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
- Clock Controller
- Memory Space
- Watchdog
- ST7 Stack
- Interrupts
- Reset System LVD
- Low Power Modes
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
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.

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

---

![Stack Manipulation Diagram]

- **Push Y**
  - Call subroutines or interrupt
- **Pop Y**
  - Return from subroutines or interrupt
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:
  - 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

HDFlash program memory (emulated data E²PROM)

XFlash program memory (data E²PROM 128 or 256 bytes)
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.
Boot-ROM allows an executable software to be downloaded in RAM through ISPCLK & ISPDATA.

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

In-Situ Programming uses 6 wires only.
ST7 IN-CIRCUIT PROGRAMMING (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 \( f_{\text{CPU}}=8\text{MHz} \)

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

<table>
<thead>
<tr>
<th>Size</th>
<th>Prog. by 32byte blocks + Verify</th>
</tr>
</thead>
<tbody>
<tr>
<td>1kbyte</td>
<td>0.16s</td>
</tr>
<tr>
<td>4kbyte</td>
<td>0.64s</td>
</tr>
<tr>
<td>8kbyte</td>
<td>1.28s</td>
</tr>
<tr>
<td>16kbyte</td>
<td>2.56s</td>
</tr>
</tbody>
</table>

Communication timings are proportionnal with \( f_{\text{CPU}} \)
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
- 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)

<table>
<thead>
<tr>
<th></th>
<th>Prog. by 32byte blocks + Verify</th>
<th>Erase + Prog + Verify</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bit Map</td>
<td>Checksum</td>
</tr>
<tr>
<td>16kbyte</td>
<td>2.8s</td>
<td>1.7s</td>
</tr>
<tr>
<td>32kbyte</td>
<td>5.6s</td>
<td>3.4s</td>
</tr>
<tr>
<td>60kbyte</td>
<td>10.2s</td>
<td>6.3s</td>
</tr>
</tbody>
</table>

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 resources
- Bidirectional communication 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
  - Piracy
  - Unexpected reprogramming
  - Unexpected trouble occurs during programming

- Protection strategy
  - READOUT/W protect on Xflash
  - RASS (keys) (Register Access security System)
  - 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

<table>
<thead>
<tr>
<th>CPU INTERRUPT</th>
<th>INTERRUPTS</th>
<th>REGISTER</th>
<th>FLAG NAME</th>
<th>INTERRUPT SOURCE</th>
<th>VECTOR ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset</td>
<td>Reset</td>
<td>N/A</td>
<td>N/A</td>
<td>-</td>
<td>FFFEh-FFFFh</td>
</tr>
<tr>
<td>Trap (instruction)</td>
<td>Software</td>
<td>N/A</td>
<td>N/A</td>
<td>I0</td>
<td>FFFCh-FFFFDh</td>
</tr>
<tr>
<td>External Interrupt 0</td>
<td>Port A</td>
<td>N/A</td>
<td>N/A</td>
<td>I1</td>
<td>FFFAh-FFFFBh</td>
</tr>
<tr>
<td>External Interrupt 1</td>
<td>Port B and Port C</td>
<td>N/A</td>
<td>N/A</td>
<td>I2</td>
<td>FFF8h-FFF9h</td>
</tr>
<tr>
<td>CSS</td>
<td>Clock Filter Interrupt</td>
<td>CRSR</td>
<td>CCSD</td>
<td>I3</td>
<td>FFF6h-FFF7h</td>
</tr>
<tr>
<td>SPI</td>
<td>Transfer Complete Mode</td>
<td>SPI Status</td>
<td>SPIF</td>
<td>I4</td>
<td>FFF4h-FFF5h</td>
</tr>
<tr>
<td>Timer A</td>
<td>Input Capture 1</td>
<td>Timer A Status</td>
<td>ICF1</td>
<td>I5</td>
<td>FFF2h-FFF3h</td>
</tr>
<tr>
<td>Timer B</td>
<td>Input Capture 1</td>
<td>Timer B Status</td>
<td>ICF1</td>
<td>I7</td>
<td>FFEEh-FFEFh</td>
</tr>
<tr>
<td>I2C</td>
<td>Byte Transfer finished</td>
<td>I2C Status</td>
<td>BTF</td>
<td>I12</td>
<td>FFE4h-FFE5h</td>
</tr>
</tbody>
</table>
ST7 INTERRUPTS
Peripheral Int management

Periph Status Register

Periph Control Register

Condition Code Register

Interrupt flag set by H/W
Interrupt Enable bit set by S/W
Interrupt Mask bit reset/set by S/W

Context switch takes 10 CPU clock cycles

Interrupt generation
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.

  ➢ .IntRoutine

  ...
  BRES Status_reg, #IT_flag ; Avoid to process the
  ; same interrupt forever
  IRET ; Return from interrupt
  ...
Software levels allows the ST7 nested interrupt process (availability of nested interrupt feature is device dependant) At least one Interrupt Vector per Peripheral

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
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.

<table>
<thead>
<tr>
<th>Interrupt Software Priority</th>
<th>Level</th>
<th>I1</th>
<th>I0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 0 (main)</td>
<td>Low</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Level 1</td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Level 2</td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Level 3 (=interrupt disable)</td>
<td>High</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- 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 a 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

- 1 External RC Oscillator
- 1 Internal RC Oscillator
- 1 Internal Safe Oscillator

- Frequency range:
  - 1 to 2 MHz
  - 2 to 4 MHz
  - 4 to 8 MHz
  - 8 to 16 MHz

- 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

### Typical internal RC freq vs RCCR values @ 25°C

<table>
<thead>
<tr>
<th>RCCCR</th>
<th>Frequency (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00h</td>
<td>0.4</td>
</tr>
<tr>
<td>64h</td>
<td>0.6</td>
</tr>
<tr>
<td>80h</td>
<td>0.8</td>
</tr>
<tr>
<td>C0h</td>
<td>1.0</td>
</tr>
<tr>
<td>FFh</td>
<td>1.2</td>
</tr>
</tbody>
</table>

### MaxTypMin Conditions RC accuracy with RCCCR=RCCR0
- 1MHz
- 5V Range

<table>
<thead>
<tr>
<th>Conditions</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>T°=25°C, Vdd=5V</td>
<td>-0.5%</td>
<td>+/-0.2%</td>
<td>0.5%</td>
</tr>
<tr>
<td>T°=25°C, Vdd=4.5 to 5.5V</td>
<td>-1%</td>
<td>+1%</td>
<td></td>
</tr>
<tr>
<td>T°=-40°C to 85°C, Vdd=5V</td>
<td>-2%</td>
<td>+1%</td>
<td></td>
</tr>
</tbody>
</table>

### MaxTypMin Conditions RC accuracy with RCCCR=RCCR1
- 700KHz or 1MHz
- 3V Range

<table>
<thead>
<tr>
<th>Conditions</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>T°=25°C, Vdd=3V</td>
<td>-2%</td>
<td>+/-0.5%</td>
<td>2%</td>
</tr>
<tr>
<td>T°=25°C, Vdd=2.7 to 3.3V</td>
<td>-20%</td>
<td>+20%</td>
<td></td>
</tr>
<tr>
<td>T°=-40°C to 85°C, Vdd=3V</td>
<td>-5%</td>
<td>+10%</td>
<td></td>
</tr>
</tbody>
</table>
When can RC calibration data be stored in non volatile memory

**Predicted calibration values**

<table>
<thead>
<tr>
<th>Vdd</th>
<th>Frequency (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,6</td>
<td></td>
</tr>
<tr>
<td>0,7</td>
<td></td>
</tr>
<tr>
<td>0,8</td>
<td></td>
</tr>
<tr>
<td>0,9</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

**Customer calibration value**

<table>
<thead>
<tr>
<th>Vdd</th>
<th>Frequency (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,6</td>
<td></td>
</tr>
<tr>
<td>0,7</td>
<td></td>
</tr>
<tr>
<td>0,8</td>
<td></td>
</tr>
<tr>
<td>0,9</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

**Application example with new calibration each 10 minutes**

<table>
<thead>
<tr>
<th>Vdd</th>
<th>Frequency (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,6</td>
<td></td>
</tr>
<tr>
<td>0,7</td>
<td></td>
</tr>
<tr>
<td>0,8</td>
<td></td>
</tr>
<tr>
<td>0,9</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

**ST FACTORY**

2 predicted calibration values for each part

**Customer Factory**

Calibrate and dedicate to the application

**In application**

Calibrate for specific conditions adaptation

- **Conditions**: (optional)
  - Freq @ Vdd @ T° application dependent
  - Software routine customized
  - Real time base needed (ex: AN1324, calibration with 50/60Hz mains)

- **Conditions**: (optional)
  - Freq @ Vdd @ T° application dependent
  - Software routine customized
  - Real time base needed (ex: AN1324, calibration with 50/60Hz mains)

- **Conditions**: (optional)
  - Freq @ Vdd @ T° application dependent
  - Software routine customized
  - Real time base needed (ex: AN1324, calibration with 50/60Hz mains)
Clock Security System

• Clock Filter Function

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

• Safe Oscillator
## Clock Source Capability

<table>
<thead>
<tr>
<th></th>
<th>FLITE0</th>
<th>FLITE2</th>
<th>F344</th>
<th>F324</th>
<th>F521</th>
<th>F561</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>QUARTZ/CERAMIC</strong></td>
<td>🍁</td>
<td>🍁</td>
<td>🍁</td>
<td>🍁</td>
<td>🍁</td>
<td>🍁</td>
</tr>
<tr>
<td><strong>EXTERNAL CLOCK</strong></td>
<td>🍁</td>
<td>🍁</td>
<td>🍁</td>
<td>🍁</td>
<td>🍁</td>
<td>🍁</td>
</tr>
<tr>
<td><strong>EXTERNAL RC</strong></td>
<td>🍁</td>
<td>🍁</td>
<td>🍁</td>
<td>🍁</td>
<td>🍁</td>
<td>🍁</td>
</tr>
<tr>
<td><strong>INTERNAL RC</strong></td>
<td>🍁</td>
<td>🍁</td>
<td>🍁</td>
<td>🍁</td>
<td>🍁</td>
<td>🍁</td>
</tr>
<tr>
<td><strong>1% INTERNAL RC</strong></td>
<td>🍁</td>
<td>🍁</td>
<td>🍁</td>
<td>🍁</td>
<td>🍁</td>
<td>🍁</td>
</tr>
<tr>
<td><strong>LOW POWER RC FOR AWU</strong></td>
<td>🍁</td>
<td>🍁</td>
<td></td>
<td></td>
<td>🍁</td>
<td></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>COMPLETE RESET SEQUENCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase with RESET pin grounded</td>
</tr>
</tbody>
</table>

- 256 in the case of the clock is provided by the internal RC or by the CLKIN 4096 in the case of a resonator on OSC1/OSC2

- From internal Watchdog Reset or Illegal Opcode: Phase # 30µs
- From internal LVD Reset: 30µs<Phase<Low voltage duration
- From external RESET pin: 30µs<Phase<Ext RESET pulse width
ST7 LVD GENERATION

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

VCC supply

VLVDf

VLVDr

Min working VDD

Internal RESET

250mV hysteresis
ST7 LVD/AVD GENERATION

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

Early Warning Interrupt (Power has dropped, MCU not yet in reset)

\[ V_{hyst} \]

\[ V_{IT+ (AVD)} \]
\[ V_{IT- (AVD)} \]
\[ V_{IT+ (LVD)} \]
\[ V_{IT- (LVD)} \]

AVDF bit

AVD INTERRUPT REQUEST
IF AVDIE bit = 1

LVD RESET

INTERUPT PROCESS

INTERUPT PROCESS
3 LVD LEVELS TO OPTIMIZE THE SAFE AREA

LVD/AVD levels are device dependant
LVD PART OF COMPLETE RESET MANAGEMENT

LVD Reset ensures a stable cleared state of the WDGRF when CPU starts.

If LVDRF is not cleared upon another Reset, the flag remains SET to keep trace of original failure.

<table>
<thead>
<tr>
<th>Original RESET source</th>
<th>LVDRF</th>
<th>WDGRF</th>
</tr>
</thead>
<tbody>
<tr>
<td>External RESET pin</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Watchdog Reset Flag</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Low Voltage Reset Flag</td>
<td>1</td>
<td>X</td>
</tr>
</tbody>
</table>

Software clearance

Vdd supply
ST7 WATCHDOG
Overview (1)

• Its purpose is to detect the occurrence 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
    \[ \Delta = 2^0 \times 12288 = 12288 \text{ clock cycles} \]
    = 1.54 ms for Fcpu = 8MHz
    Max : WDGCR = FFh
    \[ \Delta = 2^6 \times 12288 = 786432 \text{ clock cycles} \]
    = 98.30 ms for Fcpu = 8MHz

For XFlash and HDFlash:
Special formula are given in the datasheets
Clock & Reset system
Summary

• What are the maximum oscillator frequency and the maximum CPU frequency?
  \[ f_{\text{OSC}} = 16 \text{MHz} \] & \[ f_{\text{CPU}} = 8 \text{ MHZ} \] in run mode.

• How many internal RC oscillator are implemented on ST7?
  \[ 2, \text{ the first one running at 1 to 8 MHZ and the backup safety oscillator.} \]

• What are the internal reset sources?
  \[ \text{LVD, Watchdog, Illegal opcode.} \]

• Can these internal resets be detected externally?
  \[ \text{Yes, thanks to the reset pin that acts as an input/output pin.} \]
ST7 CLOCK IN LOW POWER MODES

• **RUN MODE**: F_{cpu}=F_{osc}/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 $f_{\text{osc}}$ 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)

- Enter HALT mode
- 8-Bits Interrupt (wake up from HALT)
- Wake up interval can be tuned using AWUPR register

- Internal R/C-Oscillator
- 2 kHz
- 700 µs (typ)
- 70 ms (typ)

• 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

<table>
<thead>
<tr>
<th></th>
<th>ST7FLITE0</th>
<th>ST72F264</th>
<th>ST72F324</th>
<th>ST72F521</th>
</tr>
</thead>
<tbody>
<tr>
<td>RUN</td>
<td>4.5mA</td>
<td>7.2mA</td>
<td>7.1mA</td>
<td>7.1mA</td>
</tr>
<tr>
<td>SLOW (/32)</td>
<td>0.75mA</td>
<td>0.7mA</td>
<td>1.1mA</td>
<td>1.1mA</td>
</tr>
<tr>
<td>WAIT</td>
<td>1.75mA</td>
<td>3.6mA</td>
<td>3.5mA</td>
<td>4.5mA</td>
</tr>
<tr>
<td>HALT</td>
<td>&lt;1µA</td>
<td>&lt;1µA</td>
<td>&lt;1µA</td>
<td>&lt;1µA</td>
</tr>
</tbody>
</table>
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