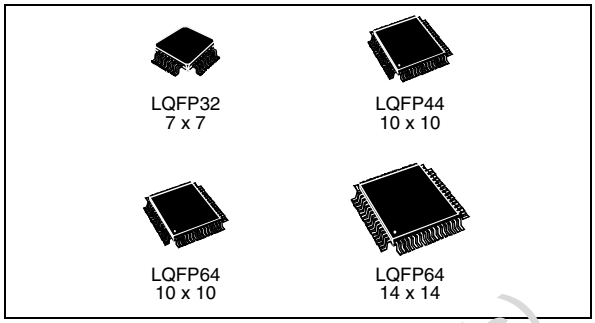
# ST72325xxx-Auto

## 8-bit MCU for automotive with 16 to 60 Kbyte Flash, ADC, CSS, 5 timers, SPI, SCI, I2C interface

## Features

- **Memories**
  - 16 to 60 Kbyte dual voltage High Density Flash (HDFlash) with readout protection capability. In-application programming and in-circuit programming for HDFlash devices
  - 512 to 2048 bytes RAM
  - HDFlash endurance: 100 cycles, data retention 20 years
- **Clock, reset and supply management**
  - Enhanced low voltage supervisor (LVD) for main supply and auxiliary voltage detector (AVD) with interrupt capability
  - Clock sources: crystal/ceramic resonator oscillators, internal RC oscillator and bypass for external clock
  - PLL for 2x frequency multiplication
  - 4 power saving modes: Halt, Active Halt, Wait and Slow
  - Clock security system
- **Interrupt management**
  - Nested interrupt controller
  - 14 interrupt vectors plus TRAP and RESET
  - Top Level Interrupt (TLI) pin on 64-pin devices
  - 9/6 external interrupt lines (on 4 vectors)
- **1 analog peripheral (low current coupling)**
  - 10-bit ADC with up to 16 input ports
- **Up to 48 I/O ports**
  - 48/32/24 multifunctional bidirectional I/O lines
  - 34/22/17 alternate function lines
  - 16/12/10 high sink outputs
- **5 timers**
  - Main clock controller with Real-time base, Beep and Clock-out capabilities
  - Configurable watchdog timer




  - Two 16-bit timers with 2 input captures, 2 output compares, external clock input on 1 timer, PWM and pulse generator modes
  - 8-bit PWM auto-reload timer with 2 input captures, 4 PWM outputs, output compare and time base interrupt, external clock with event detector
- **3 communications interfaces**
  - SPI synchronous serial interface
  - SCI asynchronous serial interface
  - $I^2C$ multimaster interface
- **Instruction set**
  - 8-bit data manipulation
  - 63 basic instructions
  - 17 main addressing modes
  - 8x8 unsigned multiply instruction
- **Development tools**
  - Full hardware/software development package
  - DM (debug module)

**Table 1. Device summary**

| Reference | Part number |
|---|---|
| ST72325xxx-Auto | ST72325K4-Auto, ST72325K6-Auto, ST72325J4-Auto, ST72325J6-Auto, ST72325J7-Auto, ST72325J9-Auto, ST72325AR6-Auto, ST72325R6-Auto, ST72325AR7-Auto, ST72325R7-Auto, ST72325AR9-Auto, ST72325R9-Auto |

Obsolete Product(s) - Obsolete Product(s)

# Contents


1 **Description . . . . . 18**

2 **Package pinout and pin description . . . . . 20**

2.1 Package pinout . . . . . 20

2.2 Pin description . . . . . 23

3 **Register and memory map . . . . . 30**

4 **Flash program memory . . . . . 34**

4.1 Introduction . . . . . 34

4.2 Main features . . . . . 34

4.3 Structure . . . . . 34

4.3.1 Readout protection . . . . . 35

4.4 ICC interface . . . . . 35

4.5 ICP (in-circuit programming) . . . . . 36

4.6 IAP (in-application programming) . . . . . 37

4.7 Related documentation . . . . . 37

4.8 Flash control/status register (FCSR) . . . . . 37

5 **Central processing unit (CPU) . . . . . 38**

5.1 Introduction . . . . . 38

5.2 Main features . . . . . 38

5.3 CPU registers . . . . . 38

5.3.1 Accumulator (A) . . . . . 39

5.3.2 Index registers (X and Y) . . . . . 39

5.3.3 Program counter (PC) . . . . . 39

5.3.4 Condition code (CC) register . . . . . 39

5.3.5 Stack pointer (SP) register . . . . . 40

6 **Supply, reset and clock management . . . . . 42**

6.1 Introduction . . . . . 42

6.2 Main features . . . . . 42

6.3 Phase locked loop . . . . . 43


Obsolete Product(s) - Obsolete Product(s)

6.4 Multi-oscillator (MO) . . . . . 43
6.5 Reset sequence manager (RSM) . . . . . 44
6.5.1 Introduction . . . . . 44
6.5.2 Asynchronous external RESET pin . . . . . 45
6.5.3 External power-on RESET . . . . . 46
6.5.4 Internal low voltage detector (LVD) RESET . . . . . 46
6.5.5 Internal watchdog RESET . . . . . 46
6.6 System integrity management (SI) . . . . . 47
6.6.1 Low voltage detector (LVD) . . . . . 47
6.6.2 Auxiliary voltage detector (AVD) . . . . . 48
6.6.3 Clock security system (CSS) . . . . . 50
6.6.4 Low power modes . . . . . 51
6.6.5 Interrupts . . . . . 51
6.6.6 System Integrity (SI) Control/Status register (SICSR) . . . . . 52

**7 Interrupts . . . . . 54**
7.1 Introduction . . . . . 54
7.2 Masking and processing flow . . . . . 54
7.3 Interrupts and low power modes . . . . . 57
7.4 Concurrent and nested management . . . . . 57
7.5 Interrupt register description . . . . . 58
7.5.1 CPU CC register interrupt bits . . . . . 58
7.5.2 Interrupt software priority registers (ISPRx) . . . . . 59
7.6 External interrupts . . . . . 61
7.6.1 I/O port interrupt sensitivity . . . . . 61
7.6.2 External interrupt control register (EICR) . . . . . 63

**8 Power saving modes . . . . . 66**
8.1 Introduction . . . . . 66
8.2 Slow mode . . . . . 66
8.3 Wait mode . . . . . 67
8.4 Active Halt and Halt modes . . . . . 68
8.4.1 Active Halt mode . . . . . 69
8.4.2 Halt mode . . . . . 70

**9 I/O ports . . . . . 74**

Obsolete Product(s) - Obsolete Product(s)

9.1 Introduction . . . 74
9.2 Functional description . . . 74
9.2.1 Input modes . . . 74
9.2.2 Output modes . . . 75
9.2.3 Alternate functions . . . 75
9.3 I/O port implementation . . . 78
9.4 Low power modes . . . 79
9.5 Interrupts . . . 79

**10 Watchdog timer (WDG) . . . 81**
10.1 Introduction . . . 81
10.2 Main features . . . 81
10.3 Functional description . . . 81
10.4 How to program the watchdog timeout . . . 82
10.5 Low power modes . . . 84
10.6 Hardware watchdog option . . . 84
10.7 Using Halt mode with the WDG (WDGHALT option) . . . 84
10.8 Interrupts . . . 84
10.9 Register description . . . 85
10.9.1 Control register (WDGCR) . . . 85

**11 Main clock controller with real-time clock and beeper (MCC/RTC) . . 86**
11.1 Introduction . . . 86
11.2 Programmable CPU clock prescaler . . . 86
11.3 Clock-out capability . . . 86
11.4 Real-time clock timer (RTC) . . . 86
11.5 Beeper . . . 86
11.6 Low power modes . . . 87
11.7 Interrupts . . . 87
11.8 Main clock controller registers . . . 88
11.8.1 MCC control/status register (MCCSR) . . . 88
11.8.2 MCC beep control register (MCCBCR) . . . 89

**12 PWM auto-reload timer (ART) . . . 91**
12.1 Introduction . . . 91

Obsolete Product(s) - Obsolete Product(s)

12.2 Functional description . . . . . 92
12.2.1 Counter . . . . . 92
12.2.2 Counter clock and prescaler . . . . . 92
12.2.3 Counter and prescaler initialization . . . . . 92
12.2.4 Output compare control . . . . . 92
12.2.5 Independent PWM signal generation . . . . . 93
12.2.6 Output compare and time base interrupt . . . . . 94
12.2.7 External clock and event detector mode . . . . . 94
12.2.8 Input capture function . . . . . 95
12.2.9 External interrupt capability . . . . . 96
12.3 ART registers . . . . . 97
12.3.1 Control/status register (ARTCSR) . . . . . 97
12.3.2 Counter access register (ARTCAR) . . . . . 98
12.3.3 Auto-reload register (ARTARR) . . . . . 98
12.3.4 PWM control register (PWMCR) . . . . . 99
12.3.5 Duty cycle registers (PWMDCRx) . . . . . 100
12.3.6 Input capture control / status register (ARTICCSR) . . . . . 100
12.3.7 Input capture registers (ARTICRx) . . . . . 101

**13 16-bit timer . . . . . 102**
13.1 Introduction . . . . . 102
13.2 Main features . . . . . 102
13.3 Functional description . . . . . 103
13.3.1 Counter . . . . . 103
13.3.2 External clock . . . . . 106
13.3.3 Input capture . . . . . 107
13.3.4 Output compare . . . . . 109
13.3.5 Forced compare output capability . . . . . 110
13.3.6 One Pulse mode . . . . . 112
13.3.7 Pulse width modulation mode . . . . . 114
13.4 Low power modes . . . . . 116
13.5 Interrupts . . . . . 116
13.6 Summary of timer modes . . . . . 116
13.7 16-bit timer registers . . . . . 117
13.7.1 Control register 1 (CR1) . . . . . 117
13.7.2 Control register 2 (CR2) . . . . . 118

Obsolete Product(s) - Obsolete Product(s)

13.7.3 Control/status register (CSR) . . . 119
13.7.4 Input capture 1 high register (IC1HR) . . . 120
13.7.5 Input capture 1 low register (IC1LR) . . . 121
13.7.6 Output compare 1 high register (OC1HR) . . . 121
13.7.7 Output compare 1 low register (OC1LR) . . . 121
13.7.8 Output compare 2 high register (OC2HR) . . . 121
13.7.9 Output compare 2 low register (OC2LR) . . . 122
13.7.10 Counter high register (CHR) . . . 122
13.7.11 Counter low register (CLR) . . . 122
13.7.12 Alternate counter high register (ACHR) . . . 122
13.7.13 Alternate counter low register (ACLR) . . . 123
13.7.14 Input capture 2 high register (IC2HR) . . . 123
13.7.15 Input capture 2 low register (IC2LR) . . . 123

**14 Serial peripheral interface (SPI) . . . 125**

14.1 Introduction . . . 125
14.2 Main features . . . 125
14.3 General description . . . 125
14.3.1 Functional description . . . 126
14.3.2 Slave select management . . . 127
14.3.3 Master mode operation . . . 128
14.3.4 Master mode transmit sequence . . . 129
14.3.5 Slave mode operation . . . 129
14.3.6 Slave mode transmit sequence . . . 129
14.4 Clock phase and clock polarity . . . 130
14.5 Error flags . . . 132
14.5.1 Master mode fault (MODF) . . . 132
14.5.2 Overrun condition (OVR) . . . 132
14.5.3 Write collision error (WCOL) . . . 132
14.5.4 Single master systems . . . 133
14.6 Low power modes . . . 134
14.6.1 Using the SPI to wake up the MCU from Halt mode . . . 134
14.7 Interrupts . . . 134
14.8 SPI registers . . . 135
14.8.1 Control register (SPICR) . . . 135
14.8.2 Control/status register (SPICSR) . . . 136

Obsolete Product(s) - Obsolete Product(s)

14.8.3 Data I/O register (SPIDR) . . . . . 137

**15 Serial communications interface (SCI) . . . . . 139**

15.1 Introduction . . . . . 139
15.2 Main features . . . . . 139
15.3 General description . . . . . 140
15.4 Functional description . . . . . 142
15.4.1 Serial data format . . . . . 142
15.4.2 Transmitter . . . . . 143
15.4.3 Receiver . . . . . 144
15.5 Low power modes . . . . . 150
15.6 Interrupts . . . . . 150
15.7 SCI registers . . . . . 151
15.7.1 Status register (SCISR) . . . . . 151
15.7.2 Control register 1 (SCICR1) . . . . . 153
15.7.3 Control register 2 (SCICR2) . . . . . 154
15.7.4 Data register (SCIDR) . . . . . 155
15.7.5 Baud rate register (SCIBRR) . . . . . 155
15.7.6 Extended receive prescaler division register (SCIERPR) . . . . . 156
15.7.7 Extended transmit prescaler division register (SCIETPR) . . . . . 157

**16 I2C bus interface (I2C) . . . . . 159**

16.1 Introduction . . . . . 159
16.2 Main features . . . . . 159
16.2.1 I2C master features . . . . . 159
16.2.2 I2C slave features . . . . . 159
16.3 General description . . . . . 160
16.3.1 Mode selection . . . . . 160
16.3.2 Communication flow . . . . . 160
16.3.3 SDA/SCL line control . . . . . 161
16.4 Functional description . . . . . 162
16.4.1 Slave mode . . . . . 162
16.4.2 Master mode . . . . . 163
16.5 Low power modes . . . . . 167
16.6 Interrupts . . . . . 167
16.7 Register description . . . . . 168

Obsolete Product(s) - Obsolete Product(s)

16.7.1 I2C control register (CR) . . . 168
16.7.2 I2C status register 1 (SR1) . . . 169
16.7.3 I2C status register 2 (SR2) . . . 171
16.7.4 I2C clock control register (CCR) . . . 172
16.7.5 I2C data register (DR) . . . 173
16.7.6 I2C own address register (OAR1) . . . 173
16.7.7 I2C own address register (OAR2) . . . 174

**17 10-bit A/D converter (ADC) . . . 176**
17.1 Introduction . . . 176
17.2 Main features . . . 176
17.3 Functional description . . . 177
17.3.1 A/D converter configuration . . . 177
17.3.2 Starting the conversion . . . 177
17.3.3 Changing the conversion channel . . . 178
17.4 Low power modes . . . 178
17.5 Interrupts . . . 178
17.6 ADC registers . . . 178
17.6.1 Control/status register (ADCCSR) . . . 178
17.6.2 Data register (ADCDRH) . . . 179
17.6.3 Data register (ADCDRL) . . . 180
17.6.4 ADC register map and reset values . . . 180

**18 Instruction set . . . 181**
18.1 CPU addressing modes . . . 181
18.1.1 Inherent . . . 182
18.1.2 Immediate . . . 183
18.1.3 Direct . . . 183
18.1.4 Indexed (no offset, short, long) . . . 183
18.1.5 Indirect (short, long) . . . 184
18.1.6 Indirect Indexed (Short, Long) . . . 184
18.1.7 Relative (Direct, Indirect) . . . 185
18.2 Instruction groups . . . 185
18.2.1 Using a prebyte . . . 186

**19 Electrical characteristics . . . 189**

Obsolete Product(s) - Obsolete Product(s)

19.1 Parameter conditions . . . 189
19.1.1 Minimum and maximum values . . . 189
19.1.2 Typical values . . . 189
19.1.3 Typical curves . . . 189
19.1.4 Loading capacitor . . . 189
19.1.5 Pin input voltage . . . 189
19.2 Absolute maximum ratings . . . 190
19.2.1 Voltage characteristics . . . 190
19.2.2 Current characteristics . . . 191
19.2.3 Thermal characteristics . . . 191
19.3 Operating conditions . . . 192
19.3.1 General operating conditions . . . 192
19.3.2 Operating conditions with low voltage detector (LVD) . . . 193
19.3.3 Auxiliary voltage detector (AVD) thresholds . . . 193
19.3.4 External voltage detector (EVD) thresholds . . . 194
19.4 Supply current characteristics . . . 195
19.4.1 Current consumption . . . 195
19.4.2 Supply and clock managers . . . 196
19.4.3 On-chip peripherals . . . 197
19.5 Clock and timing characteristics . . . 198
19.5.1 General timings . . . 198
19.5.2 External clock source . . . 198
19.5.3 Crystal and ceramic resonator oscillators . . . 199
19.5.4 RC oscillators . . . 200
19.5.5 Clock security system (CSS) . . . 200
19.5.6 PLL characteristics . . . 201
19.6 Memory characteristics . . . 201
19.6.1 RAM and hardware registers . . . 201
19.6.2 Flash memory . . . 202
19.7 EMC (electromagnetic compatibility) characteristics . . . 203
19.7.1 Functional EMS (electromagnetic susceptibility) . . . 203
19.7.2 EMI (electromagnetic interference) . . . 204
19.7.3 Absolute maximum ratings (electrical sensitivity) . . . 204
19.8 I/O port pin characteristics . . . 206
19.8.1 General characteristics . . . 206
19.8.2 Output driving current . . . 207

Obsolete Product(s) - Obsolete Product(s)

19.9 Control pin characteristics . . . 210
19.9.1 Asynchronous RESET pin . . . 210
19.9.2 ICCSEL/VPP pin . . . 212
19.10 Timer peripheral characteristics . . . 213
19.11 Communication interface characteristics . . . 214
19.11.1 SPI (serial peripheral interface) . . . 214
19.11.2 I2C - inter IC control interface . . . 217
19.12 10-bit ADC characteristics . . . 219
19.12.1 Analog power supply and reference pins . . . 220
19.12.2 General PCB design guidelines . . . 221
19.12.3 ADC accuracy . . . 222

**20 Package characteristics . . . 223**
20.1 Package mechanical data . . . 223
20.2 Thermal characteristics . . . 227
20.3 Ecopack information . . . 227
20.4 Packaging for automatic handling . . . 227

**21 Device configuration and ordering information . . . 228**
21.1 Flash devices . . . 228
21.1.1 Flash configuration . . . 228
21.1.2 Flash ordering information . . . 231
21.2 FASTROM device ordering information and transfer of customer code . 232
21.3 Development tools . . . 234
21.3.1 Introduction . . . 234
21.3.2 Evaluation tools and starter kits . . . 234
21.3.3 Development and debugging tools . . . 234
21.3.4 Programming tools . . . 234
21.3.5 Socket and emulator adapter information . . . 235
21.4 ST7 application notes . . . 235

**22 Known limitations . . . 236**
22.1 All devices . . . 236
22.1.1 Unexpected reset fetch . . . 236
22.1.2 External interrupt missed . . . 236
22.1.3 Clearing active interrupts outside interrupt routine . . . 239

Obsolete Product(s) - Obsolete Product(s)

22.1.4 SCI wrong break duration . . . . . 240
22.1.5 16-bit timer PWM mode . . . . . 240
22.1.6 TIMD set simultaneously with OC interrupt . . . . . 241
22.1.7 I2C multimaster . . . . . 241
22.1.8 Pull-up always active on PE2 . . . . . 241

**23 Revision history . . . . . 242**

Obsolete Product(s) - Obsolete Product(s)

# List of tables


Table 1. Device summary . . . 1
Table 2. Device summary . . . 18
Table 3. Device pin descriptions (LQFP64 and LQFP44) . . . 23
Table 4. Device pin descriptions (LQFP32) . . . 27
Table 5. Hardware register map . . . 31
Table 6. Sectors available in Flash devices . . . 34
Table 7. Flash control/status register address and reset value . . . 37
Table 8. Arithmetic management bits . . . 39
Table 9. Interrupt management bits . . . 40
Table 10. Interrupt software priority selection . . . 40
Table 11. ST7 clock sources . . . 44
Table 12. Effect of low power modes on SI . . . 51
Table 13. CSS AVD interrupt control/wake-up capability . . . 51
Table 14. SICSR description . . . 52
Table 15. Reset source flags . . . 53
Table 16. Interrupt software priority levels . . . 55
Table 17. CPU CC register interrupt bits description . . . 58
Table 18. Interrupt software priority levels . . . 59
Table 19. Interrupt priority bits . . . 59
Table 20. Interrupt dedicated instruction set . . . 60
Table 21. Interrupt mapping . . . 61
Table 22. EICR register description . . . 63
Table 23. Interrupt sensitivity - ei2 (port B3..0) . . . 64
Table 24. Interrupt sensitivity - ei3 (port B4) . . . 64
Table 25. Interrupt sensitivity - ei0 (port A3) . . . 64
Table 26. Interrupt sensitivity - ei1 (port F2..0) . . . 64
Table 27. Nested interrupts register map and reset values . . . 65
Table 28. MCC/RTC low power mode selection . . . 69
Table 29. I/O output mode selection . . . 75
Table 30. I/O port mode options . . . 76
Table 31. I/O port configurations . . . 77
Table 32. I/O port configuration . . . 78
Table 33. Effect of low power modes on I/O ports . . . 79
Table 34. I/O port interrupt control/wake-up capability . . . 79
Table 35. I/O port register map and reset values . . . 79
Table 36. Effect of low power modes on WDG . . . 84
Table 37. WDGCR register description . . . 85
Table 38. Watchdog timer register map and reset values . . . 85
Table 39. Effect of low power modes on MCC/RTC . . . 87
Table 40. MCC/RTC interrupt control/wake-up capability . . . 87
Table 41. MCCSR register description . . . 88
Table 42. Time base selection . . . 89
Table 43. MCCBCR register description . . . 89
Table 44. Beep frequency selection . . . 89
Table 45. Main clock controller register map and reset values . . . 90
Table 46. ARTCSR register description . . . 97
Table 47. Prescaler selection for ART . . . 97
Table 48. ARTCAR register description . . . 98


Obsolete Product(s) - Obsolete Product(s)

Table 49. ARTAAR register description . . . 98
Table 50. PWM frequency versus resolution . . . 99
Table 51. PWMCR register description . . . 99
Table 52. PWM output signal polarity selection . . . 99
Table 53. PWMDCRx register description . . . 100
Table 54. ARTICCSR register description . . . 100
Table 55. ARTICRx register description . . . 101
Table 56. PWM auto-reload timer register map and reset values . . . 101
Table 57. Effect of low power modes on 16-bit timer . . . 116
Table 58. 16-bit timer interrupt control/wake-up capability . . . 116
Table 59. Timer modes . . . 116
Table 60. CR1 register description . . . 117
Table 61. CR2 register description . . . 118
Table 62. Timer clock selection . . . 119
Table 63. CSR register description . . . 120
Table 64. 16-bit timer register map and reset values . . . 124
Table 65. Effect of low power modes on SPI . . . 134
Table 66. SPI interrupt control/wake-up capability . . . 134
Table 67. SPICR register description . . . 135
Table 68. SPI master mode SCK frequency . . . 136
Table 69. SPICSR register description . . . 136
Table 70. SPI register map and reset values . . . 138
Table 71. Frame formats . . . 148
Table 72. Effect of low power modes on SCI . . . 150
Table 73. SCI interrupt control/wake-up capability . . . 150
Table 74. SCISR register description . . . 151
Table 75. SCICR1 register description . . . 153
Table 76. SCICR2 register description . . . 154
Table 77. SCIBRR register description . . . 156
Table 78. SCIERPR register description . . . 157
Table 79. SCIETPR register description . . . 157
Table 80. Baud rate selection . . . 157
Table 81. SCI register map and reset values . . . 158
Table 82. Effect of low power modes on I2C . . . 167
Table 83. I2C interrupt control/wake-up capability . . . 167
Table 84. CR register description . . . 168
Table 85. SR1 register description . . . 169
Table 86. SR2 register description . . . 171
Table 87. CCR register description . . . 172
Table 88. DR register description . . . 173
Table 89. OAR1 register description . . . 174
Table 90. OAR2 register description . . . 174
Table 91. I2C register map and reset values . . . 175
Table 92. Effect of low power modes on ADC . . . 178
Table 93. ADCCSR register description . . . 178
Table 94. ADCDRH register description . . . 179
Table 95. ADCDRL register description . . . 180
Table 96. ADC register map and reset values . . . 180
Table 97. Addressing modes . . . 181
Table 98. CPU addressing mode overview . . . 181
Table 99. Inherent instructions . . . 182
Table 100. Immediate instructions . . . 183

Obsolete Product(s) - Obsolete Product(s)

Table 101. Instructions supporting direct, indexed, indirect, and indirect indexed addressing modes 184
Table 102. Available relative direct/indirect instructions . . . 185
Table 103. Instruction groups . . . 185
Table 104. Instruction set overview . . . 187
Table 105. Voltage characteristics . . . 190
Table 106. Current characteristics . . . 191
Table 107. Thermal characteristics . . . 191
Table 108. General operating conditions . . . 192
Table 109. Operating conditions with low voltage detector (LVD) . . . 193
Table 110. Auxiliary voltage detector (AVD) thresholds . . . 193
Table 111. External voltage detector (EVD) thresholds . . . 194
Table 112. Current consumption . . . 195
Table 113. Oscillators, CSS, PLL and LVD current consumption . . . 196
Table 114. On-chip peripherals current consumption . . . 197
Table 115. General timings . . . 198
Table 116. External clock source . . . 198
Table 117. Crystal and ceramic resonator oscillators . . . 199
Table 118. Resonator characteristics . . . 200
Table 119. RC oscillator characteristics . . . 200
Table 120. CSS characteristics . . . 200
Table 121. PLL characteristics . . . 201
Table 122. RAM supply voltage . . . 201
Table 123. Dual voltage HDFlash memory . . . 202
Table 124. EMS test results . . . 204
Table 125. EMI emissions . . . 204
Table 126. ESD absolute maximum ratings . . . 205
Table 127. Electrical sensitivities . . . 205
Table 128. I/O port pin general characteristics . . . 206
Table 129. Output driving current . . . 207
Table 130. Asynchronous RESET pin characteristics . . . 210
Table 131. ICCSEL/VPP pin characteristics . . . 212
Table 132. 8-bit PWM-ART auto-reload timer characteristics . . . 213
Table 133. 16-bit timer characteristics . . . 213
Table 134. SPI characteristics . . . 214
Table 135. I2C control interface characteristics . . . 217
Table 136. SCL frequency table . . . 218
Table 137. 10-bit ADC characteristics . . . 219
Table 138. ADC accuracy . . . 222
Table 139. 64-pin (14x14) low profile quad flat package mechanical data . . . 223
Table 140. 64-pin (10x10) low profile quad flat package mechanical data . . . 224
Table 141. 44-pin (10x10) low profile quad flat package mechanical data . . . 225
Table 142. 32-pin low profile quad flat package mechanical data . . . 226
Table 143. Thermal characteristics . . . 227
Table 144. Flash option bytes . . . 228
Table 145. Option byte 0 bit description . . . 228
Table 146. Option byte 1 bit description . . . 229
Table 147. Package selection (OPT7) . . . 230
Table 148. STMicroelectronics development tools . . . 235
Table 149. Suggested list of socket types . . . 235
Table 150. Document revision history . . . 242

Obsolete Product(s) Obsolete Product(s)

# List of figures

Figure 1. Device block diagram . . . 19
Figure 2. 64-pin LQFP 14x14 and 10x10 package pinout . . . 20
Figure 3. 44-pin LQFP package pinout . . . 21
Figure 4. 32-pin LQFP package pinout . . . 22
Figure 5. Memory map . . . 30
Figure 6. Memory map and sector address . . . 35
Figure 7. Typical ICC interface . . . 36
Figure 8. CPU registers . . . 38
Figure 9. Stack manipulation example . . . 41
Figure 10. Clock, reset and supply block diagram . . . 42
Figure 11. PLL block diagram . . . 43
Figure 12. Reset block diagram . . . 45
Figure 13. RESET sequence phases . . . 45
Figure 14. RESET sequences . . . 47
Figure 15. Low voltage detector versus reset . . . 48
Figure 16. Using the AVD to monitor VDD (AVDS bit = 0) . . . 49
Figure 17. Using the voltage detector to monitor the EVD pin (AVDS bit = 1) . . . 50
Figure 18. Clock filter function . . . 51
Figure 19. Interrupt processing flowchart . . . 55
Figure 20. Priority decision process flowchart . . . 55
Figure 21. Concurrent interrupt management . . . 57
Figure 22. Nested interrupt management . . . 58
Figure 23. External interrupt control bits . . . 62
Figure 24. Power saving mode transitions . . . 66
Figure 25. Slow mode clock transitions . . . 67
Figure 26. Wait mode flowchart . . . 68
Figure 27. Active Halt timing overview . . . 69
Figure 28. Active Halt mode flowchart . . . 70
Figure 29. Halt timing overview . . . 71
Figure 30. Halt mode flowchart . . . 72
Figure 31. I/O port general block diagram . . . 76
Figure 32. Interrupt I/O port state transitions . . . 78
Figure 33. Watchdog block diagram . . . 82
Figure 34. Approximate timeout duration . . . 82
Figure 35. Exact timeout duration (tmin and tmax) . . . 83
Figure 36. Main clock controller (MCC/RTC) block diagram . . . 87
Figure 37. PWM auto-reload timer block diagram . . . 91
Figure 38. Output compare control . . . 93
Figure 39. PWM auto-reload timer function . . . 94
Figure 40. PWM signal from 0% to 100% duty cycle . . . 94
Figure 41. External event detector example (3 counts) . . . 95
Figure 42. Input capture timing diagram . . . 96
Figure 43. Timer block diagram . . . 104
Figure 44. 16-bit read sequence . . . 105
Figure 45. Counter timing diagram, internal clock divided by 2 . . . 106
Figure 46. Counter timing diagram, internal clock divided by 4 . . . 106
Figure 47. Counter timing diagram, internal clock divided by 8 . . . 106
Figure 48. Input capture block diagram . . . 108

Obsolete Product(s) - Obsolete Product(s)

Figure 49. Input capture timing diagram . . . 108
Figure 50. Output compare block diagram . . . 111
Figure 51. Output compare timing diagram, fTIMER = fCPU/2 . . . 111
Figure 52. Output compare timing diagram, fTIMER = fCPU/4 . . . 111
Figure 53. One pulse mode cycle flowchart . . . 112
Figure 54. One pulse mode timing example . . . 113
Figure 55. Pulse width modulation mode timing example with 2 output compare functions . . . 114
Figure 56. Pulse width modulation cycle flowchart . . . 115
Figure 57. Serial peripheral interface block diagram . . . 126
Figure 58. Single master/single slave application . . . 127
Figure 59. Generic SS timing diagram . . . 128
Figure 60. Hardware/Software slave select management . . . 128
Figure 61. Data clock timing diagram . . . 131
Figure 62. Clearing the WCOL bit (Write Collision Flag) software sequence . . . 133
Figure 63. Single master / multiple slave configuration . . . 133
Figure 64. SCI block diagram . . . 141
Figure 65. Word length programming . . . 142
Figure 66. SCI baud rate and extended prescaler block diagram . . . 146
Figure 67. Bit sampling in reception mode . . . 150
Figure 68. I2C bus protocol . . . 160
Figure 69. I2C interface block diagram . . . 161
Figure 70. Transfer sequencing . . . 166
Figure 71. Interrupt control logic diagram . . . 167
Figure 72. ADC block diagram . . . 176
Figure 73. Pin loading conditions . . . 189
Figure 74. Pin input voltage . . . 189
Figure 75. fCPU max versus $V_{DD}$ . . . 192
Figure 76. Typical application with an external clock source . . . 198
Figure 77. Typical application with a crystal or ceramic resonator . . . 199
Figure 78. Typical fOSC(RCINT) versus TA . . . 200
Figure 79. Integrated PLL jitter versus signal frequency(1) . . . 201
Figure 80. Unused I/O pins configured as input . . . 207
Figure 81. Typical IPU vs VDD with VIN = VSS . . . 207
Figure 82. Typical VOL at VDD = 5V (standard) . . . 208
Figure 83. Typical VOL at VDD = 5V (high-sink) . . . 208
Figure 84. Typical VOH at VDD = 5V . . . 208
Figure 85. Typical VOL versus VDD (standard) . . . 209
Figure 86. Typical VOL versus VDD (high-sink) . . . 209
Figure 87. Typical VDD-VOH versus VDD . . . 209
Figure 88. RESET pin protection when LVD is enabled . . . 211
Figure 89. RESET pin protection when LVD is disabled . . . 212
Figure 90. Two typical applications with ICCSEL/VPP pin(1) . . . 212
Figure 91. SPI slave timing diagram with CPHA = 0(1) . . . 215
Figure 92. SPI slave timing diagram with CPHA = 1(1) . . . 215
Figure 93. SPI master timing diagram(1) . . . 216
Figure 94. Typical application with I2C BUS and timing diagram(1) . . . 218
Figure 95. RAIN maximum versus fADC with CAIN = 0pF(1) . . . 220
Figure 96. Recommended CAIN and RAIN values(1) . . . 220
Figure 97. Typical A/D converter application . . . 220
Figure 98. Power supply filtering . . . 221
Figure 99. ADC error classification . . . 222
Figure 100. 64-pin (14x14) low profile quad flat package outline . . . 223

Obsolete Product(s) - Obsolete Product(s)

Figure 101. 64-pin (10x10) low profile quad flat package outline . . . . . 224
Figure 102. 44-pin (10x10) low profile quad flat package outline . . . . . 225
Figure 103. 32-pin low profile quad flat package outline . . . . . 226
Figure 104. pin 1 orientation in tape and reel conditioning . . . . . 227
Figure 105. ST72F325xxx-Auto Flash commercial product structure . . . . . 231
Figure 106. ST72P325xxx-Auto FastROM commercial product structure . . . . . 232

Obsolete Product(s) - Obsolete Product(s)

# 1 Description

The ST72325-Auto devices are members of the ST7 microcontroller family designed for mid-range automotive applications running from 3.8 to 5.5V. Different package options offer up to 48 I/O pins.

They are derivatives of the ST72321 and ST72324 devices, with enhanced characteristics and robust Clock Security System.

All devices are based on a common industry-standard 8-bit core, featuring an enhanced instruction set and are available with Flash program memory. The ST7 family architecture offers both power and flexibility to software developers, enabling the design of highly efficient and compact application code.

The on-chip peripherals include an A/D converter, a PWM autoreload timer, two general purpose timers, $I^2C$, SPI and SCI interfaces.

For power economy, the microcontroller can switch dynamically into Wait, Slow, Active Halt or Halt mode when the application is in idle or standby state.

**Table 2. Device summary**

| Reference | Program memory | RAM (stack) | Voltage range | Temp. range | Package |
|---|---|---|---|---|---|
| ST72325K4-Auto | Flash 16 Kbytes | 512 (256) bytes | 3.8 to 5.5V | up to -40 to 125°C | LQFP32 |
| ST72325K6-Auto | Flash 32 Kbytes | 1024 (256) bytes | | | |
| ST72325J4-Auto | Flash 16 Kbytes | 512 (256) bytes | | | LQFP44 |
| ST72325J6-Auto | Flash 32 Kbytes | 1024 (256) bytes | | | |
| ST72325J7-Auto | Flash 48 Kbytes | 1536 (256) bytes | | | |
| ST72325J9-Auto | Flash 60 Kbytes | 2048 (256) bytes | | | |
| ST72325AR6-Auto | Flash 32 Kbytes | 1024 (256) bytes | | | LQFP64 10x10 |
| ST72325AR7-Auto | Flash 48 Kbytes | 1536 (256) bytes | | | |
| ST72325AR9-Auto | Flash 60 Kbytes | 2048 (256) bytes | | | |
| ST72325R6-Auto | Flash 32 Kbytes | 1024 (256) bytes | 3.8 to 5.5V | up to -40 to 125°C | LQFP64 14x14 |
| ST72325R7-Auto | Flash 48 Kbytes | 1536 (256) bytes | | | |
| ST72325R9-Auto | Flash 60 Kbytes | 2048 (256) bytes | | | |

Typical applications include

- all types of car body applications such as window lift, DC motor control, rain sensors
- safety microcontroller in airbag and engine management applications
- auxiliary functions in car radios

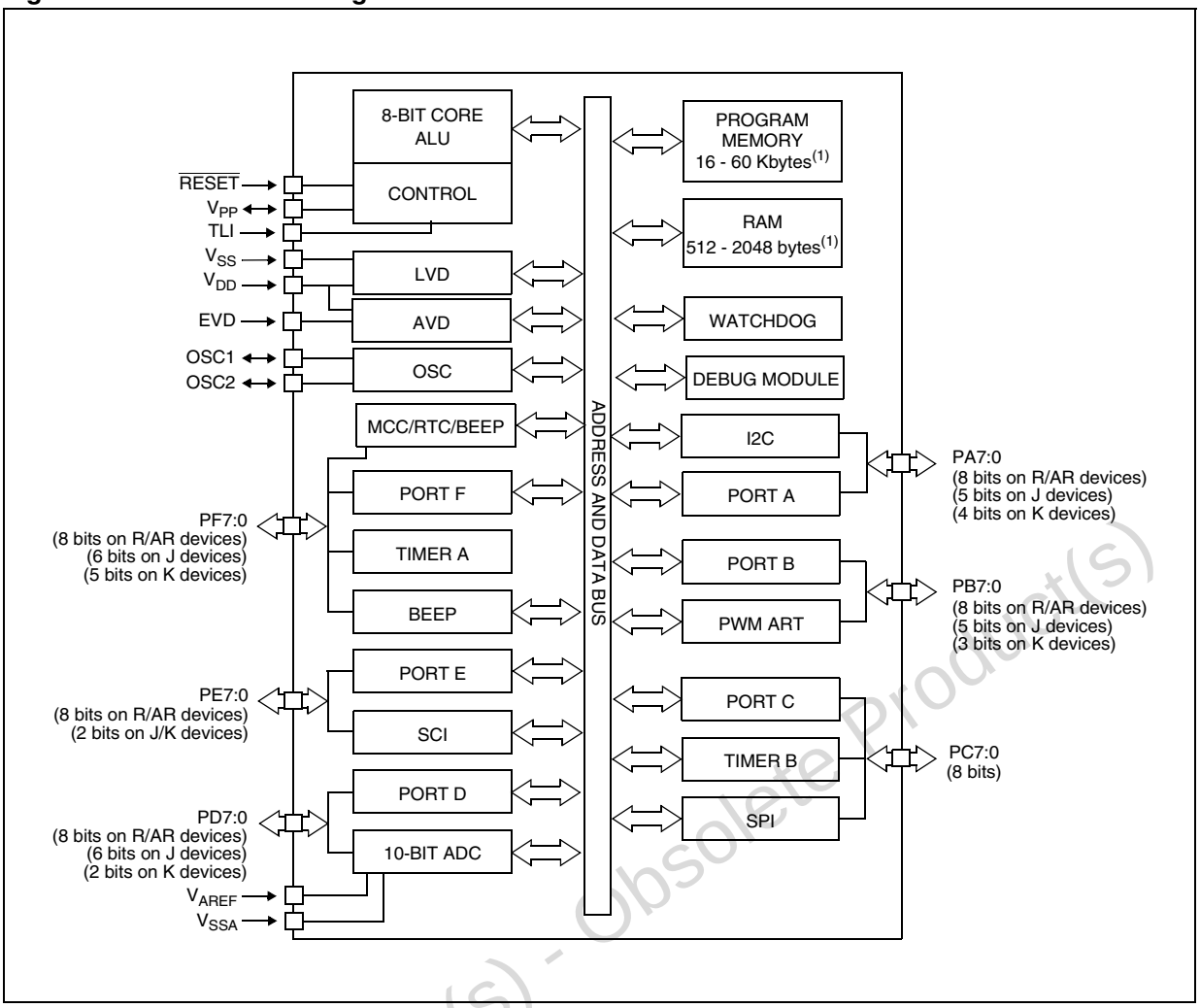
The devices feature an on-chip Debug Module (DM) to support in-circuit debugging (ICD). For a description of the DM registers, refer to the *ST7 ICC Protocol Reference Manual*.

Main differences with ST72321:

- LQFP32 package
- Clock Security System
- Internal RC, readout protection, LVD and PLL without limitations
- Negative current injection not allowed on I/O port PB0 (instead of PC6)
- External interrupts have Exit from Active Halt mode capability

Obsolete Product(s) - Obsolete Product(s)

**Figure 1. Device block diagram**



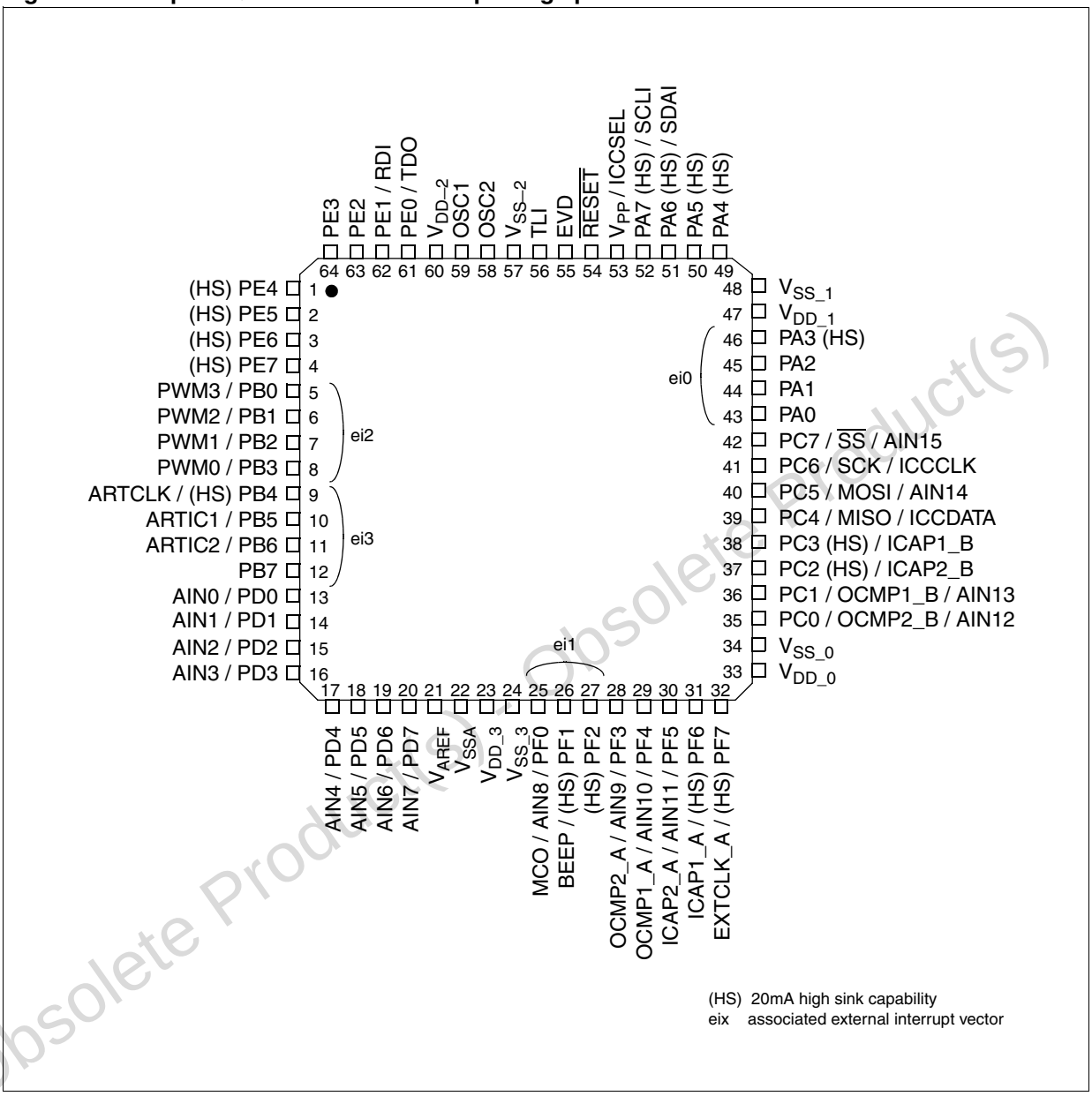

1. On certain devices only (see *Device summary*)

Obsolete Product(s) - Obsolete Product(s)

# 2 Package pinout and pin description

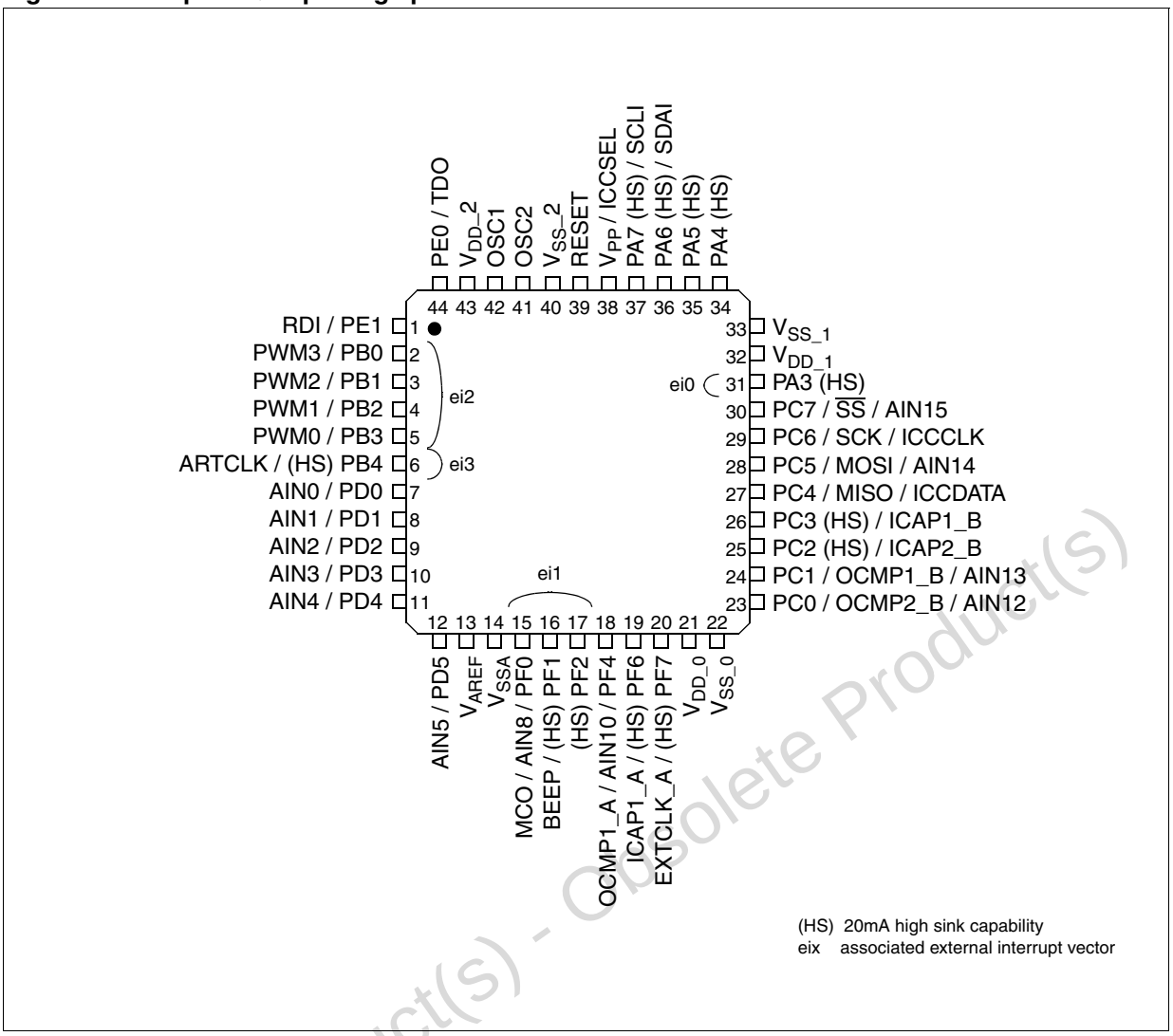
## 2.1 Package pinout

**Figure 2. 64-pin LQFP 14x14 and 10x10 package pinout**



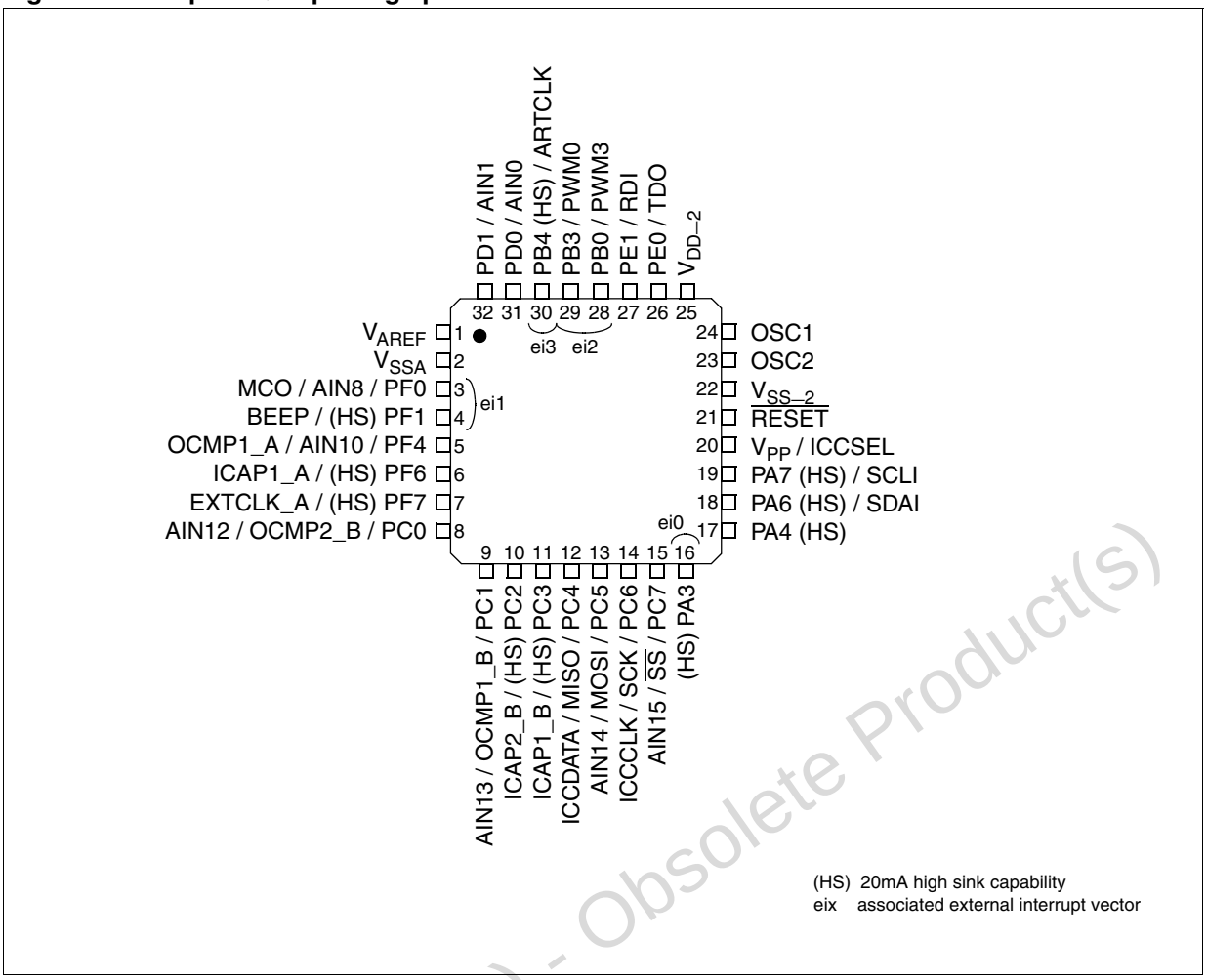

Obsolete Product(s) Obsolete Product(s)

**Figure 3. 44-pin LQFP package pinout**




Obsolete Product(s) - Obsolete Product(s)

**Figure 4. 32-pin LQFP package pinout**




For external pin connection guidelines, refer to *Section 19: Electrical characteristics*.

Obsolete Product(s) - Obsolete Product(s)

## 2.2 Pin description

In the device pin description table, the RESET configuration of each pin is shown in bold. This configuration is valid as long as the device is in reset state.

Refer to *Section 9: I/O ports* for more details on the software configuration of the I/O ports.

**Table 3. Device pin descriptions (LQFP64 and LQFP44)**

| Pin | | | Type | Level | | Port | | | | | | Main function (after reset) | Alternate function |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| No. | | Name | | Input | Output | Input | | | | Output | | | |
| LQFP64 | LQFP44 | | | | | float | wpu | int | ana | OD | PP | | |
| 1 | - | PE4 (HS) | I/O | $C_T$ | HS | **X** | X | | | X | X | Port E4 | |
| 2 | - | PE5 (HS) | I/O | $C_T$ | HS | **X** | X | | | X | X | Port E5 | |
| 3 | - | PE6 (HS) | I/O | $C_T$ | HS | **X** | X | | | X | X | Port E6 | |
| 4 | - | PE7 (HS) | I/O | $C_T$ | HS | **X** | X | | | X | X | Port E7 | |
| 5 | 2 | PB0/PWM3 | I/O | $C_T$ | | **X** | ei2 | | | X | X | Port B0 | PWM Output 3<br>**Caution:** Negative current injection not allowed on this pin |
| 6 | 3 | PB1/PWM2 | I/O | $C_T$ | | **X** | ei2 | | | X | X | Port B1 | PWM Output 2 |
| 7 | 4 | PB2/PWM1 | I/O | $C_T$ | | **X** | ei2 | | | X | X | Port B2 | PWM Output 1 |
| 8 | 5 | PB3/PWM0 | I/O | $C_T$ | | **X** | | ei2 | | X | X | Port B3 | PWM Output 0 |
| 9 | 6 | PB4 (HS)/ARTCLK | I/O | $C_T$ | HS | **X** | ei3 | | | X | X | Port B4 | PWM-ART External Clock |
| 10 | -(1) | PB5 / ARTIC1 | I/O | $C_T$ | | **X** | ei3 | | | X | X | Port B5 | PWM-ART Input Capture 1 |
| 11 | -(1) | PB6 / ARTIC2 | I/O | $C_T$ | | **X** | ei3 | | | X | X | Port B6 | PWM-ART Input Capture 2 |
| 12 | -(1) | PB7 | I/O | $C_T$ | | **X** | | ei3 | | X | X | Port B7 | |
| 13 | 7 | PD0/AIN0 | I/O | $C_T$ | | **X** | X | | X | X | X | Port D0 | ADC Analog Input 0 |
| 14 | 8 | PD1/AIN1 | I/O | $C_T$ | | **X** | X | | X | X | X | Port D1 | ADC Analog Input 1 |
| 15 | 9 | PD2/AIN2 | I/O | $C_T$ | | **X** | X | | X | X | X | Port D2 | ADC Analog Input 2 |
| 16 | 10 | PD3/AIN3 | I/O | $C_T$ | | **X** | X | | X | X | X | Port D3 | ADC Analog Input 3 |
| 17 | 11 | PD4/AIN4 | I/O | $C_T$ | | **X** | X | | X | X | X | Port D4 | ADC Analog Input 4 |
| 18 | 12 | PD5/AIN5 | I/O | $C_T$ | | **X** | X | | X | X | X | Port D5 | ADC Analog Input 5 |
| 19 | -(1) | PD6/AIN6 | I/O | $C_T$ | | **X** | X | | X | X | X | Port D6 | ADC Analog Input 6 |
| 20 | -(1) | PD7/AIN7 | I/O | $C_T$ | | **X** | X | | X | X | X | Port D7 | ADC Analog Input 7 |
| 21 | 13 | $V_{AREF}$(2) | I | | | | | | | | | Analog Reference Voltage for ADC | |
| 22 | 14 | $V_{SSA}$(2) | S | | | | | | | | | Analog Ground Voltage | |

Obsolete Product(s) - Obsolete Product(s)

**Table 3. Device pin descriptions (LQFP64 and LQFP44) (continued)**

| Pin | | | Type | Level | | Port | | | | | | Main function (after reset) | Alternate function | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| No. | | Name | | Input | Output | Input | | | | Output | | | | |
| LQFP64 | LQFP44 | | | | | float | wpu | int | ana | OD | PP | | | |
| 23 | - | $V_{DD_3}$(2) | S | | | | | | | | | Digital Main Supply Voltage | | |
| 24 | - | $V_{SS_3}$(2) | S | | | | | | | | | Digital Ground Voltage | | |
| 25 | 15 | PF0/MCO/AIN8 | I/O | $C_T$ | | X | ei1 | | X | X | X | Port F0 | Main clock out ($f_{OSC}$/2) | ADC Analog Input 8 |
| 26 | 16 | PF1 (HS)/BEEP | I/O | $C_T$ | HS | X | ei1 | | | X | X | Port F1 | Beep signal output | |
| 27 | 17 | PF2 (HS) | I/O | $C_T$ | HS | X | | ei1 | | X | X | Port F2 | | |
| 28 | -(1) | PF3/OCMP2_A/ AIN9 | I/O | $C_T$ | | X | X | | X | X | X | Port F3 | Timer A Output Compare 2 | ADC Analog Input 9 |
| 29 | 18 | PF4/OCMP1_A/ AIN10 | I/O | $C_T$ | | X | X | | X | X | X | Port F4 | Timer A Output Compare 1 | ADC Analog Input 10 |
| 30 | -(1) | PF5/ICAP2_A/AIN11 | I/O | $C_T$ | | X | X | | X | X | X | Port F5 | Timer A Input Capture 2 | ADC Analog Input 11 |
| 31 | 19 | PF6 (HS)/ICAP1_A | I/O | $C_T$ | HS | X | X | | | X | X | Port F6 | Timer A Input Capture 1 | |
| 32 | 20 | PF7 (HS)/EXTCLK_A | I/O | $C_T$ | HS | X | X | | | X | X | Port F7 | Timer A External Clock Source | |
| 33 | 21 | $V_{DD_0}$(2) | S | | | | | | | | | Digital Main Supply Voltage | | |
| 34 | 22 | $V_{SS_0}$(2) | S | | | | | | | | | Digital Ground Voltage | | |
| 35 | 23 | PC0/OCMP2_B/ AIN12 | I/O | $C_T$ | | X | X | | X | X | X | Port C0 | Timer B Output Compare 2 | ADC Analog Input 12 |
| 36 | 24 | PC1/OCMP1_B/ AIN13 | I/O | $C_T$ | | X | X | | X | X | X | Port C1 | Timer B Output Compare 1 | ADC Analog Input 13 |
| 37 | 25 | PC2 (HS)/ICAP2_B | I/O | $C_T$ | HS | X | X | | | X | X | Port C2 | Timer B Input Capture 2 | |
| 38 | 26 | PC3 (HS)/ICAP1_B | I/O | $C_T$ | HS | X | X | | | X | X | Port C3 | Timer B Input Capture 1 | |
| 39 | 27 | PC4/MISO/ICCDATA | I/O | $C_T$ | | X | X | | | X | X | Port C4 | SPI Master In / Slave Out Data | ICC Data Input |
| 40 | 28 | PC5/MOSI/AIN14 | I/O | $C_T$ | | X | X | | X | X | X | Port C5 | SPI Master Out / Slave In Data | ADC Analog Input 14 |
| 41 | 29 | PC6/SCK/ICCCLK | I/O | $C_T$ | | X | X | | | X | X | Port C6 | SPI Serial Clock | ICC Clock Output |

Obsolete Product(s) - Obsolete Product(s)

**Table 3. Device pin descriptions (LQFP64 and LQFP44) (continued)**

| Pin | | | Type | Level | | Port | | | | | | Main function (after reset) | Alternate function | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| No. | | Name | | Input | Output | Input | | | | Output | | | | |
| LQFP64 | LQFP44 | | | | | float | wpu | int | ana | OD | PP | | | |
| 42 | 30 | PC7/$\overline{SS}$/AIN15 | I/O | $C_T$ | | **X** | X | | X | X | X | Port C7 | SPI Slave Select (active low) | ADC analog Input 15 |
| 43 | -(1) | PA0 | I/O | $C_T$ | | **X** | ei0 | | | X | X | Port A0 | | |
| 44 | -(1) | PA1 | I/O | $C_T$ | | **X** | ei0 | | | X | X | Port A1 | | |
| 45 | -(1) | PA2 | I/O | $C_T$ | | **X** | ei0 | | | X | X | Port A2 | | |
| 46 | 31 | PA3 (HS) | I/O | $C_T$ | HS | **X** | | ei0 | | X | X | Port A3 | | |
| 47 | 32 | $V_{DD_1}$(2) | S | | | | | | | | | Digital Main Supply Voltage | | |
| 48 | 33 | $V_{SS_1}$(2) | S | | | | | | | | | Digital Ground Voltage | | |
| 49 | 34 | PA4 (HS) | I/O | $C_T$ | HS | **X** | X | | | X | X | Port A4 | | |
| 50 | 35 | PA5 (HS) | I/O | $C_T$ | HS | **X** | X | | | X | X | Port A5 | | |
| 51 | 36 | PA6 (HS)/SDAI | I/O | $C_T$ | HS | **X** | | | | T | | Port A6 | $I^2C$ Data | |
| 52 | 37 | PA7 (HS)/SCLI | I/O | $C_T$ | HS | **X** | | | | T | | Port A7 | $I^2C$ Clock | |
| 53 | 38 | $V_{PP}$/ ICCSEL | I | | | | | | | | | Must be tied low. In Flash programming mode, this pin acts as the programming voltage input $V_{PP}$. See *Section 19.9.2* for more details. | | |
| 54 | 39 | RESET | I/O | $C_T$ | | | | | | | | Top priority non maskable interrupt. | | |
| 55 | - | EVD | I | A | | | | | | | | External voltage detector | | |
| 56 | - | TLI | I | $C_T$ | | | | X | | | | Top level interrupt input pin | | |
| 57 | 40 | $V_{SS_2}$(2) | S | | | | | | | | | Digital Ground Voltage | | |
| 58 | 41 | OSC2(3) | I/O | | | | | | | | | Resonator oscillator inverter output | | |
| 59 | 42 | OSC1(3) | I | | | | | | | | | External clock input or Resonator oscillator inverter input | | |
| 60 | 43 | $V_{DD_2}$(2) | S | | | | | | | | | Digital Main Supply Voltage | | |
| 61 | 44 | PE0/TDO | I/O | $C_T$ | | **X** | X | | | X | X | Port E0 | SCI Transmit Data Out | |
| 62 | 1 | PE1/RDI | I/O | $C_T$ | | **X** | X | | | X | X | Port E1 | SCI Receive Data In | |
| 63 | -(1) | PE2 | I/O | $C_T$ | | **X** | X | | | X(4) | X(4) | Port E2 | | |
| 64 | -(1) | PE3 | I/O | $C_T$ | | **X** | X | | | X | X | Port E3 | | |

1. On the chip, each I/O port may have up to eight pads. Pads that are not bonded to external pins are in input pull-up configuration after reset. The configuration of these pads must be kept at reset state to avoid added current consumption.
2. It is mandatory to connect all available $V_{DD}$ and $V_{AREF}$ pins to the supply voltage and all $V_{SS}$ and $V_{SSA}$ pins to ground.
3. OSC1 and OSC2 pins connect a crystal/ceramic resonator, or an external source to the on-chip oscillator; see *Section 1: Description* and *Section 19.5: Clock and timing characteristics* for more details.
4. Pull-up always activated on PE2 (see *Section 22.1.8: Pull-up always active on PE2*).

Obsolete Product(s) - Obsolete Product(s)

**Legend / Abbreviations** for *Table 3*:

Type: I = input
O = output
S = supply

Input level: A = dedicated analog input

In/Output level: C = CMOS $0.3V_{DD}/0.7V_{DD}$
$C_T$ = CMOS $0.3V_{DD}/0.7V_{DD}$ with input trigger

Output level: HS = 20mA high sink (on N-buffer only)

Port and control configuration:

- Input: float = floating
  wpu = weak pull-up
  int = interrupt[a]
  ana = analog
- Output: OD = open-drain[b]
  PP = push-pull

Obsolete Product(s) - Obsolete Product(s)

a. In the interrupt input column, “eiX” defines the associated external interrupt vector. If the weak pull-up column (wpu) is merged with the interrupt column (int), then the I/O configuration is pull-up interrupt input, otherwise the configuration is floating interrupt input.

b. In the open-drain output column, “T” defines a true open-drain I/O (P-Buffer and protection diode to $V_{DD}$ are not implemented). See *Section 9: I/O ports* and *Section 19.8: I/O port pin characteristics* for more details.

**Table 4. Device pin descriptions (LQFP32)**

| Pin | | Type | Level | | Port | | | | | | Main function (after reset) | Alternate function | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Input | | | | Output | | | | |
| No. | Name | | Input | Output | float | wpu | int | ana | OD | PP | | | |
| 1 | $V_{AREF}$(1) | I | | | | | | | | | Analog Reference Voltage for ADC | | |
| 2 | $V_{SSA}$(1) | S | | | | | | | | | Analog Ground Voltage | | |
| 3 | PF0/MCO/AIN8 | I/O | $C_T$ | | **X** | ei1 | | X | X | X | Port F0 | Main clock out ($f_{OSC}$/2) | ADC Analog Input 8 |
| 4 | PF1 (HS)/BEEP | I/O | $C_T$ | HS | **X** | ei1 | | | X | X | Port F1 | Beep signal output | |
| 5 | PF4/OCMP1_A/AIN10 | I/O | $C_T$ | | **X** | X | | X | X | X | Port F4 | Timer A Output Compare 1 | ADC Analog Input 10 |
| 6 | PF6 (HS)/ICAP1_A | I/O | $C_T$ | HS | **X** | X | | | X | X | Port F6 | Timer A Input Capture 1 | |
| 7 | PF7 (HS)/EXTCLK_A | I/O | $C_T$ | HS | **X** | X | | | X | X | Port F7 | Timer A External Clock Source | |
| 8 | PC0/OCMP2_B/AIN12 | I/O | $C_T$ | | **X** | X | | X | X | X | Port C0 | Timer B Output Compare 2 | ADC Analog Input 12 |
| 9 | PC1/OCMP1_B/AIN13 | I/O | $C_T$ | | **X** | X | | X | X | X | Port C1 | Timer B Output Compare 1 | ADC Analog Input 13 |
| 10 | PC2 (HS)/ICAP2_B | I/O | $C_T$ | HS | **X** | X | | | X | X | Port C2 | Timer B Input Capture 2 | |
| 11 | PC3 (HS)/ICAP1_B | I/O | $C_T$ | HS | **X** | X | | | X | X | Port C3 | Timer B Input Capture 1 | |
| 12 | PC4/MISO/ICCDATA | I/O | $C_T$ | | **X** | X | | | X | X | Port C4 | SPI Master In / Slave Out Data | ICC Data Input |
| 13 | PC5/MOSI/AIN14 | I/O | $C_T$ | | **X** | X | | X | X | X | Port C5 | SPI Master Out / Slave In Data | ADC Analog Input 14 |
| 14 | PC6/SCK/ICCCLK | I/O | $C_T$ | | **X** | X | | | X | X | Port C6 | SPI Serial Clock | ICC Clock Output |
| 15 | PC7/$\overline{SS}$/AIN15 | I/O | $C_T$ | | **X** | X | | X | X | X | Port C7 | SPI Slave Select (active low) | ADC Analog Input 15 |
| 16 | PA3 (HS) | I/O | $C_T$ | HS | **X** | | ei 0 | | X | X | Port A3 | | |
| 17 | PA4 (HS) | I/O | $C_T$ | HS | **X** | X | | | X | X | Port A4 | | |
| 18 | PA6 (HS)/SDAI | I/O | $C_T$ | HS | **X** | | | | T | | Port A6 | $I^2C$ Data | |
| 19 | PA7 (HS)/SCLI | I/O | $C_T$ | HS | **X** | | | | T | | Port A7 | $I^2C$ Clock | |
| 20 | $V_{PP}$/ ICCSEL | I | | | | | | | | | Must be tied low. In Flash programming mode, this pin acts as the programming voltage input $V_{PP}$. See *Section 19.9.2: ICCSEL/VPP pin*for more details. | | |

Obsolete Product(s) - Obsolete Product(s)

**Table 4. Device pin descriptions (LQFP32) (continued)**

| Pin | | Type | Level | | Port | | | | | | Main function (after reset) | Alternate function |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Input | | | | Output | | | |
| No. | Name | | Input | Output | float | wpu | int | ana | OD | PP | | |
| 21 | RESET | I/O | $C_T$ | | | | | | | | Top priority non maskable interrupt | |
| 22 | $V_{SS_2}$ (1) | S | | | | | | | | | Digital Ground Voltage | |
| 23 | OSC2 (2) | I/O | | | | | | | | | Resonator oscillator inverter output | |
| 24 | OSC1 (2) | I | | | | | | | | | External clock input or Resonator oscillator inverter input | |
| 25 | $V_{DD_2}$ (1) | S | | | | | | | | | Digital Main Supply Voltage | |
| 26 | PE0/TDO | I/O | $C_T$ | | **X** | X | | | X | X | Port E0 | SCI Transmit Data Out |
| 27 | PE1/RDI | I/O | $C_T$ | | **X** | X | | | X | X | Port E1 | SCI Receive Data In |
| 28 | PB0/PWM3 | I/O | $C_T$ | | **X** | ei2 | | | X | X | Port B0 | PWM Output 3<br>**Caution:** Negative current injection not allowed on this pin |
| 29 | PB3/PWM0 | I/O | $C_T$ | | **X** | | ei2 | | X | X | Port B3 | PWM Output 0 |
| 30 | PB4 (HS)/ARTCLK | I/O | $C_T$ | HS | **X** | ei3 | | | X | X | Port B4 | PWM-ART External Clock |
| 31 | PD0/AIN0 | I/O | $C_T$ | | **X** | X | | X | X | X | Port D0 | ADC Analog Input 0 |
| 32 | PD1/AIN1 | I/O | $C_T$ | | **X** | X | | X | X | X | Port D1 | ADC Analog Input 1 |

1. It is mandatory to connect all available $V_{DD}$ and $V_{AREF}$ pins to the supply voltage and all $V_{SS}$ and $V_{SSA}$ pins to ground.
2. OSC1 and OSC2 pins connect a crystal/ceramic resonator, or an external source to the on-chip oscillator; see *Section 1: Description* and *Section 19.5: Clock and timing characteristics* for more details.

Obsolete Product(s) - Obsolete Product(s)

**Legend / Abbreviations** for *Table 4*:

Type: I = input
O = output
S = supply

Input level: A = Dedicated analog input

In/Output level: C = CMOS $0.3V_{DD}/0.7V_{DD}$
$C_T$ = CMOS $0.3V_{DD}/0.7V_{DD}$ with input trigger

Output level: HS = 20mA high sink (on N-buffer only)

Port and control configuration:

- Input: float = floating
  wpu = weak pull-up
  int = interrupt[a]
  ana = analog
- Output: OD = open-drain[b]
  PP = push-pull

Obsolete Product(s) - Obsolete Product(s)

a. In the interrupt input column, “eiX” defines the associated external interrupt vector. If the weak pull-up column (wpu) is merged with the interrupt column (int), then the I/O configuration is pull-up interrupt input, else the configuration is floating interrupt input.

b. In the open-drain output column, “T” defines a true open-drain I/O (P-Buffer and protection diode to $V_{DD}$ are not implemented). See *Section 9: I/O ports* and *Section 19.8: I/O port pin characteristics*for more details.

# 3 Register and memory map

As shown in *Figure 5*, the MCU is capable of addressing 64 Kbytes of memories and I/O registers.

The available memory locations consist of 128 bytes of register locations, up to 2 Kbytes of RAM and up to 60 Kbytes of user program memory. The RAM space includes up to 256 bytes for the stack from 0100h to 01FFh.

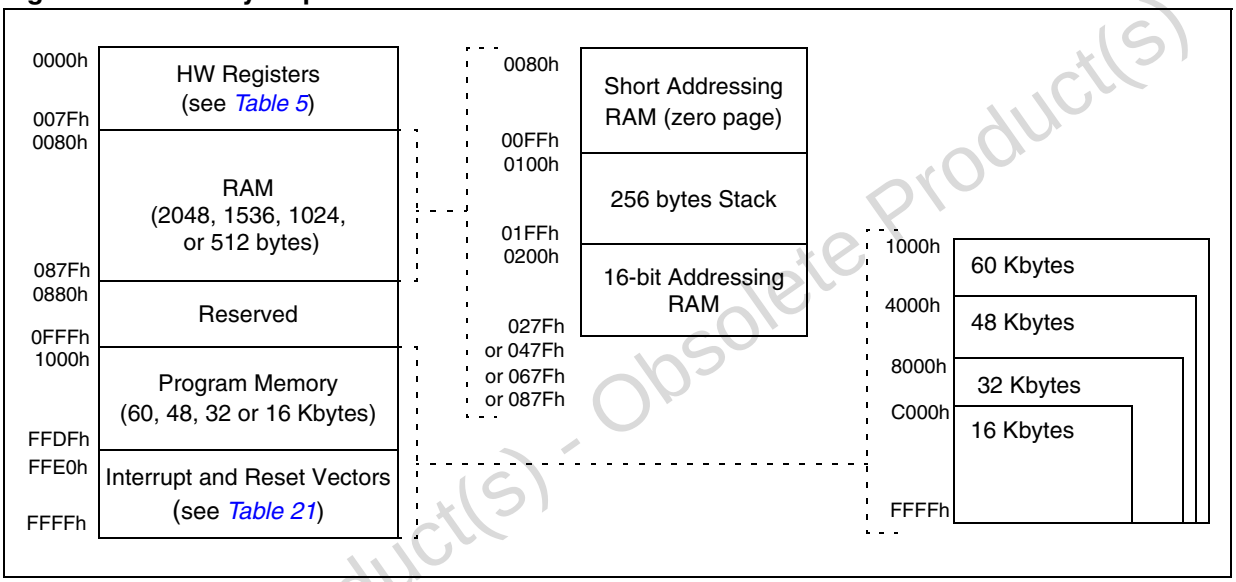
The highest address bytes contain the user reset and interrupt vectors.

**Caution:** Memory locations marked as “Reserved” must never be accessed. Accessing a reserved area can have unpredictable effects on the device.

**Related documentation**

*Executing Code in ST7 RAM (*AN 985)

**Figure 5. Memory map**




Obsolete Product(s) - Obsolete Product(s)

**Table 5. Hardware register map**

| Address | Block | Register label | Register name | Reset status | Remarks |
|---|---|---|---|---|---|
| 0000h<br>0001h<br>0002h | Port A | PADR<br>PADDR<br>PAOR | Port A Data Register<br>Port A Data Direction Register<br>Port A Option Register | 00h(1)<br>00h<br>00h | R/W<br>R/W<br>R/W |
| 0003h<br>0004h<br>0005h | Port B | PBDR<br>PBDDR<br>PBOR | Port B Data Register<br>Port B Data Direction Register<br>Port B Option Register | 00h(1)<br>00h<br>00h | R/W<br>R/W<br>R/W |
| 0006h<br>0007h<br>0008h | Port C | PCDR<br>PCDDR<br>PCOR | Port C Data Register<br>Port C Data Direction Register<br>Port C Option Register | 00h(1)<br>00h<br>00h | R/W<br>R/W<br>R/W |
| 0009h<br>000Ah<br>000Bh | Port D | PDDR<br>PDDDR<br>PDOR | Port D Data Register<br>Port D Data Direction Register<br>Port D Option Register | 00h(1)<br>00h<br>00h | R/W<br>R/W<br>R/W |
| 000Ch<br>000Dh<br>000Eh | Port E | PEDR<br>PEDDR<br>PEOR | Port E Data Register<br>Port E Data Direction Register<br>Port E Option Register | 00h(1)<br>00h<br>00h | R/W<br>R/W(2)<br>R/W(2) |
| 000Fh<br>0010h<br>0011h | Port F | PFDR<br>PFDDR<br>PFOR | Port F Data Register<br>Port F Data Direction Register<br>Port F Option Register | 00h(1)<br>00h<br>00h | R/W<br>R/W<br>R/W |
| 0018h<br>0019h<br>001Ah<br>001Bh<br>001Ch<br>001Dh<br>001Eh | $I^2C$ | I2CCR<br>I2CSR1<br>I2CSR2<br>I2CCCR<br>I2COAR1<br>I2COAR2<br>I2CDR | $I^2C$ Control Register<br>$I^2C$ Status Register 1<br>$I^2C$ Status Register 2<br>$I^2C$ Clock Control Register<br>$I^2C$ Own Address Register 1<br>$I^2C$ Own Address Register2<br>$I^2C$ Data Register | 00h<br>00h<br>00h<br>00h<br>00h<br>00h<br>00h | R/W<br>Read only<br>Read only<br>R/W<br>R/W<br>R/W<br>R/W |
| 001Fh<br>0020h | Reserved area (2 bytes) | | | | |
| 0021h<br>0022h<br>0023h | SPI | SPIDR<br>SPICR<br>SPICSR | SPI Data I/O Register<br>SPI Control Register<br>SPI Control/Status Register | xxh<br>0xh<br>00h | R/W<br>R/W<br>R/W |
| 0024h<br>0025h<br>0026h<br>0027h | ITC | ISPR0<br>ISPR1<br>ISPR2<br>ISPR3 | Interrupt Software Priority Register 0<br>Interrupt Software Priority Register 1<br>Interrupt Software Priority Register 2<br>Interrupt Software Priority Register 3 | FFh<br>FFh<br>FFh<br>FFh | R/W<br>R/W<br>R/W<br>R/W |
| 0028h | | EICR | External Interrupt Control Register | 00h | R/W |
| 0029h | FLASH | FCSR | Flash Control/Status Register | 00h | R/W |
| 002Ah | WATCHDOG | WDGCR | Watchdog Control Register | 7Fh | R/W |
| 002Bh | | SICSR | System Integrity Control/Status Register | 000x 000x b | R/W |
| 002Ch<br>002Dh | MCC | MCCSR<br>MCCBCR | Main Clock Control/Status Register<br>Main Clock Controller/Beep Control Register | 00h<br>00h | R/W<br>R/W |
| 002Eh<br>to<br>0030h | Reserved area (3 bytes) | | | | |

Obsolete Product(s) - Obsolete Products(s)

**Table 5. Hardware register map (continued)**

| Address | Block | Register label | Register name | Reset status | Remarks |
|---|---|---|---|---|---|
| 0031h | TIMER A | TACR2 | Timer A Control Register 2 | 00h | R/W |
| 0032h | | TACR1 | Timer A Control Register 1 | 00h | R/W |
| 0033h | | TACSR | Timer A Control/Status Register | xxxx x0xxb | R/W |
| 0034h | | TAIC1HR | Timer A Input Capture 1 High Register | xxh | Read only |
| 0035h | | TAIC1LR | Timer A Input Capture 1 Low Register | xxh | Read only |
| 0036h | | TAOC1HR | Timer A Output Compare 1 High Register | 80h | R/W |
| 0037h | | TAOC1LR | Timer A Output Compare 1 Low Register | 00h | R/W |
| 0038h | | TACHR | Timer A Counter High Register | FFh | Read only |
| 0039h | | TACLR | Timer A Counter Low Register | FCh | Read only |
| 003Ah | | TAACHR | Timer A Alternate Counter High Register | FFh | Read only |
| 003Bh | | TAACLR | Timer A Alternate Counter Low Register | FCh | Read only |
| 003Ch | | TAIC2HR | Timer A Input Capture 2 High Register | xxh | Read only |
| 003Dh | | TAIC2LR | Timer A Input Capture 2 Low Register | xxh | Read only |
| 003Eh | | TAOC2HR | Timer A Output Compare 2 High Register | 80h | R/W |
| 003Fh | | TAOC2LR | Timer A Output Compare 2 Low Register | 00h | R/W |
| 0040h | Reserved area (1 byte) | | | | |
| 0041h | TIMER B | TBCR2 | Timer B Control Register 2 | 00h | R/W |
| 0042h | | TBCR1 | Timer B Control Register 1 | 00h | R/W |
| 0043h | | TBCSR | Timer B Control/Status Register | xxxx x0xxb | R/W |
| 0044h | | TBIC1HR | Timer B Input Capture 1 High Register | xxh | Read only |
| 0045h | | TBIC1LR | Timer B Input Capture 1 Low Register | xxh | Read only |
| 0046h | | TBOC1HR | Timer B Output Compare 1 High Register | 80h | R/W |
| 0047h | | TBOC1LR | Timer B Output Compare 1 Low Register | 00h | R/W |
| 0048h | | TBCHR | Timer B Counter High Register | FFh | Read only |
| 0049h | | TBCLR | Timer B Counter Low Register | FCh | Read only |
| 004Ah | | TBACHR | Timer B Alternate Counter High Register | FFh | Read only |
| 004Bh | | TBACLR | Timer B Alternate Counter Low Register | FCh | Read only |
| 004Ch | | TBIC2HR | Timer B Input Capture 2 High Register | xxh | Read only |
| 004Dh | | TBIC2LR | Timer B Input Capture 2 Low Register | xxh | Read only |
| 004Eh | | TBOC2HR | Timer B Output Compare 2 High Register | 80h | R/W |
| 004Fh | | TBOC2LR | Timer B Output Compare 2 Low Register | 00h | R/W |
| 0050h | SCI | SCISR | SCI Status Register | C0h | Read only |
| 0051h | | SCIDR | SCI Data Register | xxh | R/W |
| 0052h | | SCIBRR | SCI Baud Rate Register | 00h | R/W |
| 0053h | | SCICR1 | SCI Control Register 1 | x000 0000b | R/W |
| 0054h | | SCICR2 | SCI Control Register 2 | 00h | R/W |
| 0055h | | SCIERPR | SCI Extended Receive Prescaler Register | 00h | R/W |
| 0056h | | | Reserved area | --- | |
| 0057h | | SCIETPR | SCI Extended Transmit Prescaler Register | 00h | R/W |
| 0058h | DM(3) | DMCR | DM Control Register | 00h | R/W |
| 0059h | | DMSR | DM Status Register | 00h | R/W |
| 005Ah | | DMBK1H | DM Breakpoint Register 1 High | 00h | R/W |
| 005Bh | | DMBK1L | DM Breakpoint Register 1 Low | 00h | R/W |
| 005Ch | | DMBK2H | DM Breakpoint Register 2 High | 00h | R/W |
| 005Dh | | DMBK2L | DM Breakpoint Register 2 Low | 00h | R/W |
| 005Eh to 006Fh | Reserved Area (18 bytes) | | | | |

Obsolete Product(s) - Obsolete Product(s)

**Table 5. Hardware register map (continued)**

| Address | Block | Register label | Register name | Reset status | Remarks |
|---|---|---|---|---|---|
| 0070h | ADC | ADCCSR | Control/Status Register | 00h | R/W |
| 0071h | | ADCDRH | Data High Register | 00h | Read only |
| 0072h | | ADCDRL | Data Low Register | 00h | Read only |
| 0073h | PWM ART | PWMDCR3 | PWM AR Timer Duty Cycle Register 3 | 00h | R/W |
| 0074h | | PWMDCR2 | PWM AR Timer Duty Cycle Register 2 | 00h | R/W |
| 0075h | | PWMDCR1 | PWM AR Timer Duty Cycle Register 1 | 00h | R/W |
| 0076h | | PWMDCR0 | PWM AR Timer Duty Cycle Register 0 | 00h | R/W |
| 0077h | | PWMCR | PWM AR Timer Control Register | 00h | R/W |
| 0078h | | ARTCSR | Auto-Reload Timer Control/Status Register | 00h | R/W |
| 0079h | | ARTCAR | Auto-Reload Timer Counter Access Register | 00h | R/W |
| 007Ah | | ARTARR | Auto-Reload Timer Auto-Reload Register | 00h | R/W |
| 007Bh | | ARTICCSR | AR Timer Input Capture Control/Status Reg. | 00h | R/W |
| 007Ch | | ARTICR1 | AR Timer Input Capture Register 1 | 00h | Read only |
| 007Dh | | ARTICR2 | AR Timer Input Capture Register 1 | 00h | Read only |
| 007Eh<br>007Fh | Reserved area (2 bytes) | | | | |

1. The contents of the I/O port DR registers are readable only in output configuration. In input configuration, the values of the I/O pins are returned instead of the DR register contents.
2. The bits associated with unavailable pins must always keep their reset value.
3. For a description of the Debug Module registers, see the *ST7 ICC Protocol Reference Manual*.

Legend: x = undefined, R/W = read/write

Obsolete Product(s) - Obsolete Product(s)

# 4 Flash program memory

## 4.1 Introduction

The ST7 dual voltage High Density Flash (HDFlash) is a non-volatile memory that can be electrically erased as a single block or by individual sectors and programmed on a byte-by-byte basis using an external $V_{PP}$ supply.

The HDFlash devices can be programmed and erased off-board (plugged in a programming tool) or on-board using ICP (in-circuit programming) or IAP (in-application programming).

The array matrix organization allows each sector to be erased and reprogrammed without affecting other sectors.

## 4.2 Main features

- 3 Flash programming modes:
  - Insertion in a programming tool. In this mode, all sectors including option bytes can be programmed or erased.
  - ICP (in-circuit programming). In this mode, all sectors including option bytes can be programmed or erased without removing the device from the application board.
  - IAP (in-application programming). In this mode, all sectors except Sector 0 can be programmed or erased without removing the device from the application board and while the application is running.
- ICT (in-circuit testing) for downloading and executing user application test patterns in RAM
- Readout protection
- Register Access Security System (RASS) to prevent accidental programming or erasing

## 4.3 Structure

The Flash memory is organized in sectors and can be used for both code and data storage.

Depending on the overall Flash memory size in the microcontroller device, there are up to three user sectors (see *Table 6*). Each of these sectors can be erased independently to avoid unnecessary erasing of the whole Flash memory when only a partial erasing is required.

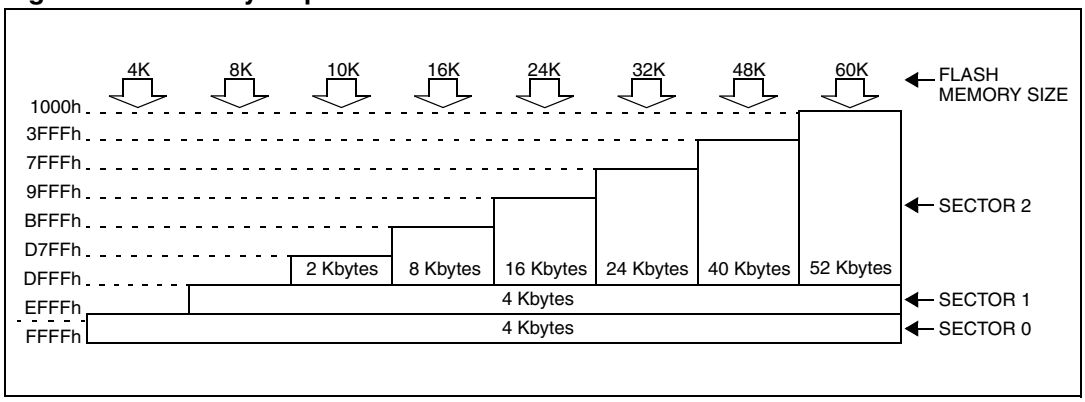
**Table 6. Sectors available in Flash devices**

| Flash size (bytes) | Available sectors |
|---|---|
| 4K | Sector 0 |
| 8K | Sectors 0, 1 |
| > 8K | Sectors 0, 1, 2 |

The first two sectors have a fixed size of 4 Kbytes (see *Figure 6*). They are mapped in the upper part of the ST7 addressing space so the reset and interrupt vectors are located in Sector 0 (F000h-FFFFh).

Obsolete Product(s) - Obsolete Product(s)

**Figure 6. Memory map and sector address**




### 4.3.1 Readout protection

Readout protection, when selected, provides a protection against program memory content extraction and against write access to Flash memory. Even if no protection can be considered as totally unbreakable, the feature provides a very high level of protection for a general purpose microcontroller.

In Flash devices, this protection is removed by reprogramming the option. In this case, the entire program memory is first automatically erased and the device can be reprogrammed.

Note: Readout protection selection is enabled and removed through the FMP_R bit in the option byte.

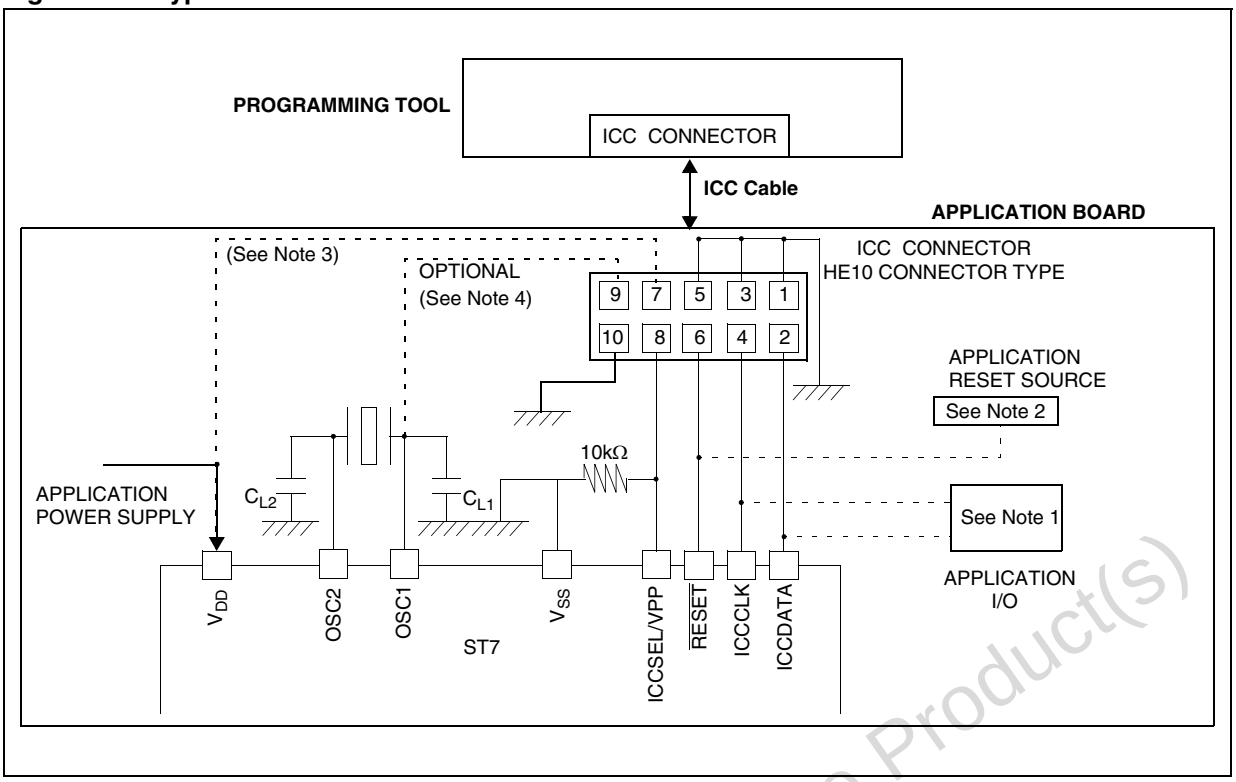
## 4.4 ICC interface

ICC needs a minimum of 4 and up to 6 pins to be connected to the programming tool (see *Figure 7*). These pins are:

- $\overline{\text{RESET}}$: device reset
- $V_{SS}$: device power supply ground
- ICCCLK: ICC output serial clock pin
- ICCDATA: ICC input/output serial data pin
- ICCSEL/$V_{PP}$: programming voltage
- OSC1 (or OSCIN): main clock input for external source (optional)
- $V_{DD}$: application board power supply (optional, see *Figure 7*, *Note 3*)

Obsolete Product(s) - Obsolete Product(s)

**Figure 7. Typical ICC interface**




1. If the ICCCLK or ICCDATA pins are only used as outputs in the application, no signal isolation is necessary. As soon as the programming tool is plugged to the board, even if an ICC session is not in progress, the ICCCLK and ICCDATA pins are not available for the application. If they are used as inputs by the application, isolation such as a serial resistor has to implemented in case another device forces the signal. Refer to the programming tool documentation for recommended resistor values.
2. During the ICC session, the programming tool must control the RESET pin. This can lead to conflicts between the programming tool and the application reset circuit if it drives more than 5mA at high level (push-pull output or pull-up resistor < 1K). A schottky diode can be used to isolate the application RESET circuit in this case. When using a classical RC network with R > 1K or a reset management IC with open-drain output and pull-up resistor > 1K, no additional components are needed. In all cases the user must ensure that no external reset is generated by the application during the ICC session.
3. The use of Pin 7 of the ICC connector depends on the programming tool architecture. This pin must be connected when using most ST programming tools (it is used to monitor the application power supply). Please refer to the programming tool manual.
4. Pin 9 has to be connected to the OSC1 or OSCIN pin of the ST7 when the clock is not available in the application or if the selected clock option is not programmed in the option byte. ST7 devices with multi-oscillator capability need to have OSC2 grounded in this case.

## 4.5 ICP (in-circuit programming)

To perform ICP the microcontroller must be switched to ICC (in-circuit communication) mode by an external controller or programming tool.

Depending on the ICP code downloaded in RAM, Flash memory programming can be fully customized (number of bytes to program, program locations, or selection serial communication interface for downloading).

When using an STMicroelectronics or third-party programming tool that supports ICP and the specific microcontroller device, the user needs only to implement the ICP hardware interface on the application board (see *Figure 7*). For more details on the pin locations, refer to the device pinout description.

Obsolete Product(s) - Obsolete Product(s)

## 4.6 IAP (in-application programming)

This mode uses a BootLoader program previously stored in Sector 0 by the user (in ICP mode or by plugging the device in a programming tool).

This mode is fully controlled by user software. This allows it to be adapted to the user application, (such as user-defined strategy for entering programming mode, choice of communications protocol used to fetch the data to be stored). For example, it is possible to download code from the serial peripheral or serial communication interface and program it in the Flash. IAP mode can be used to program any of the Flash sectors except Sector 0, which is write/erase protected to allow recovery in case errors occur during the programming operation.

## 4.7 Related documentation

For details on Flash programming and ICC protocol, refer to the *ST7 Flash Programming Reference Manual* and to the *ST7 ICC Protocol Reference Manual*.

## 4.8 Flash control/status register (FCSR)

FSCR Reset value: 0000 0000 (00h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| RW | RW | RW | RW | RW | RW | RW | RW |

This register is reserved for use by programming tool software. It controls the Flash programming and erasing operations.

**Table 7. Flash control/status register address and reset value**

| Address (Hex.) | Register label | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0029h | FCSR<br>Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Obsolete Product(s) - Obsolete Product(s)

# 5 Central processing unit (CPU)

## 5.1 Introduction

This CPU has a full 8-bit architecture and contains six internal registers allowing efficient 8-bit data manipulation.

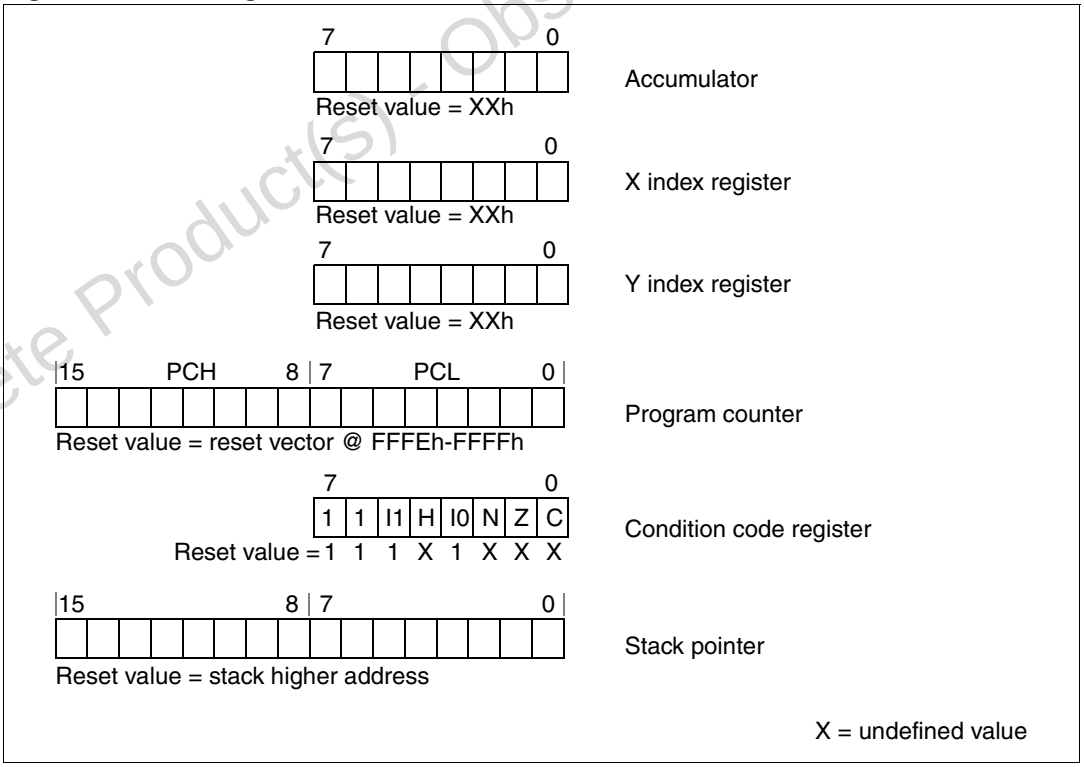
## 5.2 Main features

- Enable executing 63 basic instructions
- Fast 8-bit by 8-bit multiply
- 17 main addressing modes (with indirect addressing mode)
- Two 8-bit index registers
- 16-bit stack pointer
- Low power Halt and Wait modes
- Priority maskable hardware interrupts
- Non-maskable software/hardware interrupts

## 5.3 CPU registers

The six CPU registers shown in *Figure 8* are not present in the memory mapping and are accessed by specific instructions.

**Figure 8. CPU registers**




Obsolete Product(s) - Obsolete Product(s)

### 5.3.1 Accumulator (A)

The accumulator is an 8-bit general purpose register used to hold operands and the results of the arithmetic and logic calculations as well as data manipulations.

### 5.3.2 Index registers (X and Y)

These 8-bit registers are used to create effective addresses or as temporary storage areas for data manipulation (the Cross-Assembler generates a precede instruction (PRE) to indicate that the following instruction refers to the Y register.)

The Y register is not affected by the interrupt automatic procedures.

### 5.3.3 Program counter (PC)

The program counter is a 16-bit register containing the address of the next instruction to be executed by the CPU. It is made of two 8-bit registers PCL (Program Counter Low which is the LSB) and PCH (Program Counter High which is the MSB).

### 5.3.4 Condition code (CC) register

The 8-bit condition code register contains the interrupt masks and four flags representative of the result of the instruction just executed. This register can also be handled by the PUSH and POP instructions.

These bits can be individually tested and/or controlled by specific instructions.

CC Reset value: 111x1xxx

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | I1 | H | I0 | N | Z | C |
| | | RW | RW | RW | RW | RW | RW |

**Table 8. Arithmetic management bits**

| Bit | Name | Function |
|---|---|---|
| 4 | H | *Half carry*<br>This bit is set by hardware when a carry occurs between bits 3 and 4 of the ALU during an ADD or ADC instructions. It is reset by hardware during the same instructions.<br>0: No half carry has occurred.<br>1: A half carry has occurred.<br>This bit is tested using the JRH or JRNH instruction. The H bit is useful in BCD arithmetic subroutines. |
| 2 | N | *Negative*<br>This bit is set and cleared by hardware. It is representative of the result sign of the last arithmetic, logical or data manipulation. It is a copy of the result 7th bit.<br>0: The result of the last operation is positive or null.<br>1: The result of the last operation is negative (that is, the most significant bit is a logic 1).<br>This bit is accessed by the JRMI and JRPL instructions. |

Obsolete Product(s) - Obsolete Product(s)

**Table 8. Arithmetic management bits (continued)**

| Bit | Name | Function |
|---|---|---|
| 1 | Z | *Zero*<br>This bit is set and cleared by hardware. This bit indicates that the result of the last arithmetic, logical or data manipulation is zero.<br>0: The result of the last operation is different from zero.<br>1: The result of the last operation is zero.<br>This bit is accessed by the JREQ and JRNE test instructions. |
| 0 | C | *Carry/borrow*<br>This bit is set and cleared by hardware and software. It indicates an overflow or an underflow has occurred during the last arithmetic operation.<br>0: No overflow or underflow has occurred.<br>1: An overflow or underflow has occurred.<br>This bit is driven by the SCF and RCF instructions and tested by the JRC and JRNC instructions. It is also affected by the “bit test and branch”, shift and rotate instructions. |

**Table 9. Interrupt management bits**

| Bit | Name | Function |
|---|---|---|
| 5 | I1 | *Interrupt Software Priority 1*<br>The combination of the I1 and I0 bits gives the current interrupt software priority. |
| 3 | I0 | *Interrupt Software Priority 0*<br>The combination of the I1 and I0 bits gives the current interrupt software priority. |

**Table 10. Interrupt software priority selection**

| 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 |

These two bits are set/cleared by hardware when entering in interrupt. The loaded value is given by the corresponding bits in the interrupt software priority registers (ISPRx). They can be also set/cleared by software with the RIM, SIM, IRET, HALT, WFI and PUSH/POP instructions.

See *Section 7: Interrupts* for more details.

### 5.3.5 Stack pointer (SP) register

SP Reset value: 01 FFh

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | SP7 | SP6 | SP5 | SP4 | SP3 | SP2 | SP1 | SP0 |
| | | | | | | | | RW | RW | RW | RW | RW | RW | RW | RW |

Obsolete Product(s) - Obsolete Product(s)

The stack pointer is a 16-bit register which is always pointing to the next free location in the stack. It is then decremented after data has been pushed onto the stack and incremented before data is popped from the stack (see *Figure 9*).

Since the stack is 256 bytes deep, the 8 most significant bits are forced by hardware. Following an MCU Reset, or after a reset stack pointer instruction (RSP), the stack pointer contains its reset value (the SP7 to SP0 bits are set) which is the stack higher address.

The least significant byte of the stack pointer (called S) can be directly accessed by an LD instruction.

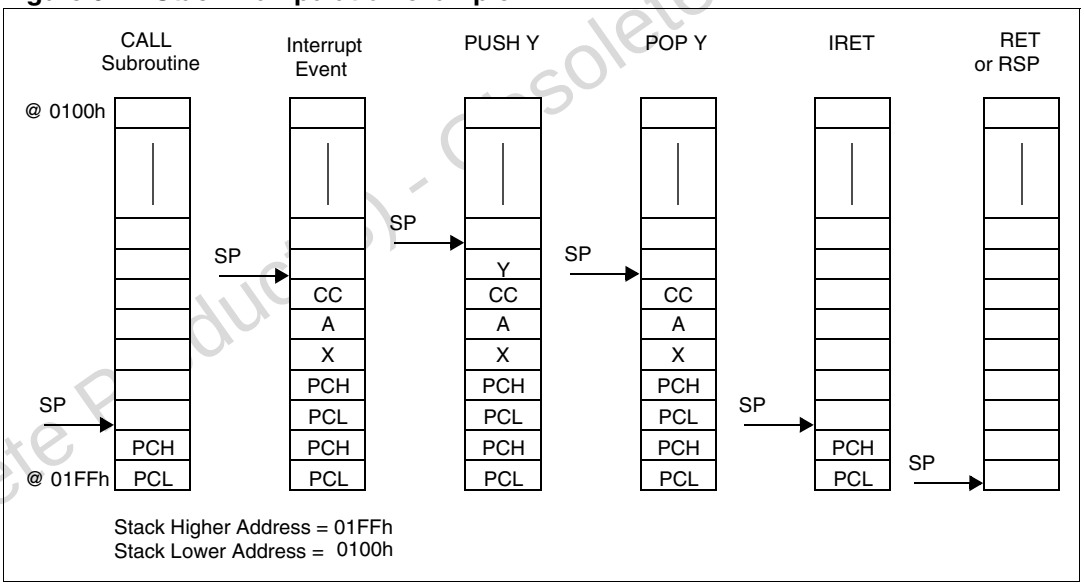
*Note: When the lower limit is exceeded, the stack pointer wraps around to the stack upper limit, without indicating the stack overflow. The previously stored information is then overwritten and therefore lost. The stack also wraps in case of an underflow.*

The stack is used to save the return address during a subroutine call and the CPU context during an interrupt. The user may also directly manipulate the stack by means of the PUSH and POP instructions. In the case of an interrupt, the PCL is stored at the first location pointed to by the SP. The other registers are then stored in the next locations as shown in *Figure 9*.

- When an interrupt is received, the SP is decremented and the context is pushed on the stack.
- On return from interrupt, the SP is incremented and the context is popped from the stack.

A subroutine call occupies two locations and an interrupt five locations in the stack area.

**Figure 9. Stack manipulation example**




Obsolete Product(s) - Obsolete Product(s)

# 6 Supply, reset and clock management

## 6.1 Introduction

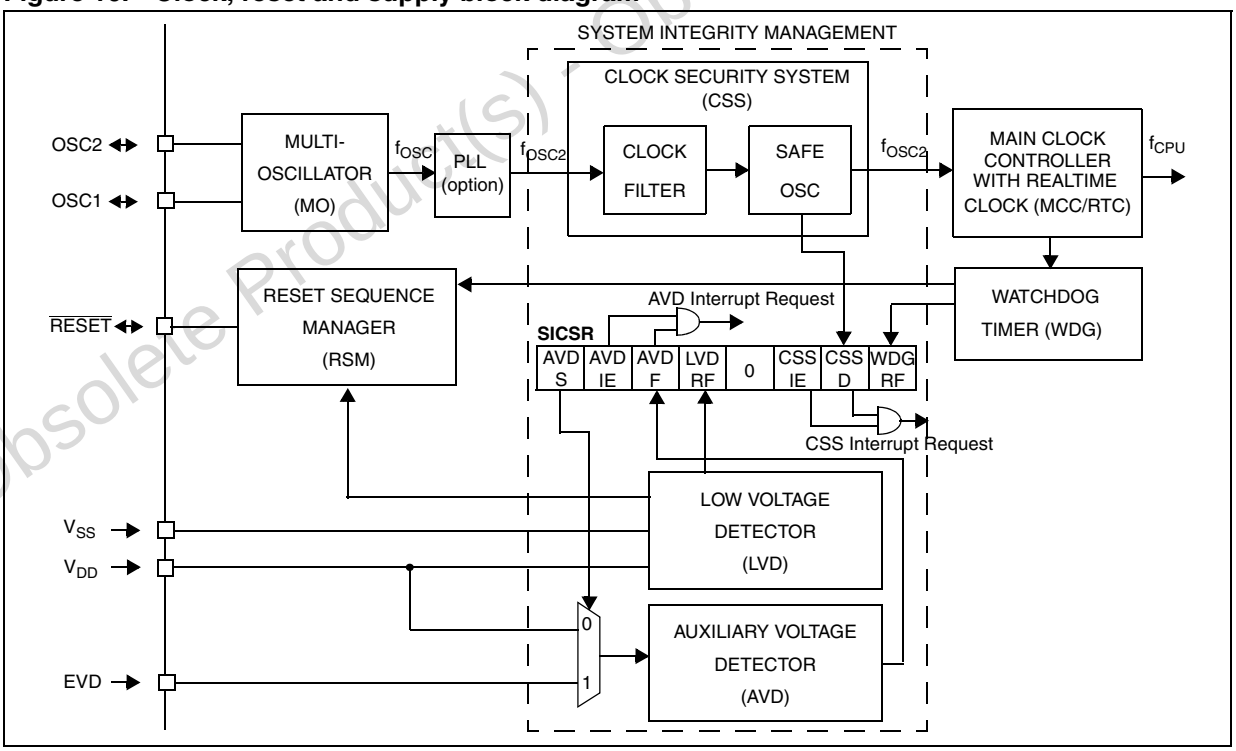
The device includes a range of utility features for securing the application in critical situations (for example in case of a power brown-out), and reducing the number of external components. An overview is shown in *Figure 10*.

For more details, refer to the dedicated parametric section.

## 6.2 Main features

- Optional PLL for multiplying the frequency by 2 (not to be used with internal RC oscillator)
- Reset Sequence Manager (RSM)
- Multi-oscillator Clock Management (MO)
  - 5 crystal/ceramic resonator oscillators
  - 1 internal RC oscillator
- System Integrity Management (SI)
  - Main supply low voltage detection (LVD)
  - Auxiliary voltage detector (AVD) with interrupt capability for monitoring the main supply
  - Clock security system (CSS) with clock filter and backup safe oscillator (enabled by option byte)

**Figure 10. Clock, reset and supply block diagram**



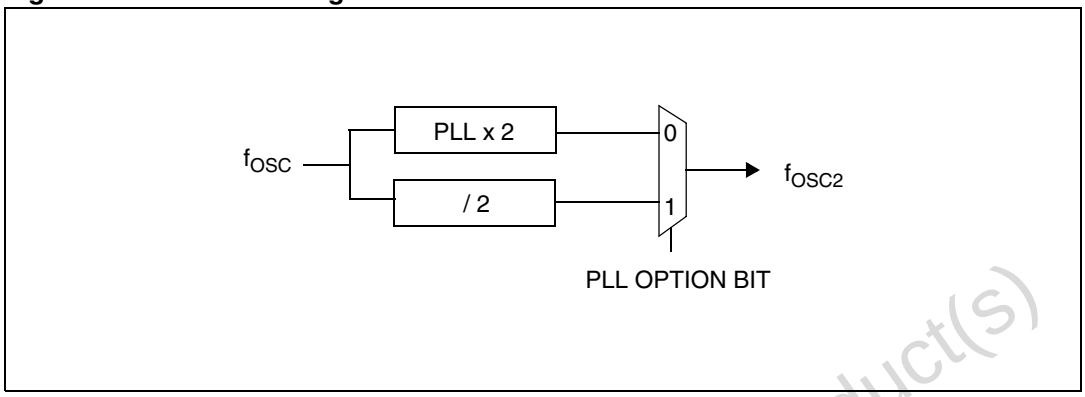

Obsolete Product(s) - Obsolete Product(s)

## 6.3 Phase locked loop

If the clock frequency input to the PLL is in the range 2 to 4 MHz, the PLL can be used to multiply the frequency by two to obtain an $f_{OSC2}$ of 4 to 8 MHz. The PLL is enabled by option byte. If the PLL is disabled, then $f_{OSC2} = f_{OSC}/2$.

**Caution:** The PLL is not recommended for applications where timing accuracy is required (see *Section 19.5.6: PLL characteristics*).

**Figure 11. PLL block diagram**




## 6.4 Multi-oscillator (MO)

The main clock of the ST7 can be generated by three different source types coming from the multi-oscillator block:

- an external source
- 4 crystal or ceramic resonator oscillators
- an internal high frequency RC oscillator

Each oscillator is optimized for a given frequency range in terms of consumption and is selectable through the option byte. The associated hardware configurations are shown in *Table 11*. Refer to *Section 19: Electrical characteristics* for more details.

### External clock source

In this external clock mode, a clock signal (square, sinus or triangle) with ~50% duty cycle has to drive the OSC1 pin while the OSC2 pin is tied to ground.

### Crystal/ceramic oscillators

This family of oscillators has the advantage of producing a very accurate rate on the main clock of the ST7. The selection within a list of four oscillators with different frequency ranges has to be done by option byte in order to reduce consumption (refer to *Section 21.1.1: Flash configuration* for more details on the frequency ranges). In this mode of the multi-oscillator, the resonator and the load capacitors have to be placed as close as possible to the oscillator pins in order to minimize output distortion and start-up stabilization time. The loading capacitance values must be adjusted according to the selected oscillator.

These oscillators are not stopped during the RESET phase to avoid losing time in the oscillator start-up phase.

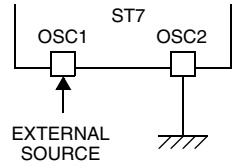
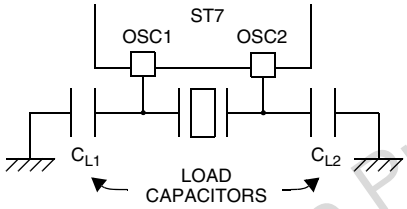
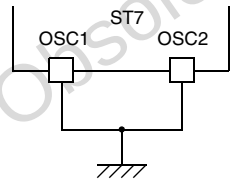
Obsolete Product(s) - Obsolete Product(s)

### Internal RC oscillator

This oscillator allows a low cost solution for the main clock of the ST7 using only an internal resistor and capacitor. Internal RC oscillator mode has the drawback of a lower frequency accuracy and should not be used in applications that require accurate timing.

In this mode, the two oscillator pins have to be tied to ground.

**Table 11. ST7 clock sources**

| | Hardware configuration |
|---|---|
| External clock |  |
| Crystal/Ceramic resonators |  |
| Internal RC oscillator |  |

## 6.5 Reset sequence manager (RSM)

### 6.5.1 Introduction

The reset sequence manager includes three RESET sources as shown in *Figure 12*:

- External $\overline{\text{RESET}}$ source pulse
- Internal LVD RESET (low voltage detection)
- Internal WATCHDOG RESET

These sources act on the $\overline{\text{RESET}}$ pin and it is always kept low during the delay phase.

The RESET service routine vector is fixed at addresses FFFEh-FFFFh in the ST7 memory map.

Obsolete Product(s) - Obsolete Product(s)

The basic RESET sequence consists of three phases as shown in *Figure 13*:

- Active phase depending on the RESET source
- 256 or 4096 CPU clock cycle delay (selected by option byte)
- RESET vector fetch

**Caution:** When the ST7 is unprogrammed or fully erased, the Flash is blank and the RESET vector is not programmed. For this reason, it is recommended to keep the $\overline{\text{RESET}}$ pin in low state until programming mode is entered, in order to avoid unwanted behavior.

The 256 or 4096 CPU clock cycle delay allows the oscillator to stabilize and ensures that recovery has taken place from the Reset state. The shorter or longer clock cycle delay should be selected by option byte to correspond to the stabilization time of the external oscillator used in the application (see *Section 21.1.1: Flash configuration*).

The RESET vector fetch phase duration is 2 clock cycles.

**Figure 12. Reset block diagram**

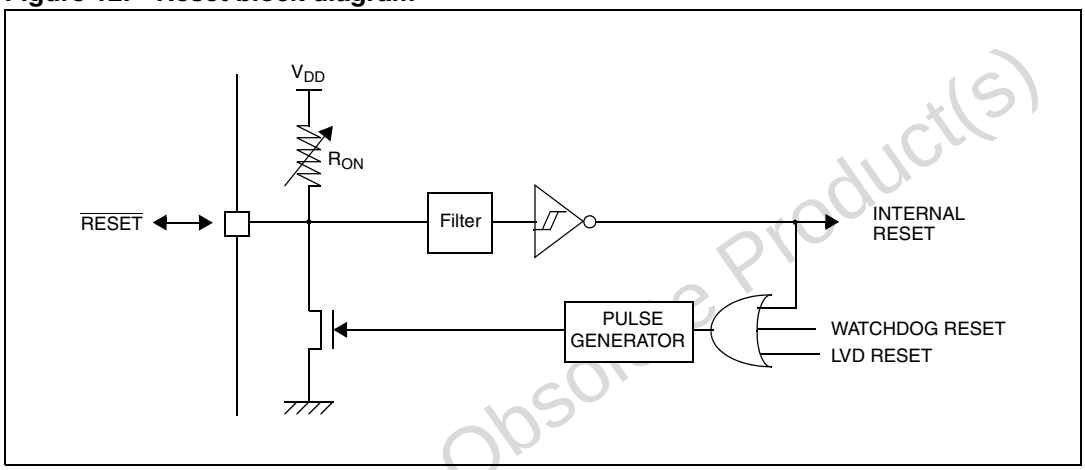
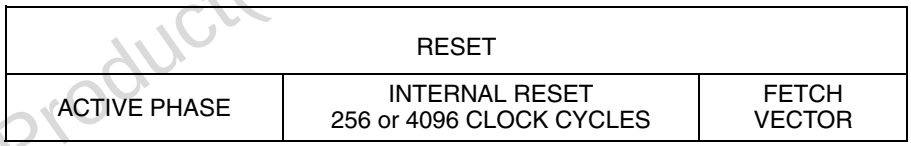



**Figure 13. RESET sequence phases**




## 6.5.2 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. See *Section 19.9: Control pin 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 14*). This detection is asynchronous and therefore the MCU can enter reset state even in Halt mode.

Obsolete Product(s) - Obsolete Product(s)

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 19: Electrical characteristics*.

If the external $\overline{\text{RESET}}$ pulse is shorter than $t_{w(RSTL)out}$ (see short ext. Reset in *Figure 14*), the signal on the $\overline{\text{RESET}}$ pin may be stretched. Otherwise the delay will not be applied (see long ext. Reset in *Figure 14*). Starting from the external RESET pulse recognition, the device $\overline{\text{RESET}}$ pin acts as an output that is pulled low during at least $t_{w(RSTL)out}$.

### 6.5.3 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 (see *Section 19.3: Operating conditions*).

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

### 6.5.4 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 14*.

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

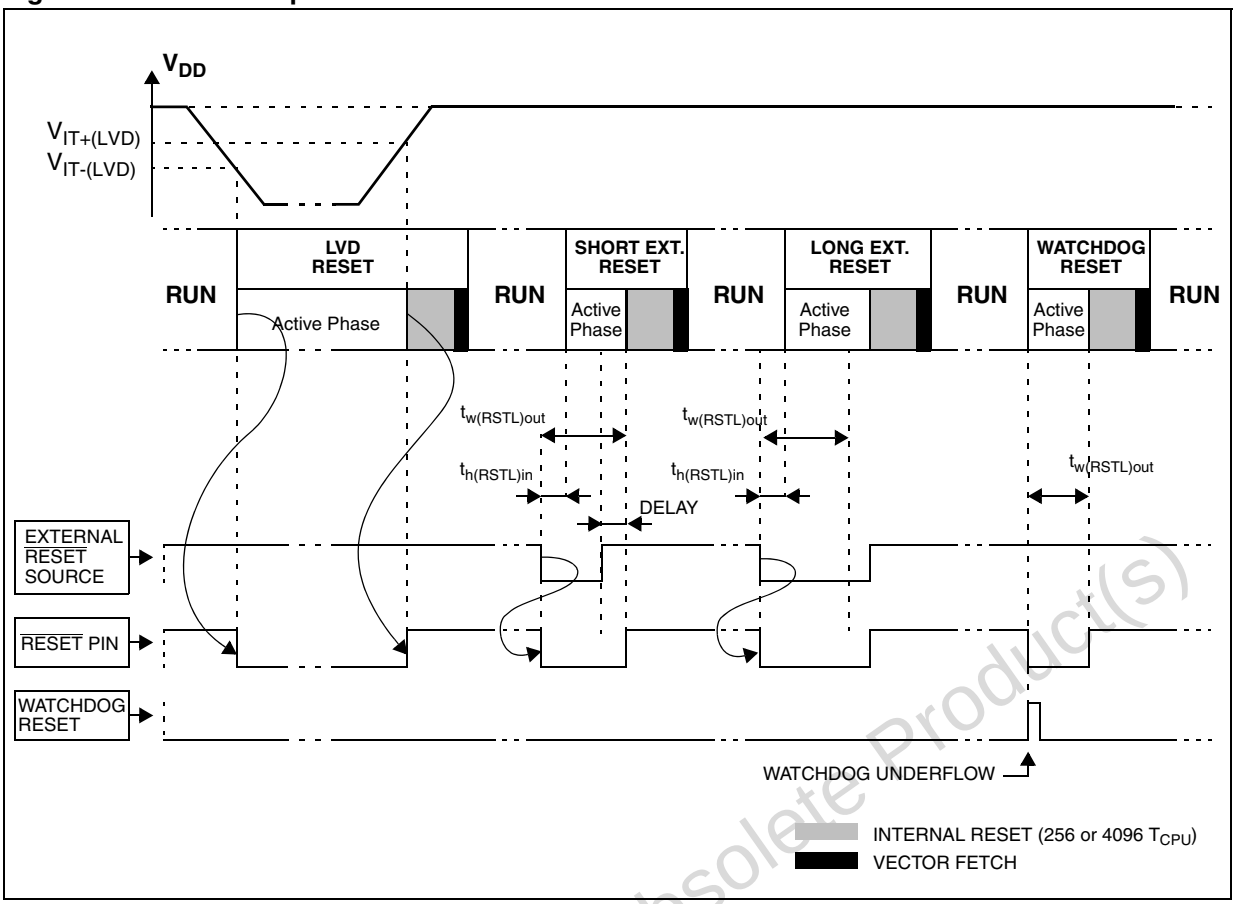
### 6.5.5 Internal watchdog RESET

The RESET sequence generated by an internal Watchdog counter overflow is shown in *Figure 14*.

Starting from the Watchdog counter underflow, the device $\overline{\text{RESET}}$ pin acts as an output that is pulled low during at least $t_{w(RSTL)out}$.

Obsolete Product(s) - Obsolete Product(s)

**Figure 14. RESET sequences**




Obsolete Product(s)

## 6.6 System integrity management (SI)

The System Integrity Management block contains the Low Voltage Detector (LVD) and Auxiliary Voltage Detector (AVD) functions and Clock Security System (CSS). It is managed by the SICSR register.

### 6.6.1 Low voltage detector (LVD)

The low voltage detector function (LVD) generates a static reset when the $V_{DD}$ supply voltage is below a $V_{IT-}$ reference value. This means that it secures the power-up as well as the power-down keeping the ST7 in reset.

The $V_{IT-}$ reference value for a voltage drop is lower than the $V_{IT+}$ reference value for power-on in order to avoid a parasitic reset when the MCU starts running and sinks current on the supply (hysteresis).

The LVD reset circuitry generates a reset when $V_{DD}$ is below:

- $V_{IT+}$ when $V_{DD}$ is rising
- $V_{IT-}$ when $V_{DD}$ is falling

The LVD function is illustrated in *Figure 15*.

The voltage threshold can be configured by option byte to be low, medium or high.

Provided the minimum $V_{DD}$ value (guaranteed for the oscillator frequency) is above $V_{IT-}$, the MCU can only be in two modes:

- under full software control
- in static safe reset

In these conditions, secure operation is always ensured for the application without the need for external reset hardware.

During a low voltage detector reset, the $\overline{\text{RESET}}$ pin is held low, thus permitting the MCU to reset other devices.

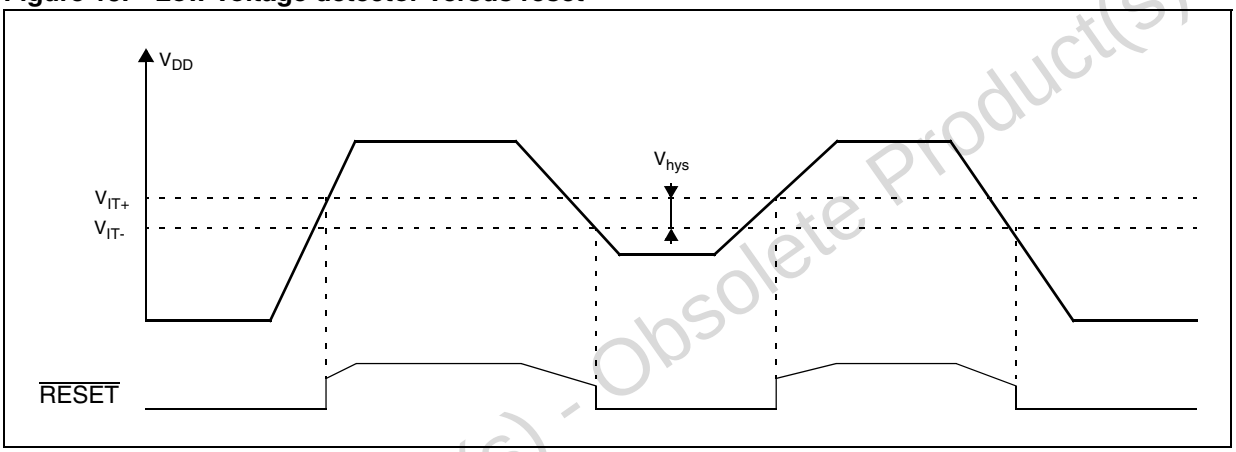
*Note: The LVD allows the device to be used without any external RESET circuitry.*

*If the medium or low thresholds are selected, the detection may occur outside the specified operating voltage range. Below 3.8V, device operation is not guaranteed.*

*The LVD is an optional function which can be selected by option byte.*

*It is recommended to make sure that the $V_{DD}$ supply voltage rises monotonously when the device is exiting from Reset, to ensure the application functions properly.*

**Figure 15. Low voltage detector versus reset**




Obsolete Product(s) - Obsolete Product(s)

## 6.6.2 Auxiliary voltage detector (AVD)

The auxiliary voltage detector function (AVD) is based on an analog comparison between a $V_{IT-(AVD)}$ and $V_{IT+(AVD)}$ reference value and the $V_{DD}$ main supply or the external EVD pin voltage level ($V_{EVD}$). The $V_{IT-}$ reference value for falling voltage is lower than the $V_{IT+}$ reference value for rising voltage in order to avoid parasitic detection (hysteresis).

The output of the AVD comparator can be read directly by the application software through a real-time status bit (AVDF) in the SICSR register. This bit is read only.

**Caution:** The AVD function is active only if the LVD is enabled through the option byte.

### Monitoring the $V_{DD}$ main supply

This mode is selected by clearing the AVDS bit in the SICSR register.

The AVD voltage threshold value is relative to the selected LVD threshold configured by option byte (see *Section 21.1.1: Flash configuration*).

If the AVD interrupt is enabled, an interrupt is generated when the voltage crosses the $V_{IT+(AVD)}$ or $V_{IT-(AVD)}$ threshold (AVDF bit toggles).

In the case of a drop in voltage, the AVD interrupt acts as an early warning, allowing software to shut down safely before the LVD resets the microcontroller. See *Figure 16*.

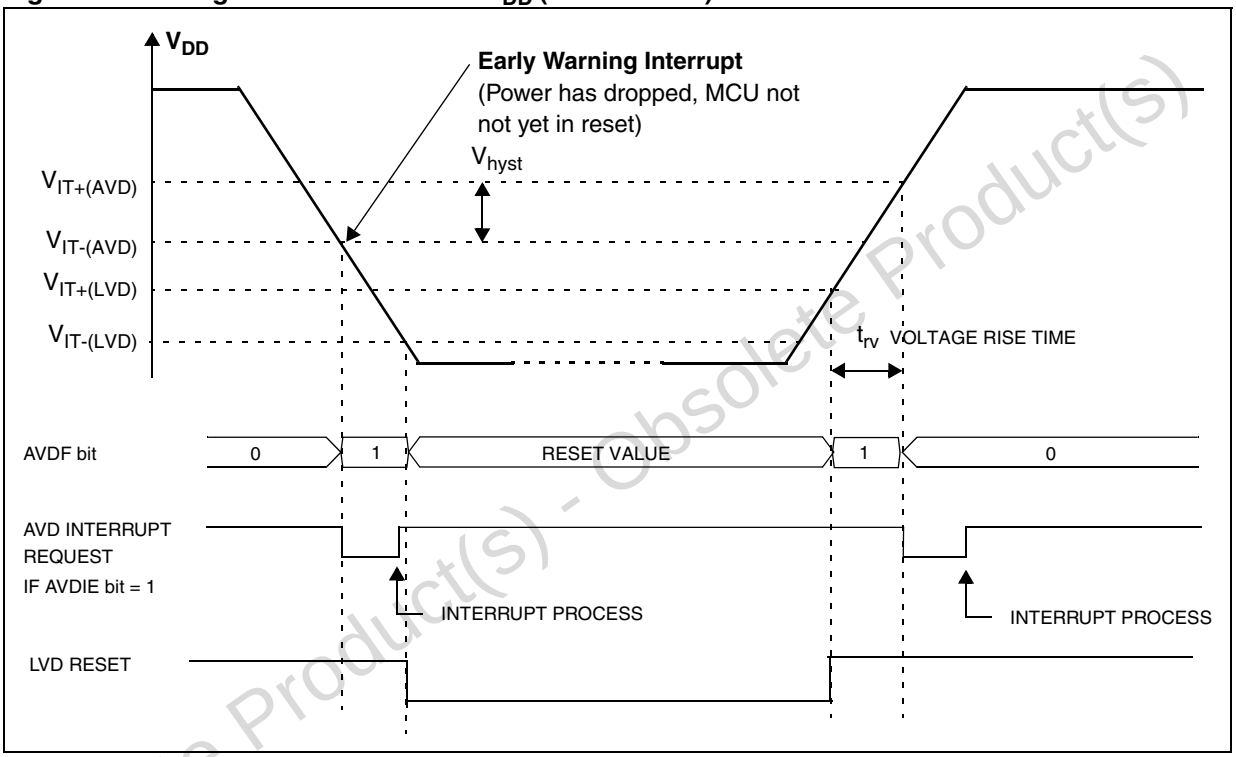
The interrupt on the rising edge is used to inform the application that the $V_{DD}$ warning state is over.

If the voltage rise time $t_{rv}$ is less than 256 or 4096 CPU cycles (depending on the reset delay selected by option byte), no AVD interrupt will be generated when $V_{IT+(AVD)}$ is reached.

If $t_{rv}$ is greater than 256 or 4096 cycles

- two AVD interrupts will be received if the AVD interrupt is enabled **before** the $V_{IT+(AVD)}$ threshold is reached: the first when the AVDIE bit is set, and the second when the threshold is reached.
- only one AVD interrupt will occur if the AVD interrupt is enabled **after** the $V_{IT+(AVD)}$ threshold is reached.

**Figure 16. Using the AVD to monitor $V_{DD}$ (AVDS bit = 0)**




Obsolete Product(s) - Obsolete Product(s)

## Monitoring a voltage on the EVD pin

This mode is selected by setting the AVDS bit in the SICSR register.

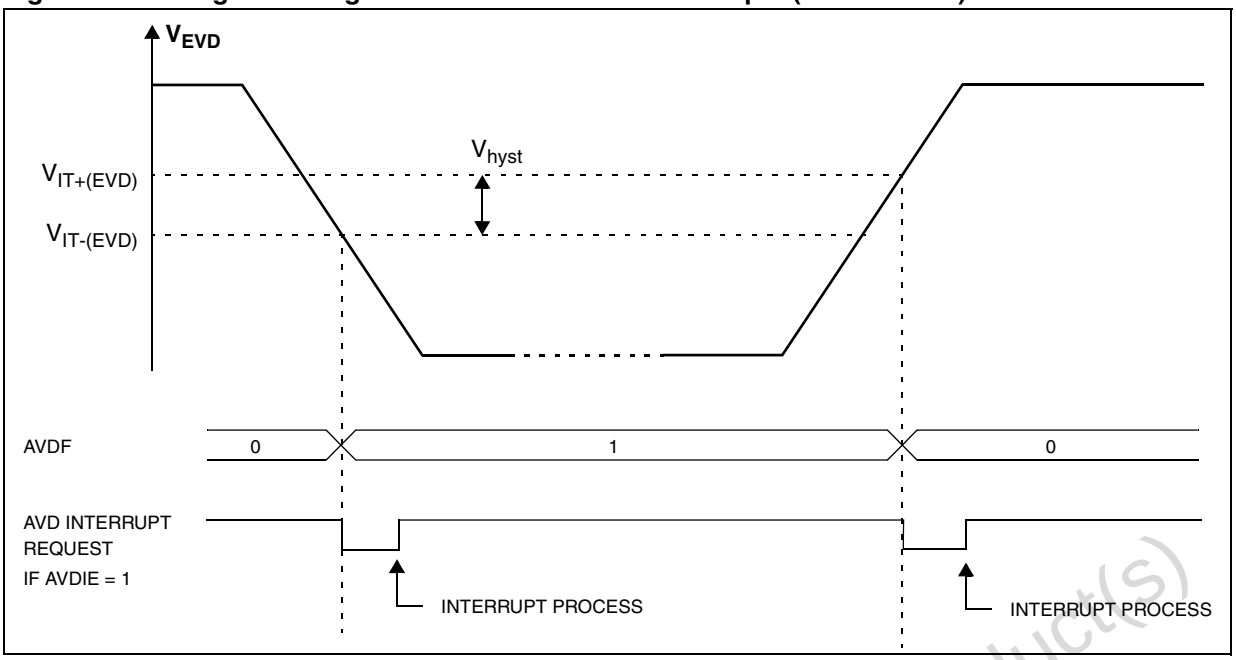
The AVD circuitry can generate an interrupt when the AVDIE bit of the SICSR register is set. This interrupt is generated on the rising and falling edges of the comparator output. This means it is generated when either one of these two events occur:

- $V_{EVD}$ rises up to $V_{IT+(EVD)}$
- $V_{EVD}$ falls down to $V_{IT-(EVD)}$

The EVD function is illustrated in *Figure 17*.

For more details, refer to *Section 19: Electrical characteristics*.

**Figure 17. Using the voltage detector to monitor the EVD pin (AVDS bit = 1)**




### 6.6.3 Clock security system (CSS)

The Clock Security System (CSS) protects the ST7 against breakdowns, spikes and overfrequencies occurring on the main clock source ($f_{OSC}$). It is based on a clock filter and a clock detection control with an internal safe oscillator ($f_{SFOSC}$).

#### Clock filter control

The PLL has an integrated glitch filtering capability making it possible to protect the internal clock from overfrequencies created by individual spikes. This feature is available only when the PLL is enabled. If glitches occur on $f_{OSC}$ (for example, due to loose connection or noise), the CSS filters these automatically, so the internal CPU frequency ($f_{CPU}$) continues delivering a glitch-free signal (see *Figure 18*).

#### Clock detection control

If the clock signal disappears (due to a broken or disconnected resonator...), the safe oscillator delivers a low frequency clock signal ($f_{SFOSC}$) which allows the ST7 to perform some rescue operations.

Automatically, the ST7 clock source switches back from the safe oscillator ($f_{SFOSC}$) if the main clock source ($f_{OSC}$) recovers.

When the internal clock ($f_{CPU}$) is driven by the safe oscillator ($f_{SFOSC}$), the application software is notified by hardware setting the CSSD bit in the SICSR register. An interrupt can be generated if the CSSIE bit has been previously set. These two bits are described in the SICSR register description.

Obsolete Product(s) - Obsolete Product(s)

### 6.6.4 Low power modes

**Table 12. Effect of low power modes on SI**

| Mode | Effect |
|---|---|
| Wait | No effect on SI. CSS and AVD interrupts cause the device to exit from Wait mode. |
| Halt | The SICSR register is frozen.The CSS (including the safe oscillator) is disabled until HALT mode is exited. The previous CSS configuration resumes when the MCU is woken up by an interrupt with “exit from HALT mode” capability or from the counter reset value when the MCU is woken up by a RESET. |

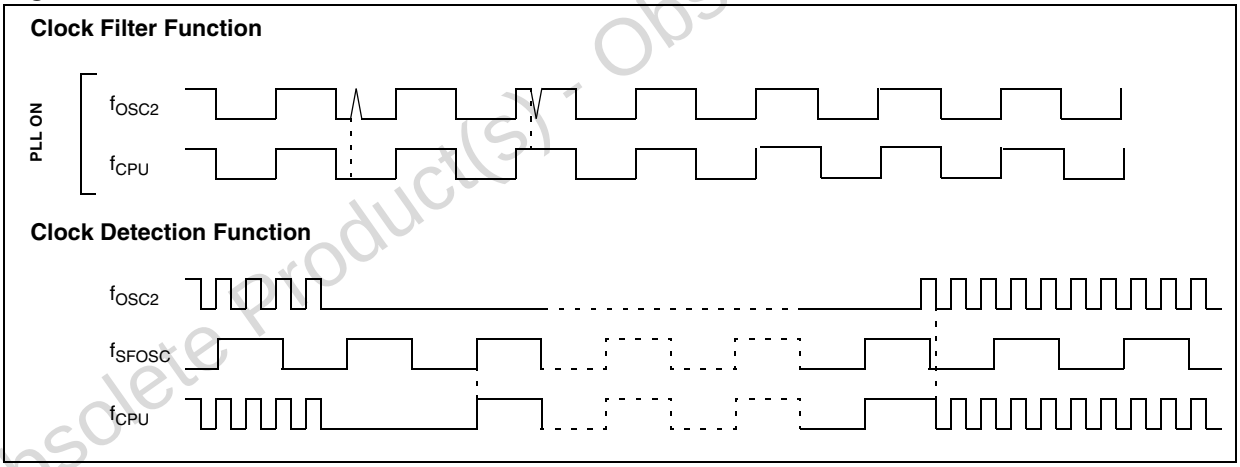
### 6.6.5 Interrupts

The CSS or AVD interrupt eventsgenerate an interrupt if the corresponding Enable Control Bit (CSSIE or AVDIE) is set and the interrupt mask in the CC register is reset (RIM instruction).

**Table 13. CSS AVD interrupt control/wake-up capability**

| Interrupt event | Event flag | Enable control bit | Exit from Wait | Exit from Halt |
|---|---|---|---|---|
| CSS event detection (safe oscillator activated as main clock) | CSSD | CSSIE | Yes | No |
| AVD event | AVDF | AVDIE | Yes | No |

**Figure 18. Clock filter function**




Obsolete Product(s) - Obsolete Product(s)

### 6.6.6 System Integrity (SI) Control/Status register (SICSR)

SICSR Reset value: 000x 000x (00h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| AVDS | AVDIE | AVDF | LVDRF | Reserved | | | WDGRF |
| RW | RW | RW | RW | - | | | RW |

**Table 14. SICSR description**

| Bit | Name | Function |
|---|---|---|
| 7 | AVDS | *Voltage Detection selection*<br>This bit is set and cleared by software. Voltage Detection is available only if the LVD is enabled by option byte.<br>0: Voltage detection on $V_{DD}$ supply<br>1: Voltage detection on EVD pin |
| 6 | AVDIE | *Voltage Detector interrupt enable*<br>This bit is set and cleared by software. It enables an interrupt to be generated when the AVDF flag changes (toggles). The pending interrupt information is automatically cleared when software enters the AVD interrupt routine.<br>0: AVD interrupt disabled<br>1: AVD interrupt enabled |
| 5 | AVDF | *Voltage Detector flag*<br>This read-only bit is set and cleared by hardware. If the AVDIE bit is set, an interrupt request is generated when the AVDF bit changes value. Refer to *Figure 16* and to *Monitoring the VDD main supply on page 48* for additional details.<br>0: $V_{DD}$ or $V_{EVD}$ over $V_{IT+(AVD)}$ threshold<br>1: $V_{DD}$ or $V_{EVD}$ under $V_{IT-(AVD)}$ threshold |
| 4 | LVDRF | *LVD reset flag*<br>This bit indicates that the last Reset was generated by the LVD block. It is set by hardware (LVD reset) and cleared by software (writing zero). See *Table 15: Reset source flags* for more details. When the LVD is disabled by OPTION BYTE, the LVDRF bit value is undefined. |
| 3 | - | Reserved, must be kept cleared. |
| 2 | CSSIE | *Clock security system interrupt enable*<br>This bit enables the interrupt when a disturbance is detected by the Clock Security System (CSSD bit set). It is set and cleared by software.<br>0: Clock security system interrupt disabled<br>1: Clock security system interrupt enabled<br>When the CSS is disabled by OPTION BYTE, the CSSIE bit has no effect. |
| 1 | CSSD | *Clock security system detection*<br>This bit indicates that the safe oscillator of the Clock Security System block has been selected by hardware due to a disturbance on the main clock signal ($f_{OSC}$). It is set by hardware and cleared by reading the SICSR register when the original oscillator recovers.<br>0: Safe oscillator is not active<br>1: Safe oscillator has been activated<br>When the CSS is disabled by OPTION BYTE, the CSSD bit value is forced to 0. |

Obsolete Product(s) - Obsolete Product(s)

**Table 14. SICSR description (continued)**

| Bit | Name | Function |
|---|---|---|
| 0 | WDGRF | *Watchdog reset flag*<br>This bit indicates that the last Reset was generated by the Watchdog peripheral. It is set by hardware (watchdog reset) and cleared by software (writing zero) or an LVD Reset (to ensure a stable cleared state of the WDGRF flag when CPU starts). Combined with the LVDRF flag information, the flag description is given in *Table 15*. |

**Table 15. Reset source flags**

| Reset sources | LVDRF | WDGRF |
|---|---|---|
| External $\overline{\text{RESET}}$ pin | 0 | 0 |
| Watchdog | 0 | 1 |
| LVD | 1 | X |

## Application notes

The LVDRF flag is not cleared when another RESET type occurs (external or watchdog); the LVDRF flag remains set to keep trace of the original failure.

In this case, software can detect a watchdog reset but cannot detect an external reset.

**Caution:** When the LVD is not activated with the associated option byte, the WDGRF flag cannot be used in the application.

Obsolete Product(s) - Obsolete Product(s)

# 7 Interrupts

## 7.1 Introduction

The ST7 enhanced interrupt management provides the following features:

- Hardware interrupts
- Software interrupt (TRAP)
- Nested or concurrent interrupt management with flexible interrupt priority and level management:
  - Up to 4 software programmable nesting levels
  - Up to 16 interrupt vectors fixed by hardware
  - 2 non-maskable events: RESET, TRAP
  - 1 maskable Top Level event: TLI

This interrupt management is based on:

- Bit 5 and bit 3 of the CPU CC register (I1:0)
- Interrupt software priority registers (ISPRx)
- Fixed interrupt vector addresses located at the high addresses of the memory map (FFE0h to FFFFh) sorted by hardware priority order

This enhanced interrupt controller guarantees full upward compatibility with the standard (not nested) ST7 interrupt controller.

## 7.2 Masking and processing flow

The interrupt masking is managed by the I1 and I0 bits of the CC register and the ISPRx registers which give the interrupt software priority level of each interrupt vector (see *Table 16*). The processing flow is shown in *Figure 19*.

When an interrupt request has to be serviced:

- Normal processing is suspended at the end of the current instruction execution.
- The PC, X, A and CC registers are saved onto the stack.
- I1 and I0 bits of CC register are set according to the corresponding values in the ISPRx registers of the serviced interrupt vector.
- The PC is then loaded with the interrupt vector of the interrupt to service and the first instruction of the interrupt service routine is fetched (refer to *Table 21: Interrupt mapping* for vector addresses).

The interrupt service routine should end with the IRET instruction which causes the contents of the saved registers to be recovered from the stack.

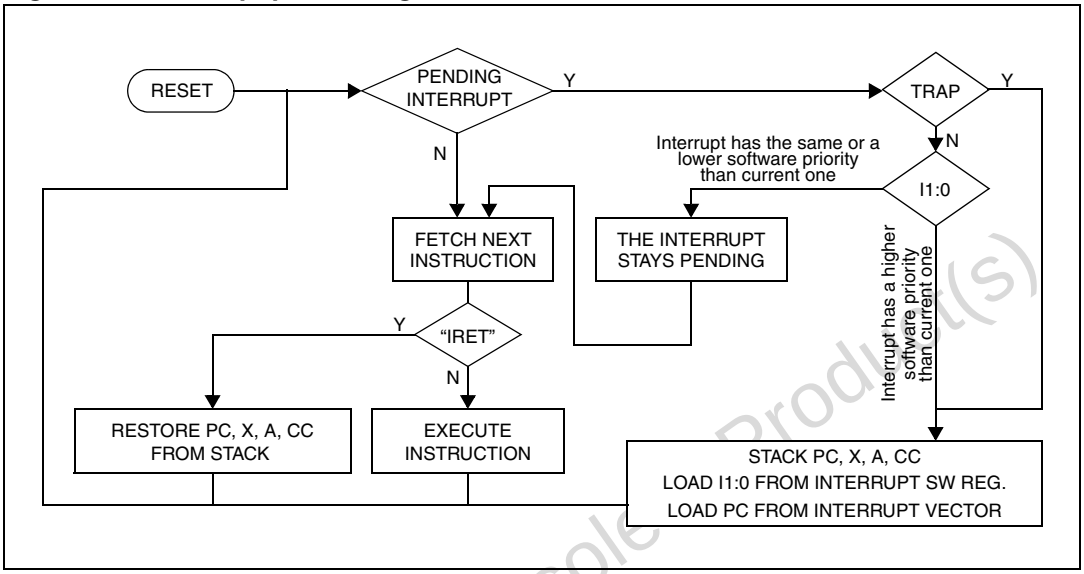
*Note: As a consequence of the IRET instruction, the I1 and I0 bits will be restored from the stack and the program in the previous level will resume.*

Obsolete Product(s) - Obsolete Product(s)

**Table 16. Interrupt software priority levels**

| Interrupt software priority | Level | I1 | I0 |
|---|---|---|---|
| Level 0 (main) | Low ↓ High | 1 | 0 |
| Level 1 | | 0 | 1 |
| Level 2 | | 0 | 0 |
| Level 3 (= interrupt disable) | | 1 | 1 |

**Figure 19. Interrupt processing flowchart**



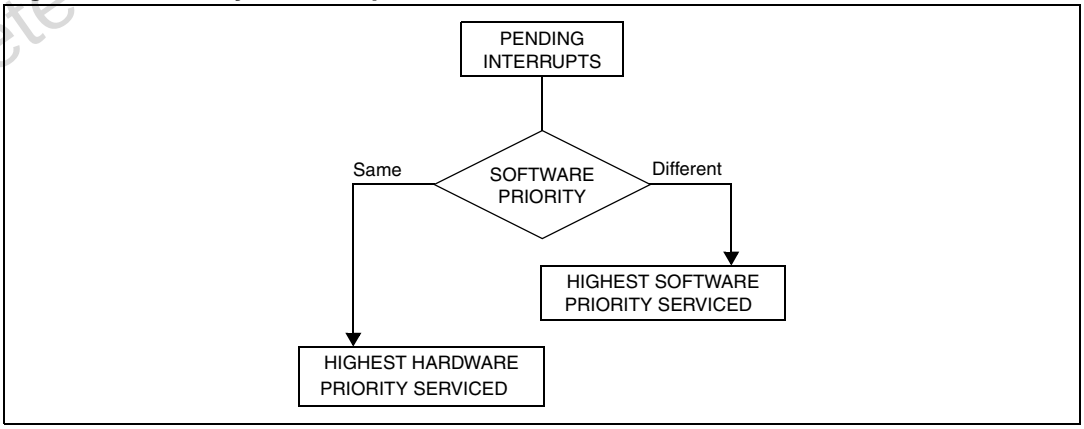

## Servicing pending interrupts

As several interrupts can be pending at the same time, the interrupt to be taken into account is determined by the following two-step process:

- the highest software priority interrupt is serviced,
- if several interrupts have the same software priority then the interrupt with the highest hardware priority is serviced first.

*Figure 20* describes this decision process.

**Figure 20. Priority decision process flowchart**




Obsolete Product(s) - Obsolete Product(s)

When an interrupt request is not serviced immediately, it is latched and then processed when its software priority combined with the hardware priority becomes the highest one.

*Note: 1 The hardware priority is exclusive while the software one is not. This allows the previous process to succeed with only one interrupt.*

*2 TLI, RESET and TRAP can be considered as having the highest software priority in the decision process.*

### Different interrupt vector sources

Two interrupt source types are managed by the ST7 interrupt controller: the non-maskable type (RESET, TRAP) and the maskable type (external or from internal peripherals).

### Non-maskable sources

These sources are processed regardless of the state of the I1 and I0 bits of the CC register (see *Figure 19*). After stacking the PC, X, A and CC registers (except for RESET), the corresponding vector is loaded in the PC register and the I1 and I0 bits of the CC are set to disable interrupts (level 3). These sources allow the processor to exit Halt mode.

- TRAP (non-maskable software interrupt)

  This software interrupt is serviced when the TRAP instruction is executed. It will be serviced according to the flowchart in *Figure 19*.

**Caution:** TRAP can be interrupted by a TLI.

- RESET

  The RESET source has the highest priority in the ST7. This means that the first current routine has the highest software priority (level 3) and the highest hardware priority.

  See *Section 6.5: Reset sequence manager (RSM)* for more details.

### Maskable sources

Maskable interrupt vector sources can be serviced if the corresponding interrupt is enabled and if its own interrupt software priority (in ISPRx registers) is higher than the one currently being serviced (I1 and I0 in CC register). If any of these two conditions is false, the interrupt is latched and thus remains pending.

- TLI (top level hardware interrupt)

**Caution:** This hardware interrupt occurs when a specific edge is detected on the dedicated TLI pin. It will be serviced according to the flowchart in *Figure 19* as a trap.
A TRAP instruction must not be used in a TLI service routine.

- External Interrupts

  External interrupts allow the processor to exit from HALT low power mode. External interrupt sensitivity is software selectable through the External Interrupt Control register (EICR).

  External interrupt triggered on edge will be latched and the interrupt request automatically cleared upon entering the interrupt service routine.
  If several input pins of a group connected to the same interrupt line are selected simultaneously, these will be logically ORed.

- Peripheral Interrupts

  Usually the peripheral interrupts cause the MCU to exit from Halt mode except those mentioned in *Table 21: Interrupt mapping*. A peripheral interrupt occurs when a specific

Obsolete Product(s) - Obsolete Product(s)

flag is set in the peripheral status registers and if the corresponding enable bit is set in the peripheral control register.

The general sequence for clearing an interrupt is based on an access to the status register followed by a read or write to an associated register.

*Note: The clearing sequence resets the internal latch. A pending interrupt (that is, waiting to be serviced) will therefore be lost if the clear sequence is executed.*

## 7.3 Interrupts and low power modes

All interrupts allow the processor to exit the Wait low power mode. On the contrary, only external and other specified interrupts allow the processor to exit from the Halt modes (see column “Exit from Halt/Active Halt” in *Table 21: Interrupt mapping*). When several pending interrupts are present while exiting Halt mode, the first one serviced can only be an interrupt with “exit from Halt mode” capability and it is selected through the same decision process shown in *Figure 20*.

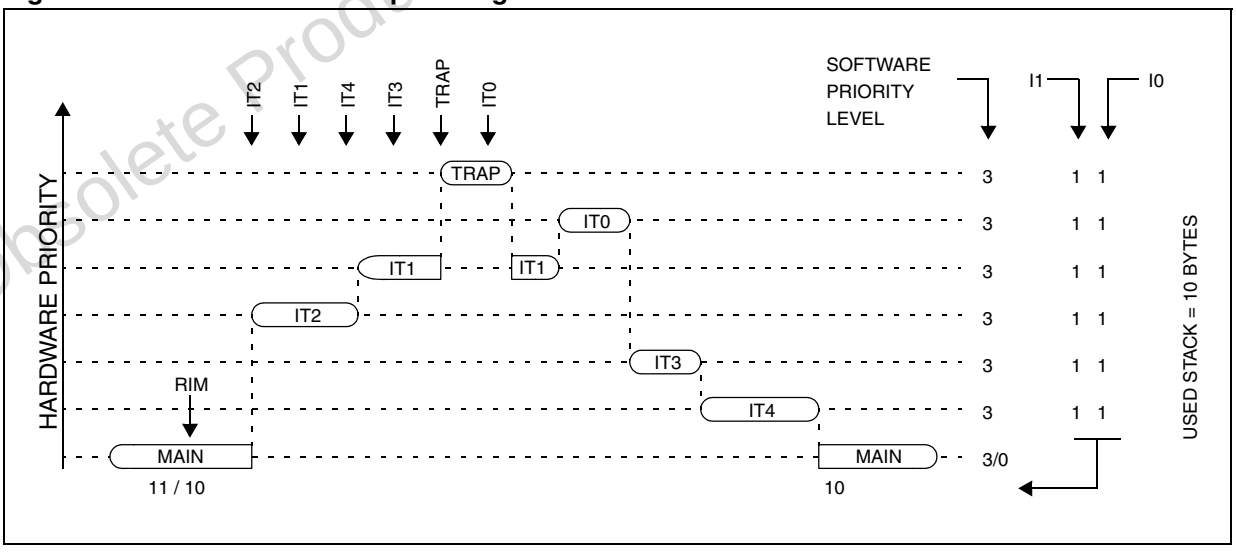
*Note: If an interrupt that is not able to exit from Halt mode is pending with the highest priority when exiting Halt mode, this interrupt is serviced after the first one serviced.*

## 7.4 Concurrent and nested management

The following *Figure 21* and *Figure 22* show two different interrupt management modes. The first is called concurrent mode and does not allow an interrupt to be interrupted, unlike the nested mode in *Figure 22*. The interrupt hardware priority is given in this order from the lowest to the highest: MAIN, IT4, IT3, IT2, IT1, IT0, TLI. The software priority is given for each interrupt.

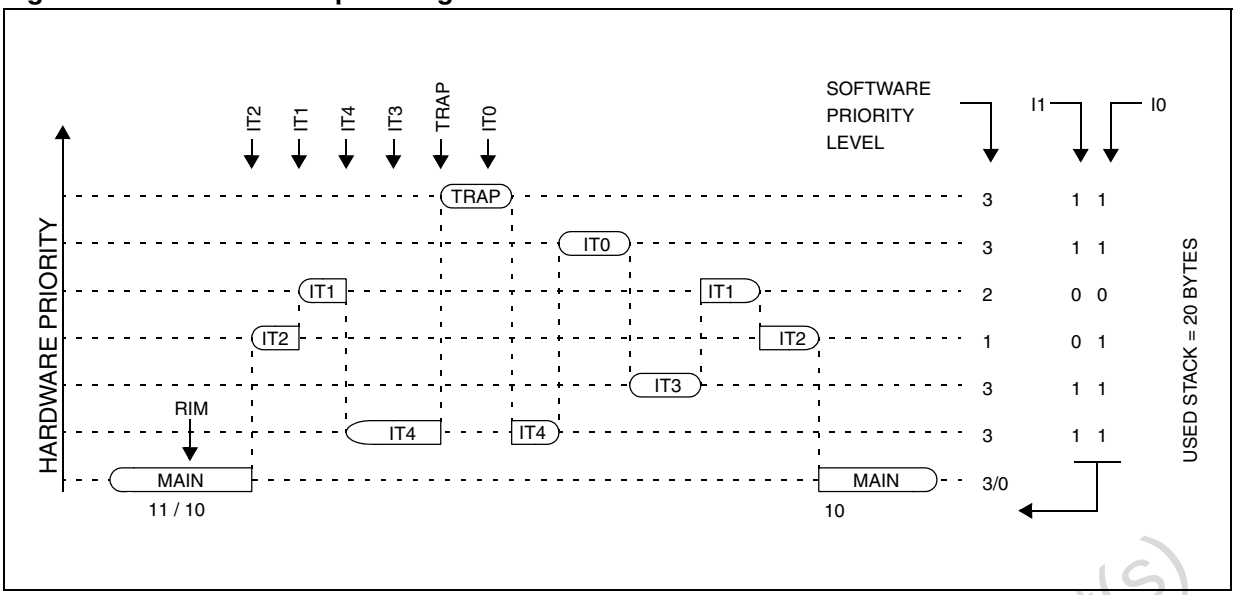
**Warning: A stack overflow may occur without notifying the software of the failure.**

**Figure 21. Concurrent interrupt management**




Obsolete Product(s) - Obsolete Product(s)

**Figure 22. Nested interrupt management**




## 7.5 Interrupt register description

### 7.5.1 CPU CC register interrupt bits

CPU CC Reset value: 111x 1010 (xAh)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | **I1** | H | **I0** | N | Z | C |
| | | RW | RW | RW | RW | RW | RW |

**Table 17. CPU CC register interrupt bits description**

| Bit | Name | Function |
|---|---|---|
| 5 | I1 | *Interrupt Software Priority 1* |
| 3 | I0 | *Interrupt Software Priority 0* |

These two bits indicate the current interrupt software priority (see *Table 18*) and are set/cleared by hardware when entering in interrupt. The loaded value is given by the corresponding bits in the interrupt software priority registers (ISPRx).

They can be also set/cleared by software with the RIM, SIM, HALT, WFI, IRET and PUSH/POP instructions (see *Table 20: Interrupt dedicated instruction set*).

Obsolete Product(s) - Obsolete Product(s)

**Table 18. Interrupt software priority levels**

| Interrupt software priority | Level | I1 | I0 |
|---|---|---|---|
| Level 0 (main) | Low ↓ High | 1 | 0 |
| Level 1 | | 0 | 1 |
| Level 2 | | 0 | 0 |
| Level 3 (= interrupt disable[1]) | | 1 | 1 |

1. TRAP and RESET events can interrupt a level 3 program.

### 7.5.2 Interrupt software priority registers (ISPRx)

These four registers are read/write, with the exception of bits 7:4 of ISPR3, which are read only.

ISPRx Reset value: 1111 1111 (FFh)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| ISPR0 | I1_3 | I0_3 | I1_2 | I0_2 | I1_1 | I0_1 | I1_0 | I0_0 |
| ISPR1 | I1_7 | I0_7 | I1_6 | I0_6 | I1_5 | I0_5 | I1_4 | I0_4 |
| ISPR2 | I1_11 | I0_11 | I1_10 | I0_10 | I1_9 | I0_9 | I1_8 | I0_8 |
| ISPR3 | 1 | 1 | 1 | 1 | I1_13 | I0_13 | I1_12 | I0_12 |

These four registers contain the interrupt software priority of each interrupt vector.

- Each interrupt vector (except RESET and TRAP) has corresponding bits in these registers where its own software priority is stored. This correspondence is shown in the following *Table 19*.

**Table 19. Interrupt priority bits**

| Vector address | ISPRx bits |
|---|---|
| FFFBh-FFFAh | I1_0 and I0_0 bits[1] |
| FFF9h-FFF8h | I1_1 and I0_1 bits |
| ... | ... |
| FFE1h-FFE0h | I1_13 and I0_13 bits |

1. Bits in the ISPRx registers which correspond to the TLI can be read and written but they are not significant in the interrupt process management.

- Each I1_x and I0_x bit value in the ISPRx registers has the same meaning as the I1 and I0 bits in the CC register.
- Level 0 cannot be written (I1_x = 1, I0_x = 0). In this case, the previously stored value is kept (Example: previous = CFh, write = 64h, result = 44h).

The TLI, RESET, and TRAP vectors have no software priorities. When one is serviced, the I1 and I0 bits of the CC register are both set.

**Caution:** If the I1_x and I0_x bits are modified while the interrupt x is executed the following behavior has to be considered: If the interrupt x is still pending (new interrupt or flag not cleared) and the new software priority is higher than the previous one, the interrupt x is re-entered.

Obsolete Product(s) - Obsolete Product(s)

Otherwise, the software priority stays unchanged up to the next interrupt request (after the IRET of the interrupt x).

**Table 20. Interrupt dedicated instruction set**

| Instruction | New description | Function/Example | I1 | H | I0 | N | Z | C |
|---|---|---|---|---|---|---|---|---|
| HALT | Entering Halt mode | | 1 | | 0 | | | |
| IRET | Interrupt routine return | Pop CC, A, X, PC | I1 | H | I0 | N | Z | C |
| JRM | Jump if I1:0 = 11 (level 3) | I1:0 = 11 ? | | | | | | |
| JRNM | Jump if I1:0 <> 11 | I1:0 <> 11 ? | | | | | | |
| POP CC | Pop CC from the Stack | Mem => CC | I1 | H | I0 | N | Z | C |
| RIM | Enable interrupt (level 0 set) | Load 10 in I1:0 of CC | 1 | | 0 | | | |
| SIM | Disable interrupt (level 3 set) | Load 11 in I1:0 of CC | 1 | | 1 | | | |
| TRAP | Software trap | Software NMI | 1 | | 1 | | | |
| WFI | Wait for interrupt | | 1 | | 0 | | | |

*Note: During the execution of an interrupt routine, the HALT, POPCC, RIM, SIM and WFI instructions change the current software priority up to the next IRET instruction or one of the previously mentioned instructions.*

Obsolete Product(s) - Obsolete Product(s)

**Table 21. Interrupt mapping**

| No. | Source block | Description | Register label | Priority order | Exit from Halt / Active Halt | Address vector |
|---|---|---|---|---|---|---|
| | RESET | Reset | N/A | | yes | FFFEh-FFFFh |
| | TRAP | Software interrupt | | | no | FFFCh-FFFDh |
| 0 | TLI | External top level interrupt | EICR | | yes | FFFAh-FFFBh |
| 1 | MCC/RTC CSS | Main clock controller time base interrupt Safe oscillator activation interrupt | MCCSRSI CSR | Higher priority | yes | FFF8h-FFF9h |
| 2 | ei0 | External interrupt port A3..0 | N/A | | yes | FFF6h-FFF7h |
| 3 | ei1 | External interrupt port F2..0 | | | yes | FFF4h-FFF5h |
| 4 | ei2 | External interrupt port B3..0 | | | yes | FFF2h-FFF3h |
| 5 | ei3 | External interrupt port B7..4 | | | yes | FFF0h-FFF1h |
| 6 | Not used | | | | | FFEEh-FFEFh |
| 7 | SPI | SPI peripheral interrupts | SPICSR | | yes(1) | FFECh-FFEDh |
| 8 | TIMER A | TIMER A peripheral interrupts | TASR | | no | FFEAh-FFEBh |
| 9 | TIMER B | TIMER B peripheral interrupts | TBSR | | no | FFE8h-FFE9h |
| 10 | SCI | SCI peripheral interrupts | SCISR | Lower priority | no | FFE6h-FFE7h |
| 11 | AVD | Auxiliary voltage detector interrupt | SICSR | | no | FFE4h-FFE5h |
| 12 | I2C | I2C peripheral interrupts | (see peripheral) | | no | FFE2h-FFE3h |
| 13 | PWM ART | PWM ART interrupt | ARTCSR | | yes(2) | FFE0h-FFE1h |

1. Exit from HALT possible when SPI is in slave mode.
2. Exit from HALT possible when PWM ART is in external clock mode.

Obsolete Product(s) - Obsolete Product(s)

## 7.6 External interrupts

### 7.6.1 I/O port interrupt sensitivity

The external interrupt sensitivity is controlled by the IPA, IPB and ISxx bits of the EICR register (*Figure 23*). This control allows to have up to four fully independent external interrupt source sensitivities.

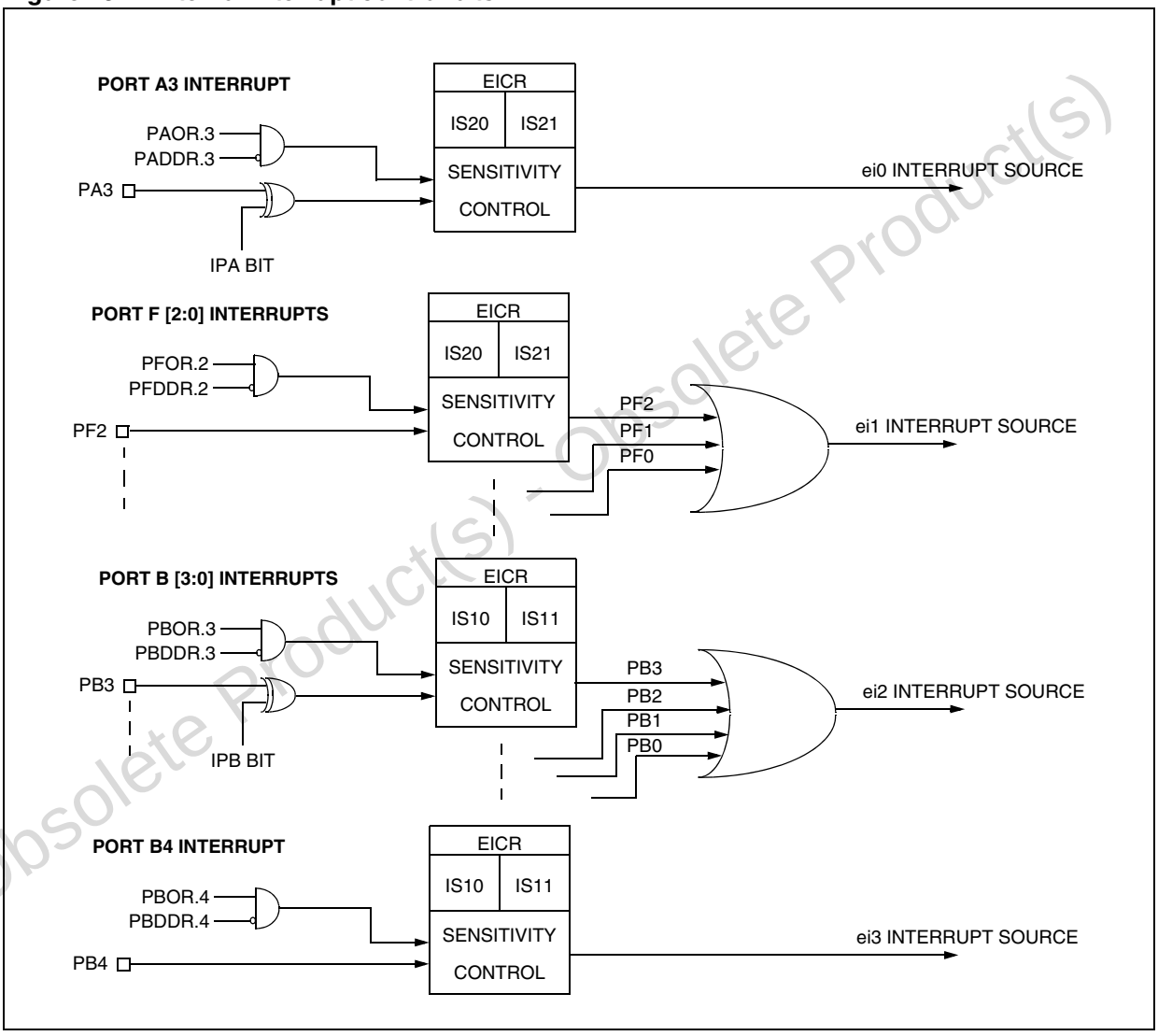
Each external interrupt source can be generated on four (or five) different events on the pin:

- Falling edge
- Rising edge
- Falling and rising edge
- Falling edge and low level
- Rising edge and high level (only for ei0 and ei2)

To guarantee correct functionality, the sensitivity bits in the EICR register can be modified only when the I1 and I0 bits of the CC register are both set to 1 (level 3). This means that interrupts must be disabled before changing sensitivity.

The pending interrupts are cleared by writing a different value in the ISx[1:0], IPA or IPB bits of the EICR.

**Figure 23. External interrupt control bits**




Obsolete Product(s) - Obsolete Product(s)

### 7.6.2 External interrupt control register (EICR)

EICR Reset value: 0000 0000 (00h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| IS1[1:0] | | IPB | IS2[1:0] | | IPA | TLIS | TLIE |
| RW | | RW | RW | | RW | RW | RW |

**Table 22. EICR register description**

| Bit | Name | Function |
|---|---|---|
| 7:6 | IS1[1:0] | *ei2 and ei3 sensitivity*<br>The interrupt sensitivity, defined using the IS1[1:0] bits, is applied to the following external interrupts:<br>- ei2 (port B3..0) (see *Table 23*)<br>- ei3 (port B4) (see *Table 24*)<br>These 2 bits can be written only when I1 and I0 of the CC register are both set to 1 (level 3). |
| 5 | IPB | *Interrupt polarity for port B*<br>This bit is used to invert the sensitivity of the port B [3:0] external interrupts. It can be set and cleared by software only when I1 and I0 of the CC register are both set to 1 (level 3).<br>0: No sensitivity inversion<br>1: Sensitivity inversion |
| 4:3 | IS2[1:0] | *ei0 and ei1 sensitivity*<br>The interrupt sensitivity, defined using the IS2[1:0] bits, is applied to the following external interrupts:<br>- ei0 (port A3) (see *Table 25*)<br>- ei1 (port F2..0) (see *Table 26*)<br>These 2 bits can be written only when I1 and I0 of the CC register are both set to 1 (level 3). |
| 2 | IPA | *Interrupt polarity for port A*<br>This bit is used to invert the sensitivity of the port A [3:0] external interrupts. It can be set and cleared by software only when I1 and I0 of the CC register are both set to 1 (level 3).<br>0: No sensitivity inversion<br>1: Sensitivity inversion |
| 1 | TLIS | *TLI sensitivity*<br>This bit allows to toggle the TLI edge sensitivity. It can be set and cleared by software only when TLIE bit is cleared.<br>0: Falling edge<br>1: Rising edge |
| 0 | TLIE | *TLI enable*<br>This bit allows to enable or disable the TLI capability on the dedicated pin. It is set and cleared by software.<br>0: TLI disabled<br>1: TLI enabled<br>*Note: A parasitic interrupt can be generated when clearing the TLIE bit.* |

Obsolete Product(s) - Obsolete Product(s)

**Table 23. Interrupt sensitivity - ei2 (port B3..0)**

| IS11 | IS10 | External interrupt sensitivity | |
|---|---|---|---|
| | | IPB bit = 0 | IPB bit = 1 |
| 0 | 0 | Falling edge and low level | Rising edge and high level |
| 0 | 1 | Rising edge only | Falling edge only |
| 1 | 0 | Falling edge only | Rising edge only |
| 1 | 1 | Rising and falling edge | |

**Table 24. Interrupt sensitivity - ei3 (port B4)**

| IS11 | IS10 | External interrupt sensitivity |
|---|---|---|
| 0 | 0 | Falling edge and low level |
| 0 | 1 | Rising edge only |
| 1 | 0 | Falling edge only |
| 1 | 1 | Rising and falling edge |

**Table 25. Interrupt sensitivity - ei0 (port A3)**

| IS21 | IS20 | External interrupt sensitivity | |
|---|---|---|---|
| | | IPA bit = 0 | IPA bit = 1 |
| 0 | 0 | Falling edge and low level | Rising edge and high level |
| 0 | 1 | Rising edge only | Falling edge only |
| 1 | 0 | Falling edge only | Rising edge only |
| 1 | 1 | Rising and falling edge | |

**Table 26. Interrupt sensitivity - ei1 (port F2..0)**

| IS21 | IS20 | External interrupt sensitivity |
|---|---|---|
| 0 | 0 | Falling edge and low level |
| 0 | 1 | Rising edge only |
| 1 | 0 | Falling edge only |
| 1 | 1 | Rising and falling edge |

Obsolete Product(s) - Obsolete Product(s)

**Table 27. Nested interrupts register map and reset values**

| Address (Hex.) | Register label | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0024h | ISPR0 | ei1 | | ei0 | | MCC + SI | | TLI | |
| | Reset value | I1_3 1 | I0_3 1 | I1_2 1 | I0_2 1 | I1_1 1 | I0_1 1 | 1 | 1 |
| 0025h | ISPR1 | SPI | | | | ei3 | | ei2 | |
| | Reset value | I1_7 1 | I0_7 1 | I1_6 1 | I0_6 1 | I1_5 1 | I0_5 1 | I1_4 1 | I0_4 1 |
| 0026h | ISPR2 | AVD | | SCI | | TIMER B | | TIMER A | |
| | Reset value | I1_11 1 | I0_11 1 | I1_10 1 | I0_10 1 | I1_9 1 | I0_9 1 | I1_8 1 | I0_8 1 |
| 0027h | ISPR3 | | | | | PWMART | | I2C | |
| | Reset value | 1 | 1 | 1 | 1 | I1_13 1 | I0_13 1 | I1_12 1 | I0_12 1 |
| 0028h | EICR | IS11 0 | IS10 0 | IPB 0 | IS21 0 | IS20 0 | IPA 0 | TLIS 0 | TLIE 0 |

Obsolete Product(s) - Obsolete Product(s)

# 8 Power saving modes

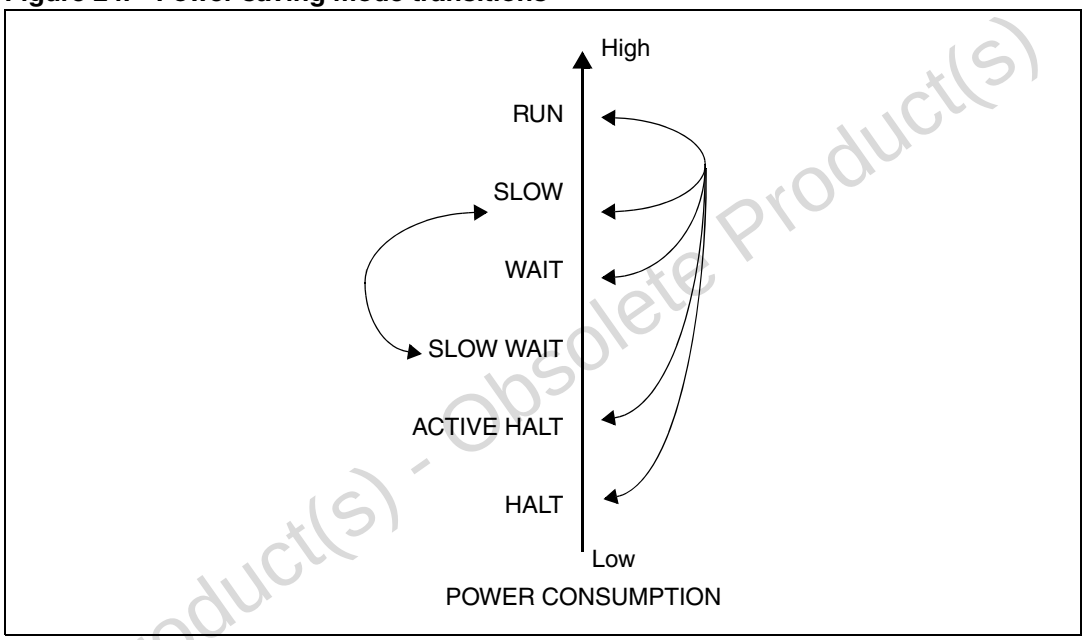
## 8.1 Introduction

To give a large measure of flexibility to the application in terms of power consumption, four main power saving modes are implemented in the ST7 (see *Figure 24*): Slow, Wait (Slow Wait), Active Halt and Halt.

After a RESET the normal operating mode is selected by default (Run mode). This mode drives the device (CPU and embedded peripherals) by means of a master clock which is based on the main oscillator frequency divided or multiplied by 2 ($f_{OSC2}$).

From Run mode, the different power saving modes may be selected by setting the relevant register bits or by calling the specific ST7 software instruction whose action depends on the oscillator status.

**Figure 24. Power saving mode transitions**




Obsolete Product(s) - Obsolete Product(s)

## 8.2 Slow mode

This mode has two targets:

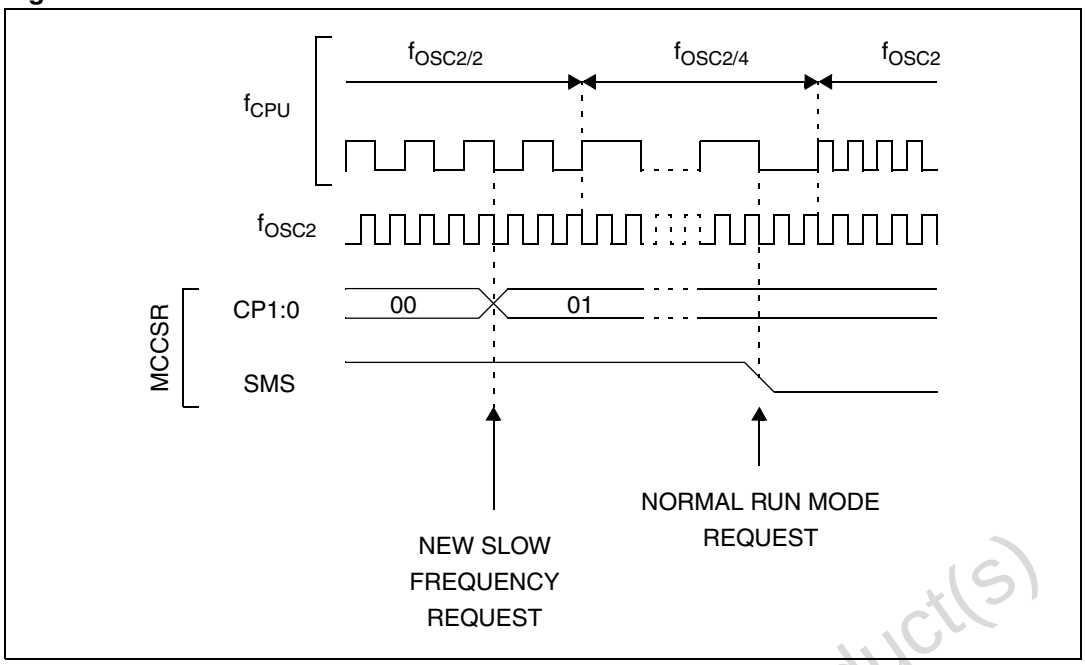
- To reduce power consumption by decreasing the internal clock in the device,
- To adapt the internal clock frequency ($f_{CPU}$) to the available supply voltage.

Slow mode is controlled by three bits in the MCCSR register: the SMS bit which enables or disables Slow mode and two CPx bits which select the internal slow frequency ($f_{CPU}$).

In this mode, the master clock frequency ($f_{OSC2}$) can be divided by 2, 4, 8 or 16. The CPU and peripherals are clocked at this lower frequency ($f_{CPU}$).

*Note: Slow Wait mode is activated when entering the Wait mode while the device is already in Slow mode.*

**Figure 25. Slow mode clock transitions**




## 8.3 Wait mode

Wait mode places the MCU in a low power consumption mode by stopping the CPU.

This power saving mode is selected by calling the 'WFI' instruction.

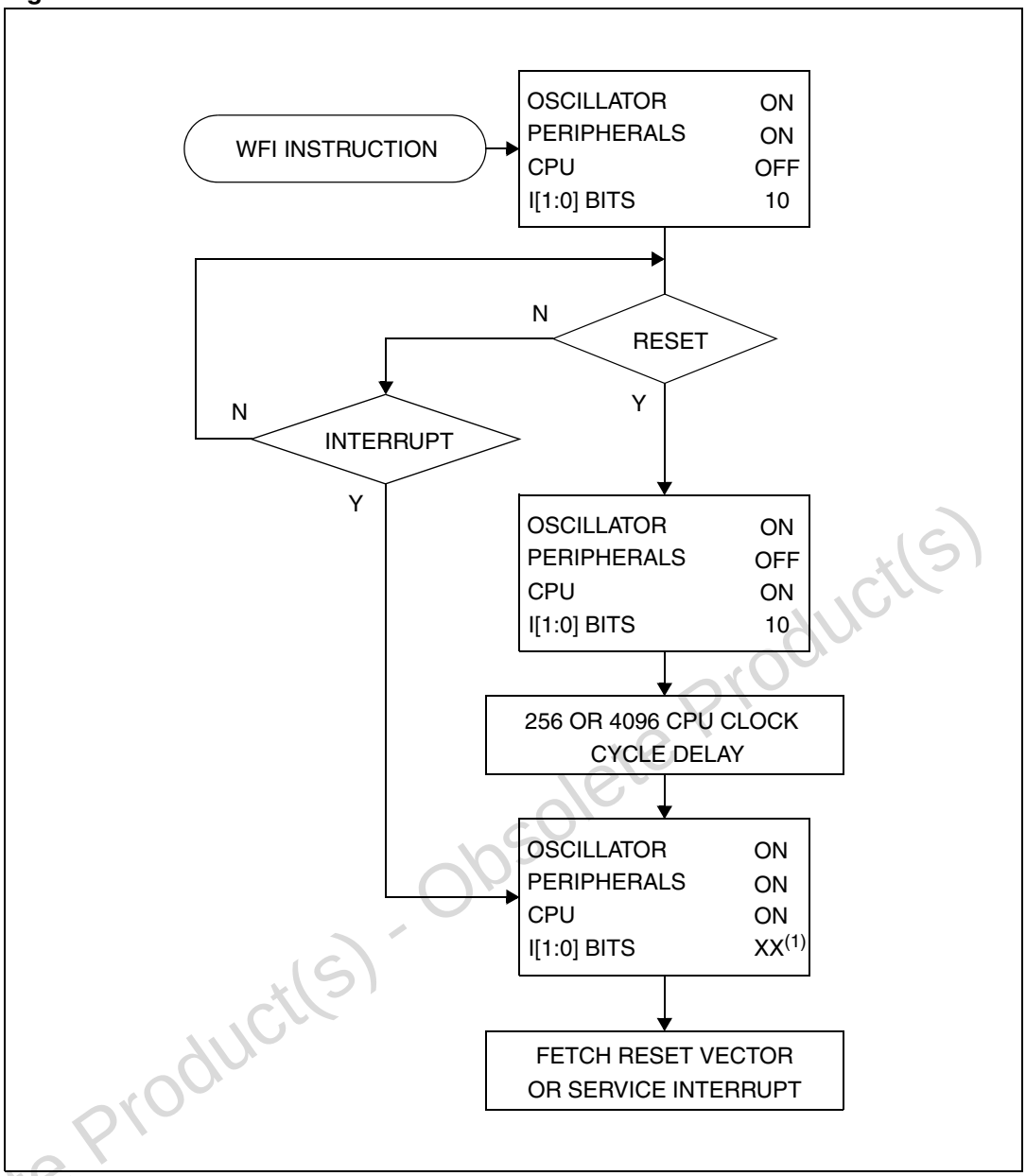
All peripherals remain active. During Wait mode, the I[1:0] bits of the CC register are forced to '10', to enable all interrupts. All other registers and memory remain unchanged. The MCU remains in Wait mode until an interrupt or RESET occurs, whereupon the Program Counter branches to the starting address of the interrupt or Reset service routine.

The MCU will remain in Wait mode until a Reset or an Interrupt occurs, causing it to wake up.

Refer to the following *Figure 26*.

Obsolete Product(s) - Obsolete Product(s)

**Figure 26. Wait mode flowchart**




1. Before servicing an interrupt, the CC register is pushed on the stack. The I[1:0] bits of the CC register are set to the current software priority level of the interrupt routine and recovered when the CC register is popped.

Obsolete Product(s) - Obsolete Product(s)

## 8.4 Active Halt and Halt modes

Active Halt and Halt modes are the two lowest power consumption modes of the MCU. They are both entered by executing the 'HALT' instruction. The decision to enter either in Active Halt or Halt mode is given by the MCC/RTC interrupt enable flag (OIE bit in MCCSR register) as shown in *Table 28*.

**Table 28. MCC/RTC low power mode selection**

| MCCSR OIE bit | Power saving mode entered when HALT instruction is executed |
|---|---|
| 0 | Halt |
| 1 | Active Halt |

### 8.4.1 Active Halt mode

Active Halt mode is the lowest power consumption mode of the MCU with a real-time clock available. It is entered by executing the 'HALT' instruction when the OIE bit of the Main Clock Controller Status register (MCCSR) is set (see *Section 12.3: ART registers* for more details on the MCCSR register).

The MCU can exit Active Halt mode on reception of an external interrupt, MCC/RTC interrupt or a RESET. When exiting Active Halt mode by means of an interrupt, no 256 or 4096 CPU cycle delay occurs. The CPU resumes operation by servicing the interrupt or by fetching the reset vector which woke it up (see *Figure 28*).

When entering Active Halt mode, the I[1:0] bits in the CC register are forced to '10b' to enable interrupts. Therefore, if an interrupt is pending, the MCU wakes up immediately.

In Active Halt mode, only the main oscillator and its associated counter (MCC/RTC) are running to keep a wake-up time base. All other peripherals are not clocked except those which get their clock supply from another clock generator (such as external or auxiliary oscillator).

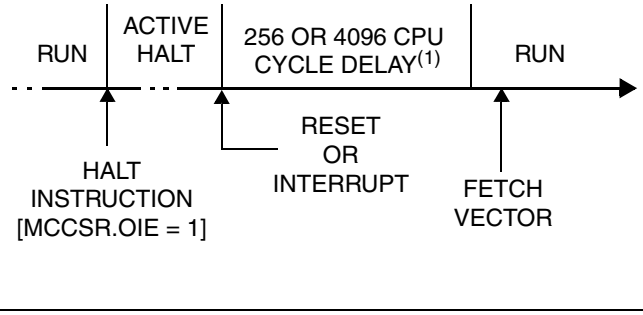
The safeguard against staying locked in Active Halt mode is provided by the oscillator interrupt.

*Note: As soon as the interrupt capability of one of the oscillators is selected (MCCSR.OIE bit set), entering Active Halt mode 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.*

**Caution:** When exiting Active Halt mode following an MCC/RTC interrupt, OIE bit of MCCSR register must not be cleared before $t_{DELAY}$ after the interrupt occurs ($t_{DELAY}$ = 256 or 4096 $t_{CPU}$ delay depending on option byte). Otherwise, the ST7 enters Halt mode for the remaining $t_{DELAY}$ period.

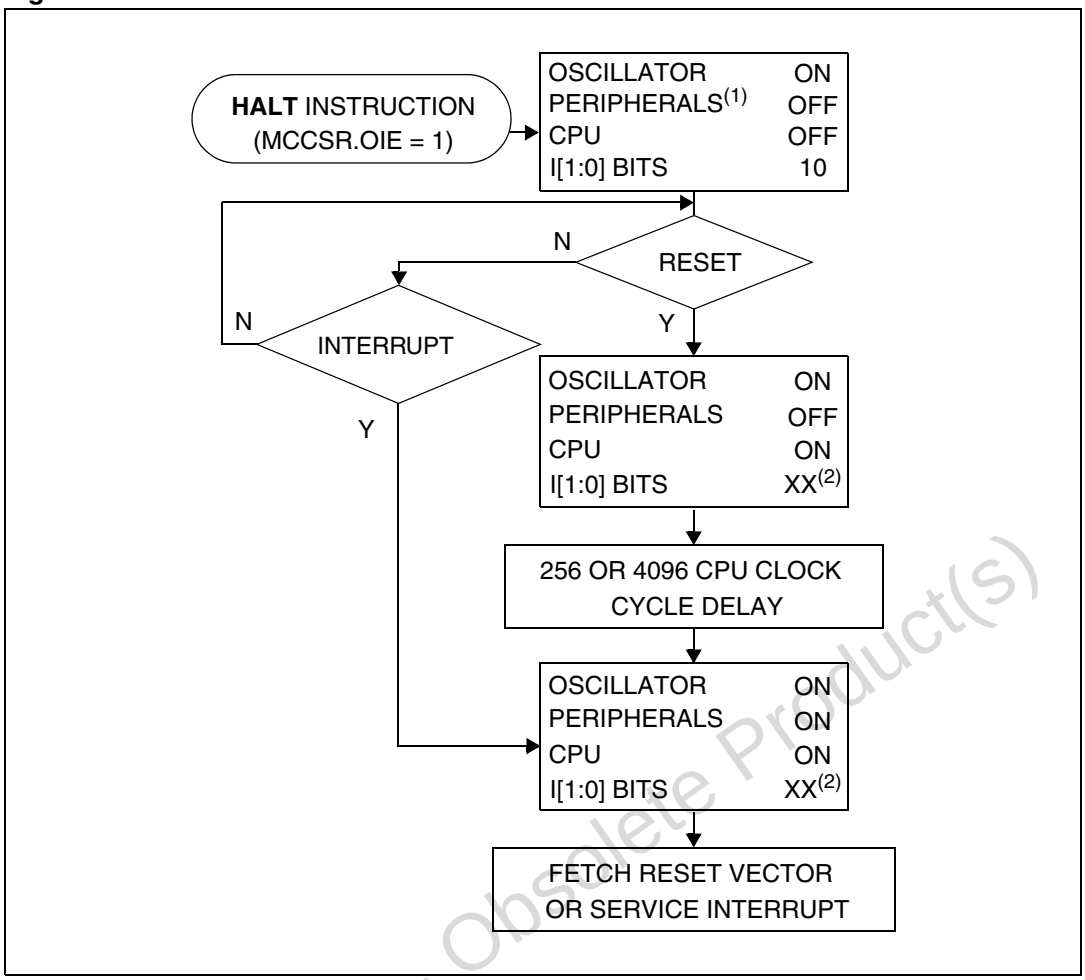
**Figure 27. Active Halt timing overview**




1. This delay occurs only if the MCU exits Active Halt mode by means of a RESET.

Obsolete Product(s) - Obsolete Product(s)

**Figure 28. Active Halt mode flowchart**




1. Peripheral clocked with an external clock source can still be active.
2. Before servicing an interrupt, the CC register is pushed on the stack. The I[1:0] bits of the CC register are set to the current software priority level of the interrupt routine and restored when the CC register is popped.

### 8.4.2 Halt mode

The Halt mode is the lowest power consumption mode of the MCU. It is entered by executing the 'HALT' instruction when the OIE bit of the Main Clock Controller Status register (MCCSR) is cleared (see *Section 11: Main clock controller with real-time clock and beeper (MCC/RTC)*for more details on the MCCSR register).

The MCU can exit Halt mode on reception of either a specific interrupt (see *Table 21: Interrupt mapping*) or a RESET. When exiting Halt mode by means of a RESET or an interrupt, the oscillator is immediately turned on and the 256 or 4096 CPU cycle delay is used to stabilize the oscillator. After the start up delay, the CPU resumes operation by servicing the interrupt or by fetching the reset vector which woke it up (see *Figure 30*).

When entering Halt mode, the I[1:0] bits in the CC register are forced to '10b' to enable interrupts. Therefore, if an interrupt is pending, the MCU wakes up immediately.

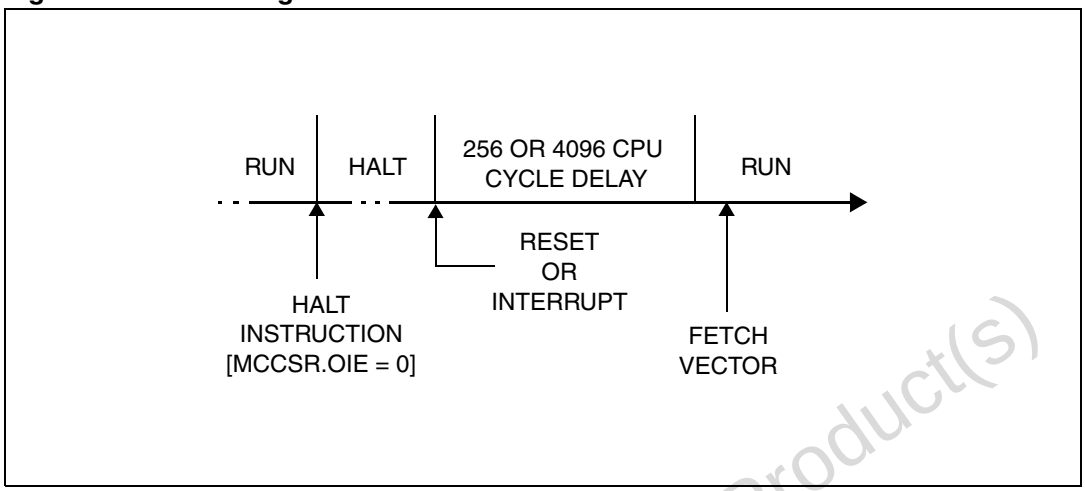
In Halt mode, the main oscillator is turned off causing all internal processing to be stopped, including the operation of the on-chip peripherals. All peripherals are not clocked except the

Obsolete Product(s) - Obsolete Product(s)

ones which get their clock supply from another clock generator (such as an external or auxiliary oscillator).

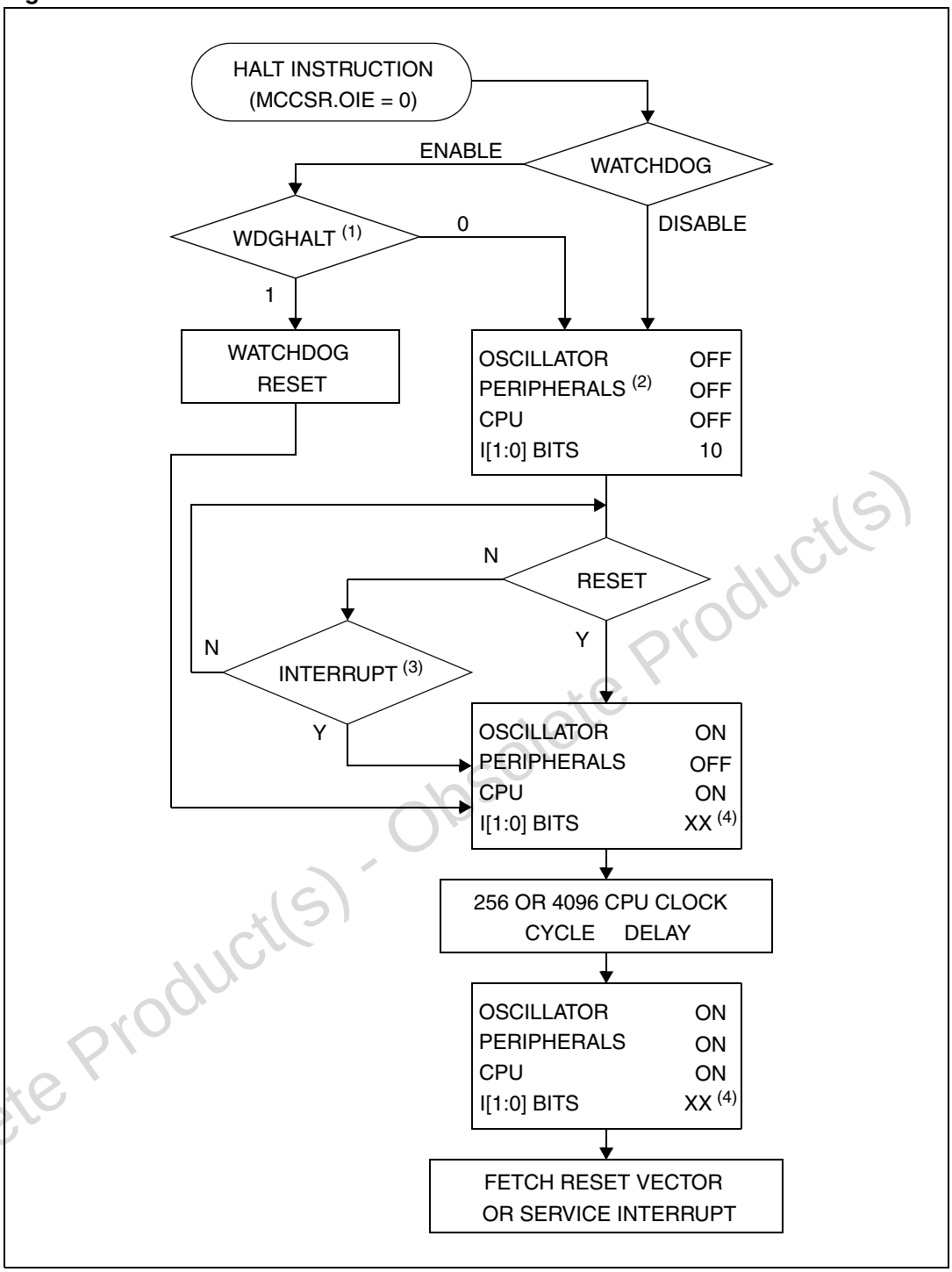
The compatibility of Watchdog operation with Halt mode is configured by the 'WDGHALT' option bit of the option byte. The HALT instruction when executed while the Watchdog system is enabled, can generate a Watchdog RESET (see *Section 21.1.1: Flash configuration* for more details).

**Figure 29. Halt timing overview**




Obsolete Product(s) - Obsolete Product(s)

**Figure 30. Halt mode flowchart**




Obsolete Product(s) - Obsolete Product(s)

1. WDGHALT is an option bit. See *Section 21.1.1: Flash configuration* 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 21: Interrupt mapping* for more details.
4. Before servicing an interrupt, the CC register is pushed on the stack. The I[1:0] bits of the CC register are set to the current software priority level of the interrupt routine and recovered when the CC register is popped.

### 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, re-initialize the corresponding I/O as “Input Pull-up with Interrupt” before executing the HALT instruction. The main reason for this is that the I/O may be wrongly configured due to external interference or by an unforeseen logical condition.
- For the same reason, reinitialize the level sensitiveness of each external interrupt as a precautionary measure.
- The opcode for the HALT instruction is 0x8E. To avoid an unexpected HALT instruction due to a program counter failure, it is advised to clear all occurrences of the data value 0x8E from memory.
- As the HALT instruction clears the interrupt mask in the CC register to allow interrupts, the user may choose to clear all pending interrupt bits before executing the HALT instruction. This avoids entering other peripheral interrupt routines after executing the external interrupt routine corresponding to the wake-up event (reset or external interrupt).

### Related documentation

*ST7 Keypad Decoding Techniques, Implementing Wake-Up on Keystroke* (AN 980)

*How to Minimize the ST7 Power Consumption* (AN1014)

*Using an active RC to wake up the ST7LITE0 from power saving mode* (AN1605)

Obsolete Product(s) - Obsolete Product(s)

# 9 I/O ports

## 9.1 Introduction

The I/O ports offer different functional modes:

- transfer of data through digital inputs and outputs

and for specific pins:

- external interrupt generation
- alternate signal input/output for the on-chip peripherals.

An I/O port contains up to eight pins. Each pin can be programmed independently as digital input (with or without interrupt generation) or digital output.

## 9.2 Functional description

Each port has two main registers:

- Data Register (DR)
- Data Direction Register (DDR)

and one optional register:

- Option Register (OR)

Each I/O pin may be programmed using the corresponding register bits in the DDR and OR registers (bit X corresponding to pin X of the port). The same correspondence is used for the DR register.

The following description takes into account the OR register (for specific ports which do not provide this register refer to *Section 9.3: I/O port implementation on page 78*). The generic I/O block diagram is shown in *Figure 31*.

### 9.2.1 Input modes

The input configuration is selected by clearing the corresponding DDR register bit.

In this case, reading the DR register returns the digital value applied to the external I/O pin.

Different input modes can be selected by software through the OR register.

*Note: 1 Writing the DR register modifies the latch value but does not affect the pin status.*

*2 When switching from input to output mode, the DR register has to be written first to drive the correct level on the pin as soon as the port is configured as an output.*

*3 Do not use read/modify/write instructions (BSET or BRES) to modify the DR register as this might corrupt the DR content for I/Os configured as input.*

#### External interrupt function

When an I/O is configured as Input with Interrupt, an event on this I/O can generate an external interrupt request to the CPU.

Each pin can independently generate an interrupt request. The interrupt sensitivity is independently programmable using the sensitivity bits in the EICR register.

Obsolete Product(s) - Obsolete Product(s)

Each external interrupt vector is linked to a dedicated group of I/O port pins (see pinout description and interrupt section). If several input pins are selected simultaneously as interrupt sources, these are first detected according to the sensitivity bits in the EICR register and then logically ORed.

The external interrupts are hardware interrupts, which means that the request latch (not accessible directly by the application) is automatically cleared when the corresponding interrupt vector is fetched. To clear an unwanted pending interrupt by software, the sensitivity bits in the EICR register must be modified.

### 9.2.2 Output modes

The output configuration is selected by setting the corresponding DDR register bit. In this case, writing the DR register applies this digital value to the I/O pin through the latch. Then reading the DR register returns the previously stored value.

Two different output modes can be selected by software through the OR register: Output push-pull and open-drain. The DR register value and output pin status are shown in the following *Table 29*.

**Table 29. I/O output mode selection**

| DR | Push-pull | Open-drain |
|---|---|---|
| 0 | $V_{SS}$ | $V_{SS}$ |
| 1 | $V_{DD}$ | Floating |

### 9.2.3 Alternate functions

When an on-chip peripheral is configured to use a pin, the alternate function is automatically selected. This alternate function takes priority over the standard I/O programming.

When the signal is coming from an on-chip peripheral, the I/O pin is automatically configured in output mode (push-pull or open-drain according to the peripheral).

When the signal is going to an on-chip peripheral, the I/O pin must be configured in input mode. In this case, the pin state is also digitally readable by addressing the DR register.

*Note: Input pull-up configuration can cause unexpected value at the input of the alternate peripheral input. When an on-chip peripheral use a pin as input and output, this pin has to be configured in input floating mode.*

Obsolete Product(s) Obsolete Product(s)

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

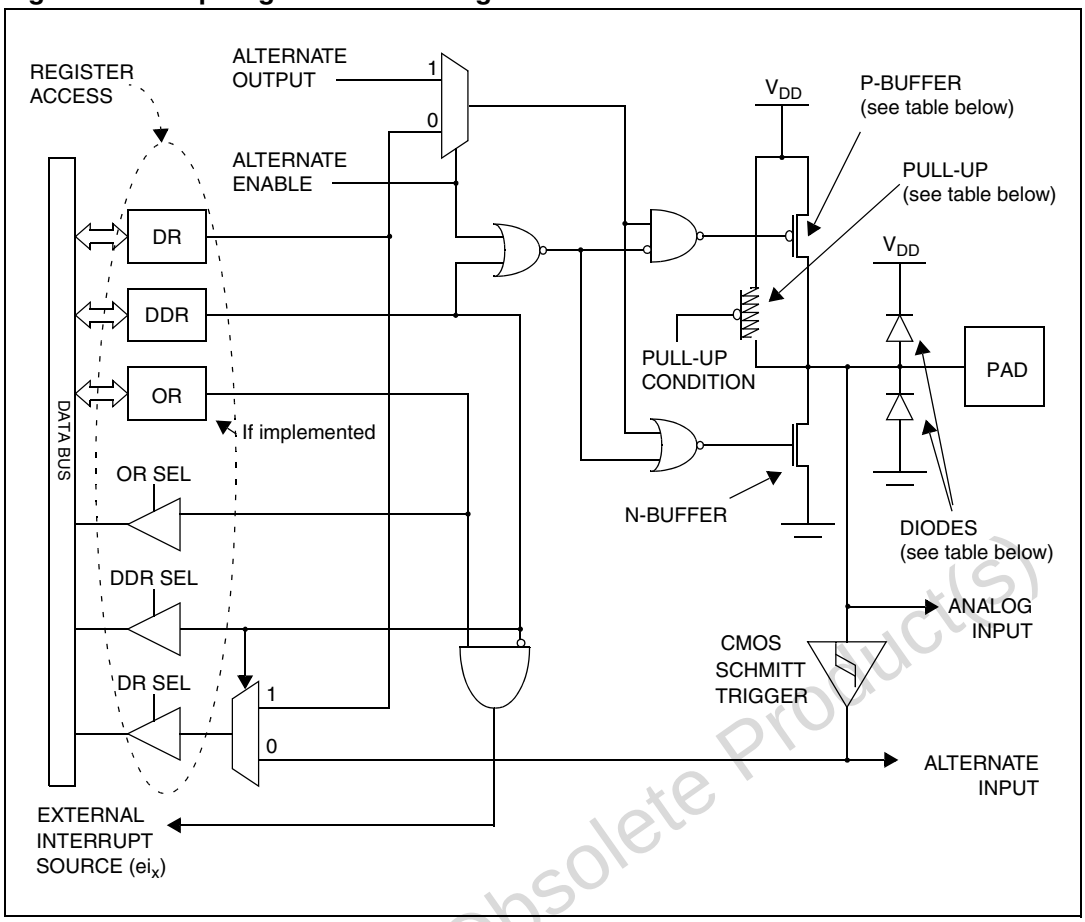



**Table 30. I/O port mode options**

| Configuration mode | | Pull-up | P-buffer | Diodes | |
|---|---|---|---|---|---|
| | | | | to $V_{DD}$ | to $V_{SS}$ |
| Input | Floating with/without Interrupt | Off | Off | On | On |
| | Pull-up with/without Interrupt | On | | | |
| Output | Push-pull | Off | On | | |
| | Open-drain (logic level) | | Off | | |
| | True open-drain | NI | NI | NI(1) | |

1. The diode to $V_{DD}$ is not implemented in the true open-drain pads. A local protection between the pad and $V_{SS}$ is implemented to protect the device against positive stress.

Legend:

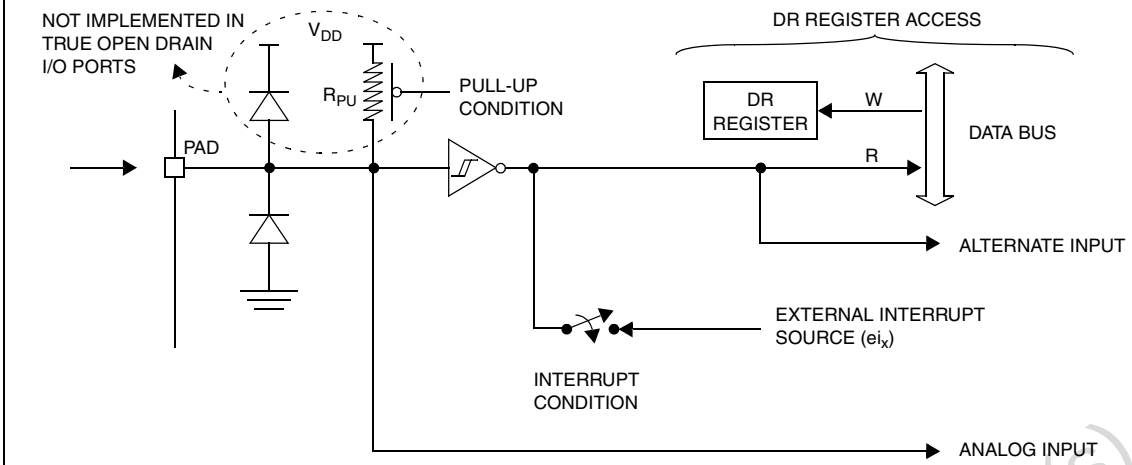
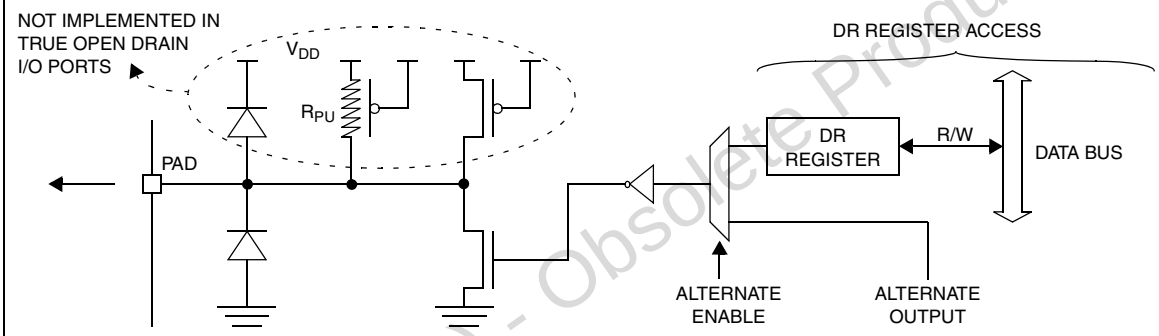
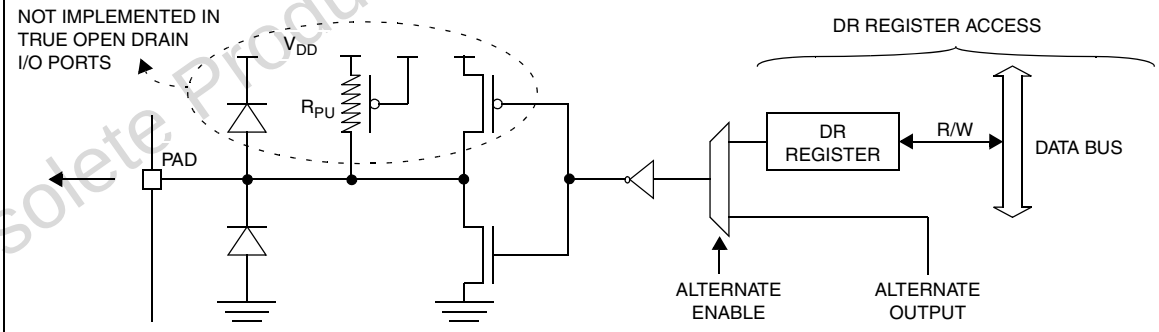
Off - Implemented not activated

On - Implemented and activated

NI - Not implemented

Obsolete Product(s) - Obsolete Product(s)

**Table 31. I/O port configurations**

| | Hardware configuration |
|---|---|
| Input(1) |  |
| Open-drain output(2) |  |
| Push-pull output(2) |  |

1. When the I/O port is in input configuration and the associated alternate function is enabled as an output, reading the DR register will read the alternate function output status.
2. When the I/O port is in output configuration and the associated alternate function is enabled as an input, the alternate function reads the pin status given by the DR register content.

Obsolete Product(s) - Obsolete Product(s)

**Caution:** The alternate function must not be activated as long as the pin is configured as input with interrupt, in order to avoid generating spurious interrupts.

### Analog alternate function

When the pin is used as an ADC input, the I/O must be configured as floating input. The analog multiplexer (controlled by the ADC registers) switches the analog voltage present on the selected pin to the common analog rail which is connected to the ADC input.

It is recommended not to change the voltage level or loading on any port pin while conversion is in progress. Furthermore it is recommended not to have clocking pins located close to a selected analog pin.

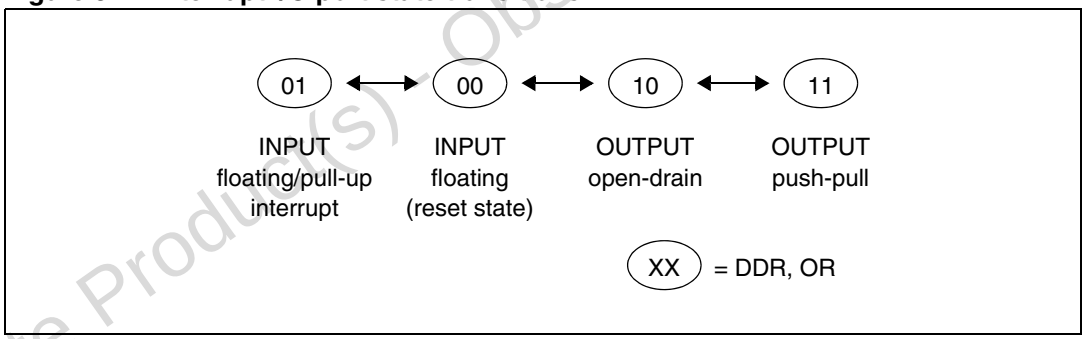
**Warning: The analog input voltage level must be within the limits stated in the absolute maximum ratings.**

## 9.3 I/O port implementation

The hardware implementation on each I/O port depends on the settings in the DDR and OR registers and specific feature of the I/O port such as ADC Input or true open-drain.

Switching these I/O ports from one state to another should be done in a sequence that prevents unwanted side effects. Recommended safe transitions are illustrated in *Figure 32*. Other transitions are potentially risky and should be avoided, since they are likely to present unwanted side-effects such as spurious interrupt generation.

**Figure 32. Interrupt I/O port state transitions**




The I/O port register configurations are summarized in the following table.

**Table 32. I/O port configuration**

| Port | Pin name | Input (DDR = 0) | | Output (DDR = 1) | |
|---|---|---|---|---|---|
| | | OR = 0 | OR = 1 | OR = 0 | OR = 1 |
| Port A | PA7:6 | floating | | true open-drain | |
| | PA5:4 | floating | pull-up | open-drain | push-pull |
| | PA3 | floating | floating interrupt | open-drain | push-pull |
| | PA2:0 | floating | pull-up interrupt | open-drain | push-pull |

Obsolete Product(s) - Obsolete Product(s)

Table 32. I/O port configuration (continued)

| Port | Pin name | Input (DDR = 0) | | Output (DDR = 1) | |
|---|---|---|---|---|---|
| | | OR = 0 | OR = 1 | OR = 0 | OR = 1 |
| Port B | PB7, PB3 | floating | floating interrupt | open-drain | push-pull |
| | PB6:5, PB4, PB2:0 | floating | pull-up interrupt | open-drain | push-pull |
| Port C | PC7:0 | floating | pull-up | open-drain | push-pull |
| Port D | PD7:0 | floating | pull-up | open-drain | push-pull |
| Port E | PE7:3, PE1:0 | floating | pull-up | open-drain | push-pull |
| | PE2 | pull-up | | open-drain(1) | push-pull(1) |
| Port F | PF7:3 | floating | pull-up | open-drain | push-pull |
| | PF2 | floating | floating interrupt | open-drain | push-pull |
| | PF1:0 | floating | pull-up interrupt | open-drain | push-pull |

1. Pull-up is always enabled leading to unwanted power consumption if output is tied to low level

## 9.4 Low power modes

Table 33. Effect of low power modes on I/O ports

| Mode | Effect |
|---|---|
| Wait | No effect on I/O ports. External interrupts cause the device to exit from Wait mode. |
| Halt | No effect on I/O ports. External interrupts cause the device to exit from Halt mode. |

## 9.5 Interrupts

The external interrupt event generates an interrupt if the corresponding configuration is selected with DDR and OR registers and the interrupt mask in the CC register is not active (RIM instruction).

Table 34. I/O port interrupt control/wake-up capability

| Interrupt event | Event flag | Enable control bit | Exit from Wait | Exit from Halt |
|---|---|---|---|---|
| External interrupt on selected external event | - | DDRx, ORx | Yes | Yes |

Table 35. I/O port register map and reset values

| Address (Hex.) | Register label | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Reset value of all I/O port registers | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0000h | PADR | MSB | | | | | | | LSB |
| 0001h | PADDR | | | | | | | | |
| 0002h | PAOR | | | | | | | | |

Obsolete Product(s) - Obsolete Product(s)

Table 35. I/O port register map and reset values (continued)

| Address (Hex.) | Register label | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Reset value of all I/O port registers | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0003h | PBDR | MSB | | | | | | | LSB |
| 0004h | PBDDR | | | | | | | | |
| 0005h | PBOR | | | | | | | | |
| 0006h | PCDR | MSB | | | | | | | LSB |
| 0007h | PCDDR | | | | | | | | |
| 0008h | PCOR | | | | | | | | |
| 0009h | PDDR | MSB | | | | | | | LSB |
| 000Ah | PDDDR | | | | | | | | |
| 000Bh | PDOR | | | | | | | | |
| 000Ch | PEDR | MSB | | | | | | | LSB |
| 000Dh | PEDDR | | | | | | | | |
| 000Eh | PEOR | | | | | | | | |
| 000Fh | PFDR | MSB | | | | | | | LSB |
| 0010h | PFDDR | | | | | | | | |
| 0011h | PFOR | | | | | | | | |

## Related documentation

*SPI Communication between ST7 and EEPROM* (AN 970)

*S/W implementation of I2C bus master* (AN1045)

*Software LCD driver* (AN1048)

Obsolete Product(s) - Obsolete Product(s)

# 10 Watchdog timer (WDG)

## 10.1 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.

## 10.2 Main features

- Programmable free-running downcounter
- 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

## 10.3 Functional description

The counter value stored in the Watchdog Control register (WDGCR bits T[6:0]), is decremented every 16384 $f_{OSC2}$ cycles (approx.), and the length of the timeout 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 the $\overline{\text{RESET}}$ pin low for typically 30µs.

The application program must write in the WDGCR register at regular intervals during normal operation to prevent an MCU reset. This downcounter is free-running: It counts down even if the watchdog is disabled. The value to be stored in the WDGCR register must be between FFh and C0h:

- The WDGA bit is set (watchdog enabled)
- The T6 bit is set to prevent generating an immediate reset
- The T[5:0] bits contain the number of increments which represents the time delay before the watchdog produces a reset (see *Figure 34: Approximate timeout duration*). The timing varies between a minimum and a maximum value due to the unknown status of the prescaler when writing to the WDGCR register (see *Figure 35*).

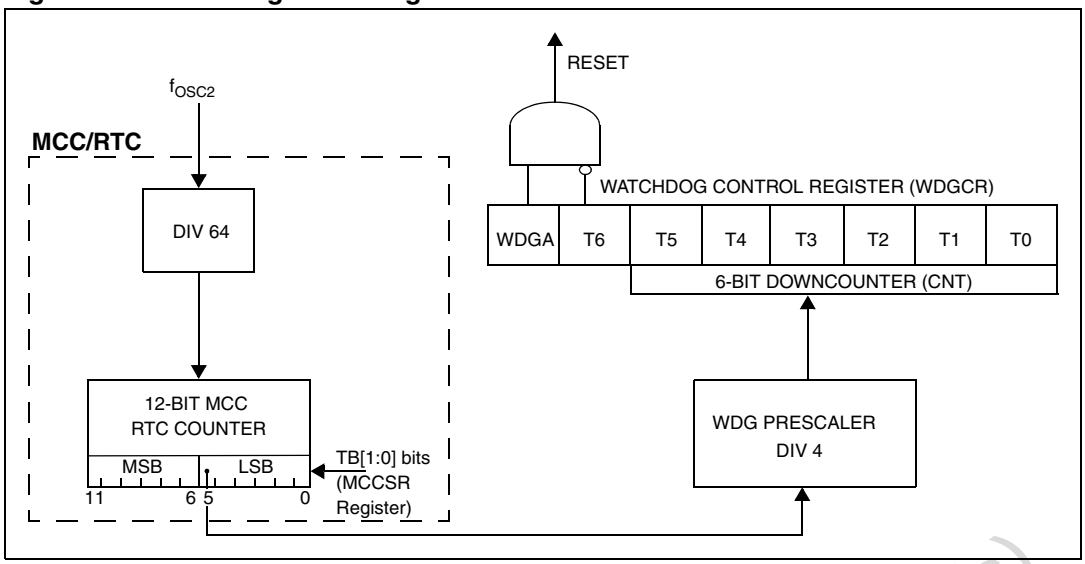
Following a reset, the watchdog is disabled. Once activated it cannot be disabled, except by a reset.

The T6 bit can be used to generate a software reset (the WDGA bit is set and the T6 bit is cleared).

If the watchdog is activated, the HALT instruction will generate a Reset.

Obsolete Product(s) - Obsolete Product(s)

**Figure 33. Watchdog block diagram**

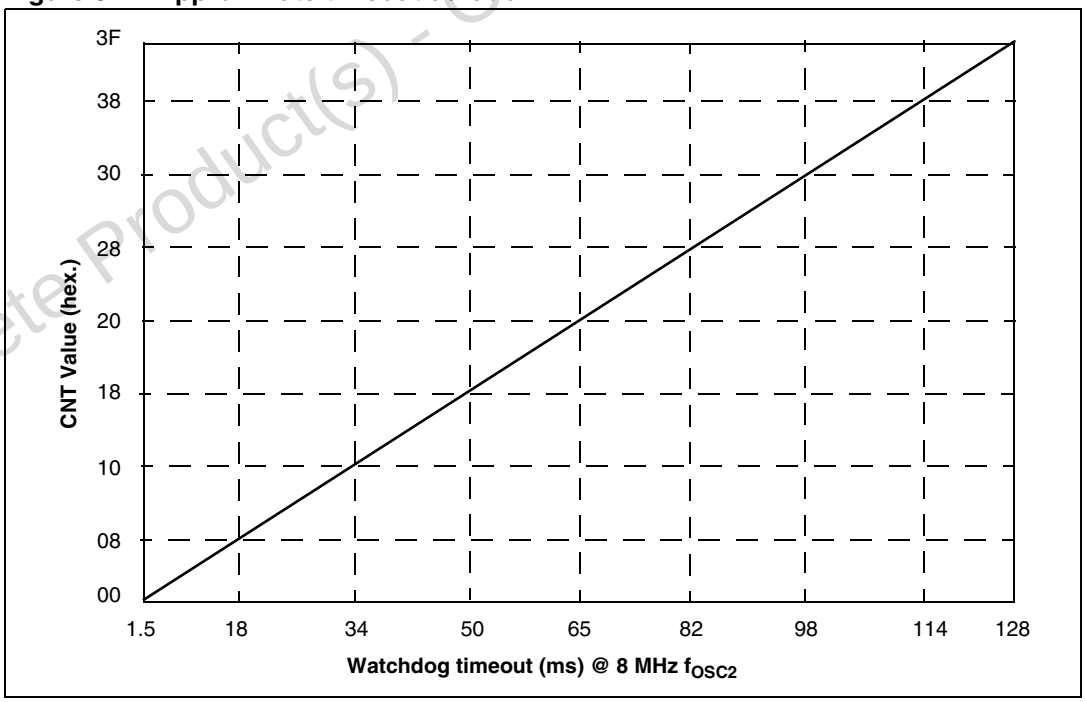



## 10.4 How to program the watchdog timeout

*Figure 34* shows the linear relationship between the 6-bit value to be loaded in the Watchdog Counter (CNT) and the resulting timeout duration in milliseconds. This can be used for a quick calculation without taking the timing variations into account. If more precision is needed, use the formulae in *Figure 35*.

**Caution:** When writing to the WDGCR register, always write 1 in the T6 bit to avoid generating an immediate reset.

**Figure 34. Approximate timeout duration**




Obsolete Product(s) - Obsolete Product(s)

**Figure 35. Exact timeout duration ($t_{min}$ and $t_{max}$)**

**WHERE**:

$t_{min0}$ = (LSB + 128) x 64 x $t_{OSC2}$

$t_{max0}$ = 16384 x $t_{OSC2}$

$t_{OSC2}$ = 125ns if $f_{OSC2}$= 8 MHz

CNT = Value of T[5:0] bits in the WDGCR register (6 bits)

MSB and LSB are values from the table below depending on the timebase selected by the TB[1:0] bits in the MCCSR register

| TB1 bit (MCCSR reg.) | TB0 bit (MCCSR reg.) | Selected MCCSR timebase | MSB | LSB |
|---|---|---|---|---|
| 0 | 0 | 2ms | 4 | 59 |
| 0 | 1 | 4ms | 8 | 53 |
| 1 | 0 | 10ms | 20 | 35 |
| 1 | 1 | 25ms | 49 | 54 |

**To calculate the minimum Watchdog Timeout ($t_{min}$):**

**IF** $CNT < \left[\frac{MSB}{4}\right]$ **THEN** $t_{min} = t_{min0} + 16384 \times CNT \times t_{osc2}$

**ELSE** $t_{min} = t_{min0} + \left[16384 \times \left(CNT - \left[\frac{4CNT}{MSB}\right]\right) + (192 + LSB) \times 64 \times \left[\frac{4CNT}{MSB}\right]\right] \times t_{osc2}$

**To calculate the maximum Watchdog Timeout ($t_{max}$):**

**IF** $CNT \leq \left[\frac{MSB}{4}\right]$ **THEN** $t_{max} = t_{max0} + 16384 \times CNT \times t_{osc2}$

**ELSE** $t_{max} = t_{max0} + \left[16384 \times \left(CNT - \left[\frac{4CNT}{MSB}\right]\right) + (192 + LSB) \times 64 \times \left[\frac{4CNT}{MSB}\right]\right] \times t_{osc2}$

**Note:** In the above formulae, division results must be rounded down to the next integer value.

**Example:**

With 2ms timeout selected in MCCSR register

| Value of T[5:0] bits in WDGCR register (Hex.) | Min. Watchdog Timeout (ms) $t_{min}$ | Max. Watchdog Timeout (ms) $t_{max}$ |
|---|---|---|
| 00 | 1.496 | 2.048 |
| 3F | 128 | 128.552 |

Obsolete Product(s) - Obsolete Product(s)

## 10.5 Low power modes

**Table 36. Effect of low power modes on WDG**

| Mode | Effect | | |
|---|---|---|---|
| Slow | No effect on Watchdog | | |
| Wait | No effect on Watchdog | | |
| Halt | **OIE bit in MCCSR register** | **WDGHALT bit in Option Byte** | |
| | 0 | 0 | No Watchdog reset is generated. The MCU enters Halt mode. The Watchdog counter is decremented once and then stops counting and is no longer able to generate a watchdog reset until the MCU receives an external interrupt or a reset.<br>If an external interrupt is received, the Watchdog restarts counting after 256 or 4096 CPU clocks. If a reset is generated, the Watchdog is disabled (reset state) unless Hardware Watchdog is selected by option byte. For application recommendations see *Section 10.7* below. |
| | 0 | 1 | A reset is generated. |
| | 1 | x | No reset is generated. The MCU enters Active Halt mode. The Watchdog counter is not decremented. It stop counting. When the MCU receives an oscillator interrupt or external interrupt, the Watchdog restarts counting immediately. When the MCU receives a reset the Watchdog restarts counting after 256 or 4096 CPU clocks. |

## 10.6 Hardware watchdog option

If Hardware Watchdog is selected by option byte, the watchdog is always active and the WDGA bit in the WDGCR is not used. Refer to the option byte description in *Section 21.1.1: Flash configuration*.

## 10.7 Using Halt mode with the WDG (WDGHALT option)

The following recommendation applies if Halt mode is used when the watchdog is enabled: Before executing the HALT instruction, refresh the WDG counter to avoid an unexpected WDG reset immediately after waking up the microcontroller.

## 10.8 Interrupts

None.

Obsolete Product(s) - Obsolete Product(s)

## 10.9 Register description

### 10.9.1 Control register (WDGCR)

WDGCR Reset value: 0111 1111 (7Fh)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| WDGA | T[6:0] | | | | | | |
| RW | RW | | | | | | |

**Table 37. WDGCR register description**

| Bit | Name | Function |
|---|---|---|
| 7 | WDGA | *Activation bit*<br>This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset.<br>0: Watchdog disabled<br>1: Watchdog enabled<br>*Note: This bit is not used if the hardware watchdog option is enabled by option byte.* |
| 6:0 | T[6:0] | *7-bit counter (MSB to LSB)*<br>These bits contain the value of the watchdog counter. It is decremented every 16384 $f_{OSC2}$ cycles (approx.). A reset is produced when it rolls over from 40h to 3Fh (T6 becomes cleared). |

**Table 38. Watchdog timer register map and reset values**

| Address (Hex.) | Register label | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 002Ah | WDGCR<br>Reset Value | WDGA<br>0 | T6<br>1 | T5<br>1 | T4<br>1 | T3<br>1 | T2<br>1 | T1<br>1 | T0<br>1 |

Obsolete Product(s) - Obsolete Product(s)

# 11 Main clock controller with real-time clock and beeper (MCC/RTC)

## 11.1 Introduction

The Main Clock Controller consists of three different functions:

- a programmable CPU clock prescaler
- a clock-out signal to supply external devices
- a real-time clock timer with interrupt capability

Each function can be used independently and simultaneously.

## 11.2 Programmable CPU clock prescaler

The programmable CPU clock prescaler supplies the clock for the ST7 CPU and its internal peripherals. It manages Slow power saving mode (see *Section 8.2: Slow mode* for more details).

The prescaler selects the $f_{CPU}$ main clock frequency and is controlled by three bits in the MCCSR register: CP[1:0] and SMS.

## 11.3 Clock-out capability

The clock-out capability is an alternate function of an I/O port pin that outputs a $f_{CPU}$ clock to drive external devices. It is controlled by the MCO bit in the MCCSR register.

**Caution:** When selected, the clock out pin suspends the clock during Active Halt mode.

## 11.4 Real-time clock timer (RTC)

The counter of the real-time clock timer allows an interrupt to be generated based on an accurate real-time clock. Four different time bases depending directly on $f_{OSC2}$ are available. The whole functionality is controlled by four bits of the MCCSR register: TB[1:0], OIE and OIF.

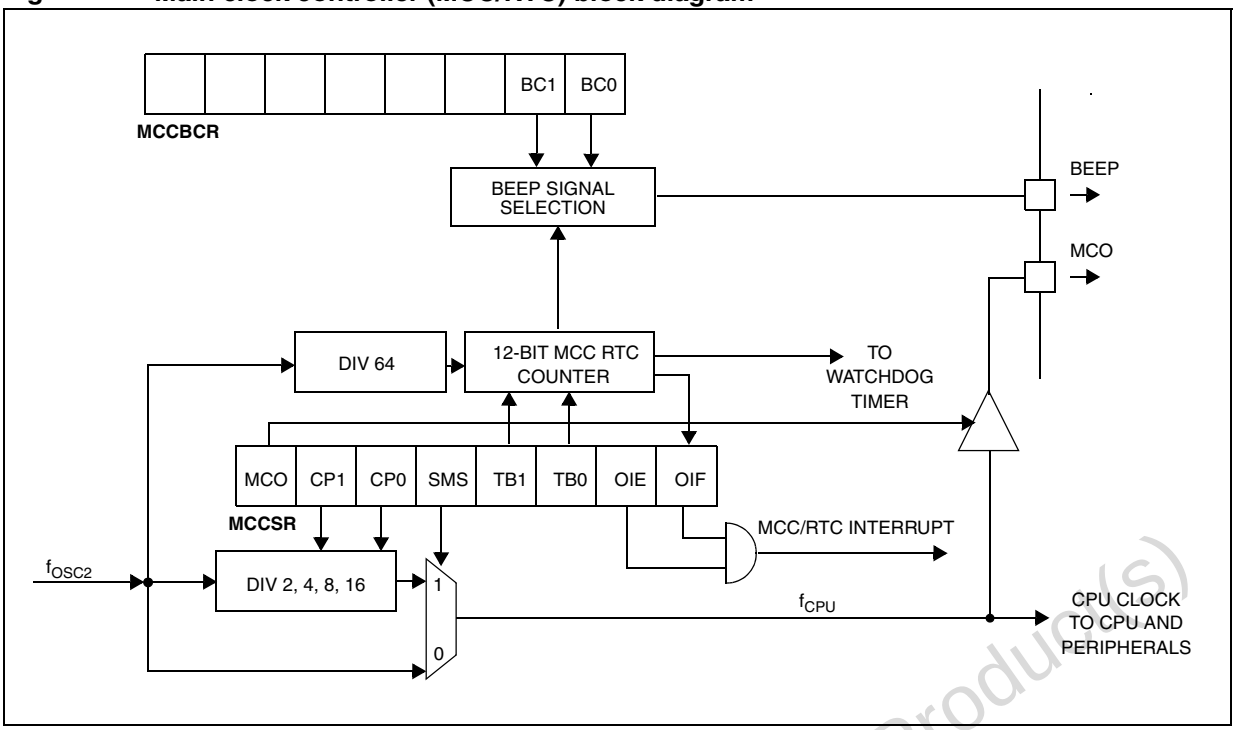
When the RTC interrupt is enabled (OIE bit set), the ST7 enters Active Halt mode when the HALT instruction is executed. See *Section 8.4: Active Halt and Halt modes* for more details.

## 11.5 Beeper

The beep function is controlled by the MCCBCR register. It can output three selectable frequencies on the BEEP pin (I/O port alternate function).

Obsolete Product(s) - Obsolete Product(s)

**Figure 36. Main clock controller (MCC/RTC) block diagram**




## 11.6 Low power modes

**Table 39. Effect of low power modes on MCC/RTC**

| Mode | Effect |
|---|---|
| Wait | No effect on MCC/RTC peripheral.<br>MCC/RTC interrupt causes the device to exit from Wait mode. |
| Active Halt | No effect on MCC/RTC counter (OIE bit is set), the registers are frozen.<br>MCC/RTC interrupt causes the device to exit from Active Halt mode. |
| Halt | MCC/RTC counter and registers are frozen.<br>MCC/RTC operation resumes when the MCU is woken up by an interrupt with "exit from HALT" capability. |

## 11.7 Interrupts

The MCC/RTC interrupt event generates an interrupt if the OIE bit of the MCCSR register is set and the interrupt mask in the CC register is not active (RIM instruction).

**Table 40. MCC/RTC interrupt control/wake-up capability**

| Interrupt event | Event flag | Enable control bit | Exit from Wait | Exit from Halt |
|---|---|---|---|---|
| Time base overflow event | OIF | OIE | Yes | No(1) |

1. The MCC/RTC interrupt wakes up the MCU from Active Halt mode, not from Halt mode.

Obsolete Product(s) - Obsolete Product(s)

## 11.8 Main clock controller registers

### 11.8.1 MCC control/status register (MCCSR)

MCCSR Reset value: 0000 0000 (00h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| MCO | CP[1:0] | | SMS | TB[1:0] | | OIE | OIF |
| RW | RW | | RW | RW | | RW | RW |

**Table 41. MCCSR register description**

| Bit | Name | Function |
|---|---|---|
| 7 | MCO | *Main clock out selection*<br>This bit enables the MCO alternate function on the PF0 I/O port. It is set and cleared by software.<br>0: MCO alternate function disabled (I/O pin free for general-purpose I/O)<br>1: MCO alternate function enabled ($f_{CPU}$ on I/O port)<br>*Note: To reduce power consumption, the MCO function is not active in Active Halt mode.* |
| 6:5 | CP[1:0] | *CPU clock prescaler*<br>These bits select the CPU clock prescaler which is applied in the different slow modes. Their action is conditioned by the setting of the SMS bit. These two bits are set and cleared by software.<br>00: $f_{CPU}$ in Slow mode = $f_{OSC2}/2$<br>01: $f_{CPU}$ in Slow mode = $f_{OSC2}/4$<br>10: $f_{CPU}$ in Slow mode = $f_{OSC2}/8$<br>11: $f_{CPU}$ in Slow mode = $f_{OSC2}/16$ |
| 4 | SMS | *Slow mode select*<br>This bit is set and cleared by software.<br>0: Normal mode. $f_{CPU} = f_{OSC2}$<br>1: Slow mode. $f_{CPU}$ is given by CP1, CP0<br>See *Section 8.2: Slow mode* and *Chapter 11: Main clock controller with real-time clock and beeper (MCC/RTC)* for more details. |
| 3:2 | TB[1:0] | *Time base control*<br>These bits select the programmable divider time base. They are set and cleared by software (see *Table 42*).<br>A modification of the time base is taken into account at the end of the current period (previously set) to avoid an unwanted time shift. This allows to use this time base as a real-time clock. |
| 1 | OIE | *Oscillator interrupt enable*<br>This bit set and cleared by software.<br>0: Oscillator interrupt disabled<br>1: Oscillator interrupt enabled<br>This interrupt can be used to exit from Active Halt mode.<br>When this bit is set, calling the ST7 software HALT instruction enters the Active Halt power saving mode. |

Obsolete Product(s) Obsolete Product(s)

**Table 41. MCCSR register description (continued)**

| Bit | Name | Function |
|---|---|---|
| 0 | OIF | *Oscillator interrupt flag*<br>This bit is set by hardware and cleared by software reading the MCCSR register. It indicates when set that the main oscillator has reached the selected elapsed time (TB1:0).<br>0: Timeout not reached<br>1: Timeout reached<br>**Caution**: The BRES and BSET instructions must not be used on the MCCSR register to avoid unintentionally clearing the OIF bit. |

**Table 42. Time base selection**

| Counter prescaler | Time base | | TB1 | TB0 |
|---|---|---|---|---|
| | $f_{OSC2}$ = 4 MHz | $f_{OSC2}$ = 8 MHz | | |
| 16000 | 4ms | 2ms | 0 | 0 |
| 32000 | 8ms | 4ms | 0 | 1 |
| 80000 | 20ms | 10ms | 1 | 0 |
| 200000 | 50ms | 25ms | 1 | 1 |

## 11.8.2 MCC beep control register (MCCBCR)

MCCBCR Reset value: 0000 0000 (00h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | BC[1:0] | |
| - | | | | | | RW | |

**Table 43. MCCBCR register description**

| Bit | Name | Function |
|---|---|---|
| 7:2 | - | Reserved, must be kept cleared. |
| 1:0 | BC[1:0] | *Beep control*<br>These 2 bits select the PF1 pin beep capability (see *Table 44*). |

**Table 44. Beep frequency selection**

| BC1 | BC0 | Beep mode with $f_{OSC2}$ = 8 MHz | |
|---|---|---|---|
| 0 | 0 | Off | |
| 0 | 1 | ~2 kHz | Output Beep signal ~50% duty cycle |
| 1 | 0 | ~1 kHz | |
| 1 | 1 | ~500 Hz | |

The beep output signal is available in Active Halt mode but has to be disabled to reduce consumption.

Obsolete Product(s) - Obsolete Product(s)

**Table 45. Main clock controller register map and reset values**

| Address (Hex.) | Register label | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 002Bh | SICSR<br>Reset value | AVDS<br>0 | AVDIE<br>0 | AVDF<br>0 | LVDRF<br>x | <br>0 | CSSIE<br>0 | CSSD<br>0 | WDGRF<br>x |
| 002Ch | MCCSR<br>Reset value | MCO<br>0 | CP1<br>0 | CP0<br>0 | SMS<br>0 | TB1<br>0 | TB0<br>0 | OIE<br>0 | OIF<br>0 |
| 002Dh | MCCBCR<br>Reset value | <br>0 | <br>0 | <br>0 | <br>0 | <br>0 | <br>0 | BC1<br>0 | BC0<br>0 |

Obsolete Product(s) - Obsolete Product(s)

# 12 PWM auto-reload timer (ART)

## 12.1 Introduction

The Pulse Width Modulated Auto-Reload Timer on-chip peripheral consists of an 8-bit auto-reload counter with compare/capture capabilities and of a 7-bit prescaler clock source.

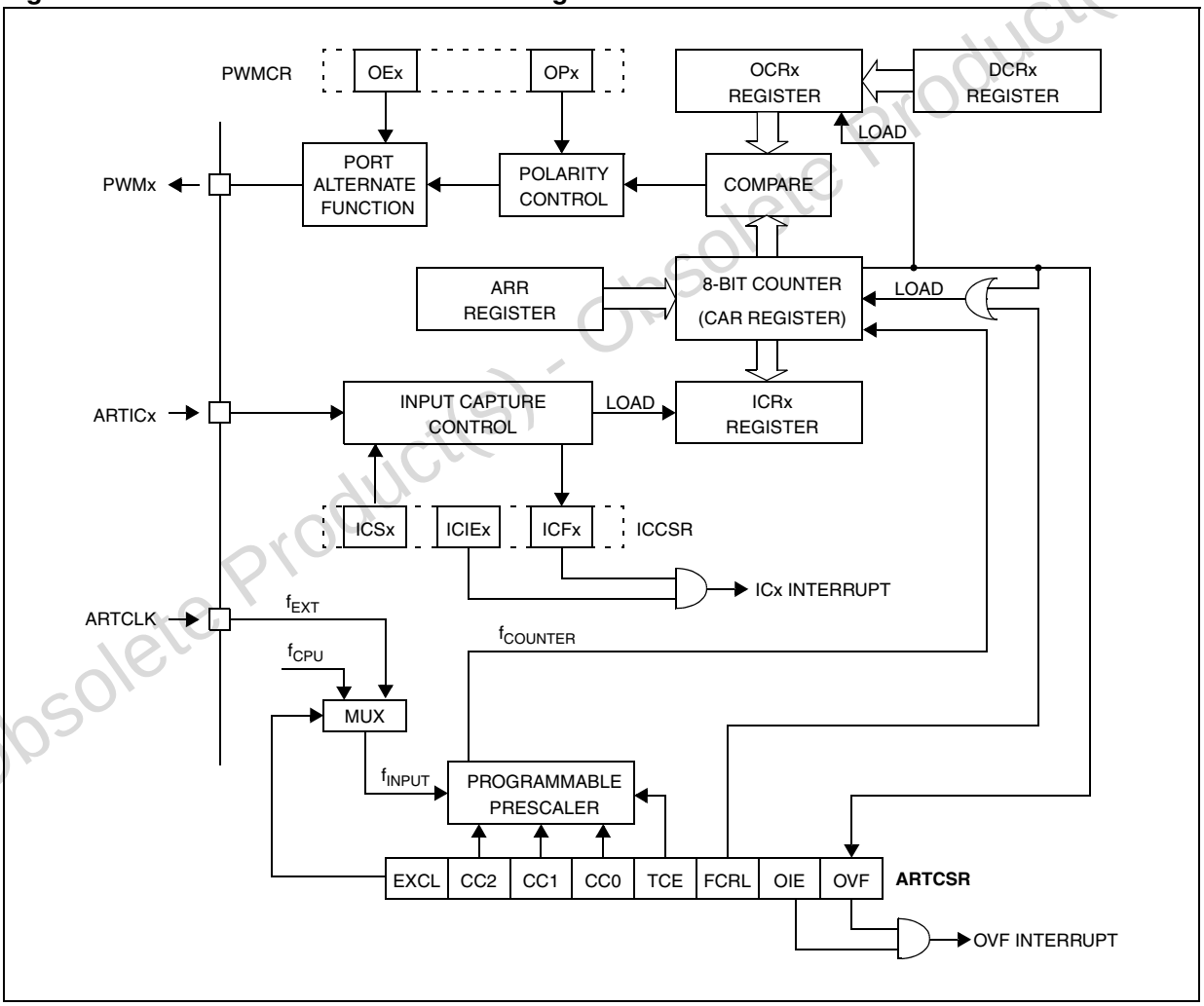
These resources allow five possible operating modes:

- Generation of up to 4 independent PWM signals
- Output compare and Time base interrupt
- Up to 2 input capture functions
- External event detector
- Up to 2 external interrupt sources

The three first modes can be used together with a single counter frequency.

The timer can be used to wake up the MCU from Wait and Halt modes.

**Figure 37. PWM auto-reload timer block diagram**




Obsolete Product(s) - Obsolete Product(s)

## 12.2 Functional description

### 12.2.1 Counter

The free running 8-bit counter is fed by the output of the prescaler, and is incremented on every rising edge of the clock signal.

It is possible to read or write the contents of the counter on the fly by reading or writing the Counter Access register (ARTCAR).

When a counter overflow occurs, the counter is automatically reloaded with the contents of the ARTARR register (the prescaler is not affected).

### 12.2.2 Counter clock and prescaler

The counter clock frequency is given by:

$$f_{COUNTER} = f_{INPUT} / 2^{CC[2:0]}$$

The timer counter's input clock ($f_{INPUT}$) feeds the 7-bit programmable prescaler, which selects one of the 8 available taps of the prescaler, as defined by CC[2:0] bits in the Control/Status Register (ARTCSR). Thus the division factor of the prescaler can be set to $2^n$ (where n = 0, 1,..7).

This $f_{INPUT}$ frequency source is selected through the EXCL bit of the ARTCSR register and can be either the $f_{CPU}$ or an external input frequency $f_{EXT}$.

The clock input to the counter is enabled by the TCE (Timer Counter Enable) bit in the ARTCSR register. When TCE is reset, the counter is stopped and the prescaler and counter contents are frozen. When TCE is set, the counter runs at the rate of the selected clock source.

### 12.2.3 Counter and prescaler initialization

After RESET, the counter and the prescaler are cleared and $f_{INPUT} = f_{CPU}$.

The counter can be initialized by:

- writing to the ARTARR register and then setting the FCRL (Force Counter Re-Load) and the TCE (Timer Counter Enable) bits in the ARTCSR register
- writing to the ARTCAR counter access register

In both cases the 7-bit prescaler is also cleared, whereupon counting will start from a known value.

Direct access to the prescaler is not possible.

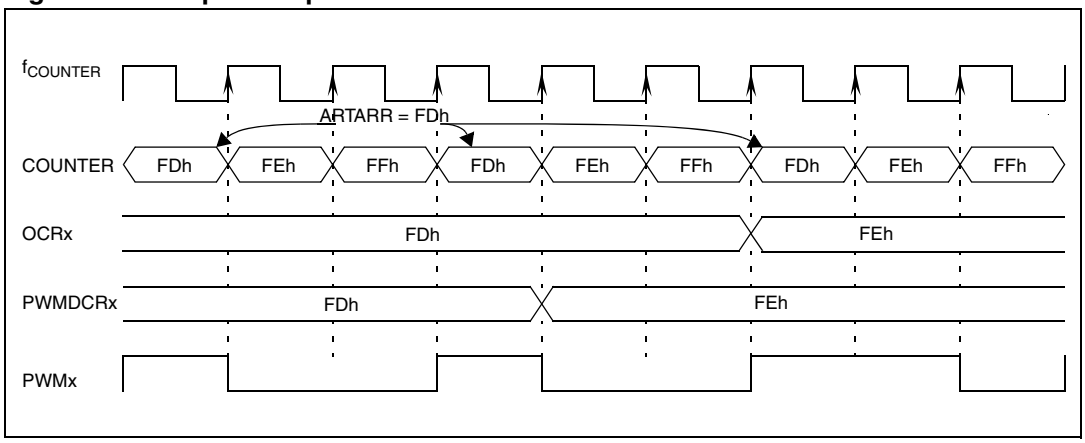
### 12.2.4 Output compare control

The timer compare function is based on four different comparisons with the counter (one for each PWMx output). Each comparison is made between the counter value and an output compare register (OCRx) value. This OCRx register can not be accessed directly, it is loaded from the duty cycle register (PWMDCRx) at each overflow of the counter.

This double buffering method avoids glitch generation when changing the duty cycle on the fly.

Obsolete Product(s) - Obsolete Product(s)

**Figure 38. Output compare control**




### 12.2.5 Independent PWM signal generation

This mode allows up to four Pulse Width Modulated signals to be generated on the PWMx output pins with minimum core processing overhead. This function is stopped during Halt mode.

Each PWMx output signal can be selected independently using the corresponding OEx bit in the PWM Control register (PWMCR). When this bit is set, the corresponding I/O pin is configured as output push-pull alternate function.

The PWM signals all have the same frequency which is controlled by the counter period and the ARTARR register value.

$$f_{PWM} = f_{COUNTER} / (256 - ARTARR)$$

When a counter overflow occurs, the PWMx pin level is changed depending on the corresponding OPx (output polarity) bit in the PWMCR register. When the counter reaches the value contained in one of the output compare register (OCRx) the corresponding PWMx pin level is restored.

It should be noted that the reload values will also affect the value and the resolution of the duty cycle of the PWM output signal. To obtain a signal on a PWMx pin, the contents of the OCRx register must be greater than the contents of the ARTARR register.

The maximum available resolution for the PWMx duty cycle is:

$$Resolution = 1 / (256 - ARTARR)$$

*Note: To get the maximum resolution (1/256), the ARTARR register must be 0. With this maximum resolution, 0% and 100% can be obtained by changing the polarity.*

Obsolete Product(s) - Obsolete Product(s)

**Figure 39. PWM auto-reload timer function**

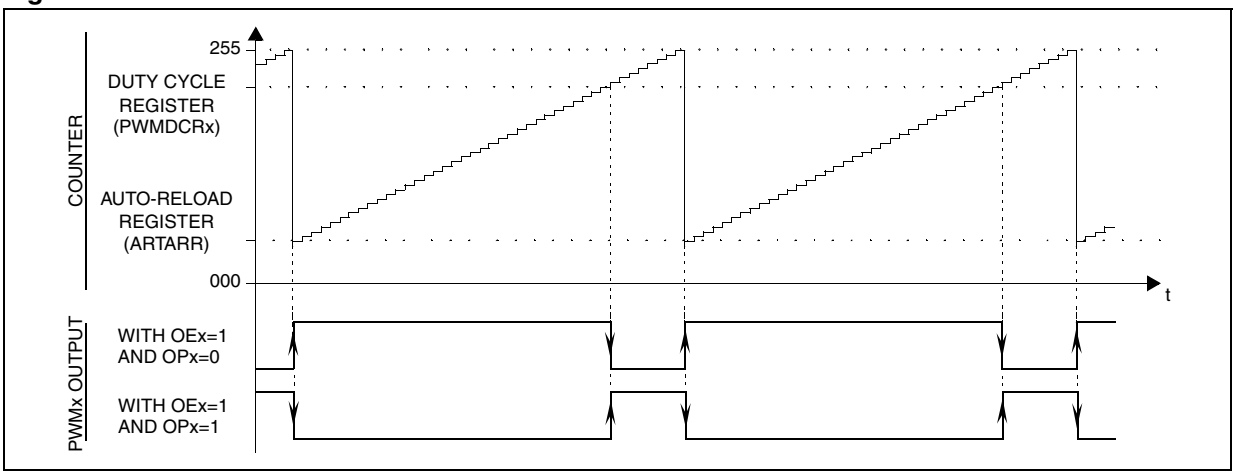
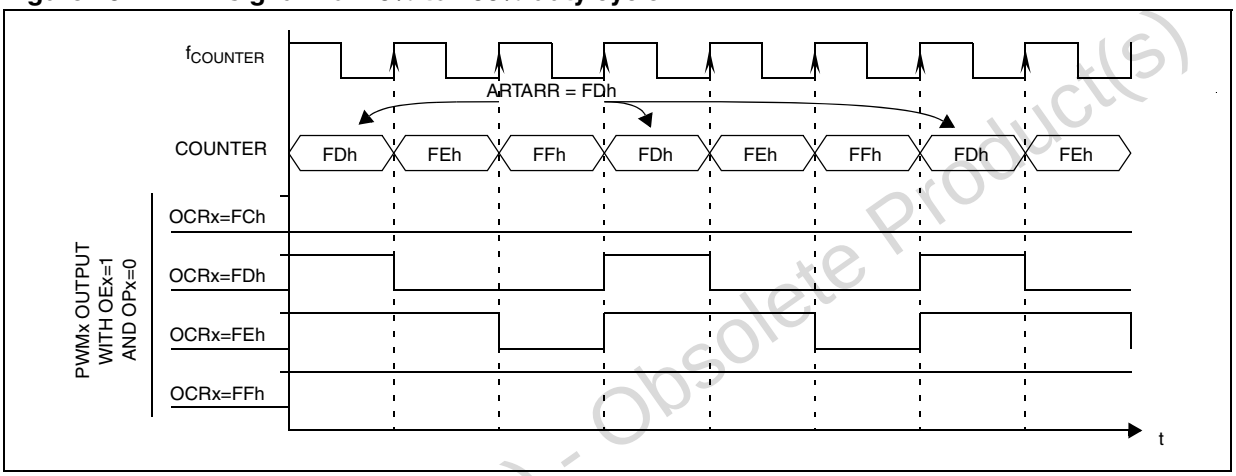



**Figure 40. PWM signal from 0% to 100% duty cycle**




Obsolete Product(s) - Obsolete Product(s)

## 12.2.6 Output compare and time base interrupt

On overflow, the OVF flag of the ARTCSR register is set and an overflow interrupt request is generated if the overflow interrupt enable bit, OIE, in the ARTCSR register, is set. The OVF flag must be reset by the user software. This interrupt can be used as a time base in the application.

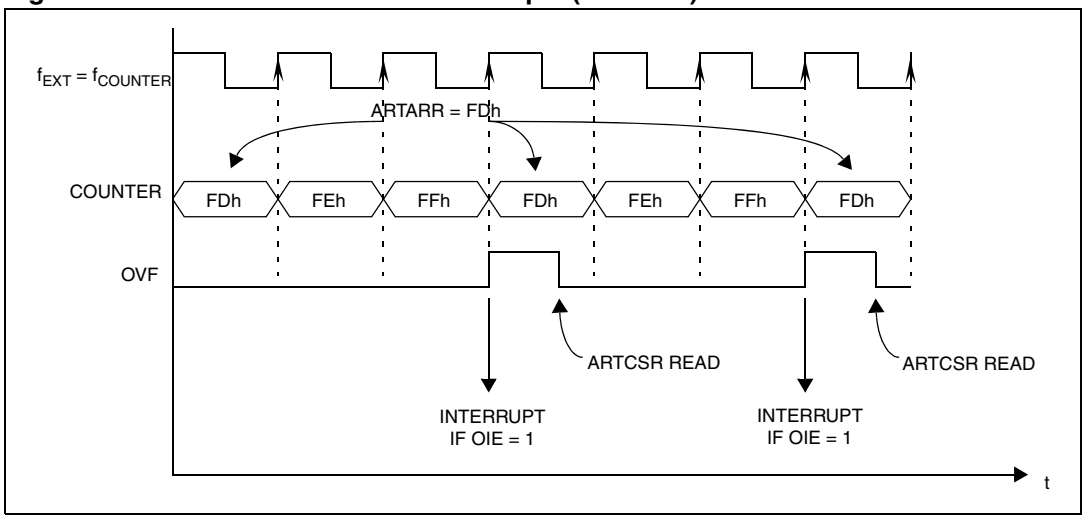
## 12.2.7 External clock and event detector mode

Using the $f_{EXT}$ external prescaler input clock, the auto-reload timer can be used as an external clock event detector. In this mode, the ARTARR register is used to select the $n_{EVENT}$ number of events to be counted before setting the OVF flag.

$$n_{EVENT} = 256 - ARTARR$$

**Caution:** The external clock function is not available in Halt mode. If Halt mode is used in the application, prior to executing the HALT instruction, the counter must be disabled by clearing the TCE bit in the ARTCSR register to avoid spurious counter increments.

**Figure 41. External event detector example (3 counts)**




## 12.2.8 Input capture function

This mode allows the measurement of external signal pulse widths through ARTICRx registers.

Each input capture can generate an interrupt independently on a selected input signal transition. This event is flagged by a set of the corresponding CFx bits of the Input Capture Control/Status register (ARTICCSR).

These input capture interrupts are enabled through the CIEx bits of the ARTICCSR register.

The active transition (falling or rising edge) is software programmable through the CSx bits of the ARTICCSR register.

The read only input capture registers (ARTICRx) are used to latch the auto-reload counter value when a transition is detected on the ARTICx pin (CFx bit set in ARTICCSR register). After fetching the interrupt vector, the CFx flags can be read to identify the interrupt source.

*Note: After a capture detection, data transfer in the ARTICRx register is inhibited until it is read (clearing the CFx bit).*

*The timer interrupt remains pending while the CFx flag is set when the interrupt is enabled (CIEx bit set). This means that the ARTICRx register has to be read at each capture event to clear the CFx flag.*

The timing resolution is given by auto-reload counter cycle time ($1/f_{COUNTER}$).

*Note: During Halt mode, if both the input capture and the external clock are enabled, the ARTICRx register value is not guaranteed if the input capture pin and the external clock change simultaneously.*

Obsolete Product(s) - Obsolete Product(s)

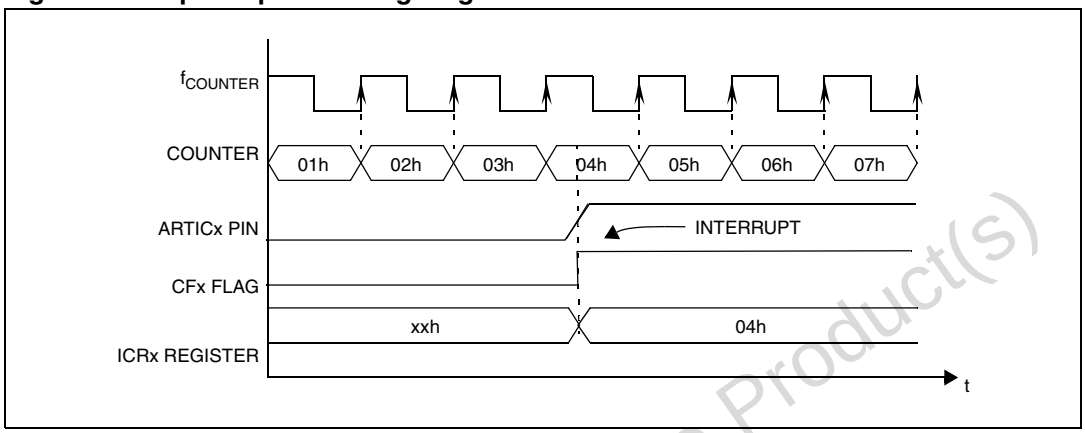
### 12.2.9 External interrupt capability

This mode allows the input capture capabilities to be used as external interrupt sources. The interrupts are generated on the edge of the ARTICx signal.

The edge sensitivity of the external interrupts is programmable (CSx bit of ARTICCSR register) and they are independently enabled through CIEx bits of the ARTICCSR register. After fetching the interrupt vector, the CFx flags can be read to identify the interrupt source.

During Halt mode, the external interrupts can be used to wake up the micro (if the CIEx bit is set).

**Figure 42. Input capture timing diagram**




Obsolete Product(s) - Obsolete Product(s)

## 12.3 ART registers

### 12.3.1 Control/status register (ARTCSR)

ARTCSR Reset value: 0000 0000 (00h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| EXCL | CC[2:0] | | | TCE | FCRL | OIE | OVF |
| RW | RW | | | RW | RW | RW | RW |

**Table 46. ARTCSR register description**

| Bit | Name | Function |
|---|---|---|
| 7 | EXCL | *External Clock*<br>This bit is set and cleared by software. It selects the input clock for the 7-bit prescaler.<br>0: CPU clock<br>1: External clock |
| 6:4 | CC[2:0] | *Counter Clock Control*<br>These bits are set and cleared by software. They determine the prescaler division ratio from $f_{INPUT}$ (see *Table 47*). |
| 3 | TCE | *Timer Counter Enable*<br>This bit is set and cleared by software. It puts the timer in the lowest power consumption mode.<br>0: Counter stopped (prescaler and counter frozen)<br>1: Counter running |
| 2 | FCRL | *Force Counter Re-Load*<br>This bit is write-only and any attempt to read it will yield a logical zero. When set, it causes the contents of ARTARR register to be loaded into the counter, and the content of the prescaler register to be cleared in order to initialize the timer before starting to count. |
| 1 | OIE | *Overflow Interrupt Enable*<br>This bit is set and cleared by software. It allows to enable/disable the interrupt which is generated when the OVF bit is set.<br>0: Overflow Interrupt disable<br>1: Overflow Interrupt enable |
| 0 | OVF | *Overflow Flag*<br>This bit is set by hardware and cleared by software reading the ARTCSR register. It indicates the transition of the counter from FFh to the ARTARR value.<br>0: New transition not yet reached<br>1: Transition reached |

**Table 47. Prescaler selection for ART**

| $f_{COUNTER}$ | With $f_{INPUT}$ = 8 MHz | CC2 | CC1 | CC0 |
|---|---|---|---|---|
| $f_{INPUT}$ | 8 MHz | 0 | 0 | 0 |
| $f_{INPUT}$ / 2 | 4 MHz | 0 | 0 | 1 |
| $f_{INPUT}$ / 4 | 2 MHz | 0 | 1 | 0 |

Obsolete Product(s) - Obsolete Product(s)

**Table 47. Prescaler selection for ART (continued)**

| $f_{COUNTER}$ | With $f_{INPUT}$ = 8 MHz | CC2 | CC1 | CC0 |
|---|---|---|---|---|
| $f_{INPUT}$ / 8 | 1 MHz | 0 | 1 | 1 |
| $f_{INPUT}$ / 16 | 500 kHz | 1 | 0 | 0 |
| $f_{INPUT}$ / 32 | 250 kHz | 1 | 0 | 1 |
| $f_{INPUT}$ / 64 | 125 kHz | 1 | 1 | 0 |
| $f_{INPUT}$ / 128 | 62.5 kHz | 1 | 1 | 1 |

### 12.3.2 Counter access register (ARTCAR)

ARTCAR Reset value: 0000 0000 (00h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CA[7:0] | | | | | | | |
| RW | | | | | | | |

**Table 48. ARTCAR register description**

| Bit | Name | Function |
|---|---|---|
| 7:0 | CA[7:0] | *Counter Access Data*<br>These bits can be set and cleared either by hardware or by software. The ARTCAR register is used to read or write the auto-reload counter “on the fly” (while it is counting). |

### 12.3.3 Auto-reload register (ARTARR)

ARTARR Reset value: 0000 0000 (00h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| AR[7:0] | | | | | | | |
| RW | | | | | | | |

**Table 49. ARTAAR register description**

| Bit | Name | Function |
|---|---|---|
| 7:0 | AR[7:0] | *Counter Auto-Reload Data*<br>These bits are set and cleared by software. They are used to hold the auto-reload value which is automatically loaded in the counter when an overflow occurs. At the same time, the PWM output levels are changed according to the corresponding OPx bit in the PWMCR register. |

This register has two PWM management functions:

- Adjusting the PWM frequency
- Setting the PWM duty cycle resolution

Obsolete Product(s) - Obsolete Product(s)

**Table 50. PWM frequency versus resolution**

| ARTARR value | Resolution | $f_{PWM}$ | |
|---|---|---|---|
| | | Min | Max |
| 0 | 8-bit | ~0.244 kHz | 31.25 kHz |
| [ 0..127 ] | > 7-bit | ~0.244 kHz | 62.5 kHz |
| [ 128..191 ] | > 6-bit | ~0.488 kHz | 125 kHz |
| [ 192..223 ] | > 5-bit | ~0.977 kHz | 250 kHz |
| [ 224..239 ] | > 4-bit | ~1.953 kHz | 500 kHz |

### 12.3.4 PWM control register (PWMCR)

PWMCR Reset value: 0000 0000 (00h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| OE[3:0] | | | | OP[3:0] | | | |
| RW | | | | RW | | | |

**Table 51. PWMCR register description**

| Bit | Name | Function |
|---|---|---|
| 7:4 | OE[3:0] | *PWM Output Enable*<br>These bits are set and cleared by software. They enable or disable the PWM output channels independently acting on the corresponding I/O pin.<br>0: PWM output disabled<br>1: PWM output enabled |
| 3:0 | OP[3:0] | *PWM Output Polarity*<br>These bits are set and cleared by software. They independently select the polarity of the four PWM output signals (see *Table 52*). |

**Table 52. PWM output signal polarity selection**

| PWMx output level | | OPx(1) |
|---|---|---|
| Counter <= OCRx | Counter > OCRx | |
| 1 | 0 | 0 |
| 0 | 1 | 1 |

1. When an OPx bit is modified, the PWMx output signal polarity is immediately reversed.

Obsolete Product(s) - Obsolete Product(s)

### 12.3.5 Duty cycle registers (PWMDCRx)

PWMDCRx Reset value: 0000 0000 (00h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DC[7:0] | | | | | | | |
| RW | | | | | | | |

**Table 53. PWMDCRx register description**

| Bit | Name | Function |
|---|---|---|
| 7:0 | DC[7:0] | *Duty Cycle Data*<br>These bits are set and cleared by software. |

A PWMDCRx register is associated with the OCRx register of each PWM channel to determine the second edge location of the PWM signal (the first edge location is common to all channels and given by the ARTARR register). These PWMDCR registers allow the duty cycle to be set independently for each PWM channel.

### 12.3.6 Input capture control / status register (ARTICCSR)

ARTICCSR Reset value: 0000 0000 (00h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | CS[2:1] | | CIE[2:1] | | CF[2:1] | |
| - | | RW | | RW | | RW | |

**Table 54. ARTICCSR register description**

| Bit | Name | Function |
|---|---|---|
| 7:6 | - | Reserved, always read as 0. |
| 5:4 | CS[2:1] | *Capture Sensitivity*<br>These bits are set and cleared by software. They determine the trigger event polarity on the corresponding input capture channel.<br>0: Falling edge triggers capture on channel x<br>1: Rising edge triggers capture on channel x |
| 3:2 | CIE[2:1] | *Capture Interrupt Enable*<br>These bits are set and cleared by software. They enable or disable the Input capture channel interrupts independently.<br>0: Input capture channel x interrupt disabled<br>1: Input capture channel x interrupt enabled |
| 1:0 | CF[2:1] | *Capture Flag*<br>These bits are set by hardware and cleared by software reading the corresponding ARTICRx register. Each CFx bit indicates that an input capture x has occurred.<br>0: No input capture on channel x<br>1: An input capture has occurred on channel x. |

Obsolete Product(s) - Obsolete Product(s)

### 12.3.7 Input capture registers (ARTICRx)

ARTICRx Reset value: 0000 0000 (00h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| IC[7:0] | | | | | | | |

RO

**Table 55. ARTICRx register description**

| Bit | Name | Function |
|---|---|---|
| 7:0 | IC[7:0] | *Input Capture Data*<br>These read only bits are set and cleared by hardware. An ARTICRx register contains the 8-bit auto-reload counter value transferred by the input capture channel x event. |

**Table 56. PWM auto-reload timer register map and reset values**

| Address (Hex.) | Register label | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0073h | PWMDCR3<br>Reset value | DC7<br>0 | DC6<br>0 | DC5<br>0 | DC4<br>0 | DC3<br>0 | DC2<br>0 | DC1<br>0 | DC0<br>0 |
| 0074h | PWMDCR2<br>Reset value | DC7<br>0 | DC6<br>0 | DC5<br>0 | DC4<br>0 | DC3<br>0 | DC2<br>0 | DC1<br>0 | DC0<br>0 |
| 0075h | PWMDCR1<br>Reset value | DC7<br>0 | DC6<br>0 | DC5<br>0 | DC4<br>0 | DC3<br>0 | DC2<br>0 | DC1<br>0 | DC0<br>0 |
| 0076h | PWMDCR0<br>Reset value | DC7<br>0 | DC6<br>0 | DC5<br>0 | DC4<br>0 | DC3<br>0 | DC2<br>0 | DC1<br>0 | DC0<br>0 |
| 0077h | PWMCR<br>Reset value | OE3<br>0 | OE2<br>0 | OE1<br>0 | OE0<br>0 | OP3<br>0 | OP2<br>0 | OP1<br>0 | OP0<br>0 |
| 0078h | ARTCSR<br>Reset value | EXCL<br>0 | CC2<br>0 | CC1<br>0 | CC0<br>0 | TCE<br>0 | FCRL<br>0 | RIE<br>0 | OVF<br>0 |
| 0079h | ARTCAR<br>Reset value | CA7<br>0 | CA6<br>0 | CA5<br>0 | CA4<br>0 | CA3<br>0 | CA2<br>0 | CA1<br>0 | CA0<br>0 |
| 007Ah | ARTARR<br>Reset value | AR7<br>0 | AR6<br>0 | AR5<br>0 | AR4<br>0 | AR3<br>0 | AR2<br>0 | AR1<br>0 | AR0<br>0 |
| 007Bh | ARTICCSR<br>Reset value | 0 | 0 | CS2<br>0 | CS1<br>0 | CIE2<br>0 | CIE1<br>0 | CF2<br>0 | CF1<br>0 |
| 007Ch | ARTICR1<br>Reset value | IC7<br>0 | IC6<br>0 | IC5<br>0 | IC4<br>0 | IC3<br>0 | IC2<br>0 | IC1<br>0 | IC0<br>0 |
| 007Dh | ARTICR2<br>Reset value | IC7<br>0 | IC6<br>0 | IC5<br>0 | IC4<br>0 | IC3<br>0 | IC2<br>0 | IC1<br>0 | IC0<br>0 |

Obsolete Product(s) - Obsolete Product(s)

# 13 16-bit timer

## 13.1 Introduction

The timer consists of a 16-bit free-running counter driven by a programmable prescaler.

It may be used for a variety of purposes, including pulse length measurement of up to two input signals (input capture) or generation of up to two output waveforms (output compare and PWM).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the CPU clock prescaler.

Some ST7 devices have two on-chip 16-bit timers. They are completely independent, and do not share any resources. They are synchronized after an MCU reset as long as the timer clock frequencies are not modified.

This description covers one or two 16-bit timers. In ST7 devices with two timers, register names are prefixed with TA (Timer A) or TB (Timer B).

## 13.2 Main features

- Programmable prescaler: $f_{CPU}$ divided by 2, 4 or 8
- Overflow status flag and maskable interrupt
- External clock input (must be at least four times slower than the CPU clock speed) with the choice of active edge
- 1 or 2 Output Compare functions each with:
  - 2 dedicated 16-bit registers
  - 2 dedicated programmable signals
  - 2 dedicated status flags
  - 1 dedicated maskable interrupt
- 1 or 2 Input Capture functions each with:
  - 2 dedicated 16-bit registers
  - 2 dedicated active edge selection signals
  - 2 dedicated status flags
  - 1 dedicated maskable interrupt
- Pulse Width Modulation mode (PWM)
- One Pulse mode
- Reduced Power mode
- 5 alternate functions on I/O ports (ICAP1, ICAP2, OCMP1, OCMP2, EXTCLK)[a]

The block diagram is shown in *Figure 43*.

*Note: When reading an input signal on a non-bonded pin, the value will always be '1'.*

a. Some timer pins may not be available (not bonded) in some ST7 devices. Refer to the device pinout description.

Obsolete Product(s) - Obsolete Product(s)

## 13.3 Functional description

### 13.3.1 Counter

The main block of the Programmable Timer is a 16-bit free running upcounter and its associated 16-bit registers. The 16-bit registers are made up of two 8-bit registers called high and low.

**Counter Register (CR)**

- Counter High Register (CHR) is the most significant byte (MS Byte)
- Counter Low Register (CLR) is the least significant byte (LS Byte)

**Alternate Counter Register (ACR)**

- Alternate Counter High Register (ACHR) is the most significant byte (MS Byte)
- Alternate Counter Low Register (ACLR) is the least significant byte (LS Byte)

These two read-only 16-bit registers contain the same value but with the difference that reading the ACLR register does not clear the TOF bit (Timer overflow flag), located in the Status register (SR) (see note at the end of paragraph entitled *16-bit read sequence*).

Writing in the CLR register or ACLR register resets the free running counter to the FFFCh value.

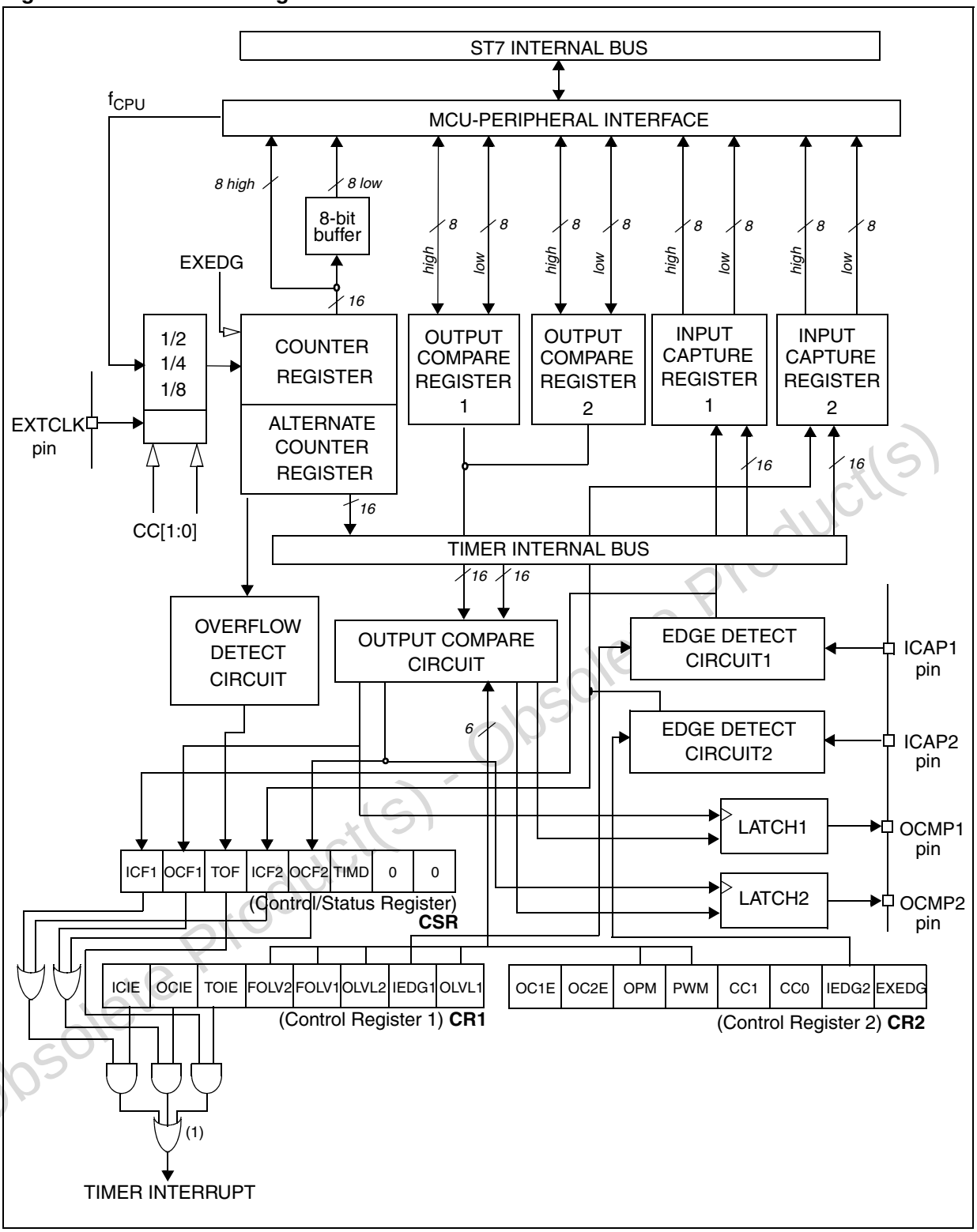
Both counters have a reset value of FFFCh (this is the only value which is reloaded in the 16-bit timer). The reset value of both counters is also FFFCh in One Pulse mode and PWM mode.

The timer clock depends on the clock control bits of the CR2 register, as illustrated in *Table 62: Timer clock selection*. The value in the counter register repeats every 131072, 262144 or 524288 CPU clock cycles depending on the CC[1:0] bits.

The timer frequency can be $f_{CPU}/2$, $f_{CPU}/4$, $f_{CPU}/8$ or an external frequency.

Obsolete Product(s) - Obsolete Product(s)

**Figure 43. Timer block diagram**



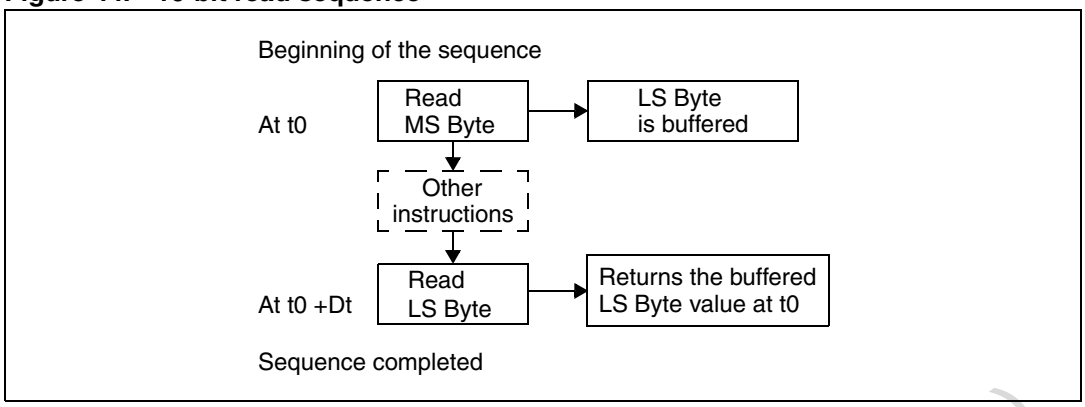

1. If IC, OC and TO interrupt request have separate vectors, then the last OR is not present (see device interrupt vector table).

Obsolete Product(s) - Obsolete Product(s)

### 16-bit read sequence

The 16-bit read sequence (from either the Counter Register or the Alternate Counter Register) is illustrated in *Figure 44*.

**Figure 44. 16-bit read sequence**




The user must read the MS Byte first; the LS Byte value is then buffered automatically.

This buffered value remains unchanged until the 16-bit read sequence is completed, even if the user reads the MS Byte several times.

After a complete reading sequence, if only the CLR register or ACLR register are read, they return the LS Byte of the count value at the time of the read.

Whatever timer mode is used (input capture, output compare, one pulse mode or PWM mode) an overflow occurs when the counter rolls over from FFFFh to 0000h, after which

- the TOF bit of the SR register is set
- a timer interrupt is generated if
  - the TOIE bit of the CR1 register is set and
  - the I bit of the CC register is cleared

If one of these conditions is false, the interrupt remains pending to be issued as soon as they are both true.

Clearing the overflow interrupt request is done in two steps:

1. Reading the SR register while the TOF bit is set
2. An access (read or write) to the CLR register

*Note: The TOF bit is not cleared by accesses to ACLR register. The advantage of accessing the ACLR register rather than the CLR register is that it allows simultaneous use of the overflow function and reading the free running counter at random times (for example, to measure elapsed time) without the risk of clearing the TOF bit erroneously.*

The timer is not affected by Wait mode.

In Halt mode, the counter stops counting until the mode is exited. Counting then resumes from the previous count (MCU awakened by an interrupt) or from the reset count (MCU awakened by a Reset).

Obsolete Product(s) - Obsolete Product(s)

### 13.3.2 External clock

The external clock (where available) is selected if CC0 = 1 and CC1 = 1 in the CR2 register.

The status of the EXEDG bit in the CR2 register determines the type of level transition on the external clock pin EXTCLK that will trigger the free running counter.

The counter is synchronized with the falling edge of the internal CPU clock.

A minimum of four falling edges of the CPU clock must occur between two consecutive active edges of the external clock; thus, the external clock frequency must be less than a quarter of the CPU clock frequency.

**Figure 45. Counter timing diagram, internal clock divided by 2**

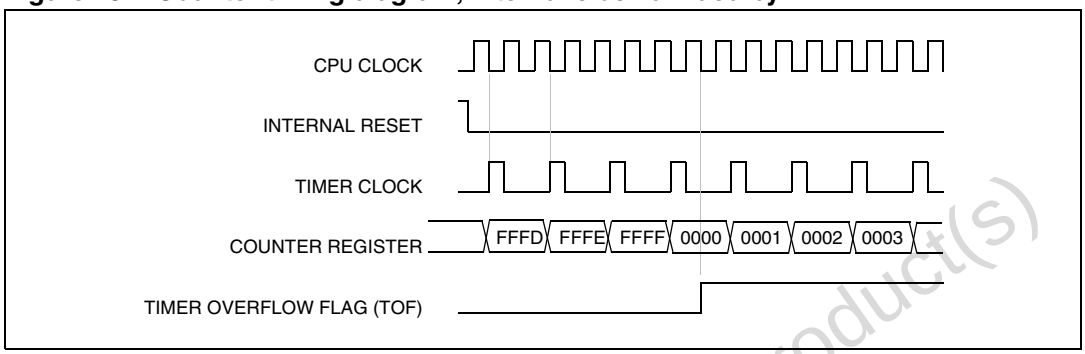



**Figure 46. Counter timing diagram, internal clock divided by 4**

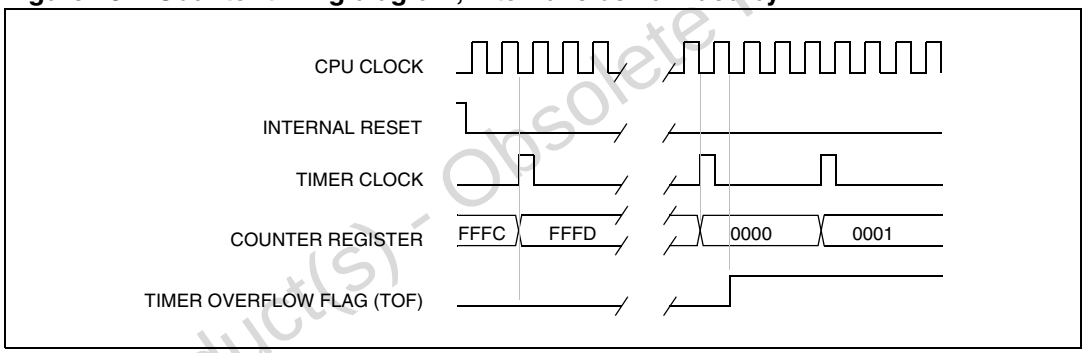
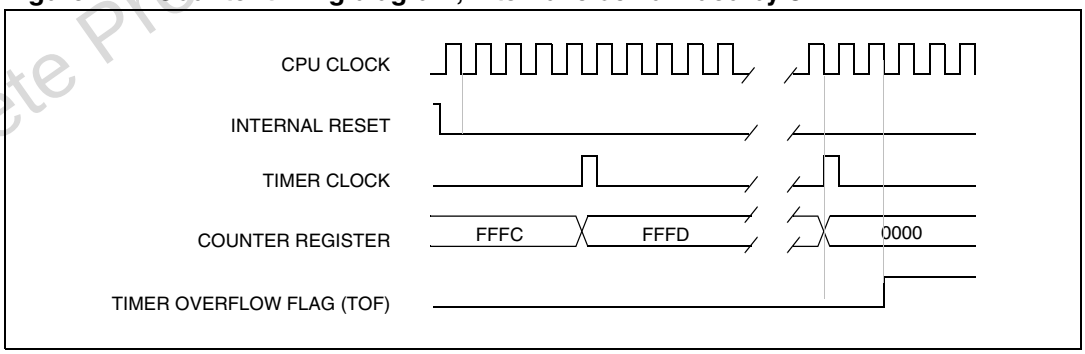



**Figure 47. Counter timing diagram, internal clock divided by 8**




*Note: The MCU is in reset state when the internal reset signal is high; when it is low the MCU is running.*

Obsolete Product(s) - Obsolete Product(s)

### 13.3.3 Input capture

In this section, the index, *i*, may be 1 or 2 because there are two input capture functions in the 16-bit timer.

The two 16-bit input capture registers (IC1R and IC2R) are used to latch the value of the free running counter after a transition is detected on the ICAP*i* pin (see *Figure 48*).

| | MS Byte | LS Byte |
|---|---|---|
| ICiR | IC*i*HR | IC*i*LR |

IC*i*R register is a read-only register.

The active transition is software programmable through the IEDG*i* bit of Control Registers (CR*i*).

Timing resolution is one count of the free running counter: ($f_{CPU}$/CC[1:0]).

Procedure:

To use the input capture function select the following in the CR2 register:

- Select the timer clock (CC[1:0]) (see *Table 62: Timer clock selection*).
- Select the edge of the active transition on the ICAP2 pin with the IEDG2 bit (the ICAP2 pin must be configured as floating input or input with pull-up without interrupt if this configuration is available).

And select the following in the CR1 register:

- Set the ICIE bit to generate an interrupt after an input capture coming from either the ICAP1 pin or the ICAP2 pin
- Select the edge of the active transition on the ICAP1 pin with the IEDG1 bit (the ICAP1pin must be configured as floating input or input with pull-up without interrupt if this configuration is available).

When an input capture occurs:

- ICF*i* bit is set.
- The IC*i*R register contains the value of the free running counter on the active transition on the ICAP*i* pin (see *Figure 49*).
- A timer interrupt is generated if the ICIE bit is set and the I bit is cleared in the CC register. Otherwise, the interrupt remains pending until both conditions become true.

Clearing the input capture interrupt request (that is, clearing the ICF*i* bit) is done in two steps:

1. Reading the SR register while the ICF*i* bit is set
2. An access (read or write) to the IC*i*LR register

*Note: 1 After reading the ICiHR register, transfer of input capture data is inhibited and ICFi will never be set until the ICiLR register is also read.*

*2 The ICiR register contains the free running counter value which corresponds to the most recent input capture.*

*3 The two input capture functions can be used together even if the timer also uses the two output compare functions.*

*4 In One pulse Mode and PWM mode only Input Capture 2 can be used.*

Obsolete Product(s) - Obsolete Product(s)

5 *The alternate inputs (ICAP1 and ICAP2) are always directly connected to the timer. So any transitions on these pins activates the input capture function.*

6 *Moreover if one of the ICAPi pins is configured as an input and the second one as an output, an interrupt can be generated if the user toggles the output pin and if the ICIE bit is set.*

7 *This can be avoided if the input capture function i is disabled by reading the ICiHR (see note 1).*

8 *The TOF bit can be used with interrupt generation in order to measure events that go beyond the timer range (FFFFh).*

**Figure 48. Input capture block diagram**

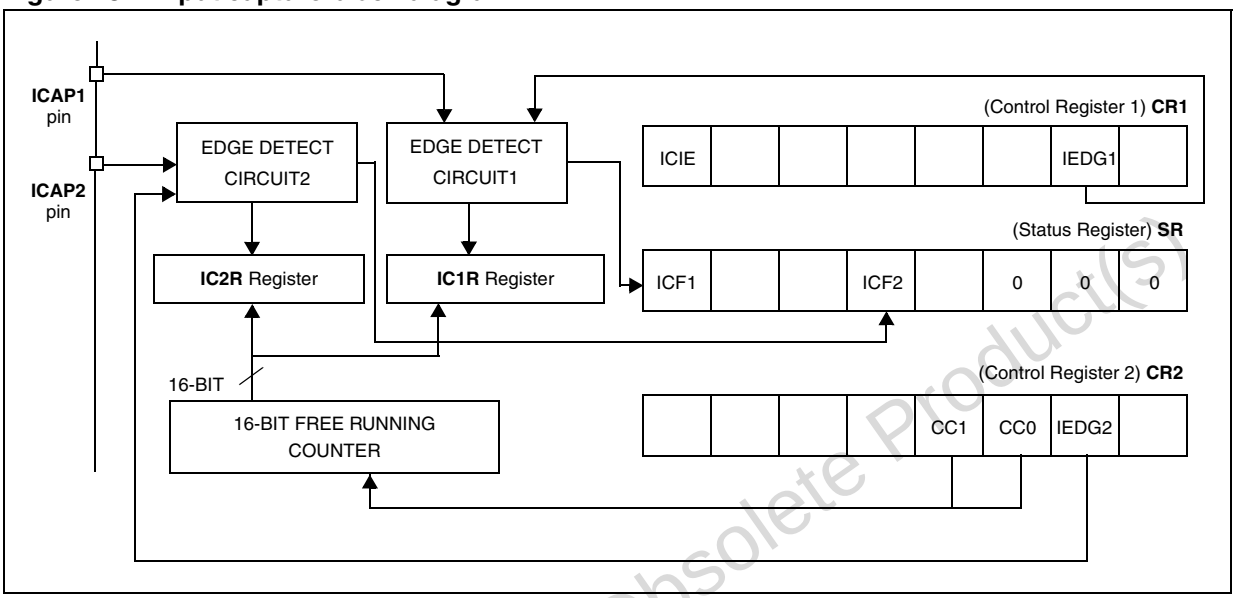
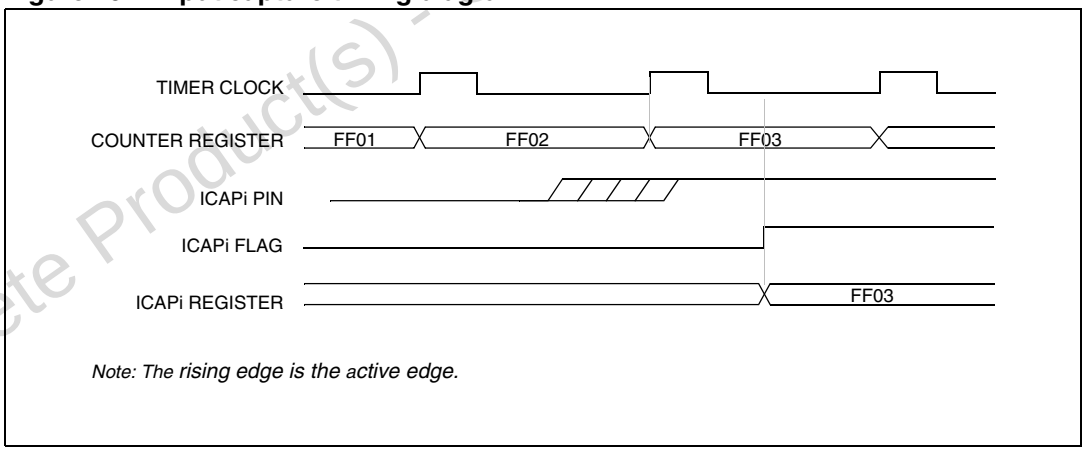



**Figure 49. Input capture timing diagram**




*Note: The rising edge is the active edge.*

Obsolete Product(s) - Obsolete Product(s)

### 13.3.4 Output compare

In this section, the index, *i*, may be 1 or 2 because there are two output compare functions in the 16-bit timer.

This function can be used to control an output waveform or indicate when a period of time has elapsed.

When a match is found between the Output Compare register and the free running counter, the output compare function:

- Assigns pins with a programmable value if the OC*i*E bit is set
- Sets a flag in the status register
- Generates an interrupt if enabled

Two 16-bit registers Output Compare Register 1 (OC1R) and Output Compare Register 2 (OC2R) contain the value to be compared to the counter register each timer clock cycle.

| | MS byte | LS byte |
|---|---|---|
| OC*i*R | OC*i*HR | OC*i*LR |

These registers are readable and writable and are not affected by the timer hardware. A reset event changes the OC*i*R value to 8000h.

Timing resolution is one count of the free running counter: ($f_{CPU/CC[1:0]}$).

#### Procedure

To use the output compare function, select the following in the CR2 register:

- Set the OC*i*E bit if an output is needed then the OCMP*i* pin is dedicated to the output compare *i* signal.
- Select the timer clock (CC[1:0]) (see *Table 62: Timer clock selection*).

And select the following in the CR1 register:

- Select the OLVL*i* bit to applied to the OCMP*i* pins after the match occurs.
- Set the OCIE bit to generate an interrupt if it is needed.

When a match is found between OCRi register and CR register:

- OCF*i* bit is set.
- The OCMP*i* pin takes OLVL*i* bit value (OCMP*i* pin latch is forced low during reset).
- A timer interrupt is generated if the OCIE bit is set in the CR1 register and the I bit is cleared in the CC register (CC).

The OC*i*R register value required for a specific timing application can be calculated using the following formula:

$$\Delta \text{ OC}i\text{R} = \frac{\Delta t * f_{CPU}}{\text{PRESC}}$$

Where:

$\Delta t$ = Output compare period (in seconds)

$f_{CPU}$ = CPU clock frequency (in hertz)

PRESC = Timer prescaler factor (2, 4 or 8 depending on CC[1:0] bits; see *Table 62: Timer clock selection*)

Obsolete Product(s) - Obsolete Product(s)

If the timer clock is an external clock, the formula is:

$$\Delta \text{ OC}i\text{R} = \Delta t * f_{EXT}$$

Where:

$\Delta t$ = Output compare period (in seconds)

$f_{CPU}$ = External timer clock frequency (in hertz)

Clearing the output compare interrupt request (that is, clearing the OCF*i* bit) is done by:

1. Reading the SR register while the OCF*i* bit is set
2. An access (read or write) to the OC*i*LR register

The following procedure is recommended to prevent the OCF*i* bit from being set between the time it is read and the write to the OC*i*R register:

- Write to the OC*i*HR register (further compares are inhibited).
- Read the SR register (first step of the clearance of the OCF*i* bit, which may be already set).
- Write to the OC*i*LR register (enables the output compare function and clears the OCF*i* bit).

*Note: 1 After a processor write cycle to the OCiHR register, the output compare function is inhibited until the OCiLR register is also written.*

*2 If the OCiE bit is not set, the OCMPi pin is a general I/O port and the OLVLi bit will not appear when a match is found but an interrupt could be generated if the OCIE bit is set.*

*3 In both internal and external clock modes, OCFi and OCMPi are set while the counter value equals the OCiR register value (see Figure 51 on page 111 for an example with $f_{CPU}/2$ and Figure 52 on page 111 for an example with $f_{CPU}/4$). This behavior is the same in OPM or PWM mode.*

*4 The output compare functions can be used both for generating external events on the OCMPi pins even if the input capture mode is also used.*

*5 The value in the 16-bit OCiR register and the OLVi bit should be changed after each successful comparison in order to control an output waveform or establish a new elapsed timeout.*

### 13.3.5 Forced compare output capability

When the FOLV*i* bit is set by software, the OLVL*i* bit is copied to the OCMP*i* pin. The OLV*i* bit has to be toggled in order to toggle the OCMP*i* pin when it is enabled (OC*i*E bit = 1). The OCF*i* bit is then not set by hardware, and thus no interrupt request is generated.

The FOLVL*i* bits have no effect in both one pulse mode and PWM mode.

Obsolete Product(s) - Obsolete Product(s)

**Figure 50. Output compare block diagram**

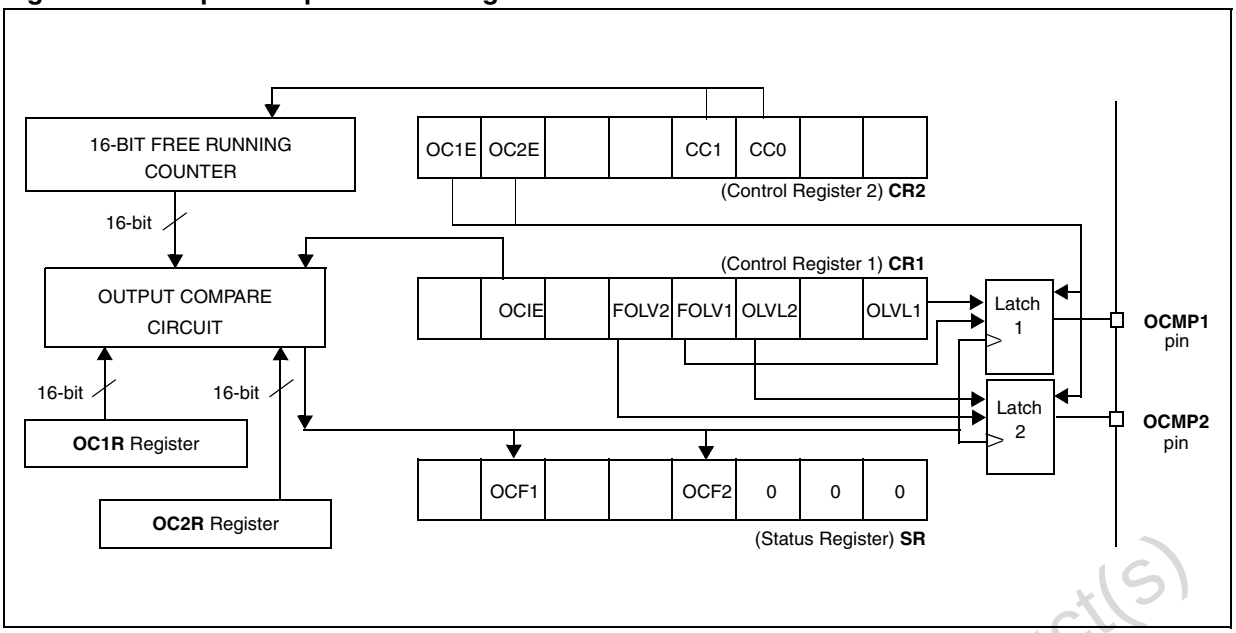



**Figure 51. Output compare timing diagram, $f_{TIMER} = f_{CPU}/2$**

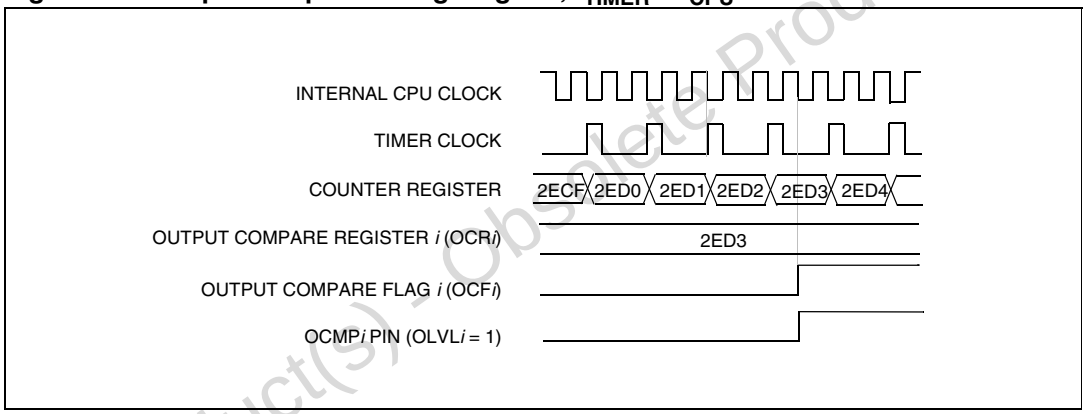
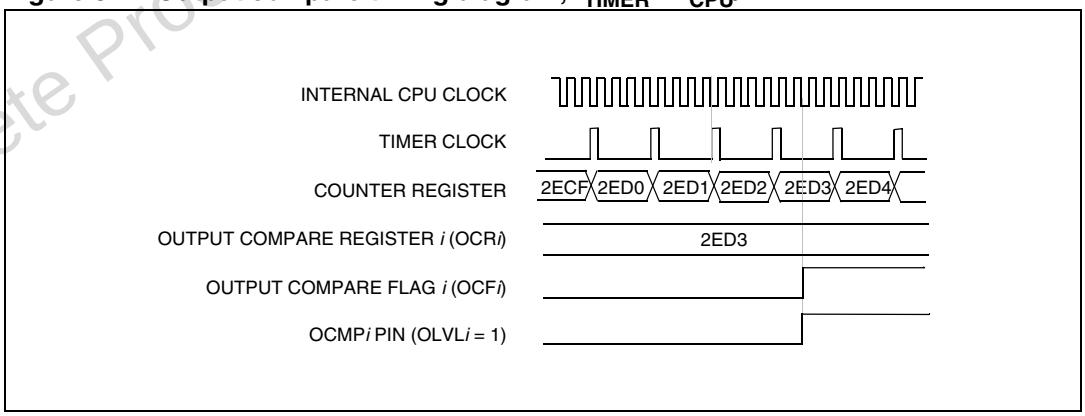



**Figure 52. Output compare timing diagram, $f_{TIMER} = f_{CPU}/4$**




Obsolete Product(s) - Obsolete Product(s)

### 13.3.6 One Pulse mode

One Pulse mode enables the generation of a pulse when an external event occurs. This mode is selected via the OPM bit in the CR2 register.

The one pulse mode uses the Input Capture1 function and the Output Compare1 function.

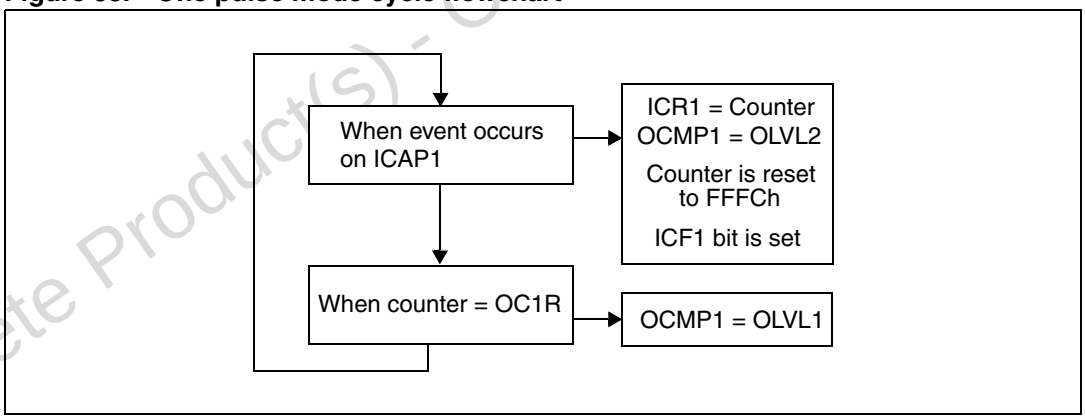
#### Procedure

To use one pulse mode:

1. Load the OC1R register with the value corresponding to the length of the pulse (using the appropriate formula below according to the timer clock source used).
2. Select the following in the CR1 register:
   - Using the OLVL1 bit, select the level to be applied to the OCMP1 pin after the pulse.
   - Using the OLVL2 bit, select the level to be applied to the OCMP1 pin during the pulse.
   - Select the edge of the active transition on the ICAP1 pin with the IEDG1 bit (the ICAP1 pin must be configured as floating input).
3. Select the following in the CR2 register:
   - Set the OC1E bit, the OCMP1 pin is then dedicated to the Output Compare 1 function.
   - Set the OPM bit.
   - Select the timer clock CC[1:0] (see *Table 62: Timer clock selection*).

Then, on a valid event on the ICAP1 pin, the counter is initialized to FFFCh and OLVL2 bit is loaded on the OCMP1 pin, the ICF1 bit is set and the value FFFDh is loaded in the IC1R register.

**Figure 53. One pulse mode cycle flowchart**




Because the ICF1 bit is set when an active edge occurs, an interrupt can be generated if the ICIE bit is set.

Clearing the input capture interrupt request (that is, clearing the ICF*i* bit) is done in two steps:

1. Reading the SR register while the ICF*i* bit is set
2. An access (read or write) to the IC*i*LR register

Obsolete Product(s) - Obsolete Product(s)

The OC1R register value required for a specific timing application can be calculated using the following formula:

$$OC\mathit{i}R\ value = \frac{t * f_{CPU} - 5}{PRESC}$$

Where:

t = Pulse period (in seconds)

$f_{CPU}$ = CPU clock frequency (in hertz)

PRESC = Timer prescaler factor (2, 4 or 8 depending on the CC[1:0] bits; see *Table 62: Timer clock selection*)

If the timer clock is an external clock the formula is:

$$OC\mathit{i}R = t * f_{EXT} - 5$$

Where:

t = Pulse period (in seconds)

$f_{EXT}$ = External clock frequency (in hertz)

When the value of the counter is equal to the value of the contents of the OC1R register, the OLVL1 bit is output on the OCMP1 pin (see *Figure 54*).

*Note: 1 The OCF1 bit cannot be set by hardware in one pulse mode but the OCF2 bit can generate an Output Compare interrupt.*

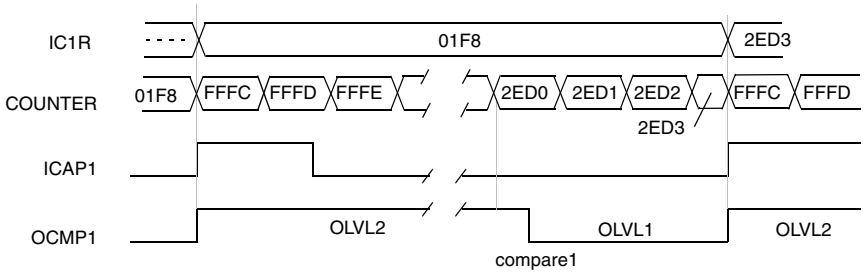
*2 When the Pulse Width Modulation (PWM) and One Pulse Mode (OPM) bits are both set, the PWM mode is the only active one.*

*3 If OLVL1 = OLVL2 a continuous signal will be seen on the OCMP1 pin.*

*4 The ICAP1 pin cannot be used to perform input capture. The ICAP2 pin can be used to perform input capture (ICF2 can be set and IC2R can be loaded) but the user must take care that the counter is reset each time a valid edge occurs on the ICAP1 pin and ICF1 can also generates interrupt if ICIE is set.*

*5 When one pulse mode is used OC1R is dedicated to this mode. Nevertheless OC2R and OCF2 can be used to indicate a period of time has been elapsed but cannot generate an output waveform because the level OLVL2 is dedicated to the one pulse mode.*

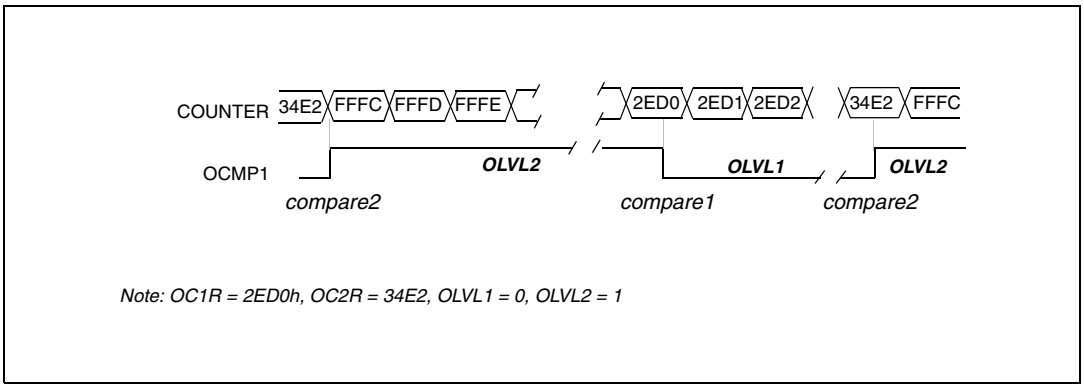
**Figure 54. One pulse mode timing example**




*Note: IEDG1 = 1, OC1R = 2ED0h, OLVL1 = 0, OLVL2 = 1*

Obsolete Product(s) - Obsolete Product(s)

**Figure 55. Pulse width modulation mode timing example with 2 output compare functions**




*Note: On timers with only one Output Compare register, a fixed frequency PWM signal can be generated using the output compare and the counter overflow to define the pulse length.*

### 13.3.7 Pulse width modulation mode

Pulse Width Modulation (PWM) mode enables the generation of a signal with a frequency and pulse length determined by the value of the OC1R and OC2R registers.

Pulse Width Modulation mode uses the complete Output Compare 1 function plus the OC2R register, and so this functionality cannot be used when PWM mode is activated.

In PWM mode, double buffering is implemented on the output compare registers. Any new values written in the OC1R and OC2R registers are taken into account only at the end of the PWM period (OC2) to avoid spikes on the PWM output pin (OCMP1).

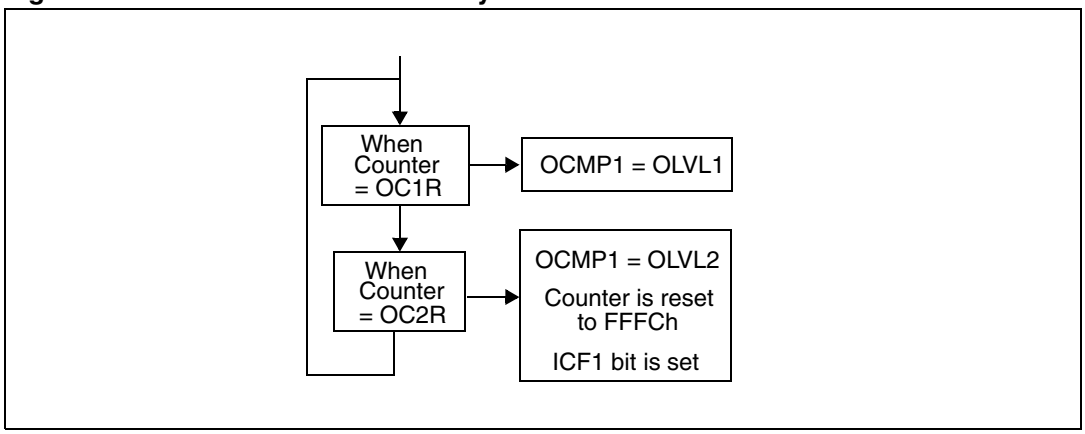
#### Procedure

To use pulse width modulation mode:

1. Load the OC2R register with the value corresponding to the period of the signal using the appropriate formula below according to the timer clock source used.
2. Load the OC1R register with the value corresponding to the period of the pulse if OLVL1 = 0 and OLVL2 = 1 using the appropriate formula below according to the timer clock source used.
3. Select the following in the CR1 register:
   - Using the OLVL1 bit, select the level to be applied to the OCMP1 pin after a successful comparison with the OC1R register.
   - Using the OLVL2 bit, select the level to be applied to the OCMP1 pin after a successful comparison with the OC2R register.
4. Select the following in the CR2 register:
   - Set OC1E bit: the OCMP1 pin is then dedicated to the output compare 1 function.
   - Set the PWM bit.
   - Select the timer clock (CC[1:0]) (see *Table 62: Timer clock selection*).

Obsolete Product(s) Obsolete Product(s)

**Figure 56. Pulse width modulation cycle flowchart**




If OLVL1 = 1 and OLVL2 = 0 the length of the positive pulse is the difference between the OC2R and OC1R registers.

If OLVL1 = OLVL2 a continuous signal will be seen on the OCMP1 pin.

The OC*i*R register value required for a specific timing application can be calculated using the following formula:

$$\text{OC}i\text{R value} = \frac{t * f_{CPU} - 5}{\text{PRESC}}$$

Where:

t = Signal or pulse period (in seconds)

$f_{CPU}$ = CPU clock frequency (in hertz)

PRESC = Timer prescaler factor (2, 4 or 8 depending on CC[1:0] bits; see *Table 62: Timer clock selection*)

If the timer clock is an external clock the formula is:

$$\text{OC}i\text{R} = t * f_{EXT} - 5$$

Where:

t = Signal or pulse period (in seconds)

$f_{EXT}$ = External timer clock frequency (in hertz)

The Output Compare 2 event causes the counter to be initialized to FFFCh (see *Figure 55*).

*Note: 1 After a write instruction to the OCiHR register, the output compare function is inhibited until the OCiLR register is also written.*

*2 The OCF1 and OCF2 bits cannot be set by hardware in PWM mode therefore the Output Compare interrupt is inhibited.*

*3 The ICF1 bit is set by hardware when the counter reaches the OC2R value and can produce a timer interrupt if the ICIE bit is set and the I bit is cleared.*

*4 In PWM mode the ICAP1 pin cannot be used to perform input capture because it is disconnected to the timer. The ICAP2 pin can be used to perform input capture (ICF2 can be set and IC2R can be loaded) but the user must take care that the counter is reset each period and ICF1 can also generates interrupt if ICIE is set.*

*5 When the Pulse Width Modulation (PWM) and One Pulse Mode (OPM) bits are both set, the PWM mode is the only active one.*

Obsolete Product(s) - Obsolete Product(s)

## 13.4 Low power modes

**Table 57. Effect of low power modes on 16-bit timer**

| Mode | Effect |
|---|---|
| Wait | No effect on 16-bit timer.<br>Timer interrupts cause the device to exit from Wait mode. |
| Halt | 16-bit timer registers are frozen.<br>In Halt mode, the counter stops counting until Halt mode is exited. Counting resumes from the previous count when the MCU is woken up by an interrupt with “exit from Halt mode” capability or from the counter reset value when the MCU is woken up by a RESET.<br>If an input capture event occurs on the ICAP*i* pin, the input capture detection circuitry is armed. Consequently, when the MCU is woken up by an interrupt with “exit from Halt mode” capability, the ICF*i* bit is set, and the counter value present when exiting from Halt mode is captured into the IC*i*R register. |

## 13.5 Interrupts

**Table 58. 16-bit timer interrupt control/wake-up capability**

| Interrupt event | Event flag | Enable control bit | Exit from Wait | Exit from Halt |
|---|---|---|---|---|
| Input Capture 1 event/Counter reset in PWM mode | ICF1 | ICIE | Yes | No |
| Input Capture 2 event | ICF2 | | | |
| Output Compare 1 event (not available in PWM mode) | OCF1 | OCIE | | |
| Output Compare 2 event (not available in PWM mode) | OCF2 | | | |
| Timer Overflow event | TOF | TOIE | | |

*Note: The 16-bit timer interrupt events are connected to the same interrupt vector (see Section 7: Interrupts). These events generate an interrupt if the corresponding Enable Control Bit is set and the interrupt mask in the CC register is reset (RIM instruction).*

## 13.6 Summary of timer modes

**Table 59. Timer modes**

| Modes | Timer resources | | | |
|---|---|---|---|---|
| | Input Capture 1 | Input Capture 2 | Output Compare 1 | Output Compare 2 |
| Input Capture (1 and/or 2) | Yes | Yes | Yes | Yes |
| Output Compare (1 and/or 2) | | | | |

Obsolete Product(s) - Obsolete Product(s)

**Table 59. Timer modes**

| Modes | Timer resources | | | |
|---|---|---|---|---|
| | Input Capture 1 | Input Capture 2 | Output Compare 1 | Output Compare 2 |
| One Pulse mode | No | Not recommended(1) | No | Partially(2) |
| PWM mode | | Not recommended(3) | | No |

1. See *Note 4* in *Section 13.3.6 One Pulse mode*
2. See *Note 5* in *Section 13.3.6 One Pulse mode*
3. See *Note 4* in *Section 13.3.7 Pulse width modulation mode*

## 13.7 16-bit timer registers

Each timer is associated with 3 control and status registers, and with 6 pairs of data registers (16-bit values) relating to the 2 input captures, the 2 output compares, the counter and the alternate counter.

### 13.7.1 Control register 1 (CR1)

CR1 Reset value: 0000 0000 (00h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| ICIE | OCIE | TOIE | FOLV2 | FOLV1 | OLVL2 | IEDG1 | OLVL1 |
| RW | RW | RW | RW | RW | RW | RW | RW |

**Table 60. CR1 register description**

| Bit | Name | Function |
|---|---|---|
| 7 | ICIE | *Input Capture Interrupt Enable*<br>0: Interrupt is inhibited<br>1: A timer interrupt is generated whenever the ICF1 or ICF2 bit of the SR register is set. |
| 6 | OCIE | *Output Compare Interrupt Enable*<br>0: Interrupt is inhibited<br>1: A timer interrupt is generated whenever the OCF1 or OCF2 bit of the SR register is set. |
| 5 | TOIE | *Timer Overflow Interrupt Enable*<br>0: Interrupt is inhibited<br>1: A timer interrupt is enabled whenever the TOF bit of the SR register is set. |
| 4 | FOLV2 | *Forced Output Compare 2*<br>This bit is set and cleared by software.<br>0: No effect on the OCMP2 pin<br>1: Forces the OLVL2 bit to be copied to the OCMP2 pin, if the OC2E bit is set and even if there is no successful comparison |

Obsolete Product(s) Obsolete Product(s)

**Table 60. CR1 register description (continued)**

| Bit | Name | Function |
|---|---|---|
| 3 | FOLV1 | *Forced Output Compare 1*<br>This bit is set and cleared by software.<br>0: No effect on the OCMP1 pin<br>1: Forces OLVL1 to be copied to the OCMP1 pin, if the OC1E bit is set and even if there is no successful comparison |
| 2 | OLVL2 | *Output Level 2*<br>This bit is copied to the OCMP2 pin whenever a successful comparison occurs with the OC2R register and OCxE is set in the CR2 register. This value is copied to the OCMP1 pin in One Pulse Mode and Pulse Width Modulation mode. |
| 1 | IEDG1 | *Input Edge 1*<br>This bit determines which type of level transition on the ICAP1 pin will trigger the capture.<br>0: A falling edge triggers the capture.<br>1: A rising edge triggers the capture. |
| 0 | OLVL1 | *Output Level 1*<br>The OLVL1 bit is copied to the OCMP1 pin whenever a successful comparison occurs with the OC1R register and the OC1E bit is set in the CR2 register. |

### 13.7.2 Control register 2 (CR2)

CR2 Reset value: 0000 0000 (00h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| OC1E | OC2E | OPM | PWM | CC[1:0] | | IEDG2 | EXEDG |
| RW | RW | RW | RW | RW | | RW | RW |

**Table 61. CR2 register description**

| Bit | Name | Function |
|---|---|---|
| 7 | OC1E | *Output Compare 1 Pin Enable*<br>This bit is used only to output the signal from the timer on the OCMP1 pin (OLV1 in Output Compare mode, both OLV1 and OLV2 in PWM and one-pulse mode). Whatever the value of the OC1E bit, the Output Compare 1 function of the timer remains active.<br>0: OCMP1 pin alternate function disabled (I/O pin free for general-purpose I/O)<br>1: OCMP1 pin alternate function enabled |
| 6 | OC2E | *Output Compare 2 Pin Enable*<br>This bit is used only to output the signal from the timer on the OCMP2 pin (OLV2 in Output Compare mode). Whatever the value of the OC2E bit, the Output Compare 2 function of the timer remains active.<br>0: OCMP2 pin alternate function disabled (I/O pin free for general-purpose I/O)<br>1: OCMP2 pin alternate function enabled |

Obsolete Product(s) - Obsolete Produc(s)

**Table 61. CR2 register description (continued)**

| Bit | Name | Function |
|---|---|---|
| 5 | OPM | *One Pulse Mode*<br>0: One Pulse Mode is not active.<br>1: One Pulse Mode is active, the ICAP1 pin can be used to trigger one pulse on the OCMP1 pin; the active transition is given by the IEDG1 bit. The length of the generated pulse depends on the contents of the OC1R register. |
| 4 | PWM | *Pulse Width Modulation*<br>0: PWM mode is not active.<br>1: PWM mode is active, the OCMP1 pin outputs a programmable cyclic signal; the length of the pulse depends on the value of OC1R register; the period depends on the value of OC2R register. |
| 3:2 | CC[1:0] | *Clock Control*<br>The timer clock mode depends on these bits (see *Table 62*). |
| 1 | IEDG2 | *Input Edge 2*<br>This bit determines which type of level transition on the ICAP2 pin will trigger the capture.<br>0: A falling edge triggers the capture.<br>1: A rising edge triggers the capture. |
| 0 | EXEDG | *External Clock Edge*<br>This bit determines which type of level transition on the external clock pin EXTCLK will trigger the counter register.<br>0: A falling edge triggers the counter register.<br>1: A rising edge triggers the counter register. |

**Table 62. Timer clock selection**

| Timer clock | CC1 | CC0 |
|---|---|---|
| $f_{CPU}$ / 4 | 0 | 0 |
| $f_{CPU}$ / 2 | 0 | 1 |
| $f_{CPU}$ / 8 | 1 | 0 |
| External clock (where available)(1) | 1 | 1 |

1. If the external clock pin is not available, programming the external clock configuration stops the counter.

### 13.7.3 Control/status register (CSR)

CSR Reset value: xxxx x0xx (xxh)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| ICF1 | OCF1 | TOF | ICF2 | OCF2 | TIMD | Reserved | |
| RO | RO | RO | RO | RO | RW | - | |

Obsolete Product(s) - Obsolete Product(s)

**Table 63. CSR register description**

| Bit | Name | Function |
|---|---|---|
| 7 | ICF1 | *Input Capture Flag 1*<br>0: No input capture (reset value)<br>1: An input capture has occurred on the ICAP1 pin or the counter has reached the OC2R value in PWM mode. To clear this bit, first read the SR register, then read or write the low byte of the IC1R (IC1LR) register. |
| 6 | OCF1 | *Output Compare Flag 1*<br>0: No match (reset value)<br>1: The content of the free running counter has matched the content of the OC1R register. To clear this bit, first read the SR register, then read or write the low byte of the OC1R (OC1LR) register. |
| 5 | TOF | *Timer Overflow Flag*<br>0: No timer overflow (reset value)<br>1: The free running counter rolled over from FFFFh to 0000h. To clear this bit, first read the SR register, then read or write the low byte of the CR (CLR) register.<br>*Note: Reading or writing the ACLR register does not clear TOF.* |
| 4 | ICF2 | *Input Capture Flag 2*<br>0: No input capture (reset value).<br>1: An input capture has occurred on the ICAP2 pin. To clear this bit, first read the SR register, then read or write the low byte of the IC2R (IC2LR) register. |
| 3 | OCF2 | *Output Compare Flag 2*<br>0: No match (reset value)<br>1: The content of the free running counter has matched the content of the OC2R register. To clear this bit, first read the SR register, then read or write the low byte of the OC2R (OC2LR) register. |
| 2 | TIMD | *Timer disable*<br>This bit is set and cleared by software. When set, it freezes the timer prescaler and counter and disabled the output functions (OCMP1 and OCMP2 pins) to reduce power consumption. Access to the timer registers is still available, allowing the timer configuration to be changed, or the counter reset, while it is disabled.<br>0: Timer enabled<br>1: Timer prescaler, counter and outputs disabled |
| 1:0 | - | Reserved, must be kept cleared |

### 13.7.4 Input capture 1 high register (IC1HR)

This is an 8-bit read only register that contains the high part of the counter value (transferred by the input capture 1 event).

IC1HR — Reset value: Undefined

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| MSB | | | | | | | LSB |
| RO | RO | RO | RO | RO | RO | RO | RO |

Obsolete Product(s) - Obsolete Product(s)

### 13.7.5 Input capture 1 low register (IC1LR)

This is an 8-bit read only register that contains the low part of the counter value (transferred by the input capture 1 event).

IC1LR — Reset value: Undefined

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| MSB | | | | | | | LSB |
| RO | RO | RO | RO | RO | RO | RO | RO |

### 13.7.6 Output compare 1 high register (OC1HR)

This is an 8-bit register that contains the high part of the value to be compared to the CHR register.

OC1HR — Reset value: 1000 0000 (80h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| MSB | | | | | | | LSB |
| RW | RW | RW | RW | RW | RW | RW | RW |

### 13.7.7 Output compare 1 low register (OC1LR)

This is an 8-bit register that contains the low part of the value to be compared to the CLR register.

OC1LR — Reset value: 0000 0000 (00h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| MSB | | | | | | | LSB |
| RW | RW | RW | RW | RW | RW | RW | RW |

### 13.7.8 Output compare 2 high register (OC2HR)

This is an 8-bit register that contains the high part of the value to be compared to the CHR register.

OC2HR — Reset value: 1000 0000 (80h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| MSB | | | | | | | LSB |
| RW | RW | RW | RW | RW | RW | RW | RW |

Obsolete Product(s) - Obsolete Product(s)

### 13.7.9 Output compare 2 low register (OC2LR)

This is an 8-bit register that contains the low part of the value to be compared to the CLR register.

OC2LR Reset value: 0000 0000 (00h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| MSB | | | | | | | LSB |
| RW | RW | RW | RW | RW | RW | RW | RW |

### 13.7.10 Counter high register (CHR)

This is an 8-bit register that contains the high part of the counter value.

CHR Reset value: 1111 1111 (FFh)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| MSB | | | | | | | LSB |
| RO | RO | RO | RO | RO | RO | RO | RO |

### 13.7.11 Counter low register (CLR)

This is an 8-bit register that contains the low part of the counter value. A write to this register resets the counter. An access to this register after accessing the CSR register clears the TOF bit.

CLR Reset value: 1111 1100 (FCh)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| MSB | | | | | | | LSB |
| RO | RO | RO | RO | RO | RO | RO | RO |

### 13.7.12 Alternate counter high register (ACHR)

This is an 8-bit register that contains the high part of the counter value.

ACHR Reset value: 1111 1111 (FFh)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| MSB | | | | | | | LSB |
| RO | RO | RO | RO | RO | RO | RO | RO |

Obsolete Product(s) - Obsolete Product(s)

### 13.7.13 Alternate counter low register (ACLR)

This is an 8-bit register that contains the low part of the counter value. A write to this register resets the counter. An access to this register after an access to CSR register does not clear the TOF bit in the CSR register.

ACLR Reset value: 1111 1100 (FCh)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| MSB | | | | | | | LSB |
| RO | RO | RO | RO | RO | RO | RO | RO |

### 13.7.14 Input capture 2 high register (IC2HR)

This is an 8-bit read only register that contains the high part of the counter value (transferred by the Input Capture 2 event).

IC2HR Reset value: Undefined

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| MSB | | | | | | | LSB |
| RO | RO | RO | RO | RO | RO | RO | RO |

### 13.7.15 Input capture 2 low register (IC2LR)

This is an 8-bit read only register that contains the low part of the counter value (transferred by the Input Capture 2 event).

IC2LR Reset value: Undefined

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| MSB | | | | | | | LSB |
| RO | RO | RO | RO | RO | RO | RO | RO |

Obsolete Product(s) - Obsolete Product(s)

**Table 64. 16-bit timer register map and reset values**

| Address (Hex.) | Register label | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Timer A: 32<br>Timer B: 42 | CR1<br>Reset value | ICIE<br>0 | OCIE<br>0 | TOIE<br>0 | FOLV2<br>0 | FOLV1<br>0 | OLVL2<br>0 | IEDG1<br>0 | OLVL1<br>0 |
| Timer A: 31<br>Timer B: 41 | CR2<br>Reset value | OC1E<br>0 | OC2E<br>0 | OPM<br>0 | PWM<br>0 | CC1<br>0 | CC0<br>0 | IEDG2<br>0 | EXEDG<br>0 |
| Timer A: 33<br>Timer B: 43 | CSR<br>Reset value | ICF1<br>x | OCF1<br>x | TOF<br>x | ICF2<br>x | OCF2<br>x | TIMD<br>0 | -<br>x | -<br>x |
| Timer A: 34<br>Timer B: 44 | IC1HR<br>Reset value | MSB<br>x | x | x | x | x | x | x | LSB<br>x |
| Timer A: 35<br>Timer B: 45 | IC1LR<br>Reset value | MSB<br>x | x | x | x | x | x | x | LSB<br>x |
| Timer A: 36<br>Timer B: 46 | OC1HR<br>Reset value | MSB<br>1 | 0 | 0 | 0 | 0 | 0 | 0 | LSB<br>0 |
| Timer A: 37<br>Timer B: 47 | OC1LR<br>Reset value | MSB<br>0 | 0 | 0 | 0 | 0 | 0 | 0 | LSB<br>0 |
| Timer A: 3E<br>Timer B: 4E | OC2HR<br>Reset value | MSB<br>1 | 0 | 0 | 0 | 0 | 0 | 0 | LSB<br>0 |
| Timer A: 3F<br>Timer B: 4F | OC2LR<br>Reset value | MSB<br>0 | 0 | 0 | 0 | 0 | 0 | 0 | LSB<br>0 |
| Timer A: 38<br>Timer B: 48 | CHR<br>Reset value | MSB<br>1 | 1 | 1 | 1 | 1 | 1 | 1 | LSB<br>1 |
| Timer A: 39<br>Timer B: 49 | CLR<br>Reset value | MSB<br>1 | 1 | 1 | 1 | 1 | 1 | 0 | LSB<br>0 |
| Timer A: 3A<br>Timer B: 4A | ACHR<br>Reset value | MSB<br>1 | 1 | 1 | 1 | 1 | 1 | 1 | LSB<br>1 |
| Timer A: 3B<br>Timer B: 4B | ACLR<br>Reset value | MSB<br>1 | 1 | 1 | 1 | 1 | 1 | 0 | LSB<br>0 |
| Timer A: 3C<br>Timer B: 4C | IC2HR<br>Reset value | MSB<br>x | x | x | x | x | x | x | LSB<br>x |
| Timer A: 3D<br>Timer B: 4D | IC2LR<br>Reset value | MSB<br>x | x | x | x | x | x | x | LSB<br>x |

## Related documentation

*SCI software communications using 16-bit timer* (AN 973)

*Real-time Clock with ST7 Timer Output Compare* (AN 974)

*Driving a buzzer through the ST7 Timer PWM function* (AN 976)

*Using ST7 PWM signal to generate analog input (sinusoid)* (AN1041)

*UART emulation software* (AN1046)

*PWM duty cycle switch implementing true 0 or 100 per cent duty cycle* (AN1078)

*Starting a PWM signal directly at high level using the ST7 16-bit timer* (AN1504)

Obsolete Product(s) - Obsolete Product(s)

# 14 Serial peripheral interface (SPI)

## 14.1 Introduction

The Serial Peripheral Interface (SPI) allows full-duplex, synchronous, serial communication with external devices. An SPI system may consist of a master and one or more slaves however the SPI interface cannot be a master in a multimaster system.

## 14.2 Main features

- Full duplex synchronous transfers (on 3 lines)
- Simplex synchronous transfers (on 2 lines)
- Master or slave operation
- 6 master mode frequencies ($f_{CPU}$/4 max.)
- $f_{CPU}$/2 max. slave mode frequency (see note)
- $\overline{SS}$ Management by software or hardware
- Programmable clock polarity and phase
- End of transfer interrupt flag
- Write collision, Master Mode Fault and Overrun flags

*Note: In slave mode, continuous transmission is not possible at maximum frequency due to the software overhead for clearing status flags and to initiate the next transmission sequence.*

## 14.3 General description

*Figure 57* shows the serial peripheral interface (SPI) block diagram. There are three registers:

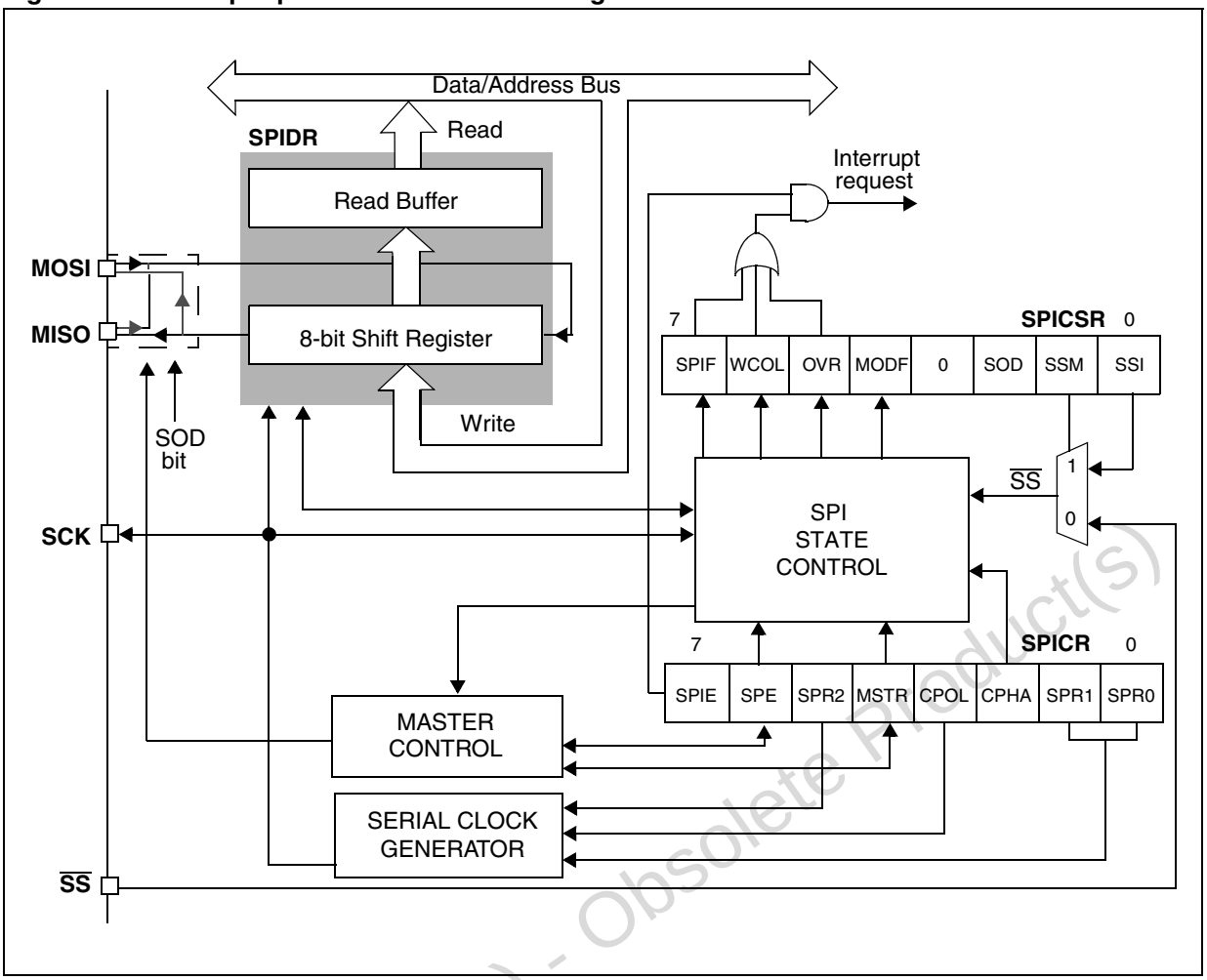
- SPI Control Register (SPICR)
- SPI Control/Status Register (SPICSR)
- SPI Data Register (SPIDR)

The SPI is connected to external devices through four pins:

- MISO (Master In / Slave Out data)
- MOSI (Master Out / Slave In data)
- SCK (Serial Clock out by SPI masters and input by SPI slaves)
- $\overline{SS}$ (Slave select): This input signal acts as a 'chip select' to let the SPI master communicate with slaves individually and to avoid contention on the data lines. Slave $\overline{SS}$ inputs can be driven by standard I/O ports on the master MCU.

Obsolete Product(s) - Obsolete Product(s)

**Figure 57. Serial peripheral interface block diagram**




## 14.3.1 Functional description

A basic example of interconnections between a single master and a single slave is illustrated in *Figure 58*.

The MOSI pins are connected together and the MISO pins are connected together. In this way data is transferred serially between master and slave (most significant bit first).

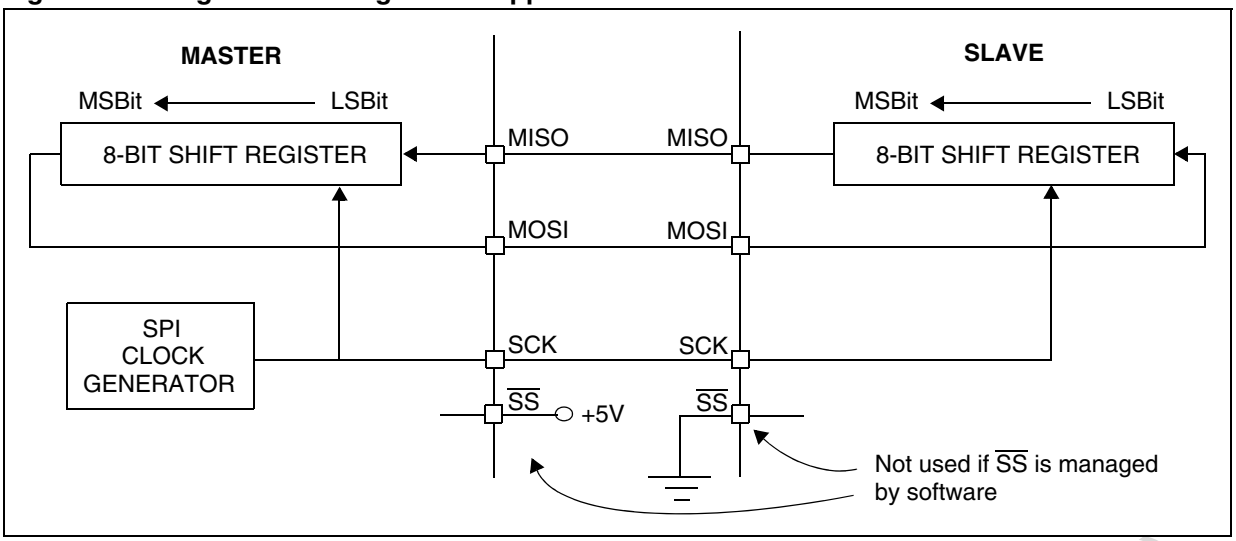
The communication is always initiated by the master. When the master device transmits data to a slave device via MOSI pin, the slave device responds by sending data to the master device via the MISO pin. This implies full duplex communication with both data out and data in synchronized with the same clock signal (which is provided by the master device via the SCK pin).

To use a single data line, the MISO and MOSI pins must be connected at each node (in this case only simplex communication is possible).

Four possible data/clock timing relationships may be chosen (see *Figure 61*) but master and slave must be programmed with the same timing mode.

Obsolete Product(s) - Obsolete Product(s)

**Figure 58. Single master/single slave application**




## 14.3.2 Slave select management

As an alternative to using the $\overline{SS}$ pin to control the Slave Select signal, the application can choose to manage the Slave Select signal by software. This is configured by the SSM bit in the SPICSR register (see *Figure 60*)

In software management, the external $\overline{SS}$ pin is free for other application uses and the internal $\overline{SS}$ signal level is driven by writing to the SSI bit in the SPICSR register.

**In Master mode**

- $\overline{SS}$ internal must be held high continuously

**In Slave mode**

There are two cases depending on the data/clock timing relationship (see *Figure 59*):

If CPHA = 1 (data latched on 2nd clock edge):

- $\overline{SS}$ internal must be held low during the entire transmission. This implies that in single slave applications the $\overline{SS}$ pin either can be tied to $V_{SS}$, or made free for standard I/O by managing the $\overline{SS}$ function by software (SSM = 1 and SSI = 0 in the in the SPICSR register)

If CPHA = 0 (data latched on 1st clock edge):

- $\overline{SS}$ internal must be held low during byte transmission and pulled high between each byte to allow the slave to write to the shift register. If $\overline{SS}$ is not pulled high, a Write Collision error will occur when the slave writes to the shift register (see *Write collision error (WCOL) on page 132*).

Obsolete Product(s) - Obsolete Product(s)

**Figure 59. Generic $\overline{SS}$ timing diagram**

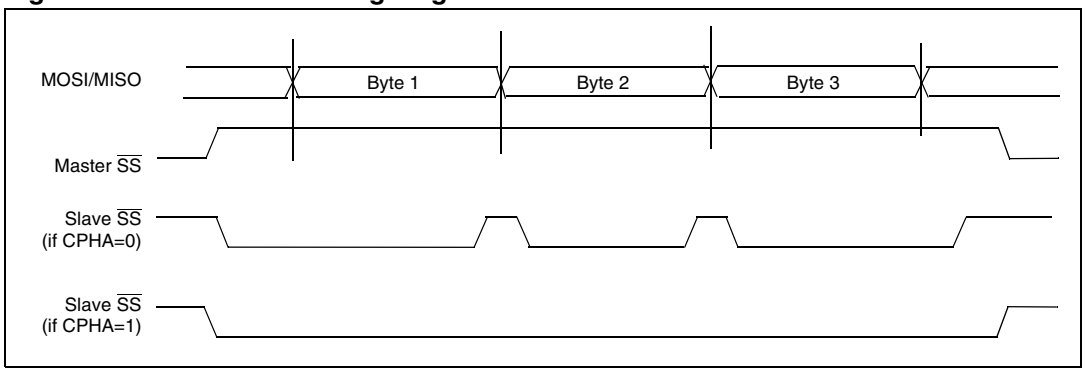
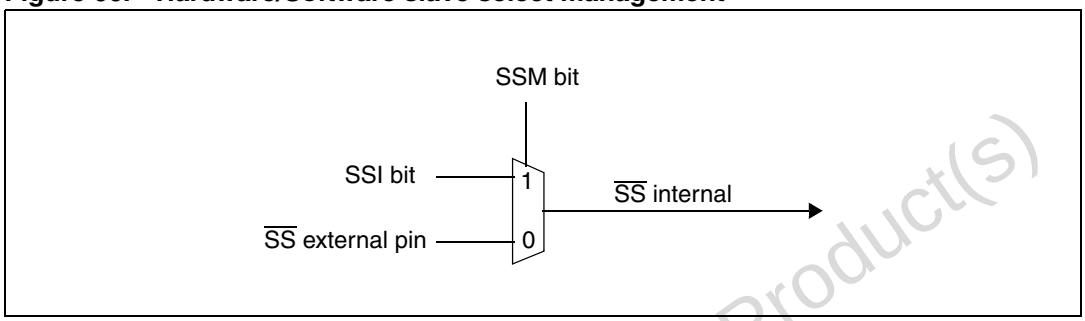



**Figure 60. Hardware/Software slave select management**




Obsolete Product(s) - Obsolete Product(s)

### 14.3.3 Master mode operation

In master mode, the serial clock is output on the SCK pin. The clock frequency, polarity and phase are configured by software (refer to the description of the SPICSR register).

*Note: The idle state of SCK must correspond to the polarity selected in the SPICSR register (by pulling up SCK if CPOL = 1 or pulling down SCK if CPOL = 0).*

**How to operate the SPI in master mode**

To operate the SPI in master mode, perform the following steps in order:

1. Write to the SPICR register:
   a) Select the clock frequency by configuring the SPR[2:0] bits.
   b) Select the clock polarity and clock phase by configuring the CPOL and CPHA bits. *Figure 61* shows the four possible configurations.

*Note: The slave must have the same CPOL and CPHA settings as the master.*

2. Write to the SPICSR register:
   Either set the SSM bit and set the SSI bit or clear the SSM bit and tie the $\overline{SS}$ pin high for the complete byte transmit sequence.
3. Write to the SPICR register:
   Set the MSTR and SPE bits

*Note: MSTR and SPE bits remain set only if $\overline{SS}$ is high).*

**IMPORTANT**: If the SPICSR register is not written first, the SPICR register setting (MSTR bit) may not be taken into account.

The transmit sequence begins when software writes a byte in the SPIDR register.

### 14.3.4 Master mode transmit sequence

When software writes to the SPIDR register, the data byte is loaded into the 8-bit shift register and then shifted out serially to the MOSI pin most significant bit first.

When data transfer is complete:

- The SPIF bit is set by hardware
- An interrupt request is generated if the SPIE bit is set and the interrupt mask in the CCR register is cleared.

Clearing the SPIF bit is performed by the following software sequence:

1. An access to the SPICSR register while the SPIF bit is set
2. A read to the SPIDR register.

*Note: While the SPIF bit is set, all writes to the SPIDR register are inhibited until the SPICSR register is read.*

### 14.3.5 Slave mode operation

In slave mode, the serial clock is received on the SCK pin from the master device.

To operate the SPI in slave mode:

1. Write to the SPICSR register to perform the following actions:
   a) Select the clock polarity and clock phase by configuring the CPOL and CPHA bits (see *Figure 61*).

*Note: The slave must have the same CPOL and CPHA settings as the master.*

   b) Manage the $\overline{SS}$ pin as described in *Slave select management on page 127* and *Figure 59*. If CPHA = 1, $\overline{SS}$ must be held low continuously. If CPHA = 0, $\overline{SS}$ must be held low during byte transmission and pulled up between each byte to let the slave write in the shift register.
2. Write to the SPICR register to clear the MSTR bit and set the SPE bit to enable the SPI I/O functions.

### 14.3.6 Slave mode transmit sequence

When software writes to the SPIDR register, the data byte is loaded into the 8-bit shift register and then shifted out serially to the MISO pin most significant bit first.

The transmit sequence begins when the slave device receives the clock signal and the most significant bit of the data on its MOSI pin.

When data transfer is complete:

- The SPIF bit is set by hardware.
- An interrupt request is generated if SPIE bit is set and interrupt mask in the CCR register is cleared.

Clearing the SPIF bit is performed by the following software sequence:

1. An access to the SPICSR register while the SPIF bit is set
2. A write or a read to the SPIDR register

*Note: While the SPIF bit is set, all writes to the SPIDR register are inhibited until the SPICSR register is read.*

Obsolete Product(s) - Obsolete Product(s)

The SPIF bit can be cleared during a second transmission; however, it must be cleared before the second SPIF bit in order to prevent an Overrun condition (see *Overrun condition (OVR) on page 132*).

## 14.4 Clock phase and clock polarity

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits (see *Figure 61*).

*Note: The idle state of SCK must correspond to the polarity selected in the SPICSR register (by pulling up SCK if CPOL = 1 or pulling down SCK if CPOL = 0).*

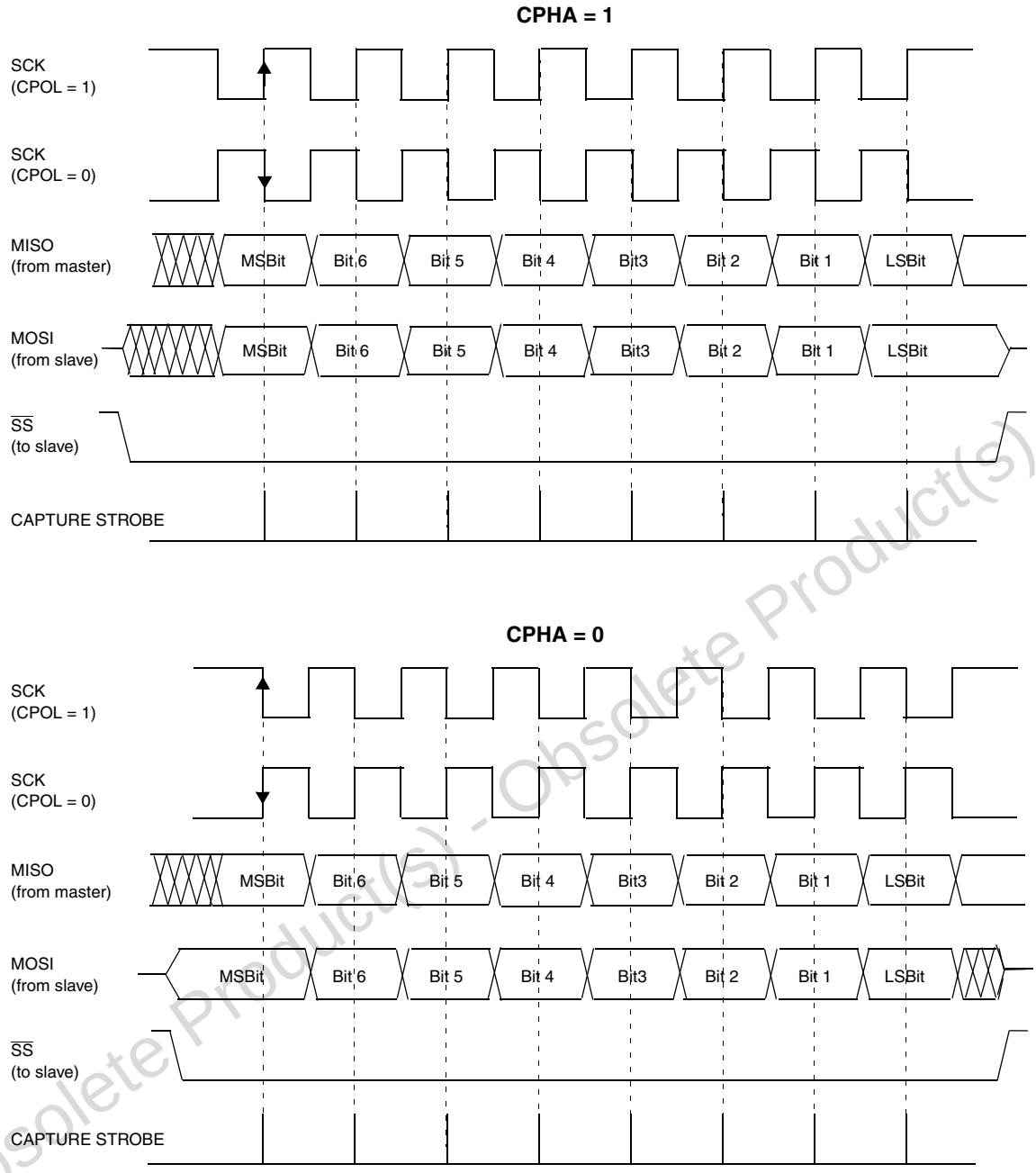
The combination of the CPOL clock polarity and CPHA (clock phase) bits selects the data capture clock edge

*Figure 61* shows an SPI transfer with the four combinations of the CPHA and CPOL bits. The diagram may be interpreted as a master or slave timing diagram where the SCK pin, the MISO pin, the MOSI pin are directly connected between the master and the slave device.

*Note: If CPOL is changed at the communication byte boundaries, the SPI must be disabled by resetting the SPE bit.*

Obsolete Product(s) - Obsolete Product(s)

**Figure 61. Data clock timing diagram**




*Note: This figure should not be used as a replacement for parametric information.*
*Refer to Section 19: Electrical characteristics.*

Obsolete Product(s) - Obsolete Product(s)

## 14.5 Error flags

### 14.5.1 Master mode fault (MODF)

Master mode fault occurs when the master device has its $\overline{SS}$ pin pulled low.

When a Master mode fault occurs:

- The MODF bit is set and an SPI interrupt request is generated if the SPIE bit is set.
- The SPE bit is reset. This blocks all output from the device and disables the SPI peripheral.
- The MSTR bit is reset, thus forcing the device into slave mode.

Clearing the MODF bit is done through a software sequence:

1. A read access to the SPICSR register while the MODF bit is set.
2. A write to the SPICR register.

*Note: To avoid any conflicts in an application with multiple slaves, the $\overline{SS}$ pin must be pulled high during the MODF bit clearing sequence. The SPE and MSTR bits may be restored to their original state during or after this clearing sequence.*

Hardware does not allow the user to set the SPE and MSTR bits while the MODF bit is set except in the MODF bit clearing sequence.

### 14.5.2 Overrun condition (OVR)

An overrun condition occurs, when the master device has sent a data byte and the slave device has not cleared the SPIF bit issued from the previously transmitted byte.

When an Overrun occurs:

- The OVR bit is set and an interrupt request is generated if the SPIE bit is set.

In this case, the receiver buffer contains the byte sent after the SPIF bit was last cleared. A read to the SPIDR register returns this byte. All other bytes are lost.

The OVR bit is cleared by reading the SPICSR register.

### 14.5.3 Write collision error (WCOL)

A write collision occurs when the software tries to write to the SPIDR register while a data transfer is taking place with an external device. When this happens, the transfer continues uninterrupted; and the software write will be unsuccessful.

Write collisions can occur both in master and slave mode. See also *Slave select management on page 127*.

*Note: A "read collision" will never occur since the received data byte is placed in a buffer in which access is always synchronous with the MCU operation.*

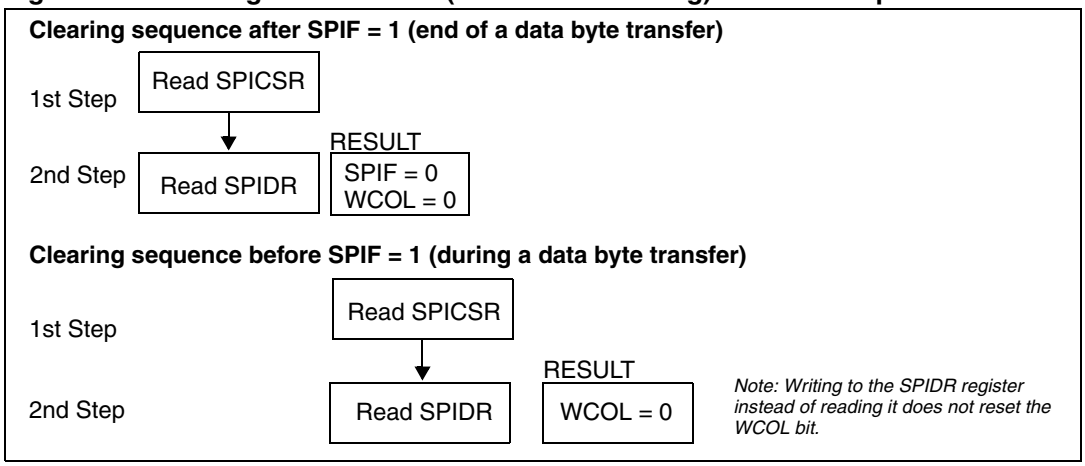
The WCOL bit in the SPICSR register is set if a write collision occurs.

No SPI interrupt is generated when the WCOL bit is set (the WCOL bit is a status flag only).

Clearing the WCOL bit is done through a software sequence (see *Figure 62*).

Obsolete Product(s) - Obsolete Product(s)

**Figure 62. Clearing the WCOL bit (Write Collision Flag) software sequence**




## 14.5.4 Single master systems

A typical single master system may be configured, using an MCU as the master and four MCUs as slaves (see *Figure 63*).

The master device selects the individual slave devices by using four pins of a parallel port to control the four $\overline{SS}$ pins of the slave devices.

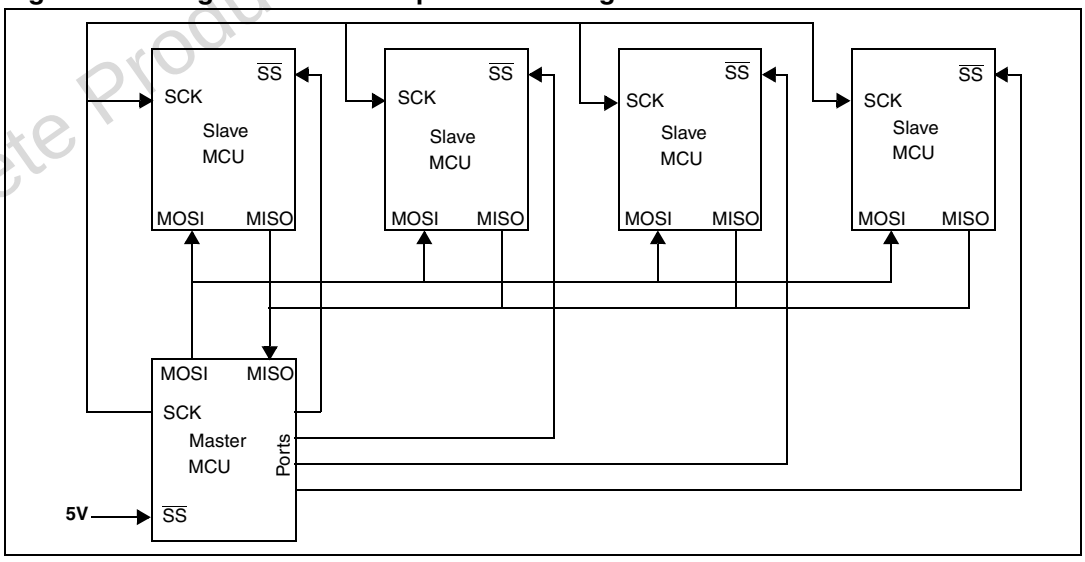
The $\overline{SS}$ pins are pulled high during reset since the master device ports will be forced to be inputs at that time, thus disabling the slave devices.

*Note: To prevent a bus conflict on the MISO line the master allows only one active slave device during a transmission.*

For more security, the slave device may respond to the master with the received data byte. Then the master will receive the previous byte back from the slave device if all MISO and MOSI pins are connected and the slave has not written to its SPIDR register.

Other transmission security methods can use ports for handshake lines or data bytes with command fields.

**Figure 63. Single master / multiple slave configuration**




Obsolete Product(s) - Obsolete Product(s)

## 14.6 Low power modes

**Table 65. Effect of low power modes on SPI**

| Mode | Effect |
|---|---|
| Wait | No effect on SPI.<br>SPI interrupt events cause the device to exit from Wait mode. |
| Halt | SPI registers are frozen.<br>In Halt mode, the SPI is inactive. SPI operation resumes when the MCU is woken up by an interrupt with “exit from Halt mode” capability. The data received is subsequently read from the SPIDR register when the software is running (interrupt vector fetching). If several data are received before the wake-up event, then an overrun error is generated. This error can be detected after the fetch of the interrupt routine that woke up the device. |

### 14.6.1 Using the SPI to wake up the MCU from Halt mode

In slave configuration, the SPI is able to wake up the ST7 device from Halt mode through a SPIF interrupt. The data received is subsequently read from the SPIDR register when the software is running (interrupt vector fetch). If multiple data transfers have been performed before software clears the SPIF bit, then the OVR bit is set by hardware.

*Note: When waking up from Halt mode, if the SPI remains in Slave mode, it is recommended to perform an extra communications cycle to bring the SPI from Halt mode state to normal state. If the SPI exits from Slave mode, it returns to normal state immediately.*

**Caution:** The SPI can wake up the ST7 from Halt mode only if the Slave Select signal (external $\overline{SS}$ pin or the SSI bit in the SPICSR register) is low when the ST7 enters Halt mode. So if Slave selection is configured as external (see *Slave select management on page 127*), make sure the master drives a low level on the $\overline{SS}$ pin when the slave enters Halt mode.

## 14.7 Interrupts

**Table 66. SPI interrupt control/wake-up capability**

| Interrupt event | Event flag | Enable control bit | Exit from Wait | Exit from Halt |
|---|---|---|---|---|
| SPI End of Transfer event | SPIF | SPIE | Yes | Yes |
| Master Mode Fault event | MODF | | | No |
| Overrun error | OVR | | | |

*Note: The SPI interrupt events are connected to the same interrupt vector (see Interrupts chapter). They generate an interrupt if the corresponding Enable Control Bit is set and the interrupt mask in the CC register is reset (RIM instruction).*

Obsolete Product(s) - Obsolete Product(s)

## 14.8 SPI registers

### 14.8.1 Control register (SPICR)

SPICR Reset value: 0000 xxxx (0xh)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SPIE | SPE | SPR2 | MSTR | CPOL | CPHA | SPR[1:0] | |
| RW | RW | RW | RW | RW | RW | RW | |

**Table 67. SPICR register description**

| Bit | Name | Function |
|---|---|---|
| 7 | SPIE | *Serial Peripheral Interrupt Enable*<br>This bit is set and cleared by software.<br>0: Interrupt is inhibited<br>1: An SPI interrupt is generated whenever SPIF = 1, MODF = 1 or OVR = 1 in the SPICSR register. |
| 6 | SPE | *Serial Peripheral Output Enable*<br>This bit is set and cleared by software. It is also cleared by hardware when, in master mode, $\overline{SS}$ = 0 (see *Master mode fault (MODF) on page 132*). The SPE bit is cleared by reset, so the SPI peripheral is not initially connected to the external pins.<br>0: I/O pins free for general purpose I/O<br>1: SPI I/O pin alternate functions enabled |
| 5 | SPR2 | *Divider Enable*<br>This bit is set and cleared by software and is cleared by reset. It is used with the SPR[1:0] bits to set the baud rate. Refer to *Table 68*.<br>0: Divider by 2 enabled<br>1: Divider by 2 disabled<br>*Note: This bit has no effect in slave mode.* |
| 4 | MSTR | *Master Mode*<br>This bit is set and cleared by software. It is also cleared by hardware when, in master mode, $\overline{SS}$ = 0 (see *Master mode fault (MODF) on page 132*).<br>0: Slave mode<br>1: Master mode. The function of the SCK pin changes from an input to an output and the functions of the MISO and MOSI pins are reversed. |
| 3 | CPOL | *Clock Polarity*<br>This bit is set and cleared by software. This bit determines the idle state of the serial Clock. The CPOL bit affects both the master and slave modes.<br>0: SCK pin has a low level idle state<br>1: SCK pin has a high level idle state<br>*Note: If CPOL is changed at the communication byte boundaries, the SPI must be disabled by resetting the SPE bit.* |
| 2 | CPHA | *Clock Phase*<br>This bit is set and cleared by software.<br>0: The first clock transition is the first data capture edge.<br>1: The second clock transition is the first capture edge.<br>*Note: The slave must have the same CPOL and CPHA settings as the master.* |

Obsolete Product(s) Obsolete Product(s)

**Table 67. SPICR register description (continued)**

| Bit | Name | Function |
|---|---|---|
| 1:0 | SPR[1:0] | *Serial Clock Frequency*<br>These bits are set and cleared by software. Used with the SPR2 bit, they select the baud rate of the SPI serial clock SCK output by the SPI in master mode.<br>*Note: These 2 bits have no effect in slave mode.* |

**Table 68. SPI master mode SCK frequency**

| Serial clock | SPR2 | SPR1 | SPR0 |
|---|---|---|---|
| $f_{CPU}/4$ | 1 | 0 | 0 |
| $f_{CPU}/8$ | 0 | 0 | 0 |
| $f_{CPU}/16$ | 0 | 0 | 1 |
| $f_{CPU}/32$ | 1 | 1 | 0 |
| $f_{CPU}/64$ | 0 | 1 | 0 |
| $f_{CPU}/128$ | 0 | 1 | 1 |

## 14.8.2 Control/status register (SPICSR)

SPICSR Reset value: 0000 0000 (00h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SPIF | WCOL | OVR | MODF | Reserved | SOD | SSM | SSI |
| RO | RO | RO | RO | - | RW | RW | RW |

**Table 69. SPICSR register description**

| Bit | Name | Function |
|---|---|---|
| 7 | SPIF | *Serial Peripheral Data Transfer Flag*<br>This bit is set by hardware when a transfer has been completed. An interrupt is generated if SPIE = 1 in the SPICR register. It is cleared by a software sequence (an access to the SPICSR register followed by a write or a read to the SPIDR register).<br>0: Data transfer is in progress or the flag has been cleared<br>1: Data transfer between the device and an external device has been completed. While the SPIF bit is set, all writes to the SPIDR register are inhibited until the SPICSR register is read. |
| 6 | WCOL | *Write Collision status*<br>This bit is set by hardware when a write to the SPIDR register is done during a transmit sequence. It is cleared by a software sequence (see *Figure 62*).<br>0: No write collision occurred.<br>1: A write collision has been detected. |
| 5 | OVR | *SPI Overrun error*<br>This bit is set by hardware when the byte currently being received in the shift register is ready to be transferred into the SPIDR register while SPIF = 1 (see *Overrun condition (OVR) on page 132*). An interrupt is generated if SPIE = 1 in SPICR register. The OVR bit is cleared by software reading the SPICSR register.<br>0: No overrun error<br>1: Overrun error detected |

Obsolete Product(s) - Obsolete Product(s)

**Table 69. SPICSR register description (continued)**

| Bit | Name | Function |
|---|---|---|
| 4 | MODF | *Mode Fault flag*<br>This bit is set by hardware when the $\overline{SS}$ pin is pulled low in master mode (see *Master mode fault (MODF) on page 132*). An SPI interrupt can be generated if SPIE = 1 in the SPICSR register. This bit is cleared by a software sequence (An access to the SPICR register while MODF = 1 followed by a write to the SPICR register).<br>0: No master mode fault detected<br>1: A fault in master mode has been detected |
| 3 | - | Reserved, must be kept cleared |
| 2 | SOD | *SPI Output Disable*<br>This bit is set and cleared by software. When set, it disables the alternate function of the SPI output (MOSI in master mode / MISO in slave mode).<br>0: SPI output enabled (if SPE = 1)<br>1: SPI output disabled |
| 1 | SSM | *$\overline{SS}$ Management*<br>This bit is set and cleared by software. When set, it disables the alternate function of the SPI $\overline{SS}$ pin and uses the SSI bit value instead. See *Slave select management on page 127*.<br>0: Hardware management ($\overline{SS}$ managed by external pin)<br>1: Software management (internal $\overline{SS}$ signal controlled by SSI bit. External $\overline{SS}$ pin free for general-purpose I/O) |
| 0 | SSI | *$\overline{SS}$ Internal Mode*<br>This bit is set and cleared by software. It acts as a 'chip select' by controlling the level of the $\overline{SS}$ slave select signal when the SSM bit is set.<br>0: Slave selected<br>1: Slave deselected |

### 14.8.3 Data I/O register (SPIDR)

SPIDR Reset value: Undefined

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| D[7:0] | | | | | | | |
| RW | | | | | | | |

The SPIDR register is used to transmit and receive data on the serial bus. In a master device, a write to this register will initiate transmission/reception of another byte.

*Note: During the last clock cycle the SPIF bit is set, a copy of the received data byte in the shift register is moved to a buffer. When the user reads the serial peripheral data I/O register, the buffer is actually being read.*

While the SPIF bit is set, all writes to the SPIDR register are inhibited until the SPICSR register is read.

Obsolete Product(s) - Obsolete Product(s)

**Warning: A write to the SPIDR register places data directly into the shift register for transmission.**

A read to the SPIDR register returns the value located in the buffer and not the content of the shift register (see *Figure 57*).

**Table 70. SPI register map and reset values**

| Address (Hex.) | Register label | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0021h | SPIDR<br>Reset value | MSB<br>x | x | x | x | x | x | x | LSB<br>x |
| 0022h | SPICR<br>Reset value | SPIE<br>0 | SPE<br>0 | SPR2<br>0 | MSTR<br>0 | CPOL<br>x | CPHA<br>x | SPR1<br>x | SPR0<br>x |
| 0023h | SPICSR<br>Reset value | SPIF<br>0 | WCOL<br>0 | OVR<br>0 | MODF<br>0 | 0 | SOD<br>0 | SSM<br>0 | SSI<br>0 |

Obsolete Product(s) - Obsolete Product(s)

# 15 Serial communications interface (SCI)

## 15.1 Introduction

The Serial Communications Interface (SCI) offers a flexible means of full-duplex data exchange with external equipment requiring an industry standard NRZ asynchronous serial data format. The SCI offers a very wide range of baud rates using two baud rate generator systems.

## 15.2 Main features

- Full duplex, asynchronous communications
- NRZ standard format (Mark/Space)
- Dual baud rate generator systems
- Independently programmable transmit and receive baud rates up to 500K baud
- Programmable data word length (8 or 9 bits)
- Receive buffer full, Transmit buffer empty and End of Transmission flags
- 2 receiver wake-up modes:
  - Address bit (MSB)
  - Idle line
- Muting function for multiprocessor configurations
- Separate enable bits for Transmitter and Receiver
- 4 error detection flags:
  - Overrun error
  - Noise error
  - Frame error
  - Parity error
- 5 interrupt sources with flags:
  - Transmit data register empty
  - Transmission complete
  - Receive data register full
  - Idle line received
  - Overrun error detected
- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte
- Reduced power consumption mode

Obsolete Product(s) - Obsolete Product(s)

## 15.3 General description

The interface is externally connected to another device by two pins (see *Figure 65*):

- TDO: Transmit Data Output. When the transmitter and the receiver are disabled, the output pin returns to its I/O port configuration. When the transmitter and/or the receiver are enabled and nothing is to be transmitted, the TDO pin is at high level.
- RDI: Receive Data Input is the serial data input. Oversampling techniques are used for data recovery by discriminating between valid incoming data and noise.

Through these pins, serial data is transmitted and received as frames comprising:

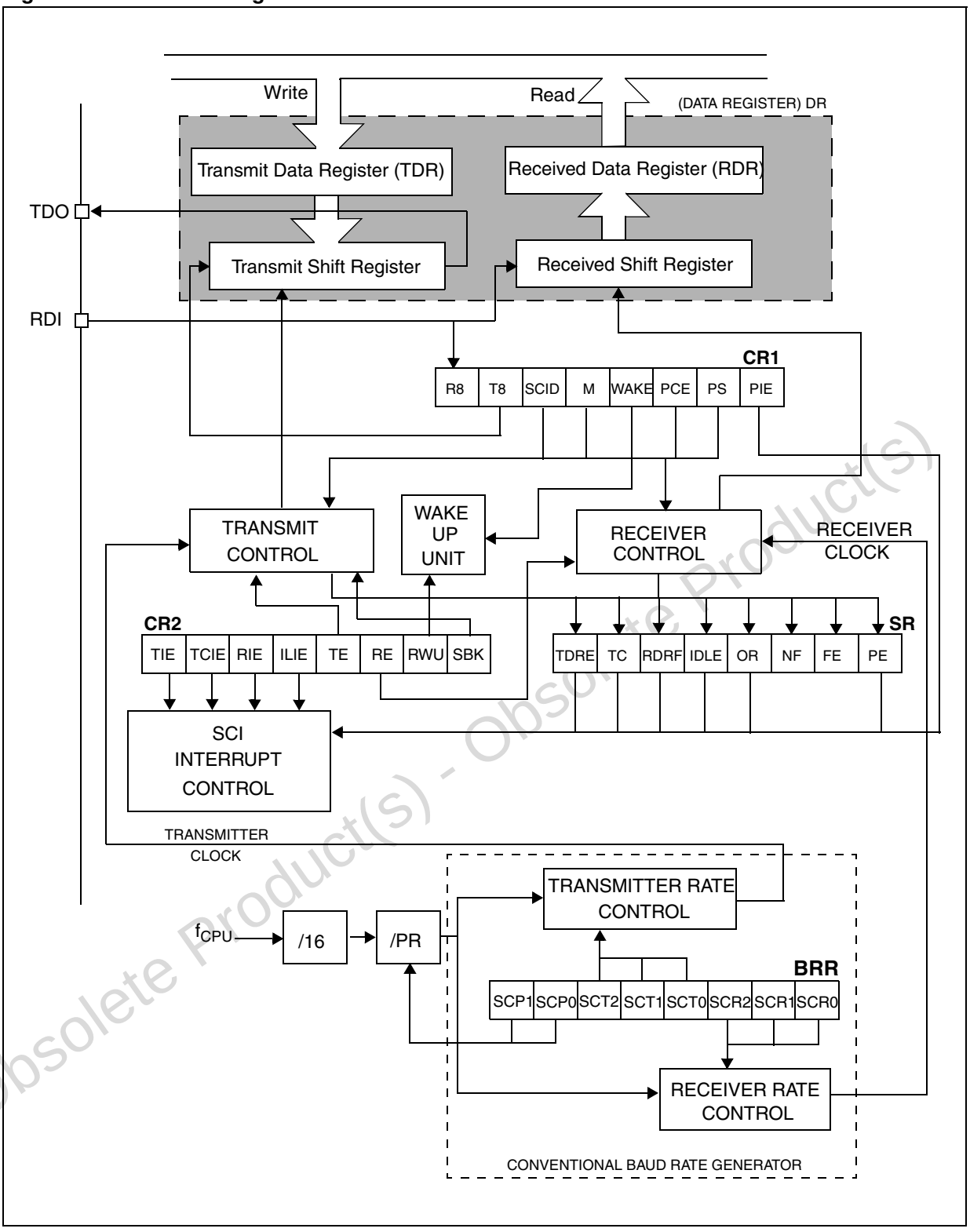
- An Idle Line prior to transmission or reception
- A start bit
- A data word (8 or 9 bits) least significant bit first
- A Stop bit indicating that the frame is complete

This interface uses two types of baud rate generator:

- A conventional type for commonly-used baud rates
- An extended type with a prescaler offering a very wide range of baud rates even with non-standard oscillator frequencies

Obsolete Product(s) - Obsolete Product(s)

**Figure 64. SCI block diagram**




Obsolete Product(s) - Obsolete Product(s)

## 15.4 Functional description

The block diagram of the Serial Control Interface, is shown in *Figure 64*. It contains six dedicated registers:

- 2 control registers (SCICR1 and SCICR2)
- a status register (SCISR)
- a baud rate register (SCIBRR)
- an extended prescaler receiver register (SCIERPR)
- an extended prescaler transmitter register (SCIETPR)

Refer to the register descriptions in *Section 15.7* for the definitions of each bit.

### 15.4.1 Serial data format

Word length may be selected as being either 8 or 9 bits by programming the M bit in the SCICR1 register (see *Figure 64*).

The TDO pin is in low state during the start bit.

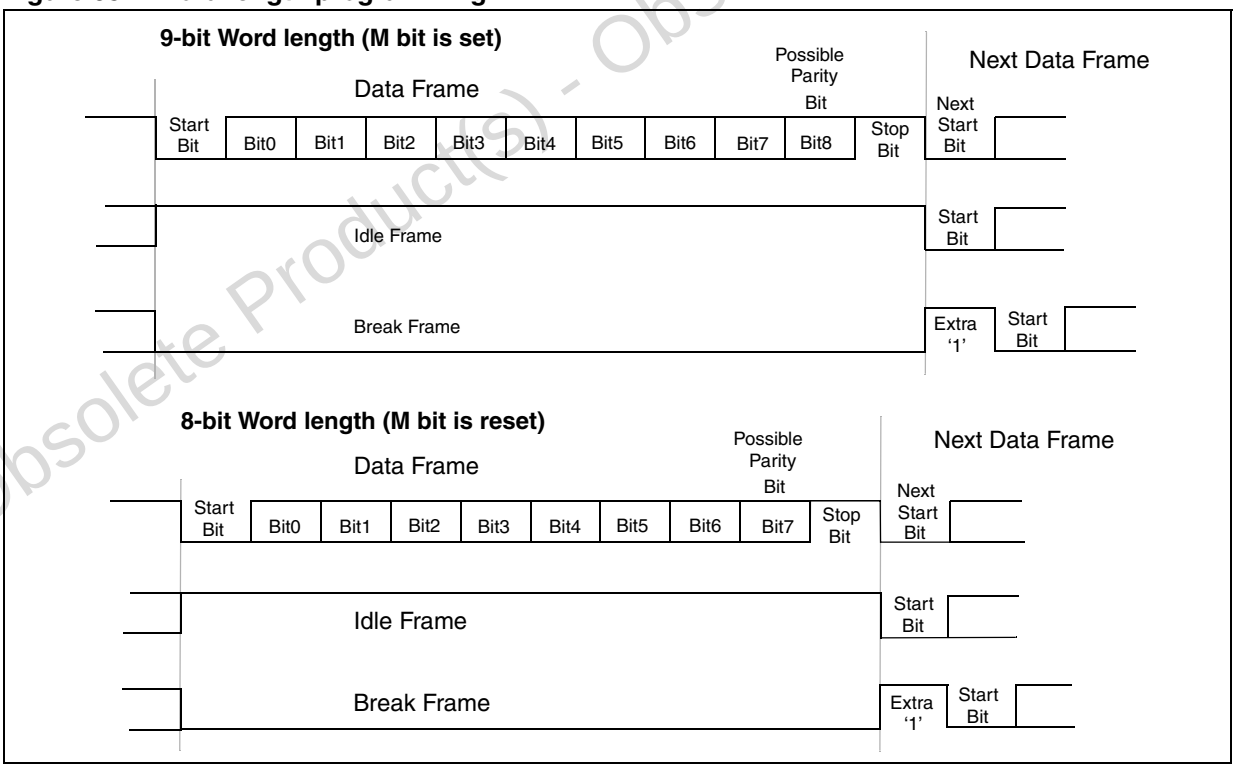
The TDO pin is in high state during the stop bit.

An Idle character is interpreted as an entire frame of ‘1’s followed by the start bit of the next frame which contains data.

A Break character is interpreted on receiving ‘0’s for some multiple of the frame period. At the end of the last break frame the transmitter inserts an extra ‘1’ bit to acknowledge the start bit.

Transmission and reception are driven by their own baud rate generator.

**Figure 65. Word length programming**




Obsolete Product(s) - Obsolete Product(s)

### 15.4.2 Transmitter

The transmitter can send data words of either 8 or 9 bits depending on the M bit status. When the M bit is set, word length is 9 bits and the 9th bit (the MSB) has to be stored in the T8 bit in the SCICR1 register.

#### Character transmission

During an SCI transmission, data shifts out least significant bit first on the TDO pin. In this mode, the SCIDR register consists of a buffer (TDR) between the internal bus and the transmit shift register (see *Figure 64*).

**Procedure**

1. Select the M bit to define the word length.
2. Select the desired baud rate using the SCIBRR and the SCIETPR registers.
3. Set the TE bit to assign the TDO pin to the alternate function and to send an idle frame as first transmission.
4. Access the SCISR register and write the data to send in the SCIDR register (this sequence clears the TDRE bit). Repeat this sequence for each data to be transmitted.

Clearing the TDRE bit is always performed by the following software sequence:

1. An access to the SCISR register
2. A write to the SCIDR register

The TDRE bit is set by hardware and it indicates:

- The TDR register is empty.
- The data transfer is beginning.
- The next data can be written in the SCIDR register without overwriting the previous data.

This flag generates an interrupt if the TIE bit is set and the I bit is cleared in the CCR register.

When a transmission is taking place, a write instruction to the SCIDR register stores the data in the TDR register and which is copied in the shift register at the end of the current transmission.

When no transmission is taking place, a write instruction to the SCIDR register places the data directly in the shift register, the data transmission starts, and the TDRE bit is immediately set.

When a frame transmission is complete (after the stop bit) the TC bit is set and an interrupt is generated if the TCIE is set and the I bit is cleared in the CCR register.

Clearing the TC bit is performed by the following software sequence:

1. An access to the SCISR register
2. A write to the SCIDR register

*Note: The TDRE and TC bits are cleared by the same software sequence.*

**Break characters**

Setting the SBK bit loads the shift register with a break character. The break frame length depends on the M bit (see *Figure 65*).

As long as the SBK bit is set, the SCI send break frames to the TDO pin. After clearing this

Obsolete Product(s) - Obsolete Product(s)

bit by software the SCI insert a logic 1 bit at the end of the last break frame to guarantee the recognition of the start bit of the next frame.

**Idle characters**

Setting the TE bit drives the SCI to send an idle frame before the first data frame.

Clearing and then setting the TE bit during a transmission sends an idle frame after the current word.

*Note: Resetting and setting the TE bit causes the data in the TDR register to be lost. Therefore the best time to toggle the TE bit is when the TDRE bit is set, that is, before writing the next byte in the SCIDR.*

### 15.4.3 Receiver

The SCI can receive data words of either 8 or 9 bits. When the M bit is set, word length is 9 bits and the MSB is stored in the R8 bit in the SCICR1 register.

#### Character reception

During a SCI reception, data shifts in least significant bit first through the RDI pin. In this mode, the SCIDR register consists or a buffer (RDR) between the internal bus and the received shift register (see *Figure 64*).

**Procedure**

1. Select the M bit to define the word length.
2. Select the desired baud rate using the SCIBRR and the SCIERPR registers.
3. Set the RE bit, this enables the receiver which begins searching for a start bit.

When a character is received:

- The RDRF bit is set. It indicates that the content of the shift register is transferred to the RDR.
- An interrupt is generated if the RIE bit is set and the I bit is cleared in the CCR register.
- The error flags can be set if a frame error, noise or an overrun error has been detected during reception.

Clearing the RDRF bit is performed by the following software sequence done by:

1. An access to the SCISR register
2. A read to the SCIDR register.

The RDRF bit must be cleared before the end of the reception of the next character to avoid an overrun error.

**Break character**

When a break character is received, the SCI handles it as a framing error.

**Idle character**

When an idle frame is detected, there is the same procedure as a data received character plus an interrupt if the ILIE bit is set and the I bit is cleared in the CCR register.

**Overrun error**

An overrun error occurs when a character is received when RDRF has not been reset. Data cannot be transferred from the shift register to the RDR register as long as the RDRF bit is not cleared.

Obsolete Product(s) - Obsolete Product(s)

When an overrun error occurs:

- The OR bit is set.
- The RDR content is not lost.
- The shift register is overwritten.
- An interrupt is generated if the RIE bit is set and the I bit is cleared in the CCR register.

The OR bit is reset by an access to the SCISR register followed by a SCIDR register read operation.

**Noise error**

Oversampling techniques are used for data recovery by discriminating between valid incoming data and noise. Normal data bits are considered valid if three consecutive samples (8th, 9th, 10th) have the same bit value, otherwise the NF flag is set. In the case of start bit detection, the NF flag is set on the basis of an algorithm combining both valid edge detection and three samples (8th, 9th, 10th). Therefore, to prevent the NF flag getting set during start bit reception, there should be a valid edge detection as well as three valid samples.

When noise is detected in a frame:

- The NF flag is set at the rising edge of the RDRF bit.
- Data is transferred from the Shift register to the SCIDR register.
- No interrupt is generated. However this bit rises at the same time as the RDRF bit which itself generates an interrupt.

The NF flag is reset by a SCISR register read operation followed by a SCIDR register read operation.

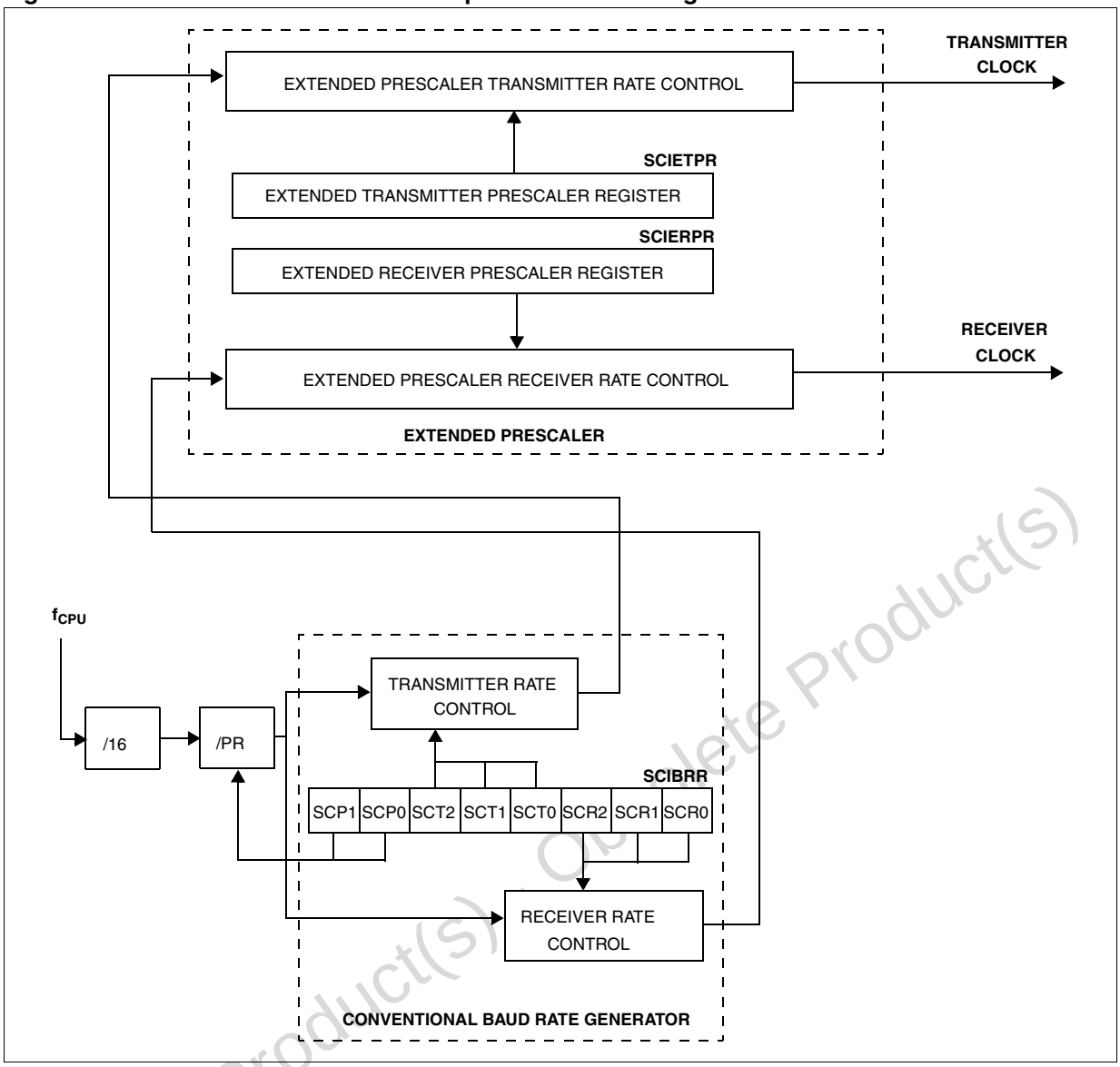
During reception, if a false start bit is detected (for example, 8th, 9th, 10th samples are 011, 101, 110), the frame is discarded and the receiving sequence is not started for this frame. There is no RDRF bit set for this frame and the NF flag is set internally (not accessible to the user). This NF flag is accessible along with the RDRF bit when a next valid frame is received.

*Note: If the application Start Bit is not long enough to match the above requirements, then the NF Flag may get set due to the short Start Bit. In this case, the NF flag may be ignored by the application software when the first valid byte is received.*

See also *Noise error causes on page 149*.

Obsolete Product(s) - Obsolete Product(s)

**Figure 66. SCI baud rate and extended prescaler block diagram**




Obsolete Product(s) - Obsolete Product(s)

### Framing error

A framing error is detected when:

- The stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.
- A break is received.

When the framing error is detected:

- The FE bit is set by hardware.
- Data is transferred from the Shift register to the SCIDR register.
- No interrupt is generated. However this bit rises at the same time as the RDRF bit which itself generates an interrupt.

The FE bit is reset by a SCISR register read operation followed by a SCIDR register read operation.

### Conventional baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are set independently and calculated as follows:

$$Tx = \frac{f_{CPU}}{(16*PR)*TR} \qquad Rx = \frac{f_{CPU}}{(16*PR)*RR}$$

with:

PR = 1, 3, 4 or 13 (see SCP[1:0] bits)

TR = 1, 2, 4, 8, 16, 32, 64,128 (see SCT[2:0] bits)

RR = 1, 2, 4, 8, 16, 32, 64,128 (see SCR[2:0] bits)

All these bits are in the SCIBRR register.

Example: If $f_{CPU}$ is 8 MHz (normal mode) and if PR = 13 and TR = RR = 1, the transmit and receive baud rates are 38400 baud.

*Note: The baud rate registers MUST NOT be changed while the transmitter or the receiver is enabled.*

### Extended baud rate generation

The extended prescaler option provides a very fine tuning of the baud rate, using a 255 value prescaler, whereas the conventional baud rate generator retains industry standard software compatibility.

The extended baud rate generator block diagram is described in the *Figure 66*.

The output clock rate sent to the transmitter or to the receiver is the output from the 16 divider divided by a factor ranging from 1 to 255 set in the SCIERPR or the SCIETPR register.

*Note: The extended prescaler is activated by setting the SCIETPR or SCIERPR register to a value other than zero. The baud rates are calculated as follows:*

$$Tx = \frac{f_{CPU}}{16*ETPR*(PR*TR)} \qquad Rx = \frac{f_{CPU}}{16*ERPR*(PR*RR)}$$

with:

ETPR = 1,..,255 (see SCIETPR register)

ERPR = 1,..,255 (see SCIERPR register)

### Receiver muting and wake-up feature

In multiprocessor configurations it is often desirable that only the intended message recipient should actively receive the full message contents, thus reducing redundant SCI service overhead for all non-addressed receivers.

The non-addressed devices may be placed in sleep mode by means of the muting function.

Setting the RWU bit by software puts the SCI in sleep mode:

- All the reception status bits cannot be set.
- All the receive interrupts are inhibited.

A muted receiver may be awakened by one of the following two ways:

- by Idle Line detection if the WAKE bit is reset
- by Address Mark detection if the WAKE bit is set

Obsolete Product(s) - Obsolete Product(s)

A receiver wakes up by Idle Line detection when the Receive line has recognized an Idle Frame. Then the RWU bit is reset by hardware but the IDLE bit is not set.

Receiver wakes up by Address Mark detection when it received a ‘1’ as the most significant bit of a word, thus indicating that the message is an address. The reception of this particular word wakes up the receiver, resets the RWU bit and sets the RDRF bit, which allows the receiver to receive this word normally and to use it as an address word.

**Caution:** In Mute mode, do not write to the SCICR2 register. If the SCI is in Mute mode during the read operation (RWU = 1) and a address mark wake-up event occurs (RWU is reset) before the write operation, the RWU bit is set again by this write operation. Consequently the address byte is lost and the SCI is not woken up from Mute mode.

### Parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the SCICR1 register. Depending on the frame length defined by the M bit, the possible SCI frame formats are as listed in *Table 71*.

**Table 71. Frame formats**

| M bit | PCE bit | SCI frame |
|---|---|---|
| 0 | 0 | ǀ SB ǀ 8 bit data ǀ STB ǀ |
| 0 | 1 | ǀ SB ǀ 7-bit data ǀ PB ǀ STB ǀ |
| 1 | 0 | ǀ SB ǀ 9-bit data ǀ STB ǀ |
| 1 | 1 | ǀ SB ǀ 8-bit data PB ǀ STB ǀ |

**Legend:** SB = Start Bit, STB = Stop Bit, PB = Parity Bit

*Note: In case of wake-up by an address mark, the MSB bit of the data is taken into account and not the parity bit*

**Even parity:** the parity bit is calculated to obtain an even number of ‘1’s inside the frame made of the 7 or 8 LSB bits (depending on whether M is equal to 0 or 1) and the parity bit.

Example: data = 00110101; 4 bits set => parity bit is 0 if even parity is selected (PS bit = 0).

**Odd parity:** the parity bit is calculated to obtain an odd number of ‘1’s inside the frame made of the 7 or 8 LSB bits (depending on whether M is equal to 0 or 1) and the parity bit.

Example: data = 00110101; 4 bits set => parity bit is 1 if odd parity is selected (PS bit = 1).

**Transmission mode:** If the PCE bit is set then the MSB bit of the data written in the data register is not transmitted but is changed by the parity bit.

**Reception mode:** If the PCE bit is set then the interface checks if the received data byte has an even number of ‘1’s if even parity is selected (PS = 0) or an odd number of ‘1’s if odd parity is selected (PS = 1). If the parity check fails, the PE flag is set in the SCISR register and an interrupt is generated if PIE is set in the SCICR1 register.

### SCI clock tolerance

During reception, each bit is sampled 16 times. The majority of the 8th, 9th and 10th samples is considered as the bit value. For a valid bit detection, all the three samples should have the same value otherwise the noise flag (NF) is set. For example: If the 8th, 9th and 10th samples are 0, 1 and 1 respectively, then the bit value is ‘1’, but the Noise Flag bit is set because the three samples values are not the same.

Obsolete Product(s) - Obsolete Product(s)

Consequently, the bit length must be long enough so that the 8th, 9th and 10th samples have the desired bit value. This means the clock frequency should not vary more than 6/16 (37.5%) within one bit. The sampling clock is resynchronized at each start bit, so that when receiving 10 bits (one start bit, 1 data byte, 1 stop bit), the clock deviation must not exceed 3.75%.

*Note: The internal sampling clock of the microcontroller samples the pin value on every falling edge. Therefore, the internal sampling clock and the time the application expects the sampling to take place may be out of sync. For example: If the baud rate is 15.625 Kbaud (bit length is 64µs), then the 8th, 9th and 10th samples are at 28µs, 32µs and 36µs respectively (the first sample starting ideally at 0µs). But if the falling edge of the internal clock occurs just before the pin value changes, the samples would then be out of sync by ~4us. This means the entire bit length must be at least 40µs (36µs for the 10th sample + 4µs for synchronization with the internal sampling clock).*

### Clock deviation causes

The causes which contribute to the total deviation are:

- $D_{TRA}$: Deviation due to transmitter error (Local oscillator error of the transmitter or the transmitter is transmitting at a different baud rate).
- $D_{QUANT}$: Error due to the baud rate quantization of the receiver.
- $D_{REC}$: Deviation of the local oscillator of the receiver: This deviation can occur during the reception of one complete SCI message assuming that the deviation has been compensated at the beginning of the message.
- $D_{TCL}$: Deviation due to the transmission line (generally due to the transceivers)

All the deviations of the system should be added and compared to the SCI clock tolerance:

$$D_{TRA} + D_{QUANT} + D_{REC} + D_{TCL} < 3.75\%$$

### Noise error causes

See also description of noise error in *Receiver on page 144*.

#### Start bit

The noise flag (NF) is set during start bit reception if one of the following conditions occurs:

1. A valid falling edge is not detected. A falling edge is considered to be valid if the 3 consecutive samples before the falling edge occurs are detected as ‘1’ and, after the falling edge occurs, during the sampling of the 16 samples, if one of the samples numbered 3, 5 or 7 is detected as a ‘1’.
2. During sampling of the 16 samples, if one of the samples numbered 8, 9 or 10 is detected as a ‘1’.

Therefore, a valid Start Bit must satisfy both the above conditions to prevent the Noise Flag getting set.

#### Data bits

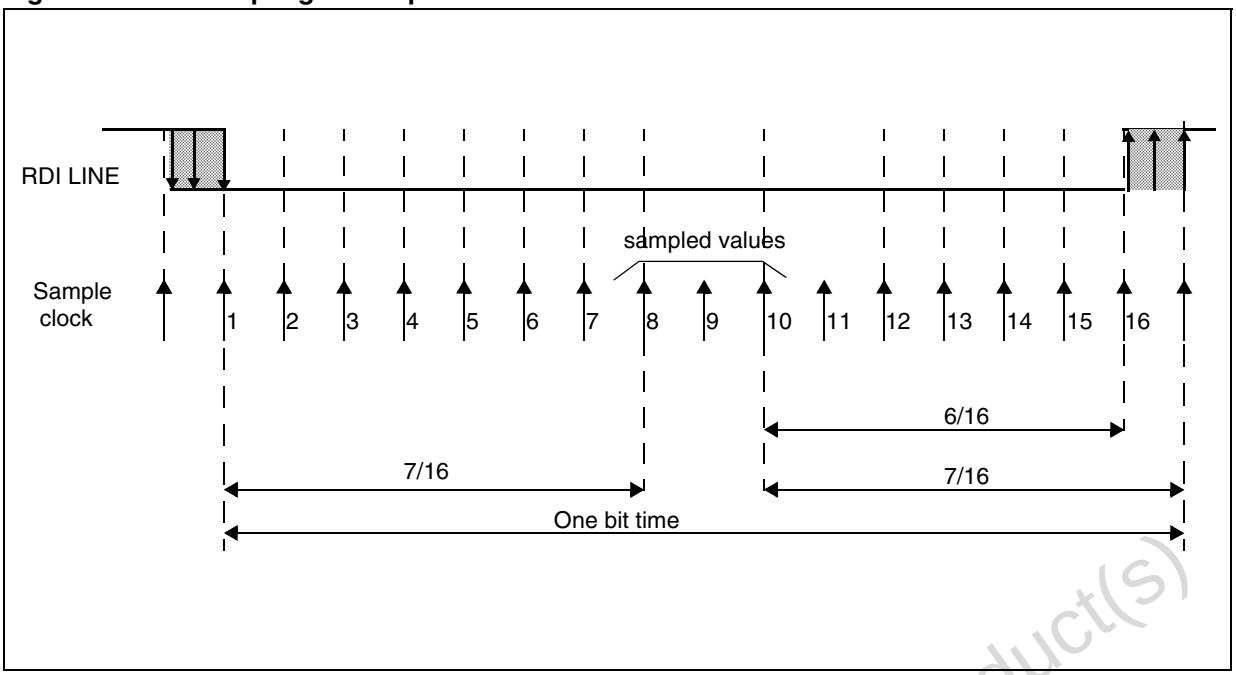
The noise flag (NF) is set during normal data bit reception if the following condition occurs:

- During the sampling of 16 samples, if all three samples numbered 8, 9 and10 are not the same. The majority of the 8th, 9th and 10th samples is considered as the bit value.

Therefore, a valid Data Bit must have samples 8, 9 and 10 at the same value to prevent the Noise Flag from getting set.

Obsolete Product(s) - Obsolete Product(s)

**Figure 67. Bit sampling in reception mode**




## 15.5 Low power modes

**Table 72. Effect of low power modes on SCI**

| Mode | Effect |
|---|---|
| Wait | No effect on SCI.<br>SCI interrupts cause the device to exit from Wait mode. |
| Halt | SCI registers are frozen.<br>In Halt mode, the SCI stops transmitting/receiving until Halt mode is exited. |

## 15.6 Interrupts

The SCI interrupt events are connected to the same interrupt vector.

These events generate an interrupt if the corresponding Enable Control Bit is set and the interrupt mask in the CC register is reset (RIM instruction).

**Table 73. SCI interrupt control/wake-up capability**

| Interrupt event | Event flag | Enable control bit | Exit from Wait | Exit from Halt |
|---|---|---|---|---|
| Transmit Data Register Empty | TDRE | TIE | Yes | No |
| Transmission Complete | TC | TCIE | Yes | No |
| Received Data Ready to be Read | RDRF | RIE | Yes | No |
| Overrun Error Detected | OR | | Yes | No |

Obsolete Product(s) - Obsolete Product(s)

**Table 73. SCI interrupt control/wake-up capability**

| Interrupt event | Event flag | Enable control bit | Exit from Wait | Exit from Halt |
|---|---|---|---|---|
| Idle Line Detected | IDLE | ILIE | Yes | No |
| Parity Error | PE | PIE | Yes | No |

## 15.7 SCI registers

### 15.7.1 Status register (SCISR)

SCISR Reset value: 1100 0000 (C0h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| TDRE | TC | RDRF | IDLE | OR | NF | FE | PE |
| RO | RO | RO | RO | RO | RO | RO | RO |

**Table 74. SCISR register description**

| Bit | Name | Function |
|---|---|---|
| 7 | TDRE | *Transmit data register empty*<br>This bit is set by hardware when the content of the TDR register has been transferred into the shift register. An interrupt is generated if the TIE bit = 1 in the SCICR2 register. It is cleared by a software sequence (an access to the SCISR register followed by a write to the SCIDR register).<br>0: Data is not transferred to the shift register<br>1: Data is transferred to the shift register<br>*Note: Data is not transferred to the shift register unless the TDRE bit is cleared.* |
| 6 | TC | *Transmission complete*<br>This bit is set by hardware when transmission of a frame containing Data is complete. An interrupt is generated if TCIE = 1 in the SCICR2 register. It is cleared by a software sequence (an access to the SCISR register followed by a write to the SCIDR register).<br>0: Transmission is not complete<br>1: Transmission is complete<br>*Note: TC is not set after the transmission of a Preamble or a Break.* |
| 5 | RDRF | *Received data ready flag*<br>This bit is set by hardware when the content of the RDR register has been transferred to the SCIDR register. An interrupt is generated if RIE = 1 in the SCICR2 register. It is cleared by a software sequence (an access to the SCISR register followed by a read to the SCIDR register).<br>0: Data is not received<br>1: Received data is ready to be read |

Obsolete Product(s) - Obsolete Product(s)

**Table 74. SCISR register description (continued)**

| Bit | Name | Function |
|---|---|---|
| 4 | IDLE | *Idle line detect*<br>This bit is set by hardware when an Idle Line is detected. An interrupt is generated if the ILIE = 1 in the SCICR2 register. It is cleared by a software sequence (an access to the SCISR register followed by a read to the SCIDR register).<br>0: No Idle Line is detected<br>1: Idle Line is detected<br>*Note: The IDLE bit is not set again until the RDRF bit has been set itself (that is, a new idle line occurs).* |
| 3 | OR | *Overrun error*<br>This bit is set by hardware when the word currently being received in the shift register is ready to be transferred into the RDR register while RDRF = 1. An interrupt is generated if RIE = 1 in the SCICR2 register. It is cleared by a software sequence (an access to the SCISR register followed by a read to the SCIDR register).<br>0: No Overrun error<br>1: Overrun error is detected<br>*Note: When this bit is set RDR register content is not lost but the shift register is overwritten.* |
| 2 | NF | *Noise flag*<br>This bit is set by hardware when noise is detected on a received frame. It is cleared by a software sequence (an access to the SCISR register followed by a read to the SCIDR register).<br>0: No noise is detected<br>1: Noise is detected<br>*Note: This bit does not generate interrupt as it appears at the same time as the RDRF bit which itself generates an interrupt.* |
| 1 | FE | *Framing error*<br>This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by a software sequence (an access to the SCISR register followed by a read to the SCIDR register).<br>0: No Framing error is detected<br>1: Framing error or break character is detected<br>*Note: This bit does not generate interrupt as it appears at the same time as the RDRF bit which itself generates an interrupt. If the word currently being transferred causes both frame error and overrun error, it will be transferred and only the OR bit will be set.* |
| 0 | PE | *Parity error*<br>This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by a software sequence (a read to the status register followed by an access to the SCIDR data register). An interrupt is generated if PIE = 1 in the SCICR1 register.<br>0: No parity error<br>1: Parity error |

Obsolete Product(s) - Obsolete Product(s)

### 15.7.2 Control register 1 (SCICR1)

SCICR1 Reset value: X000 0000 (x0h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| R8 | T8 | SCID | M | WAKE | PCE | PS | PIE |
| RW | RW | RW | RW | RW | RW | RW | RW |

**Table 75. SCICR1 register description**

| Bit | Name | Function |
|---|---|---|
| 7 | R8 | *Receive data bit 8*<br>This bit is used to store the 9th bit of the received word when M = 1. |
| 6 | T8 | *Transmit data bit 8*<br>This bit is used to store the 9th bit of the transmitted word when M = 1. |
| 5 | SCID | *Disabled for low power consumption*<br>When this bit is set the SCI prescalers and outputs are stopped and the end of the current byte transfer in order to reduce power consumption.This bit is set and cleared by software.<br>0: SCI enabled<br>1: SCI prescaler and outputs disabled |
| 4 | M | *Word length*<br>This bit determines the word length. It is set or cleared by software.<br>0: 1 Start bit, 8 Data bits, 1 Stop bit<br>1: 1 Start bit, 9 Data bits, 1 Stop bit<br>*Note: The M bit must not be modified during a data transfer (both transmission and reception).* |
| 3 | WAKE | *Wake-up method*<br>This bit determines the SCI wake-up method. It is set or cleared by software.<br>0: Idle line<br>1: Address mark |
| 2 | PCE | *Parity control enable*<br>This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).<br>0: Parity control disabled<br>1: Parity control enabled |
| 1 | PS | *Parity selection*<br>This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.<br>0: Even parity<br>1: Odd parity |

Obsolete Product(s) - Obsolete Product(s)

**Table 75. SCICR1 register description (continued)**

| Bit | Name | Function |
|---|---|---|
| 0 | PIE | *Parity interrupt enable*<br>This bit enables the interrupt capability of the hardware parity control when a parity error is detected (PE bit set). It is set and cleared by software.<br>0: Parity error interrupt disabled<br>1: Parity error interrupt enabled |

### 15.7.3 Control register 2 (SCICR2)

SCICR2 Reset value: 0000 0000 (00h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |
| RW | RW | RW | RW | RW | RW | RW | RW |

**Table 76. SCICR2 register description**

| Bit | Name | Function |
|---|---|---|
| 7 | TIE | *Transmitter interrupt enable*<br>This bit is set and cleared by software.<br>0: Interrupt is inhibited<br>1: An SCI interrupt is generated whenever TDRE = 1 in the SCISR register. |
| 6 | TCIE | *Transmission complete interrupt enable*<br>This bit is set and cleared by software.<br>0: Interrupt is inhibited<br>1: An SCI interrupt is generated whenever TC = 1 in the SCISR register. |
| 5 | RIE | *Receiver interrupt enable*<br>This bit is set and cleared by software.<br>0: Interrupt is inhibited<br>1: An SCI interrupt is generated whenever OR = 1 or RDRF = 1 in the SCISR register. |
| 4 | ILIE | *Idle line interrupt enable*<br>This bit is set and cleared by software.<br>0: Interrupt is inhibited<br>1: An SCI interrupt is generated whenever IDLE = 1 in the SCISR register. |
| 3 | TE | *Transmitter enable*<br>This bit enables the transmitter. It is set and cleared by software.<br>0: Transmitter is disabled<br>1: Transmitter is enabled<br>*Notes:*<br>*During transmission, a '0' pulse on the TE bit ('0' followed by '1') sends a preamble (idle line) after the current word.*<br>*When TE is set there is a 1 bit-time delay before the transmission starts.*<br>**Caution**: The TDO pin is free for general purpose I/O only when the TE and RE bits are both cleared (or if TE is never set). |

Obsolete Product(s) - Obsolete Product(s)

**Table 76. SCICR2 register description (continued)**

| Bit | Name | Function |
|---|---|---|
| 2 | RE | *Receiver enable*<br>This bit enables the receiver. It is set and cleared by software.<br>0: Receiver is disabled<br>1: Receiver is enabled and begins searching for a start bit |
| 1 | RWU | *Receiver wake-up*<br>This bit determines if the SCI is in mute mode or not. It is set and cleared by software and can be cleared by hardware when a wake-up sequence is recognized.<br>0: Receiver in Active mode<br>1: Receiver in Mute mode<br>*Note: Before selecting Mute mode (setting the RWU bit), the SCI must receive some data first, otherwise it cannot function in Mute mode with wake-up by idle line detection.* |
| 0 | SBK | *Send break*<br>This bit set is used to send break characters. It is set and cleared by software.<br>0: No break character is transmitted<br>1: Break characters are transmitted<br>*Note: If the SBK bit is set to '1' and then to '0', the transmitter sends a BREAK word at the end of the current word.* |

### 15.7.4 Data register (SCIDR)

This register contains the Received or Transmitted data character, depending on whether it is read from or written to.

SCIDR Reset value: Undefined

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DR[7:0] | | | | | | | |
| RW | | | | | | | |

The Data register performs a double function (read and write) since it is composed of two registers, one for transmission (TDR) and one for reception (RDR).

The TDR register provides the parallel interface between the internal bus and the output shift register (see *Figure 64*).

The RDR register provides the parallel interface between the input shift register and the internal bus (see *Figure 64*).

### 15.7.5 Baud rate register (SCIBRR)

SCIBRR Reset value: 0000 0000 (00h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SCP[1:0] | | SCT[2:0] | | | SCR[2:0] | | |
| RW | | RW | | | RW | | |

Obsolete Product(s) - Obsolete Product(s)

**Table 77. SCIBRR register description**

| Bit | Name | Function |
|---|---|---|
| 7:6 | SCP[1:0] | *First SCI Prescaler*<br>These 2 prescaling bits allow several standard clock division ranges.<br>00: PR prescaling factor = 1<br>01: PR prescaling factor = 3<br>10: PR prescaling factor = 4<br>11: PR prescaling factor = 13 |
| 5:3 | SCT[2:0] | *SCI Transmitter rate divisor*<br>These 3 bits, in conjunction with the SCP1 and SCP0 bits define the total division applied to the bus clock to yield the transmit rate clock in conventional Baud Rate Generator mode.<br>000: TR dividing factor = 1<br>001: TR dividing factor = 2<br>010: TR dividing factor = 4<br>011: TR dividing factor = 8<br>100: TR dividing factor = 16<br>101: TR dividing factor = 32<br>110: TR dividing factor = 64<br>111: TR dividing factor = 128 |
| 2:0 | SCR[2:0] | *SCI Receiver rate divisor*<br>These 3 bits, in conjunction with the SCP[1:0] bits define the total division applied to the bus clock to yield the receive rate clock in conventional Baud Rate Generator mode.<br>000: RR dividing factor = 1<br>001: RR dividing factor = 2<br>010: RR dividing factor = 4<br>011: RR dividing factor = 8<br>100: RR dividing factor = 16<br>101: RR dividing factor = 32<br>110: RR dividing factor = 64<br>111: RR dividing factor = 128 |

### 15.7.6 Extended receive prescaler division register (SCIERPR)

This register allows setting of the extended prescaler rate division factor for the receive circuit.

SCIERPR Reset value: 0000 0000 (00h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| ERPR[7:0] | | | | | | | |
| RW | | | | | | | |

Obsolete Product(s) - Obsolete Product(s)

**Table 78. SCIERPR register description**

| Bit | Name | Function |
|---|---|---|
| 7:0 | ERPR[7:0] | *8-bit Extended Receive Prescaler Register*<br>The extended baud rate generator is activated when a value different from 00h is stored in this register. Therefore the clock frequency issued from the 16 divider (see *Figure 66*) is divided by the binary factor set in the SCIERPR register (in the range 1 to 255).<br>The extended baud rate generator is not used after a reset. |

### 15.7.7 Extended transmit prescaler division register (SCIETPR)

This register allows setting of the external prescaler rate division factor for the transmit circuit.

SCIETPR Reset value: 0000 0000 (00h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| ETPR[7:0] | | | | | | | |
| RW | | | | | | | |

**Table 79. SCIETPR register description**

| Bit | Name | Function |
|---|---|---|
| 7:0 | ETPR[7:0] | *8-bit Extended Transmit Prescaler Register*<br>The extended baud rate generator is activated when a value different from 00h is stored in this register. Therefore the clock frequency issued from the 16 divider (see *Figure 66*) is divided by the binary factor set in the SCIETPR register (in the range 1 to 255).<br>The extended baud rate generator is not used after a reset. |

**Table 80. Baud rate selection**

| Symbol | Parameter | Conditions | | | Standard | Baud rate | Unit |
|---|---|---|---|---|---|---|---|
| | | $f_{CPU}$ | Accuracy versus standard | Prescaler | | | |
| $f_{Tx}$<br>$f_{Rx}$ | Communication frequency | 8 MHz | ~0.16% | Conventional mode<br>TR (or RR) = 128, PR = 13<br>TR (or RR) = 32, PR = 13<br>TR (or RR) = 16, PR = 13<br>TR (or RR) = 8, PR = 13<br>TR (or RR) = 4, PR = 13<br>TR (or RR) = 16, PR = 3<br>TR (or RR) = 2, PR = 13<br>TR (or RR) = 1, PR = 13 | 300<br>1200<br>2400<br>4800<br>9600<br>10400<br>19200<br>38400 | ~300.48<br>~1201.92<br>~2403.84<br>~4807.69<br>~9615.38<br>~10416.67<br>~19230.77<br>~38461.54 | Hz |
| | | | ~0.79% | Extended mode<br>ETPR (or ERPR) = 35,<br>TR (or RR) = 1, PR = 1 | 14400 | ~14285.71 | |

Obsolete Product(s) - Obsolete Product(s)

**Table 81. SCI register map and reset values**

| Address (Hex.) | Register label | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0050h | SCISR<br>Reset value | TDRE<br>1 | TC<br>1 | RDRF<br>0 | IDLE<br>0 | OR<br>0 | NF<br>0 | FE<br>0 | PE<br>0 |
| 0051h | SCIDR<br>Reset value | MSB<br>x | x | x | x | x | x | x | LSB<br>x |
| 0052h | SCIBRR<br>Reset value | SCP1<br>0 | SCP0<br>0 | SCT2<br>0 | SCT1<br>0 | SCT0<br>0 | SCR2<br>0 | SCR1<br>0 | SCR0<br>0 |
| 0053h | SCICR1<br>Reset value | R8<br>x | T8<br>0 | SCID<br>0 | M<br>0 | WAKE<br>0 | PCE<br>0 | PS<br>0 | PIE<br>0 |
| 0054h | SCICR2<br>Reset value | TIE<br>0 | TCIE<br>0 | RIE<br>0 | ILIE<br>0 | TE<br>0 | RE<br>0 | RWU<br>0 | SBK<br>0 |
| 0055h | SCIERPR<br>Reset value | MSB<br>0 | 0 | 0 | 0 | 0 | 0 | 0 | LSB<br>0 |
| 0057h | SCIPETPR<br>Reset value | MSB<br>0 | 0 | 0 | 0 | 0 | 0 | 0 | LSB<br>0 |

Obsolete Product(s) - Obsolete Product(s)

# 16 I²C bus interface (I2C)

## 16.1 Introduction

The I²C bus interface serves as an interface between the microcontroller and the serial I²C bus. It provides both multimaster and slave functions, and controls all I²C bus-specific sequencing, protocol, arbitration and timing. It supports fast I²C mode (400 kHz).

## 16.2 Main features

- Parallel-bus/I²C protocol converter
- Multimaster capability
- 7-bit/10-bit addressing
- SMBus V1.1 compliant
- Transmitter/Receiver flag
- End-of-byte transmission flag
- Transfer problem detection

### 16.2.1 I²C master features

- Clock generation
- I²C bus busy flag
- Arbitration Lost flag
- End of byte transmission flag
- Transmitter/Receiver flag
- Start bit detection flag
- Start and Stop generation

### 16.2.2 I²C slave features

- Stop bit detection
- I²C bus busy flag
- Detection of misplaced start or stop condition
- Programmable I²C address detection
- Transfer problem detection
- End-of-byte transmission flag
- Transmitter/Receiver flag

Obsolete Product(s) - Obsolete Product(s)

## 16.3 General description

In addition to receiving and transmitting data, this interface converts it from serial to parallel format and vice versa, using either an interrupt or polled handshake. The interrupts are enabled or disabled by software. The interface is connected to the $I^2C$ bus by a data pin (SDAI) and by a clock pin (SCLI). It can be connected both with a standard $I^2C$ bus and a fast $I^2C$ bus. This selection is made by software.

### 16.3.1 Mode selection

The interface can operate in the four following modes:

- Slave transmitter/receiver
- Master transmitter/receiver

By default, it operates in slave mode.

The interface automatically switches from slave to master after it generates a START condition and from master to slave in case of arbitration loss or a STOP generation, allowing then Multimaster capability.

### 16.3.2 Communication flow

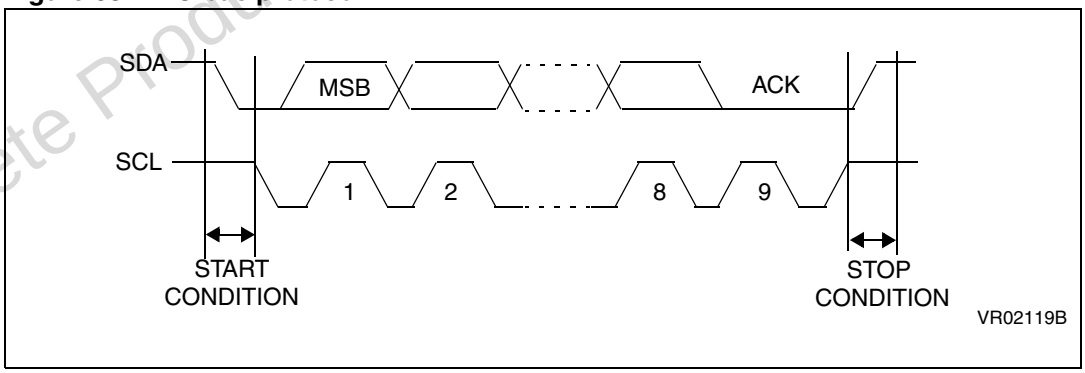
In Master mode, it initiates a data transfer and generates the clock signal. A serial data transfer always begins with a start condition and ends with a stop condition. Both start and stop conditions are generated in master mode by software.

In Slave mode, the interface is capable of recognizing its own address (7- or 10-bit), and the General Call address. The General Call address detection may be enabled or disabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The first byte(s) following the start condition contain the address (one in 7-bit mode, two in 10-bit mode). The address is always transmitted in Master mode.

A 9th clock pulse follows the 8 clock cycles of a byte transfer, during which the receiver must send an acknowledge bit to the transmitter. Refer to *Figure 68*.

**Figure 68. $I^2C$ bus protocol**




Acknowledge may be enabled and disabled by software.

The $I^2C$ interface address and/or general call address can be selected by software.

The speed of the $I^2C$ interface may be selected between standard (up to 100 kHz) and fast $I^2C$ (up to 400 kHz).

Obsolete Product(s) - Obsolete Product(s)

### 16.3.3 SDA/SCL line control

#### Transmitter mode

The interface holds the clock line low before transmission to wait for the microcontroller to write the byte in the data register.

#### Receiver mode

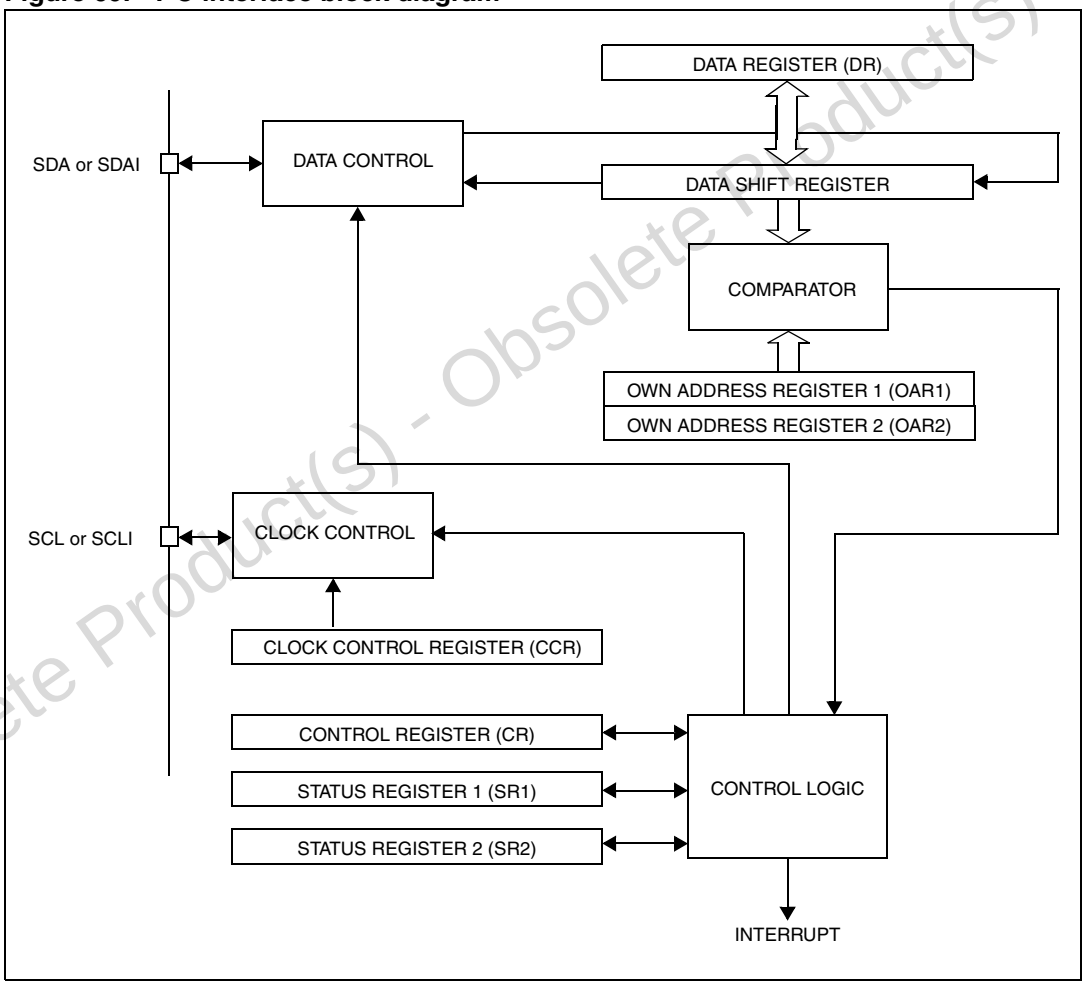
The interface holds the clock line low after reception to wait for the microcontroller to read the byte in the data register.

The SCL frequency ($f_{SCL}$) is controlled by a programmable clock divider which depends on the $I^2C$ bus mode.

When the $I^2C$ cell is enabled, the SDA and SCL ports must be configured as floating inputs. In this case, the value of the external pull-up resistor used depends on the application.

When the $I^2C$ cell is disabled, the SDA and SCL ports revert to being standard I/O port pins.

**Figure 69. $I^2C$ interface block diagram**




Obsolete Product(s) - Obsolete Product(s)

## 16.4 Functional description

Refer to the CR, SR1 and SR2 registers in *Section 16.7* for the bit definitions.

By default the $I^2C$ interface operates in Slave mode (M/SL bit is cleared) except when it initiates a transmit or receive sequence.

First the interface frequency must be configured using the FRi bits in the OAR2 register.

### 16.4.1 Slave mode

As soon as a start condition is detected, the address is received from the SDA line and sent to the shift register; then it is compared with the address of the interface or the General Call address (if selected by software).

*Note: In 10-bit addressing mode, the comparison includes the header sequence (11110xx0) and the two most significant bits of the address.*

**Header matched** (10-bit mode only): The interface generates an acknowledge pulse if the ACK bit is set.

**Address not matched**: The interface ignores it and waits for another Start condition.

**Address matched**: The interface generates in sequence:

- an acknowledge pulse if the ACK bit is set
- EVF and ADSL bits are set with an interrupt if the ITE bit is set.

Then the interface waits for a read of the SR1 register, **holding the SCL line low** (see *Figure 70: Transfer sequencing* EV1).

Next, in 7-bit mode read the DR register to determine from the least significant bit (Data Direction Bit) if the slave must enter Receiver or Transmitter mode.

In 10-bit mode, after receiving the address sequence the slave is always in receive mode. It will enter transmit mode on receiving a repeated Start condition followed by the header sequence with matching address bits and the least significant bit set (11110xx1).

#### Slave receiver

Following the address reception and after the SR1 register has been read, the slave receives bytes from the SDA line into the DR register via the internal shift register. After each byte the interface generates in sequence:

- an acknowledge pulse if the ACK bit is set
- EVF and BTF bits are set with an interrupt if the ITE bit is set.

Then the interface waits for a read of the SR1 register followed by a read of the DR register, **holding the SCL line low** (see *Figure 70: Transfer sequencing* EV2).

#### Slave transmitter

Following the address reception and after SR1 register has been read, the slave sends bytes from the DR register to the SDA line via the internal shift register.

The slave waits for a read of the SR1 register followed by a write in the DR register, **holding the SCL line low** (see *Figure 70: Transfer sequencing* EV3).

When the acknowledge pulse is received:

- The EVF and BTF bits are set by hardware with an interrupt if the ITE bit is set.

Obsolete Product(s) - Obsolete Product(s)

### Closing slave communication

After the last data byte is transferred, a Stop Condition is generated by the master. The interface detects this condition and sets:

- EVF and STOPF bits with an interrupt if the ITE bit is set.

Then the interface waits for a read of the SR2 register (see *Figure 70: Transfer sequencing* EV4).

### Error cases

- **BERR**: Detection of a Stop or a Start condition during a byte transfer. In this case, the EVF and the BERR bits are set with an interrupt if the ITE bit is set.
  If it is a Stop then the interface discards the data, released the lines and waits for another Start condition.
  If it is a Start then the interface discards the data and waits for the next slave address on the bus.
- **AF**: Detection of a non-acknowledge bit. In this case, the EVF and AF bits are set with an interrupt if the ITE bit is set.
  The AF bit is cleared by reading the I2CSR2 register. However, if read before the completion of the transmission, the AF flag will be set again, thus possibly generating a new interrupt. Software must ensure either that the SCL line is back at 0 before reading the SR2 register, or be able to correctly handle a second interrupt during the 9th pulse of a transmitted byte.

*Note: In case of errors, the SCL line is not held low; however, the SDA line can remain low if the last bits transmitted are all 0. While AF = 1, the SCL line may be held low due to SB or BTF flags that are set at the same time. It is then necessary to release both lines by software.*

#### How to release the SDA / SCL lines

Set and subsequently clear the STOP bit while BTF is set. The SDA/SCL lines are released after the transfer of the current byte.

#### SMBus compatibility

The ST7 $I^2C$ is compatible with the SMBus V1.1 protocol. It supports all SMBus addressing modes, SMBus bus protocols and CRC-8 packet error checking. Refer to *SMBus Slave Driver For ST7 $I^2C$ Peripheral* (AN1713).

## 16.4.2 Master mode

To switch from default Slave mode to Master mode a Start condition generation is needed.

### Start condition

Setting the START bit while the BUSY bit is cleared causes the interface to switch to Master mode (M/SL bit set) and generates a Start condition.

Once the Start condition is sent:

- The EVF and SB bits are set by hardware with an interrupt if the ITE bit is set.

Then the master waits for a read of the SR1 register followed by a write in the DR register with the Slave address, **holding the SCL line low** (see *Figure 70: Transfer sequencing* EV5).

Obsolete Product(s) - Obsolete Product(s)

### Slave address transmission

Then the slave address is sent to the SDA line via the internal shift register.

- In 7-bit addressing mode, one address byte is sent.
- In 10-bit addressing mode, sending the first byte including the header sequence causes the following event:
  - The EVF bit is set by hardware with interrupt generation if the ITE bit is set.

Then the master waits for a read of the SR1 register followed by a write in the DR register, **holding the SCL line low** (see *Figure 70: Transfer sequencing* EV9).

Then the second address byte is sent by the interface.

After completion of this transfer (and acknowledge from the slave if the ACK bit is set):

- The EVF bit is set by hardware with interrupt generation if the ITE bit is set.

Then the master waits for a read of the SR1 register followed by a write in the CR register (for example set PE bit), **holding the SCL line low** (see *Figure 70: Transfer sequencing* EV6).

Next, the master must enter Receiver or Transmitter mode.

*Note: In 10-bit addressing mode, to switch the master to Receiver mode, software must generate a repeated Start condition and resend the header sequence with the least significant bit set (11110xx1).*

### Master receiver

Following the address transmission and after SR1 and CR registers have been accessed, the master receives bytes from the SDA line into the DR register via the internal shift register. After each byte the interface generates in sequence:

- Acknowledge pulse if the ACK bit is set
- EVF and BTF bits are set by hardware with an interrupt if the ITE bit is set.

Then the interface waits for a read of the SR1 register followed by a read of the DR register, **holding the SCL line low** (see *Figure 70: Transfer sequencing* EV7).

To close the communication: Before reading the last byte from the DR register, set the STOP bit to generate the Stop condition. The interface goes automatically back to slave mode (M/SL bit cleared).

*Note: In order to generate the non-acknowledge pulse after the last received data byte, the ACK bit must be cleared just before reading the second last data byte.*

### Master transmitter

Following the address transmission and after SR1 register has been read, the master sends bytes from the DR register to the SDA line via the internal shift register.

The master waits for a read of the SR1 register followed by a write in the DR register, **holding the SCL line low** (see *Figure 70: Transfer sequencing* EV8).

When the acknowledge bit is received, the interface sets:

- EVF and BTF bits with an interrupt if the ITE bit is set.

To close the communication: After writing the last byte to the DR register, set the STOP bit to generate the Stop condition. The interface goes automatically back to slave mode (M/SL bit cleared).

Obsolete Product(s) - Obsolete Product(s)

### Error cases

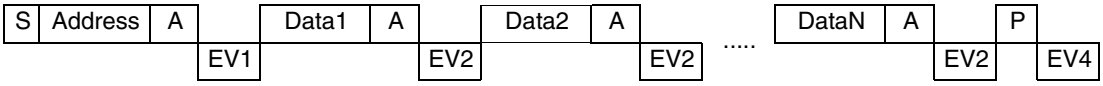
- **BERR**: Detection of a Stop or a Start condition during a byte transfer. In this case, the EVF and BERR bits are set by hardware with an interrupt if ITE is set.
  Note that BERR will not be set if an error is detected during the first or second pulse of each 9-bit transaction:
  - ***Single Master Mode***
    If a Start or Stop is issued during the first or second pulse of a 9-bit transaction, the BERR flag will not be set and transfer will continue however the BUSY flag will be reset. To work around this, slave devices should issue a NACK when they receive a misplaced Start or Stop. The reception of a NACK or BUSY by the master in the middle of communication makes it possible to re-initiate transmission.
  - ***Multimaster Mode***
    Normally the BERR bit would be set whenever unauthorized transmission takes place while transfer is already in progress. However, an issue will arise if an external master generates an unauthorized Start or Stop while the $I^2C$ master is on the first or second pulse of a 9-bit transaction. It is possible to work around this by polling the BUSY bit during $I^2C$ master mode transmission. The resetting of the BUSY bit can then be handled in a similar manner as the BERR flag being set.
- **AF**: Detection of a non-acknowledge bit. In this case, the EVF and AF bits are set by hardware with an interrupt if the ITE bit is set. To resume, set the Start or Stop bit. The AF bit is cleared by reading the I2CSR2 register. However, if read before the completion of the transmission, the AF flag will be set again, thus possibly generating a new interrupt. Software must ensure either that the SCL line is back at 0 before reading the SR2 register, or be able to correctly handle a second interrupt during the 9th pulse of a transmitted byte.
- **ARLO:** Detection of an arbitration lost condition.
  In this case the ARLO bit is set by hardware (with an interrupt if the ITE bit is set and the interface goes automatically back to slave mode (the M/SL bit is cleared).

*Note: In all these cases, the SCL line is not held low; however, the SDA line can remain low due to possible '0' bits transmitted last. It is then necessary to release both lines by software.*

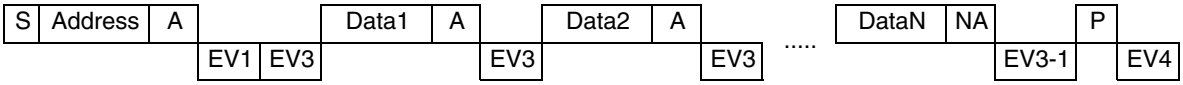
Obsolete Product(s) - Obsolete product(s)

**Figure 70. Transfer sequencing**

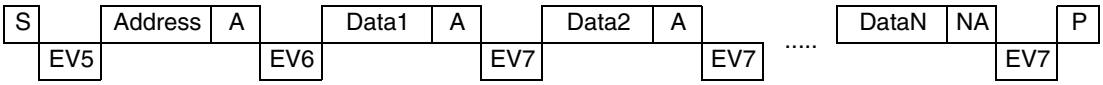
**7-bit Slave receiver:**



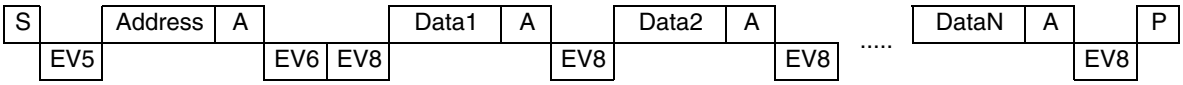

**7-bit Slave transmitter:**



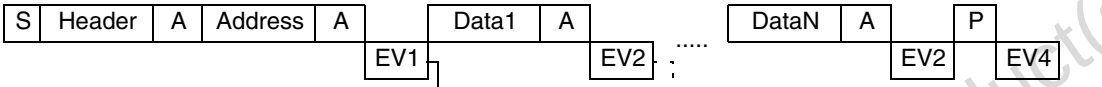

**7-bit Master receiver:**



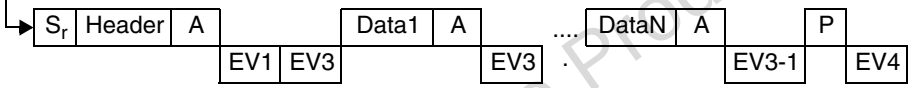

**7-bit Master transmitter:**



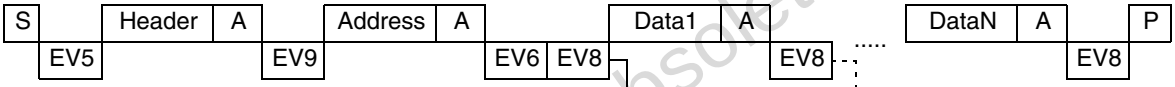

**10-bit Slave receiver:**



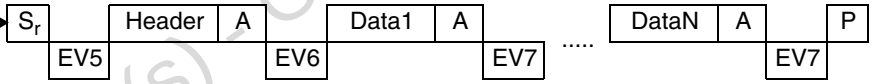

**10-bit Slave transmitter:**




**10-bit Master transmitter:**




**10-bit Master receiver:**




**Legend:**

S = Start, Sr = Repeated Start, P = Stop, A = Acknowledge, NA = Non-acknowledge, EVx = Event (with interrupt if ITE = 1)

EV1: EVF = 1, ADSL = 1, cleared by reading SR1 register.

EV2: EVF = 1, BTF = 1, cleared by reading SR1 register followed by reading DR register.

EV3: EVF = 1, BTF = 1, cleared by reading SR1 register followed by writing DR register.

EV3-1: EVF = 1, AF = 1, BTF = 1; AF is cleared by reading SR1 register. BTF is cleared by releasing the lines (STOP = 1, STOP = 0) or by writing DR register (DR = FFh). ***Note:*** *If lines are released by STOP = 1, STOP = 0, the subsequent EV4 is not seen.*

EV4: EVF = 1, STOPF = 1, cleared by reading SR2 register.

EV5: EVF = 1, SB = 1, cleared by reading SR1 register followed by writing DR register.

EV6: EVF = 1, cleared by reading SR1 register followed by writing CR register (for example PE = 1).

EV7: EVF = 1, BTF = 1, cleared by reading SR1 register followed by reading DR register.

EV8: EVF = 1, BTF = 1, cleared by reading SR1 register followed by writing DR register.

EV9: EVF = 1, ADD10 = 1, cleared by reading SR1 register followed by writing DR register.

Obsolete Product(s) - Obsolete Product(s)

## 16.5 Low power modes

**Table 82. Effect of low power modes on $I^2C$**

| Mode | Effect |
|---|---|
| Wait | No effect on $I^2C$ interface.<br>$I^2C$ interrupts cause the device to exit from Wait mode. |
| Halt | $I^2C$ registers are frozen.<br>In Halt mode, the $I^2C$ interface is inactive and does not acknowledge data on the bus. The $I^2C$ interface resumes operation when the MCU is woken up by an interrupt with “exit from Halt mode” capability. |

## 16.6 Interrupts

**Figure 71. Interrupt control logic diagram**

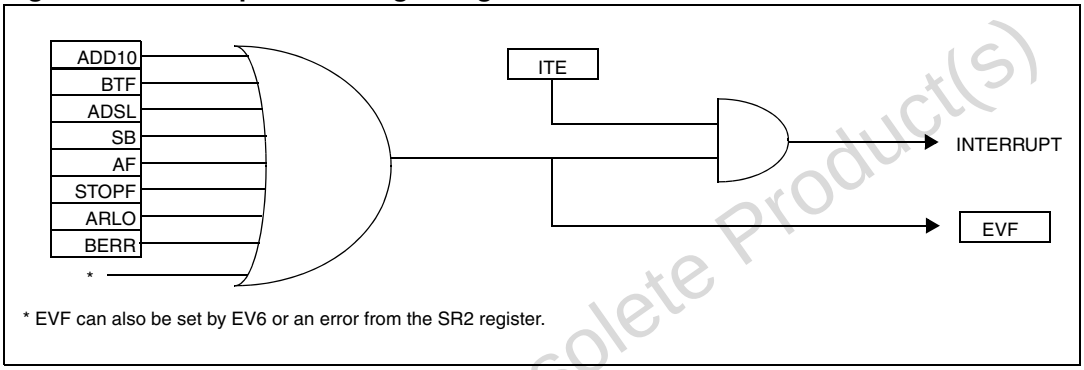



**Table 83. $I^2C$ interrupt control/wake-up capability**

| Interrupt event | Event flag | Enable control bit | Exit from Wait | Exit from Halt |
|---|---|---|---|---|
| 10-bit Address Sent Event (Master mode) | ADD10 | ITE | Yes | No |
| End of Byte Transfer Event | BTF | | | |
| Address Matched Event (Slave mode) | ADSEL | | | |
| Start Bit Generation Event (Master mode) | SB | | | |
| Acknowledge Failure Event | AF | | | |
| Stop Detection Event (Slave mode) | STOPF | | | |
| Arbitration Lost Event (Multimaster configuration) | ARLO | | | |
| Bus Error Event | BERR | | | |

*Note: The $I^2C$ interrupt events are connected to the same interrupt vector (see Section 7: Interrupts). They generate an interrupt if the corresponding Enable Control bit is set and the I-bit in the CC register is reset (RIM instruction).*

Obsolete Product(s) - Obsolete Product(s)

## 16.7 Register description

### 16.7.1 I²C control register (CR)

CR Reset value: 0000 0000 (00h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | PE | ENGC | START | ACK | STOP | ITE |
| - | | RW | RW | RW | RW | RW | RW |

**Table 84. CR register description**

| Bit | Name | Function |
|---|---|---|
| 7:6 | - | Reserved. Forced to 0 by hardware. |
| 5 | PE | *Peripheral enable*<br>This bit is set and cleared by software.<br>0: Peripheral disabled<br>1: Master/Slave capability<br>*Notes:*<br>*- When PE = 0, all the bits of the CR register and the SR register except the Stop bit are reset. All outputs are released while PE = 0*<br>*- When PE = 1, the corresponding I/O pins are selected by hardware as alternate functions.*<br>*To enable the I²C interface, write the CR register **TWICE** with PE = 1 as the first write only activates the interface (only PE is set).* |
| 4 | ENGC | *Enable General Call*<br>This bit is set and cleared by software. It is also cleared by hardware when the interface is disabled (PE = 0). The 00h General Call address is acknowledged (01h ignored).<br>0: General Call disabled<br>1: General Call enabled<br>*Note: In accordance with the I2C standard, when GCAL addressing is enabled, an I2C slave can only receive data. It will not transmit data to the master.* |
| 3 | START | *Generation of a Start condition*<br>This bit is set and cleared by software. It is also cleared by hardware when the interface is disabled (PE = 0) or when the Start condition is sent (with interrupt generation if ITE = 1).<br>**In Master mode**<br>0: No start generation<br>1: Repeated start generation<br>**In Slave mode**<br>0: No start generation<br>1: Start generation when the bus is free |
| 2 | ACK | *Acknowledge enable*<br>This bit is set and cleared by software. It is also cleared by hardware when the interface is disabled (PE = 0).<br>0: No acknowledge returned<br>1: Acknowledge returned after an address byte or a data byte is received |

Obsolete Product(s)

**Table 84. CR register description (continued)**

| Bit | Name | Function |
|---|---|---|
| 1 | STOP | *Generation of a Stop condition*<br>This bit is set and cleared by software. It is also cleared by hardware in master mode.<br>*Note: This bit is not cleared when the interface is disabled (PE = 0).*<br>**In Master mode**<br>0: No stop generation<br>1: Stop generation after the current byte transfer or after the current Start condition is sent. The STOP bit is cleared by hardware when the Stop condition is sent.<br>**In Slave mode**<br>0: No stop generation<br>1: Release the SCL and SDA lines after the current byte transfer (BTF = 1). In this mode the STOP bit has to be cleared by software. |
| 0 | ITE | *Interrupt enable*<br>This bit is set and cleared by software and cleared by hardware when the interface is disabled (PE = 0).<br>0: Interrupts disabled<br>1: Interrupts enabled<br>Refer to *Figure 71* and *Table 83* for the relationship between the events and the interrupt.<br>SCL is held low when the ADD10, SB, BTF or ADSL flags or an EV6 event (see *Figure 70*) is detected. |

## 16.7.2 I²C status register 1 (SR1)

SR1 Reset value: 0000 0000 (00h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| EVF | ADD10 | TRA | BUSY | BTF | ADSL | M/SL | SB |
| RO | RO | RO | RO | RO | RO | RO | RO |

**Table 85. SR1 register description**

| Bit | Name | Function |
|---|---|---|
| 7 | EVF | *Event flag*<br>This bit is set by hardware as soon as an event occurs. It is cleared by software reading SR2 register in case of error event or as described in *Figure 70*. It is also cleared by hardware when the interface is disabled (PE = 0).<br>0: No event<br>1: One of the following events has occurred:<br>- BTF = 1 (Byte received or transmitted)<br>- ADSL = 1 (Address matched in Slave mode while ACK = 1)<br>- SB = 1 (Start condition generated in Master mode)<br>- AF = 1 (No acknowledge received after byte transmission)<br>- STOPF = 1 (Stop condition detected in Slave mode)<br>- ARLO = 1 (Arbitration lost in Master mode)<br>- BERR = 1 (Bus error, misplaced Start or Stop condition detected)<br>- ADD10 = 1 (Master has sent header byte)<br>- Address byte successfully transmitted in Master mode |

Obsolete Product(s) - Obsolete Product(s)

**Table 85. SR1 register description (continued)**

| Bit | Name | Function |
|---|---|---|
| 6 | ADD10 | *10-bit addressing in Master mode*<br>This bit is set by hardware when the master has sent the first byte in 10-bit address mode. It is cleared by software reading SR2 register followed by a write in the DR register of the second address byte. It is also cleared by hardware when the peripheral is disabled (PE = 0).<br>0: No ADD10 event occurred.<br>1: Master has sent first address byte (header) |
| 5 | TRA | *Transmitter/Receiver*<br>When BTF is set, TRA = 1 if a data byte has been transmitted. It is cleared automatically when BTF is cleared. It is also cleared by hardware after detection of Stop condition (STOPF = 1), loss of bus arbitration (ARLO = 1) or when the interface is disabled (PE = 0).<br>0: Data byte received (if BTF = 1)<br>1: Data byte transmitted |
| 4 | BUSY | *Bus busy*<br>This bit is set by hardware on detection of a Start condition and cleared by hardware on detection of a Stop condition. It indicates a communication in progress on the bus. The BUSY flag of the I2CSR1 register is cleared if a Bus Error occurs.<br>0: No communication on the bus<br>1: Communication ongoing on the bus<br>*Note: The BUSY flag is NOT updated when the interface is disabled (PE = 0). This can have consequences when operating in Multimaster mode; that is, a second active $I^2C$ master commencing a transfer with an unset BUSY bit can cause a conflict resulting in lost data. A software workaround consists of checking that the $I^2C$ is not busy before enabling the $I^2C$ Multimaster cell.* |
| 3 | BTF | *Byte transfer finished*<br>This bit is set by hardware as soon as a byte is correctly received or transmitted with interrupt generation if ITE = 1. It is cleared by software reading SR1 register followed by a read or write of DR register. It is also cleared by hardware when the interface is disabled (PE = 0).<br>Following a byte transmission, this bit is set after reception of the acknowledge clock pulse. In case an address byte is sent, this bit is set only after the EV6 event (see *Figure 70*). BTF is cleared by reading SR1 register followed by writing the next byte in DR register.<br>Following a byte reception, this bit is set after transmission of the acknowledge clock pulse if ACK = 1. BTF is cleared by reading SR1 register followed by reading the byte from DR register.<br>The SCL line is held low while BTF = 1.<br>0: Byte transfer not done<br>1: Byte transfer succeeded |
| 2 | ADSL | *Address matched (Slave mode)*<br>This bit is set by hardware as soon as the received slave address matched with the OAR register content or a general call is recognized. An interrupt is generated if ITE = 1. It is cleared by software reading SR1 register or by hardware when the interface is disabled (PE = 0).<br>The SCL line is held low while ADSL = 1.<br>0: Address mismatched or not received<br>1: Received address matched |

Obsolete Product(s) - Obsolete Product(s)

**Table 85. SR1 register description (continued)**

| Bit | Name | Function |
|---|---|---|
| 1 | M/SL | *Master/Slave*<br>This bit is set by hardware as soon as the interface is in Master mode (writing START = 1). It is cleared by hardware after detecting a Stop condition on the bus or a loss of arbitration (ARLO = 1). It is also cleared when the interface is disabled (PE = 0).<br>0: Slave mode<br>1: Master mode |
| 0 | SB | *Start bit (Master mode)*<br>This bit is set by hardware as soon as the Start condition is generated (following a write START = 1). An interrupt is generated if ITE = 1. It is cleared by software reading SR1 register followed by writing the address byte in DR register. It is also cleared by hardware when the interface is disabled (PE = 0).<br>0: No Start condition<br>1: Start condition generated |

### 16.7.3 I²C status register 2 (SR2)

SR2 Reset value: 0000 0000 (00h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | | AF | STOPF | ARLO | BERR | GCAL |
| - | | | RO | RO | RO | RO | RO |

**Table 86. SR2 register description**

| Bit | Name | Function |
|---|---|---|
| 7:5 | - | Reserved. Forced to 0 by hardware. |
| 4 | AF | *Acknowledge failure*<br>This bit is set by hardware when no acknowledge is returned. An interrupt is generated if ITE = 1. It is cleared by software reading SR2 register or by hardware when the interface is disabled (PE = 0).<br>The SCL line is not held low while AF = 1 but by other flags (SB or BTF) that are set at the same time.<br>0: No acknowledge failure<br>1: Acknowledge failure<br>*Note: When an AF event occurs, the SCL line is not held low; however, the SDA line can remain low if the last bits transmitted are all 0. It is then necessary to release both lines by software.* |
| 3 | STOPF | *Stop detection (Slave mode)*<br>This bit is set by hardware when a Stop condition is detected on the bus after an acknowledge (if ACK = 1). An interrupt is generated if ITE = 1. It is cleared by software reading SR2 register or by hardware when the interface is disabled (PE = 0).<br>The SCL line is not held low while STOPF = 1.<br>0: No Stop condition detected<br>1: Stop condition detected |

Obsolete Product(s) - Obsolete Product(s)

**Table 86. SR2 register description (continued)**

| Bit | Name | Function |
|---|---|---|
| 2 | ARLO | *Arbitration lost*<br>This bit is set by hardware when the interface loses the arbitration of the bus to another master. An interrupt is generated if ITE = 1. It is cleared by software reading SR2 register or by hardware when the interface is disabled (PE = 0).<br>After an ARLO event the interface switches back automatically to Slave mode (M/SL = 0).<br>The SCL line is not held low while ARLO = 1.<br>0: No arbitration lost detected<br>1: Arbitration lost detected<br>*Note: In a Multimaster environment, when the interface is configured in Master Receive mode it does not perform arbitration during the reception of the Acknowledge bit. Mishandling of the ARLO bit from the I2CSR2 register may occur when a second master simultaneously requests the same data from the same slave and the $I^2C$ master does not acknowledge the data. The ARLO bit is then left at 0 instead of being set.* |
| 1 | BERR | *Bus error*<br>This bit is set by hardware when the interface detects a misplaced Start or Stop condition. An interrupt is generated if ITE = 1. It is cleared by software reading SR2 register or by hardware when the interface is disabled (PE = 0).<br>The SCL line is not held low while BERR = 1.<br>0: No misplaced Start or Stop condition<br>1: Misplaced Start or Stop condition<br>*Note: If a Bus Error occurs, a Stop or a repeated Start condition should be generated by the Master to re-synchronize communication, get the transmission acknowledged and the bus released for further communication.* |
| 0 | GCAL | *General Call (Slave mode)*<br>This bit is set by hardware when a general call address is detected on the bus while ENGC = 1. It is cleared by hardware detecting a Stop condition (STOPF = 1) or when the interface is disabled (PE = 0).<br>0: No general call address detected on bus<br>1: General call address detected on bus |

## 16.7.4 $I^2C$ clock control register (CCR)

CCR Reset value: 0000 0000 (00h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| FM/SM | CC[6:0] | | | | | | |
| RW | RW | | | | | | |

**Table 87. CCR register description**

| Bit | Name | Function |
|---|---|---|
| 7 | FM/SM | *Fast/Standard $I^2C$ mode*<br>This bit is set and cleared by software. It is not cleared when the interface is disabled (PE = 0).<br>0: Standard $I^2C$ mode<br>1: Fast $I^2C$ mode |

Obsolete Product(s) - Obsolete Product(s)

**Table 87. CCR register description (continued)**

| Bit | Name | Function |
|---|---|---|
| 6:0 | CC[6:0] | *7-bit clock divider*<br>These bits select the speed of the bus ($f_{SCL}$) depending on the $I^2C$ mode. They are not cleared when the interface is disabled (PE = 0).<br>Refer to *Section 19: Electrical characteristics* for the table of values.<br>*Note: The programmed $f_{SCL}$ assumes no load on SCL and SDA lines.* |

### 16.7.5 $I^2C$ data register (DR)

DR Reset value: 0000 0000 (00h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| D[7:0] | | | | | | | |
| RW | | | | | | | |

**Table 88. DR register description**

| Bit | Name | Function |
|---|---|---|
| 7:0 | D[7:0] | *8-bit Data Register*<br>These bits contain the byte to be received or transmitted on the bus.<br>**Transmitter mode:** Byte transmission start automatically when the software writes in the DR register.<br>**Receiver mode:** The first data byte is received automatically in the DR register using the least significant bit of the address.<br>Then, the following data bytes are received one by one after reading the DR register. |

### 16.7.6 $I^2C$ own address register (OAR1)

OAR1 Reset value: 0000 0000 (00h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| ADD7 | ADD6 | ADD5 | ADD4 | ADD3 | ADD2 | ADD1 | ADD0 |
| RW | RW | RW | RW | RW | RW | RW | RW |

Obsolete Product(s) Obsolete Product(s)

**Table 89. OAR1 register description**

| Bit | Name | Function | |
|---|---|---|---|
| | | 7-bit addressing mode | 10-bit addressing mode |
| 7:1 | ADD[7:1] | *Interface address*<br>These bits define the I²C bus address of the interface. They are not cleared when the interface is disabled (PE = 0). | Not applicable |
| 0 | ADD0 | *Address direction bit*<br>This bit is 'don't care', the interface acknowledges either 0 or 1. It is not cleared when the interface is disabled (PE = 0).<br>Address 01h is always ignored. | |
| 7:0 | ADD[7:0] | Not applicable | *Interface address*<br>These are the least significant bits of the I²C bus address of the interface. They are not cleared when the interface is disabled (PE = 0). |

## 16.7.7 I²C own address register (OAR2)

OAR2 Reset value: 0100 0000 (40h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| FR[1:0] | | Reserved | | | ADD[9:8] | | Reserved |
| RW | | - | | | RW | | - |

**Table 90. OAR2 register description**

| Bit | Name | Function |
|---|---|---|
| 7:6 | FR[1:0] | *Frequency bits*<br>These bits are set by software only when the interface is disabled (PE = 0). To configure the interface to I²C specified delays, select the value corresponding to the CPU frequency $f_{CPU}$.<br>00: $f_{CPU}$ < 6 MHz<br>01: $f_{CPU}$ = 6 to 8 MHz |
| 5:3 | - | Reserved |
| 2:1 | ADD[9:8] | *Interface address*<br>These are the most significant bits of the I²C bus address of the interface (10-bit mode only). They are not cleared when the interface is disabled (PE = 0). |
| 0 | - | Reserved |

Obsolete Product(s) - Obsolete Product(s)

**Table 91. $I^2C$ register map and reset values**

| Address (Hex.) | Register label | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0018h | I2CCR<br>Reset value | <br>0 | <br>0 | PE<br>0 | ENGC<br>0 | START<br>0 | ACK<br>0 | STOP<br>0 | ITE<br>0 |
| 0019h | I2CSR1<br>Reset value | EVF<br>0 | ADD10<br>0 | TRA<br>0 | BUSY<br>0 | BTF<br>0 | ADSL<br>0 | M/SL<br>0 | SB<br>0 |
| 001Ah | I2CSR2<br>Reset value | <br>0 | <br>0 | <br>0 | AF<br>0 | STOPF<br>0 | ARLO<br>0 | BERR<br>0 | GCAL<br>0 |
| 001Bh | I2CCCR<br>Reset value | FM/SM<br>0 | CC6<br>0 | CC5<br>0 | CC4<br>0 | CC3<br>0 | CC2<br>0 | CC1<br>0 | CC0<br>0 |
| 001Ch | I2COAR1<br>Reset value | ADD7<br>0 | ADD6<br>0 | ADD5<br>0 | ADD4<br>0 | ADD3<br>0 | ADD2<br>0 | ADD1<br>0 | ADD0<br>0 |
| 001Dh | I2COAR2<br>Reset value | FR1<br>0 | FR0<br>1 | <br>0 | <br>0 | <br>0 | ADD9<br>0 | ADD8<br>0 | <br>0 |
| 001Eh | I2CDR<br>Reset value | MSB<br>0 | <br>0 | <br>0 | <br>0 | <br>0 | <br>0 | <br>0 | LSB<br>0 |

Obsolete Product(s) - Obsolete Product(s)

# 17 10-bit A/D converter (ADC)

## 17.1 Introduction

The on-chip Analog to Digital Converter (ADC) peripheral is a 10-bit, successive approximation converter with internal sample and hold circuitry. This peripheral has up to 16 multiplexed analog input channels (refer to device pin out description) that allow the peripheral to convert the analog voltage levels from up to 16 different sources.

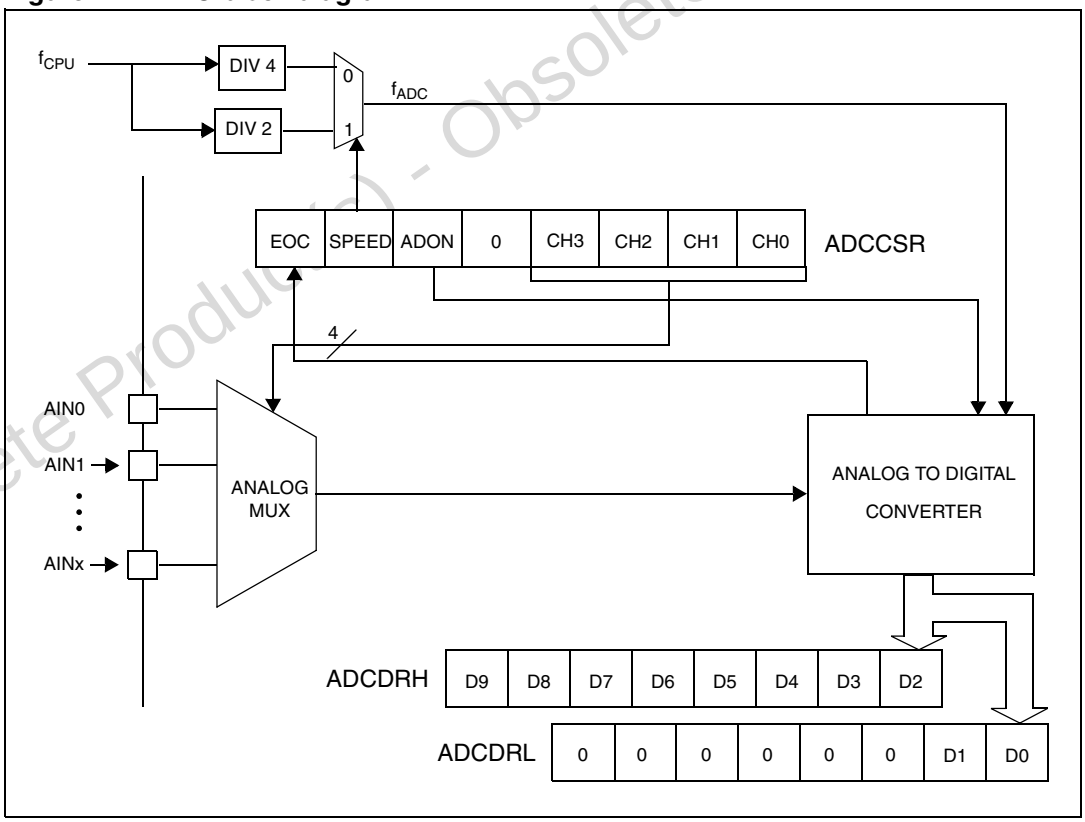
The result of the conversion is stored in a 10-bit data register. The A/D converter is controlled through a control/status register.

## 17.2 Main features

- 10-bit conversion
- Up to 16 channels with multiplexed input
- Linear successive approximation
- Data register (DR) which contains the results
- Conversion complete status flag
- On/off bit (to reduce consumption)

The block diagram is shown in *Figure 72*.

**Figure 72. ADC block diagram**




Obsolete Product(s) - Obsolete Product(s)

## 17.3 Functional description

The conversion is monotonic, meaning that the result never decreases if the analog input does not and never increases if the analog input does not.

If the input voltage ($V_{AIN}$) is greater than $V_{AREF}$ (high-level voltage reference) then the conversion result is FFh in the ADCDRH register and 03h in the ADCDRL register (without overflow indication).

If the input voltage ($V_{AIN}$) is lower than $V_{SSA}$ (low-level voltage reference) then the conversion result in the ADCDRH and ADCDRL registers is 00 00h.

The A/D converter is linear and the digital result of the conversion is stored in the ADCDRH and ADCDRL registers. The accuracy of the conversion is described in *Section 19: Electrical characteristics*.

$R_{AIN}$ is the maximum recommended impedance for an analog input signal. If the impedance is too high, this will result in a loss of accuracy due to leakage and sampling not being completed in the allotted time.

### 17.3.1 A/D converter configuration

The analog input ports must be configured as input, no pull-up, no interrupt. Refer to the *Section 9: I/O ports*. Using these pins as analog inputs does not affect the ability of the port to be read as a logic input.

In the ADCCSR register:

- Select the CS[3:0] bits to assign the analog channel to convert.

### 17.3.2 Starting the conversion

In the ADCCSR register:

- Set the ADON bit to enable the A/D converter and to start the conversion. From this time on, the ADC performs a continuous conversion of the selected channel.

When a conversion is complete:

- The EOC bit is set by hardware.
- The result is in the ADCDR registers.

A read to the ADCDRH or a write to any bit of the ADCCSR register resets the EOC bit.

To read the 10 bits, perform the following steps:

1. Poll the EOC bit.
2. Read the ADCDRL register.
3. Read the ADCDRH register. This clears EOC automatically.

*Note: The data is not latched, so both the low and the high data register must be read before the next conversion is complete, so it is recommended to disable interrupts while reading the conversion result.*

To read only 8 bits, perform the following steps:

1. Poll the EOC bit.
2. Read the ADCDRH register. This clears EOC automatically.

Obsolete Product(s) - Obsolete Product(s)

### 17.3.3 Changing the conversion channel

The application can change channels during conversion. When software modifies the CH[3:0] bits in the ADCCSR register, the current conversion is stopped, the EOC bit is cleared, and the A/D converter starts converting the newly selected channel.

## 17.4 Low power modes

*Note: The A/D converter may be disabled by resetting the ADON bit. This feature allows reduced power consumption when no conversion is needed and between single shot conversions.*

**Table 92. Effect of low power modes on ADC**

| Mode | Effect |
|---|---|
| Wait | No effect on A/D converter |
| Halt | A/D converter disabled.<br>After wake-up from Halt mode, the A/D converter requires a stabilization time $t_{STAB}$ (see *Section 19: Electrical characteristics*) before accurate conversions can be performed. |

## 17.5 Interrupts

None.

## 17.6 ADC registers

### 17.6.1 Control/status register (ADCCSR)

ADCCSR Reset value: 0000 0000 (00h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| EOC | SPEED | ADON | Reserved | CH[3:0] | | | |
| RO | RW | RW | - | RW | | | |

**Table 93. ADCCSR register description**

| Bit | Name | Function |
|---|---|---|
| 7 | EOC | *End of Conversion*<br>This bit is set by hardware. It is cleared by hardware when software reads the ADCDRH register or writes to any bit of the ADCCSR register.<br>0: Conversion is not complete<br>1: Conversion complete |
| 6 | SPEED | *ADC clock selection*<br>This bit is set and cleared by software.<br>0: $f_{ADC} = f_{CPU}/4$<br>1: $f_{ADC} = f_{CPU}/2$ |

Obsolete Product(s) - Obsolete Product(s)

**Table 93. ADCCSR register description (continued)**

| Bit | Name | Function |
|---|---|---|
| 5 | ADON | *A/D Converter on*<br>This bit is set and cleared by software.<br>0: Disable ADC and stop conversion<br>1: Enable ADC and start conversion |
| 4 | - | Reserved. Must be kept cleared |
| 3:0 | CH[3:0] | *Channel Selection*<br>These bits are set and cleared by software. They select the analog input to convert.<br>0000: Channel pin = AIN0<br>0001: Channel pin = AIN1<br>0010: Channel pin = AIN2<br>0011: Channel pin = AIN3<br>0100: Channel pin = AIN4<br>0101: Channel pin = AIN5<br>0110: Channel pin = AIN6<br>0111: Channel pin = AIN7<br>1000: Channel pin = AIN8<br>1001: Channel pin = AIN9<br>1010: Channel pin = AIN10<br>1011: Channel pin = AIN11<br>1100: Channel pin = AIN12<br>1101: Channel pin = AIN13<br>1110: Channel pin = AIN14<br>1111: Channel pin = AIN15<br>*Note: The number of channels is device dependent. Refer to the device pinout description.* |

### 17.6.2 Data register (ADCDRH)

ADCDRH Reset value: 0000 0000 (00h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| D[9:2] | | | | | | | |
| RO | | | | | | | |

**Table 94. ADCDRH register description**

| Bit | Name | Function |
|---|---|---|
| 7:0 | D[9:2] | *MSB of Converted Analog Value* |

Obsolete Product(s) - Obsolete Product(s)

### 17.6.3 Data register (ADCDRL)

ADCDRL Reset value: 0000 0000 (00h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | D[1:0] | |
| - | | | | | | RO | |

**Table 95. ADCDRL register description**

| Bit | Name | Function |
|---|---|---|
| 7:2 | - | Reserved. Forced by hardware to 0. |
| 1:0 | D[1:0] | *LSB of Converted Analog Value* |

### 17.6.4 ADC register map and reset values

**Table 96. ADC register map and reset values**

| Address (Hex.) | Register label | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0070h | ADCCSR<br>Reset value | EOC<br>0 | SPEED<br>0 | ADON<br>0 | <br>0 | CH3<br>0 | CH2<br>0 | CH1<br>0 | CH0<br>0 |
| 0071h | ADCDRH<br>Reset value | D9<br>0 | D8<br>0 | D7<br>0 | D6<br>0 | D5<br>0 | D4<br>0 | D3<br>0 | D2<br>0 |
| 0072h | ADCDRL<br>Reset value | <br>0 | <br>0 | <br>0 | <br>0 | <br>0 | <br>0 | D1<br>0 | D0<br>0 |

Obsolete Product(s) - Obsolete Product(s)

# 18 Instruction set

## 18.1 CPU addressing modes

The CPU features 17 different addressing modes which can be classified in seven main groups as listed in the following table:

**Table 97. Addressing modes**

| Group | 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 CPU instruction set is designed to minimize the number of bytes required per instruction: To do so, most of the addressing modes may be divided 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 (0000h - 00FFh 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 98. CPU addressing mode overview**

| Mode | | | Syntax | Destination | 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 |
| Short | Direct | Indexed | ld A,($10,X) | 00..1FE | | | + 1 |
| Long | Direct | Indexed | ld A,($1000,X) | 0000..FFFF | | | + 2 |
| Short | Indirect | | ld A,[$10] | 00..FF | 00..FF | byte | + 2 |
| Long | Indirect | | ld A,[$10.w] | 0000..FFFF | 00..FF | word | + 2 |
| Short | Indirect | Indexed | ld A,([$10],X) | 00..1FE | 00..FF | byte | + 2 |

Obsolete Product(s) - Obsolete Product(s)

**Table 98. CPU addressing mode overview (continued)**

| Mode | | | Syntax | Destination | Pointer address (Hex.) | Pointer size (Hex.) | Length (bytes) |
|---|---|---|---|---|---|---|---|
| Long | Indirect | Indexed | ld A,([$10.w],X) | 0000..FFFF | 00..FF | word | + 2 |
| Relative | Direct | | jrne loop | PC+/-127 | | | + 1 |
| Relative | Indirect | | jrne [$10] | PC+/-127 | 00..FF | byte | + 2 |
| Bit | Direct | | bset $10,#7 | 00..FF | | | + 1 |
| Bit | Indirect | | bset [$10],#7 | 00..FF | 00..FF | byte | + 2 |
| Bit | Direct | Relative | btjt $10,#7,skip | 00..FF | | | + 2 |
| Bit | Indirect | Relative | btjt [$10],#7,skip | 00..FF | 00..FF | byte | + 3 |

### 18.1.1 Inherent

All Inherent instructions consist of a single byte. The opcode fully specifies all the required information for the CPU to process the operation.

**Table 99. Inherent instructions**

| Instruction | Function |
|---|---|
| NOP | No operation |
| TRAP | S/W Interrupt |
| WFI | Wait For Interrupt (Low Power Mode) |
| HALT | Halt Oscillator (Lowest Power Mode) |
| RET | Sub-routine Return |
| IRET | Interrupt Sub-routine Return |
| SIM | Set Interrupt Mask (level 3) |
| RIM | Reset Interrupt Mask (level 0) |
| SCF | Set Carry Flag |
| RCF | Reset Carry Flag |
| RSP | Reset Stack Pointer |
| LD | Load |
| CLR | Clear |
| PUSH/POP | Push/Pop to/from the stack |
| INC/DEC | Increment/Decrement |
| TNZ | Test Negative or Zero |
| CPL, NEG | 1 or 2 Complement |
| MUL | Byte Multiplication |
| SLL, SRL, SRA, RLC, RRC | Shift and Rotate Operations |
| SWAP | Swap Nibbles |

Obsolete Product(s) - Obsolete Product(s)

### 18.1.2 Immediate

Immediate instructions have 2 bytes. The first byte contains the opcode and the second byte contains the operand value.

**Table 100. Immediate instructions**

| Instruction | Function |
|---|---|
| LD | Load |
| CP | Compare |
| BCP | Bit Compare |
| AND, OR, XOR | Logical Operations |
| ADC, ADD, SUB, SBC | Arithmetic Operations |

### 18.1.3 Direct

In Direct instructions, the operands are referenced by their memory address.

The direct addressing mode consists of two submodes:

#### Direct (short)

The address is a byte, thus requires only one byte after the opcode, but only allows 00 - FF addressing space.

#### Direct (long)

The address is a word, thus allowing 64 Kbyte addressing space, but requires 2 bytes after the opcode.

### 18.1.4 Indexed (no offset, short, long)

In this mode, the operand is referenced by its memory address, which is defined by the unsigned addition of an index register (X or Y) with an offset.

The indexed addressing mode consists of three submodes:

#### Indexed (no offset)

There is no offset, (no extra byte after the opcode), and allows 00 - FF addressing space.

#### Indexed (short)

The offset is a byte, thus requires only one byte after the opcode and allows 00 - 1FE addressing space.

#### Indexed (long)

The offset is a word, thus allowing 64 Kbyte addressing space and requires 2 bytes after the opcode.

Obsolete Product(s) - Obsolete Product(s)

### 18.1.5 Indirect (short, long)

The required data byte to do the operation is found by its memory address, located in memory (pointer).

The pointer address follows the opcode. The indirect addressing mode consists of two submodes:

#### Indirect (short)

The pointer address is a byte, the pointer size is a byte, thus allowing 00 - FF addressing space, and requires 1 byte after the opcode.

#### Indirect (long)

The pointer address is a byte, the pointer size is a word, thus allowing 64 Kbyte addressing space, and requires 1 byte after the opcode.

### 18.1.6 Indirect indexed (short, long)

This is a combination of indirect and short indexed addressing modes. The operand is referenced by its memory address, which is defined by the unsigned addition of an index register value (X or Y) with a pointer value located in memory. The pointer address follows the opcode.

The indirect indexed addressing mode consists of two submodes:

#### Indirect indexed (short)

The pointer address is a byte, the pointer size is a byte, thus allowing 00 - 1FE addressing space, and requires 1 byte after the opcode.

#### Indirect indexed (long)

The pointer address is a byte, the pointer size is a word, thus allowing 64 Kbyte addressing space, and requires 1 byte after the opcode.

**Table 101. Instructions supporting direct, indexed, indirect, and indirect indexed addressing modes**

| Type | Instruction | Function |
|---|---|---|
| Long and short instructions | LD | Load |
| | CP | Compare |
| | AND, OR, XOR | Logical operations |
| | ADC, ADD, SUB, SBC | Arithmetic Additions/Subtractions operations |
| | BCP | Bit Compare |

Obsolete Product(s) - Obsolete Product(s)

**Table 101. Instructions supporting direct, indexed, indirect, and indirect indexed addressing modes (continued)**

| Type | Instruction | Function |
|---|---|---|
| Short instructions only | CLR | Clear |
| | INC, DEC | Increment/Decrement |
| | TNZ | Test Negative or Zero |
| | CPL, NEG | 1 or 2 Complement |
| | BSET, BRES | Bit Operations |
| | BTJT, BTJF | Bit Test and Jump Operations |
| | SLL, SRL, SRA, RLC, RRC | Shift and Rotate Operations |
| | SWAP | Swap Nibbles |
| | CALL, JP | Call or Jump subroutine |

### 18.1.7 Relative (direct, indirect)

This addressing mode is used to modify the PC register value, by adding an 8-bit signed offset to it.

**Table 102. Available relative direct/indirect instructions**

| Instruction | Function |
|---|---|
| JRxx | Conditional Jump |
| CALLR | Call Relative |

The relative addressing mode consists of two submodes:

#### Relative (direct)

The offset is following the opcode.

#### Relative (indirect)

The offset is defined in memory, which address follows the opcode.

## 18.2 Instruction groups

The ST7 family devices use an Instruction Set consisting of 63 instructions. The instructions may be subdivided into 13 main groups as illustrated in the following table:

**Table 103. Instruction groups**

| Group | Instructions | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Load and Transfer | LD | CLR | | | | | | |
| Stack operation | PUSH | POP | RSP | | | | | |
| Increment/Decrement | INC | DEC | | | | | | |
| Compare and Tests | CP | TNZ | BCP | | | | | |

Obsolete Product(s) - Obsolete Product(s)

**Table 103. Instruction groups (continued)**

| Group | Instructions | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Logical operations | AND | OR | XOR | CPL | NEG | | | |
| Bit Operation | BSET | BRES | | | | | | |
| Conditional Bit Test and Branch | BTJT | BTJF | | | | | | |
| Arithmetic operations | ADC | ADD | SUB | SBC | MUL | | | |
| Shift and Rotates | SLL | SRL | SRA | RLC | RRC | SWAP | SLA | |
| Unconditional Jump or Call | JRA | JRT | JRF | JP | CALL | CALLR | NOP | RET |
| Conditional Branch | JRxx | | | | | | | |
| Interruption management | TRAP | WFI | HALT | IRET | | | | |
| Condition Code Flag modification | SIM | RIM | SCF | RCF | | | | |

### 18.2.1 Using a prebyte

The instructions are described with one to four opcodes.

In order to extend the number of available opcodes for an 8-bit CPU (256 opcodes), three different prebyte opcodes are defined. These prebytes modify the meaning of the instruction they precede.

The whole instruction becomes:

- PC - 2 End of previous instruction
- PC - 1 Prebyte
- PC Opcode
- PC + 1 Additional word (0 to 2) according to the number of bytes required to compute the effective address

These prebytes enable instruction in Y as well as indirect addressing modes to be implemented. They precede the opcode of the instruction in X or the instruction using direct addressing mode. The prebytes are:

- PDY 90 Replace an X based instruction using immediate, direct, indexed, or inherent addressing mode by a Y one.
- PIX 92 Replace an instruction using direct, direct bit, or direct relative addressing mode to an instruction using the corresponding indirect addressing mode.
  It also changes an instruction using X indexed addressing mode to an instruction using indirect X indexed addressing mode.
- PIY 91 Replace an instruction using X indirect indexed addressing mode by a Y one.

Obsolete Product(s) Obsolete Product(s)

**Table 104. Instruction set overview**

| Mnemo | Description | Function/Example | Dst | Src | I1 | H | I0 | N | Z | C |
|---|---|---|---|---|---|---|---|---|---|---|
| ADC | Add with Carry | A = A + M + C | A | M | | H | | N | Z | C |
| ADD | Addition | A = A + M | A | M | | H | | N | Z | C |
| AND | Logical And | A = A . M | A | M | | | | N | Z | |
| BCP | Bit compare A, Memory | tst (A . M) | A | M | | | | N | Z | |
| BRES | Bit Reset | bres Byte, #3 | M | | | | | | | |
| BSET | Bit Set | bset Byte, #3 | M | | | | | | | |
| BTJF | Jump if bit is false (0) | btjf Byte, #3, Jmp1 | M | | | | | | | C |
| BTJT | Jump if bit is true (1) | btjt Byte, #3, Jmp1 | M | | | | | | | C |
| CALL | Call subroutine | | | | | | | | | |
| CALLR | Call subroutine relative | | | | | | | | | |
| CLR | Clear | | reg, M | | | | | 0 | 1 | |
| CP | Arithmetic Compare | tst(Reg - M) | reg | M | | | | N | Z | C |
| CPL | One Complement | A = FFH-A | reg, M | | | | | N | Z | 1 |
| DEC | Decrement | dec Y | reg, M | | | | | N | Z | |
| HALT | Halt | | | | 1 | | 0 | | | |
| IRET | Interrupt routine return | Pop CC, A, X, PC | | | I1 | H | I0 | N | Z | C |
| INC | Increment | inc X | reg, M | | | | | N | Z | |
| JP | Absolute Jump | jp [TBL.w] | | | | | | | | |
| JRA | Jump relative always | | | | | | | | | |
| JRT | Jump relative | | | | | | | | | |
| JRF | Never jump | jrf * | | | | | | | | |
| JRIH | Jump if ext. INT pin = 1 | (ext. INT pin high) | | | | | | | | |
| JRIL | Jump if ext. INT pin = 0 | (ext. INT pin low) | | | | | | | | |
| JRH | Jump if H = 1 | H = 1 ? | | | | | | | | |
| JRNH | Jump if H = 0 | H = 0 ? | | | | | | | | |
| JRM | Jump if I1:0 = 11 | I1:0 = 11 ? | | | | | | | | |
| JRNM | Jump if I1:0 <> 11 | I1:0 <> 11 ? | | | | | | | | |
| JRMI | Jump if N = 1 (minus) | N = 1 ? | | | | | | | | |
| JRPL | Jump if N = 0 (plus) | N = 0 ? | | | | | | | | |
| JREQ | Jump if Z = 1 (equal) | Z = 1 ? | | | | | | | | |
| JRNE | Jump if Z = 0 (not equal) | Z = 0 ? | | | | | | | | |
| JRC | Jump if C = 1 | C = 1 ? | | | | | | | | |
| JRNC | Jump if C = 0 | C = 0 ? | | | | | | | | |
| JRULT | Jump if C = 1 | Unsigned < | | | | | | | | |
| JRUGE | Jump if C = 0 | Jmp if unsigned >= | | | | | | | | |

Obsolete Product(s) - Obsolete Product(s)

**Table 104. Instruction set overview (continued)**

| Mnemo | Description | Function/Example | Dst | Src | I1 | H | I0 | N | Z | C |
|---|---|---|---|---|---|---|---|---|---|---|
| JRUGT | Jump if (C + Z = 0) | Unsigned > | | | | | | | | |
| JRULE | Jump if (C + Z = 1) | Unsigned <= | | | | | | | | |
| LD | Load | dst <= src | reg, M | M, reg | | | | N | Z | |
| MUL | Multiply | X,A = X * A | A, X, Y | X, Y, A | | 0 | | | | 0 |
| NEG | Negate (2's compl) | neg $10 | reg, M | | | | | N | Z | C |
| NOP | No Operation | | | | | | | | | |
| OR | OR operation | A = A + M | A | M | | | | N | Z | |
| POP | Pop from the Stack | pop reg | reg | M | | | | | | |
| | | pop CC | CC | M | I1 | H | I0 | N | Z | C |
| PUSH | Push onto the Stack | push Y | M | reg, CC | | | | | | |
| RCF | Reset carry flag | C = 0 | | | | | | | | 0 |
| RET | Subroutine Return | | | | | | | | | |
| RIM | Enable Interrupts | I1:0 = 10 (level 0) | | | 1 | | 0 | | | |
| RLC | Rotate left true C | C <= A <= C | reg, M | | | | | N | Z | C |
| RRC | Rotate right true C | C => A => C | reg, M | | | | | N | Z | C |
| RSP | Reset Stack Pointer | S = Max allowed | | | | | | | | |
| SBC | Subtract with Carry | A = A - M - C | A | M | | | | N | Z | C |
| SCF | Set carry flag | C = 1 | | | | | | | | 1 |
| SIM | Disable Interrupts | I1:0 = 11 (level 3) | | | 1 | | 1 | | | |
| SLA | Shift left Arithmetic | C <= A <= 0 | reg, M | | | | | N | Z | C |
| SLL | Shift left Logic | C <= A <= 0 | reg, M | | | | | N | Z | C |
| SRL | Shift right Logic | 0 => A => C | reg, M | | | | | 0 | Z | C |
| SRA | Shift right Arithmetic | A7 => A => C | reg, M | | | | | N | Z | C |
| SUB | Subtraction | A = A - M | A | M | | | | N | Z | C |
| SWAP | SWAP nibbles | A7-A4 <=> A3-A0 | reg, M | | | | | N | Z | |
| TNZ | Test for Neg & Zero | tnz lbl1 | | | | | | N | Z | |
| TRAP | S/W trap | S/W interrupt | | | 1 | | 1 | | | |
| WFI | Wait for Interrupt | | | | 1 | | 0 | | | |
| XOR | Exclusive OR | A = A XOR M | A | M | | | | N | Z | |

Obsolete Product(s) - Obsolete Product(s)

# 19 Electrical characteristics

## 19.1 Parameter conditions

Unless otherwise specified, all voltages are referred to $V_{SS}$.

### 19.1.1 Minimum and maximum values

Unless otherwise specified the minimum and maximum values are guaranteed in the worst conditions of ambient temperature, supply voltage and frequencies by tests in production on 100% of the devices with an ambient temperature at $T_A$ = 25°C and $T_A$ = $T_{Amax}$ (given by the selected temperature range).

Data based on characterization results, design simulation and/or technology characteristics are indicated in the table footnotes and are not tested in production. Based on characterization, the minimum and maximum values refer to sample tests and represent the mean value plus or minus three times the standard deviation (mean ±3Σ).

### 19.1.2 Typical values

Unless otherwise specified, typical data is based on $T_A$ = 25°C, $V_{DD}$ = 5V. The typical values are given only as design guidelines and are not tested.

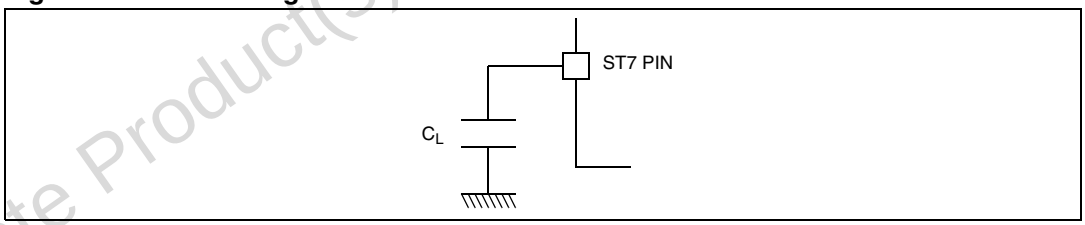
### 19.1.3 Typical curves

Unless otherwise specified, all typical curves are given only as design guidelines and are not tested.

### 19.1.4 Loading capacitor

The loading conditions used for pin parameter measurement are shown in *Figure 73*.

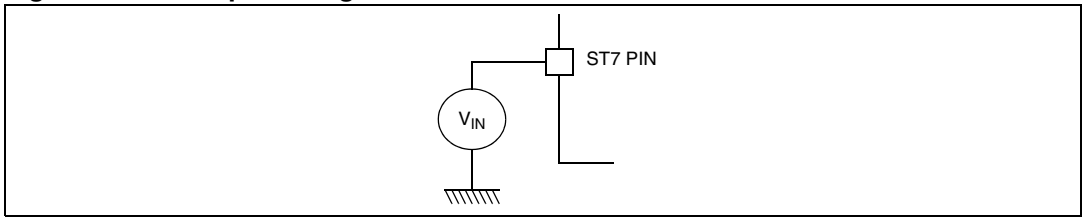
**Figure 73. Pin loading conditions**




### 19.1.5 Pin input voltage

The input voltage measurement on a pin of the device is described in *Figure 74*.

**Figure 74. Pin input voltage**




Obsolete Product(s) - Obsolete Product(s)

## 19.2 Absolute maximum ratings

Stresses above those listed as “absolute maximum ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device under these conditions is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

### 19.2.1 Voltage characteristics

**Table 105. Voltage characteristics**

| Symbol | Ratings | Maximum value | Unit |
|---|---|---|---|
| $V_{DD}$ - $V_{SS}$ | Supply voltage | 6.5 | V |
| $V_{PP}$ - $V_{SS}$ | Programming voltage | 13 | |
| $V_{IN}$ (1) | Input voltage on true open-drain pin | $V_{SS}$ - 0.3 to 6.5 | |
| | Input voltage on any other pin | $V_{SS}$ - 0.3 to $V_{DD}$ + 0.3 | |
| $\lvert\Delta V_{DDx}\rvert$ and $\lvert\Delta V_{SSx}\rvert$ | Variations between different digital power pins | 50 | mV |
| $\lvert V_{SSA} - V_{SSx}\rvert$ | Variations between digital and analog ground pins | 50 | |
| $V_{ESD(HBM)}$ | Electrostatic discharge voltage (Human Body Model) | See *Section 19.7.3 on page 204*. | |
| $V_{ESD(MM)}$ | Electrostatic discharge voltage (Machine Model) | | |

1. Directly connecting the $\overline{\text{RESET}}$ and I/O pins to $V_{DD}$ or $V_{SS}$ could damage the device if an unintentional internal reset is generated or an unexpected change of the I/O configuration occurs (for example, due to a corrupted program counter). To guarantee safe operation, this connection has to be done through a pull-up or pull-down resistor (typical: 4.7kΩ for $\overline{\text{RESET}}$, 10kΩ for I/Os). For the same reason, unused I/O pins must not be directly tied to $V_{DD}$ or $V_{SS}$.

Obsolete Product(s) - Obsolete Product(s)

### 19.2.2 Current characteristics

**Table 106. Current characteristics**

| Symbol | Ratings | | Maximum value | Unit |
|---|---|---|---|---|
| $I_{VDD}$ / $I_{VSS}$ (1) | Total current into $V_{DD}$ power lines (source) and $V_{SS}$ ground lines (sink) | 32-pin devices | 75 | mA |
| | | 44-/64-pin devices | 150 | |
| $I_{IO}$ | Output current sunk by any standard I/O and control pin | | 20 | mA |
| | Output current sunk by any high sink I/O pin | | 40 | |
| | Output current source by any I/Os and control pin | | - 25 | |
| $I_{INJ(PIN)}$ (2)(3) | Injected current on $V_{PP}$ pin | | ± 5 | |
| | Injected current on $\overline{\text{RESET}}$ pin | | ± 5 | |
| | Injected current on OSC1 and OSC2 pins | | ± 5 | |
| | Injected current on PB0 pin | | + 5 | |
| | Injected current on any other pin (4)(5) | | ± 5 | |
| $\Sigma I_{INJ(PIN)}$ (2) | Total injected current (sum of all I/O and control pins) (4) | | ± 25 | |

1. All power ($V_{DD}$) and ground ($V_{SS}$) lines must always be connected to the external supply.
2. $I_{INJ(PIN)}$ must never be exceeded. This is implicitly ensured if $V_{IN}$ maximum is respected. If $V_{IN}$ maximum cannot be respected, the injection current must be limited externally to the $I_{INJ(PIN)}$ value. A positive injection is induced by $V_{IN} > V_{DD}$ while a negative injection is induced by $V_{IN} < V_{SS}$. For true open-drain pads, there is no positive injection current, and the corresponding $V_{IN}$ maximum must always be respected.
3. Negative injection may disturb the analog performance of the device. See *Note 1* in *Table 138: ADC accuracy on page 222*.
4. When several inputs are submitted to a current injection, the maximum $\Sigma I_{INJ(PIN)}$ is the absolute sum of the positive and negative injected currents (instantaneous values). These results are based on characterization with $\Sigma I_{INJ(PIN)}$ maximum current injection on four I/O port pins of the device.
5. True open-drain I/O port pins do not accept positive injection.

### 19.2.3 Thermal characteristics

**Table 107. Thermal characteristics**

| Symbol | Ratings | Value | Unit |
|---|---|---|---|
| $T_{STG}$ | Storage temperature range | -65 to +150 | °C |
| $T_J$ | Maximum junction temperature (see *Section 20.2: Thermal characteristics*) | | |

Obsolete Product(s) - Obsolete Product(s)

## 19.3 Operating conditions

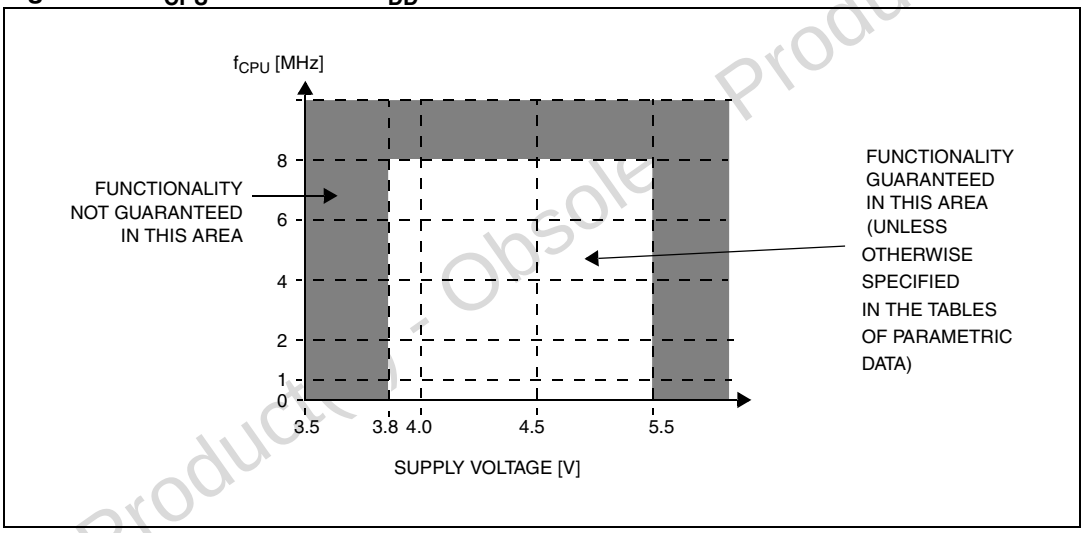
### 19.3.1 General operating conditions

**Table 108. General operating conditions**

| Symbol | Parameter | Conditions | Min | Max | Unit |
|---|---|---|---|---|---|
| $f_{CPU}$ | Internal clock frequency | | 0 | 8 | MHz |
| $V_{DD}$ | Standard voltage range (except Flash Write/Erase) | | 3.8 | 5.5 | V |
| | Operating voltage for Flash Write/Erase | $V_{PP}$ = 11.4 to 12.6V | 4.5 | 5.5 | |
| $T_A$ | Ambient temperature range | A suffix version | -40 | 85 | °C |
| | | B suffix version | | 105 | |
| | | C suffix version | | 125 | |

*Note: Some temperature ranges are only available with a specific package and memory size. Refer to Section 21: Device configuration and ordering information.*

**Figure 75. $f_{CPU}$ max versus $V_{DD}$**




Obsolete Product(s) - Obsolete Product(s)

### 19.3.2 Operating conditions with low voltage detector (LVD)

Subject to general operating conditions for $V_{DD}$, $f_{CPU}$, and $T_A$.

**Table 109. Operating conditions with low voltage detector (LVD)**

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $V_{IT+(LVD)}$ | Reset release threshold ($V_{DD}$ rise) | VD level = High in option byte | 4.0(1) | 4.2 | 4.5 | V |
| | | VD level = Med. in option byte(2) | 3.55(1) | 3.75 | 4.0(1) | |
| | | VD level = Low in option byte(2) | 2.95(1) | 3.15 | 3.35(1) | |
| $V_{IT-(LVD)}$ | Reset generation threshold ($V_{DD}$ fall) | VD level = High in option byte | 3.8 | 4.0 | 4.25(1) | |
| | | VD level = Med. in option byte(2) | 3.35(1) | 3.55 | 3.75(1) | |
| | | VD level = Low in option byte(2) | 2.8(1) | 3.0 | 3.15(1) | |
| $V_{hys(LVD)}$ | LVD voltage threshold hysteresis | $V_{IT+(LVD)}$-$V_{IT-(LVD)}$ | | 200 | | mV |
| $Vt_{POR}$ | $V_{DD}$ rise time(2) | LVD enabled | 6µs/V | | 100ms/V | - |
| $t_{g(VDD)}$ | $V_{DD}$ glitches filtered (not detected) by LVD(3) | | | | 40 | ns |

1. Data based on characterization results
2. Data based on characterization results, not tested in production
3. If the medium or low thresholds are selected, the detection may occur outside the specified operating voltage range. Below 3.8V, device operation is not guaranteed.

### 19.3.3 Auxiliary voltage detector (AVD) thresholds

Subject to general operating conditions for $V_{DD}$, $f_{CPU}$, and $T_A$.

**Table 110. Auxiliary voltage detector (AVD) thresholds**

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $V_{IT+(AVD)}$ | 1⇒0 AVDF flag toggle threshold ($V_{DD}$ rise) | VD level = High in option byte | 4.4(1) | 4.6 | 4.9(1) | V |
| | | VD level = Med. in option byte | 3.95(1) | 4.15 | 4.4(1) | |
| | | VD level = Low in option byte | 3.4(1) | 3.6 | 3.8(1) | |
| $V_{IT-(AVD)}$ | 0⇒1 AVDF flag toggle threshold ($V_{DD}$ fall) | VD level = High in option byte | 4.2(1) | 4.4 | 4.65(1) | |
| | | VD level = Med. in option byte | 3.75(1) | 4.0 | 4.2(1) | |
| | | VD level = Low in option byte | 3.2(1) | 3.4 | 3.6(1) | |
| $V_{hys(AVD)}$ | AVD voltage threshold hysteresis | $V_{IT+(AVD)}$-$V_{IT-(AVD)}$ | | 200 | | mV |
| $\Delta V_{IT-}$ | Voltage drop between AVD flag set and LVD reset activated | $V_{IT-(AVD)}$-$V_{IT-(LVD)}$ | | 450 | | |

1. Data based on characterization results

Obsolete Product(s) - Obsolete Product(s)

### 19.3.4 External voltage detector (EVD) thresholds

Subject to general operating conditions for $V_{DD}$, $f_{CPU}$, and $T_A$.

**Table 111. External voltage detector (EVD) thresholds**

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $V_{IT+(EVD)}$ | 1⇒0 AVDF flag toggle threshold ($V_{DD}$ rise)(1) | | 1.15 | 1.26 | 1.35 | V |
| $V_{IT-(EVD)}$ | 0⇒1 AVDF flag toggle threshold ($V_{DD}$ fall)(1) | | 1.1 | 1.2 | 1.3 | |
| $V_{hys(EVD)}$ | EVD voltage threshold hysteresis | $V_{IT+(EVD)}$-$V_{IT-(EVD)}$ | | 200 | | mV |

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

Obsolete Product(s) - Obsolete Product(s)

## 19.4 Supply current characteristics

The following current consumption specified for the ST7 functional operating modes over temperature range does not take into account the clock source current consumption. To obtain the total device consumption, the two current values must be added (except for Halt mode, for which the clock is stopped).

### 19.4.1 Current consumption

**Table 112. Current consumption**

| Symbol | Parameter | Conditions | Typ | Max(1) | Unit |
|---|---|---|---|---|---|
| $I_{DD}$ | Supply current in Run mode(2) | $f_{OSC}$ = 2 MHz, $f_{CPU}$ = 1 MHz<br>$f_{OSC}$ = 4 MHz, $f_{CPU}$ = 2 MHz<br>$f_{OSC}$ = 8 MHz, $f_{CPU}$ = 4 MHz<br>$f_{OSC}$ = 16 MHz, $f_{CPU}$ = 8 MHz | 1.3<br>2.0<br>3.6<br>7.1 | 3.0<br>5.0<br>8.0<br>15.0 | mA |
| | Supply current in Slow mode(2) | $f_{OSC}$ = 2 MHz, $f_{CPU}$ = 62.5 kHz<br>$f_{OSC}$ = 4 MHz, $f_{CPU}$ = 125 kHz<br>$f_{OSC}$ = 8 MHz, $f_{CPU}$ = 250 kHz<br>$f_{OSC}$ = 16 MHz, $f_{CPU}$ = 500 kHz | 600<br>700<br>800<br>1100 | 2700<br>3000<br>3600<br>4000 | µA |
| | Supply current in Wait mode(2) | $f_{OSC}$ = 2 MHz, $f_{CPU}$ = 1 MHz<br>$f_{OSC}$ = 4 MHz, $f_{CPU}$ = 2 MHz<br>$f_{OSC}$ = 8 MHz, $f_{CPU}$ = 4 MHz<br>$f_{OSC}$ = 16 MHz, $f_{CPU}$ = 8 MHz | 0.8<br>1.2<br>2.0<br>3.5 | 3.0<br>4.0<br>5.0<br>7.0 | mA |
| | Supply current in Slow Wait mode(2) | $f_{OSC}$ = 2 MHz, $f_{CPU}$ = 62.5 kHz<br>$f_{OSC}$ = 4 MHz, $f_{CPU}$ = 125 kHz<br>$f_{OSC}$ = 8 MHz, $f_{CPU}$ = 250 kHz<br>$f_{OSC}$ = 16 MHz, $f_{CPU}$ = 500 kHz | 580<br>650<br>770<br>1050 | 1200<br>1300<br>1800<br>2000 | µA |
| | Supply current in Halt mode(3) | -40°C ≤ $T_A$ ≤ +85°C | <1 | 10 | µA |
| | | -40°C ≤ $T_A$ ≤ +125°C | 5 | 50 | |
| | Supply current in Active Halt mode(4) | $f_{OSC}$ = 2 MHz<br>$f_{OSC}$ = 4 MHz<br>$f_{OSC}$ = 8 MHz<br>$f_{OSC}$ = 16 MHz | 415<br>430<br>460<br>550 | 525<br>550<br>600<br>700 | µA |

1. Data based on characterization results, tested in production at $V_{DD}$ max. and $f_{CPU}$ max.
2. Measurements are done in the following conditions:
   - Program executed from RAM, CPU running with RAM access
   - All I/O pins in input mode with a static value at $V_{DD}$ or $V_{SS}$ (no load)
   - All peripherals in reset state
   - CSS and LVD disabled
   - Clock input (OSC1) driven by external square wave
   - In Slow and Slow Wait mode, $f_{CPU}$ is based on $f_{OSC}$ divided by 32
   - To obtain the total current consumption of the device, add the clock source (*Section 19.4.2*) and the peripheral power consumption (*Section 19.4.3*).
3. All I/O pins in push-pull 0 mode (when applicable) with a static value at $V_{DD}$ or VSS (no load), LVD disabled. Data based on characterization results, tested in production at $V_{DD}$ max. and $f_{CPU}$ max.
4. Data based on characterization results, not tested in production. All I/O pins in push-pull 0 mode (when applicable) with a static value at $V_{DD}$ or $V_{SS}$ (no load); clock input (OSC1) driven by external square wave, LVD disabled. To obtain the total current consumption of the device, add the clock source consumption (*Section 19.4.2*).

Obsolete product(s) - Obsolete Product(s)

### 19.4.2 Supply and clock managers

The previous current consumption specified for the ST7 functional operating modes over temperature range does not take into account the clock source current consumption. To obtain the total device consumption, the two current values must be added (except for Halt mode).

**Table 113. Oscillators, CSS, PLL and LVD current consumption**

| Symbol | Parameter | Conditions | Typ | Max | Unit |
|---|---|---|---|---|---|
| $I_{DD(RCINT)}$ | Supply current of internal RC oscillator | | 625 | | µA |
| $I_{DD(RES)}$ | Supply current of resonator oscillator(1)(2) | | see *section 19.5.3 on page 199* | | |
| $I_{DD(CSS)}$ | Clock security system supply current | $V_{DD}$ = 5V | 250 | | |
| $I_{DD(PLL)}$ | PLL supply current | | 360 | | |
| $I_{DD(LVD)}$ | LVD supply current | | 150 | 300 | |

1. Data based on characterization results done with the external components specified in *Section 19.5.3*, not tested in production
2. As the oscillator is based on a current source, the consumption does not depend on the voltage.

Obsolete Product(s) - Obsolete Product(s)

### 19.4.3 On-chip peripherals

Measured on LQFP64 generic board $T_A$ = 25°C, $f_{CPU}$ = 4 MHz.

**Table 114. On-chip peripherals current consumption**

| Symbol | Parameter | Conditions | Typ | Unit |
|---|---|---|---|---|
| $I_{DD(TIM)}$ | 16-bit timer supply current(1) | $V_{DD}$ = 5.0V | 50 | µA |
| $I_{DD(ART)}$ | ART PWM supply current(2) | $V_{DD}$ = 5.0V | 75 | µA |
| $I_{DD(SPI)}$ | SPI supply current(3) | $V_{DD}$ = 5.0V | 400 | µA |
| $I_{DD(SCI)}$ | SCI supply current(4) | | | |
| $I_{DD(I2C)}$ | I2C supply current(5) | $V_{DD}$ = 5.0V | 175 | µA |
| $I_{DD(ADC)}$ | ADC supply current when converting(6) | $V_{DD}$ = 5.0V | 400 | µA |

1. Data based on a differential $I_{DD}$ measurement between reset configuration (timer counter running at $f_{CPU}/4$) and timer counter stopped (only TIMD bit set). Data valid for one timer.
2. Data based on a differential $I_{DD}$ measurement between reset configuration (timer stopped) and timer counter enabled (only TCE bit set).
3. Data based on a differential $I_{DD}$ measurement between reset configuration (SPI disabled) and a permanent SPI master communication at maximum speed (data sent equal to 55h). This measurement includes the pad toggling consumption.
4. Data based on a differential $I_{DD}$ measurement between SCI low power state (SCID = 1) and a permanent SCI data transmit sequence.
5. Data based on a differential $I_{DD}$ measurement between reset configuration (I2C disabled) and a permanent I2C master communication at 100 kHz (data sent equal to 55h). This measurement includes the pad toggling consumption (27k ohm external pull-up on clock and data lines).
6. Data based on a differential $I_{DD}$ measurement between reset configuration and continuous A/D conversions.

Obsolete Product(s) - Obsolete Product(s)

## 19.5 Clock and timing characteristics

Subject to general operating conditions for $V_{DD}$, $f_{CPU}$, and $T_A$.

### 19.5.1 General timings

**Table 115. General timings**

| Symbol | Parameter | Conditions | Min | Typ(1) | Max | Unit |
|---|---|---|---|---|---|---|
| $t_{c(INST)}$ | Instruction cycle time | | 2 | 3 | 12 | $t_{CPU}$ |
| | | $f_{CPU}$ = 8 MHz | 250 | 375 | 1500 | ns |
| $t_{v(IT)}$ | Interrupt reaction time(2) $t_{v(IT)} = \Delta t_{c(INST)} + 10$ | | 10 | | 22 | $t_{CPU}$ |
| | | $f_{CPU}$ = 8 MHz | 1.25 | | 2.75 | µs |

1. Data based on typical application software.
2. Time measured between interrupt event and interrupt vector fetch. $\Delta t_{c(INST)}$ is the number of $t_{CPU}$ cycles needed to finish the current instruction execution.

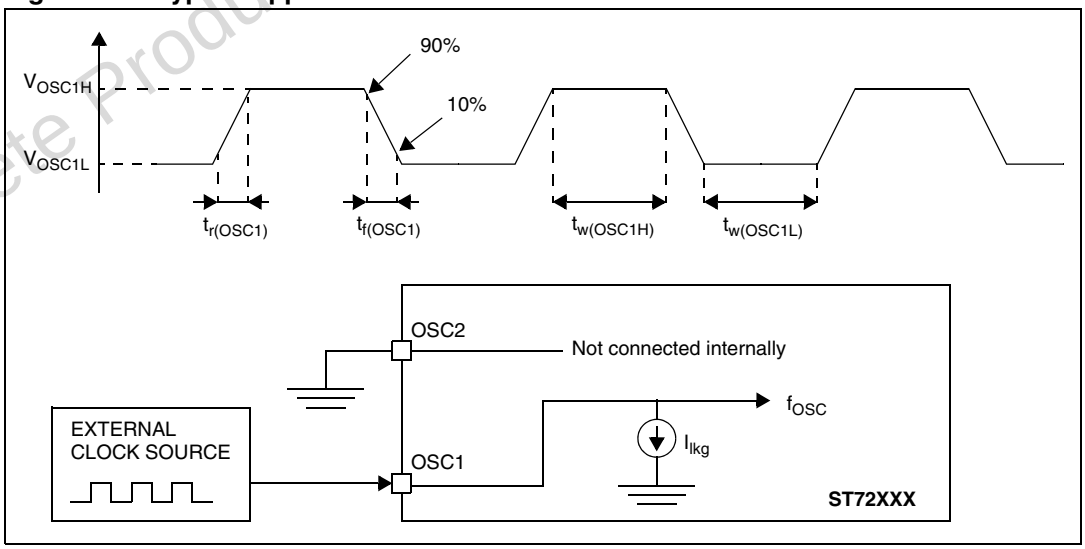
### 19.5.2 External clock source

**Table 116. External clock source**

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $V_{OSC1H}$ | OSC1 input pin high level voltage | See *Figure 76* | 0.7x$V_{DD}$ | | $V_{DD}$ | V |
| $V_{OSC1L}$ | OSC1 input pin low level voltage | | $V_{SS}$ | | 0.3x$V_{DD}$ | |
| $t_{w(OSC1H)}$ $t_{w(OSC1L)}$ | OSC1 high or low time(1) | | 5 | | | ns |
| $t_{r(OSC1)}$ $t_{f(OSC1)}$ | OSC1 rise or fall time(1) | | | | 15 | |
| $I_{lkg}$ | OSC1 input leakage current | $V_{SS} \le V_{IN} \le V_{DD}$ | | | ±1 | µA |

1. Data based on design simulation and/or technology characteristics, not tested in production.

**Figure 76. Typical application with an external clock source**




Obsolete Product(s) - Obsolete Product(s)

### 19.5.3 Crystal and ceramic resonator oscillators

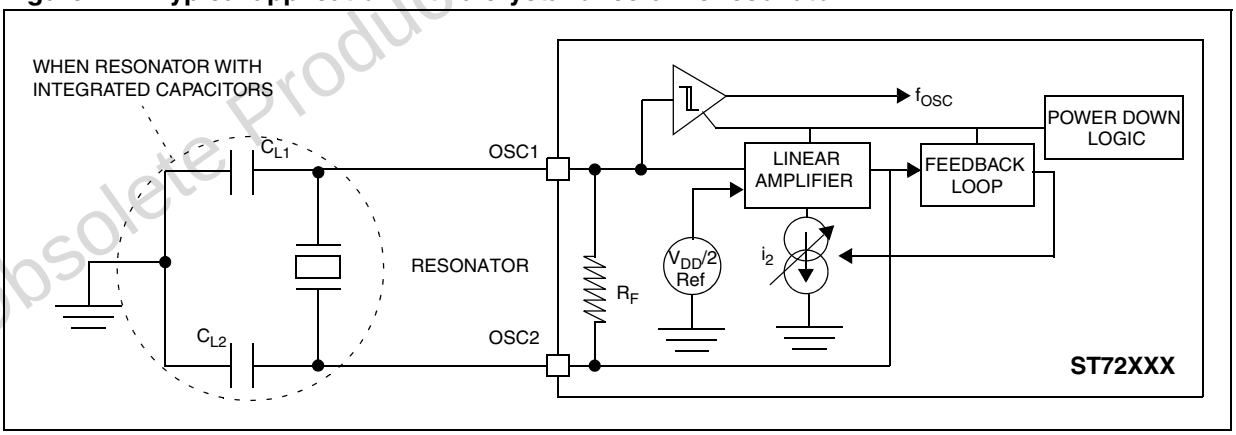
The ST7 internal clock can be supplied with four different crystal/ceramic resonator oscillators. All the information given in this paragraph is based on characterization results with specified typical external components. In the application, the resonator and the load capacitors have to be placed as close as possible to the oscillator pins in order to minimize output distortion and start-up stabilization time. Refer to the crystal/ceramic resonator manufacturer for more details (such as frequency, package or accuracy).

**Table 117. Crystal and ceramic resonator oscillators**

| Symbol | Parameter | Conditions | | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|---|
| $f_{OSC}$ | Oscillator frequency(1) | - | | 1 | | 16 | MHz |
| $R_F$ | Feedback resistor(2) | - | | 20 | - | 40 | kΩ |
| $C_{L1}$ $C_{L2}$ | Recommended load capacitance versus equivalent serial resistance of the crystal or ceramic resonator ($R_S$)(3) | $R_S$ = 200Ω<br>$R_S$ = 200Ω<br>$R_S$ = 200Ω<br>$R_S$ = 100Ω | LP oscillator<br>MP oscillator<br>MS oscillator<br>HS oscillator | 20<br>20<br>15<br>15 | - | 60<br>50<br>35<br>35 | pF |
| $i_2$ | OSC2 driving current | $V_{DD}$ = 5V, $V_{IN}$ = $V_{SS}$ | LP oscillator<br>MP oscillator<br>MS oscillator<br>HS oscillator | - | 426<br>425<br>456<br>660 | - | µA |

1. The oscillator selection can be optimized in terms of supply current using a high quality resonator with small $R_S$ value. Refer to crystal/ceramic resonator manufacturer for more details.
2. Data based on characterization results, not tested in production. The relatively low value of the RF resistor offers a good protection against issues resulting from use in a humid environment, due to the induced leakage and the bias condition change. However, it is recommended to take this point into account if the microcontroller is used in tough humidity conditions.
3. For $C_{L1}$ and $C_{L2}$ it is recommended to use high-quality ceramic capacitors in the 5pF to 25pF range (typ.) designed for high-frequency applications and selected to match the requirements of the crystal or resonator. $C_{L1}$ and $C_{L2}$ are usually the same size. The crystal manufacturer typically specifies a load capacitance which is the series combination of $C_{L1}$ and $C_{L2}$. PCB and MCU pin capacitance must be included when sizing $C_{L1}$ and $C_{L2}$ (10pF can be used as a rough estimate of the combined pin and board capacitance).

**Figure 77. Typical application with a crystal or ceramic resonator**




Obsolete Product(s) - Obsolete product(s)

**Table 118. Resonator characteristics**

| Supplier | $f_{OSC}$ (MHz) | Typical ceramic resonators(1) |
|---|---|---|
| Murata | 2 | CSTCC2M00G56Z-R0 |
| | 4 | CSTCR4M00G53Z-R0 |
| | 8 | CSTCE8M00G52Z-R0 |
| | 16 | CSTCE16M0V51Z-R0 |

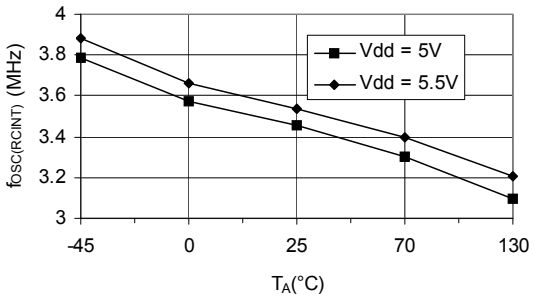
1. Resonator characteristics given by the ceramic resonator manufacturer. For more information on these resonators, please consult www.murata.com.

## 19.5.4 RC oscillators

**Table 119. RC oscillator characteristics**

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $f_{OSC(RCINT)}$ | Internal RC oscillator frequency (see *Figure 78*) | $T_A$ = 25°C, $V_{DD}$ = 5V | 2 | 3.5 | 5.6 | MHz |

**Figure 78. Typical $f_{OSC(RCINT)}$ versus $T_A$**




*Note: To reduce disturbance to the RC oscillator, it is recommended to place decoupling capacitors between $V_{DD}$ and $V_{SS}$ as shown in Figure 98.*

## 19.5.5 Clock security system (CSS)

**Table 120. CSS characteristics**

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $f_{SFOSC}$ | Safe oscillator frequency(1) | | | 3 | | MHz |

1. Data based on characterization results

Obsolete Product(s)

### 19.5.6 PLL characteristics

**Table 121. PLL characteristics**

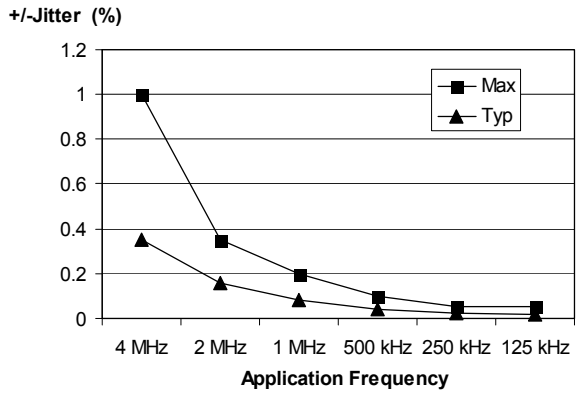
| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $f_{OSC}$ | PLL input frequency range | | 2 | | 4 | MHz |
| $\Delta f_{CPU}/f_{CPU}$ | Instantaneous PLL jitter(1) | $f_{OSC}$ = 4 MHz | | 0.7 | 2 | % |

1. Data characterized but not tested

The user must take the PLL jitter into account in the application (for example, in serial communication or sampling of high frequency signals). The PLL jitter is a periodic effect, which is integrated over several CPU cycles. Therefore, the longer the period of the application signal, the less it is impacted by the PLL jitter.

*Figure 79* shows the PLL jitter integrated on application signals in the range 125 kHz to 4 MHz. At frequencies of less than 125 kHz, the jitter is negligible.

**Figure 79. Integrated PLL jitter versus signal frequency(1)**




1. Measurement conditions: $f_{CPU}$ = 8 MHz

## 19.6 Memory characteristics

### 19.6.1 RAM and hardware registers

**Table 122. RAM supply voltage**

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $V_{RM}$ | Data retention mode(1) | Halt mode (or RESET) | 1.6 | | | V |

1. Minimum $V_{DD}$ supply voltage without losing data stored in RAM (in Halt mode or under RESET) or in hardware registers (only in Halt mode). Not tested in production.

Obsolete Product(s)

### 19.6.2 Flash memory

**Table 123. Dual voltage HDFlash memory**

| Symbol | Parameter | Conditions | Min(1) | Typ | Max(1) | Unit |
|---|---|---|---|---|---|---|
| $f_{CPU}$ | Operating frequency | Read mode | 0 | | 8 | MHz |
| | | Write / Erase mode | 1 | | 8 | |
| $V_{PP}$ | Programming voltage(2) | $4.5V \le V_{DD} \le 5.5V$ | 11.4 | | 12.6 | V |
| $I_{DD}$ | Supply current(3) | Run mode ($f_{CPU}$ = 4 MHz) | | | 3 | mA |
| | | Write / Erase | | 0 | | |
| | | Power down mode / HALT | | 1 | 10 | µA |
| $I_{PP}$ | $V_{PP}$ current(3) | Read ($V_{PP}$ = 12V) | | | 200 | µA |
| | | Write / Erase | | | 30 | mA |
| $t_{VPP}$ | Internal $V_{PP}$ stabilization time | | | 10 | | µs |
| $t_{RET}$ | Data retention | $T_A$ = 55°C | 20 | | | years |
| $N_{RW}$ | Write erase cycles | $T_A$ = 85°C | 100 | | | cycles |
| $T_{PROG}$ $T_{ERASE}$ | Programming or erasing temperature range | | -40 | 25 | 85 | °C |

1. Data based on characterization results, not tested in production
2. $V_{PP}$ must be applied only during the programming or erasing operation and not permanently for reliability reasons.
3. Data based on simulation results, not tested in production

**Warning: Do not connect 12V to $V_{PP}$ before $V_{DD}$ is powered on, as this may damage the device.**

Obsolete Product(s) - Obsolete Product(s)

## 19.7 EMC (electromagnetic compatibility) characteristics

Susceptibility tests are performed on a sample basis during product characterization.

### 19.7.1 Functional EMS (electromagnetic susceptibility)

Based on a simple running application on the product (toggling two LEDs through I/O ports), the product is stressed by two electromagnetic events until a failure occurs (indicated by the LEDs).

- **ESD**: Electrostatic discharge (positive and negative) is applied on all pins of the device until a functional disturbance occurs. This test conforms with the IEC 1000-4-2 standard.
- **FTB**: A burst of fast transient voltage (positive and negative) is applied to $V_{DD}$ and $V_{SS}$ through a 100pF capacitor until a functional disturbance occurs. This test conforms with the IEC 1000-4-4 standard.

A device reset allows normal operations to be resumed. The test results given in *Table 124* below are based on the EMS levels and classes defined in application note AN1709.

#### Designing hardened software to avoid noise problems

EMC characterization and optimization are performed at component level with a typical application environment and simplified MCU software. It should be noted that good EMC performance is highly dependent on the user application and the software in particular.

Therefore it is recommended that the user applies EMC software optimization and prequalification tests in relation with the EMC level requested for his application.

**Software recommendations**

The software flowchart must include the management of runaway conditions such as:

- Corrupted program counter
- Unexpected reset
- Critical Data corruption (control registers...)

**Prequalification trials**

Most of the common failures (unexpected reset and program counter corruption) can be reproduced by manually forcing a low state on the $\overline{\text{RESET}}$ pin or the oscillator pins for 1 second.

To complete these trials, ESD stress can be applied directly on the device, over the range of specification values. When unexpected behavior is detected, the software can be hardened to prevent unrecoverable errors occurring (see application note AN1015).

Obsolete Product(s) - Obsolete Product(s)

**Table 124. EMS test results**

| Symbol | Parameter | Conditions | Level/Class |
|---|---|---|---|
| $V_{FESD}$ | Voltage limits to be applied on any I/O pin to induce a functional disturbance | $V_{DD}$ = 5V, $T_A$ = +25°C, $f_{OSC}$ = 8 MHz, conforms to IEC 1000-4-2 | 3B |
| $V_{FFTB}$ | Fast transient voltage burst limits to be applied through 100pF on $V_{DD}$ and $V_{DD}$ pins to induce a functional disturbance | LQFP44:<br>$V_{DD}$ = 5V, $T_A$ = +25°C, $f_{OSC}$ = 8 MHz, conforms to IEC 1000-4-4 | 3B |

## 19.7.2 EMI (electromagnetic interference)

Based on a simple application running on the product (toggling two LEDs through the I/O ports), the product is monitored in terms of emission. This emission test is in line with the norm SAE J 1752/3 which specifies the board and the loading of each pin.

**Table 125. EMI emissions**

| Symbol | Parameter | Conditions | Monitored frequency band | Max vs [$f_{OSC}/f_{CPU}$](1) | | Unit |
|---|---|---|---|---|---|---|
| | | $V_{DD}$ = 5V, $T_A$ = +25°C, conforming to SAE J 1752/3 | | 8/4 MHz | 16/8 MHz | |
| $S_{EMI}$ | Peak level | 48/60 Kbyte Flash devices in LQFP44 and LQFP64 packages and 32 Kbyte Flash devices in LQFP64 package | 0.1 MHz to 30 MHz | 15 | 20 | dBµV |
| | | | 30 MHz to 130 MHz | 20 | 27 | |
| | | | 130 MHz to 1 GHz | 7 | 12 | |
| | | | SAE EMI Level | 2.5 | 3 | - |
| $S_{EMI}$ | Peak level | 16/32 Kbyte Flash devices in LQFP32 and LQFP44 packages | 0.1 MHz to 30 MHz | 13 | 14 | dBµV |
| | | | 30 MHz to 130 MHz | 20 | 25 | |
| | | | 130 MHz to 1 GHz | 16 | 21 | |
| | | | SAE EMI Level | 3 | 3.5 | - |

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

## 19.7.3 Absolute maximum ratings (electrical sensitivity)

Based on three different tests (ESD, LU and DLU) using specific measurement methods, the product is stressed in order to determine its performance in terms of electrical sensitivity. For more details, refer to the application note AN1181.

### Electrostatic discharge (ESD)

Electrostatic discharges (a positive then a negative pulse separated by 1 second) are applied to the pins of each sample according to each pin combination. The sample size depends on the number of supply pins in the device (3 parts*(n+1) supply pin). Two models can be simulated: Human Body Model and Machine Model. This test conforms to the JESD22-A114A/A115A standard.

Obsolete Product(s) - Obsolete Product(s)

**Table 126. ESD absolute maximum ratings**

| Symbol | Ratings | Conditions | Max. value(1) | Unit |
|---|---|---|---|---|
| $V_{ESD(HBM)}$ | Electrostatic discharge voltage (Human Body Model) | $T_A$ = +25°C | 2000 | V |
| $V_{ESD(MM)}$ | Electrostatic discharge voltage (Machine Model) | | 200 | |

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

## Static and dynamic latch-up

- **LU**: Three complementary static tests are required on 10 parts to assess the latch-up performance. A supply overvoltage (applied to each power supply pin) and a current injection (applied to each input, output and configurable I/O pin) are performed on each sample. This test conforms to the EIA/JESD 78 IC latch-up standard. For more details, refer to the application note AN1181.
- **DLU**: Electrostatic discharges (one positive then one negative test) are applied to each pin of three samples when the micro is running to assess the latch-up performance in dynamic mode. Power supplies are set to the typical values, the oscillator is connected as near as possible to the pins of the micro and the component is put in reset mode. This test conforms to the IEC1000-4-2 and SAEJ1752/3 standards. For more details, refer to the application note AN1181.

**Table 127. Electrical sensitivities**

| Symbol | Parameter | Conditions | Class(1) |
|---|---|---|---|
| LU | Static latch-up class | $T_A$ = +25°C<br>$T_A$ = +85°C<br>$T_A$ = +125°C | A<br>A<br>A |
| DLU | Dynamic latch-up class | $V_{DD}$ = 5.5V, $f_{OSC}$ = 4 MHz, $T_A$ = +25°C | A |

1. Class description: A Class is an STMicroelectronics internal specification. All its limits are higher than the JEDEC specifications, that means when a device belongs to Class A it exceeds the JEDEC standard. B Class strictly covers all the JEDEC criteria (international standard).

Obsolete Product(s) - Obsolete Product(s)

# 19.8 I/O port pin characteristics

## 19.8.1 General characteristics

Subject to general operating conditions for $V_{DD}$, $f_{OSC}$, and $T_A$ unless otherwise specified.

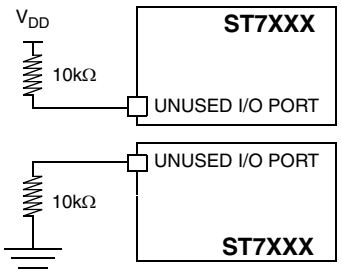
**Table 128. I/O port pin general characteristics**

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $V_{IL}$ | Input low level voltage(1) | CMOS ports | | | $0.3xV_{DD}$ | V |
| $V_{IH}$ | Input high level voltage(1) | | $0.7xV_{DD}$ | | | |
| $V_{hys}$ | Schmitt trigger voltage hysteresis(2) | | | 0.7 | | |
| $I_{INJ(PIN)}$(3) | Injected current on PB0 pin | $V_{DD}$ = 5V | 0 | | +4 | mA |
| | Injected current on an I/O pin | | | | ±4 | |
| $\Sigma I_{INJ(PIN)}$(3) | Total injected current (sum of all I/O and control pins) | | | | ±25 | |
| $I_{lkg}$ | Input leakage current | $V_{SS} \le V_{IN} \le V_{DD}$ | | | ±1 | µA |
| $I_S$ | Static current consumption | Floating input mode(4)(5) | | 400 | | |
| $R_{PU}$ | Weak pull-up equivalent resistor(6) | $V_{IN} = V_{SS}$ $V_{DD}$ = 5V | 50 | 120 | 250 | kΩ |
| $C_{IO}$ | I/O pin capacitance | | | 5 | | pF |
| $t_{f(IO)out}$ | Output high to low level fall time(1) | $C_L$ = 50pF Between 10% and 90% | | 25 | | ns |
| $t_{r(IO)out}$ | Output low to high level rise time(1) | | | 25 | | |
| $t_{w(IT)in}$ | External interrupt pulse time(7) | | 1 | | | $t_{CPU}$ |

1. Data based on characterization results, not tested in production.
2. Hysteresis voltage between Schmitt trigger switching levels. Based on characterization results, not tested.
3. When the current limitation is not possible, the $V_{IN}$ maximum must be respected, otherwise refer to $I_{INJ(PIN)}$ specification. A positive injection is induced by $V_{IN} > V_{DD}$ while a negative injection is induced by $V_{IN} < V_{SS}$. Refer to *Section 19.2.2: Current characteristics* for more details.
4. Static peak current value taken at a fixed $V_{IN}$ value, based on design simulation and technology characteristics, not tested in production. This value depends on $V_{DD}$ and temperature values.
5. The Schmitt trigger that is connected to every I/O port is disabled for analog inputs only when ADON bit is ON and the particular ADC channel is selected (with port configured in input floating mode). When the ADON bit is OFF, static current consumption may result. This can be avoided by keeping the input voltage of this pin close to $V_{DD}$ or $V_{SS}$.
6. The $R_{PU}$ pull-up equivalent resistor is based on a resistive transistor (corresponding $I_{PU}$ current characteristics described in *Figure 81*).
7. To generate an external interrupt, a minimum pulse width has to be applied on an I/O port pin configured as an external interrupt source.

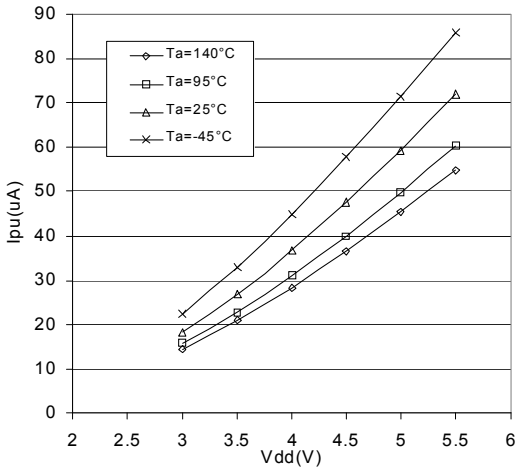
Obsolete Product(s) Obsolete Product(s)

**Figure 80. Unused I/O pins configured as input**




*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 81. Typical $I_{PU}$ vs $V_{DD}$ with $V_{IN} = V_{SS}$**




## 19.8.2 Output driving current

Subject to general operating conditions for $V_{DD}$, $f_{CPU}$, and $T_A$ unless otherwise specified.

**Table 129. Output driving current**

| Symbol | Parameter | Conditions | | Min | Max | Unit |
|---|---|---|---|---|---|---|
| $V_{OL}$(1) | Output low level voltage for a standard I/O pin when 8 pins are sunk at same time (see *Figure 82*) | $V_{DD}$ = 5V | $I_{IO}$ = +5mA | | 1.2 | V |
| | | | $I_{IO}$ = +2mA | | 0.5 | |
| | Output low level voltage for a high sink I/O pin when 4 pins are sunk at same time (see *Figure 83* and *Figure 85*) | | $I_{IO}$ = +20mA, $T_A \le 85°C$<br>$T_A \ge 85°C$ | | 1.3<br>1.5 | |
| | | | $I_{IO}$ = +8mA | | 0.6 | |
| $V_{OH}$(2) | Output high level voltage for an I/O pin when 4 pins are sourced at same time (see *Figure 84* and *Figure 87*) | | $I_{IO}$ = -5mA, $T_A \le 85°C$<br>$T_A \ge 85°C$ | $V_{DD}$ - 1.4<br>$V_{DD}$ - 1.6 | | |
| | | | $I_{IO}$ = -2mA | $V_{DD}$ - 0.7 | | |

1. The $I_{IO}$ current sunk must always respect the absolute maximum rating specified in *Section 19.2.2* and the sum of $I_{IO}$ (I/O ports and control pins) must not exceed $I_{VSS}$.
2. The $I_{IO}$ current sourced must always respect the absolute maximum rating specified in *Section 19.2.2* and the sum of $I_{IO}$ (I/O ports and control pins) must not exceed $I_{VDD}$. True open-drain I/O pins do not have $V_{OH}$.

Obsolete Product(s) - Obsolete Product(s)

**Figure 82. Typical $V_{OL}$ at $V_{DD}$ = 5V (standard)**

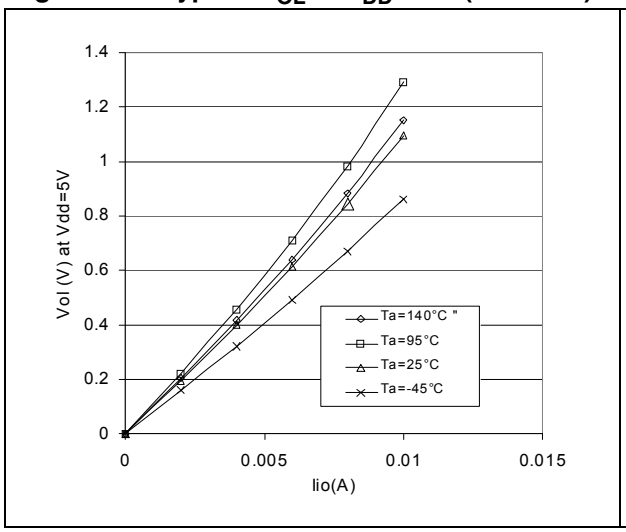



**Figure 83. Typical $V_{OL}$ at $V_{DD}$ = 5V (high-sink)**

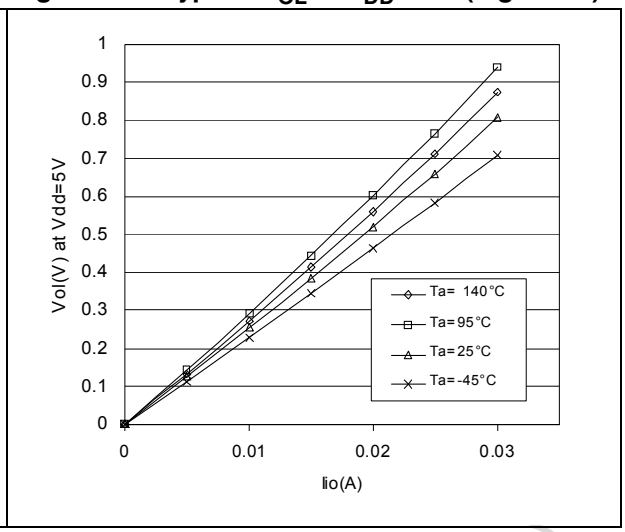
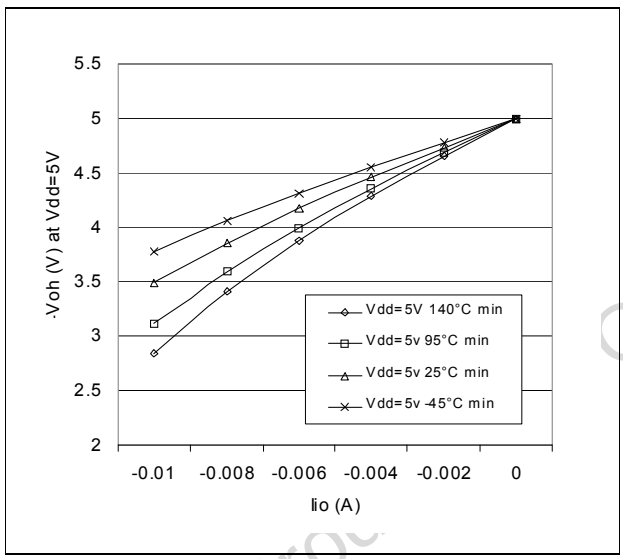



**Figure 84. Typical $V_{OH}$ at $V_{DD}$ = 5V**




Obsolete Product(s)
Obsolete Prod

**Figure 85. Typical $V_{OL}$ versus $V_{DD}$ (standard)**

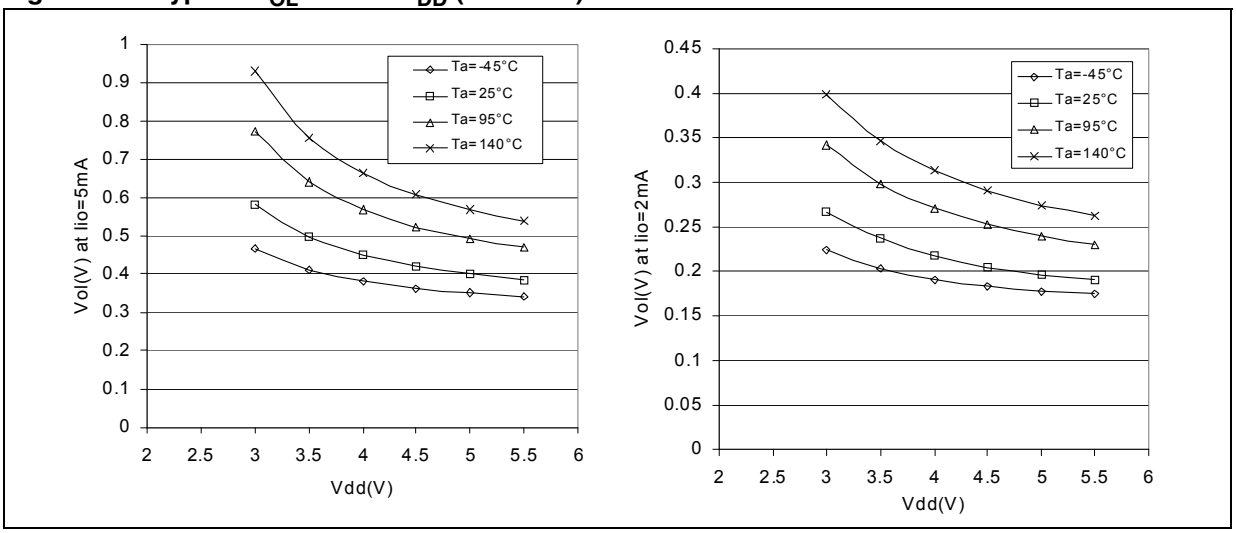



**Figure 86. Typical $V_{OL}$ versus $V_{DD}$ (high-sink)**

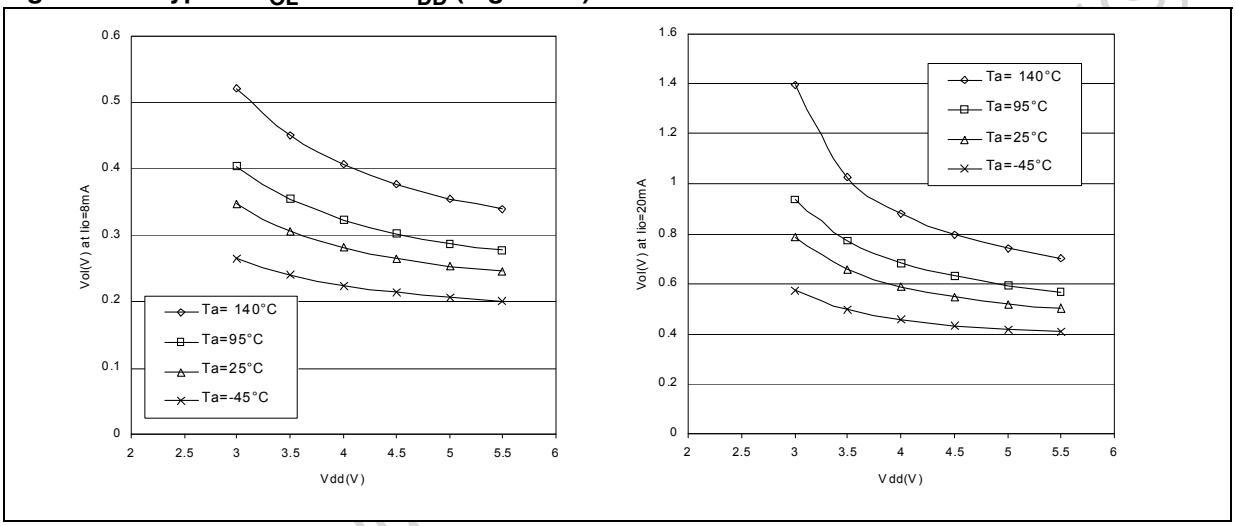
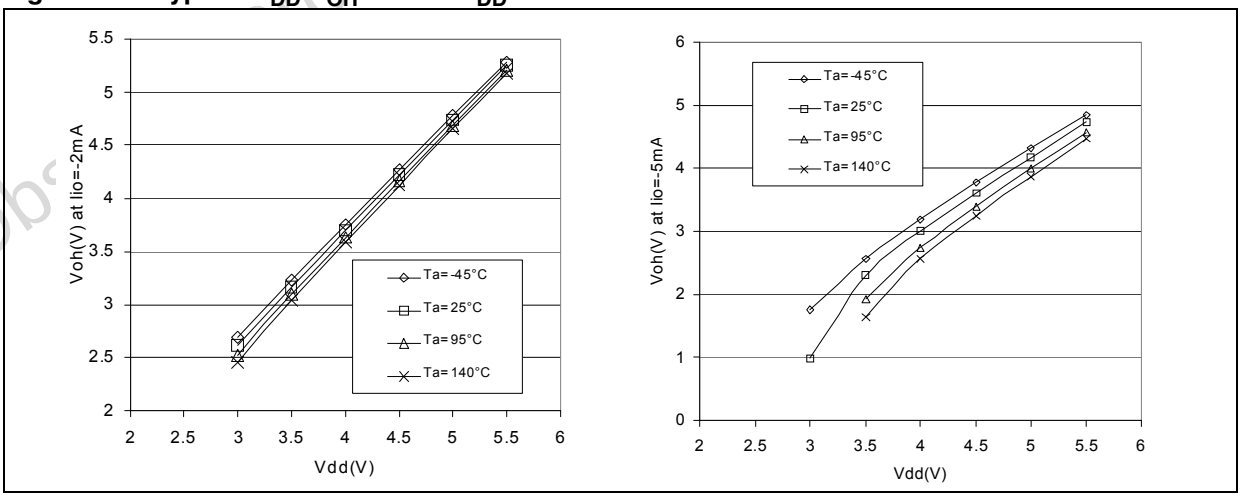



**Figure 87. Typical $V_{DD}$-$V_{OH}$ versus $V_{DD}$**



## 19.9 Control pin characteristics

### 19.9.1 Asynchronous $\overline{\text{RESET}}$ pin

Subject to general operating conditions for $V_{DD}$, $f_{CPU}$, and $T_A$ unless otherwise specified.

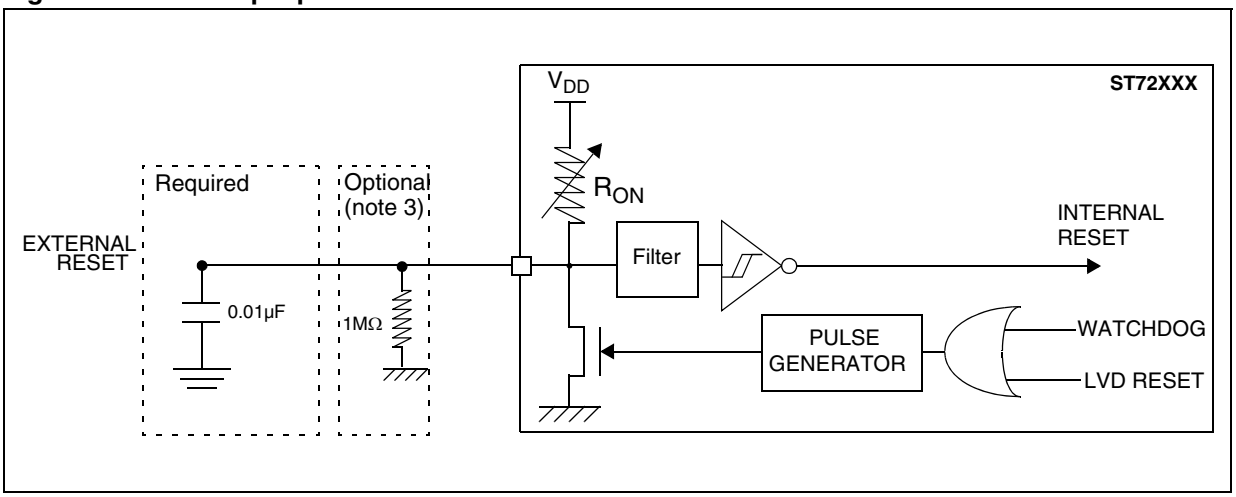
**Table 130. Asynchronous $\overline{\text{RESET}}$ pin characteristics**

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $V_{IL}$ | Input low level voltage(1) | | | | $0.3xV_{DD}$ | V |
| $V_{IH}$ | Input high level voltage(1) | | $0.7xV_{DD}$ | | | |
| $V_{hys}$ | Schmitt trigger voltage hysteresis(2) | | | 2.5 | | |
| $V_{OL}$ | Output low level voltage(3) | $V_{DD}$ = 5V, $I_{IO}$ = +2mA | | 0.2 | 0.5 | |
| $I_{IO}$ | Input current on $\overline{\text{RESET}}$ pin | | | 2 | | mA |
| $R_{ON}$ | Weak pull-up equivalent resistor | | 20 | 30 | 120 | kΩ |
| $t_{w(RSTL)out}$ | Generated reset pulse duration | Stretch applied on external pulse | 0 | | 42(4) | µs |
| | | Internal reset sources | 20 | 30 | 42(4) | |
| $t_{h(RSTL)in}$ | External reset pulse hold time(5) | | 2.5 | | | |
| $t_{g(RSTL)in}$ | Filtered glitch duration(6) | | | 200 | | ns |

1. Data based on characterization results, not tested in production.
2. Hysteresis voltage between Schmitt trigger switching levels.
3. The $I_{IO}$ current sunk must always respect the absolute maximum rating specified in *Section 19.2.2* and the sum of $I_{IO}$ (I/O ports and control pins) must not exceed $I_{VSS}$.
4. Data guaranteed by design, not tested in production.
5. To guarantee the reset of the device, a minimum pulse has to be applied to the $\overline{\text{RESET}}$ pin. All short pulses applied on the $\overline{\text{RESET}}$ pin with a duration below $t_{h(RSTL)in}$ can be ignored.
6. The reset network (the resistor and two capacitors) protects the device against parasitic resets, especially in noisy environments.

Obsolete Product(s) - Obsolete Product(s)

**Figure 88. $\overline{\text{RESET}}$ pin protection when LVD is enabled**




*Note: 1 - The reset network protects the device against parasitic resets.*
*- The output of the external reset circuit must have an open-drain output to drive the ST7 reset pad. Otherwise the device can be damaged when the ST7 generates an internal reset (LVD or watchdog).*
*- Whether the reset source is internal or external, the user must ensure that the level on the $\overline{\text{RESET}}$ pin can go below the $V_{IL}$ maximum level specified in Section 19.9.1 on page 210. Otherwise the reset will not be taken into account internally.*
*- Because the reset circuit is designed to allow the internal RESET to be output in the $\overline{\text{RESET}}$ pin, the user must ensure that the current sunk on the $\overline{\text{RESET}}$ pin is less than the absolute maximum value specified for $I_{INJ(RESET)}$ in Section 19.2.2 on page 191.*

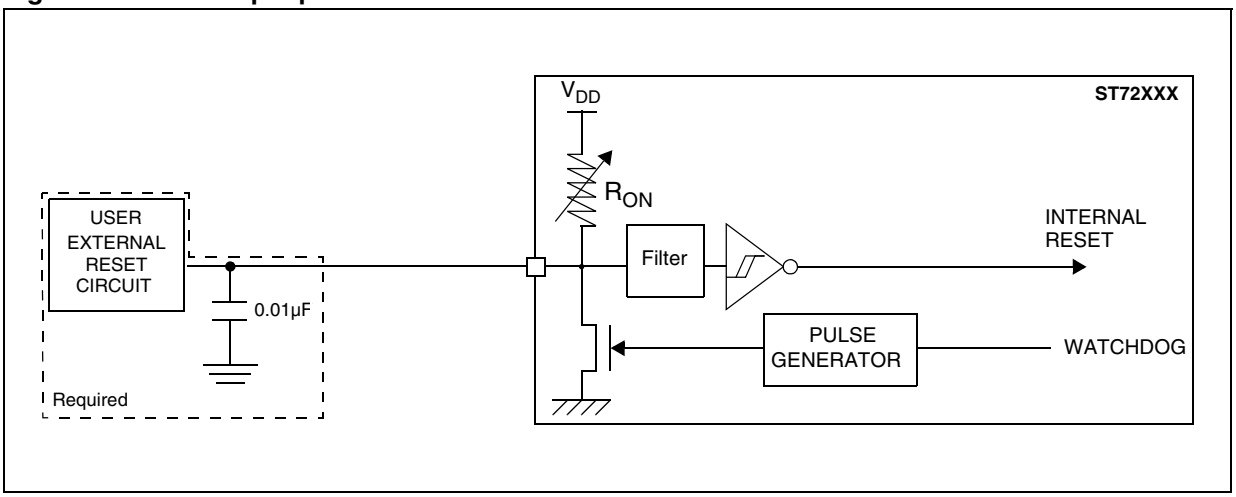
*2 When the LVD is enabled, it is recommended not to connect a pull-up resistor or capacitor. A 10nF pull-down capacitor is required to filter noise on the reset line.*

*3 In case a capacitive power supply is used, it is recommended to connect a 1MΩ pull-down resistor to the $\overline{\text{RESET}}$ pin to discharge any residual voltage induced by the capacitive effect of the power supply (this will add 5µA to the power consumption of the MCU).*

*4 Tips when using the LVD:*
*A. Check that all recommendations related to reset circuit have been applied (see notes above).*
*B. Check that the power supply is properly decoupled (100nF + 10µF close to the MCU). Refer to AN1709 and AN2017. If this cannot be done, it is recommended to put a 100nF + 1MΩ pull-down on the $\overline{\text{RESET}}$ pin.*
*C. The capacitors connected on the $\overline{\text{RESET}}$ pin and also the power supply are key to avoid any start-up marginality. In most cases, steps A and B above are sufficient for a robust solution. Otherwise, replace 10nF pull-down on the $\overline{\text{RESET}}$ pin with a 5µF to 20µF capacitor.*

Obsolete Product(s) - Obsolete Product(s)

**Figure 89. $\overline{\text{RESET}}$ pin protection when LVD is disabled**




*Note: - The reset network protects the device against parasitic resets.*
*- The output of the external reset circuit must have an open-drain output to drive the ST7 reset pad. Otherwise the device can be damaged when the ST7 generates an internal reset (LVD or watchdog).*
*- Whether the reset source is internal or external, the user must ensure that the level on the $\overline{\text{RESET}}$ pin can go below the $V_{IL}$ maximum level specified in Section 19.9.1 on page 210. Otherwise the reset will not be taken into account internally.*
*- Because the reset circuit is designed to allow the internal $\overline{\text{RESET}}$ to be output in the $\overline{\text{RESET}}$ pin, the user must ensure that the current sunk on the $\overline{\text{RESET}}$ pin is less than the absolute maximum value specified for $I_{INJ(RESET)}$ in Section 19.2.2 on page 191.*

## 19.9.2 ICCSEL/$V_{PP}$ pin

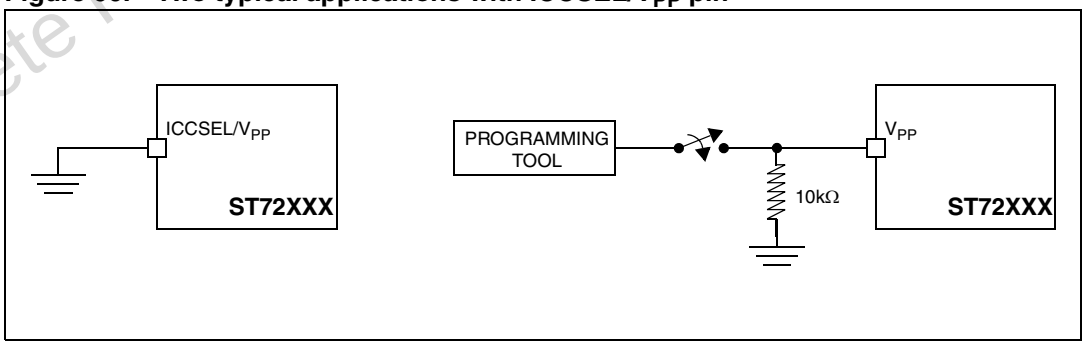
Subject to general operating conditions for $V_{DD}$, $f_{CPU}$, and $T_A$ unless otherwise specified.

**Table 131. ICCSEL/$V_{PP}$ pin characteristics**

| Symbol | Parameter | Conditions | Min | Max(1) | Unit |
|---|---|---|---|---|---|
| $I_{lkg}$ | Input leakage current | $V_{IN} = V_{SS}$ | | ±1 | µA |

1. Data based on design simulation and/or technology characteristics, not tested in production

**Figure 90. Two typical applications with ICCSEL/$V_{PP}$ pin(1)**




1. When ICC mode is not required by the application, the ICCSEL/$V_{PP}$ pin must be tied to $V_{SS}$.

Obsolete Product(s) - Obsolete Product(s)

## 19.10 Timer peripheral characteristics

Subject to general operating conditions for $V_{DD}$, $f_{OSC}$, and $T_A$ unless otherwise specified.

Refer to *Section 19.8: I/O port pin characteristics* for more details on the input/output alternate function characteristics (such as output compare, input capture, external clock, or PWM output).

**Table 132. 8-bit PWM-ART auto-reload timer characteristics**

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $t_{res(PWM)}$ | PWM resolution time | | 1 | | | $t_{CPU}$ |
| | | $f_{CPU}$ = 8 MHz | 125 | | | ns |
| $f_{EXT}$ | ART external clock frequency | | 0 | | $f_{CPU}/2$ | MHz |
| $f_{PWM}$ | PWM repetition rate | | | | | |
| $Res_{PWM}$ | PWM resolution | | | | 8 | bit |
| $V_{OS}$ | PWM/DAC output step voltage | $V_{DD}$ = 5V, Resolution = 8 bits | | 20 | | mV |

**Table 133. 16-bit timer characteristics**

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $t_{w(ICAP)in}$ | Input capture pulse time | | 1 | | | $t_{CPU}$ |
| $t_{res(PWM)}$ | PWM resolution time | | 2 | | | $t_{CPU}$ |
| | | $f_{CPU}$ = 8 MHz | 250 | | | ns |
| $f_{EXT}$ | Timer external clock frequency | | 0 | | $f_{CPU}/4$ | MHz |
| $f_{PWM}$ | PWM repetition rate | | | | | |
| $Res_{PWM}$ | PWM resolution | | | | 16 | bit |

Obsolete Product(s) - Obsolete Product(s)

## 19.11 Communication interface characteristics

### 19.11.1 SPI (serial peripheral interface)

Subject to general operating conditions for $V_{DD}$, $f_{CPU}$, and $T_A$ unless otherwise specified.

Refer to *Section 19.8: I/O port pin characteristics* for more details on the input/output alternate function characteristics ($\overline{SS}$, SCK, MOSI, MISO).

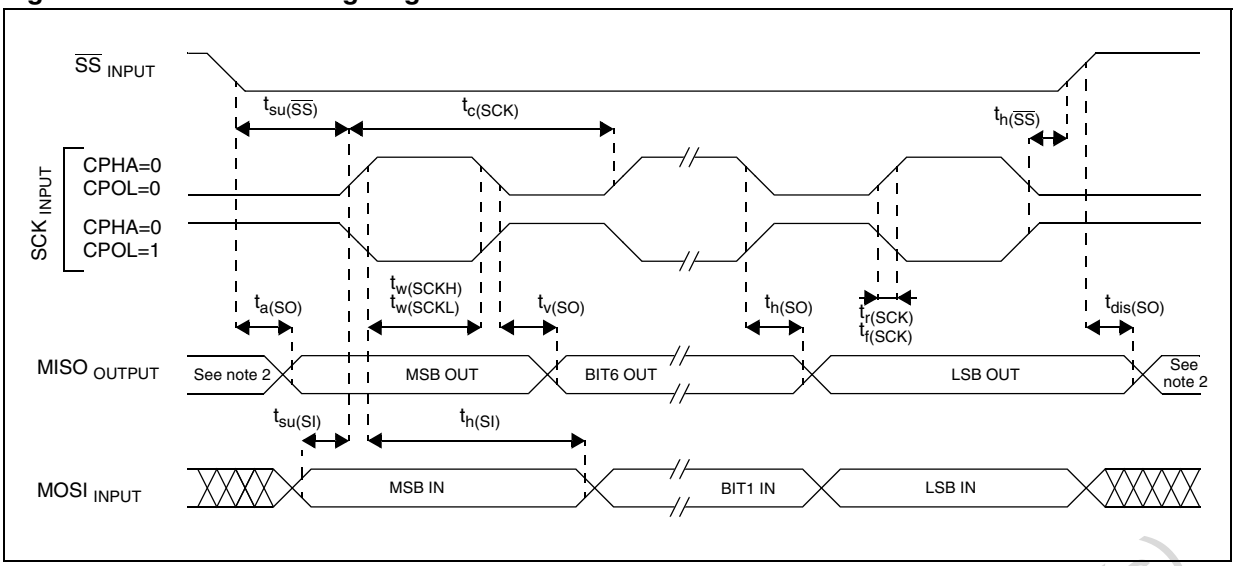
**Table 134. SPI characteristics**

| Symbol | Parameter | Conditions | Min | Max | Unit |
|---|---|---|---|---|---|
| $f_{SCK}$<br>$1/t_{c(SCK)}$ | SPI clock frequency | Master, $f_{CPU}$ = 8 MHz | $f_{CPU}/128$ = 0.0625 | $f_{CPU}/4$ = 2 | MHz |
| | | Slave, $f_{CPU}$ = 8 MHz | 0 | $f_{CPU}/2$ = 4 | |
| $t_{r(SCK)}$<br>$t_{f(SCK)}$ | SPI clock rise and fall time | | see I/O port pin description | | |
| $t_{su(\overline{SS})}$ (1) | $\overline{SS}$ setup time(2) | Slave | $t_{CPU}$ + 50 | | ns |
| $t_{h(\overline{SS})}$ (1) | $\overline{SS}$ hold time | Slave | 120 | | |
| $t_{w(SCKH)}$ (1)<br>$t_{w(SCKL)}$ (1) | SCK high and low time | Master<br>Slave | 100<br>90 | | |
| $t_{su(MI)}$ (1)<br>$t_{su(SI)}$ (1) | Data input setup time | Master<br>Slave | 100<br>100 | | |
| $t_{h(MI)}$ (1)<br>$t_{h(SI)}$ (1) | Data input hold time | Master<br>Slave | 100<br>100 | | |
| $t_{a(SO)}$ (1) | Data output access time | Slave | 0 | 120 | |
| $t_{dis(SO)}$ (1) | Data output disable time | Slave | | 240 | |
| $t_{v(SO)}$ (1) | Data output valid time | Slave (after enable edge) | | 120 | |
| $t_{h(SO)}$ (1) | Data output hold time | | 0 | | |
| $t_{v(MO)}$ (1) | Data output valid time | Master (after enable edge) | | 120 | $t_{CPU}$ |
| $t_{h(MO)}$ (1) | Data output hold time | | 0 | | |

1. Data based on design simulation and/or characterization results, not tested in production.
2. Depends on $f_{CPU}$. For example, if $f_{CPU}$ = 8 MHz, then $t_{CPU}$ = 1 / $f_{CPU}$ = 125 ns and $t_{su(\overline{SS})}$ = 175 ns.

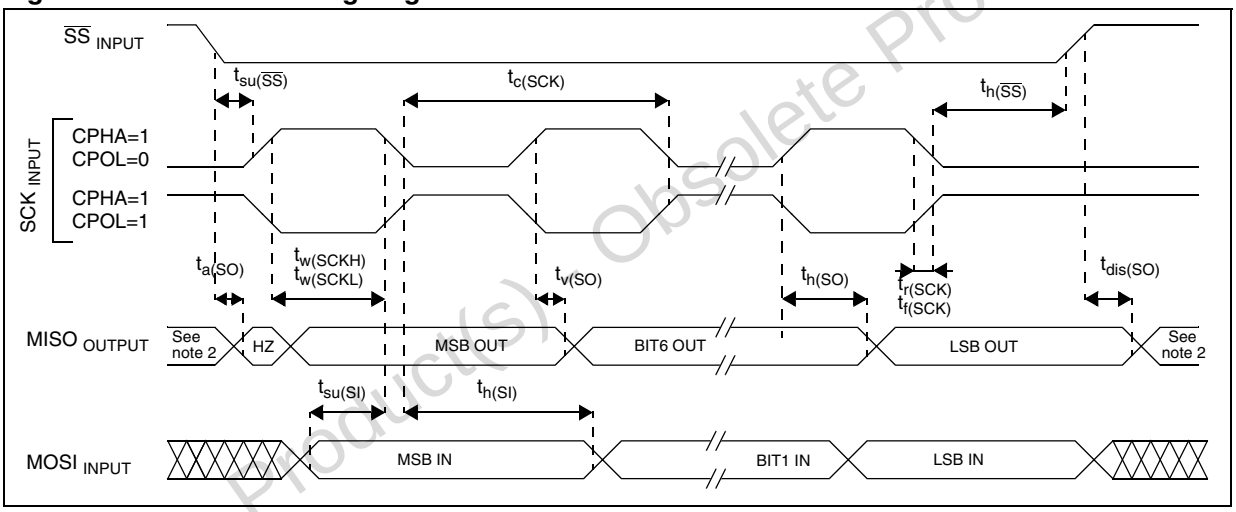
Obsolete Product(s) - Obsolete Product(s)

**Figure 91. SPI slave timing diagram with CPHA = 0(1)**




1. Measurement points are done at CMOS levels: $0.3xV_{DD}$ and $0.7xV_{DD}$.
2. When no communication is on-going the data output line of the SPI (MOSI in master mode, MISO in slave mode) has its alternate function capability released. In this case, the pin status depends on the I/O port configuration.

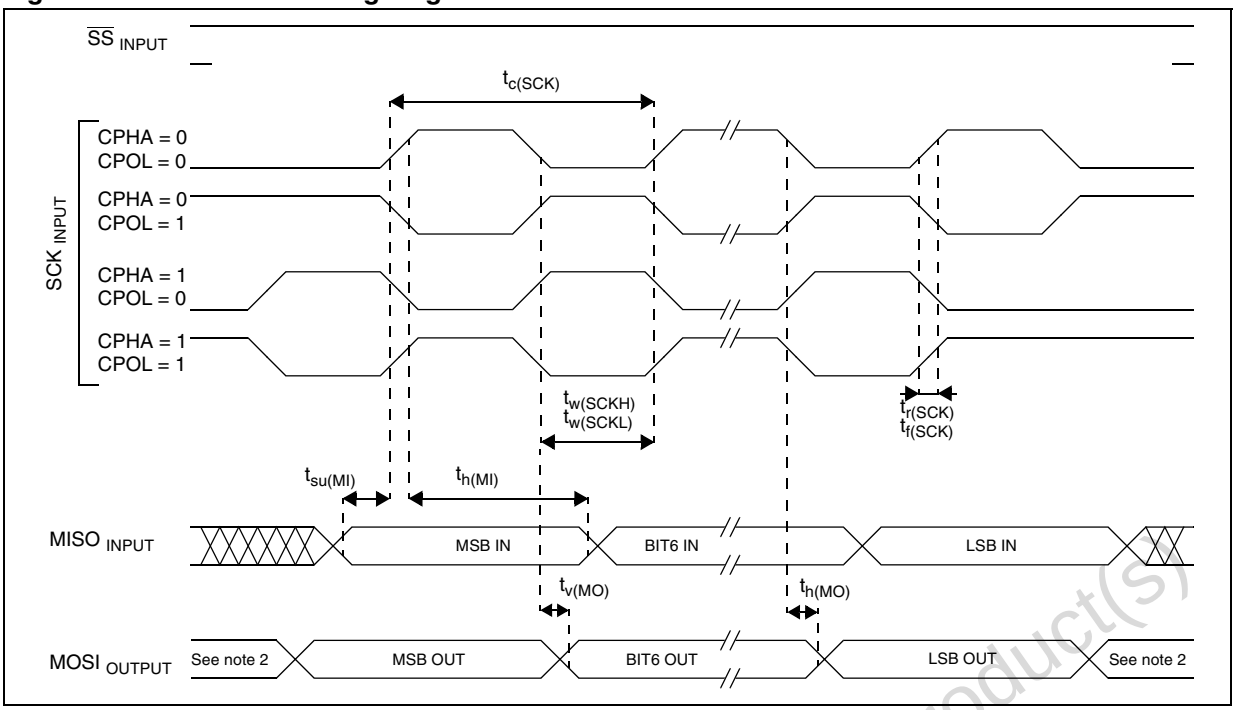
**Figure 92. SPI slave timing diagram with CPHA = 1(1)**




1. Measurement points are done at CMOS levels: $0.3xV_{DD}$ and $0.7xV_{DD}$.
2. When no communication is on-going the data output line of the SPI (MOSI in master mode, MISO in slave mode) has its alternate function capability released. In this case, the pin status depends of the I/O port configuration.

Obsolete Product(s) - Obsolete Product(s)

**Figure 93. SPI master timing diagram(1)**




1. Measurement points are done at CMOS levels: $0.3xV_{DD}$ and $0.7xV_{DD}$.
2. When no communication is on-going the data output line of the SPI (MOSI in master mode, MISO in slave mode) has its alternate function capability released. In this case, the pin status depends of the I/O port configuration.

Obsolete Product(s) - Obsolete Product(s)

### 19.11.2 I²C - inter IC control interface

Subject to general operating conditions for $V_{DD}$, $f_{CPU}$, and $T_A$ unless otherwise specified.

Refer to *Section 19.8: I/O port pin characteristics* for more details on the input/output alternate function characteristics (SDAI and SCLI). The ST7 I2C interface meets the requirements of the standard I2C communication protocol described in the following table.

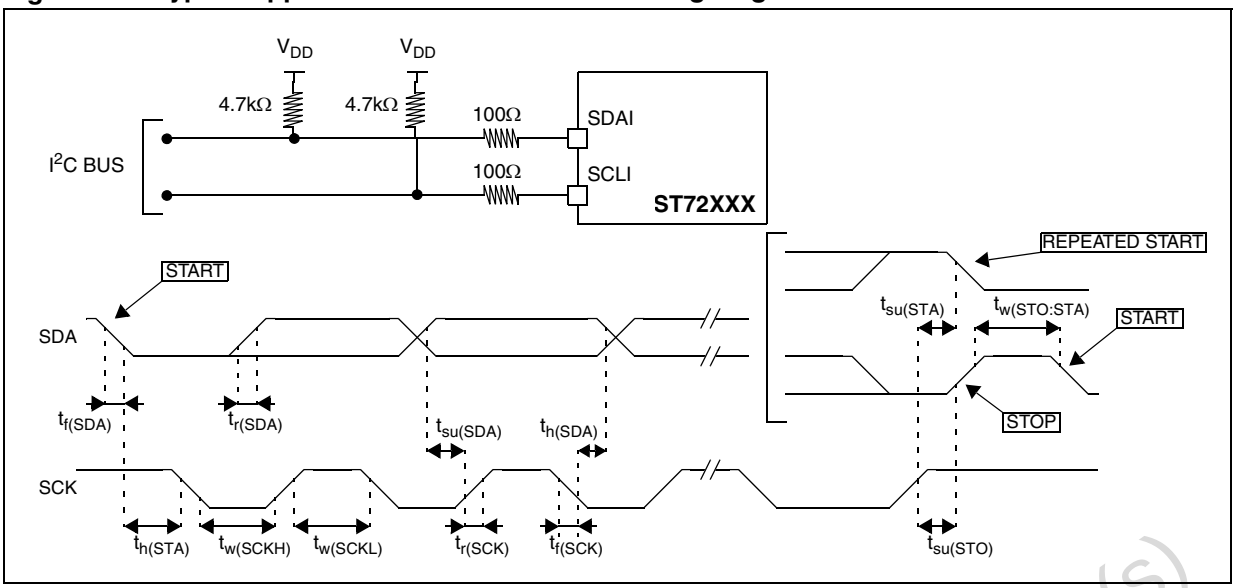
**Table 135. I2C control interface characteristics**

| Symbol | Parameter | Standard mode I²C | | Fast mode I²C(1) | | Unit |
|---|---|---|---|---|---|---|
| | | Min(2) | Max(2) | Min(2) | Max(2) | |
| $t_{w(SCLL)}$ | SCL clock low time | 4.7 | | 1.3 | | µs |
| $t_{w(SCLH)}$ | SCL clock high time | 4.0 | | 0.6 | | |
| $t_{su(SDA)}$ | SDA setup time | 250 | | 100 | | ns |
| $t_{h(SDA)}$ | SDA data hold time | 0(3) | | 0(4) | 900(3) | |
| $t_{r(SDA)}$ $t_{r(SCL)}$ | SDA and SCL rise time | | 1000 | 20+0.1$C_b$ | 300 | |
| $t_{f(SDA)}$ $t_{f(SCL)}$ | SDA and SCL fall time | | 300 | | | |
| $t_{h(STA)}$ | START condition hold time | 4.0 | | 0.6 | | µs |
| $t_{su(STA)}$ | Repeated START condition setup time | 4.7 | | | | |
| $t_{su(STO)}$ | STOP condition setup time | 4.0 | | | | |
| $t_{w(STO:STA)}$ | STOP to START condition time (bus free) | 4.7 | | 1.3 | | |
| $C_b$ | Capacitive load for each bus line | | 400 | | 400 | pF |

1. At 4 MHz $f_{CPU}$, maximum I²C speed (400 kHz) is not achievable. In this case, maximum I²C speed will be approximately 260 kHz.
2. Data based on standard I²C protocol requirement, not tested in production.
3. The maximum hold time of the START condition has only to be met if the interface does not stretch the low period of SCL signal.
4. The device must internally provide a hold time of at least 300ns for the SDA signal in order to bridge the undefined region of the falling edge of SCL.

Obsolete Product(s) - Obsolete Product(s)

**Figure 94. Typical application with I²C BUS and timing diagram(1)**




1. Measurement points are done at CMOS levels: $0.3xV_{DD}$ and $0.7xV_{DD}$.

The following table provides the values to be written in the I2CCCR register to obtain the required I²C SCL line frequency.

**Table 136. SCL frequency table**

| $f_{SCL}$ (kHz) | I2CCCR value | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | $f_{CPU}$ = 4 MHz | | | | $f_{CPU}$ = 8 MHz | | | |
| | $V_{DD}$ = 4.1V | | $V_{DD}$ = 5V | | $V_{DD}$ = 4.1V | | $V_{DD}$ = 5V | |
| | $R_P$ = 3.3kΩ | $R_P$ = 4.7kΩ | $R_P$ = 3.3kΩ | $R_P$ = 4.7kΩ | $R_P$ = 3.3kΩ | $R_P$ = 4.7kΩ | $R_P$ = 3.3kΩ | $R_P$ = 4.7kΩ |
| 400 | Not achievable | | | | 83h | | | |
| 300 | Not achievable | | | | 85h | | | |
| 200 | 83h | | | | 8Ah | 89h | 8Ah | |
| 100 | 10h | | | | 24h | 23h | 24h | 23h |
| 50 | 24h | | | | 4Ch | | | |
| 20 | 5Fh | | | | FFh | | | |

Legend:

$R_P$ = External pull-up resistance

$f_{SCL}$ = I²C speed

*Note: - For speeds around 200 kHz, the achieved speed can have a ±5% tolerance.*
*- For other speed ranges, the achieved speed can have a ±2% tolerance.*

*The above variations depend on the accuracy of the external components used.*

Obsolete Product(s) - Obsolete Product(s)

## 19.12 10-bit ADC characteristics

Subject to general operating conditions for $V_{DD}$, $f_{CPU}$, and $T_A$ unless otherwise specified.

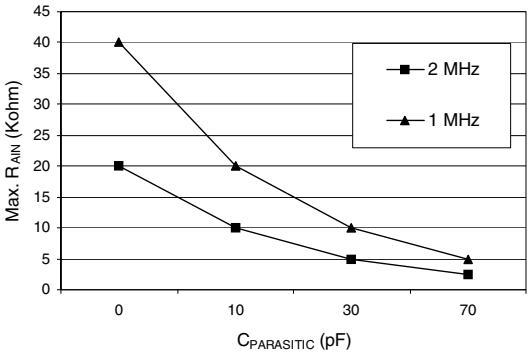
**Table 137. 10-bit ADC characteristics**

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $f_{ADC}$ | ADC clock frequency | | 0.4 | | 2 | MHz |
| $V_{AREF}$ | Analog reference voltage | $0.7*V_{DD} \le V_{AREF} \le V_{DD}$ | 3.8 | | $V_{DD}$ | V |
| $V_{AIN}$ | Conversion voltage range(1) | | $V_{SSA}$ | | $V_{AREF}$ | |
| $I_{lkg}$ | Input leakage current for analog input(2) | $-40°C \le T_A \le 85°C$ range | | | ±250 | nA |
| | | Other $T_A$ ranges | | | ±1 | µA |
| $R_{AIN}$ | External input impedance | | | | See *Figure 95* and *Figure 96* | kΩ |
| $C_{AIN}$ | External capacitor on analog input | | | | | pF |
| $f_{AIN}$ | Variation frequency of analog input signal | | | | | Hz |
| $C_{ADC}$ | Internal sample and hold capacitor | | | 12 | | pF |
| $t_{ADC}$ | Conversion time (Sample + Hold) $f_{CPU}$ = 8 MHz, speed = 0, $f_{ADC}$ = 2 MHz | | 7.5 | | | µs |
| $t_{ADC}$ | No. of sample capacitor loading cycles | | 4 | | | $1/f_{ADC}$ |
| | No. of hold conversion cycles | | 11 | | | |

1. Any added external serial resistor will downgrade the ADC accuracy (especially for resistance greater than 10kΩ). Data based on characterization results, not tested in production.
2. Injecting negative current on adjacent pins may result in increased leakage currents. Software filtering of the converted analog value is recommended.

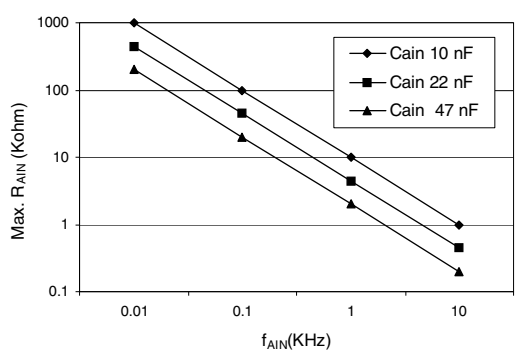
Obsolete Product(s) - Obsolete Product(s)

**Figure 95. $R_{AIN}$ maximum versus $f_{ADC}$ with $C_{AIN}$ = 0pF(1)**



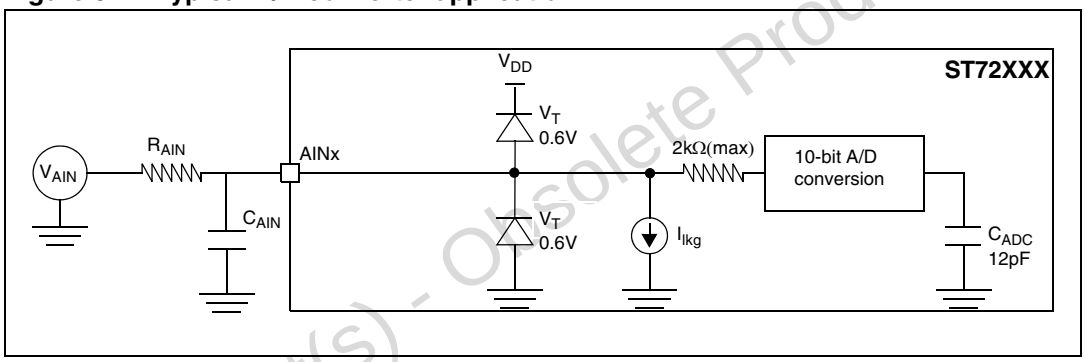

1. $C_{PARASITIC}$ represents the capacitance of the PCB (dependent on soldering and PCB layout quality) plus the pad capacitance (3pF). A high $C_{PARASITIC}$ value will downgrade conversion accuracy. To remedy this, $f_{ADC}$ should be reduced.

**Figure 96. Recommended $C_{AIN}$ and $R_{AIN}$ values(1)**




1. This graph shows that, depending on the input signal variation ($f_{AIN}$), $C_{AIN}$ can be increased for stabilization time and decreased to allow the use of a larger serial resistor ($R_{AIN}$).

**Figure 97. Typical A/D converter application**




## 19.12.1 Analog power supply and reference pins

Depending on the MCU pin count, the package may feature separate $V_{AREF}$ and $V_{SSA}$ analog power supply pins. These pins supply power to the A/D converter cell and function as the high and low reference voltages for the conversion.

Separation of the digital and analog power pins allow board designers to improve A/D performance. Conversion accuracy can be impacted by voltage drops and noise in the event of heavily loaded or badly decoupled power supply lines (see *Section 19.12.2: General PCB design guidelines*).

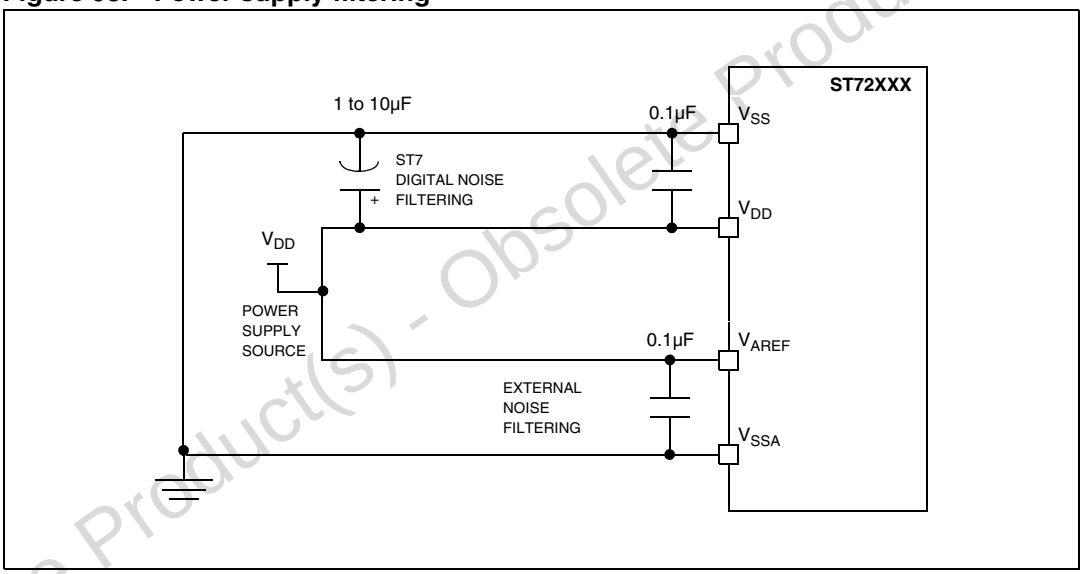
Obsolete product(s) - Obsolete Product(s)

### 19.12.2 General PCB design guidelines

To obtain best results, some general design and layout rules should be followed when designing the application PCB to shield the noise-sensitive, analog physical interface from noise-generating CMOS logic signals.

- Use separate digital and analog planes. The analog ground plane should be connected to the digital ground plane via a single point on the PCB.
- Filter power to the analog power planes. It is recommended to connect capacitors, with good high frequency characteristics, between the power and ground lines, placing 0.1µF and optionally, if needed 10pF capacitors as close as possible to the ST7 power supply pins and a 1 to 10µF capacitor close to the power source (see *Figure 98*).
- The analog and digital power supplies should be connected in a star network. Do not use a resistor, as $V_{AREF}$ is used as a reference voltage by the A/D converter and any resistance would cause a voltage drop and a loss of accuracy.
- Properly place components and route the signal traces on the PCB to shield the analog inputs. Analog signals paths should run over the analog ground plane and be as short as possible. Isolate analog signals from digital signals that may switch while the analog inputs are being sampled by the A/D converter. Do not toggle digital outputs on the same I/O port as the A/D input being converted.

**Figure 98. Power supply filtering**




Obsolete Product(s) - Obsolete Product(s)

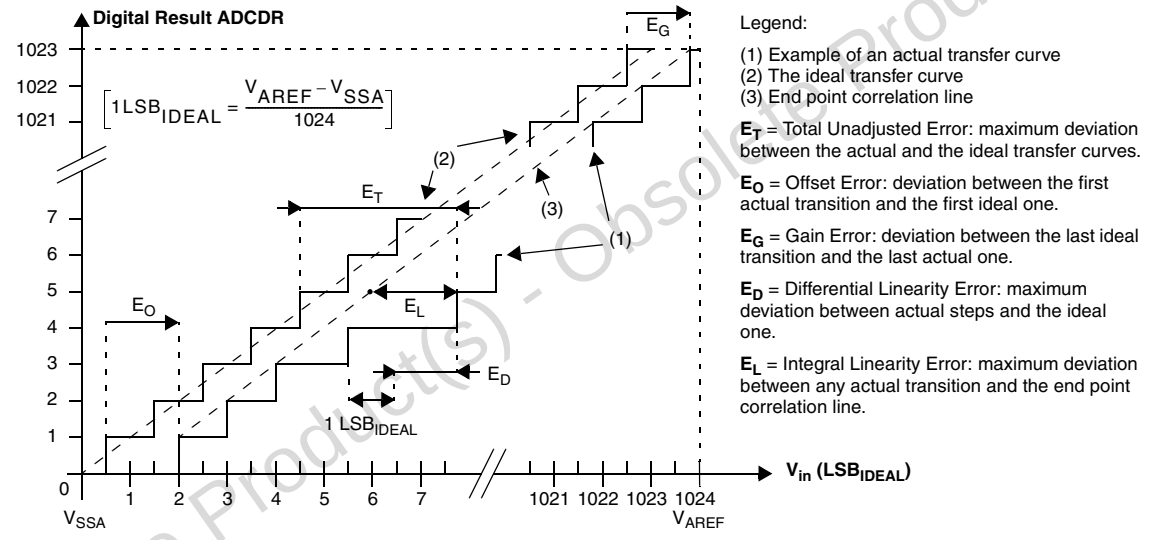
### 19.12.3 ADC accuracy

**Table 138. ADC accuracy**

| Symbol | Parameter[1] | Conditions | Typ | Max[2] | | Unit |
|---|---|---|---|---|---|---|
| | | | | 48/60 Kbyte Flash | 16/32 Kbyte Flash | |
| $\|E_T\|$ | Total unadjusted error | $V_{DD}$ = 5V[1] CPU in run mode @ $f_{ADC}$ 2 MHz | 3 | 4 | 6 | LSB |
| $\|E_O\|$ | Offset error | | 2 | 3 | 5 | |
| $\|E_G\|$ | Gain error | | 0.5 | 3 | 4.5 | |
| $\|E_D\|$ | Differential linearity error | | 1 | 2 | 2 | |
| $\|E_L\|$ | Integral linearity error | | | | 3 | |

1. ADC Accuracy versus Negative Injection Current: Injecting negative current may reduce the accuracy of the conversion being performed on another analog input. Any positive injection current within the limits specified for $I_{INJ(PIN)}$ and $\Sigma I_{INJ(PIN)}$ in *Section 19.8* does not affect the ADC accuracy.
2. Data based on characterization results, monitored in production to guarantee 99.73% within ± max value from -40°C to 125°C (± 3σ distribution limits).

**Figure 99. ADC error classification**




Obsolete Product(s) - Obsolete Product(s)

# 20 Package characteristics

## 20.1 Package mechanical data

**Figure 100. 64-pin (14x14) low profile quad flat package outline**

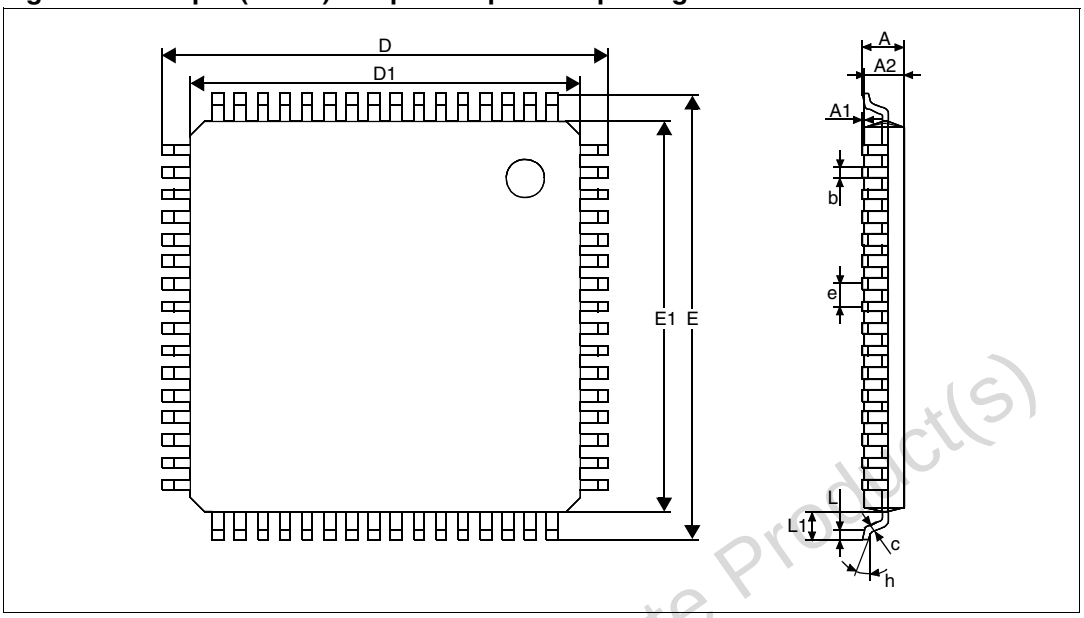



**Table 139. 64-pin (14x14) low profile quad flat package mechanical data**

| Dimension | mm | | | inches[1] | | |
|---|---|---|---|---|---|---|
| | Min | Typ | Max | Min | Typ | Max |
| A | | | 1.60 | | | 0.063 |
| A1 | 0.05 | | 0.15 | 0.002 | | 0.006 |
| A2 | 1.35 | 1.40 | 1.45 | 0.053 | 0.055 | 0.057 |
| b | 0.30 | 0.37 | 0.45 | 0.012 | 0.015 | 0.018 |
| c | 0.09 | | 0.20 | 0.004 | | 0.008 |
| D | | 16.00 | | | 0.630 | |
| D1 | | 14.00 | | | 0.551 | |
| E | | 16.00 | | | 0.630 | |
| E1 | | 14.00 | | | 0.551 | |
| e | | 0.80 | | | 0.031 | |
| θ | 0° | 3.5° | 7° | 0° | 3.5° | 7° |
| L | 0.45 | 0.60 | 0.75 | 0.018 | 0.024 | 0.030 |
| L1 | | 1.00 | | | 0.039 | |

1. Values in inches are converted from mm and rounded to 3 decimal digits.

Obsolete Product(s) - Obsolete Product(s)

**Figure 101. 64-pin (10x10) low profile quad flat package outline**

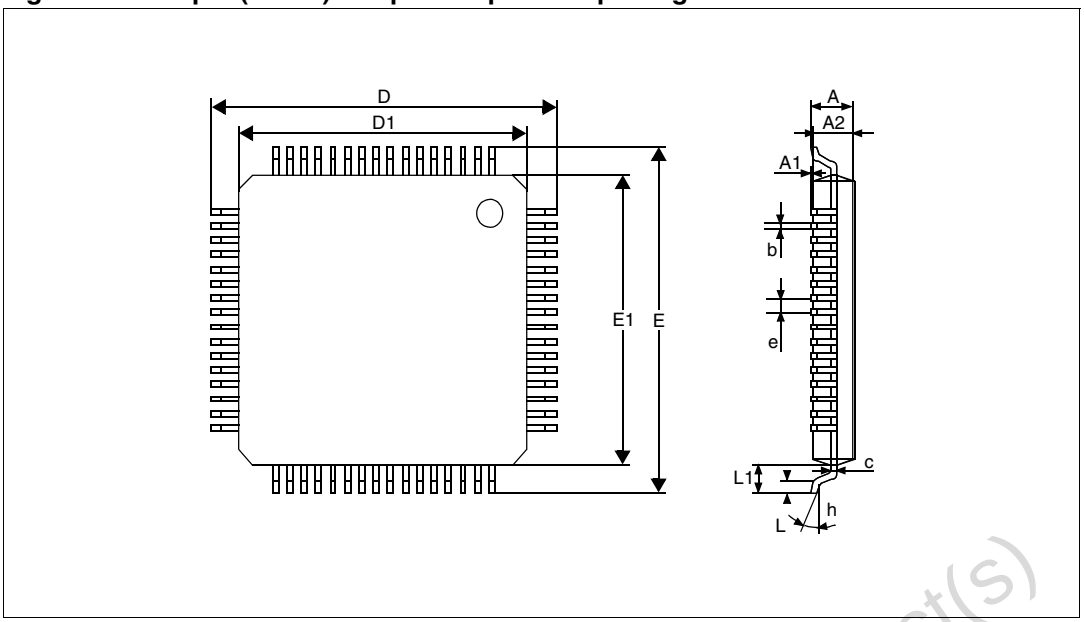



**Table 140. 64-pin (10x10) low profile quad flat package mechanical data**

| Dimension | mm | | | inches[1] | | |
|---|---|---|---|---|---|---|
| | Min | Typ | Max | Min | Typ | Max |
| A | | | 1.60 | | | 0.063 |
| A1 | 0.05 | | 0.15 | 0.002 | | 0.006 |
| A2 | 1.35 | 1.40 | 1.45 | 0.053 | 0.055 | 0.057 |
| b | 0.17 | 0.22 | 0.27 | 0.007 | 0.009 | 0.011 |
| c | 0.09 | | 0.20 | 0.004 | | 0.008 |
| D | | 12.00 | | | 0.472 | |
| D1 | | 10.00 | | | 0.394 | |
| E | | 12.00 | | | 0.472 | |
| E1 | | 10.00 | | | 0.394 | |
| e | | 0.50 | | | 0.020 | |
| θ | 0° | 3.5° | 7° | 0° | 3.5° | 7° |
| L | 0.45 | 0.60 | 0.75 | 0.018 | 0.024 | 0.030 |
| L1 | | 1.00 | | | 0.039 | |

1. Values in inches are converted from mm and rounded to 3 decimal digits.

Obsolete Product(s) - Obsolete Product(s)

**Figure 102. 44-pin (10x10) low profile quad flat package outline**

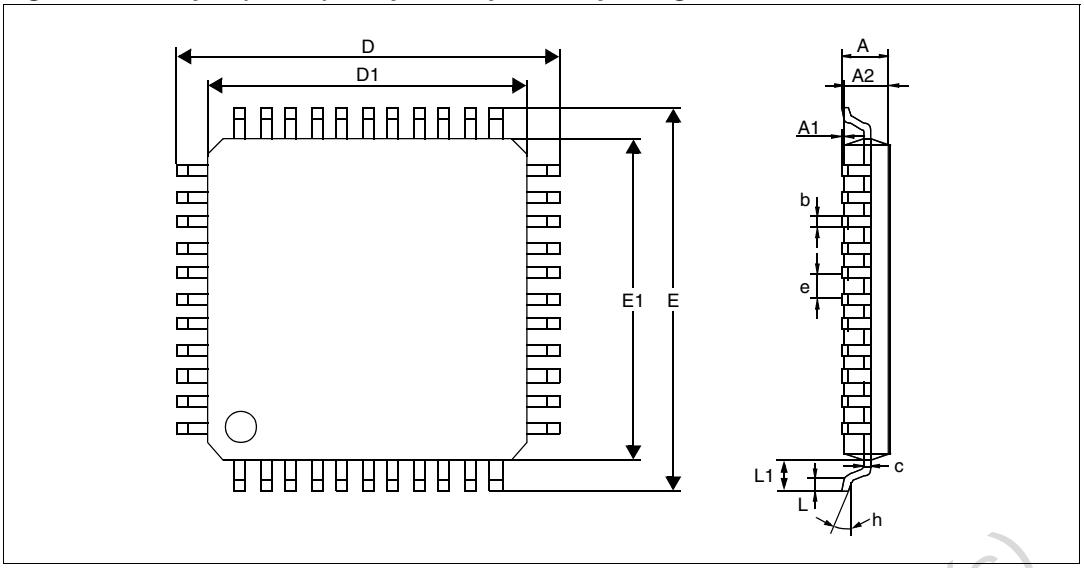



**Table 141. 44-pin (10x10) low profile quad flat package mechanical data**

| Dimension | mm | | | inches[1] | | |
|---|---|---|---|---|---|---|
| | Min | Typ | Max | Min | Typ | Max |
| A | | | 1.60 | | | 0.063 |
| A1 | 0.05 | | 0.15 | 0.002 | | 0.006 |
| A2 | 1.35 | 1.40 | 1.45 | 0.053 | 0.055 | 0.057 |
| b | 0.30 | 0.37 | 0.45 | 0.012 | 0.015 | 0.018 |
| C | 0.09 | | 0.20 | 0.004 | 0.000 | 0.008 |
| D | | 12.00 | | | 0.472 | |
| D1 | | 10.00 | | | 0.394 | |
| E | | 12.00 | | | 0.472 | |
| E1 | | 10.00 | | | 0.394 | |
| e | | 0.80 | | | 0.031 | |
| θ | 0° | 3.5° | 7° | 0° | 3.5° | 7° |
| L | 0.45 | 0.60 | 0.75 | 0.018 | 0.024 | 0.030 |
| L1 | | 1.00 | | | 0.039 | |

1. Values in inches are converted from mm and rounded to 3 decimal digits.

Obsolete Product(s) - Obsolete Product(s)

**Figure 103. 32-pin low profile quad flat package outline**

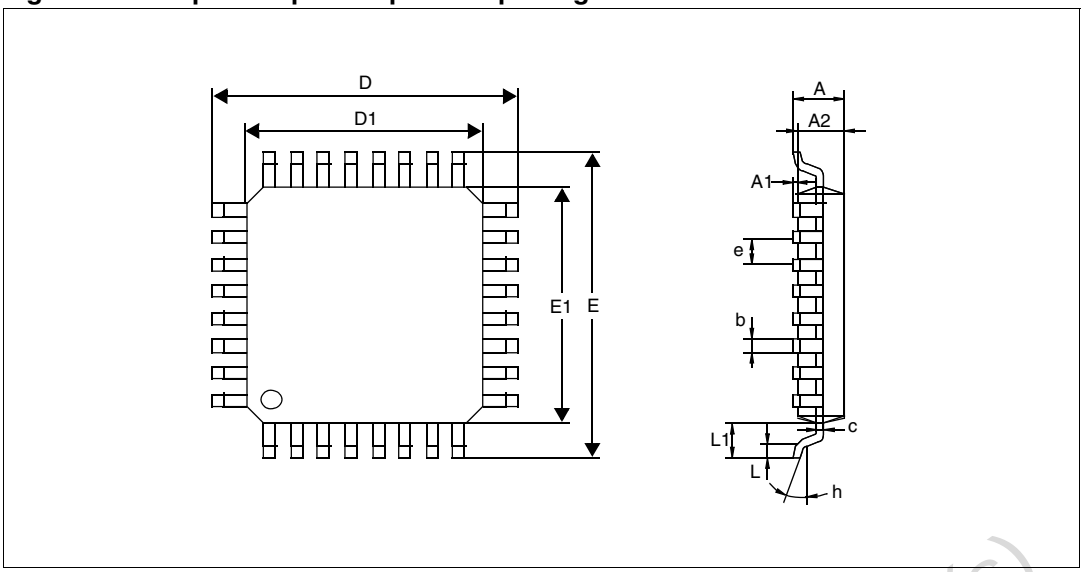



**Table 142. 32-pin low profile quad flat package mechanical data**

| Dim. | mm | | | inches[1] | | |
|---|---|---|---|---|---|---|
| | Min | Typ | Max | Min | Typ | Max |
| A | | | 1.60 | | | 0.063 |
| A1 | 0.05 | | 0.15 | 0.002 | | 0.006 |
| A2 | 1.35 | 1.40 | 1.45 | 0.053 | 0.055 | 0.057 |
| b | 0.30 | 0.37 | 0.45 | 0.012 | 0.015 | 0.018 |
| C | 0.09 | | 0.20 | 0.004 | | 0.008 |
| D | | 9.00 | | | 0.354 | |
| D1 | | 7.00 | | | 0.276 | |
| E | | 9.00 | | | 0.354 | |
| E1 | | 7.00 | | | 0.276 | |
| e | | 0.80 | | | 0.031 | |
| θ | 0° | 3.5° | 7° | 0° | 3.5° | 7° |
| L | 0.45 | 0.60 | 0.75 | 0.018 | 0.024 | 0.030 |
| L1 | | 1.00 | | | 0.039 | |

1. Values in inches are converted from mm and rounded to 3 decimal digits.

Obsolete Product(s) - Obsolete Product(s)

## 20.2 Thermal characteristics

**Table 143. Thermal characteristics**

| Symbol | Ratings | Value | Unit |
|---|---|---|---|
| $R_{thJA}$ | Package thermal resistance (junction to ambient)<br>LQFP64 14x14<br>LQFP64 10x10<br>LQFP44 10x10<br>LQFP32 7x7 | <br>47<br>50<br>52<br>70 | °C/W |
| $P_D$ | Power dissipation(1) | 500 | mW |
| $T_{Jmax}$ | Maximum junction temperature(2) | 150 | °C |

1. The maximum power dissipation is obtained from the formula $P_D = (T_J - T_A) / R_{thJA}$. The power dissipation of an application can be defined by the user with the formula: $P_D = P_{INT} + P_{PORT}$ where $P_{INT}$ is the chip internal power ($I_{DD} \times V_{DD}$) and $P_{PORT}$ is the port power dissipation depending on the ports used in the application.
2. The maximum chip-junction temperature is based on technology characteristics.

## 20.3 Ecopack information

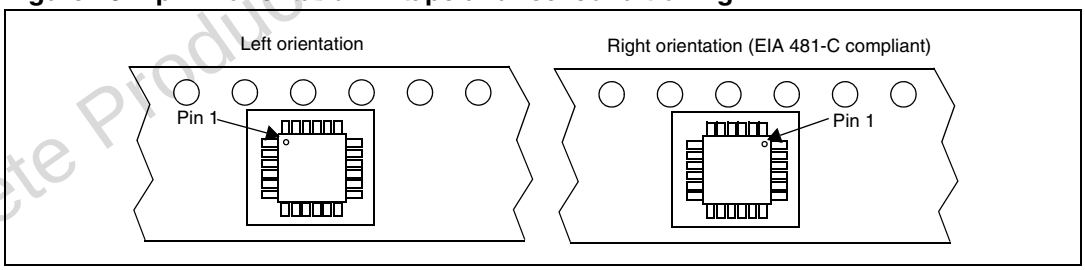
In order to meet environmental requirements, ST offers these devices in different grades of ECOPACK® packages, depending on their level of environmental compliance. ECOPACK® specifications, grade definitions and product status are available at: *www.st.com*. ECOPACK® is an ST trademark.

## 20.4 Packaging for automatic handling

The devices can be supplied in trays or with tape and reel conditioning.

Tape and reel conditioning can be ordered with pin 1 left-oriented or right-oriented when facing the tape sprocket holes as shown in *Figure 104*.

**Figure 104. pin 1 orientation in tape and reel conditioning**




See also *Figure 105: ST72F325xxx-Auto Flash commercial product structure on page 231* and *Figure 106: ST72P325xxx-Auto FastROM commercial product structure on page 232*.

Obsolete Product(s) - Obsolete Product(s)

# 21 Device configuration and ordering information

Each device is available for production in user programmable versions (Flash) as well as in factory coded versions (FASTROM).

ST72P325 devices are Factory Advanced Service Technique ROM (FASTROM) versions: They are factory-programmed HDFlash devices. Flash devices are shipped to customers with a default content, whereas ROM/FASTROM factory coded parts contain the code supplied by the customer. This implies that Flash devices have to be configured by the customer using the option bytes while the ROM/FASTROM devices are factory-configured.

Detailed device configuration and ordering information is presented in the following *Section 21.1: Flash devices* and *Section 21.2: FASTROM device ordering information and transfer of customer code*.

## 21.1 Flash devices

### 21.1.1 Flash configuration

**Table 144. Flash option bytes**

| | Static option byte 0 | | | | | | | | Static option byte 1 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | WDG | | CSS | VD | | Reserved | | FMP_R | PKG1 | RSTC | OSCTYPE | | OSCRANGE | | | PLLOFF |
| | HALT | SW | | 1 | 0 | | | | | | 1 | 0 | 2 | 1 | 0 | |
| Default value: | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | (1) | 1 | 1 | 0 | 0 | 1 | 1 | 1 |

1. Depends on device type as defined in *Table 147: Package selection (OPT7) on page 230*

The option bytes allow the hardware configuration of the microcontroller to be selected. They have no address in the memory map and can be accessed only in programming mode (for example, using a standard ST7 programming tool). The default content of the Flash is fixed to FFh. To program the Flash devices directly using ICP, Flash devices are shipped to customers with the internal RC clock source enabled. In masked ROM devices, the option bytes are fixed in hardware by the ROM code (see option list).

**Table 145. Option byte 0 bit description**

| Bit | Name | Function |
|---|---|---|
| OPT7 | WDG HALT | Watchdog and Halt mode<br>This option bit determines if a RESET is generated when entering Halt mode while the Watchdog is active.<br>0: No Reset generation when entering Halt mode<br>1: Reset generation when entering Halt mode |
| OPT6 | WDG SW | Hardware or software watchdog<br>This option bit selects the watchdog type.<br>0: Hardware (watchdog always enabled)<br>1: Software (watchdog to be enabled by software) |

Obsolete Product(s) - Obsolete Product(s)

**Table 145. Option byte 0 bit description (continued)**

| Bit | Name | Function |
|---|---|---|
| OPT5 | CSS | Clock security system on/off<br>This option bit enables or disables the clock security system (CSS) function which includes the clock filter and the backup safe oscillator.<br>0: CSS enabled<br>1: CSS disabled |
| OPT4:3 | VD[1:0] | Voltage detection<br>These option bits enable the voltage detection block (LVD and AVD) with a selected threshold for the LVD and AVD (EVD + AVD).<br>00: Selected LVD = Highest threshold ($V_{DD}$~4V)<br>01: Selected LVD = Medium threshold ($V_{DD}$~3.5V)<br>10: Selected LVD = Lowest threshold ($V_{DD}$~3V)<br>11: LVD and AVD off<br>**Caution**: If the medium or low thresholds are selected, the detection may occur outside the specified operating voltage range. Below 3.8V, device operation is not guaranteed. For details on the AVD and LVD threshold levels refer to *Section 19.3.2: Operating conditions with low voltage detector (LVD)*. |
| OPT2:1 | - | Reserved, must be kept at default value |
| OPT0 | FMP_R | Flash memory readout protection<br>Readout protection, when selected, provides a protection against program memory content extraction and against write access to Flash memory.<br>Erasing the option bytes when the FMP_R option is selected causes the whole user memory to be erased first, after which the device can be reprogrammed. Refer to *Section 4.3.1: Readout protection* and the *ST7 Flash Programming Reference Manual* for more details.<br>0: Readout protection enabled<br>1: Readout protection disabled |

**Table 146. Option byte 1 bit description**

| Bit | Name | Function |
|---|---|---|
| OPT7 | PKG1 | Package selection bit 1<br>This option bit selects the package (see *Table 147: Package selection (OPT7)*). |
| OPT6 | RSTC | RESET clock cycle selection<br>This option bit selects the number of CPU cycles applied during the RESET phase and when exiting Halt mode. For resonator oscillators, it is advised to select 4096 due to the long crystal stabilization time.<br>0: Reset phase with 4096 CPU cycles<br>1: Reset phase with 256 CPU cycles |
| OPT5:4 | OSCTYPE[1:0] | Oscillator type<br>These option bits select the ST7 main clock source type.<br>00: Clock source = Resonator oscillator<br>01: Reserved<br>10: Clock source = Internal RC oscillator<br>11: Clock source = External source |

Obsolete Product(s) - Obsolete Product(s)

**Table 146. Option byte 1 bit description (continued)**

| Bit | Name | Function |
|---|---|---|
| OPT3:1 | OSCRANGE[2:0] | Oscillator range<br>When the resonator oscillator type is selected, these option bits select the resonator oscillator current source corresponding to the frequency range of the resonator used. When the external clock source is selected, these bits are set to medium power (2 ~ 4 MHz).<br>000: Typ. frequency range = 1 ~ 2 MHz<br>001: Typ. frequency range = 2 ~ 4 MHz<br>010: Typ. frequency range = 4 ~ 8 MHz<br>011: Typ. frequency range = 8 ~ 16 MHz |
| OPT0 | PLLOFF | PLL activation<br>This option bit activates the PLL which allows multiplication by two of the main input clock frequency. The PLL is guaranteed only with an input frequency between 2 and 4 MHz. For this reason the PLL must not be used with the internal RC oscillator.<br>0: PLL x2 enabled<br>1: PLL x2 disabled<br>**Caution:** The PLL can be enabled only if the OSCRANGE (OPT3:1) bits are configured to 2 ~ 4 MHz. Otherwise, the device functionality is not guaranteed. |

**Table 147. Package selection (OPT7)**

| Version | Selected package | Flash size | PKG1 |
|---|---|---|---|
| R/AR | LQFP64 | 32/48/60 Kbytes | 1 |
| J | LQFP44 | 48/60 Kbytes | 0 |
| | | 16/32 Kbytes | 1 |
| K | LQFP32 | 16/32 Kbytes | 0 |

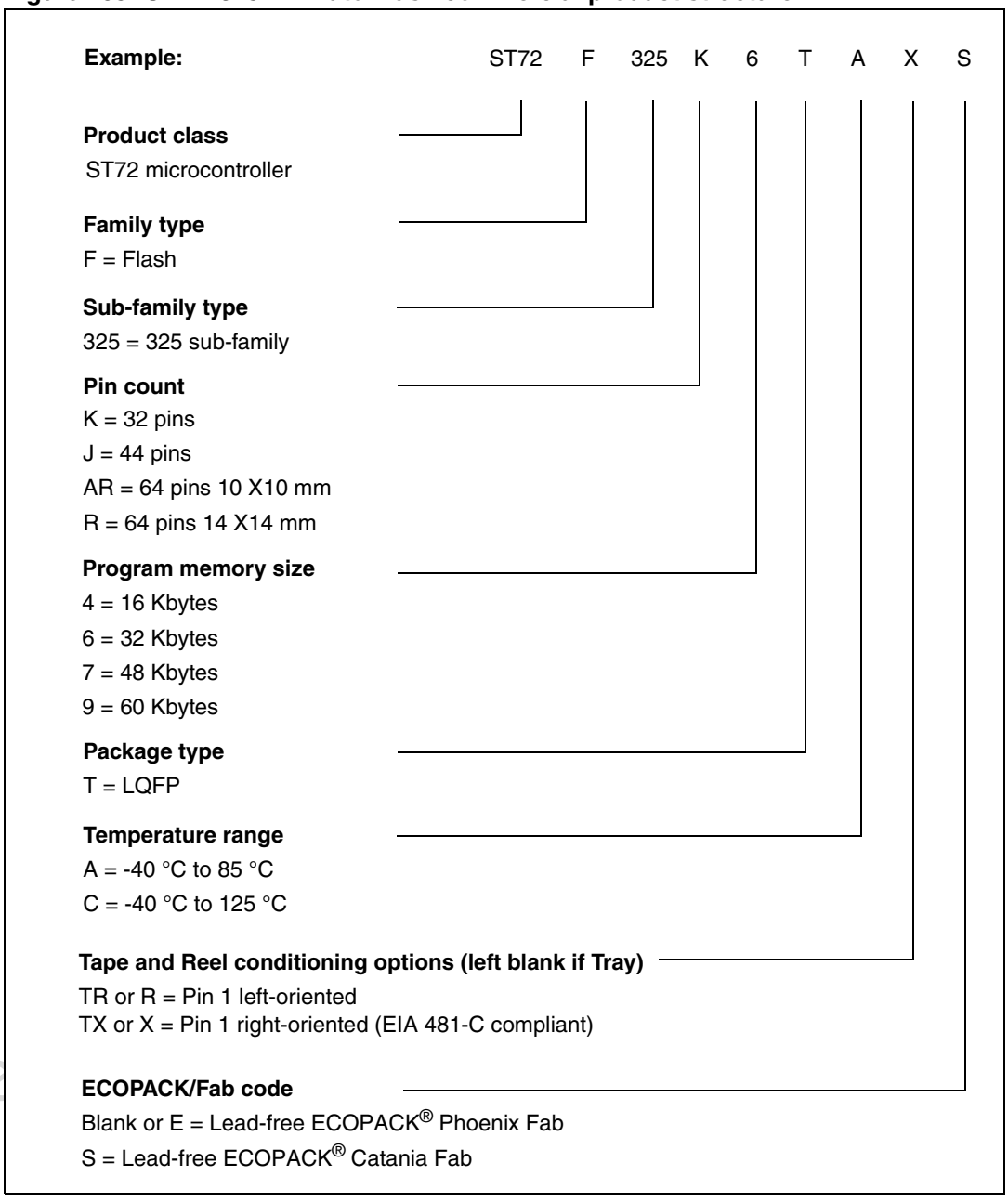
*Note: On the chip, each I/O port has up to eight pads. Pads that are not bonded to external pins are in input pull-up configuration after reset. The configuration of these pads must be kept at reset state to avoid added current consumption.*

Obsolete Product(s) - Obsolete Product(s)

### 21.1.2 Flash ordering information

The following *Figure 105* serves as a guide for ordering.

**Figure 105. ST72F325xxx-Auto Flash commercial product structure**




Obsole

1. For a list of available options (e.g. memory size, package) and orderable part numbers or for further information on any aspect of this device, please go to *www.st.com* or contact the ST Sales Office nearest to you.

## 21.2 FASTROM device ordering information and transfer of customer code

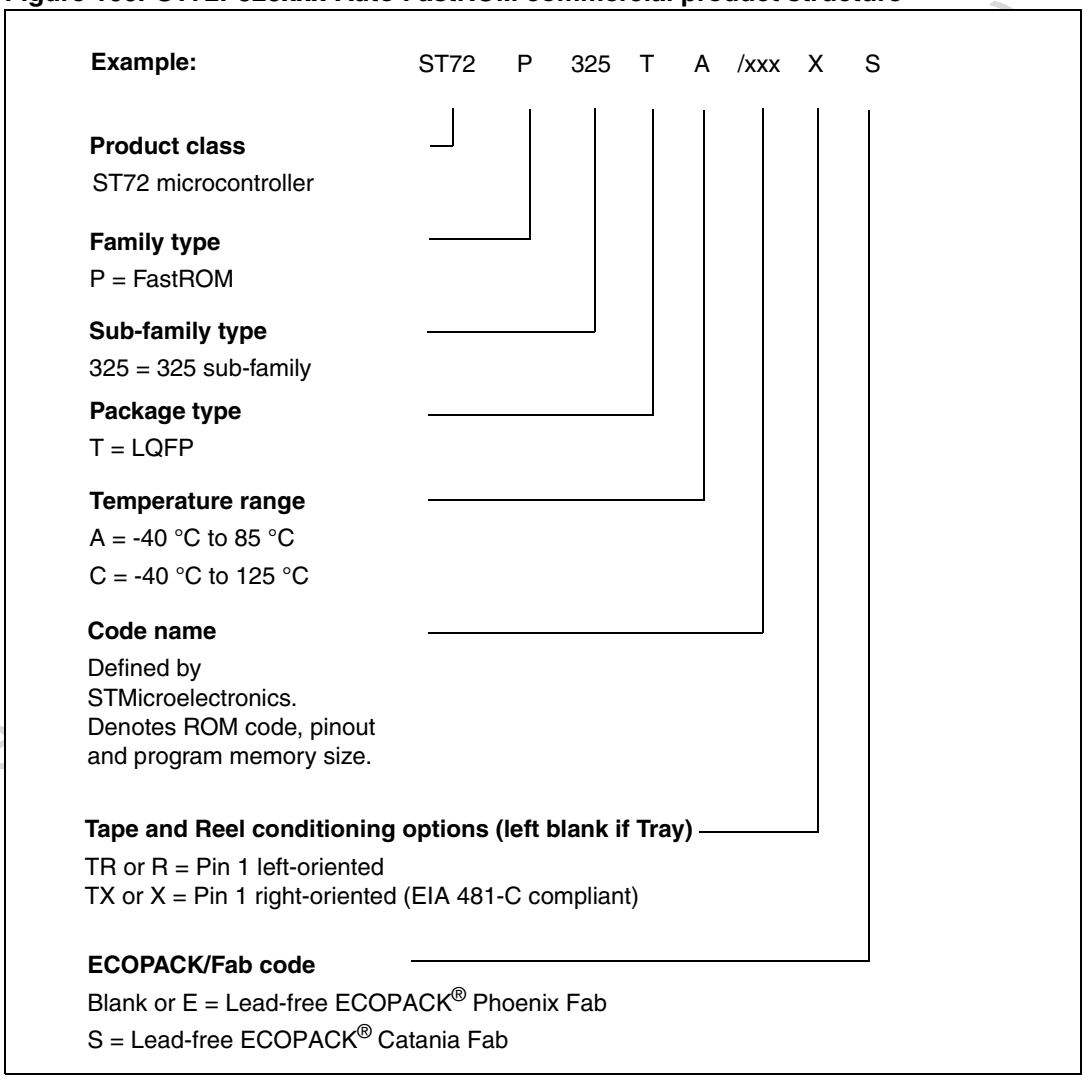
Customer code is made up of the FASTROM contents and the list of the selected options (if any). The FASTROM contents are to be sent on diskette, or by electronic means, with the S19 hexadecimal file generated by the development tool. All unused bytes must be set to FFh.

Complete the appended *ST72325-Auto Microcontroller FASTROM Option List on page 233* to communicate the selected options to STMicroelectronics and check for regular updates of the option list on the ST website or ask your ST representative.

Refer to application note AN1635 for information on the counter listing returned by ST after code has been transferred.

*Figure 105* serves as a guide for ordering. The STMicroelectronics Sales Organization will be pleased to provide detailed information on contractual points.

**Figure 106. ST72P325xxx-Auto FastROM commercial product structure**




Obsole

**ST72325-Auto Microcontroller FASTROM Option List**

(Last update: June 2010)

Customer: . . . . . .
Address: . . . . . .
. . . . . .
Contact: . . . . . .
Phone No: . . . . . .
Reference: . . . . . .

The FASTROM/ROM code name is assigned by STMicroelectronics)
FASTROM/ROM code must be sent in .S19 format. .Hex extension cannot be processed.

Device Type/Memory Size/Package (check only one option):

| FASTROM DEVICE: | 60K | 48K | 32K | 16K |
|---|---|---|---|---|
| LQFP64 14x14: | [ ] ST72P325R9T | [ ] ST72P325R7T | [ ] ST72P325R6T | |
| LQFP64 10x10: | [ ] ST72P325AR9T | [ ] ST72P325AR7T | [ ] ST72P325AR6T | |
| LQFP44 10x10: | [ ] ST72P325J9T | [ ] ST72P325J7T | [ ] ST72P325J6T | [ ] ST72P325J4T |
| LQFP32 7x7: | | | [ ] ST72P325K6T | [ ] ST72P325K4T |

Conditioning for LQFP package (check only one option):

[ ] Tape and Reel [ ] Tray

Temperature range:
[ ] A (-40°C to +85°C)
[ ] B (-40°C to +105°C)
[ ] C (-40°C to +125°C)

Special marking: [ ] No [ ] Yes "_ _ _ _ _ _ _ _ _ _ "
LQFP32 7 characters, other packages 10 characters max.
Authorized characters are letters, digits, '.', '-', '/' and spaces only.

Clock source selection:
[ ] Resonator:
[ ] LP: Low power resonator (1 to 2 MHz)
[ ] MP: Medium power resonator (2 to 4 MHz)
[ ] MS: Medium speed resonator (4 to 8 MHz)
[ ] HS: High speed resonator (8 to 16 MHz)
[ ] Internal RC
[ ] External clock (sets MP Medium Power resonator in Option Byte)

PLL(1)(2) [ ] Disabled [ ] Enabled

LVD reset [ ] Disabled [ ] High threshold [ ] Med. threshold [ ] Low threshold

Reset delay [ ] 256 Cycles [ ] 4096 Cycles

Watchdog selection [ ] Software activation [ ] Hardware activation

Halt when Watchdog on [ ] Reset [ ] No reset

Readout protection [ ] Disabled [ ] Enabled

Date . . . . . Signature . . . . . .

1. PLL must be disabled if internal RC network is selected.
2. The PLL can be enabled only if the resonator is configured to “Medium Power : 2~4 MHz”.

Please download the latest version of this option list from www.st.com.

Obsolete Product(s) - Obsolete Product(s)

## 21.3 Development tools

### 21.3.1 Introduction

Development tools for the ST7 microcontrollers include a complete range of hardware systems and software tools from STMicroelectronics and third-party tool suppliers. The range of tools includes solutions to help you evaluate microcontroller peripherals, develop and debug your application, and program your microcontrollers.

### 21.3.2 Evaluation tools and starter kits

ST offers complete, affordable starter kits and full-featured evaluation boards that allow you to evaluate microcontroller features and quickly start developing ST7 applications. Starter kits are complete, affordable hardware/software tool packages that include features and samples to help you quickly start developing your application. ST evaluation boards are open-design, embedded systems, which are developed and documented to serve as references for your application design. They include sample application software to help you demonstrate, learn about and implement your ST7's features.

### 21.3.3 Development and debugging tools

Application development for ST7 is supported by fully optimizing C Compilers and the ST7 Assembler-Linker toolchain, which are all seamlessly integrated in the ST7 integrated development environments in order to facilitate the debugging and fine-tuning of your application. The Cosmic C Compiler is available in a free version that outputs up to 16 Kbytes of code.

The range of hardware tools includes cost effective ST7-DVP3 series emulators. These tools are supported by the ST7 Toolset from STMicroelectronics, which includes the STVD7 integrated development environment (IDE) with high-level language debugger, editor, project manager and integrated programming interface.

### 21.3.4 Programming tools

During the development cycle, the ST7-DVP3 and ST7-EMU3 series emulators and the RLink provide in-circuit programming capability for programming the Flash microcontroller on your application board.

ST also provides a low-cost dedicated in-circuit programmer, the ST7-STICK, as well as ST7 socket boards which provide all the sockets required for programming any of the devices in a specific ST7 subfamily on a platform that can be used with any tool with in-circuit programming capability for ST7.

For production programming of ST7 devices, ST's third-party tool partners also provide a complete range of gang and automated programming solutions, which are ready to integrate into your production environment.

For additional ordering codes for spare parts, accessories and tools available for the ST7 (including from third party manufacturers), refer to the online product selector at www.st.com/mcu.

Obsolete Product(s) - Obsolete Product(s)

**Table 148. STMicroelectronics development tools**

| Supported products | Emulation | | | | Programming |
|---|---|---|---|---|---|
| | ST7 DVP3 series | | ST7 EMU3 series | | ICC socket board |
| | Emulator | Connection kit | Emulator | Active probe and T.E.B. | |
| ST72325AR, ST72F325AR | ST7MDT20-DVP3 | ST7MDT20-T6A/DVP | ST7MDT20M-EMU3 | ST7MDT20M-TEB | ST7SB20M/xx(1) |
| ST72325R, ST72F325R, | | ST7MDT20-T64/DVP | | | |
| ST72325J, ST72F325J | | ST7MDT20-T44/DVP | ST7MDT20J-EMU3 | ST7MDT20J-TEB | ST7SB20J/xx(1) |
| ST72325K, ST72F325K | | ST7MDT20-T32/DVP | | | |

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

**Table 149. Suggested list of socket types**

| Device | Socket | Emulator adapter |
|---|---|---|
| LQFP64 14 x14 | CAB 3303262 | CAB 3303351 |
| LQFP64 10 x10 | YAMAICHI IC149-064-*75-*5 | YAMAICHI ICP-064-6 |
| LQFP44 10 x10 | YAMAICHI IC149-044-*52-*5 | YAMAICHI ICP-044-5 |
| LQFP32 7 x 7 | IRONWOOD SF-QFE32SA-L-01 | IRONWOOD SK-UGA06/32A-01 |

### 21.3.5 Socket and emulator adapter information

For information on the type of socket that is supplied with the emulator, refer to the suggested list of sockets in *Table 149*.

*Note: Before designing the board layout, it is recommended to check the overall dimensions of the socket as they may be greater than the dimensions of the device.*

For footprint and other mechanical information about these sockets and adapters, refer to the manufacturer's datasheet.

#### Related documentation

*ST7 Visual Develop Software Key Debugging Features* (AN 978)

*ST7 Visual Develop for ST7 Cosmic C toolset users* (AN 1938)

*ST7 Visual Develop for ST7 Assembler Linker toolset users* (AN 1940)

## 21.4 ST7 application notes

All relevant ST7 application notes can be found on www.st.com.

Obsolete Product(s) Obsolete Product(s)

# 22 Known limitations

## 22.1 All devices

### 22.1.1 Unexpected reset fetch

If an interrupt request occurs while a “POP CC” instruction is executed, the interrupt controller does not recognize the source of the interrupt and, by default, passes the RESET vector address to the CPU.

#### Workaround

To solve this issue, a “POP CC” instruction must always be preceded by a “SIM” instruction.

### 22.1.2 External interrupt missed

To avoid any risk of generating a parasitic interrupt, the edge detector is automatically disabled for one clock cycle during an access to either DDR and OR. Any input signal edge during this period will not be detected and will not generate an interrupt.

This case can typically occur if the application refreshes the port configuration registers at intervals during runtime.

#### Workaround

The workaround is based on software checking the level on the interrupt pin before and after writing to the PxOR or PxDDR registers. If there is a level change (depending on the sensitivity programmed for this pin) the interrupt routine is invoked using the call instruction with three extra PUSH instructions before executing the interrupt routine (this is to make the call compatible with the IRET instruction at the end of the interrupt service routine).

But detection of the level change does not make sure that edge occurs during the critical 1 cycle duration and the interrupt has been missed. This may lead to occurrence of same interrupt twice (one hardware and another with software call).

To avoid this, a semaphore is set to ‘1’ before checking the level change. The semaphore is changed to level '0' inside the interrupt routine. When a level change is detected, the semaphore status is checked. If it is ‘1’, it means that the last interrupt has been missed. In this case, the interrupt routine is invoked with the call instruction.

There is another possible case, that is, if PxOR or PxDDR are written to with global interrupts disabled (interrupt mask bit set). In this case, the semaphore is changed to ‘1’ when the level change is detected. Detecting a missed interrupt is done after the global interrupts are enabled (interrupt mask bit reset) and by checking the status of the semaphore. If it is ‘1’, it means that the last interrupt was missed and the interrupt routine is invoked with the call instruction.

To implement the workaround, the following software sequence is to be followed for writing into the PxOR/PxDDR registers. The example is for Port PF1 with falling edge interrupt sensitivity. The software sequence is given for both cases (global interrupts disabled / global interrupts enabled):

**Case 1:** Writing to PxOR or PxDDR with global interrupts enabled:

```
LD A,#01
```

Obsolete Product(s) - Obsolete Product(s)

```
LD sema,A
; set the semaphore to '1'
LD A,PFDR

AND A,#02

LD  X,A
; store the level before writing to PxOR/PxDDR
LD A,#$90

LD PFDDR,A
; Write to PFDDR
LD A,#$ff

LD  PFOR,A
 ; Write to PFOR
LD A,PFDR

AND A,#02

LD Y,A
; store the level after writing to PxOR/PxDDR
LD A,X
; check for falling edge
cp A,#02

jrne OUT

TNZ Y

jrne OUT

LD A,sema
; check the semaphore status if edge is detected
CP A,#01

jrne OUT

call call_routine
; call the interrupt routine
OUT:LD A,#00

LD sema,A

.call_routine
; entry to call_routine
PUSH A

PUSH X

PUSH CC
```

Obsolete Product(s) - Obsolete Product(s)

```
.ext1_rt
; entry to interrupt routine
LD A,#00

LD sema,A

IRET
```

**Case 2:** Writing to PxOR or PxDDR with global interrupts disabled:

```
SIM
; set the interrupt mask
LD A,PFDR

AND A,#$02

LD  X,A
; store the level before writing to PxOR/PxDDR
LD A,#$90

LD PFDDR,A
; Write into PFDDR
LD A,#$ff

LD PFOR,A
          ; Write to PFOR
LD A,PFDR

AND A,#$02

LD Y,A
; store the level after writing to PxOR/PxDDR
LD A,X
; check for falling edge
cp A,#$02

jrne OUT

TNZ Y

jrne OUT

LD A,#$01

LD sema,A
; set the semaphore to '1' if edge is detected
RIM
 ; reset the interrupt mask
LD A,sema
; check the semaphore status
CP A,#$01
```

Obsolete Product(s) - Obsolete Product(s)

```
jrne OUT

call call_routine
; call the interrupt routine
RIM

OUT:
RIM
JP while_loop

.call_routine
; entry to call_routine
PUSH A

PUSH X

PUSH CC

.ext1_rt
; entry to interrupt routine
LD A,#$00

LD sema,A

IRET
```

### 22.1.3 Clearing active interrupts outside interrupt routine

When an active interrupt request occurs at the same time as the related flag is being cleared, an unwanted reset may occur.

*Note: Clearing the related interrupt mask will not generate an unwanted reset.*

**Concurrent interrupt context**

The symptom does not occur when the interrupts are handled normally, that is, when:

- The interrupt flag is cleared within its own interrupt routine
- The interrupt flag is cleared within any interrupt routine
- The interrupt flag is cleared in any part of the code while this interrupt is disabled

If these conditions are not met, the symptom can be avoided by implementing the following sequence:

Perform SIM and RIM operation before and after resetting an active interrupt request.

Example:

SIM
Reset interrupt flag
RIM

**Nested interrupt context**

Obsolete Product(s) Obsolete Product(s)

The symptom does not occur when the interrupts are handled normally, that is, when:

- The interrupt flag is cleared within its own interrupt routine
- The interrupt flag is cleared within any interrupt routine with higher or identical priority level
- The interrupt flag is cleared in any part of the code while this interrupt is disabled

If these conditions are not met, the symptom can be avoided by implementing the following sequence:

```
PUSH CC
SIM
Reset interrupt flag
POP CC
```

### 22.1.4 SCI wrong break duration

#### Description

A single break character is sent by setting and resetting the SBK bit in the SCICR2 register. In some cases, the break character may have a longer duration than expected:

- 20 bits instead of 10 bits if M = 0
- 22 bits instead of 11 bits if M = 1

In the same way, as long as the SBK bit is set, break characters are sent to the TDO pin. This may lead to generating one break more than expected.

#### Occurrence

The occurrence of the problem is random and proportional to the baud rate. With a transmit frequency of 19200 baud ($f_{CPU}$ = 8 MHz and SCIBRR = 0xC9), the wrong break duration occurrence is around 1%.

#### Workaround

If this wrong duration is not compliant with the communication protocol in the application, software can request that an Idle line be generated before the break character. In this case, the break duration is always correct assuming the application is not doing anything between the idle and the break. This can be ensured by temporarily disabling interrupts.

The exact sequence is:

- Disable interrupts
- Reset and Set TE (IDLE request)
- Set and Reset SBK (Break Request)
- Re-enable interrupts

### 22.1.5 16-bit timer PWM mode

In PWM mode, the first PWM pulse is missed after writing the value FFFCh in the OC1R register (OC1HR, OC1LR). It leads to either full or no PWM during a period, depending on the OLVL1 and OLVL2 settings.

Obsolete Product(s) - Obsolete Product(s)

### 22.1.6 TIMD set simultaneously with OC interrupt

If the 16-bit timer is disabled at the same time the output compare event occurs, the output compare flag then gets locked and cannot be cleared before the timer is enabled again.

#### Impact on the application

If the output compare interrupt is enabled, then the output compare flag cannot be cleared in the timer interrupt routine. Consequently, the interrupt service routine is called repeatedly.

#### Workaround

Disable the timer interrupt before disabling the timer. Again while enabling, first enable the timer, then the timer interrupts.

- Perform the following to disable the timer:
  - TACR1 or TBCR1 = 0x00h; // Disable the compare interrupt
  - TACSR | or TBCSR | = 0x40; // Disable the timer
- Perform the following to enable the timer again:
  - TACSR & or TBCSR & = ~0x40; // Enable the timer
  - TACR1 or TBCR1 = 0x40; // Enable the compare interrupt

### 22.1.7 I2C multimaster

In multimaster configurations, if the ST7 I2C receives a START condition from another I2C master after the START bit is set in the I2CCR register and before the START condition is generated by the ST7 I2C, it may ignore the START condition from the other I2C master. In this case, the ST7 master will receive a NACK from the other device. On reception of the NACK, ST7 can send a restart and Slave address to re-initiate communication.

### 22.1.8 Pull-up always active on PE2

The I/O port internal pull-up is always active on I/O port E2. As a result, if PE2 is in output mode low level, current consumption in Halt/Active Halt mode is increased.

Obsolete Product(s) - Obsolete Product(s)

# 23 Revision history

**Table 150. Document revision history**

| Date | Revision | Changes |
|---|---|---|
| 30-Aug-2007 | 1 | Initial release |
| 22-Jun-2010 | 2 | Updated *Section 20.3: Ecopack information on page 227*<br>Added *Section 20.4: Packaging for automatic handling on page 227*<br>Updated *Section 21: Device configuration and ordering information on page 228* |
| 05-Aug-2010 | 2 | Corrected partnumber in caption of *Figure 105* and *Figure 106* |

Obsolete Product(s) - Obsolete Product(s)

**Please Read Carefully:**

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST’s terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

**UNLESS OTHERWISE SET FORTH IN ST’S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.**

**UNLESS EXPRESSLY APPROVED IN WRITING BY AN AUTHORIZED ST REPRESENTATIVE, ST PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. ST PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER’S OWN RISK.**

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.

Information in this document supersedes and replaces all information previously supplied.

The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

© 2010 STMicroelectronics - All rights reserved

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Philippines - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

**www.st.com**