Never stop thinking

SmartLEWIS™ TRX - TDA5340

Software Example for TDA5340

Multi-channel RF transceiver for sub 1GHz

Application Note

Revision 1.0, 2012-05-30

Wireless Sense & Control

Edition 2012-05-30

Published by
Infineon Technologies AG
81726 Munich, Germany

Legal Disclaimer

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics. With respect to any examples or hints given herein, any typical values stated herein and/or any information regarding the application of the device, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation, warranties of non-infringement of intellectual property rights of any third party.

Information

For further information on technology, delivery terms and conditions and prices, please contact the nearest Infineon Technologies Office (**www.infineon.com**).

Warnings

Due to technical requirements, components may contain dangerous substances. For information on the types in question, please contact the nearest Infineon Technologies Office.

Infineon Technologies components may be used in life-support devices or systems only with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.

Revision History	
Page or Item	**Subjects (major changes since previous revision)**
Revision 1.0, 2012-05-30	
	Initial Version 1.0

Table of Contents

1 Introduction

The TDA5340 Software Example shows a possible low level driver software for wireless remote control applications like Remote Keyless Entry (RKE) or Home Automation driving the TDA5340 using a XC886 Micro Controller (8051 derivative) from Infineon. The available software example files include in-line documentation and are developed to enable an easy software development start and fast time to market. The software example files can be downloaded from **http://www.infineon.com/TDA5340 .** The installer for the source code of the XC886 Software Example is called *TDA5340_Software_Example_Vx.y.msi*. The installer is included in the following download packages:

- TDA5340 - Software Example

More documentation of the source code done with doxygen is also included in the download package and can be displayed with a standard browser by opening file *./TDA5340SoftwareExample/Docu/html/index.html* after running *TDA5340_Software_Example_Vx.y.msi*. installer.

The code was designed and debugged by using the µVision4 development IDE from Keil and the TDA5340 evaluation board including the UWLink interface board from Infineon.

The configuration of the XC886 peripherals like UART and SSC are done by using the Dave development tooling from Infineon. The Dave tool provides a nice and easy way to setup the registers within the Microcontroller without digging deep into the user manual of the XC886.

The register configuration of the TDA5340 can be done by using the TDA5340 Explorer Tooling which provides a graphical description of the functionality and generates the modified SFR register set in form of a header file which is used within this Software Example.

The visualization on the host PC can be done by using a simple hyper terminal (in our case putty), which is configured to monitor the communication of the emulated COM interface.

The TDA5340 evaluation board includes an interface board (UWLink) which acts as a SPI to USB bridge. This interface board simply consists of a XC886 Microcontroller and FTDI chip which takes over the job of the USB interface handling. Further information of using the UWLink main board as a first development platform can be found on our web site **http://www.infineon.com/UWLink** within the “UWLink Mainboard - User Manual”.

The software example supports the following features:

- Basic SPI Read/Write commands
  - SPI read SFR
  - SPI write SFR
  - SPI write TX FIFO
  - SPI read RX FIFO
- Initialization of the TDA5340 using SPI write and read
- TDA5340 Power up sequence
- TDA5340 Main mode selection
- RF transmission using TX FIFO mode
- Output power selection
- Basic command I/O using the UART communication
- RF link analysis

This document describes the general file structure, the program flow, the initialization of the TDA5340 and the RF-protocol which is used in the TDA5340 Software Example.

2 File structure

This chapter gives an overview over the file structure of the TDA5340 Software Example and describes the functionality implemented in the source files. **Figure 1** shows how the files are linked.

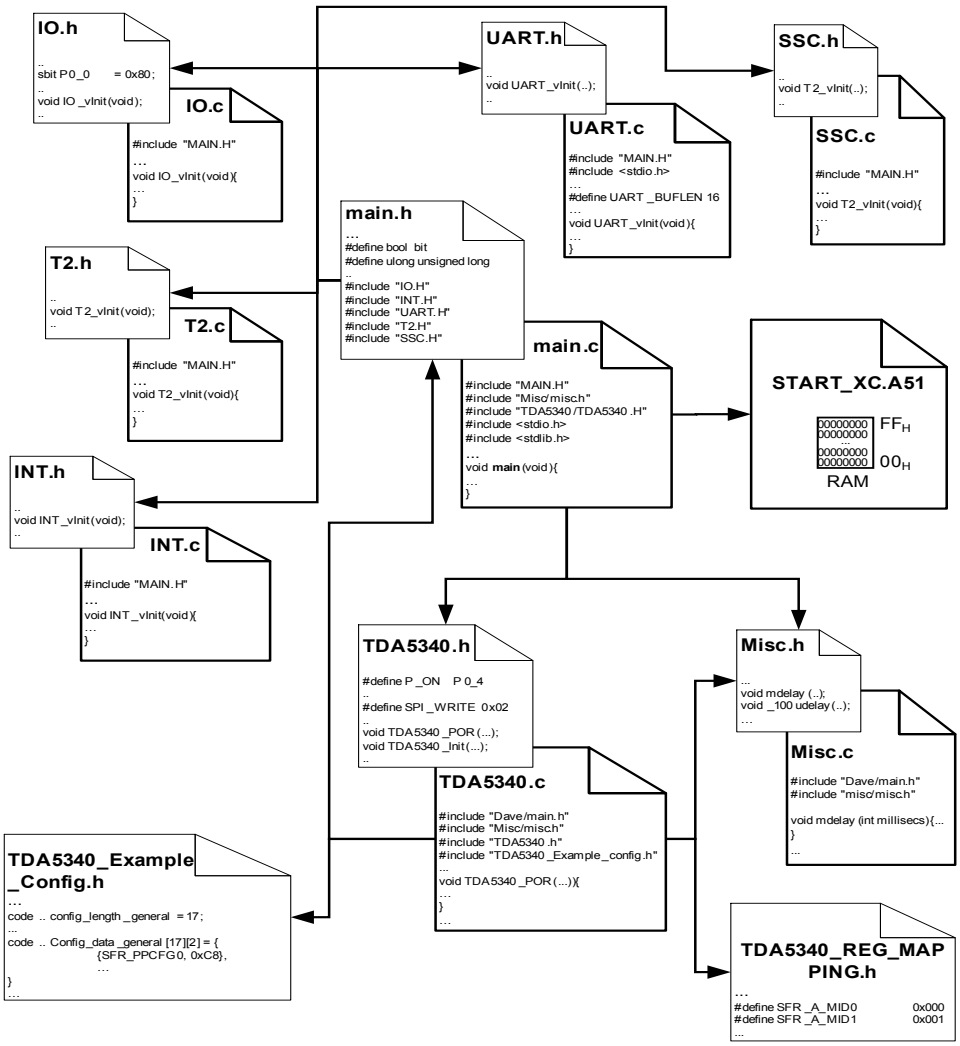

Figure 1 File Structure of TDA5340 Software Example

2.1 Header files

In the header files the interfaces to different modules are defined.

2.1.1 main.h

Within the main header file the low level driver header files for the µC peripherals are included. In addition the SFR registers for the µC are described in this file.

2.1.2 SSC.h

The SSC.h describes the interface to the synchronous serial interface of the µC.

2.1.3 UART.h

The UART.h describes the interface to the UART interface of the µC.

2.1.4 IO.h

The IO.h describes the interface to the Input and Output ports of the µC.

2.1.5 T2.h

The T2.h describes the interface to the Timer 2 Unit of the µC.

2.1.6 INT.h

The INT.h describes the interface to the Interrupt generation unit of the µC.

2.1.7 TDA5340_REG_MAPPING.h

This is the register definition file for TDA5340. Here all SFRs (Special Function Registers) of TDA5340 are defined. This file has to be added to the project.

2.1.8 TDA5340.h

TDA5340.h is the interface to the TDA5340.c low level driver file. The prototypes of the Low level driver and some declarations for the RF-Transmission and Main mode selection are defined here. This file has to be added to the project together with *TDA5340.c*. All Functions of the TDA5340.c low level driver are described in detail in **"TDA5340.c" on Page 8**.

2.1.9 TDA5340_Example_Config.h

Within this interface file the definitions of the registers which are derived out of the TDA5340 Explorer tooling are declared.

2.1.10 Misc.h

The interface to functions:

- void mdelay(int milisecs)
- void 100udelay (int 100useconds)

2.2 Source files

The source files include the implementation of the start up file, the initialization of TDA5340, µC peripherals, and the functions used for UART communication, RF framing and transmission.

2.2.1 START_XC.A51

Startup Code for the Infineon XC8xx devices.

2.2.2 main.c

This file includes the *main() and Init_check_TDA5340()* function of the TDA5340 Software Example and is executed after the Startup function.

The following global variable indicates a external interrupt from the TDA5340:

- *bool* TDA5340_int: flag which indicates an interrupt on NINT line produced by the TDA5340.

2.2.2.1 Init_check_TDA5340()

This routine resets and initializes the TDA5340. If an error is found this is indicated by sending a UART message.

2.2.3 Misc.c

The functions *mdelay() and _100udelay* are implemented within this file.

- mdelay(): performs a dealy on milli second granularity
- _100udelay: performs a delay 100 micro second basis

2.2.4 TDA5340.c

The following functions are implemented in this file:

- TDA5340_POR()
- TDA5340_Init()
- TDA5340_Set_Mode()
- TDA5340_get_IS()
- TDA5340_Change_Page()
- TDA5340_select_TX_channel()
- TDA5340_set_TX_power()
- TDA5340_get_RX_fifo_data()
- TDA5340_get_RX_Link_Info()
- TDA5340_config_IOs()
- TDA5340_start_TX_FIFO()
- TDA5340_wait_and_check_EXTINT()
- TDA5340_exit_DeepSleep_mode()
- TDA5340_enter_DeepSleep_mode()
- TDA5340_write_TX_fifo()
- TDA5340_read_RX_fifo()
- TDA5340_getReg()
- TDA5340_putReg()

2.2.4.1 TDA5340_POR()

This function will perform a power on reset of the TDA5340 and will save within the STATUS section of the TDA5340_struct the success or error messages.

2.2.4.2 TDA5340_Init()

This routine initializes the TDA5340 Configuration registers depending on the selected configuration.

2.2.4.3 TDA5340_Set_Mode()

This function sets the main Mode of TDA5340 and writes in the TRX_status structure the selected mode.

Expected Modes:

- SLEEP_MODE
- POWERDOWN_MODE
- DEEPSLEEP_MODE
- RX_SLAVE_MODE
- RX_SP_MODE
- TX_MODE

If the RX_SLAVE_MODE or the TX_MODE is selected the related configuration section must be also specified (CONFIG A-D)

2.2.4.4 TDA5340_get_IS()

This routine reads all Interrupt status registers of the TDA5340 and writes the content into the TRX_status structure.

2.2.4.5 TDA5340_Change_Page()

If a register access to a different page of the TDA5340 is initiated a change of the register page must be done prior to the intended SFR access. This function will change the address space to the page which is passed to this function.

2.2.4.6 TDA5340_select_TX_channel()

Within this function the selection of the pre-configured frequencies for transmit mode is done. If this function is not called the base frequency which is defined in the SFR set of the TDA5340 will be used.

2.2.4.7 TDA5340_set_TX_power()

This routine sets the number of used power amplifier stages. The more stages are used the lower the “ON” resistance and therefore the higher the output power delivered to the antenna.

2.2.4.8 TDA5340_get_RX_fifo_data()

This function reads the RX FIFO until the FIFO is empty.

2.2.4.9 TDA5340_get_RX_Link_Info()

Read out SFR register set which describes the RF link (RSSIPPL, RSSIRX, Channel information) and place it into TRX_status struct.

2.2.4.10 TDA5340_config_IOs()

This routine sets the function of the general purpose IOs of the TDA5340.

2.2.4.11 TDA5340_start_TX_FIFO()

This function starts the transmission with the given TX_data.

Note: TDA5340 must be in transmit mode before function call otherwise no action is done. Also make sure that FIFO transmission start with FIFO not empty is selected in the TDA5340 register set (default).

2.2.4.12 TDA5340_start_TX_FIFO_start_bit()

This routine waits until an external interrupt occurred on the µC interrupt port, after occurrence of the interrupt the interrupt status registers of the TDA5340 are read out. The status register is compared to the expected value and waits again if not expected interrupt occurred otherwise returns OK.

Note: Attention this is done only for IS Interrupt register of the TDA5340

2.2.4.13 TDA5340_exit_DeepSleep_mode()

This function will bring the receiver back from deep sleep mode to sleep mode.

Note: System Ready Interrupt must be enabled in the TDA5340 configuration

2.2.4.14 TDA5340_enter_DeepSleep_mode()

This routine enters deep sleep mode and enable SYS ready interrupt to leave Deep Sleep Mode correctly.

2.2.4.15 TDA5340_write_TX_fifo()

This function performs a SPI write to TX FIFO buffer.

2.2.4.16 TDA5340_read_RX_fifo()

This routine performs a SPI read of the RX FIFO buffer.

2.2.4.17 TDA5340_getReg()

This routine performs a TDA5340 register read.

2.2.4.18 TDA5340_putReg()

This routine performs a TDA5340 register write.

2.2.5 Timer 2 functions (T2.c)

Within this file the initialization function for the timer module 2, T2_vInit() is located.

2.2.6 IO ports functions (IO.c)

The IO_vInit() function is located within this source file. All Port pins of the XC886 are programmed which are necessary to operate the TDA5340.

2.2.7 Synchronous Serial Interface (SSC.c)

This source file includes following functions:

- SSC_vInit()
  - Initialization of the SPI interface to the TDA5340
  - Set alternate function for Port pins for SPI
  - Set SPI data rate to 1000kBaud/s
- SPI_sendByte()
  - Send byte over SPI
- SPI_readByte()
  - Read a byte over SPI
- SSC_vGetData()
  - read SPI buffer

2.2.8 Interrupt functions (INT.c)

This source file includes following functions:

- INT_vInit()
  - Pin P0.5 (input)
- INT_viExt0(void) interrupt EXT0INT
  - Interrupt callback function is setting global variable TDA5340_int
- Disable_Ext_INT()
- Enable_Ext_INT()

2.2.9 UART.c

This source file includes following functions:

- UART_vInit()
  - Pin selection for UART functionality (P1.1 and P1.0)
  - Mode 1: 8-bit data, 1 start bit, 1 stop bit

  - baudrate = 19.2308 kbaud
- UART_vIIsr(void) interrupt UARTINT
  - UART Interrupt callback function
  - if read interrupt write character into UART_RxBuffer
- UART_ubGetData8()
  - Return the received data byte
- UART_RxBufferGetByte
  - Get last received byte from RxBuffer and then clear (prevent reading twice).

3 Program flow

Figure 2 on Page 11 shows the program flow of the TDA5340 Software Example.

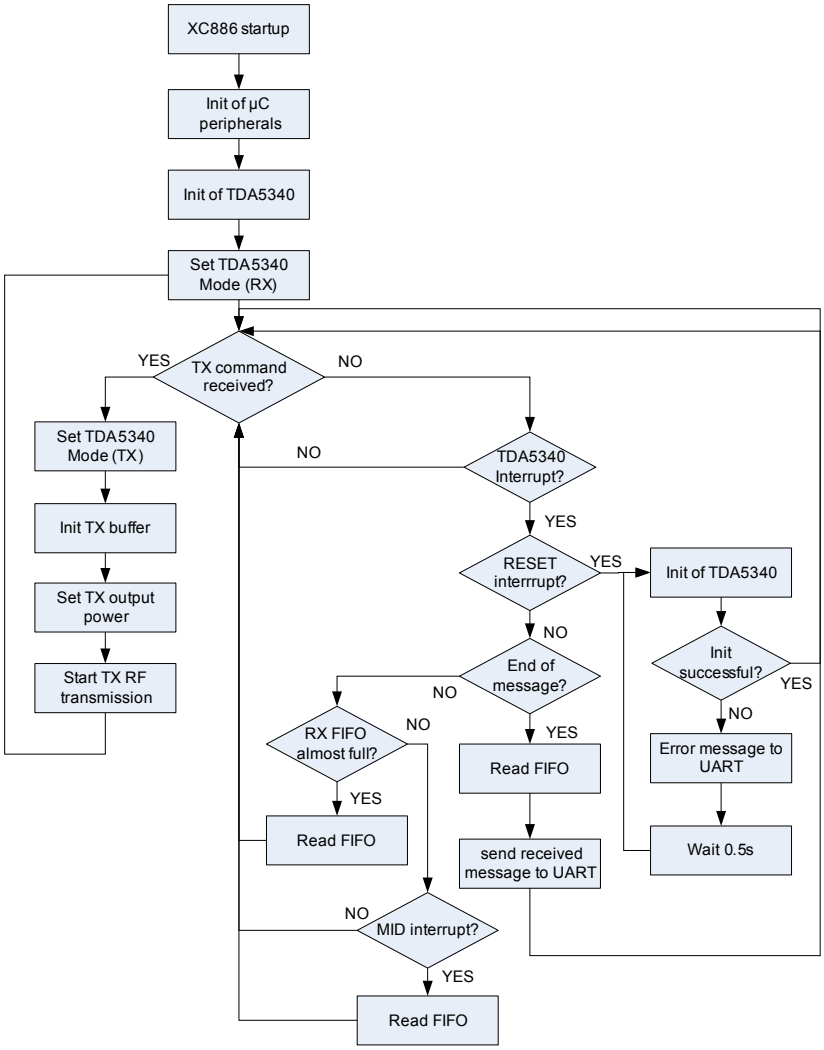

Figure 2 Program flow of the TDA5340 Software Example

The device starts program execution within the startup file. After RAM initialization the initialization of the peripheral modules of the micro controller are done.

After initialization of the µC peripherals the TDA5340 is initialized and brought into receive mode.

If no interaction from the UART interface the application will always stay in receive mode and is awaiting the configured RF protocol.

If a character ‘T’ is received from the UART interface the program will change the main state of the TDA5340 to transmit mode, initializes the transmit buffer and constructs the RF frame. For example purposes the output power is increased by one step each time the transmit cycle is performed. The output power is increased until the maximum is reached and will be set for the next transmission to minimum power.

If an external interrupt is received from the TDA5340 (indicated by a flag in the interrupt service routine) the source of the interrupt has to be evaluated by reading the Interrupt status registers (calling TDA5340_get_IS()).

There are several sources which are of interest and processed by the micro controller:

RESET Interrupt

The TDA5340 got either a brownout reset or the P_ON was set to low. Both events need an initialization of the TDA5340 to be able to operate in a wanted mode and frequency. If the re initialization is not successful an error message will be generated and after 500ms the re initialization will be started again.

End of Message (EOM) Interrupt

If the whole data frame is received the EOM interrupt is generated and the data is read out of the RX FIFO using the function TDA5340_get_RX_fifo_data. After reading the received data from the RX FIFO the first four bytes are written to the UART interface.

RX FIFO Almost Full Interrupt

If the RX FIFO almost full interrupt occurred the RX FIFO is read out to prevent a RX FIFO overflow. This decision path is very important if the expected data is larger than the RX FIFO size (36 Byte).

Message Identifier (MID) Interrupt

If the Message identifier feature of the TDA5340 is enabled the match to a pre configured ID can be signalized to micro controller via an MID interrupt. After this interrupt the receiver portion is sure that an already “known” opposite has started to open a communication channel. It is common practice to place the Data Link layer information at the beginning of the payload together with the ID. By reading out the ID after the MID interrupt also the corresponding Data Link layer data (like data payload length...) can be read out from the FIFO.

If the interrupt is handled the program will search again for UART messages and occurred TDA5340 interrupts.

4 RF-Protocol

The TDA5340 Software Example is designed to be compatible with 2kBit EU protocol example defined in the Application Note “TDA5340 - Protocol Examples for ISM band applications”. The protocol is only modified that the receiver is in Run Mode slave and therefore no Wake up pattern is sent.

The following settings are used for RF-Transmission:

- **Encoding:** Manchester
- **Modulation:** FSK
- **Baudrate:** 2000 bps
- **Frequency:** 433.92 MHz

The RF-Frame starts with a RUNIN sequence of 8 manchester coded data bits (16 chips) which are used by the receiver for internal filter setting and frequency adjustment. Then the 16 chips long TSI (Telegram Start Identifier) follows, which is used to synchronize the frame and detect the exact start of a data frame (payload). To detect the EOM (End of Message) a constant length of 88bit is used. See figure below:

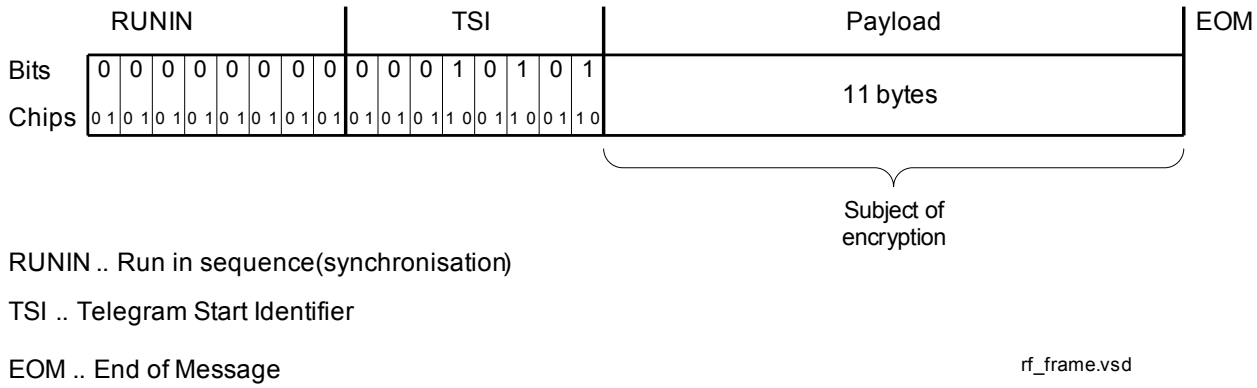

Figure 3 RF-Frame

w w w . i n f i n e o n . c o m

Published by Infineon Technologies AG