO output 100 = Reserved 101 = Reserved 110 = SAFEMON Active HI output (idles LO, default) 111 = SAFEMON Active LO output (idles HI)
GPIO1_CFG	2:0	Programs the operation of the GPIO1 pin	000 = General Purpose input 001 = Reserved (idles as a high impedance input with 2MΩ impedance to ground) 010 = General Purpose HI output 011 = General Purpose LO output 100 = Reserved 101 = Reserved 110 = SAFEMON active HI output (idles LO, default) 111 = SAFEMON active LO output (idles HI)

CONFIG_GPIO34 (R/W = 0x83/0x82)

BIT	7	6	5	4	3	2	1	0
Field	–	GPIO4_CFG[2:0]			–	GPIO3_CFG[2:0]		
Reset	–	0b101			–	0b101		
Access Type	–	Write, Read			–	Write, Read		

BITFIELD	BITS	DESCRIPTION	DECODE
GPIO4_CFG	6:4	Programs the operation of the GPIO4 pin	000 = General Purpose input 001 = SAFEMON Slave input 010 = General Purpose HI output 011 = General Purpose LO output 100 = SAFEMON one-shot HI output (idles LO) 101 = SAFEMON one-shot LO output (idles HI, default) 110 = SAFEMON active HI output (idles LO) 111 = SAFEMON active LO output (idles HI)
GPIO3_CFG	2:0	Programs the operation of the GPIO3 pin	000 = General Purpose input 001 = Reserved (idles as a high impedance input with 2MΩ impedance to ground) 010 = General Purpose HI output 011 = General Purpose LO output 100 = SAFEMON one-shot HI output (Idles LO) 101 = SAFEMON one-shot LO output (idles HI, default) 110 = SAFEMON active HI output (idles LO) 111 = SAFEMON active LO output (idles HI)

CONFIG_WD0 (R/W = 0x85/0x84)

BIT	7	6	5	4	3	2	1	0
Field	WD_OPN[3:0]				WD_CLO[3:0]			
Reset	0x0				0x0			
Access Type	Write, Read				Write, Read			

BITFIELD	BITS	DESCRIPTION	DECODE
WD_OPN	7:4	Watchdog Open Window This register sets tWD1, which is the number of watchdog clock cycles from a valid refresh until the open window starts. **Note:** Programming this register while the watchdog is enabled (WD_EN = 1) can cause unexpected operation.	$t_{WD1} = t_{WDCLK} * (WD\_OPN[3:0]+1)*8$

BITFIELD	BITS	DESCRIPTION	DECODE
WD_CLO	3:0	Watchdog Close Window This register sets tWD2, which is the time from a valid refresh until the closed window starts (i.e., the closed window plus the open window). **Note:** Programming this register while the watchdog is enabled (WD_EN = 1) can cause unexpected operation.	$t_{WD2} = t_{WD1} + t_{WDCLK} * (WD\_CLO[3:0]+1)*8$

CONFIG_WD1 (R/W = 0x87/0x86)

BIT	7	6	5	4	3	2	1	0
Field	WD_1UD[2:0]			WD_DIV[4:0]				
Reset	0x0			0x0				
Access Type	Write, Read			Write, Read				

BITFIELD	BITS	DESCRIPTION	DECODE
WD_1UD	7:5	First Update Extension This register sets the number of extra tWD2 cycles after WD_EN to the normal tWD2. **Note:** Programming this register while the watchdog is enabled (WD_EN = 1) can cause unexpected operation.	$t_{1STWD2} = t_{WD2} * (WD\_1UD[2:0]+1)$
WD_DIV	4:0	Watchdog clock divider **Note:** Programming this register while the watchdog is enabled (WD_EN = 1) can cause unexpected operation.	tWDCLK = (WD_DIV[4:0]+1) * 256µs

CONFIG_WD2 (R/W = 0x89/0x88)

BIT	7	6	5	4	3	2	1	0
Field	WD_EN	–	–	–	WD_SWW	WD_DBNC[2:0]		
Reset	0x0	–	–	–	0x0	0x0		
Access Type	Write, Read	–	–	–	Write, Read	Write, Read		

BITFIELD	BITS	DESCRIPTION	DECODE
WD_EN	7	Watchdog Enable	0 = Disabled 1 = Enabled
WD_SWW	3	Simple Windowed Watchdog Enable.	0 = Challenge/Response Watchdog Enabled 1 = Standard Windowed Watchdog Enabled

BITFIELD	BITS	DESCRIPTION	DECODE
WD_DBNC	2:0	Watchdog Debounce This register controls the number of consecutive watchdog violations that must occur before entering Safe Monitoring mode. **Note:** Programming this register while the watchdog is enabled (WD_EN = 1) can cause unexpected operation.	000 = 1 001 = 2 010 = 4 011 = 8 100 = 16 101 = 32 110 = 64 111 = 128

ALRTPCKTBUF_RD_PTR (R/W = 0x8D/0x8C)

BIT	7	6	5	4	3	2	1	0
Field	–	–	–	–	–	ALRTPCKTBUF_RD_PTR[2:0]		
Reset	–	–	–	–	–	0x0		
Access Type	–	–	–	–	–	Write, Read, Ext		

BITFIELD	BITS	DESCRIPTION
ALRTPCKTBUF_RD_PTR	2:0	Alert Packet Buffer Read Pointer The location in the Alert Packet buffer that the host is to read. This register is read/ writable and automatically incremented when ALRTPCKTBUF_RD_MSG is accessed.

ALRTPCKTBUF_RD_MSG (R/W = 0x8F/0x8E)

BIT	7	6	5	4	3	2	1	0
Field	ALRTPCKTBUF_RD_MSG[7:0]							
Reset	0x00							
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION
ALRTPCKTBUF_RD_MSG	7:0	ALRTPCKTBUF_RD_MSG Command Reads the Receive buffer starting at the address ALRTPCKTBUF_RD_PTR. Automatically increments the read pointer after the byte is read but does not increment the read pointer into the next message.

RX_RD_MSG (R/W = 0x91/0x90)

BIT	7	6	5	4	3	2	1	0
Field	RX_RD_MSG[7:0]							
Reset	0x00							
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION
RX_RD_MSG	7:0	RX_RD_MSG Command Reads the Receive buffer starting at the address RX_RD_PTR. Automatically increments the read pointer after the byte is read, but does not increment the read pointer into the next message. **Note:** When performing SPI read or write bursts to this address, the RX_RD_PTR register increments, and internal SPI address pointer does not.

RX_RD_NXT_MSG (R/W = 0x93/0x92)

BIT	7	6	5	4	3	2	1	0
Field	RX_RD_NXT_MSG[7:0]							
Reset	0x00							
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION
RX_RD_NXT_MSG	7:0	RX_RD_NXT_MSG Command Reads the Receive buffer starting at the address RX_NXT_MSG_ PTR (oldest unread message). Automatically increments the read pointer after the byte is read but does not increment the read pointer into the next message. **Note:** When performing SPI read or write bursts to this address, the RX_NXT_MSG_PTR register increments, and internal SPI address pointer does not.

TX_QUEUE_SEL (R/W = 0x95/0x94)

BIT	7	6	5	4	3	2	1	0
Field	–	–	–	–	TX_Q[1:0]		LD_Q[1:0]	
Reset	–	–	–	–	0x00		0x00	
Access Type	–	–	–	–	Read Only		Read Only	

BITFIELD	BITS	DESCRIPTION
TX_Q	3:2	Transmit Queue Select Addresses one of four queues in the Transmit buffer that the UART has selected for message transmission (sending).
LD_Q	1:0	Load Queue Select Addresses one of four queues in the Transmit buffer that the host has selected for message loading (writing).

RX_RD_PTR (R/W = 0x97/0x96)

BIT	7	6	5	4	3	2	1	0
Field	–	RX_RD_PTR[6:0]						
Reset	–	0x00						
Access Type	–	Read Only						

BITFIELD	BITS	DESCRIPTION
RX_RD_PTR	6:0	Receive Buffer Read Pointer The location in the Receive buffer that the host is to read. The UART automatically increments this pointer.

RX_WR_PTR (R/W = 0x99/0x98)

BIT	7	6	5	4	3	2	1	0
Field	–	RX_WR_PTR[6:0]						
Reset	–	0x01						
Access Type	–	Read Only						

BITFIELD	BITS	DESCRIPTION
RX_WR_PTR	6:0	Receive Buffer Write Pointer The location in the Receive buffer that is written by the UART as it receives data.

RX_NXT_MSG_PTR (R/W = 0x9B/0x9A)

BIT	7	6	5	4	3	2	1	0
Field	–	RX_NXT_MSG_PTR[6:0]						
Reset	–	0x00						
Access Type	–	Read Only						

BITFIELD	BITS	DESCRIPTION
RX_NXT_MSG_PTR	6:0	Receive Buffer Next Message Pointer The start of the next unread message in the Receive buffer. The RX_RD_Pointer is loaded with this value by the RD_NXT_MSG SPI transaction.

RX_SPACE (R/W =0x9D/0x9C)

BIT	7	6	5	4	3	2	1	0
Field	–	RX_SPACE[6:0]						
Reset	–	0x56						
Access Type	–	Read Only						

BITFIELD	BITS	DESCRIPTION
RX_SPACE	6:0	RX Space Register Number of available bytes in the Receive buffer.

RX_BYTE (R/W =0x9F/0x9E)

BIT	7	6	5	4	3	2	1	0
Field	–	–	–	–	–	RX_FIRST_BYTE	RX_BYTE_ERR	RX_LAST_BYTE
Reset	–	–	–	–	–	0x0	0x0	0x0
Access Type	–	–	–	–	–	Read Only	Read Only	Read Only

BITFIELD	BITS	DESCRIPTION
RX_FIRST_BYTE	2	RX Buffer First Byte Indicator The byte at location RX_RD_PTR is the first data byte in a message (the corresponding character is preceded by preamble character).
RX_BYTE_ERR	1	RX Byte Error Indicator The data byte at location RX_RD_PTR may contain an error (the corresponding character contained a Manchester and/or parity error).
RX_LAST_BYTE	0	RX Packet Last Byte Indicator The byte at location RX_RD_PTR is the last byte in a message (the corresponding character is a stop character and is stored as a null byte).

NXT_LDQ (R/W = 0xB1/0xB0)

BIT	7	6	5	4	3	2	1	0
Field	NXT_LDQ[7:0]							
Reset	0x00							
Access Type	Write, Read, Ext							

BITFIELD	BITS	DESCRIPTION
NXT_LDQ	7:0	NXT_LDQ Command This command increments LDQ, then writes the Transmit buffer load queue. The increment occurs whether the host loads the data or not. The command byte defines the first location to be written (locations 0 to 30). Burst writes beyond location 30 have no effect. **Note:** When performing SPI read or write bursts to this address, the LDQ_PTR register increments, and internal SPI address pointer does not.

LDQ (R/W = 0x1/0x0)

BIT	7	6	5	4	3	2	1	0
Field	LDQ[7:0]							
Reset	0x00							
Access Type	Write, Read, Ext							

BITFIELD	BITS	DESCRIPTION
LDQ	7:0	LDQ Command Reads/Writes the Transmit buffer load queue. The command byte defines the first byte written (locations 0 to 30). Burst writes beyond location 30 have no effect. **Note:** When performing SPI read or write bursts to this address, the LDQ_PTR register increments, and internal SPI address pointer does not.

LDQ_PTR (R/W = 0xC1/0xC0)

BIT	7	6	5	4	3	2	1	0
Field	–	–	–	LDQ_PTR[4:0]				
Reset	–	–	–	0x0				
Access Type	–	–	–	Write, Read, Ext				

BITFIELD	BITS	DESCRIPTION
LDQ_PTR	4:0	Transmit Buffer Read/Write Pointer The location in the Transmit LDQ buffer that the host is to read or write. This pointer automatically incremented when the LDQ register is read or written and is cleared on a NXT_LDQ write command.

CONFIGQ (R/W = 0xD1/0xD0)

BIT	7	6	5	4	3	2	1	0
Field	CONFIGQ[7:0]							
Reset	0xFF							
Access Type	Write, Read, Ext							

BITFIELD	BITS	DESCRIPTION
CONFIGQ	7:0	Configuration Data Queue Byte Reads and writes the Configuration Data Queue byte as pointed to by the CONFIG_BYTE_PTR and CONFIG_QUEUE_PTR registers. **Note:** When performing SPI read or write bursts to this address, the CONFIG_PTR register increments, and internal SPI address pointer does not.

CONFIG_PTR (R/W = 0xD3/0xD2)

BIT	7	6	5	4	3	2	1	0
Field	–	CONFIG_QUEUE_PTR[1:0]		CONFIG_BYTE_PTR[4:0]				
Reset	–	0x00		0x00				
Access Type	–	Write, Read, Ext		Write, Read, Ext				

BITFIELD	BITS	DESCRIPTION
CONFIG_QUEUE_PTR	6:5	Config Queue Pointer This Read/Write register sets the Queue Pointer used to read/write the CONFIGQ register. This register auto-increments when the CONFIGQ register is accessed. Also, this register rolls over from 'd3->'d0.

BITFIELD	BITS	DESCRIPTION
CONFIG_BYTE_PTR	4:0	Config Byte Pointer This read/write register sets the byte pointer used to read/write the CONFIGQ register. When CONFIG_QUEUE_PTR contains a value from 'd0 -> 'd2, writes to a non-valid value of 'd30 or greater results in the write being ignored. When CONFIG_QUEUE_PTR contains a value of 'd3, writes to a non-valid value of 'd6 or greater results in the write being ignored. This register auto-increments when the CONFIGQ register is accessed. Also, this register rolls over and auto-increments the CONFIG_QUEUE_PTR register when accessed.

STATE (R/W = 0xDD/0xDC)

BIT	7	6	5	4	3	2	1	0
Field	–	–	–	–	–	STATE[2:0]		
Reset	–	–	–	–	–	0x0		
Access Type	–	–	–	–	–	Read Only		

BITFIELD	BITS	DESCRIPTION
STATE	2:0	This register indicates the current state of the MAX17851. This register is intended for system level debugging.

COMM_RTRY_CNT (R/W =0xDF/0xDE)

BIT	7	6	5	4	3	2	1	0
Field	–	–	–	COMM_RTRY_CNT[4:0]				
Reset	–	–	–	0x0				
Access Type	–	–	–	Read Only				

BITFIELD	BITS	DESCRIPTION
COMM_RTRY_CNT	4:0	Communication Retry Counter This register provides the current RETRY counter data associated with COMM_RTRY in BMS Safe Monitoring or Sleep modes. This register is useful for a system level debug.

ALRTPCKT_ERR_CNT (R/W = 0xE1/0xE0)

BIT	7	6	5	4	3	2	1	0
Field	ALRTPCKT_ERR_CNT[7:0]							
Reset	0x0							
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION
ALRTPCKT_ERR_CNT	7:0	This register provides the current counter data associated with ALRTPCKT_DBNC in BMS Safe Monitoring or Sleep modes. This register is useful for a system level debug.

WD_FAULT_CNT (R/W = 0xE3/0xE2)

BIT	7	6	5	4	3	2	1	0
Field	WD_FAULT_CNT[7:0]							
Reset	0x0							
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION
WD_FAULT_CNT	7:0	Watchdog Fault Count - Indicates the number of watchdog faults that have occurred. This register is reset when WD_EN is disabled or when a valid watchdog refresh occurs. This register is useful for a system level debug.

ALIVECOUNT_SEED (R/W = 0xE5/0xE4)

BIT	7	6	5	4	3	2	1	0
Field	ALIVECOUNT_SEED[7:0]							
Reset	0x0							
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION
ALIVECOUNT_SEED	7:0	This is the seed value for the automated Alive Counter. When the Alive Counter is enabled, this seed is appended to a UART transaction and incremented by one when the transmit is complete.

ALIVECOUNT_RET (R/W = 0xE7/0xE6)

BIT	7	6	5	4	3	2	1	0
Field	ALIVECOUNT_RET[7:0]							
Reset	0x0							
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION
ALIVECOUNT_RET	7:0	When the automated Alive Counter is enabled, this read-only register stores returned Alive Counter bytes received by the MAX17851 after every transaction.

ALIVECOUNT_Q (R/W = 0xE7/0xE6)

BIT	7	6	5	4	3	2	1	0
Field	–	–	–	–	–	ALIVECOUNT_Q[2:0]		
Reset	–	–	–	–	–	0x0		
Access Type	–	–	–	–	–	Read Only		

BITFIELD	BITS	DESCRIPTION
ALIVECOUNT_Q	2:0	This read-only register contains the number of transactions sent by the TX buffer and not yet received by the RX buffer. This registeris reset by the CLR_LSSM command.

FAULT_TIMER0 (R/W = 0xEB/0xEA)

Fault Timer LSB (Seconds)

BIT	7	6	5	4	3	2	1	0
Field	FAULT_TIMER[7:0]							
Reset	0x0							
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION
FAULT_TIMER	7:0	The Fault Timer (LSB) indicates the amount of time (seconds) that has expired since a communication fault or a daisy-chain fault (as indicated by an ALERTPACKET) has occurred in Sleep or BMS Safe Monitoring Mode. The Fault Timer starts once the fault is debounced and stops once the SLP_ALRT or SAFEMON_ALRT bits are cleared. The Fault Timer is reset at the start of Sleep or BMS Safe Monitoring mode.

FAULT_TIMER1 (R/W = 0xED/0xEC)

Fault Timer MSB (Seconds)

BIT	7	6	5	4	3	2	1	0
Field	FAULT_TIMER[15:8]							
Reset	0x0							
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION
FAULT_TIMER	7:0	The Fault Timer (MSB) indicates the amount of time (seconds) that has expired since a communication fault or a daisy-chain fault (as indicated by an ALERTPACKET) has occurred in Sleep or BMS Safe Monitoring mode. The Fault Timer starts once the fault is debounced and stops once the SLP_ALRT or SAFEMON_ALRT bits are cleared. The Fault Timer is reset at the start of Sleep or BMS Safe Monitoring mode.

SLP_CBTIMER0 (R/W = 0xEF/0xEE)

BIT	7	6	5	4	3	2	1	0
Field	SLP_CBTIMER[7:0]							
Reset	0x0							
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION
SLP_CBTIMER	7:0	Sleep Mode Cell balancing Timer Readback LSB (seconds).

SLP_CBTIMER1 (R/W = 0xF1/0xF0)

BIT	7	6	5	4	3	2	1	0
Field	SLP_CBTIMER[15:8]							
Reset	0x0							
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION
SLP_CBTIMER	7:0	Sleep Mode Cell balancing Timer Readback MSB (seconds).

VERSION (R/W = 0xF3/0xF2)

BIT	7	6	5	4	3	2	1	0
Field	MODEL[3:0]				VERSION[3:0]			
Reset	0x1							
Access Type	Read Only				Read Only			

BITFIELD	BITS	DESCRIPTION
MODEL	7:4	Last digit of the Model Number.
VERSION	3:0	Mask revision.

MODEL (R/W = 0xF5/0xF4)

BIT	7	6	5	4	3	2	1	0
Field	MODEL[11:4]							
Reset	0x85							
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION
MODEL	7:0	First two digits of the Model Number.

Applications Information

System Configuration

The MAX17851 primarily supports two different system configurations. The single UART and the dual UART. The MAX17851 is configured as master only for single UART. For DUAL UART, MAX17851 devices are configured as master/slave configuration. Regardless of the which system the hardware is setup for, the host microcontroller is recommended to follow a similar sequence for device configuration, initialization of the BMS Data Acquisition devices, configuration memory, and functional operation parameters.

The following section details a step-by-step procedure that the user can follow to configure all devices in the BMS Data Acquisition System.

Dual UART Operation

The MAX17851 may be setup in a master/slave configuration in a dual UART application to allow for greater reliability and throughput. In this application there are two UART paths of communication that allow a failure in one of the communication paths without losing UART communications. The MAX17851 devices are configured as a master and a slave device using the MS_EN field of the CONFIG_GEN4 register.

The data received in the slave device is in the reverse order from that of the master device. The slave device should set the RXSWAP_EN bit in the CONFIG_GEN4 register. This changes the data received by the BMS UART devices to match that of the master MAX17851.

GPIO1 and GPIO2 are configured as outputs for both the master and slave MAX17851 devices. Both of these GPIO pins should be configured as SAFEMON outputs. The output can be set as either active high or active low. The GPIO1 and GPIO2 outputs should have some form of isolation. Figure 20 shows a diode on GPIO1 and a resistor on GPIO2. This allows for one of the GPIO outputs to fail, while the other safely asserts the Battery Contactor signal.

GPIO3 on the master MAX17851 should be set as a SAFEMON active HI output, while GPIO4 on the slave device should be set to SAFEMON slave input. This allows the master device to alert the slave when BMS Safe Monitoring mode is entered.

Figure 20 shows the remaining GPIO pins (i.e., GPIO4 on the master device, GPIO3 on the slave device) connected to a reset input on the microcontroller and a power supply enable input. These GPIO pins could be configured as SAFEMON one-shot HI output or SAFEMON one-shot LO output, depending on the polarity of the signal needed. The one-shot outputs assert for 100ms then de-assert, allowing the connected devices to go through a power on reset sequence. For more information about GPIO configuration, see the *GPIO Control* section.

The dual UART configuration also allows for more throughput when reading data from the BMS UART daisy-chain devices. For example, the master MAX17851 can set to read odd number devices, while the slave MAX17851 can read the even numbered devices. Care must be taken to be sure that the host microcontroller can service the data throughput needed to get data from two SPI interfaces (one for each MAX17851).

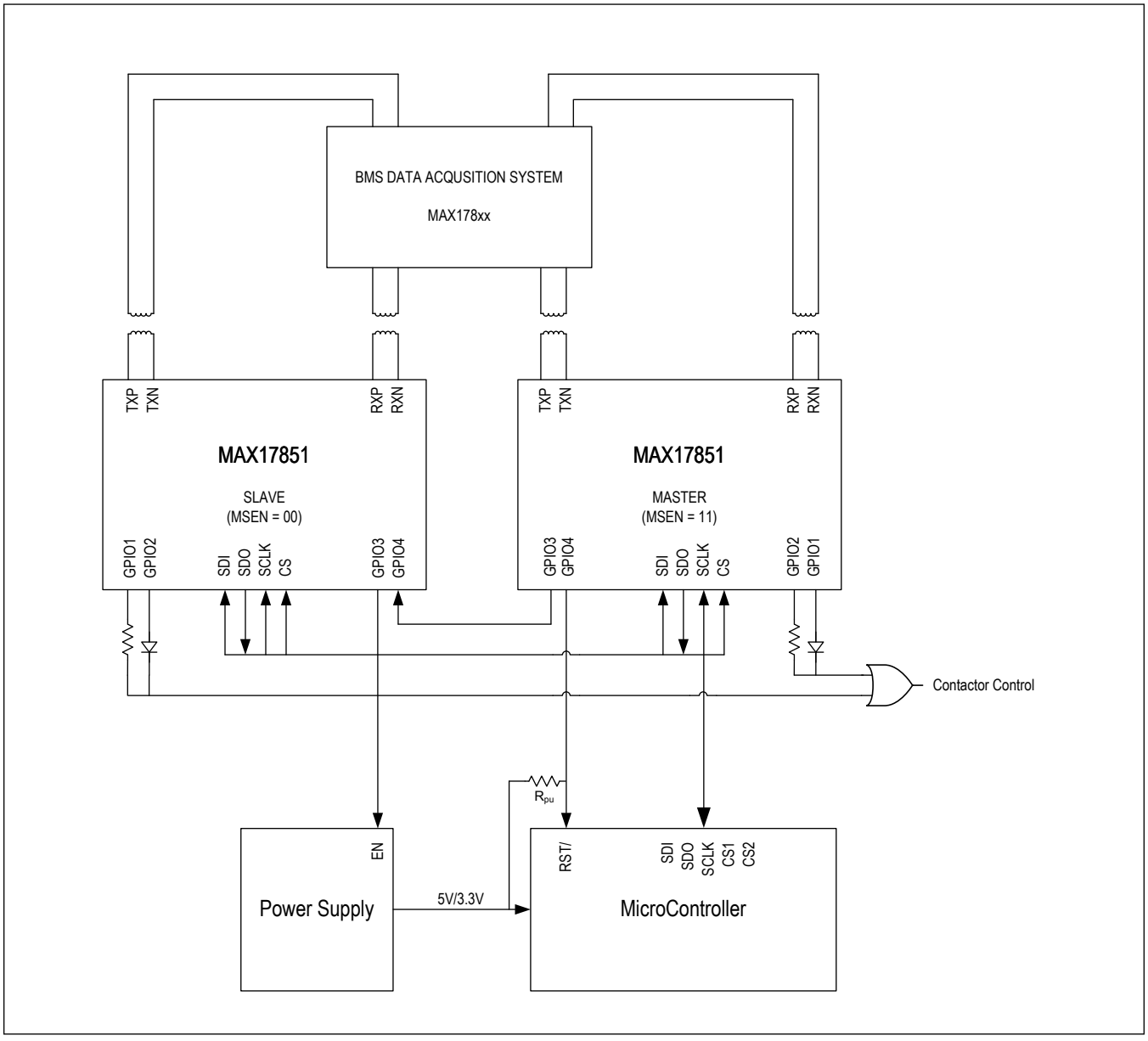


Figure 20. Dual UART

Dual UART MAX17851 Initialization

The example shown in Table 17 details a sequence for configuring the MAX17851 devices in master and slave configuration. In addition to this sequence, the host can always verify all the writes to the configuration registers by reading back the register data. This adds an extra layer of verification.

Table 17. Master/Slave Device Configuration Sequence

SPI DIN	SPI DOUT	TARGET DEVICE	DESCRIPTION
TRANSACTION 1		Master and Slave	Configure CONFIG_GEN0.
60h	xxh		Write device count.
07h	xxh		Set device count to 7 (change based on actual daisy-chain count).
TRANSACTION 2		Master and Slave	Configure CONFIG_GEN1.
62h	xxh		Configure baud rate.
30h	xxh		Set baud rate = 2Mbps.
TRANSACTION 3		Master	Configure CONFIG_GEN4.
68h	xxh		Configure RXSWAPEN and MS_EN.
38h	xxh		Enable master dual UART with normal data sequence.
TRANSACTION 4		Slave	Configure CONFIG_GEN4.
68h	xxh		Configure RXSWAPEN and MS_EN.
40h	xxh		Enable slave dual UART with data sequencing same as master.
TRANSACTION 5		Master and Slave	Configure CONFIG_GEN5.
6Ah	xxh		Configure ALRTPCKT_DBNC.
08h	xxh		Enables two consecutive errors in BMS Safe Monitoring or Sleep mode before error is flagged.
TRANSACTION 6		Master and Slave	Configure CONFIG_SAFEMON0.
6Ch	xxh		Configure GPIOREC_DLY.
5Ah	xxh		Configures 1s delay between assertion of GPIO1, 2 and GPIO3, 4. Performs the GPIO safety measure in BMS Safe Monitoring mode.
TRANSACTION 7		Master and Slave	Configure CONFIG_SAFEMON1.
6Eh	xxh		Configure CONT_TIMER_DLY.
FEh	xxh		Provides 1032 min delay upon communication error (broken wire or persistent communication fault) before driving signal to open contactor.
TRANSACTION 8		Master and Slave	Configure CONFIG_SAFEMON2.
70h	xxh		Configure SAFEMON_SCAN_DLY.
01h	xxh		Enables 1s fault polling of BMS Data Acquisition System.
TRANSACTION 9		Master and Slave	Configure CONFIG_SAFEMON3.
72h	xxh		Configure.
0Fh	xxh		Masks GPIO1, GPIO2, GPIO3, GPIO4 output upon initial entry into BMS Safe Monitoring mode. Provides time to perform GPIO safety measures.
TRANSACTION 10		Master and Slave	Configure CONFIG_SLP.
74h	xxh		Configure SLP_SCAN_DLY, SLP_ALRTPCKTEN, SLP_CBNTFY.
1Fh	xxh		Enables 1s auto polling of BMS Data Acquisition System when Sleep mode is entered, interrupt driven response to microcontroller.

Table 17. Master/Slave Device Configuration Sequence (continued)

SPI DIN	SPI DOUT	TARGET DEVICE	DESCRIPTION
TRANSACTION 11		Master and Slave	Configure CONFIG_COMM.
76h	xxh		Configure COMM_RTRY, COMM_TO_DLY.
02h	xxh		For daisy-chain = 7, READALL timing is 251us. COMM_TO_DLY is recommended to be 30% longer than the message timing.
TRANSACTION 12		Master and Slave	Configure GPIO12.
80h	xxh		Configure GPIO1_CFG and GPIO2_CFG.
66h	xxh		Enables the contactor drivers for NMOS input (BMS Safe Monitoring mode only).
TRANSACTION 13		Master	Configure GPIO34.
82h	xxh		Configure GPIO3_CFG and GPIO4_CFG.
46h	xxh		Enables master/slave communication transition in BMS Safe Monitoring mode. Enables microcontroller active high reset.
TRANSACTION 14		Slave	Configure GPIO34.
82h	xxh		Configure GPIO3_CFG and GPIO4_CFG.
16h	xxh		Enables master/slave communication transition in BMS Safe Monitoring mode. Enables microcontroller power supply active high reset.
TRANSACTION 15		Master and Slave	Configure ALRTEN_OPSTATE.
28h	xxh		Configure SLP_ALRTEN, SLP_STATUS_ERR_ALRTEN, SAFEMON_ALRTEN, SAFEMON_GPIO12_ALRTEN, SAFEMON_STATUS_ERR_ALRTEN, SAFEMON_CONFIG_ERR_ALRTEN.
3Fh	xxh		Enables Alert pin assertion for the configured Sleep and BMS Safe Monitoring mode alerts.
TRANSACTION 16		Master and Slave	Configure ALRTEN_LSSM_BYTE.
24h	xxh		Configure RX_READY_ALRTEN.
80h	xxh		Enables Alert pin assertion for the RX_READY Alert for LSSM status byte.
TRANSACTION 17		Master and Slave	Configure ALRTEN_RX.
20h	xxh		Configure RX_STOP_ALRTEN.
02h	xxh		Enables Alert pin assertion for the RX_STOP.
TRANSACTION 18		Master and Slave	Configure ALRTEN_WD.
2Ch	xxh		Configure WD_TO_ERR_ALRTEN.
10h	xxh		Enables Alert pin assertion for watchdog timeout. This allows the synchronization between host microcontroller and the MAX17851.
TRANSACTION 19		Master and Slave	Check for Receive buffer errors.
11h	xxh		Read ALERT_RX register flags.
xxh	00h		If no errors, continue; otherwise, clear and go to error routine.

After the master/slave UART device parameters are configured, the host microcontroller can initialize the BMS Data Aquisition System and well as run the configuration memory sequence to configure the BMS daisy-chain devices. See *BMS Data Acquisition System Initialization for DUAL UART* and *Configuration Memory Sequence for Daisy-Chain* for

further details.

Dual UART BMS Data Acquisition System Initialization

At the first power up or after the reset of devices, the MAX17851 needs to perform an initialization sequence to configure the BMS daisy-chain for correct operation. Before the initialization sequence, the host must verify that the hardware configuration is correct and certain parameters are programmed correctly, such as device count, baud rate, keep alive settings, etc.

It is also recommended that the host verify the device configuration register contents by writing and reading back desired settings in CONFIG_GEN registers.

Before the initialization sequence is executed, the host also needs to configure the system setup in the General Configuration register CONFIG_GEN4 by writing MS_EN[5:4]. The default configuration is master, single UART. If the hardware is configured for master/slave UART, then the slave device needs to be configured for correct data to receive configuration, so that the device can interpret the data correctly (data received in the slave device is in reverse order). This can be done by writing [RXSWAP_EN[6] in the CONFIG_GEN4 register.

See Configuration Memory Sequence in the Daisy-Chain Configuration section for details on how to program the configuration memory for the desired UART commands to be executed in BMS Safe Monitoring mode.

Once the device parameters are configured, the host should initialize the BMS Data Acquisition System.

After the host successfully performs the HELLOALL sequence, which determines the correct number of devices in the daisy-chain, it can proceed and enable the Alert Packet generation in Commanded Operation mode followed by normal operation. The Alert Packets can be enabled in CONFIG_GEN register before the initialization sequence. However, during the initialization sequence Alert Packet generation is masked, and Keep-Alive patterns are used for robust system performance. The frequency of Alert Packet generation can be configured by the ALRTPCKT_TIMING[3:0] in the CONFIG_GEN3 register. The Alert Packet generation gives a system advantage of autonomously checking the daisy-chain communication and errors without the need for host intervention.

If the host receives a device count error after the HELLOALL sequence, the sequence needs to be executed again or hardware configuration checked.

The example shown in Table 18 outlines the sequence for initialization of two UART slave devices.

Table 18. Master/Slave BMS Daisy-Chain Initialization Sequence

SPI DIN	SPI DOUT	TARGET DEVICE	DESCRIPTION
TRANSACTION 1		Master and Slave	Configure CONFIG_GEN3.
66h	xxh		Configure ALRTPCKT_TIMING.
05h	xxh		Set Keep-Alive period to 160µs.
TRANSACTION 2		Master and Slave	Configure ALRTEN_RX.
20h	xxh		Configure RX_ERR_ALRTEN and RX_OVRFLW_ERR_ALRTEN.
88h	xxh		Enables the RX_ERR_ALRTEN and RX_OVRFLW_ERR_ALRTEN bits (ALERT pin enable bits).
TRANSACTION 3		Master and Slave	Configure CONFIG_GEN2.
64h	xxh		Configure TX_PREAMBLES.
30h	xxh		Enables Transmit of Preambles (this wakes up daisy-chain devices).
TRANSACTION 4		Master and Slave	Read STATUS_RX .
01h	xxh		Poll RX_BUSY and RX_EMPTY.
xxh	21h		If RX_STATUS = 21h, continue. Otherwise, repeat transaction until true or timeout.
TRANSACTION 5		Master and Slave	Configure CONFIG_GEN2.
64h	xxh		Configure TX_PREAMBLES and TX_QUEUE Transmission.
10h	xxh		Disable Transmit of Preambles.
TRANSACTION 6		Master and Slave	CLR_RXBUF

Table 18. Master/Slave BMS Daisy-Chain Initialization Sequence (continued)

SPI DIN	SPI DOUT	TARGET DEVICE	DESCRIPTION
42h	xxh		Clears and reset Receive buffer.
00h	xxh		Clears and reset Receive buffer
TRANSACTION 7		Master and Slave	CLR_TXBUF
40h	xxh		Clears and reset Transmit buffer.
00h	xxh		Clears and reset Transmit buffer.
TRANSACTION 8		Master	Load the HELLOALL command sequence into the load queue.
C0h	xxh		Write the LD_Q SPI command byte (write the load queue).
03h	xxh		Message length.
57h	xxh		HELLOALL command byte.
00h	xxh		Register address (0x00).
00h	xxh		Initialization address of HELLOALL.
TRANSACTION 9		Master	Read RX_RD_MSG (optional).
C2h	xxh		Verify contents of load queue.
00h	xxh		Change the pointer to the load queue to be read.
TRANSACTION 10		Master	Read the buffer content (optional).
C1h	xxh		Read address for the load queue.
xxh	03h		Message length.
xxh	57h		HELLOALL command byte.
xxh	00h		Register address.
xxh	00h		Initialization address of HELLOALL.
TRANSACTION 11		Master	Transmit HELLOALL sequence.
B0h	xxh		NXT_LDQ SPI command byte (write the next load queue).
TRANSACTION 12		Master	Read STATUS_RX.
01h	xxh		Poll RX_STOP status bit.
xxh	x2h		If RX_Status[1] is true, continue. If false, then repeat transaction until true.
TRANSACTION 13		Master	Read RX_RD_NXT_MSG .
93h	xxh		Service Receive buffer. Read the HELLOALL message propagated through the daisy-chain The host should verify the device count.
xxh	57h		Sent command byte (HELLOALL).
xxh	00h		Sent address = 00h
xxh	02h		Returned address = 02h (2 devices)
xxh	84h		LSSM_BYTE
TRANSACTION 14		Master	Read ALERT_RX.
11h	xxh		Check for Receive buffer error.
xxh	00h		If no errors, continue; otherwise, clear and go to error routine.
TRANSACTION 15		Slave	Configure BMS Data Acquisition System as down host (Transactions 14,15,16 are optional and should be used if DOWNHOST).
C0h	xxh		Write the LD_Q SPI command byte (write the load queue).
03h	xxh		Message length.
09h	xxh		Downhost

Table 18. Master/Slave BMS Daisy-Chain Initialization Sequence (continued)

SPI DIN	SPI DOUT	TARGET DEVICE	DESCRIPTION
00h	xxh		DataLSB
00h	xxh		Top address.
TRANSACTION 16		Slave	Transmit data.
B0h	xxh		NXT_LDQ SPI command byte (write the next load queue).
TRANSACTION 17		Slave	Read ALERT_RX.
11h	xxh		Check for Receive buffer error.
xxh	00h		If no errors, continue; otherwise, clear and go to error routine.

After the device is successfully initialized, the host microcontroller can continue with the normal operation. The host can disable the automatic Keep-Alive generation from the MAX17851 and instead configure the device with Alert Packet generation. The Alert Packet configuration sequence is shown in Table 22

Final MAX17851 Configuration

After the MAX17851 is successfully initialized and there are no errors, the host can disable the automatic Keep-Alive generation from the MAX17851 and configure the device with the automatic ALERTPACKET generation.

The configuration sequence is shown in Table 19.

Table 19. Master/Slave UART Alert Packet Configuration

SPI DIN	SPI DOUT	TARGET DEVICE	DESCRIPTION
TRANSACTION 1		Master	Configure CONFIG_GEN2.
64h	xxh		Configure TX_PREAMBLES.
10h	xxh		Disable Transmit Preambles mode.
TRANSACTION 2		Master and Slave	Configure CONFIG_GEN3.
66h	xxh		Configure ALRTPCKT_TIMING.
08h	xxh		Alert packet timing of 1.28ms.
TRANSACTION 3		Master	Configure CONFIG_GEN4 master.
68h	xxh		Configure CO_ALRTPCKTEN.
B8h	xxh		Configures Alert Packet timing in Commanded Operation mode. In sleep or BMS Safe Monitoring mode timing is dictated by SLP_SCAN_DLY and SAFEMON_SCAN_DLY.
TRANSACTION 4		Slave	Configure CONFIG_GEN4 slave.
68h	xxh		Configure CO_ALRTPCKTEN.
88h	xxh		
TRANSACTION 5		Master and Slave	Configure CONFIG_SLP.
74h	xxh		Configure SLP_SCAN_DLY, SLP_ALRTPCKTEN.
3Fh	xxh		Enables SLP_SCAN_DLY to 1s. SLP_ALRTPCKTEN enables in order to use the delay configured by SLP_SCAN_DLY. Configure SLP_CBNTFY to off (user selected).

Single UART Operation

The MAX17851 can be configured in a single UART configuration for a single daisy-chain system. The data flow is always from the host, up to the daisy-chain path, then loops back from the daisy-chain device to the host.

To configure the device in single UART mode, the host first configures the device in the single UART configuration

(MS_EN = 0b10).

Figure 21 shows the single UART hardware configuration. After the initial power up or reset, the host can follow the *Single UART Initial Device Configuration* sequence for initial device configuration.

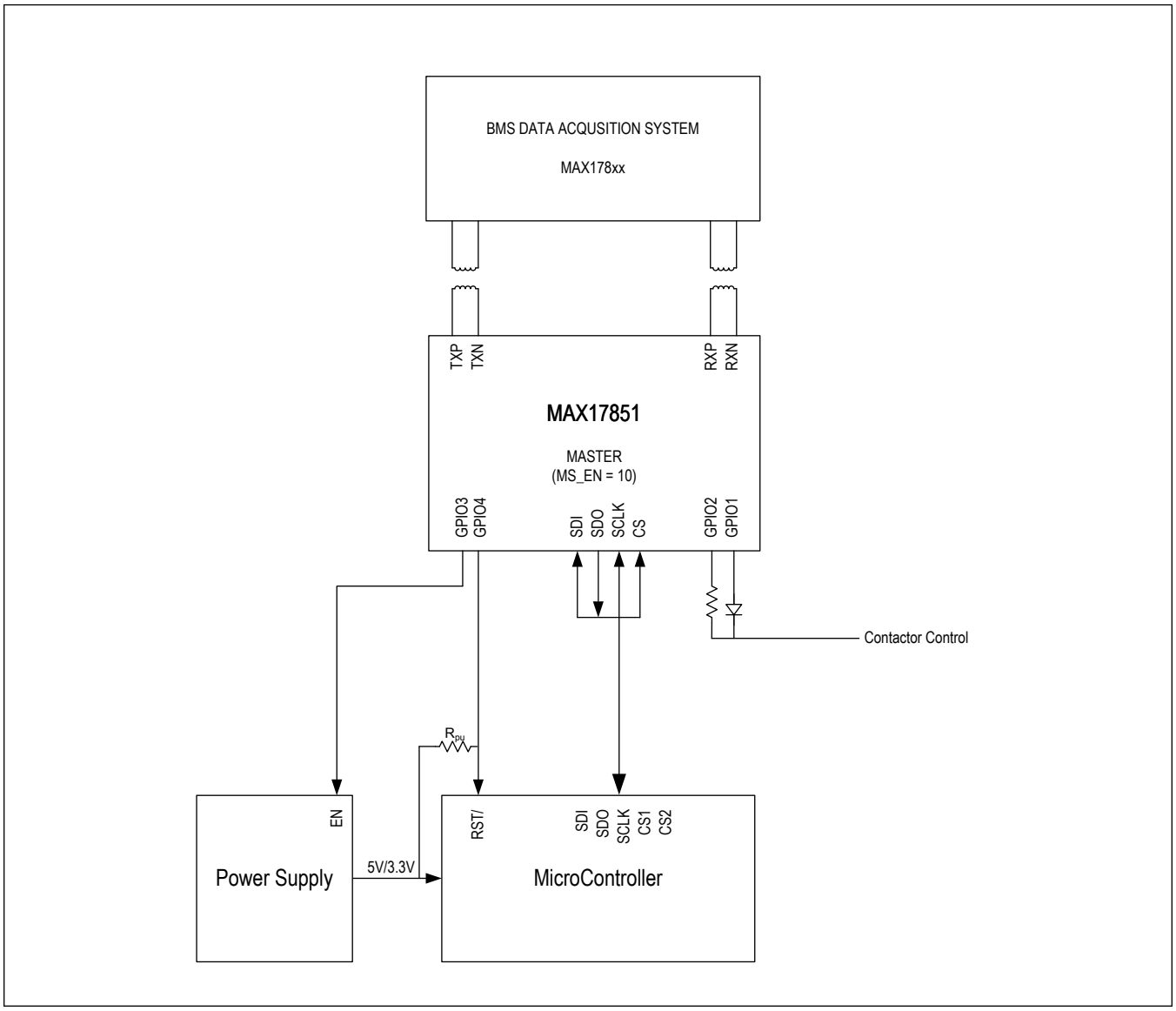


Figure 21. Single UART

Single UART MAX17851 Initialization

The example shown in Table 20 details a sequence for configuring the MAX17851 in single UART configuration. In addition to this sequence, the host can always verify all the writes to the configuration registers by reading back the register data. This adds an extra layer of verification.

Table 20. Single UART Device Configuration Sequence

SPI DIN	SPI DOUT	DESCRIPTION
TRANSACTION 1		Configure CONFIG_GEN0.
60h	xxh	Write device count.
07h	xxh	Set device count to 7 (change based on actual daisy-chain count).
TRANSACTION 2		Configure CONFIG_GEN1.
62h	xxh	Configure baud rate.
30h	xxh	Set baud rate = 2Mbps.
TRANSACTION 3		Configure CONFIG_GEN4.
68h	xxh	Configure MS_EN.
28h	xxh	Enable master single UART.
TRANSACTION 4		Configure CONFIG_GEN5.
6Ah	xxh	Configure ALRTPCKT_DBNC.
08h	xxh	Enables two consecutive errors in BMS Safe Monitoring or Sleep mode before error is flagged.
TRANSACTION 5		Configure CONFIG_SAFEMON0.
6Ch	xxh	Configure GPIOREC_DLY.
5Ah	xxh	Configures 1sec delay between assertion of GPIO1,2 and GPIO3,4. This perform the GPIO safety measure in BMS Safe Monitoring mode.
TRANSACTION 6		Configure CONFIG_SAFEMON1.
6Eh	xxh	Configure CONT_TIMER_DLY.
FEh	xxh	Provides 1032 min delay upon communication error (broken wire or persistent communication fault) before driving signal to open contactor.
TRANSACTION 7		Configure CONFIG_SAFEMON2.
70h	xxh	Configure SAFEMON_SCAN_DLY.
01h	xxh	Enables 1s fault polling of BMS Data Acquisition System.
TRANSACTION 8		Configure CONFIG_SAFEMON3.
72h	xxh	Configure.
0Fh	xxh	Masks GPIO1, GPIO2, GPIO3, GPIO4 output upon initial entry into BMS Safe Monitoring mode. Provides time to perform GPIO safety measures.
TRANSACTION 9		Configure CONFIG_SLP.
74h	xxh	Configure SLP_SCAN_DLY, SLP_ALRTPCKTEN, SLP_CBNTFY.
1Fh	xxh	Enables 1s auto polling of BMS Data Acquisition System when Sleep mode is entered, interrupt driven response to microcontroller.
TRANSACTION 10		Configure CONFIG_COMM.
76h	xxh	Configure COMM_RTRY, COMM_TO_DLY.
02h	xxh	For daisy-chain = 7 READALL timing is 251us. COMM_TO_DLY is recommended to be 30% longer than the message timing.
TRANSACTION 11		Configure GPIO12.
80h	xxh	Configure GPIO1_CFG and GPIO2_CFG.
66h	xxh	Enables the contactor drivers for NMOS input (BMS Safe Monitoring Mode only).
TRANSACTION 12		Configure GPIO34.
82h	xxh	Configure GPIO3_CFG and GPIO4_CFG.
46h	xxh	Enables master/slave communication transition in BMS Safe Monitoring mode. Enables microcontroller active high reset.

Table 20. Single UART Device Configuration Sequence (continued)

SPI DIN	SPI DOUT	DESCRIPTION
TRANSACTION 13		Configure ALRTEN_OPSTATE.
28h	xxh	Configure SLP_ALRTEN,SLP_STATUS_ERR_ALRTEN,SAFEMON_ALRTEN, SAFEMON_GPIO12_ALRTEN, SAFEMON_STATUS_ERR_ALRTEN,SAFEMON_CONFIG_ERR_ALRTEN.
3Fh	xxh	Enables Alert pin assertion for the configured Sleep and BMS Safe Monitoring mode alerts.
TRANSACTION 14		Configure ALRTEN_LSSM_BYTE.
24h	xxh	Configure RX_READY_ALRTEN.
80h	xxh	Enables Alert pin assertion for the RX_READY Alert for LSSM status byte.
TRANSACTION 15		Configure ALRTEN_RX.
20h	xxh	Configure RX_STOP_ALRTEN.
02h	xxh	Enables Alert pin assertion for the RX_STOP.
TRANSACTION 16		Configure ALRTEN_WD.
2Ch	xxh	Configure WD_TO_ERR_ALRTEN.
10h	xxh	Enables Alert pin assertion for watchdog timeout. This allows the synchronization between the host microcontroller and the MAX17851.
TRANSACTION 17		Check for Receive buffer errors.
11h	xxh	Read ALERT_RX register flags.
xxh	00h	If no errors, continue; otherwise, clear and go to error routine.

See the *BMS Data Acquisition System Initialization for Single UART* section for details on how to configure the system after the MAX17851 device configuration is complete.

Single UART BMS Data Acquisition System Initialization

The example shown in Table 21 outlines the sequence for initialization of two UART slave devices.

Table 21. Single UART BMS Daisy-Chain Initialization Sequence

SPI DIN	SPI DOUT	DESCRIPTION
TRANSACTION 1		Configure CONFIG_GEN3.
66h	xxh	Configure ALRTPCKT_TIMING.
05h	xxh	Set Keep-Alive period to 160µs.
TRANSACTION 2		Configure ALRTEN_RX.
20h	xxh	Configure RX_ERR_ALRTEN and RX_OVRFLW_ERR_ALRTEN.
88h	xxh	Enables the RX_ERR_ALRTEN and RX_OVRFLW_ERR_ALRTEN bits ($\overline{\text{ALERT}}$ pin enable bits).
TRANSACTION 3		Configure CONFIG_GEN2.
64h	xxh	Configure TX_PREAMBLES and TX_QUEUE transmission.
30h	xxh	Enables Transmit of Preambles (this would wake up daisy-chain devices).
TRANSACTION 4		Read STATUS_RX.
01h	xxh	Read STATUS.
xxh	21h	If RX_STATUS = 21h, continue. Otherwise, repeat transaction until true or timeout.
TRANSACTION 5		Configure CONFIG_GEN2.
64h	xxh	Configure TX_PREAMBLES and TX_QUEUE Transmission.
10h	xxh	Disable Transmit of Preambles.
TRANSACTION 6		CLR_RXBUF

Table 21. Single UART BMS Daisy-Chain Initialization Sequence (continued)

SPI DIN	SPI DOUT	DESCRIPTION
42h	xxh	Clears and reset Receive buffer.
00h	xxh	Clears and reset Receive buffer.
TRANSACTION 7		CLR_TXBUF
40h	xxh	Clears and reset Transmit buffer.
00h	xxh	Clears and reset Transmit buffer.
TRANSACTION 8		Load the HELLOALL command sequence into the load queue.
C0h	xxh	Write the LD_Q SPI command byte (write the load queue).
03h	xxh	Message length.
57h	xxh	HELLOALL command byte.
00h	xxh	Register address (0x00).
00h	xxh	Initialization address of HELLOALL.
TRANSACTION 9		Read RX_RD_MSG (Optional).
C2h	xxh	Verify contents of load queue.
00h	xxh	Change the pointer to the load queue to be read.
TRANSACTION 10		Read buffer content (optional).
C1h	xxh	Read address for the load queue.
xxh	03h	Message length.
xxh	57h	HELLOALL command byte.
xxh	00h	Register address.
xxh	00h	Initialization address of HELLOALL.
TRANSACTION 11		Transmit HELLOALL sequence.
B0h	xxh	NXT_LDQ SPI command byte (write the next load queue).
TRANSACTION 12		Read STATUS_RX.
01h	xxh	Poll RX_STOP Status bit.
xxh	x2h	If RX_Status[1] is true, continue. If false, then repeat transaction until true.
TRANSACTION 13		Read RX_RD_NXT_MSG.
93h	xxh	Service Receive buffer. Read the HELLOALL message that propagated through the daisy-chain and was returned back to the MAX17851. The host should verify the device count.
xxh	57h	Sent command byte (HELLOALL).
xxh	00h	Sent address = 00h
xxh	02h	Returned address = 02h (2 devices)
xxh	84h	LSSM byte.
TRANSACTION 14		Read ALERT_RX.
11h	xxh	Check for Receive buffer error.
xxh	00h	If no errors, continue. Otherwise, clear and go to error routine.

See the *Final Device Configuration* section on details for further configuration of the MAX17851.

Final MAX17851 Configuration

After the device is successfully initialized and there are no errors, the host microcontroller can continue with normal operation. The host can also disable the automatic keep alive generation from the MAX17851 and configure the device

with Alert Packet generation. The Alert Packet configuration sequence is shown below in Table 22.

Table 22. Alert Packet Configuration

SPI DIN	SPI DOUT	DESCRIPTION
TRANSACTION 1		Configure CONFIG_GEN2.
64h	xxh	Configure TX_PREAMBLES.
10h	xxh	Disable Transmit Preambles mode.
TRANSACTION 2		Configure CONFIG_GEN3.
66h	xxh	Configure ALRTPCKT_TIMING.
08h	xxh	Alert packet timing of 1.28ms.
TRANSACTION 3		Configure CONFIG_GEN4.
68h	xxh	Configure CO_ALRTPCKTEN.
A8h	xxh	Configures Alert Packet timing in Commanded Operation mode. In sleep or BMS Safe Monitoring mode timing is dictated by SLP_SCAN_DLY and SAFEMON_SCAN_DLY.
TRANSACTION 4		Configure CONFIG_SLP.
74h	xxh	Configure SLP_SCAN_DLY, SLP_ALRTPCKTEN.
3Fh	xxh	Enables SLP_SCAN_DLY to 1s. SLP_ALRTPCKTEN enables in order to use the delay configured by SLP_SCAN_DLY. Configure SLP_CBNTFY to off (user selected).

Configuration Memory Sequence for Daisy-Chain Configuration

The following table describes a typical procedure for configuration of the MAX17853 or MAX17854 for proper operation within BMS Safe Monitoring mode. All the critical threshold settings in this table are shown as an example of configuration for the auto polling operation by the BMS Data Acquisition System in the BMS Safe Monitoring mode. It should be noted that the threshold, timers, cell balancing, and other configurations are only examples to demonstrate the procedure for configuration memory programming and transmission. The actual settings would depend on the system requirements and how the user would like to configure the part in BMS Safety Monitoring mode. In this mode, the BMS Data Acquisition System would measure the critical parameters and transmit any fault information to the MAX17851 device.

Table 23. Configuration Memory Sequence

SPI TRANSACTION	SPI DESCRIPTION	DESCRIPTION OF UART COMMAND
D0	Address Configuration Memory	
60	Write Config Memory QUEUE0_BLOCK0_RA1	Write 0x00FF: All auxiliary inputs configured for Ratiometric mode.
60	Write Config Memory QUEUE0_BLOCK0_RA2	Redundant Write 0x00FF: All auxiliary inputs configured for Ratiometric mode.
FF	Write Config Memory QUEUE0_BLOCK0_DATALSB	
00	Write Config Memory QUEUE0_BLOCK0_DATAMSB	
64	Write Config Memory QUEUE0_BLOCK1_RA1	Write: 0x7FFF: Cells and block measurement enabled.
64	Write Config Memory QUEUE0_BLOCK1_RA2	Redundant write 0x7FFF.

Table 23. Configuration Memory Sequence (continued)

FF	Write Config Memory QUEUE0_BLOCK1_DATALSB	
7F	Write Config Memory QUEUE0_BLOCK1_DATAMSB	
65	Write Config Memory QUEUE0_BLOCK2_RA1	Write: 0x003F: Auxiliary measurement enabled.
65	Write Config Memory QUEUE0_BLOCK2_RA2	Redundant write 0x003F.
3F	Write Config Memory QUEUE0_BLOCK2_DATALSB	
00	Write Config Memory QUEUE0_BLOCK2_DATAMSB	
6B	Write Config Memory QUEUE0_BLOCK3_RA1	Write: 0x0012: Disable the HVMUX CTST. Enable the built in diagnostic (user selected). In this case, selection is made of die temperature and an ADC reference diagnostic.
6B	Write Config Memory QUEUE0_BLOCK3_RA2	Redundant write: 0x0012:
12	Write Config Memory QUEUE0_BLOCK3_DATALSB	
00	Write Config Memory QUEUE0_BLOCK3_DATAMSB	
6C	Write Config Memory QUEUE0_BLOCK4_RA1	Write:0x0000: Disable the cell input diagnostic current sources.
6C	Write Config Memory QUEUE0_BLOCK4_RA2	Redundant write 0x0000
00	Write Config Memory QUEUE0_BLOCK4_DATALSB	
00	Write Config Memory QUEUE0_BLOCK4_DATAMSB	
6D	Write Config Memory QUEUE0_BLOCK5_RA1	Write:0x0000: Disable the auxiliary input diagnostic current sources.
6D	Write Config Memory QUEUE0_BLOCK5_RA2	Redundant write 0x0000
00	Write Config Memory QUEUE0_BLOCK5_DATALSB	
00	Write Config Memory QUEUE0_BLOCK5_DATAMSB	
6E	Write Config Memory QUEUE0_BLOCK6_RA1	Write:0x0000: Disable the auxiliary input application of diagnostic current sources.
6F	Write Config Memory QUEUE0_BLOCK6_RA2	Write 0x0000: All balancing switches OFF for auto polling operation only.
00	Write Config Memory QUEUE0_BLOCK6_DATALSB	
00	Write Config Memory QUEUE0_BLOCK6_DATAMSB	
	PEC Calculation for Block0-6 = 0xAD	
	RESERVED BYTE	

Table 23. Configuration Memory Sequence (continued)

20	Write Config Memory QUEUE1_BLOCK7_RA1	Write 0xE664: Setting the threshold for cell Over Voltage fault (user defined).
20	Write Config Memory QUEUE1_BLOCK7_RA2	Redundant write 0xE664: Setting the threshold for cell Over Voltage fault (user defined).
64	Write Config Memory QUEUE1_BLOCK7_DATALSB	
E6	Write Config Memory QUEUE1_BLOCK7_DATAMSB	
22	Write Config Memory QUEUE1_BLOCK8_RA1	Write 0x8CCC: Setting the threshold for cell Under Voltage fault (user defined).
22	Write Config Memory QUEUE1_BLOCK8_RA2	Redundant Write 0x8CCC: Setting the threshold for cell Under Voltage fault (user defined).
CC	Write Config Memory QUEUE1_BLOCK8_DATALSB	
8C	Write Config Memory QUEUE1_BLOCK8_DATAMSB	
1A	Write Config Memory QUEUE1_BLOCK9_RA1	Write 0x7FFF: Enables the Over Voltage alert for cell and BLOCK input.
1A	Write Config Memory QUEUE1_BLOCK9_RA2	Redundant write 0x7FFF.
FF	Write Config Memory QUEUE1_BLOCK9_DATALSB	
7F	Write Config Memory QUEUE1_BLOCK9_DATAMSB	
1B	Write Config Memory QUEUE1_BLOCK10_RA1	Write 0x7FFF: Enables the Under Voltage alert for cell and BLOCK input.
1B	Write Config Memory QUEUE1_BLOCK10_RA2	Redundant write 0x7FFF.
FF	Write Config Memory QUEUE1_BLOCK10_DATALSB	
7F	Write Config Memory QUEUE1_BLOCK10_DATAMSB	
31	Write Config Memory QUEUE1_BLOCK11_RA1	Write 0xE8BC: Setting the threshold for auxiliary Over Voltage fault (user defined).
31	Write Config Memory QUEUE1_BLOCK11_RA2	Redundant write 0xE8BC: Setting the threshold for auxiliary Over Voltage fault (user defined).
BC	Write Config Memory QUEUE1_BLOCK11_DATALSB	
E8	Write Config Memory QUEUE1_BLOCK11_DATAMSB	
33	Write Config Memory QUEUE1_BLOCK12_RA1	Write 0x26C8: Setting the threshold for Auxiliary Under Voltage fault (user defined)
33	Write Config Memory QUEUE1_BLOCK12_RA2	Redundant Write 0x26C8: Setting the threshold for Auxiliary Under Voltage fault (user defined)
C8	Write Config Memory QUEUE1_BLOCK12_DATALSB	
26	Write Config Memory QUEUE1_BLOCK12_DATAMSB	

Table 23. Configuration Memory Sequence (continued)

1C	Write Config Memory QUEUE1_BLOCK13_RA1	Write 0x003F: Enables the auxiliary Over Voltage alert.
1C	Write Config Memory QUEUE1_BLOCK13_RA2	Redundant write 0x003F.
3F	Write Config Memory QUEUE1_BLOCK13_DATALSB	
00	Write Config Memory QUEUE1_BLOCK13_DATAMSB	
	PEC Calculation for Block7-13 = 0x8C	
	RESERVED BYTE	
1D	Write Config Memory QUEUE2_BLOCK14_RA1	Write 0x003F: Enables the auxiliary Under Voltage alert.
1D	Write Config Memory QUEUE2_BLOCK14_RA2	Redundant write 0x003F.
3F	Write Config Memory QUEUE2_BLOCK14_DATALSB	
00	Write Config Memory QUEUE2_BLOCK14_DATAMSB	
1F	Write Config Memory QUEUE2_BLOCK15_RA1	Write 0xD708: Cell Over Voltage alert clear threshold set.
1F	Write Config Memory QUEUE2_BLOCK15_RA2	Redundant write 0xD708: Cell Over Voltage alert clear threshold set.
D7	Write Config Memory QUEUE2_BLOCK15_DATALSB	
08	Write Config Memory QUEUE2_BLOCK15_DATAMSB	
21	Write Config Memory QUEUE2_BLOCK16_RA1	Write 0x9998: Cell Under Voltage alert clear threshold set (user defined).
21	Write Config Memory QUEUE2_BLOCK16_RA2	Redundant write 0xD708: Cell Under Voltage alert clear threshold set (user defined).
98	Write Config Memory QUEUE2_BLOCK16_DATALSB	
99	Write Config Memory QUEUE2_BLOCK16_DATAMSB	
30	Write Config Memory QUEUE2_BLOCK17_RA1	Write 0xD554: Auxiliary Over Voltage alert clear threshold set (user defined).
30	Write Config Memory QUEUE2_BLOCK17_RA2	Redundant Write 0xD554: Auxiliary Over Voltage alert clear threshold set (user defined).
54	Write Config Memory QUEUE2_BLOCK17_DATALSB	
D5	Write Config Memory QUEUE2_BLOCK17_DATAMSB	
32	Write Config Memory QUEUE2_BLOCK18_RA1	Write 0x4D94: Auxiliary Under Voltage alert clear threshold set (user defined).
32	Write Config Memory QUEUE2_BLOCK18_RA2	Redundant write 0x4D94: Auxiliary Under Voltage alert clear threshold set (user defined).
94	Write Config Memory QUEUE2_BLOCK18_DATALSB	

Table 23. Configuration Memory Sequence (continued)

4D	Write Config Memory QUEUE2_BLOCK18_DATAMSB	
70	Write Config Memory QUEUE2_BLOCK19_RA1	Write 0x03FF: This configures the expiration time for the cell balancing operation and is set to 0x3FF for the indefinite operation or in other word no expiry. It should be noted that auto polling only exits based on the timer or a timeout criteria.
70	Write Config Memory QUEUE2_BLOCK19_RA2	Redundant write 0x03FF:
FF	Write Config Memory QUEUE2_BLOCK19_DATALSB	
03	Write Config Memory QUEUE2_BLOCK19_DATAMSB	
80	Write Config Memory QUEUE2_BLOCK20_RA1	Write 0x1233: A cell balancing control register that initiates and controls the different cell balancing modes and operation. This configuration is setting the CBMODE of the daisy-chain device in auto group by sec. This command effectively starts the auto polling after the daisy-chain is configured.
80	Write Config Memory QUEUE2_BLOCK20_RA2	Redundant write 0x1233:
12	Write Config Memory QUEUE2_BLOCK20_DATALSB	
33	Write Config Memory QUEUE2_BLOCK20_DATAMSB	
	PEC Calculation for Block14-20 = 0xC3	
	RESERVED BYTE	

Write Config Memory QUEUE0_BLOCK2_DATAMSB

Watchdog Configuration

Because the watchdog is disabled by default (WD_EN = '0'), the host microcontroller must configure and enable it after startup. If the host microcontroller becomes unresponsive, the watchdog times out and enters BMS Safe Monitoring mode.

The MAX17851 watchdog functionality can be enabled and configured using the following sequence.

Table 24. Watchdog Configuration

SPI DIN	SPI DOUT	DESCRIPTION
TRANSACTION 1		Configure CONFIG_WD2.
88h	xxh	Configure WD_DBNC.
07h	xxh	Configures the number of watchdog violations that have to happen before entering BMS Safe Monitoring mode.
TRANSACTION 2		Configure WD_WD0.
84h	xxh	Configure WD_OPN and WD_CLO.
EFh	xxh	Enables WD_OPN and WD_CLO to less than watchdog timeout.
TRANSACTION 3		Configure WD1.
86h	xxh	Configure WD_1UD and WD_DIV.
FFh	xxh	Configures the extended watchdog window and watchdog clock divider.
TRANSACTION 4		Configure WD2.
88h	xxh	Configure WD_EN.

Table 24. Watchdog Configuration (continued)

SPI DIN	SPI DOUT	DESCRIPTION
87h	xxh	Enables the watchdog.
TRANSACTION 5		Monitor ALERT_WD.
1Ch	xxh	The watchdog functionality is enabled, and the Watchdog Alert register should be monitored.

Transaction Sequence for UART Write and Read

In the example shown in Table 25, the host communicates with two UART slave devices to:

- Write the value 7FFFh to the device register address 0x64 for all slave devices using a WRITEALL command sequence.
- Read back the value 7FFFh from the device register address 0x64 for all slave devices using a READALL command sequence.

This example assumes the slave devices are configured with the Alive Counter enabled. To execute these two command sequences, the host performs the SPI transactions listed in Table 25.

Table 25. Transaction Sequence for UART Write and Read

SPI DIN	SPI DOUT	DESCRIPTION
TRANSACTION 1		Load the WRITEALL command sequence into the load queue.
C0h	xxh	LDQ SPI command byte.
06h	xxh	Message length = 6.
02h	xxh	WRITEALL command byte.
64h	xxh	Register address of the device.
FFh	xxh	LS byte of register data to be written.
7Fh	xxh	MS byte of register data to be written.
24h	xxh	PEC byte for 02h, 64h, 7Fh, FFh.
00h	xxh	Alive-counter byte (seed value = 0).
TRANSACTION 2		Start transmitting the WRITEALL sequence from the transmit queue.
B0h	xxh	NXT_LDQ SPI command byte.
TRANSACTION 3		Check if a message has been received into the Receive buffer.
01h	xxh	Read RX_STATUS register.
xxh	x2h	If RX_STOP (bit 1] is true, continue. Otherwise, repeat until true or timeout.
TRANSACTION 4		Read Receive Buffer to verify the sent WRITEALL message.
93h	xxh	RX_RD_NXT_MSG SPI
xxh	02h	Sent command byte (WRITEALL).
xxh	64h	Sent address.
xxh	FFh	Sent LS byte.
xxh	7Fh	Sent MS byte.
xxh	02h	Alice-counter byte (= sent seed +2, if alive counter enabled).
xxh	84h	LSSM_BYTE
xxh	ECh	Calculated PEC.
TRANSACTION 5		Check for Receive buffer errors.
01h	xxh	Read RX_STATUS register.
xxh	00h	Check RX_ERR, RX_OVRFLW_ERR. If no errors, continue; otherwise, clear and go to error routine.

Table 25. Transaction Sequence for UART Write and Read (continued)

SPI DIN	SPI DOUT	DESCRIPTION
TRANSACTION 6		Load the READALL command sequence into the load queue.
C0h	xxh	NXT_LDQ SPI command byte.
09h	xxh	Message length.
03h	xxh	READALL command byte.
64h	xxh	Register address.
00h	xxh	Data-check byte (seed value = 00h).
A6h	xxh	PEC byte for bytes 03h, 64h, 00h.
00h	xxh	Alive Counter byte (seed value = 00h).
TRANSACTION 7		Start transmitting the READALL sequence.
B0h	xxh	NXT_LDQ SPI command byte.
TRANSACTION 8		Check if a message has been received into the Receive buffer.
01h	xxh	Read the RX_Status register.
xxh	x2h	If RX_STOP [bit 1] is true, continue. Otherwise, repeat until true or timeout.
TRANSACTION 9		Read the Receive buffer and verify that the device register data is what was written during the WRITEALL sequence.
93h	xxh	RX_RD_NXT_MSG SPI command byte.
xxh	03h	Sent command byte (READALL).
xxh	64h	Sent register address.
xxh	FFh	LS byte of device 1.
xxh	7Fh	MS byte of device 1.
xxh	FFh	LS byte of device 0.
xxh	7Fh	MS byte of device 0.
xxh	00h	Data-check byte (= 00h if all status bits have been cleared).
xxh	02h	Alive-counter byte (= sent seed + 2, if alive counter is enabled).
xxh	84h	LSSM_BYTE
xxh	D5h	PEC (for the previous 9 bytes).
TRANSACTION 10		Check for Receive buffer errors.
01h	xxh	Read RX_STATUS register.
xxh	00h	Check RX_ERR, RX_OVRFLW_ERR. If no errors, continue; otherwise, clear and go to error routine.

Additional Error Checking

The MAX17851 performs internal hardware verification and automatic UART message verification. The current status is attached to every transaction via the LSSM status byte.

If desired, the host may perform additional error checking and verification to ensure the integrity of the MAX17851 and BMS Data Acquisition System. These checks may include:

- Verification of write data received (matching values, number of bytes)
- Verification of read data received (allowed values, ranges, number of bytes)
- Verification of the received PEC, data-check, and alive-counter bytes
- Verification of FMEA registers of connected daisy-chain devices

Corrupted Preamble Character

If the received preamble is corrupt, the message is not written to the Receive buffer. This condition is detected by the expiration of the communications timeout timer and result in the COMM_ERR flag being set.

For multiple identical message transmissions (e.g., if the host is polling a daisy-chain device register), this condition may also be detected by the transmission and verification of unique Alive Counter seeds. When the automatic Alive Counter is enabled, this process is automatic, and an error is reported in ALIVECOUNT_ERR. When the manual Alive Counter is enabled, the host must manually increment and verify the Alive Count seed. The Alive Counter byte is sent after the PEC. As such, the PEC is not affected by the Alive Count value.

Corrupt Message Content

Manchester, parity, and PEC errors are indications that the data in the message may be corrupt. For each UART message received, the host should perform the appropriate computations on any error-checking data available in the received message. This data includes:

- Data Check byte: Error status provided by the slave device(s); sent and returned on reads of slave device data as described in the slave device data sheet.
- LSSM Status byte: Error status provided by the MAX17851 and appended to each received message.
- PEC: CRC-8 packet error-checking byte protecting each received message.
- Alive Counter byte: Used to verify the number of devices responding to a transmitted message; can be sent and received with every message as described in the slave device data sheet.
- RX_ERR_ALRT bit: A Manchester and/or parity error is detected in at least one of the received characters in the message during the course of a Receive buffer read transaction.

Note: Because Manchester and parity errors can be introduced anywhere in the UART data stream, the errors denoted by RX_ERR_ALRT are not necessarily reflected in the error checking bytes returned by the slave device(s). Therefore, the host should check and clear this flag after, but not before, reading each message in the Receive buffer. The host should set the RX_ERR_ALRTEN bit to be notified of such an event.

Corrupt or Missing Stop Character

If a stop character is corrupt or missing, there is no data loss, because the message is still framed either by a subsequent valid stop character (automatically sent in Keep-Alive mode), a frame timeout, or by the preamble of the next UART message. A corrupt stop character may be interpreted as a data character, and as such may be stored in the Receive buffer. In this case, if a valid stop character is eventually received before the next preamble, the message length is one byte too long. The host should check for this condition by computing the received message length and comparing it to the expected message length.

Before reading the next message, the host may also verify that all of the following are true:

1. The last byte in the message is a null byte (00h).
2. The RX_LAST_BYTE bit is set.
3. The RX_BYTE_ERR bit is cleared.

Unintended Preamble

The presence of an unintended preamble in the middle of a message creates an unintended message in the Receive buffer. If the unintended preamble is the result of a corrupt data character within a message, then the message is prematurely terminated and a second, unexpected message is created. This event can be detected by comparing the number of received bytes in the message to the expected number.

Unintended Stop Character

The presence of an unintended stop character prematurely terminates the message. This is detected by comparing the number of received bytes in the message to the expected number.

Fault Handling Guidelines

Table 26. Fault Handling Guidelines

ERROR BIT	REGISTER	BIT POSITION	ERROR BIT DESCRIPTION	PRIMARY ACTION FOR INTERMITTENT ERRORS	SECONDARY ACTION FOR PERSISTENT ERRORS

Table 26. Fault Handling Guidelines (continued)

ALRTPCKT_STATUS_ERR	STATUS_LSSM_BYTE	6	A safety critical STATUS register error within one or more of the daisy-chain units is reported within the ALRTPCKT message.	1. Read ALRTPCKT_RD_MSG to determine daisy-chain error(s) and device location(s) of error 2. Diagnose cause of error(s) and follow required actions dictated by system safety requirements.	Follow required actions dictated by system safety requirements.
COMM_ERR	STATUS_LSSM_BYTE	5	A communication error is observed with corrupted, inserted, or omitted data within the UART message.	Resend communication command.	Report communication failures and consider terminating normal system behavior.
ALRTPCKT_COMM_ERR	STATUS_LSSM_BYTE	4	A communication error is observed in the auto-generated ALERTPACKET message which corrupted, inserted, or omitted data.	1. Increase COMM_RTRY counter to decrease sensitivity to communication channel noise. 2. Decrease the frequency of issuance of the ALRTPACKET_TIMING. Normal operation is still permissible.	Verify that communication to the daisy-chain is still possible. If errors within user defined message, re-initialize the daisy-chain assuming a wire break. Report communication failures and consider terminating normal system behavior.
ALRTPCKT_HW_ERR	STATUS_LSSM_BYTE	4	A hardware error is reported within the ALERTPACKET buffer.	Manually query to the daisy-chain STATUS register and compare to the ALERTPACKET STATUS register. If the STATUS registers match, the error effected the redundant buffer only. Normal operation is still permissible.	Manually query to the daisy-chain STATUS register and compare to the ALERTPACKET STATUS register. If the STATUS registers match, the error effected the redundant buffer only. Normal operation is still permissible.

Table 26. Fault Handling Guidelines (continued)

COMM_MSMTCH_ERR	STATUS_LSSM_BYTE	3	An unexpected communication packet is received that differs from what is transmitted.	1. Reset the LSSM buffer, TX buffer, and RX buffer. 2. Resend communication command.	Report communication failures and consider terminating normal system behavior.
ALIVECOUNT_ERR	STATUS_LSSM_BYTE	1	The rolling Alive Counter returned value does not match the seed incremented by the number of BMS daisy-chained units.	1. Verify the integrity of the returned data message (message is free of error if COMM_ERR and COMM_MSMTCH_ERR are a logical 0). 2. Continue to transmit communication packets and monitored returned value. Normal operation is still permissible.	Disable rolling Alive Counter. Normal operation is still permissible.
HW_ERR	STATUS_LSSM_BYTE[0]	0	A hardware error is reported, which may affect device operation. Errors include supply voltages that exceed specified operating range, oscillator drift errors that may effect communication timing, corrupt device trim, corrupt memory (stuck at or transient) and internal register errors.	Issue a SWPOR and re-initialize the device. or Verify configuration registers and re-initialize if required, and resend communication command.	Report communication failures and consider terminating normal system behavior.
GPIO_ERR	STATUS_GEN	4	The GPIO output level does not match the intended programmed value.	Perform SAFEMON GPIO verification diagnostic	Report communication failures and consider terminating normal system behavior.
ALRTRST	ALERT_OPSTATE	7	Supply voltages tripped the device POR, causing a reset.	Reinitialize the daisy-chain.	Report communication failures and consider terminating normal system behavior.

Table 26. Fault Handling Guidelines (continued)

DEV_COUNT_ERR	STATUS_GEN	6	The HELLOALL returns a device address (DA) that does not match the programmed DEV_COUNT bitfield.	Reinitialize the daisy-chain.	Report communication failures and consider terminating normal system behavior.
WD_ERR	STATUS_GEN	5	The watchdog response is not configured appropriately.	Resend watchdog command.	If system microcontroller remains functional with valid communication, SWPOR can be issued, and the system must be re-initialized. Report failure to system and consider terminating normal system behavior.
DATAPATH_ERR	STATUS_GEN	3	A hardware error occurs during Datapath and Configuration Memory Verification.	1. Verify the data and PEC of the Configuration Memory queue 4 and the data and PEC of queue 1 -3. 2. Perform Configuration Memory and Datapath diagnostic using VER_CONFIG.	Report communication failures and consider terminating normal system behavior.
SPI_ERR	STATUS_GEN	6	A SPI transaction is rejected due to device operation within Sleep or BMS Safe Monitoring mode.	Configure device while operating in Commanded Operation mode.	Configure device while operating in Commanded Operation mode.
SLP_STATUS_ERR	STATUS_OPSTATE	4	An error within one or more of the daisy-chain units is reported during Sleep mode operation with SLP_ALRTPCKT_EN = 1.	1. Read ALRTPCKT_RD_MSG to determine daisy-chain error(s) and device location(s) of error. 2. Diagnose cause of error(s) and follow required actions dictated by system safety requirements.	Follow required actions dictated by system safety requirements.

Table 26. Fault Handling Guidelines (continued)

SAFEMON_STATUS_ERR	STATUS_OPSTATE	1	An error within one or more of the daisy-chain units is reported during BMS Safe Monitoring mode.		Critical system failure experienced and system forced into safe state.

Sleep Mode Configuration

MAX17851 Sleep mode is designed to autonomously monitor safety critical parameters in the daisy-chain devices while the host microcontroller is in a low power mode.

Sleep mode is entered by asserting the SLP_EN bit in the SLP_EN command register. Sleep mode cannot be entered if verify memory config or load config processes are active or in BMS Safe Monitoring mode. SLP_EN is also auto-cleared when Sleep mode is exited.

When the MAX17851 transitions from Commanded Operation to Sleep mode, auto-generated Alert Packets programmed in CO_ALRTPACKET and ALRTPCKT_TIMING are disabled. In Sleep mode, Alert Packets are enabled with the SLP_ALRTPCKTEN register, and the Alert Packet frequency is configured with the SLP_SCAN_DLY register.

The daisy-chain devices can be configured for auto polling/cell balancing operation while the MAX17851 is in Sleep mode. The host can program the Cell Balancing timer expiration in CONFIG_SLP register using the SLP_CBNTFY bits. When the SLP_CBNTFY timer expires, the device automatically exits Sleep mode. The SLP_ALRT bit in ALERT_OPSTATE register flags, and the ALERT pin asserts to wake the host.

Hardware-In-The-Loop (HIL) Testing Using the TX_AUTO Feature

The MAX17851 supports Hardware-In-The-Loop (HIL) testing with the TX_AUTO feature.

When two MAX17851s are populated in a master/slave application circuit, and the TX/RX lines of the master are bypassed from the daisy-chain and connected to the RX/TX lines of the slave, the slave can be programmed to automatically respond to a master transaction with any desired sequence (up to the maximum size of the TX buffer.

This is achieved by enabling the TX_AUTO bit on the slave device and programming it with the desired response via the LD_Q command. When enabled, TX_AUTO causes the MAX17851 to automatically transmit the contents of its Transmit buffer upon receipt of a preamble.

In this mode, the MAX17851 is configured as shown in Figure 22

HIL is intended for hardware verification and is not operational during SAFEMON or Sleep operation. The host microcontroller can inject and identify multiple data transmission or corruption issues (corrupt data, corrupt PEC, data check error, etc). This can be used to test the expected master response to a multitude of communication failures.

After the slave device is configured in Automatic Transmit mode (TX_AUTO = 1), the master device can transmit data to the slave device. When a valid preamble is received, the slave device transmits the data already in the load queue back to the master device. The host can evaluate the data and determine proper functionality.

The example shown below in the Hardware-In-The-Loop Test Sequence outlines the sequence for the 2 MAX17851 devices configured for testing after TX_AUTO = 1.

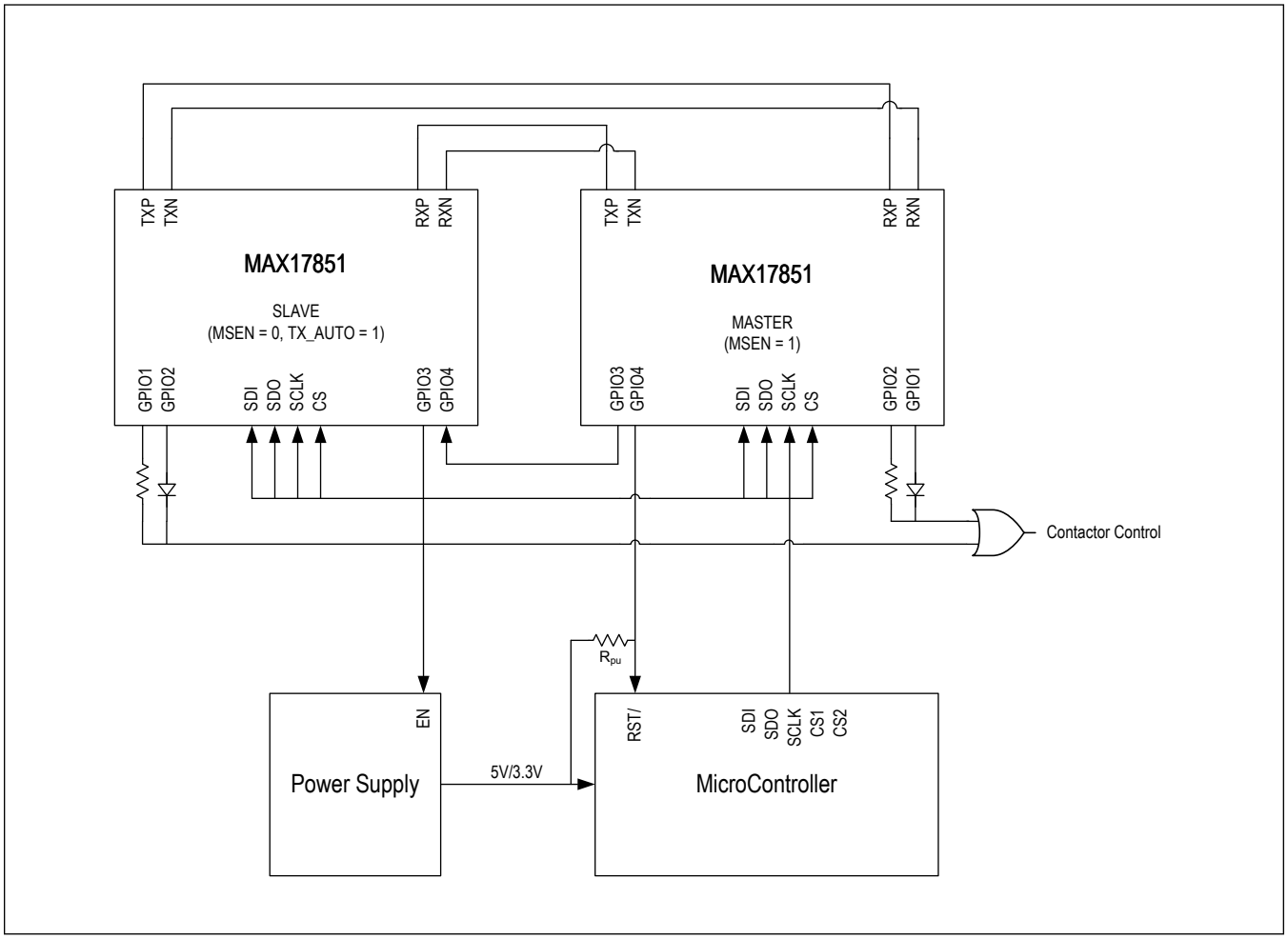


Figure 22. Hardware In Loop Test Setup

Table 27. Hardware-In-The-Loop Test Sequence

DIN	DOUT	DESCRIPTION
TRANSACTION 1		Enable Rx Interrupt flags for RX_Error and RX_Overflow.
20h	xxh	Write Alert Enable register
88h	xxh	Set the RX_ERR_ALRTEN and RX_OVRFLW_ERR_ALRTEN bits ($\overline{\text{ALERT}}$ pin enable bits).
TRANSACTION 2		Clear Receive buffer.
42h	xxh	Clear Receive buffer.
00h	xxh	
TRANSACTION 3		Clear Transmit buffer.
40h	xxh	Clear Transmit buffer.
00h	xxh	
TRANSACTION 4		Set CS for slave device
TRANSACTION 5		Set TX_AUTO of the slave device.

Table 27.
Hardware-In-The-Loop Test Sequence (continued)

DIN	DOUT	DESCRIPTION
66h	xxh	Configure TX_AUTO of slave device.
40h	xxh	Enables the Automatic Transmit mode of the slave device.
TRANSACTION 6		Load the READALL command sequence in the load queue of slave device.
C0h	xxh	WR_LD_Q SPI command byte.
06h	xxh	Message Length = 6.
03h	xxh	READALL command byte.
00h	xxh	Register Address of the device.
00h	xxh	LSB data.
00h	xxh	MSB data.
00h	xxh	Data Check byte.
F0h	xxh	PEC
TRANSACTION 7		Set CS for master device.
TRANSACTION 8		Load the READALL command sequence in load queue of master device.
C0h	xxh	WR_LD_Q SPI command byte.
06h	xxh	Message length = 6.
03h	xxh	READALL command byte.
00h	xxh	Register Address of the device.
00h	xxh	Data Check byte.
58h	xxh	PEC
TRANSACTION 9		Wait for null message to be received (poll RX_EMPTY Status bit).
01h	xxh	Read RX_Status register.
xxh	11h	
TRANSACTION 10		Transmit the load queue.
B0h	xxh	
TRANSACTION 11		Poll RX_STOP
01h	xxh	
xxh	x2h	
TRANSACTION 12		Read RX buffer.
93h	xxh	READALL command.
xxh	03h	Command byte.
xxh	00h	Register address.
xxh	00h	LSB data.
xxh	00h	MSB data.
xxh	00h	Data Check byte.
xxh	84h	LSSM byte.
xxh	70h	PEC
TRANSACTION 13		Read STATUS_RX.
01h	xxh	
xxh	11h	If no errors, proceed with normal operation; otherwise, the host needs to check errors.

Using a similar sequence, the user can also verify other system faults (no preambles, corrupt data, corrupt register

address, etc).

UART Physical Layer

UART Single Ended Mode Operation

By default, UART ports are configured for differential communication. For single-ended operation, the host can set the SINGLE_ENDED configuration bit. This mode enables the UART to receive a single-ended signal by shifting the input threshold negative so that zero differential voltage is a logic one. The RXP input is connected to ground, and the RXN input receives the inverted signal. In this mode, the TX port operates the same as in Differential mode.

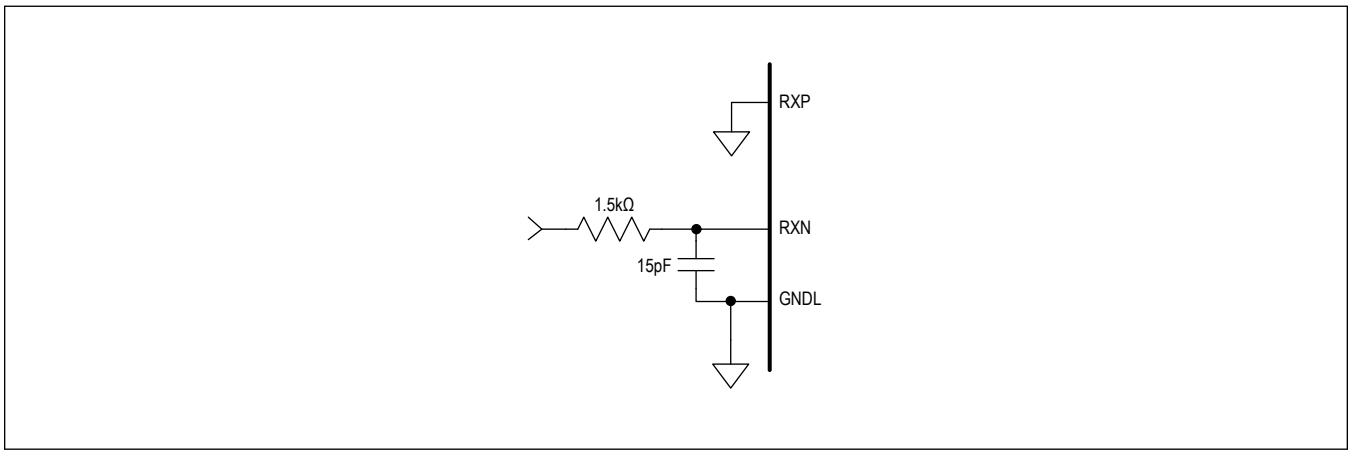


Figure 23. Single-Ended Mode

UART Transformer Coupling

The UART signals can be transformer-coupled because of the DC-balanced signaling. Placing an isolation transformer between the UART's transmitter and the slave device's receiver provides common-mode isolation for the case where the slave device is operating at a different voltage level. The center-tap of a signal transformer may be used to enhance common-mode rejection as shown in Figure 24 by AC-coupling the node to local ground. Common-mode currents that are able to pass through the parasitic coupling of the primary and secondary are shunted to ground to make a very effective common-mode noise filter.

When no data is being transmitted (idle state) and the TX_HI_Z register bit is 0, the transmitter drives both outputs to a logic low level to prevent any current flow through the transformer winding.

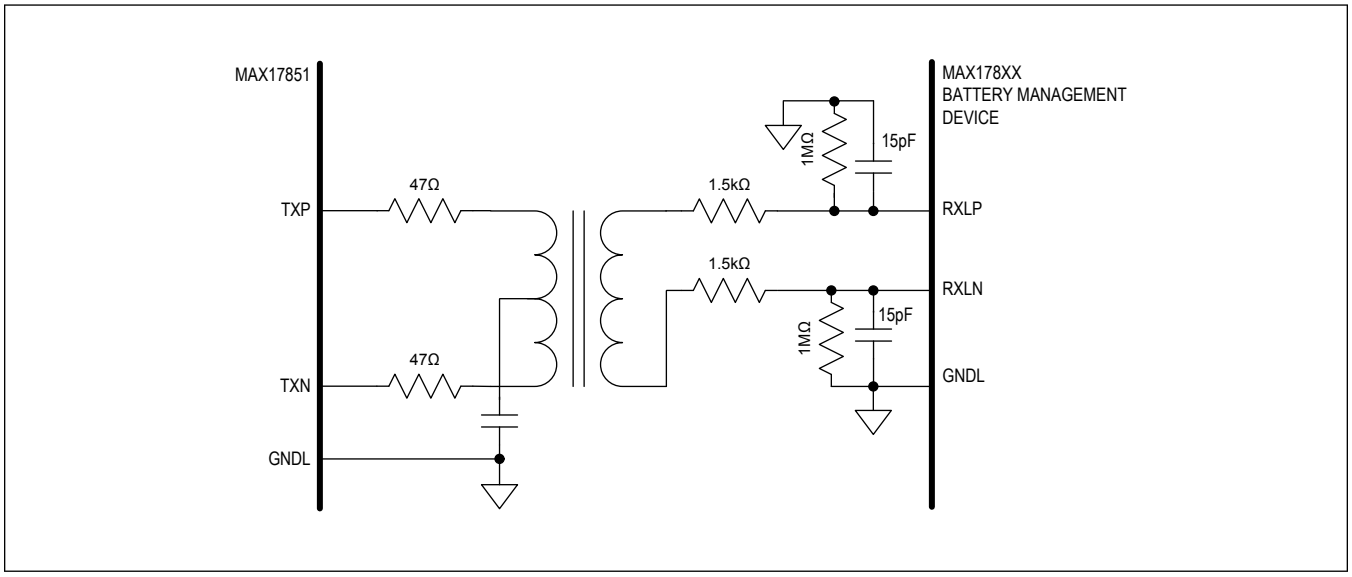


Figure 24. Transformer Coupling of UART Signals

A minimum transformer magnetizing inductance of 480uH is recommended.

UART Supplemental ESD Protection

The UART ports may require supplemental protection to meet IEC 61000-4-2 requirements for contact discharge. The recommended circuits to meet ±8kV protection levels are shown in Figure 25 and Figure 26. The protection components should be placed as near as possible to the signal's entry point on the PCB. The diodes should be placed as close as possible to the external connector.

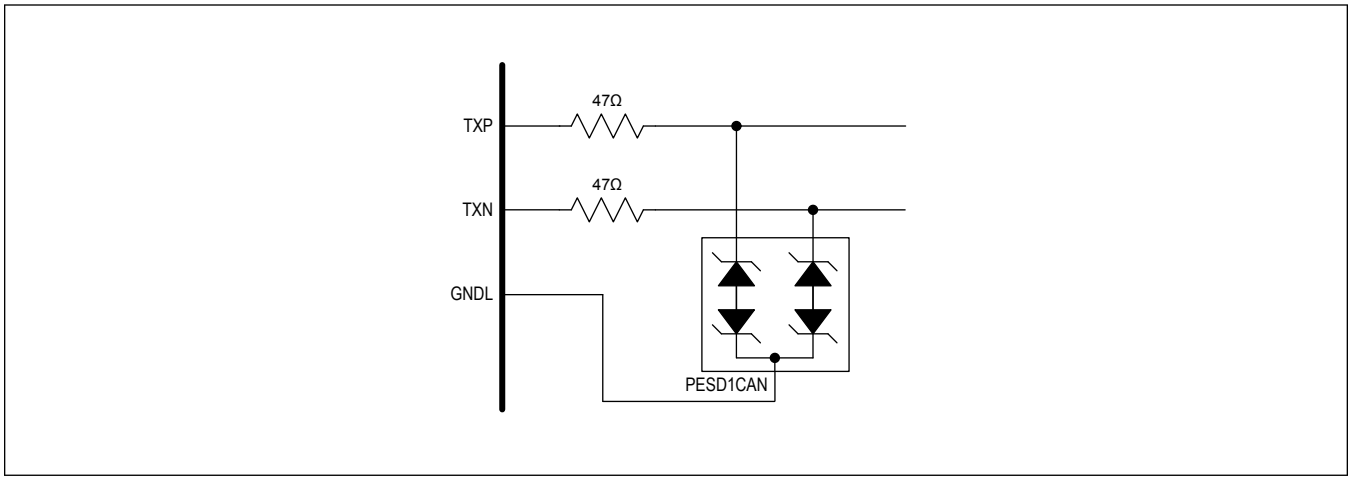


Figure 25. Supplemental ESD Protection for UART Transmitter

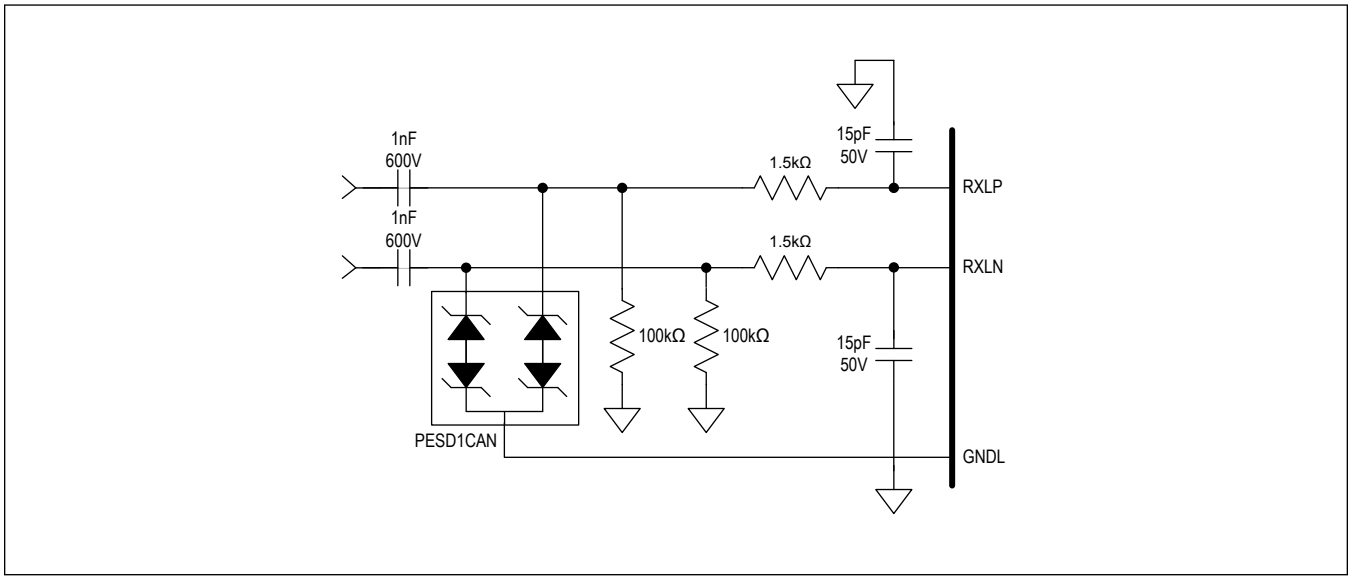


Figure 26. Supplemental ESD Protection for UART Receiver (Shown with Capacitive Coupling)

Power Supply Considerations

The MAX17851 has two regulator supplies. V_{DDL1} is a regulated supply that powers the internal oscillators and logic. V_{DDL2} is a regulated supply that powers the UART interface.

V_{DDL2} regulation may be bypassed and powered directly from a 3.3V or 5V power supply as shown below in Figure 27.

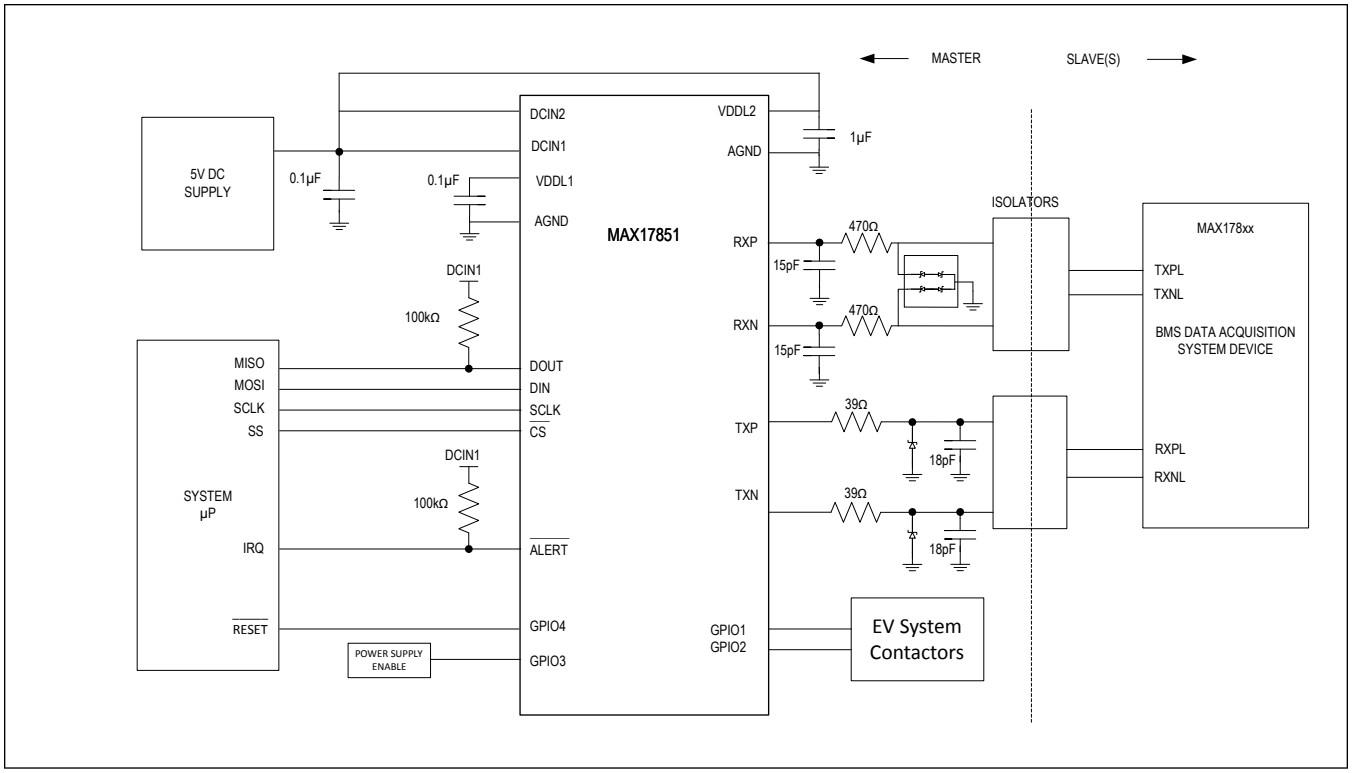


Figure 27. Power Supply Considerations

PCB Layout Recommendations

Careful PCB layout is critical to achieving the best accuracy performance and robust performance against environmental conditions.

1. Place decoupling capacitors close to their respective pins and on the same layer as the MAX17851. Capacitors should not share a ground return and each should via directly to the AGND internal layer.
2. UART RX and TX ports should be routed for an 100Ω differential impedance. UART ESD protection is recommended to be placed close the MAX17851 with the ground return via'ed directly to the AGND plane to clamp transient events before they can couple to other nodes, which may affect device performance.
3. SPI operation is driven by a system microcontroller located on the same ground plane as the MAX17851. Depending on trace length, optional source termination may be required to ensure that overshoot or undershoot does not violate the Absolute Maximum Operating Conditions. This source termination should be placed close to device transmit pins.

Layout Example (Silkscreens)

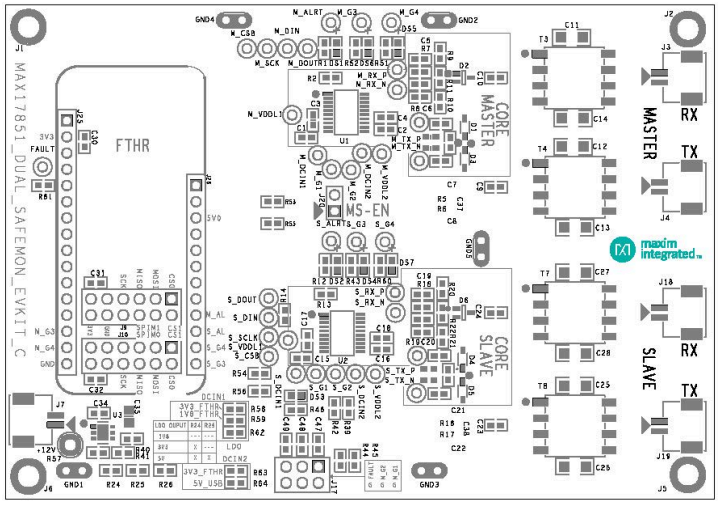


Figure 28. Layout Example: Top Layer Silkscreen

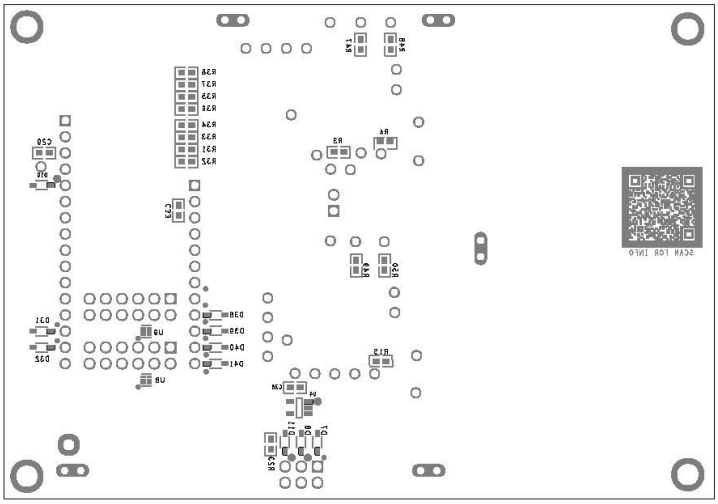


Figure 29. Layout Example: Bottom Layer Silkscreen

Layout Example (Metal)

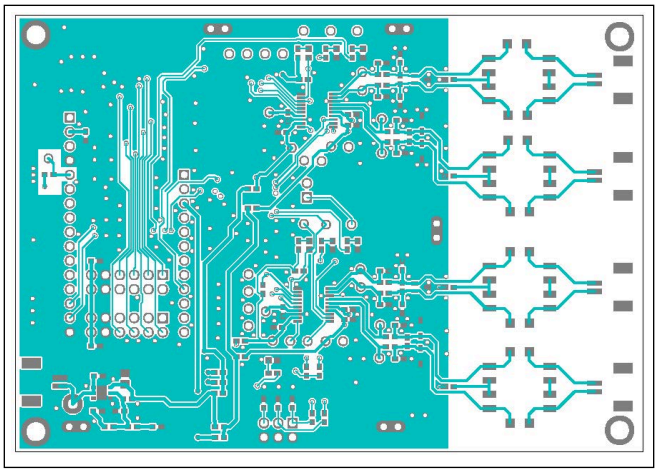


Figure 30. Layout Example: Top Metal

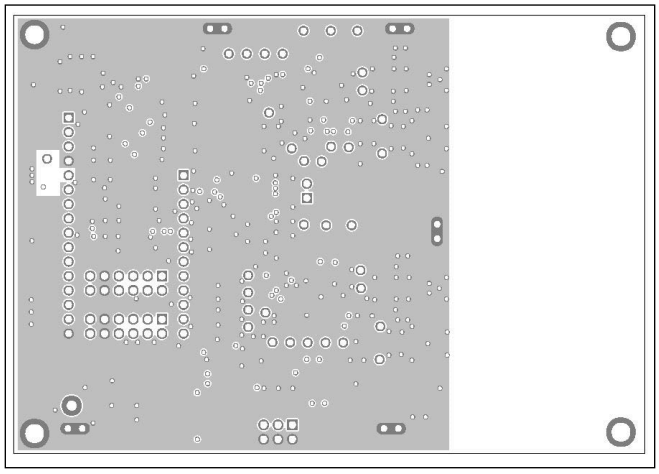


Figure 31. Layout Example: Layer Two Metal

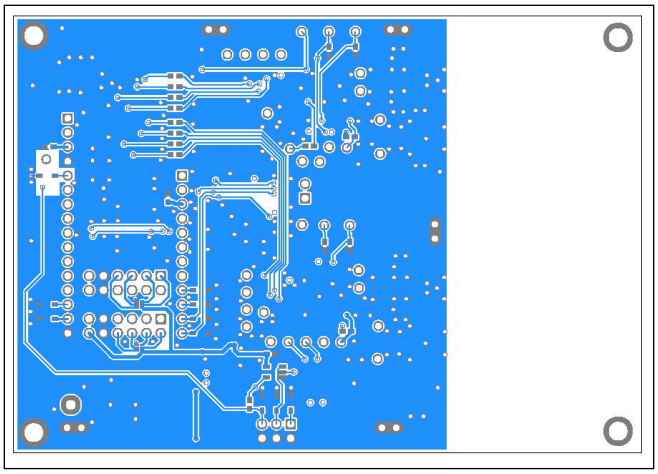


Figure 32. Layout Example: Bottom Metal

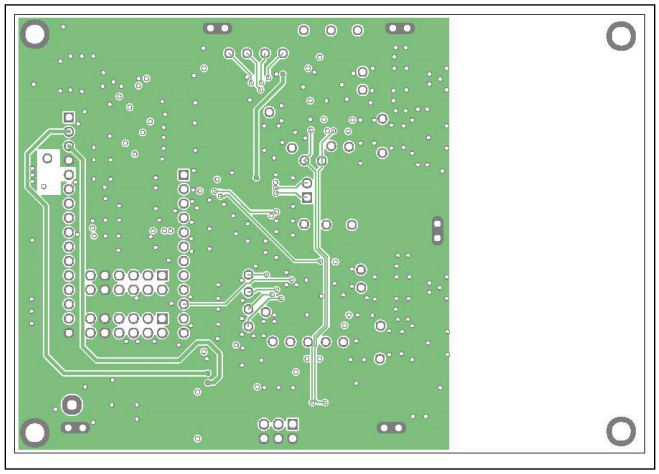
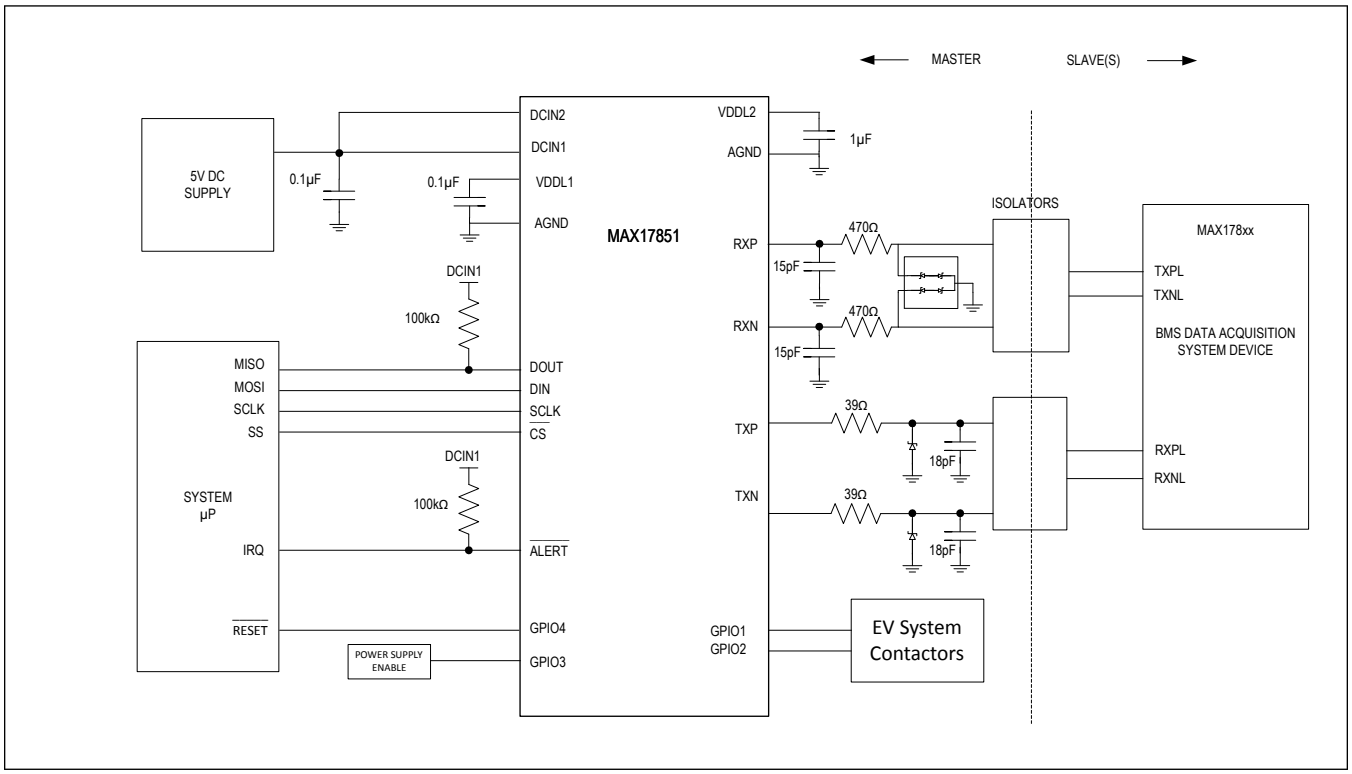


Figure 33. Layout Example: Layer Three Metal

Typical Application Circuits

MAX17851 Application Circuit



Ordering Information

PART NUMBER	TEMP RANGE	PIN-PACKAGE
MAX17851AUP/V+	-40°C to +125°C	TSSOP U20+7C
MAX17851AUP/V+T	-40°C to +125°C	TSSOP U20+7C

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

T Denotes tape-and-reel.

Revision History

REVISION NUMBER	REVISION DATE	DESCRIPTION	PAGES CHANGED
0	08/21	Initial release.	—

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Maxim Integrated:

MAX17851AUP/V+