STM32F4xx Technical Training

STM32 product series

4 product series

Common core peripherals and architecture:

- Communication peripherals: USART, SPI, I²C
- Multiple general-purpose timers
- Integrated reset and brown-out warning
- Multiple DMA
- 2x watchdogs Real-time clock
- Integrated regulator PLL and clock circuit
- External memory interface (FSMC)
- Dual 12-bit DAC
- Up to 3x 12-bit ADC (up to 0.41 µs)
- Main oscillator and 32 kHz oscillator
- Low-speed and high-speed internal RC oscillators
- -40 to +85 °C and up to 105 °C operating temperature range
- Low voltage 2.0 to 3.6 V or 1.65/1.7 to 3.6 V (depending on series) 5.0 V tolerant I/Os
- Temperature sensor

STM32 F4 series - High performance with DSP (STM32F405/415/407/417)

168 MHz Cortex-M4 with DSP and FPU	Up to 192-Kbyte SRAM	Up to 1-Mbyte Flash	2x USB 2.0 OTG FS/HS	3-phase MC timer	2x CAN 2.0B	SDIO 2x I²S audio Camera IF	Ethernet IEEE 1588	Crypto/hash processor and RNG

STM32 F2 series - High performance (STM32F205/215/207/217)

120 MHz Cortex-M3 CPU	Up to 128-Kbyte SRAM	Up to 1-Mbyte Flash	2x USB 2.0 OTG FS/HS	3-phase MC timer	2x CAN 2.0B	SDIO 2x I²S audio Camera IF	Ethernet IEEE 1588	Crypto/hash processor and RNG

STM32 F1 series - Connectivity line (STM32F105/107)

72 MHz Cortex-M3 CPU	Up to 64-Kbyte SRAM	Up to 256-Kbyte Flash	USB 2.0 OTG FS	3-phase MC timer	2x CAN 2.0B	2x I²S audio	Ethernet IEEE 1588

STM32 F1 series - Performance line (STM32F103)

72 MHz Cortex-M3 CPU	Up to 96-Kbyte SRAM	Up to 1-Mbyte Flash	USB FS device	3-phase MC timer	CAN 2.0B	SDIO 2x I²S

STM32 F1 series - USB Access line (STM32F102)

+

48 MHz Cortex-M3 CPU	Up to 16-Kbyte SRAM	Up to 128-Kbyte Flash	USB FS device

STM32 F1 series - Access line (STM32F101)

36 MHz Cortex-M3 CPU	Up to 80-Kbyte SRAM	Up to 1-Mbyte Flash

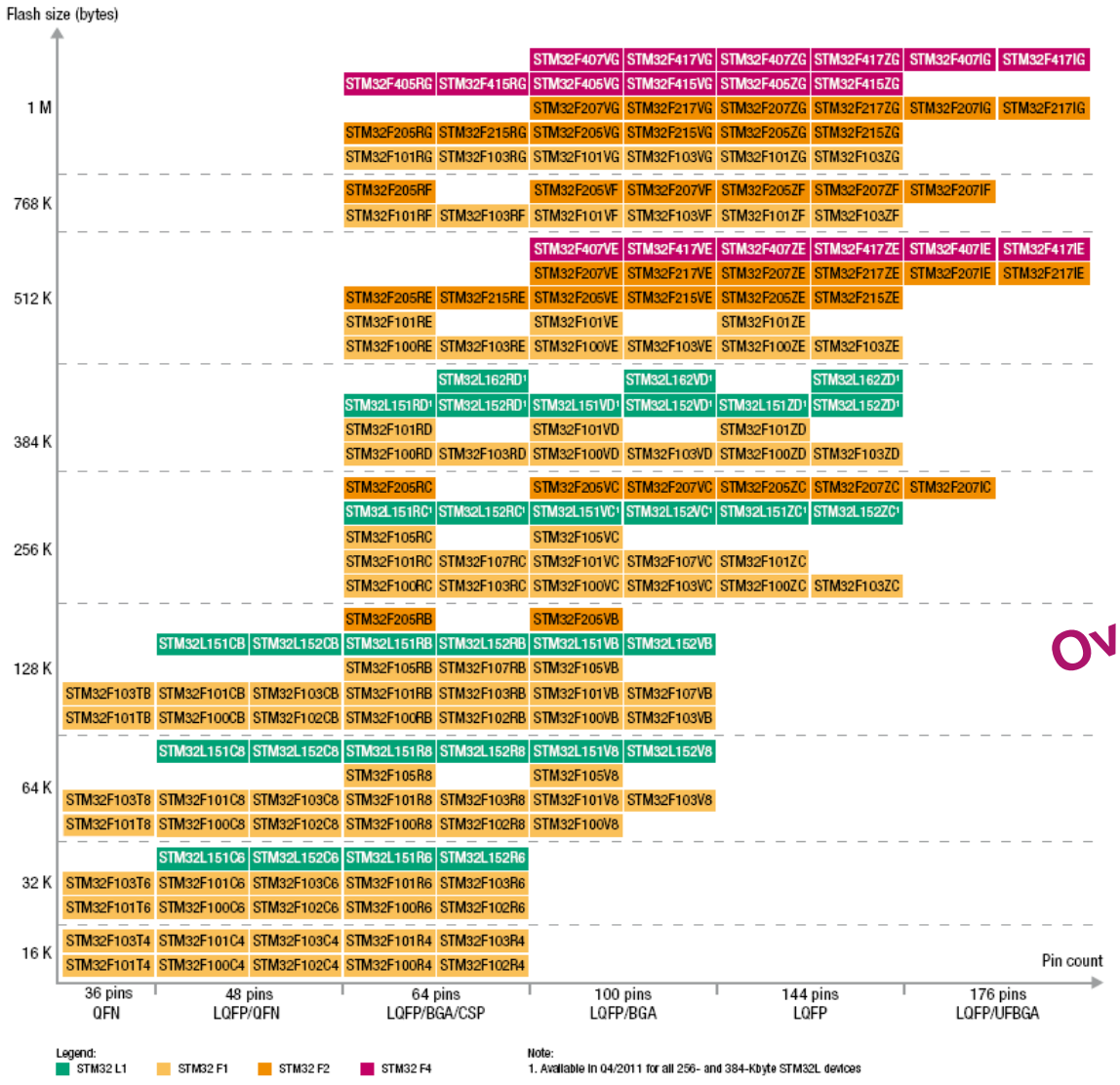
STM32 F1 series - Value line (STM32F100)

24 MHz Cortex-M3 CPU	Up to 32-Kbyte SRAM	Up to 512-Kbyte Flash	3-phase MC timer	CEC

STM32 L1 series - Ultra-low-power (STM32F151/152)

32 MHz Cortex-M3 CPU	Up to 48-Kbyte SRAM	Up to 384-Kbyte Flash	USB FS device	Data EEPROM up to 12 Kbytes	LCD 8x40 4x44	Comparator	BOR MSI VScal	AES 128-bit

STM32 – leading Cortex-M portfolio



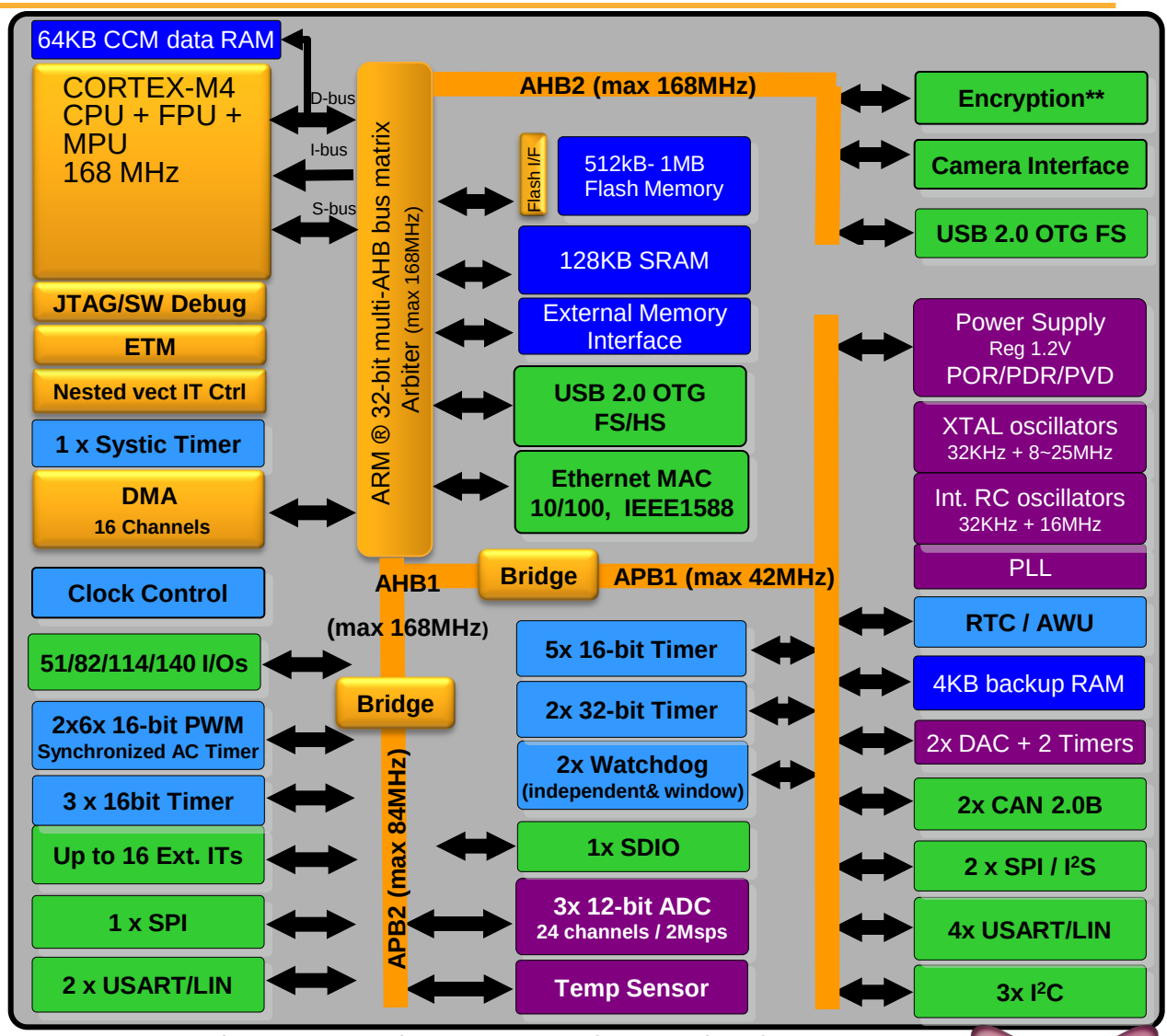
Over 250 pin-to-pin compatible part numbers

STM32 F4 series

High-performance Cortex™-M4 MCU

STM32F4xx Block Diagram

- **Cortex-M4 w/ FPU, MPU and ETM**
- **Memory**
  - Up to 1MB Flash memory
  - 192KB RAM (including 64KB CCM data RAM
  - FSMC up to 60MHz
- **New application specific peripherals**
  - USB OTG HS w/ ULPI interface
  - Camera interface
  - HW Encryption**: DES, 3DES, AES 256-bit, SHA-1 hash, RNG.
- **Enhanced peripherals**
  - USB OTG Full speed
  - ADC: 0.416µs conversion/2.4Msps, up to 7.2Msps in interleaved triple mode
  - ADC/DAC working down to 1.8V
  - Dedicated PLL for I²S precision
  - Ethernet w/ HW IEEE1588 v2.0
  - 32-bit RTC with calendar
  - 4KB backup SRAM in VBAT domain
  - 2 x 32bit and 8 x 16bit Timers
  - high speed USART up to 10.5Mb/s
  - high speed SPI up to 37.5Mb/s
- **RDP (JTAG fuse)**
- **More I/Os in UFBGA 176 package**



HS requires an external PHY connected to ULPI interface,
** Encryption is only available on STM32F415 and STM32F417

STM32F4 Series highlights 1/3

- Based on Cortex M4 core
  - The **new DSP and FPU instructions** combined to 168MHz
- Over 30 new part numbers **pin-to-pin and software compatible** with existing STM32 F2 Series.

Advanced technology and process from ST:

- **Memory accelerator**: ART Accelerator™
- Multi AHB Bus Matrix
- 90nm process

Outstanding results:

- **210DMIPS** at 168MHz.
- Execution from Flash equivalent to **0-wait state** performance up to 168MHz thanks to ST ART Accelerator

STM32F4 Series highlights 2/3

More Memory

- Up to **1MB Flash** with option to permanent readout protection (**JTAG fuse**),
- **192kB SRAM**: 128kB on bus matrix + 64kB (Core Coupled Memory) on data bus dedicated to the CPU usage

Advanced peripherals

- **USB OTG High speed** 480Mbit/s
- **Ethernet MAC** 10/100 with IEEE1588
- PWM **High speed timers**: 168MHz max frequency
- **Crypto/Hash processor**, 32-bit random number generator (**RNG**)
- 32-bit RTC with calendar: with sub 1 second accuracy, and <1uA

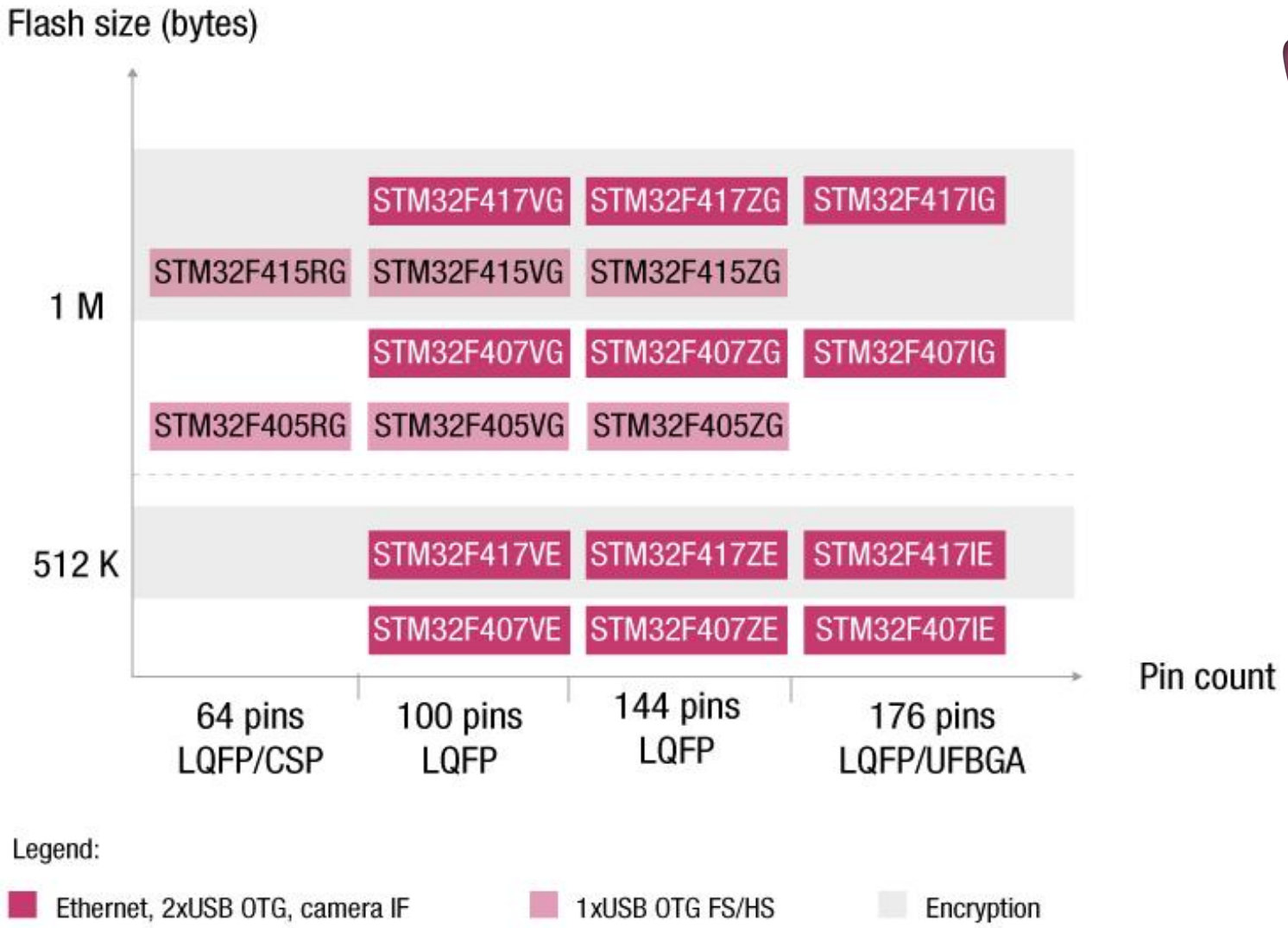
STM32F4 Series highlights 3/3

Further improvements

- Low voltage: 1.8V to 3.6V VDD , down to 1.7*V on most packages
- **Full duplex I²S** peripherals
- **12-bit ADC**: 0.41µs conversion/2.4Msps (**7.2Msps** in interleaved mode)
- High speed **USART** up to **10.5Mbits/s**
- High speed **SPI** up to **37.5Mbits/s**
- **Camera interface** up to **54MBytes/s**

*external reset circuitry required to support 1.7V

STM32F4 portfolio



Extensive tools and SW

- Evaluation board for full product feature evaluation
  - Hardware evaluation platform for all interfaces
  - Possible connection to all I/Os and all peripherals
- Discovery kit for cost-effective evaluation and prototyping
- Starter kits from 3rd parties available soon
- Large choice of development IDE solutions from the STM32 and ARM ecosystem

STM3240G-EVAL

$349

STM32F4DISCOVERY

$14.90

RAISONANCE Embbeded Systems Development Tools · aiji system 아이지시스템 · hitex DEVELOPMENT TOOLS · IAR SYSTEMS · atollic · iSYSTEM · Green Hills SOFTWARE, INC. · SEGGER · KEIL An ARM® Company · LAUTERBACH · SIGNUM SYSTEMS · TASKING · expresslogic · CMX SYSTEMS

Tools for development – SW (examples)

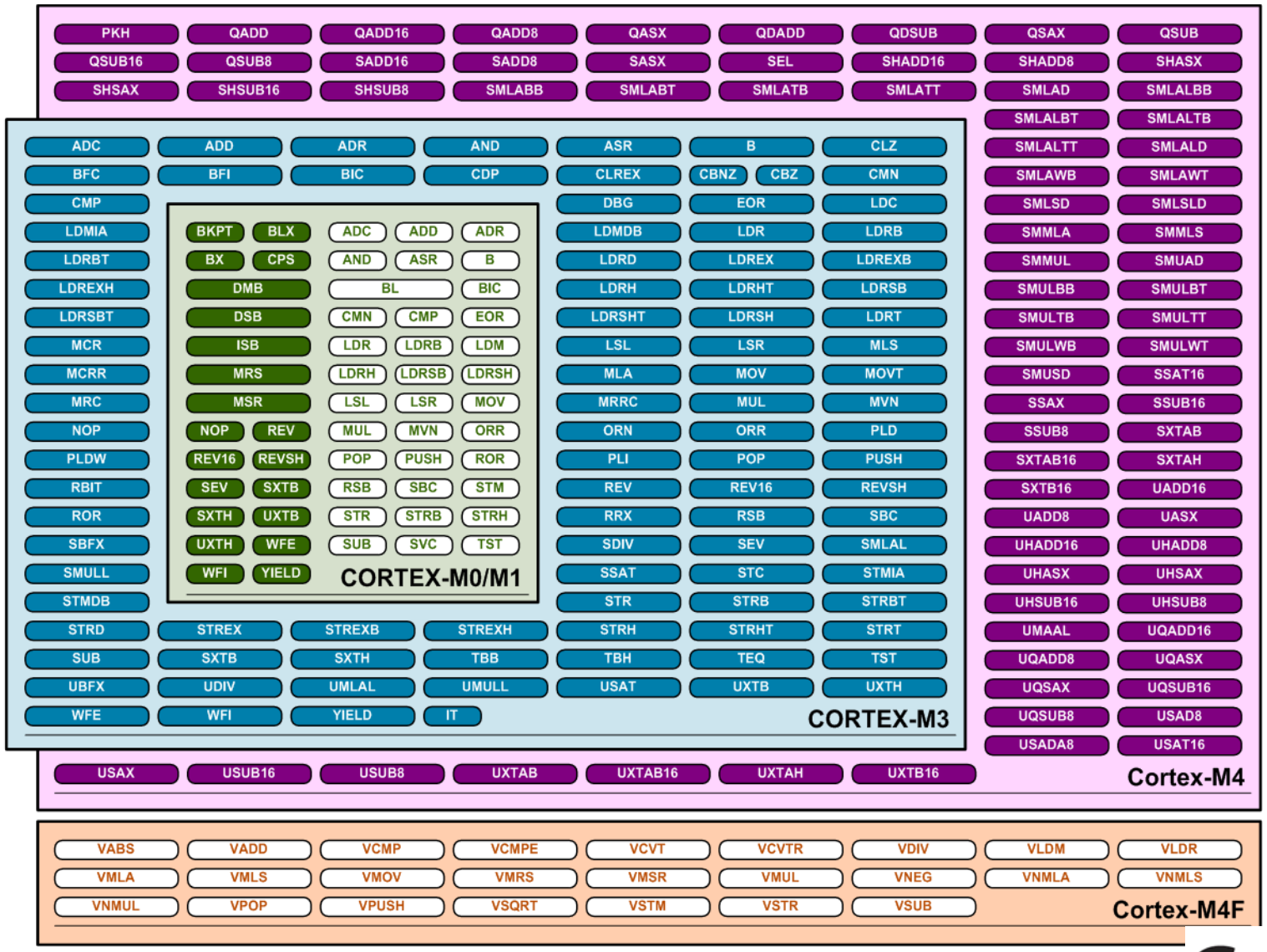
- **Commercial ones:**
  - IAR – eval 32kB/30days for test [RK-System]
  - Keil (ARM) – eval 32kB for test [WG Electronics]
- **Based on GCC commercial:**
  - Atollic – Lite (no hex/bin, limited debug), [Kamami]
  - Raisonance – debug limited to 32kB
  - Rowley Crossworks – 30 days for test
- **Free**
  - STVP – FLASH prog.
  - STLink utility – FLASH prog. (+cmd line)
  - ST FlashLoader – FLASH prog.
- **Libraries (free)**
  - Standard peripherals library with CMSIS
  - USB device library

LITE/PRO VERSION FEATURE COMPARISION

FEATURE	LITE	PRO
Price	Free	Low-cost
Supported languages	Assembler, C	Assembler, C and C++
ARM build & debug tools	✔	✔
PC build & debug tools	–	✔
GUI configuration of command line tool options	–	✔
Extensive IDE	✔	✔
Additional IDE features	–	✔
Graphical UML editors	–	✔
Integrated version control system client	–	✔
Integrated bug/task management system client	–	✔
Runtime libraries	Precompiled	Adaptable
JTAG dongle support	ST ST-LINK	Extensive
Technical support	–	Available
Unlimited code size	✔	✔
Unlimited usage time	✔	✔

ARM Cortex M4 in few words

Cortex-M processors binary compatible



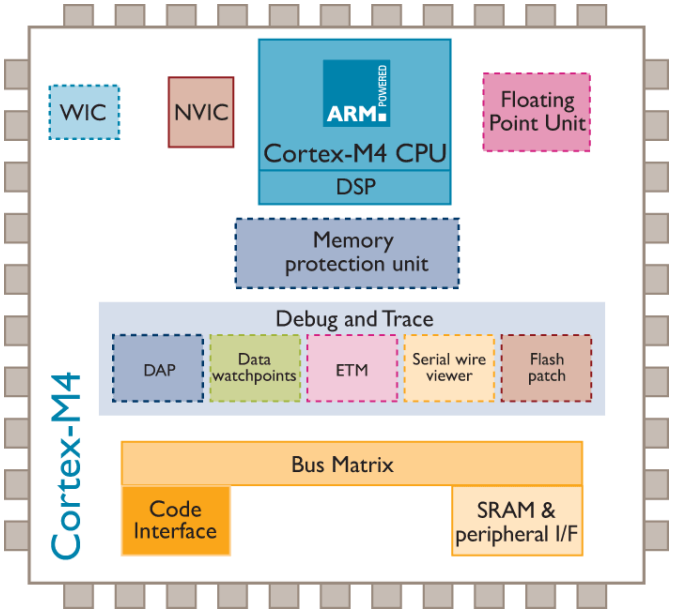
Cortex-M feature set comparison

	Cortex-M0	Cortex-M3	Cortex-M4
Architecture Version	V6M	v7M	v7ME
Instruction set architecture	Thumb, Thumb-2 System Instructions	Thumb + Thumb-2	Thumb + Thumb-2, DSP, SIMD, FP
DMIPS/MHz	0.9	1.25	1.25
Bus interfaces	1	3	3
Integrated NVIC	Yes	Yes	Yes
Number interrupts	1-32 + NMI	1-240 + NMI	1-240 + NMI
Interrupt priorities	4	8-256	8-256
Breakpoints, Watchpoints	4/2/0, 2/1/0	8/4/0, 2/1/0	8/4/0, 2/1/0
Memory Protection Unit (MPU)	No	Yes (Option)	Yes (Option)
Integrated trace option (ETM)	No	Yes (Option)	Yes (Option)
Fault Robust Interface	No	Yes (Option)	No
Single Cycle Multiply	Yes (Option)	Yes	Yes
Hardware Divide	No	Yes	Yes
WIC Support	Yes	Yes	Yes
Bit banding support	No	Yes	Yes
Single cycle DSP/SIMD	No	No	**Yes**
Floating point hardware	No	No	**Yes**
Bus protocol	AHB Lite	AHB Lite, APB	AHB Lite, APB
CMSIS Support	Yes	Yes	Yes

Cortex M4

Cortex-M4 processor architecture

- **ARMv7ME Architecture**
  - Thumb-2 Technology
  - DSP and SIMD extensions
  - Single cycle MAC (Up to 32 x 32 + 64 -> 64)
  - Optional single precision FPU
  - Integrated configurable NVIC
  - Compatible with Cortex-M3
- **Microarchitecture**
  - 3-stage pipeline with branch speculation
  - 3x AHB-Lite Bus Interfaces
- **Configurable for ultra low power**
  - Deep Sleep Mode, Wakeup Interrupt Controller
  - Power down features for Floating Point Unit
- **Flexible configurations for wider applicability**
  - Configurable Interrupt Controller (1-240 Interrupts and Priorities)
  - Optional Memory Protection Unit
  - Optional Debug & Trace



Cortex-M4 overview

- Main Cortex-M4 processor features
  - ARMv7-ME architecture revision
    - Fully compatible with Cortex-M3 instruction set
  - Single-cycle multiply-accumulate (MAC) unit
  - Optimized single instruction multiple data (SIMD) instructions
  - Saturating arithmetic instructions
  - Optional single precision Floating-Point Unit (FPU)
  - Hardware Divide (2-12 Cycles), same as Cortex-M3
  - Barrel shifter (same as Cortex-M3)
  - Hardware divide (same as Cortex-M3)

Single-cycle multiply-accumulate unit

- The multiplier unit allows any MUL or MAC instructions to be executed in a single cycle
  - Signed/Unsigned Multiply
  - Signed/Unsigned Multiply-Accumulate
  - Signed/Unsigned Multiply-Accumulate Long (64-bit)

- Benefits : Speed improvement vs. Cortex-M3
  - 4x for 16-bit MAC (dual 16-bit MAC)
  - 2x for 32-bit MAC
  - up to 7x for 64-bit MAC

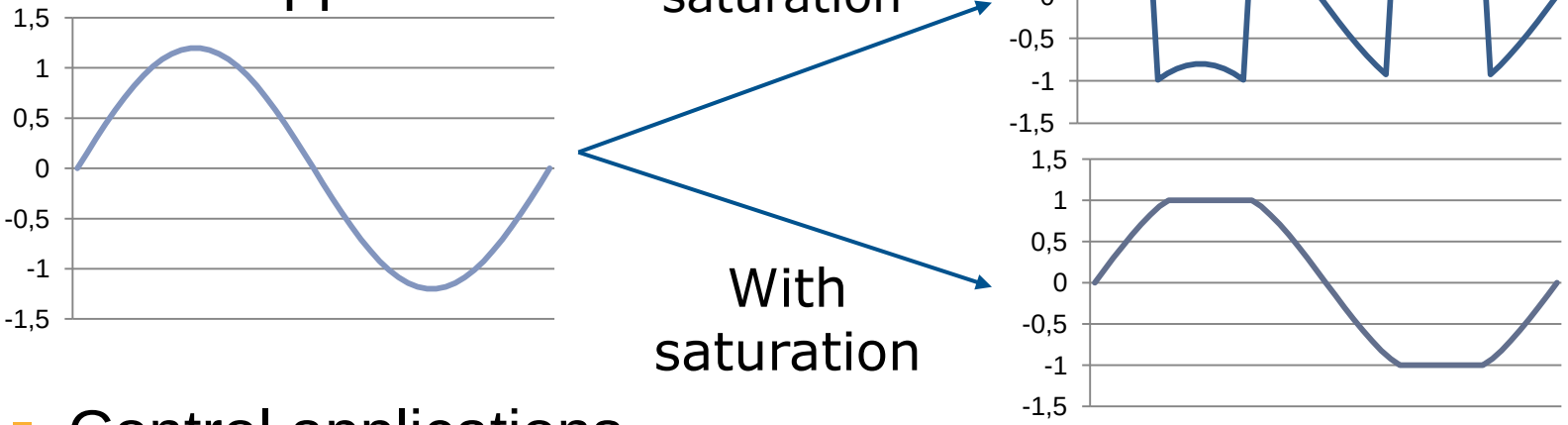
Cortex-M4 extended single cycle MAC

OPERATION	INSTRUCTIONS	CM3	CM4
16 x 16 = 32	SMULBB, SMULBT, SMULTB, SMULTT	n/a	1
16 x 16 + 32 = 32	SMLABB, SMLABT, SMLATB, SMLATT	n/a	1
16 x 16 + 64 = 64	SMLALBB, SMLALBT, SMLALTB, SMLALTT	n/a	1
16 x 32 = 32	SMULWB, SMULWT	n/a	1
(16 x 32) + 32 = 32	SMLAWB, SMLAWT	n/a	1
(16 x 16) ± (16 x 16) = 32	SMUAD, SMUADX, SMUSD, SMUSDX	n/a	1
(16 x 16) ± (16 x 16) + 32 = 32	SMLAD, SMLADX, SMLSD, SMLSDX	n/a	1
(16 x 16) ± (16 x 16) + 64 = 64	SMLALD, SMLALDX, SMLSLD, SMLSLDX	n/a	1
32 x 32 = 32	MUL	1	1
32 ± (32 x 32) = 32	MLA, MLS	2	1
32 x 32 = 64	SMULL, UMULL	5-7	1
(32 x 32) + 64 = 64	SMLAL, UMLAL	5-7	1
(32 x 32) + 32 + 32 = 64	UMAAL	n/a	1
32 ± (32 x 32) = 32 (upper)	SMMLA, SMMLAR, SMMLS, SMMLSR	n/a	1
(32 x 32) = 32 (upper)	SMMUL, SMMULR	n/a	1

All the above operations are single cycle on the Cortex-M4 processor

Saturated arithmetic

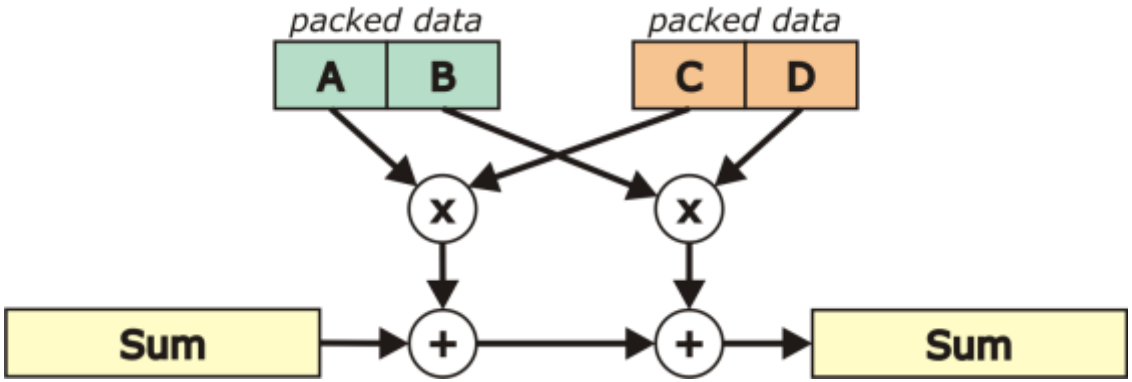
- Intrinsically prevents overflow of variable by clipping to min/max boundaries and remove CPU burden due to software range checks
- Benefits
  - Audio applications



  - Control applications
    - The PID controllers' integral term is continuously accumulated over time. The saturation automatically limits its value and saves several CPU cycles per regulators

Single-cycle SIMD instructions

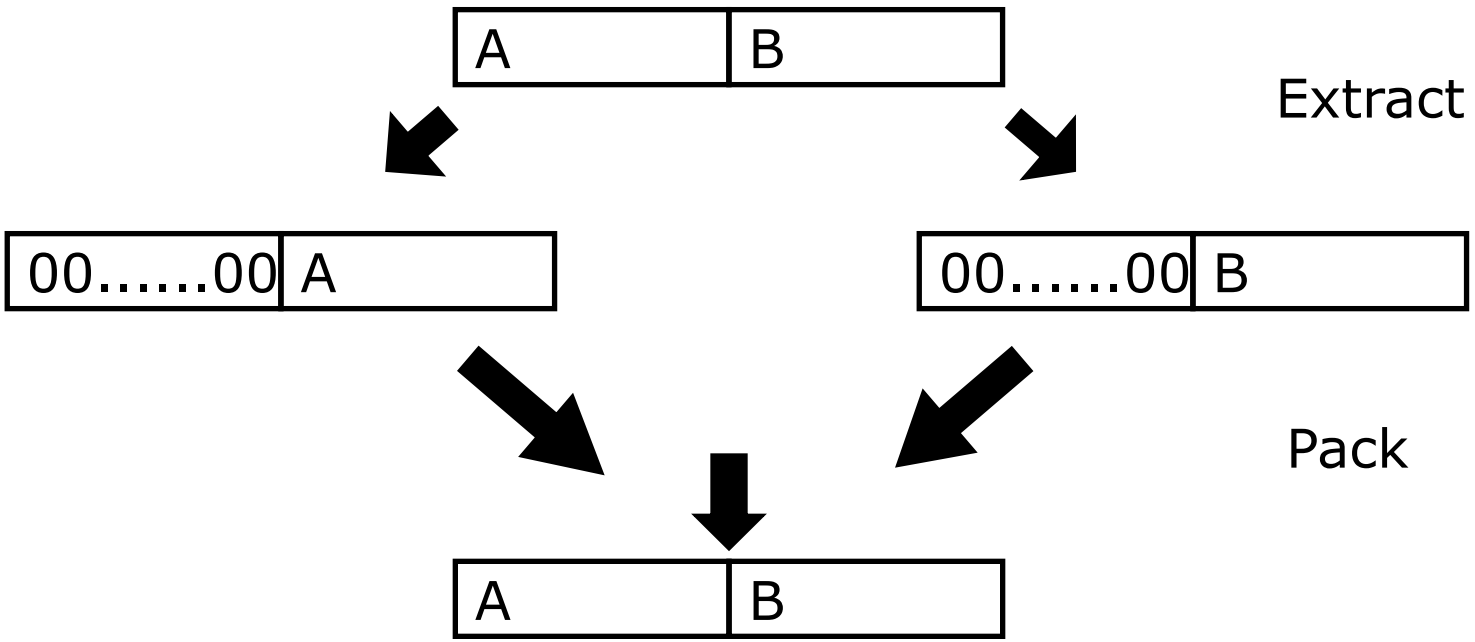
- Stands for **Single Instruction Multiple Data**
- It operates with **packed data**
- Allows to do simultaneously several operations with 8-bit or 16-bit data format
  - i.e.: dual 16-bit MAC (Result = 16x16 + 16x16 + 32)



- Benefits
  - Parallelizes operations (2x to 4x speed gain)
  - Minimizes the number of Load/Store instruction for exchanges between memory and register file (2 or 4 data transferred at once), if 32-bit is not necessary
  - Maximizes register file use (1 register holds 2 or 4 values)

Packed data types

- Byte or halfword quantities packed into words
- Allows more efficient access to packed structure types
- SIMD instructions can act on packed data
- Instructions to extract and pack data



IIR – single cycle MAC benefit

	Cortex-M3 cycle count	Cortex-M4 cycle count
`xN = *x++;`	2	2
`yN = xN * b0;`	3-7	1
`yN += xNm1 * b1;`	3-7	1
`yN += xNm2 * b2;`	3-7	1
`yN -= yNm1 * a1;`	3-7	1
`yN -= yNm2 * a2;`	3-7	1
`*y++ = yN;`	2	2
`xNm2 = xNm1;`	1	1
`xNm1 = xN;`	1	1
`yNm2 = yNm1;`	1	1
`yNm1 = yN;`	1	1
`Decrement loop counter`	1	1
`Branch`	2	2

$$y[n] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] - a_1 y[n-1] - a_2 y[n-2]$$

- Only looking at the inner loop, making these assumptions
  - Function operates on a block of samples
  - Coefficients b0, b1, b2, a1, and a2 are in registers
  - Previous states, x[n-1], x[n-2], y[n-1], and y[n-2] are in registers
- Inner loop on Cortex-M3 takes 27-47 cycles per sample
- Inner loop on Cortex-M4 takes 16 cycles per sample

Further optimization strategies

- Circular addressing alternatives
- Loop unrolling
- Caching of intermediate variables
- Extensive use of SIMD and intrinsics

FIR Filter Standard C Code

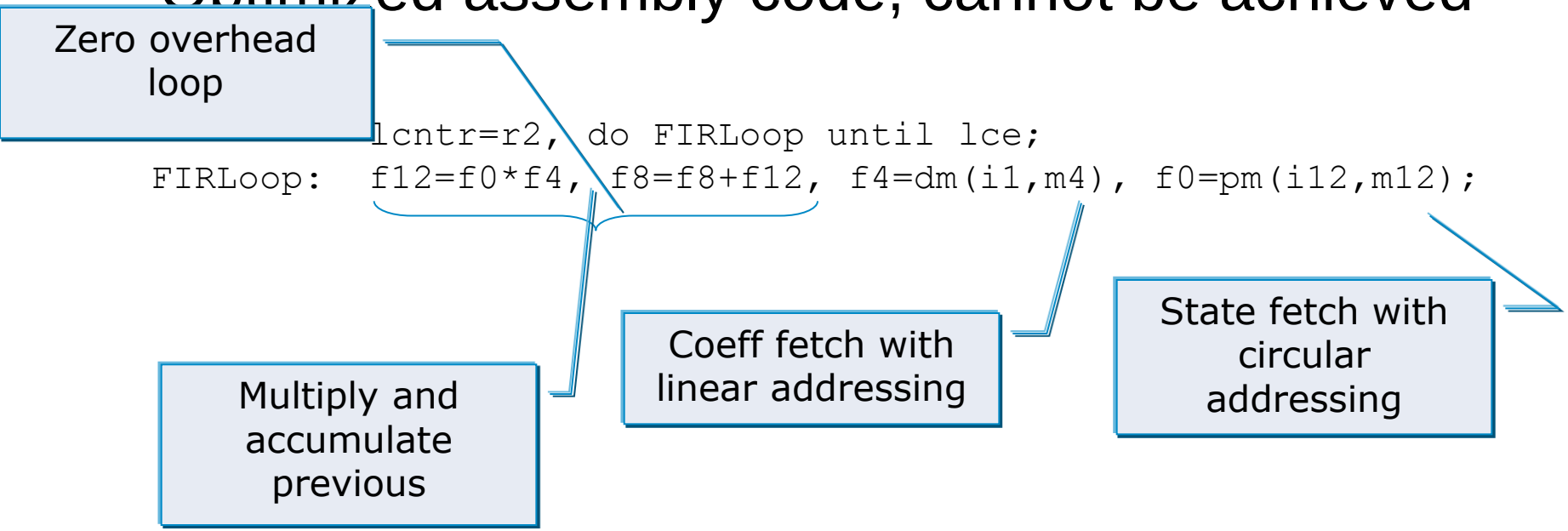
```
void fir(q31_t *in, q31_t *out, q31_t *coeffs, int *stateIndexPtr,
                    int filtLen, int blockSize)
{
  int sample;
  int k;
  q31_t sum;
  int stateIndex = *stateIndexPtr;

  for(sample=0; sample < blockSize; sample++)
    {
      state[stateIndex++] = in[sample];
      sum=0;
      for(k=0;k<filtLen;k++)
          {
            sum += coeffs[k] * state[stateIndex];
            stateIndex--;
            if (stateIndex < 0)
              {
                stateIndex = filtLen-1;
              }
          }
      out[sample]=sum;
    }
    *stateIndexPtr = stateIndex;
}
```

- Block based processing
- Inner loop consists of:
  - Dual memory fetches
  - MAC
  - Pointer updates with circular addressing

circular addressing

FIR Filter DSP Code

- 32-bit DSP processor assembly code
- Only the inner loop is shown, executes in a single cycle
- Optimized assembly code, cannot be achieved



Cortex-M4 - Final FIR Code

```
sample = blockSize/4;
    do
    {
       sum0 = sum1 = sum2 = sum3 = 0;
       statePtr = stateBasePtr;
       coeffPtr = (q31_t *)(S->coeffs);
       x0 = *(q31_t *)(statePtr++);
       x1 = *(q31_t *)(statePtr++);
       i = numTaps>>2;
       do
       {
            c0 = *(coeffPtr++);
            x2 = *(q31_t *)(statePtr++);
            x3 = *(q31_t *)(statePtr++);
            sum0  = __SMLALD(x0, c0, sum0);
            sum1  = __SMLALD(x1, c0, sum1);
            sum2  = __SMLALD(x2, c0, sum2);
            sum3  = __SMLALD(x3, c0, sum3);

            c0 = *(coeffPtr++);
            x0 = *(q31_t *)(statePtr++);
            x1 = *(q31_t *)(statePtr++);

            sum0  = __SMLALD(x0, c0, sum0);
            sum1  = __SMLALD(x1, c0, sum1);
            sum2  = __SMLALD (x2, c0, sum2);
            sum3  = __SMLALD (x3, c0, sum3);
       } while(--i);
       *pDst++ = (q15_t) (sum0>>15);

       *pDst++ = (q15_t) (sum1>>15);
       *pDst++ = (q15_t) (sum2>>15);
       *pDst++ = (q15_t) (sum3>>15);

       stateBasePtr= stateBasePtr + 4;
    } while(--sample);
```

Uses loop unrolling, SIMD intrinsics, caching of states and coefficients, and work around circular addressing by using a large state buffer.

Inner loop is 26 cycles for a total of 16, 16-bit MACs.

Only 1.625 cycles per filter tap!

Cortex-M4 - FIR performance

- DSP assembly code = 1 cycle

- Cortex-M4 standard C code takes 12 cycles
- Using circular addressing alternative = 8 cycles
- After loop unrolling < 6 cycles
- After using SIMD instructions < 2.5 cycles
- After caching intermediate values ~ 1.6 cycles

Cortex-M4 C code now comparable in performance

Cortex M4

Floating Point Unit

Overview

- **FPU : Floating Point Unit**
  - Handles “real” number computation
  - Standardized by **IEEE.754-2008**
    - Number format
    - Arithmetic operations
    - Number conversion
    - Special values
    - 4 rounding modes
    - 5 exceptions and their handling

- **ARM Cortex-M FPU ISA**
  - Supports
    - Add, subtract, multiply, divide
    - Multiply and accumulate
    - Square root operations

C language example

```
float function1(float number1, float number2)
{
        float temp1, temp2;

        temp1 = number1 + number2;
        temp2 = number1/temp1;

        return temp2;
}
```

```
# float function1(float number1, float number2)
# {
#       float temp1, temp2;
#
#       temp1 = number1 + number2;
        VADD.F32 S1,S0,S1
#       temp2 = number1/temp1;
        VDIV.F32 S0,S0,S1
#
#       return temp2;
        BX      LR
# }
```

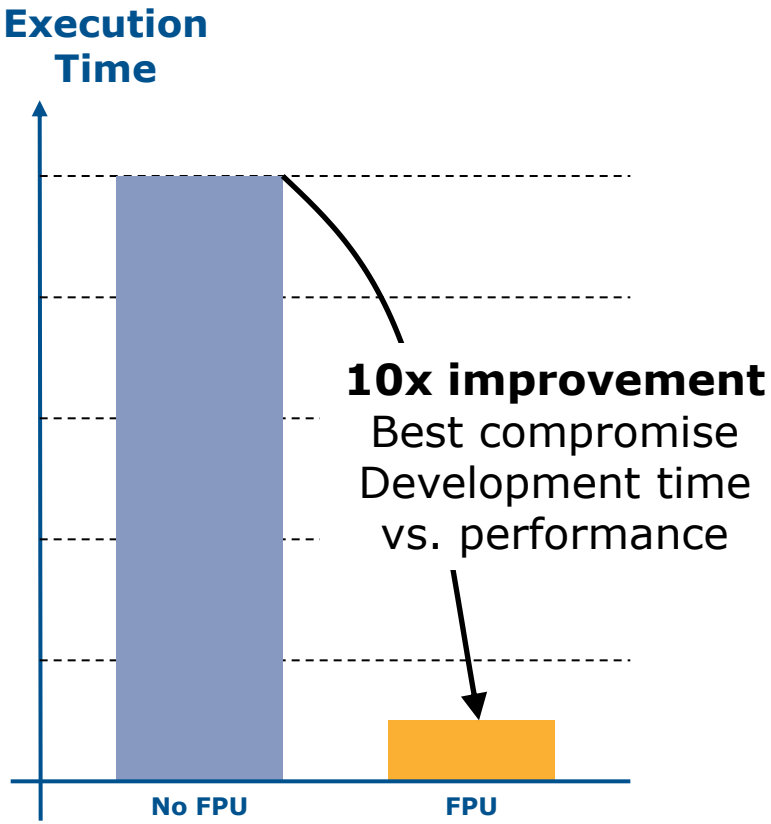
1 assembly instruction

```
# float function1(float number1, float number2)
# {
        PUSH    {R4,LR}
        MOVS    R4,R0
        MOVS    R0,R1
#       float temp1, temp2;
#
#       temp1 = number1 + number2;
        MOVS    R1,R4
        BL      __aeabi_fadd
        MOVS    R1,R0
#       temp2 = number1/temp1;
        MOVS    R0,R4
        BL      __aeabi_fdiv
#
#       return temp2;
        POP     {R4,PC}
# }
```

Call Soft-FPU

Performances

- Time execution comparison for a 29 coefficient FIR on float 32 with and without FPU (CMSIS library)



Rounding issues

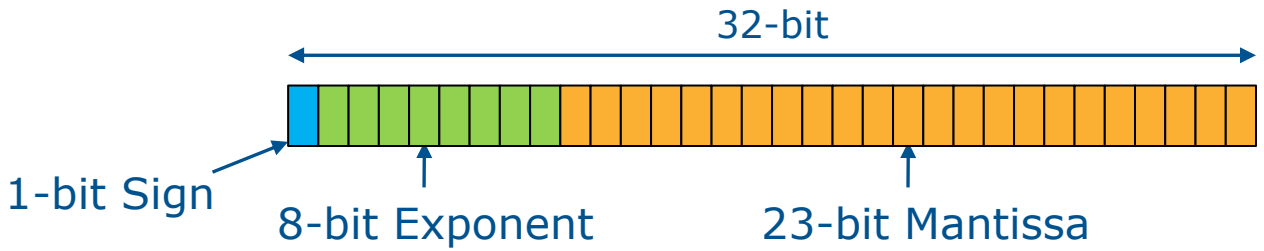
- **The precision has some limits**
  - Rounding errors can be accumulated along the various operations an may provide unaccurate results (do not do financial operations with floatings…)
- **Few examples**
  - If you are working on two numbers in different base, the hardware automatically « denormalize » on of the two number to make the calculation in the same base
  - If you are substracting two numbers very closed you are loosing the relative precision (also called cancellation error)
- **If you are « reorganizing » the various operations, you may not obtain the same result as because of the rounding errors…**

IEEE 754

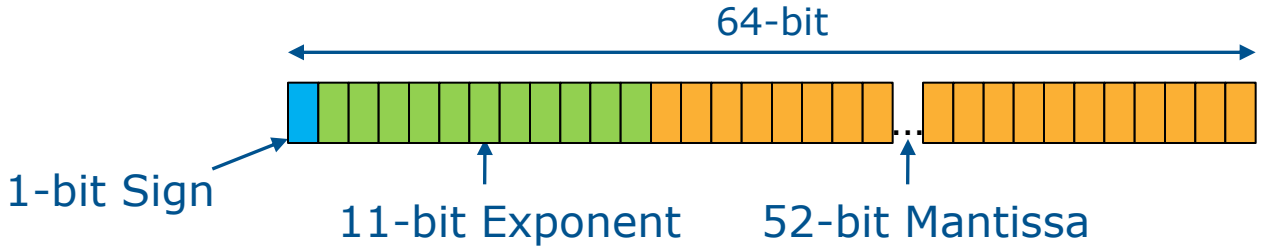
Number format

- **3 fields**
  - Sign
  - Biased exponent (sum of an exponent plus a constant bias)
  - Fractions (or mantissa)

- **Single precision : 32-bit coding**

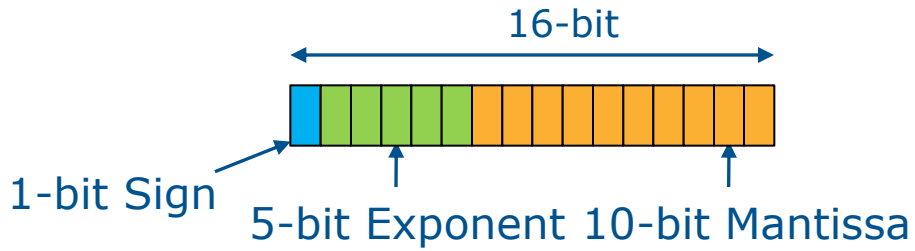


- **Double precision : 64-bit coding**



Number format

- **Half precision : 16-bit coding**



  - Can also be used for storage in higher precision FPU
  - ARM has an alternative coding for Half precision

Normalized number value

- **Normalized number**
  - Code a number as :

    A sign + Fixed point number between 1.0 and 2.0 multiplied by 2^N
- **Sign field (1-bit)**
  - 0 : positive
  - 1 : negative
- **Single precision exponent field (8-bit)**
  - **Exponent range** : 1 to 254 (0 and 255 reserved)
  - **Bias** : 127
  - **Exponent - bias range** : -126 to +127
- **Single precision fraction (or mantissa) (23-bit)**
  - **Fraction** : value between 0 and 1 : $\sum(N_i.2^{-i})$ with i in 1 to 24 range
  - The 23 N_i values are store in the fraction field

$$\mathbf{(-1)^s \times (1 + \sum(N_i.2^{-i})) \times 2^{exp-bias}}$$

Number value

- **Single precision coding of -7**
  - **Sign bit** = 1
  - **7** = 1.75 x 4 = (1 + ½ + ¼) x 4 = (1 + ½ + ¼) x 2^2
    = (1 + 2^{-1} + 2^{-2}) x 2^2
  - **Exponent** = 2 + bias = 2 + 127 = 129 = 0b10000001
  - **Mantissa** = 2^{-1} + 2^{-2} = 0b11000000000000000000000
- **Result**
  - Binary coding : 0b 1 10000001 11000000000000000000000
  - Hexadecimal value : **0xC0E00000**

Special values

- **Denormalized (Exponent field all “0”, Mantisa non 0)**
  - Too small to be normalized (but some can be normalized afterward)
  - **$(-1)^s \times (\sum(N_i.2^{-i}) \times 2^{-bias}$**
- **Infinity (Exponent field “all 1”, Mantissa “all 0”)**
  - Signed
  - Created by an overflow or a division by 0
  - Can not be an operand
- **Not a Number : NaN (Exponent filed “all1”, Mantisa non 0)**
  - Quiet NaN : propagated through the next operations (ex: 0/0)
  - Signalled NaN : generate an error
- **Signed zero**
  - Signed because of saturation

ARM Cortex-M FPU

Introduction

- **Single precision FPU**
- **Conversion between**
  - Integer numbers
  - Single precision floating point numbers
  - Half precision floating point numbers
- **Handling floating point exceptions** (Untrapped)
- **Dedicated registers**
  - 32 single precision registers (S0-S31) which can be viewed as 16 Doubleword registers for load/store operations (D0-D15)
  - FPSCR for status & configuration

Modifications vs IEEE 754

- **Full Compliance mode**
  - Process all operations according to IEEE 754
- **Alternative Half-Precision format**
  - $(-1)^s \times (1 + \sum(N_i.2^{-i})) \times 2^{16}$ and no de-normalize number support
- **Flush-to-zero mode**
  - De-normalized numbers are treated as zero
  - Associated flags for input and output flush
- **Default NaN mode**
  - Any operation with an NaN as an input or that generates a NaN returns the default NaN

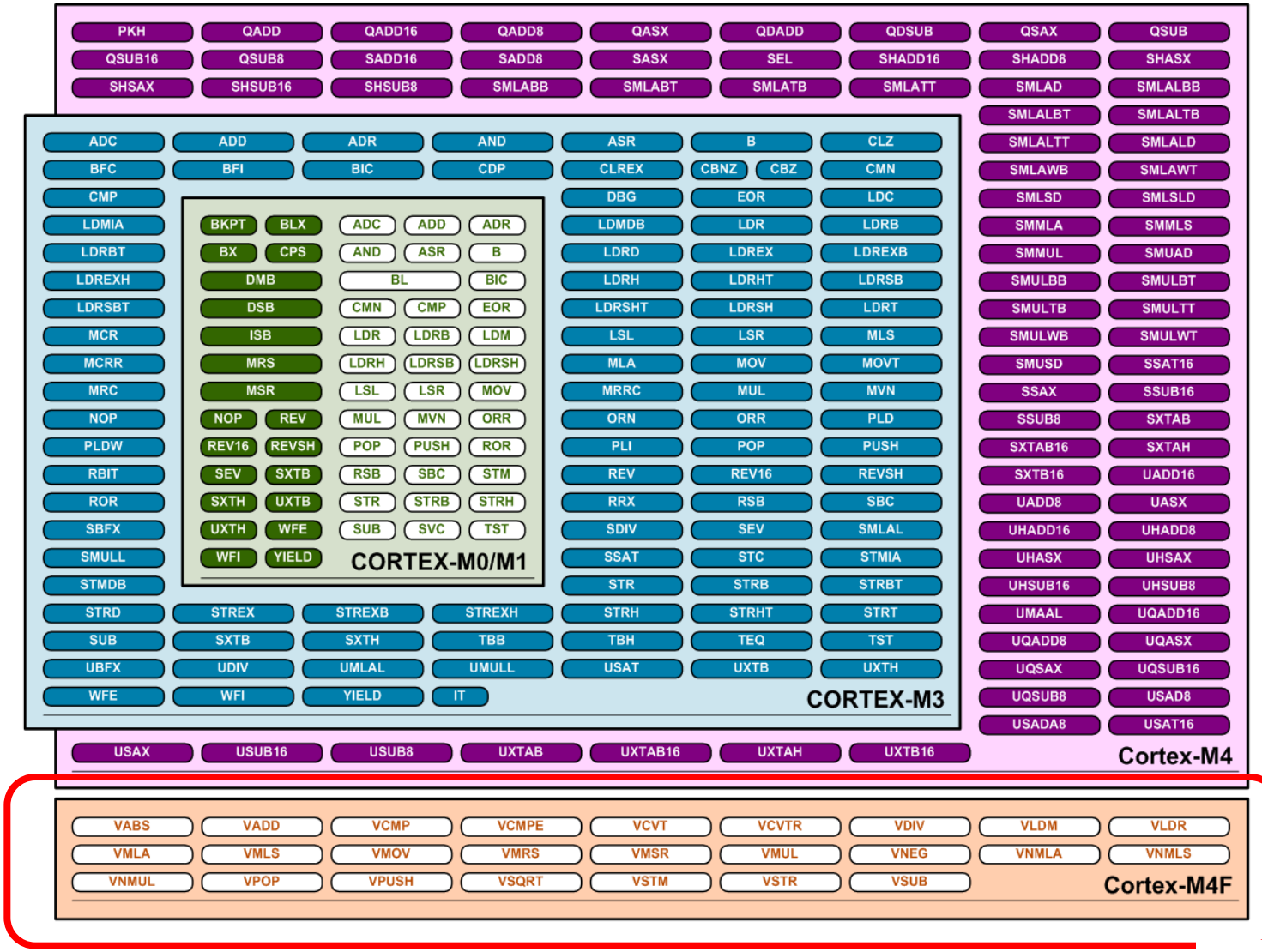
Complete implementation

- **Cortex-M4F does NOT support all operations of IEEE 754-2008**
- **Full implementation is done by software**
- **Unsupported operations**
  - Remainder (% operator)
  - Round FP number to integer-value FP number
  - Binary to decimal conversions
  - Decimal to binary conversions
  - Direct comparison of Single Precision (SP) and Double Precision (DP) values

Floating-Point Status & Control Register

- **Condition code bits**
  - negative, zero, carry and overflow (update on compare operations)
- **ARM special operating mode configuration**
  - half-precision, default NaN and flush-to-zero mode
- **The rounding mode configuration**
  - nearest, zero, plus infinity or minus infinity
- **The exception flags**
  - Inexact result flag may not be routed to the interrupt controller…

FPU instructions



FPU arithmetic instructions

Operation	Description	Assembler	Cycle
Absolute value	of float	VABS.F32	1
Negate	float and multiply float	VNEG.F32 VNMUL.F32	1 1
Addition	floating point	VADD.F32	1
Subtract	float	VSUB.F32	1
Multiply	float then accumulate float then subtract float then accumulate then negate float the subtract the negate float	VMUL.F32 VMLA.F32 VMLS.F32 VNMLA.F32 VNMLS.F32	1 3 3 3 3
Multiply (fused)	then accumulate float then subtract float then accumulate then negate float then subtract then negate float	VFMA.F32 VFMS.F32 VFNMA.F32 VFNMS.F32	3 3 3 3
Divide	float	VDIV.F32	14
Square-root	of float	VSQRT.F32	14

FPU compare & convert instructions

Operation	Description	Assembler	Cycle
Compare	float with register or zero float with register or zero	VCMP.F32 VCMPE.F32	1 1
Convert	between integer, fixed-point, half precision and float	VCVT.F32	1

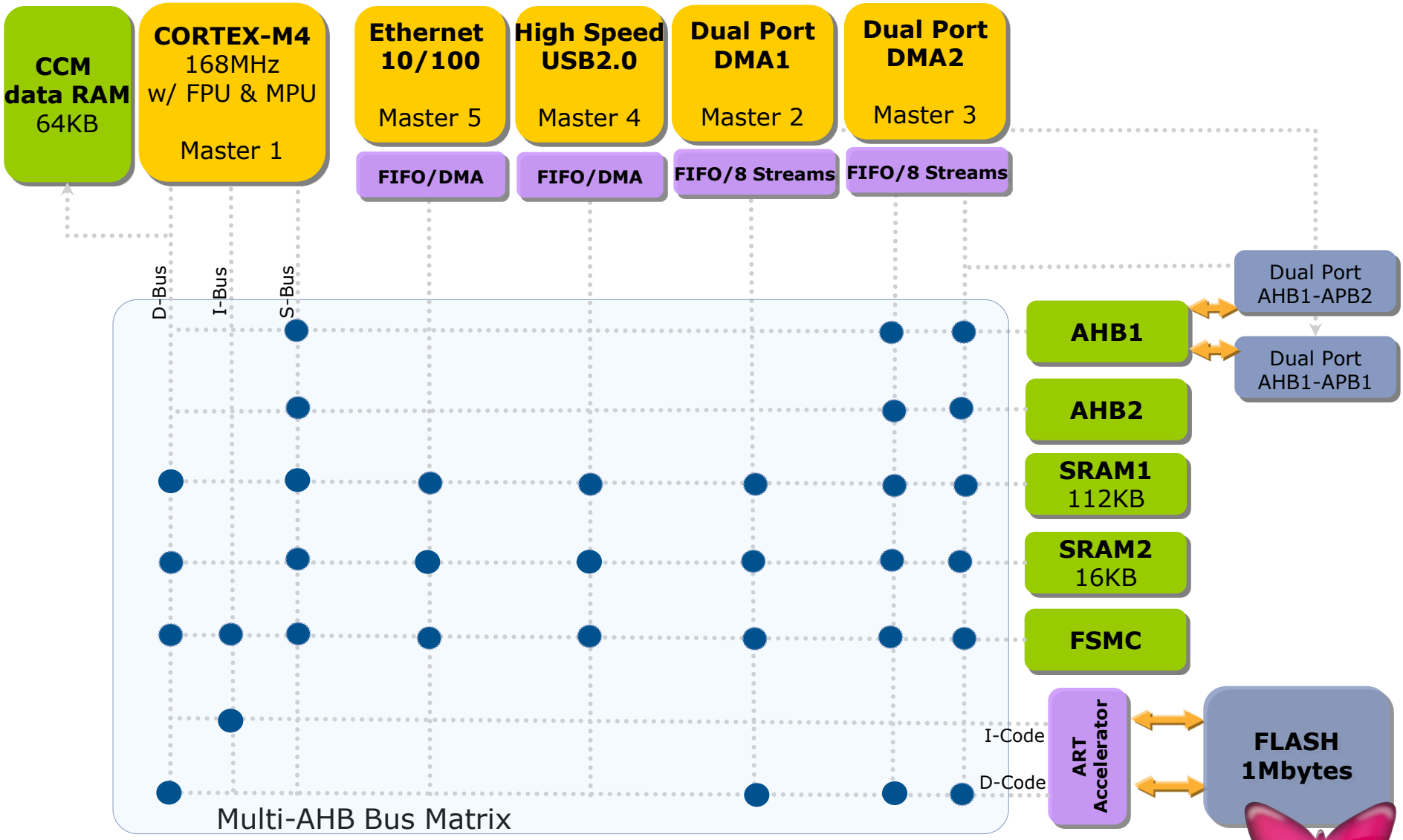
FPU Load/Store Instructions

Operation	Description	Assembler	Cycle
Load	multiple doubles (N doubles)	VLDM.64	1+2*N
	multiple floats (N floats)	VLDM.32	1+N
	single double	VLDR.64	3
	single float	VLDR.32	2
Store	multiple double registers (N doubles)	VSTM.64	1+2*N
	multiple float registers (N doubles)	VSTM.32	1+N
	single double register	VSTR.64	3
	single float register	VSTR.32	2
Move	top/bottom half of double to/from core register	VMOV	1
	immediate/float to float-register	VMOV	1
	two floats/one double to/from core registers	VMOV	2
	one float to/from core register	VMOV	1
	floating-point control/status to core register	VMRS	1
	core register to floating-point control/status	VMSR	1
Pop	double registers from stack	VPOP.64	1+2*N
	float registers from stack	VPOP.32	1+N
Push	double registers to stack	VPUSH.64	1+2*N
	float registers to stack	VPUSH.32	1+N

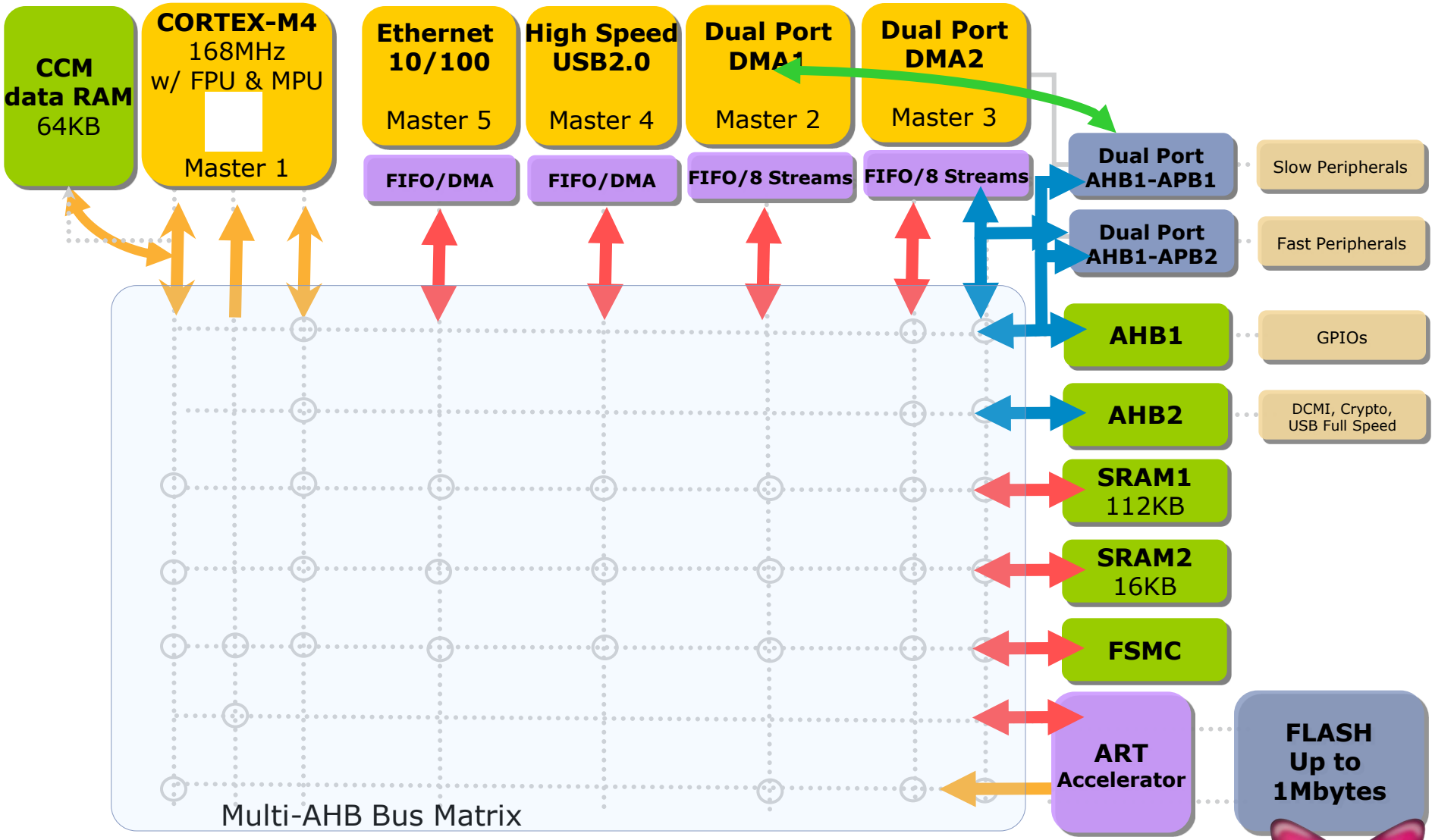
STM32F4xx

System Peripherals

Innovative system Architecture



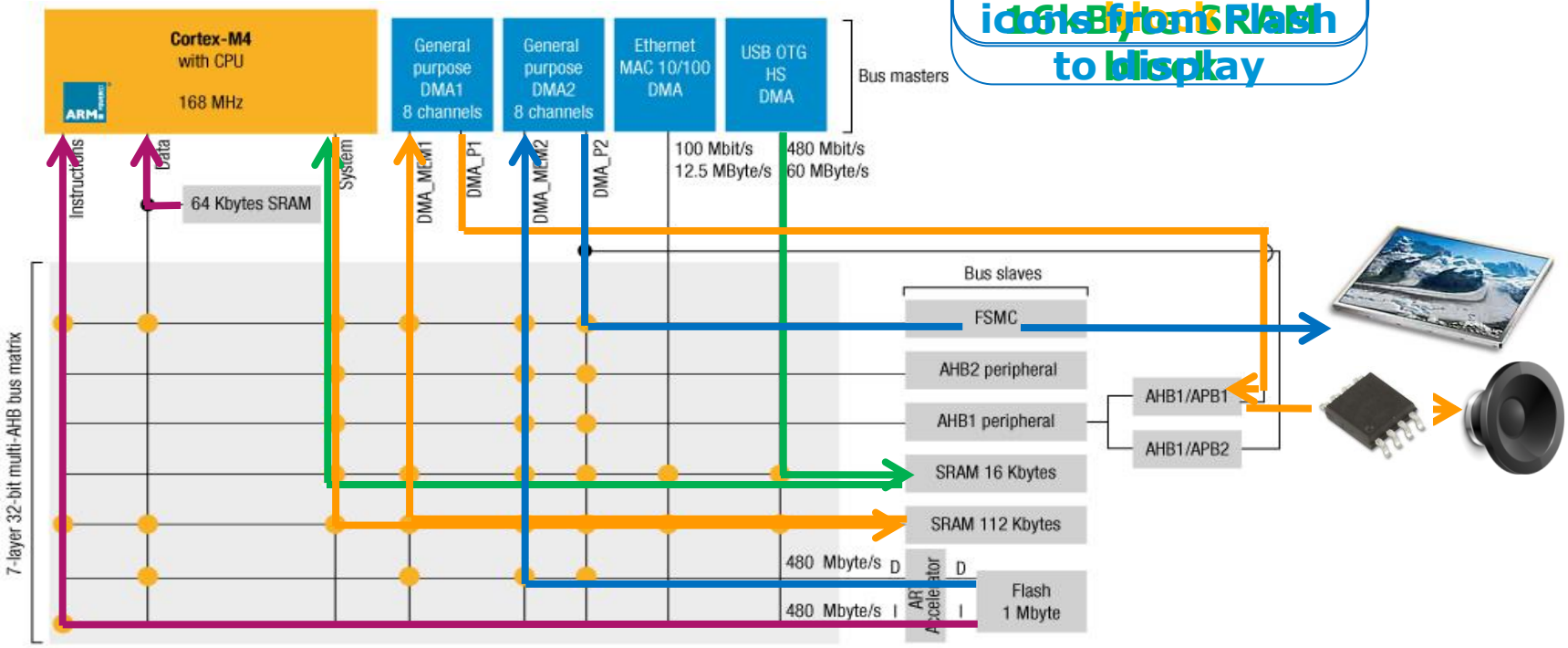
Architecture : CPU, DMA & Multi-Bus Matrix



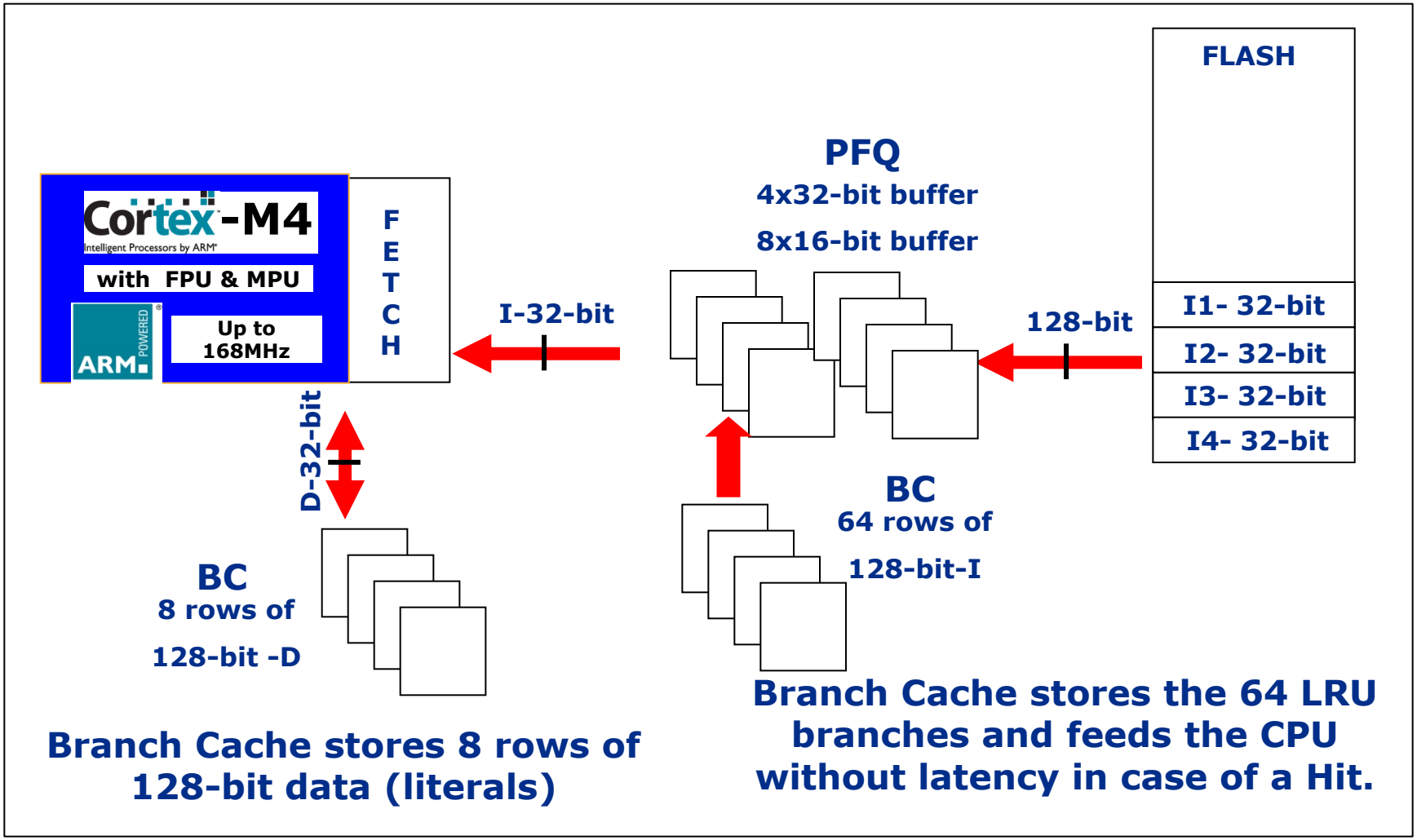
Real-time performance

32-bit multi-AHB bus matrix

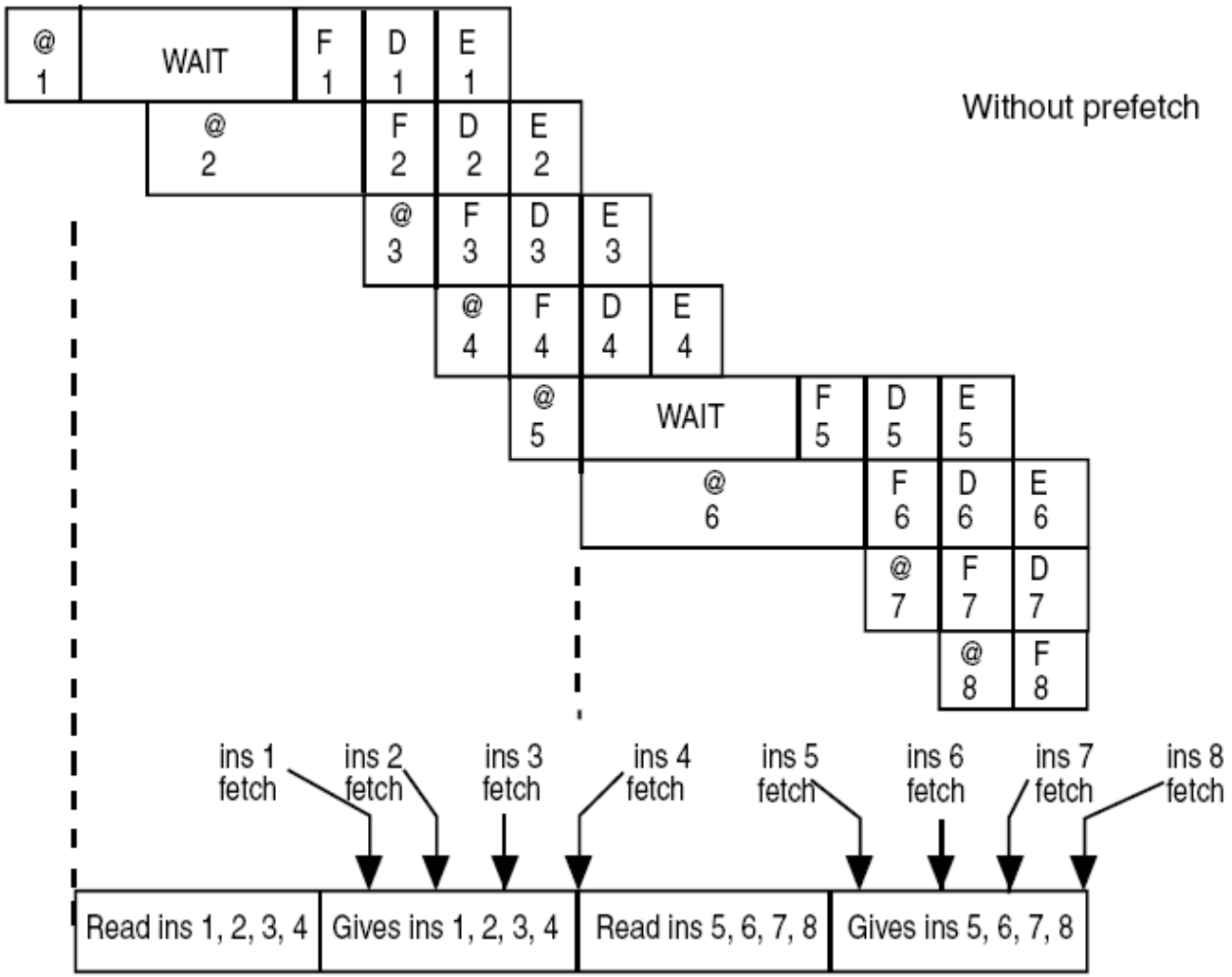
User interface:
DMA transfers
of the graphical
icons from Flash
to display



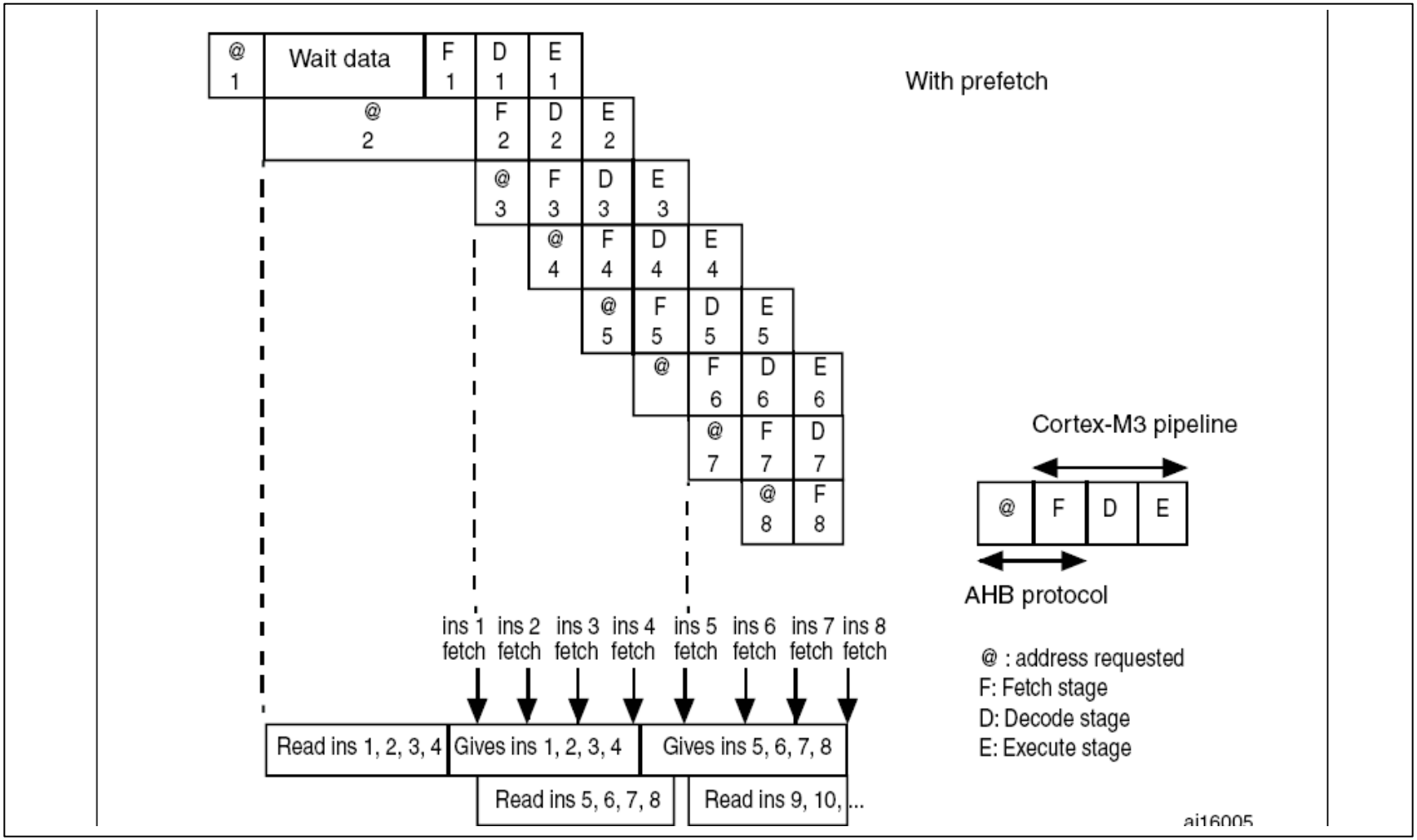
System Architecture – Role of the ART accelerator



System Architecture – Flash performance



System Architecture – Flash performance



System Architecture - Bootloader

BOOT Mode Selection Pins		Boot Mode	Aliasing
BOOT1	BOOT0		
x	0	**Flash memory**	Main Flash memory is selected as boot space
0	1	**System memory**	System memory is selected as boot space
1	1	**Embedded SRAM**	Embedded SRAM is selected as boot space

- The **Bootloader** supports
  - USART1(PA9/PA10)
  - USART3(PC10/PC11 or PB10/PB11)
  - CAN2(PB5PB13)
  - USB OTG FS in Device mode (PA11/PA12) through DFU (device firmware upgrade)

Note

- The DFU/CAN may work w/ different value of external quartz in the range of 4-26 MHz, and the USART uses the internal HSI
- This Bootloader uses the same USART, CAN and DFU protocols as for STM32F2xx/STM32F10x

System Architecture - Boot mode through I-D code bus

- STM32F4xx allows to execute from 3 different memory space mapped on the I-Code/D-Code busses ➔ Faster code execution than System bus

- This is done by SW in SYSCFG_MEMRMP register, 2 bits are used to select the physical remap and so, bypass the BOOT pins.
  - 00: Main Flash memory mapped at 0x0000 0000
  - 01: System Flash memory mapped at 0x0000 0000
  - 10: FSMC (NOR/SRAM bank1 NE1/NE2) mapped at 0x0000 0000
  - 11: Embedded SRAM (112kB) mapped at 0x0000 0000

	BOOT/REMAP in Main Flash memory	BOOT/REMAP in Embedded SRAM	BOOT/REMAP in System memory	REMAP in FSMC
0x2001 C000 - 0x2001 FFFF	SRAM2 (16kB)	SRAM2 (16kB)	SRAM2 (16kB)	SRAM2 (16kB)
0x2000 0000 - 0x2001 BFFF	SRAM1 (112kB)	SRAM1 (112kB)	SRAM1 (112kB)	SRAM1 (112kB)
0x1FFF 0000 - 0x1FFF 77FF	System memory	System memory	System memory	System memory
0x1000 0000 - 0x1000 FFFF	CCM Data RAM (64KB)	CCM Data RAM (64KB)	CCM Data RAM (64KB)	CCM Data RAM (64KB)
0x0810 0000 - 0x0FFF FFFF	Reserved	Reserved	Reserved	Reserved
0x0800 0000 - 0x080F FFFF	FLASH (1MB)	FLASH (1MB)	FLASH (1MB)	FLASH (1MB)
0x0010 0000 - 0x07FF FFFF	Reserved	Reserved	Reserved	FSMC NOR/SRAM 2 Bank1 (Aliased)
0x0000 0000 - 0x000F FFFF	FLASH (1MB) Aliased	SRAM1 (112kB) Aliased	System memory (30KB) Aliased	FSMC NOR/SRAM 1 Bank1 (Alia[illegible]d)

Flash Features Overview

- **Flash Features:**
  - Up to 1MB (sectors 16kB, 64kB and 128kB)
  - Endurance: 10K cycles by sector / 20 years retention
  - 32-bit Word Program time: 12µs(Typ)
- **Flash interface (FLITF) Features:**
  - 128b wide interface with prefetch buffer and data cache, instruction cache
  - Option Bytes loader
  - Flash program/Erase operations
  - Types of Protection:
    - Readout Protection: Level 1 and Level 2 (**JTAG Fuse**)
    - Write Protection (sector by sector)
- **The Information Block consists of:**
  - 30 kB for System Memory : contains embedded Bootloader.
  - 16 B for Small Information block (SIF): contains 8 option bytes + its complementary part (write/read protection, BOR configuration, IWDG configuration, user data)
  - 512 Bytes OTP: one-time programmable

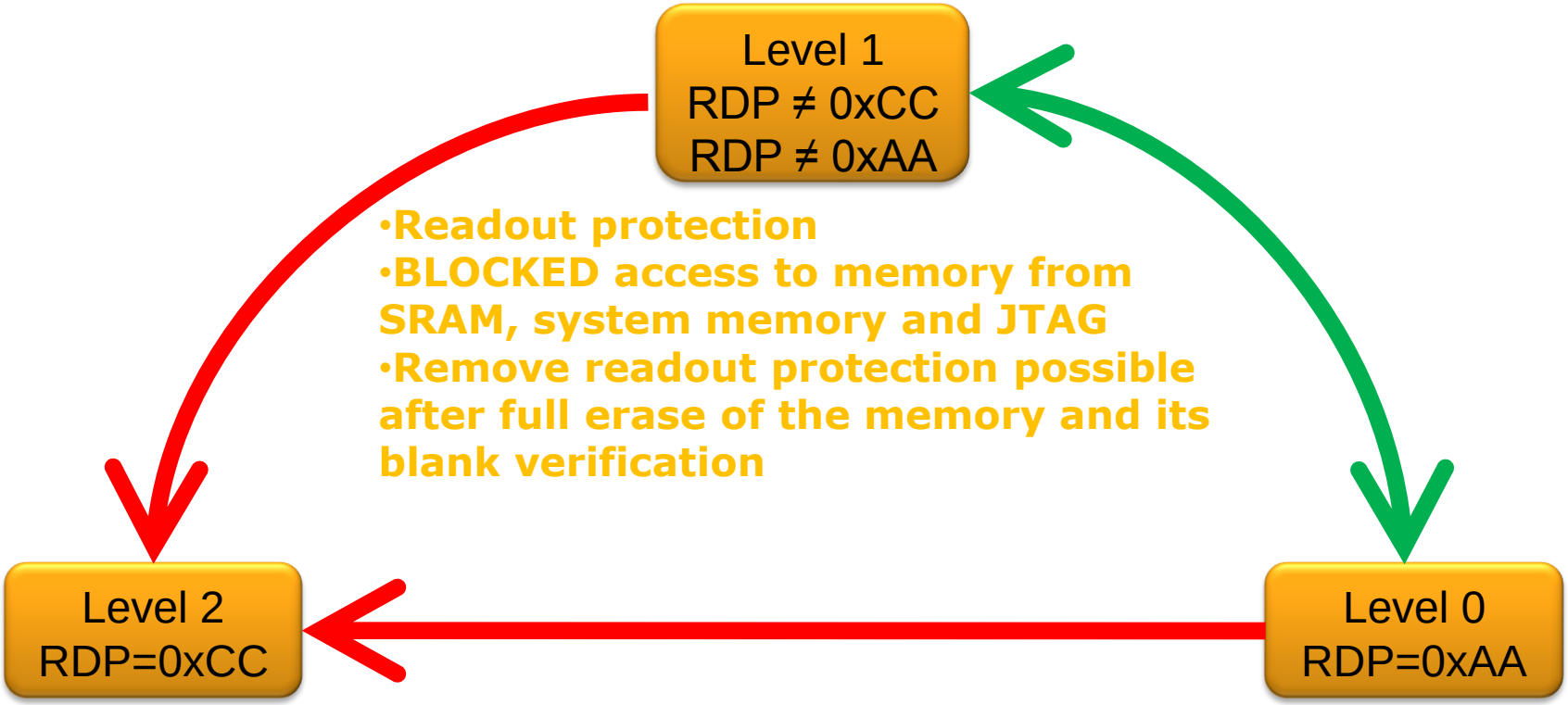
Flash Operations

Relation between CPU clock frequency and Flash memory read time

Wait states(WS) (LATENCY)	HCLK clock frequency (MHz)			
	Voltage range 2.7 V - 3.6 V	Voltage range 2.4 V - 2.7 V	Voltage range 2.1 V - 2.4 V	Voltage range 1.8V - 2.1 V
0WS(1CPU cycle)	0 < HCLK <= 30	0 < HCLK <= 24	0 < HCLK <= 18	0 < HCLK <= 16
1WS(2CPU cycle)	30 < HCLK <= 60	24 < HCLK <= 48	18 < HCLK <= 36	16 < HCLK <= 32
2WS(3CPU cycle)	60 < HCLK <= 90	48 < HCLK <= 72	36 < HCLK <= 54	32 < HCLK <= 48
3WS(4CPU cycle)	90 < HCLK <= 120	72 < HCLK <= 96	54 < HCLK <= 72	48 < HCLK <= 64
4WS(5CPU cycle)	120 < HCLK <= 150	96 < HCLK <= 120	72 < HCLK <= 90	64 < HCLK <= 80
5WS(6CPU cycle)	150 < HCLK <= 168	120 < HCLK <= 144	90 < HCLK <= 108	80 < HCLK <= 96
6WS(7CPU cycle)		144 < HCLK <= 168	108 < HCLK <= 126	96 < HCLK <= 112
7WS(8CPU cycle)			126 < HCLK <= 144	112 < HCLK <= 128

Note: Latency when **VOS** bit in **PWR_CR** is equal to '**1**'

Flash Protections



- **JTAG fuse**
- **No un-protection possible**
- **JTAG disabled**
- **System memory disabled**
- **User settings protected**

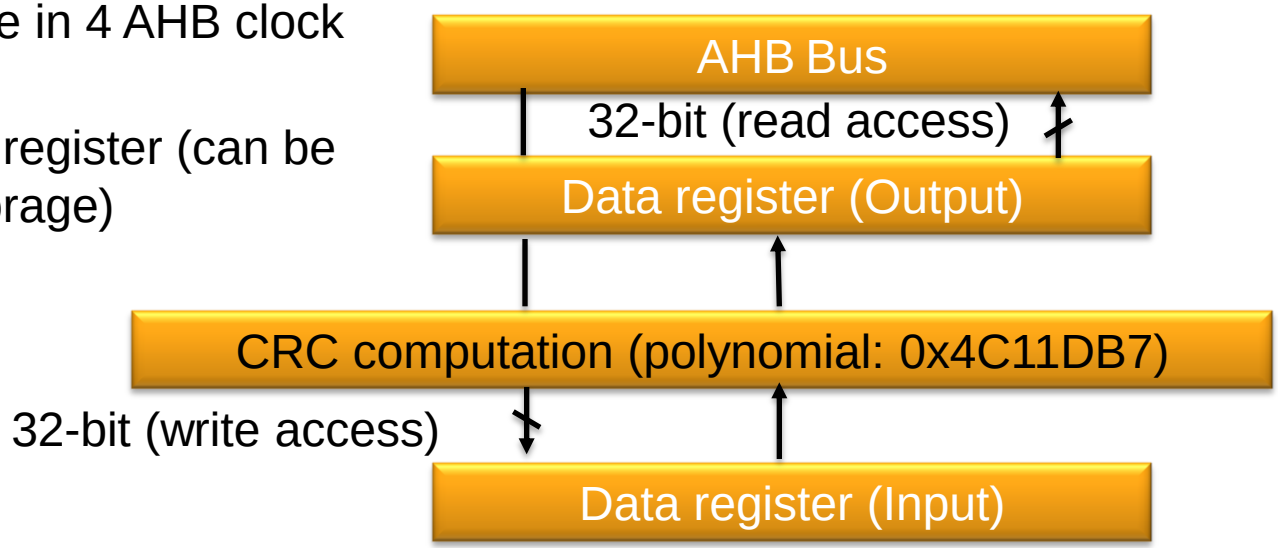
- **No readout protection**
- **Full access to memory from SRAM, system memory and JTAG**

CRC Features

- CRC-based techniques are used to verify data transmission or storage integrity
- Uses CRC-32 (Ethernet) polynomial: 0x4C11DB7

$$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$$

- Single input/output 32-bit data register
- CRC computation done in 4 AHB clock cycles (HCLK)
- General-purpose 8-bit register (can be used for temporary storage)



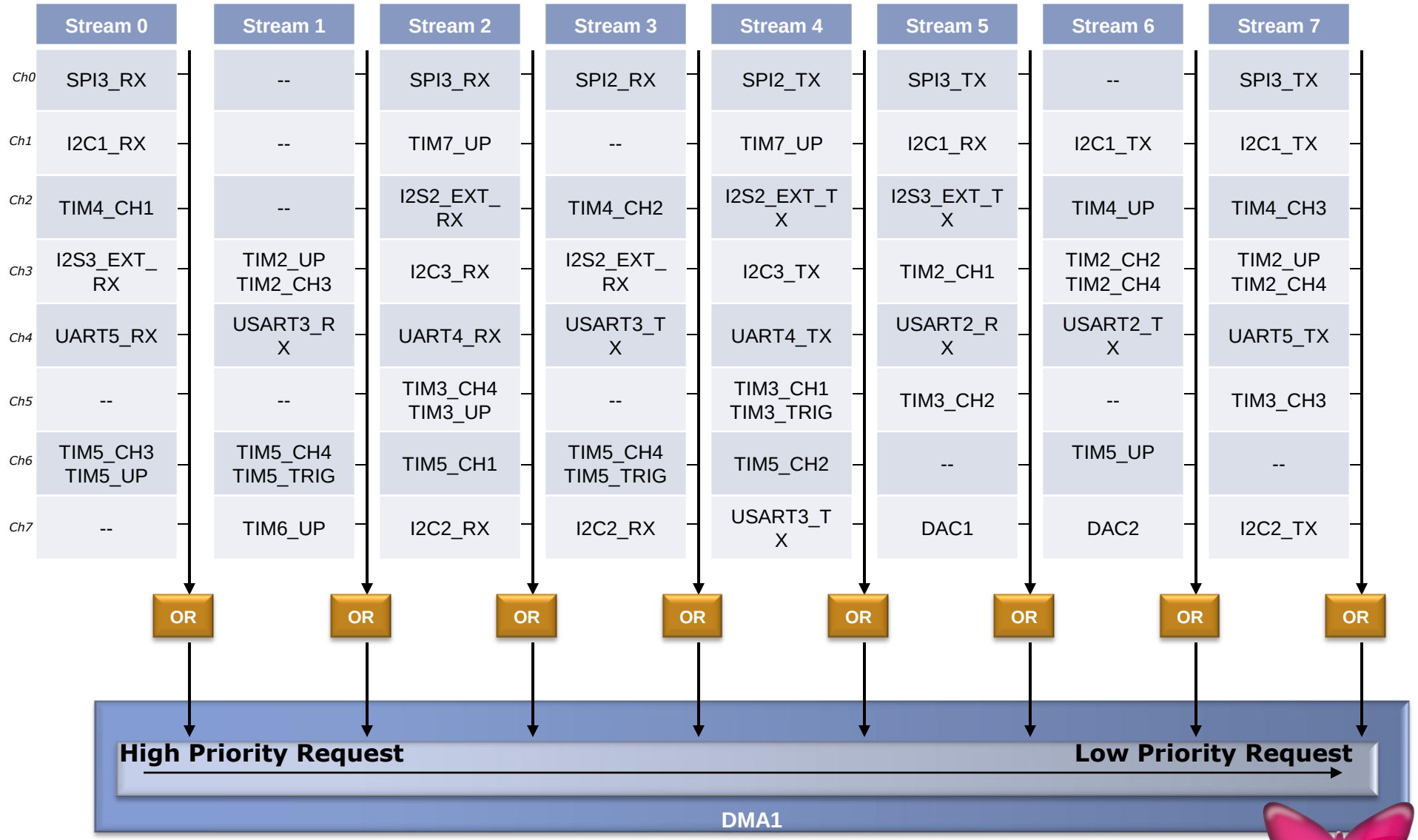
DMA Features

- Dual AHB master bus architecture, one dedicated to memory accesses and one dedicated to peripheral accesses.
- 8 streams for each DMA controller, up to 8 channels (requests) per stream (2 DMA controllers in STM32F4xx family). Channel selection for each stream is software-configurable.
- 4x32-Bits FIFO memory for each Stream (FIFO mode can be enabled or disabled).
- Independent source and destination transfer width (byte, half-word, word): when the source and destination data widths are different, the DMA automatically packs/unpacks data to optimize the bandwidth. (this feature is available only when FIFO mode is enabled)
- Double buffer mode (double buffer mode can enabled or disabled).
- Support software trigger for memory-to-memory transfers (available for the DMA2 controller streams only)

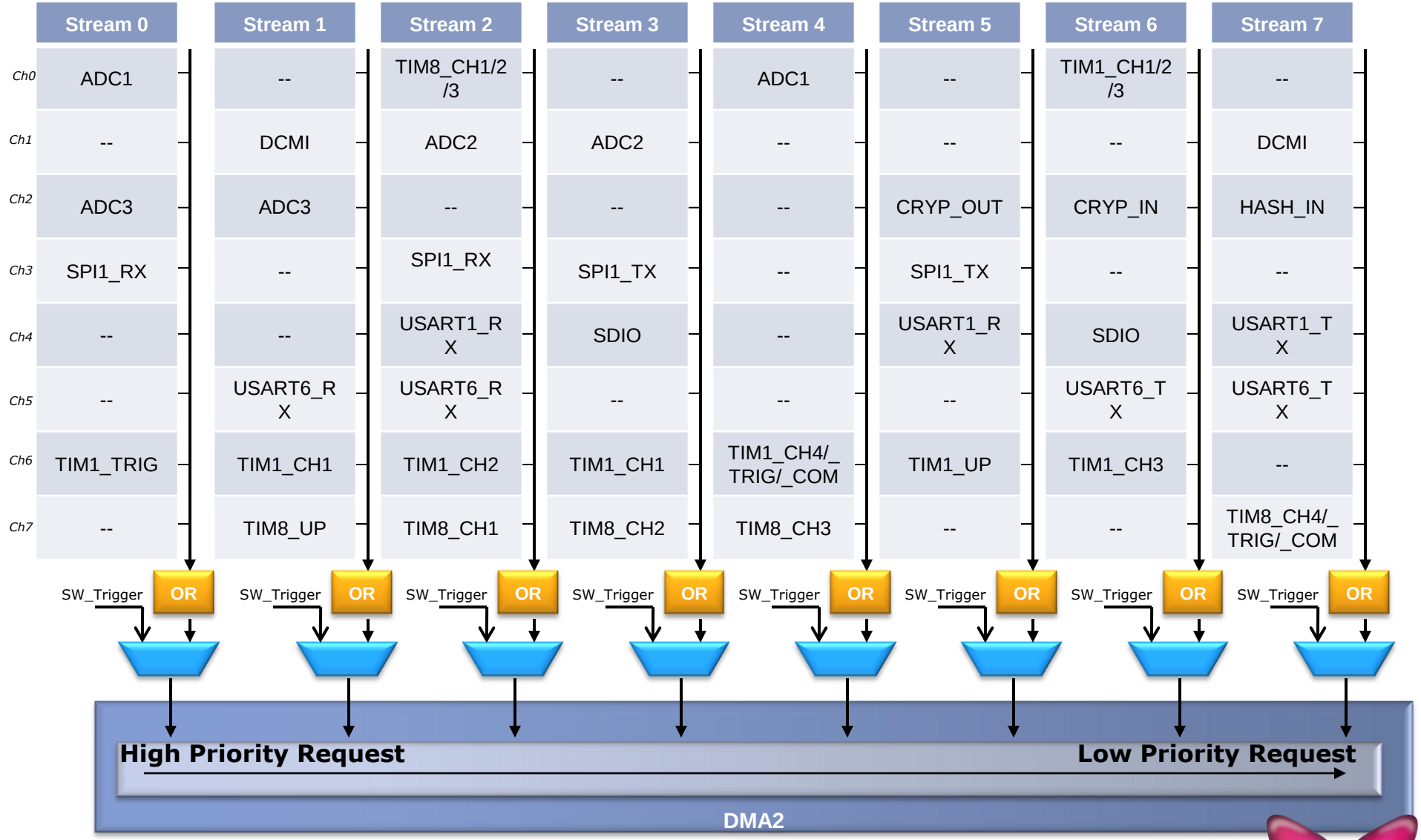
DMA Features

- The **number of data** to be transferred **can be managed either by the DMA controller or by the peripheral**
- Independent Incrementing or Non-Incrementing addressing for source and destination. Possibility to set **increment offset for peripheral address**.
- Supports **incremental burst transfers** of 4, 8 or 16 beats. The size of the burst is software-configurable, usually equal to half the FIFO size of the peripheral
- Each stream supports **circular buffer management**.
- 5 event flags logically ORed together in a single interrupt request for each stream
- **Priorities between DMA** stream requests are **software-programmable**

DMA1 Controller



DMA2 Controller



Streams and Channels configuration

- Each DMA Stream is connected to 8 channels (requests).
- Software selection of which channel should be active for a given stream by setting CHSEL[2:0] bits in DMA_SxCR register.
- Only one Channel can be active for a given Stream.
- A Channel may be not connected to any physical request on the product (ie. DMA1 Stream1 Channel 0).
- A Channel may also be connected to more than one request from the same peripheral (ie. DMA1 Stream1 Channel 4 is connected to TIM2_UP and TIM2_CH3 requests).
- Software requests are used for Memory-to-Memory transfers and are available only on DMA2 controller.

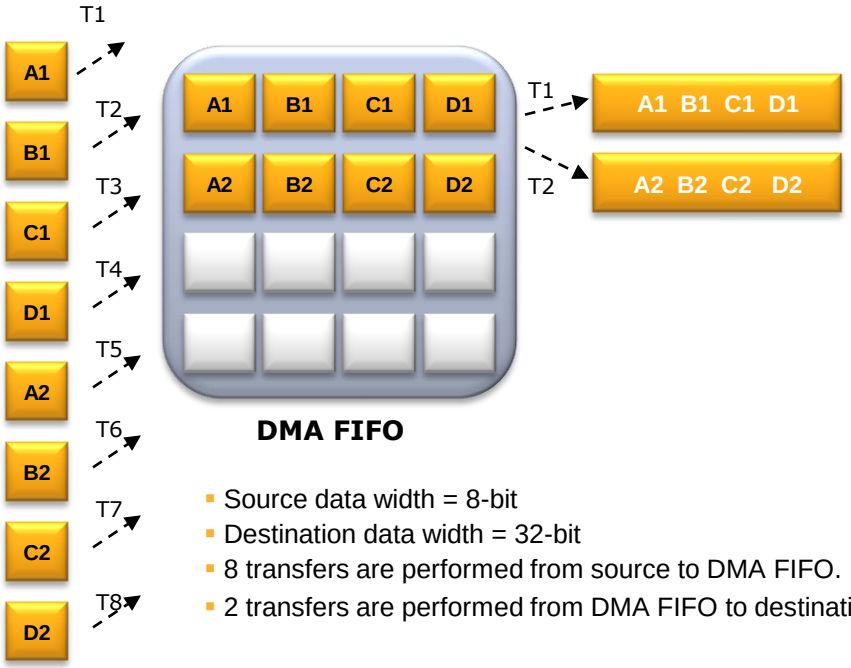
Transfer size and Flow controller

- Either the DMA or the Peripheral determine the amount of data to transfer
  - **DMA is the flow controller**: (to most applied)
    - Number of data items to be transferred is determined by the DMA through the value in register DMA_SxNDTR.
    - DMA_SxNDTR register: from 1 to 65535 bytes/half-words/words and decrements
    - Number of data items is relative only to Peripheral side
      - in Memory-to-Memory mode, the source memory is considered as peripheral
  - **Peripheral is the flow controller: SDIO only**
    - The number of transfers is determined only by the peripheral.
    - Used when the transfer size is unknown to the DMA
    - When transfer is complete, the peripheral sends End of Transfer Signal to DMA when number of transfers is reached.
- **DMA_SxNDTR register can be read when transfer is ongoing to know the remaining number of transfers.**

FIFO: Data Packing/Unpacking

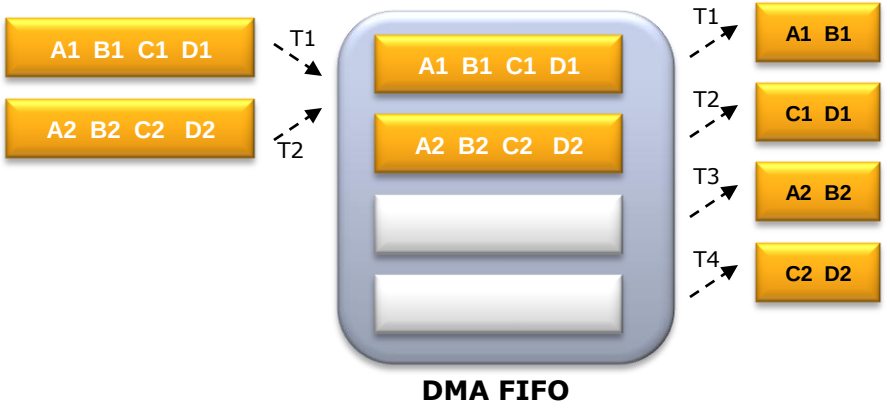
- When FIFO mode is enabled (direct mode disabled) the DMA manage the data format difference between source and destination (data Packing and Unpacking).
- Supported operations:
  - 8-bit / 16-bit → 32-bit / 16-bit (Packing)
  - 32-bit / 16-bit → 8-bit / 16-bit (Unpacking)
- This feature allows to reduce software overhead and CPU load.

Data Packing Example (8-bit → 32-bit)



- Source data width = 8-bit
- Destination data width = 32-bit
- 8 transfers are performed from source to DMA FIFO.
- 2 transfers are performed from DMA FIFO to destination.

Data Unpacking Example (32-bit → 16-bit)



- Source data width = 32-bit
- Destination data width = 16-bit
- 2 transfers are performed from source to DMA FIFO.
- 4 transfers are performed from DMA FIFO to destination.

FIFO: Threshold & Burst mode

- **Threshold:**
  - Threshold level determines when the data in the FIFO should be transferred to/from Memory.
  - There are 4 threshold levels:
    - ¼ FIFO Full ,1/2 FIFO Full, ¾ FIFO Full, FIFO Full
  - When the FIFO threshold is reached, the FIFO is filled/flushed from/to the Memory location.

- **Burst/Single mode:**
  - Burst mode is available only when FIFO mode is enabled (direct mode disabled)
  - Burst mode allows to configure the amount of data to be transferred without CPU/DMA interruption.
  - Available Burst modes:
    - INC4: 1 burst = 4-beats (4 Words, 8 Half-Words or 16 Bytes)
    - INC8: 1 burst = 8-beats (8 Half-Words or 16 Bytes)
    - INC16: 1 burst = 16-beats (16 Bytes)
  - When setting Burst mode, the FIFO threshold should be compatible with Burst size:

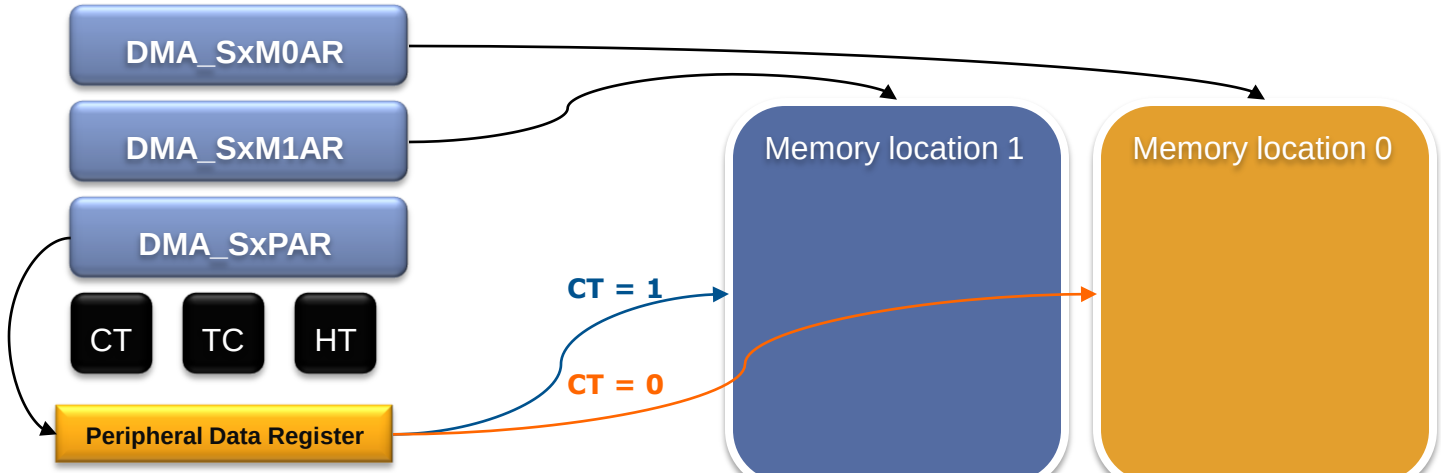
Memory Data Size	Burst Size	Allowed Threshold levels
Byte	4-Beats (INC4)	¼, ½, ¾ and Full
	8-Beats (INC8)	½ & Full
	16-Beats (INC16)	Full
Half-Word	4-Beats (INC4)	½ & Full
	8-Beats (INC8)	Full
Word	4-Beats (INC4)	Full

Notes:

- For Half-Word Memory size, INC16 is not possible.
- For Word Memory size, INC8 and INC16 are not possible.

Circular & Double Buffer modes

- Circular mode:
  - All FIFO features and DMA events (TC, HT, TE) are available in this mode.
  - The number of data items is automatically reloaded and transfer restarted
  - This mode is NOT available for Memory-to-Memory transfers .
- Double Buffer mode: (circular mode only)
  - Two Memory address registers are available (DMA_SxM0AR & DMA_SxM1AR)
  - Allows switch between two Memory buffers to be managed by hardware.
  - Memory-to-Memory mode is not allowed
  - A flag & control bit (CT) is available to monitor which destination is being used for data transfer.
  - TC flag is set when transfer to memory location 0 or 1 is complete.

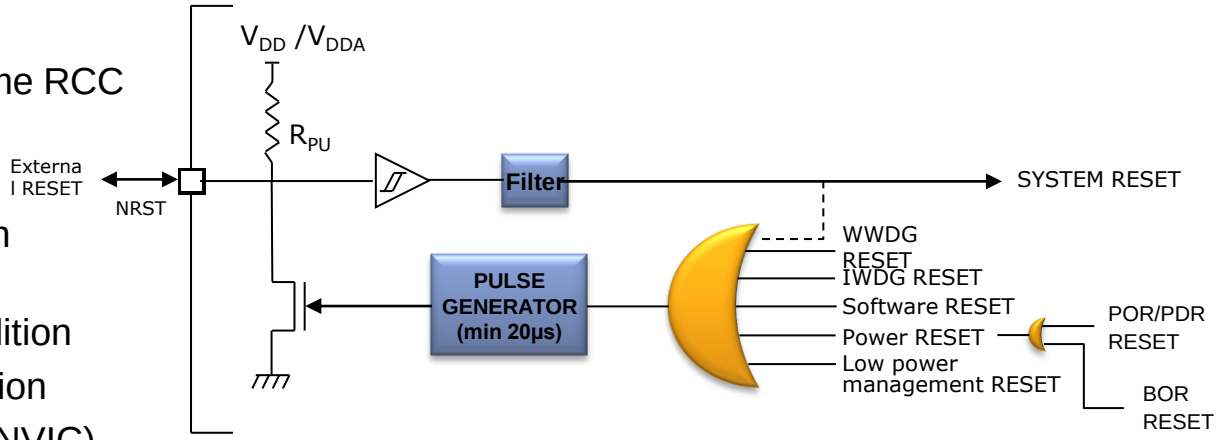


Transfer modes summary

DMA transfer mode	Flow Controller	Circular mode	Transfer Type	Direct Mode	Double Buffer mode
Peripheral-to-Memory	DMA	Possible	Single	Possible	Possible
			Burst	Forbidden	
	Peripheral	Forbidden	Single	Possible	Forbidden
			Burst	Forbidden	
Memory-to-Peripheral	DMA	Possible	Single	Possible	Possible
			Burst	Forbidden	
	Peripheral	Forbidden	Single	Possible	Forbidden
			Burst	Forbidden	
Memory-to-Memory	DMA	Forbidden	Single	Forbidden	Forbidden
			Burst		

RESET Sources

- System RESET
  - Resets all registers except some RCC registers and Backup domain
  - Sources
    - Low level on the NRST pin (External Reset)
    - WWDG end of count condition
    - IWDG end of count condition
    - A software reset (through NVIC)
    - Low power management Reset

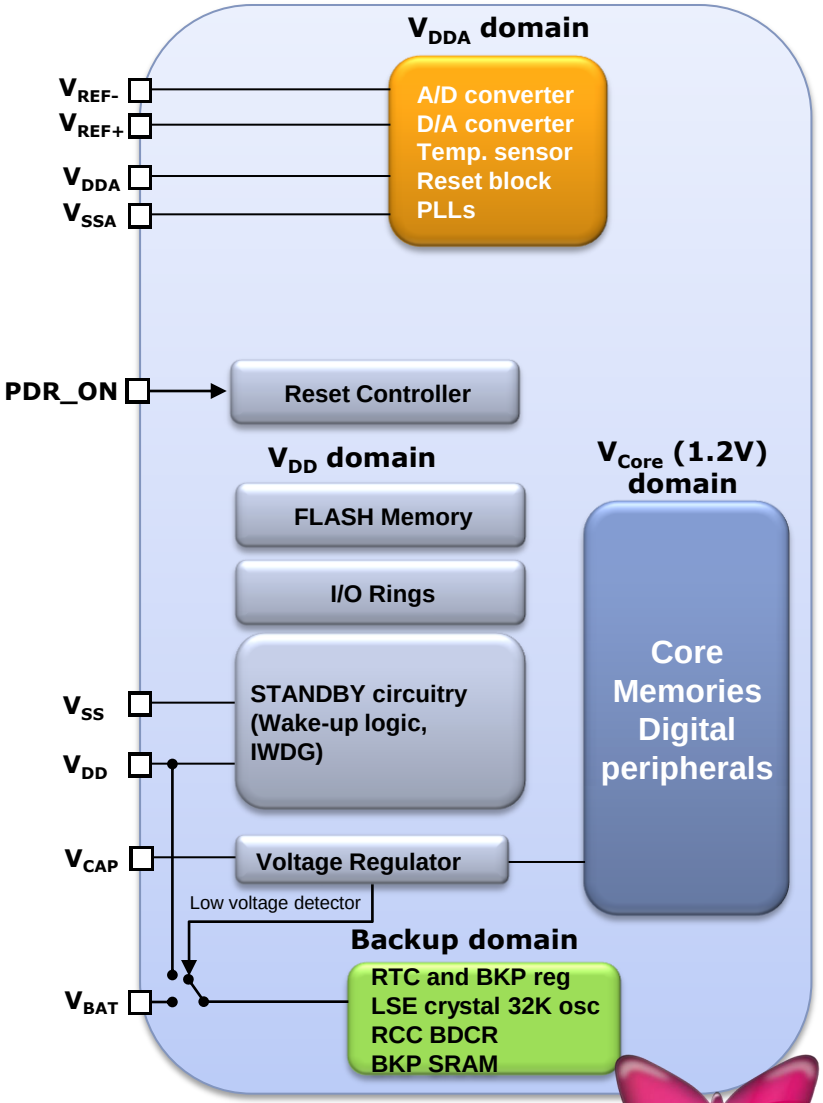


- Power RESET
  - Resets all registers except the Backup domain
  - Sources
    - Power On/Power down Reset (POR/PDR)
    - BOR
    - Exit from STANDBY

- Backup domain RESET
  - Resets in the Backup domain: RTC registers + Backup Registers + RCC BDCR register
  - Sources
    - BDRST bit in RCC BDCR register
    - POWER Reset

Power Supply

- V_{DD} = 1.8 V to 3.6 V. External Power Supply for I/Os and the internal regulator. The supply voltage can drop to 1.7 when the PDR_ON is connected to VSS and the device operates in the 0 to 70°C.
- V_{DDA} = 1.8 V to 3.6 V : External Analog Power supplies for ADC, DAC, Reset blocks, RCs and PLLs.
- V_{CAP} = Voltage regulator external capacitors (also 1.2V supply in Regulator bypass mode)
- V_{BAT} = 1.65 to 3.6 V: power supply for Backup domain when V_{DD} is not present.
- Power pins connection:
  - V_{DD} and V_{DDA} must be connected to the same power source
  - V_{SS}, V_{SSA} must be tight to ground
  - 2.4V ≤ V_{REF+} ≤ V_{DDA} when V_{DDA} ≥ 2.4
  - V_{REF+} = V_{DDA} when V_{DDA} < 2.4



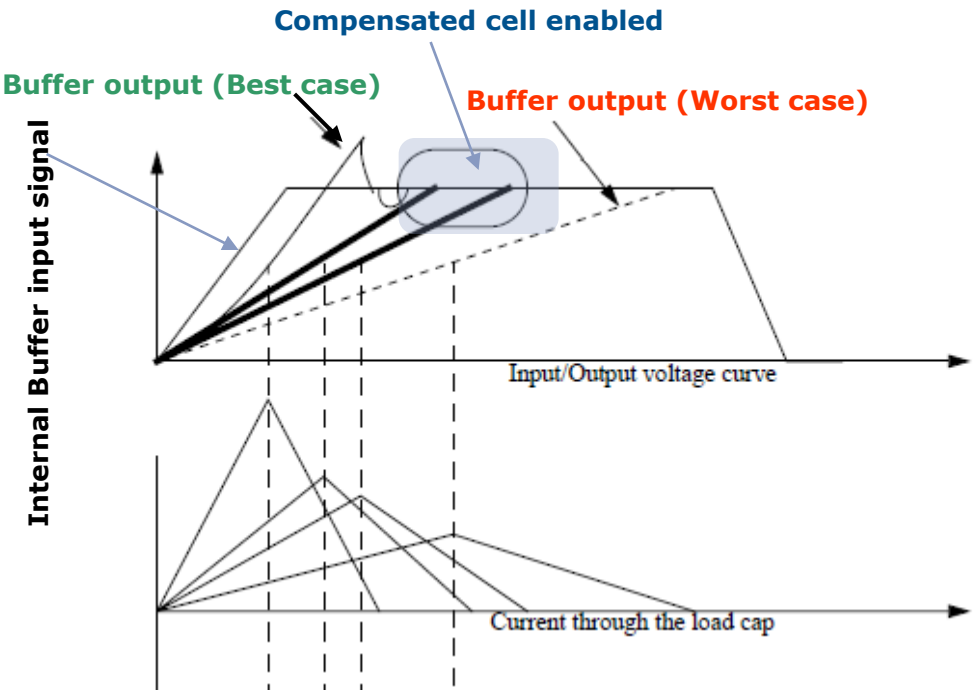
Limitations depending on the operating power supply range

Limitations depending on the operating power supply range					
Operating power supply range	**ADC operation**	**Maximum CPU frequency (fCPUmax)**	**I/O operation**	**FSMC controller operation**	**Possible Flash memory operations**
V_{DD} = 1.8 to 2.1 V	Conversion time up to **1 .2Msps**	– **128 MHz** with 7 Flash memory wait states – **16MHz** with no Flash memory wait state	– Degraded speed performance – No I/O compensation	up to **30 MHz**	**8-bit** erase and program operations only
V_{DD} = 2.1 V to 2.4 V	Conversion time up to **1.2 Msps**	– **138 MHz** with 7 Flash memory wait states – **18MHz** with no Flash memory wait state	– Degraded speed performance – No I/O compensation	up to **30 MHz**	**16-bit** erase and program operations
V_{DD} = 2.4 V to 2.7 V	Conversion time up to **2 .4Msps**	– **168 MHz**(*) with 6 Flash memory wait states – **24MHz** with no Flash memory wait state	– Degraded speed performance – I/O compensation works	up to **48 MHz**	**16-bit** erase and program operations
V_{DD} = 2.7 V to 3.6 V	Conversion time up to **2.4 Msps**	– **168 MHz**(*) with 5 Flash memory wait states – **30MHz** with no Flash memory wait state	– Full-speed operation – I/O compensation works	– up to **60 MHz** when VDD = 3.0 V to 3.6 V – up to **48 MHz** when VDD =2.7 V to 3.0 V	**32-bit** erase and program operations

(*) when **VOS** bit in **PWR_CR** is equal to '**1**'

What is the I/O Compensation cell?

- An automatic slew rate adjustment depending on PVT for high frequency IO toggling:
  - IOs Rise and fall edge slopes are adjusted according to Power Voltage and Temperature
  - Recommended for 50MHz/100MHz drive IO settings
  - Allows a greater I/O noise immunity from VDD
- Feature is enabled by software (disabled by default) and switched off in low power mode
- Enabled with VDD operating range: 2.4V-3.6V
- Power consumption is increased when enabled



Voltage Regulators (1/2)

- 2 voltage regulators are embedded
  - **A Main linear voltage regulator** supplies all the digital circuitries (except for the Standby circuitry and Backup domain). The regulator output voltage (V_{CORE}) is 1.2 V (typical) and can supply up to 200mA.
  - **Low voltage regulator** exclusively for the backup RAM (in VBAT mode)

- The Main Voltage regulator has three different modes
  - Run and Sleep modes (200mA max)
  - Low power mode for STOP mode (5mA max)
  - Regulator OFF in STANDBY/VBAT mode.

- Regulator bypass mode
  - It allows to supply externally a 1.2 V voltage source through V_{CAP_1} and V_{CAP_2} pins, in addition to a second external V_{DD} supply source.

Voltage Regulators (2/2)

- In order to achieve a tradeoff between performance and power consumption, ‘VOS’ dedicated bit in ‘PWR_CR’ register, allows to controls the main internal voltage regulator output voltage

Condition	Max AHB clock frequency
VOS bit in **PWR_CR** register equal to ‘**0**’	**144 MHz**
VOS bit in **PWR_CR** register equal to ‘**1**’	**168 MHz**

- The voltage scaling allows to optimize the power consumption when the device is clocked below the maximum system frequency.

Voltage Regulator Bypass

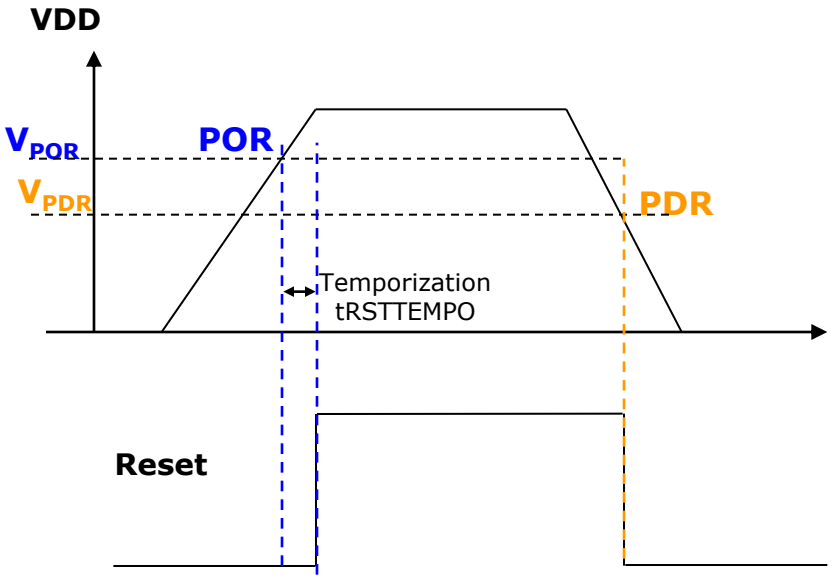
- Available only on CSP64 and BGA176 packages.
  - CSP by bonding option
  - BGA dedicated pin “Bypass-Reg”

- Power consumption gains, but…
  - Need to control the 1.2V logic circuitry “by hand”
    - PA0 pin dedicated to reset the 1.2V logic.
    - VDD should always be higher than VDD12
    - If VDD12=1.08V supply slope faster than VDD=1.8V supply
      - can just connect PA0 to NRST
    - Otherwise reset sequence should be controlled externally
      - PA0 should be asserted low until VDD12 =1.08V.
  - Standby mode not allowed

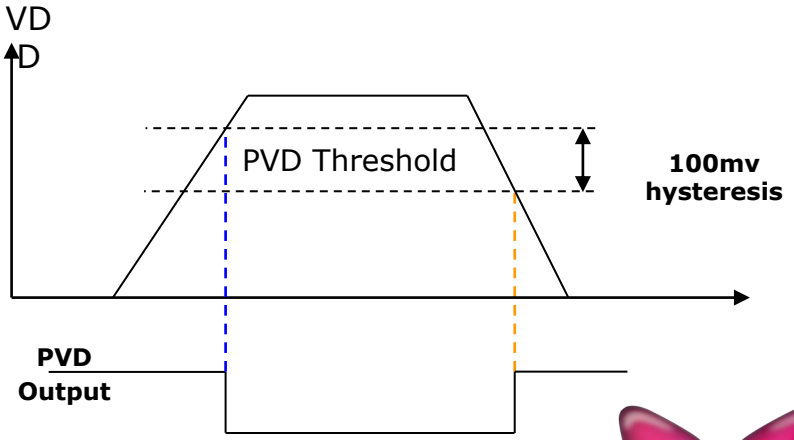
Power supply monitoring POR, PDR, PVD

- Integrated POR / PDR circuitry:
  - For devices operating from 1.8 to 3.6 V, there is no BOR and the reset is released when V_{DD} goes above POR level and asserted when V_{DD} goes below PDR level
  - POR and PDR have 40mV hysteresis
- Programmable Voltage Detector
  - Enabled by software
  - Monitor the V_{DD} power supply by comparing it to a threshold
  - Threshold configurable from 1.9V to 3.1V by step of 100mV
  - Generate interrupt through EXTI Line16 (if enabled) when VDD < Threshold and/or VDD > Threshold.

Can be used to generate a warning message and/or put the MCU into a safe state

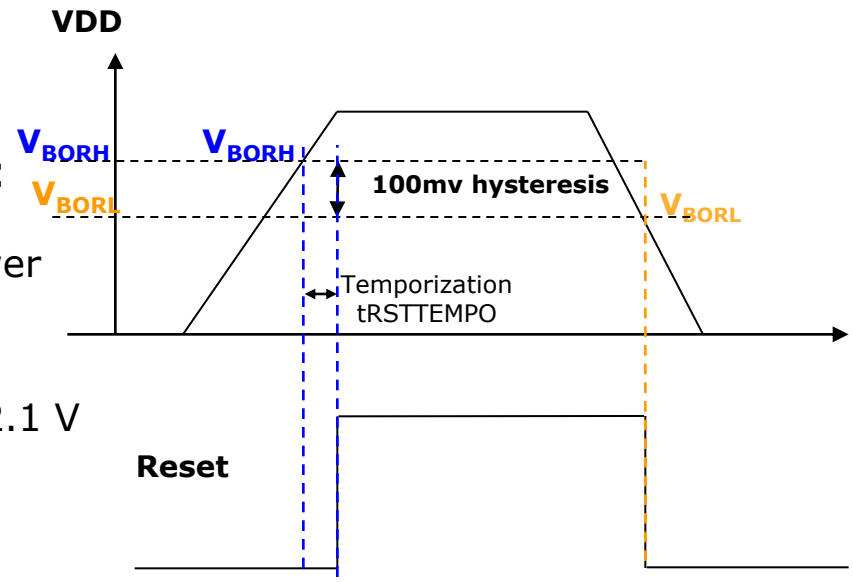


$V_{PDR} = V_{POR} = 1.8V$



Brown Out Reset (BOR)

- During power on, the Brown out reset (BOR) keeps the device under reset until the supply voltage reaches the specified V_{BOR} threshold.
  - No need for external reset circuit
- BOR have a typical hysteresis of 100mV
- BOR Levels are configurable by option bytes:
  - BOR OFF: 2.1 V at power on and 1.62 V at power down
  - BOR LOW (DEFAULT) : 2.4 V at power on and 2.1 V at power down
  - BOR MEDIUM: 2.7 V at power on and 2.4 V at power down
  - BOR HIGH: 3.6 V at power on and 2.7 V at power down



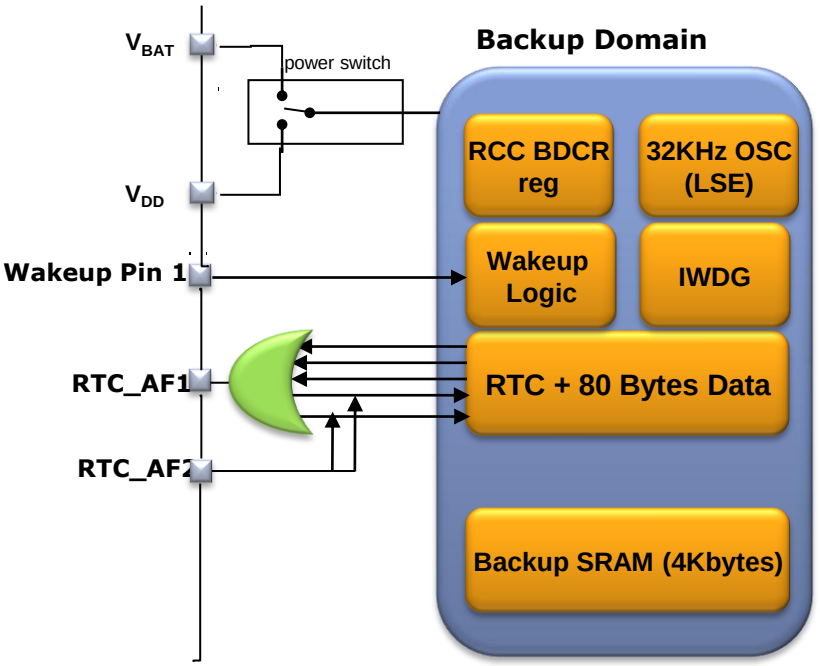
Supply monitoring and Reset circuitry

At startup:

- POR/PDR - Always ON (or CSP64 bounding option)
- Brown Out Reset (BOR) - Always ON (can be switched off after option byte loading)
- Programmable Voltage Detection (PVD) – ON/OFF.
  - PVD enable/disable bit is controlled by software via a dedicated bit (PVDE).

Backup Domain

- Backup Domain
  - RTC unit and 4KB Backup RAM
  - LVR for the backup RAM (with switch off option)
- VBAT independent voltage supply
  - Automatic switch-over to V_{BAT} when V_{DD} goes below PDR level
    - No current sunk on V_{BAT} when V_{DD} present
    - Prevent from power line down
- 1 Wakeup pin and 2 RTC Alternate functions pins (RTC_AF1 and RTC_AF2)
- Backup SRAM
  - 4 kB of backup SRAM accessible only from the CPU
  - Can store sensitive data (crypto keys)
  - Backup SRAM is powered by a dedicated low power regulator in V_{BAT} mode. Its content is retained even in Standby and V_{BAT} mode when the low power backup regulator is enabled.
  - The backup SRAM is not mass erased by an tamper event.



STM32F4xx Low power modes features

- The STM32F4xx features 3 low power modes
  - **SLEEP** (core stopped, peripherals running) ~2mA @2MHz (38mA @120MHz)
  - **STOP** (clocks stopped, RAM, registers kept) ~1mA current consumption
  - **STANDBY** (only backup domain kept, return via RESET)
- VBAT mode (like in STANDBY mode).

- The STM32F4xx features options to decrease the consumption during low power modes
  - Peripherals clock stopped automatically during sleep mode (S/W)
  - Flash Power Down mode
  - LVR and Backup RAM disable option

- The STM32F4xx features many sources to wakeup the system from low power modes:
  - Wakeup pin (PA0) / NRST pin
  - RTC Alarm (Alarm A and Alarm B)
  - RTC Wakeup Timer interrupt
  - RTC Tamper events
  - RTC Time Stamp Event
  - IWDG Reset event

Wakeup time from Low Power Modes

Low power mode	Conditions	Wakeup time in µs
Sleep mode		1 Typ
Stop mode	regulator in Run mode	13 Typ
Stop mode	regulator in low power mode	17 Typ
Stop mode	regulator in low power mode and Flash in Deep power down mode	110 Typ
Standby mode		375 Typ

STM32F4 - clock features

Four oscillators on board

- **HSE** (High Speed External Osc) 4..26MHz (can be bypassed by and ext. Oscillator)
- **HSI** (High Speed Internal RC): factory trimmed internal RC oscillator 16MHz +/- 1
- **LSI** (Low Speed Internal RC): 32kHz internal RC used for **IWDG**, optionally RTC and AWU
- **LSE** (Low Speed External oscillator): 32.768kHz osc (can be bypassed by an external Osc)
  - precise time base with very low power consumption (max 1µA).
  - optionally drives the RTC for Auto Wake-Up (AWU) from STOP/STANDBY mode.

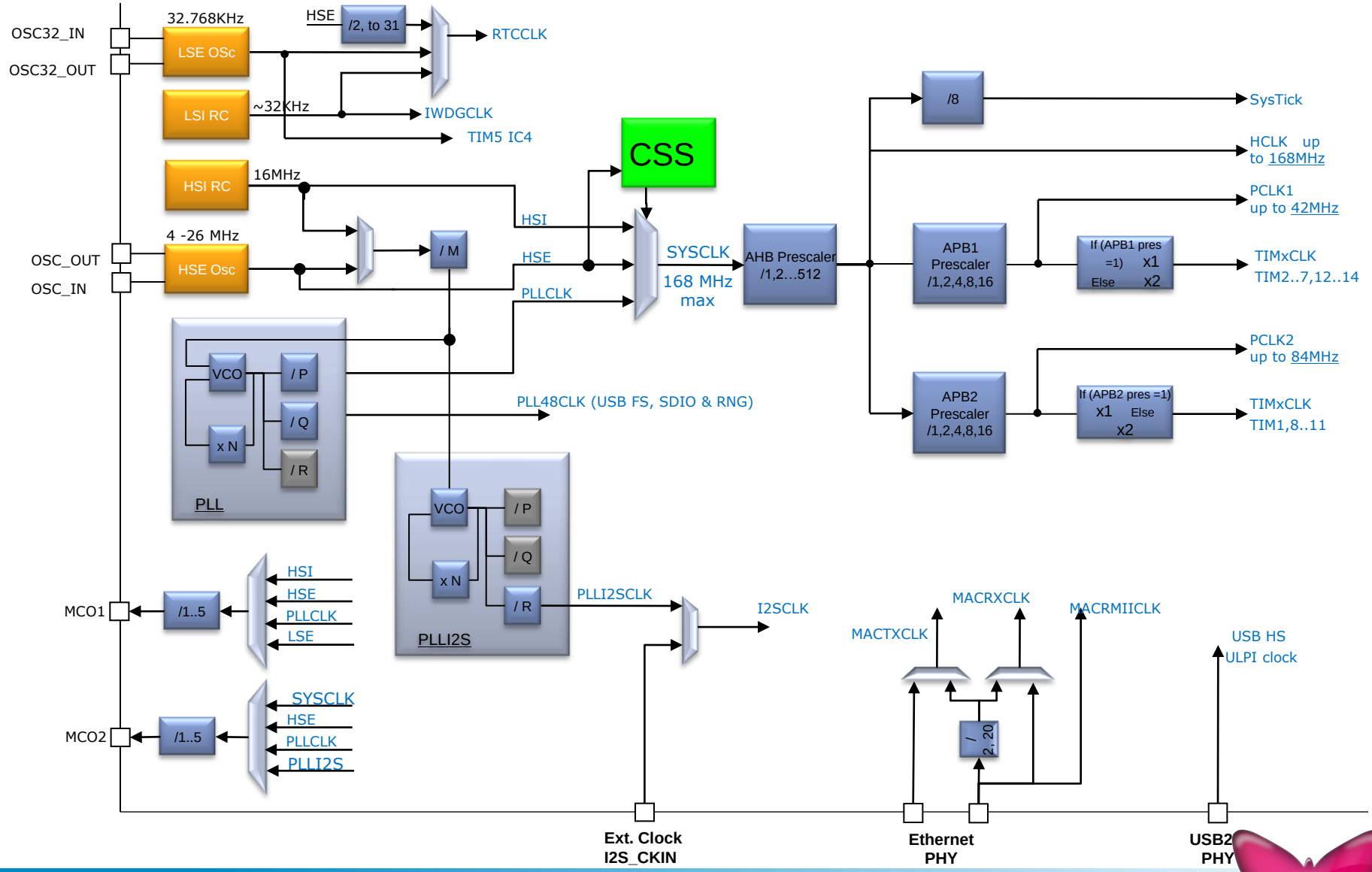
Two PLLs

- **Main PLL (PLL)** clocked by HSI or HSE used to generate the System clock (up to 168MHz), and 48 MHz clock for USB OTG FS, SDIO and RNG. PLL input clock in the range 1-2 MHz.
- **PLLI2S PLL (PLLI2S)** used to generate a clock to achieve HQ audio performance on the I^2S interface.

More security

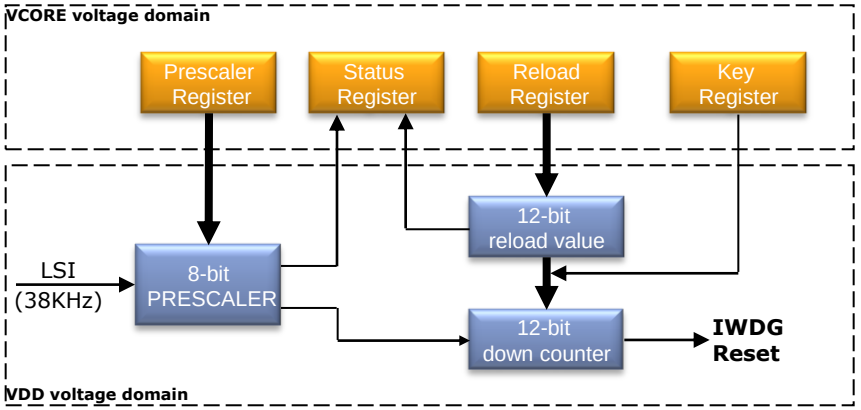
- **Clock Security System (CSS,** enabled by software**)** to backup clock in case of HSE clock failure (HSI feeds the system clock) – linked to Cortex NMI interrupt
- **Spread Spectrum Clock Generation** (SSCG, enabled by software) to reduce the spectral density of the electromagnetic interference (EMI) generated by the device

STM32F4 - clock scheme

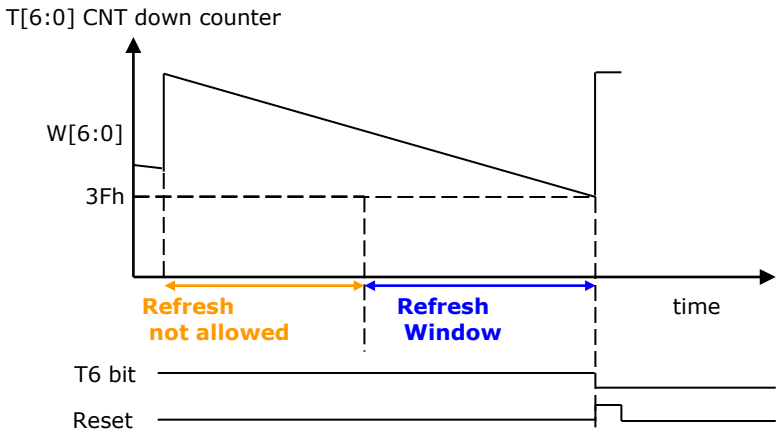


Watchdogs

- **Independent Watchdog (IWDG)**
  - Dedicated low speed clock (LSI)
  - HW and SW way of enabling
  - IWDG clock still active if main clock fails
  - Still functional in Stop/Standby
  - Wake-up from stop/standby
  - Min-max Timeout values *125us …32.7s*



- **Window Watchdog (WWDG)**
  - Configurable Time Window
  - Can detect abnormally early or late application behavior
  - Conditional Reset
  - WWDG Reset flag
  - Timeout value @42MHz (PCLK1): *97.52us … 49.93ms*



STM32F4

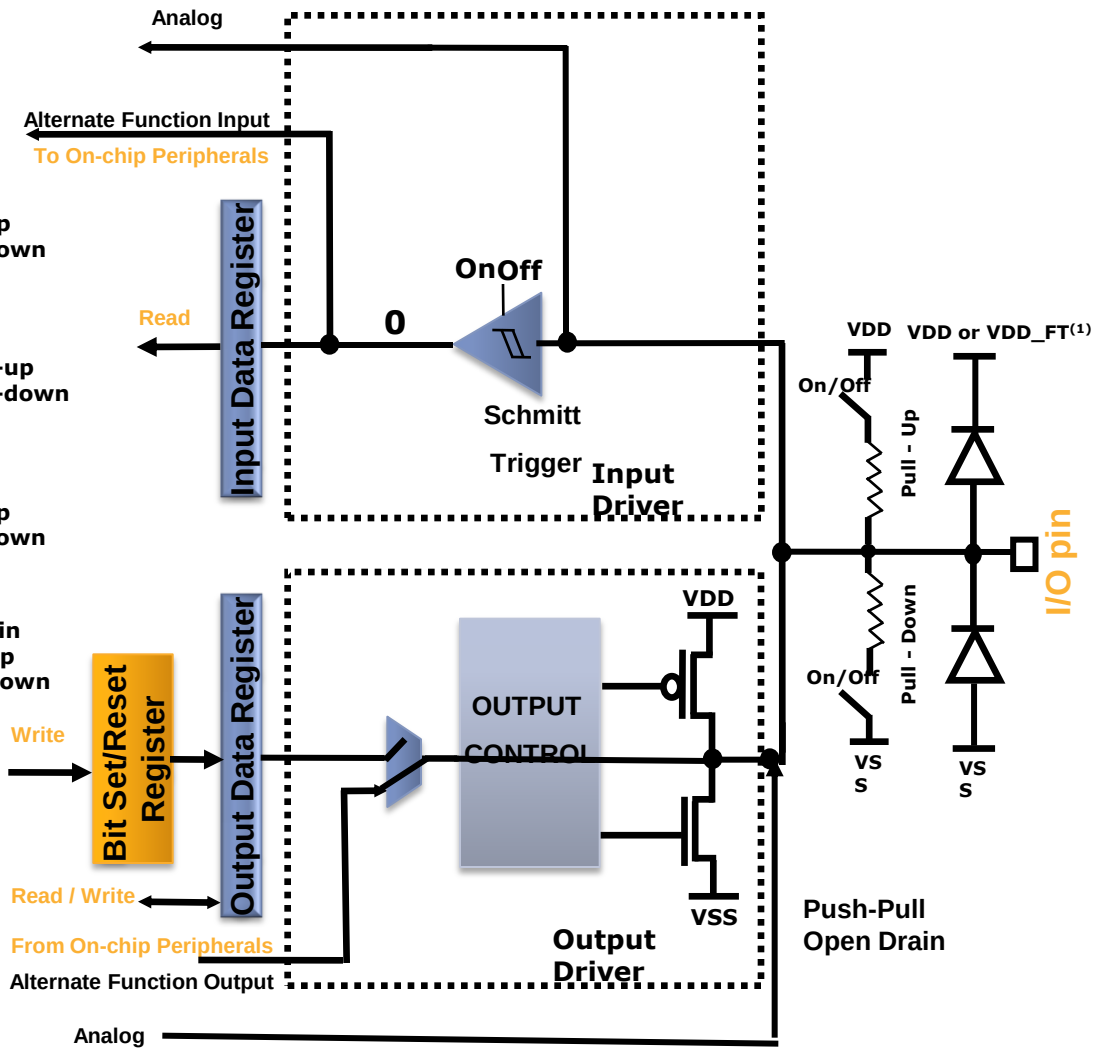
Standard Peripherals

GPIO features

- Up to 140 multifunction bi-directional I/O ports available on 176 pin package
- Almost standard I/Os are 5V tolerant
- All Standard I/Os are shared in 9 ports (GPIOA..GPIOI)
- Atomic Bit Set and Bit Reset using BSRR register
- GPIO connected to AHB bus: max toggling frequency = $f_{AHB}/2$ = 84 MHz
- Configurable Output Speed up to 100 MHz (2MHz,25MHz,50MHz,100MHz)
- Locking mechanism (GPIOx_LCKR) provided to freeze the I/O configuration
- Up to 140 GPIOs can be set-up as external interrupt (up to 16 lines at time) able to wake-up the MCU from low power modes
- Most of the I/O pins are shared with Alternate Functions pins connected to onboard peripherals through a multiplexer that allows only one peripheral's alternate function to be connected to an I/O pin at a time

GPIO Configuration Modes

MODER(i) [1:0]	OTYPER(i) [1:0]	PUPDR(i) [1:0]	I/O configuration
01	0	0 0 0 1 1 0	Output Push Pull Output Push Pull with Pull-up Output Push Pull with Pull-down
	1	0 0 0 1 1 0	Output Open Drain Output Open Drain with Pull-up Output Open Drain with Pull-down
10	0	0 0 0 1 1 0	Alternate Function Push Pull Alternate Function PP Pull-up Alternate Function PP Pull-down
	1	0 0 0 1 1 0	Alternate Function Open Drain Alternate Function OD Pull-up Alternate Function OD Pull-down
00	x	0 0 0 1 1 0	Input floating Input with Pull-up Input with Pull-down
11	x	x	Analog mode

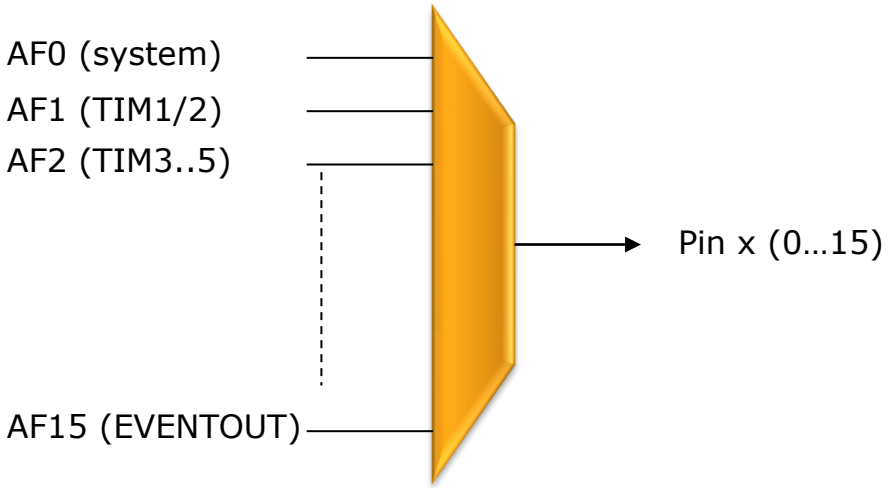


* In output mode, the I/O speed is configurable through OSPEEDR register: 2MHz, 25MHz, 50MHz or 100 MHz

(1) VDD_FT is a potential specific to five-volt tolerant I/Os and different from VDD.

Alternate Functions features

- Most of the peripherals shares the same pin (like USARTx_Tx, TIMx_CH2, I2Cx_SCL, SPIx_MISO, EVENTOUT…)
- Alternate functions multiplexers prevent to have several peripheral's function pin to be connected to a specific I/O at a time.
- Some Alternate function pins are remapped to give the possibility to optimize the number of peripherals used in parallel.

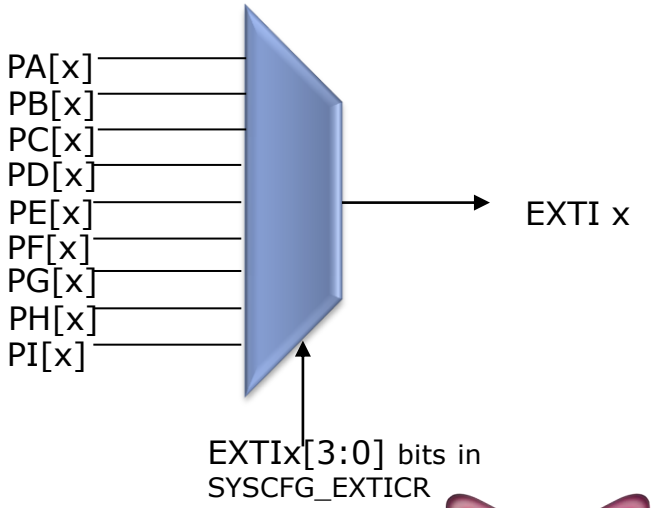


System Configuration

- Configure (2 bits) the type of memory accessible at address 0x00000000. These bits are used to select the physical remap by software and so, bypass the BOOT pins.
  - 00: Main Flash memory mapped at 0x0000 0000
  - 01: System Flash memory mapped at 0x0000 0000
  - 10: FSMC (NOR/SRAM bank1) mapped at 0x0000 0000
  - 11: Embedded SRAM (112kB) mapped at 0x0000 0000

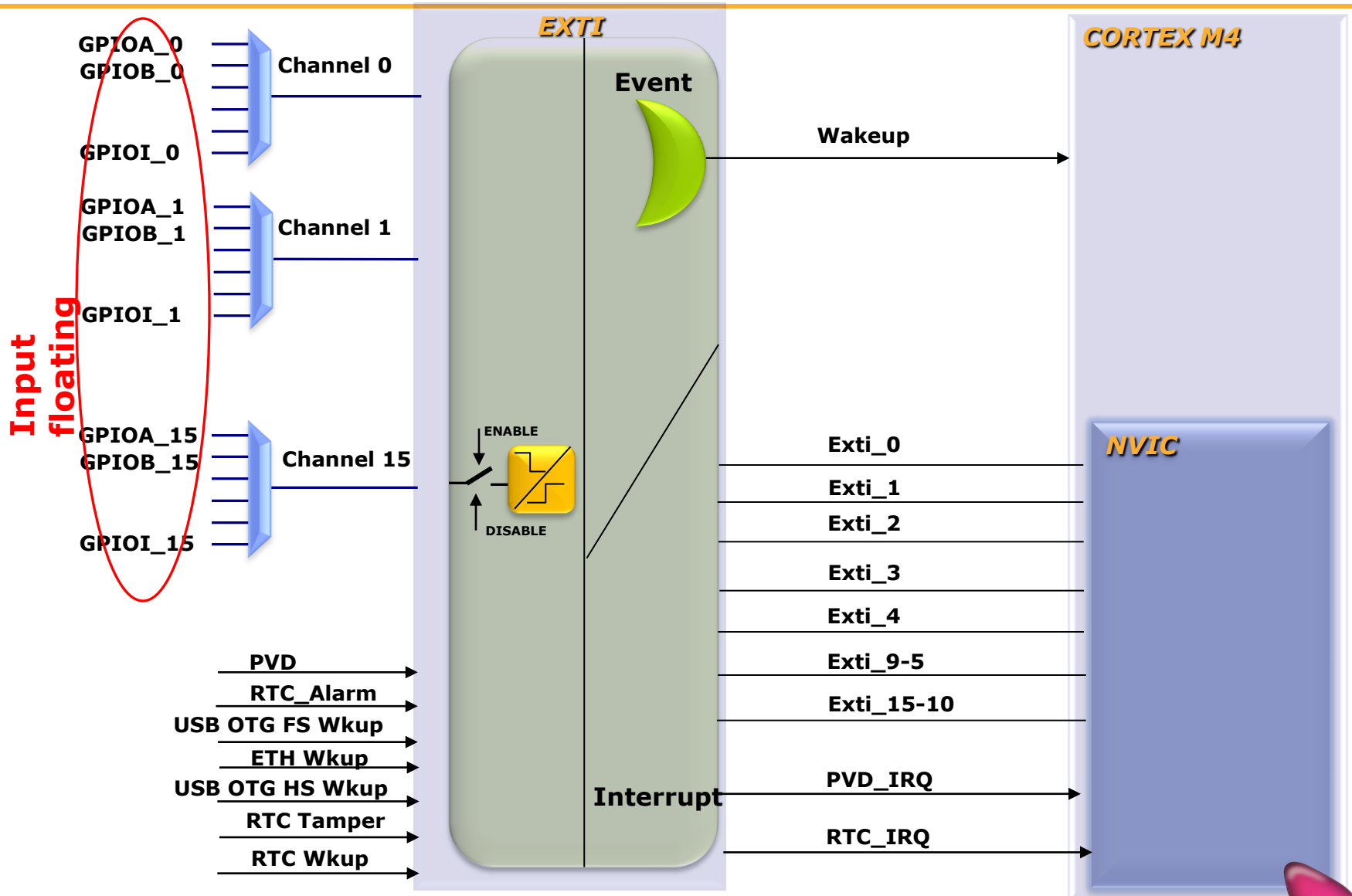
- Select the Ethernet PHY interface (MII or RMII)

- Manage the external interrupt line connection to the GPIOs:



* x can be 0 to 15 for all ports]

EXTI module: from pin to NVIC



ADC Features (1/2)

- 3 ADCs : ADC1 (master), ADC2 and ADC3 (slaves)
- Maximum frequency of the ADC analog clock is 36MHz.
- 12-bits, 10-bits, 8-bits or 6-bits configurable resolution.
- ADC conversion rate with 12 bit resolution is up to:
  - 2.4 M.sample/s in single ADC mode,
  - 4.5 M.sample/s in dual interleaved ADC mode,
  - 7.2 M.sample/s in triple interleaved ADC mode.
- Conversion range: 0 to 3.6 V.
- ADC supply requirement: VDDA = 2.4V to 3.6V at full speed and down to 1.65V at lower speed.
- Up to 24 external channels.
- 3 ADC1 internal channels connected to:
  - Temperature sensor,
  - Internal voltage reference : V_{REFINT} (1.2V typ),
  - VBAT for internal battery monitoring.

ADC Features (2/2)

- External trigger option for both regular and injected conversion.
- Single and continuous conversion modes.
- Scan mode for automatic conversion of channel 0 to channel ‘n’.
- Left or right data alignment with in-built data coherency.
- Channel by channel programmable sampling time.
- Discontinuous mode.
- Dual/Triple mode (with ADC1 and ADC2 or all 3 ADCs).
- DMA capability
- Analog Watchdog on high and low thresholds.
- Interrupt generation on:
  - End of Conversion
  - End of Injected conversion
  - Analog watchdog
  - Overrun

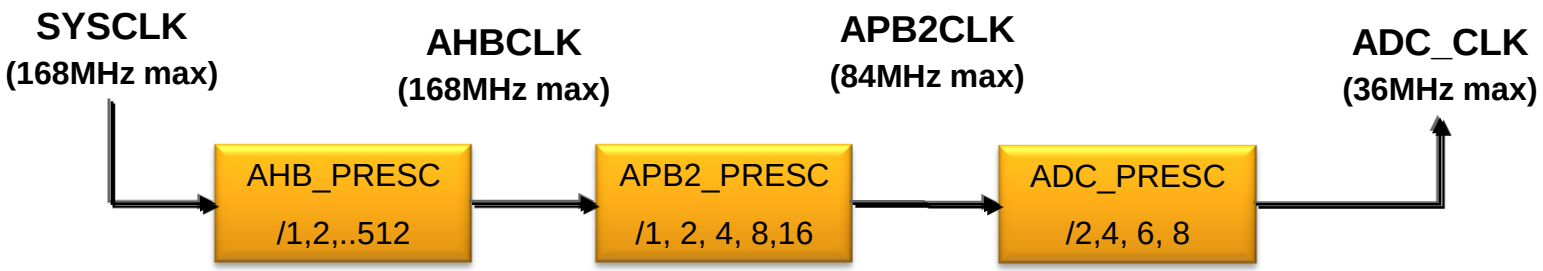
ADC speed performances

AHBCLK	APB2CLK	ADC_CLK	ADC speed (15 cycles)
168MHz	(a) 84MHz	(2) 21MHz	0.714µs **1.4 Msample/s**
144MHz	(a) 72MHz	(1) 36MHz	0.416µs **2.4 Msample/s**
120MHz	(a) 60MHz	(1) 30MHz	0.5µs **2 Msample/s**
96MHz	(a) 48MHz	(1) 24MHz	0.625µs **1.6 Msample/s**
72MHz	(b) 72MHz	(1) 36MHz	0.416µs **2.4 Msample/s**

(1). ADC_PRESC = /2
(2). ADC_PRESC = /4

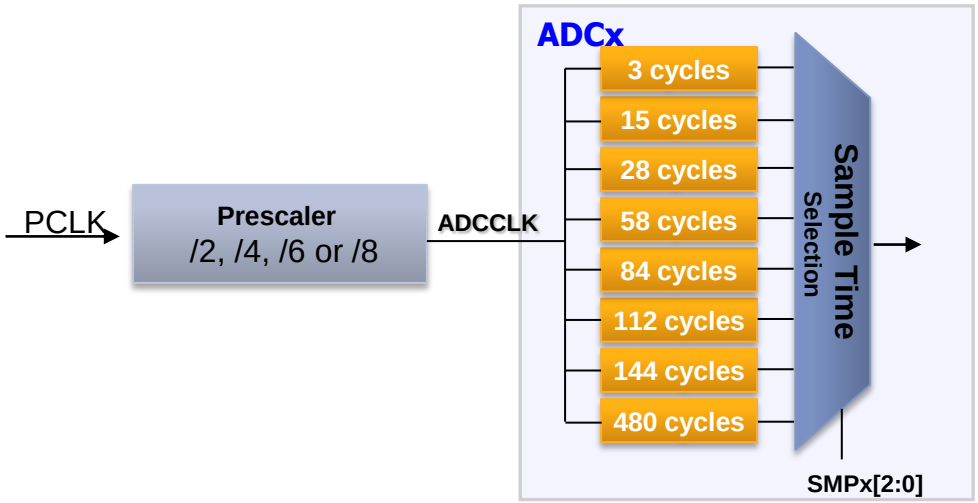
(a) APB_PRESC = /2

(b) APB_PRESC = /1



ADC ConversionTime

- ADCCLK, up to 36MHz, taken from PCLK through a prescaler (Div2, Div4, Div6 and Div8).
- Programmable sample time for each channel (from 4 to 480 clock cycles)
- Total conversion Time = $T_{Sampling} + T_{conversion}$



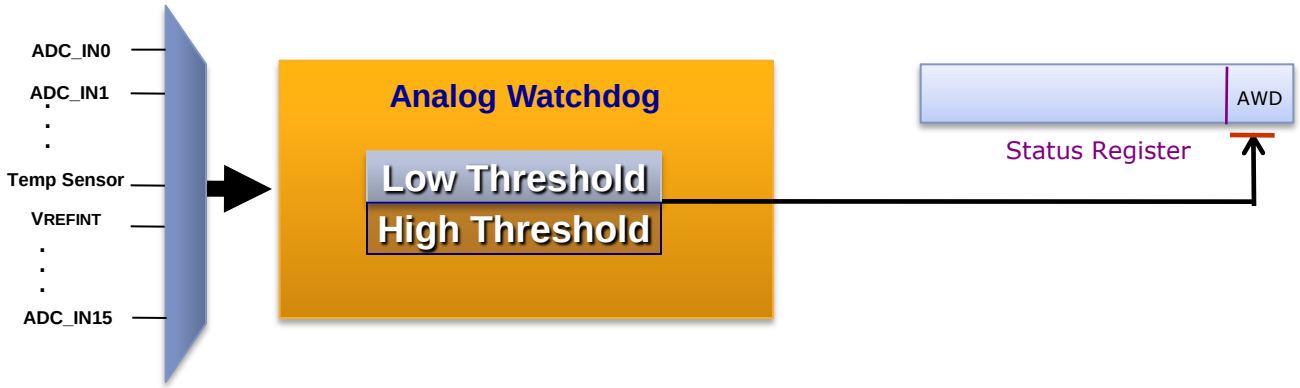
Resolution	$T_{Conversion}$
12 bits	12 Cycles
10 bits	10 Cycles
8 bits	8 Cycles
6 bits	6 Cycles

- With Sample time= 3 cycles @ ADC_CLK = 36MHz ➔ total conversion time is equal to:

resolution	Total conversion Time	
12 bits	**12 + 3 = 15cycles**	**0.416 us → 2.4 Msps**
10 bits	10 + 3 = 13 cycles	0.361 us → 2.71 Msps
8 bits	8 + 3 = 11 cycles	0.305 us → 3.27 Msps
6 bits	6 + 3 = 9 cycles	0.25 us → 4 Msps

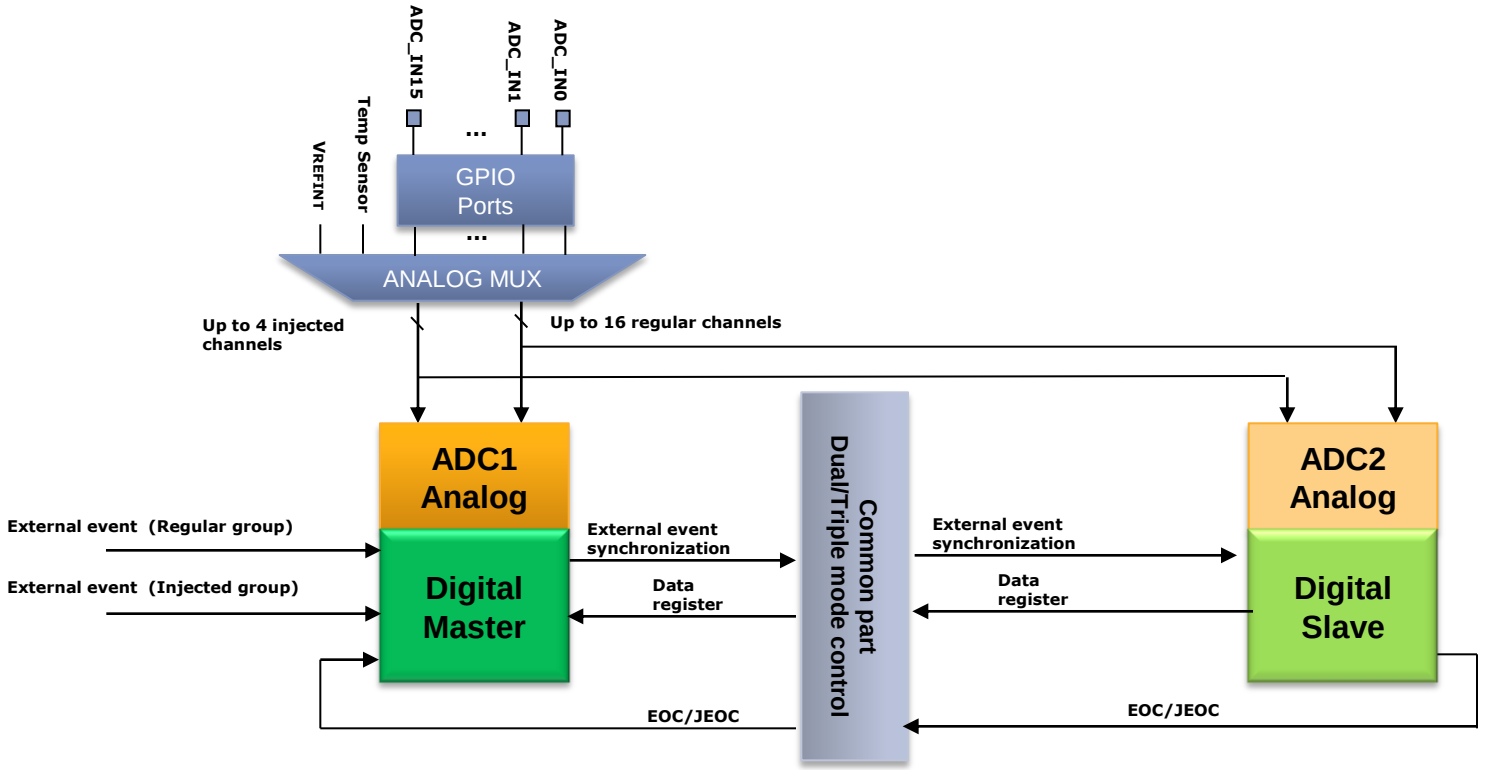
ADC Analog Watchdog

- 12-bit programmable analog watchdog low and high thresholds
- Enabled on one or all converted channels: one regular or/and injected channel, all injected or/and regular channels.
- Interrupt generation on low or high thresholds detection



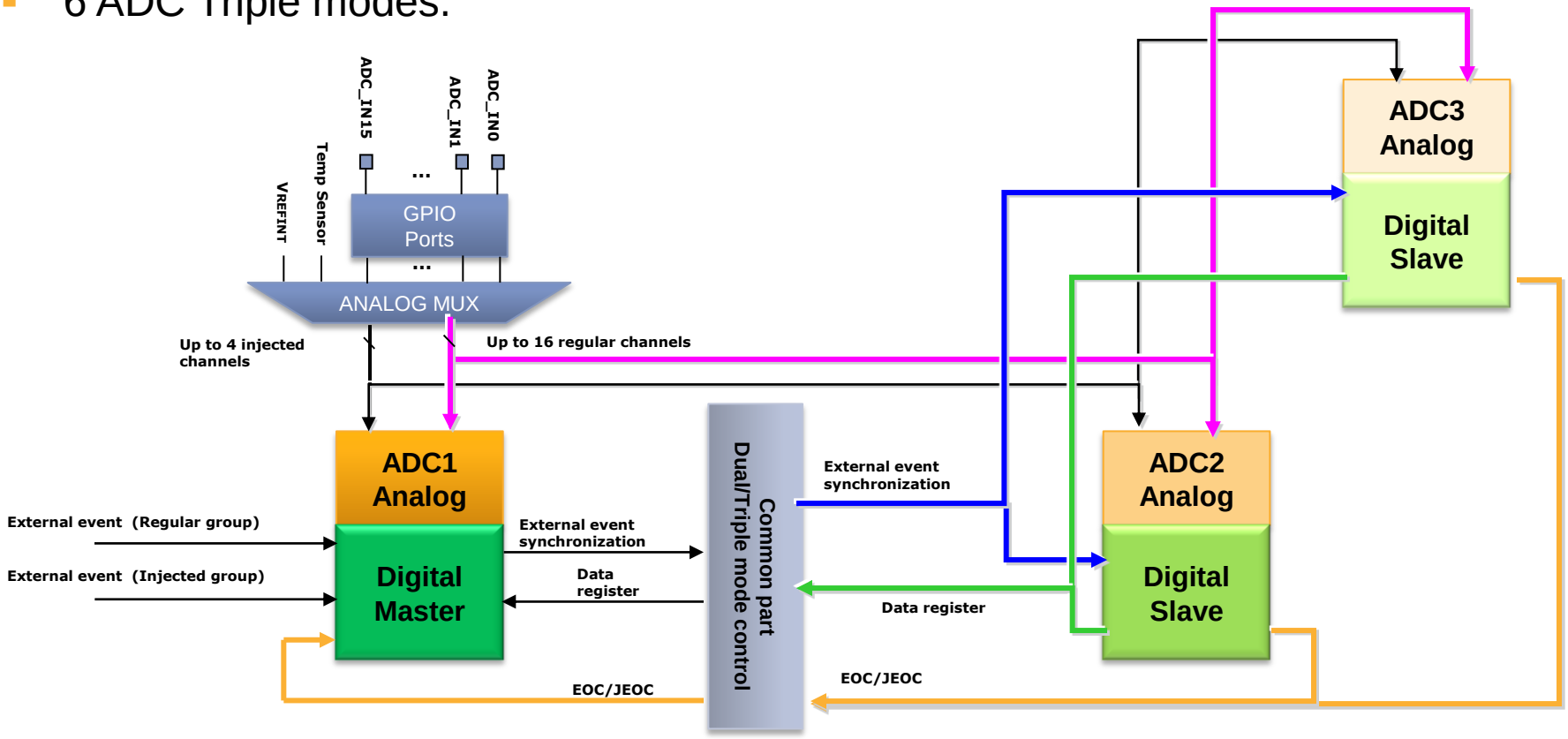
ADC dual modes

- ADCs: ADC1 master and ADC2 slave, ADC3 is independently.
- The start of conversion is triggered alternately or simultaneously by the ADC1 master to the ADC2 slave depending on the mode selected.
- 6 ADC dual modes



ADC Triple modes

- ADCs: ADC1 master, ADC2 and ADC3 slaves.
- The start of conversion is triggered alternately or simultaneously by the ADC1 master to the ADC2 and ADC3 slaves depending on the mode selected.
- 6 ADC Triple modes.



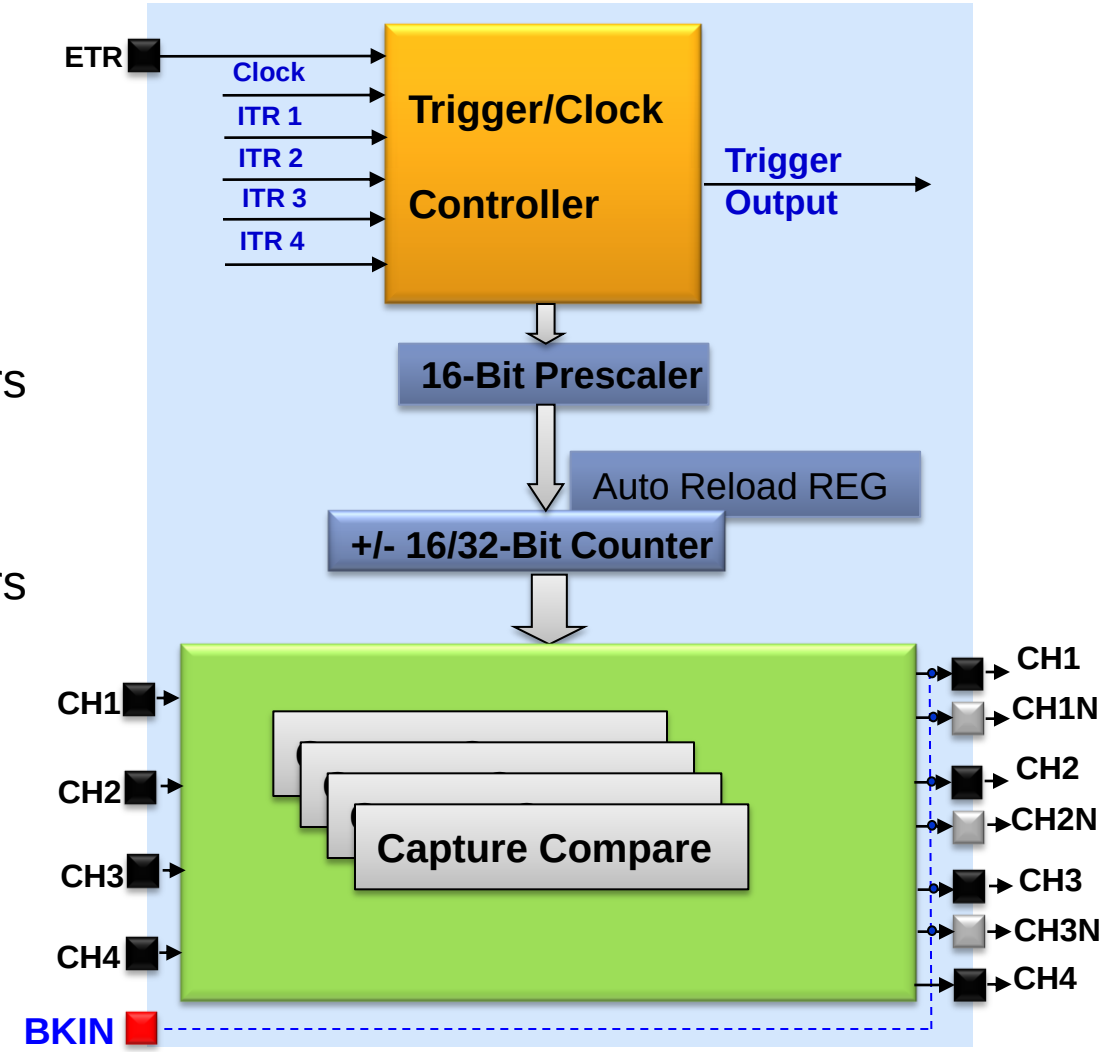
DAC Features

- Two DAC converters: one output channel for each one
- 8-bit or 12-bit monotonic output
- Left or right data alignment in 12-bit mode
- Synchronized update capability
- Noise-wave or Triangular-wave generation
- Dual DAC channel independent or simultaneous conversions
- 11 dual channel modes
- DMA capability for each channel
- External triggers for conversion
- DAC supply requirement: 1.8V to 3.6 V
- Conversion range: 0 to 3.6 V
- DAC outputs range: 0 ≤ DAC_OUTx ≤ VREF+ (VREF+ is available only in 100, 144 and 176 pins package)

Timers on STM32F4

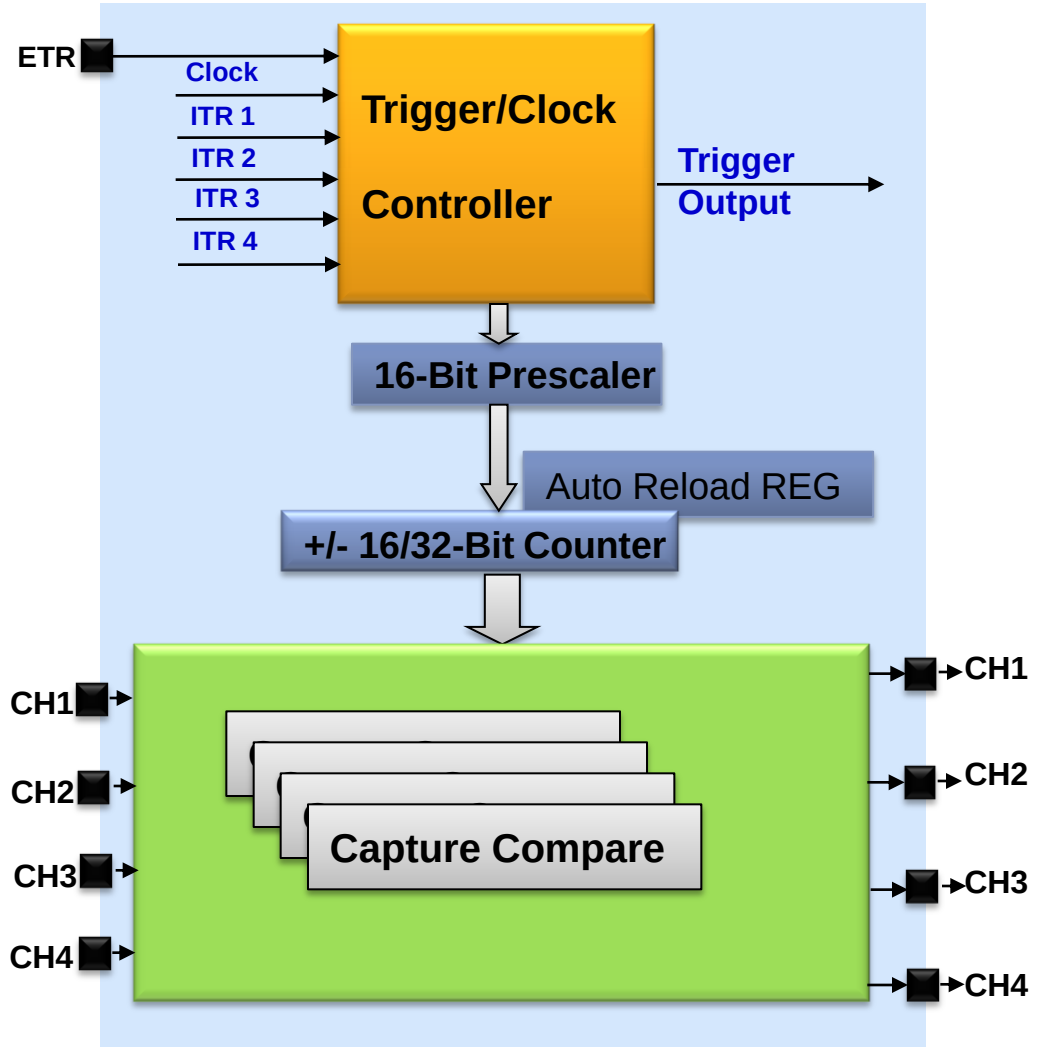
On board there are following timers available:

- **2x advanced 16bit** timers (TIM1,8)
- **2x general purpose 32bit** timers (TIM2,5)
- **8x general purpose 16bit** timers (TIM3,4,9,10..14)
- **2x simple 16bit** timers for DAC (TIM6,7)
- **1x 24bit system timer** (SysTick)



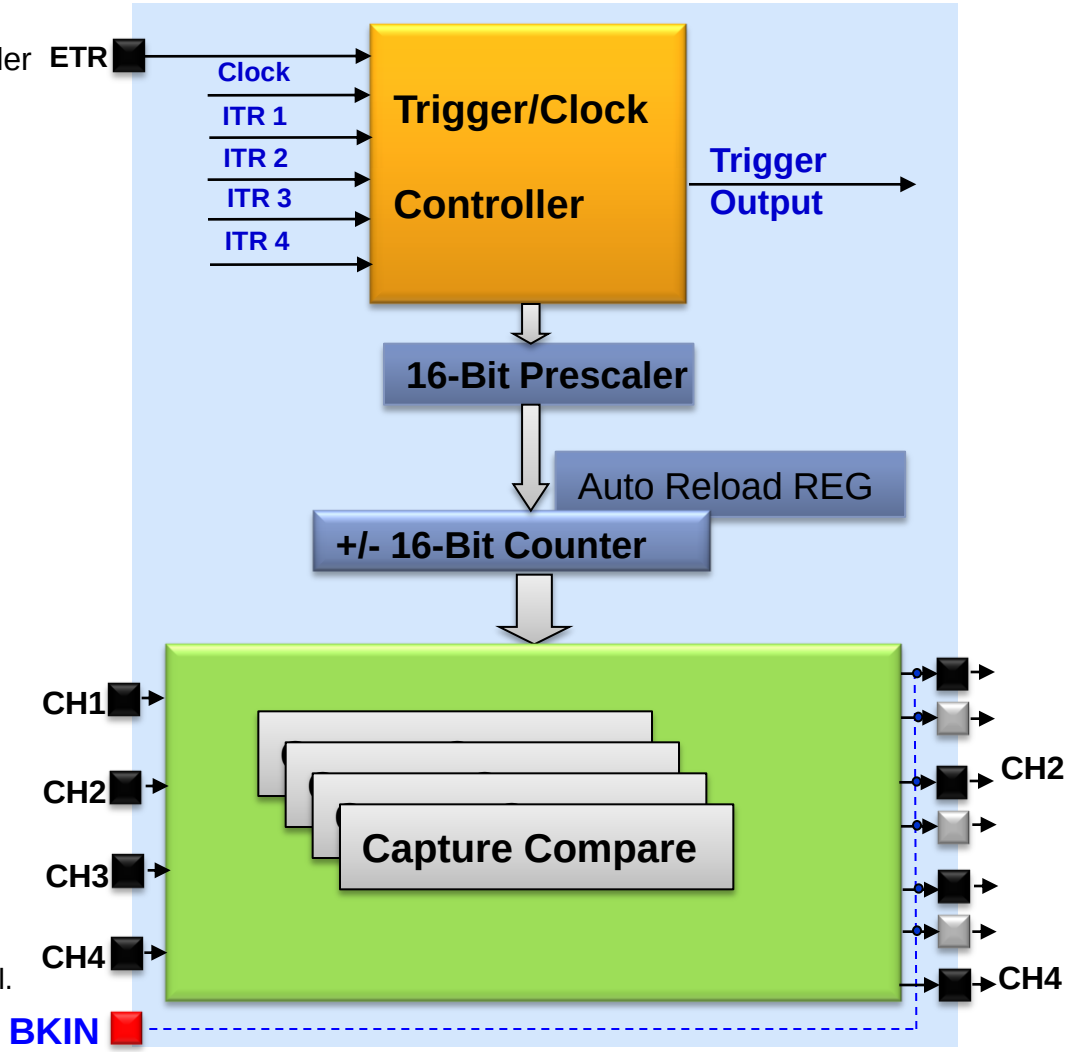
General Purpose timer Features overview

- TIM2, 3, 4 and 5 on Low Speed APB (APB1)
- Internal clock up to **84 MHz** (if AHB/APB1 prescaler distinct from 1)
- 16-bit Counter for TIM3 and 4
- 32-bit Counter for TIM2 and 5
  - Up, down and centered counting modes
  - Auto Reload
- 4 x 16 High resolution Capture Compare Channels
  - Programmable direction of the channel: input/output
  - Output Compare
  - PWM
  - Input Capture, PWM Input Capture
  - One Pulse Mode
- Synchronization
  - Timer Master/Slave
  - Synchronisation with external trigger
  - Triggered or gated mode
- Encoder interface
- 6 Independent IRQ/DMA Requests generation
  - At each Update Event
  - At each Capture Compare Events
  - At each Input Trigger



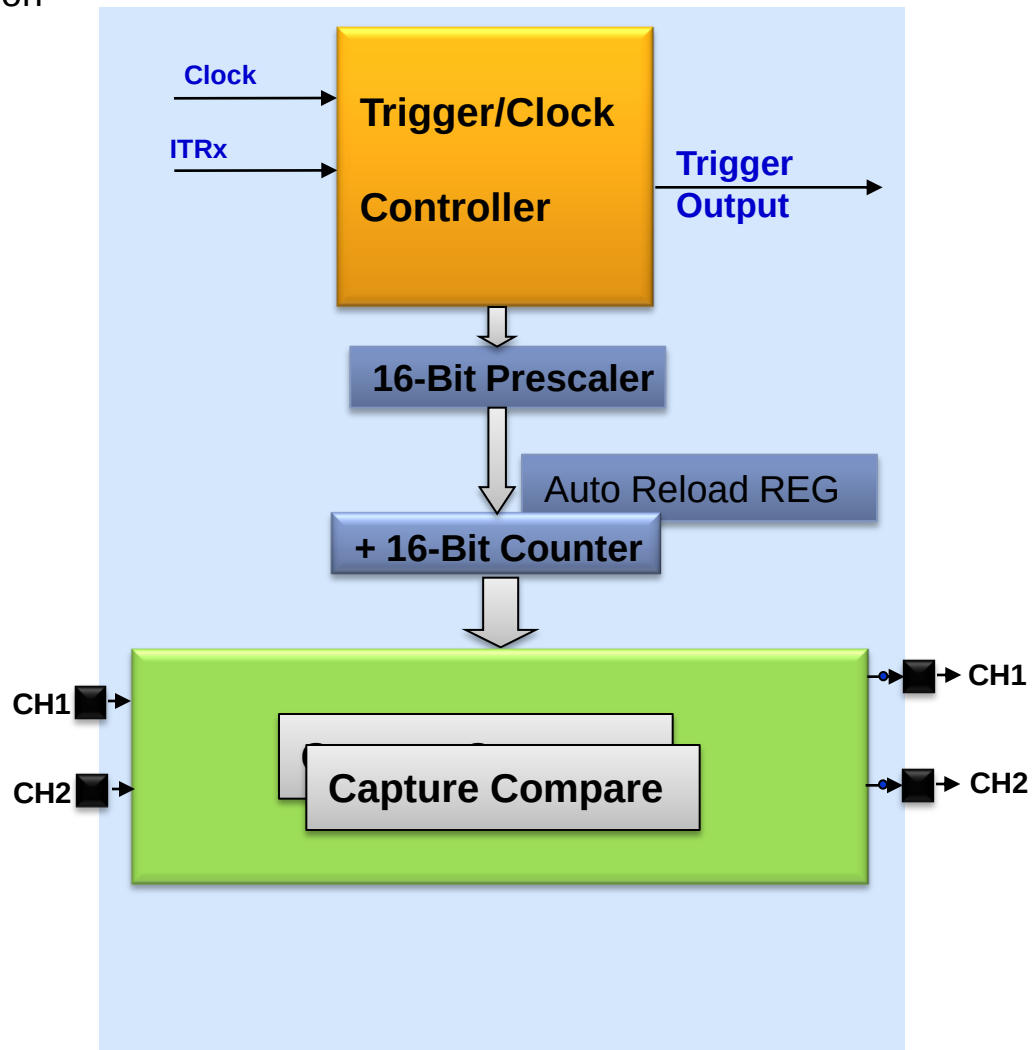
Advanced timer Features overview

- TIM1 and TIM8 on High Speed APB (APB2)
- Internal clock up to **168 MHz** (if AHB/APB2 prescaler distinct from 1)
- 16-bit Counter
  - Up, down and centered counting modes
  - Auto Reload
- 4 x 16 High resolution Capture Channels
  - Output Compare
  - PWM
  - Input Capture, PWM input Capture
  - One Pulse Mode
- 6 Complementary outputs: Channel1, 2 and 3
- Output Idle state selection independently for each output
- Polarity selection independently for each output
- Programmable PWM repetition counter
- Hall sensor interface
- Encoder interface
- 8 Independent IRQ/DMA Requests Generation
  - At each Update Event
  - At each Capture Compare Events
  - At each Trigger Input Event
  - At each Break Event
  - At each Capture Compare Update
- Embedded Safety features
  - Break input
  - Lockable unit configuration: 3 possible Lock level.



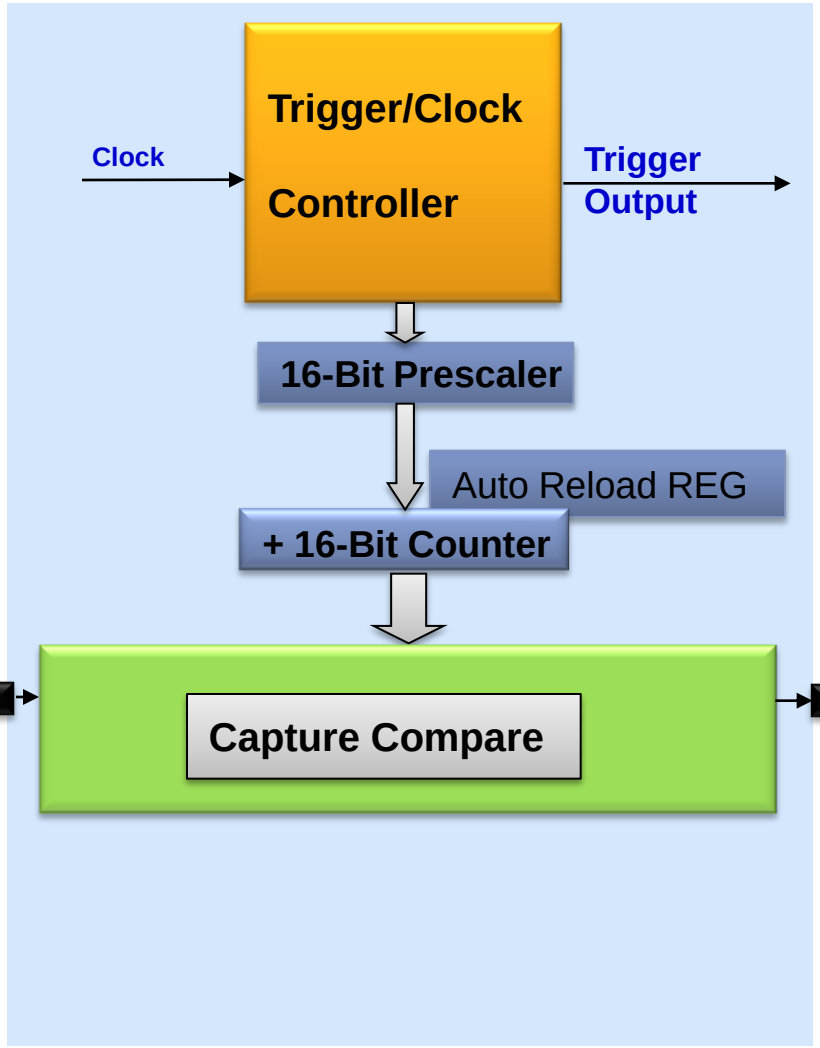
General Purpose 2 Channels timer (TIM9 & TIM12) Features overview

- TIM9 on High speed APB (APB2) and TIM12 on Low Speed APB (APB1)
- Internal clock up to **168 MHz** and **84 MHz** respectively
- 16-bit Counter
  - Up counting mode
  - Auto Reload
- 2 x 16 High resolution Capture Compare Channels
  - Programmable direction of the channel: input/output
  - Output Compare
  - PWM
  - Input Capture, PWM Input Capture
  - One Pulse Mode
- Synchronization Timer Master/Slave
  - Synchronization with external trigger
  - Triggered or gated mode
- Independent IRQ Requests generation
  - At each Update Event
  - At each Capture Compare Events
  - At each Input Trigger



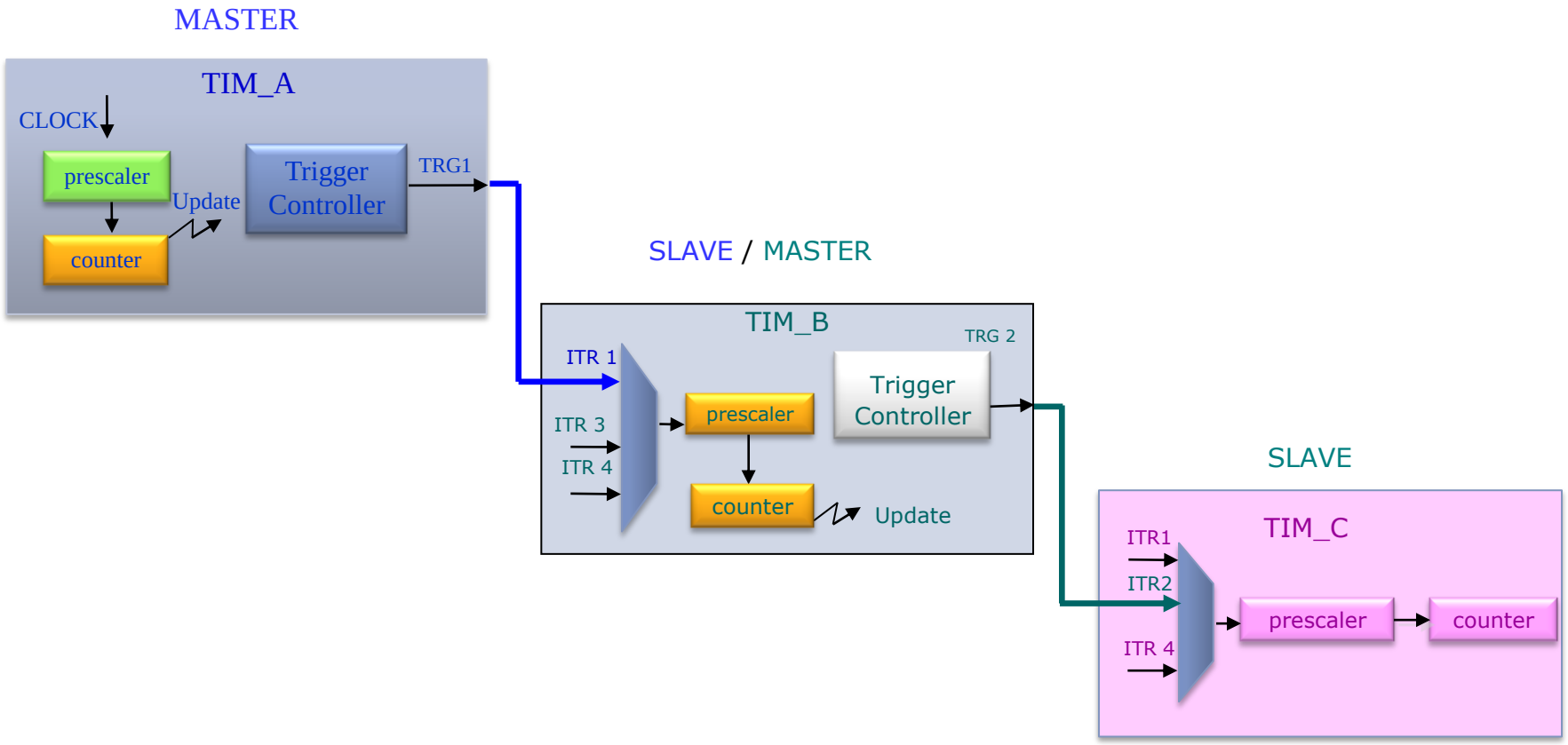
General Purpose 1 Channels timer (TIM10..11 & TIM13..14) Features overview

- TIM10..11 on High speed APB (APB2) and TIM13..14 on Low Speed APB (APB1)
- Internal clock up to 168 MHz for TIM10/11
- Internal clock up to 84 MHz for TIM13/14
- 16-bit Counter
  - Up counting mode
  - Auto Reload
- 2 x 16 High resolution Capture Compare Channels
  - Programmable direction of the channel: input/output
  - Output Compare
  - PWM
  - Input Capture
- Independent IRQ Requests generation
  - At each Update Event
  - At each Capture Compare Events



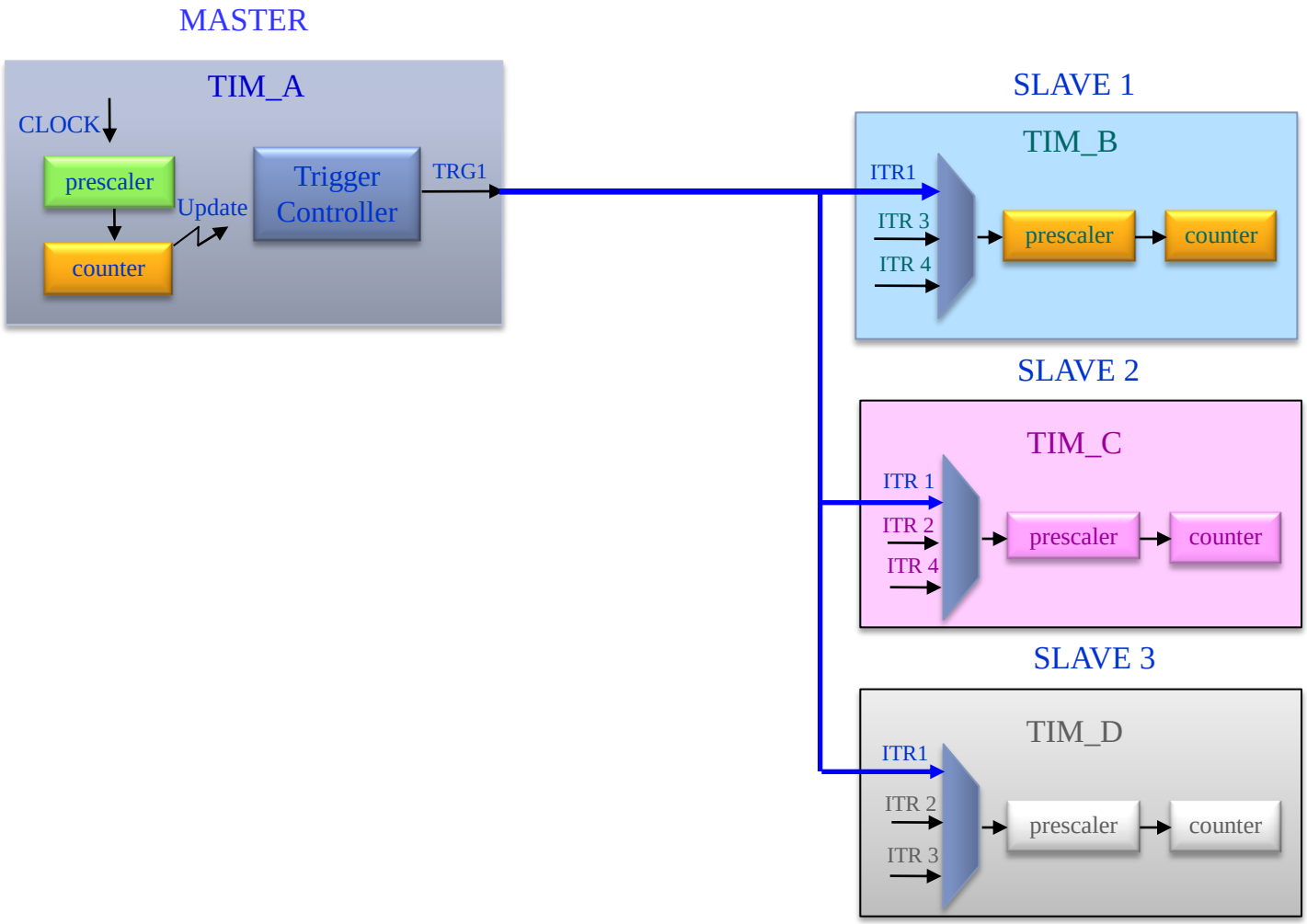
Synchronization – Configuration examples (1/3)

- Cascade mode:
  - TIM_A used as master timer for TIM_B, TIM_B configured as TIM_A slave and master for TIM_C.



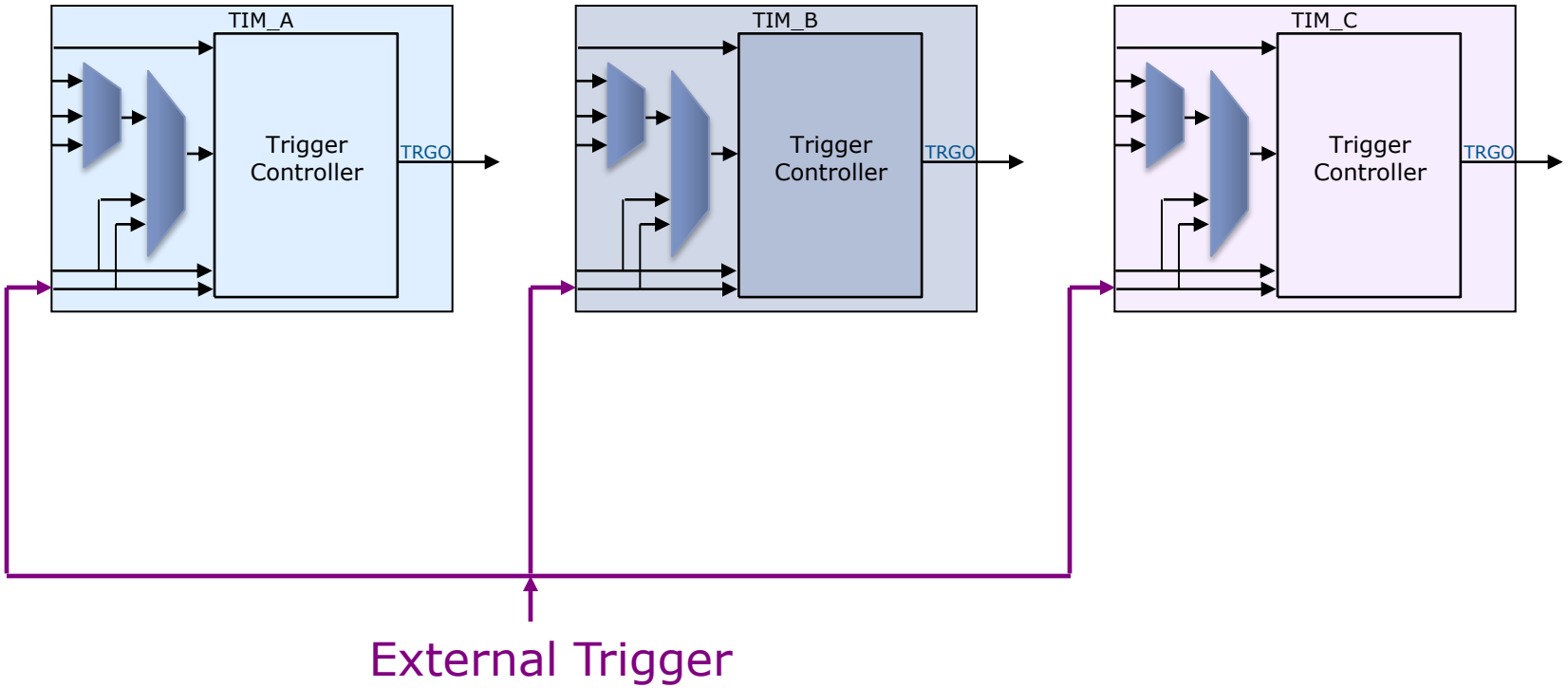
Synchronization – Configuration examples (2/3)

- One Master several slaves: TIM_A used as master for TIM_B, TIM_C and TIM_D.



Synchronization – Configuration examples (3/3)

- Timers and external trigger synchronization
  - TIM_A, TIM_B and TIM_C are slaves for an external signal connected to respective Timers inputs.



STM32F4xx Timer features overview (1/2)

	Counter resolution	Counter type	Prescaler factor	DMA	Capture Compare Channels	Complementary output	Synchronization	
							Master Config	Slave Config
Advanced TIM1 and TIM8	16 bit	up, down and up/down	1..65536	YES	4	3	YES	YES
General purpose (1) TIM2 and TIM5	32 bit	up, down and up/down	1..65536	YES	4	0	YES	YES
General purpose TIM3 and TIM4	16 bit	up, down and up/down	1..65536	YES	4	0	YES	YES
Basics TIM6 and TIM7	16 bit	up	1..65536	YES	0	0	YES	NO
1 Channel (2) TIM10..11 and TIM13..14 (2)	16 bit	up	1..65536	NO	1	0	YES (OC signal)	NO
2 Channel(2) TIM9 and TIM12	16 bit	up	1..65536	NO	2	0	NO	YES

(1) Same as STM32F2xx 32-Bit Timers

(2) These Timers are identical to STM32F2xx and STM32F1 XL Timers

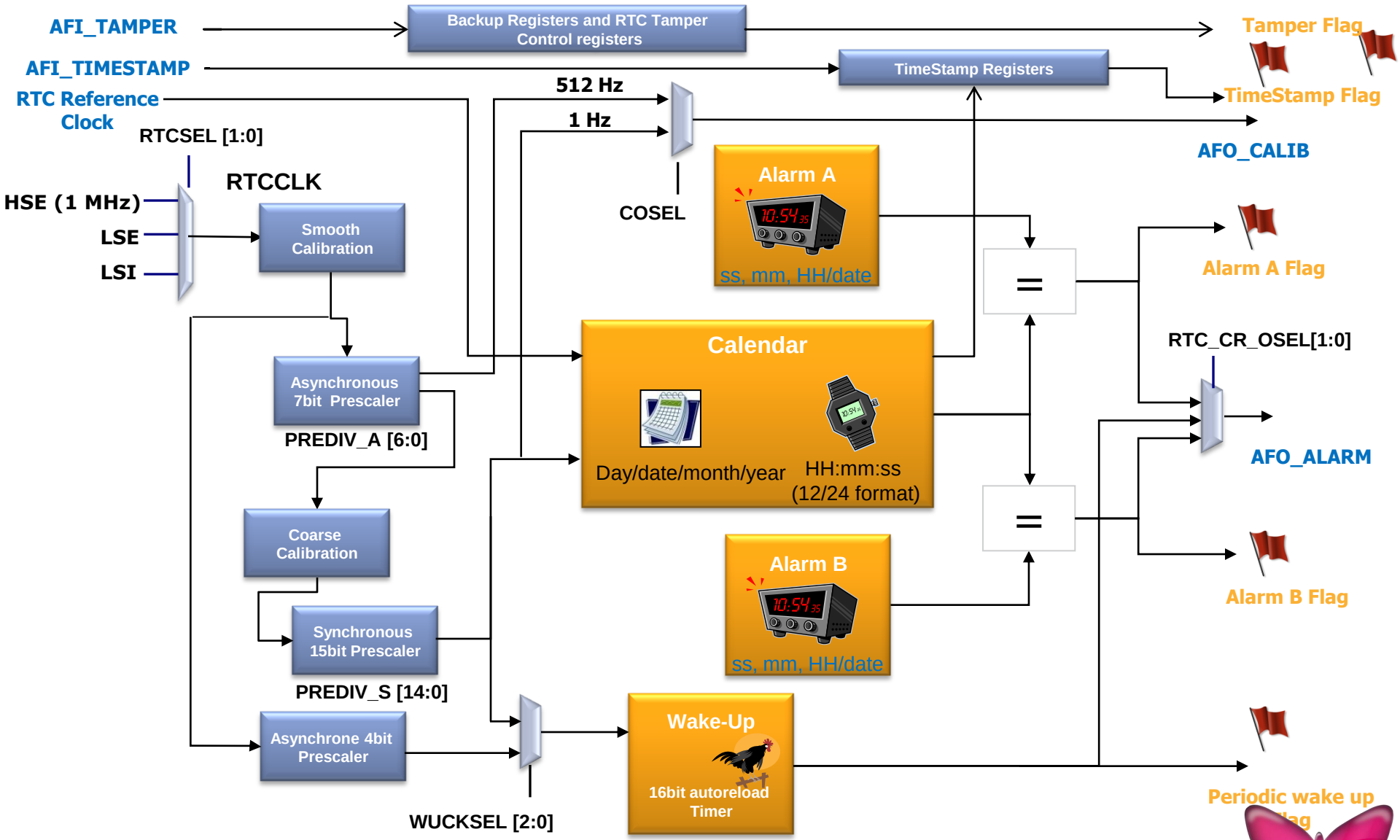
STM32F4xx Timer features overview 2/2

	Counter clock source	Output Compare	PWM	Input Capture	PWMI	OPM	Encoder interface	Hall sensor interface	XOR Input
Advanced TIM1 and TIM8	-Internal clock APB2 -External clock: ETR/TI1/TI2/TI3/TI4 pins -Internal Trigger: ITR1/ITR2/ITR3/ITR4 -Slave mode	7	7	4	2	2	Yes	Yes	Yes
General Purpose TIM2 and TIM5	-Internal clock APB1 -External clock: ETR/TI1/TI2/TI3/TI4 pins -Internal Trigger: ITR1/ITR2/ITR3/ITR4 -Slave mode	4	4	4	2	2	Yes	No	Yes
General Purpose TIM3 and TIM4	-Internal clock APB1 -External clock: ETR/TI1/TI2/TI3/TI4 pins -Internal Trigger: ITR1/ITR2/ITR3/ITR4 -Slave mode	4	4	4	2	2	Yes	No	Yes
Basics TIM6 and TIM7	-Internal clock APB1	No	No	No	No	No	No	No	No
1 Channel TIM10/11 and 13/14	-Internal clock APB1/APB2	1	1	1	No	No	No	No	No
2 Channel TIM9 and TIM12	-Internal clock APB1/APB2 -External clock: TI1/TI2/TI3/TI4 pins -Internal Trigger: ITR1/ITR2/ITR3/ITR4 -Slave mode	2	2	2	2	2	No	No	No

RTC Features

- Ultra-low power battery supply current < 1uA with RTC ON.
- Calendar with sub seconds, seconds, minutes, hours, week day, date, month, and year.
- Daylight saving compensation programmable by software
- Two programmable alarms with interrupt function. The alarms can be triggered by any combination of the calendar fields.
- A periodic flag triggering an automatic wakeup interrupt. This flag is issued by a 16-bit auto-reload timer with programmable resolution. This timer is also called ‘wakeup timer’.
- A second clock source (50 or 60Hz) can be used to update the calendar.
- Maskable interrupts/events:
  - Alarm A, Alarm B, Wakeup interrupt, Time-stamp, Tamper detection
- Digital calibration circuit (periodic counter correction) to achieve 5 ppm accuracy
- Time-stamp function for event saving with sub second precision (1 event)
- 20 backup registers (80 bytes) which are reset when an tamper detection event occurs.

RTC Block Diagram



STM32F4

Communication peripherals

Communication Peripherals

INTER-INTEGRATED CIRCUIT INTERFACE (I^2C)

I²C Features (1/2)

- Multi Master and slave capability
- Controls all I²C bus specific sequencing, protocol, arbitration and timing
- Standard and fast I²C mode (up to 400kHz)
- 7-bit and 10-bit addressing modes
- Dual Addressing Capability to acknowledge 2 slave addresses
- Status flags:
  - Transmitter/Receiver mode flag
  - End-of-Byte transmission flag
  - I²C busy flag
- Configurable PEC (Packet Error Checking) Generation or Verification:
  - PEC value can be transmitted as last byte in Tx mode
  - PEC error checking for last received byte

I²C Features (2/2)

- Error flags:
  - Arbitration lost condition for master mode
  - Acknowledgement failure after address/ data transmission
  - Detection of misplaced start or stop condition
  - Overrun/Underrun if clock stretching is disabled
- 2 Interrupt vectors:
  - 1 Interrupt for successful address/ data communication
  - 1 Interrupt for error condition
- 1-byte buffer with DMA capability
- SMBus 2.0 Compatibility
- PMBus Compatibility

Communication Peripherals

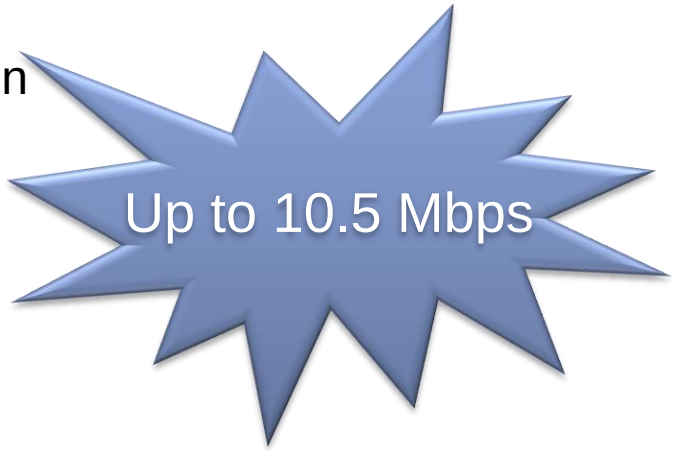
UNIVERSAL SYNCHRONOUS ASYNCHRONOUS RECEIVER TRANSMITTER (USART)

USART Features (1/2)

- **6 USARTs**: USART1 & USART6 on APB2 and USART2,3,4,5 on APB1
- Fully-programmable serial interface characteristics:
  - Data can be 8 or 9 bits
  - Even, odd or no-parity bit generation and detection
  - 0.5, 1, 1.5 or 2 stop bit generation
  - **Oversampling by 16 (default) or by 8**
  - Programmable baud rate generator
    - Integer part (12 bits)
    - Fractional part (4 bits)
    - Baud rate for standard USART (SPI mode included)

$$Tx/Rx\ baud = fck/8x(2-OVR8)xUSARTDIV$$

      - Where:
        - **Tx/Rx baud**: desired baudrate
        - **OVR8**: oversampling by 8 (1 if enabled, 0 if disabled)
        - **fck**: APB frequency
        - **USARTDIV**: value to be programmed to the BRR register



USART Features (2/2)

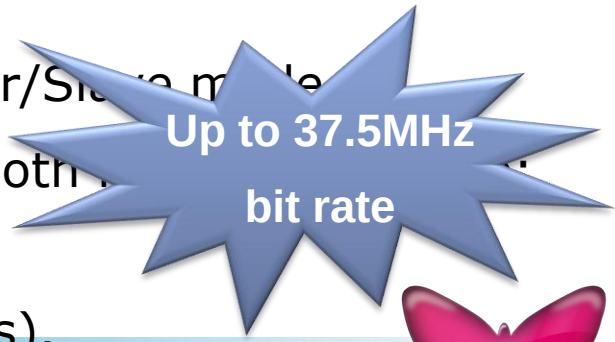
- Support hardware flow control (CTS and RTS)
- Dedicated transmission and reception flags (TxE and RxNE) with interrupt capability
- Support for DMA
  - Receive DMA request and Transmit DMA request
- 10 interrupt sources to ease software implementation
- LIN Master/Slave compatible
- Synchronous Mode: Master mode only
- IrDA SIR Encoder Decoder
- Smartcard Capability
- Single wire Half Duplex Communication
- Multi-Processor communication
  - USART can enter Mute mode
  - Mute mode: disable receive interrupts until next header detected
  - Wake up from mute mode (by idle line detection or address mark detection)
- Support One Sample Bit method: allows to disable noise detection (for noise-free applications) in order to increase the receiver's tolerance to clock deviations.

Communication Peripherals

SERIAL PERIPHERAL INTERFACE (SPI)

SPI Features (1/2)

- Up to 3 SPIs: SPI1 on high speed APB2 and SPI2,SPI2 on low speed APB1
- SPI2, SPI3 can work as SPI or I^2S interface
- Full duplex synchronous transfers on 3 lines
- Simplex synchronous transfers on 2 lines with or without a bi-directional data line
- Programmable data frame size :8- or 16-bit transfer frame format selection
- Programmable data order with MSB-first or LSB-first shifting
- Master or slave operation
- Programmable bit rate: up to 37.5 MHz in Master/Slave mode
- NSS management by hardware or software for both Master and Slave: Dynamic change of Master/Slave operations
- Motorola / TI mode (master and slave operations).



SPI Features (2/2)

- Programmable clock polarity and phase
- Dedicated transmission and reception flags (Tx buffer Empty and Rx buffer Not Empty) with interrupt capability
- SPI bus busy status flag
- Master mode fault and overrun flags with interrupt capability
- Hardware CRC feature for reliable communication (CRC8, CRC16)
- Support for DMA
- Each SPI has a DMA Tx and Rx requests
- Each of the SPIs requests is mapped on a different DMA Stream: possibility to use DMA for all SPIs transfer direction at the same time
- Calculated CRC value is automatically transmitted at the end of data transfer

I²S Features (1/2)

- Two I²Ss: Available on SPI2 and SPI3 peripherals.
- Two I²Ss extension added for Full-Duplex communication.
- Dedicated PLL for high quality audio clock generation.
- Simplex/or Full duplex communication (transmitter and receiver)
- Can operate in master or slave configuration.
- 8-bit programmable linear prescaler to support all standard audio sample frequencies up to 192KHz.
- Programmable data format (16-, 24- or 32-bit data formats)
- Programmable packet frame (16-bit and 32-bit packet frames).
- Underrun flag in slave transmit mode, Overrun flag in receive mode and new de-synchronization flag in slave transmit/receive mode.
- 16-bit register for transmission and reception.
- Support for DMA (16-bit wide).

I²S Features (2/2)

- I²S protocols supported:
  - I²S Phillips standard.
  - MSB Justified standard (Left Justified).
  - LSB Justified standard (Right Justified).
  - PCM standard (with short and long frame synchronization on 16-bit channel frame or 16-bit data frame extended to 32-bit channel frame)

- Master clock may be output to drive an external audio component. Ratio is fixed at 256xFs (where Fs is the audio sampling frequency).

- Note: Since some SPI3/I²S3 pins are shared with JTAG pins, they are not controlled by the I/O controller and are reserved for JTAG usage (after each Reset). Prior to configure these pins, the user has to disable the JTAG and use the SWD interface (when debugging the application), or disable both JTAG/SWD interfaces (for standalone application).

Communication Peripherals

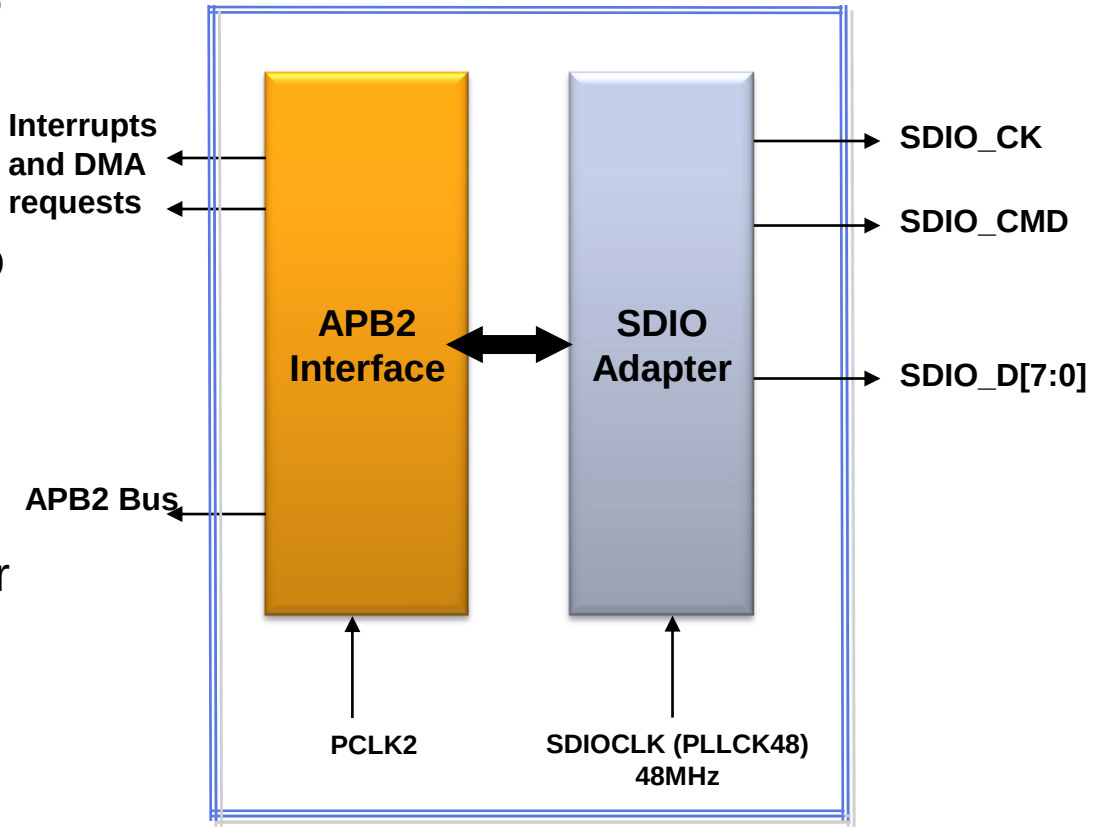
SD/SDIO MMC CARD HOST INTERFACE (SDIO)

SDIO Features

- Cards Clock Management: Rising and Falling edge, 8-bit prescaler, bypass, power save..
- Hardware Flow Control: to avoid FIFO underrun (TX mode) and overrun (RX mode) errors.
- A 32-bit wide, 32-word FIFO for Transmit and Receive
- DMA Transfer Capability
- Data Transfer: Configurable mode (Block or Stream), configurable data block size from1 to 16384 bytes, configurable TimeOut
- 24 interrupt sources to ease software implementation
- CRC Check and generation
- SD I/O mode: SD I/O Interrupt, suspend/resume and Read Wait
- Data transfer up to 48 MHz

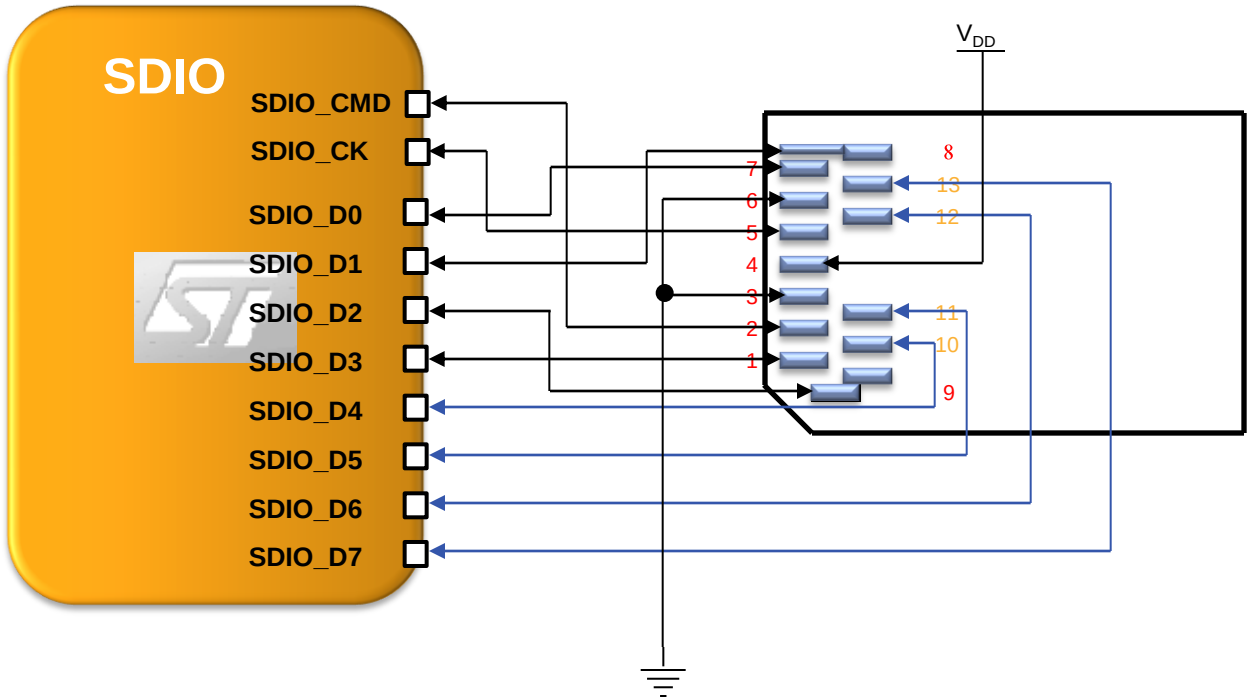
SDIO Block Diagram

- The SDIO consists of two parts:
  - The SDIO adapter block provides all functions specific to the MMC/SD/SD I/O card such as the clock generation unit, command and data transfer.
  - The APB2 interface accesses the SDIO adapter registers, and generates interrupt and DMA request signals.



SD/SDIO & MMC Cards

- The SDIO has 10 pins to control different kinds of memory cards
  - Only **6 pins** (SDIO_CMD, SDIO_CK, SDIO_D[3:0]) at **most for SD cards** (SD full size, miniSD, microSD)
  - Only **6 pins** (SDIO_CMD, SDIO_CK, SDIO_D[3:0]) at **most for SDIO cards** (SD full size, miniSD, microSD)
  - **10 pins** (SDIO_CMD, SDIO_CK, SDIO_D[7:0]) at **most for MMC cards** (MMC full size, RS-MMC, MMC+ and MMCMobile)



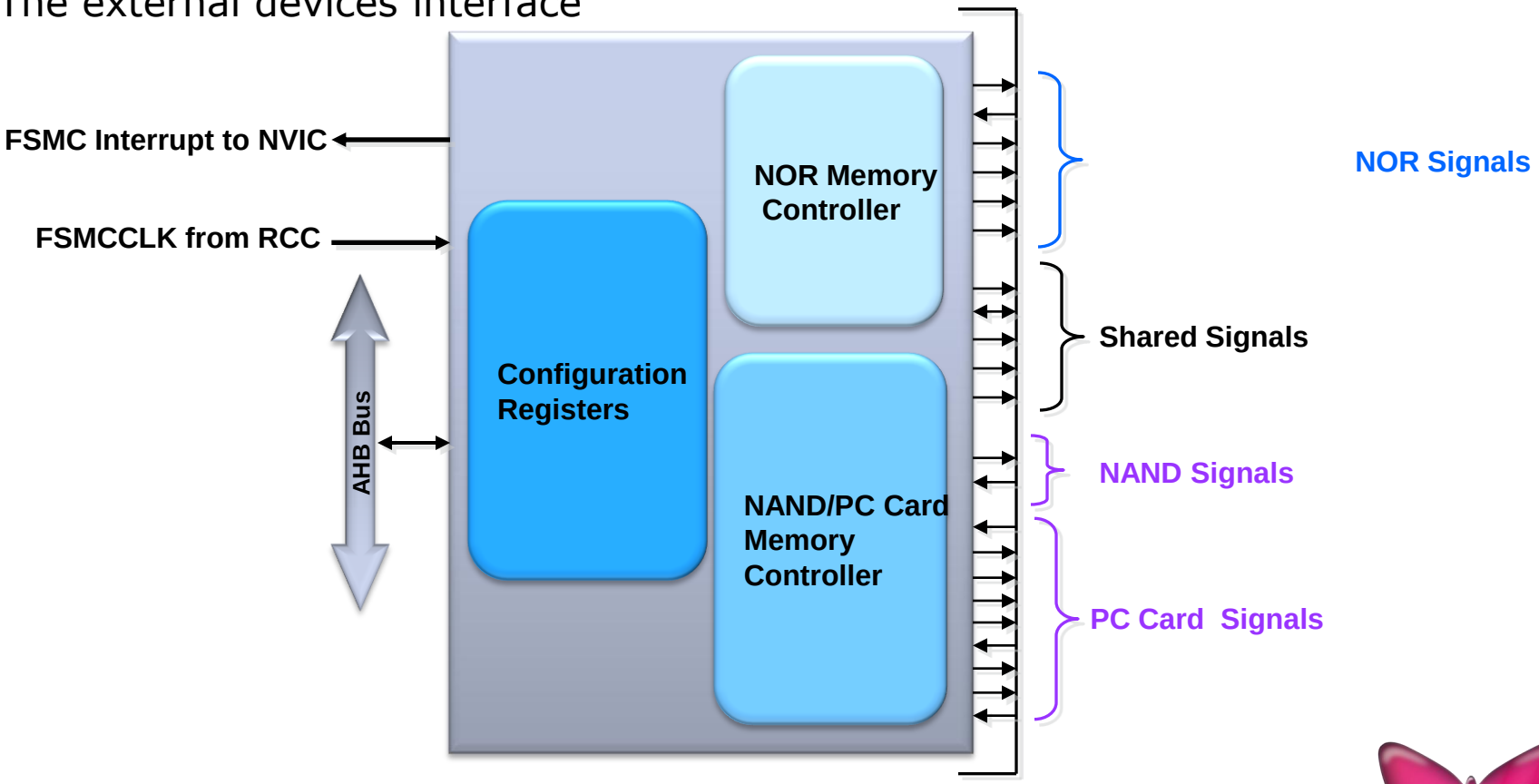
FLEXIBLE STATIC MEMORY CONTROLLER (FSMC)

FSMC Features

- The Flexible Static Memory Controller has the following main features:
  - **4 Banks** to support External memory
  - FSMC external access frequency is **60MHz when HCLK is at 168Hz**
  - Independent chip select control for each memory bank
  - Independent configuration for each memory bank
  - Interfaces with static memory-mapped devices including:
    - static random access memory (SRAM)
    - read-only memory (ROM)
    - NOR/ OneNAND Flash memory
    - PSRAM
  - Interfaces parallel LCD modules: Intel 8080 and Motorola 6800
  - Supports burst mode access to synchronous devices (NOR Flash and PSRAM)
  - NAND Flash and 16-bit PC Cards
    - With ECC hardware up to 8 Kbyte for NAND memory
    - 3 possible interrupt sources (Level, Rising edge and falling edge)
  - Programmable timings to support a wide range of devices
  - External asynchronous wait control
  - Enhanced performance vs. STM32F10x

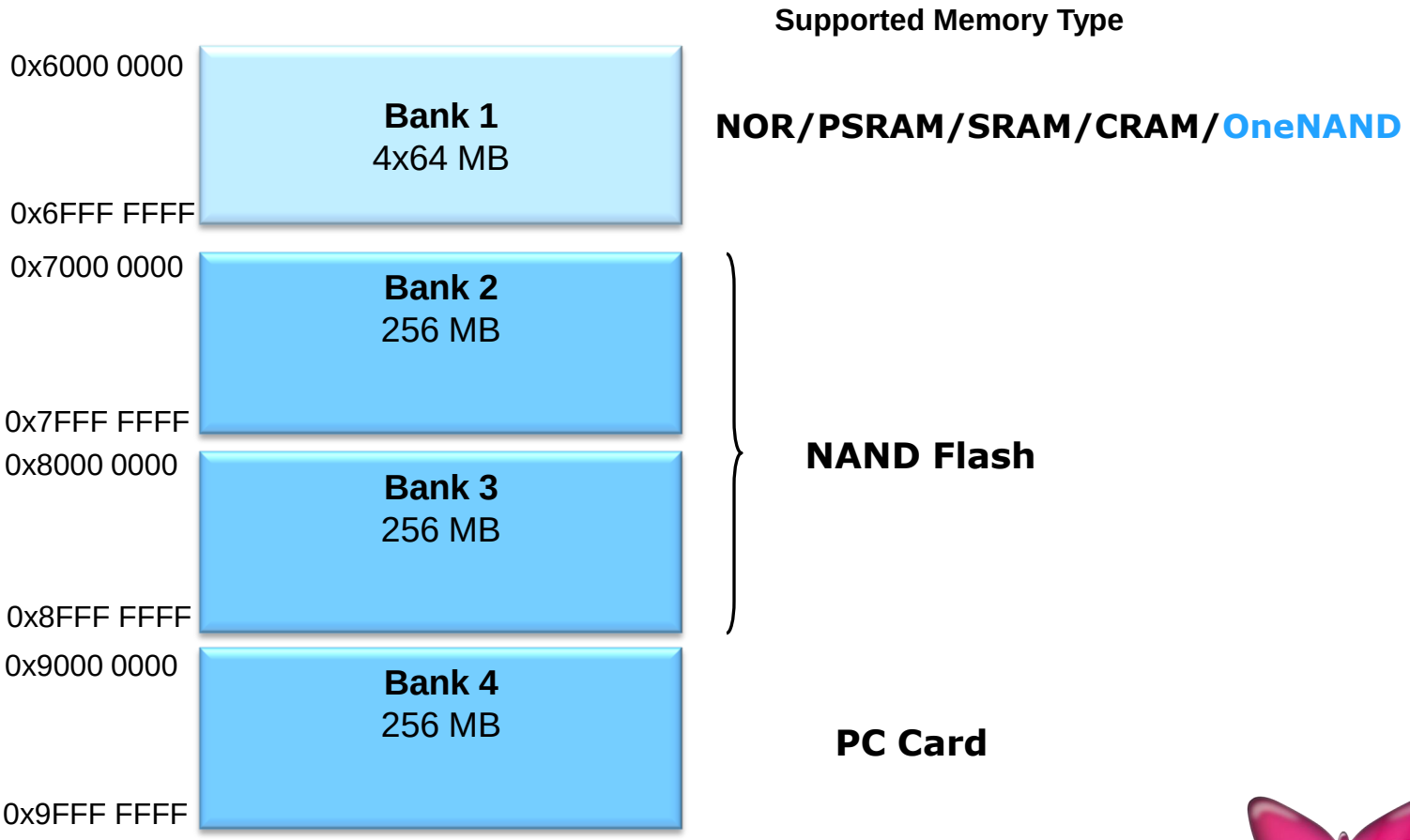
FSMC Block Diagram

- The FSMC consists of four main blocks:
  - The AHB interface (including the IP configuration registers)
  - The NOR Flash/PSRAM controller
  - The NAND Flash/PC Card controller
  - The external devices interface



FSMC Bank memory mapping

- For the FSMC, the external memory is divided into 4 fixed size banks of 4x64 MB each:
  - Bank 1 can be used to address NOR Flash, OneNAND or PSRAM memory devices.
  - Banks 2 and 3 can be used to address NAND Flash devices.
  - Bank 4 can be used to address a PC Card device.



Communication Peripherals

USB 2.0 ON-THE-GO FULL SPEED (OTG FS)

General Features

- Fully compliant with Universal Serial Bus Revision 2.0 specification
- Dual Role Device (DRD) controller that supports both device and host functions compliant with On-The-Go (OTG) Supplement Revision 1.3
- Can be configured as host-only or device-only controller
- Integrated PHY with full support of the OTG mode
- Full-speed (12 Mbits/s) and low-speed (1.5 Mbits/s) operation (only full speed for device)
- Dedicated RAM of 1.25 kB with advanced FIFO management and dynamic memory allocation

Device Mode Features

- 1 bidirectional control endpoint0
- Up to 3 IN and 3 OUT endpoints configurable to support Bulk, Interrupt or Isochronous mode
- Shared RxFIFO for OUT endpoints
- Dedicated TxFIFO for each IN endpoint
- FIFO management with multi-packet transfer support
- Soft disconnection feature (removing internal D+ pull-up)
- USB suspend/resume with exit from STOP mode

Host Mode Features

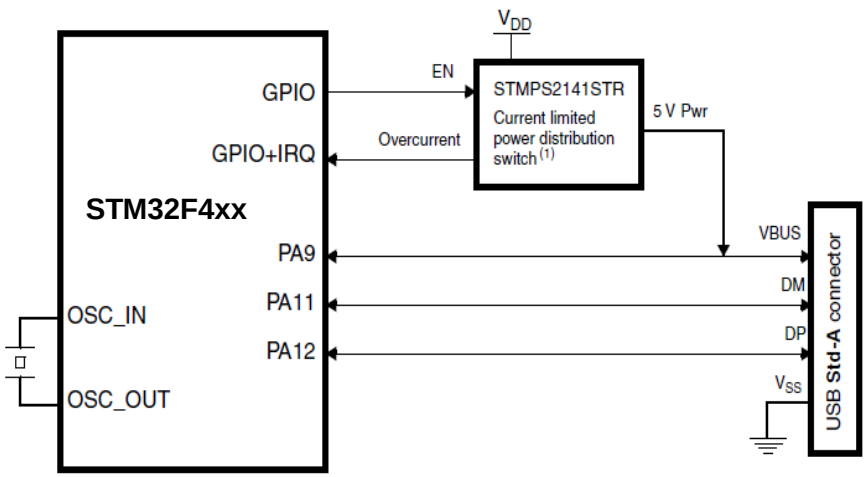
- Up to 8 host channels (pipes) dynamically reconfigurable to any type of USB transfer
- Shared RxFIFO for IN channels
- Shared periodic TxFIFO for interrupt and isochronous OUT channels
- Shared non-periodic TxFIFO for bulk and control OUT channels
- Separate queue management for periodic and non-periodic transfer requests with up to 8 requests for each queue
- Built-in hardware scheduler for giving priority to periodic transfers request over non-periodic transfers requests
- FIFO management with Mutli-packet transfer support

Embedded Full-speed OTG PHY Features

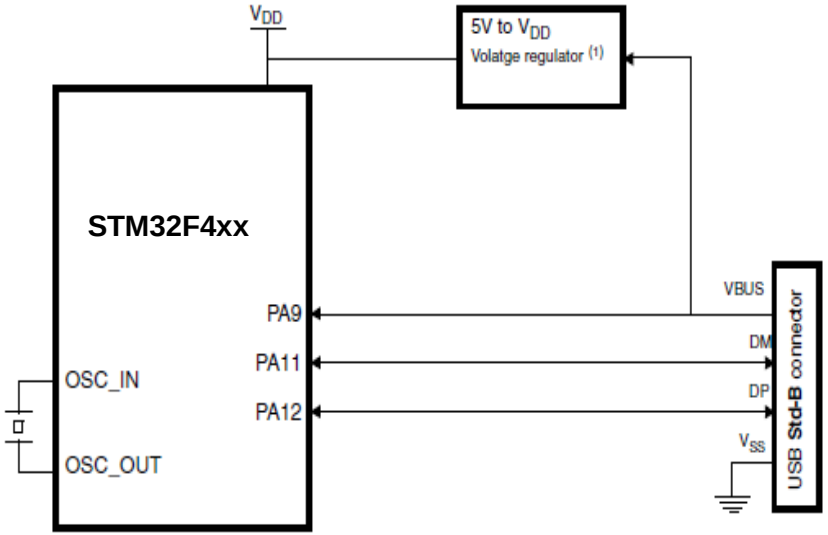
- FS/LS transceiver module used for Host/Device operation
- ID line detection for A/B device identification in OTG mode
- DP/DM integrated pull-up/pull-down resistors controlled by the USB core for device/host operation
- Vbus sensing and pulsing used for Session Request Protocol (SRP) in OTG mode

Hardware connections

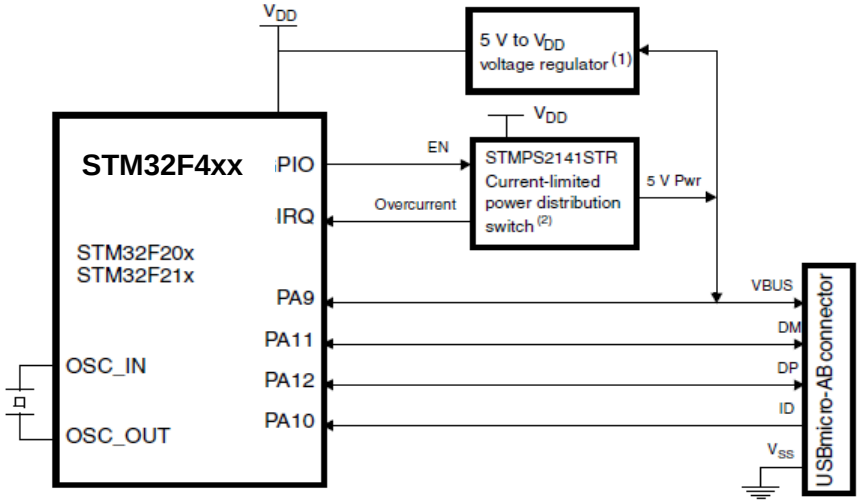
Host-only Operation



Device-only Operation



Dual role OTG (Host/Device) Operation



Communication Peripherals

USB 2.0 ON-THE-GO HIGH SPEED (OTG HS)

Main Features

- Fully compatible (@ register level) with the full-speed USB OTG peripheral
- High-speed (480 Mbit/s), full-speed and low speed operation in host mode and High-speed/Full-speed in device mode
- Three PHY interfacing options
  - Internal full-speed PHY (as for FS peripheral)
  - I2C interface for full-speed I2C PHY
  - ULPI bus interface for high-speed PHY
- DMA support with a dedicated FIFO of 4Kbytes

Device mode Features

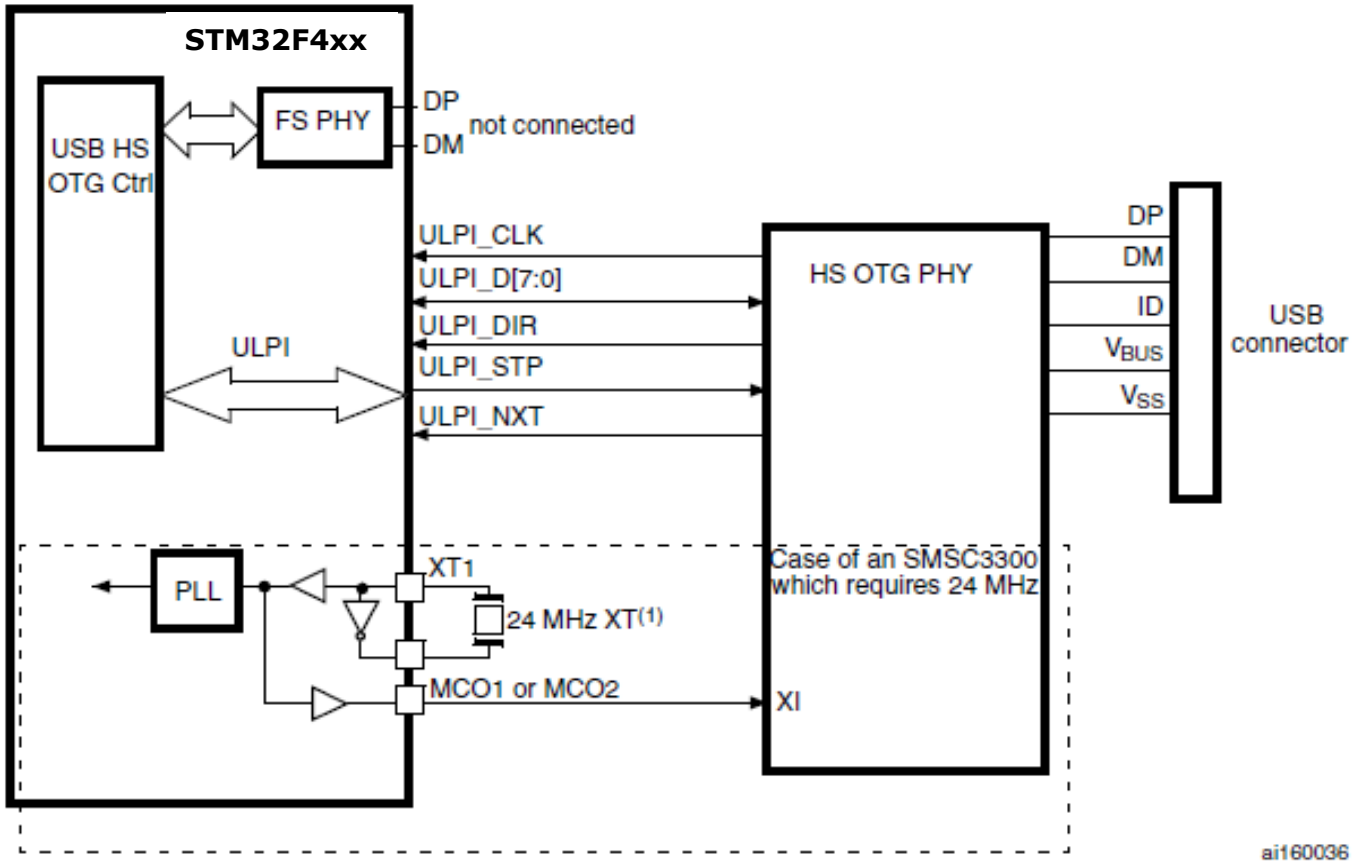
- Same as Full-speed mode with some extended/new features:
  - Up to 5 IN bulk, interrupt or isochronous endpoints (Vs 3 in FS)
  - Up to 5 OUT bulk, interrupt or isochronous endpoints (Vs 3 in FS)
  - Separate NVIC interrupt vector for EP1_IN
  - Separate NVIC interrupt vector for EP1_OUT

- NYET handshake sending
  - In High Speed mode, after receiving a packet, the core sends NYET handshake if it does not find threshold amount of free space available in the RxFIFO

Host mode features

- Same as Full-speed mode features
  - Up to 12 channels (Vs 8 channels in FS peripheral)
- High-speed protocol specific features
  - **PING protocol**: when a HS device is not ready to accept new packet, it sends the NYET or NAK handshake, in this case the host shouldn't continue data sending, but it should start the PING protocol to check periodically if device is ready to resume operation
  - **SPLIT protocol**: when the host is connected to a HS HUB, on which is connected a full/low speed device, the host will not wait response from the device, but it can do other HS transactions then after a period of time return to check if HS HUB has received any response from device
  - **Multi-transaction** during one micro-frame (125us) on isochronous transfers using (DATA0, DATA1, DATA2 and MDATA data PIDs)

ULPI High Speed PHY connection



USB High-Speed DMA features overview

- User can enable or disable the DMA mode
- The USB High-Speed DMA is connected to system bus matrix, it can support Burst transfers to/from SRAM or FSMC
- DMA can be enabled with or without FIFO thresholding
  - When FIFO thresholding is enabled a FIFO Rx/Tx levels can be configured to trigger DMA data transfer from RxFIFO to application buffer or from FIFO to USB bus
  - When FIFO thresholding is disabled, the trigger level is fixed to max packet size
- In DMA mode, as part of the transfer configuration parameters, application should program:
  - The application buffer destination address for OUT endpoints/IN channels
  - The application buffer source address for IN endpoints/OUT channels
- When DMA is enabled, a full transfer will be handled by DMA without CPU intervention for copying data to/from FIFOs
  - No more need for interrupts needed for FIFO management (TXFIFO empty interrupt , RxFIFO level interrupt)

Same as STM32F-2

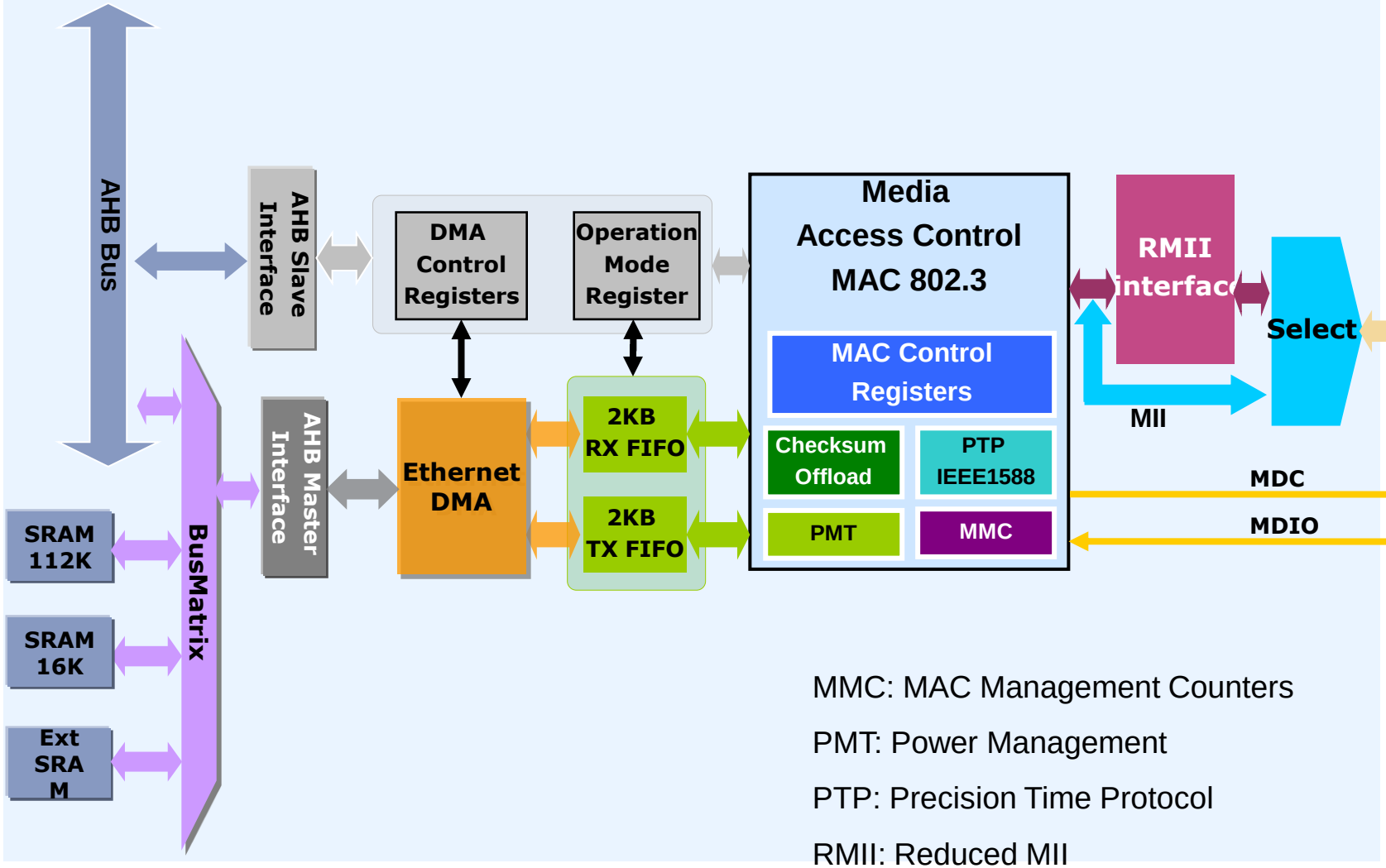
Communication Peripherals

ETHERNET MAC 10/100

Main Features

- Supports 10/100Mbits Half/Full-duplex operations modes
- MII/RMII PHY interface
- Several options for MAC address filtering
- IPv4 checksum offload during receive and transmit operation
- Dedicated DMA controller with two FIFOs (Rx/Tx) of 2KBytes each
  - Connected as AHB master to system bus matrix
- Ethernet Time Stamping support - IEEE1588 version 2
- Power management: Wake on LAN with Magic Packet or Wakeup frame
- MAC management Counters for statistics
- MII loopback mode for debug purpose

Ethernet Block Diagram



MMC: MAC Management Counters

PMT: Power Management

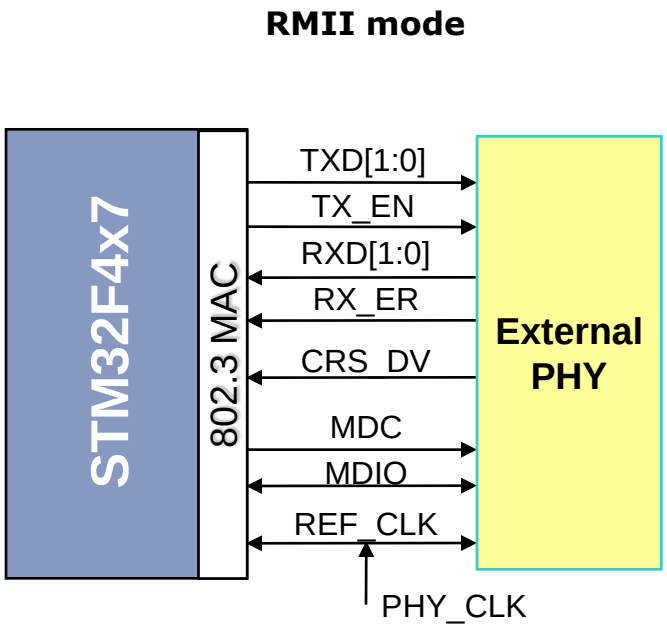
PTP: Precision Time Protocol

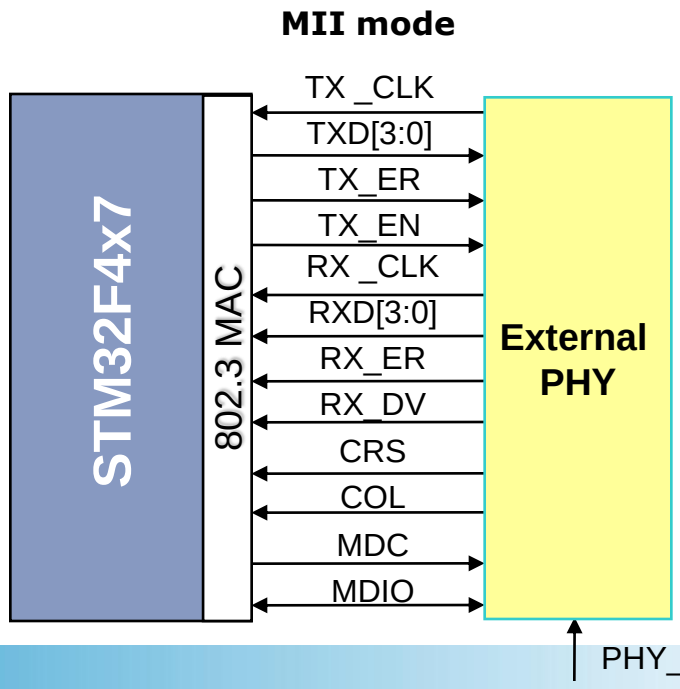
RMII: Reduced MII

MII: Media Independent Interface

Physical Layer Interface

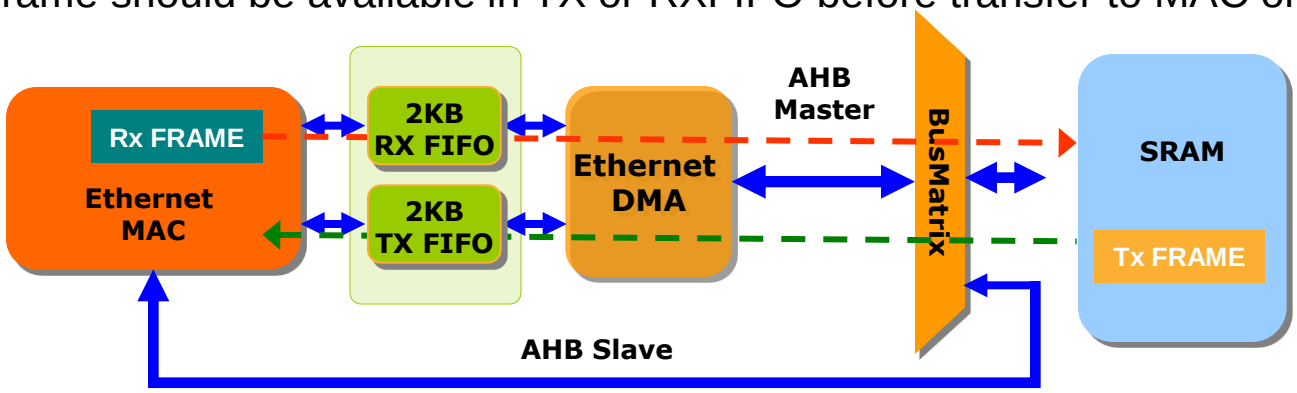
- Supports both Media Independent Interface (MII) and Reduced Media Independent Interface (RMII)
- RMII is a lower pin count alternative, which targets multi-port applications and low cost design
  - **MII** = 16 pins (8 data and 8 control)
  - **RMII** = 7 pins (4 data and 3 control)





MAC FIFOs

- Two modes for data transfer between FIFOs and SRAM:
  - Threshold mode
  - Store and forward mode
- Threshold mode
  - During frame transmission as soon as the TXFIFO level crosses a defined FIFO level (default is 64 bytes), the data start to be pushed to MAC for frame transmission
  - During frame reception as soon as the RXFIFO level crosses a defined FIFO level(default is 64 bytes), the DMA start to transmit the data to SRAM
- Store and forward mode
  - A full frame should be available in TX or RXFIFO before transfer to MAC or SRAM



Same as STM32F-2

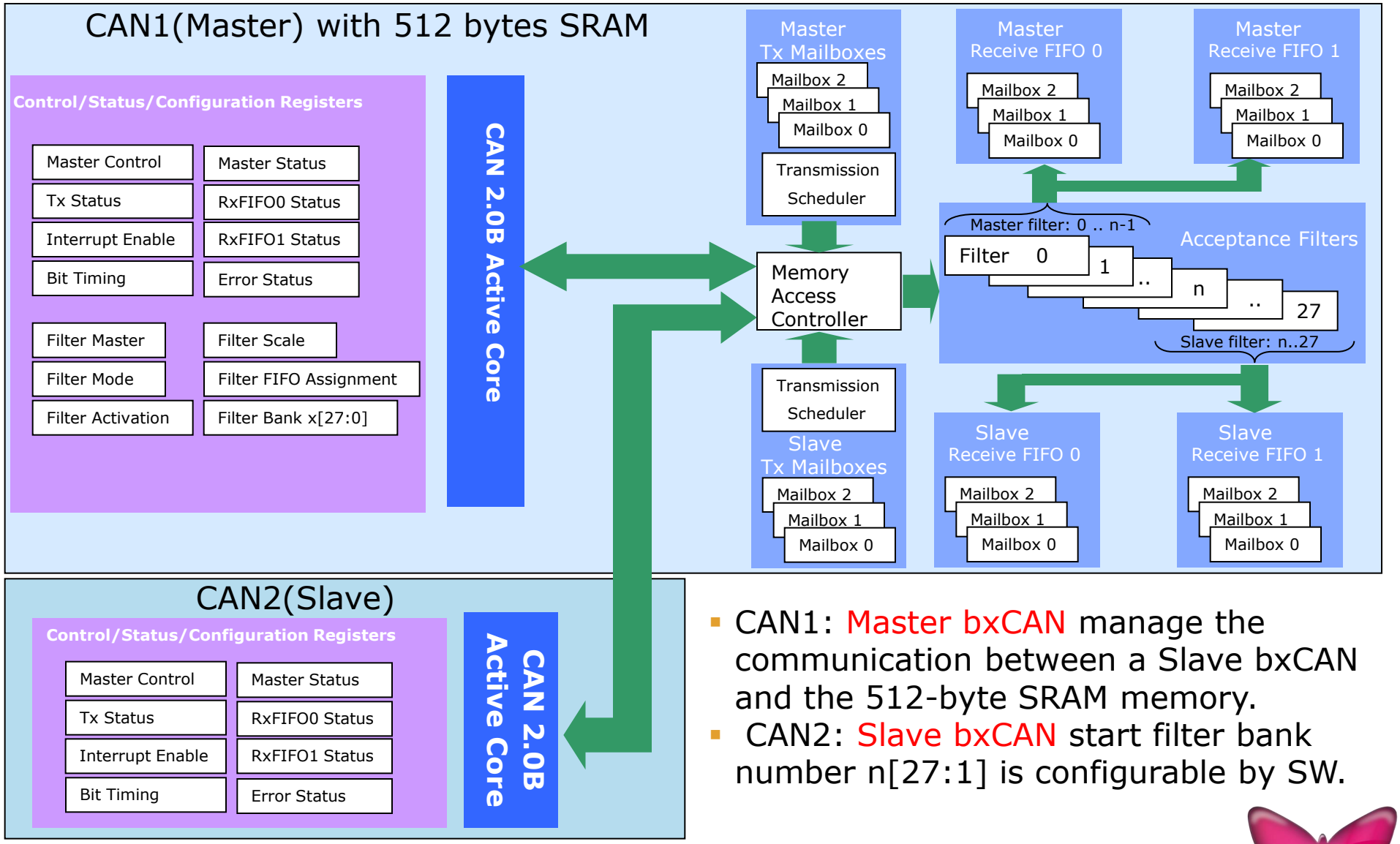
Communication Peripherals

CONTROLLER AREA NETWORK (BXCAN)

CAN Features

- **Dual CAN** 2.0 A, B Active w/ Bit rates up to 1Mbit/s, mapped on APB1
- Support time Triggered Communication
- Three transmit mailboxes w/ configurable transmit priority
- Two receive FIFOs with three stages and 28 filter banks shared between CAN1 and CAN2
- Time Stamp on SOF reception and transmission
- Maskable interrupts for easy software management
- Software efficient mailbox mapping at a unique address space
- 4 dedicated interrupt vectors: transmit interrupt, FIFO0 interrupt, FIFO1 interrupt and status change error interrupt
- The two CAN cells share a dedicated 512-byte SRAM memory and capable to work simultaneously with USB OTG FS peripheral

Block Diagram – Dual CAN



- CAN1: Master bxCAN manage the communication between a Slave bxCAN and the 512-byte SRAM memory.
- CAN2: Slave bxCAN start filter bank number n[27:1] is configurable by SW.

STM32F4

Advanced peripherals

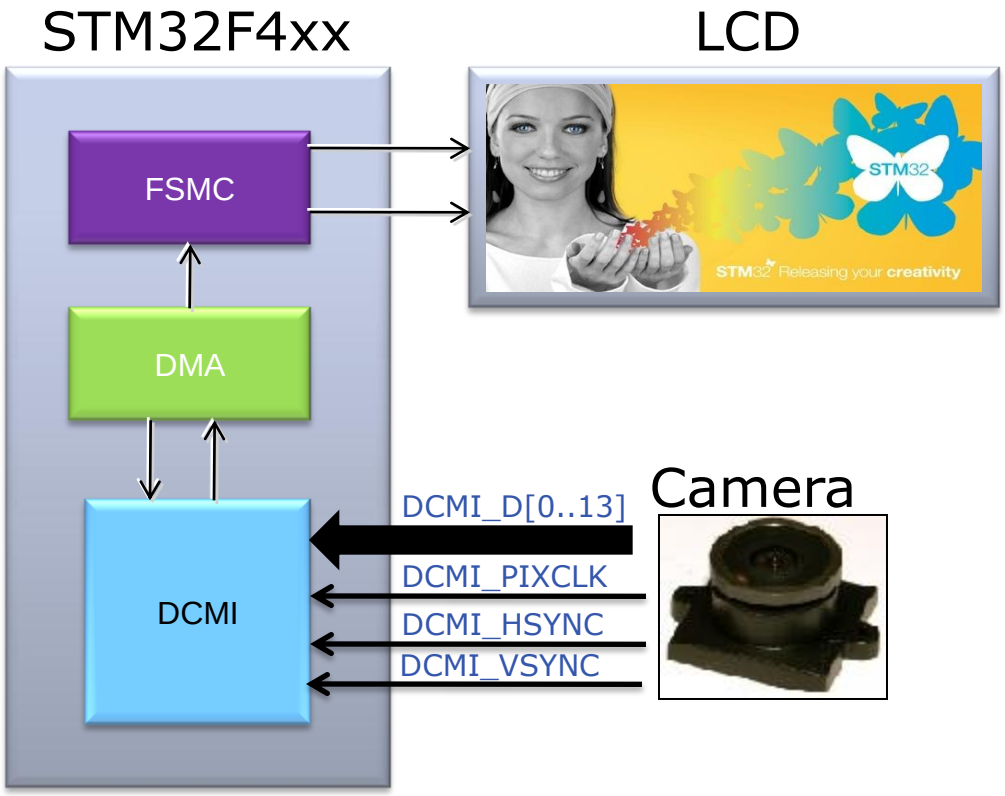
DIGITAL CAMERA INTERFACE (DCMI)

DCMI Features

- The Digital Camera Interface has the following main features:
  - 8-, 10-, 12- or 14-bit parallel interface
  - Continuous or snapshot mode
  - Crop feature
  - Supports the following data formats:
    - 8/10/12/14- bit progressive scan: either monochrome or raw bayer
    - YCbCr 4:2:2 progressive scan
    - RGB 565 progressive video
    - Compressed data: JPEG

- With a 48MHz PIXCLK and 8-bit parallel input data interface it is possible to receive:
  - up to 15fps uncompressed data stream in SXGA resolution (1280x1024) with 16-bit per pixel
  - up to 30fps uncompressed data stream in VGA resolution (640x480) with 16-bit per pixel

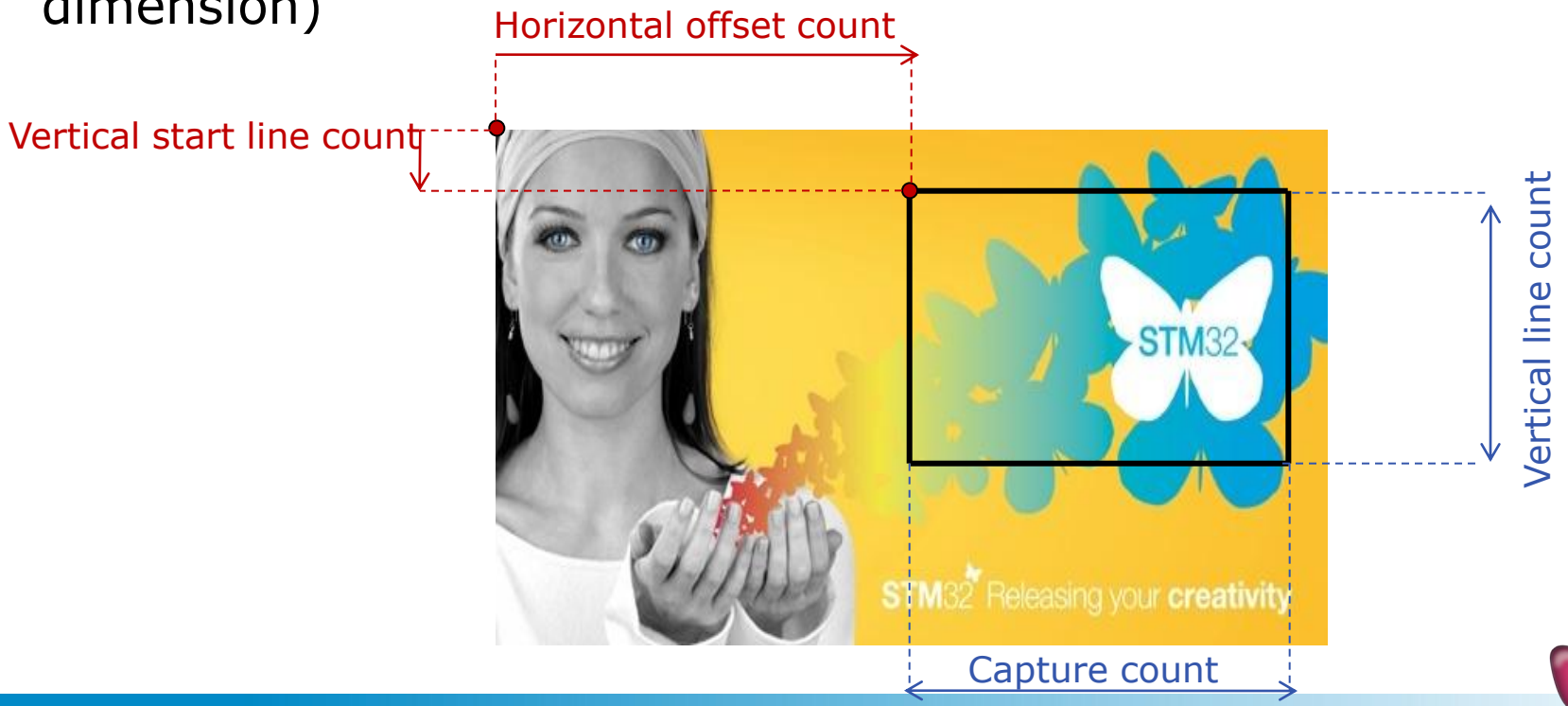
DCMI Data transfer



- The data are packed into a 32-bit data register (DCMI_DR) connected to the AHB bus
- 8x32-bit FIFO with DMA handling.

DCMI CROP feature

- The DCMI interface supports two types of capture:
  - The DCMI can select a rectangular window from the received image
  - The start coordinates and size are specified using two 32-bit registers DCMI_CWSTRT and DCMI_CWSIZE.
- The size of the window is specified in number of pixel clocks (horizontal dimension) and in number of lines (vertical dimension)



CRYPTOGRAPHIC PROCESSOR (CRYP)

Definitions

- **AES** : Advanced Encryption Standard
- **DES** : Data Encryption Standard
- **TDES** : Triple Data Encryption Standard

- Encryption/ Decryption modes
  - **ECB** : **Electronic code book** mode
  - **CBC** : **Cipher block chaining** mode or chained encryption
  - **CTR** : **Counter** mode (used for GCM : Galois Counter Mode)

    GCM is a combination of CTR and GHASH.

CRYP algorithms overview

	AES	DES	TDES
Key sizes	128, 192 or 256 bits	64* bits * 8 parity bits	192***, 128** or 64* bits * 8 parity bits : Keying option 1 ** 16 parity bits: Keying option 2 ***24 parity bits: Keying option 3
Block sizes	128 bits	64 bits	64 bits
Time to process one block	14 HCLK cycle for key = 128bits 16 HCLK cycle for key = 192bits 18 HCLK cycle for key = 256bits	16 HCLK cycles	48 HCLK cycles
Type	block cipher	block cipher	block cipher
Structure	Substitution-permutation network	Feistel network	Feistel network
First published	1998	1977 (standardized on January 1979)	1998 (ANS X9.52)

- AES : Advanced Encryption Standard
- DES : Data Encryption Standard
- TDES : Triple Data Encryption Standard

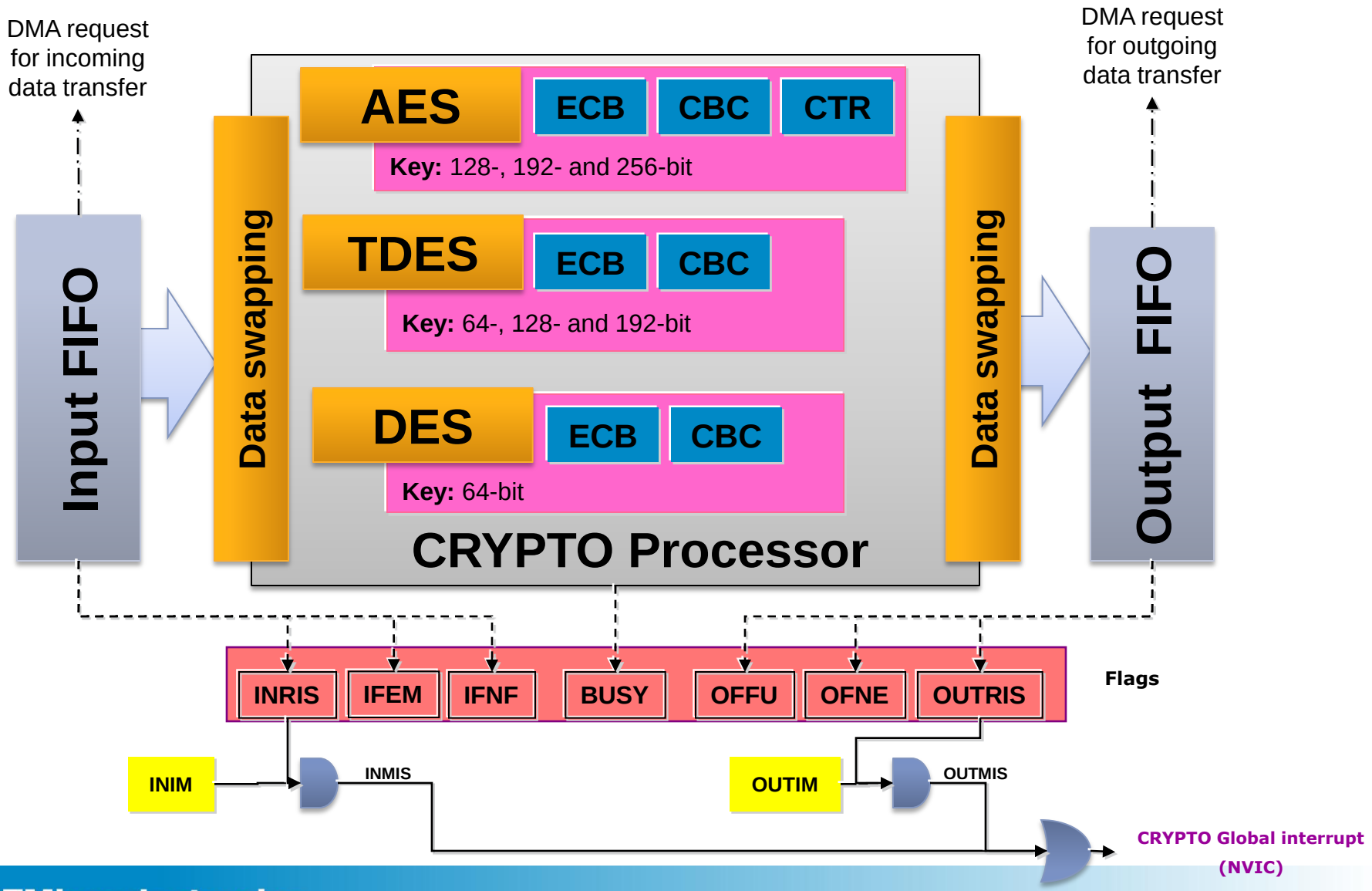
CRYP Features (1/2)

- Suitable for AES, DES and TDES enciphering and deciphering operations
- Runs at the same frequency as the CPU, up to 168 MHz.
- DES/TDES
  - Direct implementation of simple DES algorithms (a single key, K1, is used)
  - Supports the ECB and CBC chaining algorithms
  - Supports 64-, 128- and 192-bit keys (including parity)
  - 64-bit initialization vectors (IV) used in the CBC mode
  - 16 HCLK cycles to process one 64-bit block in DES
  - 48 HCLK cycles to process one 64-bit block in TDES

CRYP Features (2/2)

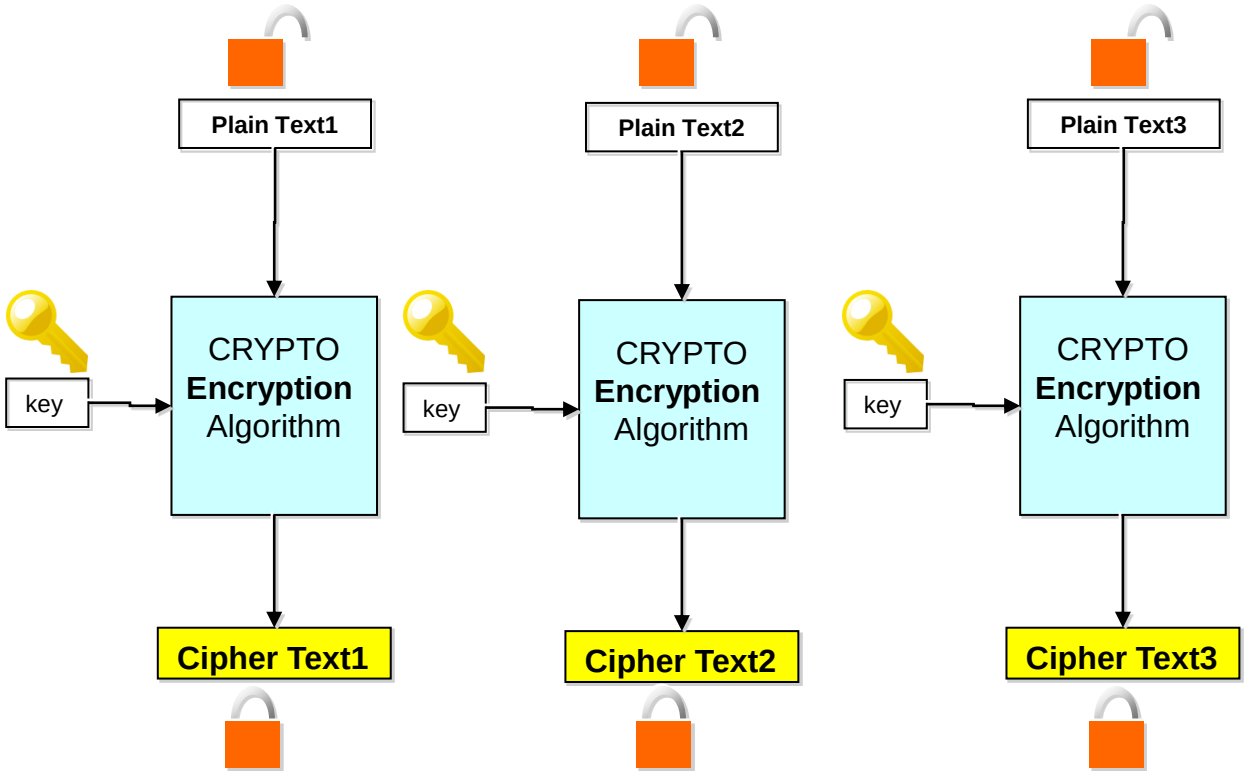
- AES
  - Supports the ECB, CBC and CTR chaining algorithms
  - Supports 128-, 192- and 256-bit keys
  - 128-bit initialization vectors (IV) used in the CBC and CTR modes
  - 14, 16 or 18 HCLK cycles (depending on the key size) to transform one 128-bit block in AES
- Common to DES/TDES and AES
  - IN and OUT FIFO (each with an 8-word depth, a 32-bit width, corresponding to 4 DES blocks or 2 AES blocks)
  - Automatic data flow control with support of direct memory access (DMA) (using 2 channels, one for incoming data the other for processed data)
  - Data swapping logic to support 1-, 8-, 16- or 32-bit data

CRYP Block Diagram



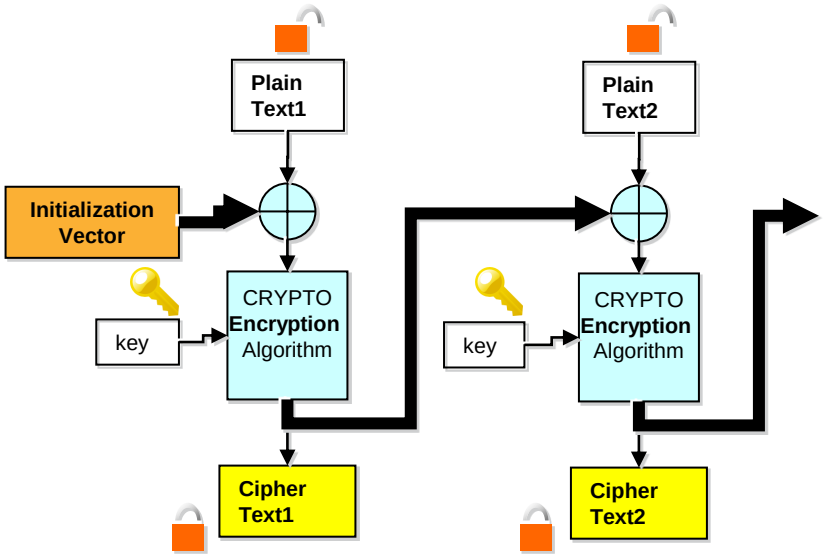
ECB Encryption

- The simplest of the encryption modes is the **Electronic codebook** (ECB) mode. The message is divided into blocks and each block is encrypted separately.
- The disadvantage of this method is that identical plaintext blocks are encrypted into identical cipher text blocks; thus, it does not hide data patterns well. To avoid this weakness, CBC or CTR modes can be used.

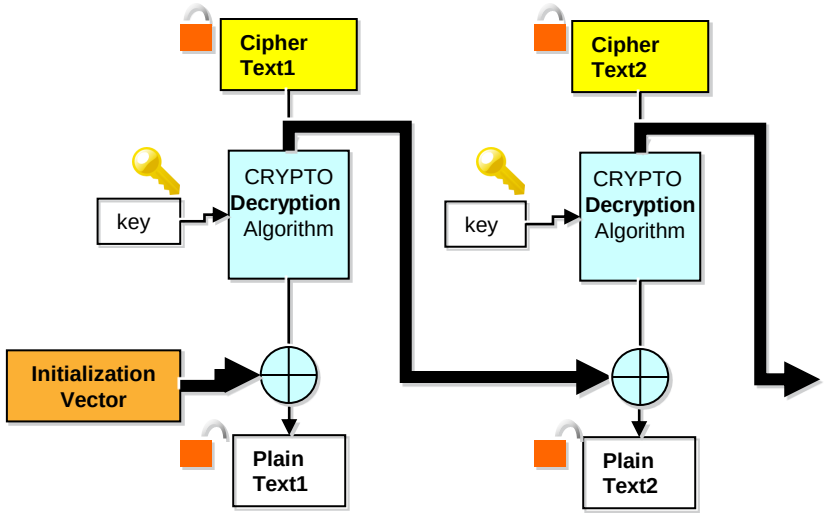


Cipher block chaining mode (CBC)

- CBC mode of operation was invented by IBM in 1976.
- In the CBC mode, each block of plaintext is XORed with the previous cipher text block before being encrypted.
- This way, each cipher text block is dependent on all plaintext blocks processed up to that point.
- To make each message unique, an initialization vector must be used in the first block.



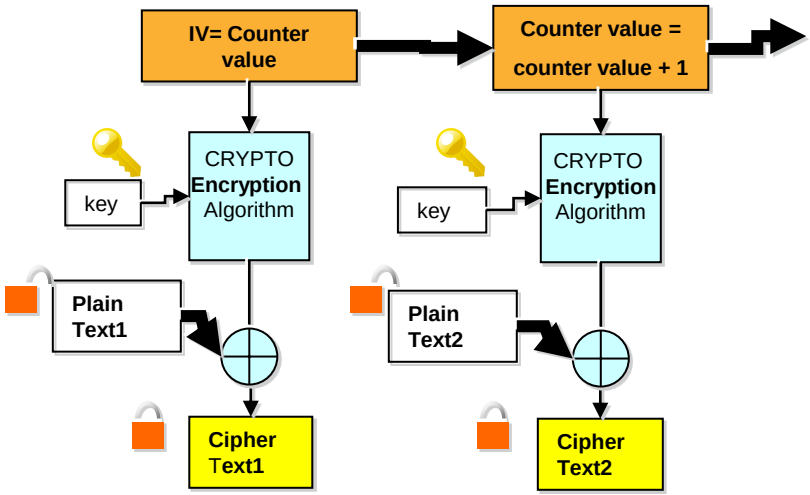
Encryption



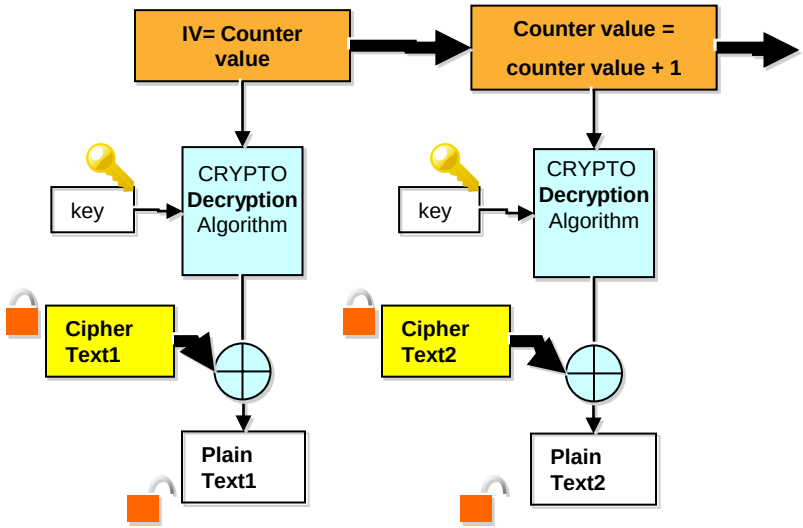
Decryption

Counter mode (CTR): AES only

- Counter mode turns a block cipher into a stream cipher. It generates the next key stream block by encrypting successive values of a "counter".
- The counter can be any function which produces a sequence which is guaranteed not to repeat for a long time, although an actual counter is the simplest and most popular.
- CTR mode is well suited to operation on a multi-processor machine where blocks can be encrypted in parallel.
- The IV/nonce and the counter can be concatenated, added, or XORed together to produce the actual unique counter block for encryption.



Encryption



Decryption

CRYP throughput

- Throughput in MB/s at 168 MHz for the various algorithms and implementations

	AES-128	AES-192	AES-256	DES	TDES
HW Theoretical	192.00	168.00	149.33	84.00	28.00
HW Without DMA	72.64	72.64	62.51	43.35	16.00
HW With DMA	**128.00**	**168.00**	**149.33**	**84.00**	**28.00**
Pure SW	**1.38**	**1.14**	**0.96**	**0.74**	**0.25**

CRYP and DMA

- The cryptographic processor provides an interface to connect to the DMA controller. The DMA operation is controlled through the CRYP DMA control register, CRYP_DMACR.
- 2 requests are available
  - Request DMA for outgoing data transfer from FIFO OUT
  - Request DMA for incoming data transfer to FIFO IN
- All request signals are de-asserted if the CRYP peripheral is disabled or the DMA enable bit is cleared (DIEN bit for the IN FIFO and DOEN bit for the OUT FIFO in the CRYP_DMACR register).

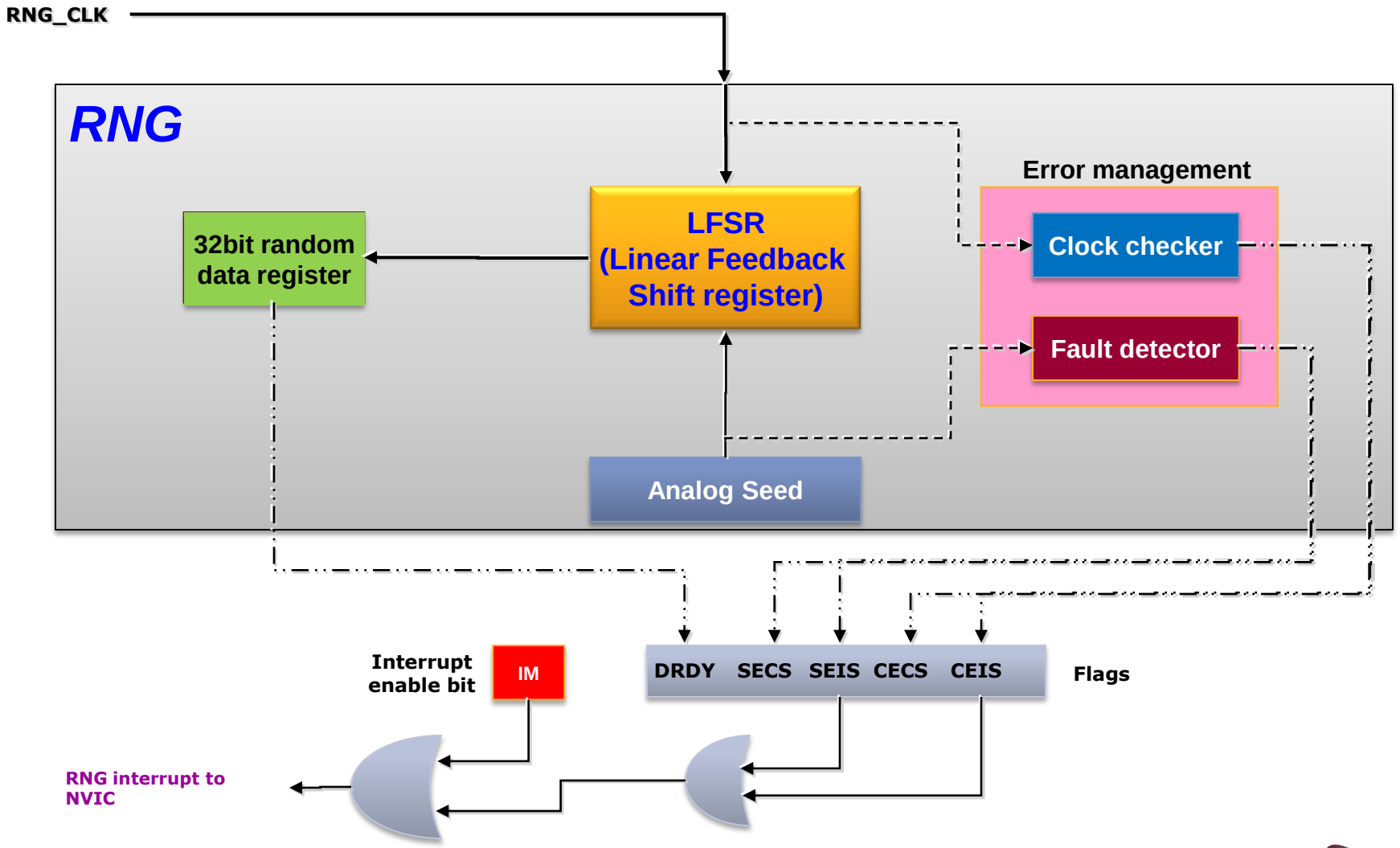
- **Important to know**
  - The DMA controller must be configured to perform burst of 4 words or less. Otherwise some data could be lost.
  - In order to let the DMA controller empty the OUT FIFO before filling up the IN FIFO, the OUTDMA Stream should have a higher priority than the INDMA Stream.

RANDOM NUMBER GENERATOR (RNG)

RNG Features

- 32-bit random numbers, produced by an analog generator (based on a continuous analog noise)
- Clocked by a dedicated clock (PLL48CLK)
- 40 periods of the PLL48CLK clock signal between two consecutive random numbers
- Can be disabled to reduce power-consumption
- Provide a success ratio of more than 85% to FIPS 140-2 (Federal Information Processing Standards Publication 140-2) tests for a sequence of 20 000 bits.
- 5 Flags
  - 1 flag occurs when Valid random Data is ready
  - 2 Flags to an abnormal sequence occurs on the seed.
  - 2 flags for frequency error (PLL48CLK clock is too low).
- 1 interrupt
  - To indicate an error (an abnormal sequence error or a frequency error)

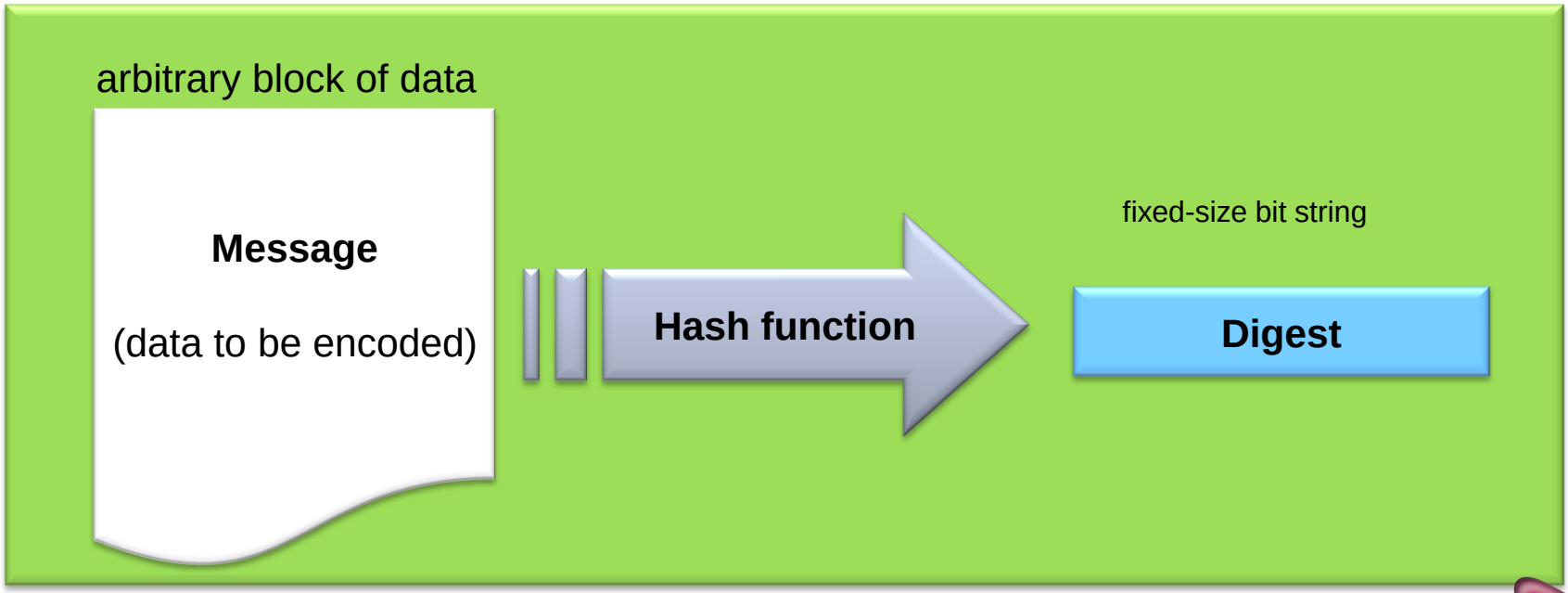
RNG Block Diagram



HASH PROCESSOR (HASH)

Definitions

- A **cryptographic hash function** is a deterministic procedure that takes an arbitrary block of data and returns a fixed-size bit string, the (**cryptographic**) **hash value**, such that an accidental or intentional change to the data will change the hash value. The data to be encoded is often called the **"message",** and the hash value is sometimes called the **message digest** or simply **digest**.



Definitions

- **SHA-1** : the **S**ecure **H**ash **a**lgorithm
- **MD5** : **M**essage-**D**igest algorithm **5** hash algorithm
- **HMAC** : (keyed-**H**ash **M**essage **A**uthentication **C**ode) algorithm
- **HASH** : Computes a SHA-1 and MD5 message digest for messages of up to $(2^{64} – 1)$ bits
- HMAC algorithms provide a way of authenticating messages by means of hash functions.
- HMAC algorithms consist in calling the SHA-1 or MD5 hash function twice on message in combination with a secret value (key).

HASH Features

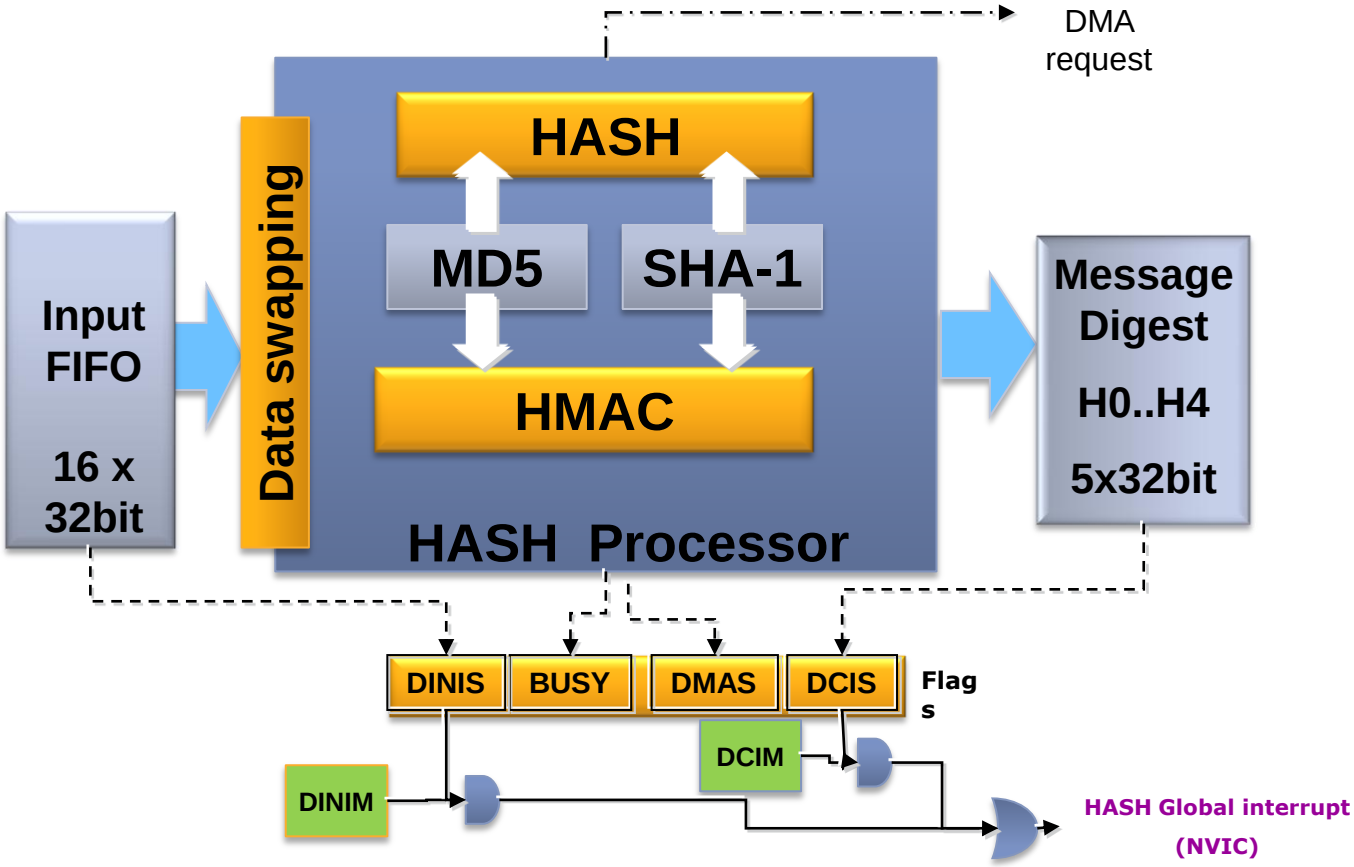
- Suitable for Integrity check and data authentication applications, compliant with:
  - FIPS PUB 180-2 (Federal Information Processing Standards Publication 180-2)
  - Secure Hash Standard specifications (SHA-1)
  - IETF RFC 1321 (Internet Engineering Task Force Request For Comments number 1321) specifications (MD5)

- AHB slave peripheral
- Fast computation of SHA-1 and MD5 :
  - 66 HCLK clock cycles in SHA-1
  - 50 HCLK clock cycles in MD5

- 5 × (32-bit) words (H0, H1, H2, H3 and H4) for output message digest, reload able to continue interrupted message digest computation
- Automatic data flow control with support for direct memory access (DMA)

- 32-bit data words for input data, supporting word, half-word, byte and bit bit-string representations, with little-endian data representation only

HASH Block Diagram



HASH throughput

- Throughput in MB/s at 168 MHz for SHA-1 and MD5 algorithms with different implementations

	MD5	SHA1
HW Theoretical	162.9	131.12
HW Without DMA	77.35	71.68
HW With DMA	**105.40**	**91.11**
Pure SW	**11.52**	**5.15**

Thank you

www.st.com/stm32f4