UM0563
User manual

L9952GXP software driver user manual

Introduction

The L9952GXP is a complex device described in the L9952GXP datasheet and the L9952GXP power management system IC application note (AN2751). For further information on these documents, please contact STMicroelectronics. To simplify use of L9952GXP device, an associated software driver has been developed. This document decribes how to use the software driver to control the L9952GXP device.

Contents

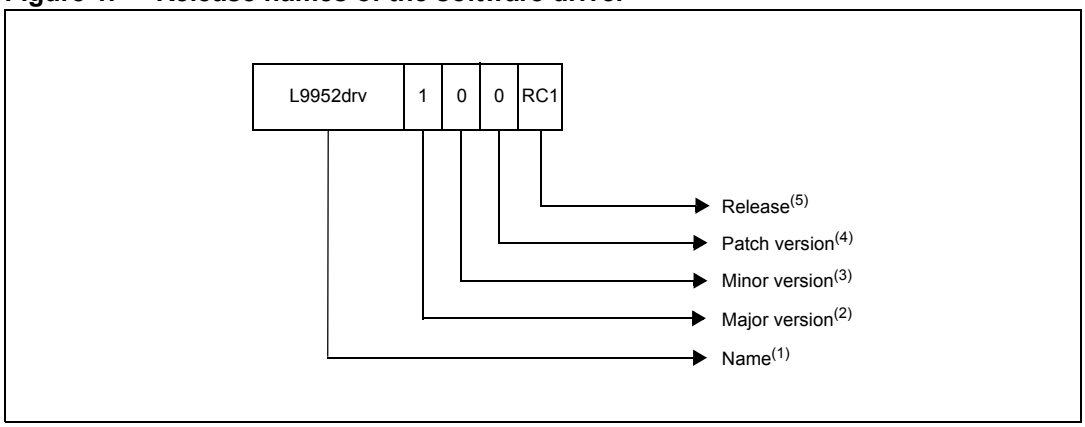
List of tables

List of figures

1 Naming different releases of the L9952GXP software driver

A standardized naming system exists for naming different releases of the software driver for the L9952GXP device. The name includes all relevant information, such as project name, functionality, patch version and release type. An example of a release name is the L9952drv_1-0-0_RC1, which is explained in *Figure 1*.

Figure 1. Release names of the software driver



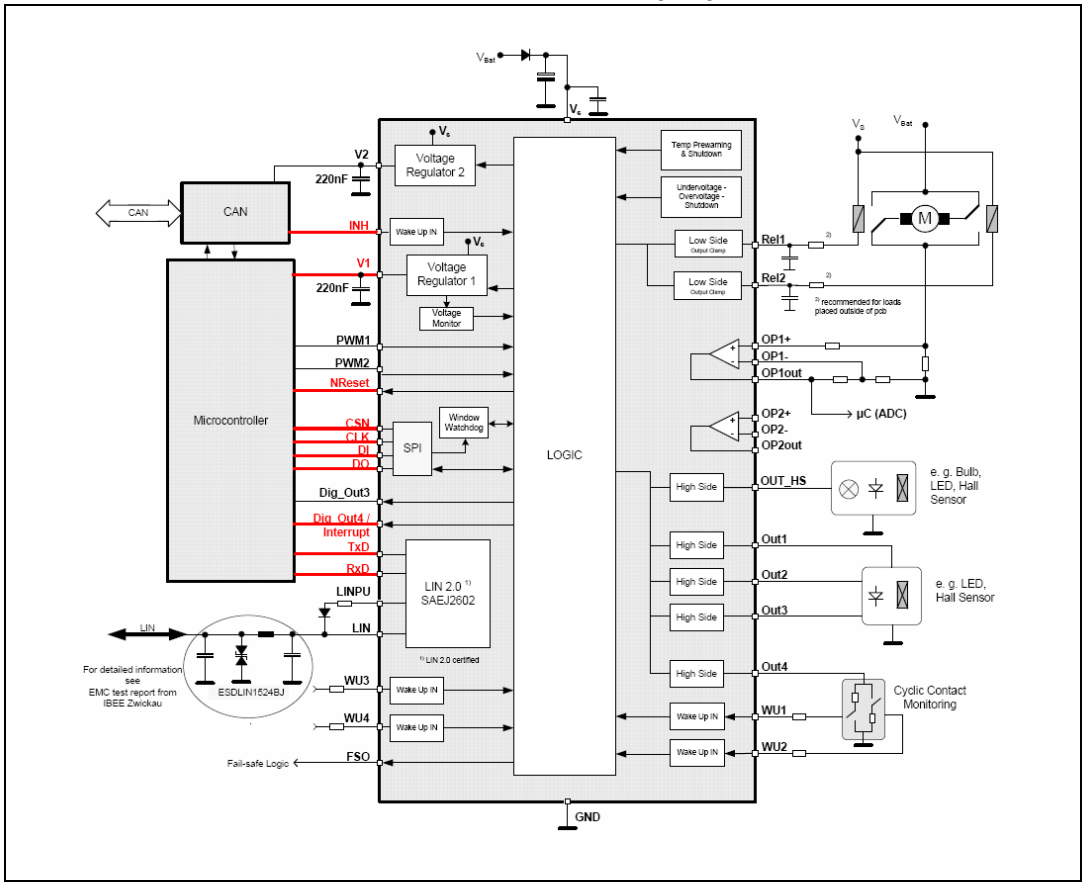
1. Name: Project name
2. Major version: Major functionality
3. Minor version: Minor functionality
   Odd numbers indicate the driver is still in development
   Even numbers indicate the driver is mature and a public version is available.
4. Patch version: No additional functionality permitted, except bug fixing.
5. Release: Type of release
   Official release (contains no additional marking)
   Release candidate (contains additional marking)

2 Purpose of the L9952GXP software driver

The L9952GXP device is generally used with a microcontroller. The microcontroller is supplied from the device and the two communicate via a standard SPI interface. The software driver for the L9952GXP is used to control the device (see *Figure 2*). Please refer to the L9952GXP datasheet for further details.

Note: The source code of the software driver is MISRA II compliant and follows Autosar requirements.

Figure 2. Block diagram of the L9952GXP software driver and the L9952GXP device with its associated microcontroller and peripherals



3 Architecture of the L9952GXP software driver

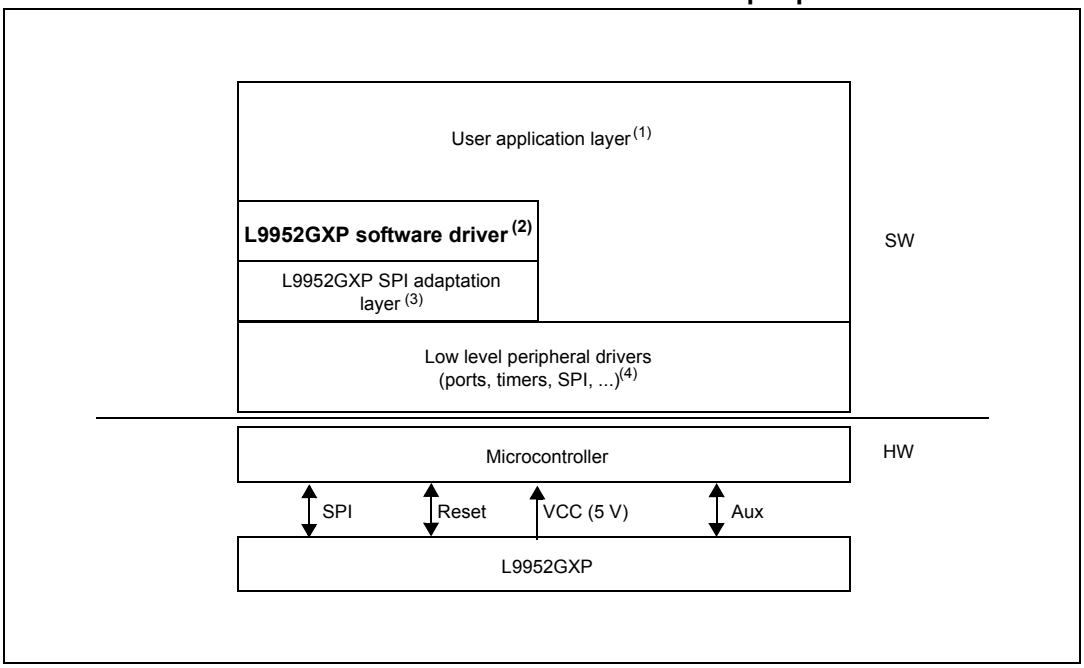
The software driver for the L9952GXP is independent from the microcontroller which physically controls the device. It is controlled by the user application layer of the software driver (see *Figure 3*).

The glue layer, between the software driver and the low level peripheral drivers of the microcontroller, is the L9952GXP SPI adaptation layer. This layer adapts the SPI interface of the software driver to the microcontroller dependent SPI driver. Such a mechanism decouples the software driver and makes its hardware independent (see *Figure 3*).

A detailed description of the low level peripheral drivers is ouside the scope of this document. They are dependent on the microcontroller in question, but, decouple the user application layer, making it hardware indpendent (see *Figure 3*). They provide the user application layer with all the functions necessary to control the L9952GXP device.

A detailed description of the user application layer is also outside the scope of this user manual. Please refer to the AN2751 (L9952GXP power management system IC application note) for further details.

Figure 3. Top level architecture of the L9952GXP software driver and the L9952GXP device with its associated microcontroller and peripherals

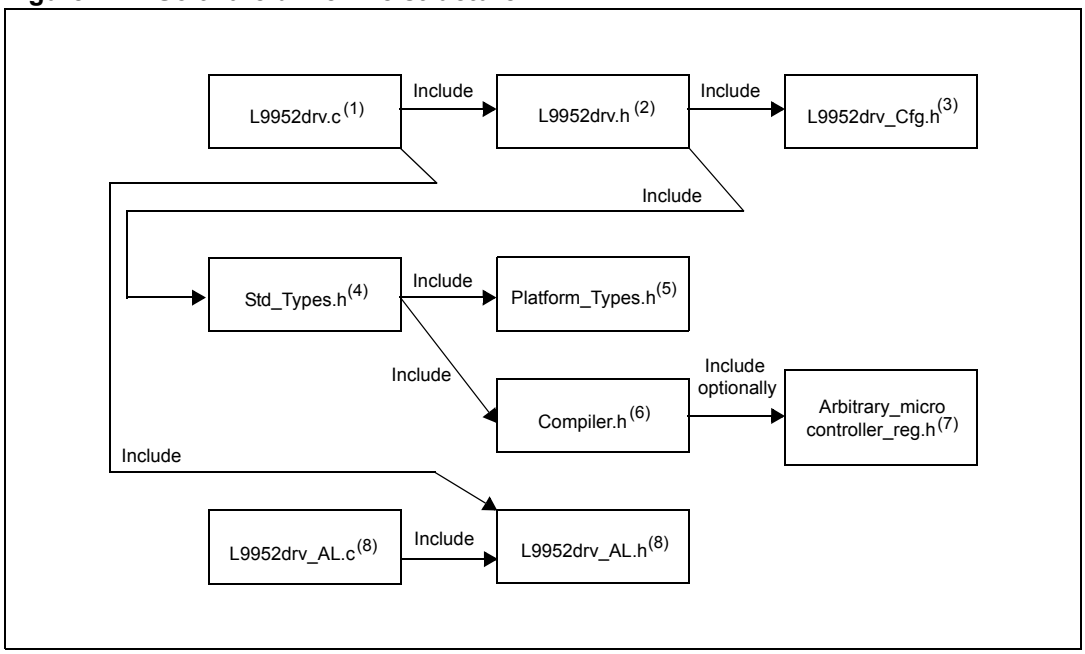


1. User application layer: User software which uses the application programming interface (API) of the software driver to control the device. Such software is ouside the scope of this document.
2. Software driver for the L9952GXP: The software driver for the L9952GXP which interfaces the user application layer.
3. L9952GXP SPI adaptation layer: A thin layer which adapts the software driver SPI interface to the microcontroller dependent SPI driver.
4. Low level peripheral drivers: Provide the user application layer with all the functions necessary to control the L9952GXP device. They control the peripherals of the microncontroller at the lowest power level.

3.1 L9952GXP software driver file structure

The L9952GXP sofware driver includes a set of source files which are shown in *Figure 4*. These files are described in more detail in *Section 8: Using the L9952GXP software driver on page 43*.

Figure 4. Software driver file structure



1. L9952drv.c file contains all functionality of the L9952GXP software driver.
2. L9952drv.h file contains the headers of the L9952GXP software driver functions.
3. L9952drv_Cfg.h file contains the configuration of the L9952GXP software driver functions.
4. Std_Types.h file contains the standard platform types of the L9952GXP software driver functions.
5. Platform_Types.h file contains the standard platform types for an 8-bit microcontroller platform.
6. Compiler.h: File contains the compiler dependent definitions.
7. Arbitrary_microcontroller_reg.h file contains the microcontroller dependent definitions. This is not necessary for the L9952GXP driver itself, which is hardware independent. This file is useful when the Std_Types.h/Compiler.h file is shared with other modules.
8. L9952drv_AL.c and L9952drv_AL.h files adapt the L9952GXP software driver to the low level SPI driver which allows the software driver to control the L9952GXP device.

3.2 L9952GXP software driver error detection

Errors are detected in the software driver by checking library usage at runtime (see flowchart in *Figure 5*).

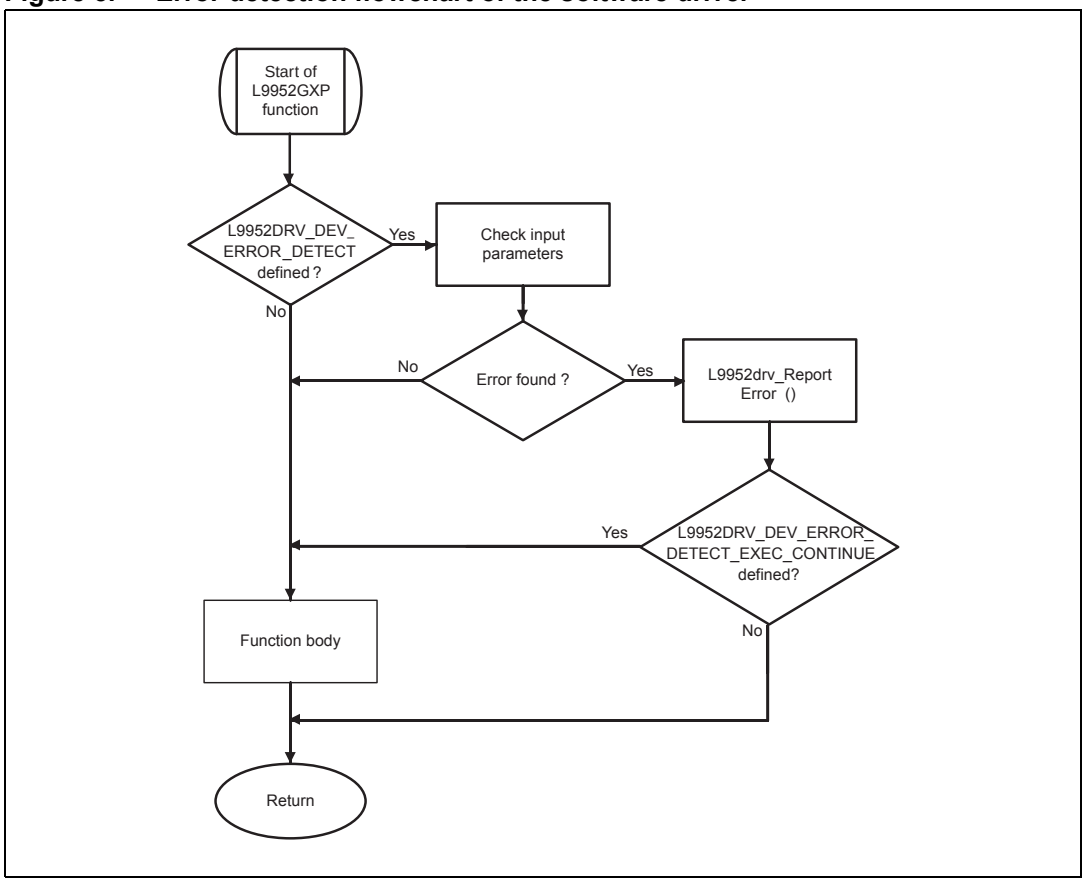
When the L9952DRV_DEV_ERROR_DETECT preprocessor switch is defined, the L9952GXP software driver function checks the initialization status of the library and the value range of the input parameters at runtime. If an error is detected, the L9952drv_ReportError() function is called.

Inside the L9952drv_ReportError() function, the L9952DRV_DEV_ERROR_DETECT_EXEC_CONTINUE switch allows the user to define the behavior of the library function.

If the L9952DRV_DEV_ERROR_DETECT_EXEC_CONTINUE switch is not activated, the library function is terminated immediately.

If the L9952DRV_DEV_ERROR_DETECT_EXEC_CONTINUE switch is defined, the function body of the library function is executed after exiting the L9952drv_ReportError() function.

Figure 5. Error detection flowchart of the software driver



3.3 L9952GXP low-level peripheral drivers

Hardware of the low-level peripheral drivers depend on the microcontroller being used together with the L9952GXP device. Functionality of the peripherals lies outside the scope of the software driver being described in this document. However, an example of low-level ST7 peripheral driver implementation with the software driver for the L9952GXP and an ST7 microcontroller was developed. This example is delivered with the L9952GXP software driver.

4 L9952GXP software driver functions

The L9952GXP sofware driver mainly carries out simple functions. It does not perform its own timing or processes. However, it must service the watchdog trigger, in the correct time window, of the L9952GXP device. Other timing functions are checked by the user application.

The following tables summarize the main functions of the L9952GXP software driver.

Table 1. L9952drv_Init

Service name	L9952drv_Init
Syntax	Void L9952drv_Init (void)
Service ID	0
Sync/async	Synchronous
Reentrancy	Non reentrant
Parameters (in)	None
Return value	None
Description	This function initializes the software driver. It has no affect on the device, the L9952GXP SPI adaptation layer, or the SPI interface. However, The SPI must be initialized prior to the function being used.
Caveats	L9952drv controlsonly one L9952 device. This function should not be called when an operation is running.
Configuration	None

Table 2. L9952drv_SetStandbyMode

Service name	L9952drv_SetStandbyMode
Syntax	Void L9952drv_SetStandbyMode (L9952drv_StandbyModeType **mode**)
Service ID	1
Sync/async	Asynchronous
Reentrancy	Non reentrant
Parameters (in)	Parameter **mode** defines the type of standby mode to be set
Return value	None
Description	This module sets selected standby modes of the L9952GXP device. The registers affected are the CR0.20..21, CR2.20 (by L9952drv_SetVReg1CurrentMonitorOn)(1)(2)(3)
Caveats	Power consumption of all components connected to the V1 voltage regulator must be decreased after standby mode V1 is set. This is done by setting halt mode in the microcontroller. The watchdog should be set in its regular watchdog time window (to avoid triggering it) by calling L9952drv_WdgTrigger(). This function waits until both control registers are completely transferred to the L9952GXP device via the SPI.
Configuration	None

1. The L9952drv_SetVReg1CurrentMonitorOn() function is called before control register 0 is modified prior to entering standby mode. This is for security reasons.
2. This function updates the software driver's internal copy of control register 0 and sends it immediately to the L9952GXP via the SPI.
3. This function refreshes the software driver's internal copy of status 0 register from the L9952GXP device via the SPI.

Table 3. L9952drv_SetOutMode

Service name	L9952drv_SetOutMode
Syntax	Void L9952drv_SetOutMode (uint8 **mask** L9952drv_OutModeType **mode**)
Service ID	2
Sync/async	Asynchronous
Reentrancy	Non reentrant
Parameters (in)	Parameter **mask** defines the outputs which should be configured Address range: 0x00 .. 0xF1 Bit 0: 1 = Mask is active for OUTHS, 0 = Mask is not active for OUTHS Bit 4: 1 = Mask is active for OUT1, 0 = Mask is not active for OUT1 Bit 5: 1 = Mask is active for OUT2, 0 = Mask is not active for OUT2 Bit 6: 1 = Mask is active for OUT3, 0 = Mask is not active for OUT3 Bit 7: 1 = Mask is active for OUT4, 0 = Mask is not active for OUT4 The following constants are defined in L9952drv.h file: #define L9952DRV_MASK_OUTHS 0x01 #define L9952DRV_MASK_OUT1 0x10 #define L9952DRV_MASK_OUT2 0x20 #define L9952DRV_MASK_OUT3 0x40 #define L9952DRV_MASK_OUT4 0x80 Example: OUT_MODE_MASK_OUT1 | OUT_MODE_MASK_OUT2 ... mask is active for OUT1 and OUT2. Parameter **mode** sets the mode for selected outputs
Return value	None
Description	This function controls high side (HS) outputs of the L9952GXP The registers affected are the CR0.0..14[1][2]
Caveats	This function waits until the data are completely transferred to the L9952GXP via the SPI.
Configuration	None

1. This function updates the software driver's internal copy of control register 0 and sends it immediately to the L9952GXP via the SPI.
2. This function refreshes the software driver's internal copy of status 0 register from the L9952GXP via the SPI.

Table 4. L9952drv_SetOutHSAutorecovery

Service name	L9952drv_SetOutHSAutorecovery
Syntax	Void L9952drv_SetOutHSAutorecovery (L9952drv_OutHSAutorecoveryType **mode**)
Service ID	3
Sync/async	Asynchronous
Reentrancy	Non reentrant
Parameters (in)	Parameter **mode** sets the autorecovery functionality of the OUT HS output
Return value	None
Description	This function enables/disables the autorecovery functionality of the OUT HS outputs of the L9952GXP. The register affected is the CR2.5(1)(2)
Caveats	This function waits until the data are completely transferred to the L9952GXP via the SPI.
Configuration	None

1. This function updates the software driver's internal copy of control register 2 and sends it immediately to the L9952GXP via the SPI.
2. This function refreshes the software driver's internal copy of status 0 register from the L9952GXP via the SPI.

Table 5. L9952drv_SetRelayOutput

Service name	L9952drv_SetRelayOutput
Syntax	Void L9952drv_SetRelayOutput (L9952drv_RelayOutputType **value**)
Service ID	4
Sync/Async	Asynchronous
Reentrancy	Non reentrant
Parameters (in)	Parameter **value** describes the state a relay is set to (or not set to)
Return value	None
Description	This function controls the low side (LS) relay outputs of the L9952GXP. The registers affected are the CR0.15..16(1)(2)
Caveats	This function waits until the data are completely transferred to the L9952GXP via the SPI.
Configuration	None

1. This function updates the software driver's internal copy of control register 0 and sends it immediately to the L9952GXP via the SPI.
2. This function refreshes the software driver's internal copy of status 0 register from the L9952GXP via the SPI.

Table 6. L9952drv_SetVoltageReg2Mode

Service name	L9952drv_SetVoltageReg2Mode
Syntax	Void L9952drv_SetVoltageReg2Mode (L9952drv_VoltageReg2ModeType **mode**)
Service ID	5
Sync/async	Asynchronous
Reentrancy	Non reentrant
Parameters (in)	Parameter **mode** defines the voltage regulator 2 mode to be set
Return value	None
Description	This function controls the voltage regulator 2 mode of the L9952GXP. The registers affected are the CR0.17..18(1)(2)
Caveats	This function waits until the data are completely transferred to the L9952GXP via the SPI.
Configuration	None

1. This function updates the software driver's internal copy of control register 0 and sends it immediately to the L9952GXP via the SPI.
2. This function refreshes the software driver's internal copy of status 0 register from the L9952GXP via the SPI.

Table 7. L9952drv_SetTimer1

Service name	L9952drv_SetTimer1
Syntax	Void L9952drv_SetTimer1 (L9952drv_Timer1PeriodType **period** L9952drv_Timer1ONTimeType **ontime**)
Service ID	6
Sync/async	Asynchronous
Reentrancy	Non reentrant
Parameters (in)	Parameter **period** defines the timer 1 period of the L9952GXP. Parameter **ontime** defines the period where the L9952GXP timer 1 output is set to 1.
Return value	None
Description	This function controls timer1 of the L9952GXP. The registers affected are the CR1.15..18(1)(2)
Caveats	This function waits until the data are completely transferred to the L9952GXP via the SPI.
Configuration	None

1. This function updates the software driver's internal copy of control register 1 and sends it immediately to the L9952GXP via the SPI.
2. This function refreshes the software driver's internal copy of status 1 register from the L9952GXP via the SPI.

Table 8. L9952drv_SetTimer2

Service name	L9952drv_SetTimer2
Syntax	Void L9952drv_SetTimer2(L9952drv_Timer2ONTimeType **ontime**)
Service ID	7
Sync/async	Asynchronous
Reentrancy	Non reentrant
Parameters (in)	Parameter **ontime** defines the part of the timing period in which the L9952GXP timer 2 output is set to 1.
Return value	None
Description	This function controls timer2 of the L9952GXP The register affected is the CR1.19(1)(2)
Caveats	This function waits until the data are completely transferred to the L9952GXP via the SPI.
Configuration	None

1. This function updates the software driver's internal copy of the control register 1 and sends it immediately to the L9952GXP via the SPI.
2. This function refreshes the software driver's internal copy of the status 1 register from the L9952GXP via the SPI.

Table 9. L9952drv_SetDigOutput

Service name	L9952drv_SetDigOutput
Syntax	Void L9952drv_SetDigOutput (L9952drv_DigOutputModeType **mode**)
Service ID	8
Sync/async	Asynchronous
Reentrancy	Non reentrant
Parameters (in)	Parameter **mode** describes the modes for digital outputs Dig_Out3 and Dig_Out4 of the L9952GXP.
Return value	None
Description	This function controls digital outputs 3 and 4 of the L9952GXP The registers affected are the CR1.12..14(1)(2)
Caveats	The modes of digital outputs 3 and 4 are coupled due to the internal construction of the L9952GXP. This function waits until the data are completely transferred to the L9952GXP via the SPI.
Configuration	None

1. This function updates the software driver's internal copy of control register 1 and sends it immediately to the L9952GXP via the SPI.
2. This function refreshes the software driver's internal copy of status 1 register from the L9952GXP via the SPI.

Table 10. L9952drv_SetWUInputMode

Service name	L9952drv_SetWUInputMode
Syntax	Void L9952drv_SetWUInputMode (uint8 **mask** L9952drv_WUInputModeType **mode**)
Service ID	9
Sync/async	Asynchronous
Reentrancy	Non reentrant
Parameters (in)	Parameter **mask** sets the wakeup (WU) input mode as current source (active) or current sink (not active): Address range: 0x00 .. 0x0F Bit 0: 1 = Mask active for WU1, 0 = Mask not active for WU1 Bit 1: 1 = Mask active for WU2, 0 = Mask not active for WU2 Bit 2: 1 = Mask active for WU3, 0 = Mask not active for WU3 Bit 3: 1 = Mask active for WU4, 0 = Mask not active for WU4 The following constants are defined in the L9952drv.h file: #define L9952DRV_MASK_WU1 0x01 #define L9952DRV_MASK_WU2 0x02 #define L9952DRV_MASK_WU3 0x04 #define L9952DRV_MASK_WU4 0x08 Example: L9952DRV_MASK_WU2 | L9952DRV_MASK_WU3 … mask is active for WU2 and WU3. Parameter **mode** describes the modes (current source/current sink) of the digital WU inputs.
Return value	None
Description	This function sets the mode (current source or current sink) for the digital wakeup inputs WU1, WU2, WU3, and WU4 of the L9952GXP. The registers affected are the CR1.8..11[1][2]
Caveats	This function waits until the data are completely transferred to the L9952GXP via the SPI.
Configuration	None

1. This function updates the software driver's internal copy of control register 1 and sends it immediately to the L9952GXP via the SPI.
2. This function refreshes the software driver's internal copy of status 1 register from the L9952GXP via the SPI.

Table 11. L9952drv_DisableWakeupSource

Service name	L9952drv_DisableWakeupSource
Syntax:	Void L9952drv_DisableWakeupSource (uint8 **mask** uint8 **bitpattern**)
Service ID	10
Sync/async	Asynchronous
Reentrancy	Non reentrant
Parameters (in)	Parameter **mask** sets the wakeup source. It disables the wakeup source from OUT1 and WU2 and enables the wakeup source from OUT2. Address range: 0x00 . .0xFF Bit 0: 1 = Mask active for WU1 0 = Mask not active for WU1 Bit 1: 1 = Mask active for WU2 0 = Mask not active for WU2 Bit 2: 1 = Mask active for WU3 0 = Mask not active for WU3 Bit 3: 1 = Mask active for WU4 0 = Mask not active for WU4 Bit 4: 1 = Mask active for openload of OUT1 wakeup source, 0 = Mask not active for openload of OUT1 wakeup source Bit 5: 1 = Mask active for openload of OUT2 wakeup source, 0 = Mask not active for openload of OUT2 wakeup source Bit 6: 1 = Mask active for openload of OUT3 wakeup source, 0 = Mask not active for openload of OUT3 wakeup source Bit 7: 1 = Mask active for openload of OUT4 wakeup source, 0 = Mask not active for openload of OUT4 wakeup source The following constants are defined in the L9952drv.h file: #define L9952DRV_MASK_WU1 0x01 #define L9952DRV_MASK_WU2 0x02 #define L9952DRV_MASK_WU3 0x04 #define L9952DRV_MASK_WU4 0x08 #define L9952DRV_MASK_OUT1 0x10 #define L9952DRV_MASK_OUT2 0x20 #define L9952DRV_MASK_OUT3 0x40 #define L9952DRV_MASK_OUT4 0x80 Example: L9952DRV_MASK_OUT1 | L9952DRV_MASK_WU4 … mask is active for WU1 and openload of OUT4 Parameter **bitpattern** disables/enables wakeup functionality separately for each wakeup source. Particular bits correspond with specifice wakeup sources in the same order as defined by mask (see above). Bits with a value of 1, disable a corresponding wakeup source. Bits with a value of 0, enable a corresponding wakeup source.

Table 11. L9952drv_DisableWakeupSource (continued)

Service name	L9952drv_DisableWakeupSource
Parameters (in) cont'd....	Example: L9952drv_DisableWakeupSource(0x32, 0x12) A better method, is to use the predefined constants below: L9952drv_DisableWakeupSource(L9952DRV_MASK_OUT1 | L9952DRV_MASK_OUT2 | L9952DRV_MASK_WU2, L9952DRV_MASK_OUT1 | L9952DRV_MASK_WU2); This example disables the wakeup source from OUT1 and WU2; and enables wakeup source from OUT2. The rest reminds unchanged.
Return value	None
Description	This function enables/disables wakeup functionality separately for each wakeup source defined by mask. The registers affected are the CR1.0..7(1)(2)
Caveats	This function waits until the data are completely transferred to the L9952GXP via the SPI.
Configuration	None

1. This function updates the software driver's internal copy of control register 1 and sends it immediately to the L9952GXP via the SPI.
2. This function refreshes the software driver's internal copy of status 1 register from the L9952GXP via the SPI.

Table 12. L9952drv_SetResetThresholdLevel

Service name	L9952drv_SetResetThresholdLevel
Syntax	Void L9952drv_SetResetThresholdLevel (L9952drv_ResetThresholdLevelType **level**)
Service ID	11
Sync/async	Asynchronous
Reentrancy	Non reentrant
Parameters (in)	Parameter **level** defines the reset threshold level value
Return value	None
Description	This function sets the reset threshold level of the L9952GXP The registers affected are the CR2.8..9(1)(2)
Caveats	This function waits until the data are completely transferred to the L9952GXP via the SPI.
Configuration	None

1. This function updates the software driver's internal copy of control register 2 and sends it immediately to the L9952GXP via the SPI.
2. This function refreshes the software driver's internal copy of status 0 register from the L9952GXP via the SPI.

Table 13. L9952drv_SetInputFilterMode

Service name	L9952drv_SetInputFilterMode
Syntax	Void L9952drv_SetInputFilterMode (uint8 **mask** L9952drv_InputFilterModeType **mode**)
Service ID	12
Sync/async	Asynchronous
Reentrancy	Non reentrant
Parameters (in)	Parameter **mask** sets the filter timing of the wakeup inputs Address range: 0x00 .. 0x0F Bit 0: 1 = Mask active for WU1, 0 = Mask not active for WU1 Bit 1: 1 = Mask active for WU2, 0 = Mask not active for WU2 Bit 2: 1 = Mask active for WU3, 0 = Mask not active for WU3 Bit 3: 1 = Mask active for WU4, 0 = Mask not active for WU4 The following constants are defined in the L9952drv.h file: #define L9952DRV_MASK_WU1 0x01 #define L9952DRV_MASK_WU2 0x02 #define L9952DRV_MASK_WU3 0x04 #define L9952DRV_MASK_WU4 0x08 Example: L9952DRV_MASK_WU1 | L9952DRV_MASK_WU4 … mask is active for WU1 and WU4. Parameter **mode** sets filter timing for related WU inputs
Return value	None
Description	This function controls the filter timing for digital wakeup inputs WU1, WU2, WU3 and WU4 of the L9952GXP. The registers affected are the CR2.10..17(1)(2)
Caveats	This function waits until the data are completely transferred to the L9952GXP via the SPI.
Configuration	None

1. This function updates the software driver's internal copy of control register 2 and sends it immediately to the L9952GXP via the SPI.
2. This function refreshes the software driver's internal copy of status 0 register from the L9952GXP via the SPI.

Table 14. L9952drv_SetOutOLThresholdLevel

Service name	L9952drv_SetOutOLThresholdLevel
Syntax	Void L9952drv_SetOutOLThresholdLevel (uint8 **mask** L9952drv_OutOLThresholdLevelType **level**)
Service ID	13
Sync/async	Asynchronous
Reentrancy	Non reentrant
Parameters (in)	Parameter **mask** sets the openload detection level (OLT) for related HS outputs Address range: 0x00 .. 0xF0 Bit 4: 1 = Mask active for OUT 1, 0 = Mask not active for OUT 1 Bit 5: 1 = Mask active for OUT 2, 0 = Mask not active for OUT 2 Bit 6: 1 = Mask active for OUT 3, 0 = Mask not active for OUT 3 Bit 7: 1 = Mask active for OUT 4, 0 = Mask not active for OUT 4 Following constants are defined in L9952drv.h file #define L9952DRV_MASK_OUT1 0x10 #define L9952DRV_MASK_OUT2 0x20 #define L9952DRV_MASK_OUT3 0x40 #define L9952DRV_MASK_OUT4 0x80 Example: L9952DRV_MASK_OUT1 | L9952DRV_MASK_OUT 4 ... mask is active for OUT1 and OUT4. Parameter **level** defines level for open load detection for related HS outputs
Return value	None
Description	This function sets the openload detection level for selected HS outputs 1..4 of the L9952GXP. The registers affected are the CR2.0..3(1)(2)
Caveats	This function waits until the data are completely transferred to the L9952GXP via the SPI.
Configuration	None

1. This function updates the software driver's internal copy of control register 2 and sends it immediately to the L9952GXP via the SPI.
2. This function refreshes the software driver's internal copy of status 0 register from the L9952GXP via the SPI.

Table 15. L9952drv_LinSetup

Service name	L9952drv_LinSetup
Syntax	Void L9952drv_LinSetup (L9952drv_LinSetupType **value**)
Service ID	14
Sync/async	Asynchronous
Reentrancy	Non reentrant
Parameters (in)	Parameter **value** LIN setup values for pull up, on bits 0..2, which enables dominant TxD timeout and LIN slope
Return value	None
Description	This function sets the LIN configuration. The registers affected are the CR2.6, 2.7, 2.18(1)(2)
Caveats	This function waits until the data are completely transferred to the L9952GXP via the SPI.
Configuration	None

1. This function updates the software driver's internal copy of control register 2 and sends it immediately to the L9952GXP via the SPI.
2. This function refreshes the software driver's internal copy of status 0 register from the L9952GXP via the SPI.

Table 16. L9952drv_ClearStatusRegisters

Service name	L9952drv_ClearStatusRegisters
Syntax	Void L9952drv_ClearStatusRegisters (void)
Service ID	15
Sync/async	Asynchronous
Reentrancy	Non reentrant
Parameters (in)	None
Return value	None
Description	This function clears the content of both L9952GXP status registers The register affected is the CR1.21(1)(2)
Caveats	This function waits until the data are completely transferred to the L9952GXP via the SPI.
Configuration	None

1. This function updates the software driver's internal copy of control register 1 and sends it immediately to the L9952GXP via the SPI. It then changes the internal copy of control register 1 and sends it again to the L9952GXP via the SPI.
2. This function refreshes the software driver's internal copy of status 1 register from the L9952GXP via the SPI.

Table 17. L9952drv_SetVsLockoutMode

Service name	L9952drv_SetVsLockoutMode
Syntax	Void L9952drv_SetVsLockoutMode (L9952drv_VsLockoutModeType **mode**)
Service ID	16
Sync/async	Asynchronous
Reentrancy	Non reentrant
Parameters (in)	Parameter **mode** selects the behavior of the L9952GXP after Vs (power supply voltage) recovery
Return value	None
Description	This function enables automatic recovery of outputs after Vs over/under voltage recovery of the L9952GXP. The register affected is the CR2.4(1)(2)
Caveats	This function waits until the data are completely transferred to the L9952GXP via the SPI.
Configuration	None

1. This function updates the software driver's internal copy of control register 2 and sends it immediately to the L9952GXP via the SPI.
2. This function refreshes the software driver's internal copy of status 0 register from the L9952GXP via the SPI.

Table 18. L9952drv_WdgTrigger

Service name	L9952drv_WdgTrigger
Syntax	Void L9952drv_WdgTrigger (void)
Service ID	17
Sync/async	Synchronous
Reentrancy	Non reentrant
Parameters (in)	None
Return value	None
Description	This function triggers the watchdog hardware. It is called cyclically by the upper layer function (usually the watchdog manager), to prevent the watchdog hardware from expiring. Right timing is essential. For more details see the L9952GXP datasheet. The register affected is the CR0.19(1)(2)(3) This function ensures the user pays attention to special conditions, such as, watchdog handling after wakeup from standby mode.
Caveats	This function waits until the data are completely transferred to the L9952GXP via the SPI.
Configuration	None

1. For security reasons, this function periodically refreshes dedicated bit positions of the main internal variables.
2. This function writes internal copies of control register 0 to the L9952GXP device via the SPI.
3. This function refreshes the software driver's internal copy of status 0 register from the L9952GXP via the SPI.

Table 19. L9952drv_SetRelayShutdownMode

Service name	L9952drv_SetRelayShutdownMode
Syntax	Void L9952drv_SetRelayShutdownMode (L9952drv_RelayShutdownModeType **mode**)
Service ID	18
Sync/async	Asynchronous
Reentrancy	Non reentrant
Parameters (in)	Parameter **mode** selects the behavior of the relay output of the L9952GXP after under/over voltage
Return value	None
Description	This function enables automatic shutdown of the relay outputs during Vs over/under voltage of the L9952GXP. The register affected is the CR2.19(1)(2)
Caveats	This function waits until the data are completely transferred to the L9952GXP via the SPI.
Configuration	None

1. This function updates the software driver's internal copy of control register 2 and sends it immediately to the L9952GXP via the SPI.
2. This function refreshes the software driver's internal copy of status 0 register from the L9952GXP via the SPI.

Table 20. L9952drv_GetGlobalErrorStatus

Service name	L9952drv_GetGlobalErrorStatus
Syntax	Bool L9952drv_GetGlobalErrorStatus (void)
Service ID	19
Sync/async	Synchronous
Reentrancy	Non reentrant
Parameters (in)	None
Return value	True: Global error status is set (at least one error flag is set) False: Global error status is reset (no error)
Description	This function gets the status of the cumulative error bit of the L9952GXP. It is implemented using the L9952drv_GetStatus0() function(1).
Caveats	Reading the DO bit of the SPI while the chip select (CS) pin is active is not standard SPI operation. The microcontroller-related SPI driver may not support such functionality and problems may arise in the SPI adaptation layer. This function waits until the data are completely transferred to the L9952GXP via the SPI.
Configuration	None

1. It is possible to get the status of the cumulative error bit by regularly reading status register 0/1 or by reading the first byte of either register. An alternative and quicker method is to read the DO bit of the SPI while the SPI is not clocked and the chipselect (CS) pin is active. The SPI must not be communicating during this procedure.

Table 21. L9952drv_GetStatus0

Service name	L9952drv_GetStatus0
Syntax	Void L9952drv_GetStatus0 (L9952drv_StatusRegType ***DataPtr**)
Service ID	20
Sync/async	Asynchronous
Reentrancy	Non reentrant
Parameters (in)	Parameter ***DataPtr** points to variables which are filled with data from the status 0 register.
Return value	None
Description	This function reads the current status from the L9952GXP via the SPI and then files the data structure in the staus 0 register(1)
Caveats	This function waits until the data are completely transferred to the L9952GXP via the SPI.
Configuration	None

1. Control register 2 is refreshed to the L9952GXP via the SPI using the software driver's interanl copy of it.

Table 22. L9952drv_ReadStatus0

Service name	L9952drv_ReadStatus0
Syntax	Void L9952drv_ReadStatus0 (L9952drv_StatusRegType ***DataPtr**)
Service ID	21
Sync/async	Asynchronous
Reentrancy	Non reentrant
Parameters (in)	Parameter ***DataPtr** points to variables which are filled with data from the status 0 register.
Return value	None
Description	This function files data structure with the most recent software driver internal copy of status 0 data.
Caveats	This function returns the internal copy of the data. The status 0 register is not refreshed by reading its status from the L9952GXP via the SPI(1).
Configuration	None

1. The internal copy of the status 0 register is refreshed at every watchdog trigger (every 10 ms) by calling the L9952drv_WdgTrigger() function.

Table 23. L9952drv_GetStatus1

Service name	L9952drv_GetStatus1
Syntax	Void L9952drv_GetStatus1 (L9952drv_StatusRegType ***DataPtr**)
Service ID	22
Sync/async	Asynchronous
Reentrancy	Non reentrant
Parameters (in)	Parameter ***DataPtr** points to variables which are filled with data from the status 1 register.
Return value	None
Description	This function reads the current status from the L9952GXP via the SPI and then files the data structure in the staus 1 register(1)
Caveats	This function waits until the data are completely transferred to the L9952GXP via the SPI.
Configuration	None

1. Control register 1 is refreshed to the L9952GXP via the SPI using the software driver's interanl copy of it.

Table 24. L9952drv_ReadStatus1

Service name	L9952drv_ReadStatus1
Syntax	Void L9952drv_ReadStatus1 (L9952drv_StatusRegType ***DataPtr**)
Service ID	23
Sync/async	Asynchronous
Reentrancy	Non reentrant
Parameters (in)	Parameter ***DataPtr** points to variables which are filled with data from the status 1 register.
Return value	None
Description	This function files data structure with the most recent software driver internal copy of status 1 data.
Caveats	This function returns the internal copy of the data. The status 1 register is not refreshed by reading its status from the L9952GXP via the SPI(1).
Configuration	None

1. Use the L9952drv_GetStatus1() function to get actual refreshed data via the SPI.

Table 25. L9952drv_SetVReg1CurrentMonitorOn

Service name	L9952drv_SetVReg1CurrentMonitorOn
Syntax	Void L9952drv_SetVReg1CurrentMonitorOn (void)
Service ID	24
Sync/async	Asynchronous
Reentrancy:	Non reentrant
Parameters (in)	None
Return value	None
Description	This function switches on voltage regulator 1 of the L9952GXP in stand by mode. The register affected is the CR2.20(1)(2)
Caveats	This function waits until the data are completely transferred to the L9952GXP via the SPI.
Configuration	None

1. This function updates the software driver's internal copy of control register 2 and sends it immediately to the L9952GXP via the SPI.
2. This function refreshes the software driver's internal copy of status 0 register from the L9952GXP via the SPI.

Table 26. L9952drv_SetIntMode

Service name	L9952drv_SetIntMode
Syntax	Void L9952drv_SetIntMode (L9952drv_IntModeType **mode**)
Service ID	25
Sync/async	Asynchronous
Reentrancy	Non reentrant
Parameters (in)	Parameter **mode** selects the interrupt mode of L9952GXP
Return value	None
Description	This function selects the interrupt mode of L9952GXP The register affected is the CR1.20(1)(2)
Caveats	This function waits until the data are completely transferred to the L9952GXP via the SPI.
Configuration	None

1. This function updates the software driver's internal copy of control register 1 and sends it immediately to the L9952GXP via the SPI.
2. This function refreshes the software driver's internal copy of status 1 register from the L9952GXP via the SPI.

5 L9952GXP software driver platform types

The following tables summarize the L9952GXP software driver platform types.

Table 27. L9952drv_StandbyModeType

Type	enum	
Range	L9952DRV_STANDBYMODE_V1	Switches L9952GXP to V1 standby mode
	L9952DRV_STANDBYMODE_VBAT	Switches L9952GXP to Vbat stand by mode
Description	Defines the standby modes of the L9952GXP	

Table 28. L9952drv_OutModeType

Type:	enum	
Range	L9952DRV_OUT_MODE_OFF	Driver is off in all modes
	L9952DRV_OUT_MODE_ON	Driver is on in active mode and off in standby mode
	L9952DRV_OUT_MODE_TIMER1	Driver is cyclic: It is on when Timer1 is in active or standby mode
	L9952DRV_OUT_MODE_TIMER2	Driver is cyclic: It is on when Timer2 is in active or standby mode
	L9952DRV_OUT_MODE_PWM1	Driver is controlled by the pulse-width modulation 1 (PWM1) input
	L9952DRV_OUT_MODE_PWM2	Driver is controlled by the PWM2 input
Description	Defines the output modes of the HS outputs	

Table 29. L9952drv_OutHSAutorecoveryType

Type	enum	
Range	L9952DRV_OUTHS_AUTORECOVERY_OFF	Disables the autorecovery functionality for the OUT HS outputs of the L9952GXP
	L9952DRV_OUTHS_AUTORECOVERY_ON	Enables the autorecovery functionality for the OUT HS outputs of the L9952GXP
Description	Defines the states for the OUT HS outputs which control autorecovery functionality	

Table 30. L9952drv_RelayOutputType

Type	enum	
Range	RELAYOUTPUT_OFF_OFF	Switches both relays off
	RELAYOUTPUT_ON_OFF	Switches relay REL1 on and REL2 off
	RELAYOUTPUT_OFF_ON	Switches relay REL1 off and REL2 on
	RELAYOUTPUT_ON_ON	Switches both relays on
	RELAYOUTPUT_OFF_X	Switches relay REL1 off and refreshes the previous state of REL2
	RELAYOUTPUT_ON_X	Switches relay REL1 on and refreshes the previous state of REL2
	RELAYOUTPUT_X_OFF	Switches relay REL2 off and refreshes the previous state of REL1
	RELAYOUTPUT_X_ON	Switches relay REL2 on and refreshes the previous state of REL1
Description	Defines the states to control the relay outputs	

Table 31. L9952drv_VoltageReg2ModeType

Type	enum	
Range	L9952DRV_VOLTAGE_REG2_OFF	Switches voltage regulator 2 off
	L9952DRV_VOLTAGE_REG2_ON_ACTIVE	Switches voltage regulator 2 on when the L9952GXP is in active mode
	L9952DRV_VOLTAGE_REG2_ON_V1	Switches voltage regulator 2 on when the L9952GXP is in active or V1 standby mode
	L9952DRV_VOLTAGE_REG2_ON	Switches voltage regulator 2 on
Description	Defines the voltage regulator 2 modes of the L9952GXP	

Table 32. L9952drv_Timer1PeriodType

Type	enum	
Range	L9952DRV_TIMER1PERIOD_500	Sets timer 1 period to 0.5 s
	L9952DRV_TIMER1PERIOD_1000	Sets timer 1 period to 1 s
	L9952DRV_TIMER1PERIOD_1500	Sets timer 1 period to 1.5 s
	L9952DRV_TIMER1PERIOD_2000	Sets timer 1 period to 2 s
	L9952DRV_TIMER1PERIOD_2500	Sets timer 1 period to 2.5 s
	L9952DRV_TIMER1PERIOD_3000	Sets timer 1 period to 3 s
	L9952DRV_TIMER1PERIOD_3500	Sets timer 1 period to 3.5 s
	L9952DRV_TIMER1PERIOD_4000	Sets timer 1 period to 4 s
Description	Defines the timer 1 period of the L9952GXP	

Table 33. L9952drv_Timer1ONTimeType

Type	enum	
Range	L9952DRV_TIMER1ONTIME_10	Defines the ‘on’ time value as 10 ms
	L9952DRV_TIMER1ONTIME_20	Defines ‘on’ time value as 20 ms
Description	Defines the ‘on’ time of timer 1 of the L9952GXP	

Table 34. L9952drv_Timer2ONTimeType

Type	enum	
Range	L9952DRV_TIMER2ONTIME_01	Defines the ‘on’ time value as 100 µs
	L9952DRV_TIMER2ONTIME_1	Defines the ‘on’ time value as 1 ms
Description	Defines the ‘on’ time of timer 2 of the L9952GXP	

Table 35. L9952drv_DigOutputModeType

Type	enum	
Range	L9952DRV_DO_WU3_WU4	Sets Dig_Out3 to loopback of WU3 Sets Dig_Dut4 to loopback of WU4
	L9952DRV_DO_HIZ_WU4	Sets Dig_Out3 to high impedance Sets Dig_Dut4 to loopback of WU4
	L9952DRV_DO_WU3_HIZ	Sets Dig_Out3 to loopback of WU3 Sets Dig_Dut4 to high impedance
	L9952DRV_DO_WU3_OLHS2	Sets Dig_Out3 to loopback of WU3 Sets Dig_Out4 to loopback of open load state of OUT2
	L9952DRV_DO_OLHS1_WU4	Sets Dig_Out3 to loopback of open load state of OUT1 Sets Dig_Out4 to loopback of WU4
	L9952DRV_DO_OLHS1_OLHS2	Sets Dig_Out3 to loopback of open load state of OUT1 Sets Dig_Out4 to loopback of open load state of OUT2
	L9952DRV_DO_OLHS1_HIZ	Sets Dig_Out3 to loopback of open load state of OUT1 Sets Dig_Out4 to high impedance
	L9952DRV_DO_HIZ_OLHS2	Sets Dig_Out3 to high impedance Sets Dig_Out4 to loopback of open load state of OUT2
Description	Defines which signals are looped to digital outputs Dig_Out3 and Dig_Out4	

Table 36. L9952drv_WUInputModeType

Type	enum	
Range	L9952DRV_WU_INPUT_MODE_CUR_SINK	Defines the current sink (pull down) mode for related WU inputs
	L9952DRV_WU_INPUT_MODE_CUR_SOURCE	Defines the current source (pull up) mode for related WU inputs
Description	Defines current source/current sink modes for related WU inputs	

Table 37. L9952drv_InputFilterModeType

Type	enum	
Range	L9952DRV_IN_FILTER_64	Defines the input filter time value as 64 µs and leaves it unsynchronized
	L9952DRV_IN_FILTER_80_TIMER2	Defines the input filter time value as 80 µs and synchronizes it with timer 2
	L9952DRV_IN_FILTER_800_TIMER2	Defines the input filter time value as 800 µs and synchronizes it with timer 2
	L9952DRV_IN_FILTER_800_TIMER1	Defines the input filter time value as 800 µs and synchronizes it with timer 1
Description	Defines the input filter configuration for related WU inputs	

Table 38. L9952drv_ResetThresholdLevelType

Type	enum	
Range	L9952DRV_RESET_THRESHOLD_4650	Defines the reset threshold level as 4.65 V
	L9952DRV_RESET_THRESHOLD_4350	Defines the reset threshold level as 4.35 V
Description	Defines the reset threshold level value	

Table 39. L9952drv_OutOLThresholdLevelType

Type	enum	
Range	L9952DRV_OUT_OLTHRESHOLD_2	Defines the output openload threshold level value as 2 mA
	L9952DRV_OUT_OLTHRESHOLD_8	Defines the output openload threshold level value as 8 mA
Description	Defines the output openload threshold level value for related outputs	

Table 40. L9952drv_LinSetupType

Type	enum	
Range	L9952DRV_LIN_SL_DI_TIM_EN_PUP_EN	Enables master pullup (LINPU) and TxD dominant timeout for LIN; disables alternative LIN slope
	L9952DRV_LIN_SL_DI_TIM_EN_PUP_DI	Disbles master pullup (LINPU) and alternative LIN slope; enables TxD dominant timeout for LIN
	L9952DRV_LIN_SL_DI_TIM_DI_PUP_EN	Enables master pullup (LINPU); disables TxD dominant timeout for LIN and alternative LIN slope
	L9952DRV_LIN_SL_DI_TIM_DI_PUP_DI	Disbles master pullup (LINPU), TxD dominant timeout for LIN, and alternative LIN slope
	L9952DRV_LIN_SL_EN_TIM_EN_PUP_EN	Enables master pullup (LINPU), TxD dominant timeout for LIN, and alternative LIN slope
	L9952DRV_LIN_SL_EN_TIM_EN_PUP_DI	Disbles master pullup (LINPU); enables TxD dominant timeout for LIN and alternative LIN slope
	L9952DRV_LIN_SL_EN_TIM_DI_PUP_EN	Enables master pullup (LINPU) and alternative LIN slope; disables TxD dominant timeout for LIN
	L9952DRV_LIN_SL_EN_TIM_DI_PUP_DI	Disbles master pullup (LINPU) and TxD dominant timeout for LIN; enables alternative LIN slope
Description	Defines all possible configurations related to the L9952GXP LIN interface. One configuration must be chosen.	

Table 41. L9952drv_VsLockoutModeType

Type	enum	
Range	L9952DRV_VS_LOCKOUT_DISABLE	Disables Vs lockout after a Vs over/under voltage condition has disappeared. Outputs automatically recover according to the output settings in the CR0 register.
	L9952DRV_VS_LOCKOUT_ENABLE	Enables Vs lockout. After a Vs over/under voltage recovery, outputs remain off until the status register 1 bits 0 and 1 are cleared by the CLR command, using bit 21 of control regsiter 1.
Description	Defines Vs lockout modes for controlling automatic recovery after Vs over/under voltage recovery of the L9952GXP	

Table 42. L9952drv_RelayShutdownModeType

Type	enum	
Range	L9952DRV_RELAY_SHUTDOWN_ENABLE	Enables shutdown of relay outputs during Vs over/under voltage of the L9952GXP. REL1.2 turned off in case of Vs over/under voltage.
	L9952DRV_RELAY_SHUTDOWN_DISABLE	Disables shutdown of relay outputs during Vs over/under voltage of the L9952GXP. REL1.2 remain on in case of Vs over/under voltage.
Description	Defines modes which control Vs over/under voltage shutdown of REL1.2 (low side drivers)	

Table 43. L9952drv_IntModeType

Type	enum	
Range	L9952DRV_INT_DISABLE	Disables interrupt mode
	L9952DRV_INT_ENABLE	Enables interrupt mode
Description	Defines modes which control the interrupt mode of the L9952GXP	

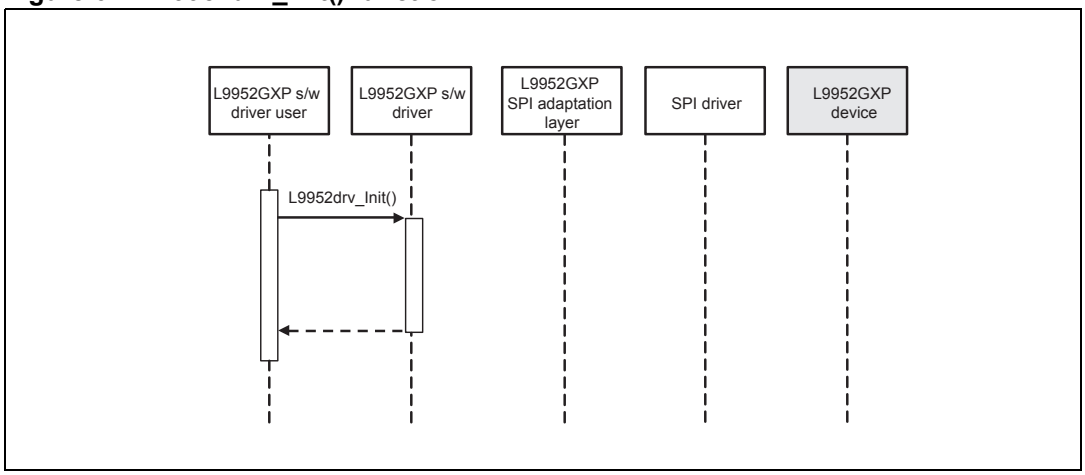
Table 44. L9952drv_StatusRegType

Type	uint32
Range	0..0x00FFFFFF
Description	The L9952drv_StatusRegType defines global data types for access to a complete local copy of the L9952GXP status registers 0 and 1. Access to the internal structure and particular bit variables of each status register is provided by a set of masks related to each bit variable inside the status register.

6 Dynamic view of the L9952GXP software driver functions

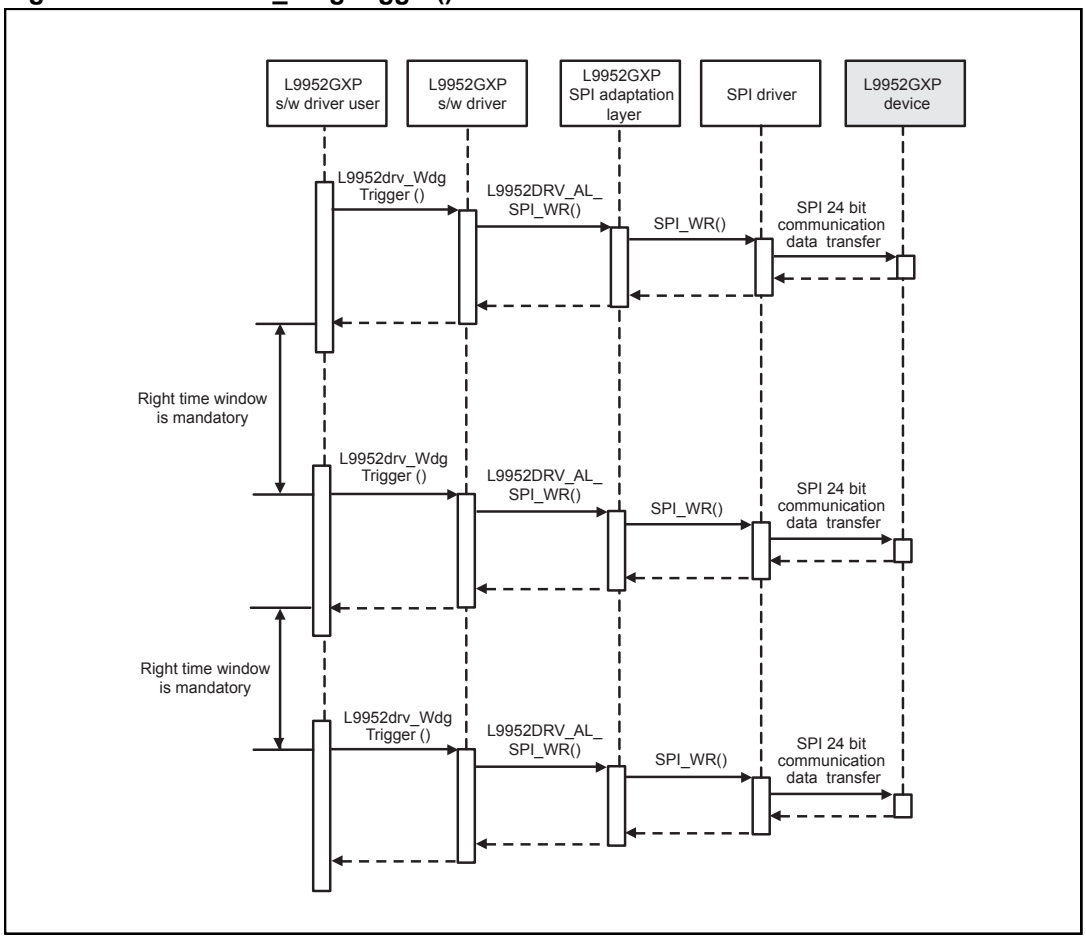
The following figures give a dynamic overview of the L9952GXP software driver functions.

Figure 6. L9952drv_Init() function



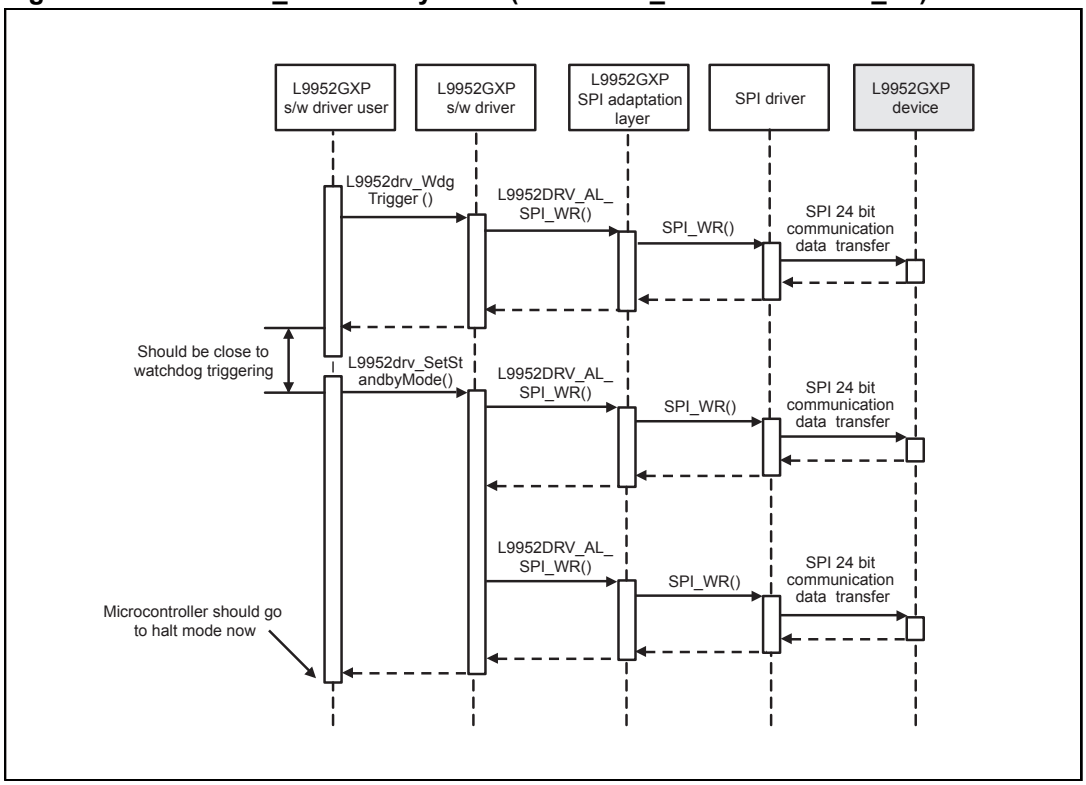
1. The hardware layer is depicted with a grey background and the software layer is depicted with a white background.

Figure 7. L9952drv_WdgTrigger() function



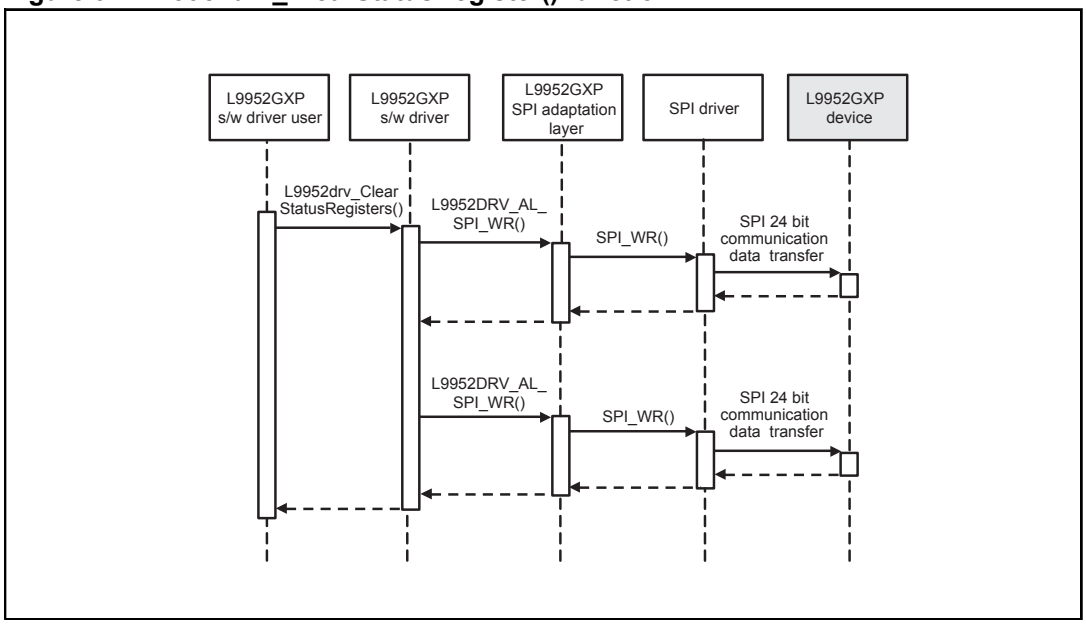
1. The hardware layer is depicted with a grey background and the software layer is depicted with a white background.

Figure 8. L9952drv_SetStandbyMode (L9952DRV_STANDBYMODE_V1) function



1. The hardware layer is depicted with a grey background and the software layer is depicted with a white background.

Figure 9. L9952drv_ClearStatusRegister() function

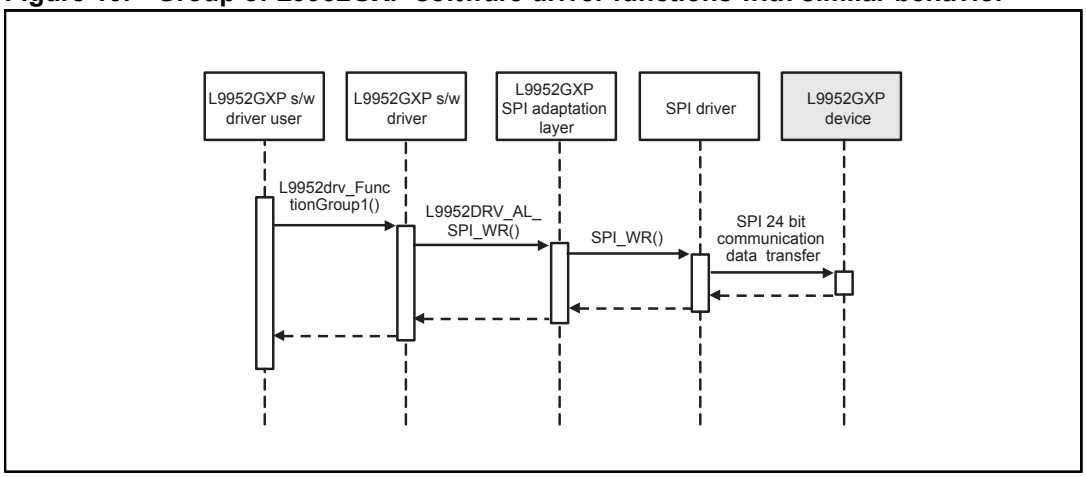


1. The hardware layer is depicted with a grey background and the software layer is depicted with a white background.

The L9952drv_SetStandbyMode(L9952DRV_STANDBYMODE_VBAT) is one of a group of functions of the L9952GXP which have the same behavior. Other functions in this group include:

- L9952drv_SetOutMode()
- L9952drv_SetRelayOutput()
- L9952drv_SetVoltageReg2Mode()
- L9952drv_SetOutHSAutorecovery()
- L9952drv_SetTimer1()
- L9952drv_SetTimer2()
- L9952drv_SetDigOutput()
- L9952drv_SetWUInputMode()
- L9952drv_DisableWakeupSource()
- L9952drv_SetResetThresholdLevel()
- L9952drv_SetInputFilterMode()
- L9952drv_SetOutOLThresholdLevel()
- L9952drv_LinSetup()
- L9952drv_SetVsLockoutMode()
- L9952drv_SetRelayShutdownMode()
- L9952drv_GetStatus0()
- L9952drv_GetStatus1()
- L9952drv_SetVReg1CurrentMonitorOn()
- L9952drv_SetIntMode

Figure 10. Group of L9952GXP software driver functions with similar behavior

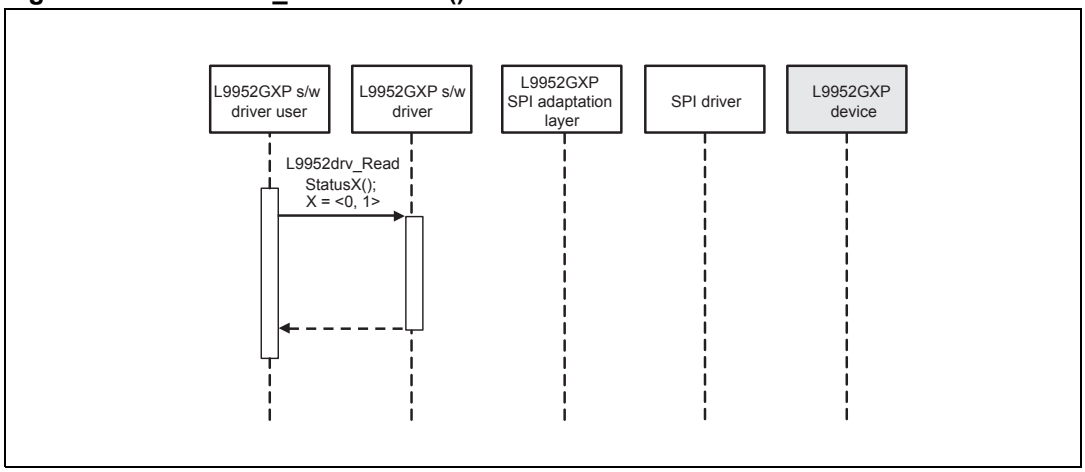


1. The hardware layer is depicted with a grey background and the software layer is depicted with a white background.

The ReadStatusx() function is comprised of two similarly behaved functions:

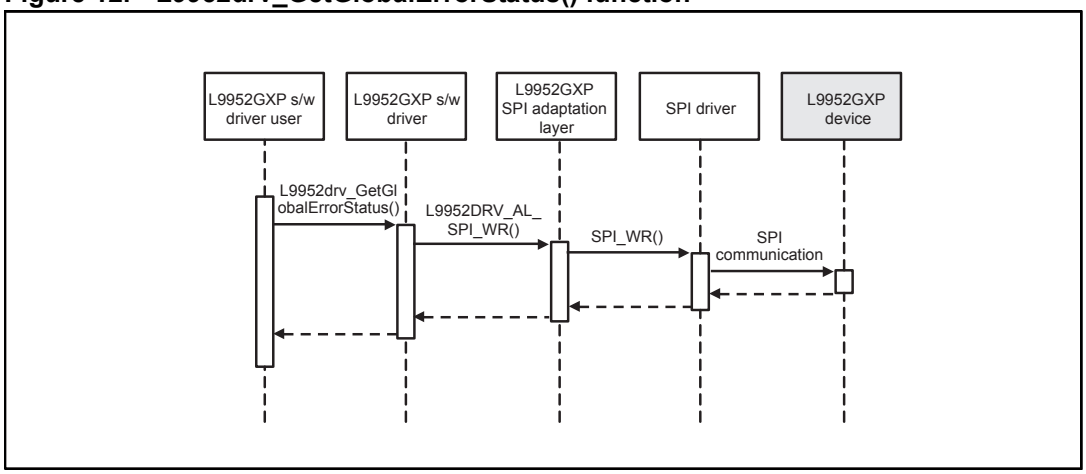
- L9952drv_ReadStatus0()
- L9952drv_ReadStatus1()

Figure 11. L9952drv_ReadStatusx() function



1. The hardware layer is depicted with a grey background and the software layer is depicted with a white background.

Figure 12. L9952drv_GetGlobalErrorStatus() function



1. The hardware layer is depicted with a grey background and the software layer is depicted with a white background.

6.1 L9952GXP software driver error handling function

Runtime error handling is supported via the L9952drv_ReportError() function. This function is called from any of the L9952GXP driver functions when an error is detected. It is originally implemented as a dummy function. Runtime error handling allows the user to implement the required functionality, for example, error logging, setting some user global error variables or flags for detailed error localization, and so on. The function is available only if the preprocessor switch L9952DRV_DEV_ERROR_DETECT is set.

For further details, please refer to *Section 3.2: L9952GXP software driver error detection on page 9*.

The error handling function does not care about the arbitrary SPI communication driver. All SPI errors and timeouts must be solved inside the SPI driver.

Note: If there is an error lock and the the watchdog expires, the microcontroller may be reset by the L9952GXP device.

Table 45. L9952drv_ReportError

Service name	L9952drv_ReportError
Syntax	Bool L9952drv_ReportError (uint8 **ServiceId** uint8 **ErrorId**)
Service ID	-
Sync/async	Asynchronous
Reentrancy	Non reentrant
Parameters (in)	Parameter **ServiceId** defines the ID of the failing function (service) Parameter **ErrorId** defines ID of the error
Return value	True: The software driver continues in an interrupted function in run mode without the L9952DRV_DEV_ERROR_DETECT. False: The software driver stops execution of the interrupted function by returning (avoiding) any communication with the L9952GXP device(1).
Description	Service for reporting errors, especially during the development phase
Caveats	This function is originally implemented as a dummy function. It allows the user to implement the required functionality, for example error logging, setting of some user global error variables for detailed error localization, and so on.
Configuration	None

1. The return value is driven in the L9952drv_Cfg.h file by #DEFINE L9952DRV_DEV_ERROR_DETECT_EXEC_CONTINUE.

7 L9952GXP software driver system requirements

7.1 Operating system, compiler and interrupts

- The L9952GXP driver is written in ANSI C and is MISRA II compliant
- No operating system is required or directly supported
- No interrupts are used or required
- The compiler depends on the microcontroller being used

7.2 Hardware requirements

There are no special hardware requirements for the L9952GXP software driver. For hardware details, please refer to the L9952GXP datasheet.

7.3 Memory mapping

Memory mapping depends on the microcontroller being used and the setup of the compiler. *Table 46* gives two setups of the ST7L3 microcontroller which is compiled by Cosmic C (version 4.5.5).

Table 46. Memory mapping examples

Map file items	Setup 1	Setup 2
Microcontroller	ST7L3	ST7L3
Memory model	Compact memory	Long stack
Memory model short cut	modc	modsl
Version	Run	Debug
Version note	Not defined L9952DRV_DEV_ERROR_DETECT	Defined L9952DRV_DEV_ERROR_DETECT
Filename	l9952drv_T5_1_R_c.map	l9952drv_T5_1_D_sl.map
Map file items	**Result**	
Text (ROM)	1445	3183
Const (ROM)	4	4
Ubsct (RAM)	23	8
Share (RAM)	15	0
Data (RAM)	0	1
Bss (RAM stack)	0	15

8 Using the L9952GXP software driver

The L9952GXP software driver includes a set of source files which can be modified by the user if necessary to customize the driver. Some changes are essential, for example, see *Section 8.7: L9952drv_AL.c and L9952drv_AL.h files* below.

8.1 Main.c file

Before beginning, this file is empty because it exists to compile the software driver for the L9952GXP device.

Note: The user must regularly service the L9952GXP device watchdog.

8.2 L9952drv.C file

The L9952_drv.c file contains all functionality of the L9952GXP software driver.

It is recommended to change nothing in this file except the L9952drv_ReportError function.

The L9952drv_ReportError function is called from the L9952GXP software driver function if the L9952DRV_DEV_ERROR_DETECT switch is active. This function is originally implemented as a dummy function. It allows the user to implement the required functionality, for example, error logging, settings of some user global error variables for detailed error localization, and so on. See *Section 3.2: L9952GXP software driver error detection on page 9*.

8.3 L9952_Cfg file

The source file, L9952_Cfg.h, is a dedicated area where L9952GXP software driver configuration items are placed. Before the L9952GXP is configured, two preprocessor switches allow the user to tune driver behavior. No changes are required, but, it is possible to make them. For more details about preprocessor switches see *Section 3.2: L9952GXP software driver error detection on page 9*.

8.4 Std_Types.h file

The Std_Types.h file is an Autosar compliant file. It defines standard platform types of the L9952GXP software driver functions. No user changes are necessary.

8.5 Platform_Types.h file

File Platform_Types.h is an Autosar compliant file which defines global symbols depending on the microcontroller platform (8-bit, 16-bit, or 32-bit). By default, this file contains values for an 8-bit microcontroller platform, for which no user changes are necessary.

8.6 Compiler.h file

File Compiler.h is an Autosar compliant file which defines compiler types. Currently, this file contains values for the Cosmic C-compiler. The L9952GXP software driver is hardware independent, therefore, the Compiler.h file does not include any microcontroller dependent file. No user changes are necessary.

8.7 L9952drv_AL.c and L9952drv_AL.h files

These files adapt the L9952GXP software driver to the low level SPI driver which allows the software driver to control the L9952GXP device (see *Section 3*).

The L9952drv_AL.h file contains the macro:

```
#define L9952DRV_AL_SPI_WR SPI_Send
```

This macro assigns a function, SPI_Send(), to transfer the SPI driver to the L9952GXP device. SPI_Send() replaces the general function L9952DRV_AL_SPI_WR. By default, SPI_Send() exists as a dummy function which needs to be updated or replaced. If it does not exist, it must be created in the L9952drv_AL.c file.

The SPI_Send() prototype is:

```
extern void SPI_Send (uint8 *DataTX, uint8 *DataRX);
```

A description of SPI_Send() is as follows:

- DataTX defines a pointer to 3 bytes of data to be transmitted via the SPI
- DataRX defines a pointer to 3 bytes of data to be received via the SPI

Data transmitted via the SPI are 3 bytes (24 bits) in length. They are transmitted in the order: byte 1, byte 2, and byte 3. The MSB is always transmitted as the first bit from the byte. Data are received in the same order as those transmitted.

Note: The SPI driver must be initialized before the L9952GXP software driver is used for the first time.

9 Revision history

Table 47. Document revision history

Date	Revision	Changes
08-Jul-2008	1	Initial release
22-Sep-2013	2	Updated Disclaimer.

Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST’s terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST’S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

ST PRODUCTS ARE NOT DESIGNED OR AUTHORIZED FOR USE IN: (A) SAFETY CRITICAL APPLICATIONS SUCH AS LIFE SUPPORTING, ACTIVE IMPLANTED DEVICES OR SYSTEMS WITH PRODUCT FUNCTIONAL SAFETY REQUIREMENTS; (B) AERONAUTIC APPLICATIONS; (C) AUTOMOTIVE APPLICATIONS OR ENVIRONMENTS, AND/OR (D) AEROSPACE APPLICATIONS OR ENVIRONMENTS. WHERE ST PRODUCTS ARE NOT DESIGNED FOR SUCH USE, THE PURCHASER SHALL USE PRODUCTS AT PURCHASER’S SOLE RISK, EVEN IF ST HAS BEEN INFORMED IN WRITING OF SUCH USAGE, UNLESS A PRODUCT IS EXPRESSLY DESIGNATED BY ST AS BEING INTENDED FOR “AUTOMOTIVE, AUTOMOTIVE SAFETY OR MEDICAL” INDUSTRY DOMAINS ACCORDING TO ST PRODUCT DESIGN SPECIFICATIONS. PRODUCTS FORMALLY ESCC, QML OR JAN QUALIFIED ARE DEEMED SUITABLE FOR USE IN AEROSPACE BY THE CORRESPONDING GOVERNMENTAL AGENCY.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.
Information in this document supersedes and replaces all information previously supplied.
The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Philippines - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

www.st.com