SP300 Series Pressure Sensors with Embedded Micro Controller

Fully Compensated, Multipurpose Pressure Sensors with Embedded Microcontroller

The SP300 series represents Sensonor Technnolgies' latest generation multipurpose, absolute pressure sensors. Based on already known technology, SP300 adds a level of integration by including a micro controller (µC), 2 LF-input stages, 5 available general purposer I/O pins, and many other sofisticated features to meet market demands for flexible, customer specific behaviour and solutions, also resulting in over-all system cost reductions.

This user manual supports programmers in creating application programs for the sensor, and therefore most of the descriptions herein are presented as seen from the micro controller.

SP300 sensors are delivered with pressure sensors (ranges from 50 to 1600 kPa), temperature and supply voltage sensors, acceleration sensor (range from –12 to 115 g) (option), and tube connection (option).

See product datasheets for performance of the specific SP300 sensors.

SP300-0.5(T)
SP300-1(T)
SP300-2(T)
SP300-7(T)
SP300-7A(T)
SP300-16A

SP300 Series Pressure Sensors with Embedded Micro Controller

Edition 2010-03-12
Published bySensonor Technologies AS
P.O. Box 196
N-3192 Horten
Norway

Information furnished by Sensonor Technologies is believed to be accurate and reliable. However, no responsibility is assumed by Sensonor Technologies for its use, nor for any infringements of patents or other rights of third parties that may result from its use. Sensonor Technologies reserves the right to make changes without further notice to any products herein. Sensonor Technologies makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Sensonor Technologies assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. No license is granted by implication or otherwise under any patent or patent rights of Sensonor Technologies. Trademarks and registered trademarks are the property of their respective owners. Sensonor Technologies products are not intended for any application in which the failure of the Sensonor Technologies product could create a situation where personal injury or death may occur. Should Buyer purchase or use Sensonor Technologies products for any such unintended or unauthorized application, Buyer shall indemnify and hold Sensonor Technologies and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable legal fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Sensonor Technologies was negligent regarding the design or manufacture of the part.

SP300 Series Pressure Sensors with Embedded Micro Controller

TABLE OF CONTENTS

TABLE OF FIGURES

1 REFERENCES, ABBREVIATIONS AND DEFINITIONS

1.1 REFERENCES

Ref.	Document title	Doc no
[1]	RISC Architecture and Instruction Set SP300	TN10030201
[2]	Monitor and Download Interface SP300 EROM	TN03103003

1.2 ABBREVIATIONS AND DEFINITIONS

The following abbreviations and definitions are used throughout the document:

AGC	Automatic Gain Control
ALU	Arithmetic Logic Unit
ASIC	Application Specific Integrated Circuit
ASK	Amplitude Shift Key
BCD	Binary-Coded Decimal
EEPROM	Electrically Erasable Programmable Read-Only Memory
EROM	Physically the same as EEPROM. “EROM” is used for Program Memory, and “EEPROM” is used for device specific memory (which holds calibration coefficients) in SP300
LF	Low Frequency (for wireless communication with SP300)
LNA	Low Noise Amplifier
LSB	Least Significant Bit
MSB	Most Significant Bit
MUX	Multiplexer
N.A.	Not Applicable
PCB	Printed-Circuit Board
PLL	Phase-Locked Loop
PTAT	Proportional To Absolute Temperature
RISC	Reduced Instruction Set Controller

2 PRODUCT INFORMATION

2.1 SYSTEM DESCRIPTION

The SP300 sensors are micro systems consisting of two dies packaged in a low cost, plastic moulded 14 pin SOIC package as shown in Figure 2-1. The chip shown to the left is an example of a micro machined SP300 pressure and acceleration sensor die. The die to the right is the ASIC containing the remaining part of the micro system. Wire bonds connect the dies internally and forward the ASIC signals to the more accessible package pins. A block diagram of the sensors is shown in Figure 2-3.

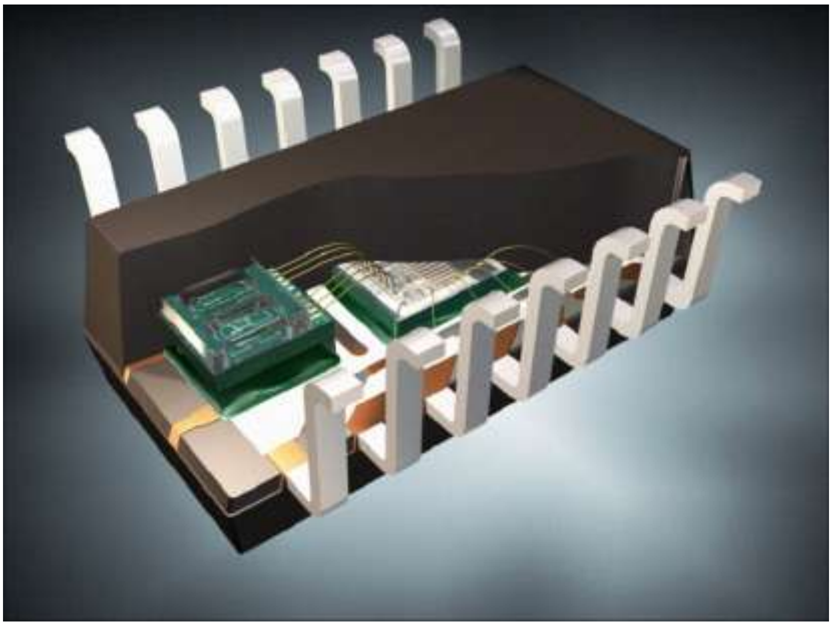

Figure 2-1 View of SP300 sensor internals

2.2 PIN OUT AND LASER MARKING

Figure 2-2 shows SP300 pin out and laser marking.

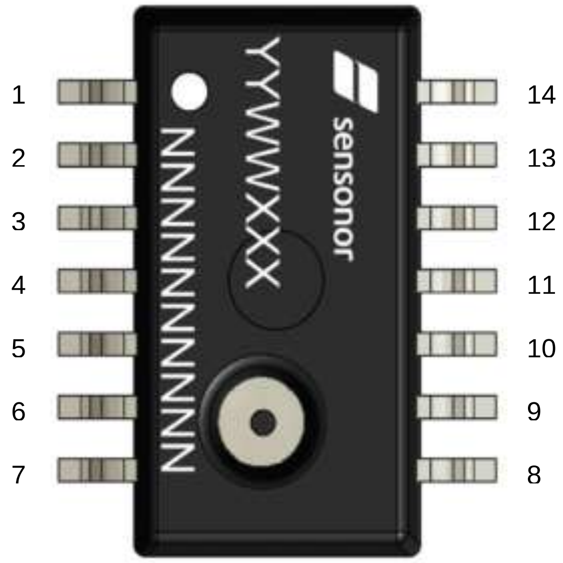

Figure 2-2 SP300 package (top view)

Reading the laser marking:

YYWWXXX :	Lot number
NNNNNNNNNN:	Product name
O:	Pin 1 marking

Table 2-1 shows the name and function of each pin. Pin connection for a typical application is shown in 9.1.

Table 2-1 SP300 pin names and function

PIN NO, TYPE 6	NAME	FUNCTION	NOTE
1	IN4	LF receiver channel 2, negative input	
2	P10	General purpose I/O with external wakeup, internal pull-up/pull-down	
3	P11	General purpose I/O with external wakeup, internal pull-up/pull-down	
4	MSDA	Monitor Serial Data I/O, internal pull-up	
5	MSCL	Monitor Serial Clock input	
6	VDD	Supply pad VDD (battery/ supply voltage, positive terminal)	
7	VSS	Common ground (battery/ supply voltage, negative terminal)	
8	VSS	Common ground (battery/ supply voltage, negative terminal)	
9	P17	General purpose I/O (or digital modulator output)	
10	P15	General purpose I/O or external clock	
11	P14	General purpose I/O (or digital modulator output)	
12	IN1	LF receiver channel 1, positive input	
13	IN2	LF receiver channel 1, negative input	
14	IN3	LF receiver channel 2, positive input	

Figure 2-3 shows SP300 block diagram. The pressure and acceleration (optinonal) sensors are indicated to be on a separate sensor die by the dashed lines. The five general purpose I/O's (P10, P11, P14, P15, and P17) are indicated to the right. MSDA and MSCL pins for the monitor interface is indicated on top.

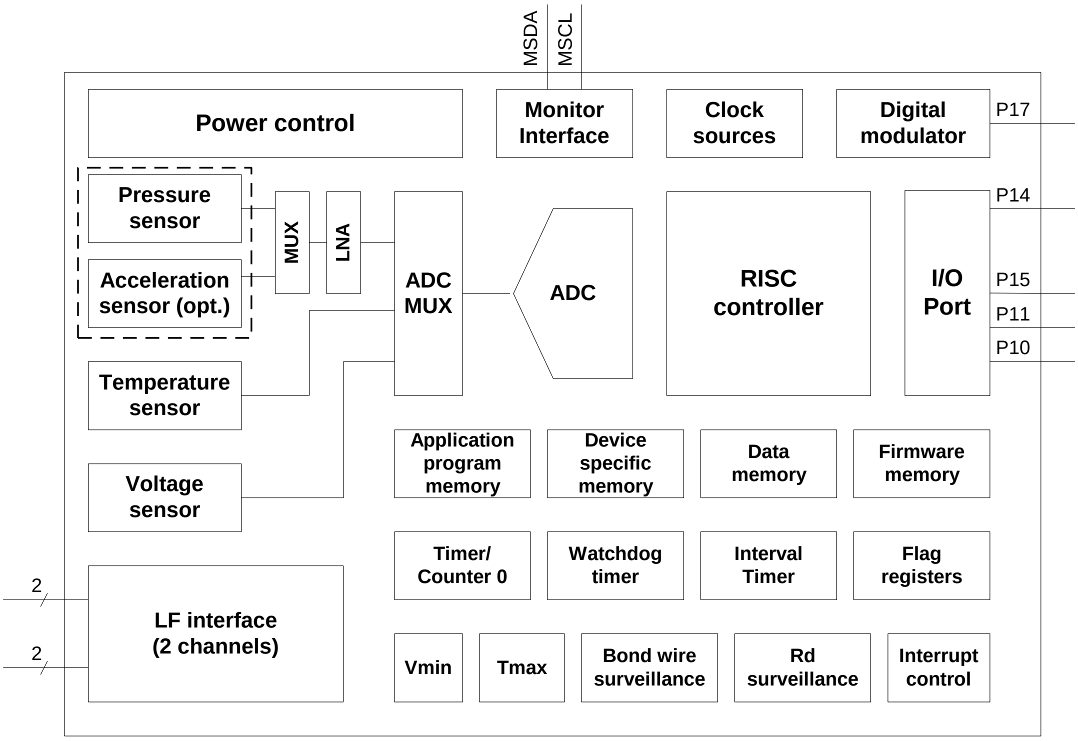

Figure 2-3 SP300 block diagram

2.3 RISC CONTROLLER

The SP300 RISC controller employs a low-power RISC to control device operation according to the application program. The instruction set for the controller is found in [1].

The RISC is clocked from an on-chip RC oscillator, while the peripheral unit Timer/Counter 0 may be clocked from an external source, for example the reference clock of a PLL chip in case a higher precision or synchronous timing is desired.

The RISC features an 8 bit Harvard architecture with 16 bit instructions. Due to a two-stage pipeline concept the instructions execute in a single instruction cycle, featuring fast execution time and low-power consumption. The RISC block diagram is shown in Figure 2-4.

Byte-wise read access for the Application Program Memory is provided, whereas the firmware is not visible to the application program due to the implemented shadow mechanism. A software interrupt (SYS) allows the execution of library functions in ROM, see chapter 6.2. The SYS mechanism handles the transition to the "shadowed" ROM.

The RISC features 128 byte of internal Data Memory. The Data Memory is also used as 16-bit program stack during subroutine calls and interrupts. Although no specific instructions are provided to manipulate this stack (e.g. Push/Pop), the RISC features auto-increment and auto-decrement addressing modes to enable data stack handling by the application program.

The peripherals of the RISC (timer, I/O, etc) are accessible via a Special Function Register File (SFR) that is mapped into the Data Address Space. The RISC allows byte-oriented as well as bit wise access to both Data Memory and SFR's.

Eight general purpose registers (8 bit) are provided. Four of them may be used in the context of indirect addressing. Two of these registers provide additional post-increment and pre-decrement addressing modes in order to support e.g. a software data stack.

The ALU supports instructions for arithmetic, logical and Boolean data manipulation.

The single priority level Interrupt Control system features an additional wakeup function from IDLE mode, which provides the application program with a convenient means to synchronize with the peripherals.

A number of special loop control and bit shift instructions are available to optimize code speed and size, ref. [1].

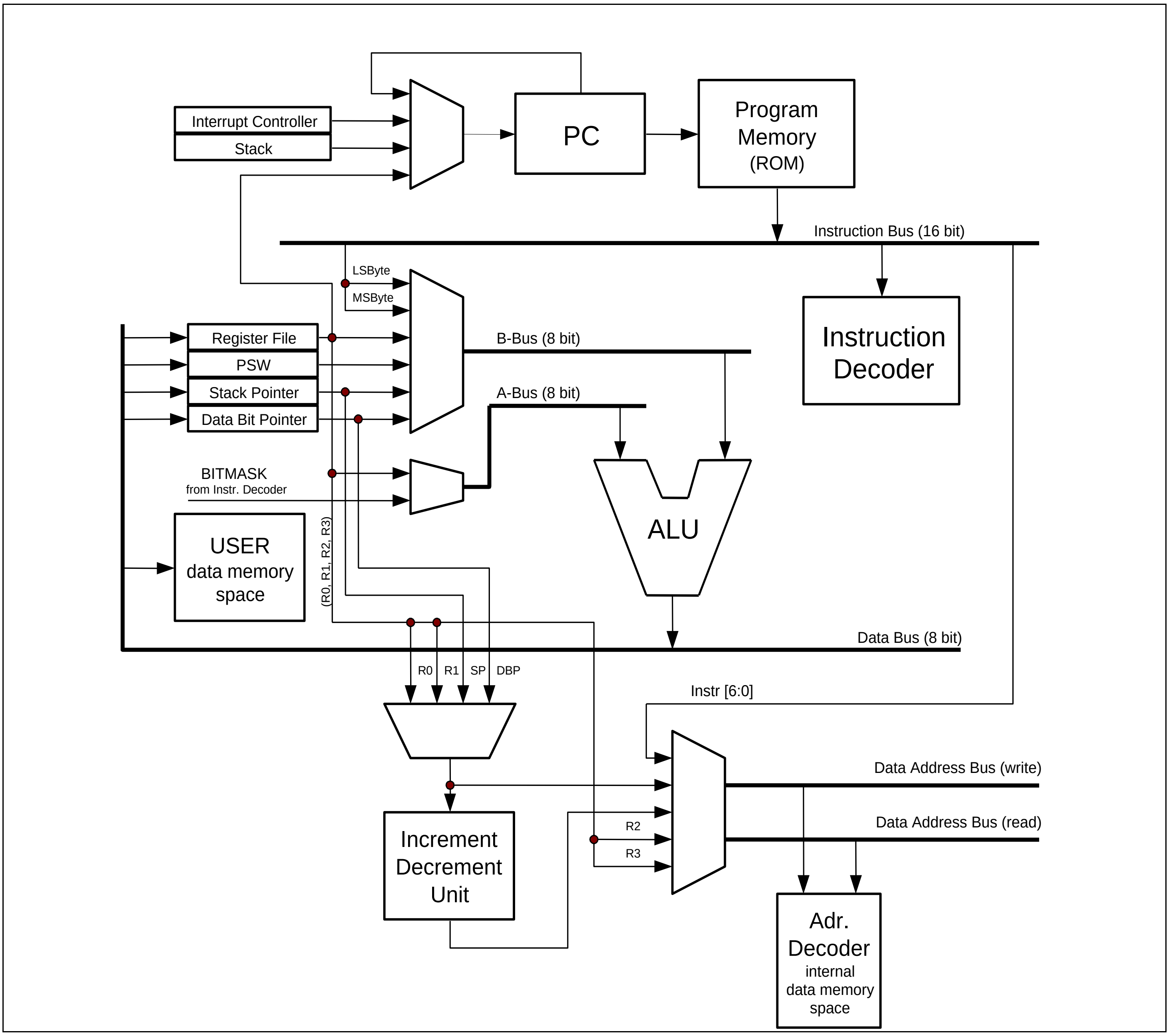

Figure 2-4 SP300 RISC Controller block diagram

2.4 OVERVIEW OF SPECIAL FUNCTION REGISTERS

The Special Function Registers (SFR) is a key element for controlling the device and hence it is important to know and understand its contents. Table 2-2 gives an overview of the SFR, while description of each individual bit is explained on the referred pages (follw the links directly for more details).

Table 2-2 Special Function Register Summary

NAME	REF. PAGE	DESCRIPTION	ADDR	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	RESET VALUE
PSW	40	Program Status Word	12h	C	H	OV	X	C‘	H‘	OV‘	X	
SP	41	Stack Pointer	13h	SP7	SP6	SP5	SP4	SP3	SP2	SP1	X	1100000XB
SBIT	Ref. [1]	Indirect Bit Address	14h	MAP	X	X	X	X	SBIT2	SBIT1	SBIT0	
SPTR	Ref. [1]	Indirect Byte Address	15h	SPTR7	SPTR6	SPTR5	SPTR4	SPTR3	SPTR2	SPTR1	SPTR0	
IE	27	Interrupt Enable	16h	EA	X	EE	EI4	EADC	ET0	EP	X	00000000B
IFF	27	Interrupt Flag	17h	X	X	FE	FI4	FADC	FT0	FP	X	X0000000B
T0	22	Timer/Counter 0	18h									
TR0	22	Timer/Counter 0 Reload	19h									
TCON	22	Timer/Counter 0 Control	1Ah	X	TPS2	TPS1	TPS0	TCS1	TCS0	X	TRS0	X00000X0B
WTCON	25	Watchdog Timer Control	1Bh	X	WPS2	WPS1	WPS0	X	X	X	WCLR	X000XXXXB
TVCON	56	Thermal Shutdown Flags,	1Ch	TSTHI	X	X	X	X	X	X	X	XXX0XX0XB
MODCON	29	Modulator Control	1Dh	MDB	X	X	X	SCEN	EP17	EP14	X	XXXX000XB
X	N.A.	Reserved	1Eh - 1Fh	X	X	X	X	X	X	X	X	
P1OUT	31	Port 1 Output	20h	P17	X	P15	P14	X	X	P11	P10	0X00X000B
P1INS	32	Port 1 Input sense	21h	P17S	X	P15S	P14S	X	X	P11S	P10S	
P1DIR	31	Port 1 Direction	22h	IO17	X	IO15	IO14	X	X	IO11	IO10	0X00X000B
X	N.A.	Reserved	23h - 25h	X	X	X	X	X	X	X	X	
PCON	16	Power Control	26h	X	X	X	X	RST	SHUT	PDWN	IDLE	XXXX0X00B
SCSL	18	System Clock Select	27h	X	X	X	X	X	CSL2	CSL1	CSL0	XXXXX000B
X	N.A.	Reserved	28h - 33h	X	X	X	X	X	X	X	X	
RICAR	33	RISC Interface-Address	34h	X	X	X	RIA4	RIA3	RIA2	RIA1	RIA0	XXX00000B
RIDR	33	RISC Interface-Data	35h	RID7	RID6	RID5	RID4	RID3	RID2	RID1	RID0	XXXXXXXXB
	N.A.	Reserved	36h - 3Fh	Reserved								

Note

Bits marked ‘X’ are reserved for future use or device test. Any read operation yields an undefined result.

3 OPERATING MODES

The SP300 features a versatile Power Management scheme to control and minimise the system power consumption. The different device operating modes and wakeup mechanisms are controlled by the RISC circuitry and the corresponding program memory.

The device operates in one of four modes, according to Figure 3-1.

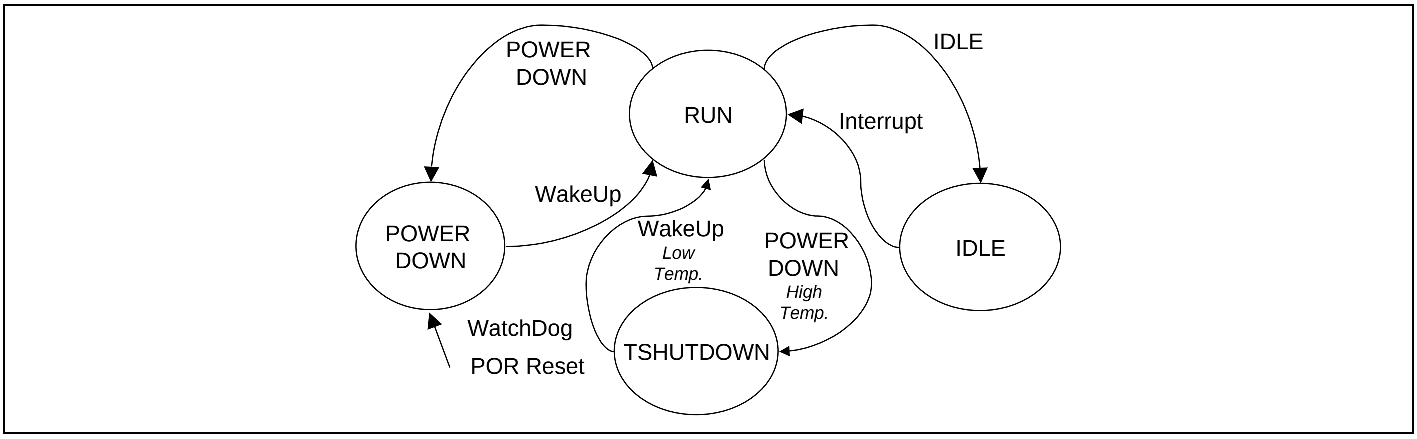

Figure 3-1 Device operating modes

3.1 POWER DOWN

The POWER DOWN mode offers a minimum of current consumption, enabling a long battery lifetime for wireless applications. Hence, SP300 should be kept in POWER DOWN mode as much as possible.

In POWER DOWN mode most of the chip circuitry is disabled. Only the Interval Timer, Flag Register, TMAX and LF Interface, if enabled, remain operational.

Upon a device wakeup condition, the POWER DOWN mode is terminated, which enables and powers up the whole chip circuitry. As soon as the internal chip supply is settled, the device enters RUN mode, causing the RISC circuitry to commence program execution. The device will execute a Boot Sequence prior to passing control to the program memory reset vector.

A device wakeup applies when the Interval Timer or LT Timer overflow, port P10 or port P11 sense a high-to-low transition, or the LF Interface detects an LF Header, provided that the LF Interface has been enabled by instruction. The actual wakeup source is signaled in the corresponding register and may be tested by the application program.

If the WatchDog counter is allowed to time-out while operating in RUN or IDLE mode, the device will cease operation and be forced into POWER DOWN mode.

In case the supply voltage drops below the Power On Reset threshold, the device is forced into reset state causing the device to cease operation.

3.2 RUN

When operating in RUN mode the RISC is operated at configured clock speed. Peripheral units may be enabled.

The device may be forced into POWER DOWN mode at anytime, when a RISC controlled device operation is not required. The POWER DOWN mode is invoked upon instruction by setting the control bit PDWN, which is part of Special Function Register PCON.

3.3 THERMAL SHUTDOWN

If the ambient temperature is above the specified threshold, the device can be forced from RUN mode into THERMAL SHUTDOWN mode in order to prevent undefined operation. As a result, all chip circuitry and port lines are disabled, including the Interval Timer. The only active circuitry is the on-chip temperature sensor TMAX (ref. 4.11) and the Flag registers.

The device will remain in THERMAL SHUTDOWN mode until the ambient temperature drops below the thermal shutdown release temperature. When the THERMAL SHUTDOWN mode is released, a wakeup is generated, and the device enters RUN mode resuming normal operation.

3.4 IDLE

Since some device circuitry once triggered operates independently from the RISC core (Timer Counter, ADC, LF Interface etc.) the RISC core may be put into IDLE mode until the corresponding circuitry complete operation. The IDLE mode is terminated and the RUN mode resumed when an Interrupt Request is detected. As a result, the program execution commences. Depending on the interrupt configuration the corresponding interrupt is serviced or not, see also section 4.14.

The IDLE mode may be invoked by the “Idle” library function (ref. 8.4.16). Another possibility is to set the control bit IDLE as part of Special Function Register PCON. This will however cause a higher current consumption, because the EROM is not turned off.

3.5 DEVICE WAKEUP

SP300 supports four mechanisms for device wakeup (WUP):

1. Periodic, by means of an on-chip Interval Timer or the LF Timer
2. By port P10 or P11, which may be connected to external circuitry, like a roll switch
3. The LF Interface when detecting an LF Header
4. Wakeup from THERMAL SHUTDOWN mode

3.5.1 INTERVAL TIMER WAKEUP

A periodic device wakeup from POWER DOWN mode is accomplished by an Interval Timer. The Interval Timer is active regardless of the mode of operation and is clocked from an independent low-power RC oscillator, ref. 4.6.2. The Interval Timer features a programmable period, controlled by the IT<1:0> in the Timer Register of the RISC Interface Register Set (ref. 5.7). In addition to the Interval Timer tick, a second tick may be generated by the LF Timer (ref. 4.7).

If operating in POWER DOWN mode, a device wakeup is generated when the Interval timer overflows. When operating in RUN or IDLE mode a corresponding interrupt (IT Overflow) will be generated. This will trigger an Interrupt request via INT4, see also section 4.14.

The Interval Timer Overflow flag (ITOV) is set when the Interval Timer overflows. The flag may be tested by the application program, in order to identify the wakeup or interrupt sources. Since the flag is not reset, it should be cleared by the application program prior to forcing the device into POWER DOWN mode in order to serve its identification function.

The ITOV flag is located in the Status Register of the RISC Interface Register Set. See section 5.2.

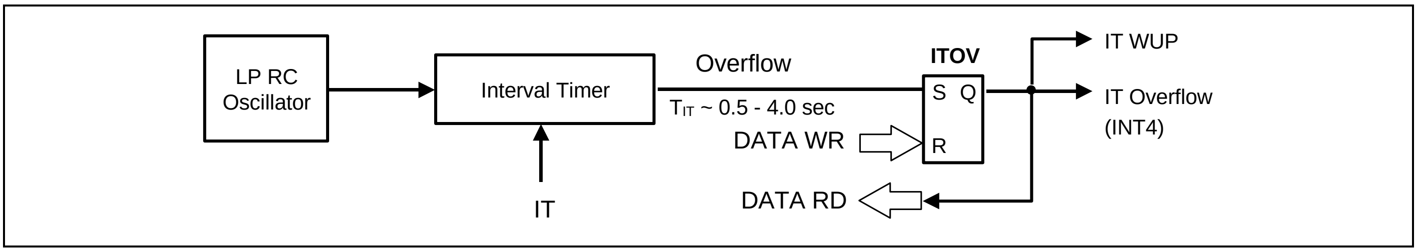

Figure 3-2 Interval Timer block diagram

3.5.2 PORT WAKEUP

Device wakeup from POWER DOWN mode may be accomplished by port P10 and P11, see Figure 3-3. A port wakeup is detected upon a high-to-low transition at port P10 or P11 and triggers a Port10/11 interrupt request. See also section 4.16.

Port P10 and P11 feature an on-chip programmable pull-up/pull-down resistor.

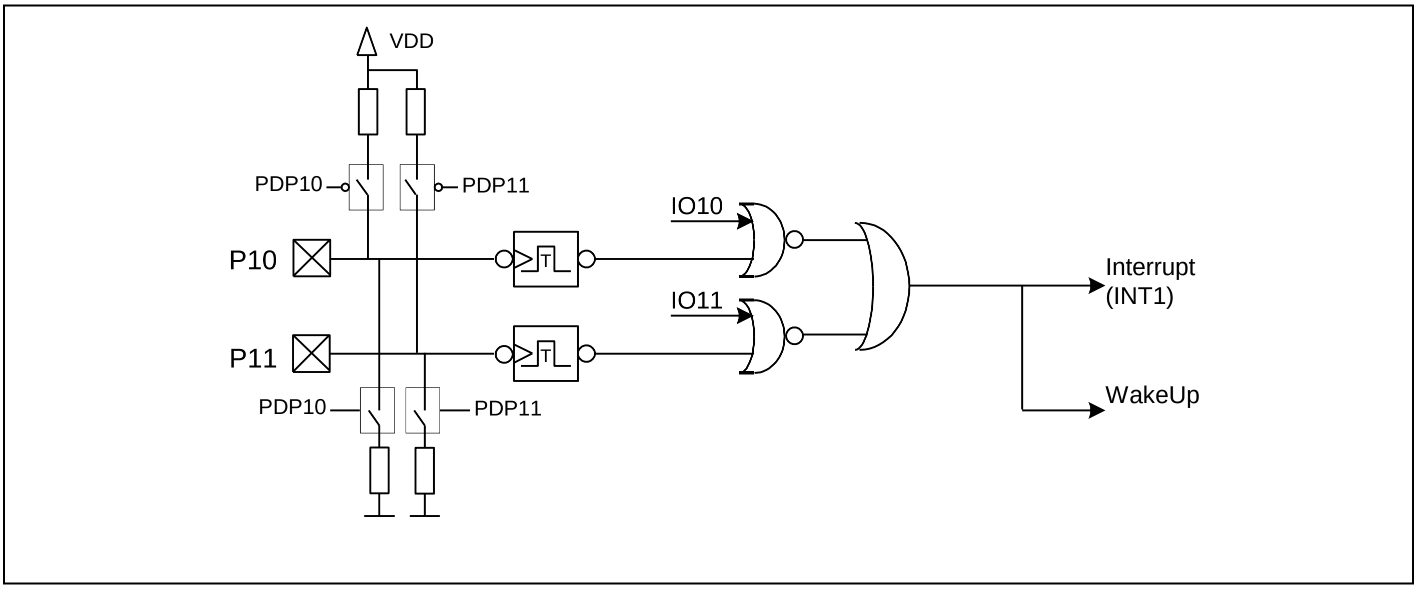

Figure 3-3 Port wakeup block diagram

3.5.3 LF WAKEUP

Device wakeup from POWER DOWN mode may be generated by means of an LF Header received and detected by the LF Interface. However, this requires that the LF Interface has been enabled by the application program prior to forcing SP300 into POWER DOWN mode.

In this case, the LF WUP flags (WUP1, WUP2) are set and may be tested by the application program as desired, in order to identify the wakeup source. Since the flags are not reset by hardware, they shall be cleared by the application program prior to forcing the device into POWER DOWN mode in order to serve their identification function.

The WUP1 and WUP2 flags are located in the RISC Interface Register Set, Status Register (ref. Table 5-5).

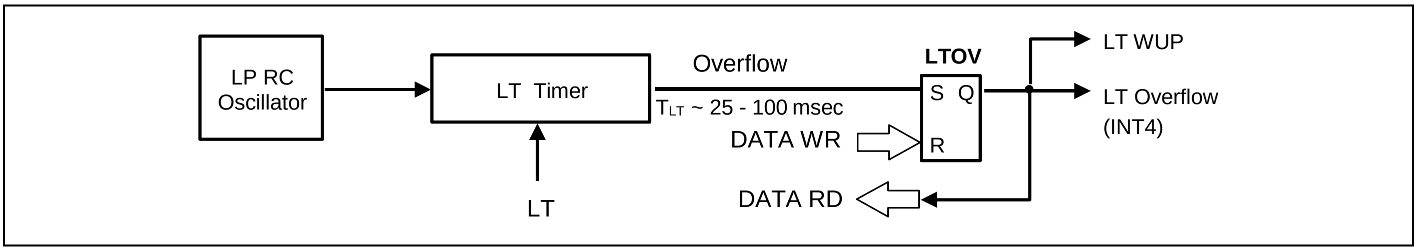

Figure 3-4 LF Timer block diagram

3.5.4 WAKEUP FROM THERMAL SHUTDOWN

Wakeup from THERMAL SHUTDOWN mode when temperature goes below thermal shutdown release temperature is described in 3.3.

3.6 RESTART

Upon instruction the application program may reset the entire device, causing the device to resume program execution at the RESET vector. A RESTART is forced by setting the control bit RST as part of Special Function Register PCON (ref. 4.5).

4 PERIPHERAL UNITS

This section describes the peripheral units seen from the SP300 micro controller core. For memory description, refer to chapter 5.

4.1 ADC

The implemented ADC is a first order Sigma-Delta ADC with differential input. The resolution is selectable between 5 and 12 bits. The ADC is used when performing measurements of pressure, acceleration, temperature, and supply voltage, and is used by the implemented library functions for the required measurements.

The ADC input is connected via the ADC MUX to the LNA output, the on-chip supply voltage sensor or the on-chip temperature sensor.

4.2 LNA AND INPUT MUX

The LNA operates in a differential fashion with programmable gain in order to cope with the output voltage range of the connected sensor bridge. The LNA features a MUX, selecting one of two independent differential sensor inputs.

4.2.1 PRESSURE SENSOR

The pressure sensor consists of a single-crystal silicon, bulk micro machined membrane with an integrated full Wheatstone piezo-resistive bridge. The piezo-resistors are placed inside a vacuum reference chamber, whilst the pressure media to be measured in the application is applied to the opposite side of the membrane. This gives good long-term properties and exceptional media compatibilty as the measurement bridge is protected from the environment. Only pure silicon and glass are exposed to the measurement media in the pressure inlet hole. Figure 4-1 shows a drawing of the combined pressure and acceleration sensor die. The sensor cell furthest away is for measuring pressure. Pressure measurement is performed by a dedicated library function, as described in Chapter 8.

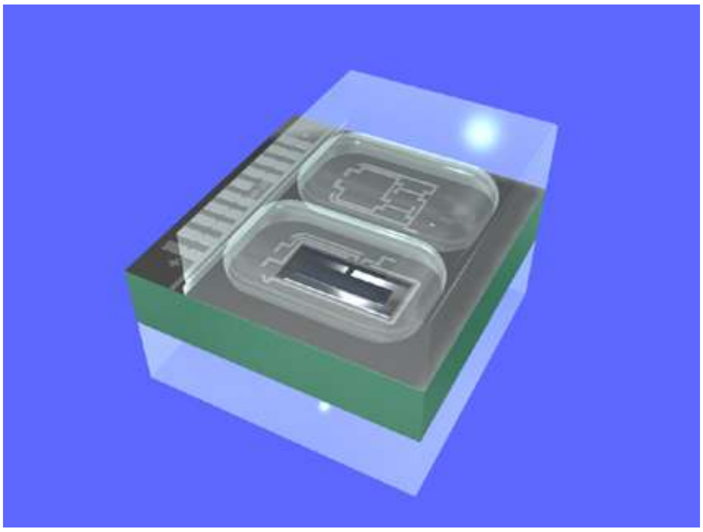

Figure 4-1 Sensor die

4.2.2 ACCELERATION SENSOR

The (optional) acceleration sensor consists of a single-crystal silicon, bulk micro machined beam with an integrated full Wheatstone piezo-resistive bridge. The whole beam is placed inside a hermetically sealed vacuum chamber and is therefore well protected from the environment. A diagnostic resistor, Rd, is integrated along the edge of the beam to be used to check the mechanical integrity of the beam. In Figure 4-1, the nearest sensor cell shows the accelerometer (beam and mass). Acceleration measurement is performed by a dedicated library function, as described in Chapter 8.

4.3 TEMPERATURE SENSOR

The temperature sensor is placed on the ASIC and is a PTAT circuit. This is read by the ADC referenced to a fixed (band gap) voltage. Temperature measurement is performed by a dedicated library function, as described in Chapter 8.

4.4 VOLTAGE SENSOR

The on-chip voltage sensor is a circuit that gives out a voltage proportional to the supply voltage. The voltage is read by the ADC referenced to a fixed (band gap) voltage. Voltage measurement is performed by a dedicated library function, as described in Chapter 8.

4.5 POWER CONTROL

The device operating mode is controlled by the bits located in the PCON register that is mapped into the SFR space of the data memory, see Table 4-1.

Table 4-1 Power Control Register, PCON

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
X	X	X	X	RST	SHUT	PDWN	IDLE
				R0/W	R/W	R0/W	R0/W

Note Address = 26h

Bits marked 'X' are not connected and reserved for future use. Any read operation yields an undefined result. For future compatibility any write operation should assign a '0'.

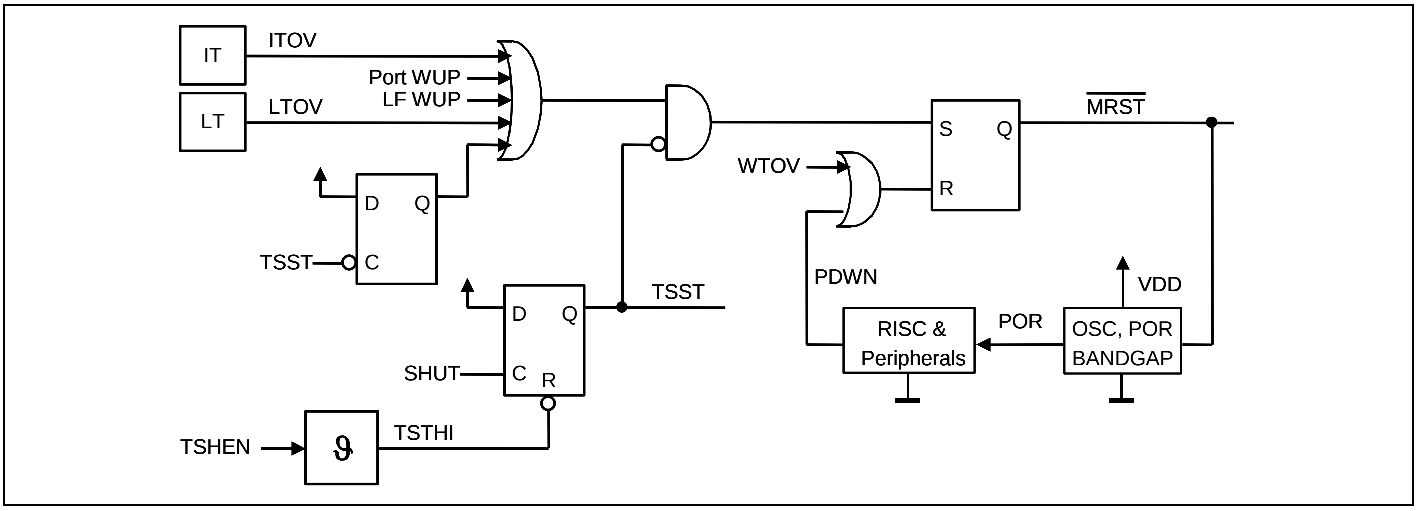

Figure 4-2 Power Control block diagram

The Power Control as shown in Figure 4-2 offers different wakeup methods such as the LF Interface, the Interval or LF Timer, or Port wakeup, as described in section 3.5.

RST, PDWN and IDLE provide a trigger signal for the corresponding circuitry. Any read operation yields zero as result.

When SHUT is set, THERMAL SHUTDOWN mode is prepared (ref. 8.4.9) and TSST is set to 1. A read instruction of SHUT will return TSST information.

4.6 CLOCK SOURCES

SP300 includes two on-chip oscillators; a high-precision oscillator and a low-power oscillator. Furthermore, an external clock source for Timer/Counter 0 may be connected to P15.

4.6.1 HIGH-PRECISION OSCILLATOR

The high-precision oscillator operates at a nominal frequency of 8 MHz. The clock generation circuitry features a programmable clock divider in order to enable a clock rate selection for the RISC according to the speed and power consumption requirements of the application program, ref. Figure 4-3.

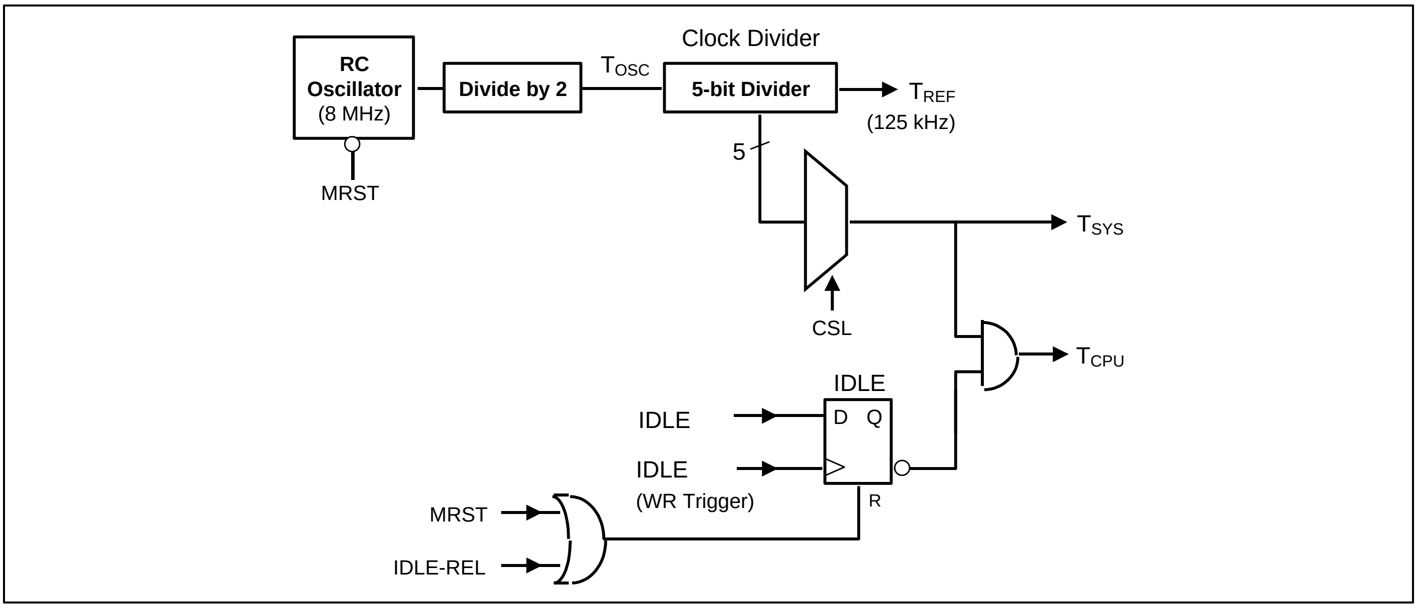

Figure 4-3 System clock generation

The application program may change the clock selection at any time “on-the-fly”. Each write-access to the SCSL register forces a new synchronization. The system clock, T_{SYS}, is determined by the Clock Select control bits, CSL, according to Table 4-2.

Table 4-2 Clock Select, CSL

CSL2	CSL1	CSL0	T_{SYS}	F_{SYS} (typ)	Note
0	0	X	T_{OSC} * 32	125 kHz	
0	1	0	T_{OSC} * 16	250 kHz	
0	1	1	T_{OSC} * 8	500 kHz	
1	0	0	T_{OSC} * 4	1 MHz	
1	0	1	T_{OSC} * 2	2 MHz	
1	1	0	Reserved	Reserved	1
1	1	1	Reserved	Reserved	1

Note

1. Reserved for device test purposes

The System Clock Select Register is located in the SFR space of the data memory. The bits are assigned as shown in Table 4-3.

Table 4-3 System Clock Select Register, SCSL

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
X	X	X	X	X	CSL2	CSL1	CSL0
W0	W0	W0	W0	W0	R/W	R/W	R/W

Note Address = 27h

Bits marked ‘X’ are not connected and reserved for future use. Any read operation yields an undefined result. For future compatibility any write operation should assign a ‘0’.

A 125 kHz clock (T_{REF}) is derived from the high-precision oscillator, which serves the function of an internal reference clock for timing purposes. The reference clock is also available for the application program as clock input for the on-chip timers, see sections 4.8 and 4.9.

Initially after reset, the SCSL register is cleared and the lowest speed for the RISC is selected.

4.6.2 LOW-POWER OSCILLATOR

Due to the extreme low-power requirements, the low-power oscillator is implemented to serve as a periodic wakeup source, ref 3.5.1. In addition, the low-power oscillator is used internally by the LF interface.

4.6.3 EXTERNAL CLOCK

The external clock on P15 can be used as an input to Timer/Counter 0 (see section 4.8), e.g. to assure a high accuracy baud rate of the signal train generated in the Digital Modulator circuitry (see section 4.15) or to synchronize with an external RF transmitter. The baud rate is dictated by the frequency of the external clock input combined with the value in the prescaler register in Timer/Counter 0.

4.7 LF INTERFACE

A block diagram of the LF interface is shown in Figure 4-4.

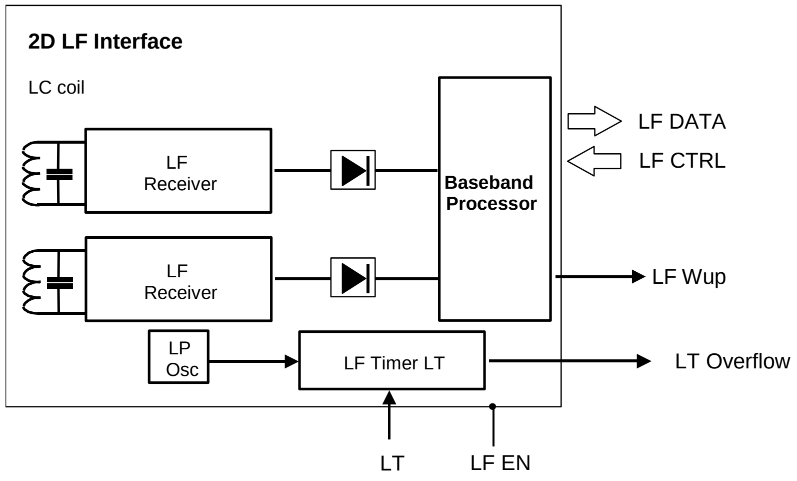

Figure 4-4 LF Interface block diagram

The LF interface may be enabled by the “LF_Interface_Control” library function (ref. 8.4.7), to be operational in RUN, IDLE or POWER DOWN modes.

The LF Interface supports Manchester coded command or data reception, employing ASK modulation of the LF carrier at 3.9 kbps. After detection of a valid LF telegram header, reception of data may be handled by the library function LF_Data_Read (ref. 8.4.8).

While the LF interface monitors the coil inputs, the RISC may be put into IDLE or POWER DOWN mode, in order to minimise the system power consumption. In case POWER DOWN or IDLE mode is entered, the LF Timer or Timer/Counter 0 respectively may be programmed to define a certain time-out period, in order to limit the LF scan time per sequence.

4.7.1 LF TELEGRAM FORMAT

When the LF Interface is enabled, it will autonomously monitor the coil inputs for a modulated LF carrier that must assemble an LF Telegram featuring a distinct format, in order to cause a device wakeup. Subsequently, data might follow that is received and demodulated by the LF circuitry. The LF Telegram format is shown in Figure 4-5. The LF Telegram comprises of Preamble, Synchronisation, wakeup ID, and optional Data.

4.7.1.1 Preamble

The Preamble is a sequence Manchester coded “zeros” which allows the LF Interface to settle its analog circuitry. The Preamble must feature a minimum length as specified.

4.7.1.2 Synchronisation

To initialise the on-chip Manchester decoder and to detect the start of the wakeup ID frame, a distinct synchronisation pattern needs to be received by the device (ref. Figure 4-5). The Synchronisation frame features a fixed length.

4.7.1.3 Wakeup ID

In order to force a wakeup, the device needs to receive a Manchester coded wakeup ID that matches the user programmed bit pattern. The wakeup ID is a string assembled by up to 16 bits, ref. 5.5.

When the received wakeup ID is detected to match the pre-programmed one, the device will generate an interrupt after the last bit of the wakeup ID has been received. Subsequently, the device will continue sampling the LF interface in order to detect and decode data sent optionally.

4.7.1.4 Data

After device wakeup Manchester coded data may optionally be sent to the device. Data is sent in 8-bit frames. These are decoded by the on-chip Manchester decoder and buffered in a UART-like 8-bit register. Reception of a full byte is signalled by setting interrupt flag 4 if enabled. If subsequent data frames are being received, the 8-bit register is overwritten at the end of each frame as well as an interrupt is being generated.

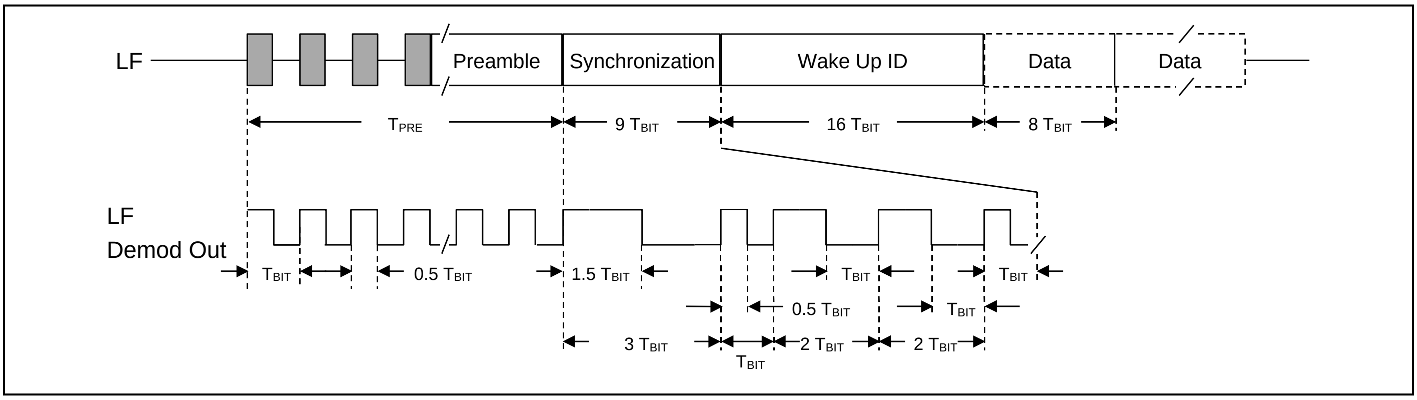

Figure 4-5 LF Telegram format

The Manchester coding is defined as follows: A zero of a Manchester coded bit is coded as a transition from high to low state, a one is coded as a transition from low to high state.

The Manchester-coded bit and the corresponding LF pattern is shown in Figure 4-6. During the high state of the coded bit the LF signal is on, during the low state the LF signal is off. The nominal bit rate is 3.9 kbit/s. This results in a bit length T of 256 µs nominal and a minimum pulse width of 128 µs.

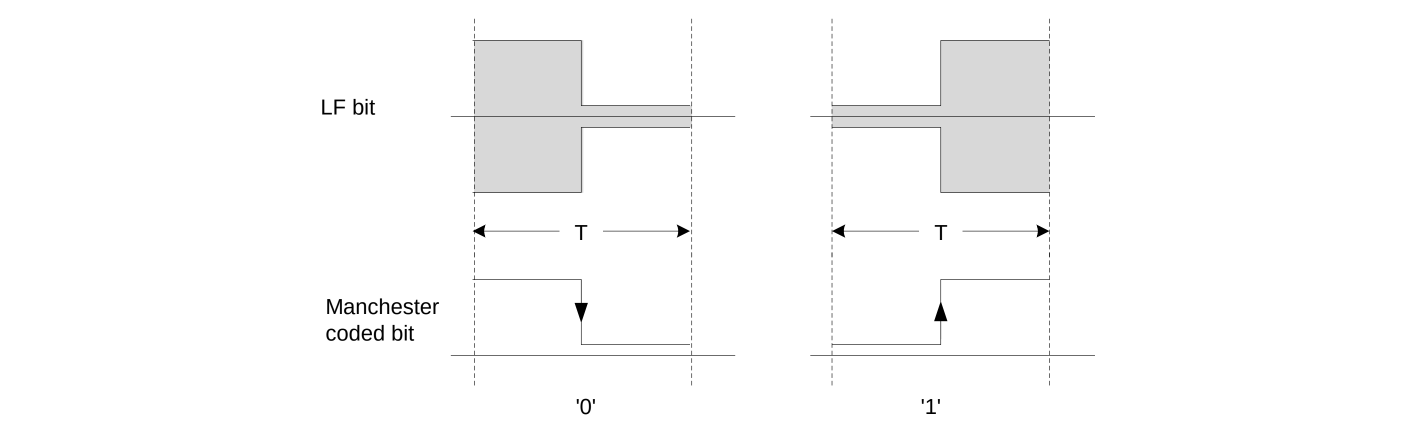

Figure 4-6 LF Manchester coding

4.7.2 LF CARRIER DETECTOR

The LF carrier detector provides a simple wakeup circuitry when no data/instructions are required to be sent to the sensor. Compared to the LF interface, the circuit has been designed less sensitive only to react on carrier amplitudes above a certain value. In addition, the carrier must be detected for a certain period of time before an interrupt is generated. For trigger level and time, refer to the specifications. The LF carrier detector is controlled by the Control Register located in the RISC Interface Register Set, ref. 5.3.

The Carrier detector interface sets an interrupt and a flag as soon as it detects a short 125 kHz carrier signal (i.e. LF signal). The flag is set as long as this LF signal is detected.

In case of a short LF pulse, the pulse duration must be larger than a specified value, ref. Figure 4-7.

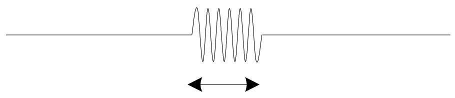

Figure 4-7 Detection of a short LF "pulse" requires a minimum signal duration

In case of a continuous LF signal, the detection will occur after a setup time + the minimum duration time, ref. Figure 4-8.

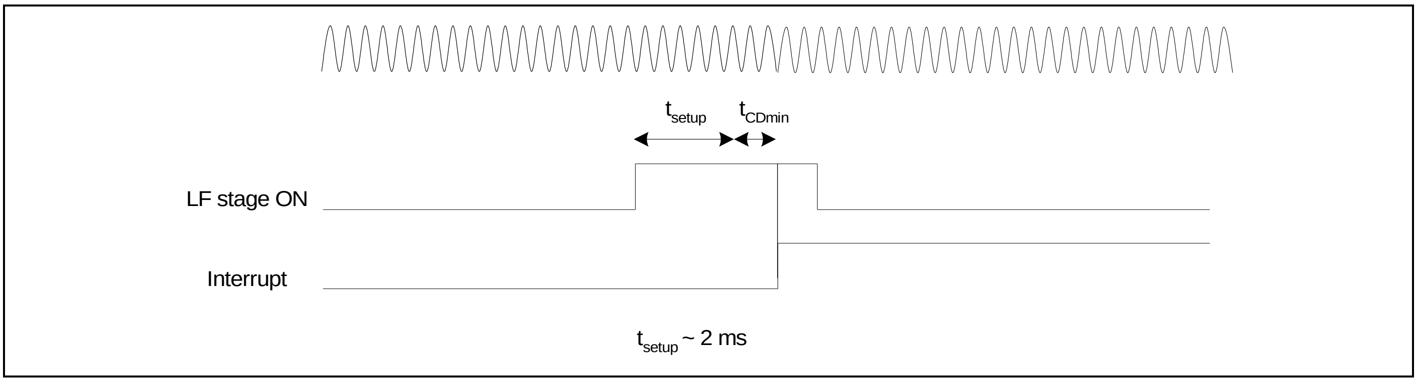

Figure 4-8 Detection of a continuous LF signal

If the LF signal duration is below t_{CDmin} or the amplitude is below S_{notdet}, the Carrier detector interface will not detect the LF signal.

The minimum LF signal to detect with respect to sensitivity and signal duration is shown in Figure 4-9.

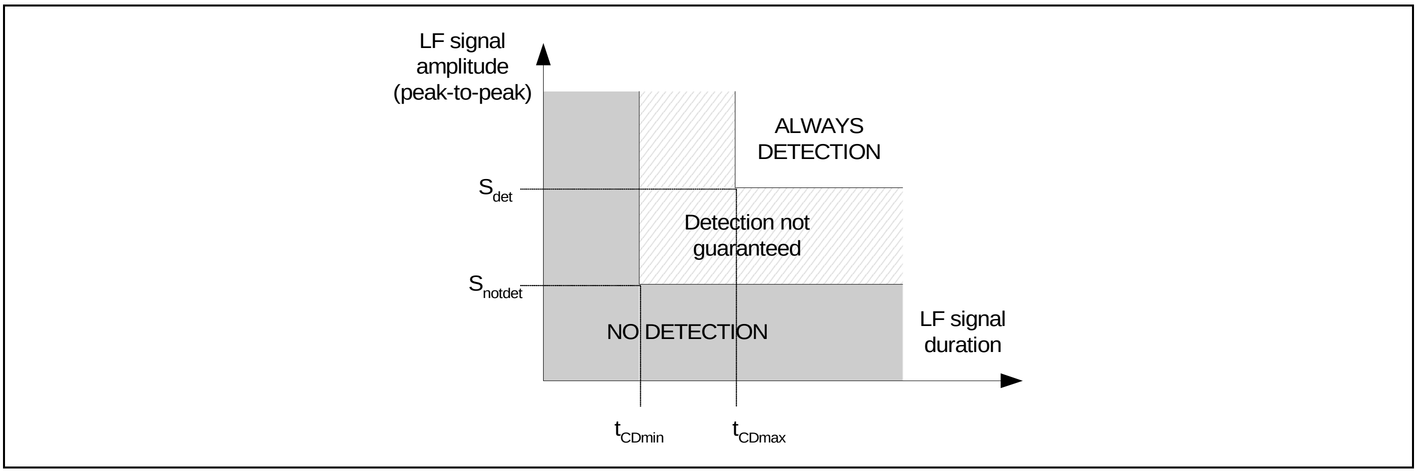

Figure 4-9 LF detection levels

4.8 TIMER/COUNTER 0

The device incorporates an asynchronous 8-bit timer/counter with auto reload feature and a 6-bit prescaler, ref. Figure 4-10.

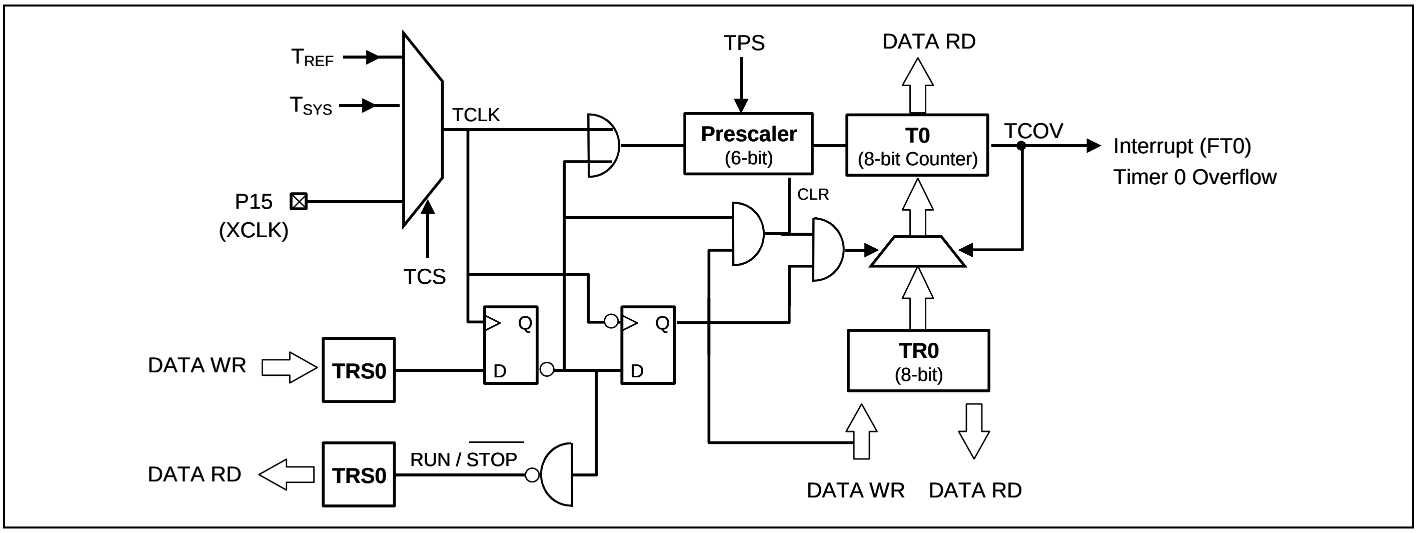

Figure 4-10 Timer/Counter 0 block diagram

The Timer/Counter 0 control bits are located in the Special Function Register TCON, see Table 4-4.

Table 4-4 Timer/Counter 0 Control Register, TCON

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
X	TPS2	TPS1	TPS0	TCS1	TCS0	X	TRS0
W0	R/W	R/W	R/W	R/W	R/W	W0	R/W

Note Address = 1Ah

Bits marked ‘X’ are not connected and reserved for future use. Any read operation yields an undefined result. For future compatibility a write operation should assign a ‘0’.

Timer/Counter 0 features a programmable 6-bit prescaler that provides prescaler values of 2^N for N = 0 to 6, selected by TPS<2:0> (ref. Table 4-5).

Writing to the Timer/Counter 0 Reload Register TR0 while the timer/counter is stopped, will clear the prescaler. Writing to and reading from the prescaler is not supported.

Table 4-5 Timer/Counter 0 Prescaler Select, TPS

TPS2	TPS1	TPS0	Prescaler Value	Note
0	0	0	1	
0	0	1	2	
0	1	0	4	
0	1	1	8	
1	0	0	16	
1	0	1	32	
1	1	0	64	
1	1	1	Reserved	

Timer/Counter 0 can operate as a timer or as an event counter, depending on the clock source selected by the corresponding control bits TCS, Table 4-6.

Table 4-6 Timer/Counter 0 Clock Source Select, TCS

TCS1	TCS0	Clock Source	Note
0	0	Reference clock, T_{REF}	1
0	1	System clock, T_{SYS}	1
1	0	Reserved	
1	1	External clock/event at Pin P15	

Note

1. The clock is derived from the on-chip high-precision oscillator

Timer/Counter 0 is incremented in response to a rising edge at its clock input. Upon overflow of the timer/counter register T0, the Timer/Counter 0 interrupt request flag FT0 is set (see section 4.14 for details). At the same time the timer/counter register is overwritten with the value stored in the timer/counter reload register TR0.

The timer/counter register T0 reload operation does not affect the prescaler state.

The Run/Stop control bit TRS0 controls the operation of timer/counter 0. A Run/Stop request is synchronized with the clock source (TCLK) and the timer/counter is incremented in response to a rising edge of the clock (TCLK), according to Figure 4-11.

To force Timer/Counter 0 into Run mode, a ‘1’ has to be written to the TRS0 flip-flop. To force the Stop mode, a ‘0’ has to be written. The corresponding request is latched upon the next rising edge of the clock signal TCLK and signaled by the Run/Stop mode flip-flop. Subsequent clocks will be recognized/ignored respectively by the timer/counter.

Note that the prescaler does not recognize the clock which acknowledges the RUN mode, while it does recognize the clock which acknowledges the STOP mode, ref. Figure 4-11.

The counter and prescaler states are not changed in Stop mode. Reading the control bit TRS0 signals, if Timer/Counter 0 is running or stopped.

The timer/counter register, T0, is located in the SFR address range and available for reading only. A write operation has no effect. The register T0 is initialized via the reload register TR0.

The timer/counter reload register, TR0, is located in the SFR address range and is available for reading and writing. In case Timer/Counter 0 is stopped (TRS0 = 0), writing to TR0 will also affect T0, since T0 does receive a copy of the value loaded into TR0. If the timer/counter is running (TRS0 =1), T0 is not affected by a write operation to TR0.

It is important to notice that the system clock (instruction clock) and Timer/Counter 0 clock may be asynchronous to each other, depending on the selected clock sources. Thus reading from T0 by software may happen at the exact moment in which the timer/counter is incremented. In this case, the read value may be undefined. Thus the timer/counter should be stopped before reading T0. Alternatively, successive readings of T0 should be performed in order to verify the read results against each other.

The Timer/Counter 0 control register is initially cleared after reset, and the timer is stopped.

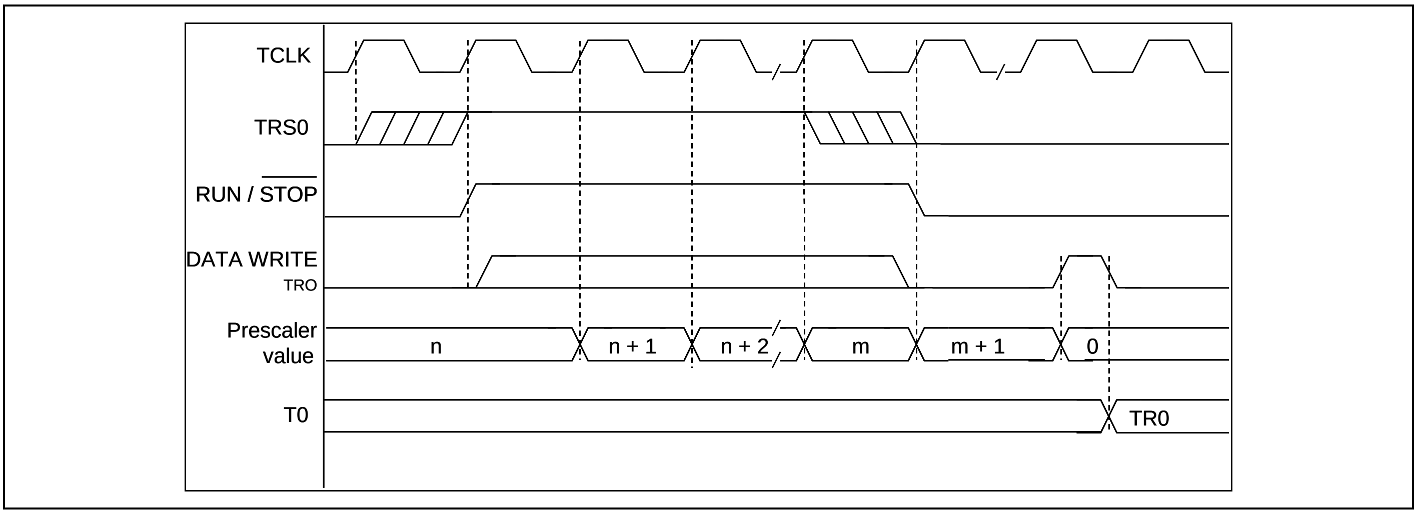

Figure 4-11 Timer/Counter 0 timing

4.9 WATCHDOG TIMER

The device incorporates a Watchdog Timer, to recover the system from situations in which the application program has run into a deadlock situation. This helps to prevent system malfunction and unintended battery discharge for wireless applications. If the Watchdog Timer overflows, it will force SP300 into POWER DOWN mode.

The Watchdog Timer as shown in 0 is *always* active when the system operates in RUN or IDLE Mode. When active, the watchdog prescaler is clocked from the reference clock (T_{ref}, see section 4.6).

The Watchdog Timer consists of an 8-bit prescaler and an 8-bit main timer, WT. The main timer is clocked by a tap taken from the prescaler according to Table 4-7.

Table 4-7 Watchdog Prescaler Select, WPS

WPS2	WPS1	WPS0	Prescaler Ratio	Note
0	0	0	256	
0	0	1	128	
0	1	0	64	
0	1	1	32	
1	0	0	16	
1	0	1	8	
1	1	0	4	
1	1	1	2	

To prevent the main timer from overflow, the application program has to write a ‘1’ to the control bit WCLR prior to the timeout event. Writing a ‘0’ to WCLR has no effect. WCLR is not latched internally. Reading from WCLR always yields a ‘0’.

The Watchdog Timer control bits are located in the Special Function Register WTCON, see Table 4-8.

Table 4-8 Watchdog Timer Control Register, WTCON

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
X	WPS2	WPS1	WPS0	X	X	X	WCLR
W0	R/W	R/W	R/W	W0	W0	W0	R0/W

Note Address = 1Bh

1. Bits marked 'X' are not connected and reserved for future use. Any read operation yields an undefined result. 2. Reading of the bit WCLR yields a '0'.

Initially after reset, the WTCON register is cleared, which configures the longest possible timeout time.

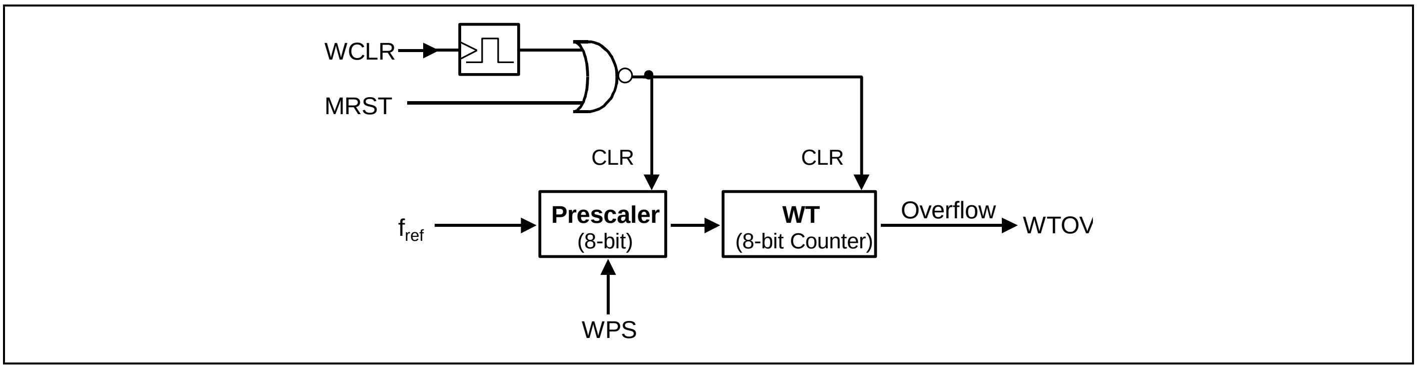

Figure 4-12 Watchdog Timer block diagram

4.10 VMIN

The VMIN circuitry checks whether the supply voltage is above the minimum level for measurements. The circuitry is used by some of the implemented library functions to check if measurements are performed at a valid supply voltage. This information is part of the Status byte (ref. 8.2) returned by these functions.

4.11 TMAX

In order to avoid undefined device operation above the normal operating temperature range, an on-chip Thermal Shutdown circuitry, TMAX, is included. TMAX is enabled by the library function "Thermal_Shutdown_Enable" (ref. 8.4.10) to put the device into THERMAL SHUTDOWN mode (ref. 3.3).

4.12 BOND WIRE SURVEILLANCE

The bond wire surveillance circuitry checks the integrity of the wire bond connections to the acceleration and pressure measurement bridges. The circuitry is used by the implemented functions for pressure and acceleration measurement and the result is placed in the STATUS byte (ref. Chapter 8).

4.13 RD SURVEILLANCE

The Rd surveillance circuitry checks the integrity of the acceleration beam. The circuitry is used by the implemented library function for acceleration measurement and the result is placed in the STATUS byte (ref Chapter 8).

4.14 INTERRUPT AND IDLE CONTROL

The SP300 employs a single-level interrupt architecture with four independently maskable interrupt sources (INT1 to INT4), see Figure 4-13. Interrupt masking is accomplished by the Interrupt Enable Register, IE. An Interrupt Flag register, IFF, stores pending interrupt requests generated by the corresponding peripheral.

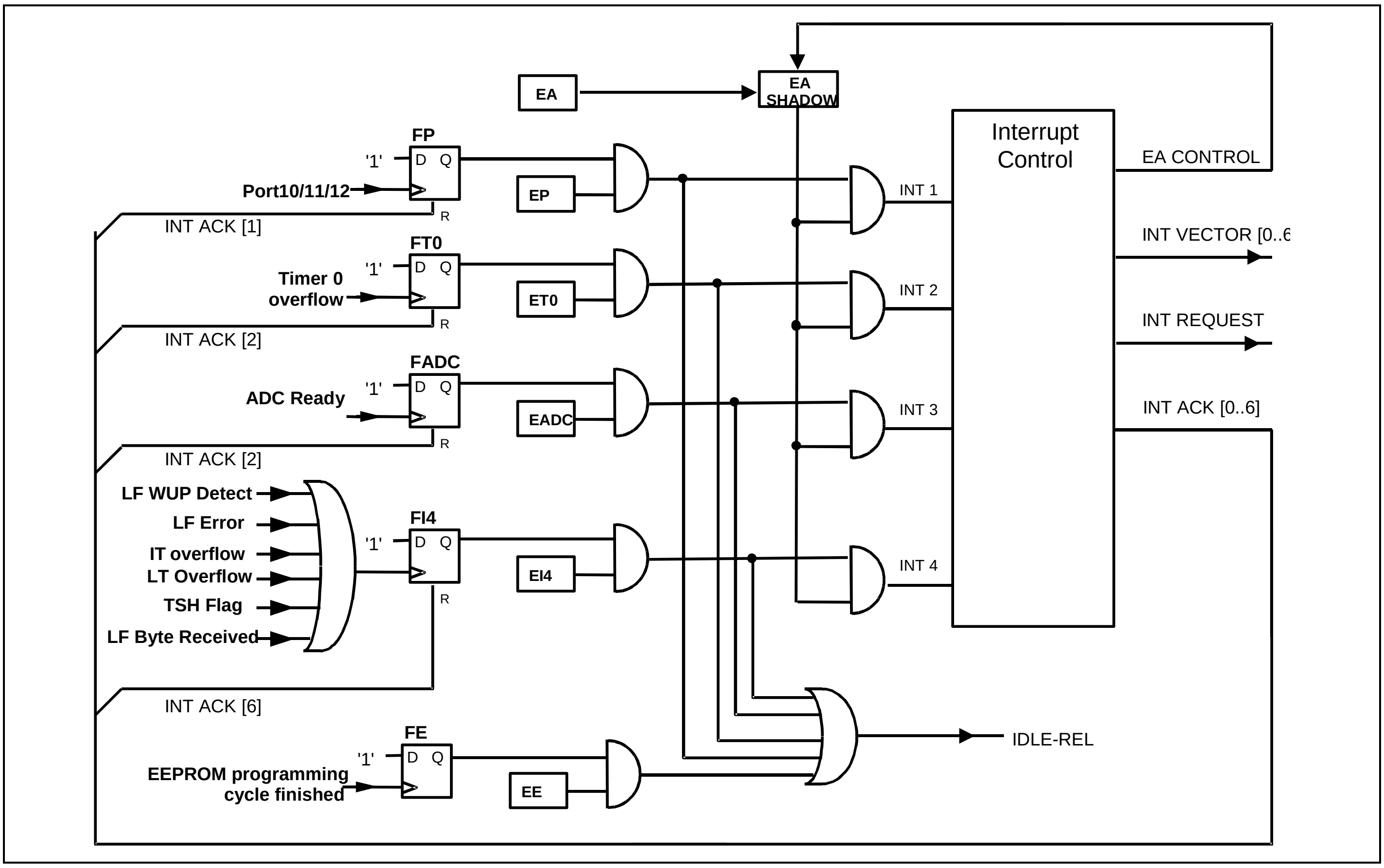

Figure 4-13 Interrupt control system

The interrupt sources operate at a common priority level, meaning that any interrupt service cannot be interrupted by subsequent interrupt requests until it is terminated by a RETI instruction. However, a multi-level interrupt structure can be constructed in software by manipulating the EA flag during interrupt service.

In case of simultaneous interrupts (e.g. occurred during the execution of an interrupt service), the interrupt with the lowest vector address will be serviced next. However, at least one instruction of the main program is executed between successive interrupts.

Interrupt vectors are assigned to fixed locations in the Application Program Memory. The assigned address space for each vector is shown in the Application Program Memory. See Figure 6-1.

All interrupts are initially disabled after a device Reset.

4.14.1 INTERRUPT ENABLE REGISTER

The Interrupt Enable Register, IE, is located in the SFR address space and contains several bits that control the interrupt system to feature interrupt masking, see Table 4-9.

Interrupts will not be serviced while EA is cleared. If EA is set, interrupts are serviced according to the setting of the corresponding interrupt enable bit. In any case, interrupts will be latched until vectored and may alternatively serve to terminate the IDLE mode, if enabled by the corresponding bit.

Table 4-9 Interrupt Enable Register, IE

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EA	X	EE	EI4	EADC	ET0	EP	X
R/W	W0	R/W	R/W	R/W	R/W	R/W	W0

Note Address = 16h

Bits marked ‘X’ are not connected and reserved for future use. Any read operation yields an undefined result. Interrupt request flags

Interrupt requests are latched in corresponding flags of the Interrupt Flag Register, IFF. See Table 4-10

Table 4-10 Interrupt Flag Register, IFF

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
X	X	FE	FI4	FADC	FT0	FP	X
W1		R/0	R/0	R/0	R/0	R/0	R/0

Note Address = 17h

1. Bits marked ‘X’ are not connected and reserved for future use.
2. Bits marked ‘R/0’ may be cleared only by a corresponding instruction. Any set operation does not affect the bit at all.

The corresponding interrupt request flip-flop is cleared automatically when the interrupt is serviced, except for the interrupt flag FE, which serves a dedicated function in the context of IDLE mode release.

4.14.2 INTERRUPT SOURCE ASSIGNMENT

The interrupts vectors, located in the Application Program Memory, are assigned to the sources as listed in Table 4-11. Since each interrupt vector leaves space for not more than a single instruction, the application program should place a JMP instruction to the address of the actual interrupt service routine. A RETI instruction should be placed instead at all unused vector addresses.

Table 4-11 Interrupt Source Assignment

Vector	Address	Source	Note
INT 0	0002H	Reserved	
INT 1	0004H	Port10/11	
INT 2	0006H	Timer 0 Overflow	
INT 3	0008H	ADC Ready	
INT 4	000AH	LF WUP Detect LF Error IT Overflow LT Overflow Temp SHTD LF Byte Received	1
INT 5	000CH	Reserved	
INT 6	000EH	Reserved	

Note

1. INT4 serves a number of interrupt sources and the corresponding service required need to be determined by testing the corresponding flags provided for each source.

4.14.3 INTERRUPT SERVICE

When an interrupt request is detected, the Interrupt Control logic clears the EA SHADOW flag (ref. Figure 4-13) rather than the EA bit itself, to prevent subsequent interrupt requests from being serviced. However, the subsequent interrupt is being latched.

The interrupt in service is acknowledged automatically by the Interrupt Control Logic, by clearing the corresponding bit in the IFF register. Subsequently, the RISC is forced to perform a CALL instruction to the corresponding vector address. The CALL instruction saves the processor status (Program Counter, PC, and Program Status Word, PSW) on the Call Stack, see also sections 6.4.1 and 6.4.2.

The Interrupt Service Routine must be terminated by executing a RETI instruction. The RETI instruction restores the program status (PC and PSW), in order to resume program execution at the corresponding location. Subsequently, the Interrupt Logic sets the EA SHADOW flag to enable pending or new interrupts to be serviced.

To enable interrupt nesting, the application program may set the EA flag, causing the EA SHADOW flag to be set again, enabling interrupt nesting and service for pending or future interrupts.

4.14.4 IDLE MODE

Any interrupts enabled by setting their corresponding bit in the IE register will terminate IDLE mode and force the device to resume program execution, see Figure 4-13.

The IDLE mode is typically used alternatively and mutually exclusive to interrupt services. Thus the Global Interrupt Enable bit EA is cleared in this case and no interrupts are invoked. However, if the control bit EA is set, thus general interrupt service is enabled, the corresponding interrupt will be serviced after termination of the IDLE mode.

The device executes the instruction following the one that forced the device into IDLE mode first, before the corresponding interrupt is serviced.

The IDLE mode will not be entered, if the corresponding interrupt flag (IFF) is already set for an interrupt that is enabled, while EA is cleared. Thus the corresponding bits in the IFF register should be acknowledged prior to IDLE mode invocation. The Interrupt Control logic does not perform this step, because the interrupt has not been serviced. Clearing of any interrupt request bits shall be accomplished by the INTA instruction, while masking all interrupts that shall not be affected. A Read-Modify-Write instruction (such as bit manipulation) should not be used, in order to avoid unintentional clearing of interrupt request bits and potential loss of an interrupt event, when latter one occurs after the Read but before the Write phase of the instruction.

The involved steps before and after using the IDLE mode are as follows:

1. The application programs the IE register by clearing the EA bit and setting the corresponding bits of all interrupt sources that shall be able to release the IDLE mode. The same bits in the IFF register should be cleared, in case they are set by unintended interrupt requests that occurred in the past.

2. The application program calls the “Idle” function (ref. 8.4.16) and enters the IDLE mode. Alternatively, the application program could directly set the IDLE bit (ref. 3.4).

3. The IDLE control logic detects an enabled interrupt request and clears the IDLE bit, terminating the IDLE mode (wakeup).

4. The RISC resumes execution at the first instruction following the instruction that sets the idle mode bit. If more than one interrupt source was enabled in step 1, the IFF register should be read to determine the source that has caused the wakeup. In any case, the application program is obliged to clear the corresponding bit in the IFF register to acknowledge the interrupt request. Note that the Interrupt Control logic does not perform this step, because the interrupt has not been processed and vectored in this case (see section 4.14.3).

Clearing of any interrupt request bits should be accomplished by the INTA instruction, while masking all bits that shall not be affected by a 1. The interrupt request bits do not support setting by instruction. A Read-Modify-Write instruction (such as bit manipulation) should not be used, in order to avoid unintentional clearing of interrupt request bits and potential loss of an interrupt event, when latter one occurs after the Read but before the Write phase of the instruction.

If any enabled interrupt sources have already signaled an interrupt request by setting the corresponding bit in the Interrupt Flag register (IFF), the IDLE mode cannot be entered. Setting the IDLE bit by the application program has no effect in this case.

4.15 DIGITAL MODULATOR

SP300 features an on-chip digital modulator circuitry for use with the I/O Port. The modulator supports signal train generation, e.g. with Manchester/Bi-Phase or Pulse Width coding and features a sub-carrier mode.

The digital modulator circuitry may be configured to take control of the port line P14 or P17.

Signal train timing is controlled by a data clock (DCLK), which is provided from Timer/Counter 0 (ref. Section 4.8).

Modulator operation is configured by the control bits located in the Special Function Register MODCON, see Table 4-12.

Table 4-12 Modulator Control Register, MODCON

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MDB	X	X	X	SCEN	EP17	EP14	X
R/W	W0	W0	W0	R/W	R/W	R/W	W0

Note Address = 1DH

Bits marked ‘X’ are not connected and reserved for future use. Any read operation yields an undefined result.

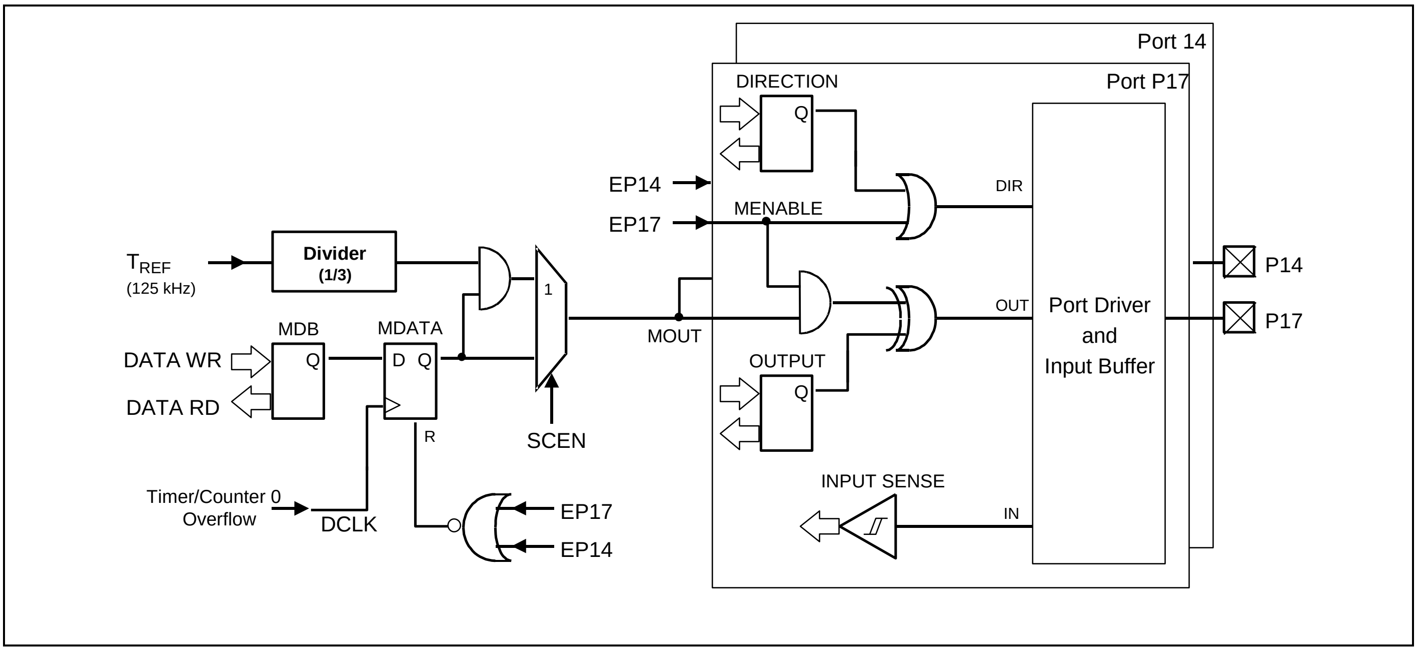

Figure 4-14 Modulator configuration and interaction with port

The flip-flop MDATA is clocked by the Timer/Counter 0 overflow, in order to immediately latch the data that is stored in the Modulator Data Buffer, MDB, avoiding signal train phase jitter due to the interrupt or polling response time. A corresponding Timer/Counter 0 interrupt service routine has to serve the Modulator Data Buffer with the value designated to be output with the next Timer/Counter 0 overflow.

When for example Manchester coding shall be implemented, the Timer/Counter 0 consequently needs to be operated at twice the bit rate.

The flip-flop MDATA is cleared when EP14 and EP17 is cleared.

The modulator may be configured to control either P17 or P14, according to the setting of the configuration bits EP14 and EP17. If more than one bit is set, the circuits involved will be served with the same signal train in parallel.

If the EP17 (or EP14) is set, the corresponding port direction flip-flop is overruled, which causes the port line to be configured for output mode. The port line is controlled by an XOR function of the Modulator Data flip-flop, MDATA, and the corresponding port output flip-flop. Thus the modulator state after reset may be adapted to the port line configuration that shall be present before and after the modulator controlled operation ("HIGH" or "LOW").

The modulator circuitry features a sub-carrier mode that can be applied for the signal train generated at port P17 or P14. The sub-carrier is enabled if the control bit SCEN is set. The sub-carrier is derived from the reference clock by division by 3 and features a duty cycle of 33%, however, is not synchronised with the bit clock (DCLK). The SCEN bit should be cleared when the modulator is not used, in order to minimize power consumption.

Initially after reset, the Modulator Control register, MODCON, is cleared, which will disable the modulator by default.

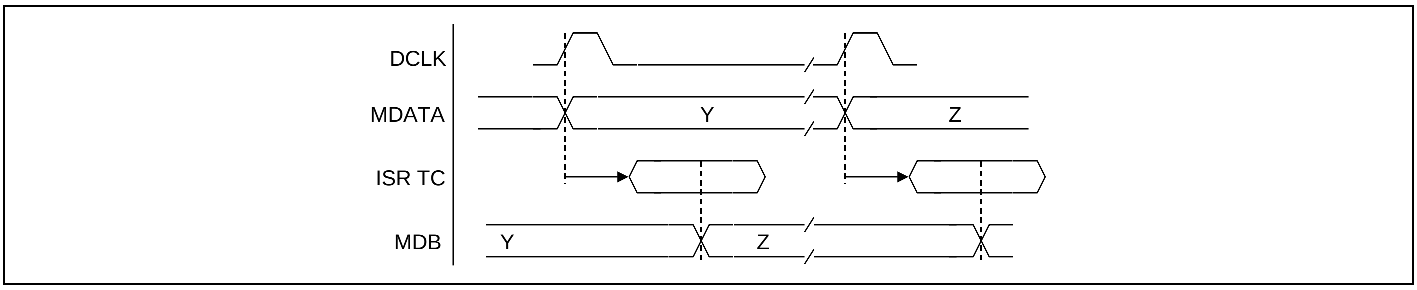

Figure 4-15 Port modulator timing

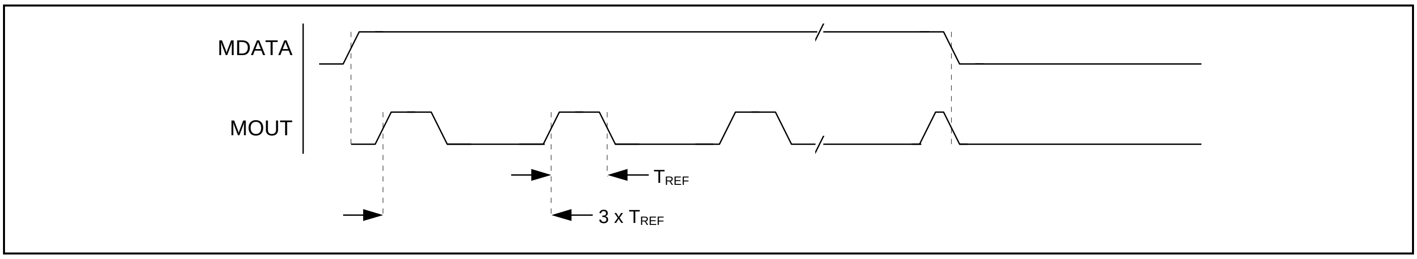

Figure 4-16 Sub-carrier timing

4.16 I/O PORTS

The device features an I/O port structure, Port 1, to interact with external peripherals. 5 independently configurable port lines are available.

All port lines as shown in Figure 4-17 are configured in "push-pull" fashion when used in output mode. Port P10 and P11 feature an additional pull-up/pull-down resistor, providing a defined level when used in input mode. For all other ports, external resistors need to be provided, when needed.

Although fully controlled by the application program, some I/O lines are assigned typical control functions. Table 4-13 gives an overview of the functions the corresponding port line serves in addition to its generic I/O function.

Table 4-13 Port Function Assignment

Port line	Typical function in a wireless application	Note
P10	wakeup sense	1
P11	wakeup sense	1
P14	Digital Modulator output	
P15	External clock input	
P17	Digital modulator output	

Note

1. Features internal programmable pull-up/pull-down resistor

P10 and P11: May serve as wakeup input and connects to the wakeup sense circuitry. The wakeup sense circuitry features to release the device from POWER DOWN mode by a high-to-low transition at the input. For applications where P10 or P11 can be held low for longer periods (i.e. roller switch), the pull-up resistors can be disabled in order to reduce current consumption during POWER DOWN mode. This is controlled by PDP11 and PDP10 (ref. Table 5-11).

P14 and P17: May be controlled by the on-chip digital modulator circuitry that supports signal train generation, e.g. Manchester/Bi-Phase, Pulse Width, etc. (ref. 4.15).

P15: May serve as an external clock input (ref. 4.7.2).

All port lines may be independently configured for input or output, as defined by the Special Function Register P1DIR, see Table 4-14. Other peripheral units sharing I/O pin resources may overrule pin direction register.

Table 4-14 P1 Direction Register, P1DIR

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IO17	X	IO15	IO14	X	X	IO11	IO10
R/W	W0	R/W	R/W	W0	R/W	R/W	R/W

Note Address = 22h

Bits marked ‘X’ are not connected and reserved for future use. Any read operation yields an undefined result.

If the corresponding direction bit is set, the port line is configured for output and the corresponding line driver is enabled. As a result, the port line will be forced high or low, depending on the state of the output flip-flop. If the corresponding direction bit is cleared, the port line is configured for input and the corresponding line driver is disabled (tri-state), see Figure 4-17.

The port output flip-flop controls the state of the corresponding port line, if latter one is configured for output mode. Any read operation from the port output flip-flop will be executed by sampling the state of the flip-flop rather than the state of the port line, see Figure 4-17.

The port output register of Port 1, P1OUT, is located in the SFR range, see Table 4-15.

Table 4-15 P1 Output Register, P1OUT

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P17	X	P15	P14	X	X	P11	P10
R/W	W0	R/W	R/W	W0	R/W	R/W	R/W

Note Address = 23h

Bits marked ‘X’ are not connected and reserved for future use. Any read operation yields an undefined result.

Reading from the port lines is accomplished by means of the Special Function Register Port 1 Input Sense, P1INS, see Table 4-16.

Table 4-16 P1 Input Sense Register, P1INS

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P17S	X	P15S	P14S	X	X	P11S	P10S
R	W0	R	R	W0	R	R	R

Note Address = 21h

Bits marked ‘X’ are not connected and reserved for future use. Any read operation yields an undefined result.

P1INS directly senses the port pin and return the corresponding states of the I/O lines, see Figure 4-17.

Initially after a Reset, the port direction and output flip-flops are cleared, which will configure all port pins for input. In this situation, ports P14, P15 and P17 need to be driven externally in order to prevent them from floating. A weak pull-down or pull-up resistor is sufficient to “tie” the port to VSS or VDD.

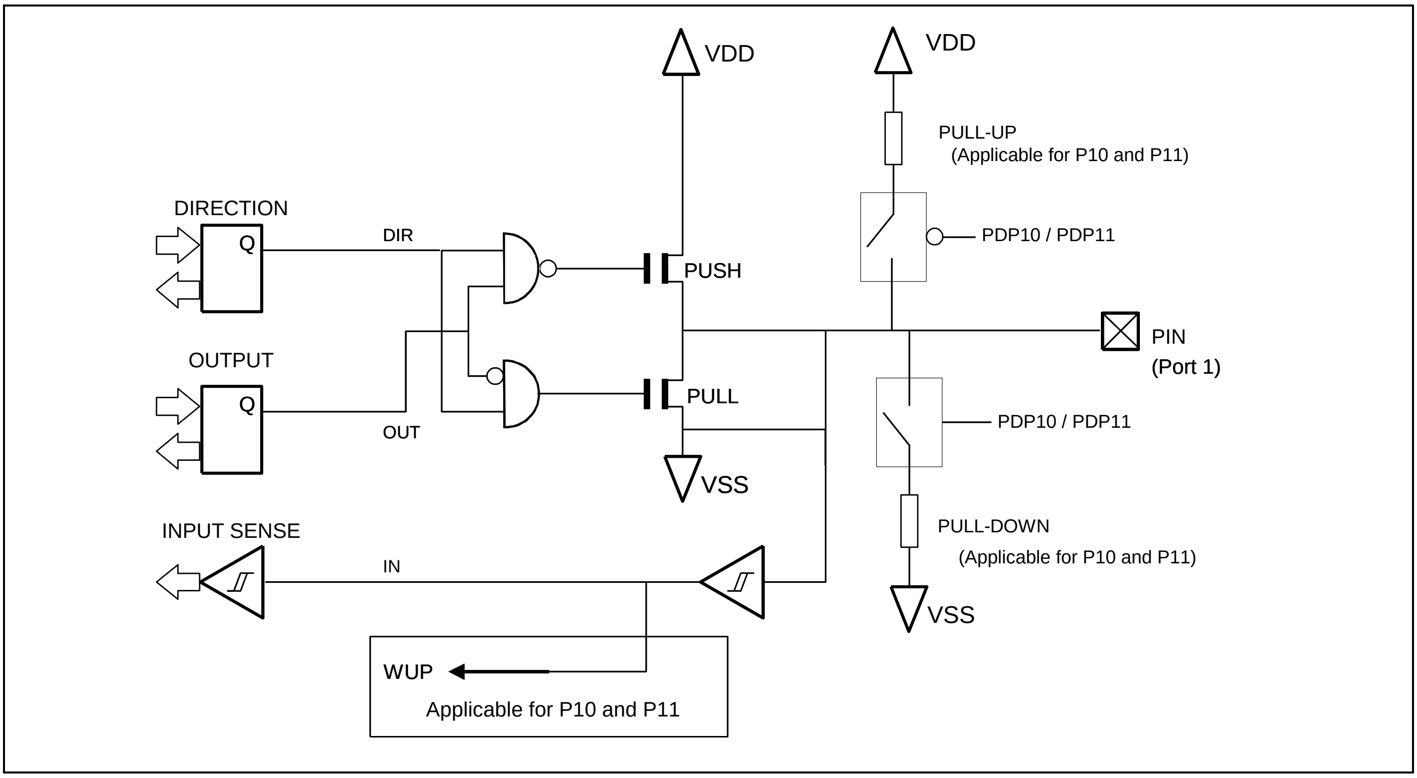

Figure 4-17 Port 1 configuration

5 RISC INTERFACE REGISTER SET

The RISC Interface Register Set is an independent address space that is powered in all operating modes. The address space cannot directly be accessed by the RISC. Two registers in the SFR (ref. Table 2-2) are used for indirect access to the RISC Interface Register Set; RICAR and RIDR.

The RISC Interface Register designated for R/W access is selected by address bits, RISC Interface Address, RIA.

Table 5-1 RISC Interface Control/Address Register, RICAR

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
X	X	RIA5	RIA4	RIA3	RIA2	RIA1	RIA0
W0	W0	W0	R/W	R/W	R/W	R/W	R/W

Note Address = 34h

Bits marked 'X' are not connected and reserved for future use. Any read operation yields an undefined result.

Read and write access to the designated RISC Interface registers is executed via the RISC Interface Data register, RIDR. Accessing the RIDR register for Read or Write results in a Read or Write access of the corresponding RISC Interface Registers, as selected by RIA. Write access to the RISC Interface registers location 08h – 0Eh is not possible.

Table 5-2 RISC Interface Data Register, RIDR

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RID7	RID6	RID5	RID4	RID3	RID2	RID1	RID0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Note Address = 35h

Table 5-3 gives an overview of the RISC Interface Register Set, while description of each individual bit are explained on the referred pages.

Table 5-3 RISC Interface Register Set Summary

NAME	ADDR	REF. PAGE	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	RESET VALUE
LF Data Register	00h	34	RID7	RID6	RID5	RID4	RID3	RID2	RID1	RID0	11111111b
Status Register	01h	34	LTOV	TSH	CDET	BITF	ITOV	NBYTE	WUP2	WUP1	XX00X000b
Control Register	02h	35	TSHEN	LFEN	ENCD	CDF	X	SWCD	WUL2	WUL1	XX000011b
Power Down Register	03h	35	PDAGC	PDBB	0	0	PDC21	PDC20	PDC11	PDC10	11111111b
WUP1 Byte0 Register	04h	36	W1_7	W1_6	W1_5	W1_4	W1_3	W1_2	W1_1	W1_0	11111111b
WUP1 Byte1 Register	05h	36	W1_F	W1_E	W1_D	W1_C	W1_B	W1_A	W1_9	W1_8	11111111b
WUP2 Byte0 Register	06h	36	W2_7	W2_6	W2_5	W2_4	W2_3	W2_2	W2_1	W2_0	11111111b
WUP2 Byte1 Register	07h	36	W2_F	W2_E	W2_D	W2_C	W2_B	W2_A	W2_9	W2_8	11111111b
Test and Trim Registers	08h-0Eh	36									
Timer Register	0Fh	37	LT1	LT0	IT1	IT0	ELT	X	PDP11	PDP10	XXXXXXXXb
Flag registers	10h-1Fh	37									

Note

Bits marked 'X' are reserved for future use or device test. Any read operation yields an undefined result.

5.1 LF DATA REGISTER

After receiving a full byte from the LF interface it will be stored in the data register.
This register is handled by library function LF_Data_Read, ref. Chapter 8.4.8.

Table 5-4 LF Data Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RID7	RID6	RID5	RID4	RID3	RID2	RID1	RID0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Address = 00h

5.2 STATUS REGISTER

The bits in the Status Register are used as a wakeup for Power Management, for start/end detection of the active protocol and for overflow monitoring of the LF and IT timer ticks generating an interrupt.

Table 5-5 Status Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LTOV	TSH	CDET	BITF	ITOV	NBYTE	WUP2	WUP1
R/E	R/E	R/E	R/E	R/E	R/E	R/E	R/E

Note Address = 01h

Bits marked ‘X’ are not connected and reserved for future use. Any read operation yields an undefined result.

LTOV: Overflow of LF Interface Timer tick

TSH: Reactivation after Thermal shutdown

CDET: Carrier detection

BITF: End of active protocol detected (used by library function LF_Data_Read)

ITOV: Interval timer overflow

NBYTE: New byte received by active protocol (used by library function LF_Data_Read)

WUP2: wakeup pattern 2 received, byte reception started

WUP1: wakeup pattern 1 received, byte reception started

Read/Write access (indicated with R/E in Table 5-5) to the Status Register has a different function than write access to other RISC Interface registers. By performing a write access to the Status Register the contents of the register can be reset. Which bit of the Status Register is reset can be determined by the contents of RIDR. A ‘1’ at a specific bit in RIDR will result in a reset of the corresponding bit (same bit position) in the Status Register (for example: RIDR = 01h, bit 0 of status register will be reset).

5.3 CONTROL REGISTER

This register is handled by the library functions Thermal_Shutdown_Enable (ref. 8.4.9), LF_Interface_Control (ref. 8.4.7), LF_Data_Read (ref. 8.4.8) and LF_Interface_On_Off (ref. 8.4.9).

Table 5-6 Control Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TSHEN	LFEN	ENCD	CDF	X	SWCD	WUL2	WUL1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Note Address = 02h

Bits marked ‘X’ are not connected and reserved for future use. Any read operation yields an undefined result.

TSHEN: Enables thermal shutdown block. This bit must be cleared during temperature measurements (ref. 8.4.2).

LFEN: Enables LF interface

ENCD: enables carrier detect function, disables LF protocol reception

CDF: indication if carrier is present

SWCD: switches between unmodulated and modulated carrier detection

WUL2: wakeup 2 pattern length (ref. 5.5)

WUL1: wakeup 1 pattern length (ref. 5.5)

5.4 POWER DOWN REGISTER

This register is handled by the library function LF_Interface_Control, ref. Chapter 8.4.7.

Table 5-7 Power Down Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PD AGC	PD BB	X	X	PD C21	PD C20	PD C11	PD C10
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Note Address = 03h

The different PD settings disable gain-stages of the preamplifier and the baseband amplifier. Bits marked ‘X’ are not connected and reserved for future use. Any read operation yields an undefined result.

Setting of the PDAGC-bit discharges the internal node of the Automatic Gain Control, AGC. This would be necessary during or after active communication to set the active reception unit into the ‘default‘ state (highest sensitivity).

Setting of PDBB disables gain-stage in baseband amplifier.

The PDCx1 and PDCx0 are used for LF interface calibration:

Table 5-8 Gains stages

PD Cx1	PD Cx0	Mode
0	0	All enabled
0	1	3rd amplifier in chain disabled
1	0	2nd and 3rd amplifier in chain disabled
1	1	Whole channel disabled (+bias + rectifier)

5.5 WUP REGISTERS

An important part of the LF Telegram is the wakeup ID, ref. Figure 4-5. Two wakeup IDs can be programmed. In combination with bit 0 and 1 of the Control register (WUL1 and WUL2), a wakeup ID length from 1 to 17 bits can be defined. (Most significant bit of the wakeup Pattern is always defined as '0'.) The most significant '0' in the 17-bit wakeup pattern will determine the pattern length.

WUP1 and WUP2 registers must be written after enabling the LF-circuit since disabling LF will reset the content of the WUP registers (ref. 8.4.7.)

The wakeup IDs should be defined according to Table 5-9 and Table 5-10.

WUP1 Register

Table 5-9 WUP1 byte0 and byte1register

Byte 0

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W1_7	W1_6	W1_5	W1_4	W1_3	W1_2	W1_1	W1_0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Address = 04h

Byte 1

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W1_F	W1_E	W1_D	W1_C	W1_B	W1_A	W1_9	W1_8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Note

Address = 05h

If short reception is required (ref. Chapter 8.4.7) the length is determined by the most significant "0".

WUP2 Register

Table 5-10 WUP2 byte0 and byte1register

Byte 0

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W2_7	W2_6	W2_5	W2_4	W2_3	W2_2	W2_1	W2_0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Address = 06h

Byte 1

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W2_F	W2_E	W2_D	W2_C	W2_B	W2_A	W2_9	W2_8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Note

Address = 07h

If short reception is required (ref. Chapter 8.4.7) the length is determined by the most significant "0".

5.6 TEST AND TRIM REGISTERS

These registers (Addresses 08h – 0Eh) contain trimming information and are not used by the application program.

5.7 TIMER REGISTER

The Timer Register controls two timers used for generating periodic wakeups; the Interval Timer (ref. 3.5.1) and the LF Interface Timer (ref. 4.7). A typical application would be to use these two wakeups to turn the LF interface on and off regularly. The LF Interface Timer is started by the Interval Timer.

The Timer Register also inlcudes two bits for P10 and P11 pull-up / pull-down configuration, PDP10 and PDP11 respectively.

Table 5-11 Timer Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LT1	LT0	IT1	IT0	ELT	X	PDP11	PDP10
R/W	R/W	R/W	R/W	R/W	W0	W0	W0

Address = 0Fh

LT: LF Timer control

LT1	LT0	Mode
0	0	25 ms
0	1	50 ms
1	0	75 ms
1	1	100 ms

IT: Interval Timer control

IT1	IT0	Mode
0	0	0.5 s
0	1	1 s
1	0	2 s
1	1	4 s

ELT: Enable LF Timer

ELT	Mode
0	LF Timer disabled
1	LF Timer enabled

PDP10/PDP11: pull-up/pull-down

PDP10 / PDP11	Mode
0	P10/P11 connected to pull-up
1	P10/P11 connected to pull-down

5.8 FLAG REGISTER

The Flag Register provides 16 byte of static memory implemented as latches for system state control purposes and storage of measurement data in the context of averaging successive readings.

The Flag Register has no reset condition and keeps its contents during POWER DOWN mode and THERMAL SHUTDOWN mode.

6 MEMORY

The RISC features a separate address space for program and data memory and an independent data bus and instruction bus structure. Each instruction consists of 16 bit and thus occupies two bytes. The program counter (PC) addresses the program memory instruction by instruction, which is always aligned to even addresses.

The RISC features byte wise addressing of code data, e.g. for lookup tables.

A monitor program is included that supports application program download and debug (ref. [2]).

6.1 APPLICATION PROGRAM MEMORY

The Application Program Memory is implemented as a 4 kbyte EROM holding the application and customer specific code. SP300 supports in-circuit programming and debugging. The Application Program Memory is organized according to Figure 6-1.

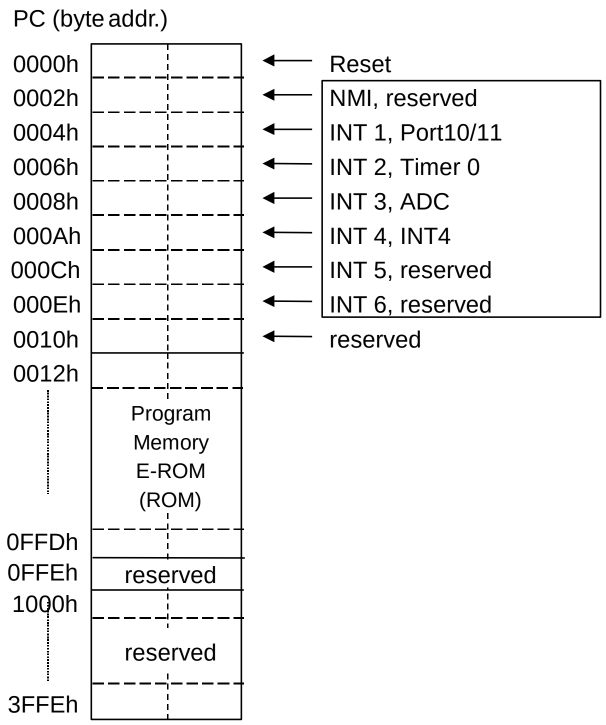

Figure 6-1 Application program memory

Note

Location 0FFEh is reserved for device configuration purposes and not available to the application.

After a device wakeup, execution of the application program commences from location 0000h.

The RISC features 7 interrupt vectors. Each interrupt is assigned a fixed location in the Application Program Memory. The device uses 4 of these 7 interrupt vectors only; the others are reserved.

Interrupt vectors INT1 to INT4 can independently be enabled or disabled. If an interrupt is enabled, an interrupt event will cause the RISC controller to perform a CALL operation to the corresponding location, where it commences execution of the Interrupt Service Routine (ISR). For example in case of an INT2 interrupt (Timer 0), program execution would commence at memory location 0006h. A Jump instruction needs to be placed at the corresponding location, in order to redirect program execution to the final location of the ISR (interrupt service routine).

6.1.1 CONTENT OF TWO LAST BYTES OF EROM.

All bytes of the EROM should be programmed, also when the application code size is less than available memory space. The bytes not in use should be filled up with a defined pattern, and the two last bytes should always be programmed with ‘00AFh’. An example for how to do this is shown below.

Code to be included in source:

```
------------------------------------------
      :
      :
?PR?Top_Of_EROM         SEGMENT CODE AT 0FFEh
      :
      :
ASEG    ?PR?Top_Of_EROM
        dw  0AF00h
      :
      :
-----------------------------------------
```

```
Hex dump generated in an example program:
------------------------------------------
:10026800837202F80F4A300A397C842284DB070A39
:0E0278000F4B300B397C842384DB070B00E531
:020FFE0000AF42
:00000001FF
------------------------------------------
```

When programming the last two bytes of EROM within RIDE software, the message “Code larger than the EROM size (1 byte ignored)” will appear. This can be ignored, since the locations are not used for program execution.

If the EDDC program is used for downloading, the last byte ‘AFh’ will be ignored and replaced with ‘AAh’ or ‘A5h’ if in VIRGIN or INIT mode respectively. This will affect the second nibble of the signature of the sensor (SX-SS-SS).

NOTE: To avoid lock-ups in field, the device must be put into PROTECT mode once the EROM and EEPROM are initialized and the debugging is completed.

6.2 FIRMWARE MEMORY

The Firmware Memory is implemented as a 4 kbyte ROM holding a predefined implemented set of library functions and generic low-level routines for use with the on-chip peripherals that may be called by the application program. The Firmware Memory space is not visible to the application. The provided functions are called by a dedicated instruction (SYS) that passes control back to the application program, when completed.

For more firmware description and library functions see ‘Library Functions’, Chapter 8.

6.3 DEVICE SPECIFIC MEMORY

The device specific memory is an EEPROM and features 128 bytes that are split into device calibration and configuration data, stored during device manufacturing (sensor calibration and configuration).

Eight pages (32 bytes) of the device specific memory (pages 00h – 07h) are available for the application program.

6.4 DATA MEMORY

The Data Memory address space is split into a register file (R0 to R7), the Program Status Word (PSW), Special Function Registers (SFR) and User RAM, see Figure 6-2.

The SFR space enables I/O from/to the peripherals, the transponder interface and EEPROM as well as control of the interrupt system.

The User segment provides 96 bytes of RAM for data storage for the application program. 32 of these bytes (60h to 7Fh) are directly addressed, and 64 bytes (80h to BFh) are indirectly addressed. The address space from 08h to 11h are reserved for internal RISC usage. The library functions have reserved the memory space from 40h to 5Fh. This area may however be used by the application program as a scratch area between library function calls.

The memory is disconnected from the supply voltage in POWER DOWN mode and thus volatile. Application data and flags required to be stored during POWER DOWN mode need to be placed in the Flag Register, see section 6.5.

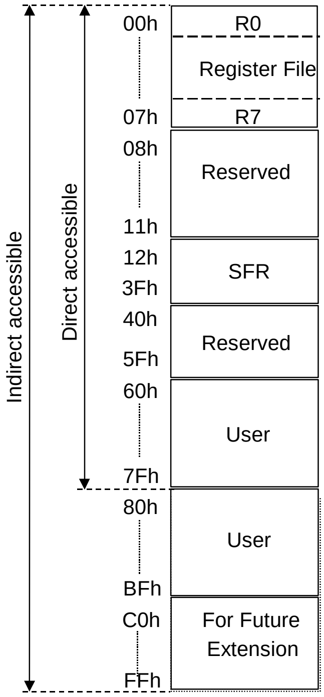

Figure 6-2 Data memory

6.4.1 PROGRAM STATUS WORD, PSW

The Program Status Word (PSW) contains several status flags that reflect the current state of the RISC. The PSW resides in the data memory and contains the Carry flag, Half-carry flag for BCD operations and the Overflow flag for signed operations, see Table 6-1.

Table 6-1 Program Status Word, PSW

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
C	H	OV	X	C’	H’	OV’	X

Note Address = 12h

Bits marked ‘X’ are not connected and reserved for future use. Any read operation yields an undefined result.

The Carry flag C and the Half-carry flag H serve as carry bit in arithmetic operations. The Carry flag receives a carry out from bit 7 of arithmetic operations while the Half-carry flag does for bit 3 of ALU operations. The Carry flag also serves as an “Accumulator“ bit for a number of Boolean operations.

The Overflow Flag, OV serves the function of an overflow during signed arithmetic operations. The overflow flag is set if there is a carry-out of bit 6 but not out of bit 7, or a carry-out of bit 7 but not bit 6. Otherwise OV is cleared.

The flags are located in the most significant nibble. The least significant nibble (C', H' and OV') may be used at any time to save and/or restore the actual flag status by a SWAP instruction.

6.4.2 CALL STACK

The RISC incorporates a stack which is used to store the return addresses during subroutine calls and during interrupt services. The stack is mapped into the Data Memory and organized in 16-bit words, i.e. each stack entry occupies two bytes of Data Memory. The stack is aligned to even byte addresses.

The Stack Pointer SP is defined by Special Function Register as shown in Table 6-2. Since the stack is aligned to even addresses, bit 0 of SP is not implemented and internally substituted with '0' during all accesses to the stack.

Table 6-2 Stack Pointer, SP

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SP7	SP6	SP5	SP4	SP3	SP2	SP1	X
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R0

Note Address = 13h

Bits marked 'X' are not connected and reserved for future use. Any read operation yields an undefined result.

Instructions/events which perform a "Push" on the stack (CALL, XCALL and interrupt service) first decrement the stack pointer by 2 before storing program address to the resulting address in the Data Memory ("pre-decrement") as shown in Table 6-3. For an interrupt event, also the three status bits of the PSW register (ref. 6.4.1) will be stored together with the program address. Note that bit 0 of the program counter (PC) is always zero.

Table 6-3 Organization of stack entry

at address SP-2 (even address)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1

at address SP-1 (odd address)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
C	H	OV	PC13	PC12	PC11	PC10	PC9

Instructions that perform a "pop" operation (RET, RETI) will use the address that the stack pointer SP currently points to, and retrieve the address from that location in the Data Memory. The RETI instruction will additionally retrieve the saved PSW contents (C, H and OV flags). The SP is incremented by 2 afterwards ("post-increment").

The PSW is pushed upon an interrupt service and restored again by the RETI instruction, which shortens the execution time of the interrupt service routine.

Size and location of the stack within the Data Memory are fully determined by the application program by programming an initial value to the stack pointer (SP) immediately after reset. Due to the "pre-decrement" concept the initial value is the address of the "top most" stack entry, plus 2. The application program is responsible to check for stack overflows. Stack depth for each library function is given in Chapter 8.

The reset value of the stack pointer is C0h, which places the beginning of the stack at the end of available memory. The first pushed address will thus be stored at addresses BEh and BFh in RAM. If desired, the application may initialize the SP with a new start value immediately after reset.

6.4.3 SPECIAL FUNCTION REGISTERS, SFR

A set of Special Function Register, SFR, is provided to enable access to the peripherals and to control operation of the interrupt system and power management. Table 2-2 in Chapter 2.4 provides a comprehensive overview of the SFR organisation and their corresponding values after a device reset.

6.5 FLAG REGISTER

Ref. 5.8.

7 BOOT SEQUENCE AND MONITOR MODE

7.1 BOOT SEQUENCE

After any device wakeup or Restart, the program execution starts with the ROM based boot routine. The boot routine executes a sequence of instructions that calibrate the oscillators, LF bias and VMIN circuits of the device and subsequently invoke the application program or the monitor program.

The boot sequence is further described in [2].

7.2 IN-CIRCUIT MONITOR AND DOWNLOAD ROUTINE

If activated, the firmware operates the in-circuit monitor and download interface, which features communication via a serial interface (MSDA/MSCL), in order to provide means for EROM and EEPROM initialization and to monitor and manipulate the embedded peripherals in the context of system debugging.

The monitor interface is further described in [2].

8 LIBRARY FUNCTIONS

8.1 FUNCTION INTERFACE DEFINITION

Figure 8-1 shows the default assignment of the parameters and return values to the corresponding registers, flags and RAM locations at the interface to the library function. Before the particular library function is called, the parameters and optionally the RAM have to be initialized by the application program.

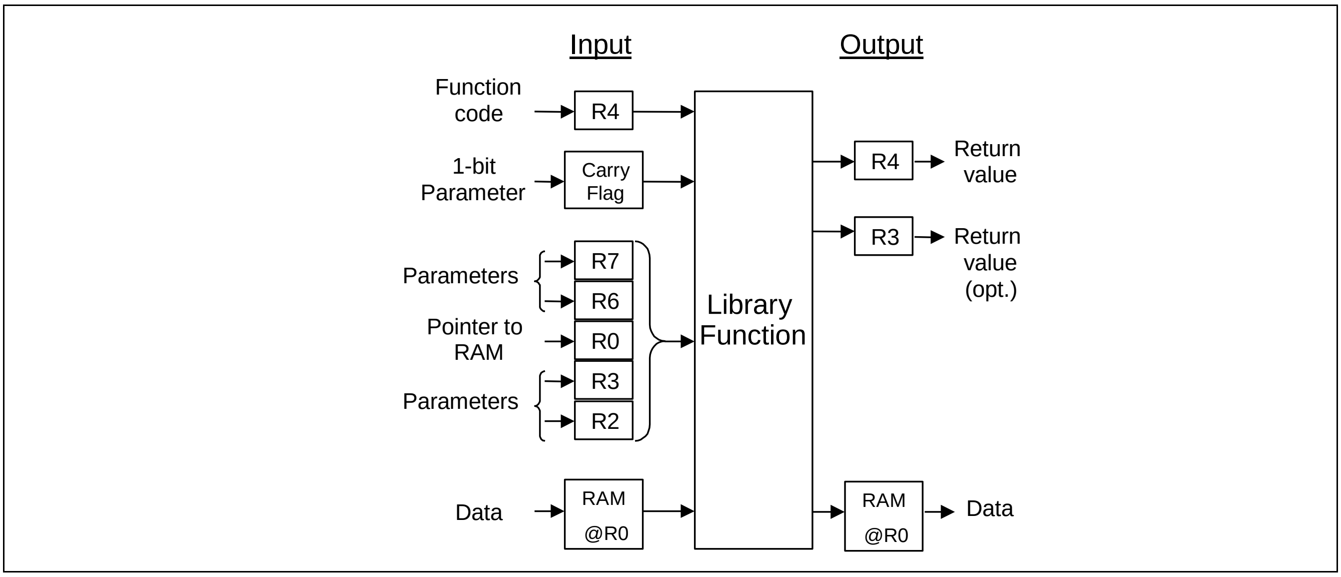

Figure 8-1 Function interface definition

8.2 STATUS BYTE

For several functions a status byte is returned in R4. The following convention applies:

- ‘0’ = OK or not tested
- ‘1’ = Fault or error detected

All functions except EEPROM functions (ref. Table 8-2) use the following definition for Status byte:

Table 8-1 Status byte definition

Bit	Definition	Note
0	Overflow or underflow in compensated measurement	1, 2
1	Low supply voltage. This bit indicates if the supply voltage is below the VMIN threshold during ADC conversion.	
2	Sensor fault. This bit indicates broken accelerometer or broken bond wire between ASIC and sensor	3
3	EEPROM error. This bit indicates parity error in calibration coefficients read from EEPROM.	
4	Reserved	
5	Reserved	
6	This bit is set if the Thermal_Shutdown_Enable function is called at too low temperature.	
7	Reserved	

Note

1. Bit 0 is set if compensated value is saturated at min or max value
2. For some library revisions, also ADC overflow/underflow is detected (ref. 11.2.2)
3. Tested if accelerometer is available

8.3 FUNCTION CALL

The functions included in this chapter can easily be accessed by the application programs, as shown in Figure 8-2.

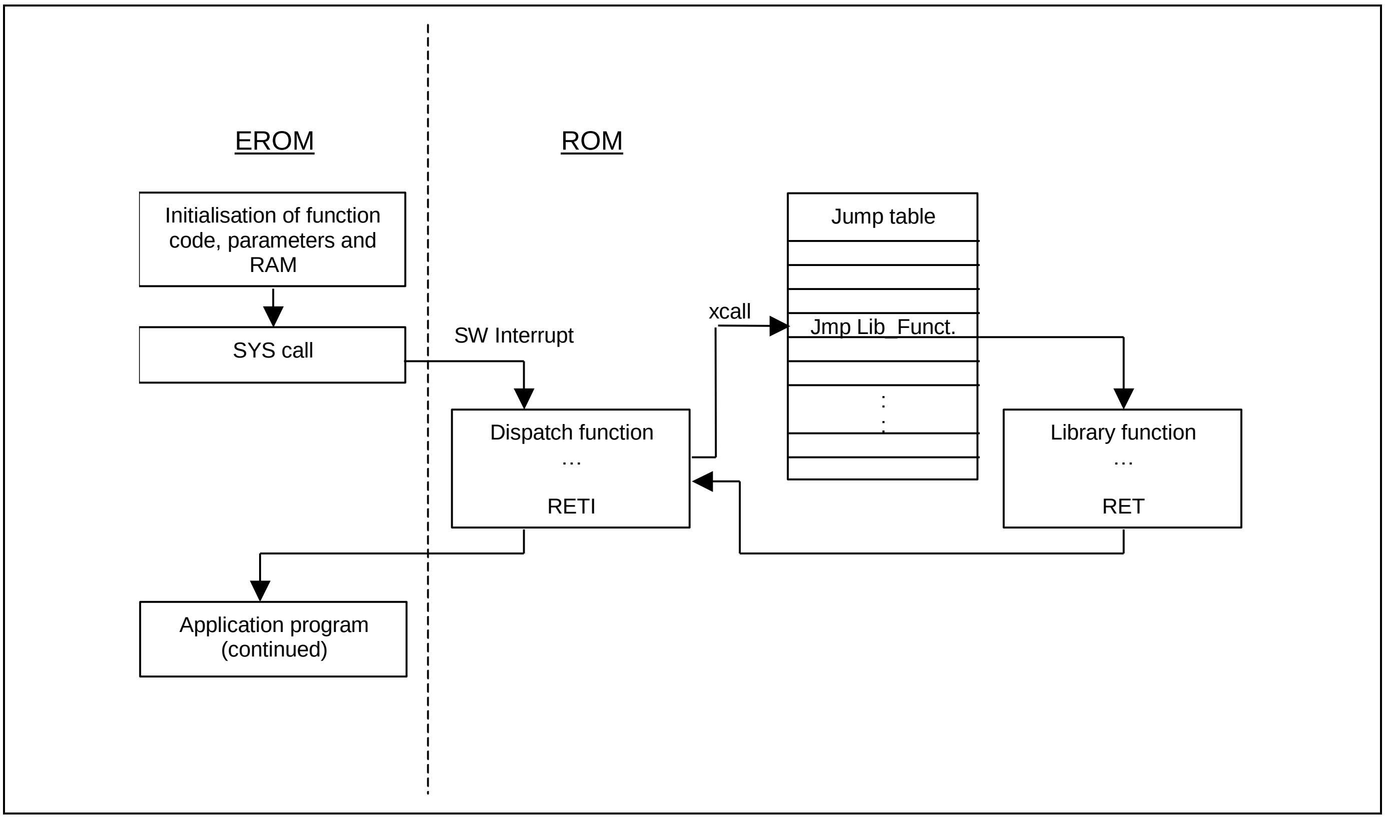

Figure 8-2 ROM library access mechanism

First the function parameters and optionally the RAM have to be initialized by the application program, then a SYS instruction has to be executed. The SYS instruction generates an interrupt, which disables the EROM and enables the execution of the desired function in ROM. During execution, the library function stores the defined return values in registers and optionally in RAM. The execution of the ROM library function is finished by a RETI instruction. After this, the application program commences in EROM.

Note that the execution of ROM library functions is not interruptible (EA flag is cleared and NMI is inhibited). Moreover, the ROM functions have full access to the device including its peripherals (e.g. the EEPROM).

8.4 FUNCTIONS

Table 8-2 Overview of general library functions

NAME	FUNCTION CODE	DESCRIPTION	REF
Sensor Measurements			
Meas_Sensor	32h	Measure pressure or acceleration	8.4.1
Meas_Temperature_Vdd	2Ch	Measure temperature	8.4.2
Meas_Supply_Voltage	16h	Measure supply voltage	8.4.3
Start_Offset_Voltage	34h	Start supply voltage offset measurement	8.4.4
Start_Supply_Voltage	18h	Start supply voltage measurement	8.4.5
Get_Supply_Voltage	1Ah	Complete supply voltage measurement	8.4.6
LF interface control			
LF_Interface_Control	2Eh	LF Interface on/off	8.4.7
LF_Data_Read	1Eh	LF Data Read	8.4.8
LF_Interface_On_Off	1Ch	LF Interface on/off, incl. calibration of carrier detection threshold	8.4.9
Miscellaneous			
Thermal_Shutdown_Enable	20h	Enable Thermal Shutdown mode	8.4.10
Multiply_Signed_16	28h	16x16 signed multiplication	8.4.11
Get_Library_Rev	2Ah	Get library revision	8.4.12
CRC8_Calc	22h	CRC-8	8.4.13
Signature_Test	24h	Signature test of program memory	8.4.14
Read_ID	26h	Read Serial Number	8.4.15
IDLE	7Ch	Enters IDLE mode and returns after an enabled interrupt has occurred	8.4.16
EEPROM			
EE_Read	48h	Reads a string of bits from the EEPROM	8.4.17
EE_Read_W_Parity	46h	Reads a string of bits from the EEPROM, incl. Parity check	8.4.18
EE_Write_Ext	52h	Programs a string of bits into an EEPROM page	8.4.19

The following sections describe each library function in more details.

Note: “Registers affected” shows which registers are changed by the library function. Input registers are listed here if the value is changed during function execution. Output registers are not listed.

8.4.1 MEAS_SENSOR

Description: This function performs a sensor measurement on specified channel and with specified resolution. By specifying sensor type (pressure or acceleration), the function selects correct sensor check and calibration coefficients. The result is calibrated and compensated for sensitivity, offset and temperature. The output format is described in Chapter 10.

Remarks: The pressure or acceleration raw data is the ADC result normalized to 12-bit resolution (right-adjusted). The value is represented as a 16-bit signed number.

Inputs:

R4	Function code, ref. Table 8-2
R2	Compensated temperature data low byte, ref. 8.4.2
R3	Compensated temperature data high byte, ref. 8.4.2
R6	ADC resolution, ref. Table 8-3
R7	LNA gain, sensor type and sensor channel, ref. Table 8-5

Outputs:

R4	Status byte, ref. 8.2. (Bits 0, 1, 2 and 3 affected)
R0	Pointer to return values placed in RAM

RAM:

@R0	Compensated pressure/acceleration low byte
@R0+1	Compensated pressure/acceleration high byte
@R0+2	Pressure/acceleration raw data low byte
@R0+3	Pressure/acceleration raw data high byte

Registers affected: R2, R3, R5, R6, R7

Stack depth: 8 bytes

Execution time: Execution time for this function is given by ADC resolution and µC clock frequency, ref. Table 8-4. Worst case values are used for data dependent calculation time.

Table 8-3 Contents of R6 for Meas_Sensor

Resolution [bits]	R6
5	0000 0000
6	0000 0001
7	0000 0010
8	0000 0011
9	0000 0100
10	0000 0101
11	0000 0110
12	0000 0111

Table 8-4 Execution time for Meas_Sensor

Resolution [bits]	Max Execution time [ms] given by µC clock frequency				
	2 MHz	1 MHz	500 kHz	250 kHz	125 kHz
5	1,5	2,6	4,7	9,1	17,8
6	1,5	2,6	4,8	9,1	17,8
7	1,6	2,7	4,8	9,2	17,9
8	1,7	2,8	5,0	9,3	17,9
9	2,0	3,1	5,2	9,6	18,2
10	2,6	3,7	5,8	10,1	18,7
11	3,8	4,8	6,9	11,2	19,8
12	6,3	7,4	9,5	13,7	22,1

Table 8-5 Contents of R7 for Meas_Sensor

Product	Pressure	Acceleration
SP300-0.5(T)	90h	X
SP300-1(T)	91h	X
SP300-2(T)	51h	X
SP300-7(T)	A0h	X
SP300-7A(T)	A0h	93h
SP300-16A	50h	93h

8.4.2 MEAS_TEMPERATURE_VDD

Description: This function performs a temperature measurement with specified resolution. The result is calibrated and compensated for offset, sensitivity drift and supply voltage dependency.

Remarks: The compensated temperature data output is placed in R2 and R3 to make it available for the next sensor measurements. The output format is described in Chapter 10.
The pressure raw data is the ADC result normalized to 12-bit resolution (right-adjusted). The value is represented as a 16-bit signed number.
Note: R1 is affected by this function.
Note: TSHEN must be cleared when this function is called.

Inputs:

R4	Function code, ref. Table 8-2
R6	Resolution, ref. Table 8-6

Outputs:

R4	Status byte, ref. 8.2. (Bits 0, 1 and 3 affected)
R2	Compensated temperature data low byte
R3	Compensated temperature data high byte
R0	Pointer to return values placed in RAM

RAM:

@R0	Compensated temperature data low byte
@R0+1	Compensated temperature data high byte
@R0+2	Temperature raw data low byte
@R0+3	Temperature raw data high byte

Registers affected: R1, R5, R6, R7

Stack depth: 12 bytes

Execution time: Execution time for this function is given by ADC resolution and µC clock frequency, ref. Table 8-7. Worst case values are used for data dependent calculation time.

Table 8-6 Contents of R6 for Meas_Temperature_Vdd

Resolution [bits]	R6
5	0000 0000
6	0000 0001
7	0000 0010
8	0000 0011
9	0000 0100
10	0000 0101
11	0000 0110
12	0000 0111

Table 8-7 Execution time for Meas_Temperature_Vdd

Resolution [bits]	Execution time [ms] given by µC clock frequency				
	2 MHz	1 MHz	500 kHz	250 kHz	125 kHz
5	1,0	1,5	2,5	4,5	8,6
6	1,0	1,5	2,5	4,5	8,6
7	1,1	1,6	2,6	4,6	8,6
8	1,2	1,7	2,7	4,7	8,7
9	1,5	2,0	3,0	5,0	8,9
10	2,1	2,6	3,5	5,5	9,4
11	3,3	3,8	4,8	6,6	10,5
12	5,9	6,4	7,3	9,2	12,9

8.4.3 MEAS_SUPPLY_VOLTAGE

Description: This function performs a supply voltage measurement with specified resolution. The result is calibrated and compensated for offset.

Remarks: The pressure raw data is the ADC result normalized to 12-bit resolution (right-adjusted). The value is represented as a 16-bit signed number. The compensated output format is described in Chapter 10.

Inputs:

R4	Function code, ref. Table 8-2
R6	Resolution, ref. Table 8-8

Outputs:

R4	Status byte, ref. 8.2. (Bits 0 and 3 affected)
R0	Pointer to return values placed in RAM

RAM:

@R0	Supply voltage low byte
@R0+1	Supply voltage high byte
@R0+2	Supply voltage raw data low byte
@R0+3	Supply voltage raw data high byte

Registers affected: R2, R3, R5, R6, R7

Stack depth: 8 bytes

Execution time: Execution time for this function is given by ADC resolution and µC clock frequency, ref. Table 8-9. Only nominal values are shown in the table.

Table 8-8 Contents of R6 for Meas_Supply_Voltage

Resolution [bits]	R6
5	0000 0000
6	0000 0001
7	0000 0010
8	0000 0011
9	0000 0100
10	0000 0101
11	0000 0110
12	0000 0111

Table 8-9 Execution time for Meas_Supply_Voltage

Resolution [bits]	Execution time [ms] given by µC clock frequency				
	2 MHz	1 MHz	500 kHz	250 kHz	125 kHz
5	0,2	0,3	0,5	1,0	1,9
6	0,2	0,3	0,5	1,0	1,9
7	0,3	0,4	0,6	1,0	1,9
8	0,4	0,5	0,7	1,1	2,0
9	0,7	0,8	1,0	1,4	2,2
10	1,4	1,4	1,6	2,0	2,7
11	2,6	2,7	2,9	3,2	3,8
12	5,2	5,3	5,4	5,7	6,3

8.4.4 START_OFFSET_VOLTAGE

Description: This function starts an offset measurement for supply voltage measurement.

Remarks: Start_Offset_Voltage is the first part of a command set consisting of three commands (Start_Offset_Voltage, Start_Supply_Voltage and Get_Supply_Voltage) that should be called in sequence to form a complete supply voltage measurement. This function starts a supply voltage offset measurement with specified resolution. It returns to the application program immediately without waiting for the A/D conversion to complete. If Power Down mode is entered before the “Start_Supply_Voltage” function is called, the results from the A/D conversion will be lost.

Inputs:

R4	Function code, ref. Table 8-2
R6	Resolution, ref. Table 8-10

Outputs:

R4	Status byte = 00h

Registers affected: R0, R5, R6

Stack depth: 6 bytes

Execution time: Execution time for this function is given by ADC resolution and µC clock frequency, ref. Table 8-11. Only nominal values are shown in the table.

Table 8-10 Contents of R6 for Start_Offset_Voltage

Resolution [bits]	R6
5	0000 0000
6	0000 0001
7	0000 0010
8	0000 0011
9	0000 0100
10	0000 0101
11	0000 0110
12	0000 0111

Table 8-11 Execution time for Start_Offset_Voltage

Resolution [bits]	Execution time [µs] given by µC clock frequency				
	2 MHz	1 MHz	500 kHz	250 kHz	125 kHz
5	35	53	90	164	312
6	35	53	90	164	312
7	35	53	90	164	312
8	51	69	106	180	328
9	83	101	138	212	360
10	147	165	202	276	424
11	275	293	330	404	552
12	531	549	586	660	808

8.4.5 START_SUPPLY_VOLTAGE

Description: This function starts a supply voltage measurement.

Remarks: Start_Supply_Voltage is the second part of a command set consisting of three commands (Start_Offset_Voltage**,** Start_Supply_Voltage and Get_Supply_Voltage) that should be called in sequence to form a complete supply voltage measurement. This function stores the offset measurement from "Start_Offset_Voltage", and starts a supply voltage measurement with specified resolution. It returns to the application program immediately without waiting for the A/D conversion to complete. If Power Down mode is entered before the "Get_Supply_Voltage" function is called, the results from the A/D conversion will be lost.

Inputs: R4 Function code, ref. Table 8-2
R6 Resolution, ref. Table 8-12

Outputs: R4 Status byte = 00h

Registers affected: R0, R5, R6

Stack depth: 6 bytes

Execution time: Execution time for this function is given by ADC resolution and µC clock frequency, ref. Table 8-13. Only nominal values are shown in the table.

Table 8-12 Contents of R6 for Start_Supply_Voltage

Resolution [bits]	R6
5	0000 0000
6	0000 0001
7	0000 0010
8	0000 0011
9	0000 0100
10	0000 0101
11	0000 0110
12	0000 0111

Table 8-13 Execution time for Start_Supply_Voltage

Resolution [bits]	Execution time [µs] given by µC clock frequency				
	2 MHz	1 MHz	500 kHz	250 kHz	125 kHz
5	41	65	114	212	408
6	41	65	114	212	408
7	41	65	114	212	408
8	57	81	130	228	424
9	89	113	162	260	456
10	153	177	226	324	520
11	281	305	354	452	648
12	537	561	610	708	904

8.4.6 GET_SUPPLY_VOLTAGE

Description: This function completes the supply voltage measurement started by the “Start_Supply_Voltage” function.

Remarks: Get_Supply_Voltage is the last part of a command set consisting of three commands (Start_Offset_Voltage, Start_Supply_Voltage and Get_Supply_Voltage) that should be called in sequence to form a complete supply voltage measurement. This function completes ADC measurement started by the “Start_Supply_Voltage”. It then calibrates the result. The output format is described in Chapter 10.

Inputs:

R4	Function code, ref. Table 8-2
R6	Resolution, ref. Table 8-14

Outputs:

R4	Status byte, ref. 8.2. (Bits 0 and 3 affected)
R0	Pointer to return values placed in RAM

RAM:

@R0	Supply voltage low byte
@R0+1	Supply voltage high byte
@R0+2	Supply voltage raw data low byte
@R0+3	Supply voltage raw data high byte

Registers affected: R2, R3, R5, R6, R7

Stack depth: 8 bytes

Execution time: 164 + 6 * (12-Res) clock cycles + 57 clock cycles @ 2MHz
(Res = ADC resolution)

Table 8-14 Contents of R6 for Get_Supply_Voltage

Resolution [bits]	R6
5	0000 0000
6	0000 0001
7	0000 0010
8	0000 0011
9	0000 0100
10	0000 0101
11	0000 0110
12	0000 0111

8.4.7 LF_INTERFACE_CONTROL

Description: This function enables/disables the LF interface.

Remarks: The LF receiver is active also when the µC is in POWER DOWN mode, ref. 3.5.1 and 5.7.
The LF circuit must be enabled before the wakeup patterns (ref. 5.5) are defined. The WU pattern registers will be reset when the LF circuit is turned off.

Inputs: R4 Function code, ref. Table 8-2
R6 Ref. Table 8-15

Outputs: R4 Status byte, ref. 8.2. (Bit 3 affected)

Registers affected: R0, R2, R3, R5, R6, R7

Stack depth: 10 bytes

Example: Ref 9.5 and 9.6

Table 8-15 Contents of R6 for LF_Interface_Control

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0		Note
0	0	0	X	X	0	X	X	LF interface off	
1	1	0	X	X	0	X	X	Unmodulated carrier detection mode (always channel 1)	
1	1	0	E2	E1	1	X	X	Modulated carrier detection mode	1
1	0	0	E2	E1	0	0	0	LF telegram mode, long WU word 2 and 1	1
1	0	0	E2	E1	0	0	1	LF telegram mode, long WU word 2, short WU word 1	1
1	0	0	E2	E1	0	1	0	LF telegram mode, short WU word 2, long WU word 1	1
1	0	0	E2	E1	0	1	1	LF telegram mode, short WU word 2 and 1	1, 2

Notes:

1. LF channels are enabled/disabled independently:
   E2 = ‘0’: Channel 2 disabled, E2 = ‘1’: Channel 2 enabled
   E1 = ‘0’: Channel 1 disabled, E1 = ‘1’: Channel 1 enabled
2. When short LF wakeup is selected, the most significant ‘0’ in the wakeup pattern defines the length of the wakeup ID.

8.4.8 LF_DATA_READ

Description: This function reads and stores the data received by the LF interface.

Remarks: This function may be applied when the LF Telegram Interface is turned on and a valid wakeup ID has been defined.
Maximum number of bytes to read is given by R6 (R6 = 0 not allowed).
The function will return when End of Protocol is detected, or number of read bytes equals maximum.

Inputs:	R4	Function code, ref. Table 8-2
	R0	Pointer to start of data space
	R6	Maximum number of bytes to read
Outputs:	R4	Status byte = 00h
	R0	Pointer to first received byte (= R0 input)
	R3	Number of received bytes

Registers affected: R5

Execution time: N/A (dependent of LF bit rate)

Stack depth: 4 bytes

Example: Ref 9.5

8.4.9 LF_INTERFACE_ON_OFF

Description: This function enables/disables the LF interface and performs a calibration of LF carrier detection threshold.

Remarks: The LF receiver is active also when the µC is in POWER DOWN mode.
The LF circuit must be enabled before the wakeup patterns are defined. The WU pattern registers will be reset when the LF circuit is turned off.
This function calls the library function LF_Interface_Control as a sub-routine after calibration.

Inputs:

R3	Temperature data high byte
R4	Function code, ref. Table 8-2
R6	Ref. Table 8-16

Outputs: R4 Status byte, ref. 8.2. (Bit 3 affected)

Registers affected: R0, R2, R3, R5, R6, R7

Stack depth: 10 bytes

Table 8-16 Contents of R6 for LF_Interface_On_Off

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0		Note
0	0	0	X	X	0	X	X	LF interface off	
1	1	0	X	X	0	X	X	Unmodulated carrier detection mode (always channel 1)	
1	1	0	E2	E1	1	X	X	Modulated carrier detection mode	1
1	0	0	E2	E1	0	0	0	LF telegram mode, long WU word 2 and 1	1
1	0	0	E2	E1	0	0	1	LF telegram mode, long WU word 2, short WU word 1	1
1	0	0	E2	E1	0	1	0	LF telegram mode, short WU word 2, long WU word 1	1
1	0	0	E2	E1	0	1	1	LF telegram mode, short WU word 2 and 1	1, 2

Notes:

1. LF channels are enabled/disabled independently:
   E2 = ‘0’: Channel 2 disabled, E2 = ‘1’: Channel 2 enabled
   E1 = ‘0’: Channel 1 disabled, E1 = ‘1’: Channel 1 enabled
2. When short LF wakeup is selected, the most significant ‘0’ in the wakeup pattern defines the length of the wakeup ID.

8.4.10 THERMAL_SHUTDOWN_ENABLE

Description: This function enables thermal shutdown if the temperature is above the specified threshold.

Remarks: This function should be called when the measured temperature is approaching the specified maximum operating temperature. The returned Status byte indicates whether it is possible to go to Thermal Shutdown. See also Figure 8-3.

Inputs: R4 Function code, ref. Table 8-2

Outputs: R4 Status byte, ref. 8.2. (Bit 3 affected)

Registers affected: R0, R2, R3, R5, R6, R7

Stack depth: 8 bytes

Execution time: 94 clock cycles + 33 clock cycles @ 2 MHz + 56 clock cycles @ 500 kHz

Example: Use of this function is shown in 9.9

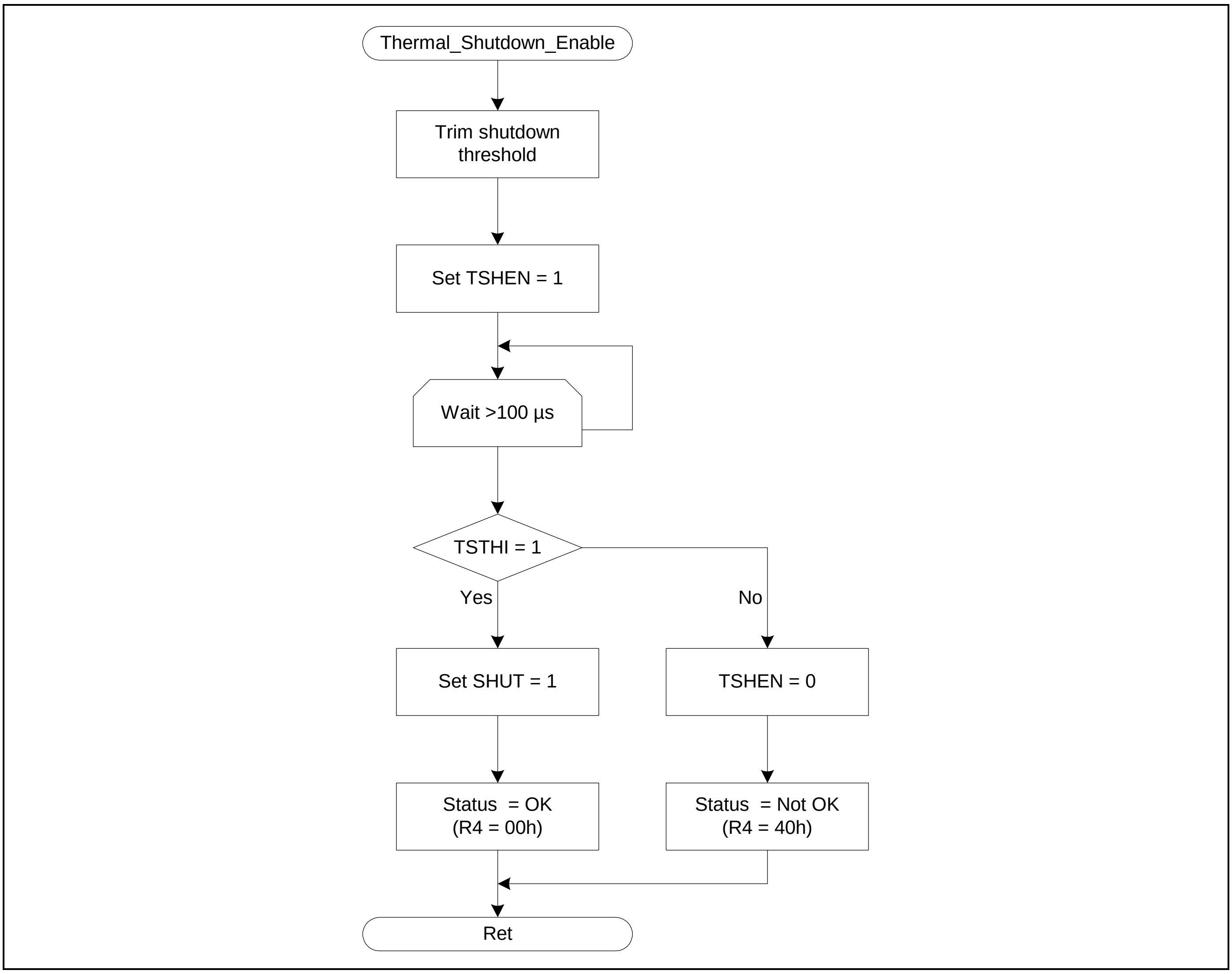

Figure 8-3 Flowchart for Thermal_Shutdown_Enable

8.4.11 MULTIPLY_SIGNED_16

Description:	This function performs a 16 x 16 bit signed multiplication	
Remarks:	This routine is invalid for Multiplicand = -32768	
Inputs:	R4	Function code, ref. Table 8-2
	R0	Pointer to input values placed in RAM
	@R0	Multiplicator LSB
	@R0+1	Multiplicator MSB
	@R0+2	Multiplicand LSB
	@R0+3	Multiplicand MSB
Outputs:	R4	Status byte = 00h
	R0	Pointer to multiplication result placed in RAM
RAM:	@R0	Multiplication result Byte 0 (LSB)
	@R0+1	Multiplication result Byte 1
	@R0+2	Multiplication result Byte 2
	@R0+3	Multiplication result Byte 3 (MSB)
Registers affected:	R2, R3, R5, R6, R7	
Stack depth:	6 bytes	
Execution time:	Min 204 clock cycles (0000h x 0000h) Max 284 clock cycles (FFFFh x FFFFh)	

8.4.12 GET_LIBRARY_REV

Description:	This function returns a constant identifying the revision status of the ROM Library	
Remarks:	The revision number is an 8-bit value incremented for each mask-ROM release	
Inputs:	R4	Function code, ref. Table 8-2
Outputs:	R4	Status byte = 00h
	R3	Library revision (ref. 8.5)
Registers affected:	R5	
Stack depth:	4 bytes	
Execution time:	11 clock cycles	

8.4.13 CRC8_CALC

Description: This function calculates the CRC-8 value of data pointed to by R0 in RAM using a programmable polynomial. The routine can check any number of bits to allow for optimized length of transmission data. The CRC-8 is calculated using a long division without carry propagation. The returned 8-bit checksum (R6) is the remainder of this long division. The 9-bit divisor (polynomial) may be specified freely, except the MSB must always be 1. The remaining 8 LSB is specified in R7.

Remarks: Typically used for data transmission validation

Inputs:

R4	Function code, ref. Table 8-2
R0	Pointer to start of data set
R2	Total number of bits for computation (Low byte)
R3	Total number of bits for computation (High byte)
R6	Running CRC-hash value (00h for new calculation)
R7	Polynomial = $1 \cdot x^8 + a_7 \cdot x^7 + a_6 \cdot x^6 + a_5 \cdot x^5 + a_4 \cdot x^4 + a_3 \cdot x^3 + a_2 \cdot x^2 + a_1 \cdot x^1 + a_0 \cdot x^0$

Outputs:

R4	Status byte = 00h
R6	CRC-8 result

Registers affected: R2, R3, R5

Stack depth: 4 bytes

Execution time: 16 + 7 * number_of_bits + (2 x Value in MSB part of counter) clock cycles

Example: A frequently used polynomial is g(X) = x^8 + x^2 + x + 1. This can be represented as 100000111b or 107h. Since bit 8 is fixed to ‘1’, R7 should be set to 00000111b (=07h) for this polynomial.

8.4.14 SIGNATURE_TEST

Description: This function performs a signature test of the specified memory and returns a 3-byte signature.

Remarks: The function does not compare the result to a stored signature. This must be performed by the application program.
See also 11.2.3 (Errata sheet).

Inputs:	R4	Function code, ref. Table 8-2
	R3	Defines which memory to test (00h for ROM, 01h for EROM, 02h for EEPROM)
Outputs:	R4	Status byte = 00h
	R3	Signature byte 0 (compare to EEPROM page 24, byte 1)
	R6	Signature byte 1 (compare to EEPROM page 24, byte 2)
	R7	Signature byte 2 (compare to EEPROM page 24, byte 3)

Registers affected: R0, R2, R5

Stack depth: 8 bytes

Algorithm: The algorithm uses three registers during calculation (24-stage MISR). For each two data bytes the algorithm is executed. The value of the result registers must not be changed during signature calculation. After finishing the calculation the result is stored in the result registers.

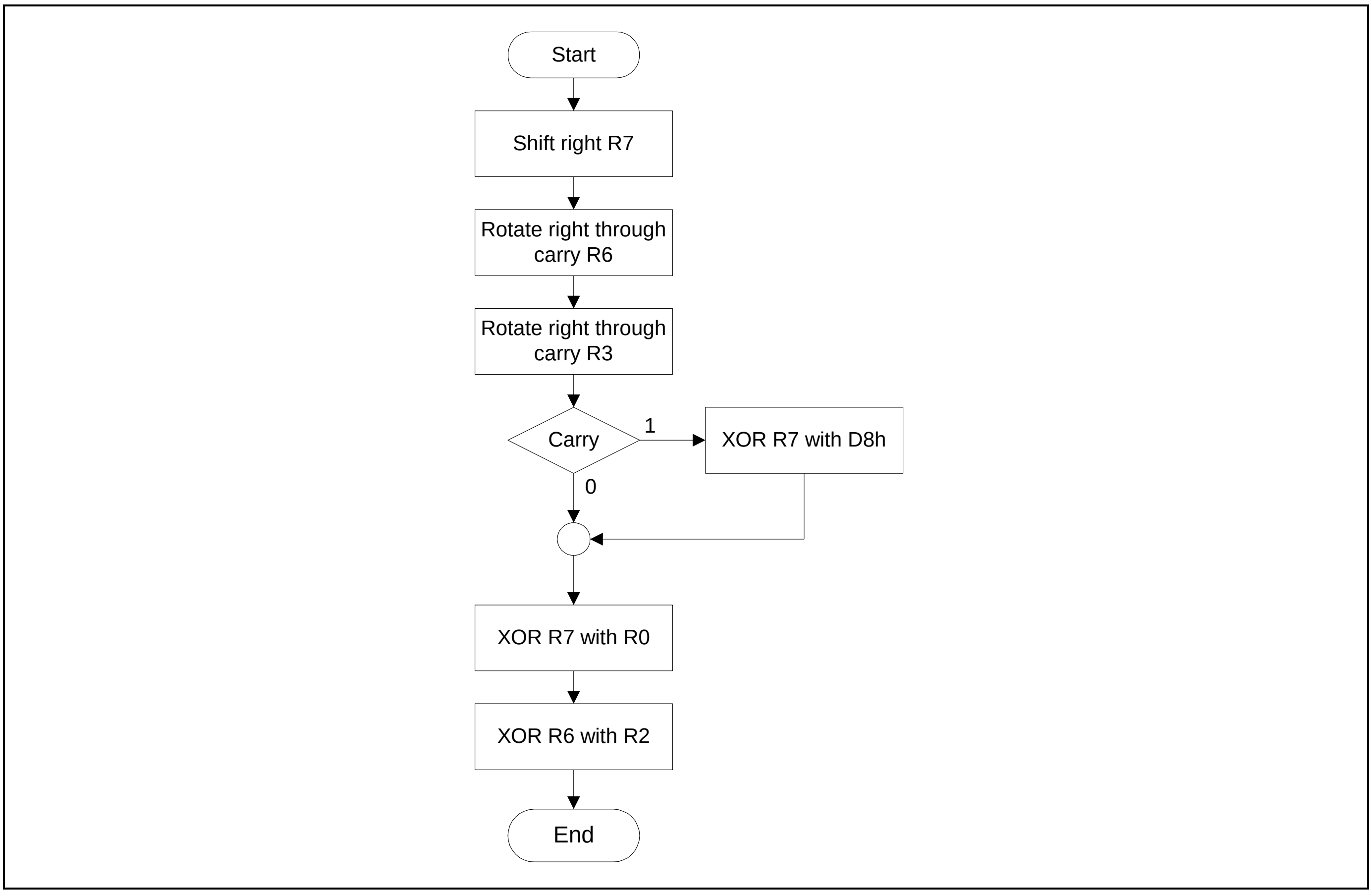

Figure 8-4 Flowchart for Signature_Test

8.4.15 READ_ID

Description:	This function reads the 4-byte unique serial number stored in the EEPROM (Device Specific Memory).	
Remarks:	None	
Inputs:	R4	Function code, ref. Table 8-2
Outputs:	R4	Status byte, ref. 8.2. (Bit 3 affected)
	R0	Pointer to ID placed in RAM
RAM:	@R0	ID0
	@R0+1	ID1
	@R0+2	ID2
	@R0+3	ID3
Registers affected:	R2, R3, R5, R6, R7	
Stack depth:	8 bytes	
Execution time:	56 clock cycles + 201 clock cycles @ 2MHz	

8.4.16 IDLE

Description: This function enters IDLE mode by executing the instruction “setb IDLE” and returns after an enabled interrupt has occurred. Since the EROM is disabled during this time, the device power consumption is minimized.

Remarks: If no wakeup is generated, the function will never return. In that case, the Watchdog Timer will force the device into POWER DOWN.

Input: R4 Function code, ref. Table 8-2

Output: None

Registers affected: R4, R5

Stack depth: 4 bytes

Execution time: 9 cycles until entering IDLE mode, plus 5 cycles after IDLE.

Example: The following sequence waits in low-power IDLE mode until one of the wakeup ports has detected a valid transition.

```
clr     IE                  ; disable all interrupts
setb    EP                  ; enable Port Wakeup
clrb    FP                  ; clear corresponding interrupt flip-flop
mov     R4,#7Ch
sys                         ; call IDLE
clrb    FP                  ; acknowledge wakeup
```

8.4.17 EE_READ

Description: Reads a number of bits from the EEPROM. The function can be configured to start the read operation with a certain bit offset within the addressed EEPROM page. The bits are stored MSB first in RAM.

Remarks: If the RISC is clocked with 2MHz when entering this function, *EE_READ* temporarily switches to 1MHz to meet the EEPROM read access time specification. The initial RISC clock setting is restored when leaving *EE_READ*.

Input: R0 = Pointer to RAM location were the read bits shall be stored
R4 = Function code, ref. Table 8-2
R7 = EEPROM page number (0...31). Bit 5...7 is ignored (masked out).
R6 = Bit offset in EEPROM page (0...31). Bits 5...7 are ignored (masked out).
R3 = Number of bits to read (1…256). To read 256 bits, specify 00h here.

Output: R0 = Pointer to RAM location were the read bits are stored.

Registers affected: R3, R4, R5, R6

Stack depth: 4 bytes

Execution cycles: 35 + number_of_bits_to_read * 4 (if system clock is less than 2MHz)
37 + number_of_bits_to_read * 8 (if system clock is 2MHz)

Example: The EEPROM holds the value 01020304h (00000001-00000010-00000011-00000100b) in page 8. After execution of the command sequence,

```
mov     R0,#RAM_addr
mov     R7,#8               ; EEPROM page address
mov     R6,#16              ; EEPROM bit offset
mov     R3,#16              ; number of bits to read
mov     R4,#48h
sys                         ; call EE_READ
```

The value 03h is stored at the symbolic RAM address *RAM_addr*, and the value 04h is stored at *RAM_addr* +1.

8.4.18 EE_READ_W_PARITY

Description: Reads a number of bits from the EEPROM. The function can be configured to start the read operation with a certain bit offset within the addressed EEPROM page. The bits are stored MSB first in RAM.
R5 will contain a count of ‘1’-s in the data set read. By looking at R5 bit <0> parity information can be extracted. Alternatively the number of ‘1’-s can be counted, to detect multiple bit errors.

Remarks: If the RISC is clocked with frequency < 2MHz when entering this function, *EE_READ_W_PARITY* temporarily switches to 2 MHz. The initial RISC clock setting is restored when leaving *EE_READ_W_PARITY*.
Bit <0> of R5 returns parity information

Input: R0 = Pointer to RAM location were the read bits shall be stored.
R4 = Function code, ref. Table 8-2
R7 = EEPROM page number (0...31). Bit 5...7 is ignored (masked out).
R6 = Bit offset in EEPROM page (0...31). Bits 5...7 are ignored (masked out).
R3 = Number of bits to read (1…256). To read 256 bits, specify 00h here.

Output: R0 = Pointer to RAM location were the read bits are stored.
R5 = Number of ‘1’-s read

Registers affected: R3, R4, R6

Stack depth: 4 bytes

Execution cycles: 3 + number_of_bits_to_read * 6 @ 2 MHz clock
+ 19 @ selected application clock speed

Example: The EEPROM holds the value 01020304h (00000001-00000010-00000011-00000100b) in page 8. After execution of the command sequence,

```
mov     R0,#RAM_addr
mov     R7,#8                       ; EEPROM page address
mov     R6,#16                      ; EEPROM bit offset
mov     R3,#16                      ; number of bits to read (including
                                    ; parity bit)
mov     R4,#46h
sys                                 ; call EE_READ_W_PARITY
sbc     R5.0                        ; Test if equal parity
call    Parity_Error_Routine        ; Take Parity error action
:
(Application)
:
```

The value 03h is stored at the symbolic RAM address *RAM_addr*, and the value 04h is stored at *RAM_addr* +1. Since the number of ones read in the selected data set is odd number, the Parity error routine will be called.

8.4.19 EE_WRITE_EXT

Description: With this function, a number of data bits can be programmed into a single EEPROM page.

Remarks: Due to design limitations, the function may not be used to write to the same bit location(s) in the EEPROM page buffer more than once. The result, which would actually be written to the EEPROM, would be undefined. The EEPROM page buffer operates cyclically and restarts at bit position 0 within the same page if the page boundary is crossed. It follows that,

- The number of bits to write (R3) shall be limited to 32 by the application.
- If the upper page border is crossed during writing (R6 + R3 > 32), the write pointer wraps around and starts all over at the beginning of the EEPROM page to write the remaining bits.

Before this function is called, the application shall ensure that no other sources will generate any wakeup(s) from IDLE mode. To ensure this, it is recommended to always enter this function with all interrupt sources disabled (the IE register initialized to 00h). The function modifies only the interrupt source(s) it uses, and disables them before returning.

Input: R0 = Pointer to RAM location containing the bits to program
R4 = Function code, ref. Table 8-2
R7 = EEPROM page number (0...31). Bit 5...7 is ignored (masked out)
R6 = Bit offset in EEPROM page (0...31). Bits 5...7 are ignored (masked out)
R3 = Number of bits to write (1...32)

Output: R4 = 00h if programming successful
R4 = 01h if programming failed. Page write-protected, or programming voltage failure (PERR=1)

Registers affected: R0, R3, R5, R6

Stack depth: 6 bytes

Execution time: (43 + 4 * number_of_bits_to_write) @ Tsys + 394 clock cycles @ 125 kHz

Example: The RAM content starting at the symbolic address *RAM_addr* equals 12h, 34h, 56h, 78h. After successful execution of the following command sequence (label *ee_write_ok* is reached),

```
mov     R0,#RAM_addr
mov     R7,#8               ; EEPROM page address
mov     R6,#16              ; EEPROM bit offset
mov     R3,#32              ; number of bits to program
mov     R4,#52h
sys                         ; call EE_WRITE_EXT
sz      R4
jmp     ee_write_error      ; programming failed / page wr. Protected
ee_write_ok:
…
ee_write_error:
…
```

Page 8 of the EEPROM holds the value 56781234h (01010110-01111000-00010010-00110100b). Note that the write cycle started with an offset of 16 bits and forced the page pointer to wrap around at the upper page border.

8.5 LIBRARY REVISIONS

There are two different library revisions available. Apart from corrected errors (ref. Chapter 11), there are two different boot sequences:

Type A: P10 and P11 pins are not defined during boot sequence

Type B: P10 and P11 pins are defined to pull-up configuration during boot sequence

Table 8-17 shows which boot sequence that is used for which library revision.

Table 8-17 Boot sequences related to different library versions

HW rev + Library rev / Ref.	V1A		V1B		V1C	
	02h	04h	02h	04h	02h	04h
Type A	X		X		X	
Type B		X		X		X

9 PROGRAMMER’S GUIDE

9.1 PIN CONNECTION

Pin-out is described in 2.2. Table 9-1 shows how these pins could be connected in a typical application.

Table 9-1 Pin connection for typical applications

Pin	Connection
IN1, IN2, IN3, IN4	LF receiver inputs. If not used, these pins should be connected to VSS, or left unconnected. In the latter case, the corresponding channel should be disabled by SW.
P10, P11	General purpose I/O with external wakeup, internal pull-up/pull-down. If not used, these pins should be left unconnected. For definitions during boot, refer to 8.5.
P14, P15, P17	General purpose I/O. If not used, these pins should be connected to VSS. P15 may be connected to an external RF transmitter.
MSDA	Monitor Serial Data I/O. Used during debug and download. Should be left unconnected in application, or connected to VDD.
MSCL	Monitor Serial Clock output. Used during debug and download. Should be left unconnected in application.

9.2 STARTUP ACTIONS

After any device wakeup the program execution starts with the ROM based boot routine. Prior to passing control to the program memory reset vector, the boot routine executes a sequence of instructions that calibrate the oscillators, LF bias and VMIN circuits of the device. A consistency check of the corresponding EEPROM data is performed. The result of this test can be checked by the application program by reading R4, bit 7. ‘1’ means OK, ‘0’ means that the calibration data is not reliable.

After the boot sequence, the reset conditions of SFR (ref. 2.4) apply.

Some registers are not defined by power-on reset or when going from Power Down mode to Run mode. The following sections suggest precautions to be taken by the application program.

9.2.1 STATUS REGISTER

LTOV, TSH and ITOV (ref. 5.2) should be checked after each boot sequence, and then cleared.

9.2.2 CONTROL REGISTER

TSHEN and LFEN (ref. 5.3) should be cleared after the first power-up reset.

9.2.3 TIMER REGISTER

All bits except bit 2 of the Timer register (ref. 5.7) should be defined after the first power-up reset.

9.3 FLAGS AND CONTROL BITS

9.3.1 ITOV

The Interval Timer Overflow flag ITOV (ref. Table 5-5) is set when the Interval Timer overflows. The flag may be tested by the application program, in order to identify the wakeup or interrupt sources. Since the flag is not reset, it should be cleared by the application program prior to forcing the device into POWER DOWN mode in order to serve its identification function.

9.3.2 LFWUP

Device wakeup from POWER DOWN mode may be generated by means of an LF Header received and detected by the LF Interface. However, this requires that the LF Interface has been enabled by the application program prior to forcing SP300 into POWER DOWN mode.

In this case, the LF WUP flags WUP1 and WUP2 (ref. Table 5-5) are set and may be tested by the application program as desired, in order to identify the wakeup source. Since the flags are not reset by hardware, they shall be cleared by the application program prior to forcing the device into POWER DOWN mode in order to serve their identification function.

9.3.3 SCEN

The SCEN bit (ref. Table 4-12) should be cleared when the modulator is not used, in order to minimize power consumption.

9.4 STATUS BYTE

Most library functions return a status byte in R4 (ref. 8.2). This status byte is cleared at the beginning of the library function. Only relevant bits are updated by the function; all other bits are kept at zero. Thus, the application program should check the status byte after each function call (where relevant).

9.5 LF TELEGRAM ON

The following code example shows how to turn on channel 1 in telegram mode with 8 bits LF wakeup ID 1, R6=85h.

```
; LF Interface on

      mov R4, #2Eh
      mov R6, #10001001b                       ; LF enable channel 1 short WUL on WUP1

      sys                                      ; SYS call to LF_Interface_Control

; Set wakeup pattern

      mov R0, #W1_0
      mov RICAR, R0
      mov R0, #0Ach
      mov RIDR, R0                             ; wup1 byte 0 = AC

      mov R0, #W1_1
      mov RICAR, R0
      mov R0, #0Feh
      mov RIDR, R0                             ; wup1 byte 1 bit 0 is most significant zero

      ; Read LF data

      mov R4, #1Eh
      mov R0, #Data_IO_Area                    ; Set data area for LF-data
      mov R6, #5
      sys                                      ; SYS call to LF_Data_Read
```

Figure 9-1 Code example: LF Interface on

9.6 LF TELEGRAM OFF

The following code example shows how to turn off LF reception.

```
; LF Interface off

      mov    R4,# 2Eh
      clr    R6                                ; Turn off LF interface
      sys                                      ; SYS call to LF_Interface_Control
```

Figure 9-2 Code example: LF Interface off

9.7 LF CARRIER DETECTION

The LF_Interface_On_Off function (ref. 8.4.9) enables/disables the LF interface and performs a calibration of LF carrier detection threshold. The calibration setting is kept also during Power Down mode. To reduce current consumption, this function could be called only the first time LF Carrier Detection is used, and then the LF_Interface_Control function (8.4.7) could be used.

9.8 TIMING RELATED TO LF

Maximum time from LF Wakeup to calling LF data read is shown in Figure 9-3 (not to scale).

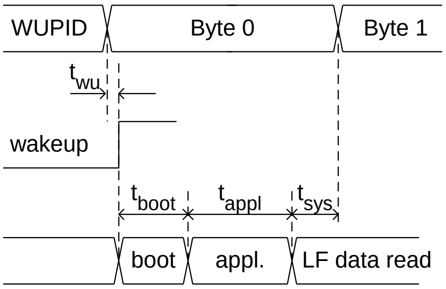

Figure 9-3 ROM library access mechanism

Time from LF wakeup ID received to wakeup, t_{wu}, is approximately 10 µs. Then the boot sequence needs approximately t_{boot} = 530 µs (worst case). The “LF_Data_Read” function must be called minimum t_{sys} = 5 µs (at 2 MHz clock) before Byte 1 appears at the input. This means that the time from the application program is invoked to “LF_Data_Read” must be called, is maximum t_{appl} = 1500 µs (at 3.9 kbit/s data rate).

9.9 THERMAL SHUTDOWN

Figure 9-4 shows the use of the Thermal_Shutdown_Enable function (ref. 8.4.10).

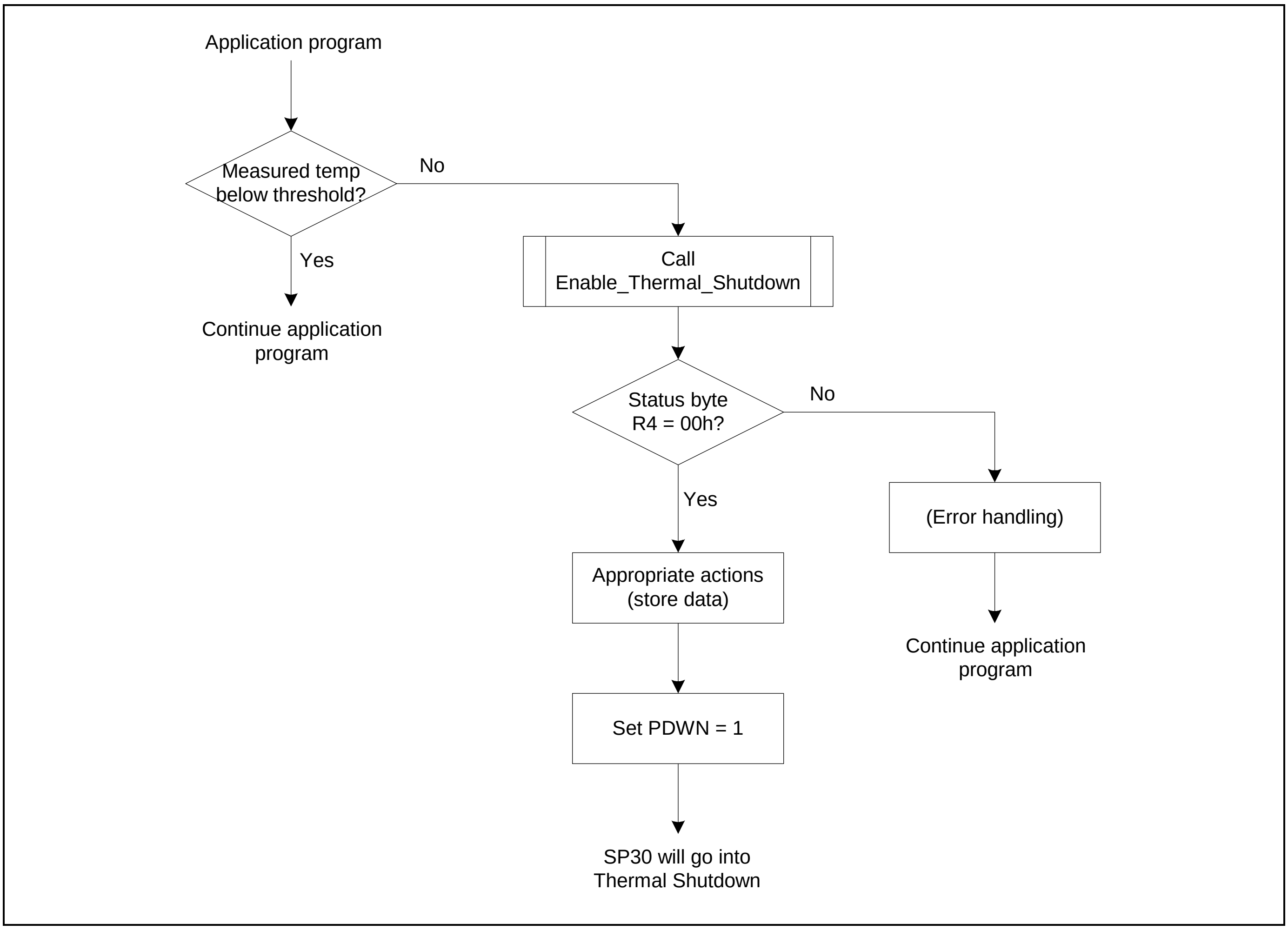
Application program
Measured temp below threshold?
No
Yes
Continue application program
Call Enable_Thermal_Shutdown
Status byte R4 = 00h?
No
Yes
(Error handling)
Continue application program
Appropriate actions (store data)
Set PDWN = 1
SP30 will go into Thermal Shutdown

Figure 9-4 Thermal shutdown

In addition, the application program may check the TSST flag before setting the PDWN bit, just to be sure that the device will go into Thermal Shutdown mode.

The TSH flag can be tested by the application program in order to identify if there has been a wakeup from Thermal Shutdown. In that case, the application program should clear the TSHEN bit.

9.10 THREE-STEP SUPPLY VOLTAGE MEASUREMENT

Normally, supply voltage measurement is performed by the “Meas_Supply_Voltage” function. A possible application could want to measure supply voltage during bit transmission. Since the “Meas_Supply_Voltage” function could be too time-consuming to be run within transmission of one bit, a three-step solution could be used:

1. “Start_Offset_Voltage” sets up the ADC and starts a conversion of the ADC offset voltage. It then returns the control to the application program.
2. “Start_Supply_Voltage” ensures that the offset measurement is finished before storing the offset value to reserved RAM area. It then starts a supply voltage conversion before returning to the application program.
3. “Get_Supply_Voltage” ensures that the ADC measurement is finished before turning off the ADC. It then offset compensate the ADC result before calculating the compensated supply voltage value.

The first two steps should be called at as equal conditions as possible, e.g. during transmission of a ‘1’-bit. Then the measurement could be completed by calling the “Get_Supply_Voltage” function after bit-transmission. The result will then show the supply voltage that was present during bit transmission.

As mentioned above, the two first functions will start an A/D conversion and return before this is completed. In order to reduce current consumption, the application program may turn off the ADC at the right time. An example of this is shown in Figure 9-5 and Figure 9-6.

Note: The application program should not reset the ADC ready interrupt flag. The flag should only be “hidden” by turning off the EADC bit (ADC Ready interrupt enable).

```
; Start supply voltage offset conversion
        mov    R6,# ADC_RESOLUTION
        mov    R4,# INS_START_OFFS_VOLT        ; Call “Start_Supply_Offset”
        sys
        :
        (Application)
        :

; Wait for the ADC to complete the offset voltage conversion (1)
        ADC_Wait_1:
        mov    R4,# INS_IDLE (07Ch)            ; Use IDLE to reduce current consumption
        sys

; Check the idle wakeup source
        :
        (Application)                         ; Service other wakeup sources (e.g. timer)
        :
        sbs    FADC                           ; Check if ADC is the wakeup source
        jmp    ADC_Wait_1                     ; ADC not finished, wait!

; Turn off the ADC power and interrupt source (2)
        clrb   ADCEN                          ; Turn off ADC
        clrb   EADC                           ; Turn off ADC ready interrupt enable bit
        :
        (Application)
        :

; Start supply voltage conversion
        mov    R6,# ADC_RESOLUTION
        mov    R4,# INS_START_SUPPL_VOLT       ; Call “Start_Supply_Voltage”
        sys
        :
        (Application)
        :

; Wait for the ADC to complete the supply voltage conversion (3)
```

```
        ADC_Wait_3:
        mov    R4,# INS_IDLE (07Ch)            ; Use IDLE to reduce current consumption
        sys

; Check the idle wakeup source
        :
        (Application)                          ; Service other wakeup sources (e.g. timer)
        :
        sbs    FADC                            ; Check if ADC is the wakeup source
        jmp    ADC_Wait_3                      ; ADC not finished, wait!

; Turn off the ADC power and interrupt source (4)
        clrb   ADCEN                           ; Turn off ADC
        clrb   EADC                            ; Turn off ADC ready interrupt enable bit
        :
        (Application)
        :

; Get data and complete calculations
        mov    R6,# ADC_RESOLUTION
        mov    R4,# INS_GET_SUPPL_VOLT         ; "Call Get_Supply_Voltage"
        sys
```

Figure 9-5 Code example: Three-step supply voltage measurement

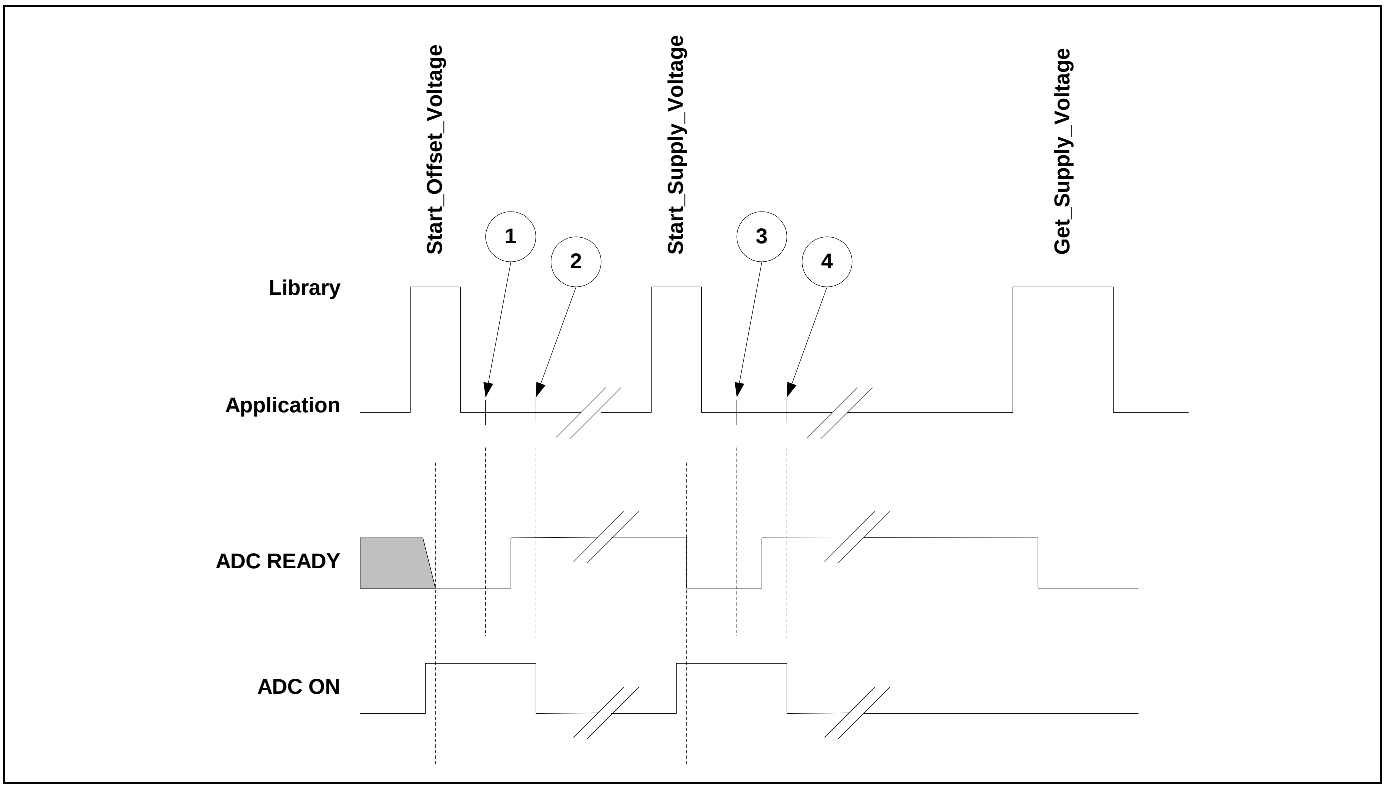

Figure 9-6 Timing of three-step supply voltage measurement

Numbers in parentheses in Figure 9-5 corresponds to events indicated by numbers in circles in Figure 9-6.

10 MEASUREMENT OUTPUT

10.1 SUPPLY VOLTAGE

Voltage measurements are performed by the function Meas_Supply_Voltage (ref. 8.4.3) or a sequence of the three functions Start_Offset_Voltage, Start_Supply_Voltage and Get_Supply_Voltage (ref. 8.4.4 to 8.4.6). Measured voltage [V] is found from the two-byte output V as

$$V_{out} = C_v - V \cdot R_v \cdot 2^{-8}$$

where

V_{out}	= Measured supply voltage	[V]
V	= Digital compensated supply voltage	[LSB]
R_v	= $10.8 \cdot 10^{-3}$	[V/LSB]
C_v	= 4.0	[V]

Under normal conditions (supply voltage within spec), the digital compensated supply voltage will not saturate and the overflow/underflow flag will not be set. Quite theoretically, saturation and overflow/underflow is given at the following conditions:

$V_{out} > 4.0$ V	=>	V is saturated at 0 (=00000h)	Overflow/underflow flag is set
$V_{out} < 1.235$ V	=>	V is saturated at 2^{16}-1 (=FFFFh)	Overflow/underflow flag is set

10.2 TEMPERATURE

Temperature measurements are performed by the function Meas_Temperature_Vdd (ref. 8.4.2). Measured temperature [°C] is found from the two-byte output T as

$$T_{out} = T \cdot 2^{-8} + C_T$$

where

T_{out}	= Measured temperature	[°C]
T	= Digital compensated temperature	[LSB]
C_T	= -50	[°C]

Under normal conditions (applied temperature within spec), the digital compensated temperature will not saturate and the overflow/underflow flag will not be set. Quite theoretically, saturation and overflow/underflow is given at the following conditions:

$T_{out} < -50$ °C	=>	V is saturated at 0 (=00000h)	Overflow/underflow flag is set
$T_{out} \geq 206$ °C	=>	V is saturated at 2^{16}-1 (=FFFFh)	Overflow/underflow flag is set

10.3 PRESSURE

Pressure measurements are performed by the function Meas_Sensor (ref. 8.4.1). Measured pressure [kPa] is found from the two-byte output P as

$$p_{out} = R_p \cdot P$$

where

p_{out}	= Measured pressure	[kPa]
P	= Digital compensated pressure	[LSB]
R_p	= Constant decided by pressure range, ref. Table 10-1	[kPa/LSB]

Under normal conditions (applied pressure within spec), the digital compensated pressure will not saturate and the overflow/underflow flag will not be set. Depending on the sensor configuration, the output is saturated as described in Table 10-1.

Table 10-1 Output resolution and saturation limits for pressure measurements

Product	R_p	P saturated to 2^{15}-1 (=7FFFh)
SP300-0.5(T)	0.15*2^{-6} kPa/lsb	≥76.80 kPa
SP300-1(T)	0.25*2^{-6} kPa/lsb	≥128.0 kPa
SP300-2(T)	0.50*2^{-6} kPa/lsb	≥256.0 kPa
SP300-7(T)	4.00*2^{-6} kPa/lsb	≥2048 kPa
SP300-7A(T)	4.00*2^{-6} kPa/lsb	≥2048 kPa
SP300-16A	4.00*2^{-6} kPa/lsb	≥2048 kPa

Note: All pressure values are given as absolute pressure.

10.3.1 OUTPUT FORMAT CONVERSION

Meas_Sensor (ref. 8.4.1) generates a 16-bit signed output, as shown above. The number format covers both negative and positive pressure, even though negative pressure is not applicable in real applications. For some applications, it could be desirable to convert the 16-bit signed output to an 8-bit unsigned output, covering only the specified pressure range. This can be performed with a first order transfer function:

$$F_p = \left(N_p - \frac{Pmin}{R_p}\right) \cdot \frac{R_p \cdot (Fmax)}{(Pmax - Pmin)}$$

where

N_p	= Normal 16-bit signed output	[LSB]
F_p	= Formatted 8-bit unsigned output	[LSB]
Fmax	= Number to represent maximum input pressure	[LSB]
Pmin	= Minimum input pressure	[kPa]
Pmax	= Maximum input pressure	[kPa]

Minimum input pressure is represented as 00h, while maximum input pressure is represented as either FFh or 100h.

<u>Example SP300-7(T):</u>

If the wanted number format for 700 kPa is that 100 kPa corresponds to 00h output while 700 kPa corresponds to 100h output, then the pressure output should be calculated by the following formula:

$$F_p = \left(N_p - \frac{100}{4 \cdot 2^{-6}}\right) \cdot \frac{4 \cdot 2^{-6} \cdot (256)}{(700 - 100)}$$

$$F_p = \left(N_p - 1600\right) \cdot 1748 \cdot 2^{-16}$$

This can be implemented with add + multiply + shift operations.

10.4 ACCELERATION

Acceleration measurements are performed by the function Meas_Sensor (ref. 8.4.1). Measured acceleration [g] is found from the two-byte output A as

$a_{out} = R_a \cdot A$

where

a_{out}	= Measured acceleration	[g]
A	= Digital compensated acceleration	[LSB]
R_a	= Constant decided by acceleration range, ref. Table 10-2	[g/LSB]

Under normal conditions (applied acceleration within spec), the digital compensated acceleration will not saturate and the overflow/underflow flag will not be set. Depending on the sensor configuration, the output is saturated as described in Table 10-2.

Table 10-2 Output resolution and saturation limits for acceleration measurements

Product	Acc. Range	R_a	A saturated to 2^{15}-1 (=7FFFh)	A saturated to -2^{15} (=8000h)
SP300-0.5(T)	X	X	X	X
SP300-1(T)	X	X	X	X
SP300-2(T)	X	X	X	X
SP300-7(T)	X	X	X	X
SP300-7A(T)	-12 to 115 g	$0.50*2^{-6}$	≥256 g	< -256 g
SP300-16A	-12 to 115 g	$0.50*2^{-6}$	≥256 g	< -256 g

10.4.1 OUTPUT FORMAT CONVERSION

Meas_Sensor (ref. 8.4.1) generates a 16-bit signed output, as shown above. The number format covers both negative and positive acceleration, even though negative acceleration may not be applicable in real applications. For some applications, it could be desirable to convert the 16-bit signed output to an 8-bit unsigned output, covering only the specified acceleration range. This can be performed with a first order transfer function:

$$F_a = \left(N_a - \frac{\mathrm{Amin}}{R_a} \right) \cdot \frac{R_a \cdot (\mathrm{Fmax})}{(\mathrm{Amax} - \mathrm{Amin})}$$

where

N_a	= Normal 16-bit signed output	[LSB]
F_a	= Formatted 8-bit unsigned output	[LSB]
Fmax	= Number to represent maximum input acceleration	[LSB]
Amin	= Minimum input acceleration	[g]
Amax	= Maximum input acceleration	[g]

Minimum input acceleration is represented as 00h, while maximum input acceleration is represented as either FFh or 100h.

Example –12 to 116 g:

If the wanted number format is that –12 g corresponds to 00h output while 116 g corresponds to 100h output, then the acceleration output should be calculated by the following formula:

$$F_a = \left[N_a - \frac{(-12)}{2^{-7}} \right] \cdot \frac{2^{-7} \cdot (256)}{(116 - (-12))}$$

$$F_a = (N_a + 1536) \cdot 2^{-6}$$

This can be implemented with add + shift operations.

11 ERRATA SHEET

11.1 OVERVIEW

Table 11-1 shows an overview of errors in different library revisions and HW versions. The errors are explained in 11.2.

HW revisions are V1A, V1B and V1C. Library revisions are further described in 8.5.

Table 11-1 Errata for different HW versions and library revisions

HW rev + Library rev / Ref.	V1A		V1B		V1C	
	02h	04h	02h	04h	02h	04h
11.2.1 Temp measurement	X	OK	X	OK	X	OK
11.2.2 ADC overflow	X	OK	X	OK	X	OK
11.2.3 Signature test	X	OK	X	OK	X	OK
11.2.4 Parity, temp	X	X	X	X	X	X
11.2.5 Parity, LF	X	X	X	X	X	X
11.2.6 Lo-temp error (HW)	X	X	OK	OK	OK	OK
11.2.7 IDDQ error (HW)	X	X	X	X	OK	OK

X:	This error is present in this version
OK:	This error is corrected in this version
NA:	Not applicable

11.2 KNOWN ERRORS

11.2.1 TEMPERATURE MEASUREMENT

For all library functions, R1 should be untouched by the library function. One exception is the “Meas_Temperature_Vdd” function (ref. 8.4.2).

11.2.2 ADC OVERFLOW

This is not really an error, only a difference between different library revisions. For the versions marked with “X” in Table 11-1, overflow/underflow in compensation is flagged in the Status Byte (ref. 8.2). For the versions marked with “OK”, overflow/underflow in ADC is also flagged in the same bit. The reason for this change is to discover if the ADC is saturated even though the compensated output is not saturated. This is possible e.g. if the applied acceleration is beyond the specified range.

11.2.3 SIGNATURE TEST

The EROM part of the Signature_Test function (ref. 8.4.14) does not work correctly. A correct functionality could be obtained by including the following code in the application program:

```
;+---------------------------------------------------------------------------
;| Name:
;|   sig_erom
;|
;| Description:
;|   Calculates the EROM signature
;|
;| Parameters:
;|    none
;|
;| Result:
;|   R3: signature byte 0
;|   R6: signature byte 1
;|   R7: signature byte 2
;|
;| Broken Registers:
;|   R0, R2, R3, R4, R5, R6, R7
;+-----------------------------------------------------------------------------

RSEG ?PR?SIG_EROM
sig_erom:
;********************************************************************************
;| Calculation of EROM signature from EROM
;********************************************************************************
     clr    R3                               ; prepare R3 for signature calculation
     clr    R6                               ; prepare R6 for signature calculation
     clr    R7                               ; prepare R7 for signature calculation
     clr    R4                               ; set EROM address 0000h
     clr    R5                               ; set EROM address 0000h

sig_erom_loop:
     movc                                    ; get data byte
     mov R2, R0                              ; save it into R2
     inc R4                                  ; increment EROM address
     movc                                    ; get data byte
     call sig_calc                           ; calculate signature
     add    R4, #01h                         ; increment EROM address
     addc   R5, #00h                         ; add overflow into R5
     cpse   R5, #10h                         ; address > 4K ?
     jmp    sig_erom_loop                    ; EROM end
     ret

;+-----------------------------------------------------------------------------
;| Name:
;|   sig_calc
;|
;| Description:
;|   calculates the signature of one word (16 bits)
;|
;| Parameters:
;|   R0:          data byte 0
;|   R2:          data byte 1
;|   R3, R6, R7:  either cleared for calculation start or
;|                the result of a previous signature calculation
;|
;| Result:
;|   R3:          signature byte 0
;|   R6:          signature byte 1
;|   R7:          signature byte 2
;|
;| Broken Registers:
;|   R3, R6, R7
;+-----------------------------------------------------------------------------

RSEG ?PR?SIG_CALC
sig_calc:
     shr    R7
     rrc    R6
     rrc    R3
     sbc    C
     xor    R7, #0d8h
     xor    R7, R0
     xor    R6, R2
     ret
```

Figure 11-1 Code example: Signature test

Note: This code will not calculate a correct signature during debugging in the RIDE environment if any breakpoints are defined. The reason for this is that the RIDE environment will replace the actual instruction in the address location of a breakpoint with a TRAP instruction.

11.2.4 DETECTION OF PARITY ERROR IN TEMPERATURE CALIBRATION

An error in a subroutine of the Meas_Temperature_Vdd function (ref. 8.4.2) results in only partial error detection. Calibration coefficients in EEPROM are read in two separate operations. If there is a parity error in the first read operation, the status bit will be overwritten by the second read operation.

Work-around: Add the following sequence in the application program, prior to the temperature measurement:

```
;Check Parity of Temperature coefficients
       mov    R0,# 050h     ; Coefficient buffer address
       mov    R7,# 011h     ; EEPROM Page Temperature coefficients
       mov    R6,# 0        ; EEPROM Page offset
       mov    R3,# 33       ; Number of bits to read (including parity bit)
       mov    R4,# 046h     ; EE_READ_W_PARITY
       sys
       sbc    R5.0          ; Test if parity bit set
       jmp    M_T_VDD_Parity_Error_handler
```

Figure 11-2 Work-around for detection of parity error in temperature calibration coefficients

11.2.5 DETECTION OF PARITY ERROR IN LF CALIBRATION

An error in a subroutine of the LF_Interface_On_Off function (ref. 8.4.9) results in only partial error detection. Calibration coefficients in EEPROM are read in two separate operations. If there is a parity error in the first read operation, the status bit will be overwritten by the second read operation.

Work-around: Add the following sequence in the application program, prior to turning the LF interface on:

```
; Check Parity of TRIMOFFS_LT coefficients
       mov    R0,# 050h     ; Coefficient buffer address
       mov    R7,# 01Bh     ; EEPROM Page LF Bias
       mov    R6,# 0        ; EEPROM Page offset LT
       mov    R3,# 5        ; Number of bits to read (including parity bit)
       mov    R4,# 046h     ; EE_READ_W_PARITY
       sys
       sbc    R5.0          ; Test if parity bit set.
       jmp    LF_Parity_Error_handler

; Check Parity of TRIMOFFS_HT coefficients
       mov    R7,# 01Bh     ; EEPROM Page LF Bias
       mov    R6,# 8        ; EEPROM Page offset HT
       mov    R3,# 5        ; Number of bits to read (including parity bit)
       mov    R4,# 046h     ; EE_READ_W_PARITY
       sys
       sbc    R5.0          ; Test if parity bit set.
       jmp    LF_Parity_Error_handler
```

Figure 11-3 Work-around for detection of parity error in LF calibration coefficients

11.2.6 LO-TEMP ERROR

For V1A and M2A, there was a design error causing wrong temperature measurements at low temperature (close to -40 °C) and low voltages (close to 2.1 V), especially at sudden drops in supply voltage. The error was present at <1%

of the components. After this error was detected, screening was performed on V1A components. The root cause is removed for V1B, V1C and M2B components.

11.2.7 IDDQ ERROR

For V1A, V1B and M2A, there was a design error causing increasing Power Down current in special cases. The error was present at approximately 0.2% of the components, starting 1 to 20 s after ADC was turned off. Screening was performed on V1A, V1B and M2A components. The root cause is removed for V1C and M2B components.

11.3 OTHER INFORMATION

11.3.1 AVOID ACCIDENTAL DEVICE WAKEUP

As shown in 8.5, each library revision uses either Type A or Type B boot sequence (ref. Table 8-17). For boot sequence type A, P10 and P11 pins are not defined. The application program must then define P10 and P11 to pull-up or pull-down.

Unintented wakeup events may occur if P10/P11 is configured to pull-down and VDD is changed rapidly: The reason for the port wakeup is a mono-flop having an internal capacitor, which is charged and discharged to generate a port wakeup pulse based on a negative edge on the port. The capacitor of the mono-flop, which is connected to port, is charged to VDD. The mono-flop generates a pulse causing a wakeup, if VDD is changed faster than charging the capacitor (5 – 20 µs), with a voltage increase of approx. 1V or more.

If the port is configured with pull-up resistor this behavior cannot be observed.

In case a pull-down load is connected to P10/11 and the device drives the corresponding port line HIGH before entering the Power Down mode, an unintended device wakeup condition may apply. Since the port P10/P11 is forced into input mode when entering Power Down mode, a high-to-low transition will occur (e.g. eventually delayed due to capacitive load) that will trigger the corresponding Port Sense mono-flop. Similar, operating the ports in output mode and forcing a high-to-low transition triggers the mono-flop also. If the mono-flop is still in its "triggered" state, in the moment the Power On Reset condition applies, the device will instantly power-up again (see also section 3.5.3). Depending on the application program, this may initiate an endless loop.

To prevent the application from such situations, the following sequence of instructions is recommended, before the Power Down mode is entered:

```
; Forcing the device into POWER-OFF mode and
; avoiding accidentally wakeup

      clr   P1DIR                          ; Switch all ports to input
      call  delay                          ; T = TPSMF + application delay
 inf: setb  PDWN                           ; Switch to Power Down mode
      jmp   inf                            ; Wait for Power Down
```

Figure 11-4 Code example: Avoid accidental device wakeup

The above sequence forces all ports to input mode and waits for a certain time. The delay needs to exceed the Port Sense mono-flop duration ($T_{PSMF,}$ min 5µs, max 100µs) plus the time the external circuitry needs to establish static conditions at the port lines. The Power Down mode is invoked and an endless loop is entered to prevent further execution before Power Down mode is entered.

11.3.2 INTERRUPT CONTROL SYSTEM

The Interrupt Control shows the following mal-function: The Interrupt Control may cause the device to vector the wrong interrupt if the interrupt enable register (IE) is manipulated and an interrupt event happens at the same moment in time. This may occur in two cases:

1. The control bit EA is cleared and an interrupt is detected within the same instruction clock cycle.

2. Any of the dedicated interrupt enable (IE) control bits is cleared, and the interrupt corresponding to these bits is detected within the same instruction clock cycle.

In both cases the Interrupt Control vectors to the wrong location and does not clear the corresponding interrupt request flag. Because Interrupt Control is disabled during program execution from system code, the error will not occur during execution of library functions.

Ad case 1:

The device always vectors to the WARM BOOT location (0000H). As a workaround to this problem, we suggest to modify the WARM BOOT code of the application code according to following scheme:

```
; WARM BOOT handler to cope with situations in which the Interrupt Control
; accidentally vectors the location 0000H
;
;-----------------------------------------
; Local declarations
;-----------------------------------------
       rseg   ?PR?main_cd
  ...
;-----------------------------------------
; Device State Start Addresses
;-----------------------------------------
       cseg   at 0000h
       jmp    modfied_WARM_BOOT
  ...
;-----------------------------------------
; Program code
;-----------------------------------------
       rseg   ?PR?main_cd

modified_WARM_BOOT:
       sbs    SP.7
       reti
       sbs    SP.6
       reti   jmp application_WARM_BOOT
  ...
```

Figure 11-5 Code example: Modification of "warm boot" code

The code lines as suggested serve to identify whether a hardware reset or a misled interrupt caused program execution to commence at the WARM BOOT location (0000H). The criteria chosen above is the value of the Stack Pointer, SP. In case of a hardware reset, thus execution of the BOOT routine, the SP is initialized to its maximum value (C0h). However, in case of a misled interrupt, the SP value is already decremented, as the interrupt is in service. If SP is found to equal C0h, a hardware reset is assumed and the application WARM BOOT sequence executed, otherwise the misled interrupt is simply ignored and control returned to the application program. The interrupt request flags are not changed.

Ad case 2:

The modified WARM BOOT sequence will resolve this situation, as described above.

In rare cases, a lower priority interrupt could occur within the same clock cycle. The device will then service the lower priority interrupt, and its request flag is cleared as desired. (Other interrupt request flags are not affected.) To avoid this problem, manually disable the interrupt source before clearing the dedicated IE bit (e.g. stop timer before turning off ET0 bit).