Sensium™ High Level API Library

This document explains the library of functions to configure the modules/peripherals of the Sensium™ chip. The Sensium™ API Library files consists of functions to configure the UART, SPI, Watchdog Timer and the three Timers of the Sensium™ to perform various functions. Thus, it is easier for the developer to configure these peripherals without the need of understanding the low level design.

Contents

1 Including Sensium™ API Libraries in a Project

Users need a minimum of two files to make use of any Library in the Sensium™ API Library package. These are:

a. One of the Sensium™ API Libraries (SensiumUartApiLibrary.Lib / SensiumSpiLibrary.Lib / SensiumTimerApiLibrary.Lib / SensiumMiscellaneousApiLibrary.Lib)
b. SensiumLibraryDefines.H (header file containing all the necessary definitions required by the all the four library files)

Step 1 – Once an IAR project is created, the user needs to add SensiumXXXXApiLibrary.Lib file to the project.

Step 2 – SensiumLibraryDefines.h needs to be included in the source files where the library functions are being used using the #include<path of the headerfile> directive. E.g. #include "SensiumLibraryDefines.h"

Once these two steps are performed, then the project is ready to use the Library functions to configure the modules. The functions and their syntax for each module are discussed later in this document.

2 Sensium™ UART Library

The SensiumUartApiLibrary.Lib file provides high level functions to configure the hardware, and transmit and receive bytes through the UART.

2.1 Configure UART

Function Name	:	configureUart
Number of Function Parameters	:	5
Syntax	:	void configureUart(parameter1, parameter2, parameter3, parameter4, parameter5);
Parameter 1	:	Used to enable/disable UART Receiver Mode.
Parameter 2	:	Used to enable/disable UART Loop-back Mode.
Parameter 3	:	Used to set the transmission Parity Mode.
Parameter 4	:	Used to set the UART Baud rate.
Parameter 5	:	Used to enable/disable UART interrupt. If interrupt disabled, the driver will operate in the polling mode.

The following table shows the possible values for each of the UART parameters.

Parameter 1 (Receiver Mode)	**Parameter 2 (Loop-Back Mode)**	**Parameter 3 (Parity Mode)**	**Parameter 4 (Baud Rate)**	**Parameter 5**
UART_NO_RECIEVER	UART_NO_LOOPBACK	UART_NO_PARITY	UART_BAUDRATE_1843200	UART_INTERRUPT_ENABLE
UART_RECIEVER	UART_LOOPBACK	UART_EVEN_PARITY	UART_BAUDRATE_921600	UART_INTERRUPT_DISABLE
		UART_ODD_PARITY	UART_BAUDRATE_460800	
			UART_BAUDRATE_230400	
			UART_BAUDRATE_115200	
			UART_BAUDRATE_57600	
			UART_BAUDRATE_38400	
			UART_BAUDRATE_19200	
			UART_BAUDRATE_9600	
			UART_BAUDRATE_4800	
			UART_BAUDRATE_2400	

For example:

configureUart (UART_RECIEVER, UART_NO_LOOPBACK, UART_EVEN_PARITY, UART_BAUDRATE_115200, UART_INTERRUPT_DISABLE); will configure the UART to work in the polling mode, in the receiver mode, no loop back, even parity transmission, and with a transmission baud rate of 115200 Hz.

2.2 UART Transmit Byte

Function Name	:	uartTxByte
Number of Function Parameters	:	1
Syntax	:	void uartTxByte(unsigned char Parameter1);
Parameter 1	:	Byte to be transmitted through the UART.

For example:

uartTxByte(0x48) ;

This function transmits Hex Value of 48 through the UART.

2.3 UART Receive Byte

Function Name	:	uartRxByte
Number of Function Parameters	:	0
Syntax	:	Unsigned char receiveByte = uartRxByte(void).
Return Code	:	This function returns a byte received through the UART.

For example:

ReceiveByte = uartRxByte() ;

This function receives a byte from the UART, and stores it in the ReveiveByte Variable.

2.4 UART Library Version Number

Function Name	:	getUartApiVersionNumber
Number of Function Parameters	:	None
Syntax	:	unsigned char versionNumber = getUartApiVersionNumber();

This function gives the version number of the UART API library.

For example:

versionNumber = getUartApiVersionNumber();

3 Sensium™ SPI Library

The SensiumSpiApiLibrary.Lib file has functions to configure the hardware, and transmit and receive bytes through the SPI port. It also includes functions to perform read and write operations on the EEPROM (25AA256).

3.1 SPI Configure

Function Name	:	configureSpi
Number of Function Parameters	:	8
Syntax	:	void configureSpi(parameter1, parameter2, parameter3, parameter4, parameter5, parameter6, parameter7, parameter8);
Parameter 1	:	Used to set SPI Master Slave Mode.
Parameter 2	:	Used to set Transmission Byte Length.
Parameter 3	:	Used to set the Loop Back Mode.
Parameter 4	:	Used to set the SPI Baud rate.
Parameter 5	:	Used to enable/disable the Receive Error interrupt.
Parameter 6	:	Used to enable/disable the Transmit Error interrupt.
Parameter 7	:	Used to enable/disable the Receive interrupt.
Parameter 8	:	Used to enable/disable the Transmit interrupt.

The following tables show the possible values for each of the above parameters.

Parameter 1 (Master/Slave Mode)	**Parameter 2 (Transmission Byte Length)**	**Parameter 3 (Loop Back Mode)**	**Parameter 4 (Baud Rate)**
1. SPI_MASTER	1. SPI_LENGTH_8	1. SPI_LOOPBACK	1. SPI_BAUDRATE_8000000
2. SPI_SLAVE	2. SPI_LENGTH_7	2. SPI_NO_LOOPBAC	2. SPI_BAUDRATE_4000000
	3. SPI_LENGTH_6		3. SPI_BAUDRATE_2000000
	4. SPI_LENGTH_5		4. SPI_BAUDRATE_1000000
	5. SPI_LENGTH_4		5. SPI_BAUDRATE_500000
	6. SPI_LENGTH_3		6. SPI_BAUDRATE_250000
	7. SPI_LENGTH_2		7. SPI_BAUDRATE_125000
			8. SPI_BAUDRATE_62500
			9. SPI_BAUDRATE_31250
			10. SPI_BAUDRATE_15625
			11. SPI_BAUDRATE_7812
			12. SPI_BAUDRATE_3906
			13. SPI_BAUDRATE_1953
			14. SPI_BAUDRATE_976
			15. SPI_BAUDRATE_488
			16. SPI_BAUDRATE_244

Parameter 5 (Rx Error Interrupt)	**Parameter 6 (Tx Error Interrupt)**	**Parameter 7 (Rx Interrupt)**	**Parameter 8 (Tx Interrupt)**
1. SPI_RX_ERROR_ENABLE	1. SPI_TX_ERROR_ENABLE	1. SPI_RX_INTERRUPT_ENABLE	1. SPI_TX_INTERRUPT_ENABLE
2. SPI_RX_ERROR_DISABLE	2. SPI_TX_ERROR_DISABLE	2. SPI_RX_INTERRUPT_DISABLE	2. SPI_TX_INTERRUPT_DISABLE

NOTE:

SPI Error Detection Mechanism – In addition to the SPI Receive and the Transmit Interrupts, the SPI module is able to detect two different error conditions – Receive Error and Transmit Error. SPI_RX_ERROR_ENABLE and SPI_TX_ERROR_ENABLE parameters in the configureSPI function above will enable the SPI module to generate an SPI interrupt if there is an error in communication.

A transmit error occurs only when Sensium™ is in the SLAVE mode. It occurs if the MASTER initiates a data transfer even before the SLAVE's (Sensium™) SPI transmit buffer is updated with a new value.

A receive error occurs when Sensium™ is in both MASTER and SLAVE modes. It occurs when a new data byte is received even before the old data in the receive buffer is completely read.

In either case, regardless of their enable bits, a flag is set for each of the error conditions in the SSPCON register. If the error interrupts are enabled, the application might need to check the flags to determine which error caused the interrupt.

SPI Slave Mode – If the SPI is configured in SLAVE mode, then the fourth parameter, SPI_BAUDRATE_XXXX, in the configureSPI function is a don't care parameter.

For Example:

configureSPI (SPI_MASTER, SPI_LENGTH_8, SPI_NO_LOOPBACK, SPI_BAUDRATE_1000000,SPI_RX_ERROR_DISABLE, SPI_TX_ERROR_DISABLE, SPI_TX_INTERRUPT_DISABLE, SPI_RX_INTERRUPT_DISABLE);

The function configures the SPI as a Master, to transmit 8 bits of data word, with no loopback, at a baud rate of 1000000 Hz, and disables all the interrupts associated with SPI.

3.2 SPI Write

Function Name	:	spiWriteByte
Number of Function Parameters	:	1
Syntax	:	void spiWriteByte(parameter1);
Parameter 1	:	This function writes a byte through the SPI.

For example:

spiWriteByte(0x48);

This function writes hex value of 48 through the SPI

3.3 SPI Read

Function Name	:	spiReadByte
Number of Function Parameters	:	None
Syntax	:	unsigned char receiveByte = spiReadByte();
Parameter 1	:	This function returns a byte that is read through the SPI.

For example:

receiveByte = spiReadByte();

This function reads "data_length" number of bits (as configured in configureSpi function) through the SPI, and stores it in the receiveByte variable.

3.4 EEPROM Read

Function Name	:	readEe
Number of Function Parameters	:	3
Syntax	:	void readEe(parameter1, parameter2, parameter3);
Parameter 1	:	An Unsigned short data type – specifies the start address of the block to be read.
Parameter 2	:	An Unsigned short data type – specifies the size of the block (number of bytes in the block).

Parameter 3	:	A pointer to a variable where the read data is stored.

For example:

readEe(0x5555, 1, &ee_read_data);

This function reads 1 byte from the start of 0x5555 in the EEPROM and writes the values into the variable ee_read_data.

3.5 EEPROM Write

Function Name	:	writeEe
Number of Function Parameters	:	3
Syntax	:	void writeEe(parameter1, parameter2, parameter3);
Parameter 1	:	A short data type – specifies the start address of the block where the data is to be written.
Parameter 2	:	A short data type – specifies the size of the block (number of bytes in the block).
Parameter 3	:	A pointer to a variable where the data to be written is stored.

For example:

unsigned char ee_write_data[10];

readEe(0x5555, 10, &ee_write_data);

This function writes 10 bytes from the ee_write_data array to the EEPROM starting at 0x5555.

3.6 EEPROM Read Status Register

Function Name	:	readEeStatusRegister
Number of Function Parameters	:	None
Syntax	:	unsigned char registerValue = readEeStatusRegister();

For example:

ee_status_register = readEeStatusRegister();

This function reads the status register of the EEPROM into the ee_status_register variable.

3.7 EEPROM Write Status Register

Function Name	:	writeEeStatusRegister
Number of Function Parameters	:	1
Syntax	:	void writeEeStatusRegister(parameter1);
Parameter 1	:	This holds the value to be written into the EEPROM status register.

For example:

writeEeStatusRegister(0x02);

This function writes 0x02 into the status register of the EEPROM.

Note: It is important to check the status register of the EEPROM before any write operation for any possible write protection.

3.8 SPI Library Version Number

Function Name	:	getSpiApiVersionNumber
Number of Function Parameters	:	None
Syntax	:	unsigned char versionNumber = getSpiApiVersionNumber();

This function gives the version number of the SPI API library.

For example:

versionNumber = getSpiApiVersionNumber();

4 Sensium™ Timer Library

The SensiumTimerApiLibrary.Lib has functions to configure the watchdog timer and the three timers of the Sensium™ chip.

4.1 Watchdog Timer

Function Name	:	setWatchDog
Number of Function Parameters	:	1
Syntax	:	void setWatchDog(parameter1);
Parameter 1	:	Used to set the time with in which the watchdog must be reset, or else the software will automatically reset.

Note: The same function is used to reset the watchdog

Parameter (Watchdog reset time)
1. WDT_320_MS
2. WDT_160_MS
3 WDT_1_255_MS

For Example:

configureWatchDog(WDT_320_MS);

This function configures the watchdog to generate software reset after 320 milliseconds if the watchdog is not reset. The same function can be called to reset the watchdog.

4.2 Disable Watchdog Timer

Function Name	:	disableWatchDog
Number of Function Parameters	:	1
Syntax	:	void disableWatchDog ();

4.3 Enable Timer

Any of the three timers can be configured using this timer module in the API Library.

Function Name	:	enableTimer
Number of Function Parameters	:	3
Syntax	:	void enableTimer (parameter1, parameter2, parameter3, parameter4);
Parameter 1	:	Used to select Timer A, Timer B or Timer C.
Parameter 2	:	Used to enable/disable Auto-Reload mode
Parameter 3	:	Used to set the time in milliseconds

The following table shows the possible values for each of the Enable Timer parameters.

Parameter 1 (Timer Selection)	**Parameter 2 (Reload)**	**Parameter 3 (Time)**
1. TIMER_A	1. AUDO_RELOAD	Any integer value which is a multiple of 100
2.TIMER_B	2. NO_AUTO_RELOAD	
3.TIMER_C		

For Example:

enableTimer (TIMER_C, AUTO_RELOAD, 5000);

This function configures Timer C in access mode 1 with auto reload that generates an interrupt every 5000 milliseconds.

4.4 Disable Tmer

Any of the three timers can be configured using this timer module in the API Library.

Function Name	:	disableTimer
Number of Function Parameters	:	1
Syntax	:	void disableTimer (parameter1);
Parameter 1		Used to select Timer A, Timer B or Timer C

For Example:

disableTimer (TIMER_C);

This function will disable Timer C.

4.5 Timer Library Version Number

Function Name	:	getTimerApiVersionNumber
Number of Function Parameters	:	None
Syntax	:	unsigned char versionNumber = getTimerApiVersionNumber();

This function gives the version number of the Timer API library.

For example:

versionNumber = getTimerApiVersionNumber();

5 Sensium™ Miscellaneous Library

The SensiumMiscellaneousApiLibrary.Lib has functions to reset the software

5.1 Software Reset

Function Name	:	softwareReset
Number of Function Parameters	:	None
Syntax	:	void softwareReset();

This function performs a software reset.

5.2 Miscellaneous Library Version Number

Function Name	:	getMiscellaneousApiVersionNumber
Number of Function Parameters	:	None
Syntax	:	unsigned char versionNumber = getMiscellaneousApiVersionNumber();

This function gives the version number of the Miscellaneous API library.

For example:

versionNumber = getMiscellaneousApiVersionNumber();

EVALUATION BOARD/KIT IMPORTANT NOTICE

Texas Instruments (TI) provides the enclosed product(s) under the following conditions:

This evaluation board/kit is intended for use for **ENGINEERING DEVELOPMENT, DEMONSTRATION, OR EVALUATION PURPOSES ONLY** and is not considered by TI to be a finished end-product fit for general consumer use. Persons handling the product(s) must have electronics training and observe good engineering practice standards. As such, the goods being provided are not intended to be complete in terms of required design-, marketing-, and/or manufacturing-related protective considerations, including product safety and environmental measures typically found in end products that incorporate such semiconductor components or circuit boards. This evaluation board/kit does not fall within the scope of the European Union directives regarding electromagnetic compatibility, restricted substances (RoHS), recycling (WEEE), FCC, CE or UL, and therefore may not meet the technical requirements of these directives or other related directives.

Should this evaluation board/kit not meet the specifications indicated in the User's Guide, the board/kit may be returned within 30 days from the date of delivery for a full refund. THE FOREGOING WARRANTY IS THE EXCLUSIVE WARRANTY MADE BY SELLER TO BUYER AND IS IN LIEU OF ALL OTHER WARRANTIES, EXPRESSED, IMPLIED, OR STATUTORY, INCLUDING ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE.

The user assumes all responsibility and liability for proper and safe handling of the goods. Further, the user indemnifies TI from all claims arising from the handling or use of the goods. Due to the open construction of the product, it is the user's responsibility to take any and all appropriate precautions with regard to electrostatic discharge.

EXCEPT TO THE EXTENT OF THE INDEMNITY SET FORTH ABOVE, NEITHER PARTY SHALL BE LIABLE TO THE OTHER FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES.

TI currently deals with a variety of customers for products, and therefore our arrangement with the user **is not exclusive.**

TI assumes **no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein.**

Please read the User's Guide and, specifically, the Warnings and Restrictions notice in the User's Guide prior to handling the product. This notice contains important safety information about temperatures and voltages. For additional information on TI's environmental and/or safety programs, please contact the TI application engineer or visit www.ti.com/esh.

No license is granted under any patent right or other intellectual property right of TI covering or relating to any machine, process, or combination in which such TI products or services might be or are used.

FCC Warning

This evaluation board/kit is intended for use for **ENGINEERING DEVELOPMENT, DEMONSTRATION, OR EVALUATION PURPOSES ONLY** and is not considered by TI to be a finished end-product fit for general consumer use. It generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of computing devices pursuant to part 15 of FCC rules, which are designed to provide reasonable protection against radio frequency interference. Operation of this equipment in other environments may cause interference with radio communications, in which case the user at his own expense will be required to take whatever measures may be required to correct this interference.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DLP® Products	www.dlp.com	Broadband	www.ti.com/broadband
DSP	dsp.ti.com	Digital Control	www.ti.com/digitalcontrol
Clocks and Timers	www.ti.com/clocks	Medical	www.ti.com/medical
Interface	interface.ti.com	Military	www.ti.com/military
Logic	logic.ti.com	Optical Networking	www.ti.com/opticalnetwork
Power Mgmt	power.ti.com	Security	www.ti.com/security
Microcontrollers	microcontroller.ti.com	Telephony	www.ti.com/telephony
RFID	www.ti-rfid.com	Video & Imaging	www.ti.com/video
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265