STM32

From the Mikrocontroller.net collection of articles, with contributions by various authors (see version history)

STM32 is a microcontroller family from ST with a 32-bit ARM Cortex-M0 / M3 / M4 CPU. This architecture has been specially developed for use in microcontrollers and thus largely replaces the previous ARM7-based controllers. The STM32 is available from ST in countless versions with variable peripherals and different housing sizes and shapes. Due to the small chip area of the core, ST is able to offer a 32-bit CPU for less than € 1.

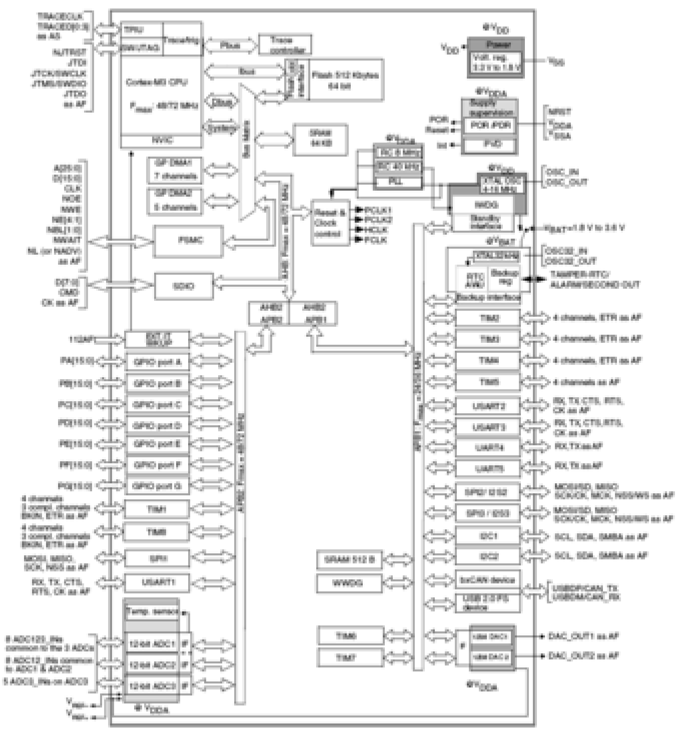
Block diagram STM32F103xC / D / E

table of contents

STM32 families [edit]

So far there are eleven STM32 families:

- STM32F0
  - Cortex M0

  - Microcontroller to get you started
  - Up to 48MHz (38 DMIPS)
- STM32F1
  - Cortex M3
  - Up to 72MHz (61 DMIPS)
  - Different subfamilies:
    - Connectivity line
    - Performance line
    - USB access line
    - Access Line
    - Value line
- STM32F2
  - Cortex M3
  - Up to 120MHz (150 DMIPS)
  - Like the STM32F1 series, camera interface, 32-bit timer, crypto engine ...
- STM32F3
  - Cortex M4F
  - DSP and FPU
  - Up to 72MHz (90 DMIPS)
  - Fast 12-bit 5 MSPS and precise 16-bit sigma-delta ADCs
  - Touch sensing controller (TSC)
- STM32F4
  - Cortex M4F
  - DSP and FPU
  - Up to 180MHz (225 DMIPS)
  - Up to 2MB flash
- STM32F7
  - Cortex M7
  - DSP and FPU (Single / Double Precision)
  - Up to 216MHz (462 DMIPS)
  - More peripherals: SPDIF-IN / OUT, SAI, HDMI-CEC, Dual Quad SPI
  - On-chip graphic LCD controller
  - DMAs also for Ethernet, USB and Chrome ART
- STM32H7
  - Cortex M7
  - Up to 400MHz (856 DMIPS)
- STM32L0
  - Cortex M0 +
  - Low power
  - with LCD driver
  - Up to 32MHz (26 DMIPS)
- STM32L1

  - Cortex M3
  - Low power
  - with LCD driver
  - Up to 32MHz (33 DMIPS)
- STM32L4
  - Cortex M4F
  - DSP and FPU (Single Precision)
  - Ultra Low Power (up to 8nA with I / O wake-up)
  - Up to 80MHz (100 DMIPS)
  - 128KB ... 1MB Flash, 64 / 128KB SRAM
  - optional segment LCD driver
  - Quartz-free operation also possible with CAN (1% ex works) or USB (synch via host)
  - Digital filter for ΣΔ modulators
- STM32G0
  - Cortex M0 +
  - Up to 64MHz (approx. 60MIPS)
  - 16KB ... 512KB Flash, up to 128KB SRAM
  - 8-100 pins
  - SO-8, QFP TW in 0.8mm grid
  - Not all listed variants are in production yet (8.2019)
- STM32G4
  - Cortex M4F
  - FPU and DSP
  - "Math Accelerator" For trigonometric functions, FIR, IIR
  - Up to 170MHz (213 DMIPS)
  - up to 512KB Flash, 128KB SRAM
  - 48-128 pins
  - "Mixed Signal MCU"
  - Not all listed variants are in production yet (8.2019)

- STM32WB
  - Cortex M4 + Cortex M0 +
  - 64 MHz (M4), 32 Mhz (M0 +)
  - IEEE 802.15.4
  - Bluetooth 5.0

- STM32T - no longer in production
  - Cortex M3
  - 72MHz
  - Touch sensing
- STM32W - no longer in production

- Cortex M3
- UP TO 24MHz
- RF MCU

Here is an overview of how to select an STM32Fxxx

Features [edit]

- Cortex-M0 (+) / Cortex-M3 / Cortex-M4 (F) / Cortex-M7 core (with FPU)
- 16KB ... 2MB Flash-ROM
- 4KB ... 512KB SRAM
- 2KB ... 16KB EEPROM (STM32L)
- SDRAM controller for the STM32F42xxx and STM32F43xxx , up to 512 MByte external SDRAM addressable
- 512 one-time programmable bytes (STM32F2 / 4)
- Housing 8 ... 216 pins as SO, LCSP, TSSOP, QFN, QFP and BGA
- **Over 700** STM32 derivatives / variants are currently available
- Up to 72MHz CPU clock, up to 120MHz for the STM32F2xx, up to 168/180 MHz for the STM32F4xx, whereby a special prefetch hardware achieves a speed of up to 120/168 MHz that corresponds to 0 wait states. The CPU clock is derived from the internal RC clock or an external quartz clock via a multiplier. Up to 216MHz CPU clock with STM32F7xx.
- Additional FPU "Math Accelerator" for Trigonom. Functions (CORDIC), FIR, IIR (FMAC) (STM32G4)
- External bus interface (only for housings from 100 pin and only for STM32F4, STM32F2 and STM32F1 Performance line)
- LCD driver for up to 8x40 segments (not for the STM32F2xx)
- TFT driver for STM32F429 / STM32F439 STM32F469 / STM32F479
- Voltage range 1.65 ... 3.6V, only one operating voltage required
- Temperature range up to 125 ° C
- Up to 168 IOs, many of them 5V tolerant
- Internal, calibrated RC oscillator with 8MHz (16MHz for STM32F2 / F4xx)
- External quartz
- Real time clock with its own quartz and separate power supply
- Up to 16 timers , each timer up to 4 IC / OC / PWM outputs. Including 2x motion control timers (with STM32F103xF / G), (up to 32 PWM outputs)
- Systick counter
- Up to 3 12-bit AD converters with a total of 24 AD inputs, integrated temperature sensor , reference voltage Vrefint and VBatt voltage measurement (STM32F4xx)
- Up to 2 12-bit DA converters (up to 3 for the STM32F3xx)
- Up to 2 DMA controllers with up to 12 channels (16 for the STM32F2 / 4xx)
- Up to 2x I²C
- Up to 5x USART with LIN, IrDA and Modem Control (up to 8 for the STM32F2 / F4xx)

- Up to 3x SPI (up to 6 for the STM32F4xx)
- Up to 2x I²S
- Up to 3x CAN
- Hardware CRC unit, with the STM32F3xx series with an adjustable polynomial
- Unique device ID register (96 bits)
- TRNG - True Random Number Generator (STM32F2 / 4xx), based on an analog circuit
- Cryptographic Processor (CRYP) (STM32F2 / 4xx)
- Hash Processor (HASH) (STM32F2 / 4xx)
- Camera interface (DCMI) (STM32F2 / 4xx)
- USB 2.0 full speed / OTG
- USB 2.0 Hi Speed OTG with extra PHY chip (STM32F2 / 4xx)
- USB Type-C ™ Power Delivery controller (STM32G0 / G4)
- HDMI CEC interface (STM32G0)
- SDIO interface (e.g. SD card reader)
- Ethernet
- Watchdog with window mode
- Each peripheral module is switched separately, which significantly save electricity can
- JTAG and SWD (Serial Wire Debug) interface
- Up to 6 hardware breakpoints for debugging
- and much more ...

Structure of the documentation [edit]

The documentation of the STM32 is more extensive and complex compared to the AVR family. It is divided into several documents. The STM32F103RC should be named as an example of the documentation . The ST page contains all the necessary information for this processor.

These documents from ST describe the controller:

- The STM32F103xC / D / E datasheet describes the special properties of a certain model series and lists the exact data and pinouts, as well as the assignment of chip name to flash / RAM size. The peripheral modules are only listed, not described in detail.
- In the Reference Manual (RM0008) all peripheral modules of the respective STM32 family of controllers are described in detail.
- The ARMv7M Architecture Reference Manual describes in detail the abstract ARMv7M architecture, such as the exception model, the CPU instructions including encoding, etc.
- The Cortex-M4 Technical Reference Manual or the Cortex-M3 Technical Reference Manual describes properties of the Cortex-M3 / 4 implementation of the architecture, in particular the speed of the individual processor instructions.
- The STM32 Cortex-M3 Programming Manual is a summary of the ARMv7M Architecture Reference Manual related to the STM32.

- If you do not use the ST firmware library, you also need the Flash Programming Manual for the operating mode of the Flash ROM, ie the frequency-dependent configuration of the wait states.

In addition, the errata sheets should also be observed. The app note " AN2586 Getting started with STM32F10xxx hardware development " is also recommended . The respective documentation PDFs are linked on the ST product page of each microcontroller.

Hardware access libraries [edit]

CMSIS [edit]

The CMSIS (ARM® **C** ortex ™ **M** icrocontroller **S** oftware **I** nterface **S** tandard) is a library of ARM to access the vendor-independent functions of the ARM core. In the case of the Cortex-M4F cores, this also includes the DSP and floating point functionality. There are also a number of helper functions for the NVIC, the Sys-Tick-Counter, and a SystemInit function that takes care of the PLL.

The header files have been standardized within the framework of the CMSIS standard (www.onARM.com), the registers are accessed via **Peripheral-> Register** . The CMSIS C files or headers also contain adaptations for the various compilers. The porting of a real-time operating system should be possible in a greatly simplified manner using the CMSIS for chips from the various manufacturers (e.g. uniform addresses for core hardware / system tick counter).

The CMSIS is included in the download of the STM32 Standard Peripheral Library. The compiler manufacturers deliver a library (including CMSIS) that is suitable or tested for their tool version. In contrast to the downloads from the chip manufacturer, these libs can also contain older versions.

STM32 Standard Peripheral Library (SPL) [edit]

ST offers an extensive peripheral library suitable for the CMSIS for every controller family. All functions for using the periphery are encapsulated in simple structures and function calls. So you don't have to worry about the peripheral registers yourself. This library and its documentation require a basic understanding of the function of the respective peripheral module, as conveyed by the above reference and various app notes. The library also contains several examples for almost every peripheral. There is an extra library for the USB interface, as well as for Ethernet.

The Standard Peripheral Library is now out of date, ST recommends that you no longer use it.

The library for the respective controller can be downloaded from the "Design Resources" page of the ST product page of each STM32 microcontroller, e.g. here for the above-mentioned STM32F103RC .

Library for STM32F4xx: STSW-STM32065 STM32F4 DSP and standard peripherals library

STM32 Cube HAL [edit]

Has replaced the SPL since 2012.

- https://www.st.com/en/embedded-software/stm32cube-mcu-packages.html

Programming [edit]

There are several ways to program the STM32, both commercial proprietary and free software.

Free software / freeware [edit]

Put it together yourself [edit]

You take...:

- A development environment of your choice:
  - Eclipse with C / C ++ Development Tooling and GNU ARM plug-in (Linux, Windows)
  - TrueStudio for STM32 is based on Eclipse
  - System Workbench for STM32 is based on Eclipse
  - Netbeans with GDBserver plug-in (Linux, Windows)
  - KDevelop (Linux, Windows)
  - Geany (Linux, Windows)
  - Or a simple text editor
- A C, C ++ compiler:
  - One of the GCC binary distributions , see also GCC (depending on the distribution Linux, Windows)
- Programming software for flashing the target:
  - OpenOCD supports many debug / programming adapters (Linux, Windows)
  - Texane stlink works well with the ST-Link adapters like them. can be found on the STM32 Discovery Boards (Linux)
  - When using a Segger J-Link, the Segger GDB server in connection with the GDB supplied with the GCC (Linux, Windows).
  - Black Magic Probe as microcontroller firmware simulates a serial port that can be used directly by GDB. For FTDI-MPPSE based adapters and ST-Link V2, Blackmagic runs on the host and provides port: 2000 for GDB. If you can get programming access to the STM32F103 of the STlink, STlinks can also be reflashed with BMP firmware.

Complete IDEs [edit]

- ARM mbed Developer Site is a complete development platform for various ARM controllers based on an RTOS with hardware abstraction and web-based online and offline IDE. Similar to the Arduino concept, simple tasks can be implemented quickly with mbed. mbed is based on C ++ and supports various compilers. Projects can also be exported and downloaded for other IDEs. The mbed library is open source and hosted on github.

- Atollic TrueStudio has been reduced to STM32 microcontrollers since it was acquired by ST and is now available free of charge. Based on Eclipse, OpenOCD and ARM GCC . Without size limit.
- Codesourcery Lite Edition You have to be able to make friends with this environment. There are only a few sample projects available. No longer available for free.
- Coocox Eclipse IDE free IDE for STM32F0 / F1 / F2 / F3 / F4, which is no longer further developed. Based on the ARM GCC and widely supported. There is even a free RTOS available. A good choice with no limits with broad debugger support. There is helpful information here and here in the forum, article: STM32 CooCox installation
- emIDE free IDE from Segger. The emIDE is based on Code :: Blocks. It is based on ARM GCC and supports a large number of different JTAG / SWD debuggers - including the J-Link from the same company, of course.
- EmBlocks free IDE, Code :: Blocks based, supports STM32 L1 / F0 / F1 / F2 / F3 / F4 / W, integrated compiler (ARM GCC), integrated GDB debugger, Jlink / ST-Link, system view (peripheral register show) when debugging, Project-Wizard (own wizards can be written with Squirrel), based on Code :: Blocks. Article: STM32 - Getting started with Em :: Blocks
  - is now called EmBitz -> https://www.embitz.org
- Development environment GNU / Linux for STM32F1 with OpenOCD and Olimex ARM-USB-OCD-H, operation via Eclipse IDE or command line.
- System Workbench for STM32 (SW4STM32) is an unrestricted and free IDE. It is officially supported by ST . Version 1.0 of the development environment has been available since 5.2.2015. A version for Linux has been available since February 2016.

Other programming languages [edit]

- Mecrisp-Stellaris , a native Forth implementation for ARM Cortex M0 / M3 / M4. Several STM32 targets are already supported and new ports are very welcome. Chips from TI, NXP and Freescale are also included in the current package.

Commercial environments [Edit]

- Wedge µVision(Demo max. 32KB Code / Free for STM32F0 / STM32L0): In addition to the ARM compiler, the very convenient µVison IDE can also be configured for any GNU compiler using a menu. This means that the 32k limit only applies to the integrated debugger / simulator. In connection with a ULINK (ULINK2, ULINK pro for trace, ULINK Plus for current measurement) the environment is very easy to use, the compiler supports parallel build. With µVision you can write external ELF or HEX files into the controller or into the controller's flash memory (a bit tricky, see "Options for Target"). The IDE supports the graphic analysis of different data in a common display. A good choice for the beginner. However, the price is a good reason to switch to other free IDEs.download If you limit yourself to the STM32 Cortex M0 / L0, you can also use the Keil MDK freely without 32K limitation. download
- IAR Embedded Workbench (demo max. 32KB code) download

- winIDEAOpen No code limitation, GCC and test tool included. Runs with the iTag50 adapter, Segger J-Link and the ST-Link
- Raisonance Ride7 (GCC Compiler, free version limited to debugging of max. 32KB code, no limitation when compiling)
- Rowley Crossworks (30-day unlimited demo, $ 150 for non-commercial use, based on GCC). I don't understand why you should pay money for this IDE. The GNU compiler is free and so are the development environments based on Eclipse. However, this setting work is something for the somewhat experienced developer.
- SiSy ARM or SiSy Micrcontroller ++ (demo available, no size limit, based on GNU compiler, graphic programming with UML possible, integrated debugger)
- EPS Debugger Plugin, for STM32 Development with Code :: Blocks
- In addition to Pascal and Basic, MikroE also offers C with a complete user interface with compiler etc. pp relatively inexpensive
- VIsualGDB Anyone who comes from Atmel Studio or who otherwise works with Visual Studio will get a plug-in that is really fun and works. Not only STM32 are supported. Just look at the free trial version and try it out.

STM32CubeMX [edit]

This is software made by ST itself that simplifies the selection and configuration of STM32 microcontrollers:

- Selection of controllers or development boards with a parametric search
- graphical configuration of the pins and alternate functions (including checking for collisions - with development boards, certain pins are already preconfigured and are displayed)
- graphical configuration of the clock tree
- Generation of C code according to the graphic configuration. This only works with the new STM32CubeMX Libraries (HAL, LL), not with the old Standard Peripheral Libraries (SPL).
- Simulation of electricity consumption by selecting a wide variety of power sources and batteries

STM32CubeMX is Java-based and therefore runs smoothly on Windows, OS X and Linux. The zip file, which can be downloaded from ST , contains the appropriate installers for the individual operating systems.

Tutorials for various tool combinations [edit]

- Windows, Linux, Eclipse + Yagarto / CodeSourcery + OpenOCD / ST-Link
- Windows, Linux, Eclipse + GCC-ARM-Embedded + JLink
- Linux on STM32 (ucLinux)

- Windows
    - Eclipse
        - Windows, Eclipse, codesourcery, st-link

        - Eclipse plugin "GDB Hardware Debugging" with OpenOCD
    - Code :: Blocks
        - Windows, Code :: Blocks, STM32F4
    - STM32 with EmBlocks
        - Download EmBlocks
        - Video STM32 Project Wizzard in EmBlocks
    - Atollic TrueSTUDIO
        - Atollic TrueSTUDIO installation + demo
    - MDK-ARM Lite with settings for STM32F0 / F4-Discovery Board
        - KEIL MDK-ARM Download
        - Installation video STM32F4 Discovery Board
        - Settings STM32F0 Discovery Board Video
    - SiSy ARM, STM32
        - Download: SiSy DEMO no code size limit
        - Video example
    - Microsoft Visual Studio
        - "STM32F4-Discovery tutorial with Visual Studio"
- Ubuntu
    - Installing a toolchain for Cortex-M3 / STM32 on GNU / Linux - How-to manual, for STM32F1 under GNU / Linux with OpenOCD and Olimex ARM-USB-OCD-H. Integrated make files, linker scripts, startup code, various tools and demo project / program. Integration in Eclipse IDE or operation via command line.
    - Ubuntu, self-compiled GCC, STM32 / Cortex-M3
    - The ToolChain - automatically installing development environment with own and external drivers, supports QtCreator as IDE, flexibly expandable via shell scripts

- Tips for installing with Eclipse
- ARM assembler tutorial on Windows / Linux (English)

Programming adapter [edit]

- The ST-LINK / V2is a debugger offered by ST itself. Every STM32 Discovery- or Nucleo-Board has an ST-LINK V2 or ST-Link V2-1 for programming / debugging via SWD on-board (partially interruptible), which can also be used for its own STM32 target hardware and, in principle, other Cortex-M can be used. Although it is a very slow representative of its kind with a clock rate of 1.8MHz, it can be used to debug and flash foreign hex and binary files. The ST-LINK variant on the Nucleo or Discovery boards only supports SWD and no JTAG, whereas the ST-Link in the adapter version with housing also supports JTAG and is also available in a variant with electrical isolation. The ST-LINK / V2-1 on the NUCLEO and Discovery boards can also be converted to a J-Link OB using a software update.here . Copies of the ST-Link V2 are available as a "mini" version, among other things, very cheaply (<5 €) from Ebay, Aliexpress and Co. However, these also do not support JTAG and also have the disadvantage that the reset pin is not brought out or

the pin labeled "Reset" is only intended for STM8. All ST-Link V2 and V2 / 1 can be brought up to date with the update software offered by ST .
- SEGGER J-LINK / J-TRACE for all ARM7 / 9/11, Cortex-M0 / M1 / M3 / M4 / A5 / A8 / A9 / R4 as non-commercial J-LINK-EDU for approx. 50 € , runs in µVision, IAR, GDB (Linux & Windows via their own GDB server), ... The J-Link is by far the fastest debugger that I have been able to test so far. If you are in a hurry when debugging, the J-Link from Segger is the right choice.
- Keil ULINK-ME , ULINK2 , ULINK pro If you don't want to leave the µVision IDE, you can make friends with these adapters, because they only work with this IDE. You do not need any USB drivers, as they cleverly use the HID device of the operating system. No foreign binary or hex file can be flashed. The ULINK2 costs exactly as much as a Segger J-Link Basic with the same range of functions, but which can also be used in conjunction with other IDEs (GDB, etc.).
- Raisonance RLink
- iTag can be ordered from Amazon for € 50 , alternatively as a do-it-yourself version (open design) runs with the free winIDEAiTag version (see above)

As a rule, the JTAG adapters have a 20-pin connector that can be plugged directly into the demo boards with a 20-pin JTAG connector. The pin assignment is standardized, see article JTAG . The Discovery boards do not have a separate JTAG connector, but at least for the STM32F4 Discovery you can easily build an adapter pin header-> JTAG connector yourself.

Programming adapter open source [edit]

- ARM-JTAG-COOCOX , CoLinkEX replica from Olimex, supports JTAG and SWD
  - supported uC
  - Supported IDEs: Keil MDK-ARM 4.03 or newer, IAR Embedded Workbench 5.xx or newer as well as the CooCox CoIDE
- Olimex ARM-USB-OCD (approx. 60.-, also has a voltage output and a COM port)
- Raft JTAG

The controller also has a built-in boot loader. This means that it can also be programmed via a normal serial interface without the need for a JTAG adapter. This may require a corresponding configuration via the BOOTx pins and / or the option bytes, and a program such as stm32flash .

Demo projects [edit]

- Introduction to GPIO programming of the STM32F10x and STM32F30x processors using the example of the STM32F3 Discovery Board and comparison to the AVR IO register structure [1]
- Program example for the use of Timer2 together with the interrupt
- Printf () debugging with minimal effort
- Program example for BLDC motor control (timer 1) with HALL sensor (timer 3)
- Cortex_M3_OCM3U
- Martin Thomas has created an extensive project that contains the Eclipse settings:
  - "ChaN's FAT-Module with STM32 SPI"

- STM32 USB-FS-Device Lib
- Model transmitter based on STM32 with many drivers www.rcos.eu
- Detailed introductory tutorial in code form for the STM32F4 discovery board
- Swiss gondola control via web server on ETT STM32F ARM KIT board in Keil RTOS with webcam
- The Ethernut SVN version now supports many STM32 types, many devices and some STM32 demo boards
- Uwe Becker's Libraries for the STM32F4
- Uwe Becker's STM32F429 Discovery Board Oscilloscope , here the thread
- Uwe Becker's STM32F429 Discovery Board ZX-Spectrum Emulator
- USB tutorial with STM32 includes a complete sample project

Debug and trace interface (CoreSight ™ debug and trace technology) [edit]

Overview of both functionalities and the interfaces:
http://www.keil.com/support/man/docs/ulink2/ulink2_cs_core_sight.htm

The Coresight debug architecture enables non-invasive debugging, ie data can be read from the memory and written to it during operation (mostly) without influencing the processor.

Debugger functions [edit]

The debugger part has three functions:

- Run Control: e.g. program start, stop and single steps.
- (Program) Break Points: A program stops when the program counter reaches a certain program address.
    - The maximum number of simultaneously possible break points is limited (e.g. 6 for an STM32).
    - The number of break points is almost unlimited if a debugger supports so-called flash break points via the memory access (see below). A loaded program is reprogrammed in the flash to stop the debugger. This functionality is usually a chargeable additional feature of the debugger manufacturer.
    - Does not contain any data watch functionality, which is implemented in the trace part (DWT).
- Memory Access: Reading and writing of memory addresses.
    - This functionality does not include direct flash programming. The programming process for a flash is manufacturer-specific and must be supported by the debugger used.

Trace functions [edit]

The trace functionality is divided into three functions:

- ETM (Embedded Trace Macrocell): Optional, not every CPU has this hardware (cost factor, equipment).
- ITM (Instrumentation Trace Macrocell): A simplified trace of the core can be enabled via this channel, and "printf-like" data can be sent via ITM Channel 0 and output in the debugger.
- DWT (Data Watchpoint & Trace Unit):
  - Data watch: 4 access breakpoints (e.g. the debugger stops when the program accesses a memory or the value of a variable assumes a certain value).
  - Trace Unit: Track program progress (by reading the program counter) and interrupt calls, as well as time measurements.

Some of the trace functionalities can be addressed via the JTAG interface. The fast trace functionality (with 4 bit parallel port) is only available with the extended DEBUG + ETM interface. In contrast to the debugger part (run control, break points and memory access), trace functions are not supported by all debuggers. Debuggers with full trace functionality cost significantly more.

- Examples of trace port activations for different manufacturers: http://www.keil.com/support/man/docs/jlink/jlink_capture_tracedata.htm

Depending on the CPU manufacturer, the activation of the parallel trace port requires additional debugger macros for activation and port activation. In addition, the interface selection and setting (frequencies) in the development tool (IDE) are important in order to be able to successfully "trace" the program process.

Debug and trace interfaces [edit]

Two variants are available as a debug interface:

- JTAG : At least 6 control lines are required for this. Supports device chaining: Several connected devices can be controlled simultaneously with a debugger / programmer.
- SWD (Serial Wire Debug): Here at least 2 control lines (3 with SWO, plus GND and 3.3V). The SWD interface is usually faster and can also contain functions from the trace part (e.g. ITM, the SWO pin is required for this). Device chaining is not possible with this interface.

Standard JTAG connector assignments:
http://www.keil.com/support/man/docs/ulink2/ulink2_hw_connectors.htm

The 10-pin JTAG connector from mmvisual [edit]

mmvisual has expanded the standard JTAG interface with this pin assignment:

I've moved this part to the JTAG article . In addition, the adapter board 10-pin to standard JTAG 20-pin with TTL / V24 converter. Look here.

Hardware wiring [edit]

For operation, the STM32 only needs (minimum wiring):

- VCC 2..3.3V (depending on type)
- AVCC 2..3.3V (very important, the STM32 cannot be programmed without this voltage)
- GND
- Reset pin 100nF to GND (a pull-up resistor of approx. 40k is available internally)
- Boot pins

otherwise only a few single Cs 100nF to VCC / GND.

In order to be able to program, either the serial interface (programming via the pre-programmed bootloader) or JTAG or the SWD interface is required.

Boot modes [edit]

Different boot modes can be selected using the PINs BOOT0 and BOOT1. See Application Note AN2606 . Besides F1, newer families have a SYSCFG_MEMR register. The desired Boot0 / 1 values can be written into this register and after a core reset (! = System_Reset) the processor starts in the desired mode. A new or de-initialization of the periphery is recommended!

Boot from FLASH [edit]

Start address is loaded from 0x08000004

```
BOOT0 Lo
BOOT1 X
```

Boot from SRAM [edit]

PC start address is jumped to directly at 0x200001E0.

```
BOOT0 Hi
BOOT1 Hi
```

Since, according to the data sheet, the internal FLASH of the stm32f1x is only designed for 10,000 write operations, BOOT0 (high) and BOOT1 (high) can also be used to boot from the SRAM previously described with the debugger (JTAG / SWD). Please note:

```
Remap VTOR to the NVIC table in SRAM before triggering the first interrupt.
```

```
In order to achieve a start behavior comparable to the FLASH, it is advisable to
0xF1E0F85F to write to 0x200001E0. This implicit execution of "ldr.w pc,
[pc, # -0x01E0] "at start forces the start address of 0x20000004 to be loaded.
```

Boot from SYSMEM (RS232, CAN and USB) [edit]

PC start address is loaded from 0x1FFFF004

```
BOOT0 Hi
BOOT1 Lo
```

From F2 on there is also a SYSCFG_MEMRMR register. If you write the value for "System Flash" here and do a core reset (not a system reset), you also end up in the bootloader, regardless of the value of the boot pins.

An STM32 can also be programmed without JTAG (bootloader activation). Depending on the CPU type, different options are available:

- RS-232 (previously all STMs)
- USB (all USB capable CPUs> F103)
- CAN (like USB only in certain MCUs)

3 additional connections need to be patched on the board. For a test it is also possible to use buttons for RESET and BOOT0.
RESET = RTS (L-active)
BOOT0 = DTR (H-active)
BOOT1 = LOW

Details are here in the forum: STM32 programming tool

Tools for downloading via the STM32 bootloader:

- STSW-MCU005 STM32 and STM8 Flash loader demonstrator
- stm32flash - Open source flash program (RS-232)
- dfu-util - Open source flash program (USB)

Rating [edit]

Advantages [edit]

Advantages over ARM7:

- Interrupt controller now part of the processor (as core peripheral), the vector table is now a real vector table, not a jump list as with the ARM7. Automatic functions between the core and NVIC (auto register save r0..r3, lr, sp, pc) for interrupt entry result in a significantly faster execution time for interrupts. The interrupt code no longer has to worry about backing up the above registers and there is no special configuration of the handler in the compiler. If further interrupts are pending before the end of an ISR (ie return to the user code), these are executed without a complete pop-push sequence of the registers being necessary. It's nicely described here in the

Insider's Guide under 2.4.5 / page 20 (if the link no longer works, googling directly for isg-stm32-v18d-scr.pdf can help ...).
- Thumb-2 instruction set, significantly faster than Thumb-1 and just as compact
- SWD requires fewer pins for debugging
- More hardware breakpoints make debugging easier
- Software is easier because switching between ARM mode and thumb mode is no longer necessary

Advantages over LPC1700 and LPC1300:

- More flexible housing shapes with more peripherals in small housings
- FW-Lib the same for all STM32, all AppNotes / Demos refer to this one FW-Lib, which accelerates the development of your own application very much.
- More accurate and flexible ADC, especially compared to LPC1300
- More flexible variants of the periphery >> with less a clear price advantage
- from EUR 0.85 (as of 2010) However, the LPC1100 with Cortex-M0 is available for as little as $ 0.65!

Advantages over SAM3 / 4:

- Almost all pins are 5 volt tolerant.

Advantages over other "little ones" such as PIC, Atmel etc.

- almost the same price for hobby applications
- 32 bits can be calculated directly in assembler
- Fast direct offset addressing enables efficient access to stack variables, locally stored flash constants, struct / array elements
- Simple, uniform addressing of the entire address space, ie pointers to peripheral registers, RAM & Flash can be handled in exactly the same way, no banking / switchover mechanisms required, even with large Flash / RAM
- Interrupt priorities and priority groups
- Efficient pointer arithmetic because register width = address width
- better peripherals like USB, ethernet, variety of timers
- the ARM core has a higher clock frequency and at the same time can calculate more in fewer clocks
- Hardware division, for some FPUs for efficient float calculation
- Available with larger flash / RAM
- Code can be executed directly from the RAM, memory protection and privileged execution mode can protect "kernel" from "application" code, so the dynamic reloading of applications from external memory is possible efficiently and safely
- ... and another 1000 points ...

Left

- Code size analysis between different µC

Disadvantages [edit]

Disadvantage compared to LPC1700:

- STM32F1xx: only 72 MHz instead of 100 MHz (LPC1759: 120 MHz) clock frequency; STM32F2xx does not have this disadvantage (also 120MHz, STM32F4xx with 180MHz)
- The LPC1700 has significantly more mechanisms to reduce the effect of the wait states of the flash ROM on code and data access, and that means more performance with the same clock frequency. This disadvantage does not apply to the STM32F2 due to the ART accelerator.
- All LPC1xxx have 32 bit timers. With the STM32, only the STM32F2xx and STM32F4xx (2 pieces) have this
- I2S unit from ST has no FIFO and in 24/32 bit mode 2x16 bit half words have to be transmitted. In general, with new ARM processors, the existing DMA channels (based on their own BUS channels and memory accesses) mean FIFO of any size. (Does not apply to certain STM32F4xx)

Disadvantage for hobby users

- Not directly "suitable for breadboard" as there is no DIL housing available. However, the ebay shop dipmicro offers very cheap soldering adapters for converting LQFP48 to DIP48. QFP64 with 0.5mm pin spacing and not 0.8mm like AVR. NXP offers Cortex-M0 µC in a DIL housing.

- A lot of peripherals, clocks must all be set correctly, if necessary, adaptation of the startup code, etc.

- The price-performance ratio is generally worse because of the lower number of units sold

Errata, tips and tricks [edit]

Hardware [edit]

- AD converter PA0: The errata says that conversion errors could occur here, so use a different pin.
- CAN bus PD0 / PD1: Remap only works from the 100-pin version. Is in the RM0008 under 9.3.3 .: "CAN1 alternate function remapping". All information from RM0008 9.3.x is interesting
- In the F1 series, CAN and USB can only be used simultaneously with the "◦Connectivity-Line". See data sheets.
- With an internal RC oscillator, the CPU can be operated at a maximum of 64MHz. With an external crystal, 72MHz are possible.
- For USB operation, the CPU must be operated at 48MHz or 72MHz (with STM32F1xx).
- The idle interrupt from the Usart is triggered, but not indicated by the corresponding status flag

- When activated, the DMA always starts from the beginning, even if it was only stopped briefly
- STM32F2xx does not have a flash size register, the STM32F4xx does have a flash size register, but the address collides with another register
- Derivatives with internal EEPROM and only one memory bank have the "feature" of writing / erasing the data flash (EEPROM) to cause a complete stall of the code execution (incl. ISRs, DMA). The same applies to write / erase the internal flash (ISP routines, EEPROM emulation).
- The I2C has various errors, which can be found in the errata of the respective model (e.g. STM32F105xx and STM32F107xx Errata sheet). Workarounds for this can be found in Application Note AN2824 . However, it is best to use the I2C Communication peripheral application library (CPAL) from ST (STSW-STM32127)
- more undocumented features
- Interrupt flags in the status registers of the various peripherals such as the timer must be reset at the **beginning** (or as far as possible before the return) of the ISR, otherwise the ISR may be executed twice (see STM32 FAQ and forum).

Software [edit]

General [edit]

Standard GPIOs of the STM32 and in particular the BSRR

- The registers consist of two parts, the upper part BR0-15 signals the bits to be deleted in the IO-ODR register with a set bit, the lower part BS0-15 signals the bits to be set with a set bit. It is particularly important if both bits (upper and lower part) are set, the set bit has priority. A clever combination of the upper and lower part can save memory access. For example, you can use such a construct to change the lower 8 bits of the IO-ODR register "uint32_t temp = GPIOC-> ODR & 0xff00; GPIOC-> ODR = temp | (input & 0x00ff)" to shorten a memory access to "GPIOC- > BSRR = (input & 0x00ff) | ((0x00ff) << 16) "

GCC [edit]

To use the GCC directly (e.g. with a self-made makefile), if this cannot be done by a development environment, see ARM GCC first . STM32-specific is:

- If the STM32 Standard Peripheral Library and a crystal are used, the frequency of the crystal must be specified via preprocessor definition using, for example, -DHSE_VALUE = 8000000 for 8MHz (as on the STM32F4 Discovery).

Startup code & linker script [edit]

- So that the compiled code ends up in the right places in the controller (ie the flash), the linker must be given a linker script. This goes to the linker command via "-T *pfad_zum_linkerscript.ld* ". In the archive of the STM32 Standard Peripheral Library there is a sample linker script for the

Atollic TrueSTUDIO IDE, this can be used directly with the GCC. For example for the STM32F4 the script is in the path "/STM32F4xx_DSP_StdPeriph_Lib_V1.1.0/Project/STM32F4xx_StdPeriph_Templates/TrueSTUDIO/STM324x7I_EVAL/stm32_flash.ld" of the archive.
- So that the correct initializations are carried out when starting (such as global variables and with C ++ constructors of global object instances), a startup code must first run, which then calls the main () function. The startup code is mostly written in assembler, but C code is also possible. The archive of the STM32 Standard Peripheral Library contains a sample startup code for the Atollic TrueSTUDIO IDE, which can be used directly with the GCC. For example for the STM32F4 the code is in assembler form in the path "/STM32F4xx_DSP_StdPeriph_Lib_V1.1.0/Libraries/CMSIS/Device/ST/STM32F4xx/Source/Templates/TrueSTUDIO/startup_stm32f40xx.s" of the archive. The assembler code can be assembled using arm-none-eabi-as (flags see above), the resulting .o file is linked normally.

Together, the two files offer the application a standard C interface, ie you can use global variables as usual and write your code in the main () function.

Tips for those switching from Atmel / PIC / 8051 [edit]

- Processor clock has different clock sources and a PLL.
- All peripheral modules have an extra clock that has to be activated.
- For example, if you want to use a UART, you have to activate the clock from the UART, Alternate Function IO (AFIO) and the GPIO port.
- Otherwise you have almost twice as many options in the peripheral modules.
- Interrupt flags must be deleted in the ISR itself
- Forum on interrupts vs. events

Errata from the STM32F4xx that are not in the errata from ST [edit]

- Activating DMA , if more than 3 DMA channels are activated, they may not all be operated correctly. The DMA does not always work reliably with the FSMC. see here and here
- Annoying bug in "stm32f4xx.h" Change structure GPIO_TypeDef
- Battery is drawn empty , only with some chips with Rev. A
- List of documentation errors

Sources of supply [edit]

Controller [edit]

Mail order companies for private individuals

- Reichelt
- Darisus

- TME
- RS-Online
- Mouser
- Conrad

Of course, many deliver commercially such as EBV, Future Electronics , Farnell, Digikey etc.

Evaluation boards [edit]

- various Nucleo and Discovery boards at Conrad
- various Nucleo boards at Reichelt
- STM32 at Watterott (including Olimex and Nucleo boards)
- STM32 boards directly from Olimex
- various STM32 boards myAVR
- Sander Electronic
- Futurlec evaluation board, as well as header board
- Article in the Wiki, ARM with USB and 4MB storage
- Cortex M3 article in the wiki
- STM32Discovery at Farnell microcontroller board (STM32F100RBT6B) with onboard USB programming interface for approx. € 12.50
- SO-DIMM modules with STM32F4, Spartan-6 & DDR3 RAM breakout board with CAN (2x isolated), UART, LAN, SPI, I2C and USB-OTG and much more

Web links, forums, communities, tutorials [edit]

- STM32 - Getting started with Em :: Blocks tutorial
- Search the forum
- STM32 for beginners
- http://www.openstm32.org/System+Workbench+for+STM32
- Forum on the ST homepage
- Entry: STM32board with camera (German)
- STM32 Tutorial for Standard Peripheral Library in German
- STM32 C and C ++ tutorial in German
- Tutorial for graphic libraries and SiSy projects in German
- MicroXplorer MCU graphical configuration tool
- Test report about CoreMark 1.0 on Cortex-M3 / M4 with different compiler and MCU settings
- STM32 toolchain with Eclipse CDT 4.3, GnuArmEclipse, OpenOCD 0.8.0, Gnu Arm GCC 4.8, STM32CubeMX
- libopenmc3 OpenSource libs for STM32 and similar
- Dead time calculator for STM32
- Download book in English, min. $ 25 "Mastering STM32"
- STM32 L0 / F1 / F3 instructions

- "Discovering the STM32 Microcontroller" eBook (free)
- Insight into modern electronics without a lot of theory **PDF** , programming instructions for beginners
- USB tutorial with STM32
- Very nice STM32 beginners tutorial with code analysis right down to the registers
- Introduction to C programming for microcontrollers, focus on STM32 Prof. Dr.-Ing.Otto Parzhuber University of Munich FK06

Categories :

- POOR
- STM32
- Microcontroller