# NETMF for STM32 – HAL Driver Overview

## Version History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Revision Date | Summary of Changes | Author |
| 0.1 | 27.04.2012 | Initial draft. | Mark Munte |
| 0.2 | 30.04.2012 | Review of all chapters | Cuno Pfister |
| 0.3 | 10.05.2012 | Synchronized to latest source code version | Mark Munte |
| 0.4 | 10.05.2012 | Document title renamed | Cuno Pfister |
| 0.5 | 11.05.2012 | Memory map updated | Beat Heeb |

## Purpose

Creating a NET MF port is all about implementing the HAL drivers needed by NET MF for the chosen microprocessor. Early in implementation, the port team has to decide which MCU peripherals to use to implement the HAL drivers.

Knowing which MCU peripherals are used by the HAL is not especially useful for those planning to "just" use the features exposed by NET MF. But for those planning to create additional drivers, for example exposing more advanced features than provided by NET MF, this information is pretty essential.

The purpose of this document is to provide a quick overview of which MCU peripherals are used by the STM32F4 HAL drivers, helping decide which peripherals one should better not touch and which ones are free to use (e.g. when writing custom Interop drivers).

The document does not describe the bootstrap procedure, flash and scatter file configuration, and HAL drivers in detail. To gain deep knowledge bout the port inner workings, one shall look at the code.

## General Information

The STM32F4 port makes extensive use of the ARM CMSIS and ST's CMSIS extensions libraries. Looking at the ports HAL drivers one will almost always find some usage of CMSIS defined structures. The following table describes the most important header files.

|  |  |
| --- | --- |
| File | Notes |
| DeviceCode\Targets\Native\STM32F2\DeviceCode\  core\_cm4.h | CMSIS Cortex-M4 Core Peripheral Access Layer Header File  V2.10  19. July 2011  Provides access to all core peripherals. |
| DeviceCode\Targets\Native\STM32F2\DeviceCode\  stm32f4xx.h | CMSIS Cortex-M4 Device Peripheral Access Layer Header File  V1.0.0  30-September-2011  This is ST's CMSIS extension for the STMF4 MCU's. It contains access to all peripherals not part of the core. |
| Solutions\Discovery4\  platform\_selector.h | Contains lot's of #define's used by the port's HAL driver implementations. Here it is possible to make some adjustments about which MCU peripheral is used by a HAL driver - off course, within constraints. |

## HAL drivers peripheral usage

The following table describes which MCU peripherals are used by the HAL drivers.

|  |  |
| --- | --- |
| HAL Driver | Notes |
| STM32F2\_Analog | The driver can use either ADC1 or ADC3, configured in platform\_selector.h  For the Discovery4 solution, ADC1 is used. Though the STM32F4 ADCs have 16 channels each, the HAL driver will only "expose" seven of them, see STM32F2\_AD\_CHANNELS in platform\_selector.h. Probably to avoid pin conflicts with other drivers.  Sample time can be adjusted in the driver through a #define It defaults to 28 cycles. |
| STM32F2\_Bootstrap | Mostly processor clock setup done in here. All clocks be configured through platform\_selector.h  What is configured and/or enabled:   * PLL, HSE, AHB, APB1, APB2 clocks * JTAG debugging * MCU Cache & Prefetch (on rev Z devices) * 64KB CCM SRAM * FPU * RTC (if defined in platform\_selector.h)   What is disabled:   * HSI clock * flash remap to boot area - instead configures system memory to boot area |
| STM32F2\_ETH\_lwip | TODO |
| STM32F2\_Flash | Flash driver implementation. Flash is configured for 16bit programming through a #define. Not sure if it makes any sense to change this. |
| STM32F2\_GPIO | There's nothing to configure in here but this is certainly a driver one should examine closer if planning to extend NET MF's capabilities. Almost every STM32F4 peripheral will use one or more pins - requiring those to be correctly setup by the GPIO driver.  The STM32F4 's MCU GPIO peripheral allows/requires more fine grained pin settings than possible by using the CPU\_GPIO\_DisablePin PAL function. To overcome this, pin attributes like speed, alternate function and mode are packed into the functions GPIO\_*ALT\_MODE alternate* parameter. Please look into the source code for details. |
| STM32F2\_I2C | Though the STM32F4 has three I2C peripherals, NET MF currently only exposes a single "I2CDevice". The I2C HAL driver can be configured to use any of the three available I2C peripherals, see platform\_selector.h. Default is I2C #1. |
| STM32F2\_IntC | There's nothing to configure about the interrupt controller.  NOTE: the STM32F4 port makes room for 84 interrupt vectors. That's actually three vectors less than defined by the NVIC. Should this be improved? |
| STM32F2\_Power | Nothing to configure here. |
| STM32F2\_PWM | Uses a single timer peripheral to generate up to four PWM outputs. Allowed timers are: TIM1, TIM4, TIM5, TIM8  Default is TIM1 and TIM4, see platform\_selector.h  The STM32F4 has three kinds of timers, the PWM HAL expects a timer with four OC channels to be used in order to be able to generate up to four PWM signals.  TIM1 & TIM8 are "Advanced-control timers"   * 16 bit bit up, down, etc counter * 16 bit on-the-fly changeable pre-scaler * four channels for input/output capture, pwm, one-pulse * many more features...!   TIM4 & TIM5 are "General-purpose timers" supporting the feature described above. One notable difference: TIM5 is a 32bit timer - which does not really matter for pwm generation. |
| STM32F2\_security | TDB: not sure what this does. Looks like a stub implementation. |
| STM32F2\_SPI | All three MCU SPI peripherals are exposed. Nothing to configure in here. |
| STM32F2\_Time | Uses a 32bit and a 16bit timer to build a chained 48bit timer suitable for use with NET MF.  Wich TIM\* timer is used can be configured in platform\_selector.h,  Nothing to configure in here.  Don't touch these timers:) |
| STM32F2\_USART | All six USART peripherals are exposed by the driver. Default Debug port is COM3 (RX pin D9, TX pin D8) |
| STM32F2\_USB | TODO |

## A word about the memory configuration

The STM32F4 chip used on the Discovery4 board has 192KB embedded SRAM split into three blocks:

|  |  |  |
| --- | --- | --- |
| Address | Size | Notes |
| 0x10000000 - 0x1000FFF8 | 64KB | CCM, "Core Coupled Memory"  Only accessible by the CPU core, not by peripherals! |
| 0x20000000 - 0x2001BFFF | 112KB | SRAM1 |
| 0x2001C000 - 0x2001FFFF | 16KB | SRAM2 |

SRAM1 and SRAM2 are almost entirely used for the heap.

The stack is placed at the first 16KB of the CCM, so that the stack would overflow into unmapped address space and cause an exception. RW and ZI data is placed after the stack. At the end of the CCM, there’s currently around 20KB of currently unused SRAM.

Keep in mind that only the CPU has access to the CCM SRAM, meaning peripherals and the DMA engine have no access to it. So make sure to only use heap allocated buffers if you plan to use DMA.