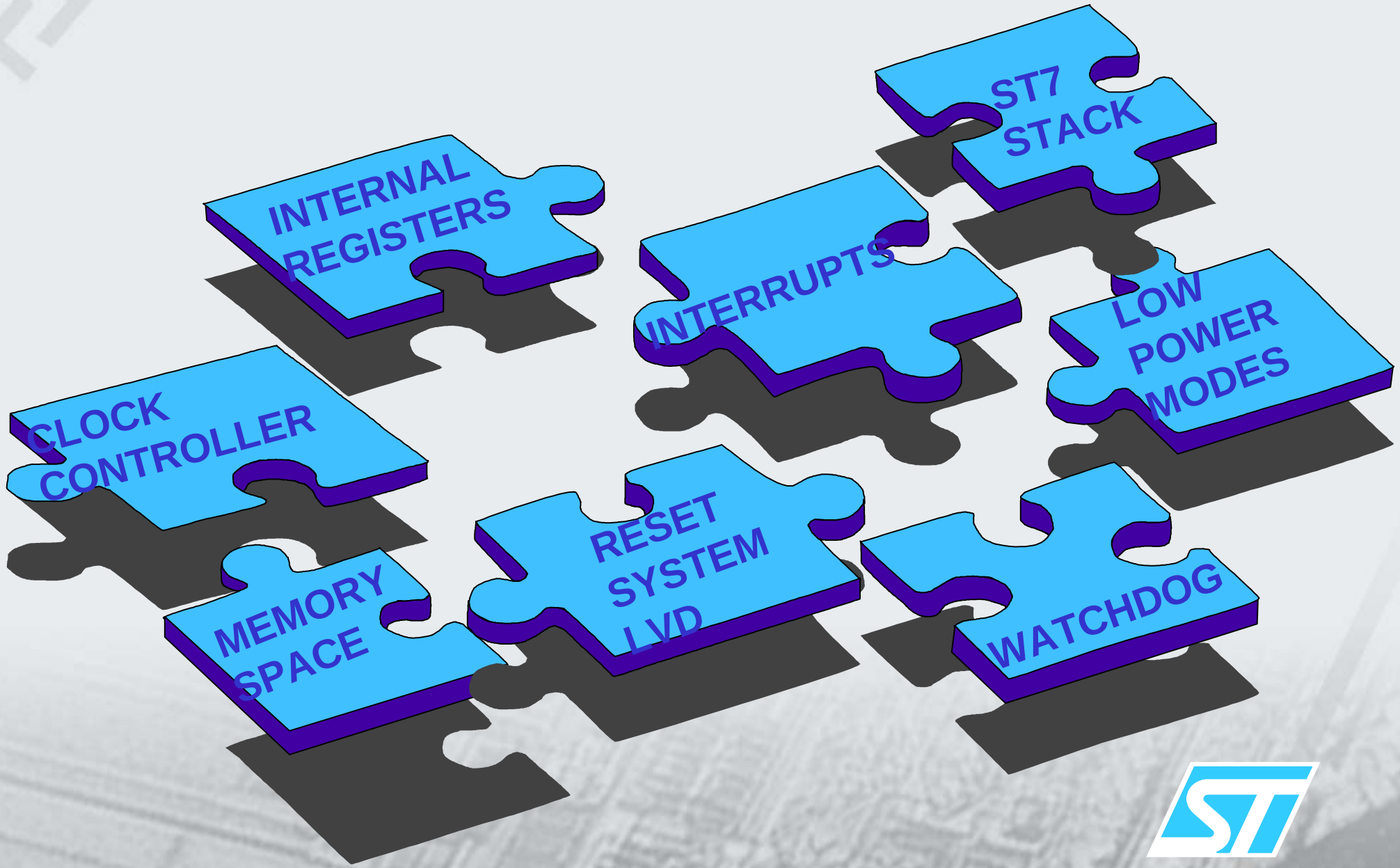
ST7 TECHNICAL TRAINING

1. INTRODUCTION
2. CORE
3. ADDRESSING MODES
4. ASSEMBLY TOOLCHAIN
5. STVD7 DEBUGGER
6. HARDWARE TOOLS
7. PERIPHERALS
8. ST-REALIZER II
9. C TOOLCHAINS

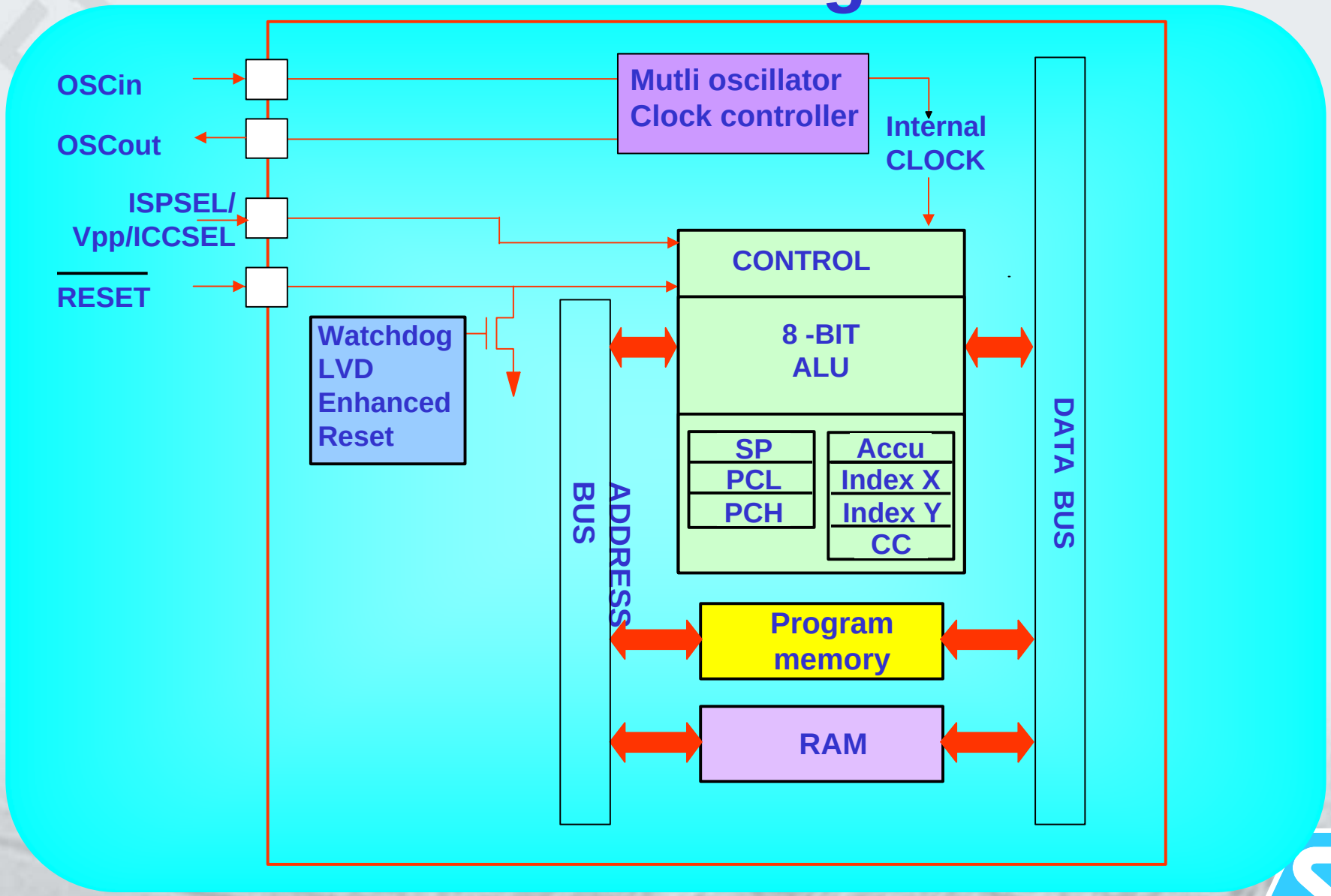
ST7 CORE


INTERNAL REGISTERS
ST7 STACK
INTERRUPTS
LOW POWER MODES
CLOCK CONTROLLER
RESET SYSTEM LVD
MEMORY SPACE
WATCHDOG

ST7 CORE
General Description

- The ST7 core (von neuman architecture) is built around:
  - An 8-bit Arithmetic and Logic Unit (ALU)
  - 6 internal registers: Accumulator (A), X and Y index registers, Program Counter (PC), the Stack Pointer (SP) and the Code Condition register (CC)
  - A controller block
- It interfaces with:
  - An on-chip oscillator
  - A reset block
  - Address and data buses to access memories and peripherals
  - An interrupt controller

ST7 CORE
Block Diagram



ST7 CORE
Internal Registers (1)

- The ACCUMULATOR is an 8-bit general purpose register used to hold:
  - Operands
  - Results of arithmetic and logic operation

- The X and Y REGISTERS are two 8-bit registers used to:
  - Create effective addresses
  - Store temporary data

Y is not automatically stacked. If needed, it must be done using the PUSH and POP instructions
Instructions using X are faster than the ones using Y

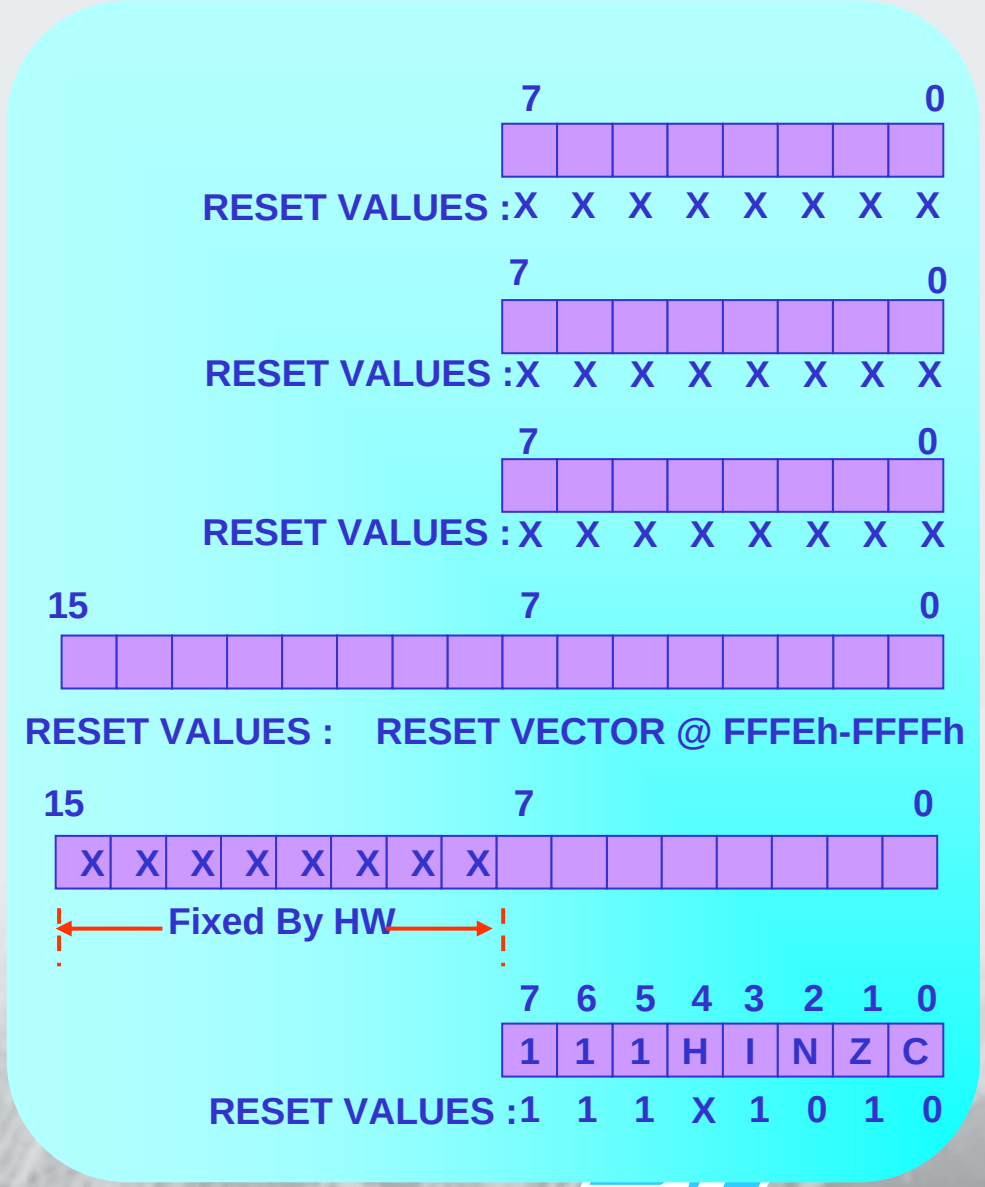
ST7 CORE
Internal Registers (2)

- The PROGRAM COUNTER PC is a 16-bit register used to store the address of the next instruction to be executed by the CPU. As a result, the ST7 can address up to 64k of program memory
- The STACK POINTER SP is a 16-bit register. The MSB is fixed by hardware
- The CODE CONDITION CC is a 5-bit register

BIT	NAME	DESCRIPTION
H	Half Carry Bit	H=1 when a carry occurs during ADD and ADC instructions
I	Interrupt mask	I=1 disabled the interrupt
N	Negative bit	N=1 if the result of the last operation is negative
Z	Zero bit	Z=1 if the result of the last operation is zero
C	Carry/Borrow bit	Affected when carry or borrow out occur and some inst. are executed

ST7 CORE
Internal Registers (3)

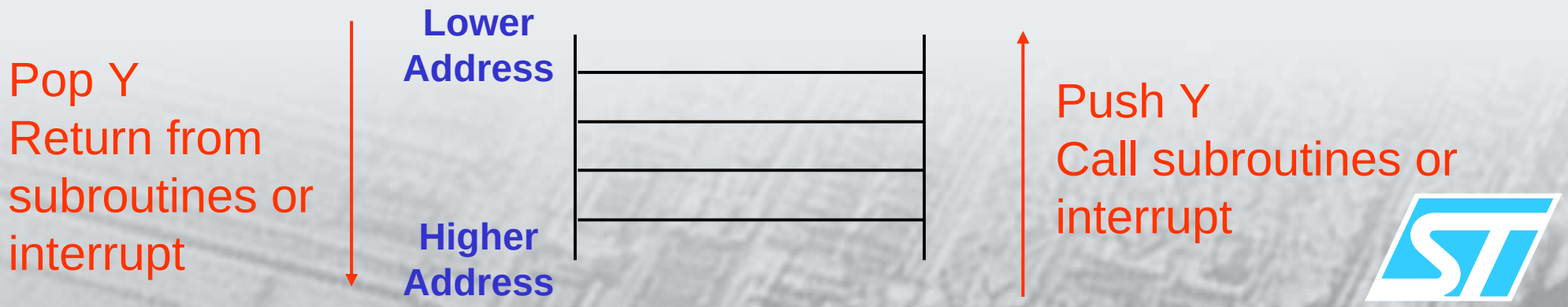
- Accumulator:
- X Index register:
- Y Index register:
- Program counter:
- Stack pointer:
- Condition code register:



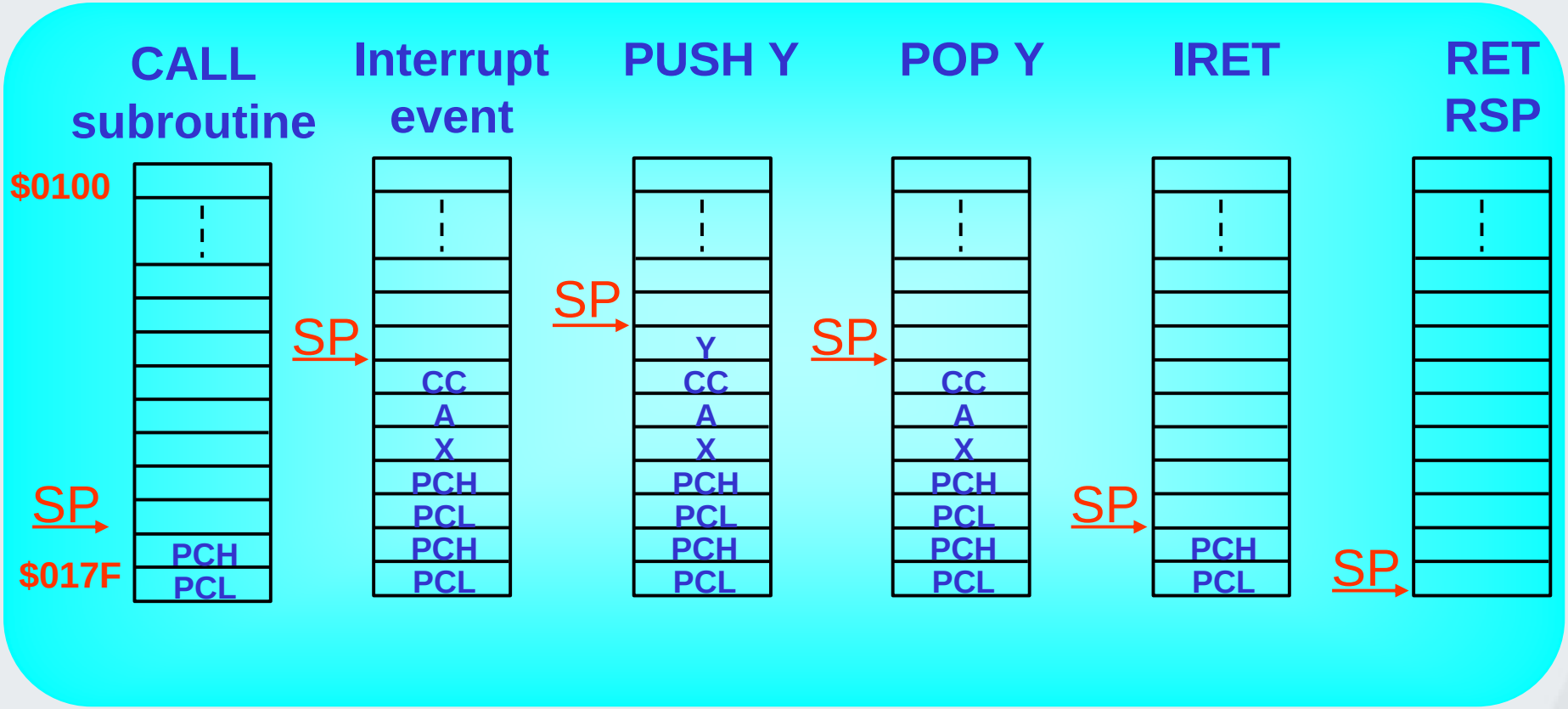
ST7 CORE
Stack manipulation (1)

- Purpose:
  - Save the CPU context during subroutine calls or interrupts
  - Save temporary user's data (PUSH and POP instructions)

- In case of overflow (lower limit exceeded):
  - SP rolls over to the higher address
  - Value at the higher address is overwritten so lost
  - Stack overflow is not indicated



ST7 CORE
Stack manipulation (2)



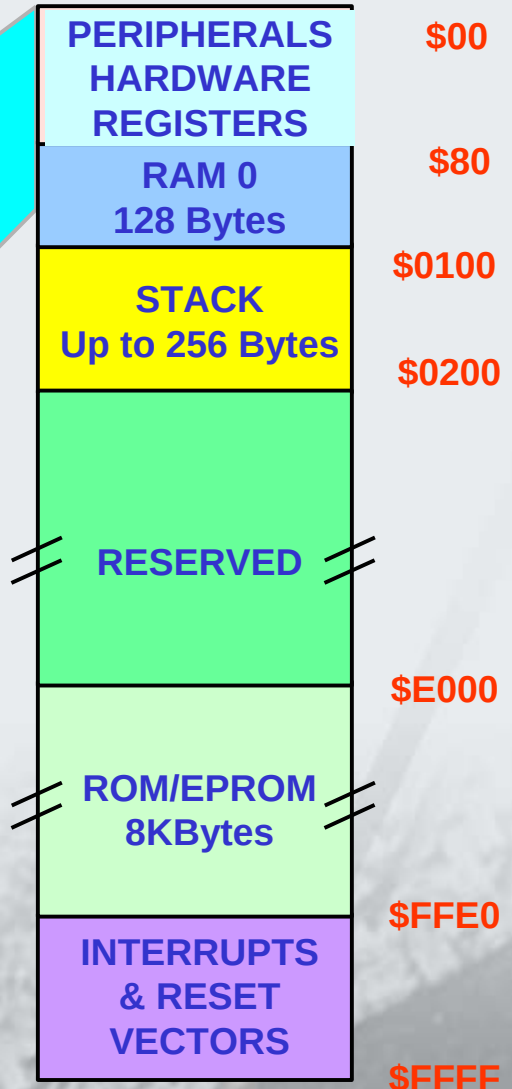
- Stack size and position is device dependent:
  - ST72254: 128 bytes ($0100 to $017F)
  - ST72521: 256 bytes ($0100 to $01FF)
  - ST7Lite0: 64 bytes ($00C0 to $00FF)
- Stack position can be reinitialized thanks to “ld SP, A” or reset by “RSP”

ST7 CORE
The memory space

- The memory can be made of 6 different blocks:

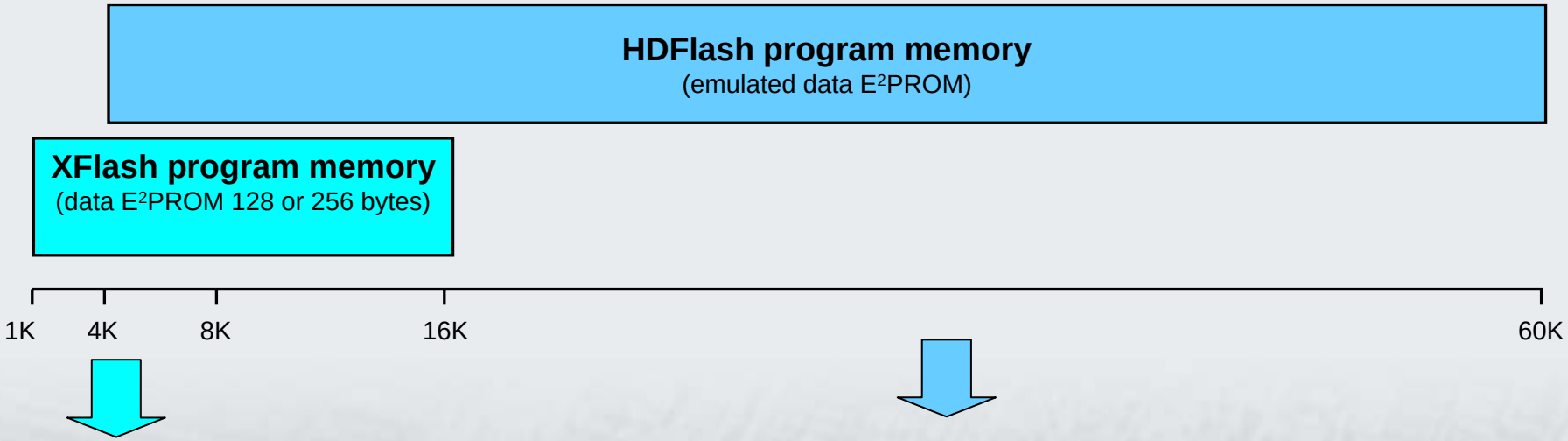
Short Addressing Mode Location

- ✓ Peripherals hardware register I/O Ports, TIM, ADC, WDG,SPI, I2C, EEPROM …
- ✓ Ram 0: ram in first page
- ✓ Stack: from 128 to 256 bytes (device dependent)
- ✓ EEPROM Data (up to 256 bytes)
- ✓ Program memory
- ✓ Interrupt and Reset vectors



TWO FLASH TECHNOLOGIES for cost vs. feature optimization

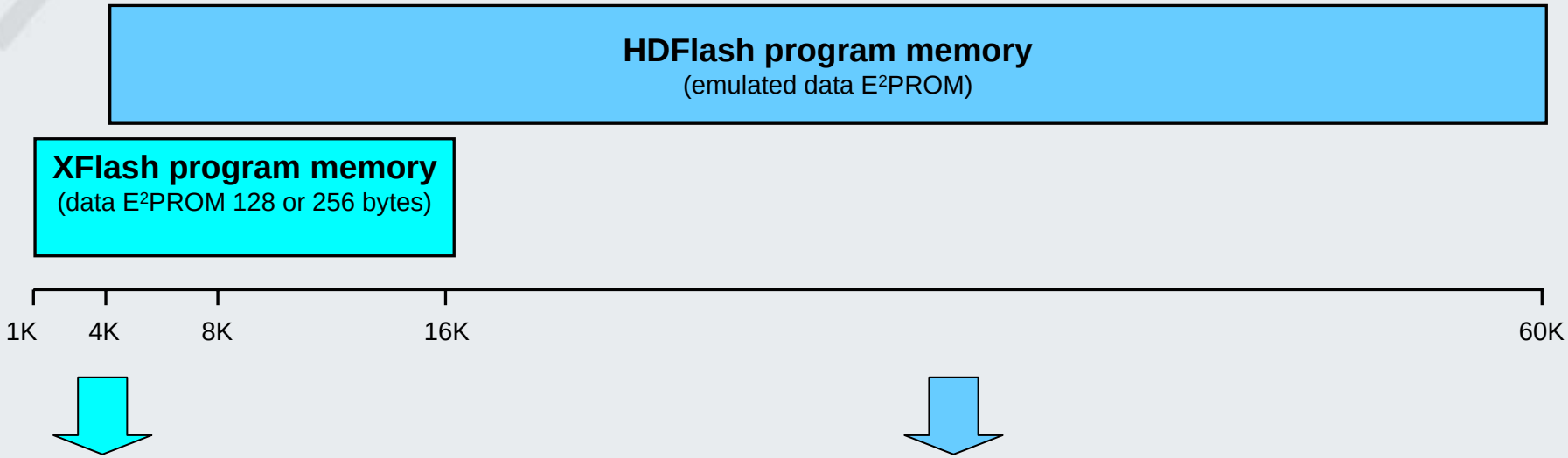
- General features:
  - **0.5µ technology**
  - **Very high reliability with** 20 year retention
  - **High cycling capability up to 300 K W/E**
  - **Patented high** memory protection **level**



- **XFlash**
  - **eXtended** Flash
  - E²prom technology
- **HDFlash**
  - **High Density** Flash
  - True Flash technology

TWO FLASH TECHNOLOGIES

for ST7 Product Family



- **XFlash** ST7 families
  - ST7Lite family from 1k to 8k
  - ST72F344 (16k)

- **HDFlash** ST7 families
  - ST72F52x/32x
  - ST72F56x
  - ST72F6xx
  - ST72F32x

TWO FLASH TECHNOLOGIES

for cost vs. feature optimization

- **XFlash**
  - Single voltage
  - Cycling:
    - 10K in program memory
    - 300K in data e²prom
  - Programming method:
    - Byte per byte or parallel (x32) programming with hardware control
  - Prog/Erase time:
    - 5ms for 1 to 32Bytes (including erasing)
  - Supply conditions
    - $2.4V \leq V_{DD} \leq 5.5V$

- **HDFlash**
  - Dual voltage
  - Cycling:
    - 100 in program memory
    - Emulated Data E²prom
  - Programming method:
    - Byte per byte or parallel (x256) programming and sectorized erasing with « Embedded Commands »
  - Prog/Erase time:
    - See dedicated slide
  - Supply conditions
    - $3.8V \leq V_{DD} \leq 5.5V$ (dedicated low voltage version)
    - $11.4V \leq V_{PP} \leq 12.6V$

FLASH PROGRAMMING

- How to program:
  - **ICP:** allows to PROGRAM or REPROGRAM (Xflash and HDFlash devices) the Program Memory when the micro is soldered on the application board or on an EPB socket.
  - **IAP:** allows to program the whole FLASH memory except sector 0, when the application is running

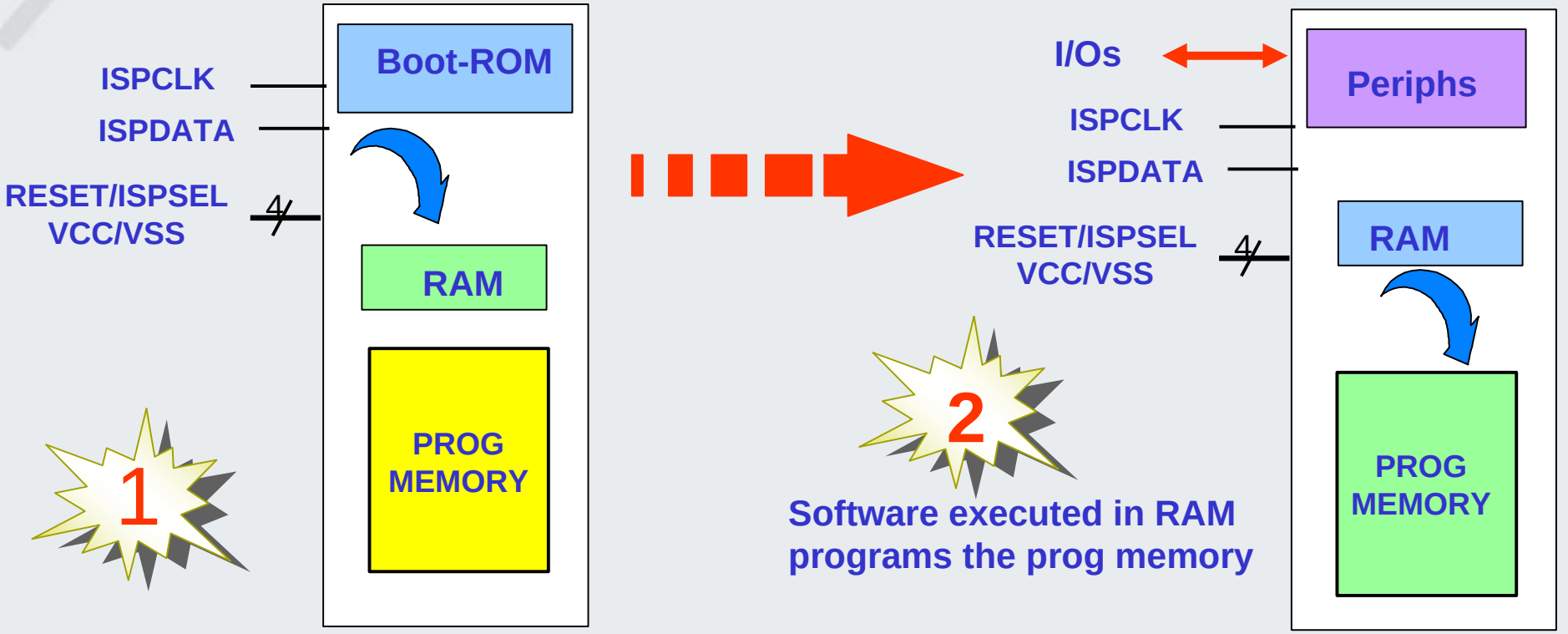
ST7 IN-APPLICATION PROGRAMMING (IAP)

- In-Application Programming
- Possible with Xflash and HDFlash
- Requirements:
  - Standard application reset sequence
  - Any interface supported by the microcontroller (SPI, USB, CAN…)
  - Always go through a RAM execution to program FLASH parts
- Capabilities:
  - Programming of the whole FLASH memory except sector 0

ST7 IN-SITU PROGRAMING (ISP)

- What is it for ?
  - To PROGRAM or REPROGRAM (flash devices) the Program Memory when the micro is soldered on the application board.
- Main features:
  - Only 6 wires are used (including VDD & VSS).
  - Do not need double voltage on the application board.
  - Supported by ST Visual Programmer (STVP7) tools.
- Performances:
  - ~ 5 s to program 8Kbytes.

ST7 IN-SITU PROGRAMING (ISP)



Boot-ROM allows an executable software to be downloaded in RAM through ISPCLK & ISPDATA

Software executed in RAM programs the prog memory

Software executed in RAM runs any applicative software with I/Os and peripherals access

In-Situ Programming uses 6 wires only

ST7 IN-CIRCUIT PROGRAMING (ICP)

- What is it for ?
  - To PROGRAM or REPROGRAM (flash devices) the Program Memory when the micro is soldered on the application board.
- Main features:
  - Only 4 to 7 wires are used (including VSS and clock depending the configuration).
  - Do not need double voltage on the application board.
  - Supported by the ST Visual Programmer (STVP7) tools.
- Performances:
  - XFlash: 1.3s to program 8Kbytes.
  - HDFlash: 640ms to program 8Kbytes
- ICP versus ISP
  - ICP based on a different protocol called ICC (In-Circuit Communication)
  - ICP more flexible cause the protocol adapts to the slowest part.
  - Flash Programming and ICC Reference Manual available on Internet.

Xflash Programming Timings with fCPU=8MHz

- 1 to 32 Bytes erasing/programming
  - 1~32 byte 5ms
  - 1 kbyte 160ms (// by 32)
- ICC protocol
  - ST7 Receive ~ 20.0 kbyte/s 32byte in ~1.6ms
  - ST7 Transmit ~ 15 kbyte/s 32byte in ~2.2ms
  - the ICC communication can be done during programming therefore masked (receive data+verify)
- Conditions: Fcpu=8MHz, no speed limitation from the programming tool for the communication

	Prog. by 32byte blocks + Verify
1kbyte	0.16s
4kbyte	0.64s
8kbyte	1.28s
16kbyte	2.56s

Communication timings are proportionnal with f_{CPU}

HDFlash Program & Erase Timings with fCPU=8MHz

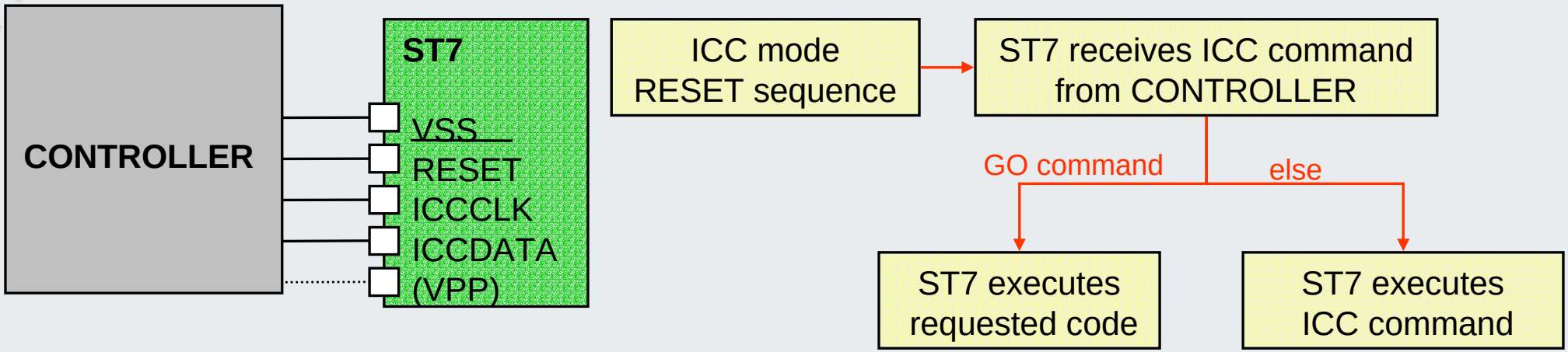
- Block programming
  - 256 bytes = 12.5ms
  - 32 kbyte ~ 1.6s
  - 60 kbyte ~ 3s
- Byte programming
  - 1 byte ~ 49µs
- VDD=5V and VPP=12V
- Conditions: Fcpu=8MHz, no speed limitation from the programming tool for the communication

- Sector erasing
  - S0 (4kbyte) 1.5~2s
  - S1 (4kbyte) 1.5~2s
  - S2 (8kbyte) 2~2.50s
    (24kbyte) 4~4.5s
    (52kbyte) 8~8.5s
- ICC protocol
  - ST7 Receive ~ 20.0 kbyte/s
  - ST7 Transmit ~ 17.5 kbyte/s
- Verify: CRC (time optimized)

	Prog. by 32byte blocks + Verify		Erase + Prog + Verify	
	Bit Map	Checksum	Bit Map	Checksum
16kbyte	2.8s	1.7s	7.8s~9.3s	6.7s~8.2s
32kbyte	5.6s	3.4s	12.6s~14.1s	10.4s~11.9s
60kbyte	10.2s	6.3s	21.2s~22.7s	17.3s~18.8s

All these timings are proportionnal with f_{CPU}

ICC MODE
Communicating with ICC Protocol

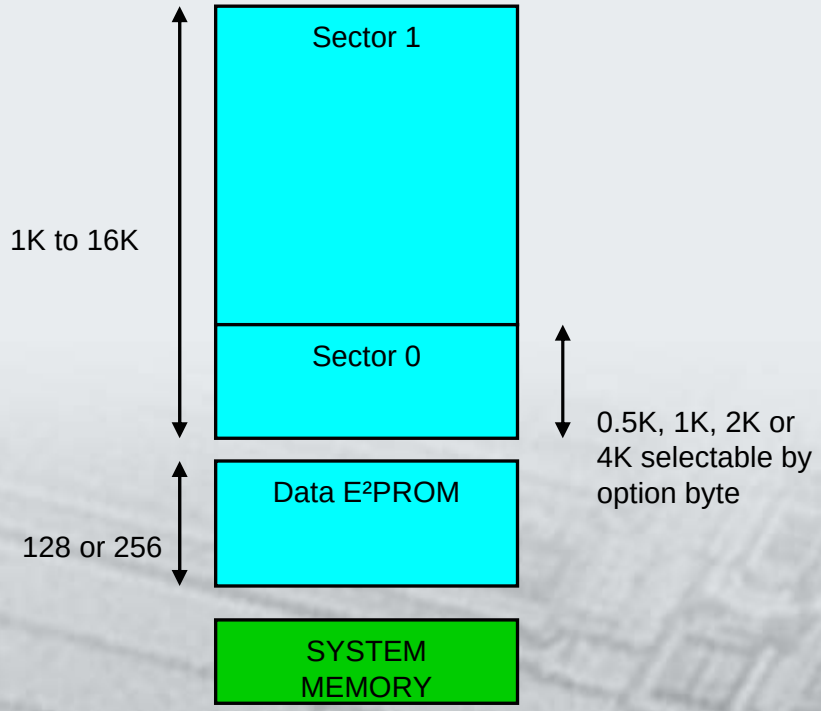


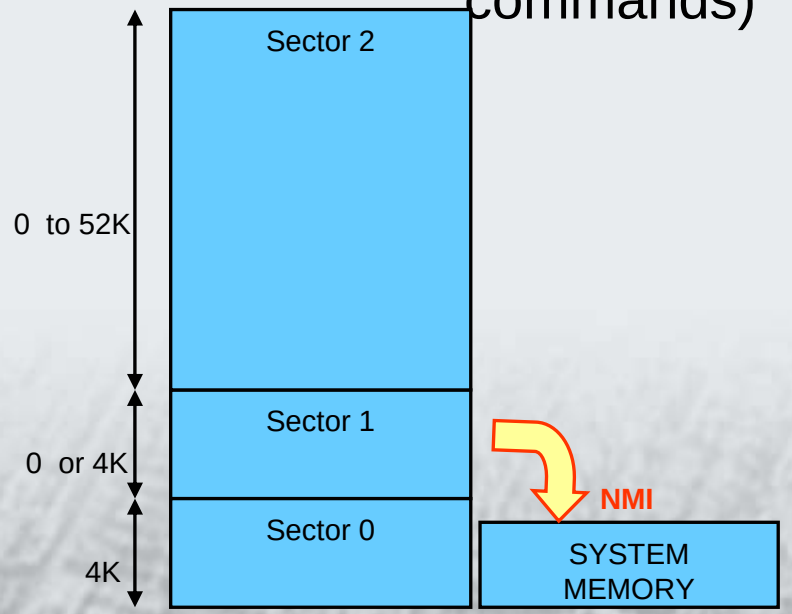
- **Dedicated RESET sequence to enter ICC mode**
- **Real 4 pin ICC protocol**
  - **ICCCLK, ICCDATA, RESET, VSS (and VPP when needed)**
- **No speed constraint for ST7 or Controller (PC/Tester)**
- **Minimum hardware needs to connect to a PC parallel port**
- **Allow ICD emulation and ICT without any other added external HW ressources**
- **Bidirectionnal comunication protocol**
- **Optimum command base protocol to manage ICP, ICT, ICD**
  - **Commands: Read, Write, Go, Get SP**

SYSTEM MEMORY

- Xflash :
  - Location
    - ✓ User space (read accessible by user)
  - Contains
    - ✓ ICC communication routines

- In HDFlash :
  - Location
    - ✓ Paginated (no read accessible by user)
  - Contains
    - ✓ ICC communication routines
    - ✓ « Embedded commands » (Flash programming commands)





PATENTED MEMORY PROTECTIONS

• Protection against	• Protection strategy
➢ Piracy	➢ READOUT/W protect on Xflash
➢ Unexpected reprogramming	➢ RASS (keys) (Register Access security System)
➢ Unexpected trouble occurs during programming	➢ Write Protected bootloader

IN-CIRCUIT DEBUGGING (ICD)

- Ability to debug a flash device using the ICC protocol
- Allows low-cost emulator strategy (InDart,…)
- ICD features:
  - Complex data or instructions breakpoints (up to 23 combinations)
  - Abort
  - Step by step
- Two ICD methods:
  - H/W : devices with Debug Module
  - S/W : XFlash without Debug Module (ST7FLITE0)
    - ✓ Limited (less features)
    - ✓ 1 or 2 hundred bytes needed in Flash
    - ✓ No real-time after a stop

Memory & CPU register Summary

- **How Many CPU registers belong to the ST7 Core?**
  - 6 registers (A, CC, X, Y, PC, SP)
- **Is it possible to place and read data in ROM (program memory)?**
  - Yes, lookup table can be placed in ROM
- **Is it possible to execute code located in RAM ?**
  - Yes, it is used in ISP and ICP modes to program the Program memory
- **Is the Stack handled automatically by the ST7 core ?**
  - Yes, the return address is loaded when a CALL is executed
  - The return address & the CPU registers are saved when the interrupt process is activated

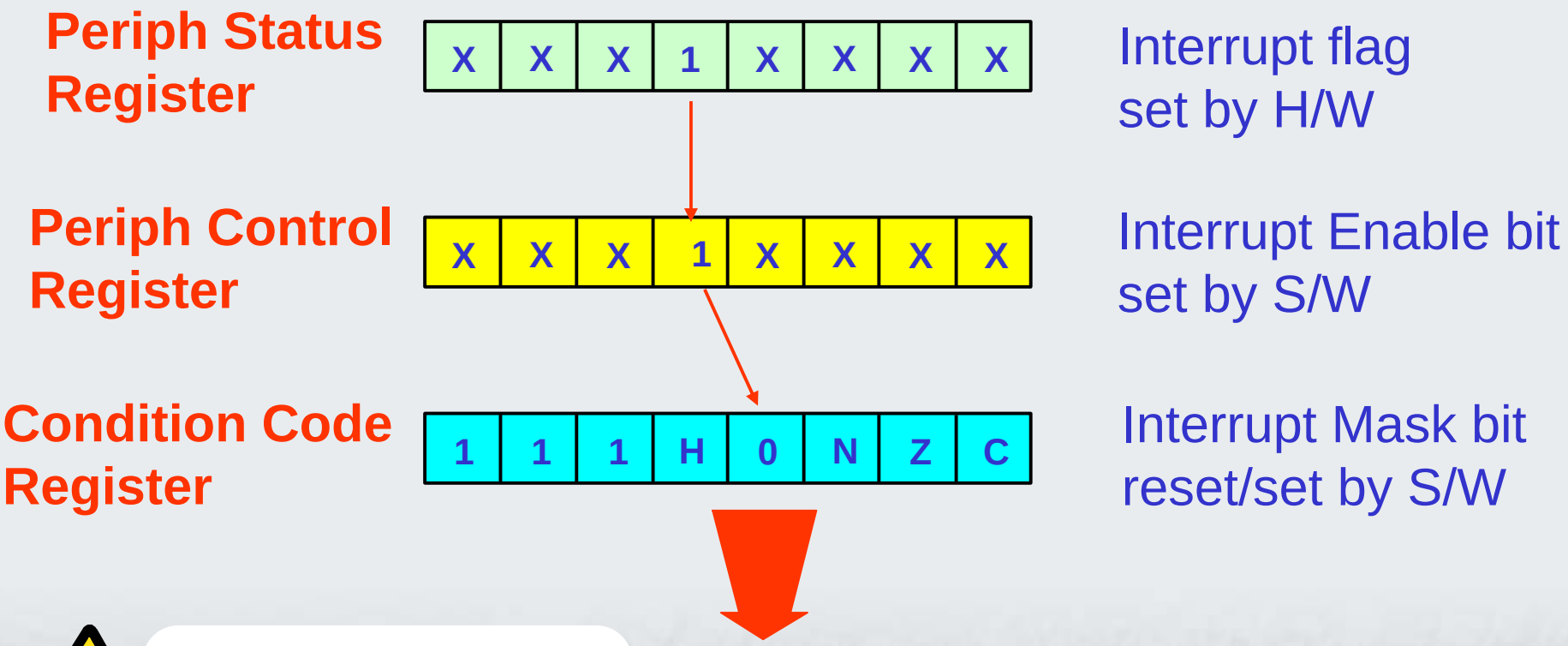
ST7 INTERRUPTS
Overview

- Except for the software interrupt (TRAP Instruction), all interrupts can be masked by setting the I BIT in CC
- When an interrupt occurs:
  - The context is saved on the stack (CC, A, X, PC)
  - All other interrupts are masked (the I bit is set By H/W)
  - The interrupt vector is loaded in the Program Counter
- When return from interrupt is executed:
  - The original context is automatically restored (CC, A, X, PC)
  - Interrupts are enabled (I bit reset)
- Priority between interrupts is given by the interrupt address vector (Higher address = higher priority)

ST7 INTERRUPTS
ST72254 Interrupt mapping

CPU INTERRUPT	INTERRUPTS	REGISTER	FLAG NAME	INTERRUPT SOURCE	VECTOR ADDRESS
Reset	Reset	N/A	N/A	-	FFFEh-FFFFh
Trap (instruction)	Software	N/A	N/A	I0	FFFCh-FFFDh
External Interrupt 0	Port A	N/A	N/A	I1	FFFAh-FFFBh
External Interrupt 1	Port B and Port C	N/A	N/A	I2	FFF8h-FFF9h
CSS	Clock Filter Interrupt	CRSR	CCSD	I3	FFF6h-FFF7h
SPI	Transfer Complete Mode Fault	SPI Status	SPIF MODF	I4	FFF4h-FFF5h
Timer A	Input Capture 1 Output Compare 1 Input Capture 2 Output Compare 2 Timer Overflow	Timer A Status	ICF1 OCF1 ICF2 OCF2 TOF	I5	FFF2h-FFF3h
Timer B	Input Capture 1 Output Compare 1 Input Capture 2 Output Compare 2 Timer Overflow	Timer B Status	ICF1 OCF1 ICF2 OCF2 TOF	I7	FFEEh-FFEFh
I2C	Byte Transfer finished Bus Error STOP Detection	I2C Status	BTF BERR SSTOP	I12	FFE4h-FFE5h

ST7 INTERRUPTS
Peripheral Int management



Context switch takes 10 CPU clock cycles

ST7 INTERRUPT
Peripheral Int management

- SOFTWARE EXAMPLE
  - **.Main**

```
...
BSET  Control_reg, #IT_enable          ; Enable Periph interrupt
RIM                                    ; Clear I bit in CC regis
...                                    ; ie interrupt enabled.
```

  - **.Int_routine**

```
...
BRES Status_reg, #IT_flag              ; Avoid to process the
                                       ; same interrupt forever
IRET                                   ; Return from interrupt
...
```

ST7 Interrupt Summary

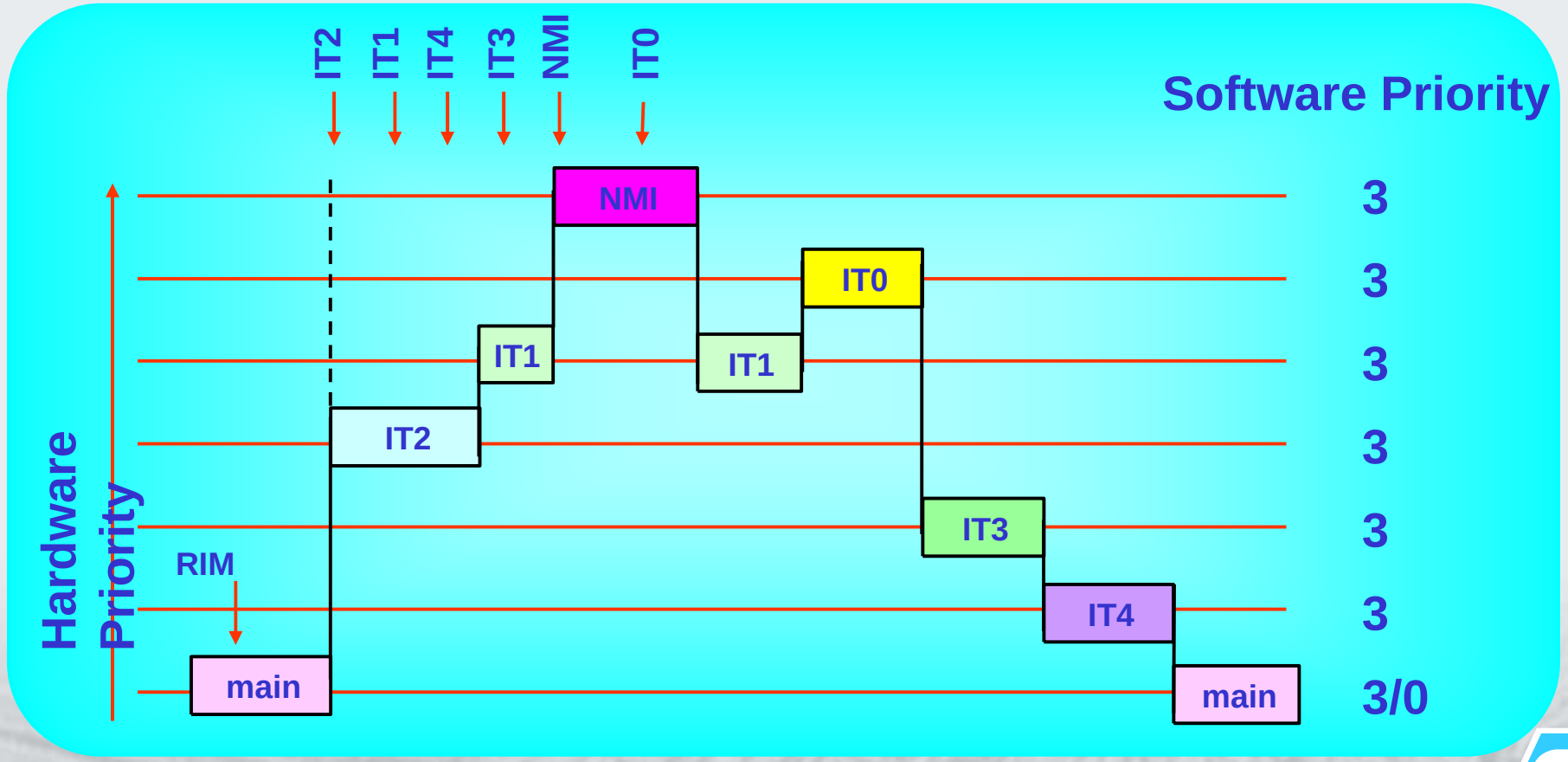
- Interrupt Vectors:
- Number:
- S/W Priority (Nested mode only):
- Interrupt Reaction Time:
- Automatic register pushed:

- up 16 Vectors
- 16 levels hardwired
- 4 levels user configurable
- 1.250µs to 2.750 µs (end of the current instruction +10 cpu cycles)
- Program counter, accumulator, CC, X

Software levels allows the ST7 nested interrupt process (availability of nested interrupt feature is device dependant)
At least one Interrupt Vector per Peripheral

Concurrent Interrupt Management

- An interrupt can not be interrupted by another one
- Except by the NMI (Non Maskable Interrupt) or TLI (Top level interrupt)



Nested Interrupt

- The 4 interrupt S/W levels are set thanks to the pair of bits I0, I1.

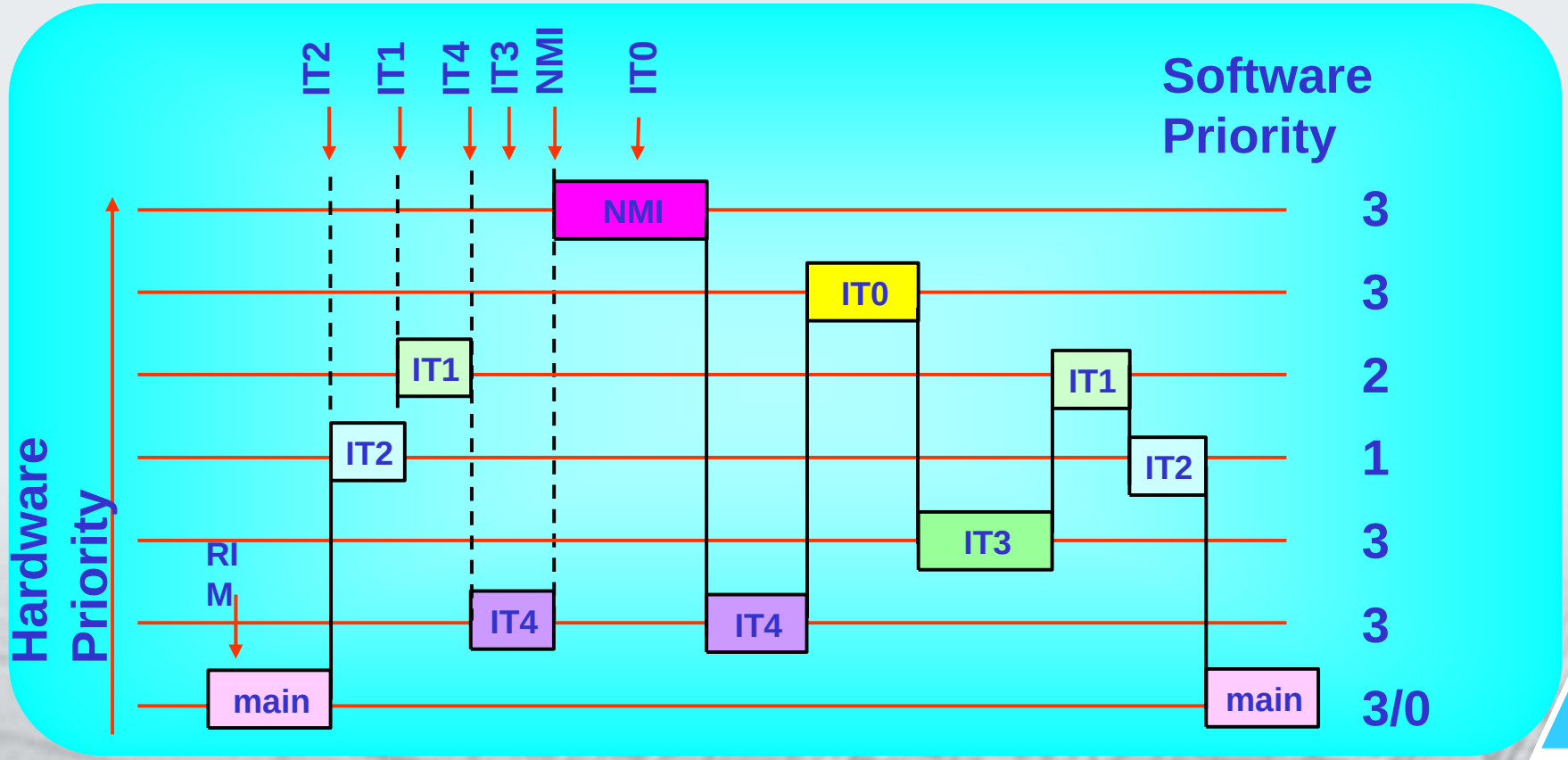
Interrupt Software Priority	Level	I1	I0
Level 0 (main)	Low	1	0
Level 1		0	1
Level 2		0	0
Level 3 (=interrupt disable)	High	1	1

to be compatible with previous bit I

- 1 pair of bits by interrupt vector stored in the ISPR registers
- The pair of bits is copied in the CC register when the corresponding IT is activated (software level greater than the current one).

Nested Interrupt Management

- An interrupt can be interrupted by:
  - The NMI (Non Maskable Interrupt) or TLI (Top Level Interrupt)
  - An interrupt request having an higher software Priority

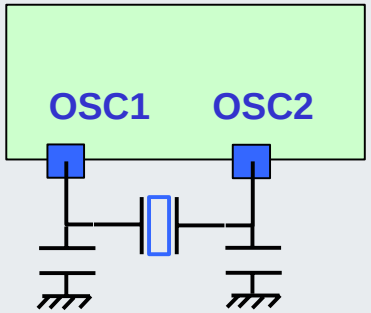


Interrupts Summary

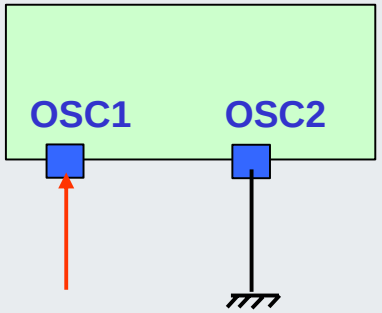
- **How many interrupt vectors can be used in the ST7 ?**
  - Up to 16, at least one per peripheral.
- **Are the software interrupt levels able to be modified during the application?**
  - Yes, in that case the interrupt routines priority can be modified during the program execution
- **What are the instructions that enable & disable the Interrupts?**
  - RIM: Reset Interrupt Mask (I bit =0, allows the interrupts)
  - SIM: Set interrupt Mask (I bit =1, disable the interrupts)

Multi oscillator (1)

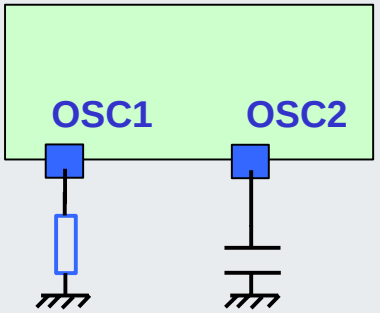
- Advanced ST7 clock system



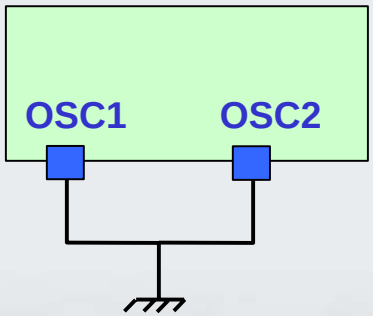
- QUARTZ/CERAMIC



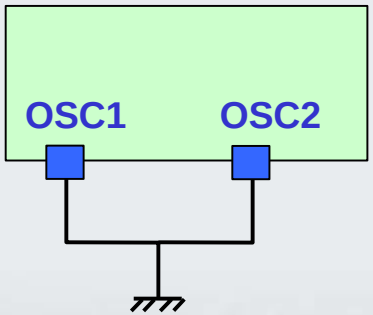
- EXTERNAL SOURCE



- EXTERNAL RC



- INTERNAL # 4Mhz



- 1% ACCURACY INTERNAL RC

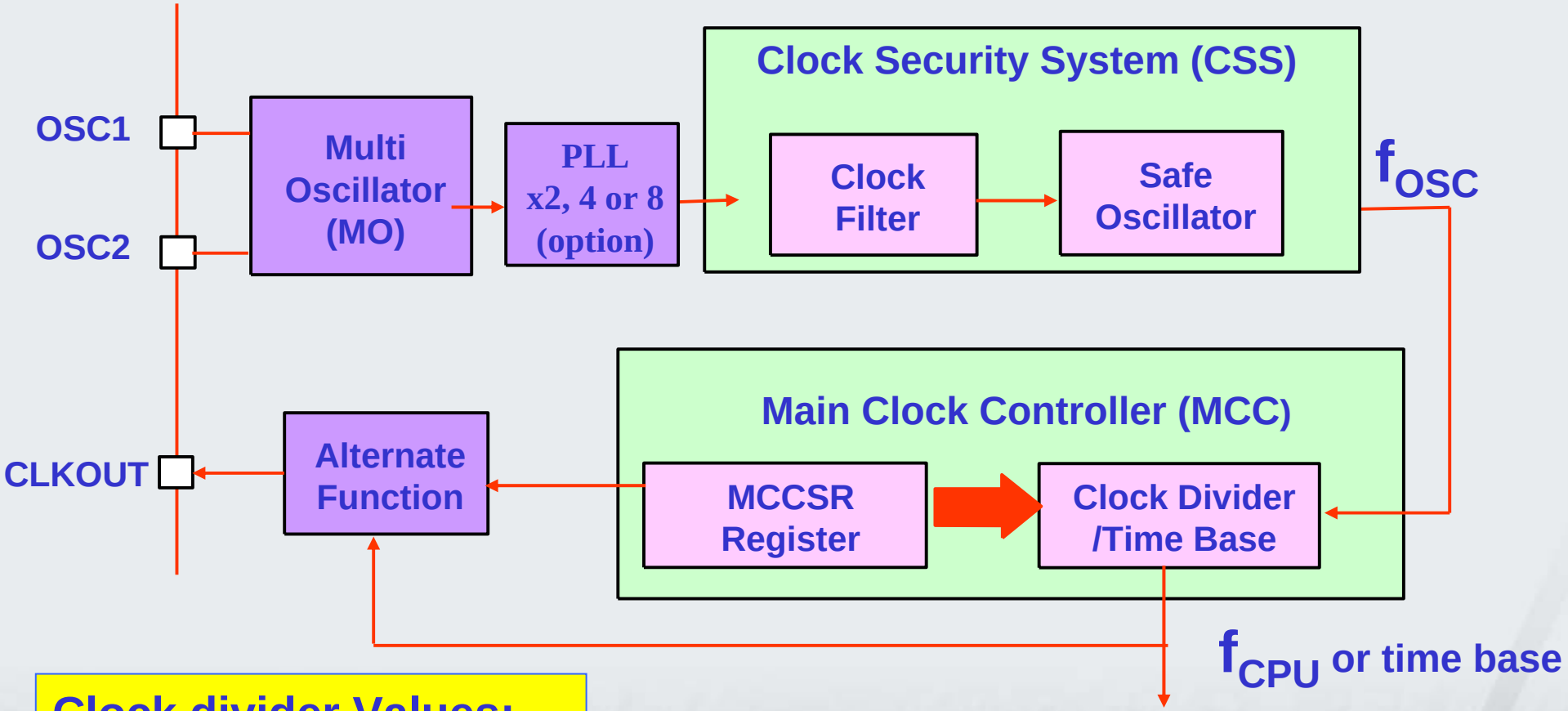
➢ **+ Low frequency backup safety oscillator (option byte)**

Multi Oscillator (2)

- 4 Crystal/Ceramic Oscillators
  - Designed to reduce EMI & consumption
    - ✓Low speed
    - ✓Mid low speed
    - ✓Mid high speed
    - ✓High speed
- Frequency range:
  - ✓1 to 2 MHz
  - ✓2 to 4 MHz
  - ✓4 to 8 MHZ
  - ✓8 to 16 MHz
- 1 External RC Oscillator
- 1 Internal RC Oscillator
- 1 Internal Safe Oscillator
- 1 to 14 MHz
- # 1 to 8 MHz
- # 250KHz and #3MHz for new products (Xflash and HDFlash)

Oscillator selected by option byte

Main Clock Controller

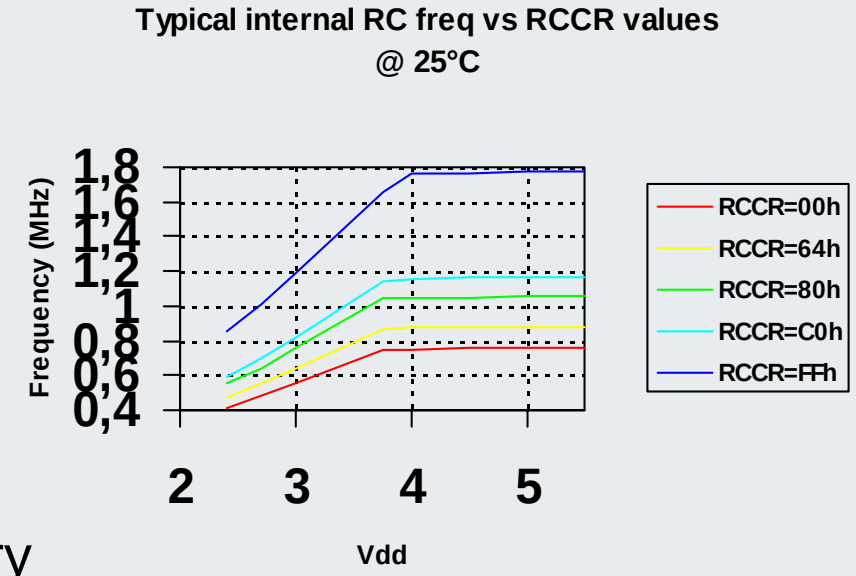


Clock divider Values:
2,4,8,16,32

Time Base Prescaler (Active Halt):
32000,64000,160000,400000

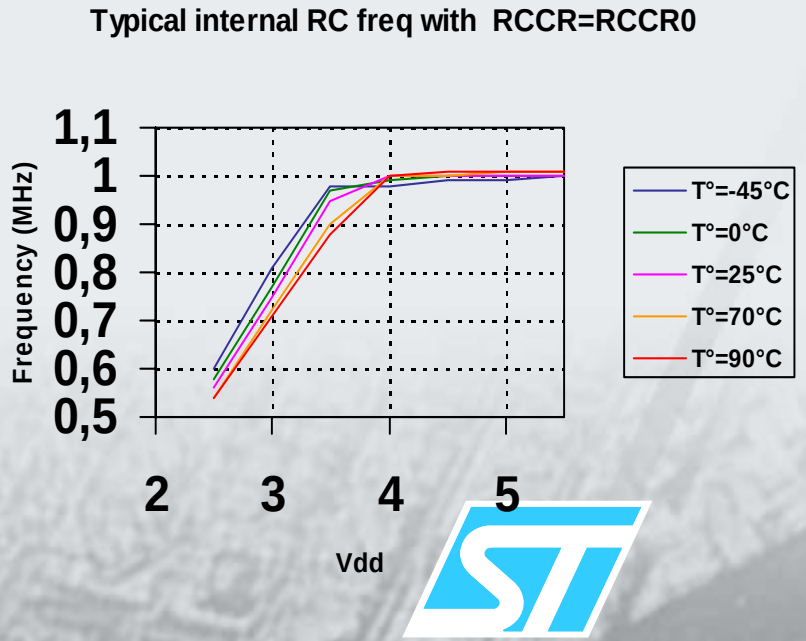
RC1% Architecture

- Calibration used for process, temperature and voltage derive compensation
- Expected frequency&accuracy obtained by SW tuning and by loading RCCR register after each reset
  - LD A, $1000h
  - LD RCCR, A
- Predefined RCCRx values stored in memory after calibration during ST test flow



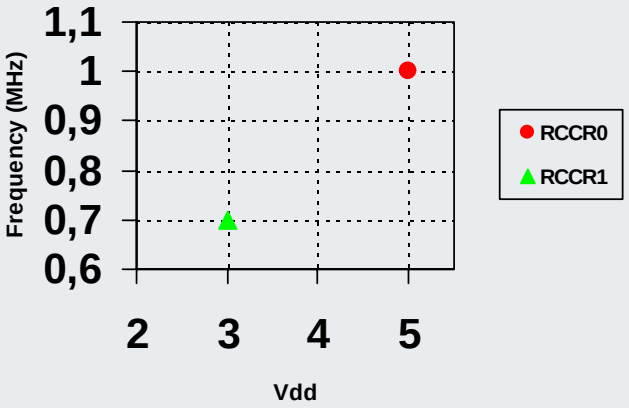
RC accuracy with RCCR=RCCR0 1MHZ 5V Range	Conditions	Min	Typ	Max
	T°=25°C, Vdd=5V	-0.5%	+/-0.2%	0.5%
	T°=25°C, Vdd=4.5V to 5.5V	-1%		+1%
	T°=-40°C to 85°C, Vdd=5V	-2%		+1%

RC accuracy with RCCR=RCCR1 700KHz or 1MHz 3V Range	Conditions	Min	Typ	Max
	T°=25°C, Vdd=3V	-2%	+/-0.5%	2%
	T°=25°C, Vdd=2.7V to 3.3V	-20%		+20%
	T°=-40°C to 85°C, Vdd=3V	-5%		+10%

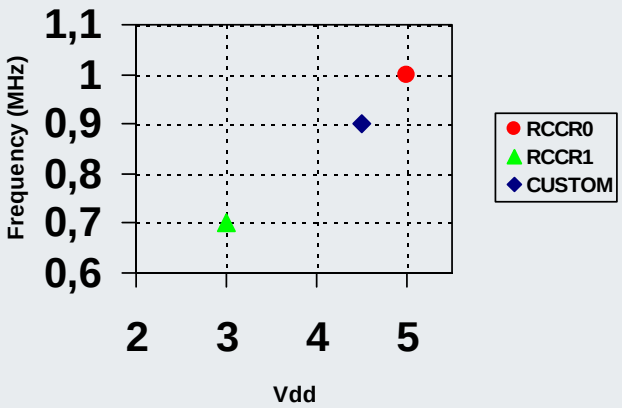


When can RC calibration data be stored in non volatile memory

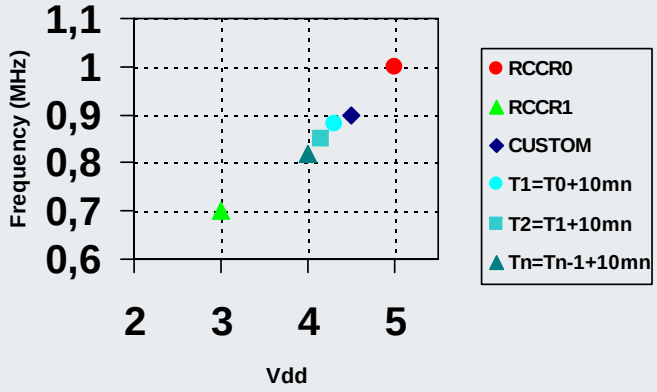
Predifined calibration values



Customer calibration value



Application exemple with new calibartion each 10 minutes



ST FACTORY
2 PREDIFINED CALIBRATION VALUES FOR EACH PART

- **CONDITIONS :**
  - RCCR0 : 1MHz @ 5V @ 25°C
  - RCCR1 :
    - ✓ 0.7MHz @ 3V @ 25°C (for Lite0 and 2)
    - ✓ 1MHz @ 3V @ 25°C (for Lite3, 1B, Ultra …)

CUSTOMER FACTORY
CALIBRATE AND DEDICATE TO THE APPLICATION

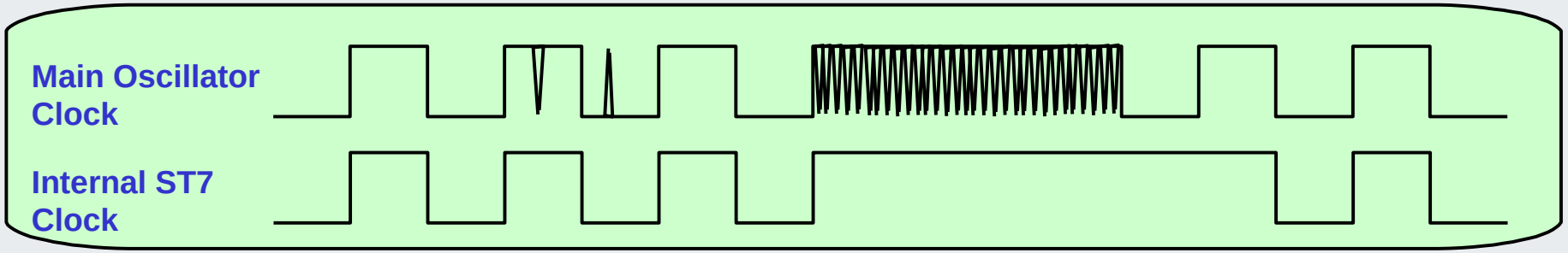
- **CONDITIONS : (OPTIONAL)**
  - FREQ @ VDD @ T° CUSTOMER CHOICE FOR BEST APPLICATION FITTING

IN APPLICATION
CALIBRATE FOR SPECIFIC CONDITIONS ADAPTATION

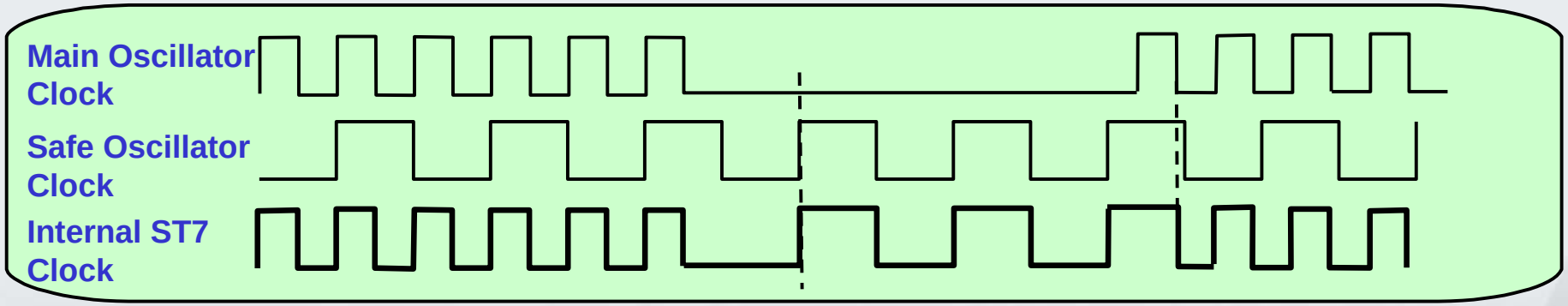
- CONDITIONS : (OPTIONAL)
  - FREQ @ VDD @ T° APPLICATION DEPENDANT
  - SOFTWARE ROUTINE CUSTOMIZED
  - REAL TIME BASE NEEDED (EX : AN1324, CALIBRATION WITH 50/60Hz MAINS)

Clock Security System

- Clock Filter Function



- Safe Oscillator



CSSD bit is set by H/W if one of the safety function is activated and can generate a maskable interrupt request

Clock Source Capability

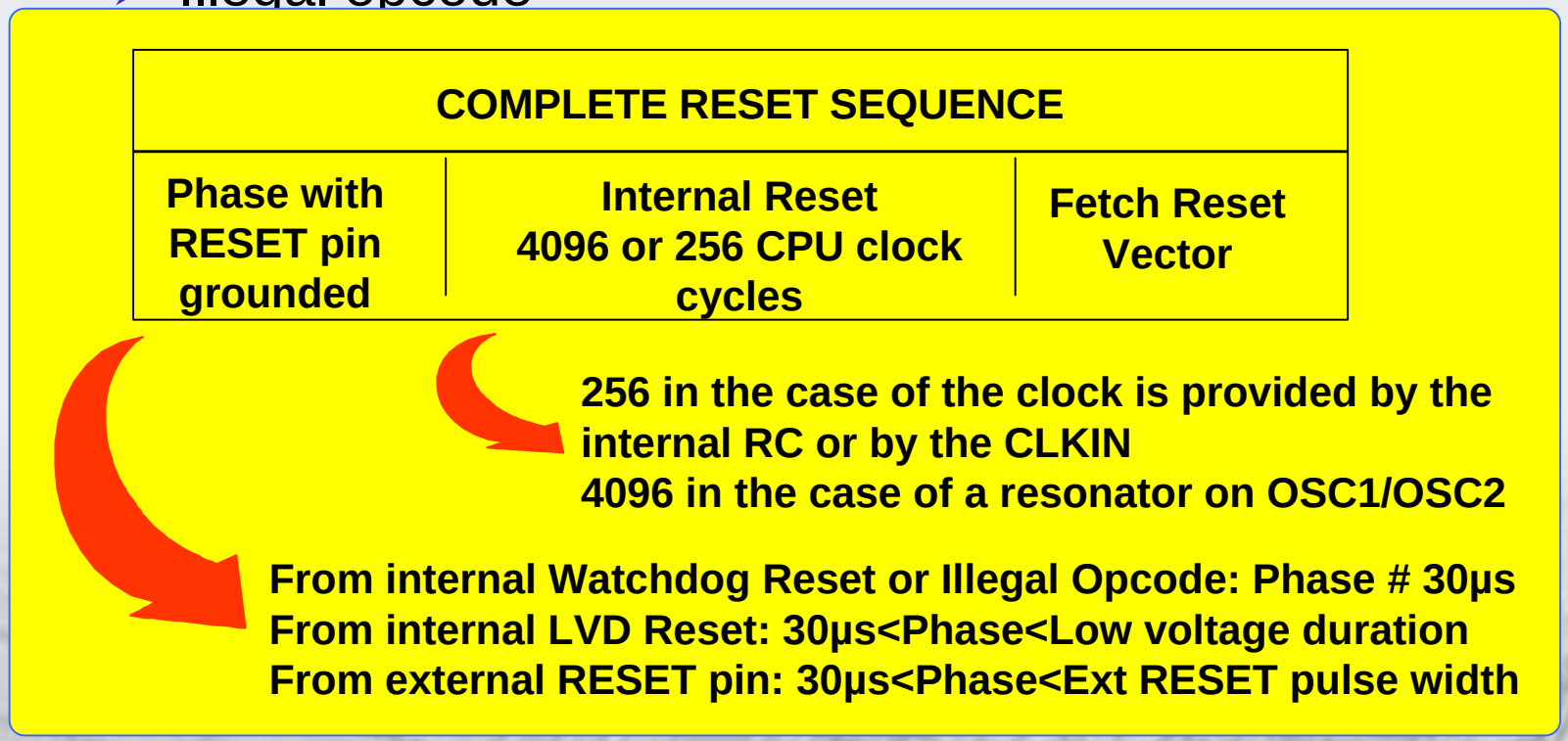
	FLITE0	FLITE2	F344	F324	F521	F561
QUARTZ/ CERAMIC		X	X	X	X	X
EXTERNAL CLOCK	X	X	X	X	X	X
EXTERNAL RC			X	X	X	X
INTERNAL RC				X	X	X
1% INTERNAL RC	X	X	X			
LOW POWER RC FOR AWU		X	X			X

ST7 CORE
Reset sources

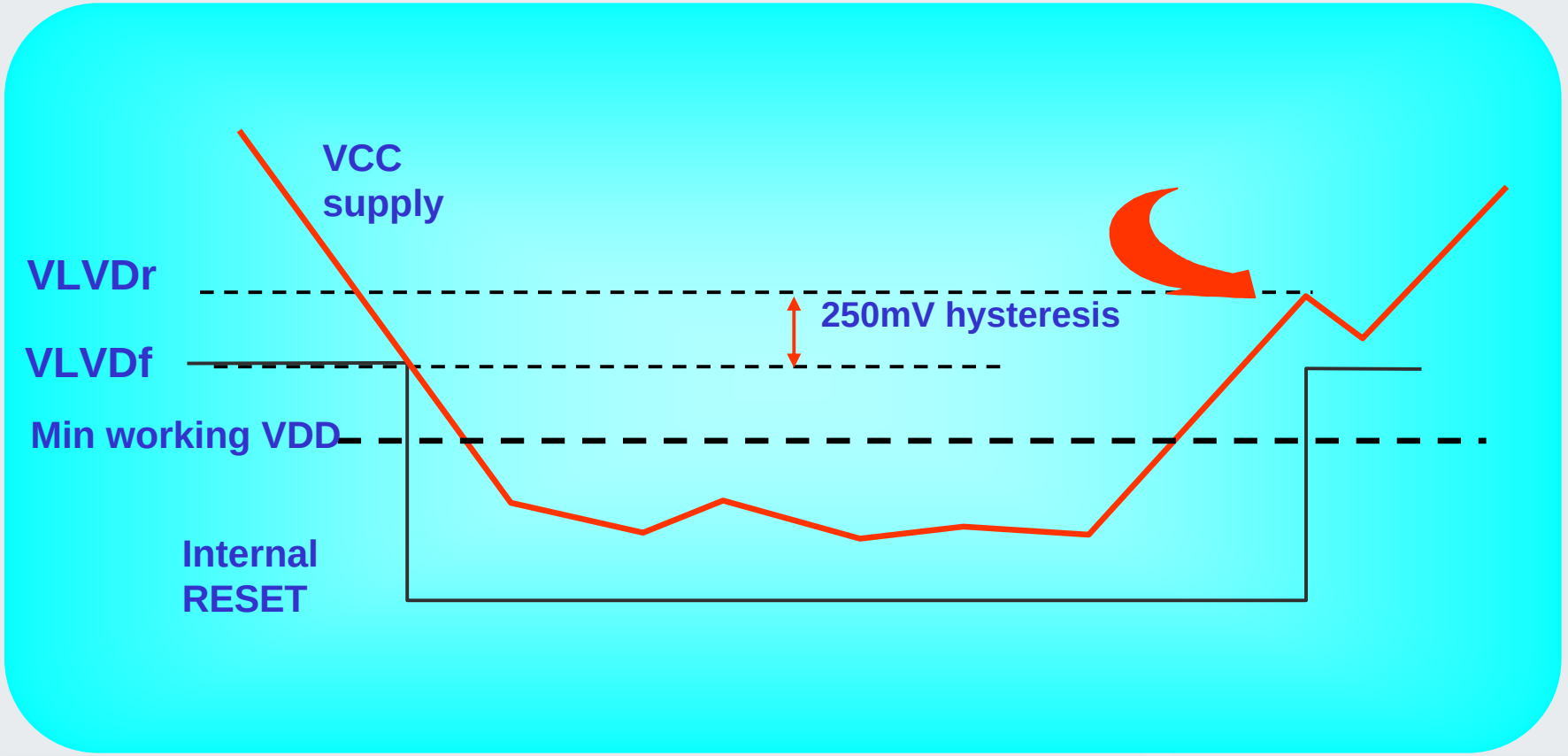
- External reset using reset pin
  - Purpose: allow to generate an external reset
  - Condition: reset pin pull low
- Power supply dependent reset using LVD
  - Purpose: ensure the MCU is in a known state whatever Vcc
  - Condition: internal reset when Vcc reaches Vcc min
- Watchdog reset using the watchdog timer
  - Purpose: guarantee the safety in case of software trouble
  - Condition: internal reset when the WD register is not refreshed
- Illegal opcode
  - Purpose: guarantee the safety in case of software trouble
  - Condition: internal reset when executing an undefined prebyte or opcode

ST7 ENHANCED RESET SYSTEM

- 4 Reset sources
  - Watchdog
  - Low Voltage Detection (LVD)
  - External RESET pin
  - Illegal opcode
- Complete reset management
  - Flags on Reset sources
  - Internal Reset externally issued to reset the whole application

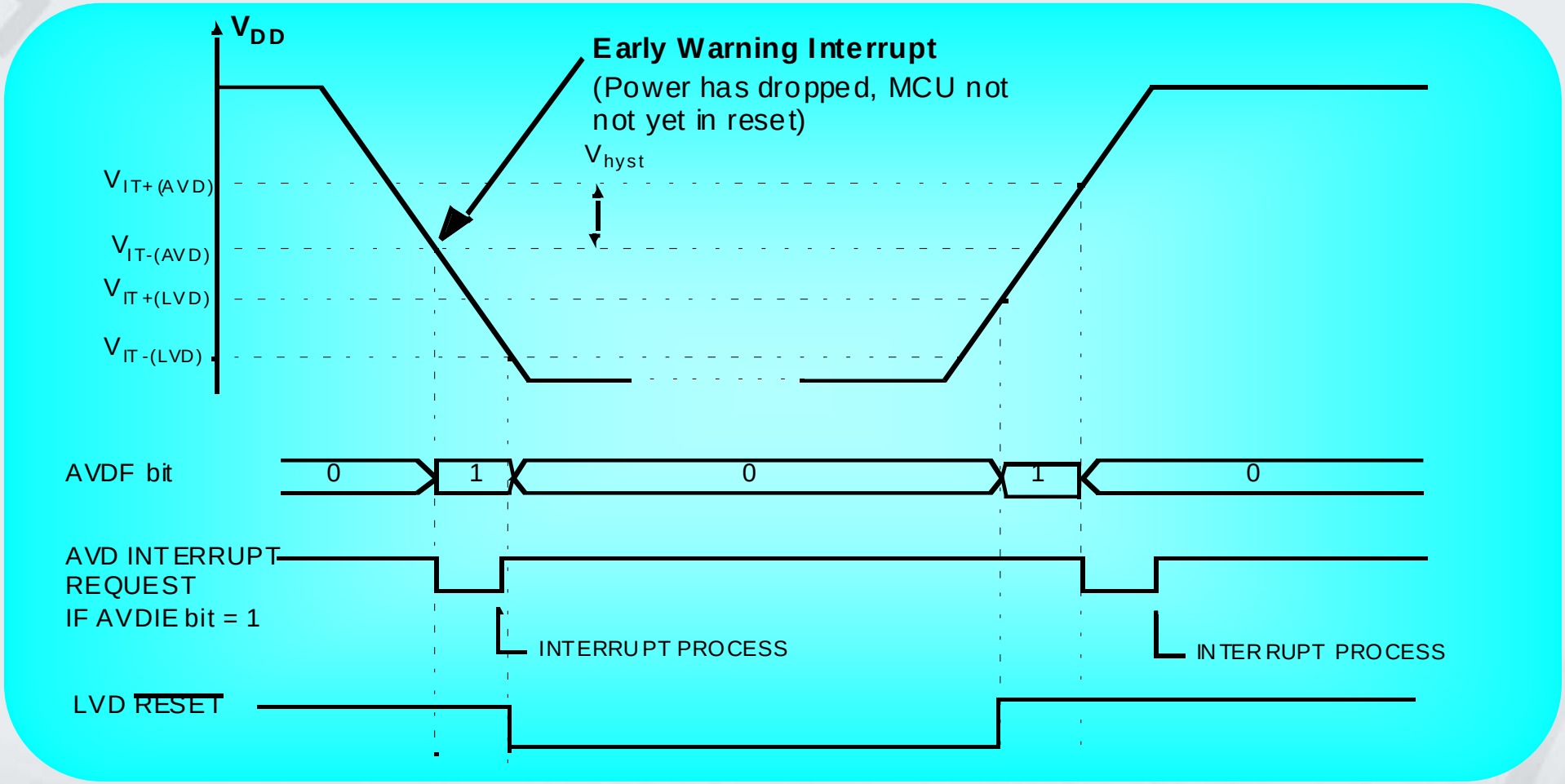


ST7 LVD GENERATION



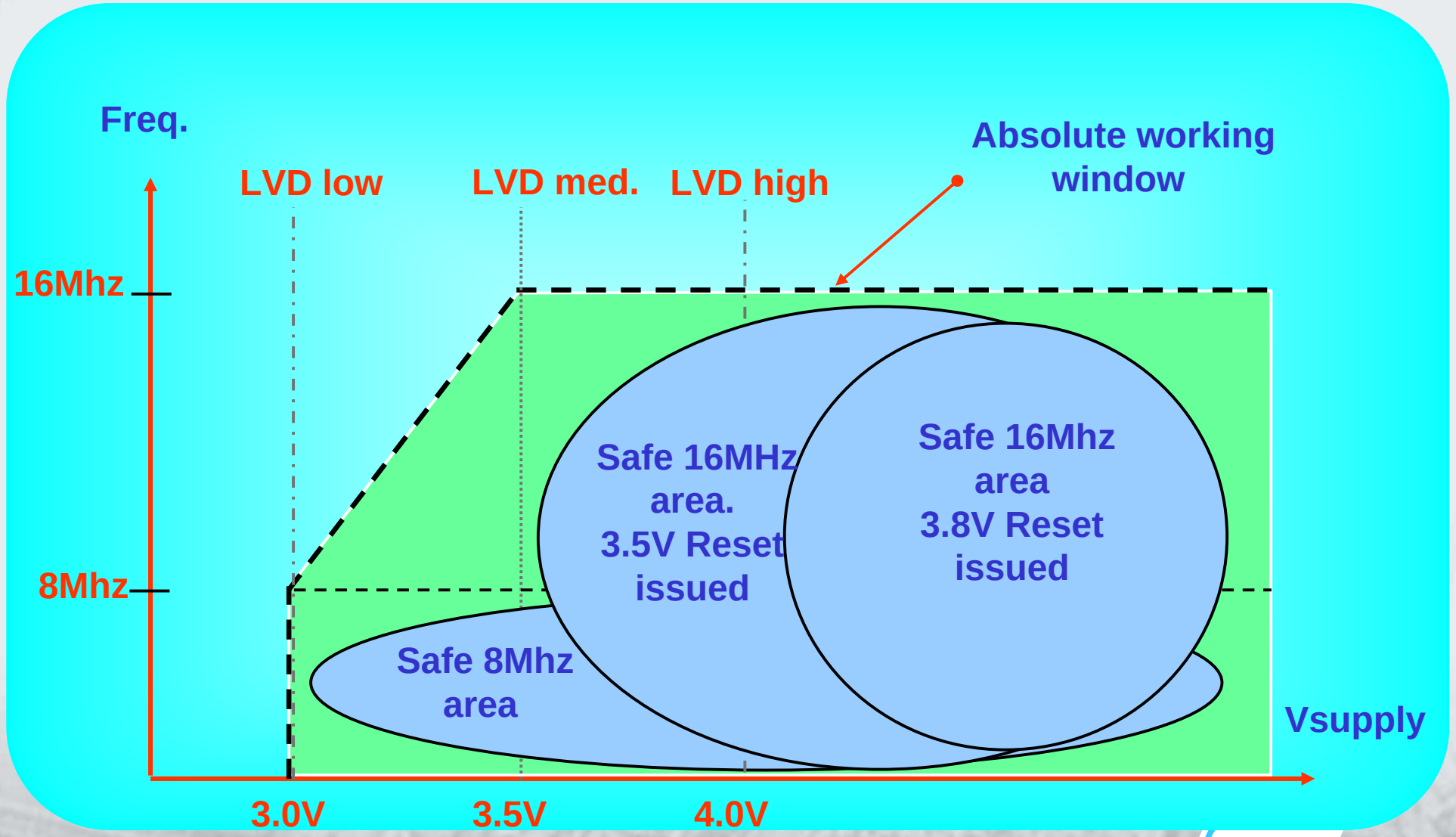
- 3 Selectable levels
- Activation Flag
- RESET pin tied to GND

ST7 LVD/AVD GENERATION



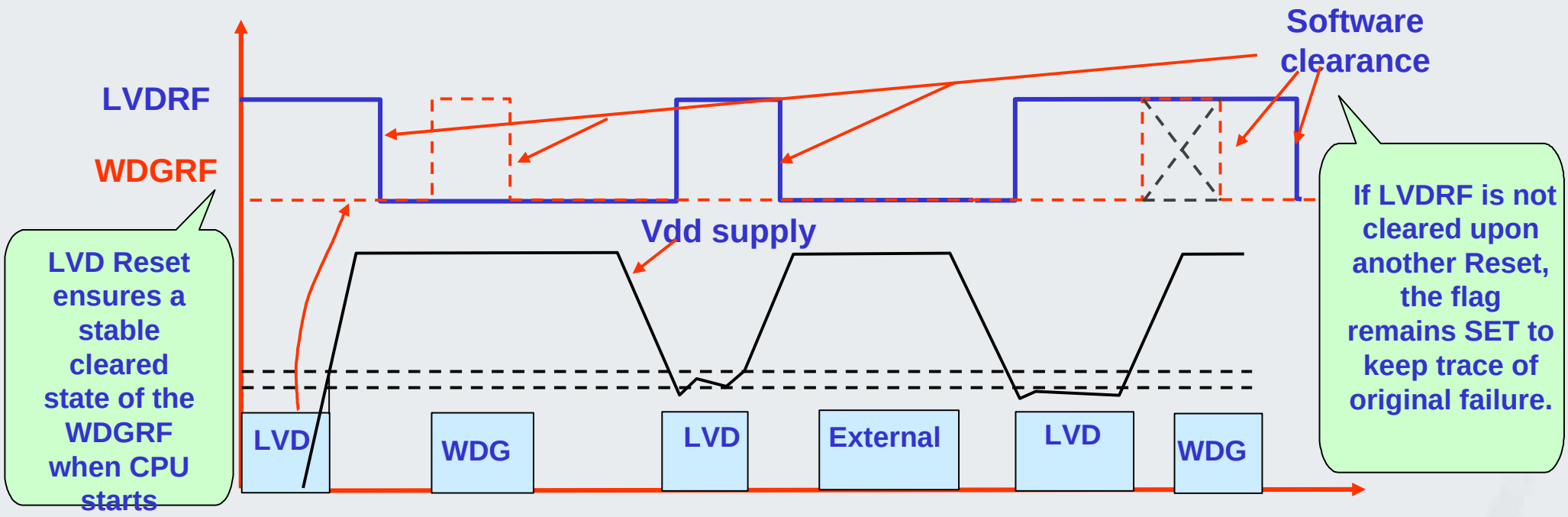
- **3 Selectable levels**
- **Activation Flag**
- **RESET pin tied to GND**

3 LVD LEVELS TO OPTIMIZE THE SAFE AREA



LVD/AVD levels are device dependant

LVD PART OF COMPLETE RESET MANAGEMENT



Original RESET source	LVDRF	WDGRF
External RESET pin	0	0
Watchdog Reset Flag	0	1
Low Voltage Reset Flag	1	X

ST7 WATCHDOG Overview (1)

- Its purpose is to detect the occurence of a software fault. It must be regularly refreshed by the program:
  - LD A,#$FF
  - LD WDGCR, A ; Reload WD

- 2 different watchdogs can be selected by option bit:
  - Hardware Watchdog:
    - ✓ WD automatically activated upon reset
  - Software Watchdog:
    - ✓ It is activated by software (bit 7 = 1). Once activated, it cannot be disabled

ST7 WATCHDOG
Overview (2)

- Reset and watchdog:
  - HALT instruction can generate a reset if watchdog is activated and if the option byte allows it
  - The Watchdog can be used to generate a software reset
    - ✓(bit 7 = 1, bit 6 = 0) WDGCR = $80 ; Reset !
- Difference between the Lite family and the other ST7
  - On Lite, the watchdog period is X*16000 Tcpu with X between 1 and 3Fh
  - On ST7, the watchdog period varies between :

Min : WDGCR = C0h
Δ = 2^0 × 12288 = 12288 clock cycles
= 1.54 ms for Fcpu = 8MHz
Max : WDGCR = FFh
Δ = 2^6 × 12288 = 786432 clock cycles
= 98.30 ms for Fcpu = 8MHz
For XFlash and HDFlash:
Special formula are given in the datasheets

Clock & Reset system Summary

- **What are the maximun oscillator frequency and the maximun CPU frequency ?**
  - f_{OSC} = 16MHz & fCPU = 8 MHZ in run mode.
- **How many internal RC oscillator are implemented on ST7 ?**
  - 2, the first one running at 1 to 8 MHZ and the backup safety oscillator.
- **What are the internal reset sources?**
  - LVD, Watchdog, Illegal opcode.
- **Can these internal resets be detected externally?**
  - Yes, thanks to the reset pin that acts as an input/output pin.

ST7 CLOCK IN LOW POWER MODES

- **RUN MODE**: Fcpu=Fosc/2
  - Core & periph. running except if WAIT (Core stopped) selected
- **SLOW MODE**: Division ratio from 4 to 32 by software
  - Core & periph. running except if WAIT (Core stopped) selected
- **ACTIVE HALT**: Division ratio from 32000 to 400000 by software
  - Core & periph. stopped but periodic wake-up through interrupts
- **AUTO WAKE UP** : Based on a low power internal RC allowing a periodic wake up
- **HALT MODE**: Oscillator stopped
  - Core & periph. stopped

ST7 LOW CONSUMPTION MODES Overview

- Typical ST7 consumption:
  - HDFlash : 7 mA with f_{osc} = 16 MHZ and VDD = 5V
  - XFlash : 5 mA with f_{osc} = 16 MHZ and VDD = 5V
- To reach lowest power consumption
  - Switch off unused peripherals
  - Configure I/Os as output low level and connect them to GND
  - Use the lowest oscillator frequency possible
  - Use the Slow mode, Wait mode or better the Halt mode

ST7 LOW CONSUMPTION MODES
Slow mode

- Goal: reduce the consumption by reducing the clock speed keeping the same Oscillator frequency
- Enter by: configuring the miscellaneous register
- Causes: the CPU clock slows down
  - The f_{osc} can be divided by 4, 8 ,16 or 32 rather than 2, or only 32 for small products
- Exit by: configuring the miscellaneous register

ST7 LOW CONSUMPTION MODES
Wait mode

- Goal: Reduce the consumption while monitoring external events

- Enter by: execution of the Wait For Interrupt instruction (WFI)
- Causes: the micro is software frozen
  - Program execution stopped
  - Memory and registers remain unchanged
  - The oscillator still provides a clock to the peripherals

- Exit by
  - Reset (Watchdog, reset pin)
  - Internal interrupts (timer A, timer B, A/D, SPI etc)
  - External interrupts (I/O ports)

ST7 LOW CONSUMPTION MODES
Active Halt mode

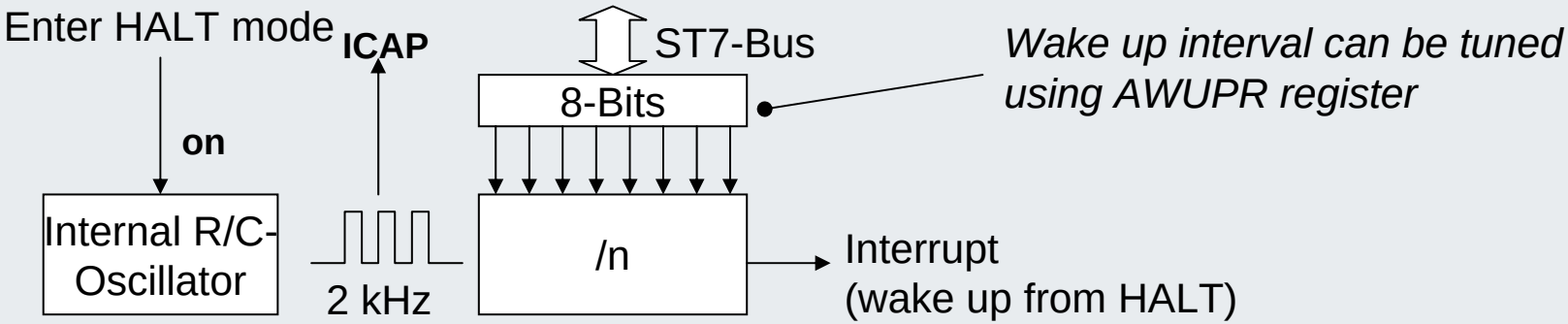
- Goal: Reduce the consumption to the lowest value while monitoring a real time clock.
- Enter by: execution of the HALT instruction while the OIE (Oscillator Interrupt Enable) bit is set.
- Causes: the micro is SW frozen, all the peripherals are stopped, only the Oscillator & the Main Oscillator Counter are running.
- Exit by
  - External Reset
  - Interrupts with exit from halt capability (External IT,..)
  - Time Base Interrupt (32000,64000,160000,400000 *T_{CPU})
  - From 2ms to 25ms with f_{OSC}=16MHZ

Auto wake up mode

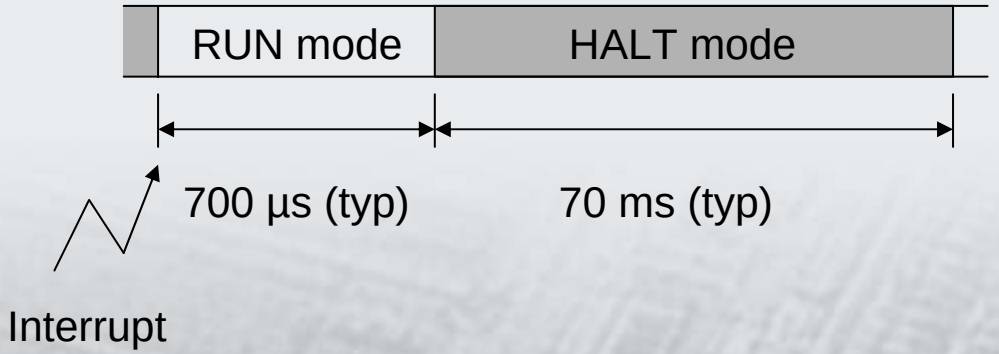
i.e. cyclic wakeup every 70 ms with total conso < 50µA

- Basic Principle

  Based on internal RC-oscillator (low power)



- Power Consumption



- **Low system power consumption**
  <50µA in total in this mode (application dependent)
- **Fast startup after 256 cycles possible** around 30µs @ 8MHz internal

ST7 LOW CONSUMPTION MODES
Halt mode

- Goal: Reduce the consumption to the lowest value
- Enter by: execution of the HALT instruction
- Causes: the micro is SW and HW frozen
  - Program execution stopped
  - Memory and registers remain unchanged
  - The oscillator stopped
- Exit by
  - External Reset
  - External interrupts (I/O ports)

TYPICAL CONSUMPTIONS

- During wait mode or HALT MODE, BIT I (INTERRUPT BIT) of CC Register is automatically reset to enable interrupt

- Typical consumptions at Vdd=5V, Fcpu=8MHz:
  Auto wake up ~30µA for the ST72F561,FLITE2
  Active Halt ~50µA + oscillator consumption

	ST7FLITE0	ST72F264	ST72F324	ST72F521
RUN	4.5mA	7.2mA	7.1mA	7.1mA
SLOW (/32)	0.75mA	0.7mA	1.1mA	1.1mA
WAIT	1.75mA	3.6mA	3.5mA	4.5mA
HALT	<1µA	<1µA	<1µA	<1µA

PROGRAMMING TIPS
Low Consumption Modes

- After exiting from halt mode or wait mode after reset, the micro waits 256 or 4096 CPU clock cycle (STABILIZATION TIME) before being operational (clock source dependant)

- Source that allows to exit from WFI Or halt mode
  - Internal Interrupts => Wait and Halt modes
  - External Interrupts => Halt mode

- During wait mode or HALT MODE, BIT I (INTERRUPT BIT) of CC Register is automatically reset to enable interrupt