Hitex Germany
– Head Quarters –
Greschbachstr. 12
76229 Karlsruhe
Germany

☎ +049-721-9628-0
Fax +049-721-9628-149
E-mail: Sales@hitex.de
WEB: www.hitex.de

Hitex UK
Warwick University
Science Park
Coventry CV47EZ
United Kingdom

☎ +44-24-7669-2066
Fax +44-24-7669-2131
E-mail: Info@hitex.co.uk
WEB: www.hitex.co.uk

Hitex USA
2062 Business Center Drive
Suite 230
Irvine, CA 92612
U.S.A.

☎ 800-45-HITEX (US only)
☎ +1-949-863-0320
Fax +1-949-863-0331
E-mail: Info@hitex.com
WEB: www.hitex.com

Embedding Software Quality

Application Example

for

STM32-comStick / STM32F107VC

This documentation describes an application example for the Hitex STM32-comStick debugger system with the integrated STM32F107VC microcontroller from STMicroelectronics.

Architecture:	ARM Cortex M3
Controller:	STM32F107VCT6
Chip Manufacturer:	STMicroelectronics
Peripherals:	LEDs, SYStick, bitbanding
Compiler:	GNU Cortex v.4.4.0

Author:	Application / Hitex Germany
Contact:	applications@hitex.de
Revision:	07/2009 - 001

Preface

In order to support you when working with our products, we provide documents containing specific examples, additional topics, special hints, detailed procedures etc.

This document explains features and functions of an application example to run with the HiTOP user interface.

For more information on the current software and hardware revisions, as well as our update service, please contact www.hitex.de, www.hitex.co.uk or www.hitex.com.

Contents

1 Scope and Disclaimer

This software is a simple example using the SYStick and bitbanding features on the ARM-Cortex-based STM32F107 controller by STMicroelectronics. The software includes the hardware-related driver and definitions as well as the standard library used in the application.

Subject Matter

The design is made to show the implementation and configuration of the STM32F107 core on the STM32-comStick. Basic software is the GPL licensing model, laid open on the corresponding WEB Sites. The implementation is 'as it is' and no warranties are made. The use and the change of the code is free and not under control of the originators.

Requirements

- Hitex Cortex-GNUTool chain GCC 4.4.0 for STM32-comStick
- STM32-comStick device
- STM32F107VCT6 controller

Features and Components

- Platform: STM32F107 Cortex core
- STM32-comStick device hardware
- GCC Cortex compilation tools version 4.4.0
- LIBs (Thumb2 / GNU Cortex compiler)
- LED peripheral is used

Procedure

Installing

1. Step: Install the Hitex ARM-GNUTool chain for STM32-comStick.
2. Step: Plug the STM32-comStick device into a USB port.

Starting

1. Step: Copy the project data to the working directory.
2. Step: Load the project 'Systick\HiTOP\STM32-comStick\project.htp' using the **Project > Open** command of the HiTOP user interface.
3. Step: Perform a 'build target' using the **Project > Build** command if object code is not available. **Download** the application to the STM32-comStick.
4. Step: Perform a 'target reset' using the **Debug > Reset** command and run the project with the pre-configured settings.

Customizing

1. Step: Change the SYStick preload value to change the frequency of the SYStick interrupt occurrence in the 'main.c' file by changing the SysTick_SetReload() in line 243.
2. Step: Change the signalling LED by changing the code SET_LED_BIT and CLR_LED_BIT used in the ToggleLED routine. These routines are macros defined in the line 54 and following. The LEDs available are on port B pin0, pin1, pin5, pin9, and on port E pin15. A first try can be made changing the numbers in the macros line 66 and 67 to 0, 5 or 9.
3. Step: Recompile the project, download to the target hardware and run.

2 Software and Documentation

For more information on the STM library, see the related documents on http://www.st.com/mcu/. For more information on the CMSIS system, see the related documents on http://www.arm.com/news/23722.html. The current version 1.10 - 24. Feb. 2009 and the library version 3.0 from STMicroelectronics offers basic functions of the CMSIS implementation. Other functions can be applied on this basic software.

3 Used Modules

This chapter describes source files used to run the application.

3.1 C Sources

All sources are written in pure C following the ANSI rules nearly strictly. For all sources originated by STMicroelectronics the originator is responsible.

startup.c	C file for controller start and RAM memory setup
main.c	Setup routines and main loop
stm32f10x_it.c	Interrupt vectors
stm32f10x_lib.c	Library definition module
stm32f10x_gpio.c	Library for GPIO module
stm32f10x_rcc.c	Library for RCC module
stm32f10x_systick.c	Library for SYSTICK module
stm32f10x_nvic.c	Library for NVIC module
cortexm3_macro.s	Supporting macros and core functions

3.1.1 main.c

The main program (main.c) provides the following routines:

Main()

The main routine contains calls for all related hardware setup routines (**RCC_init(), IO_init()** and **SYSTICK_init()**). The startup code in the file 'startup.c' initializes no hardware components like PLL and clock. It only initializes the RAM with the data segment.

The hardware initialization (RCC_init() configures the PLL and enables the peripheral clocks of the GPIO modules. IO_init() enables the LED pins and SYSTICK_init() configures the systick module.

The following 'while loop' is not relevant for further program execution.

RCC_init()

The RCC_init uses the library functions of the stm32f10x_rcc library module. PLL is set up and the GPIO modules' clocks are enabled.

IO_init()

Set the LED pins to 'output' and 'push-pull' mode with high current enabled.

SYSTICK_init()

The SYSTICK module of the core is setup here. Therefore the clock source is enabled, the reload value is set and the interrupt is enabled. The SYSTICK module will generate an interrupt after counting down from the reload value. This interrupt will call the **ToggleLED()** routine

ToggleLED()

This routine toggles the variable "toggle" and calls the corresponding macros writing to the bitbanding area of the IO port. This is how bits can be set or reset in a simple way.

3.1.2 startup.c

One of the main aspects on Cortex-M3 controllers is the usage of appropriate startup code in C. The basic memory initialization is performed via this file.

3.1.3 stm32f10x_it.c

This module includes the interrupt vectors which are referenced in the vector table of the Cortex-M3. The NVIC module controls most of the sources but not the system interrupts including the SYSTICK events.

4 Header Files

main.h	General inclusions
stm32f10x_conf.h	Library configuration file
stm32f10x_lib.h	Library inclusions file
stm32f10x_type.h	Definitions and types
stm32f10x_it.h	Interrupt vector pre-declarations

5 Customizing

- The software provides all library functions of the ST library. To activate the functions, the file **stm32f10x_conf.h** must be edited. The configurations and options are described in the related documents of the library. In this software only the necessary library files are used.
- The added library sources must be added to the FWLib source tree in the project workspace view. If added, the sources are compiled and linked automatically.

6 Compiler-Specific Keys and Options

This chapter describes the compiler-related specific keys and options to handle the code with the GCC compiler.

6.1 __attribute__ ((section("vector")))

The attribute **__attribute__ ((section("vector")))** is used to build the initial interrupt vector table necessary for the Cortex architecture.

6.2 -mcpu=cortex-m3

The attribute **-mcpu=cortex-m3** enables the compiler to generate thumbs code for the Cortex-M3 cores.

6.3 -gdwarf-2

This attribute forces the compiler to generate debug information. This option is necessary for debugging but should be changed for release code.

6.4 -O0

This attribute forces the compiler to generate an un-optimized (optimization level 0) code. This option is best for debugging but should be changed for release code.

6.5 --gc-sections

This attribute forces the compiler not to generate eh_frame sections in the Flash memory. The section may occur in the map file but will not use more than 0 (zero) bytes of memory.

■