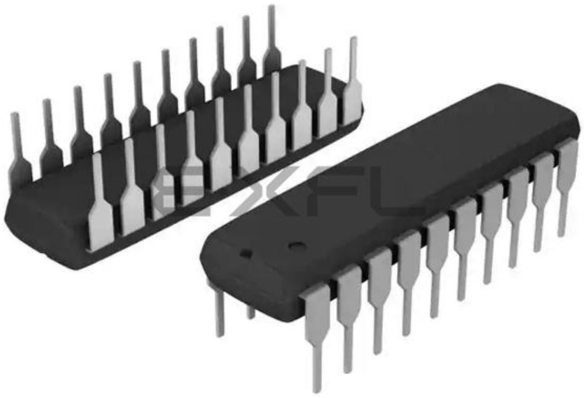
# STMicroelectronics - ST7FLITE20F1B6 Datasheet



**Welcome to E-XFL.COM**

## What is "Embedded - Microcontrollers"?

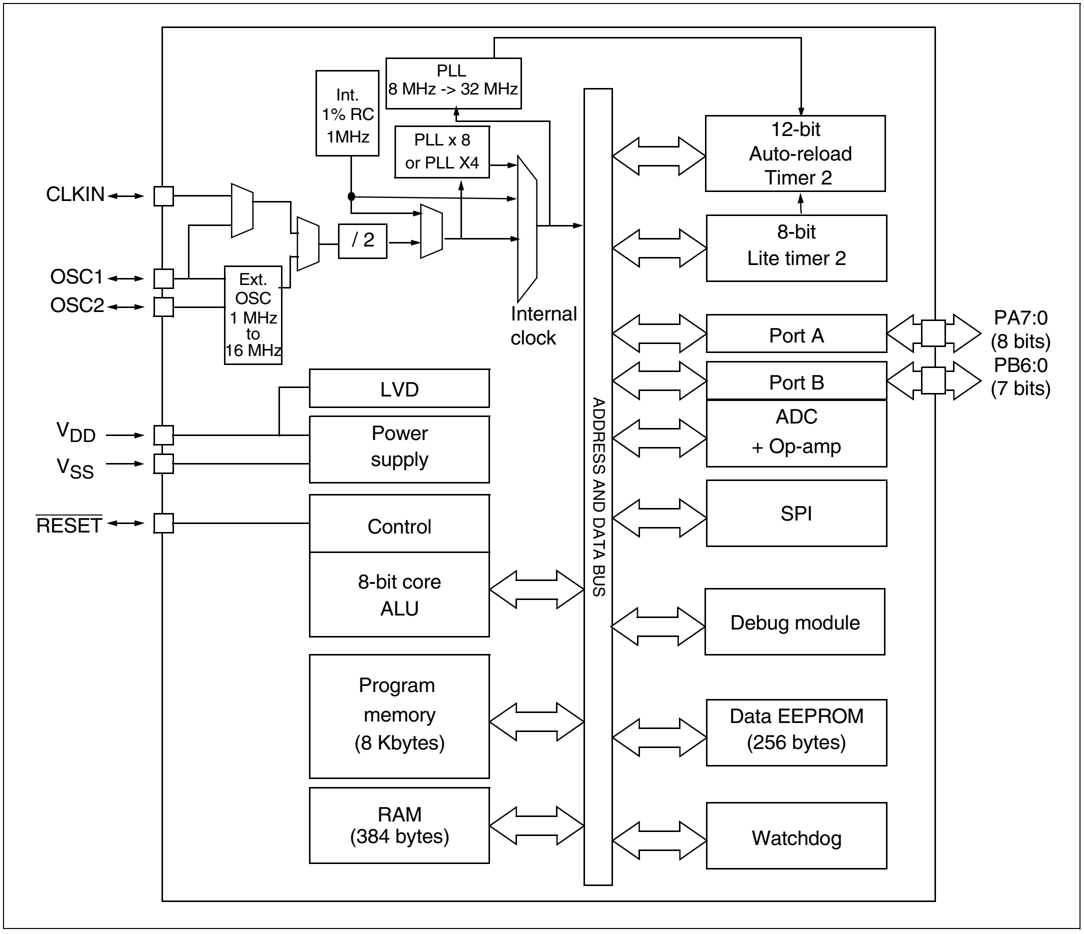
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

## Applications of "Embedded - Microcontrollers"

| **Details** | |
|---|---|
| Product Status | Obsolete |
| Core Processor | ST7 |
| Core Size | 8-Bit |
| Speed | 8MHz |
| Connectivity | SPI |
| Peripherals | LVD, POR, PWM, WDT |
| Number of I/O | 15 |
| Program Memory Size | 8KB (8K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 384 x 8 |
| Voltage - Supply (Vcc/Vdd) | 2.4V ~ 5.5V |
| Data Converters | A/D 7x10b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Through Hole |
| Package / Case | 20-DIP (0.300", 7.62mm) |
| Supplier Device Package | - |
| Purchase URL | https://www.e-xfl.com/product-detail/stmicroelectronics/st7flite20f1b6 |

# List of figures

**Figure 1. General block diagram**



### Read operation (E2LAT=0)

The EEPROM can be read as a normal ROM location when the E2LAT bit of the EECSR register is cleared.

On this device, Data EEPROM can also be used to execute machine code. Take care not to write to the Data EEPROM while executing from it. This would result in an unexpected code being executed.

### Write operation (E2LAT=1)

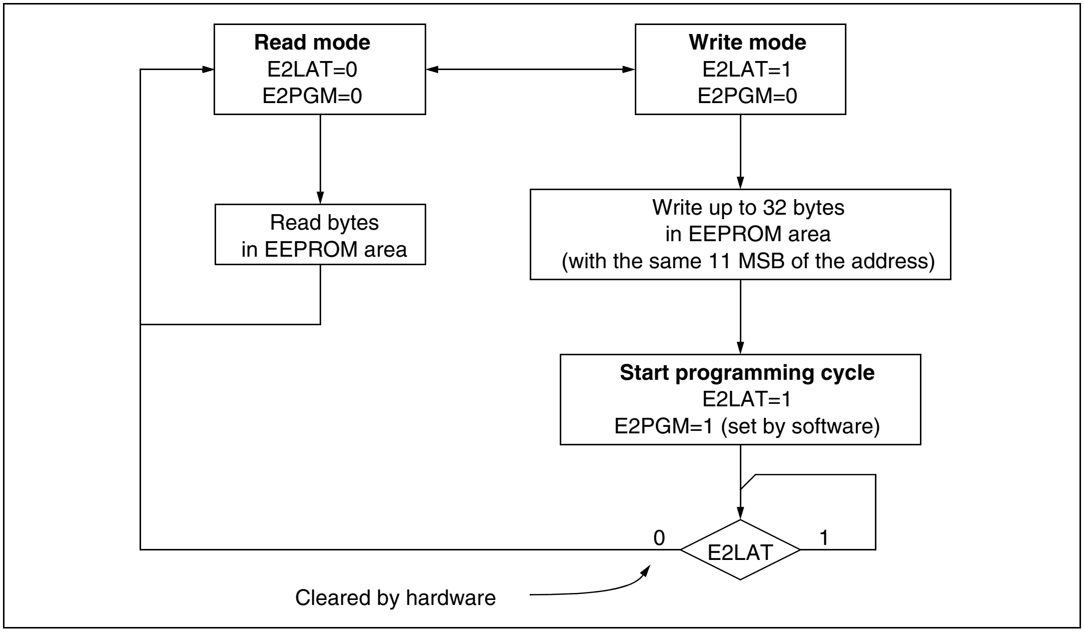
To access the write mode, the E2LAT bit has to be set by software (the E2PGM bit remains cleared). When a write access to the EEPROM area occurs, the value is latched inside the 32 data latches according to its address.

When PGM bit is set by the software, all the previous bytes written in the data latches (up to 32) are programmed in the EEPROM cells. The effective high address (row) is determined by the last EEPROM write sequence. To avoid wrong programming, the user must take care that all the bytes written between two programming sequences have the same high address: only the five Least Significant Bits of the address can change.

At the end of the programming cycle, the PGM and LAT bits are cleared simultaneously.

*Note: Care should be taken during the programming cycle. Writing to the same memory location will over-program the memory (logical AND between the two write access data result) because the data latches are only cleared at the end of the programming cycle and by the falling edge of the E2LAT bit.*
*It is not possible to read the latched data.*
*This note is illustrated by the Figure 9: Data EEPROM programming cycle.*

**Figure 7. Data EEPROM programming flowchart**



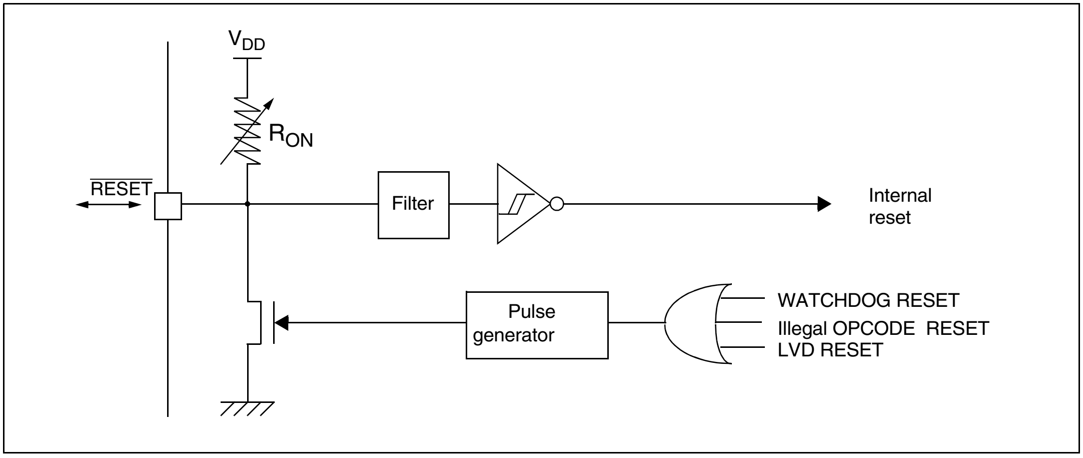
### 7.5.2 Asynchronous external $\overline{\text{RESET}}$ pin

The $\overline{\text{RESET}}$ pin is both an input and an open-drain output with integrated $R_{ON}$ weak pull-up resistor. This pull-up has no fixed value but varies in accordance with the input voltage. It can be pulled low by external circuitry to reset the device.

*Note: See Section 13: Electrical characteristics for more details.*

A RESET signal originating from an external source must have a duration of at least $t_{h(RSTL)in}$ in order to be recognized (see *Figure 16: RESET sequences*). This detection is asynchronous and therefore the MCU can enter reset state even in HALT mode.

**Figure 15. Reset block diagram**



*Note: See Section 12.2.1: Illegal opcode reset for more details on illegal opcode reset conditions.*

The $\overline{\text{RESET}}$ pin is an asynchronous signal which plays a major role in EMS performance. In a noisy environment, it is recommended to follow the guidelines mentioned in *Section 13: Electrical characteristics*.

### 7.5.3 External power-on RESET

If the LVD is disabled by option byte, to start up the microcontroller correctly, the user must ensure by means of an external reset circuit that the reset signal is held low until $V_{DD}$ is over the minimum level specified for the selected $f_{OSC}$ frequency.

A proper reset signal for a slow rising $V_{DD}$ supply can generally be provided by an external RC network connected to the RESET pin.

### 7.5.4 Internal low voltage detector (LVD) RESET

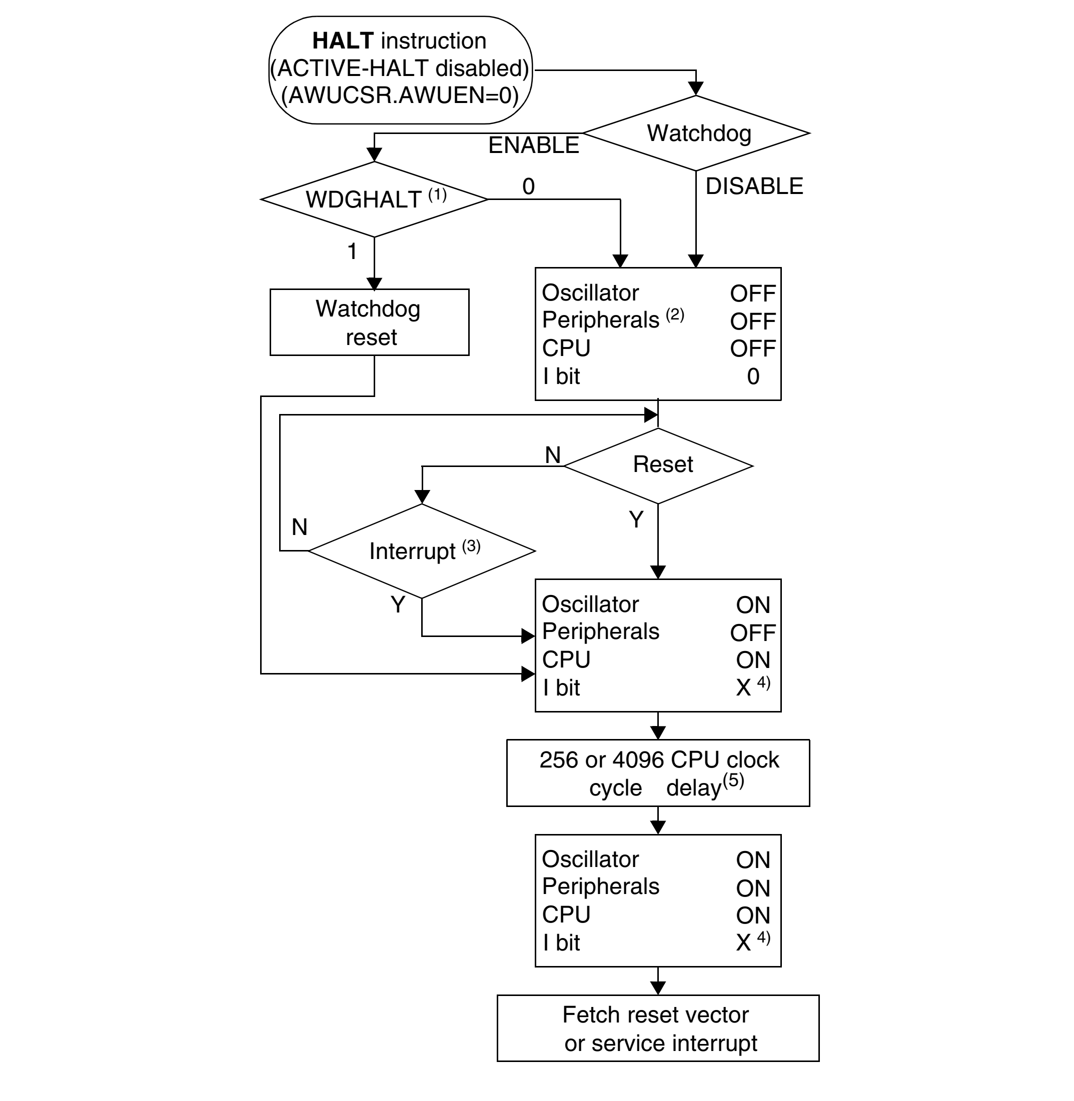
Two different RESET sequences caused by the internal LVD circuitry can be distinguished:

- Power-on RESET
- Voltage drop RESET.

The device $\overline{\text{RESET}}$ pin acts as an output that is pulled low when $V_{DD}<V_{IT+}$ (rising edge) or $V_{DD}<V_{IT-}$ (falling edge) as shown in *Figure 16: RESET sequences*.

The LVD filters spikes on $V_{DD}$ larger than $t_{g(VDD)}$ to avoid parasitic resets.

Figure 25. HALT mode flowchart



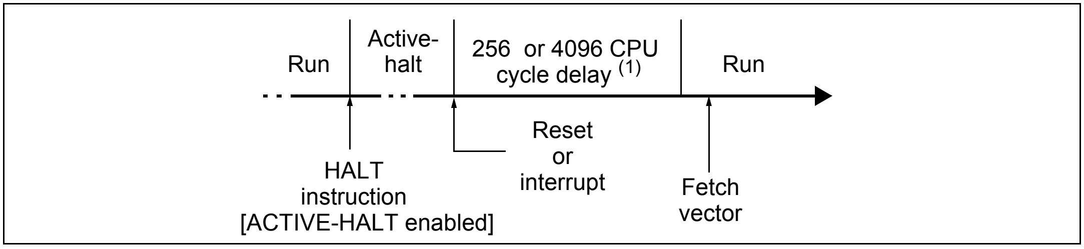
1. WDGHALT is an option bit (see *Section 15.1: Option bytes* for more details).
2. Peripheral clocked with an external clock source can still be active.
3. Only some specific interrupts can exit the MCU from HALT mode (such as external interrupt). Refer to *Table 12: Interrupt mapping* for more details.
4. Before servicing an interrupt, the CC register is pushed on the stack. The I bit of the CC register is set during the interrupt routine and cleared when the CC register is popped.
5. If the PLL is enabled by option byte, it outputs the clock after a delay of $t_{STARTUP}$ (see *Figure 12: PLL output frequency timing diagram*).

### 9.4.1 HALT mode recommendations

- Make sure that an external event is available to wake up the microcontroller from Halt mode.
- When using an external interrupt to wake up the microcontroller, reinitialize the corresponding I/O as “Input Pull-up with Interrupt” before executing the HALT

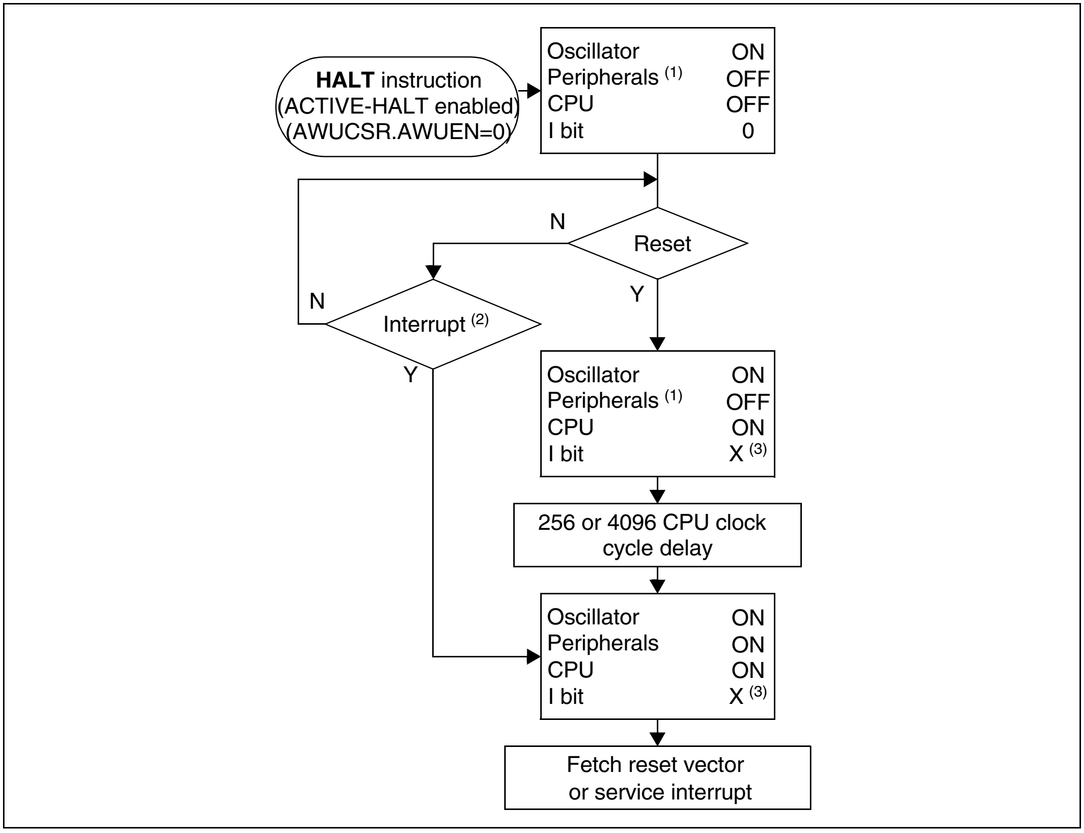
*Note: As soon as ACTIVE-HALT is enabled, executing a HALT instruction while the Watchdog is active does not generate a RESET.*
*This means that the device cannot spend more than a defined delay in this power saving mode.*

### Figure 26. ACTIVE-HALT timing overview


Run
Active-halt
256 or 4096 CPU cycle delay (1)
Run
HALT instruction [ACTIVE-HALT enabled]
Reset or interrupt
Fetch vector

1. This delay occurs only if the MCU exits ACTIVE-HALT mode by means of a RESET.

### Figure 27. ACTIVE-HALT mode Flow-chart



1. Peripherals clocked with an external clock source can still be active.
2. Only the RTC1 interrupt and some specific interrupts can exit the MCU from ACTIVE-HALT mode. Refer to *Table 12: Interrupt mapping* for more details.
3. Before servicing an interrupt, the CC register is pushed on the stack. The I bit of the CC register is set during the interrupt routine and cleared when the CC register is popped.

### 10.2.3 Alternate functions

Many ST7s I/Os have one or more alternate functions. These may include output signals from, or input signals to, on-chip peripherals. The Device Pin Description table describes which peripheral signals can be input/output to which ports.

A signal coming from an on-chip peripheral can be output on an I/O. To do this, enable the on-chip peripheral as an output (enable bit in the peripheral's control register). The peripheral configures the I/O as an output and takes priority over standard I/O programming. The I/O's state is readable by addressing the corresponding I/O data register.

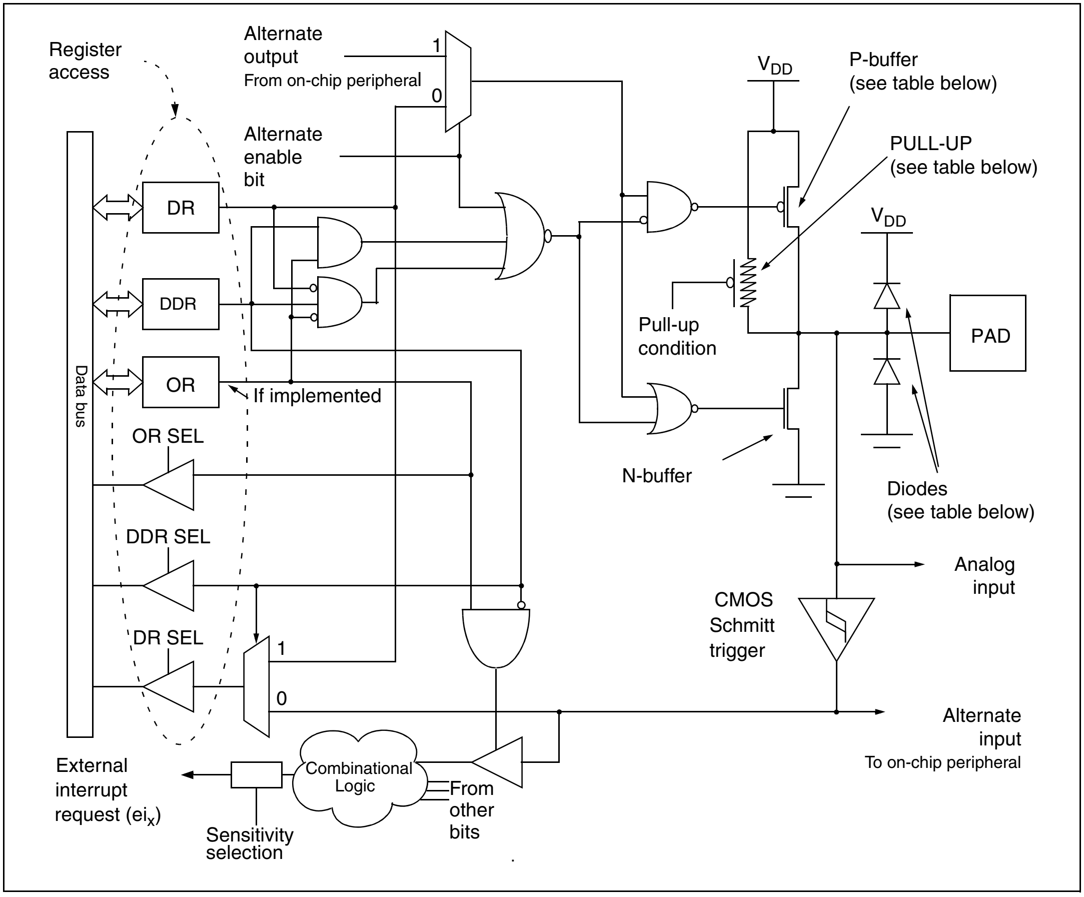
Configuring an I/O as floating enables alternate function input. It is not recommended to configure an I/O as pull-up as this will increase current consumption.

Before using an I/O as an alternate input, configure it without interrupt. Otherwise spurious interrupts can occur.

Configure an I/O as input floating for an on-chip peripheral signal which can be input and output.

**Caution:** I/Os which can be configured as both an analog and digital alternate function need special attention.
The user must control the peripherals so that the signals do not arrive at the same time on the same pin. If an external clock is used, only the clock alternate function should be employed on that I/O pin and not the other alternate function.

**Figure 31. I/O port general block diagram**



# 11 On-chip peripherals

## 11.1 Watchdog timer (WDG)

### 11.1.1 Introduction

The Watchdog timer is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence. The Watchdog circuit generates an MCU reset on expiry of a programmed time period, unless the program refreshes the counter's contents before the T6 bit becomes cleared.

### 11.1.2 Main features

- Programmable free-running downcounter (64 increments of 16000 CPU cycles)
- Programmable reset
- Reset (if watchdog activated) when the T6 bit reaches zero
- Optional reset on HALT instruction (configurable by option byte)
- Hardware Watchdog selectable by option byte.

### 11.1.3 Functional description

The counter value stored in the CR register (bits T[6:0]), is decremented every 16000 machine cycles, and the length of the time-out period can be programmed by the user in 64 increments.

If the watchdog is activated (the WDGA bit is set) and when the 7-bit timer (bits T[6:0]) rolls over from 40h to 3Fh (T6 becomes cleared), it initiates a reset cycle pulling low the reset pin for typically 30µs.

**Figure 33. Watchdog block diagram**

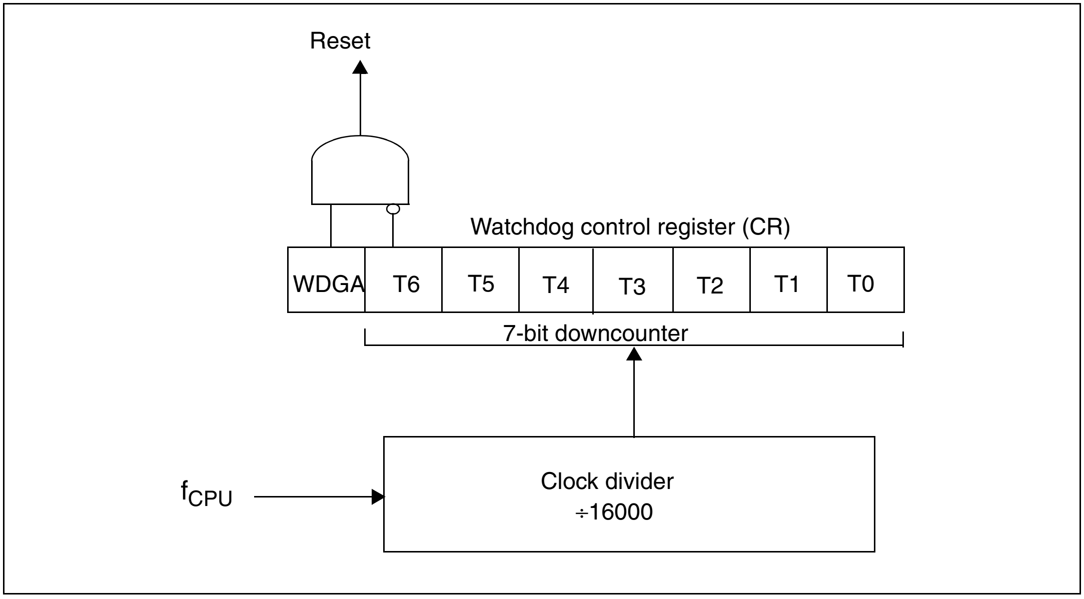


Figure 36. PWM function

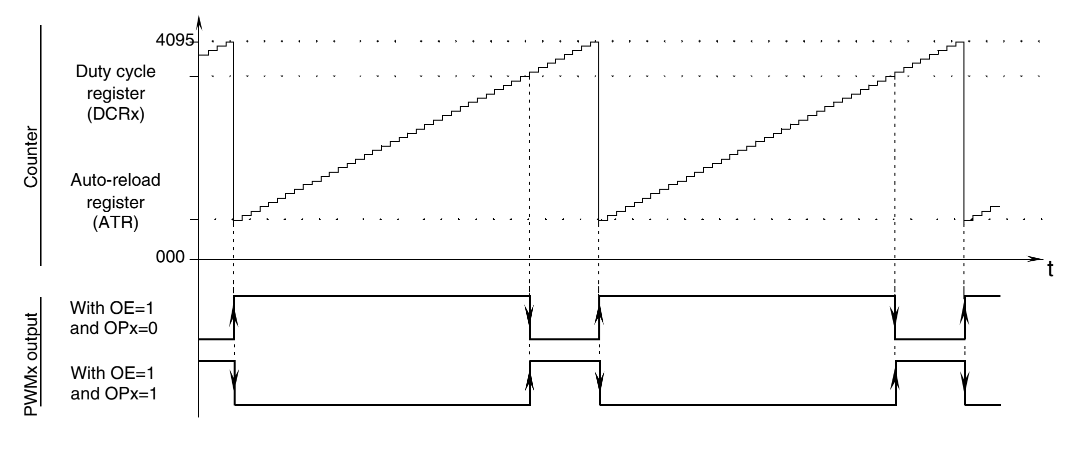
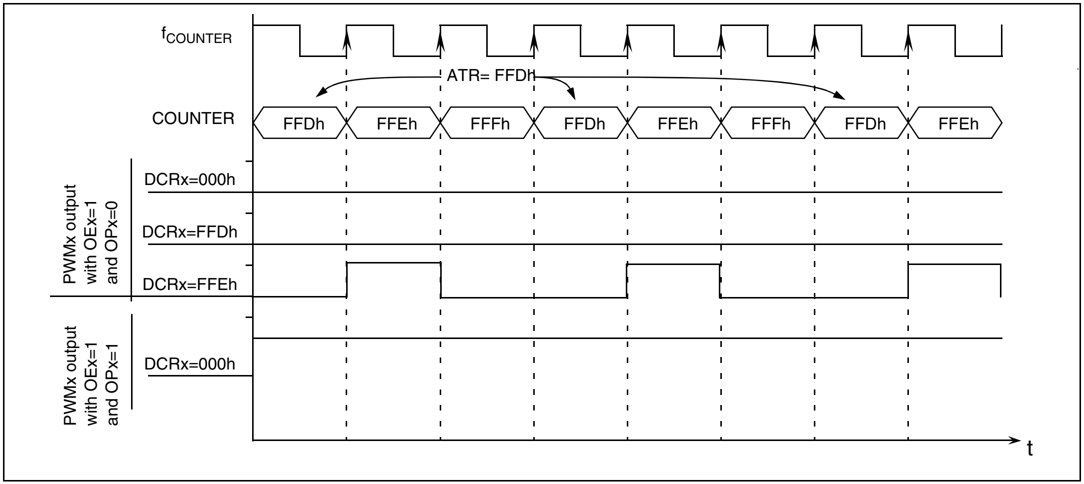


Figure 37. PWM signal from 0% to 100% duty cycle



## Output compare mode

To use this function, load a 12-bit value in the DCRxH and DCRxL registers.

When the 12-bit upcounter (CNTR) reaches the value stored in the DCRxH and DCRxL registers, the CMPF bit in the PWMxCSR register is set and an interrupt request is generated if the CMPIE bit is set.

*Note: The output compare function is only available for DCRx values other than 0 (reset value).*

## Break function

The break function is used to perform an emergency shutdown of the power converter.

The break function is activated by the external BREAK pin (active low). In order to use the BREAK pin it must be previously enabled by software setting the BPEN bit in the BREAKCR register.

When a low level is detected on the BREAK pin, the BA bit is set and the break function is activated.

### 11.2.6 Register description

#### Timer control status register (ATCSR)

Read / Write

Reset value: 0x00 0000 (x0h)

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| 0 | ICF | ICIE | CK1 | CK0 | OVF | OVFIE | CMPIE |

- Bit 7 = Reserved.
- Bit 6 = **ICF** *Input capture flag*
  This bit is set by hardware and cleared by software by reading the ATICR register (a read access to ATICRH or ATICRL will clear this flag). Writing to this bit does not change the bit value.
  0: No input capture
  1: An input capture has occurred
- Bit 5 = **ICIE** *IC interrupt enable*
  This bit is set and cleared by software.
  0: Input capture interrupt disabled
  1: Input capture interrupt enabled
- Bits 4:3 = **CK[1:0]** *Counter clock selection*
  These bits are set and cleared by software and cleared by hardware after a reset. They select the clock frequency of the counter.

**Table 35. Counter clock selection**

| Counter clock selection | CK1 | CK0 |
|---|---|---|
| OFF | 0 | 0 |
| $f_{LTIMER}$ (1 ms timebase @ 8 MHz) [(1)] | 0 | 1 |
| $f_{CPU}$ | 1 | 0 |
| 32 MHz [(2)] | 1 | 1 |

1. PWM mode and Output Compare modes are not available at this frequency.
2. ATICR counter may return inaccurate results when read. It is therefore not recommended to use Input Capture mode at this frequency.

- Bit 2 = **OVF** *Overflow flag*
  This bit is set by hardware and cleared by software by reading the TCSR register. It indicates the transition of the counter from FFFh to ATR value.
  0: No counter overflow occurred
  1: Counter overflow occurred
- Bit 1 = **OVFIE** *Overflow interrupt enable*
  This bit is read/write by software and cleared by hardware after a reset.
  0: OVF interrupt disabled.
  1: OVF interrupt enabled.
- Bit 0 = **CMPIE** *Compare interrupt enable*
  This bit is read/write by software and cleared by hardware after a reset. It can be used to mask the interrupt generated when the CMPF bit is set.

### Break control register (BREAKCR)

Read/Write

Reset Value: 0000 0000 (00h)

| 7 |  |  |  |  |  |  | 0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | BA | BPEN | PWM3 | PWM2 | PWM1 | PWM0 |

- Bits 7:6 = Reserved. Forced by hardware to 0.
- Bit 5 = **BA** *Break Active*
  This bit is read/write by software, cleared by hardware after reset and set by hardware when the BREAK pin is low. It activates/deactivates the Break function.
  0: Break not active
  1: Break active
- Bit 4 = **BPEN** *Break pin enable*
  This bit is read/write by software and cleared by hardware after Reset.
  0: Break pin disabled
  1: Break pin enabled
- Bits 3:0 = **PWM[3:0]** *Break pattern*
  These bits are read/write by software and cleared by hardware after a reset. They are used to force the four PWMx output signals into a stable state when the Break function is active.

### PWMx duty cycle register high (DCRxH)

Read / Write

Reset Value: 0000 0000 (00h)

| 15 |  |  |  |  |  |  | 8 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | DCR11 | DCR10 | DCR9 | DCR8 |

### PWMx duty cycle register low (DCRxL)

Read / Write

Reset value: 0000 0000 (00h)

| 7 |  |  |  |  |  |  | 0 |
|---|---|---|---|---|---|---|---|
| DCR7 | DCR6 | DCR5 | DCR4 | DCR3 | DCR2 | DCR1 | DCR0 |

- Bits 15:12 = Reserved
- Bits 11:0 = **DCR[11:0]** *PWMx duty cycle value*
  This 12-bit value is written by software. It defines the duty cycle of the corresponding PWM output signal (see *Figure 36*).
  In PWM mode (OEx=1 in the PWMCR register) the DCR[11:0] bits define the duty cycle of the PWMx output signal (see *Figure 36*). In Output Compare mode, they define the value to be compared with the 12-bit upcounter value.

# 12 Instruction set

## 12.1 ST7 addressing modes

The ST7 Core features 17 different addressing modes which can be classified in seven main groups:

**Table 48. Addressing mode groups**

| Addressing mode | Example |
|---|---|
| Inherent | nop |
| Immediate | ld A,#$55 |
| Direct | ld A,$55 |
| Indexed | ld A,($55,X) |
| Indirect | ld A,([$55],X) |
| Relative | jrne loop |
| Bit operation | bset byte,#5 |

The ST7 Instruction set is designed to minimize the number of bytes required per instruction: To do so, most of the addressing modes may be subdivided in two submodes called long and short:

- Long addressing mode is more powerful because it can use the full 64 Kbyte address space, however it uses more bytes and more CPU cycles.
- Short addressing mode is less powerful because it can generally only access page zero (0000h00FFh range), but the instruction size is more compact, and faster. All memory to memory instructions use short addressing modes only (CLR, CPL, NEG, BSET, BRES, BTJT, BTJF, INC, DEC, RLC, RRC, SLL, SRL, SRA, SWAP).

The ST7 Assembler optimizes the use of long and short addressing modes.

**Table 49. ST7 addressing mode overview**

| Mode | | | Syntax | Destination/ source | Pointer address (Hex.) | Pointer size (Hex.) | Length (bytes) |
|---|---|---|---|---|---|---|---|
| Inherent | – | – | nop | – | – | – | + 0 |
| Immediate | – | – | ld A,#$55 | – | – | – | + 1 |
| Short | Direct | – | ld A,$10 | 00..FF | – | – | + 1 |
| Long | Direct | – | ld A,$1000 | 0000..FFFF | – | – | + 2 |
| No offset | Direct | Indexed | ld A,(X) | 00..FF | – | – | + 0 (with x register)<br>+ 1 (with Y register) |
| Short | Direct | Indexed | ld A,($10,X) | 00..1FE | – | – | + 1 |

Figure 54. RC Osc Freq vs $V_{DD}$ (calibrated with RCCR0: 5V@ 25°C)

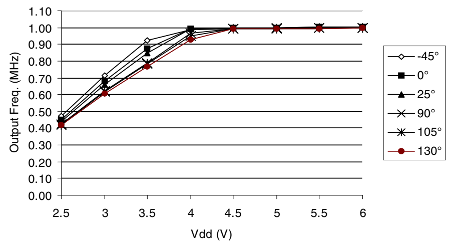


Figure 55. Typical RC oscillator Accuracy vs temperature @ $V_{DD}$=5V (calibrated with RCCR0: 5V @ 25°C)

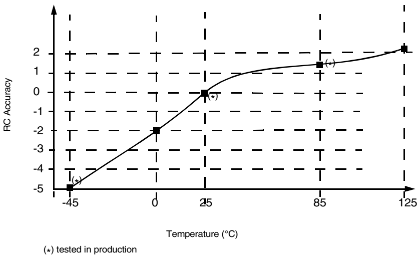


Figure 56. RC Osc Freq vs $V_{DD}$ and RCCR Value

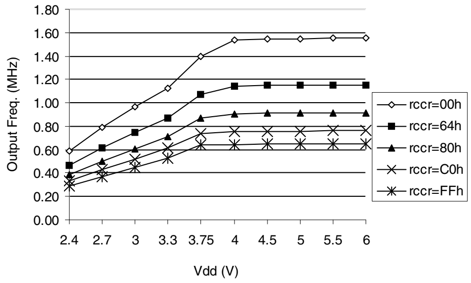
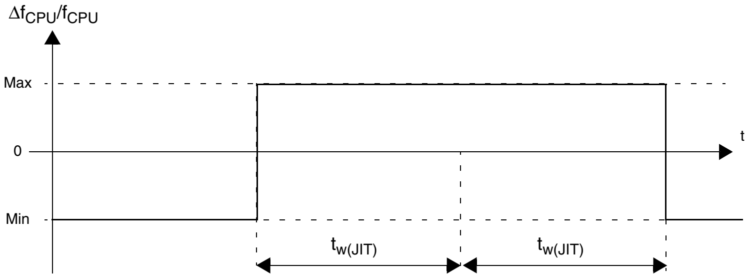
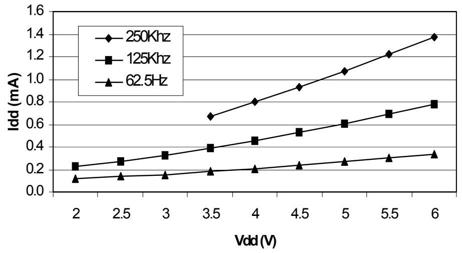


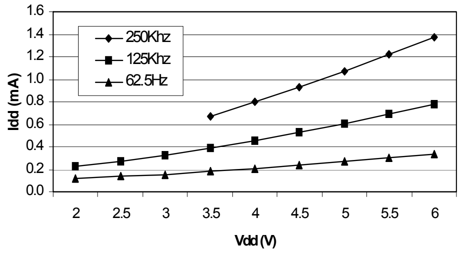
Figure 57. PLL $\Delta f_{CPU}/f_{CPU}$ versus time



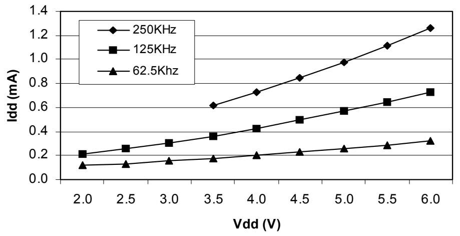
### Figure 61. Typical $I_{DD}$ in SLOW vs. $f_{CPU}$



### Figure 62. Typical $I_{DD}$ in WAIT vs. $f_{CPU}$



### Figure 63. Typical $I_{DD}$ in SLOW-WAIT vs. $f_{CPU}$



### Figure 64. Typical $I_{DD}$ in AWUF mode at $T_A$ = 25°C

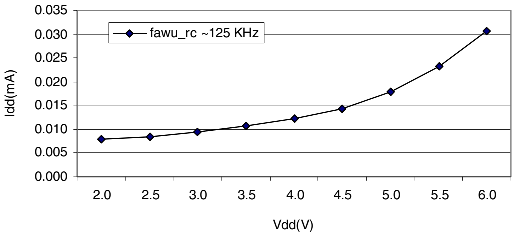
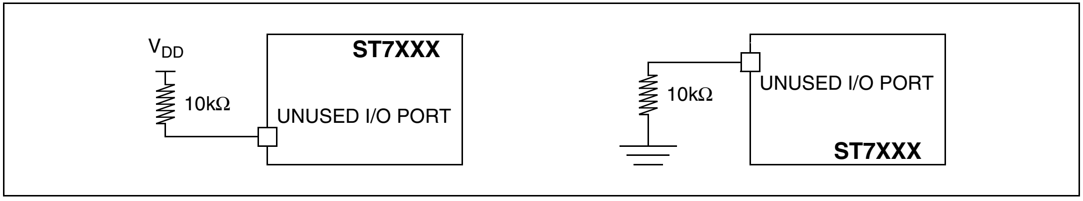


Figure 67. Two typical applications with unused I/O Pin



**Caution:** To avoid entering ICC mode unexpectedly during a reset, the ICCCLK pin must be pulled-up internally or externally during normal operation (external pull-up of 10k mandatory in noisy environment).

*Note: I/O can be left unconnected if it is configured as output (0 or 1) by the software. This has the advantage of greater EMC robustness and lower cost.*

Figure 68. Typical $I_{PU}$ vs. $V_{DD}$ with $V_{IN}$ =$V_{SS}$

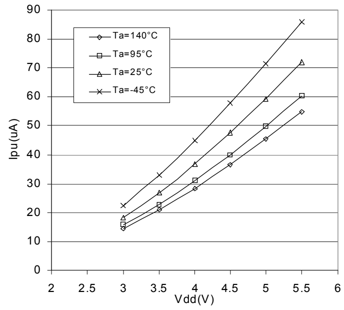


Figure 82. Typical $V_{OL}$ vs. $V_{DD}$ (high-sink I/Os)

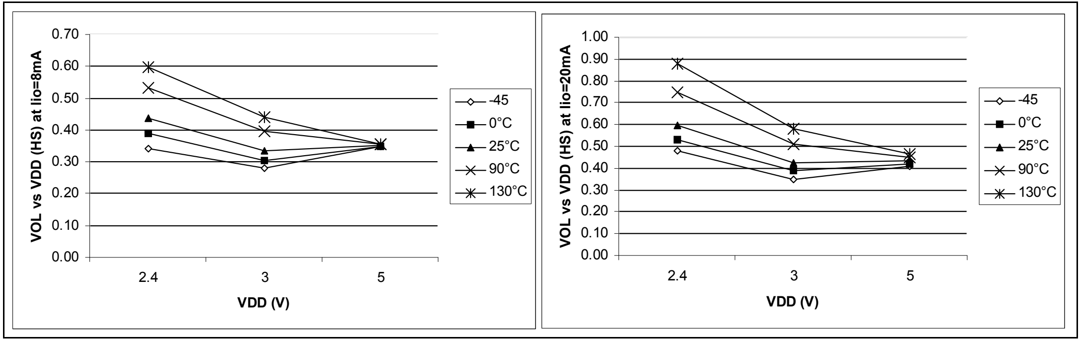
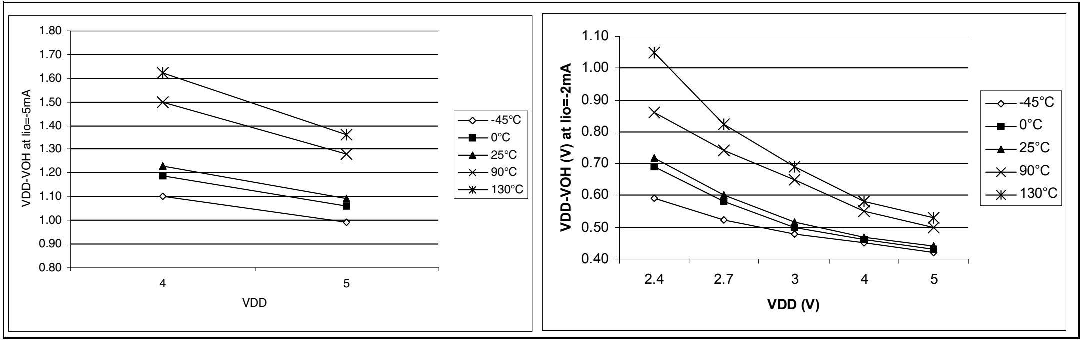


Figure 83. Typical $V_{DD}$-$V_{OH}$ vs. $V_{DD}$



## 13.9 Control pin characteristics

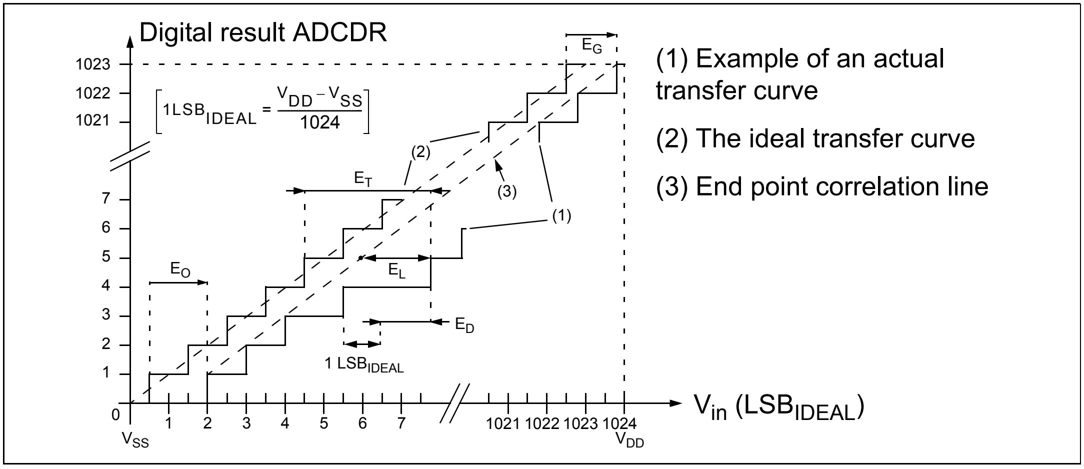
Table 82. Asynchronous $\overline{\text{RESET}}$ Pin(1)

| Symbol | Parameter | Conditions | | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|---|
| $V_{IL}$ | Input low level voltage | – | | $V_{SS}$ - 0.3 | – | $0.3xV_{DD}$ | V |
| $V_{IH}$ | Input high level voltage | – | | $0.7xV_{DD}$ | – | $V_{DD}$ + 0.3 | |
| $V_{hys}$ | Schmitt trigger voltage hysteresis(2) | – | | – | 2 | – | V |
| $V_{OL}$ | Output low level voltage(3) | $V_{DD}$=5V | $I_{IO}$=+5mA$T_A$≤85°C<br>$T_A$≥85°C | – | 0.5 | 1.0<br>1.2 | V |
| | | | $I_{IO}$=+2mA$T_A$≤85°C<br>$T_A$≥85°C | – | 0.2 | 0.4<br>0.5 | |
| $R_{ON}$ | Pull-up equivalent resistor(2)(4) | $V_{DD}$=5V | | 20 | 40 | 80 | kΩ |
| | | $V_{DD}$=3V | | 40 | 70 | 120 | |
| $t_{w(RSTL)out}$ | Generated reset pulse duration | Internal reset sources | | – | 30 | – | µs |
| $t_{h(RSTL)in}$ | External reset pulse hold time(5) | – | | 20 | – | – | µs |
| $t_{g(RSTL)in}$ | Filtered glitch duration | – | | – | 200 | – | ns |

1. TA = -40°C to 85°C, unless otherwise specified.

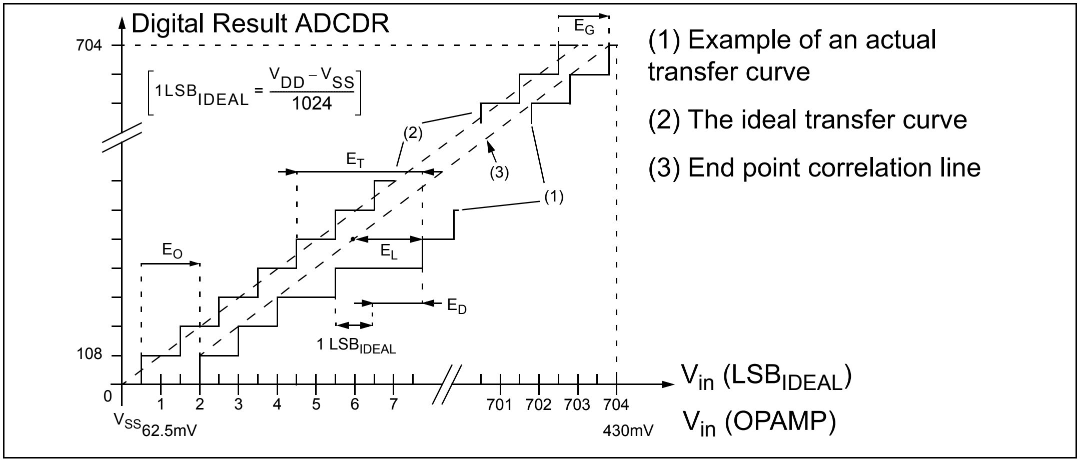
2. Data based on characterization results, not tested in production.

Figure 90. ADC accuracy characteristics with amplifier disabled



- $E_T$=Total Unadjusted Error: maximum deviation between the actual and the ideal transfer curves.
- $E_O$=Offset Error: deviation between the first actual transition and the first ideal one.
- $E_G$=Gain Error: deviation between the last ideal transition and the last actual one.
- $E_D$=Differential Linearity Error: maximum deviation between actual steps and the ideal one.
- $E_L$=Integral Linearity Error: maximum deviation between any actual transition and the end point correlation line.

Figure 91. ADC accuracy characteristics with amplifier enabled



Note: *When the AMPSEL bit in the ADCDRL register is set, it is mandatory that $f_{ADC}$ be less than or equal to 2 MHz (if $f_{CPU}$=8MHz, then SPEED=0, SLOW=1).*

- $E_T$=Total Unadjusted Error: maximum deviation between the actual and the ideal transfer curves.
- $E_O$=Offset Error: deviation between the first actual transition and the first ideal one.
- $E_G$=Gain Error: deviation between the last ideal transition and the last actual one.
- $E_D$=Differential Linearity Error: maximum deviation between actual steps and the ideal one.
- $E_L$=Integral Linearity Error: maximum deviation between any actual transition and the end point correlation line.

Table 99. STMicroelectronics development tools

| Supported Products | Emulation | | | | Programming |
|---|---|---|---|---|---|
| | ST7 DVP3 Series | | ST7 EMU3 series | | ICC Socket Board |
| | Emulator | Connection kit | Emulator | Active Probe & T.E.B. | |
| ST7FLITE20<br>ST7FLITE25<br>ST7FLITE29 | ST7MDT10-DVP3 | ST7MDT10-20/ DVP | ST7MDT10-EMU3 | ST7MDT10-TEB | ST7SB10/123[1] |

[1] Add suffix /EU, /UK, /US for the power supply of your region.

## 15.4 Application notes

Table 100. ST7 application notes

| Identification | Description |
|---|---|
| **Application examples** | |
| AN1658 | Serial Numbering Implementation |
| AN1720 | Managing the Read-out Protection in Flash Microcontrollers |
| AN1755 | A High Resolution/Precision thermometer using ST7 and NE555 |
| AN1756 | Choosing a DALI Implementation Strategy with ST7DALI |
| AN1812 | A High Precision, Low Cost, Single Supply ADC for Positive and Negative Input Voltages |
| **Example drivers** | |
| AN 969 | SCI Communication Between ST7 and PC |
| AN 970 | SPI Communication Between ST7 and EEPROM |
| AN 971 | I²C Communication Between ST7 and M24Cxx EEPROM |
| AN 972 | ST7 Software SPI Master Communication |
| AN 973 | SCI Software Communication with a PC Using ST72251 16-Bit Timer |
| AN 974 | Real Time Clock with ST7 Timer Output Compare |
| AN 976 | Driving a Buzzer Through ST7 Timer PWM Function |
| AN 979 | Driving An Analog Keyboard with the ST7 ADC |
| AN 980 | ST7 Keypad Decoding Techniques, Implementing wakeup on Keystroke |
| AN1017 | Using the ST7 Universal Serial Bus Microcontroller |
| AN1041 | Using ST7 PWM Signal to Generate Analog Output (Sinusoïd) |
| AN1042 | ST7 Routine for I²C Slave Mode Management |
| AN1044 | Multiple Interrupt Sources Management for ST7 MCUs |
| AN1045 | ST7 S/W Implementation of I²C Bus Master |
| AN1046 | UART Emulation Software |
| AN1047 | Managing Reception Errors with the ST7 SCI Peripherals |

# 17 Revision history

Table 101. Revision history

| Date | Revision | Description of changes |
|---|---|---|
| 30-Aug-2004 | 3 | Updated *Figure 62. Typical IDD in WAIT vs. fCPU* with correct data<br>Added data for Fcpu @ 1MHz into *Section 13.4.1 Supply Current* table.<br>EnabledProgramming Capability for EMU3, *Table 26*<br>Reset delay in *section 11.1.3 on page 53* changed to 30µs<br>Altered note 1 for *section 13.2.3 on page 94* removing references to $\overline{\text{RESET}}$<br>Removed sentence relating to an effective change only after overflow for CK[1:0], *page 61*<br>MOD00 replaced by 0Ex in *Figure 37 on page 58*<br>Added Note 2 related to Exit from Active Halt, *section 11.2.5 on page 60*<br>Changed *section 11.4.2 on page 71*<br>Changed *section 11.4.3.3 on page 74*<br>Added illegal opcode detection to page 1, *section 7.6 on page 30*, *section 12 on page 87*<br>Clarification of Flash Readout protection, *section 4.5.1 on page 14*<br>Added note 4 and description relating to Total Percentage in Error and Amplifier Output Offset Variation to the ADC Characteristics subsection and table, *page 120*<br>Added note 5 and description relating to Offset Variation in Temperature to ADC Characteristics subsection and table, *page 120*<br>$f_{PLL}$ value of 1MHz quoted as Typical instead of a Minimum in *section 13.3.4.1 on page 97*<br>Updated $f_{SCK}$ in *section 13.10.1 on page 115* to $f_{CPU}/4$ and $f_{CPU}/2$<br>Corrected$f_{CPU}$ in SLOW and SLOW WAIT modes in *section 13.4.1 on page 101*<br>Max values updated for ADC Accuracy, *page 118*<br>Socket Board development kit details added in *Table 27 on page 126*<br>Notes indicating that PB4 cannot be used as an external interrupt in HALT mode, *section 16.6 on page 132* and *Section 8.3 PERIPHERAL INTERRUPTS*<br><br>-Removed “optional” referring to $V_{DD}$ in *Figure 5 on page 13*<br>-Changed FMP_R option bit description in *section 15.1 on page 124*<br>-Added “CLEARING ACTIVE INTERRUPTS OUTSIDE INTERRUPT ROUTINE” on page 132 |