

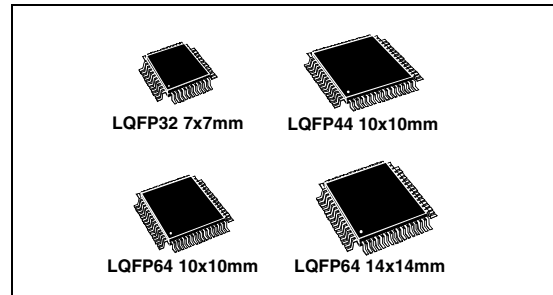


ST72561xx-Auto

8-bit MCU for automotive with Flash or ROM,
10-bit ADC, 5 timers, SPI, LINSCI™, active CAN

Features

- Memories
 - 16 K to 60 K High Density Flash (HDFlash) or ROM with read-out protection capability. In-application programming and in-circuit programming for HDFlash devices
 - 1 to 2 K RAM
 - HDFlash endurance: 100 cycles, data retention 20 years at 55 °C
- Clock, reset and supply management
 - Low power crystal/ceramic resonator oscillators and bypass for external clock
 - PLL for 2 x frequency multiplication
 - 5 power saving modes: halt, auto wake up from halt, active halt, wait and slow
- Interrupt management
 - Nested interrupt controller
 - 14 interrupt vectors plus TRAP and RESET
 - TLI top level interrupt (on 64-pin devices)
 - Up to 21 ext. interrupt lines (on 4 vectors)
- Up to 48 I/O ports
 - Up to 48 multifunctional bidirectional I/Os
 - Up to 36 alternate I/O functions
 - Up to 6 high sink outputs
- 5 timers
 - 16-bit timer with 2 input captures, 2 output compares, external clock input, PWM and pulse generator modes
 - 8-bit timer with 1 or 2 input captures, 1 or 2 output compares, PWM and pulse generator modes
 - 8-bit PWM auto-reload timer with 1 or 2 input captures, 2 or 4 independent PWM output channels, output compare and time base interrupt, external clock with event detector
 - Main clock controller with real-time base and clock output
 - Window watchdog timer



- Up to 4 communications interfaces
 - SPI synchronous serial interface
 - Master/ slave LINSCI™ asynchronous serial interface
 - Master only LINSCI™ asynchronous serial interface
 - CAN 2.0B active
- Analog peripheral (low current coupling)
 - 10-bit A/D converter with up to 16 inputs
 - Up to 9 robust ports (low current coupling)
- Instruction set
 - 8-bit data manipulation
 - 63 basic instructions, 17 main addressing modes
 - 8 x 8 unsigned multiply instruction
- Development tools
 - Full hardware/ software development package

Table 1. Device summary

| Reference | Part number |
|----------------|--|
| ST72561xx-Auto | ST72561K4-Auto, ST72561K6-Auto, ST72561K7-Auto, ST72561K9-Auto, ST72561J4-Auto, ST72561J6-Auto, ST72561J7-Auto, ST72561J9-Auto, ST72561R4-Auto, ST72561R6-Auto, ST72561R7-Auto, ST72561R9-Auto, ST72561AR4-Auto, ST72561AR6-Auto, ST72561AR7-Auto, ST72561AR9-Auto |

Contents

| | | |
|----------|---|-----------|
| 1 | Description | 20 |
| 1.1 | Pin description | 22 |
| 2 | Register and memory map | 28 |
| 3 | Flash program memory | 32 |
| 3.1 | Introduction | 32 |
| 3.2 | Main features | 32 |
| 3.3 | Structure | 32 |
| 3.3.1 | Read-out protection | 33 |
| 3.4 | ICC interface | 33 |
| 3.5 | ICP (in-circuit programming) | 34 |
| 3.6 | IAP (in-application programming) | 35 |
| 3.7 | Related documentation | 35 |
| 3.8 | Register description | 35 |
| 4 | Central processing unit | 36 |
| 4.1 | Introduction | 36 |
| 4.2 | Main features | 36 |
| 4.3 | CPU registers | 36 |
| 4.3.1 | Accumulator (A) | 36 |
| 4.3.2 | Index registers (X and Y) | 36 |
| 4.3.3 | Program counter (PC) | 36 |
| 4.3.4 | Condition code register (CC) | 37 |
| 4.3.5 | Stack pointer (SP) | 39 |
| 5 | Supply, reset and clock management | 41 |
| 5.1 | Introduction | 41 |
| 5.2 | Main features | 41 |
| 5.3 | Phase locked loop | 41 |
| 5.4 | Multi-oscillator (MO) | 42 |
| 5.5 | Reset sequence manager (RSM) | 43 |
| 5.5.1 | Introduction | 43 |

| | | |
|----------|---|-----------|
| 5.5.2 | Asynchronous external RESET pin | 44 |
| 5.5.3 | External power-on reset | 44 |
| 5.5.4 | Internal low voltage detector (LVD) reset | 44 |
| 5.5.5 | Internal watchdog reset | 45 |
| 5.6 | System integrity management (SI) | 45 |
| 5.6.1 | Low voltage detector (LVD) | 45 |
| 5.6.2 | Auxiliary voltage detector (AVD) | 46 |
| 5.6.3 | Low power modes | 47 |
| 5.6.4 | Interrupts | 47 |
| 5.6.5 | Register description | 48 |
| 6 | Interrupts | 50 |
| 6.1 | Introduction | 50 |
| 6.2 | Masking and processing flow | 50 |
| 6.3 | Interrupts and low power modes | 53 |
| 6.4 | Concurrent & nested management | 53 |
| 6.5 | Interrupt register description | 54 |
| 6.5.1 | CPU CC register interrupt bits | 54 |
| 6.5.2 | Interrupt software priority registers (ISPRX) | 55 |
| 6.6 | External interrupts | 58 |
| 6.6.1 | I/O port interrupt sensitivity | 58 |
| 6.6.2 | Register description | 60 |
| 7 | Power saving modes | 63 |
| 7.1 | Introduction | 63 |
| 7.2 | Slow mode | 63 |
| 7.3 | Wait mode | 64 |
| 7.4 | Halt mode | 65 |
| 7.5 | Active halt mode | 67 |
| 7.6 | Auto wake-up from halt mode | 68 |
| 7.6.1 | Register description | 71 |
| 8 | I/O ports | 73 |
| 8.1 | Introduction | 73 |
| 8.2 | Functional description | 73 |
| 8.2.1 | Input modes | 73 |

| | | |
|-----------|---|-----------|
| 8.2.2 | Output modes | 74 |
| 8.2.3 | Alternate functions | 74 |
| 8.3 | I/O port implementation | 77 |
| 8.4 | I/O port register configurations | 77 |
| 8.4.1 | Standard ports | 77 |
| 8.4.2 | Interrupt ports | 78 |
| 8.4.3 | Pull-up input port (CANTX requirement) | 79 |
| 8.5 | Low power modes | 80 |
| 8.6 | Interrupts | 80 |
| 9 | Window watchdog (WWDG) | 82 |
| 9.1 | Introduction | 82 |
| 9.2 | Main features | 82 |
| 9.3 | Functional description | 82 |
| 9.4 | Using halt mode with the WDG | 84 |
| 9.5 | How to program the watchdog timeout | 84 |
| 9.6 | Low power modes | 86 |
| 9.7 | Hardware watchdog option | 86 |
| 9.8 | Using halt mode with the WDG (WDGHALT option) | 87 |
| 9.9 | Interrupts | 87 |
| 9.10 | Register description | 87 |
| 9.10.1 | Control register (WDGCR) | 87 |
| 9.10.2 | Window Register (WDGWR) | 87 |
| 10 | Main clock controller with real time clock MCC/RTC | 89 |
| 10.1 | Programmable CPU clock prescaler | 89 |
| 10.2 | Clock-out capability | 89 |
| 10.3 | Real time clock timer (RTC) | 89 |
| 10.4 | Low power modes | 90 |
| 10.5 | Interrupts | 90 |
| 10.6 | Register description | 90 |
| 10.6.1 | MCC control/status register (MCCSR) | 90 |
| 11 | PWM auto-reload timer (ART) | 93 |
| 11.1 | Introduction | 93 |

| | | |
|-----------|--|------------|
| 11.2 | Functional description | 94 |
| 11.2.1 | Counter | 94 |
| 11.2.2 | Counter clock and prescaler | 94 |
| 11.2.3 | Counter and prescaler Initialization | 94 |
| 11.2.4 | Output compare control | 94 |
| 11.2.5 | Independent PWM signal generation | 95 |
| 11.2.6 | Output compare and Time base interrupt | 96 |
| 11.2.7 | External clock and event detector mode | 96 |
| 11.2.8 | Input capture function | 96 |
| 11.2.9 | External interrupt capability | 98 |
| 11.3 | Register description | 99 |
| 12 | 16-bit timer | 104 |
| 12.1 | Introduction | 104 |
| 12.2 | Main features | 104 |
| 12.3 | Functional description | 105 |
| 12.3.1 | Counter | 105 |
| 12.3.2 | External clock | 107 |
| 12.3.3 | Input capture | 108 |
| 12.3.4 | Procedure | 109 |
| 12.3.5 | Output compare | 110 |
| 12.3.6 | Procedure | 111 |
| 12.3.7 | Forced compare output capability | 112 |
| 12.3.8 | One pulse mode | 113 |
| 12.3.9 | Pulse width modulation mode | 115 |
| 12.4 | Low power modes | 117 |
| 12.5 | Interrupts | 117 |
| 12.6 | Summary of timer modes | 118 |
| 12.7 | Register description | 118 |
| 12.7.1 | Control register 1 (CR1) | 118 |
| 12.7.2 | Control register 2 (CR2) | 119 |
| 12.7.3 | Control/status register (CSR) | 120 |
| 12.7.4 | Input capture 1 high register (IC1HR) | 121 |
| 12.7.5 | Input capture 1 low register (IC1LR) | 122 |
| 12.7.6 | Output compare 1 high register (OC1HR) | 122 |
| 12.7.7 | Output compare 1 low register (OC1LR) | 122 |

| | | | |
|-----------|---------|--|------------|
| | 12.7.8 | Output compare 2 high register (OC2HR) | 122 |
| | 12.7.9 | Output compare 2 low register (OC2LR) | 123 |
| | 12.7.10 | Counter high register (CHR) | 123 |
| | 12.7.11 | Counter low register (CLR) | 123 |
| | 12.7.12 | Alternate counter high register (ACHR) | 123 |
| | 12.7.13 | Alternate counter low register (ACLR) | 124 |
| | 12.7.14 | Input capture 2 high register (IC2HR) | 124 |
| | 12.7.15 | Input capture 2 low register (IC2LR) | 124 |
| 13 | | 8-bit timer (TIM8) | 126 |
| | 13.1 | Introduction | 126 |
| | 13.2 | Main features | 126 |
| | 13.3 | Functional description | 126 |
| | 13.3.1 | Counter | 126 |
| | 13.3.2 | Input capture | 130 |
| | 13.3.3 | Output compare | 131 |
| | 13.3.4 | Forced compare output capability | 133 |
| | 13.3.5 | One pulse mode | 134 |
| | 13.3.6 | Pulse width modulation mode | 136 |
| | 13.4 | Low power modes | 138 |
| | 13.5 | Interrupts | 138 |
| | 13.6 | Summary of timer modes | 139 |
| | 13.7 | Register description | 139 |
| | 13.7.1 | Control register 1 (CR1) | 139 |
| | 13.7.2 | Control register 2 (CR2) | 140 |
| | 13.7.3 | Control/status register (CSR) | 141 |
| | 13.7.4 | Input capture 1 register (IC1R) | 142 |
| | 13.7.5 | Output compare 1 register (OC1R) | 142 |
| | 13.7.6 | Output compare 2 register (OC2R) | 143 |
| | 13.7.7 | Counter register (CTR) | 143 |
| | 13.7.8 | Alternate counter register (ACTR) | 143 |
| | 13.7.9 | Input capture 2 register (IC2R) | 143 |
| | 13.8 | 8-bit timer register map | 144 |
| 14 | | Serial peripheral interface (SPI) | 145 |
| | 14.1 | Introduction | 145 |

| | | |
|-----------|---|------------|
| 14.2 | Main features | 145 |
| 14.3 | General description | 145 |
| 14.3.1 | Functional description | 146 |
| 14.3.2 | Slave select management | 147 |
| 14.3.3 | Master mode operation | 148 |
| 14.3.4 | Master mode transmit sequence | 148 |
| 14.3.5 | Slave mode operation | 149 |
| 14.3.6 | Slave mode transmit sequence | 149 |
| 14.4 | Clock phase and clock polarity | 149 |
| 14.5 | Error flags | 150 |
| 14.5.1 | Master mode fault (MODF) | 150 |
| 14.5.2 | Overrun condition (OVR) | 151 |
| 14.5.3 | Write collision error (WCOL) | 151 |
| 14.6 | Low power modes | 153 |
| 14.7 | Interrupts | 154 |
| 14.8 | Register description | 154 |
| 14.8.1 | Control register (SPICR) | 154 |
| 14.8.2 | Control/status register (SPICSR) | 155 |
| 14.8.3 | Data I/O register (SPIDR) | 157 |
| 15 | LINSCI serial communication interface (LIN master/slave) | 158 |
| 15.1 | Introduction | 158 |
| 15.2 | SCI features | 158 |
| 15.3 | LIN features | 159 |
| 15.4 | General description | 159 |
| 15.5 | SCI mode - functional description | 160 |
| 15.5.1 | Conventional baud rate generator mode | 160 |
| 15.5.2 | Extended prescaler mode | 161 |
| 15.5.3 | Serial data format | 161 |
| 15.5.4 | Transmitter | 161 |
| 15.5.5 | Receiver | 163 |
| 15.5.6 | Extended baud rate generation | 165 |
| 15.5.7 | Receiver muting and wake-up feature | 166 |
| 15.5.8 | Parity control | 167 |
| 15.6 | Low power modes | 168 |
| 15.7 | Interrupts | 168 |

| | | |
|-----------|--|------------|
| 15.8 | SCI mode register description | 169 |
| 15.8.1 | Status register (SCISR) | 169 |
| 15.8.2 | Control register 1 (SCICR1) | 170 |
| 15.8.3 | Control register 2 (SCICR2) | 171 |
| 15.8.4 | Data register (SCIDR) | 172 |
| 15.8.5 | Baud rate register (SCIBRR) | 173 |
| 15.8.6 | Extended receive prescaler division register (SCIERPR) | 174 |
| 15.8.7 | Extended transmit prescaler division register (SCIETPR) | 175 |
| 15.9 | LIN mode - functional description. | 175 |
| 15.9.1 | Entering LIN mode | 175 |
| 15.9.2 | LIN transmission | 176 |
| 15.9.3 | LIN reception | 177 |
| 15.9.4 | LIN error detection | 179 |
| 15.9.5 | LIN baud rate | 182 |
| 15.9.6 | LIN slave baud rate generation | 182 |
| 15.9.7 | LINSPI clock tolerance | 183 |
| 15.9.8 | Clock deviation causes | 184 |
| 15.9.9 | Error due to LIN synch measurement | 185 |
| 15.9.10 | Error due to baud rate quantization | 185 |
| 15.9.11 | Impact of clock deviation on maximum baud rate | 185 |
| 15.10 | LIN mode register description | 186 |
| 15.10.1 | Status register (SCISR) | 186 |
| 15.10.2 | Control Register 1 (SCICR1) | 187 |
| 15.10.3 | Control Register 2 (SCICR2) | 187 |
| 15.10.4 | Control register 3 (SCICR3) | 188 |
| 15.10.5 | LIN divider registers | 190 |
| 15.10.6 | LIN prescaler register (LPR) | 190 |
| 15.10.7 | LIN prescaler fraction register (LPFR) | 190 |
| 15.10.8 | LIN header length register (LHLR) | 192 |
| 16 | LINSPI serial communication interface (LIN master only) | 195 |
| 16.1 | Introduction | 195 |
| 16.2 | Main features | 195 |
| 16.3 | General description | 196 |
| 16.4 | Functional description | 197 |
| 16.4.1 | Serial data format | 198 |

| | | |
|-----------|---|------------|
| 16.4.2 | Transmitter | 198 |
| 16.4.3 | Receiver | 200 |
| 16.4.4 | Conventional baud rate generation | 202 |
| 16.4.5 | Extended baud rate generation | 203 |
| 16.4.6 | Receiver muting and wake-up feature | 203 |
| 16.4.7 | Parity control | 204 |
| 16.5 | Low power modes | 205 |
| 16.6 | Interrupts | 205 |
| 16.7 | SCI synchronous transmission | 205 |
| 16.8 | Register description | 207 |
| 16.8.1 | Status register (SCISR) | 207 |
| 16.8.2 | Control register 1 (SCICR1) | 209 |
| 16.8.3 | Control register 2 (SCICR2) | 210 |
| 16.8.4 | Control Register 3 (SCICR3) | 211 |
| 16.8.5 | Data register (SCIDR) | 212 |
| 16.8.6 | Baud rate register (SCIBRR) | 213 |
| 16.8.7 | Extended receive prescaler division register (SCIERPR) | 214 |
| 16.8.8 | Extended transmit prescaler division register (SCIETPR) | 214 |
| 17 | beCAN controller (beCAN) | 216 |
| 17.1 | Main features | 216 |
| 17.2 | General description | 216 |
| 17.3 | Operating modes | 218 |
| 17.4 | Functional description | 221 |
| 17.4.1 | Transmission handling | 221 |
| 17.4.2 | Reception handling | 222 |
| 17.4.3 | Identifier filtering | 224 |
| 17.4.4 | Message storage | 227 |
| 17.4.5 | Error management | 229 |
| 17.4.6 | Bit timing | 230 |
| 17.5 | Interrupts | 232 |
| 17.6 | Register access protection | 232 |
| 17.7 | beCAN cell limitations | 233 |
| 17.7.1 | FIFO corruption | 233 |
| 17.8 | Register description | 238 |
| 17.8.1 | Control and status registers | 238 |

| | | |
|-----------|---|------------|
| 17.8.2 | CAN transmit status register (CTSR) | 240 |
| 17.8.3 | Mailbox registers | 248 |
| 17.8.4 | CAN filter registers | 251 |
| 18 | 10-bit A/D converter (ADC) | 259 |
| 18.1 | Introduction | 259 |
| 18.2 | Main features | 259 |
| 18.3 | Functional description | 259 |
| 18.3.1 | Digital A/D conversion result | 259 |
| 18.3.2 | A/D conversion | 260 |
| 18.3.3 | Changing the conversion channel | 261 |
| 18.3.4 | ADCDR consistency | 261 |
| 18.4 | Low power modes | 261 |
| 18.5 | Interrupts | 261 |
| 18.6 | Register description | 261 |
| 18.6.1 | Control/status register (ADCCSR) | 261 |
| 18.6.2 | Data register (ADCDRH) | 263 |
| 18.6.3 | Data register (ADCDDL) | 263 |
| 19 | Instruction set | 264 |
| 19.1 | CPU addressing modes | 264 |
| 19.1.1 | Inherent | 265 |
| 19.1.2 | Immediate | 266 |
| 19.1.3 | Direct | 266 |
| 19.1.4 | Indexed (no offset, short, long) | 266 |
| 19.1.5 | Indirect (short, long) | 266 |
| 19.1.6 | Indirect indexed (short, long) | 267 |
| 19.1.7 | Relative mode (direct, indirect) | 268 |
| 19.2 | Instruction groups | 268 |
| 19.2.1 | Using a prebyte | 269 |
| 20 | Electrical characteristics | 272 |
| 20.1 | Parameter conditions | 272 |
| 20.1.1 | Minimum and maximum values | 272 |
| 20.1.2 | Typical values | 272 |
| 20.1.3 | Typical curves | 272 |

| | | |
|---------|--|-----|
| 20.1.4 | Loading capacitor | 272 |
| 20.1.5 | Pin input voltage | 272 |
| 20.2 | Absolute maximum ratings | 273 |
| 20.2.1 | Voltage characteristics | 273 |
| 20.2.2 | Current characteristics | 274 |
| 20.2.3 | Thermal characteristics | 274 |
| 20.3 | Operating conditions | 275 |
| 20.3.1 | General operating conditions | 275 |
| 20.3.2 | Operating conditions with low voltage detector (LVD) | 275 |
| 20.3.3 | Auxiliary voltage detector (AVD) thresholds | 276 |
| 20.4 | Supply current characteristics | 276 |
| 20.4.1 | Supply and clock managers | 277 |
| 20.4.2 | On-chip peripherals | 278 |
| 20.5 | Clock and timing characteristics | 279 |
| 20.5.1 | Crystal and ceramic resonator oscillators | 280 |
| 20.5.2 | PLL characteristics | 281 |
| 20.6 | Auto wakeup from halt oscillator (AWU) | 282 |
| 20.7 | Memory characteristics | 282 |
| 20.7.1 | RAM and hardware registers | 282 |
| 20.7.2 | Flash memory | 282 |
| 20.8 | EMC characteristics | 283 |
| 20.8.1 | Functional EMS (electromagnetic susceptibility) | 283 |
| 20.8.2 | Electromagnetic interference (EMI) | 284 |
| 20.8.3 | Absolute maximum ratings (electrical sensitivity) | 284 |
| 20.9 | I/O port pin characteristics | 286 |
| 20.9.1 | General characteristics | 286 |
| 20.9.2 | Output driving current | 288 |
| 20.10 | Control pin characteristics | 290 |
| 20.10.1 | Asynchronous RESET pin | 290 |
| 20.10.2 | ICCSEL/ VPP pin | 292 |
| 20.11 | Timer peripheral characteristics | 293 |
| 20.12 | Communication interface characteristics | 295 |
| 20.12.1 | SPI - serial peripheral interface | 295 |
| 20.12.2 | CAN - controller area network interface | 297 |
| 20.13 | 10-bit ADC characteristics | 297 |

| | | |
|-----------|---|------------|
| 21 | Package characteristics | 302 |
| 21.1 | ECOPACK® | 302 |
| 21.2 | Package mechanical data | 302 |
| 21.3 | Thermal characteristics | 304 |
| 21.4 | Packaging for automatic handling | 304 |
| 22 | Device configuration and ordering information | 306 |
| 22.1 | Introduction | 306 |
| 22.2 | Flash devices | 306 |
| 22.2.1 | Flash configuration | 306 |
| 22.2.2 | Flash ordering information | 310 |
| 22.3 | Transfer of customer code | 311 |
| 23 | Development tools | 314 |
| 24 | Important notes | 315 |
| 24.1 | All devices | 315 |
| 24.1.1 | RESET pin protection with LVD enabled | 315 |
| 24.1.2 | Clearing active interrupts outside interrupt routine | 315 |
| 24.1.3 | External interrupt missed | 316 |
| 24.1.4 | Unexpected reset fetch | 318 |
| 24.1.5 | Header time-out does not prevent wake-up from mute mode | 318 |
| 24.1.6 | CAN FIFO corruption | 319 |
| 24.2 | Flash/FastROM devices only | 320 |
| 24.2.1 | LINSCI wrong break duration | 320 |
| 24.2.2 | 16-bit and 8-bit timer PWM mode | 321 |
| 24.3 | ROM devices only | 321 |
| 24.3.1 | 16-bit timer PWM mode buffering feature change | 321 |
| 25 | Revision history | 322 |

List of tables

| | | |
|-----------|--|-----|
| Table 1. | Device summary | 1 |
| Table 2. | Product overview | 20 |
| Table 3. | Device pin description | 25 |
| Table 4. | Hardware register map | 28 |
| Table 5. | Sectors available in Flash devices | 32 |
| Table 6. | Flash control/status register address and reset value | 35 |
| Table 7. | Interrupt software priority selection | 38 |
| Table 8. | ST7 clock sources | 43 |
| Table 9. | Effect of low power modes on SI | 47 |
| Table 10. | Interrupt control/wake-up capability | 47 |
| Table 11. | Reset source flags | 49 |
| Table 12. | Interrupt software priority levels | 51 |
| Table 13. | Interrupt software priority levels | 54 |
| Table 14. | Interrupt priority bits | 55 |
| Table 15. | Dedicated interrupt instruction set | 55 |
| Table 16. | Interrupt mapping | 57 |
| Table 17. | Interrupt sensitivity - ei3 | 60 |
| Table 18. | Interrupt sensitivity - ei2 | 60 |
| Table 19. | Interrupt sensitivity - ei1 | 60 |
| Table 20. | Interrupt sensitivity - ei0 | 61 |
| Table 21. | Nested interrupts register map and reset values | 62 |
| Table 22. | MCC/RTC low power mode selection | 67 |
| Table 23. | AWUPR prescaler | 71 |
| Table 24. | AWU register map and reset values | 72 |
| Table 25. | DR register value and output pin status | 74 |
| Table 26. | I/O port mode options | 75 |
| Table 27. | I/O port configurations | 76 |
| Table 28. | Configuration of PB7:6, PC0, PC3, PC7:5, PD3:2, PD5, PE7:0, PF7:0 | 77 |
| Table 29. | Configuration of PA0, 2, 4, 6; PB0, 2,4; PC1; PD0,6 (with pull-up) | 78 |
| Table 30. | Configuration of PA1, 3, 5, 7; PB1,3,5; PC2; PD1, 4, 7 (without pull-up) | 78 |
| Table 31. | Configuration of PC4 | 79 |
| Table 32. | Port configuration | 79 |
| Table 33. | Effect of low power modes on I/O ports | 80 |
| Table 34. | I/O port interrupt control/wake-up capability | 80 |
| Table 35. | I/O port register map and reset values | 81 |
| Table 36. | Effect of low power modes on WDG | 86 |
| Table 37. | Watchdog timer register map and reset values | 88 |
| Table 38. | Effect of low power modes on MCC/RTC | 90 |
| Table 39. | MCC/RTC Interrupt control wake-up capability | 90 |
| Table 40. | CPU clock frequency in SLOW mode | 91 |
| Table 41. | Time base selection | 91 |
| Table 42. | Main clock controller register map and reset values | 92 |
| Table 43. | Counter clock control | 99 |
| Table 44. | PWM frequency vs resolution | 100 |
| Table 45. | PWMx output level and polarity | 101 |
| Table 46. | PWM auto-reload timer register map and reset values | 102 |
| Table 47. | Effect of low power modes on 16-bit timer | 117 |
| Table 48. | Timer interrupt control and wake-up capability | 117 |

| | | |
|------------|--|-----|
| Table 49. | Timer modes | 118 |
| Table 50. | Clock control bits | 120 |
| Table 51. | 16-bit timer register map | 124 |
| Table 52. | Effect of low power modes on TIM8 | 138 |
| Table 53. | TIM8 interrupt control and wake-up capability | 138 |
| Table 54. | Timer modes | 139 |
| Table 55. | Clock control bits | 141 |
| Table 56. | Effect of low power modes on SPI | 153 |
| Table 57. | SPI interrupt control and wake-up capability | 154 |
| Table 58. | SPI master mode SCK frequency | 155 |
| Table 59. | SPI register map and reset values | 157 |
| Table 60. | Character formats | 167 |
| Table 61. | Effect of low power modes on SCI | 168 |
| Table 62. | SCI interrupt control and wake-up capability | 168 |
| Table 63. | PR prescaler | 173 |
| Table 64. | Transmitter rate divider | 173 |
| Table 65. | Receiver rate divider | 174 |
| Table 66. | LIN mode configuration | 188 |
| Table 67. | LDIV mantissa | 190 |
| Table 68. | LDIV fraction | 191 |
| Table 69. | LHL mantissa coding | 192 |
| Table 70. | LHL fraction coding | 193 |
| Table 71. | LINSCI1 register map and reset values | 194 |
| Table 72. | Frame formats | 204 |
| Table 73. | Effect of low power modes on SCI | 205 |
| Table 74. | SCI interrupt control and wake-up capability | 205 |
| Table 75. | LIN sync break duration | 211 |
| Table 76. | SCI clock on SCLK pin | 212 |
| Table 77. | PR prescaler | 213 |
| Table 78. | Transmitter rate divider | 213 |
| Table 79. | Receiver rate divider | 214 |
| Table 80. | Baud rate selection | 215 |
| Table 81. | LINSCI2 register map and reset values | 215 |
| Table 82. | Transmit mailbox mapping | 228 |
| Table 83. | Receive mailbox mapping | 228 |
| Table 84. | While loop timing | 236 |
| Table 85. | LEC error types | 243 |
| Table 86. | Filter page selection | 247 |
| Table 87. | beCAN control and status page - register map and reset values | 257 |
| Table 88. | beCAN mailbox pages - register map and reset values | 257 |
| Table 89. | beCAN filter configuration page - register map and reset values | 258 |
| Table 90. | Effect of low power modes on ADC | 261 |
| Table 91. | A/D clock selection | 262 |
| Table 92. | ADC channel selection | 262 |
| Table 93. | ADC register map and reset values | 263 |
| Table 94. | Addressing mode groups | 264 |
| Table 95. | CPU addressing mode overview | 264 |
| Table 96. | Instructions supporting direct, indexed, indirect and indirect indexed addressing (part 1) | 267 |
| Table 97. | Instructions supporting direct, indexed, indirect and indirect indexed addressing (part 2) | 267 |
| Table 98. | Instruction groups | 268 |
| Table 99. | Supply current consumption | 277 |
| Table 100. | Clock source current consumption | 278 |

| | | |
|------------|---|-----|
| Table 101. | Peripheral consumption | 278 |
| Table 102. | General timings | 279 |
| Table 103. | External clock source | 279 |
| Table 104. | Oscillator characteristics | 280 |
| Table 105. | PLL characteristics | 281 |
| Table 106. | AWU oscillator characteristics | 282 |
| Table 107. | RAM supply voltage | 282 |
| Table 108. | Dual voltage HDFlash memory | 282 |
| Table 109. | EMS test results | 284 |
| Table 110. | EMI emissions | 284 |
| Table 111. | Absolute maximum ratings | 285 |
| Table 112. | Electrical sensitivities | 285 |
| Table 113. | I/O characteristics | 286 |
| Table 114. | Output driving current | 288 |
| Table 115. | $\overline{\text{RESET}}$ pin characteristics | 290 |
| Table 116. | ICCSEL/ V_{PP} pin characteristics | 292 |
| Table 117. | 8-bit PWM-ART auto reload timer characteristics | 293 |
| Table 118. | 8-bit timer characteristics | 293 |
| Table 119. | 16-bit timer characteristics | 293 |
| Table 120. | SPI characteristics | 295 |
| Table 121. | CAN characteristics | 297 |
| Table 122. | ADC characteristics | 297 |
| Table 123. | ADC accuracy with $f_{\text{CPU}} = 8 \text{ MHz}$, $f_{\text{ADC}} = 4 \text{ MHz}$ $R_{\text{AIN}} < 10\text{k}\Omega$, $V_{\text{DD}} = 5\text{V}$ | 300 |
| Table 124. | Package selection | 307 |
| Table 125. | Alternate function remapping 1 | 308 |
| Table 126. | Alternate function remapping 0 | 308 |
| Table 127. | OSCTYPE selection | 308 |
| Table 128. | OSCRANGE selection | 308 |
| Table 129. | Document revision history | 322 |

List of figures

| | | |
|------------|---|-----|
| Figure 1. | Device block diagram | 21 |
| Figure 2. | LQFP 64-pin package pinout | 22 |
| Figure 3. | LQFP 44-pin package pinout | 23 |
| Figure 4. | LQFP 32-pin package pinout | 24 |
| Figure 5. | Memory map. | 28 |
| Figure 6. | Memory map and sector address | 33 |
| Figure 7. | Typical ICC interface | 34 |
| Figure 8. | CPU registers | 37 |
| Figure 9. | Stack manipulation example. | 40 |
| Figure 10. | PLL block diagram | 41 |
| Figure 11. | Clock, reset and supply block diagram | 42 |
| Figure 12. | RESET sequence phases. | 44 |
| Figure 13. | Reset block diagram | 44 |
| Figure 14. | Reset sequences | 45 |
| Figure 15. | Low voltage detector vs reset | 46 |
| Figure 16. | Using the AVD to monitor VDD | 47 |
| Figure 17. | Interrupt processing flowchart | 51 |
| Figure 18. | Priority decision process | 51 |
| Figure 19. | Concurrent interrupt management | 53 |
| Figure 20. | Nested interrupt management | 54 |
| Figure 21. | External interrupt control bits | 59 |
| Figure 22. | Power saving mode transitions. | 63 |
| Figure 23. | SLOW mode clock transitions. | 64 |
| Figure 24. | WAIT mode flow-chart | 65 |
| Figure 25. | HALT timing overview. | 66 |
| Figure 26. | HALT mode flow-chart | 66 |
| Figure 27. | ACTIVE HALT timing overview. | 68 |
| Figure 28. | ACTIVE HALT mode flow-chart | 68 |
| Figure 29. | AWUFH mode block diagram | 69 |
| Figure 30. | AWUFH halt timing diagram | 70 |
| Figure 31. | AWUFH mode flow-chart | 70 |
| Figure 32. | I/O port general block diagram | 75 |
| Figure 33. | Interrupt I/O port state transitions | 77 |
| Figure 34. | Watchdog block diagram | 83 |
| Figure 35. | Approximate timeout duration. | 84 |
| Figure 36. | Exact timeout duration (t _{min} and t _{max}) | 85 |
| Figure 37. | Window watchdog timing diagram | 86 |
| Figure 38. | Main clock controller (MCC/RTC) block diagram | 89 |
| Figure 39. | PWM auto-reload timer block diagram | 93 |
| Figure 40. | Output compare control | 95 |
| Figure 41. | PWM auto-reload timer function | 95 |
| Figure 42. | PWM signal from 0% to 100% duty cycle. | 96 |
| Figure 43. | External event detector example (3 counts) | 96 |
| Figure 44. | Input capture timing diagram, f _{COUNTER} = f _{CPU} | 97 |
| Figure 45. | Input capture timing diagram, f _{COUNTER} = f _{CPU} / 4 | 98 |
| Figure 46. | ART external interrupt in halt mode | 98 |
| Figure 47. | Timer block diagram | 106 |
| Figure 48. | 16-bit read sequence: (from counter or alternate counter register) | 106 |

| | | |
|-------------|--|-----|
| Figure 49. | Counter timing diagram, internal clock divided by 2 | 108 |
| Figure 50. | Counter timing diagram, internal clock divided by 4 | 108 |
| Figure 51. | Counter timing diagram, internal clock divided by 8 | 108 |
| Figure 52. | Input capture block diagram | 110 |
| Figure 53. | Input capture timing diagram | 110 |
| Figure 54. | Output compare block diagram | 112 |
| Figure 55. | Output compare timing diagram, fTIMER = fCPU/2 | 113 |
| Figure 56. | Output compare timing diagram, fTIMER = fCPU/4 | 113 |
| Figure 57. | One pulse mode timing example | 115 |
| Figure 58. | Pulse width modulation mode timing example with 2 output compare functions | 115 |
| Figure 59. | Timer block diagram | 128 |
| Figure 60. | Counter timing diagram, internal clock divided by 2 | 129 |
| Figure 61. | Counter timing diagram, internal clock divided by 4 | 129 |
| Figure 62. | Counter timing diagram, internal clock divided by 8 | 129 |
| Figure 63. | Input capture block diagram | 131 |
| Figure 64. | Input capture timing diagram | 131 |
| Figure 65. | Output compare block diagram | 133 |
| Figure 66. | Output compare timing diagram, fTIMER = fCPU/2 | 133 |
| Figure 67. | Output compare timing diagram, fTIMER = fCPU/4 | 134 |
| Figure 68. | One pulse mode timing example | 136 |
| Figure 69. | Pulse width modulation mode timing example | 136 |
| Figure 70. | Serial peripheral interface block diagram | 146 |
| Figure 71. | Single master/ single slave application | 147 |
| Figure 72. | Generic SS timing diagram | 147 |
| Figure 73. | Hardware/software slave select management | 148 |
| Figure 74. | Data clock timing diagram | 150 |
| Figure 75. | Clearing the WCOL bit (write collision flag) software sequence | 152 |
| Figure 76. | Single master / multiple slave configuration | 153 |
| Figure 77. | SCI block diagram (in conventional baud rate generator mode) | 160 |
| Figure 78. | Word length programming | 161 |
| Figure 79. | SCI baud rate and extended prescaler block diagram | 166 |
| Figure 80. | LIN characters | 176 |
| Figure 81. | SCI block diagram in LIN slave mode | 177 |
| Figure 82. | LIN header reception timeout | 180 |
| Figure 83. | LIN synch field measurement | 181 |
| Figure 84. | LDIV read / write operations when LDUM = 0 | 183 |
| Figure 85. | LDIV read / write operations when LDUM = 1 | 183 |
| Figure 86. | Bit sampling in reception mode | 184 |
| Figure 87. | LSF bit set and clear | 189 |
| Figure 88. | SCI block diagram | 197 |
| Figure 89. | Word length programming | 198 |
| Figure 90. | SCI baud rate and extended prescaler block diagram | 202 |
| Figure 91. | SCI example of synchronous and asynchronous transmission | 206 |
| Figure 92. | SCI data clock timing diagram (M = 0) | 206 |
| Figure 93. | SCI data clock timing diagram (M = 1) | 207 |
| Figure 94. | CAN network topology | 217 |
| Figure 95. | CAN block diagram | 218 |
| Figure 96. | beCAN operating modes | 218 |
| Figure 97. | beCAN in silent mode | 220 |
| Figure 98. | beCAN in loop back mode | 220 |
| Figure 99. | beCAN in combined mode | 221 |
| Figure 100. | Transmit mailbox states | 222 |

| | |
|---|-----|
| Figure 101. Receive FIFO states | 223 |
| Figure 102. Filter bank scale configuration - register organization | 225 |
| Figure 103. Filtering mechanism - example | 227 |
| Figure 104. CAN error state diagram | 229 |
| Figure 105. Bit timing | 230 |
| Figure 106. CAN frames (part 1/2). | 231 |
| Figure 107. CAN frames (part 2/2). | 231 |
| Figure 108. Event flags and interrupt generation | 232 |
| Figure 109. FIFO corruption | 234 |
| Figure 110. Workaround 1 | 234 |
| Figure 111. Critical window timing diagram | 236 |
| Figure 112. Reception of a sequence of frames | 236 |
| Figure 113. Reception at maximum CAN baud rate | 237 |
| Figure 114. Workaround 2 | 237 |
| Figure 115. CAN register mapping | 255 |
| Figure 116. Page mapping for CAN | 256 |
| Figure 117. ADC block diagram | 260 |
| Figure 118. Pin loading conditions | 272 |
| Figure 119. Pin input voltage | 273 |
| Figure 120. fCPU maximum vs V_{DD} | 275 |
| Figure 121. LVD startup behavior | 276 |
| Figure 122. Typical application with an external clock source | 279 |
| Figure 123. Typical application with a crystal or ceramic resonator | 280 |
| Figure 124. PLL jitter vs signal frequency ⁽¹⁾ | 281 |
| Figure 125. AWU oscillator freq. @ TA 25°C | 282 |
| Figure 126. Connecting unused I/O pins | 287 |
| Figure 127. RPU vs VDD with VIN = VSS | 287 |
| Figure 128. IPU vs VDD with VIN = VSS | 287 |
| Figure 129. Typical VOL at VDD = 5V (standard) | 288 |
| Figure 130. Typical VOL at VDD = 5V (high-sink) | 288 |
| Figure 131. Typical VOH at VDD = 5V | 289 |
| Figure 132. Typical VOL vs VDD (standard I/Os) | 289 |
| Figure 133. Typical VOL vs VDD (high-sink I/Os) | 289 |
| Figure 134. Typical VOH vs VDD | 290 |
| Figure 135. RESET pin protection when LVD is disabled | 291 |
| Figure 136. RESET pin protection when LVD is enabled | 291 |
| Figure 137. RESET RPU vs VDD | 292 |
| Figure 138. Two typical applications with ICCSEL/VPP pin | 292 |
| Figure 139. SPI slave timing diagram with CPHA = 0 | 296 |
| Figure 140. SPI slave timing diagram with CPHA = 1 | 296 |
| Figure 141. SPI master timing diagram | 297 |
| Figure 142. RAIN max vs fADC with CAIN = 0pF | 298 |
| Figure 143. Recommended CAIN/RAIN values | 299 |
| Figure 144. Typical application with ADC | 299 |
| Figure 145. Power supply filtering | 300 |
| Figure 146. ADC accuracy | 301 |
| Figure 147. 64-pin low profile quad flat package (14x14) | 302 |
| Figure 148. 32-pin low profile quad flat package (7x7) | 303 |
| Figure 149. 44-pin low profile quad flat package (10x10) | 303 |
| Figure 150. 64-pin low profile quad flat package (10 x10) | 304 |
| Figure 151. pin 1 orientation in tape and reel conditioning | 305 |
| Figure 152. ST72F561xx-Auto Flash commercial product structure | 310 |

| | |
|---|-----|
| Figure 153. ST72P561xxx-Auto FastROM commercial product structure | 311 |
| Figure 154. ST72561xx-Auto ROM commercial product structure | 312 |
| Figure 155. Header reception event sequence | 319 |
| Figure 156. LINSI interrupt routine | 319 |

1 Description

The ST72561xx-Auto devices are members of the ST7 microcontroller family designed for automotive mid-range applications with CAN (Controller Area Network) and LIN (Local Interconnect Network) interface.

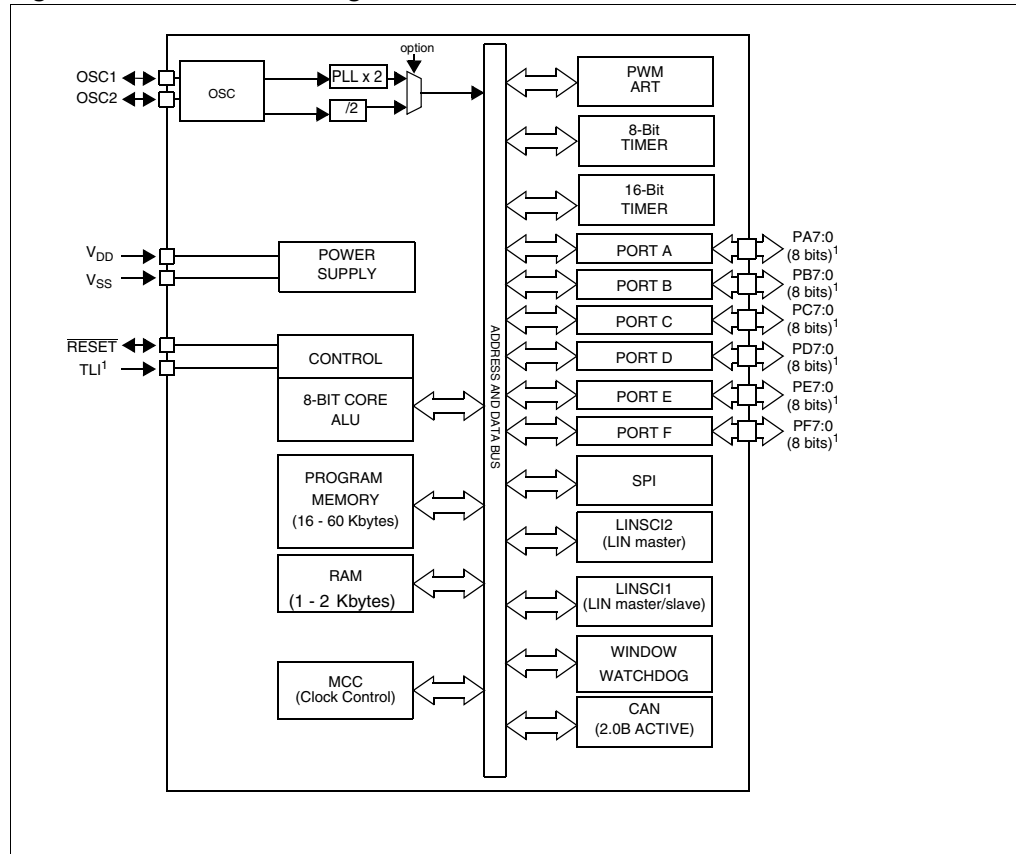
All devices are based on a common industry-standard 8-bit core, featuring an enhanced instruction set and are available with Flash or ROM program memory.

The enhanced instruction set and addressing modes of the ST7 offer both power and flexibility to software developers, enabling the design of highly efficient and compact application code. In addition to standard 8-bit data management, all ST7 microcontrollers feature true bit manipulation, 8x8 unsigned multiplication and indirect addressing modes.

Table 2. Product overview

| Features | ST72561(AR/R/J/K)9 | ST72561(AR/R/J/K)7 | ST72561(AR/R/J/K)6 | ST72561(AR/R/J/K)4 |
|---------------------------|--|--------------------|--------------------|--------------------|
| Program memory - bytes | 60K | 48K | 32K | 16K |
| RAM (stack) - bytes | 2K (256) | 2K (256) | 1K (256) | 1K (256) |
| Operating supply | 4.5V to 5.5V | | | |
| CPU frequency | External resonator oscillator w/ PLLx2/8 MHz | | | |
| Maximum temperature range | -40°C to +125°C | | | |
| Packages | LQFP64 10x10mm (AR), LQFP44 10x10mm (J), LQFP32 7x7mm (K) LQFP64 14x14 (R) | | | |

Figure 1. Device block diagram



1. On some devices only (see [Table 2: Product overview](#))

1.1 Pin description

Figure 2. LQFP 64-pin package pinout

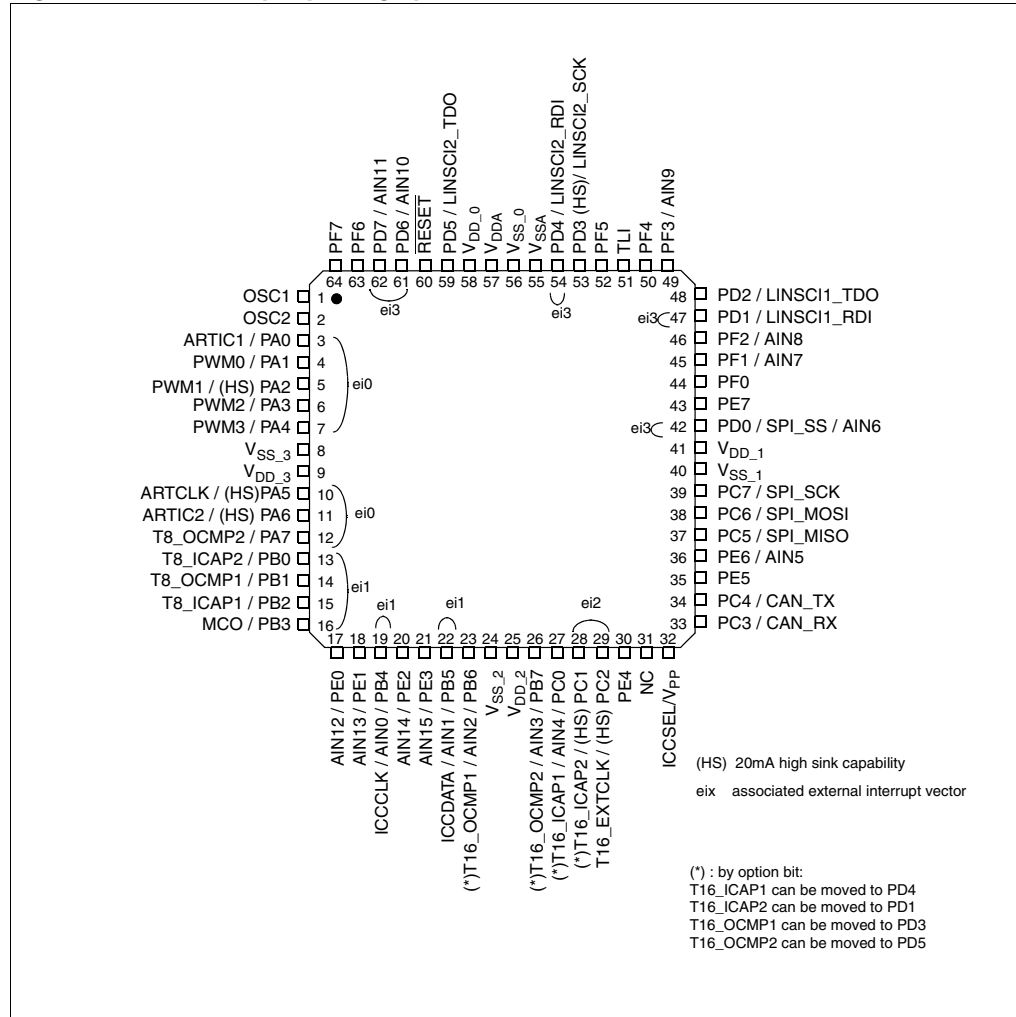


Figure 3. LQFP 44-pin package pinout

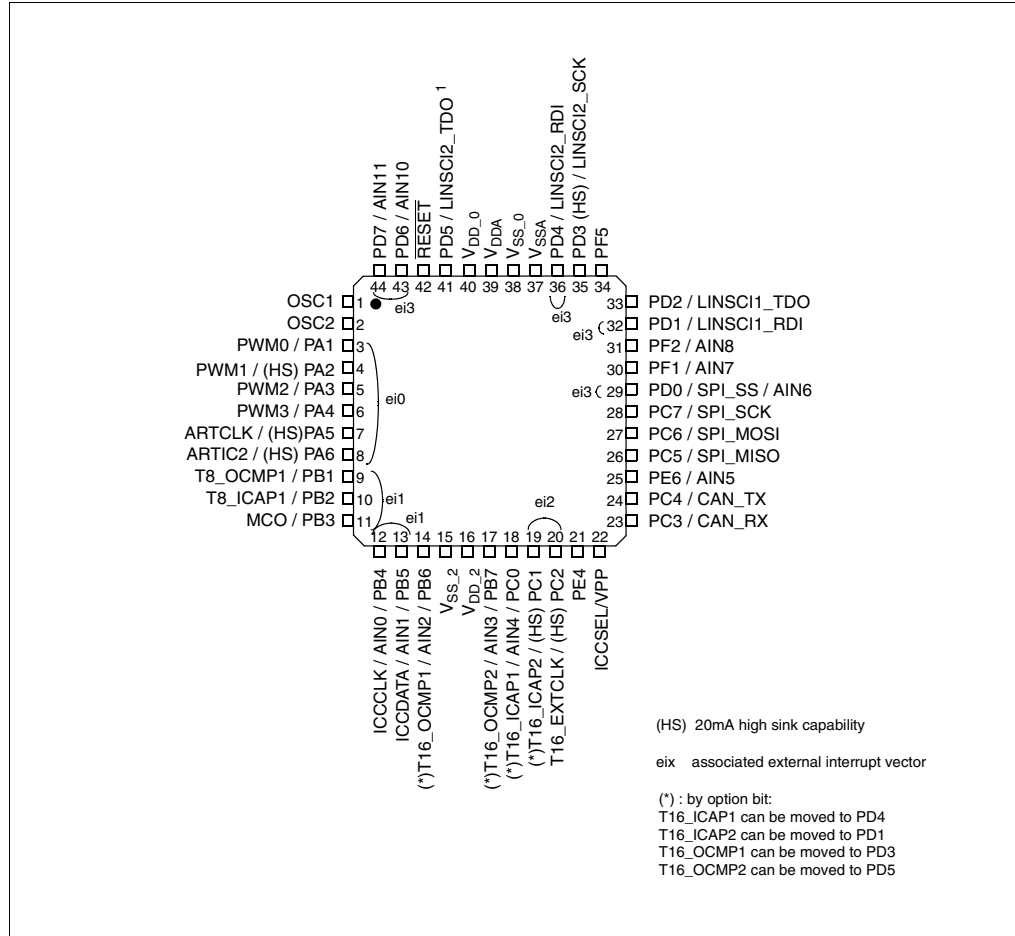
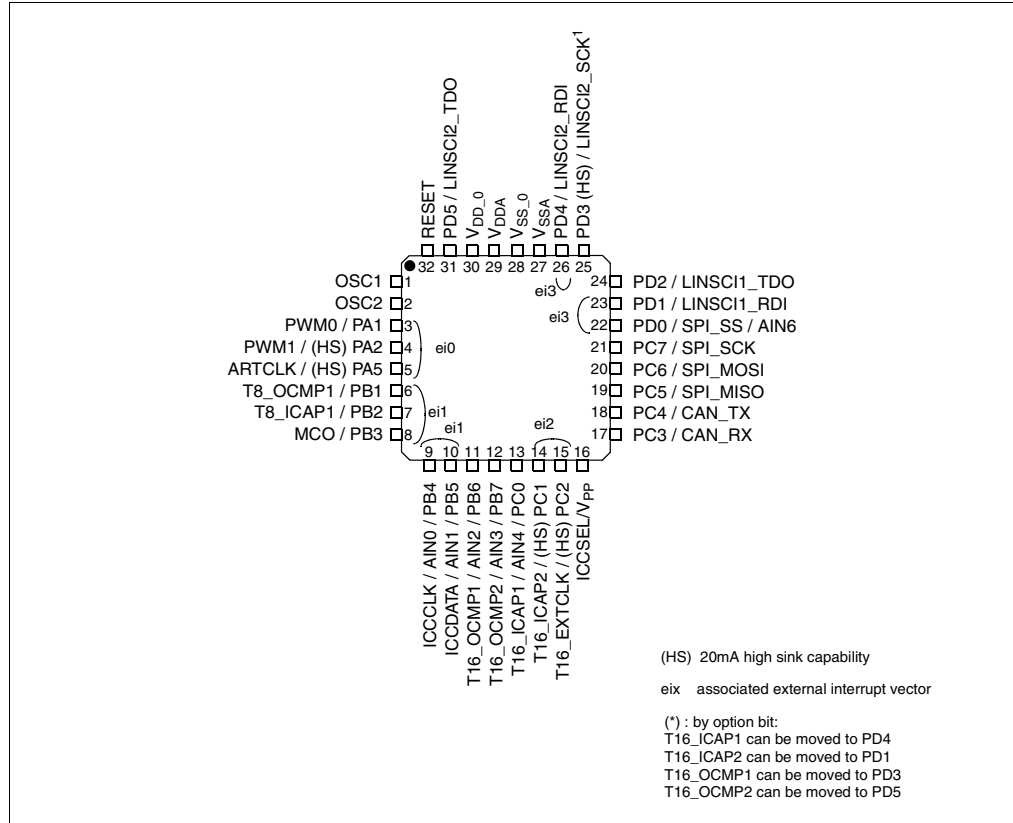


Figure 4. LQFP 32-pin package pinout



For external pin connection guidelines, refer to [Chapter 20: Electrical characteristics](#).

List of abbreviations used in Table 3

Type: I = input, O = output, S = supply

In/Output level: $C_T = \text{CMOS } 0.3V_{DD}/0.7V_{DD}$ with Schmitt trigger

$T_T = \text{TTL } 0.8V / 2V$ with Schmitt trigger

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

Port and control configuration:

Input: float = floating, wpu = weak pull-up, int = interrupt, ana = analog,

RB = robust

Output: OD = open drain, PP = push-pull

Refer to [Chapter 8: I/O ports](#) for more details on the software configuration of the I/O ports. The RESET configuration of each pin is shown in bold which is valid as long as the device is in reset state.

Table 3. Device pin description

| Pin n° | | | Pin name | Type | Level | | Port | | | | | | Main function (after reset) | Alternate function | |
|--------|--------|--------|---------------------|------|-------|--------|----------------------|-----|-----|-----|--------|----|-----------------------------|--------------------|---|
| LQFP64 | LQFP44 | LQFP32 | | | Input | Output | Input ⁽¹⁾ | | | | Output | | | | |
| | | | | | | | float | wpu | int | ana | OD | PP | | | |
| 1 | 1 | 1 | OSC1 ⁽²⁾ | I | | | | | | | | | | | External clock input or resonator oscillator inverter input |
| 2 | 2 | 2 | OSC2 ⁽²⁾ | I/O | | | | | | | | | | | Resonator oscillator inverter output |
| 3 | - | - | PA0 / ARTIC1 | I/O | C_T | | X | | ei0 | | X | X | Port A0 | | ART input capture 1 |
| 4 | 3 | 3 | PA1 / PWM0 | I/O | C_T | | X | | ei0 | | X | X | Port A1 | | ART PWM output 0 |
| 5 | 4 | 4 | PA2 (HS) / PWM1 | I/O | C_T | HS | X | | ei0 | | X | X | Port A2 | | ART PWM output 1 |
| 6 | 5 | - | PA3 / PWM2 | I/O | C_T | | X | | ei0 | | X | X | Port A3 | | ART PWM output 2 |
| 7 | 6 | - | PA4 / PWM3 | I/O | C_T | | X | | ei0 | | X | X | Port A4 | | ART PWM output 3 |
| 8 | - | - | V_{SS_3} | S | | | | | | | | | | | Digital ground voltage |
| 9 | - | - | V_{DD_3} | S | | | | | | | | | | | Digital main supply voltage |
| 10 | 7 | 5 | PA5 (HS) / ARTCLK | I/O | C_T | HS | X | | ei0 | | X | X | Port A5 | | ART external clock |
| 11 | 8 | - | PA6 (HS) / ARTIC2 | I/O | C_T | HS | X | | ei0 | | X | X | Port A6 | | ART input capture 2 |
| 12 | - | - | PA7 / T8_OCMP2 | I/O | C_T | | X | | ei0 | | X | X | Port A7 | | TIM8 output capture 2 |
| 13 | - | - | PB0 / T8_ICAP2 | I/O | C_T | | X | | ei1 | | X | X | Port B0 | | TIM8 input capture 2 |
| 14 | 9 | 6 | PB1 / T8_OCMP1 | I/O | C_T | | X | | ei1 | | X | X | Port B1 | | TIM8 output capture 1 |
| 15 | 10 | 7 | PB2 / T8_ICAP1 | I/O | C_T | | X | | ei1 | | X | X | Port B2 | | TIM8 input capture 1 |
| 16 | 11 | 8 | PB3 / MCO | I/O | C_T | | X | | ei1 | | X | X | Port B3 | | Main clock out (f_{OSC2}) |
| 17 | - | - | PE0 / AIN12 | I/O | T_T | | X | X | | RB | X | X | Port E0 | | ADC analog input 12 |
| 18 | - | - | PE1 / AIN13 | I/O | T_T | | X | X | | RB | X | X | Port E1 | | ADC analog input 13 |
| 19 | 12 | 9 | PB4 / AIN0 / ICCCLK | I/O | C_T | | X | | ei1 | RB | X | X | Port B4 | ICC clock input | ADC analog input 0 |

Table 3. Device pin description (continued)

| Pin n° | | | Pin name | Type | Level | | Port | | | | | | Main function (after reset) | Alternate function | | |
|--------|--------|--------|------------------------------|---------------|----------------|--------|----------------------|-----|-----|-----|--------|------------------|-----------------------------|-----------------------------|--|--------------------|
| LQFP64 | LQFP44 | LQFP32 | | | Input | Output | Input ⁽¹⁾ | | | | Output | | | | | |
| | | | | | | | float | wpu | int | ana | OD | PP | | | | |
| 20 | - | - | PE2 / AIN14 | I/O | T _T | | X | X | | | RB | X | X | Port E2 | ADC analog input 14 | |
| 21 | - | - | PE3 / AIN15 | I/O | T _T | | X | X | | | RB | X | X | Port E3 | ADC analog input 15 | |
| 22 | 13 | 10 | PB5 / AIN1 / ICCDATA | I/O | C _T | | X | | ei1 | | RB | X | X | Port B5 | ICC data input | ADC analog input 1 |
| 23 | 14 | 11 | PB6 / AIN2 / T16_OCMP1 | I/O | C _T | | X | X | | | RB | X | X | Port B6 | TIM16 output compare 1 | ADC analog input 2 |
| 24 | 15 | - | V _{SS_2} | S | | | | | | | | | | Digital ground voltage | | |
| 25 | 16 | - | V _{DD_2} | S | | | | | | | | | | Digital main supply voltage | | |
| 26 | 17 | 12 | PB7 / AIN3 / T16_OCMP2 | I/O | C _T | | X | X | | | RB | X | X | Port B7 | TIM16 output compare 2 | ADC analog input 3 |
| 27 | 18 | 13 | PC0 / AIN4 / T16_ICAP1 | I/O | C _T | | X | X | | | RB | X | X | Port C0 | TIM16 input capture 1 | ADC analog input 4 |
| 28 | 19 | 14 | PC1 (HS) / T16_ICAP2 | I/O | C _T | HS | X | | ei2 | | | X | X | Port C1 | TIM16 input capture 2 | |
| 29 | 20 | 15 | PC2 (HS) / T16_EXTCLK | I/O | C _T | HS | X | | ei2 | | | X | X | Port C2 | TIM16 external clock input | |
| 30 | 21 | - | PE4 | I/O | T _T | | X | X | | | | X | X | Port E4 | | |
| 31 | - | - | NC | Not Connected | | | | | | | | | | | | |
| 32 | 22 | 16 | V _{PP} | I | | | | | | | | | | | Flash programming voltage. Must be tied low in user mode | |
| 33 | 23 | 17 | PC3 / CANRX | I/O | C _T | | X | X | | | | X | X | Port C3 | CAN receive data input | |
| 34 | 24 | 18 | PC4 / CANTX | I/O | C _T | | X | | | | | X ⁽³⁾ | | Port C4 | CAN transmit data output | |
| 35 | - | - | PE5 | I/O | T _T | | X | X | | | | X | X | Port E5 | | |
| 36 | 25 | - | PE6 / AIN5 | I/O | T _T | | X | X | | | X | X | X | Port E6 | ADC analog input 5 | |
| 37 | 26 | 19 | PC5 / MISO | I/O | C _T | | X | X | | | | X | X | Port C5 | SPI master in/slave out | |
| 38 | 27 | 20 | PC6 / MOSI | I/O | C _T | | X | X | | | | X | X | Port C6 | SPI master out/slave in | |
| 39 | 28 | 21 | PC7 / SCK | I/O | C _T | | X | X | | | | X | X | Port C7 | SPI serial clock | |
| 40 | - | - | V _{SS_1} | S | | | | | | | | | | Digital ground voltage | | |
| 41 | - | - | V _{DD_1} | S | | | | | | | | | | Digital main supply voltage | | |
| 42 | 29 | 22 | PD0 / \overline{SS} / AIN6 | I/O | C _T | | X | | ei3 | | X | X | X | Port D0 | SPI slave select | ADC analog input 6 |

Table 3. Device pin description (continued)

| Pin n° | | | Pin name | Type | Level | | Port | | | | | | Main function (after reset) | Alternate function |
|--------|--------|--------|---------------------|------|----------------|--------|----------------------|-----|-----|-----|--------|----|--------------------------------------|------------------------------|
| LQFP64 | LQFP44 | LQFP32 | | | Input | Output | Input ⁽¹⁾ | | | | Output | | | |
| | | | | | | | float | wpu | int | ana | OD | PP | | |
| 43 | - | - | PE7 | I/O | T _T | | X | X | | | X | X | Port E7 | |
| 44 | - | - | PF0 | I/O | T _T | | X | X | | | X | X | Port F0 | |
| 45 | 30 | - | PF1 / AIN7 | I/O | T _T | | X | X | | X | X | X | Port F1 | ADC analog input 7 |
| 46 | 31 | - | PF2 / AIN8 | I/O | T _T | | X | X | | X | X | X | Port F2 | ADC analog input 8 |
| 47 | 32 | 23 | PD1 / SCI1_RDI | I/O | C _T | | X | | ei3 | | X | X | Port D1 | LINSCI1 receive data input |
| 48 | 33 | 24 | PD2 / SCI1_TDO | I/O | C _T | | X | X | | | X | X | Port D2 | LINSCI1 transmit data output |
| 49 | - | - | PF3 / AIN9 | I/O | T _T | | X | X | | X | X | X | Port F3 | ADC analog input 9 |
| 50 | - | - | PF4 | I/O | T _T | | X | X | | | X | X | Port F4 | |
| 51 | - | - | TLI | I | C _T | | X | | X | | | | Top level interrupt input pin | |
| 52 | 34 | - | PF5 | I/O | T _T | | X | X | | | X | X | Port F5 | |
| 53 | 35 | 25 | PD3 (HS) / SCI2_SCK | I/O | C _T | HS | X | X | | | X | X | Port D3 | LINSCI2 serial clock output |
| 54 | 36 | 26 | PD4 / SCI2_RDI | I/O | C _T | | X | | ei3 | | X | X | Port D4 | LINSCI2 receive data input |
| 55 | 37 | 27 | V _{SSA} | S | | | | | | | | | Analog ground voltage | |
| 56 | 38 | 28 | V _{SS_0} | S | | | | | | | | | Digital ground voltage | |
| 57 | 39 | 29 | V _{DDA} | I | | | | | | | | | Analog reference voltage for ADC | |
| 58 | 40 | 30 | V _{DD_0} | S | | | | | | | | | Digital main supply voltage | |
| 59 | 41 | 31 | PD5 / SCI2_TDO | I/O | C _T | | X | X | | | X | X | Port D5 | LINSCI2 transmit data output |
| 60 | 42 | 32 | RESET | I/O | C _T | | | | | | | | Top priority non maskable interrupt. | |
| 61 | 43 | - | PD6 / AIN10 | I/O | C _T | | X | | ei3 | X | X | X | Port D6 | ADC analog input 10 |
| 62 | 44 | - | PD7 / AIN11 | I/O | C _T | | X | | ei3 | X | X | X | Port D7 | ADC analog input 11 |
| 63 | - | - | PF6 | I/O | T _T | | X | X | | | X | X | Port F6 | |
| 64 | - | - | PF7 | I/O | T _T | | X | X | | | X | X | Port F7 | |

1. 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.
2. OSC1 and OSC2 pins connect a crystal/ceramic resonator, or an external source to the on-chip oscillator; see [Chapter 1: Description and Section 20.5: Clock and timing characteristics](#) for more details.
3. Input mode can be used for general purpose I/O, output mode only for CANTX. On the chip, each I/O port has 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 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.

Figure 5. Memory map

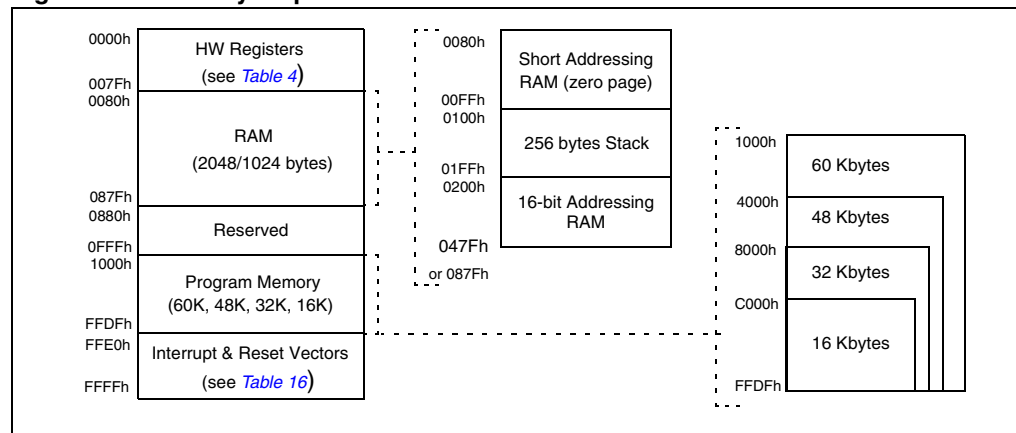


Table 4. Hardware register map

| Address | Block | Register label | Register name | Reset status | Remarks ⁽¹⁾ |
|-------------------------|--------|-----------------------|--|----------------------------------|--|
| 0000h 0001h 0002h | Port A | PADR PADDR PAOR | Port A Data Register Port A Data Direction Register Port A Option Register | 00h ⁽²⁾ 00h 00h | R/W ⁽³⁾ R/W ⁽³⁾ R/W ⁽³⁾ |
| 0003h 0004h 0005h | Port B | PBDR PBDDR PBOR | Port B Data Register Port B Data Direction Register Port B Option Register | 00h ⁽²⁾ 00h 00h | R/W ⁽³⁾ R/W ⁽³⁾ R/W ⁽³⁾ |
| 0006h 0007h 0008h | Port C | PCDR PCDDR PCOR | Port C Data Register Port C Data Direction Register Port C Option Register | 00h ⁽²⁾ 00h 00h | R/W ⁽³⁾ R/W ⁽³⁾ R/W ⁽³⁾ |
| 0009h 000Ah 000Bh | Port D | PDDR PDDDR PDOR | Port D Data Register Port D Data Direction Register Port D Option Register | 00h ⁽²⁾ 00h 00h | R/W ⁽³⁾ R/W ⁽³⁾ R/W ⁽³⁾ |
| 000Ch 000Dh 000Eh | Port E | PEDR PEDDR PEOR | Port E Data Register Port E Data Direction Register Port E Option Register | 00h ⁽²⁾ 00h 00h | R/W ⁽³⁾ R/W ⁽³⁾ R/W ⁽³⁾ |

Table 4. Hardware register map (continued)

| Address | Block | Register label | Register name | Reset status | Remarks ⁽¹⁾ |
|---|--------------------------|---|---|---|---|
| 000Fh 0010h 0011h | Port F | PFDR PFDDR PFOR | Port F Data Register Port F Data Direction Register Port F Option Register | 00h ⁽²⁾ 00h 00h | R/W ⁽³⁾ R/W ⁽³⁾ R/W ⁽³⁾ |
| 0012h to 0020h | Reserved Area (15 bytes) | | | | |
| 0021h 0022h 0023h | SPI | SPIDR SPICR SPICSR | SPI Data I/O Register SPI Control Register SPI Control/Status Register | xxh 0xh 00h | R/W R/W R/W |
| 0024h | FLASH | FCSR | Flash Control/Status Register | 00h | R/W |
| 0025h 0026h 0027h 0028h 0029h 002Ah | ITC | ISPR0 ISPR1 ISPR2 ISPR3 EICR0 EICR1 | Interrupt Software Priority Register 0 Interrupt Software Priority Register 1 Interrupt Software Priority Register 2 Interrupt Software Priority Register 3 External Interrupt Control Register 0 External Interrupt Control Register 1 | FFh FFh FFh FFh 00h 00h | R/W R/W R/W R/W R/W R/W |
| 002Bh 002Ch | AWU | AWUCSR AWUPR | Auto Wake up f. Halt Control/Status Register Auto Wake Up From Halt Prescaler | 00h FFh | R/W R/W |
| 002Dh 002Eh | CKCTRL | SICSR MCCSR | System Integrity Control / Status Register Main Clock Control / Status Register | 0xh 00h | R/W R/W |
| 002Fh 0030h | WWDG | WDGCR WDGWR | Watchdog Control Register Watchdog Window Register | 7Fh 7Fh | R/W R/W |
| 0031h 0032h 0033h 0034h 0035h 0036h 0037h 0038h 0039h 003Ah 003Bh | PWM ART | PWMDCR3 PWMDCR2 PWMDCR1 PWMDCR0 PWMCR ARTCSR ARTCAR ARTARR ARTICCSR ARTICR1 ARTICR2 | Pulse Width Modulator Duty Cycle Register 3 PWM Duty Cycle Register 2 PWM Duty Cycle Register 1 PWM Duty Cycle Register 0 PWM Control register Auto-Reload Timer Control/Status Register Auto-Reload Timer Counter Access Register Auto-Reload Timer Auto-Reload Register ART Input Capture Control/Status Register ART Input Capture Register 1 ART Input Capture register 2 | 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h | R/W R/W R/W R/W R/W R/W R/W R/W R/W Read Only Read Only |
| 003Ch 003Dh 003Eh 003Fh 0040h 0041h 0042h 0043h 0044h | 8-BIT TIMER | T8CR2 T8CR1 T8CSR T8IC1R T8OC1R T8CTR T8ACTR T8IC2R T8OC2R | Timer Control Register 2 Timer Control Register 1 Timer Control/Status Register Timer Input Capture 1 Register Timer Output Compare 1 Register Timer Counter Register Timer Alternate Counter Register Timer Input Capture 2 Register Timer Output Compare 2 Register | 00h 00h 00h xxh 00h FCh FCh xxh 00h | R/W R/W Read Only Read Only R/W Read Only Read Only Read Only R/W |
| 0045h 0046h 0047h | ADC | ADCCSR ADCDRH ADCRL | Control/Status Register Data High Register Data Low Register | 00h 00h 00h | R/W Read Only Read Only |

Table 4. Hardware register map (continued)

| Address | Block | Register label | Register name | Reset status | Remarks ⁽¹⁾ |
|---|-------------------------------|--|---|---|--|
| 0048h 0049h 004Ah 004Bh 004Ch 004Dh 004Eh 004Fh | LINSCI1 (LIN Master/Slave) | SCI1ISR SCI1DR SCI1BRR SCI1CR1 SCI1CR2 SCI1CR3 SCI1ERPR SCI1ETPR | SCI1 Status Register SCI1 Data Register SCI1 Baud Rate Register SCI1 Control Register 1 SCI1 Control Register 2 SCI1 Control Register 3 SCI1 Extended Receive Prescaler Register SCI1 Extended Transmit Prescaler Register | C0h xxh 00h xxh 00h 00h 00h 00h | Read Only R/W R/W R/W R/W R/W R/W R/W |
| 0050h | Reserved Area (1 byte) | | | | |
| 0051h 0052h 0053h 0054h 0055h 0056h 0057h 0058h 0059h 005Ah 005Bh 005Ch 005Dh 005Eh 005Fh | 16-BIT TIMER | T16CR2 T16CR1 T16CSR T16IC1HR T16IC1LR T16OC1HR T16OC1LR T16CHR T16CLR T16ACHR T16ACLR T16IC2HR T16IC2LR T16OC2HR T16OC2LR | Timer Control Register 2 Timer Control Register 1 Timer Control/Status Register Timer Input Capture 1 High Register Timer Input Capture 1 Low Register Timer Output Compare 1 High Register Timer Output Compare 1 Low Register Timer Counter High Register Timer Counter Low Register Timer Alternate Counter High Register Timer Alternate Counter Low Register Timer Input Capture 2 High Register Timer Input Capture 2 Low Register Timer Output Compare 2 High Register Timer Output Compare 2 Low Register | 00h 00h 00h xxh xxh 80h 00h FFh FCh FFh FCh xxh xxh 80h 00h | R/W R/W R/W Read Only Read Only R/W R/W Read Only Read Only Read Only Read Only Read Only R/W R/W |
| 0060h 0061h 0062h 0063h 0064h 0065h 0066h 0067h | LINSCI2 (LIN Master) | SCI2SR SCI2DR SCI2BRR SCI2CR1 SCI2CR2 SCI2CR3 SCI2ERPR SCI2ETPR | SCI2 Status Register SCI2 Data Register SCI2 Baud Rate Register SCI2 Control Register 1 SCI2 Control Register 2 SCI2 Control Register 3 SCI2 Extended Receive Prescaler Register SCI2 Extended Transmit Prescaler Register | C0h xxh 00h xxh 00h 00h 00h 00h | Read Only R/W R/W R/W R/W R/W R/W R/W |

Table 4. Hardware register map (continued)

| Address | Block | Register label | Register name | Reset status | Remarks ⁽¹⁾ | |
|---------|------------|----------------|--------------------------------|------------------|------------------------|-----|
| 0068h | Active CAN | CMCR | CAN Master Control Register | | R/W | |
| 0069h | | CMSR | CAN Master Status Register | | R/W | |
| 006Ah | | CTSR | CAN Transmit Status Register | | R/W | |
| 006Bh | | CTPR | CAN Transmit Priority Register | | R/W | |
| 006Ch | | CRFR | CAN Receive FIFO Register | | R/W | |
| 006Dh | | CIER | CAN Interrupt Enable Register | | R/W | |
| 006Eh | | CDGR | CAN Diagnosis Register | | R/W | |
| 006Fh | | CPSR | CAN Page Selection Register | | R/W | |
| 0070h | | PAGES | | PAGE REGISTER 0 | | R/W |
| 0071h | | | | PAGE REGISTER 1 | | R/W |
| 0072h | | | | PAGE REGISTER 2 | | R/W |
| 0073h | | | | PAGE REGISTER 3 | | R/W |
| 0074h | | | | PAGE REGISTER 4 | | R/W |
| 0075h | | | | PAGE REGISTER 5 | | R/W |
| 0076h | | | | PAGE REGISTER 6 | | R/W |
| 0077h | | | | PAGE REGISTER 7 | | R/W |
| 0078h | | | | PAGE REGISTER 8 | | R/W |
| 0079h | | | | PAGE REGISTER 9 | | R/W |
| 007Ah | | | | PAGE REGISTER 10 | | R/W |
| 007Bh | | | | PAGE REGISTER 11 | | R/W |
| 007Ch | | | | PAGE REGISTER 12 | | R/W |
| 007Dh | | | | PAGE REGISTER 13 | | R/W |
| 007Eh | | | | PAGE REGISTER 14 | | R/W |
| 007Fh | | | | PAGE REGISTER 15 | | R/W |

1. x = undefined, R/W = read/write

2. 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.

3. The bits associated with unavailable pins must always keep their reset value.

3 Flash program memory

3.1 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.

3.2 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
- Read-out protection
- Register Access Security System (RASS) to prevent accidental programming or erasing

3.3 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 5](#)). Each of these sectors can be erased independently to avoid unnecessary erasing of the whole Flash memory when only a partial erasing is required.

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).

Table 5. Sectors available in Flash devices

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

3.3.1 Read-out protection

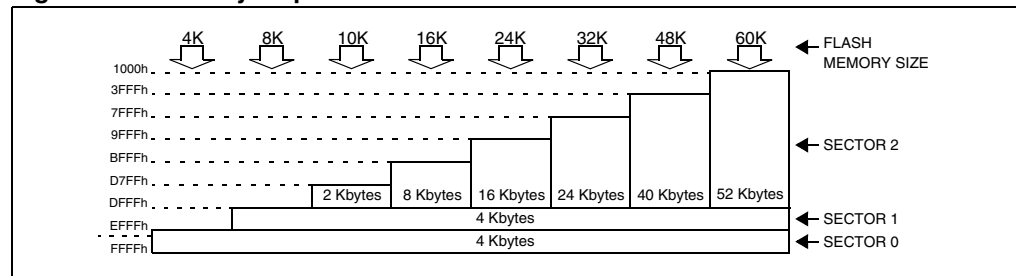
Read-out protection, when selected, provides a protection against Program Memory content extraction and against write access to Flash memory. Even if no protection can be considered as totally unbreakable, the feature provides a very high level of protection for a general purpose microcontroller.

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.

Read-out protection selection depends on the device type:

- In Flash devices it is enabled and removed through the FMP_R bit in the option byte.
- In ROM devices it is enabled by mask option specified in the Option List.

Figure 6. Memory map and sector address

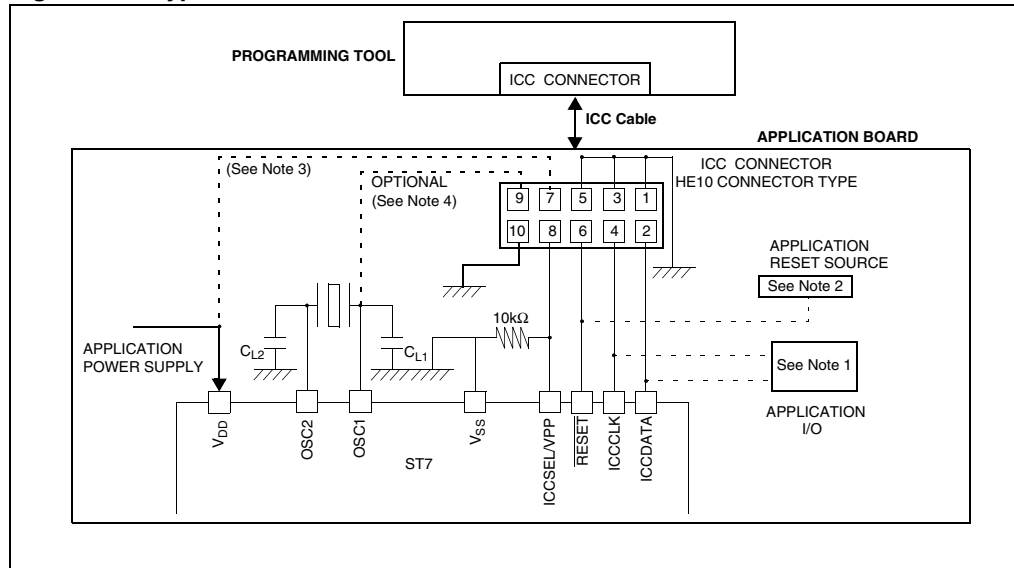


3.4 ICC interface

ICC (In-Circuit Communication) needs a minimum of four and up to six 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 (see [Figure 7](#), Note 3)

Figure 7. Typical ICC interface



- Note:
- 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 be 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 $\overline{\text{RESET}}$ 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.

3.5 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.

3.6 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, (user-defined strategy for entering programming mode, choice of communications protocol used to fetch the data to be stored, etc.). For example, it is possible to download code from the SPI, SCI or other type of serial 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.

3.7 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*.

3.8 Register description

Flash Control/Status Register (FCSR)

Read/ write

Reset value: 0000 0000 (00h)

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 7 | | | | | | | | | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

Table 6. Flash control/status register address and reset value

| Address (Hex.) | Register label | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------------|---------------------|---|---|---|---|---|---|---|---|
| 0024h | FCSR Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

4 Central processing unit

4.1 Introduction

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

4.2 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

4.3 CPU registers

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

4.3.1 Accumulator (A)

The Accumulator is an 8-bit general purpose register used to hold operands and the results of the arithmetic and logic calculations and to manipulate data.

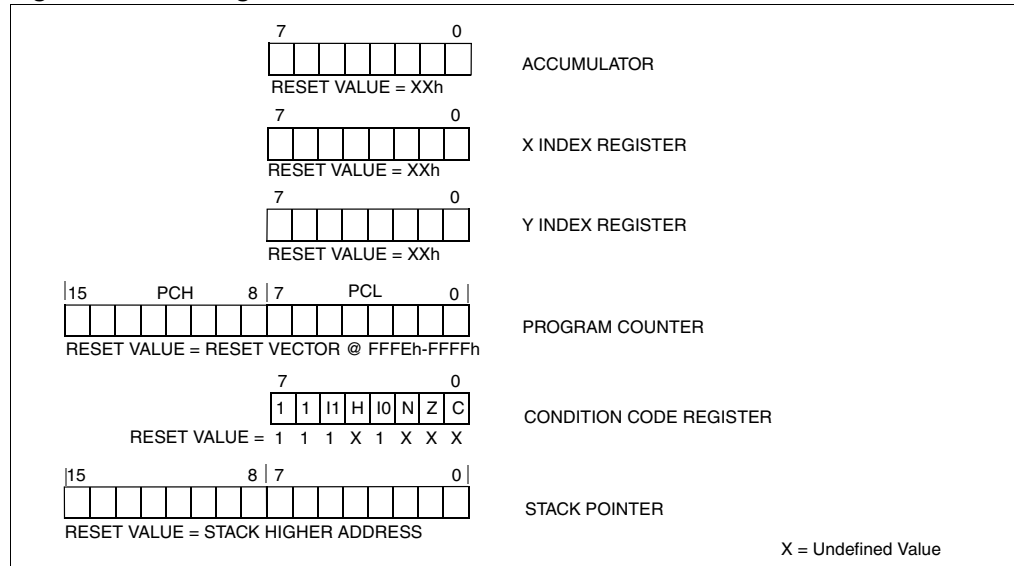
4.3.2 Index registers (X and Y)

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.

4.3.3 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).

Figure 8. CPU registers

4.3.4 Condition code register (CC)

Read/ write

Reset value: 111x1xxx



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.

Arithmetic management bits

Bit 4 = **H** *Half carry*.

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.

- 0: No half carry has occurred.
- 1: A half carry has occurred.

This bit is tested using the JRH or JRNH instruction. The H bit is useful in BCD arithmetic subroutines.

Bit 2 = **N** *Negative*.

This bit is set and cleared by hardware. It is representative of the result sign of the last arithmetic, logical or data manipulation. It's a copy of the result 7th bit.

- 0: The result of the last operation is positive or null.
- 1: The result of the last operation is negative (that is, the most significant bit is a logic 1).

This bit is accessed by the JRMI and JRPL instructions.

Bit 1 = **Z** *Zero*.

This bit is set and cleared by hardware. This bit indicates that the result of the last arithmetic, logical or data manipulation is zero.

0: The result of the last operation is different from zero.

1: The result of the last operation is zero.

This bit is accessed by the JREQ and JRNE test instructions.

Bit 0 = **C** *Carry/borrow*.

This bit is set and cleared by hardware and software. It indicates an overflow or an underflow has occurred during the last arithmetic operation.

0: No overflow or underflow has occurred.

1: An overflow or underflow has occurred.

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.

Interrupt management bits

Bit 5,3 = **I1, I0** *Interrupt*

The combination of the I1 and I0 bits gives the current interrupt software priority.

Table 7. Interrupt software priority selection

| Interrupt software priority | I1 | I0 |
|-------------------------------|----|----|
| Level 0 (main) | 1 | 0 |
| Level 1 | 0 | 1 |
| Level 2 | 0 | 0 |
| Level 3 (= interrupt disable) | 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 (IxSPR). They can be also set/cleared by software with the RIM, SIM, IRET, HALT, WFI and PUSH/POP instructions.

See the interrupt management chapter for more details.

4.3.5 Stack pointer (SP)

Read/ write

Reset value: 01 FFh

| | | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|---|
| 15 | | | | | | | | 8 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | |
| 7 | | | | | | | | 0 |
| SP7 | SP6 | SP5 | SP4 | SP3 | SP2 | SP1 | SP0 | |

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 a LD instruction.

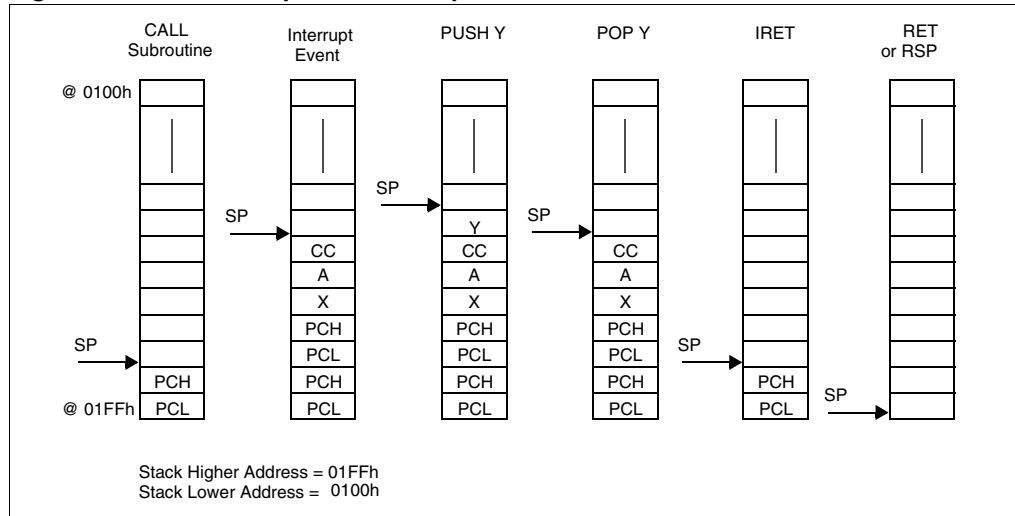
Note: When the lower limit is exceeded, the Stack Pointer wraps around to the stack upper limit, without indicating the stack overflow. The previously stored information is then overwritten and therefore lost. The stack also wraps in case of an underflow.

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. Then the other registers are 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



5 Supply, reset and clock management

5.1 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 11](#).

For more details, refer to dedicated parametric section.

5.2 Main features

- Optional PLL for multiplying the frequency by 2
- Reset Sequence Manager (RSM)
- Multi-Oscillator Clock Management (MO)
 - 4 Crystal/Ceramic resonator oscillators
- System Integrity Management (SI)
 - Main supply Low voltage detection (LVD)
 - Auxiliary Voltage detector (AVD) with interrupt capability for monitoring the main supply

5.3 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. [Section 20.5.2: PLL characteristics](#)

Figure 10. PLL block diagram

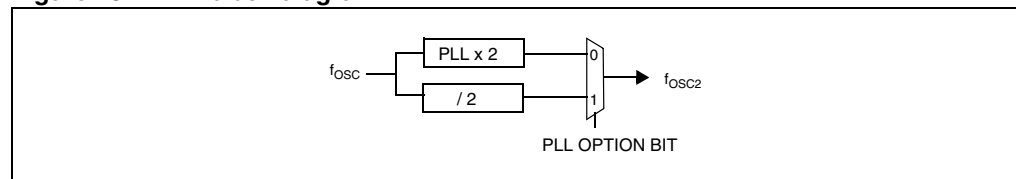
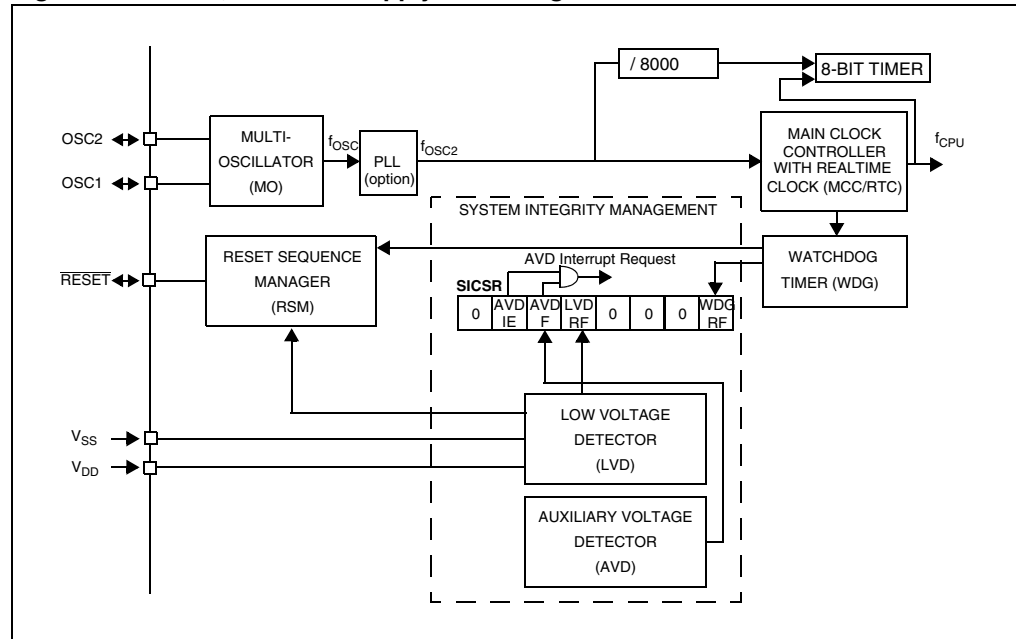


Figure 11. Clock, reset and supply block diagram



5.4 Multi-oscillator (MO)

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

- an external source
- a crystal or ceramic resonator oscillator

Each oscillator is optimized for a given frequency range in terms of consumption and is selectable through the option byte. The associated hardware configuration are shown in [Table 8](#). Refer to the electrical characteristics section for more details.

Caution: The OSC1 and/or OSC2 pins must not be left unconnected. For the purposes of Failure Mode and Effect Analysis, it should be noted that if the OSC1 and/or OSC2 pins are left unconnected, the ST7 main oscillator may start and, in this configuration, could generate an f_{OSC} clock frequency in excess of the allowed maximum (> 16 MHz), putting the ST7 in an unsafe/undefined state. The product behavior must therefore be considered undefined when the OSC pins are left unconnected.

External clock source

In 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.

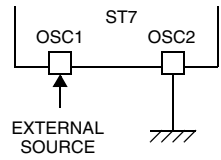
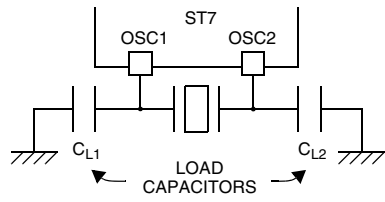
Crystal/ceramic oscillators

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 five oscillators with different frequency ranges must be done by option byte in order to reduce consumption (refer to [Section 22.2.1: Flash configuration](#) for more details on the frequency ranges). The resonator and the load capacitors must 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.

Table 8. ST7 clock sources

| | Hardware configuration |
|----------------------------|--|
| External clock |  |
| Crystal/Ceramic resonators |  |

5.5 Reset sequence manager (RSM)

5.5.1 Introduction

The reset sequence manager includes three RESET sources as shown in [Figure 13](#):

- 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.

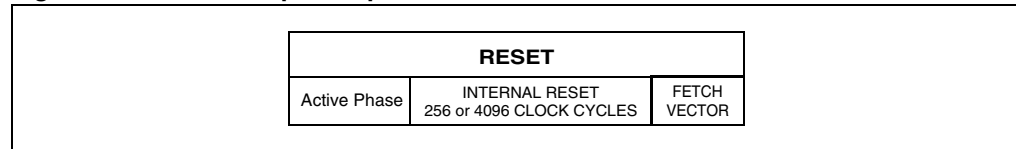
The basic RESET sequence consists of three phases as shown in [Figure 12](#):

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

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.

The reset vector fetch phase duration is two clock cycles.

Figure 12. RESET sequence phases

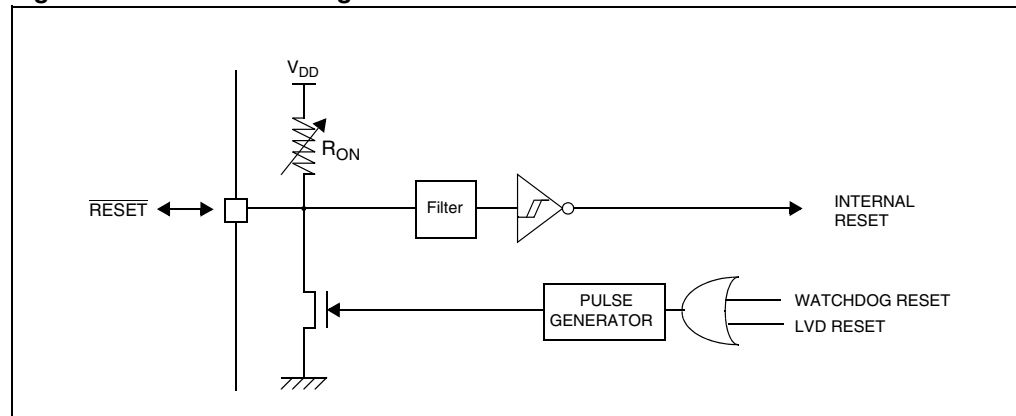


5.5.2 Asynchronous external $\overline{\text{RESET}}$ pin

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 [Chapter 20: Electrical characteristics](#) for more details.

A reset signal originating from an external source must have a duration of at least $t_{\text{h(RSTL)}}_{\text{in}}$ in order to be recognized (see [Figure 14](#)). This detection is asynchronous and therefore the MCU can enter reset state even in HALT mode.

Figure 13. Reset block diagram



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 the electrical characteristics section.

5.5.3 External power-on reset

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

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

5.5.4 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](#).

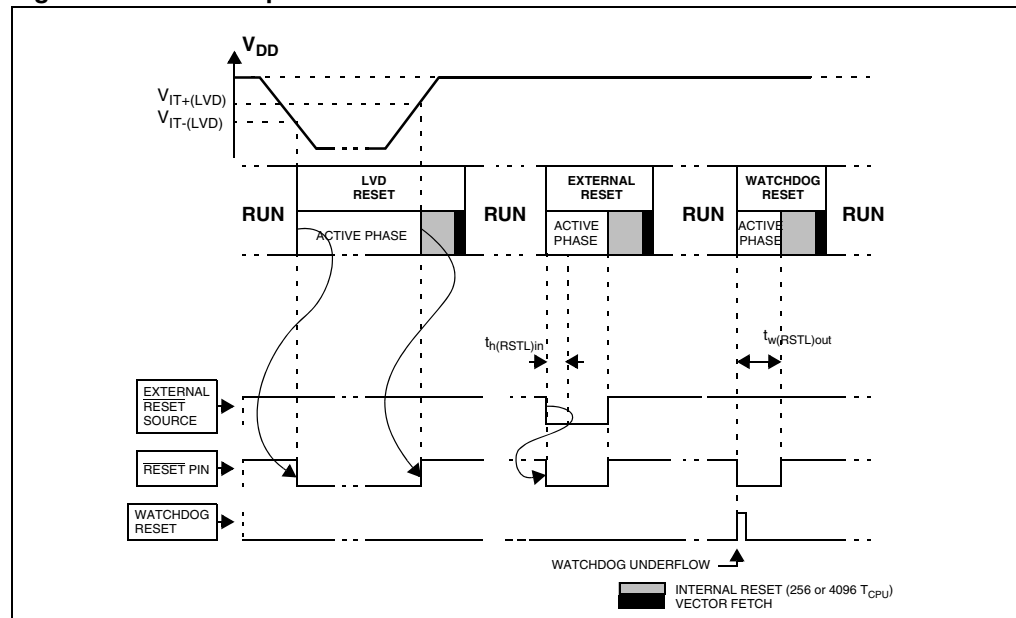
The LVD filters spikes on V_{DD} larger than $t_{g(VDD)}$ to avoid parasitic resets.

5.5.5 Internal watchdog reset

The RESET sequence generated by a 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 $t_{w(RSTL)out}$.

Figure 14. Reset sequences



5.6 System integrity management (SI)

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

5.6.1 Low voltage detector (LVD)

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

The $V_{IT-(LVD)}$ reference value for a voltage drop is lower than the $V_{IT+(LVD)}$ 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:

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

The LVD function is illustrated in [Figure 15](#).

Provided the minimum V_{DD} value (guaranteed for the oscillator frequency) is above $V_{IT-(LVD)}$, 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.

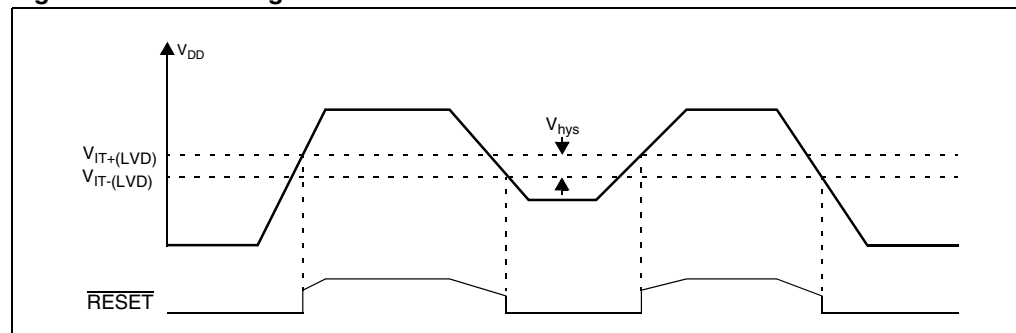
During a Low Voltage Detector Reset, the \overline{RESET} pin is held low, thus permitting the MCU to reset other devices.

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

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 vs reset



5.6.2 Auxiliary voltage detector (AVD)

The 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. The $V_{IT-(AVD)}$ reference value for falling voltage is lower than the $V_{IT+(AVD)}$ reference value for rising voltage in order to avoid parasitic detection (hysteresis).

The output of the AVD comparator is directly readable 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

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](#)).

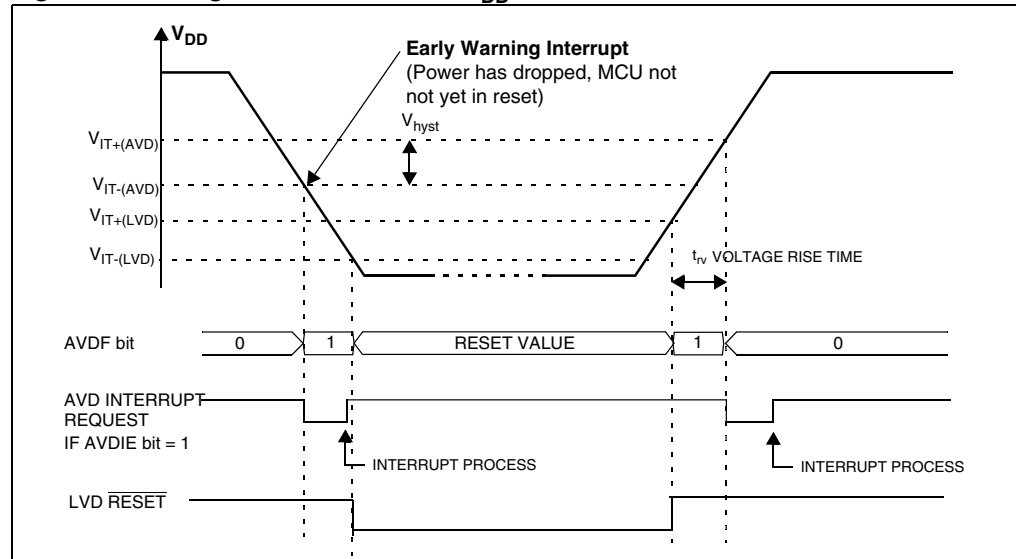
The interrupt on the rising edge is used to inform the application that the V_{DD} warning state is over.

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 then:

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

Figure 16. Using the AVD to monitor V_{DD}



5.6.3 Low power modes

Table 9. Effect of low power modes on SI

| Mode | Description |
|------|--|
| WAIT | No effect on SI. AVD interrupts cause the device to exit from Wait mode. |
| HALT | The SICSR register is frozen. |

5.6.4 Interrupts

The AVD interrupt event generates an interrupt if the AVDIE bit is set and the interrupt mask in the CC register is reset (RIM instruction).

Table 10. Interrupt control/wake-up capability

| Interrupt event | Event flag | Enable control bit | Exit from wait | Exit from halt |
|-----------------|------------|--------------------|----------------|----------------|
| AVD event | AVDF | AVDIE | Yes | No |

5.6.5 Register description

System integrity (SI) control/status register (SICSR)

Read/Write

Reset value: 000x 000x (00h)

| | | | | | | | |
|---|-------|------|-------|---|---|---|-------|
| 7 | | | | | | | 0 |
| 0 | AVDIE | AVDF | LVDRF | 0 | 0 | 0 | WDGRF |

Bit 7 = Reserved, must be kept cleared.

Bit 6 = **AVDIE** *Voltage Detector interrupt enable*

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.

- 0: AVD interrupt disabled
- 1: AVD interrupt enabled

Bit 5 = **AVDF** *Voltage Detector flag*

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](#) for additional details.

- 0: V_{DD} over $V_{IT+(AVD)}$ threshold
- 1: V_{DD} under $V_{IT-(AVD)}$ threshold

Bit 4 = **LVDRF** *LVD reset flag*

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 WDGRF flag description for more details. When the LVD is disabled by OPTION BYTE, the LVDRF bit value is undefined.

Bits 3:1 = Reserved, must be kept cleared.

Bit 0 = **WDGRF** *Watchdog reset flag*

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 by the following table.

Table 11. Reset source flags

| RESET sources | LVDRF | WDGRF |
|--|-------|-------|
| External $\overline{\text{RESET}}$ pin | 0 | 0 |
| Watchdog | | 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, a watchdog reset can be detected by software while an external reset can not.

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

6 Interrupts

6.1 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.

6.2 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 12](#)). The processing flow is shown in [Figure 17](#).

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 “Interrupt Mapping” table for vector addresses).

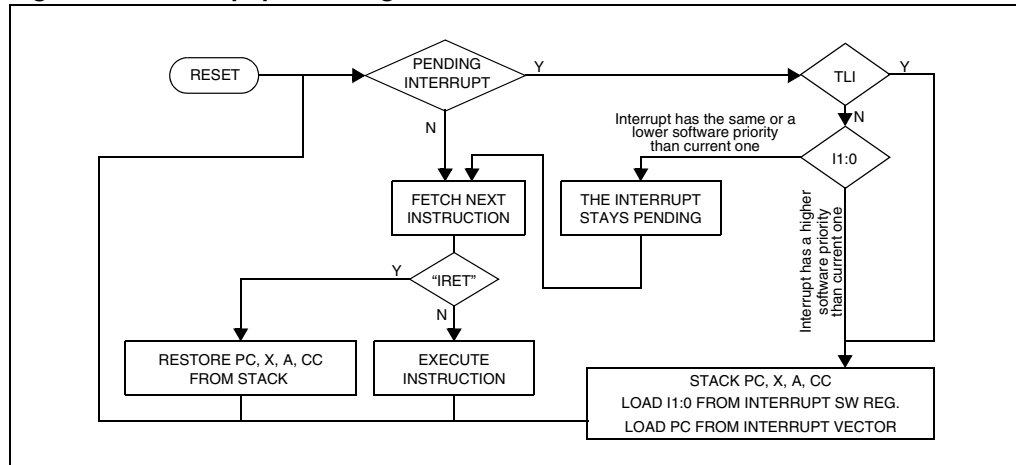
The interrupt service routine should end with the IRET instruction which causes the contents of the saved registers to be recovered from the stack.

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.

Table 12. Interrupt software priority levels

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

Figure 17. Interrupt processing flowchart



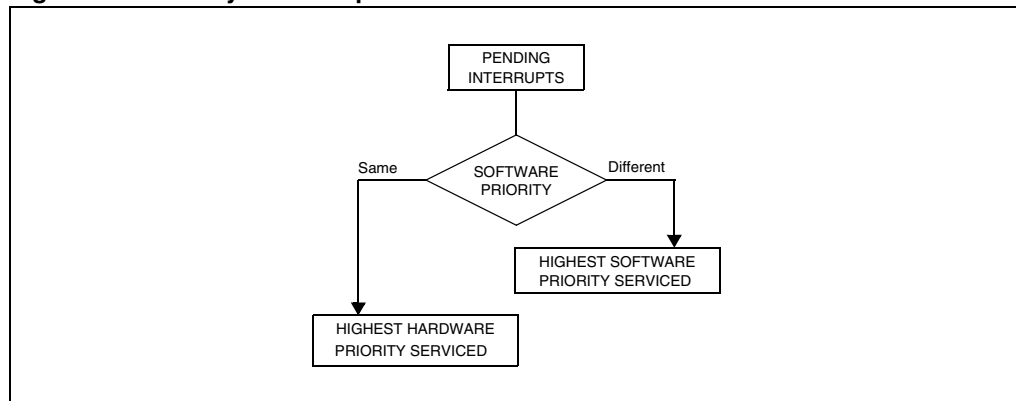
Servicing pending interrupts

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 18 describes this decision process.

Figure 18. Priority decision process



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 *RESET, TRAP and TLI 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 17](#)). 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)

Caution: This software interrupt is serviced when the TRAP instruction is executed. It will be serviced according to the flowchart in [Figure 17](#) as a TLI. 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 the RESET chapter 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. 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 the "Interrupt Mapping" table.

A peripheral interrupt occurs when a specific 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 for being serviced) will therefore be lost if the clear sequence is executed.

6.3 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” in “Interrupt Mapping” table). 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 18](#).

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.

6.4 Concurrent & nested management

The following [Figure 19](#) and [Figure 20](#) 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 20](#). 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 19. Concurrent interrupt management

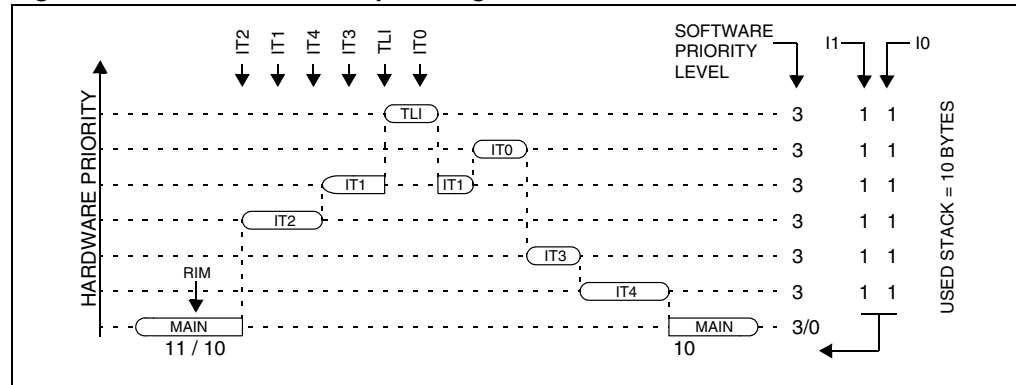
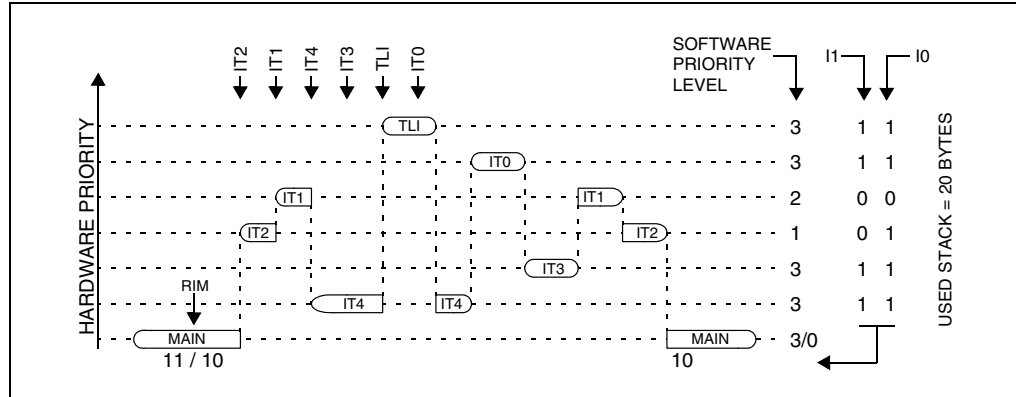


Figure 20. Nested interrupt management



6.5 Interrupt register description

6.5.1 CPU CC register interrupt bits

Read/Write

Reset value: 11x 1010 (xAh)

| | | | | | | | | | |
|---|---|---|----|---|----|---|---|---|---|
| 7 | 1 | 1 | I1 | H | I0 | N | Z | 0 | C |
|---|---|---|----|---|----|---|---|---|---|

Bit 5, 3 = I1, I0 Software Interrupt Priority

These two bits indicate the current interrupt software priority.

Table 13. 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⁾) | High | 1 | 1 |

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

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, HALT, WFI, IRET and PUSH/POP instructions (see [Table 15: Dedicated interrupt instruction set on page 55](#)).

6.5.2 Interrupt software priority registers (ISPRX)

Read/ write (bit 7:4 of ISPR3 are read only)

Reset value: 1111 1111 (FFh)

| | 7 | | | | 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.

Table 14. 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 RESET, TRAP and TLI 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. Otherwise, the software priority stays unchanged up to the next interrupt request (after the IRET of the interrupt x).

Table 15. Dedicated interrupt 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? | | | | | | |

Table 15. Dedicated interrupt instruction set (continued)

| Instruction | New description | Function/example | I1 | H | I0 | N | Z | C |
|-------------|---------------------------------|-----------------------|----|---|----|---|---|---|
| POP CC | Pop CC from the Stack | Mem => CC | I1 | H | I0 | N | Z | C |
| RIM | Enable interrupt (level 0 set) | Load I0 in I1:0 of CC | 1 | | 0 | | | |
| SIM | Disable interrupt (level 3 set) | Load I1 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 instruction

Table 16. Interrupt mapping

| N° | Source block | Description | Register label | Priority order | Exit from Halt ⁽¹⁾ | Address vector |
|----|--------------|---|----------------|--|-------------------------------|----------------|
| | RESET | Reset | N/A | Highest Priority ↓ Lowest Priority | yes | FFFEh-FFFFh |
| | TRAP | Software interrupt | | | no | FFFCh-FFFDh |
| 0 | TLI | External top level interrupt | EICR | | yes | FFFAh-FFFBh |
| 1 | MCC/RTC | Main clock controller time base interrupt | MCCSR | | yes | FFF8h-FFF9h |
| 2 | ei0/AWUFH | External interrupt ei0/ Auto wake-up from Halt | EICR/AWUCSR | | yes ⁽²⁾ | FFF6h-FFF7h |
| 3 | ei1/AVD | External interrupt ei1/Auxiliary Voltage Detector | EICR/SICSR | | | FFF4h-FFF5h |
| 4 | ei2 | External interrupt ei2 | EICR | | | FFF2h-FFF3h |
| 5 | ei3 | External interrupt ei3 | EICR | | | FFF0h-FFF1h |
| 6 | CAN | CAN peripheral interrupt - RX | CIER | | no | FFEEh-FFEFh |
| 7 | CAN | CAN peripheral interrupt - TX / ER / SC | CIER | | yes ⁽³⁾ | FFEC-FFEDh |
| 8 | SPI | SPI peripheral interrupts | SPICSR | | yes | FFEAh-FFEBh |
| 9 | TIMER8 | 8-bit TIMER peripheral interrupts | T8_TCR1 | | no | FFE8h-FFE9h |
| 10 | TIMER16 | 16-bit TIMER peripheral interrupts | TCR1 | | no | FFE6h-FFE7h |
| 11 | LINSCI2 | LINSCI2 Peripheral interrupts | SCI2CR1 | no | FFE4h-FFE5h | |
| 12 | LINSCI1 | LINSCI1 Peripheral interrupts (LIN Master/Slave) | SCI1CR1 | no ⁽⁴⁾ | FFE2h-FFE3h | |
| 13 | PWM ART | 8-bit PWM ART interrupts | PWMCR | yes | FFE0h-FFE1h | |

- Valid for HALT and ACTIVE HALT modes except for the MCC/RTC interrupt source which exits from ACTIVE HALT mode only.
- Except AVD interrupt.
- Exit from Halt only when a wake-up condition is detected, generating a Status Change interrupt. See [Section 16.6: Interrupts](#).
- It is possible to exit from Halt using the external interrupt which is mapped on the RDI pin.

6.6 External interrupts

6.6.1 I/O port interrupt sensitivity

The external interrupt sensitivity is controlled by the ISxx bits in the EICR register (*Figure 21*). This control allows up to four fully independent external interrupt source sensitivities.

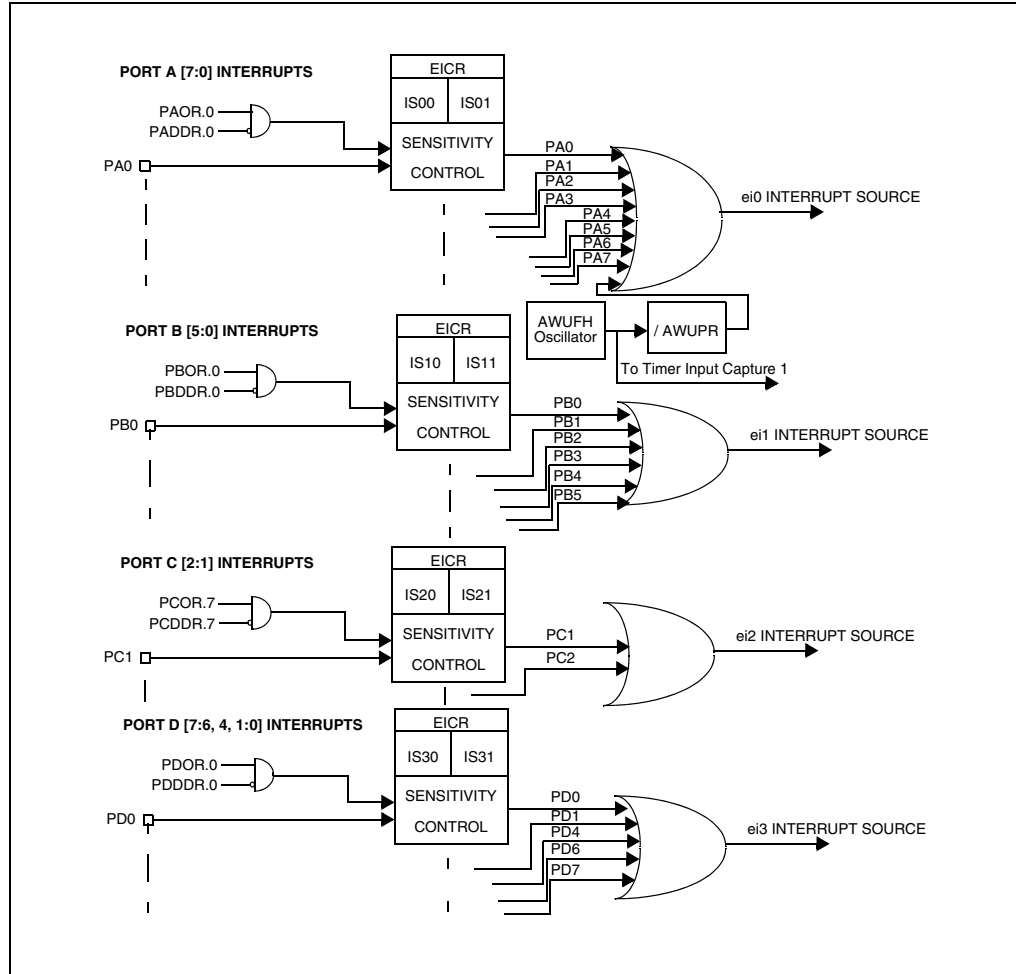
Each external interrupt source can be generated on four (or five) different events on the pin:

- Falling edge
- Rising edge
- Falling and rising edge
- Falling edge and low level

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] of the EICR.

Figure 21. External interrupt control bits



6.6.2 Register description

External interrupt control register 0 (EICR0)

Read/Write

Reset value: 0000 0000 (00h)

| | | | | | | | |
|------|------|------|------|------|------|------|------|
| 7 | 0 | | | | | | |
| IS31 | IS30 | IS21 | IS20 | IS11 | IS10 | IS01 | IS00 |

Bits 7:6 = **IS3[1:0]** *ei3 sensitivity*

The interrupt sensitivity, defined using the IS3[1:0] bits, is applied to the ei3 external interrupts:

Table 17. Interrupt sensitivity - ei3

| IS31 | IS30 | External interrupt sensitivity |
|------|------|--------------------------------|
| 0 | 0 | Falling edge and low level |
| | 1 | Rising edge only |
| 1 | 0 | Falling edge only |
| | 1 | Rising and falling edge |

These 2 bits can be written only when I1 and I0 of the CC register are both set to 1 (level 3).

Bits 5:4 = **IS2[1:0]** *ei2 sensitivity*

The interrupt sensitivity, defined using the IS2[1:0] bits, is applied to the ei2 external interrupts:

Table 18. Interrupt sensitivity - ei2

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

These 2 bits can be written only when I1 and I0 of the CC register are both set to 1 (level 3).

Bits 3:2 = **IS1[1:0]** *ei1 sensitivity*

The interrupt sensitivity, defined using the IS1[1:0] bits, is applied to the ei1 external interrupts:

Table 19. Interrupt sensitivity - ei1

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

Table 19. Interrupt sensitivity - ei1

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

These 2 bits can be written only when I1 and I0 of the CC register are both set to 1 (level 3).

Bits 1:0 = **IS0[1:0]** *ei0 sensitivity*

The interrupt sensitivity, defined using the IS0[1:0] bits, is applied to the ei0 external interrupts:

Table 20. Interrupt sensitivity - ei0

| IS01 | IS00 | External interrupt sensitivity |
|------|------|--------------------------------|
| 0 | 0 | Falling edge and low level |
| | 1 | Rising edge only |
| 1 | 0 | Falling edge only |
| | 1 | Rising and falling edge |

These 2 bits can be written only when I1 and I0 of the CC register are both set to 1 (level 3).

External Interrupt Control Register 1 (EICR1)

Read/Write

Reset value: 0000 0000 (00h)

| | | | | | | | |
|---|---|---|---|---|---|---|-----------|
| 7 | | | | | | | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | TLIS TLIE |

Bits 7:2 = Reserved

Bit 1 = **TLIS** *Top Level Interrupt sensitivity*

This bit configures the TLI edge sensitivity. It can be set and cleared by software only when TLIE bit is cleared.

- 0: Falling edge
- 1: Rising edge

Bit 0 = **TLIE** *Top Level Interrupt enable*

This bit allows to enable or disable the TLI capability on the dedicated pin. It is set and cleared by software.

- 0: TLI disabled
- 1: TLI enabled

Note: *A parasitic interrupt can be generated when clearing the TLIE bit.*

In some packages, the TLI pin is not available. In this case, the TLIE bit must be kept low to avoid parasitic TLI interrupts.

Table 21. Nested interrupts register map and reset values

| Address (Hex.) | Register label | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------------|-----------------------------|--------------|------------|------------|------------|------------|------------|------------|------------|
| 0025h | ISPR0 Reset value | ei1 | | ei0 | | CLKM | | TLI | |
| | | I1_3 1 | I0_3 1 | I1_2 1 | I0_2 1 | I1_1 1 | I0_1 1 | 1 | 1 |
| 0026h | ISPR1 Reset value | CAN TX/ER/SC | | CAN RX | | ei3 | | ei2 | |
| | | I1_7 1 | I0_7 1 | I1_6 1 | I0_6 1 | I1_5 1 | I0_5 1 | I1_4 1 | I0_4 1 |
| 0027h | ISPR2 Reset value | LINSICI 2 | | TIMER 16 | | TIMER 8 | | SPI | |
| | | I1_11 1 | I0_11 1 | I1_10 1 | I0_10 1 | I1_9 1 | I0_9 1 | I1_8 1 | I0_8 1 |
| 0028h | ISPR3 Reset value | | | | | ART | | LINSICI 1 | |
| | | 1 | 1 | 1 | 1 | I1_13 1 | I0_13 1 | I1_12 1 | I0_12 1 |
| 0029h | EICR0 Reset value | IS31 0 | IS30 0 | IS21 0 | IS20 0 | IS11 0 | IS10 0 | IS01 0 | IS00 0 |
| 002Ah | EICR1 Reset value | 0 | 0 | 0 | 0 | 0 | 0 | TLIS 0 | TLIE 0 |

7 Power saving modes

7.1 Introduction

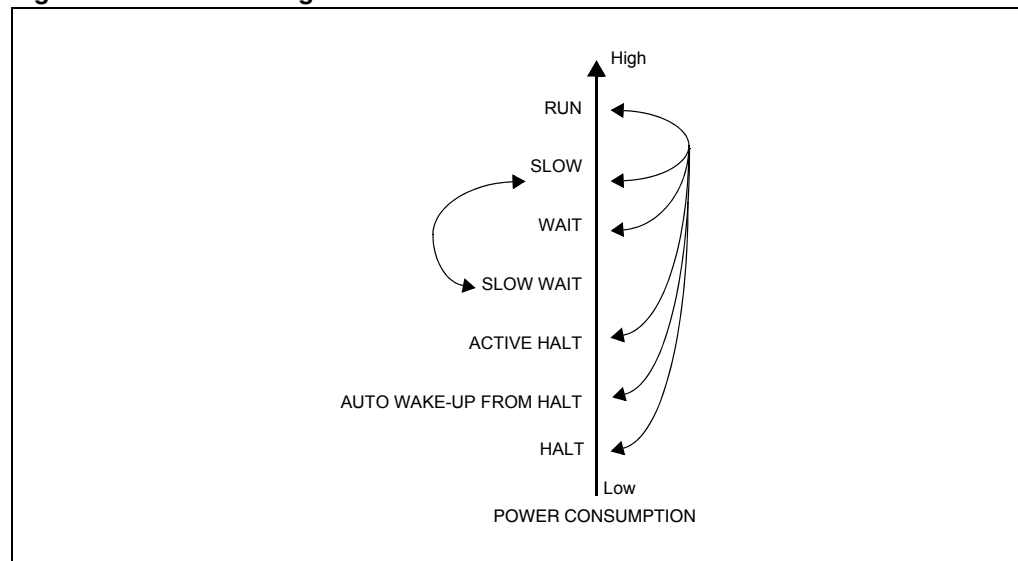
To give a large measure of flexibility to the application in terms of power consumption, five main power saving modes are implemented in the ST7 (see [Figure 22](#)):

- Slow
- Wait (and Slow-Wait)
- Active Halt
- Auto Wake-up From Halt (AWUFH)
- 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 22. Power saving mode transitions



7.2 Slow mode

This mode has two targets:

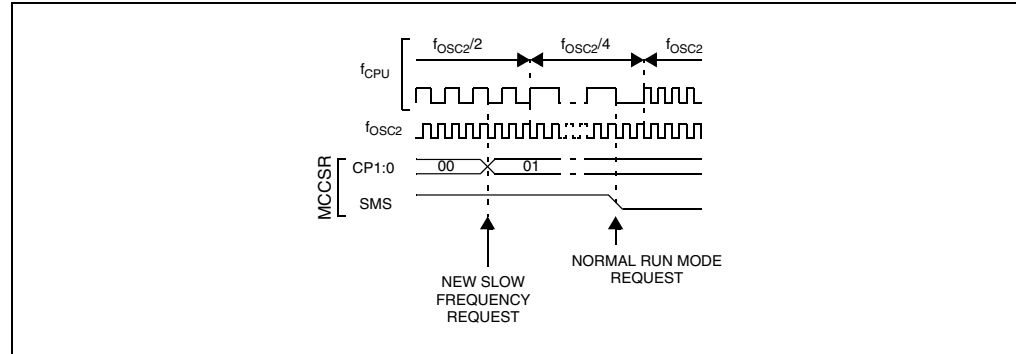
- To reduce power consumption by decreasing the internal clock in the device,
- To adapt the internal clock frequency (f_{CPU}) to the available supply voltage.

SLOW mode is controlled by three bits in the MCCR 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 by entering WAIT mode while the device is in SLOW mode.

Figure 23. SLOW mode clock transitions



7.3 Wait mode

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

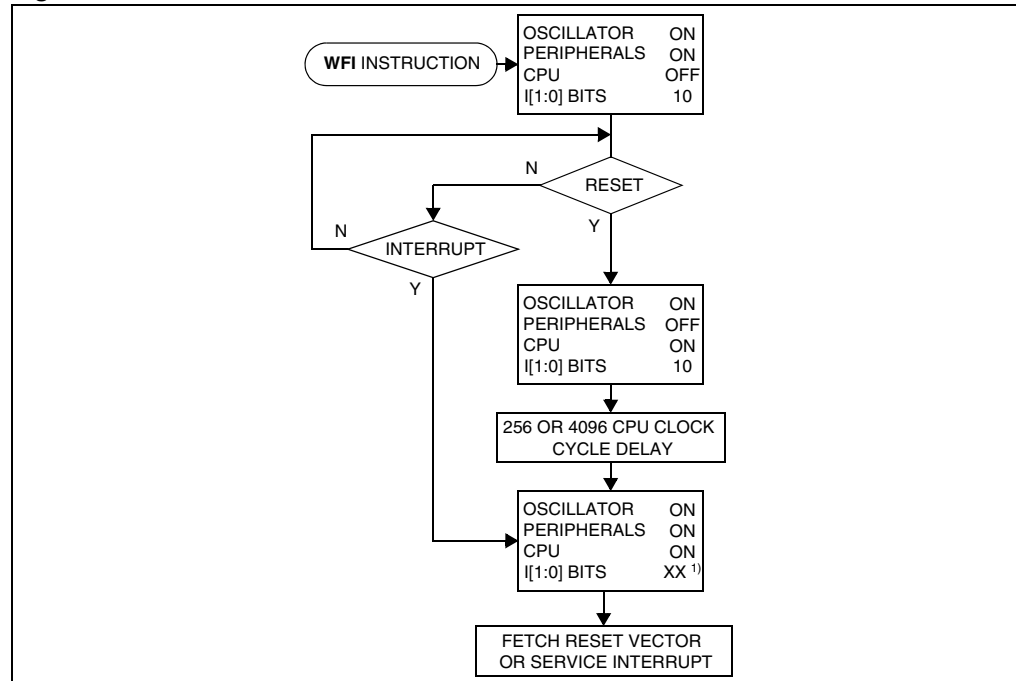
This power saving mode is selected by calling the 'WFI' instruction.

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 [Figure 24](#)

Figure 24. WAIT mode flow-chart



Note: 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.

7.4 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 10: Main clock controller with real time clock MCC/RTC](#) for more details on the MCCSR register) and when the AWUEN bit in the AWUCSR register is cleared.

The MCU can exit HALT mode on reception of either a specific interrupt (see [Table 16](#)) 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 26](#)).

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 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 22.1: Introduction](#) for more details).

Figure 25. HALT timing overview

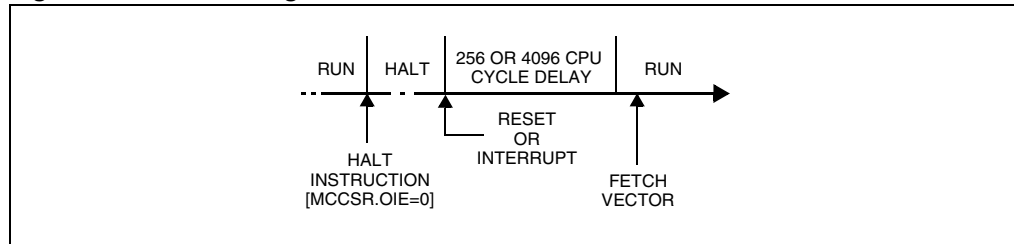
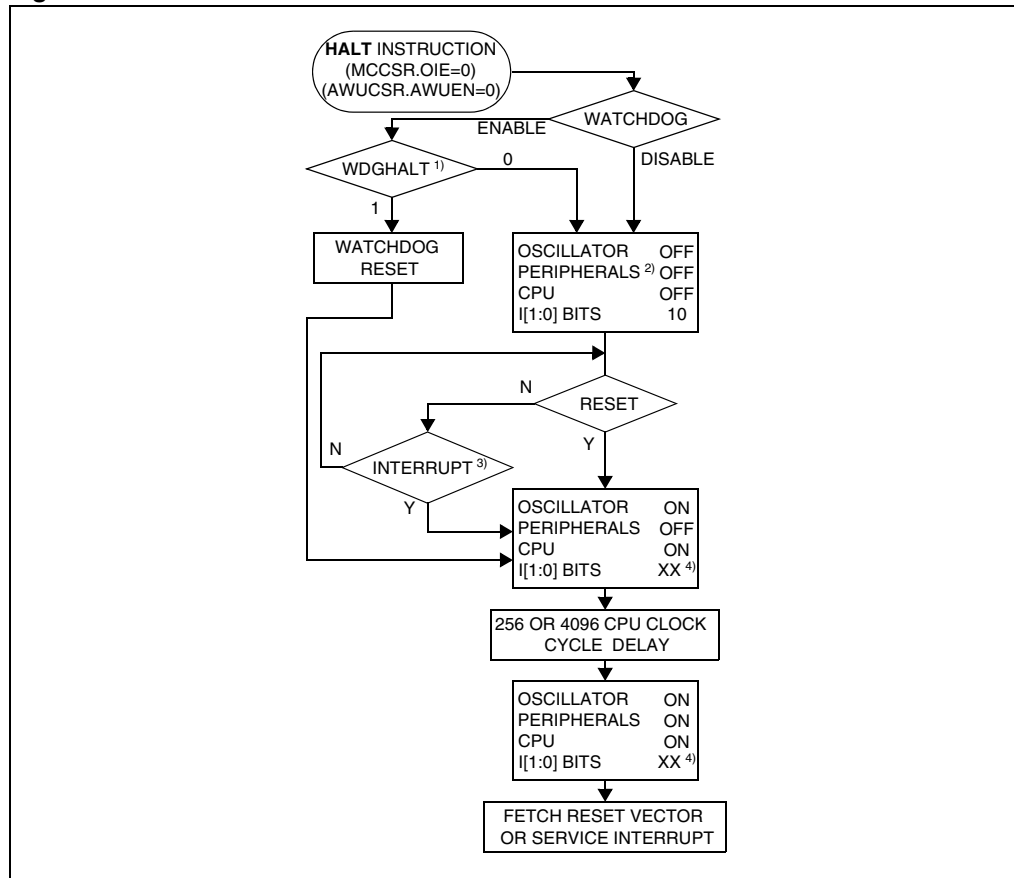


Figure 26. HALT mode flow-chart



- Note:
- 1 WDGHALT is an option bit. See option byte section 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 16](#) 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, reinitialize 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. For example, avoid defining a constant in ROM with the value 0x8E.
- 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).

7.5 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 MCC/RTC interrupt enable flag (OIE bit in MCCR register) is set and when the AWUEN bit in the AWUCSR register is cleared ([Section 7.6.1: Register description](#))

Table 22. MCC/RTC low power mode selection

| MCCR OIE bit | Power saving mode entered when HALT instruction is executed |
|-----------------|---|
| 0 | HALT mode |
| 1 | ACTIVE HALT mode |

The MCU can exit ACTIVE HALT mode on reception of the RTC interrupt and some specific interrupts (see [Table 16](#)) or a RESET. When exiting ACTIVE HALT mode by means of a RESET a 4096 or 256 CPU cycle delay occurs (depending on the option byte). 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 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 active halt is enabled, executing a HALT instruction while the Watchdog is active does not generate a RESET. This means that the device cannot spend more than a defined delay in this power saving mode.

Figure 27. ACTIVE HALT timing overview

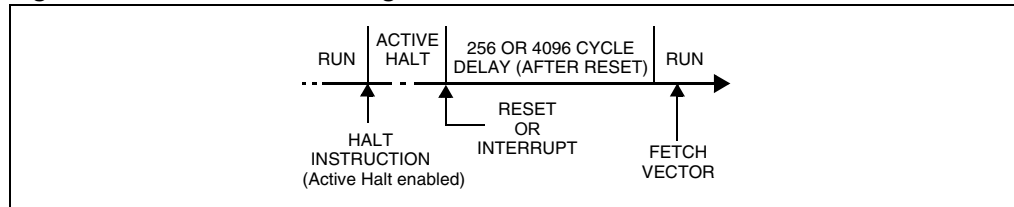
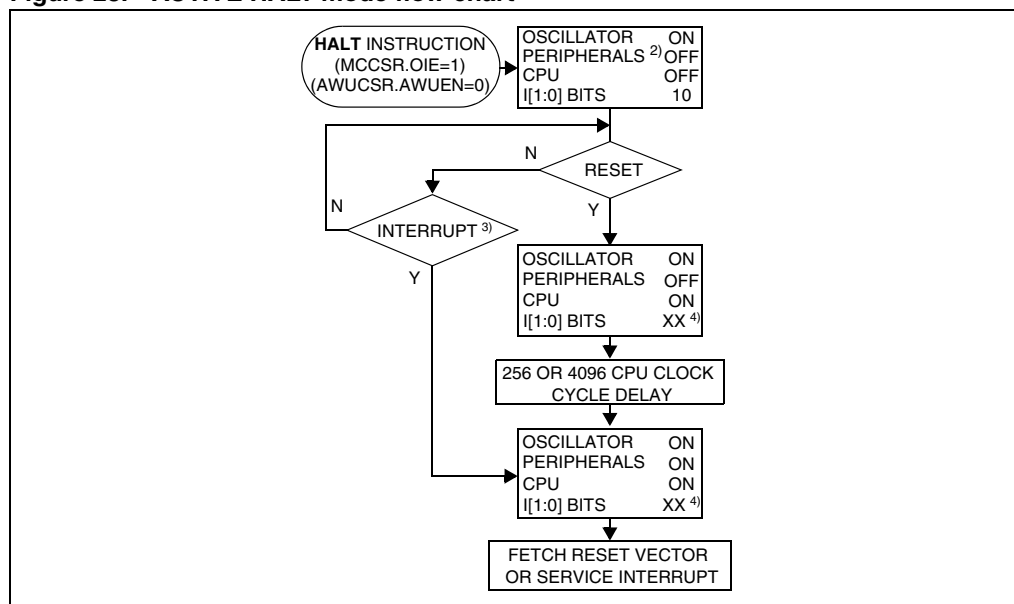


Figure 28. ACTIVE HALT mode flow-chart



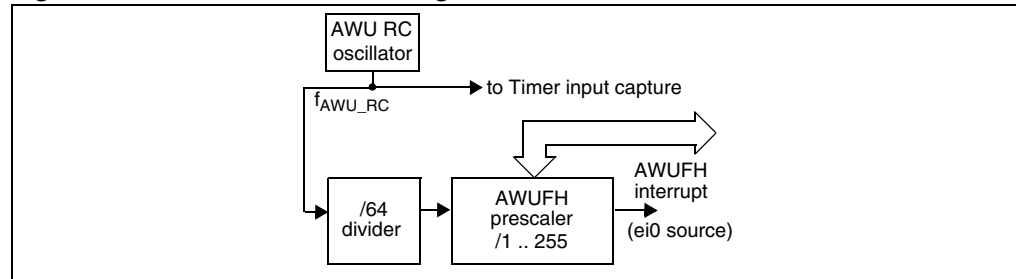
- Note:**
- 1 This delay occurs only if the MCU exits ACTIVE HALT mode by means of a RESET.
 - 2 Peripheral clocked with an external clock source can still be active.
 - 3 Only the RTC interrupt and some specific interrupts can exit the MCU from ACTIVE HALT mode (such as external interrupt). Refer to [Table 16](#) for more details.
 - 4 Before servicing an interrupt, the CC register is pushed on the stack. The I[1:0] bits in the CC register are set to the current software priority level of the interrupt routine and restored when the CC register is popped.

7.6 Auto wake-up from halt mode

Auto Wake-Up From Halt (AWUFH) mode is similar to Halt mode with the addition of an internal RC oscillator for wake-up. Compared to ACTIVE HALT mode, AWUFH has lower power consumption because the main clock is not kept running, but there is no accurate realtime clock available.

It is entered by executing the HALT instruction when the AWUEN bit in the AWUCSR register has been set and the OIE bit in the MCCSR register is cleared (see [Section 10: Main clock controller with real time clock MCC/RTC](#) for more details).

Figure 29. AWUFH mode block diagram



As soon as HALT mode is entered, and if the AWUEN bit has been set in the AWUCSR register, the AWU RC oscillator provides a clock signal (f_{AWU_RC}). Its frequency is divided by a fixed divider and a programmable prescaler controlled by the AWUPR register. The output of this prescaler provides the delay time. When the delay has elapsed the AWUF flag is set by hardware and an interrupt wakes up the MCU from Halt mode. At the same time the main oscillator is immediately turned on and a 256 or 4096 cycle delay is used to stabilize it. After this start-up delay, the CPU resumes operation by servicing the AWUFH interrupt. The AWU flag and its associated interrupt are cleared by software reading the AWUCSR register.

To compensate for any frequency dispersion of the AWU RC oscillator, it can be calibrated by measuring the clock frequency f_{AWU_RC} and then calculating the right prescaler value. Measurement mode is enabled by setting the AWUM bit in the AWUCSR register in Run mode. This connects f_{AWU_RC} to the ICAP1 input of the 16-bit timer, allowing the f_{AWU_RC} to be measured using the main oscillator clock as a reference time base.

Similarities with halt mode

The following AWUFH mode behavior is the same as normal Halt mode:

- The MCU can exit AWUFH mode by means of any interrupt with exit from Halt capability or a reset (see [Section 7.4: Halt mode](#)).
- When entering AWUFH 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 AWUFH mode, the main oscillator is turned off causing all internal processing to be stopped, including the operation of the on-chip peripherals. None of the peripherals are clocked except those which get their clock supply from another clock generator (such as an external or auxiliary oscillator like the AWU oscillator).
- The compatibility of Watchdog operation with AWUFH mode is configured by the WDGHALT option bit in the option byte. Depending on this setting, the HALT instruction when executed while the Watchdog system is enabled, can generate a Watchdog RESET.

Figure 30. AWUF halt timing diagram

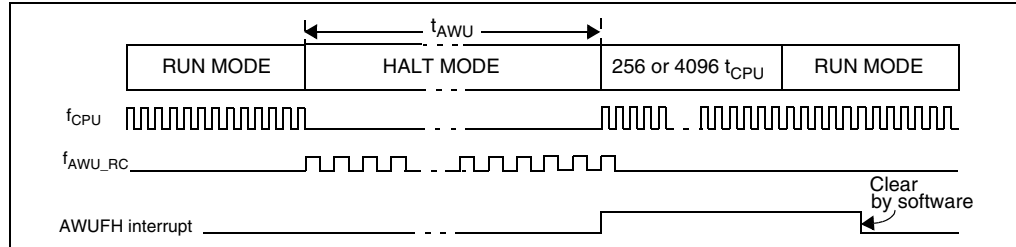
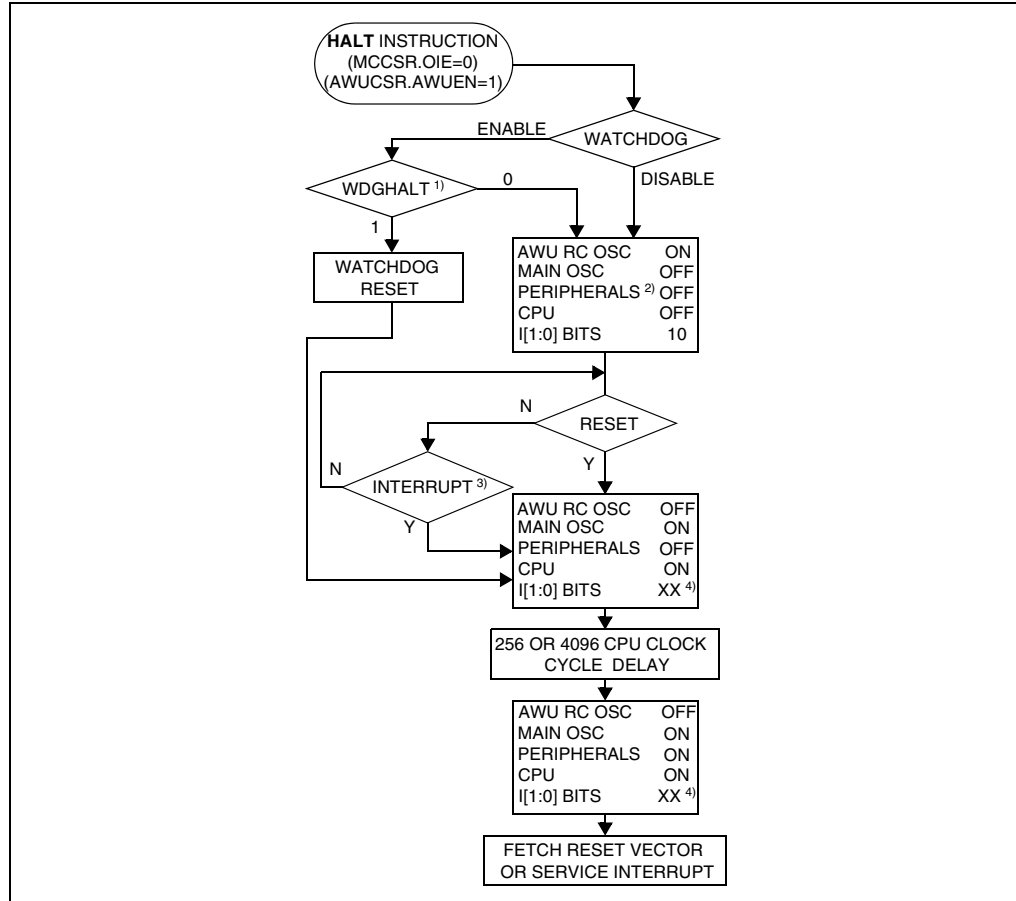


Figure 31. AWUFH mode flow-chart



- Note:
- 1 WDGHALT is an option bit. See option byte section for more details.
 - 2 Peripheral clocked with an external clock source can still be active.
 - 3 Only an AWUFH interrupt and some specific interrupts can exit the MCU from HALT mode (such as external interrupt). Refer to [Table 16](#) 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.

7.6.1 Register description

AWUFH control/status register (AWUCSR)

Read/Write (except bit 2 read only)

Reset value: 0000 0000 (00h)

| | | | | | | | | |
|---|---|---|---|---|---|------|------|-------|
| 7 | | | | | | | 0 | |
| 0 | 0 | 0 | 0 | 0 | 0 | AWUF | AWUM | AWUEN |

Bits 7:3 = Reserved.

Bit 2 = **AWUF** *Auto Wake-Up Flag*

This bit is set by hardware when the AWU module generates an interrupt and cleared by software on reading AWUCSR.

- 0: No AWU interrupt occurred
- 1: AWU interrupt occurred

Bit 1 = **AWUM** *Auto Wake-Up Measurement*

This bit enables the AWU RC oscillator and connects its output to the ICAP1 input of the 16-bit timer. This allows the timer to be used to measure the AWU RC oscillator dispersion and then compensate this dispersion by providing the right value in the AWUPR register.

- 0: Measurement disabled
- 1: Measurement enabled

Bit 0 = **AWUEN** *Auto Wake-Up From Halt Enabled*

This bit enables the Auto Wake-Up From Halt feature: once HALT mode is entered, the AWUFH wakes up the microcontroller after a time delay defined by the AWU prescaler value. It is set and cleared by software.

- 0: AWUFH (Auto Wake-Up From Halt) mode disabled
- 1: AWUFH (Auto Wake-Up From Halt) mode enabled

AWUFH prescaler register (AWUPR)

Read/Write

Reset value: 1111 1111 (FFh)

| | | | | | | | |
|--------|--------|--------|--------|--------|--------|--------|--------|
| 7 | | | | | | | 0 |
| AWUPR7 | AWUPR6 | AWUPR5 | AWUPR4 | AWUPR3 | AWUPR2 | AWUPR1 | AWUPR0 |

Bits 7:0 = **AWUPR[7:0]** *Auto Wake-Up Prescaler*

These 8 bits define the AWUPR Dividing factor as explained below:

Table 23. AWUPR prescaler

| AWUPR[7:0] | Dividing factor |
|------------|----------------------|
| 00h | Forbidden (See note) |
| 01h | 1 |
| ... | ... |

Table 23. AWUPR prescaler (continued)

| AWUPR[7:0] | Dividing factor |
|------------|-----------------|
| FEh | 254 |
| FFh | 255 |

In AWU mode, the period that the MCU stays in Halt Mode (t_{AWU} in [Figure 30](#)) is defined by

$$t_{AWU} = 64 \times AWUP \times \frac{1}{t_{AWURC}} + t_{RCSTRT}$$

This prescaler register can be programmed to modify the time that the MCU stays in Halt mode before waking up automatically.

Note: If 00h is written to AWUPR, depending on the product, an interrupt is generated immediately after a HALT instruction or the AWUPR remains unchanged.

Table 24. AWU register map and reset values

| Address (Hex.) | Register Label | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------------|-----------------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| 002Bh | AWUCSR Reset value | 0 | 0 | 0 | 0 | 0 | AWUF 0 | AWUM 0 | AWUEN 0 |
| 002Ch | AWUPR Reset value | AWUPR7 1 | AWUPR6 1 | AWUPR5 1 | AWUPR4 1 | AWUPR3 1 | AWUPR2 1 | AWUPR1 1 | AWUPR0 1 |

8 I/O ports

8.1 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 8 pins. Each pin can be programmed independently as digital input (with or without interrupt generation) or digital output.

8.2 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 the I/O Port Implementation section). The generic I/O block diagram is shown in [Figure 32](#)

8.2.1 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.

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.

8.2.2 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.

Table 25. DR register value and output pin status

| DR | Push-pull | Open-drain |
|----|-----------------|-----------------|
| 0 | V _{SS} | V _{SS} |
| 1 | V _{DD} | Floating |

8.2.3 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.

Figure 32. I/O port general block diagram

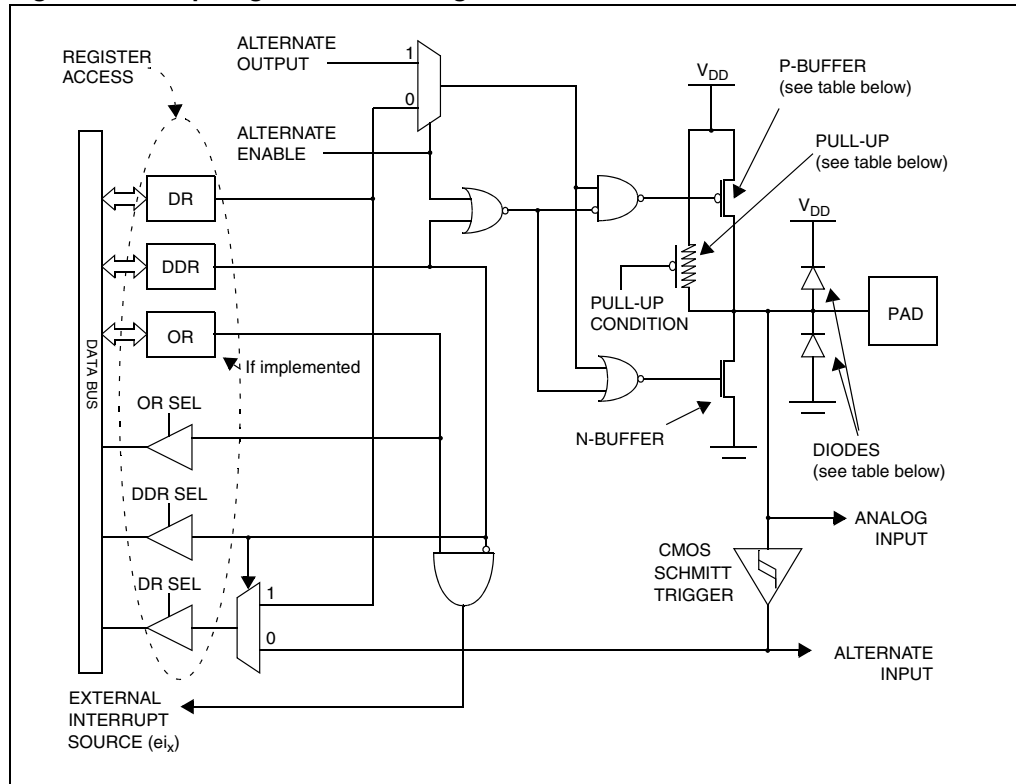


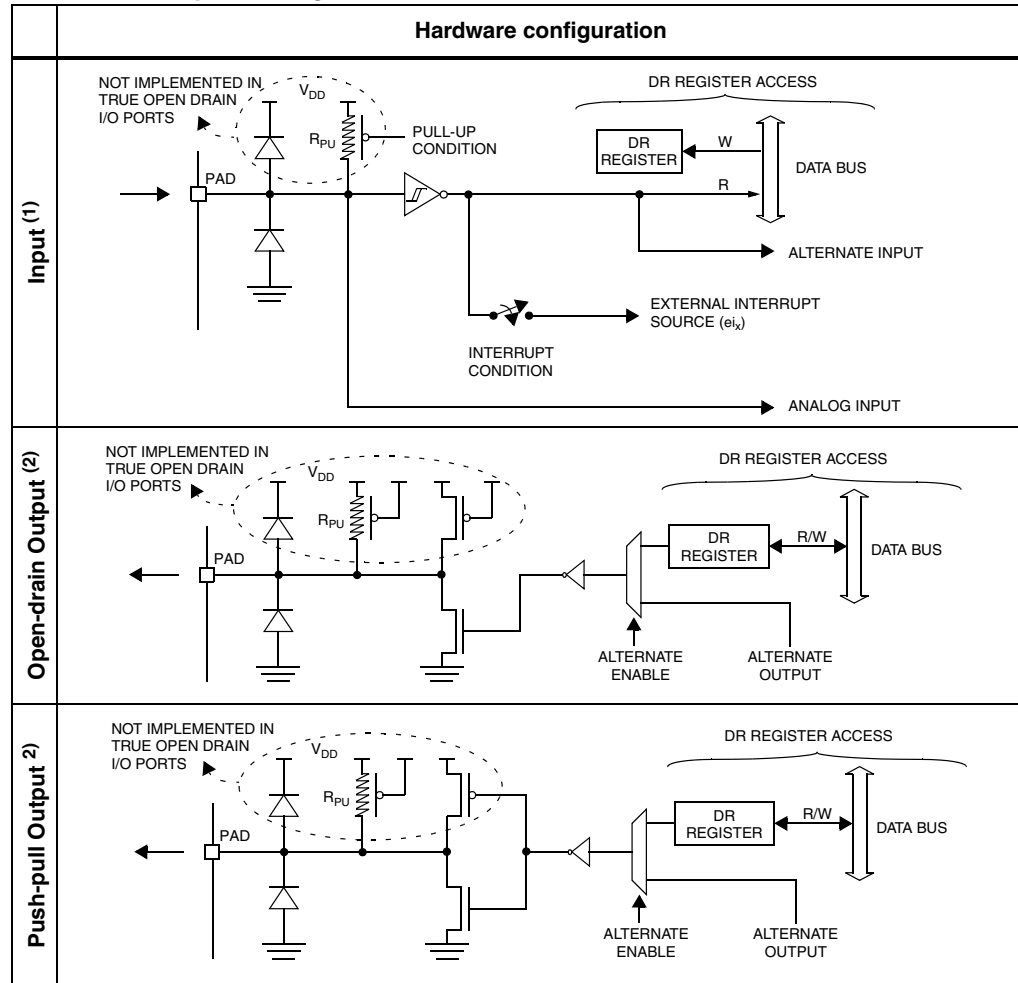
Table 26. I/O port mode options⁽¹⁾

| Configuration mode | | Pull-Up | P-Buffer | Diodes | |
|--------------------|---------------------------------|---------|----------|---------------|----|
| | | | | | |
| Input | Floating with/without Interrupt | Off | Off | On | On |
| | Pull-up with/without Interrupt | On | | | |
| Output | Push-pull | Off | On | On | On |
| | Open Drain (logic level) | | Off | | |
| | True Open Drain | NI | NI | NI (see note) | |

1. NI - not implemented
 Off - implemented not activated
 On - implemented and activated

Note: 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.

Table 27. I/O port configurations



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.

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.

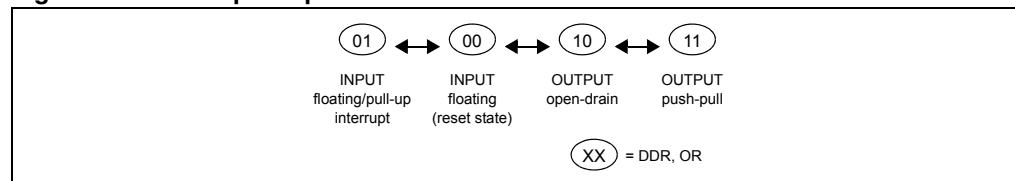
Warning: The analog input voltage level must be within the limits stated in the absolute maximum ratings.

8.3 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 33](#). Other transitions are potentially risky and should be avoided, since they are likely to present unwanted side-effects such as spurious interrupt generation.

Figure 33. Interrupt I/O port state transitions



8.4 I/O port register configurations

The I/O port register configurations are summarized as follows.

8.4.1 Standard ports

Table 28. Configuration of PB7:6, PC0, PC3, PC7:5, PD3:2, PD5, PE7:0, PF7:0

| Mode | DDR | OR |
|-------------------|-----|----|
| Floating input | 0 | 0 |
| Pull-up input | | 1 |
| Open drain output | 1 | 0 |
| Push-pull output | | 1 |

8.4.2 Interrupt ports

Table 29. Configuration of PA0, 2, 4, 6; PB0, 2,4; PC1; PD0,6 (with pull-up)

| Mode | DDR | OR |
|-------------------------|-----|----|
| Floating input | 0 | 0 |
| Pull-up interrupt input | | 1 |
| Open drain output | 1 | 0 |
| Push-pull output | | 1 |

Table 30. Configuration of PA1, 3, 5, 7; PB1,3,5; PC2; PD1, 4, 7 (without pull-up)

| Mode | DDR | OR |
|--------------------------|-----|----|
| Floating input | 0 | 0 |
| Floating interrupt input | | 1 |
| Open drain output | 1 | 0 |
| Push-pull output | | 1 |

8.4.3 Pull-up input port (CANTX requirement)

Table 31. Configuration of PC4

| Mode |
|---------------|
| pull-up input |

The PC4 port cannot operate as a general purpose output. The CAN peripheral controls it directly when enabled. Otherwise, PC4 is a pull-up input.

If DDR = 1 it is still possible to read the port through the DR register.

Table 32. Port configuration

| Port | Pin name | Input | | Output | |
|--------|----------|----------|--------------------------|------------------------------------|------------|
| | | OR = 0 | OR = 1 | OR = 0 | OR = 1 |
| Port A | PA0 | floating | pull-up interrupt (ei0) | open drain | push-pull |
| | PA1 | | floating interrupt (ei0) | | |
| | PA2 | | pull-up interrupt (ei0) | | |
| | PA3 | | floating interrupt (ei0) | | |
| | PA4 | | pull-up interrupt (ei0) | | |
| | PA5 | | floating interrupt (ei0) | | |
| | PA6 | | pull-up interrupt (ei0) | | |
| | PA7 | | floating interrupt (ei0) | | |
| Port B | PB0 | floating | pull-up interrupt (ei1) | open drain | push-pull |
| | PB1 | | floating interrupt (ei1) | | |
| | PB2 | | pull-up interrupt (ei1) | | |
| | PB3 | | floating interrupt (ei1) | | |
| | PB4 | | pull-up interrupt (ei1) | | |
| | PB5 | | floating interrupt (ei1) | | |
| Port C | PC0 | floating | pull-up | open drain | push-pull |
| | PC1 | | pull-up interrupt (ei2) | | |
| | PC2 | | floating interrupt (ei2) | | |
| | PC3 | | pull-up | | |
| | PC4 | | pull-up | controlled by CANTX ⁽¹⁾ | |
| | PC7:5 | floating | | pull-up | open drain |

Table 32. Port configuration (continued)

| Port | Pin name | Input | | Output | |
|--------|----------|----------------|--------------------------|------------|-----------|
| | | OR = 0 | OR = 1 | OR = 0 | OR = 1 |
| Port D | PD0 | floating | pull-up interrupt (ei3) | open drain | push-pull |
| | PD1 | | floating interrupt (ei3) | | |
| | PD3:2 | | pull-up | | |
| | PD4 | | floating interrupt (ei3) | | |
| | PD5 | | pull-up | | |
| | PD6 | | pull-up interrupt (ei3) | | |
| | PD7 | | floating interrupt (ei3) | | |
| Port E | PE7:0 | floating (TTL) | pull-up (TTL) | open drain | push-pull |
| Port F | PF7:0 | floating (TTL) | pull-up (TTL) | open drain | push-pull |

1. When the CANTX alternate function is selected, the I/O port operates in output push-pull mode.

8.5 Low power modes

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

| Mode | Description |
|------|--|
| 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. |

8.6 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 | |

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 IO port registers | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0000h | PADR | MSB | | | | | | | LSB |
| 0001h | PADDR | | | | | | | | |
| 0002h | PAOR | | | | | | | | |
| 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 | | | | | | | | |

9 Window watchdog (WWDG)

9.1 Introduction

The Window Watchdog 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 contents of the downcounter before the T6 bit becomes cleared. An MCU reset is also generated if the 7-bit downcounter value (in the control register) is refreshed before the downcounter has reached the window register value. This implies that the counter must be refreshed in a limited window.

9.2 Main features

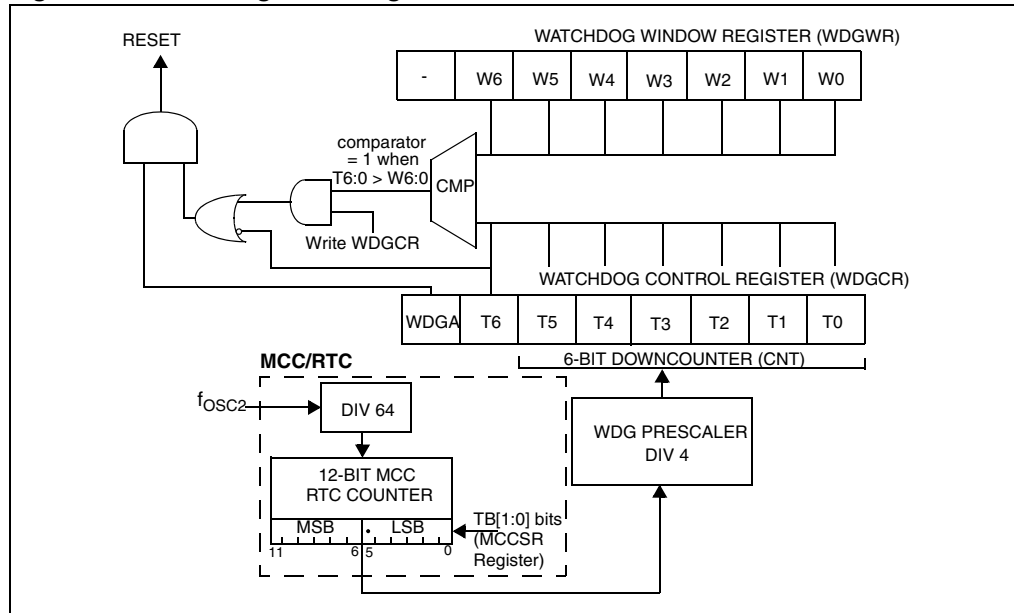
- Programmable free-running down counter
- Conditional reset
 - Reset (if watchdog activated) when the downcounter value becomes less than 40h
 - Reset (if watchdog activated) if the downcounter is reloaded outside the window (see [Figure 37](#))
- Hardware/Software Watchdog activation (selectable by option byte)
- Optional reset on HALT instruction (configurable by option byte)

9.3 Functional description

The counter value stored in the WDGCR register (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 downcounter (T[6:0] bits) rolls over from 40h to 3Fh (T6 becomes cleared), it initiates a reset cycle pulling low the reset pin for typically 30 μ s. If the software reloads the counter while the counter is greater than the value stored in the window register, then a reset is generated.

Figure 34. Watchdog block diagram



The application program must write in the WDGCR register at regular intervals during normal operation to prevent an MCU reset. This operation must occur only when the counter value is lower than the window register value. The value to be stored in the WDGCR register must be between FFh and C0h (see [Figure 35](#)):

- **Enabling the watchdog:**
when Software Watchdog is selected (by option byte), the watchdog is disabled after a reset. It is enabled by setting the WDGA bit in the WDGCR register, then it cannot be disabled again except by a reset.
When Hardware Watchdog is selected (by option byte), the watchdog is always active and the WDGA bit is not used.
- **Controlling the downcounter:**
this downcounter is free-running: It counts down even if the watchdog is disabled. When the watchdog is enabled, the T6 bit must be 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 35](#)). 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 36](#)).
The window register (WDGWR) contains the high limit of the window: To prevent a reset, the downcounter must be reloaded when its value is lower than the window register value and greater than 3Fh. [Figure 37](#) describes the window watchdog process.

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

- **Watchdog Reset on Halt option:**
if the watchdog is activated and the watchdog reset on halt option is selected, then the HALT instruction will generate a Reset.

9.4 Using halt mode with the WDG

If Halt mode with Watchdog is enabled by option byte (no watchdog reset on HALT instruction), it is recommended before executing the HALT instruction to refresh the WDG counter, to avoid an unexpected WDG reset immediately after waking up the microcontroller.

9.5 How to program the watchdog timeout

Figure 35 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 36*.

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

Figure 35. Approximate timeout duration

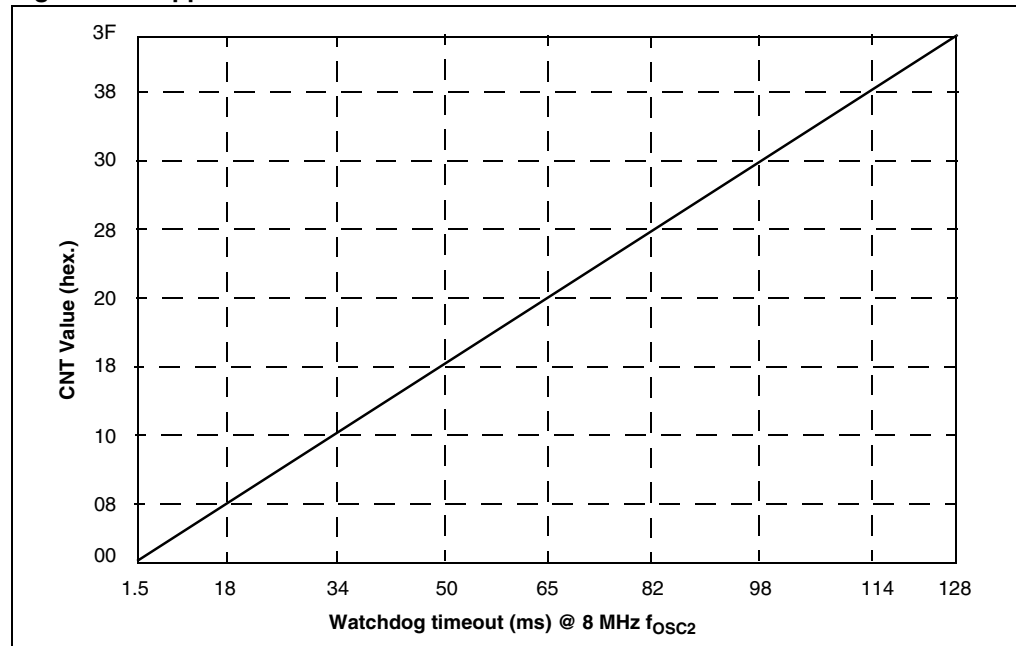


Figure 36. Exact timeout duration (t_{min} and t_{max})**WHERE:**

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

$$t_{max0} = 16384 \times t_{OSC2}$$

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

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 MCCR register

| TB1 Bit (MCCR Reg.) | TB0 Bit (MCCR Reg.) | Selected MCCR 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}):

$$\text{IF } CNT < \left\lfloor \frac{MSB}{4} \right\rfloor \text{ THEN } t_{min} = t_{min0} + 16384 \times CNT \times t_{osc2}$$

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

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

$$\text{IF } CNT \leq \left\lfloor \frac{MSB}{4} \right\rfloor \text{ THEN } t_{max} = t_{max0} + 16384 \times CNT \times t_{osc2}$$

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

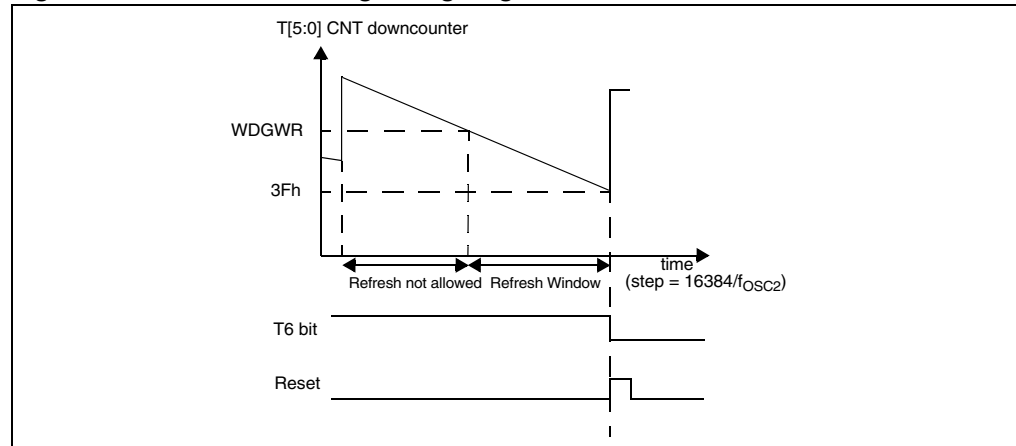
Note: In the above formulae, division results must be rounded down to the next integer value.

Example 1:

With 2ms timeout selected in MCCR 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 |

Figure 37. Window watchdog timing diagram



9.6 Low power modes

Table 36. Effect of low power modes on WDG

| Mode | Description | | |
|-------------|--|----------------------------|--|
| SLOW | No effect on Watchdog: the downcounter continues to decrement at normal speed. | | |
| WAIT | No effect on Watchdog: the downcounter continues to decrement. | | |
| 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. If an interrupt is received (refer to interrupt table mapping to see interrupts which can occur in halt mode), 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 9.8: Using halt mode with the WDG (WDGHALT option) below. |
| | 0 | 1 | A reset is generated instead of entering halt mode. |
| ACTIVE HALT | 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. |

9.7 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.

9.8 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.

9.9 Interrupts

None.

9.10 Register description

9.10.1 Control register (WDGCR)

Read/Write

Reset value: 0111 1111 (7Fh)

| | | | | | | | |
|------|----|----|----|----|----|----|----|
| 7 | | | | | | | 0 |
| WDGA | T6 | T5 | T4 | T3 | T2 | T1 | T0 |

Bit 7 = **WDGA** Activation bit.

This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset.

- 0: Watchdog disabled
- 1: Watchdog enabled

Note: This bit is not used if the hardware watchdog option is enabled by option byte.

Bits 6:0 = **T[6:0]** 7-bit counter (MSB to LSB).

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).

9.10.2 Window Register (WDGWR)

Read/ write

Reset value: 0111 1111 (7Fh)

| | | | | | | | |
|---|----|----|----|----|----|----|----|
| 7 | | | | | | | 0 |
| - | W6 | W5 | W4 | W3 | W2 | W1 | W0 |

Bit 7 = Reserved

Bits 6:0 = **W[6:0]** 7-bit window value

These bits contain the window value to be compared to the downcounter.

Table 37. Watchdog timer register map and reset values

| Address (Hex.) | Register label | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------------|----------------------|-----------|---------|---------|---------|---------|---------|---------|---------|
| 2F | WDGCR Reset value | WDGA 0 | T6 1 | T5 1 | T4 1 | T3 1 | T2 1 | T1 1 | T0 1 |
| 30 | WDGWR Reset value | - 0 | W6 1 | W5 1 | W4 1 | W3 1 | W2 1 | W1 1 | W0 1 |

10 Main clock controller with real time clock MCC/RTC

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.

10.1 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 7.2: 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.

10.2 Clock-out capability

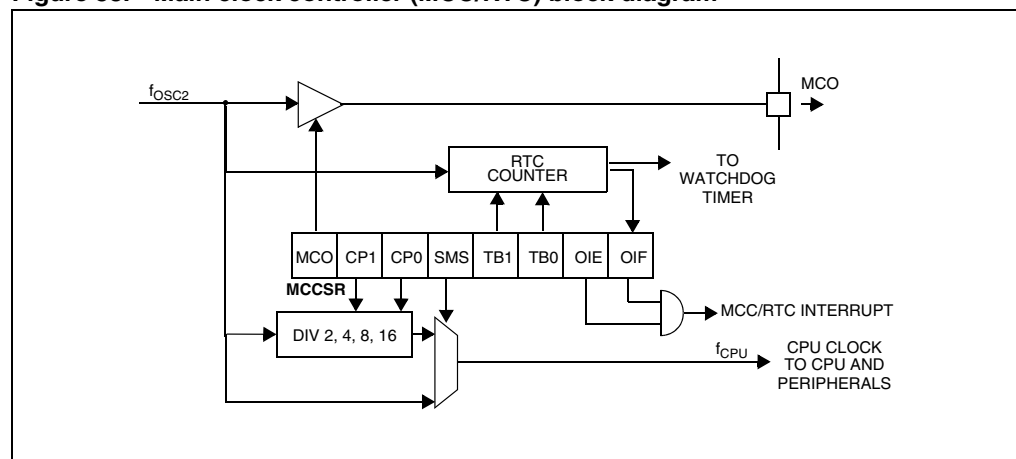
The clock-out capability is an alternate function of an I/O port pin that outputs a f_{OSC2} clock to drive external devices. It is controlled by the MCO bit in the MCCSR register.

10.3 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 4 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 7.5: Active halt mode](#) for more details.

Figure 38. Main clock controller (MCC/RTC) block diagram



10.4 Low power modes

Table 38. Effect of low power modes on MCC/RTC

| Mode | Description |
|--------------------------|---|
| WAIT | No effect on MCC/RTC peripheral. MCC/RTC interrupt cause the device to exit from WAIT mode. |
| ACTIVE HALT | No effect on MCC/RTC counter (OIE bit is set), the registers are frozen. MCC/RTC interrupt cause the device to exit from ACTIVE HALT mode. |
| HALT and AWUF HALT | MCC/RTC counter and registers are frozen. MCC/RTC operation resumes when the MCU is woken up by an interrupt with "exit from HALT" capability. |

10.5 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 39. 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⁾ |

Note: The MCC/RTC interrupt wakes up the MCU from ACTIVE HALT mode, not from HALT or AWUF HALT mode.

10.6 Register description

10.6.1 MCC control/status register (MCCSR)

Read/Write

Reset value: 0000 0000 (00h)

| | |
|-----|-----|
| 7 | 0 |
| MCO | OIF |

Bit 7 = **MCO** Main clock out selection

This bit enables the MCO alternate function on the corresponding I/O port. It is set and cleared by software.

- 0: MCO alternate function disabled (I/O pin free for general-purpose I/O)
- 1: MCO alternate function enabled (f_{OSC2} on I/O port)

Bits 6:5 = **CP[1:0]** CPU clock prescaler

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

Table 40. CPU clock frequency in SLOW mode

| f_{CPU} in SLOW mode | CP1 | CP0 |
|-------------------------------|-----|-----|
| $f_{\text{OSC2}} / 2$ | 0 | 0 |
| $f_{\text{OSC2}} / 4$ | | 1 |
| $f_{\text{OSC2}} / 8$ | 1 | 0 |
| $f_{\text{OSC2}} / 16$ | | 1 |

Bit 4 = **SMS** *Slow mode select*

This bit is set and cleared by software.

0: Normal mode. $f_{\text{CPU}} = f_{\text{OSC2}}$

1: Slow mode. f_{CPU} is given by CP1, CP0

See [Section 7.2: Slow mode](#) and [Section 10: Main clock controller with real time clock MCC/RTC](#) for more details.

Bits 3:2 = **TB[1:0]** *Time base control*

These bits select the programmable divider time base. They are set and cleared by software.

Table 41. Time base selection

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

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.

Bit 1 = **OIE** *Oscillator interrupt enable*

This bit set and cleared by software.

0: Oscillator interrupt disabled

1: Oscillator interrupt enabled

This interrupt can be used to exit from ACTIVE HALT mode.

When this bit is set, calling the ST7 software HALT instruction enters the ACTIVE HALT power saving mode

Bit 0 = **OIF** *Oscillator interrupt flag*

This bit is set by hardware and cleared by software reading the CSR register. It indicates when set that the main oscillator has reached the selected elapsed time (TB1:0).

0: Timeout not reached

1: Timeout reached

Caution: The BRES and BSET instructions must not be used on the MCCSR register to avoid unintentionally clearing the OIF bit.

Table 42. Main clock controller register map and reset values

| Address (Hex.) | Register label | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------------|----------------------|----------|----------|----------|----------|----------|----------|----------|------------|
| 002Dh | SICSR Reset value | 0 | AVDIE | AVDF | LVDRF | 0 | 0 | 0 | WDGRF x |
| 002Eh | MCCSR Reset value | MCO 0 | CP1 0 | CP0 0 | SMS 0 | TB1 0 | TB0 0 | OIE 0 | OIF 0 |

11 PWM auto-reload timer (ART)

11.1 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.

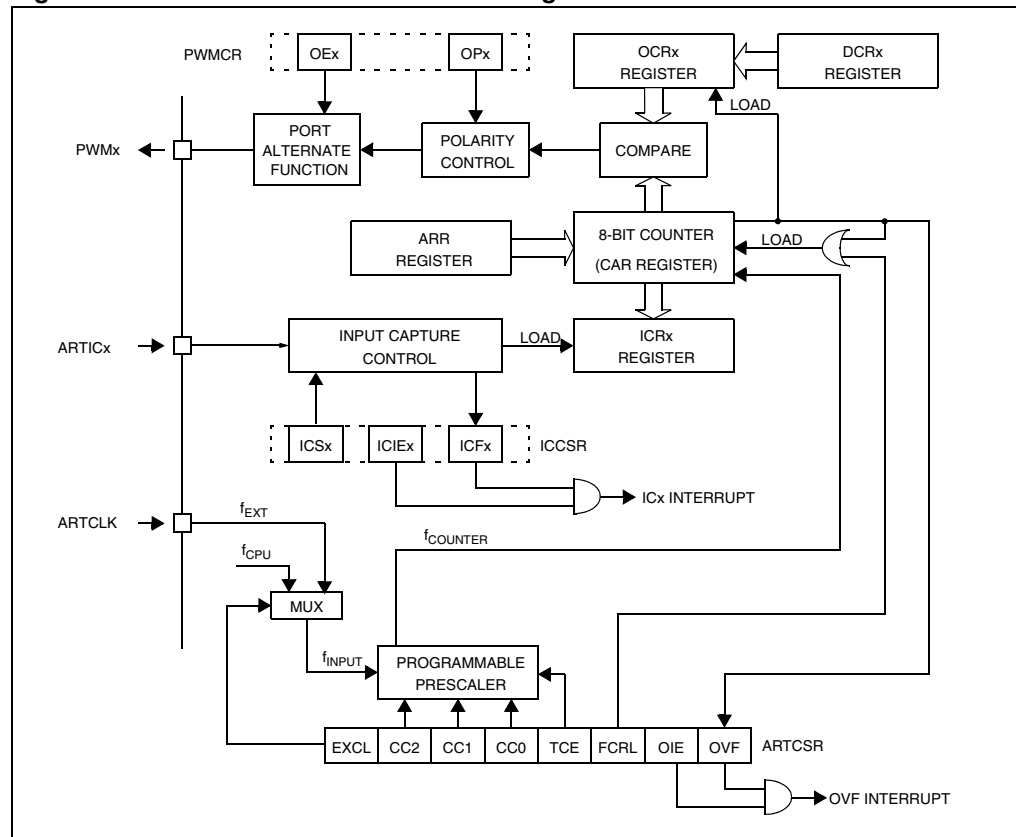
These resources allow five possible operating modes:

- Generation of up to four independent PWM signals
- Output compare and Time base interrupt
- Up to two input capture functions
- External event detector
- Up to two 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 39. PWM auto-reload timer block diagram



11.2 Functional description

11.2.1 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).

11.2.2 Counter clock and prescaler

The counter clock frequency is given by:

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

The timer counter's input clock (f_{INPUT}) feeds the 7-bit programmable prescaler, which selects one of the eight 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 $n = 0, 1 \dots 7$).

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.

11.2.3 Counter and prescaler Initialization

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

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.

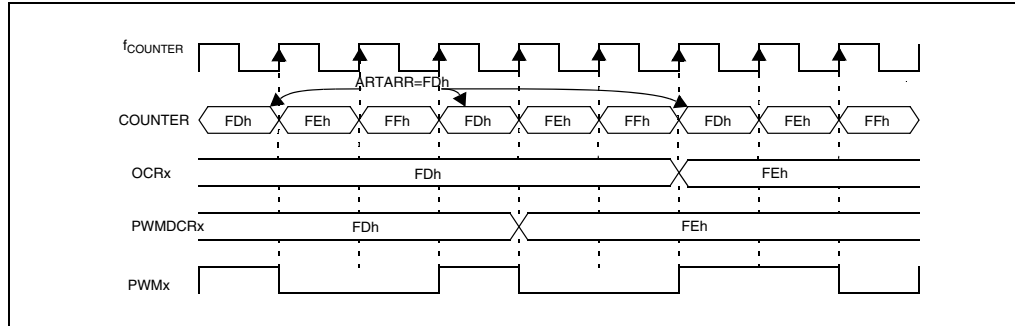
Direct access to the prescaler is not possible.

11.2.4 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.

Figure 40. Output compare control



11.2.5 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.

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

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:

$$\text{Resolution} = 1 / (256 - \text{ARTARR})$$

Note: To have 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.

Figure 41. PWM auto-reload timer function

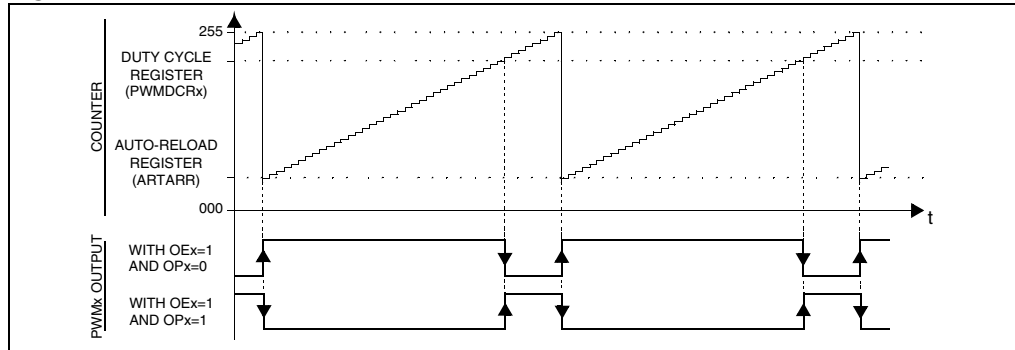
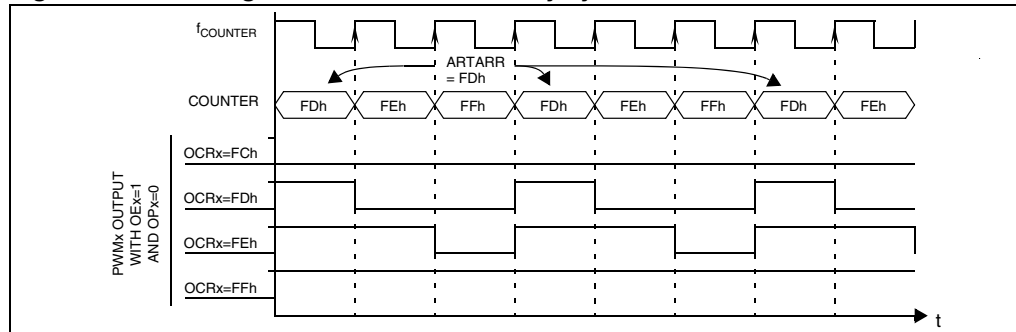


Figure 42. PWM signal from 0% to 100% duty cycle



11.2.6 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.

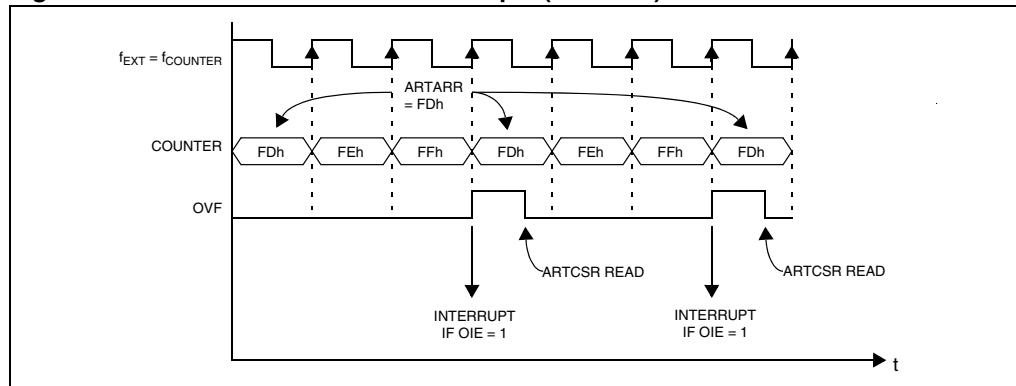
11.2.7 External clock and event detector mode

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.

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

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 43. External event detector example (3 counts)



11.2.8 Input capture function

Input Capture 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 the next 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, 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 ($1/f_{\text{COUNTER}}$).

Note: *During HALT mode, input capture is inhibited (the ARTICRx is never reloaded) and only the external interrupt capability can be used.*

The ARTICx signal is synchronized on CPU clock. It takes two rising edges until ARTICRx is latched with the counter value. Depending on the prescaler value and the time when the ICAP event occurs, the value loaded in the ARTICRx register may be different.

If the counter is clocked with the CPU clock, the value latched in ARTICRx is always the next counter value after the event on ARTICx occurred ([Figure 44](#)).

If the counter clock is prescaled, it depends on the position of the ARTICx event within the counter cycle ([Figure 45](#)).

Figure 44. Input capture timing diagram, $f_{\text{COUNTER}} = f_{\text{CPU}}$

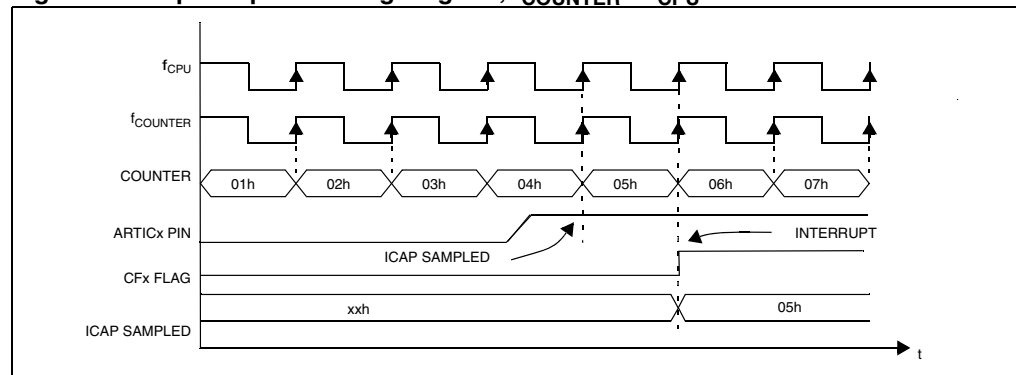
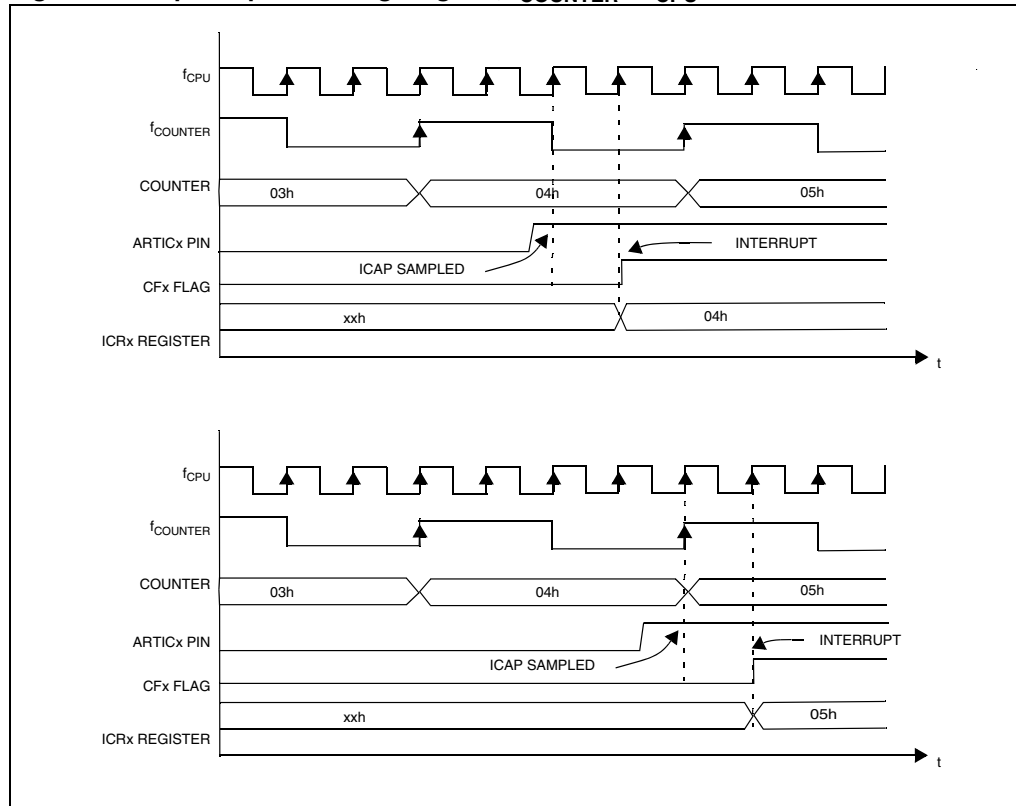


Figure 45. Input capture timing diagram, $f_{\text{COUNTER}} = f_{\text{CPU}} / 4$



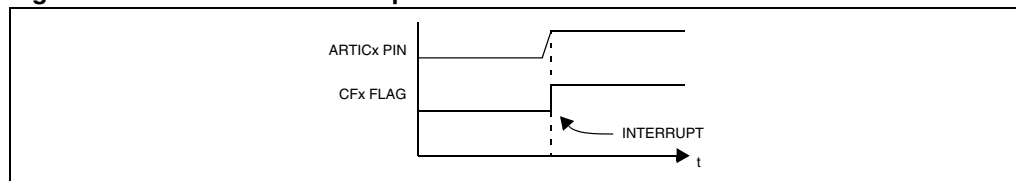
11.2.9 External interrupt capability

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). In this case, the interrupt synchronization is done directly on the ARTICx pin edge (Figure 46).

Figure 46. ART external interrupt in halt mode



11.3 Register description

Control/status register (ARTCSR)

Read/Write

Reset value: 0000 0000 (00h)

| | |
|------|------|
| 7 | 0 |
| EXCL | OVF |
| CC2 | OIE |
| CC1 | FCRL |
| CC0 | TCE |

Bit 7 = **EXCL** *External Clock*

This bit is set and cleared by software. It selects the input clock for the 7-bit prescaler.

- 0: CPU clock.
- 1: External clock.

Bit 6:4 = **CC[2:0]** *Counter Clock Control*

These bits are set and cleared by software. They determine the prescaler division ratio from f_{INPUT} .

Table 43. Counter clock control

| f_{COUNTER} | With $f_{\text{INPUT}}=8 \text{ MHz}$ | CC2 | CC1 | CC0 |
|--------------------------|---------------------------------------|-----|-----|-----|
| f_{INPUT} | 8 MHz | 0 | 0 | 0 |
| $f_{\text{INPUT}} / 2$ | 4 MHz | 0 | 0 | 1 |
| $f_{\text{INPUT}} / 4$ | 2 MHz | 0 | 1 | 0 |
| $f_{\text{INPUT}} / 8$ | 1 MHz | 0 | 1 | 1 |
| $f_{\text{INPUT}} / 16$ | 500 kHz | 1 | 0 | 0 |
| $f_{\text{INPUT}} / 32$ | 250 kHz | 1 | 0 | 1 |
| $f_{\text{INPUT}} / 64$ | 125 kHz | 1 | 1 | 0 |
| $f_{\text{INPUT}} / 128$ | 62.5 kHz | 1 | 1 | 1 |

Bit 3 = **TCE** *Timer Counter Enable*

This bit is set and cleared by software. It puts the timer in the lowest power consumption mode.

- 0: Counter stopped (prescaler and counter frozen).
- 1: Counter running.

Bit 2 = **FCRL** *Force Counter Re-Load*

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.

Bit 1 = **OIE** *Overflow Interrupt Enable*

This bit is set and cleared by software. It allows to enable/disable the interrupt which is generated when the OVF bit is set.

- 0: Overflow Interrupt disable.
- 1: Overflow Interrupt enable.

Bit 0 = **OVF** *Overflow Flag*

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.

- 0: New transition not yet reached
- 1: Transition reached

Counter Access Register (ARTCAR)

Read/Write

Reset value: 0000 0000 (00h)

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| | | | | | | | |
| 7 | | | | | | | 0 |
| CA7 | CA6 | CA5 | CA4 | CA3 | CA2 | CA1 | CA0 |

Bit 7:0 = **CA[7:0]** *Counter Access Data*

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).

Auto-Reload Register (ARTARR)

Read/Write

Reset value: 0000 0000 (00h)

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| | | | | | | | |
| 7 | | | | | | | 0 |
| AR7 | AR6 | AR5 | AR4 | AR3 | AR2 | AR1 | AR0 |

Bit 7:0 = **AR[7:0]** *Counter Auto-Reload Data*

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

Table 44. PWM frequency vs 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 |



PWM control register (PWMCR)

Read/write

Reset value: 0000 0000 (00h)

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 7 | 0 | | | | | | |
| OE3 | OE2 | OE1 | OE0 | OP3 | OP2 | OP1 | OP0 |

Bit 7:4 = **OE[3:0]** *PWM Output Enable*

These bits are set and cleared by software. They enable or disable the PWM output channels independently acting on the corresponding I/O pin.

0: PWM output disabled.

1: PWM output enabled.

Bit 3:0 = **OP[3:0]** *PWM Output Polarity*

These bits are set and cleared by software. They independently select the polarity of the four PWM output signals.

Table 45. PWMx output level and polarity

| PWMx output level | | OPx |
|-------------------|----------------|-----|
| Counter ≤ OCRx | Counter > OCRx | |
| 1 | 0 | 0 |
| 0 | 1 | 1 |

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

Duty cycle registers (PWMDCRx)

Read/write

Reset value: 0000 0000 (00h)

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 7 | 0 | | | | | | |
| DC7 | DC6 | DC5 | DC4 | DC3 | DC2 | DC1 | DC0 |

Bit 7:0 = **DC[7:0]** *Duty Cycle Data*

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.

Input Capture control / status register (ARTICCSR)

Read/Write

Reset value: 0000 0000 (00h)

| | | | | | | | | |
|---|---|-----|-----|------|------|-----|-----|---|
| 7 | | | | | | | | 0 |
| 0 | 0 | CS2 | CS1 | CIE2 | CIE1 | CF2 | CF1 | |

Bit 7:6 = Reserved, always read as 0.

Bit 5:4 = **CS[2:1] Capture Sensitivity**

These bits are set and cleared by software. They determine the trigger event polarity on the corresponding input capture channel.

- 0: Falling edge triggers capture on channel x.
- 1: Rising edge triggers capture on channel x.

Bit 3:2 = **CIE[2:1] Capture Interrupt Enable**

These bits are set and cleared by software. They enable or disable the Input capture channel interrupts independently.

- 0: Input capture channel x interrupt disabled.
- 1: Input capture channel x interrupt enabled.

Bit 1:0 = **CF[2:1] Capture Flag**

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.

- 0: No input capture on channel x.
- 1: An input capture has occurred on channel x.

Input Capture Registers (ARTICRx)

Read only

Reset value: 0000 0000 (00h)

| | | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|---|
| 7 | | | | | | | | 0 |
| IC7 | IC6 | IC5 | IC4 | IC3 | IC2 | IC1 | IC0 | |

Bit 7:0 = **IC[7:0] Input Capture Data**

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 46. PWM auto-reload timer register map and reset values

| Address (Hex.) | Register label | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------------|------------------------|----------|----------|----------|----------|----------|----------|----------|----------|
| 0031h | PWMDCR3 Reset value | DC7 0 | DC6 0 | DC5 0 | DC4 0 | DC3 0 | DC2 0 | DC1 0 | DC0 0 |
| 0032h | PWMDCR2 Reset value | DC7 0 | DC6 0 | DC5 0 | DC4 0 | DC3 0 | DC2 0 | DC1 0 | DC0 0 |
| 0033h | PWMDCR1 Reset value | DC7 0 | DC6 0 | DC5 0 | DC4 0 | DC3 0 | DC2 0 | DC1 0 | DC0 0 |
| 0034h | PWMDCR0 Reset value | DC7 0 | DC6 0 | DC5 0 | DC4 0 | DC3 0 | DC2 0 | DC1 0 | DC0 0 |

Table 46. PWM auto-reload timer register map and reset values (continued)

| Address (Hex.) | Register label | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------------|-------------------------|-----------|----------|----------|----------|----------|-----------|----------|----------|
| 0035h | PWMCR Reset value | OE3 0 | OE2 0 | OE1 0 | OE0 0 | OP3 0 | OP2 0 | OP1 0 | OP0 0 |
| 0036h | ARTCSR Reset value | EXCL 0 | CC2 0 | CC1 0 | CC0 0 | TCE 0 | FCRL 0 | RIE 0 | OVF 0 |
| 0037h | ARTCAR Reset value | CA7 0 | CA6 0 | CA5 0 | CA4 0 | CA3 0 | CA2 0 | CA1 0 | CA0 0 |
| 0038h | ARTARR Reset value | AR7 0 | AR6 0 | AR5 0 | AR4 0 | AR3 0 | AR2 0 | AR1 0 | AR0 0 |
| 0039h | ARTICCSR Reset value | 0 | 0 | CE2 0 | CE1 0 | CS2 0 | CS1 0 | CF2 0 | CF1 0 |
| 003Ah | ARTICR1 Reset value | IC7 0 | IC6 0 | IC5 0 | IC4 0 | IC3 0 | IC2 0 | IC1 0 | IC0 0 |
| 003Bh | ARTICR2 Reset value | IC7 0 | IC6 0 | IC5 0 | IC4 0 | IC3 0 | IC2 0 | IC1 0 | IC0 0 |

12 16-bit timer

12.1 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 a 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).

12.2 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 47](#).

a. Some timer pins may not be available (not bonded) in some ST7 devices. Refer to the device pin out description.
When reading an input signal on a non-bonded pin, the value will always be '1'.

12.3 Functional description

12.3.1 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 titled 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 50](#). 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.

Figure 47. Timer block diagram

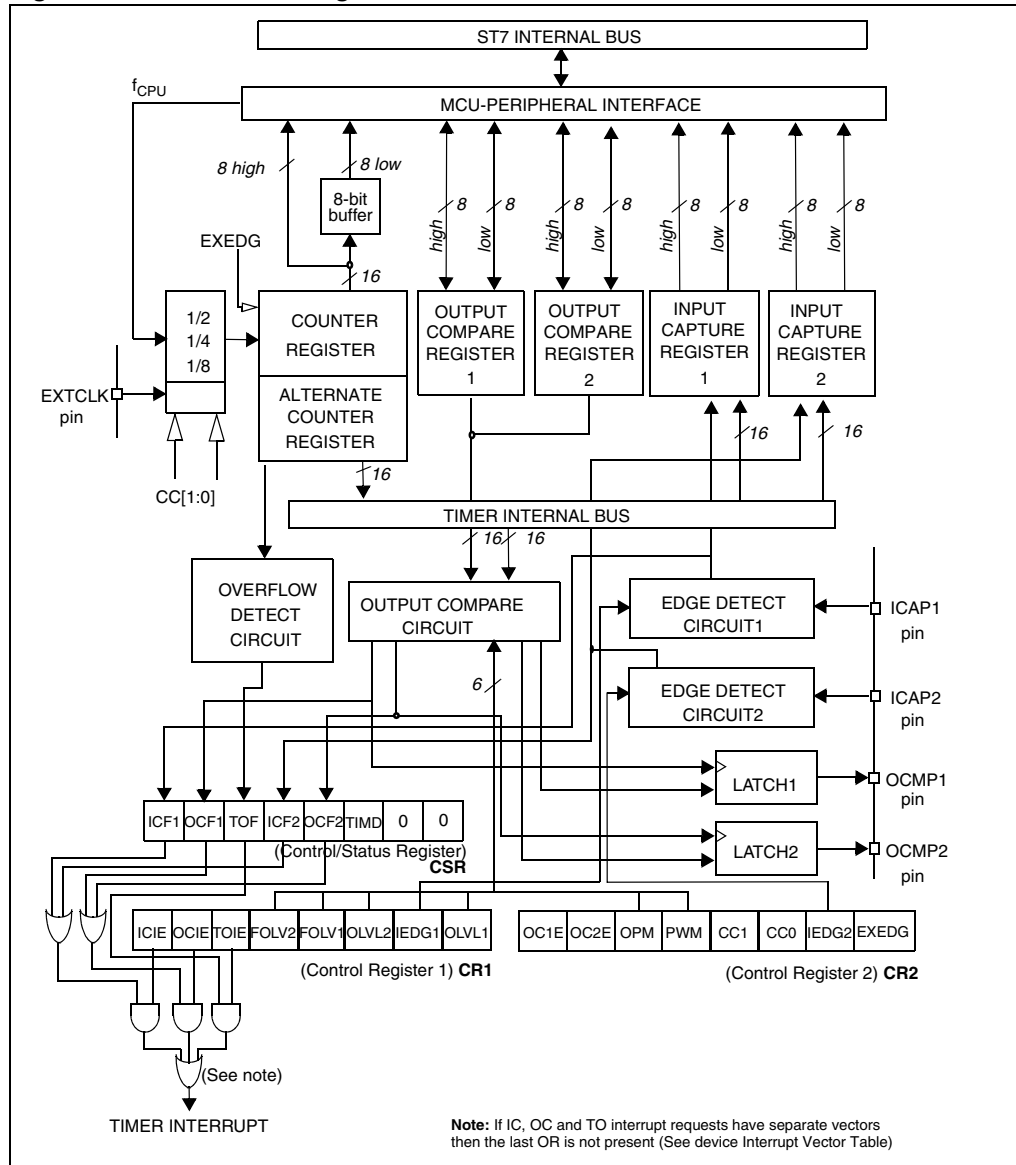
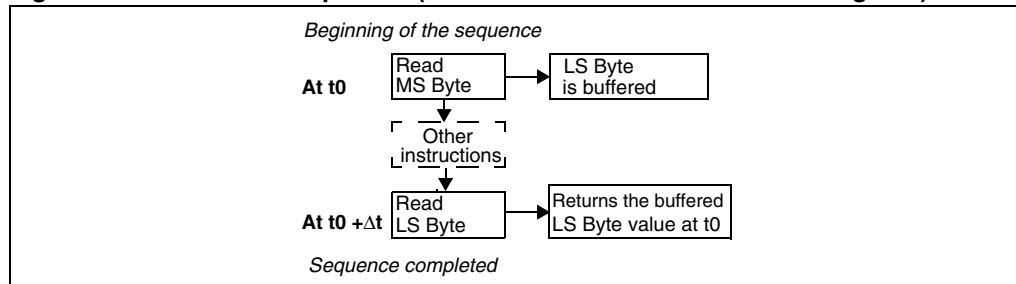


Figure 48. 16-bit read sequence: (from counter or alternate counter register)



The user must read the MS Byte first, then the LS Byte value is 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 the timer mode used (input capture, output compare, One Pulse mode or PWM mode) an overflow occurs when the counter rolls over from FFFFh to 0000h then:

- The TOF bit of the SR register is set.
- A timer interrupt is generated if:
 - TOIE bit of the CR1 register is set and
 - 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).

12.3.2 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 49. Counter timing diagram, internal clock divided by 2

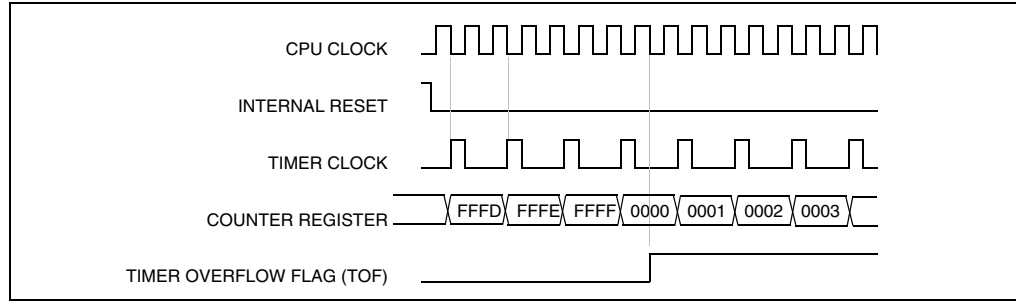


Figure 50. Counter timing diagram, internal clock divided by 4

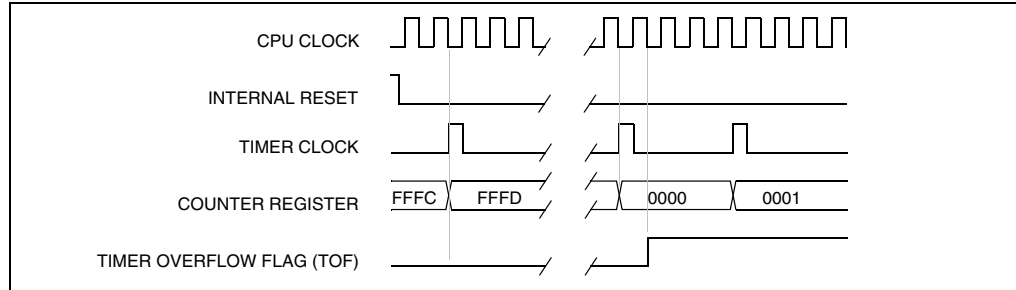
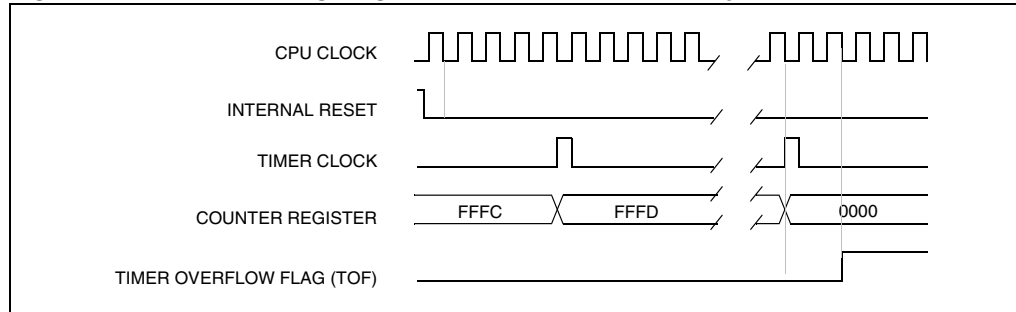


Figure 51. Counter timing diagram, internal clock divided by 8

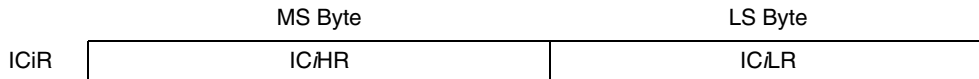


Note: The MCU is in reset state when the internal reset signal is high, when it is low the MCU is running.

12.3.3 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 5](#)).



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

The active transition is software programmable through the IEDG i bit in the control register (CR i).

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

12.3.4 Procedure

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

- Select the timer clock (CC[1:0]) (see [Table 50](#)).
- 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 ICAP1 pin 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 53](#)).
- 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 IC i HR register, transfer of input capture data is inhibited and ICF i will never be set until the IC i LR register is also read.
 - 2 The IC i R 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.
 - 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. Moreover if one of the ICAP i 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. This can be avoided if the input capture function i is disabled by reading the IC i HR (see note 1).
 - 6 The TOF bit can be used with interrupt generation in order to measure events that go beyond the timer range (FFFFh).

Figure 52. Input capture block diagram

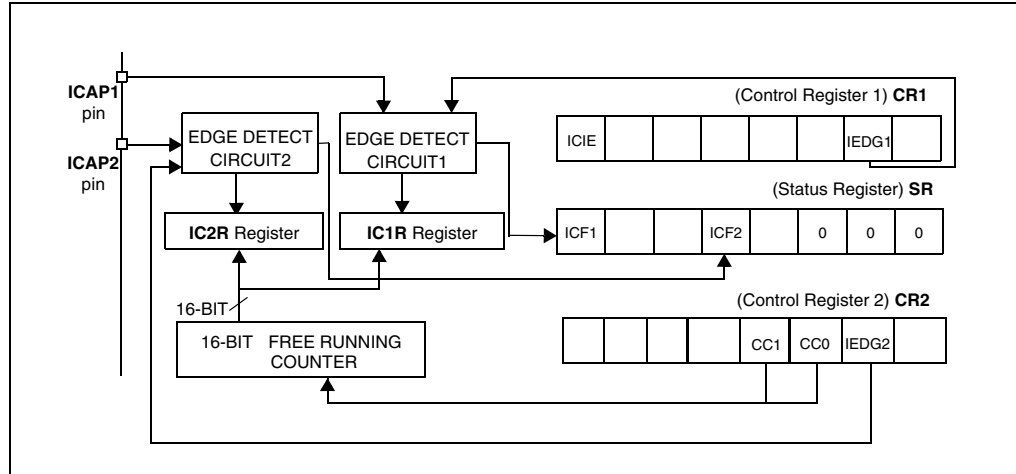
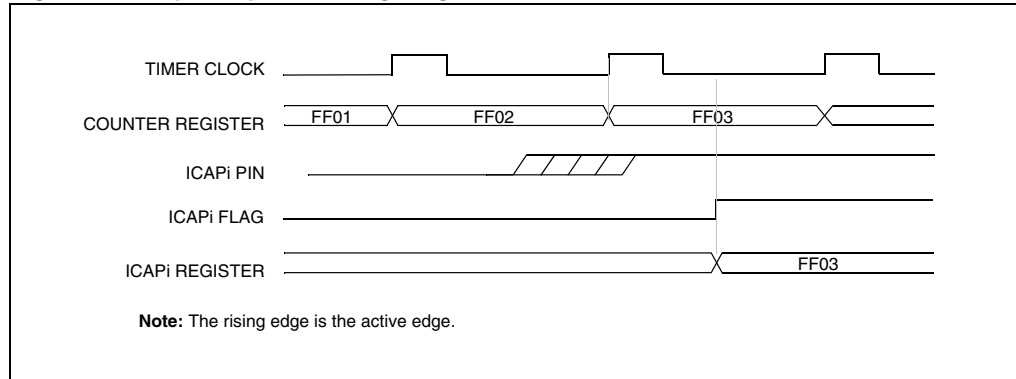


Figure 53. Input capture timing diagram



12.3.5 Output compare

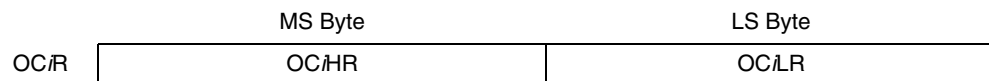
In this section, the index, *i*, may be 1 or 2 because there are two output compare functions in the 16-bit timer.

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/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.



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

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

12.3.6 Procedure

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

- Set the OC_iE 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 50](#)).

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

$$\Delta \text{OCR} = \frac{\Delta t \cdot f_{\text{CPU}}}{\text{PRESC}}$$

Where:

Δt = 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 50](#))

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

$$\Delta \text{OCR} = \Delta t \cdot f_{\text{EXT}}$$

Where:

Δt = output compare period (in seconds)

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

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

- Reading the SR register while the OCF_i bit is set.
- An access (read or write) to the OCLR 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 $OCiR$ register:

- Write to the $OCiHR$ 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 $OCiLR$ 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 $OCMP_i$ pin is a general I/O port and the OLV_i bit will not appear when a match is found but an interrupt could be generated if the $OCiE$ bit is set.
 - 3 When the timer clock is $f_{CPU}/2$, OCF_i and $OCMP_i$ are set while the counter value equals the $OCiR$ register value (see Figure 55). This behavior is the same in OPM or PWM mode. When the timer clock is $f_{CPU}/4$, $f_{CPU}/8$ or in external clock mode, OCF_i and $OCMP_i$ are set while the counter value equals the $OCiR$ register value (see Figure 56).
 - 4 The output compare functions can be used both for generating external events on the $OCMP_i$ pins even if the input capture mode is also used.
 - 5 The value in the 16-bit $OCiR$ register and the OLV_i bit should be changed after each successful comparison in order to control an output waveform or establish a new elapsed timeout.

12.3.7 Forced compare output capability

When the $FOLV_i$ bit is set by software, the OLV_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 ($OCiE$ bit = 1). The OCF_i bit is then not set by hardware, and thus no interrupt request is generated.

The $FOLV_i$ bits have no effect in both One Pulse mode and PWM mode.

Figure 54. Output compare block diagram

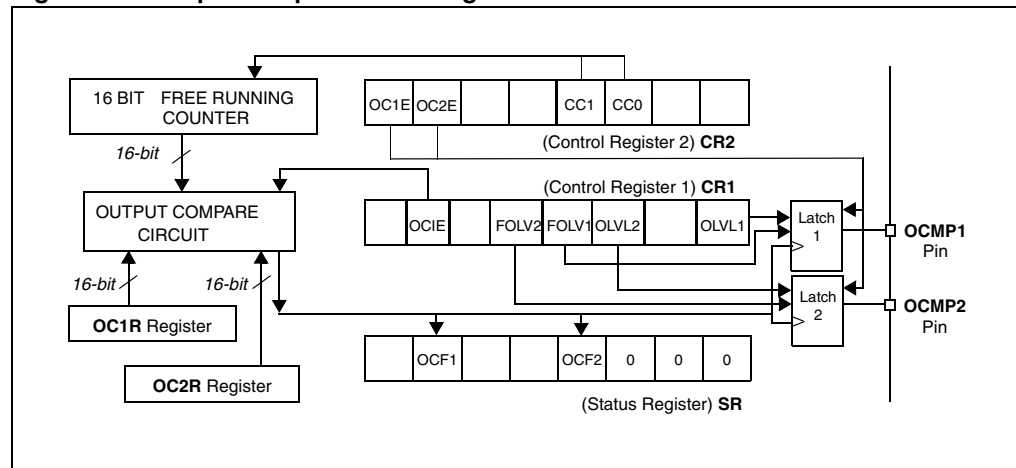
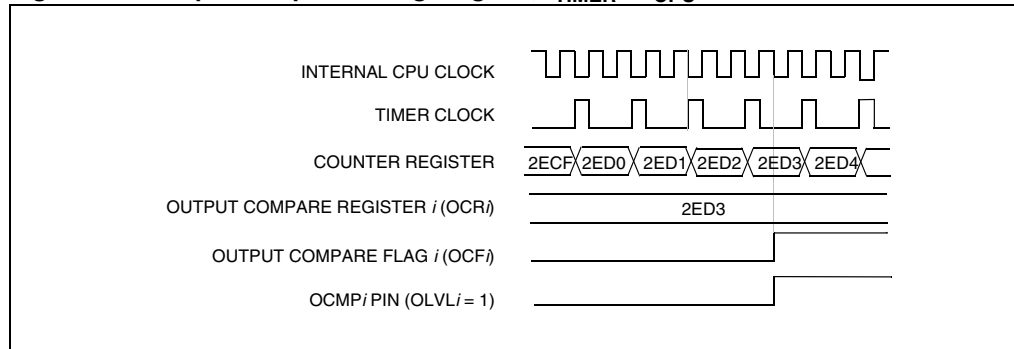
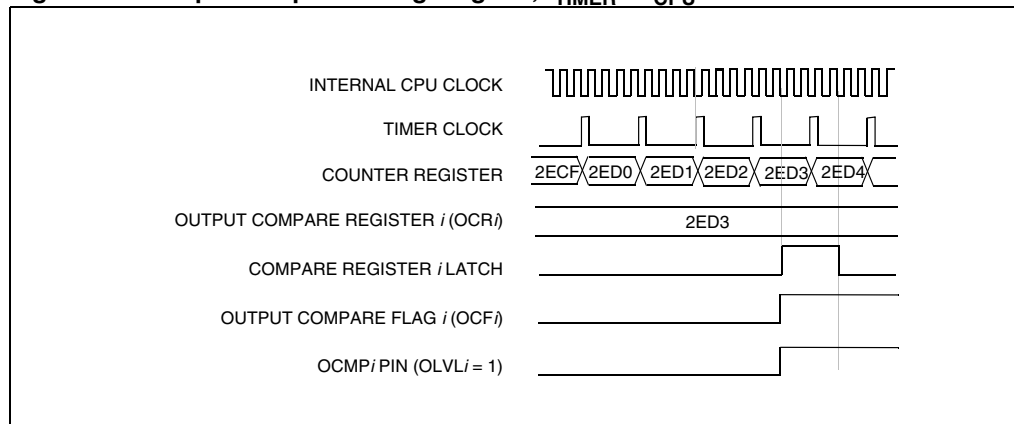


Figure 55. Output compare timing diagram, $f_{\text{TIMER}} = f_{\text{CPU}}/2$ Figure 56. Output compare timing diagram, $f_{\text{TIMER}} = f_{\text{CPU}}/4$ 

12.3.8 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.

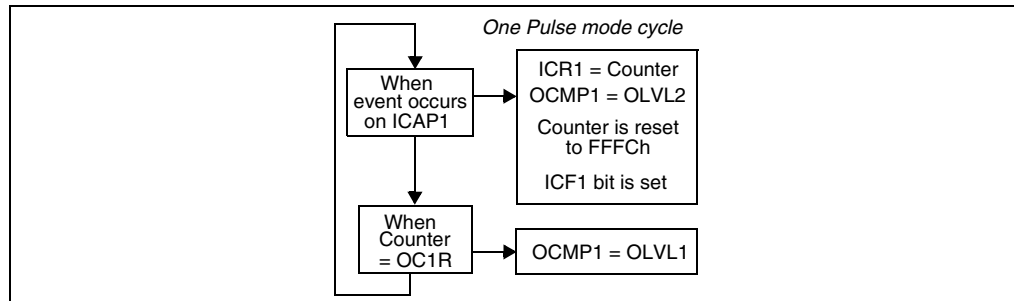
The One Pulse mode uses the Input Capture1 function and the Output Compare1 function.

Procedure

To use One Pulse mode:

1. Load the OC1R register with the value corresponding to the length of the pulse (see the formula in the opposite column).
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 50](#)).



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.

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.

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

$$\text{OC1R Value} = \frac{t \cdot f_{\text{CPU}}}{\text{PRESC}} - 5$$

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 50](#))

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

$$\text{OC1R} = t \cdot f_{\text{EXT}} - 5$$

Where:

t = Pulse period (in seconds)

f_{EXT} = External timer 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 57](#)).

- 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 can not 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 57. One pulse mode timing example

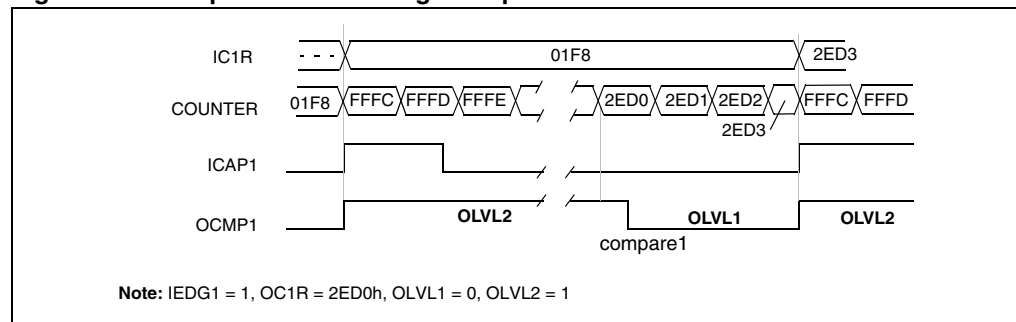
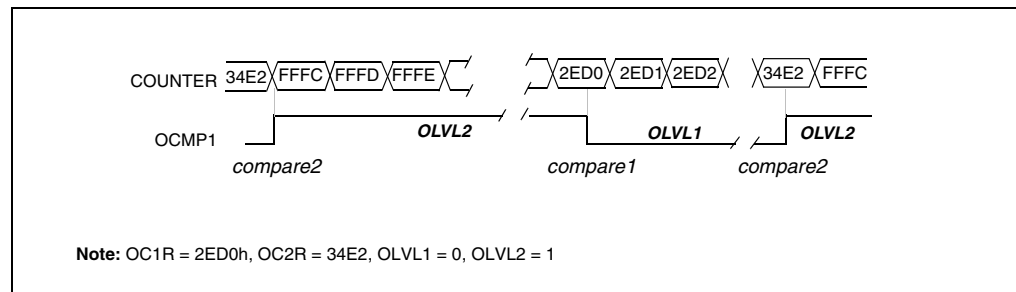


Figure 58. Pulse width modulation mode timing example with 2 output compare functions



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.

12.3.9 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.

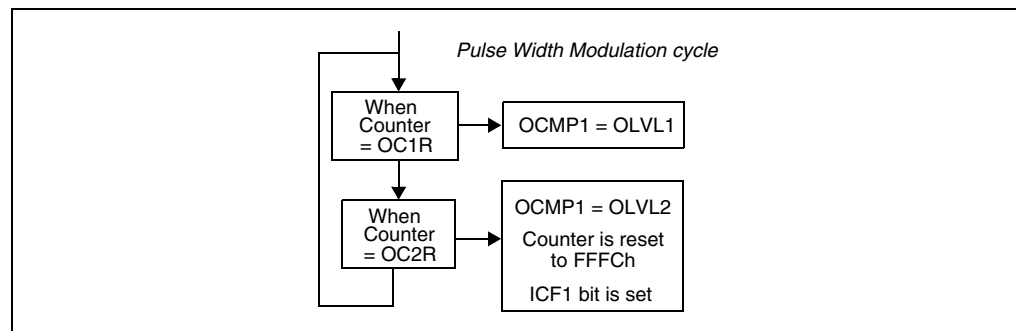
Pulse Width Modulation mode uses the complete Output Compare 1 function plus the OC2R register, and so this functionality can not be used when PWM mode is activated.

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 formula in the opposite column.
2. Load the OC1R register with the value corresponding to the period of the pulse if (OLVL1 = 0 and OLVL2 = 1) using the formula in the opposite column.
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 50](#)).



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

$$\text{OC/R Value} = \frac{t \cdot f_{\text{CPU}}}{\text{PRESC}} - 5$$

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 50](#))

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

$$\text{OC/R} = t \cdot f_{\text{EXT}} - 5$$

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 58](#))

- 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 can not 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 generate 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.

12.4 Low power modes

Table 47. Effect of low power modes on 16-bit timer

| Mode | Description |
|------|--|
| WAIT | No effect on 16-bit Timer. Timer interrupts cause the device to exit from WAIT mode. |
| HALT | 16-bit Timer registers are frozen. 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. If an input capture event occurs on the ICAP <i>i</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>i</i> bit is set, and the counter value present when exiting from HALT mode is captured into the IC <i>i</i> R register. |

12.5 Interrupts

Table 48. Timer interrupt control and 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 *Interrupts chapter*). 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).

12.6 Summary of timer modes

Table 49. 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) | | | | |
| One pulse mode | No | Not recommended ⁽¹⁾ | No | Partially ⁽²⁾ |
| PWM Mode | | Not recommended ⁽³⁾ | | No |

1. See note 4 in *One pulse mode*

2. See note 5 in *One pulse mode*

3. See note 4 in *Pulse width modulation mode*

12.7 Register description

Each Timer is associated with three control and status registers, and with six pairs of data registers (16-bit values) relating to the two input captures, the two output compares, the counter and the alternate counter.

12.7.1 Control register 1 (CR1)

Read/ write

Reset value: 0000 0000 (00h)

| | | | | | | | |
|------|------|------|-------|-------|-------|-------|-------|
| 7 | | | | | | | 0 |
| ICIE | OCIE | TOIE | FOLV2 | FOLV1 | OLVL2 | IEDG1 | OLVL1 |

Bit 7 = **ICIE** *Input Capture Interrupt Enable*.

0: Interrupt is inhibited.

1: A timer interrupt is generated whenever the ICF1 or ICF2 bit of the SR register is set.

Bit 6 = **OCIE** *Output Compare Interrupt Enable*.

0: Interrupt is inhibited.

1: A timer interrupt is generated whenever the OCF1 or OCF2 bit of the SR register is set.

Bit 5 = **TOIE** *Timer Overflow Interrupt Enable*.

0: Interrupt is inhibited.

1: A timer interrupt is enabled whenever the TOF bit of the SR register is set.

Bit 4 = **FOLV2** *Forced Output Compare 2*.

This bit is set and cleared by software.

0: No effect on the OCMP2 pin.

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.

Bit 3 = **FOLV1** *Forced Output Compare 1*.

This bit is set and cleared by software.

0: No effect on the OCMP1 pin.

1: Forces OLVL1 to be copied to the OCMP1 pin, if the OC1E bit is set and even if there is no successful comparison.

Bit 2 = **OLVL2** *Output Level 2*.

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.

Bit 1 = **IEDG1** *Input Edge 1*.

This bit determines which type of level transition on the ICAP1 pin will trigger the capture.

0: A falling edge triggers the capture.

1: A rising edge triggers the capture.

Bit 0 = **OLVL1** *Output Level 1*.

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.

12.7.2 Control register 2 (CR2)

Read/ write

Reset value: 0000 0000 (00h)

| | | | | | | | |
|------|------|-----|-----|-----|-----|-------|-------|
| 7 | 0 | | | | | | |
| OC1E | OC2E | OPM | PWM | CC1 | CC0 | IEDG2 | EXEDG |

Bit 7 = **OC1E** *Output Compare 1 Pin Enable*.

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.

0: OCMP1 pin alternate function disabled (I/O pin free for general-purpose I/O).

1: OCMP1 pin alternate function enabled.

Bit 6 = **OC2E** *Output Compare 2 Pin Enable*.

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.

0: OCMP2 pin alternate function disabled (I/O pin free for general-purpose I/O).

1: OCMP2 pin alternate function enabled.

Bit 5 = **OPM** *One Pulse Mode*.

0: One Pulse mode is not active.

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.

Bit 4 = **PWM** *Pulse Width Modulation*.

0: PWM mode is not active.

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.

Bit 3, 2 = **CC[1:0]** *Clock Control*.

The timer clock mode depends on these bits:

Table 50. Clock control bits

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

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

Bit 1 = **IEDG2** *Input Edge 2*.

This bit determines which type of level transition on the ICAP2 pin will trigger the capture.

0: A falling edge triggers the capture.

1: A rising edge triggers the capture.

Bit 0 = **EXEDG** *External Clock Edge*.

This bit determines which type of level transition on the external clock pin EXTCLK will trigger the counter register.

0: a falling edge triggers the counter register.

1: a rising edge triggers the counter register.

12.7.3 Control/status register (CSR)

Read/ write (bits 7:3 read only)

Reset value: xxxx x0xx (xxh)

| | | | | | | | |
|------|------|-----|------|------|------|---|---|
| 7 | | | | | | | 0 |
| ICF1 | OCF1 | TOF | ICF2 | OCF2 | TIMD | 0 | 0 |

Bit 7 = **ICF1** *Input Capture Flag 1*.

0: no input capture (reset value).

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.

Bit 6 = **OCF1** *Output Compare Flag 1*.

0: nomatch (reset value).

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.

Bit 5 = **TOF** *Timer Overflow Flag*.

0: No timer overflow (reset value).

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.

Note: Reading or writing the ACLR register does not clear TOF.

Bit 4 = **ICF2** *Input Capture Flag 2*.

0: No input capture (reset value).

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.

Bit 3 = **OCF2** *Output Compare Flag 2*.

0: No match (reset value).

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.

Bit 2 = **TIMD** *Timer disable*.

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.

0: Timer enabled

1: Timer prescaler, counter and outputs disabled

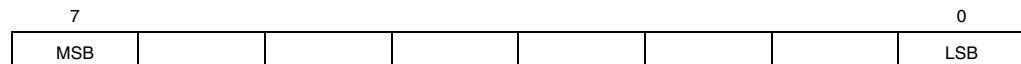
Bits 1:0 = Reserved, must be kept cleared.

12.7.4 Input capture 1 high register (IC1HR)

Read Only

Reset value: Undefined

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

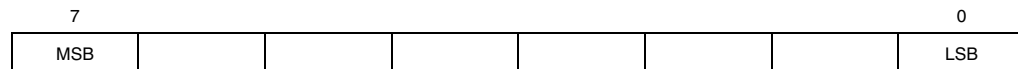


12.7.5 Input capture 1 low register (IC1LR)

Read only

Reset value: Undefined

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

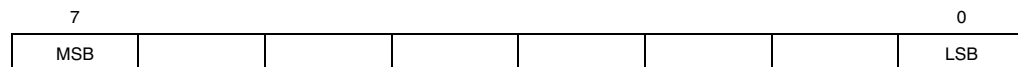


12.7.6 Output compare 1 high register (OC1HR)

Read/ write

Reset value: 1000 0000 (80h)

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

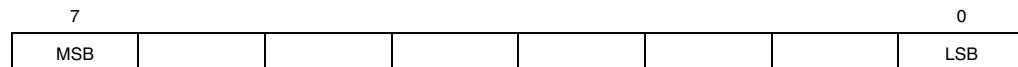


12.7.7 Output compare 1 low register (OC1LR)

Read/ write

Reset value: 0000 0000 (00h)

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

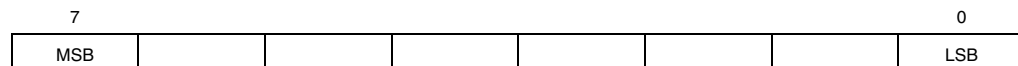


12.7.8 Output compare 2 high register (OC2HR)

Read/ write

Reset value: 1000 0000 (80h)

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

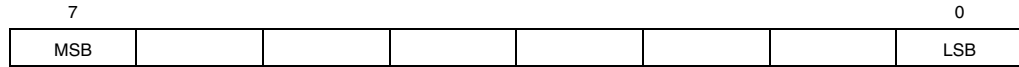


12.7.9 Output compare 2 low register (OC2LR)

Read/ write

Reset value: 0000 0000 (00h)

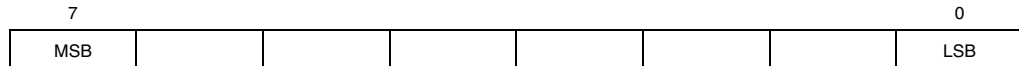
This is an 8-bit register that contains the low part of the value to be compared to the CLR register.

**12.7.10 Counter high register (CHR)**

Read only

Reset value: 1111 1111 (FFh)

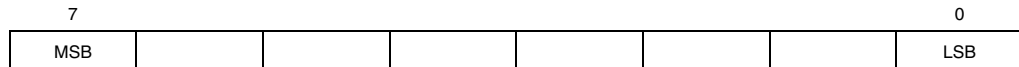
This is an 8-bit register that contains the high part of the counter value.

**12.7.11 Counter low register (CLR)**

Read only

Reset value: 1111 1100 (FCh)

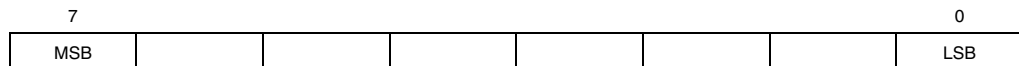
This is an 8-bit register that contains the low part of the counter value. A write to this register resets the counter. An access to this register after accessing the CSR register clears the TOF bit.

**12.7.12 Alternate counter high register (ACHR)**

Read Only

Reset value: 1111 1111 (FFh)

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

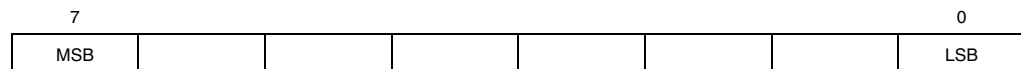


12.7.13 Alternate counter low register (ACLR)

Read only

Reset value: 1111 1100 (FCh)

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.

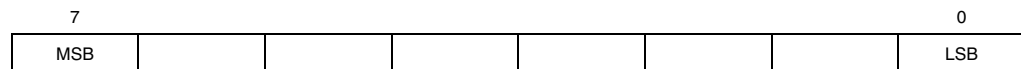


12.7.14 Input capture 2 high register (IC2HR)

Read only

Reset value: Undefined

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



12.7.15 Input capture 2 low register (IC2LR)

Read only

Reset value: Undefined

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

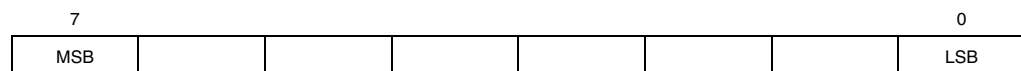


Table 51. 16-bit timer register map

| Address (Hex.) | Register name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------------|---------------|------|------|------|-------|-------|-------|-------|-------|
| 51 | CR2 | OC1E | OC2E | OPM | PWM | CC1 | CC0 | IEDG2 | EXEDG |
| 52 | CR1 | ICIE | OCIE | TOIE | FOLV2 | FOLV1 | OLVL2 | IEDG1 | OLVL1 |
| 53 | CSR | ICF1 | OCF1 | TOF | ICF2 | OCF2 | TIMD | | |

Table 51. 16-bit timer register map (continued)

| Address (Hex.) | Register name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------------|---------------|-----|---|---|---|---|---|---|-----|
| 54 | IC1HR | MSB | | | | | | | LSB |
| 55 | IC1LR | | | | | | | | |
| 56 | OC1HR | | | | | | | | |
| 57 | OC1LR | | | | | | | | |
| 58 | CHR | | | | | | | | |
| 59 | CLR | | | | | | | | |
| 5A | ACHR | | | | | | | | |
| 5B | ACLR | | | | | | | | |
| 5C | IC2HR | | | | | | | | |
| 5D | IC2LR | | | | | | | | |
| 5E | OC2HR | | | | | | | | |
| 5F | OC2LR | | | | | | | | |

13 8-bit timer (TIM8)

13.1 Introduction

The timer consists of a 8-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 clock prescaler.

13.2 Main features

- Programmable prescaler: f_{CPU} divided by 2, 4, 8 or f_{OSC2} divided by 8000.
- Overflow status flag and maskable interrupt
- Output compare functions with
 - 2 dedicated 8-bit registers
 - 2 dedicated programmable signals
 - 2 dedicated status flags
 - 1 dedicated maskable interrupt
- Input capture functions with
 - 2 dedicated 8-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
- 4 alternate functions on I/O ports (ICAP1, ICAP2, OCMP1, OCMP2)*

The Block Diagram is shown in [Figure 59](#).

Note: Some timer pins may not be available (not bonded) in some ST7 devices. Refer to the device pin out description.
When reading an input signal on a non-bonded pin, the value will always be '1'.

13.3 Functional description

13.3.1 Counter

The main block of the Programmable Timer is a 8-bit free running upcounter and its associated 8-bit registers.

These two read-only 8-bit registers contain the same value but with the difference that reading the ACTR register does not clear the TOF bit (Timer overflow flag), located in the Status register, (SR).

Writing in the CTR register or ACTR register resets the free running counter to the FCh value.

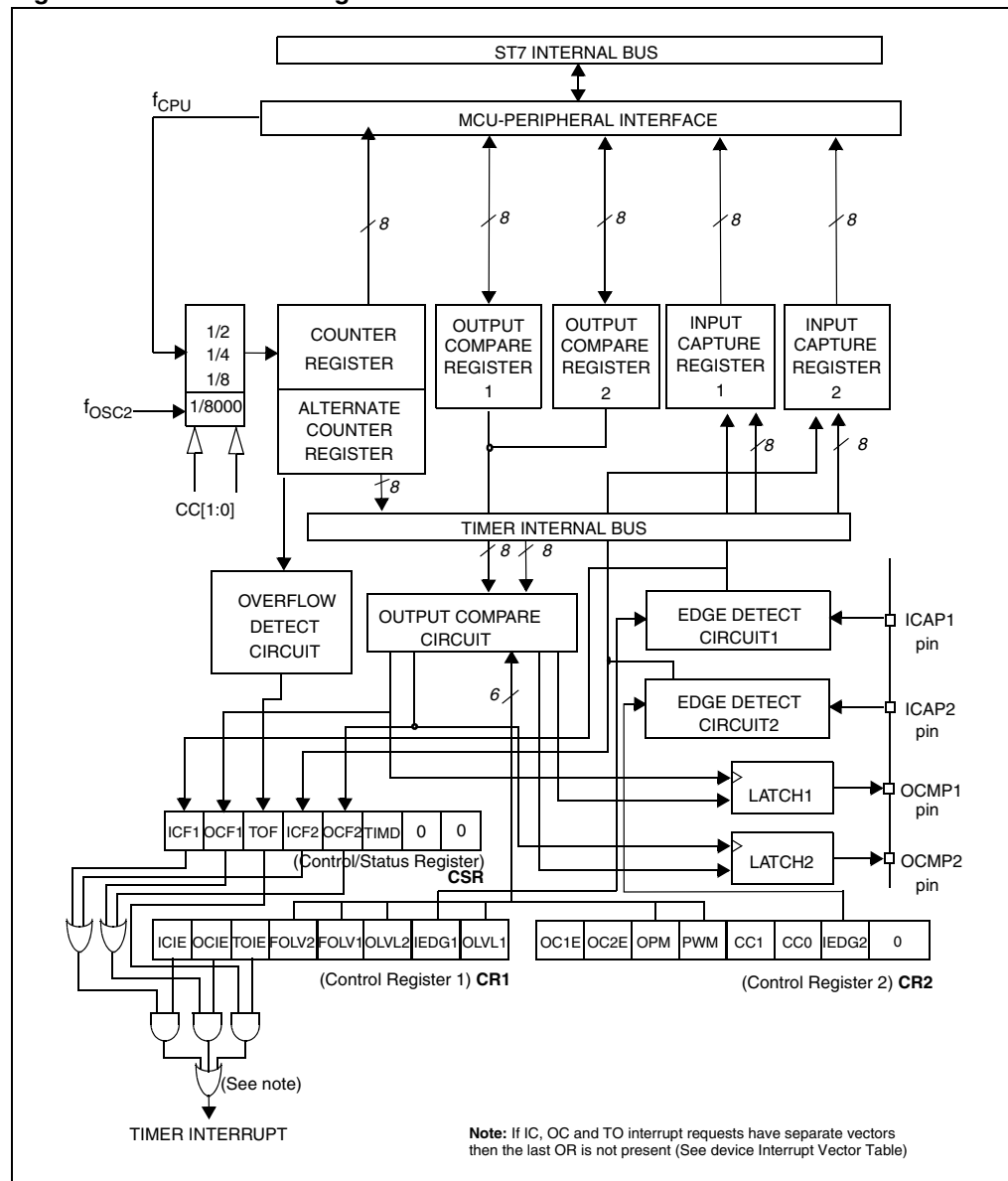
Both counters have a reset value of FCh (this is the only value which is reloaded in the 8-bit timer). The reset value of both counters is also FCh in One Pulse mode and PWM mode.

The timer clock depends on the clock control bits of the CR2 register, as shown in [Table 55](#). The value in the counter register repeats every 512, 1024, 2048 or 20480000 f_{CPU} clock cycles depending on the CC[1:0] bits.

The timer frequency can be $f_{\text{CPU}}/2$, $f_{\text{CPU}}/4$, $f_{\text{CPU}}/8$ or $f_{\text{OSC2}}/8000$.

For example, if $f_{\text{OSC2}}/8000$ is selected, and $f_{\text{OSC2}} = 8 \text{ MHz}$, the timer frequency will be 1 ms. Refer to [Table 55](#).

Figure 59. Timer block diagram



Whatever the timer mode used (input capture, output compare, one pulse mode or PWM mode) an overflow occurs when the counter rolls over from FFh to 00h then:

- The TOF bit of the SR register is set.
- A timer interrupt is generated if:
 - TOIE bit of the CR1 register is set and
 - 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 CTR register.

Note: *The TOF bit is not cleared by accesses to ACTR register. The advantage of accessing the ACTR register rather than the CTR 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).

Figure 60. Counter timing diagram, internal clock divided by 2

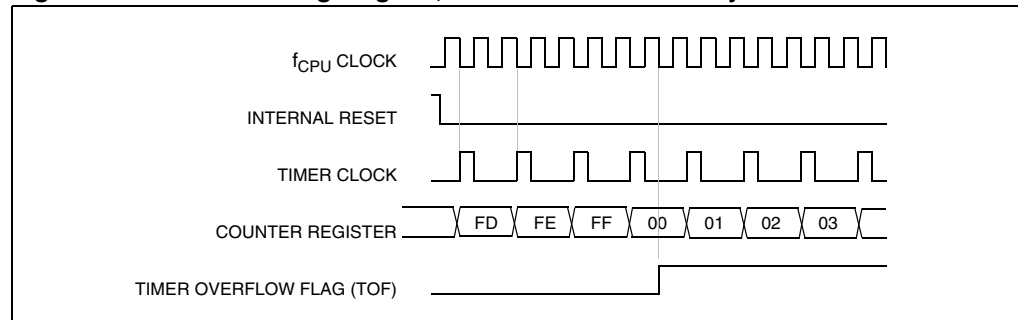


Figure 61. Counter timing diagram, internal clock divided by 4

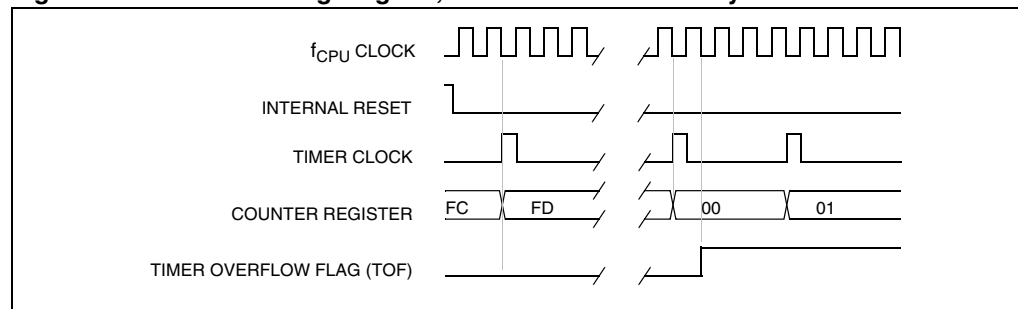
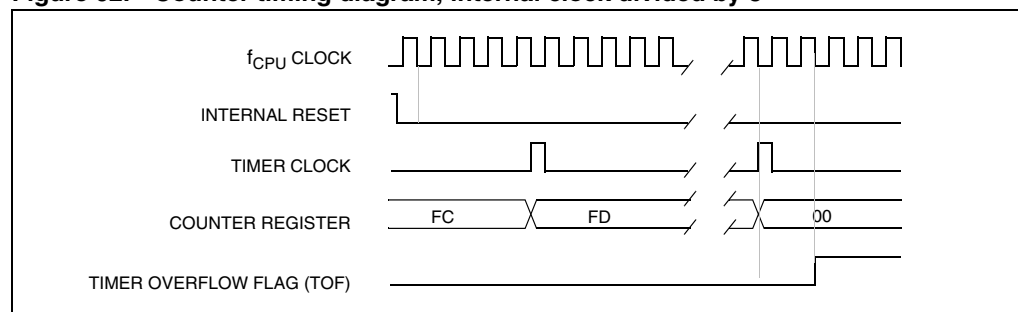


Figure 62. Counter timing diagram, internal clock divided by 8



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

13.3.2 Input capture

In this section, the index, i , may be 1 or 2 because there are two input capture functions in the 8-bit timer.

The two 8-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 63](#)).

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 (see [Table 55](#)).

Procedure

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

- Select the timer clock (CC[1:0]) (see [Table 55](#)).
- 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 ICAP1 pin 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 64](#)).
- A timer interrupt is generated if the ICIE bit is set and the interrupt mask 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 R register.

- Note:*
- 1 The IC i R register contains the free running counter value which corresponds to the most recent input capture.
 - 2 The two input capture functions can be used together even if the timer also uses the two output compare functions.
 - 3 Once the ICIE bit is set both input capture features may trigger interrupt requests. If only one is needed in the application, the interrupt routine software needs to discard the unwanted capture interrupt. This can be done by checking the ICF1 and ICF2 flags and resetting them both.
 - 4 In One pulse Mode and PWM mode only Input Capture 2 can be used.
 - 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.

Moreover if one of the ICAP_i 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.

- 6 The TOF bit can be used with interrupt generation in order to measure events that go beyond the timer range (FFh).

Figure 63. Input capture block diagram

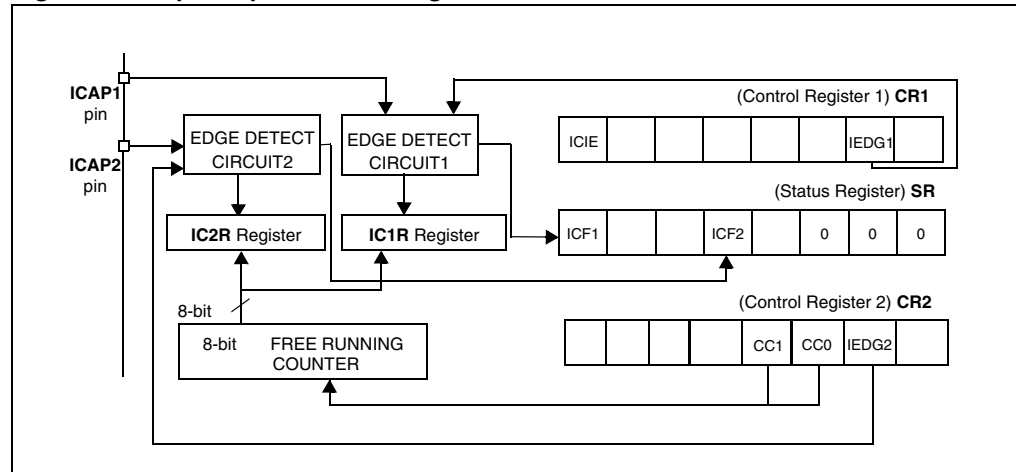
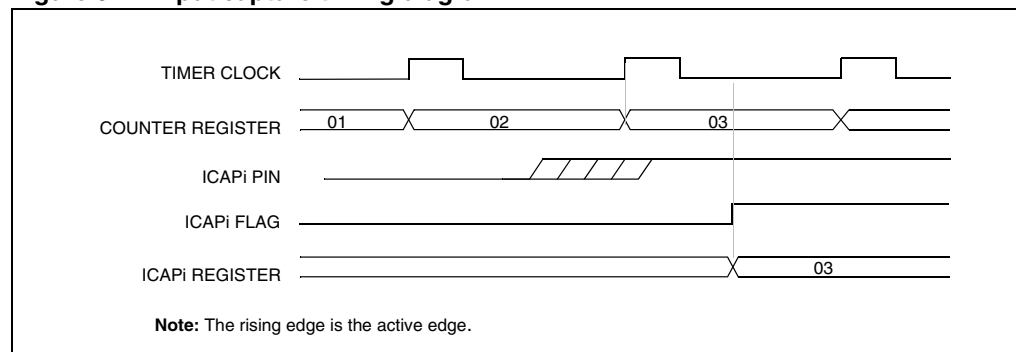


Figure 64. Input capture timing diagram



13.3.3 Output compare

In this section, the index, i , may be 1 or 2 because there are two output compare functions in the 8-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/E bit is set
- Sets a flag in the status register
- Generates an interrupt if enabled

Two 8-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.

These registers are readable and writable and are not affected by the timer hardware. A reset event changes the OC_iR value to 00h.

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

Procedure

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

- Set the OC_iE 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 55](#)).

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

$$\Delta \text{OC}_i\text{R} = \frac{\Delta t \cdot f_{\text{CPU}}}{\text{PRESC}}$$

Where:

Δt = Output compare period (in seconds)

f_{CPU} = PLL output x2 clock frequency in hertz (or $f_{\text{OSC}}/2$ if PLL is not enabled)

PRESC = Timer prescaler factor (2, 4, 8 or 8000 depending on CC[1:0] bits, see [Table 55](#))

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_iR register.

- Note:**
- 1 *Once the OCIE bit is set both output compare features may trigger interrupt requests. If only one is needed in the application, the interrupt routine software needs to discard the unwanted compare interrupt. This can be done by checking the OCF1 and OCF2 flags and resetting them both.*
 - 2 *If the OCIE bit is not set, the OCMP_i pin is a general I/O port and the OLVL_i bit will not appear when a match is found but an interrupt could be generated if the OCIE bit is set.*
 - 3 *When the timer clock is $f_{\text{CPU}}/2$, OCF_i and OCMP_i are set while the counter value equals the OC_iR register value (see [Figure 66](#)). This behavior is the same in OPM or PWM mode.*

When the timer clock is $f_{CPU}/4$, $f_{CPU}/8$ or $f_{CPU}/8000$, $OCFi$ and $OCMPi$ are set while the counter value equals the $OCiR$ register value plus 1 (see Figure 67).

- 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 8-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.

13.3.4 Forced compare output capability

When the $FOLVi$ bit is set by software, the $OLVLi$ bit is copied to the $OCMPi$ pin. The $OLVi$ bit has to be toggled in order to toggle the $OCMPi$ pin when it is enabled ($OCiE$ bit = 1). The $OCFi$ bit is then not set by hardware, and thus no interrupt request is generated.

The $FOLVLi$ bits have no effect in both one pulse mode and PWM mode.

Figure 65. Output compare block diagram

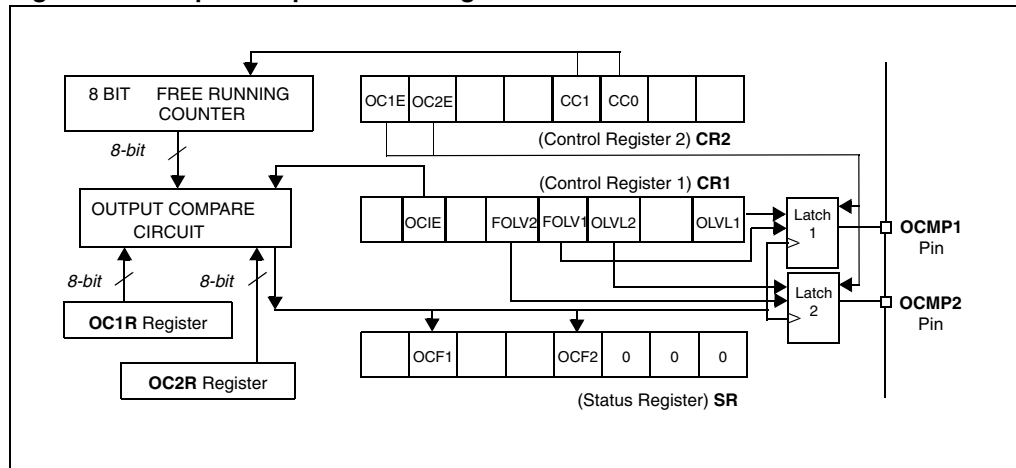


Figure 66. Output compare timing diagram, $f_{TIMER} = f_{CPU}/2$

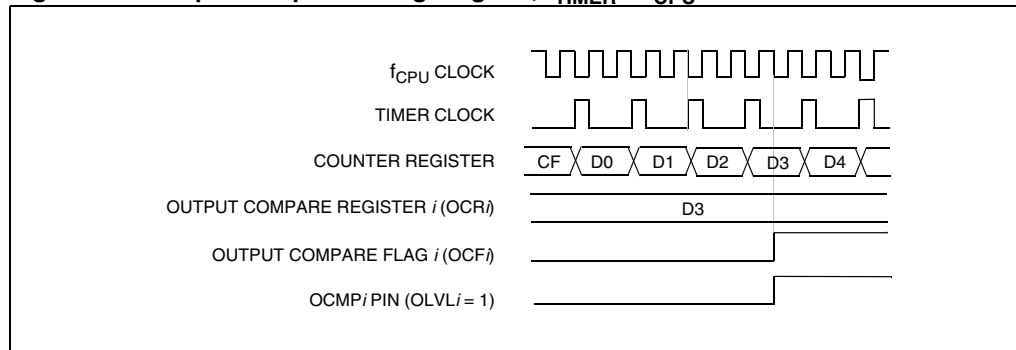
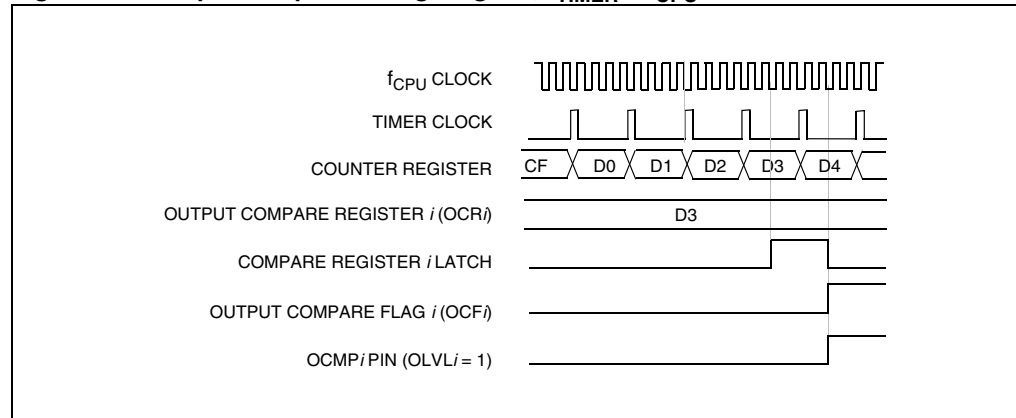


Figure 67. Output compare timing diagram, $f_{\text{TIMER}} = f_{\text{CPU}}/4$ 

13.3.5 One pulse mode

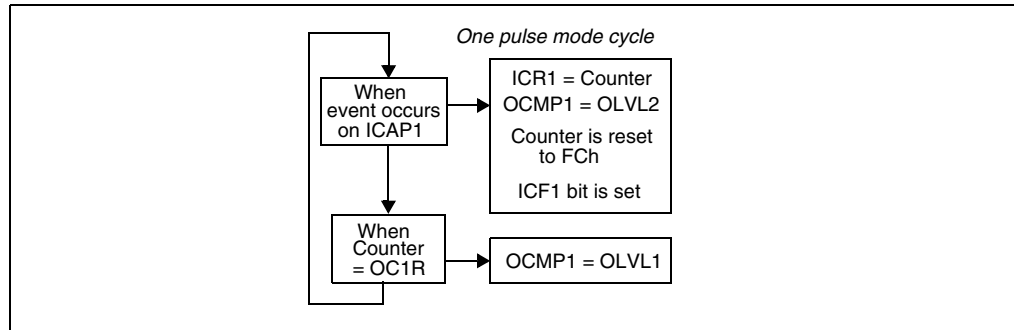
One Pulse mode enables the generation of a pulse when an external event occurs. This mode is selected via the OPM bit in the CR2 register.

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 (see the formula in the opposite column).
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 55](#)).



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

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.

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

$$\text{OC1R Value} = \frac{t \cdot f_{\text{CPU}}}{\text{PRESC}} - 5$$

Where:

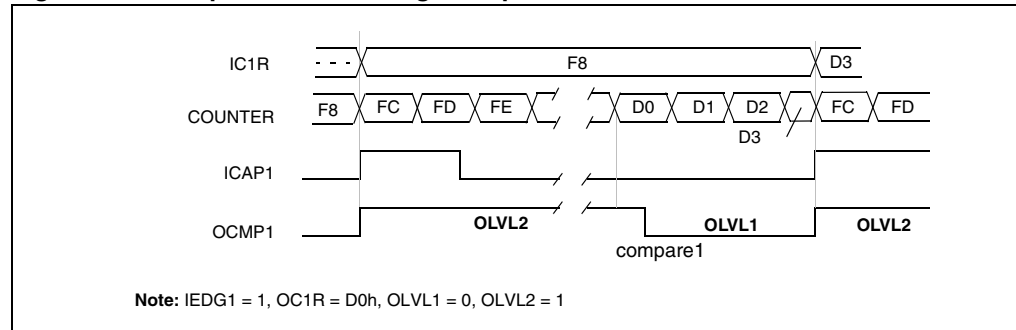
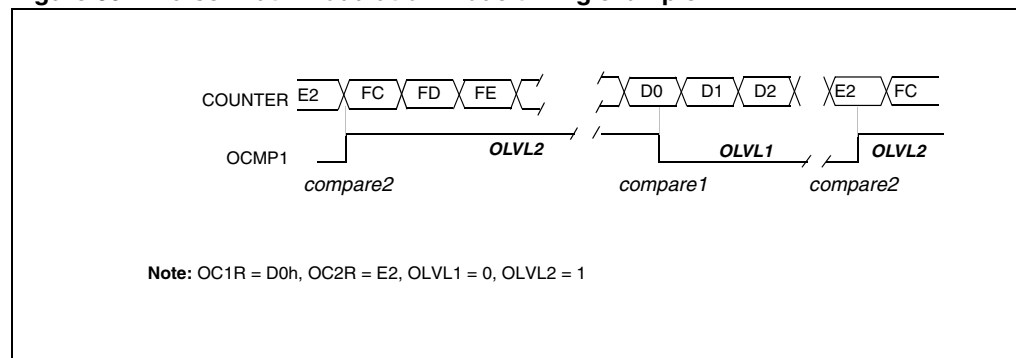
t = Pulse period (in seconds)

f_{CPU} = PLL output x2 clock frequency in hertz (or $f_{\text{OSC}}/2$ if PLL is not enabled)

PRESC = Timer prescaler factor (2, 4, 8 or 8000 depending on the CC[1:0] bits, see [Table 55](#))

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 68](#)).

- 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 can not 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 68. One pulse mode timing example**Figure 69. Pulse width modulation mode timing example**

13.3.6 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.

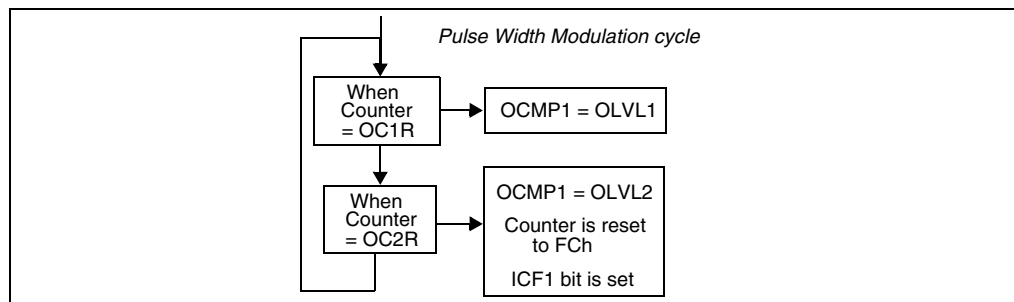
Pulse Width Modulation mode uses the complete Output Compare 1 function plus the OC2R register, and so this functionality can not be used when PWM mode is activated.

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 formula in the opposite column.
2. Load the OC1R register with the value corresponding to the period of the pulse if (OLVL1 = 0 and OLVL2 = 1) using the formula in the opposite column.
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 55](#)).



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:

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

Where:

- t = Signal or pulse period (in seconds)
- f_{CPU} = PLL output x2 clock frequency in hertz (or f_{OSC}/2 if PLL is not enabled)
- PRESC = Timer prescaler factor (2, 4, 8 or 8000 depending on CC[1:0] bits, see [Table 55](#))

The Output Compare 2 event causes the counter to be initialized to FCh (see [Figure 69](#))

- Note:**
- 1 The OCF1 and OCF2 bits cannot be set by hardware in PWM mode therefore the Output Compare interrupt is inhibited.
 - 2 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.
 - 3 In PWM mode the ICAP1 pin can not 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 generate interrupt if ICIE is set.

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

13.4 Low power modes

Table 52. Effect of low power modes on TIM8

| Mode | Description |
|------|---|
| WAIT | No effect on 8-bit Timer. Timer interrupts cause the device to exit from WAIT mode. |
| HALT | 8-bit Timer registers are frozen. 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. If an input capture event occurs on the ICAP <i>i</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>i</i> bit is set, and the counter value present when exiting from HALT mode is captured into the IC/R register. |

13.5 Interrupts

Table 53. TIM8 interrupt control and 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 8-bit Timer interrupt events are connected to the same interrupt vector (see *Interrupts chapter*). 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 54. Timer modes

| Modes | Available 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) | | | | |
| One Pulse Mode | No | Not Recommended ⁽¹⁾ | No | Partially ⁽²⁾ |
| PWM Mode | | Not Recommended ⁽³⁾ | | No |

1. See note 4 in *One pulse mode*.

2. See note 5 in *One pulse mode*.

3. See note 4 in *Pulse width modulation mode*.

13.7 Register description

Each Timer is associated with three control and status registers, and with six data registers (8-bit values) relating to the two input captures, the two output compares, the counter and the alternate counter.

13.7.1 Control register 1 (CR1)

Read/ write

Reset value: 0000 0000 (00h)

| | | | | | | | |
|------|------|------|-------|-------|-------|-------|-------|
| 7 | | | | | | | 0 |
| ICIE | OCIE | TOIE | FOLV2 | FOLV1 | OLVL2 | IEDG1 | OLVL1 |

Bit 7 = **ICIE** *Input Capture Interrupt Enable*.

0: Interrupt is inhibited.

1: A timer interrupt is generated whenever the ICF1 or ICF2 bit of the SR register is set.

Bit 6 = **OCIE** *Output Compare Interrupt Enable*.

0: Interrupt is inhibited.

1: A timer interrupt is generated whenever the OCF1 or OCF2 bit of the SR register is set.

Bit 5 = **TOIE** *Timer Overflow Interrupt Enable*.

0: Interrupt is inhibited.

1: A timer interrupt is enabled whenever the TOF bit of the SR register is set.

Bit 4 = **FOLV2** *Forced Output Compare 2.*

This bit is set and cleared by software.

0: No effect on the OCMP2 pin.

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.

Bit 3 = **FOLV1** *Forced Output Compare 1.*

This bit is set and cleared by software.

0: No effect on the OCMP1 pin.

1: Forces OLVL1 to be copied to the OCMP1 pin, if the OC1E bit is set and even if there is no successful comparison.

Bit 2 = **OLVL2** *Output Level 2.*

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.

Bit 1 = **IEDG1** *Input Edge 1.*

This bit determines which type of level transition on the ICAP1 pin will trigger the capture.

0: A falling edge triggers the capture.

1: A rising edge triggers the capture.

Bit 0 = **OLVL1** *Output Level 1.*

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)

Read/ write

Reset value: 0000 0000 (00h)

| | | | | | | | |
|------|------|-----|-----|-----|-----|-------|---|
| 7 | | | | | | | 0 |
| OC1E | OC2E | OPM | PWM | CC1 | CC0 | IEDG2 | 0 |

Bit 7 = **OC1E** *Output Compare 1 Pin Enable.*

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.

0: OCMP1 pin alternate function disabled (I/O pin free for general-purpose I/O).

1: OCMP1 pin alternate function enabled.

Bit 6 = **OC2E** *Output Compare 2 Pin Enable.*

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.

0: OCMP2 pin alternate function disabled (I/O pin free for general-purpose I/O).

1: OCMP2 pin alternate function enabled.

Bit 5 = **OPM** *One Pulse Mode.*

0: One Pulse Mode is not active.

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.

Bit 4 = **PWM Pulse Width Modulation**.

0: PWM mode is not active.

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.

Bit 3, 2 = **CC[1:0] Clock Control**.

The timer clock mode depends on these bits:

Table 55. Clock control bits

| Timer clock | CC1 | CC0 |
|-------------------------|-----|-----|
| $f_{CPU} / 4$ | 0 | 0 |
| $f_{CPU} / 2$ | 0 | 1 |
| $f_{CPU} / 8$ | 1 | 0 |
| $f_{OSC2} / 8000^{(1)}$ | 1 | 1 |

1. Not available in Slow mode in ST72F561.

Bit 1 = **IEDG2 Input Edge 2**.

This bit determines which type of level transition on the ICAP2 pin will trigger the capture.

0: A falling edge triggers the capture.

1: A rising edge triggers the capture.

Bit 0 = Reserved, must be kept at 0.

13.7.3 Control/status register (CSR)

Read only (except bit 2 R/W)

Reset value: 0000 0000 (00h)

| | | | | | | | |
|------|------|-----|------|------|------|---|---|
| 7 | | | | | | | 0 |
| ICF1 | OCF1 | TOF | ICF2 | OCF2 | TIMD | 0 | 0 |

Bit 7 = **ICF1 Input Capture Flag 1**.

0: no input capture (reset value).

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 IC1R register.

Bit 6 = **OCF1 Output Compare Flag 1**.

0: no match (reset value).

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 OC1R register.

Bit 5 = **TOF Timer Overflow Flag**.

0: no timer overflow (reset value).

1: the free running counter rolled over from FFh to 00h. To clear this bit, first read the SR register, then read or write the CTR register.

Note: Reading or writing the ACTR register does not clear TOF.

Bit 4 = **ICF2** Input Capture Flag 2.

0: no input capture (reset value).

1: an input capture has occurred on the ICAP2 pin. To clear this bit, first read the SR register, then read or write the IC2R register.

Bit 3 = **OCF2** Output Compare Flag 2.

0: no match (reset value).

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 OC2R register.

Bit 2 = **TIMD** Timer disable.

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.

0: Timer enabled

1: Timer prescaler, counter and outputs disabled

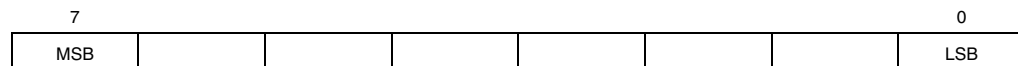
Bits 1:0 = Reserved, must be kept cleared.

13.7.4 Input capture 1 register (IC1R)

Read only

Reset value: Undefined

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

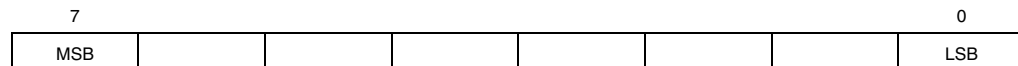


13.7.5 Output compare 1 register (OC1R)

Read/write

Reset value: 0000 0000 (00h)

This is an 8-bit register that contains the value to be compared to the CTR register.

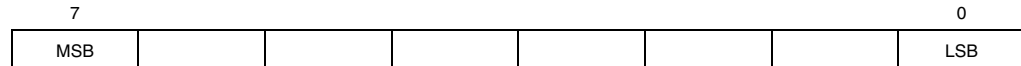


13.7.6 Output compare 2 register (OC2R)

Read/ write

Reset value: 0000 0000 (00h)

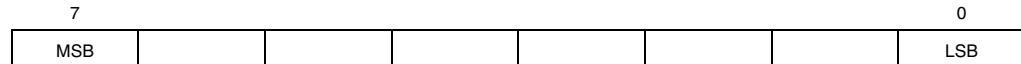
This is an 8-bit register that contains the value to be compared to the CTR register.

**13.7.7 Counter register (CTR)**

Read only

Reset value: 1111 1100 (FCh)

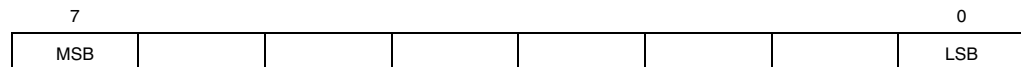
This is an 8-bit register that contains the counter value. A write to this register resets the counter. An access to this register after accessing the CSR register clears the TOF bit.

**13.7.8 Alternate counter register (ACTR)**

Read only

Reset value: 1111 1100 (FCh)

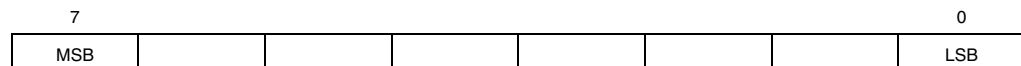
This is an 8-bit register that contains the counter value. A write to this register resets the counter. An access to this register after an access to CSR register does not clear the TOF bit in the CSR register.

**13.7.9 Input capture 2 register (IC2R)**

Read Only

Reset value: Undefined

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



13.8 8-bit timer register map

| Address (Hex.) | Register name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------------|---------------|------|------|------|-------|-------|-------|-------|-------|
| 3C | CR2 | OC1E | OC2E | OPM | PWM | CC1 | CC0 | IEDG2 | 0 |
| 3D | CR1 | ICIE | OCIE | TOIE | FOLV2 | FOLV1 | OLVL2 | IEDG1 | OLVL1 |
| 3E | CSR | ICF1 | OCF1 | TOF | ICF2 | OCF2 | TIMD | | |
| 3F | IC1R | MSB | | | | | | | LSB |
| 40 | OC1R | | | | | | | | |
| 41 | CTR | | | | | | | | |
| 42 | ACTR | | | | | | | | |
| 43 | IC2R | | | | | | | | |
| 44 | OC2R | | | | | | | | |

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 or a system in which devices may be either masters or slaves.

14.2 Main features

- Full duplex synchronous transfers (on three lines)
- Simplex synchronous transfers (on two 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

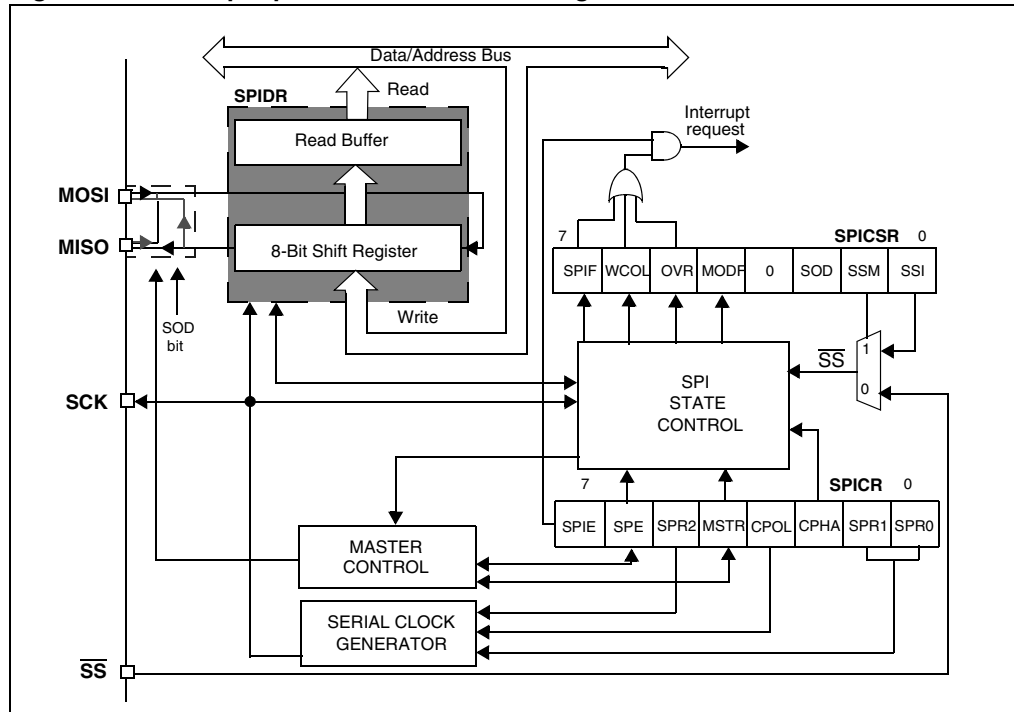
Figure 70 shows the serial peripheral interface (SPI) block diagram. There are three registers:

- 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 Device.

Figure 70. 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 71](#).

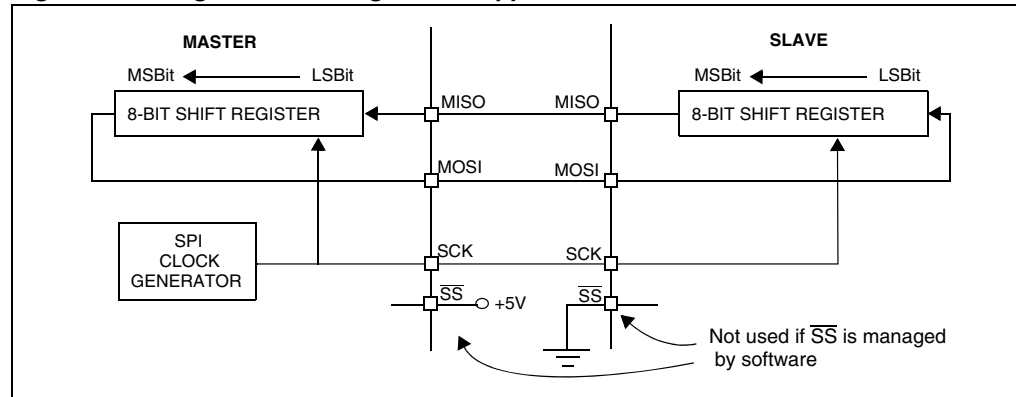
The MOSI pins are connected together and the MISO pins are connected together. In this way data is transferred serially between master and slave (most significant bit first).

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 74](#)) but master and slave must be programmed with the same timing mode.

Figure 71. 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 73](#)).

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 72](#)):

If CPHA = 1 (data latched on second 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 first 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 \(WCOL\)](#)).

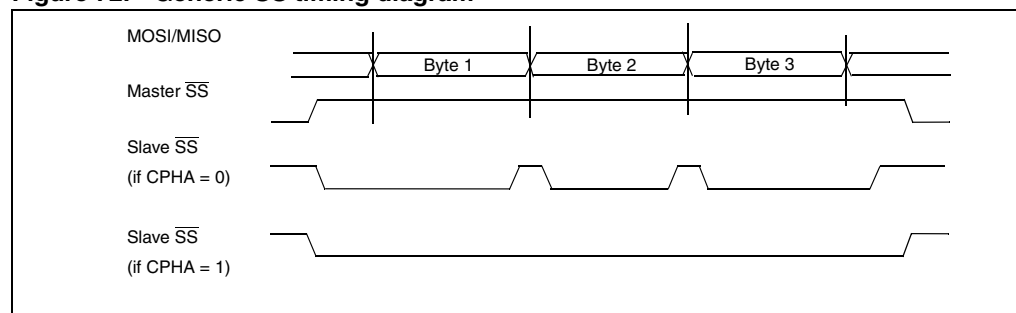
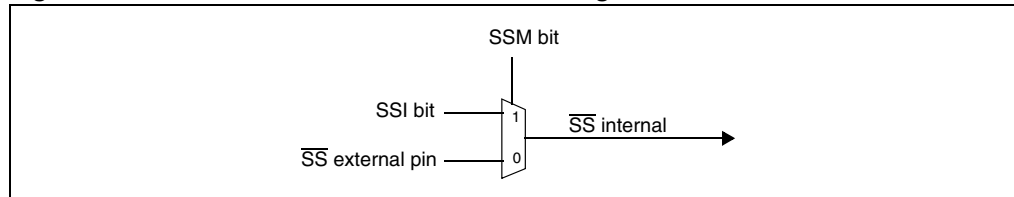
Figure 72. Generic \overline{SS} timing diagram

Figure 73. Hardware/software slave select management

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:
 - Select the clock frequency by configuring the SPR[2:0] bits.
 - Select the clock polarity and clock phase by configuring the CPOL and CPHA bits. [Figure 74](#) 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).

Caution: If the SPICSR register is not written first, the SPICR register setting (MSTR bit) may be not 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:
 - Select the clock polarity and clock phase by configuring the CPOL and CPHA bits (see [Figure 74](#)).

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

- Manage the \overline{SS} pin as described in [Slave select management](#) and [Figure 72](#). 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.

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 \(OVR\)](#)).

14.4 Clock phase and clock polarity

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits (See [Figure 74](#)).

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).

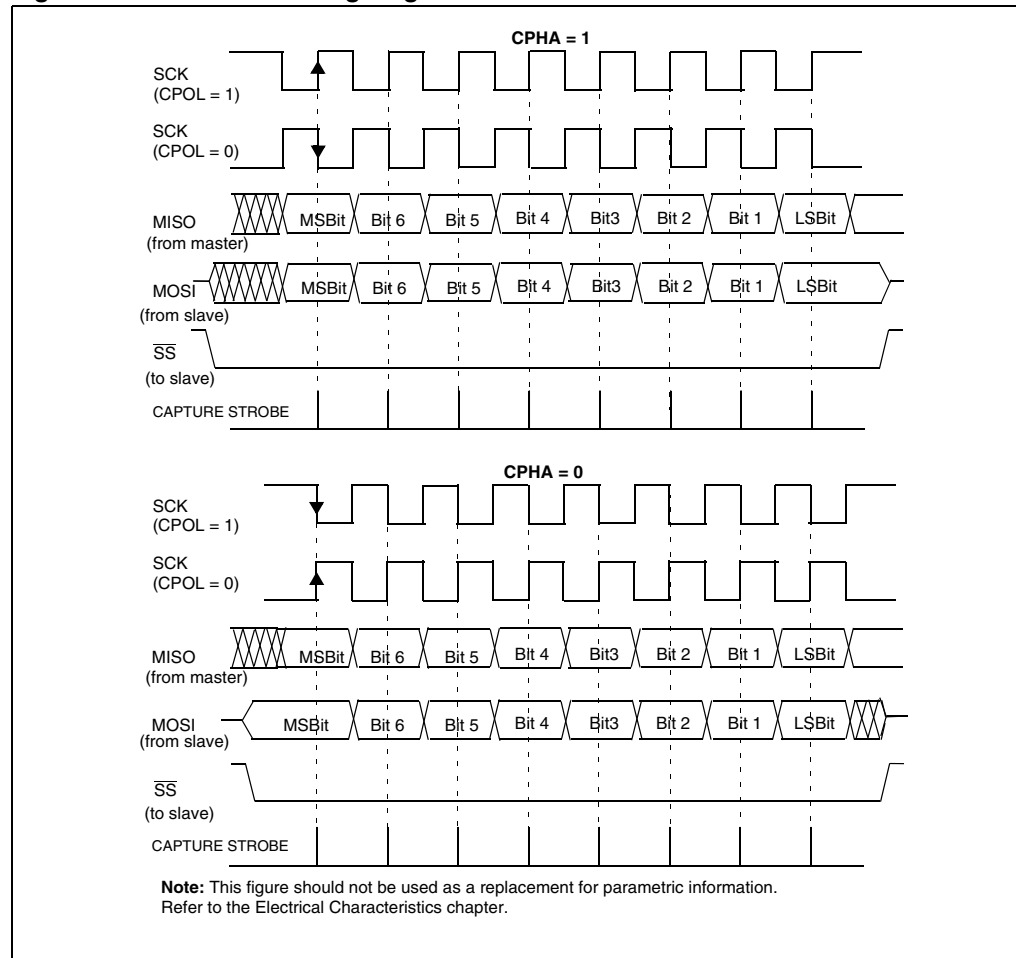
The combination of the CPOL clock polarity and CPHA (clock phase) bits selects the data capture clock edge.

[Figure 74](#) 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 and 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.

Figure 74. Data clock timing diagram



14.5 Error flags

14.5.1 Master mode fault (MODF)

Master mode fault occurs when the master device's \overline{SS} pin is 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.

In a slave device, the MODF bit can not be set, but in a multimaster configuration the device can be in slave mode with the MODF bit set.

The MODF bit indicates that there might have been a multimaster conflict and allows software to handle this using an interrupt routine and either perform a reset or return to an application default state.

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.

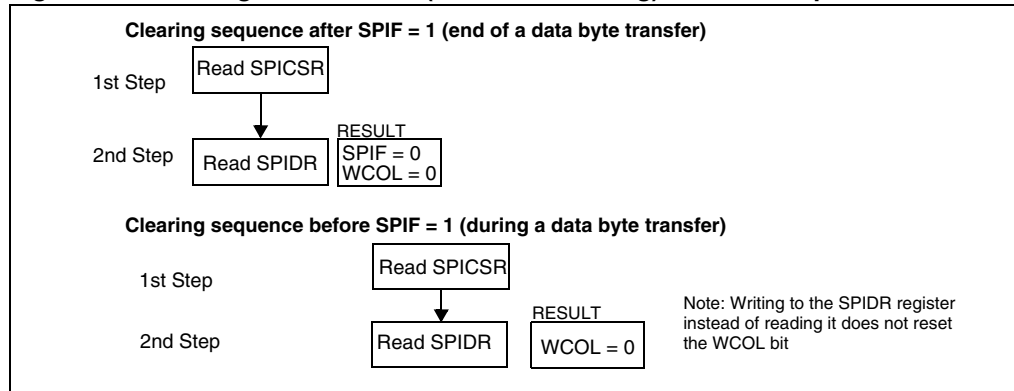
Write collisions can occur both in master and slave mode. See also [Slave select management](#).

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 CPU 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 75](#)).

Figure 75. Clearing the WCOL bit (write collision flag) software sequence

Single master and multimaster configurations

There are two types of SPI systems:

- Single Master System
- Multimaster System

Single master system

A typical single master system may be configured using a device as the master and four devices as slaves (see [Figure 76](#)).

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.

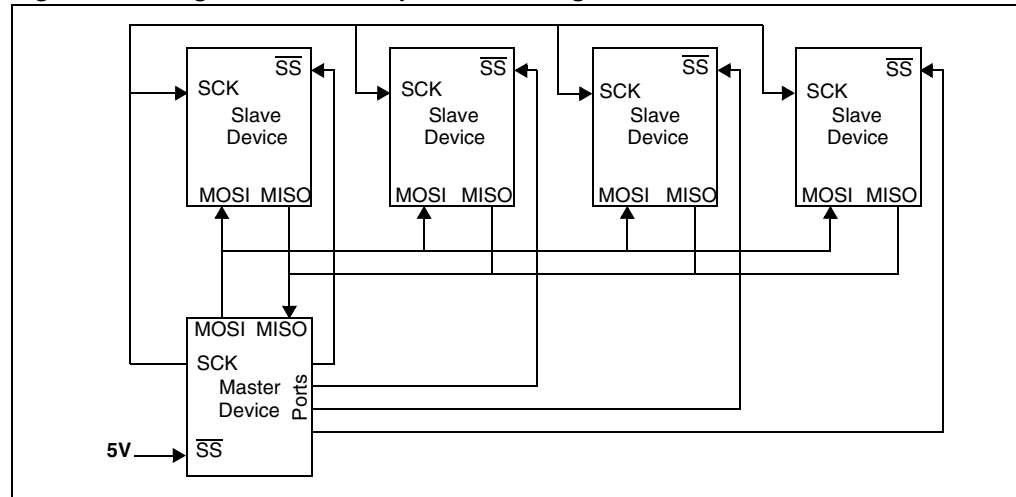
Other transmission security methods can use ports for handshake lines or data bytes with command fields.

Multimaster system

A multimaster system may also be configured by the user. Transfer of master control could be implemented using a handshake method through the I/O ports or by an exchange of code messages through the serial peripheral interface system.

The multimaster system is principally handled by the MSTR bit in the SPICR register and the MODF bit in the SPICSR register.

Figure 76. Single master / multiple slave configuration



14.6 Low power modes

Table 56. Effect of low power modes on SPI

| Mode | Description |
|------|---|
| WAIT | No effect on SPI. SPI interrupt events cause the device to exit from WAIT mode. |
| HALT | SPI registers are frozen. In HALT mode, the SPI is inactive. SPI operation resumes when the device 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. |

Using the SPI to wake up the device from halt mode

In slave configuration, the SPI is able to wake up the 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 device 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 device enters HALT mode. So, if Slave selection is configured as external (see [Slave select management](#)), make sure the master drives a low level on the \overline{SS} pin when the slave enters HALT mode.

14.7 Interrupts

Table 57. SPI interrupt control and 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).

14.8 Register description

14.8.1 Control register (SPICR)

Read/ write

Reset value: 0000 xxxx (0xh)

| | | | | | | | |
|------|-----|------|------|------|------|------|------|
| 7 | | | | | | | 0 |
| SPIE | SPE | SPR2 | MSTR | CPOL | CPHA | SPR1 | SPR0 |

Bit 7 = **SPIE** *Serial Peripheral Interrupt Enable*

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An SPI interrupt is generated whenever an End of Transfer event, Master Mode Fault or Overrun error occurs (SPIF = 1, MODF = 1 or OVR = 1 in the SPICSR register)

Bit 6 = **SPE** *Serial Peripheral Output Enable*

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 \(MODF\)](#)). The SPE bit is cleared by reset, so the SPI peripheral is not initially connected to the external pins.

0: I/O pins free for general purpose I/O

1: SPI I/O pin alternate functions enabled

Bit 5 = **SPR2** *Divider Enable*

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 58](#).

0: Divider by 2 enabled

1: Divider by 2 disabled

Note: This bit has no effect in slave mode.

Bit 4 = MSTR Master Mode

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 \(MODF\)](#)).

0: Slave mode

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.

Bit 3 = CPOL Clock Polarity

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.

0: SCK pin has a low level idle state

1: SCK pin has a high level idle state

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

Bit 2 = CPHA Clock Phase

This bit is set and cleared by software.

0: The first clock transition is the first data capture edge.

1: The second clock transition is the first capture edge.

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

Bits 1:0 = SPR[1:0] Serial Clock Frequency

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.

Note: These 2 bits have no effect in slave mode.

Table 58. SPI master mode SCK frequency

| Serial clock | SPR2 | SPR1 | SPR0 |
|---------------|------|------|------|
| $f_{CPU}/4$ | 1 | 0 | 0 |
| $f_{CPU}/8$ | 0 | | 1 |
| $f_{CPU}/16$ | | 1 | 1 |
| $f_{CPU}/32$ | 0 | 0 | |
| $f_{CPU}/64$ | | 1 | |
| $f_{CPU}/128$ | | | 1 |

14.8.2 Control/status register (SPICSR)

Read/ write (some bits Read Only)

Reset value: 0000 0000 (00h)

| | |
|------|-----|
| 7 | 0 |
| SPIF | SSI |
| WCOL | SSM |
| OVR | SOD |
| MODF | - |

Bit 7 = SPIF Serial Peripheral Data Transfer Flag (Read only)

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).

0: Data transfer is in progress or the flag has been cleared.

1: Data transfer between the device and an external device has been completed.

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

Bit 6 = **WCOL** Write Collision status (Read only)

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 75](#)).

0: No write collision occurred

1: A write collision has been detected

Bit 5 = **OVR** SPI Overrun error (Read only)

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 \(OVR\)](#)). An interrupt is generated if SPIE = 1 in the SPICR register. The OVR bit is cleared by software reading the SPICSR register.

0: No overrun error

1: Overrun error detected

Bit 4 = **MODF** Mode Fault flag (Read only)

This bit is set by hardware when the \overline{SS} pin is pulled low in master mode (see [Master mode fault \(MODF\)](#)). An SPI interrupt can be generated if SPIE = 1 in the SPICR register. This bit is cleared by a software sequence (An access to the SPICSR register while MODF = 1 followed by a write to the SPICR register).

0: No master mode fault detected

1: A fault in master mode has been detected

Bit 3 = Reserved, must be kept cleared.

Bit 2 = **SOD** SPI Output Disable

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)

0: SPI output enabled (if SPE = 1)

1: SPI output disabled

Bit 1 = **SSM** \overline{SS} Management

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](#).

0: Hardware management (\overline{SS} managed by external pin)

1: Software management (internal \overline{SS} signal controlled by SSI bit. External \overline{SS} pin free for general-purpose I/O)

Bit 0 = **SSI** \overline{SS} Internal Mode

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.

0: Slave selected

1: Slave deselected

14.8.3 Data I/O register (SPIDR)

Read/ write

Reset value: Undefined

| | | | | | | | | |
|----|----|----|----|----|----|----|----|---|
| 7 | | | | | | | | 0 |
| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | |

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.

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 70](#)).

Table 59. SPI register map and reset values

| Address (Hex.) | Register label | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------------|-----------------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| 21 | SPIDR Reset value | MSB x | x | x | x | x | x | x | LSB x |
| 22 | SPICR Reset value | SPIE 0 | SPE 0 | SPR2 0 | MSTR 0 | CPOL x | CPHA x | SPR1 x | SPR0 x |
| 23 | SPICSR Reset value | SPIF 0 | WCOL 0 | OVR 0 | MODF 0 | 0 | SOD 0 | SSM 0 | SSI 0 |

15 LINSICI serial communication interface (LIN master/slave)

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.

The LIN-dedicated features support the LIN (Local Interconnect Network) protocol for both master and slave nodes.

This chapter is divided into SCI Mode and LIN mode sections. For information on general SCI communications, refer to the SCI mode section. For LIN applications, refer to both the SCI mode and LIN mode sections.

15.2 SCI features

- Full duplex, asynchronous communications
- NRZ standard format (Mark/Space)
- 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
- Overrun, Noise and Frame error detection
- 6 interrupt sources
 - Transmit data register empty
 - Transmission complete
 - Receive data register full
 - Idle line received
 - Overrun error
 - Parity interrupt
- Parity control:
 - Transmits parity bit
 - Checks parity of received data byte
- Reduced power consumption mode

15.3 LIN features

- LIN master
 - 13-bit LIN synch break generation
- LIN slave
 - Automatic header handling
 - Automatic baud rate resynchronization based on recognition and measurement of the LIN synch field (for LIN slave nodes)
 - Automatic baud rate adjustment (at CPU frequency precision)
 - 11-bit LIN synch break detection capability
 - LIN parity check on the LIN identifier field (only in reception)
 - LIN error management
 - LIN header timeout
 - Hot plugging support

15.4 General description

The interface is externally connected to another device by two pins:

- TDO: Transmit Data Output. When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is 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 characters 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 character is complete

This interface uses three 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
- A LIN baud rate generator with automatic resynchronization

15.5.2 Extended prescaler mode

Two additional prescalers are available in extended prescaler mode. They are shown in [Figure 79](#).

- An extended prescaler receiver register (SCIERPR)
- An extended prescaler transmitter register (SCIETPR)

15.5.3 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 78](#)).

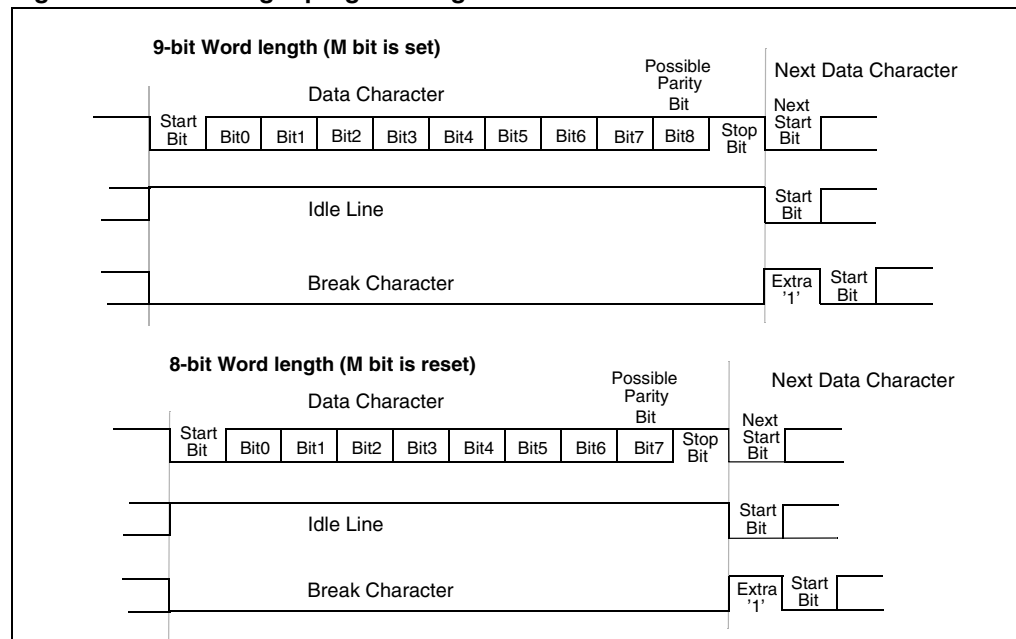
The TDO pin is in low state during the start bit.

The TDO pin is in high state during the stop bit.

An idle character is interpreted as a continuous logic high level for 10 (or 11) full bit times.

A break character is a character with a sufficient number of low level bits to break the normal data format followed by an extra "1" bit to acknowledge the start bit.

Figure 78. Word length programming



15.5.4 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 77](#)).

Procedure

- Select the M bit to define the word length.
- Select the desired baud rate using the SCIBRR and the SCIETPR registers.
- Set the TE bit to send a preamble of 10 (M = 0) or 11 (M = 1) consecutive ones (Idle Line) as first transmission.
- 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[1:0] bits are 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 character 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[1:0] bits are 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 character length depends on the M bit (see [Figure 78](#)).

As long as the SBK bit is set, the SCI sends break characters to the TDO pin. After clearing this bit by software, the SCI inserts a logic 1 bit at the end of the last break character to guarantee the recognition of the start bit of the next character.

Idle line

Setting the TE bit drives the SCI to send a preamble of 10 (M = 0) or 11 (M = 1) consecutive '1's (idle line) before the first character.

In this case, clearing and then setting the TE bit during a transmission sends a preamble (idle line) after the current word. Note that the preamble duration (10 or 11 consecutive '1's depending on the M bit) does not take into account the stop bit of the previous character.

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.

15.5.5 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 of a buffer (RDR) between the internal bus and the received shift register (see [Figure 77](#)).

Procedure

- Select the M bit to define the word length.
- Select the desired baud rate using the SCIBRR and the SCIERPR registers.
- 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[1:0] bits are 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.

Idle line

When an idle line is detected, there is the same procedure as a data received character plus an interrupt if the ILIE bit is set and the I[1:0] bits are cleared in the CCR register.

Overrun error

An overrun error occurs when a character is received when RDRF has not been reset. Data can not be transferred from the shift register to the TDR register as long as the RDRF bit is not cleared.

When an overrun error occurs:

- The OR bit is set.
- The RDR content will not be lost.
- The shift register will be overwritten.
- An interrupt is generated if the RIE bit is set and the I[1:0] bits are 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.

When noise is detected in a character:

- The NF bit 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 bit is reset by a SCISR register read operation followed by a SCIDR register read operation.

Framing error

A framing error is detected when:

- The stop bit is not recognized on reception at the expected time, following either a desynchronization 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.

Break character

- When a break character is received, the SCI handles it as a framing error. To differentiate a break character from a framing error, it is necessary to read the SCIDR. If the received value is 00h, it is a break character. Otherwise it is a framing error.

Conventional baud rate generation

The baud rates for the receiver and transmitter (Rx and Tx) are set independently and calculated as follows:

$$Tx = \frac{f_{CPU}}{(16 \cdot PR) \cdot TR} \quad Rx = \frac{f_{CPU}}{(16 \cdot PR) \cdot RR}$$

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 1: 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.

15.5.6 Extended baud rate generation

The extended prescaler option gives a very fine tuning on 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 [Figure 79](#).

The output clock rate sent to the transmitter or to the receiver will be the output from the 16 divider divided by a factor ranging from 1 to 255 set in the SCIETPR or the SCIERPR 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:

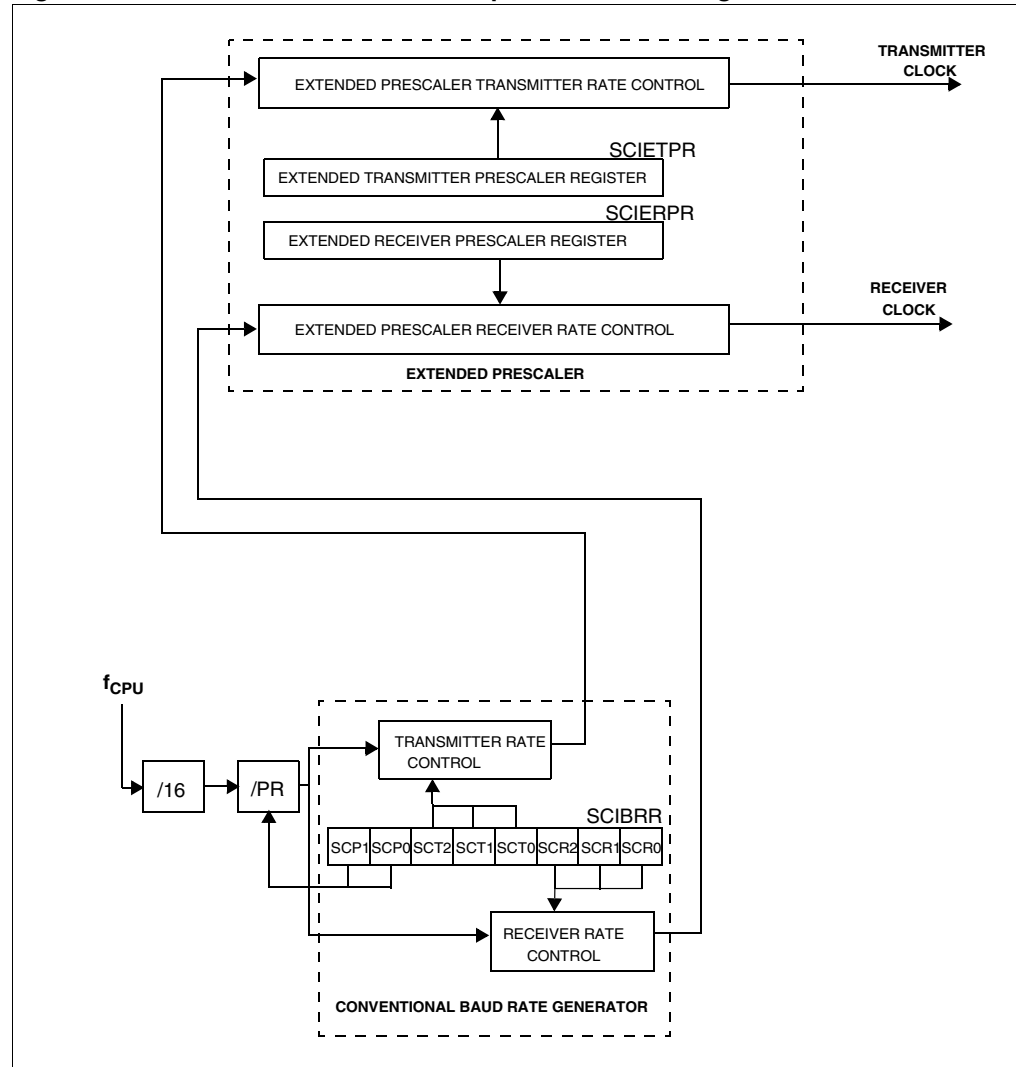
$$T_x = \frac{f_{\text{CPU}}}{16 \cdot \text{ETPR} \cdot (\text{PR} \cdot \text{TR})} \quad R_x = \frac{f_{\text{CPU}}}{16 \cdot \text{ERPR} \cdot (\text{PR} \cdot \text{RR})}$$

with:

ETPR = 1...255 (see SCIETPR register)

ERPR = 1...255 (see SCIERPR register)

Figure 79. SCI baud rate and extended prescaler block diagram



15.5.7 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 can not be set.

All the receive interrupts are inhibited.

A muted receiver may be woken up in one of the following ways:

- by Idle Line detection if the WAKE bit is reset,
- by Address Mark detection if the WAKE bit is set.

Idle line detection

Receiver wakes up by idle line detection when the receive line has recognized an Idle Line. Then the RWU bit is reset by hardware but the IDLE bit is not set.

This feature is useful in a multiprocessor system when the first characters of the message determine the address and when each message ends by an idle line: As soon as the line becomes idle, every receivers is waken up and analyze the first characters of the message which indicates the addressed receiver. The receivers which are not addressed set RWU bit to enter in mute mode. Consequently, they will not treat the next characters constituting the next part of the message. At the end of the message, an idle line is sent by the transmitter: this wakes up every receivers which are ready to analyze the addressing characters of the new message.

In such a system, the inter-characters space must be smaller than the idle time.

Address mark detection

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.

This feature is useful in a multiprocessor system when the most significant bit of each character (except for the break character) is reserved for Address Detection. As soon as the receivers received an address character (most significant bit = '1'), the receivers are waken up. The receivers which are not addressed set RWU bit to enter in mute mode. Consequently, they will not treat the next characters constituting the next part of the message.

15.5.8 Parity control

Hardware byte 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 character format defined by the M bit, the possible SCI character formats are as listed in [Table 60](#).

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

Table 60. Character formats⁽¹⁾

| M bit | PCE bit | Character format |
|-------|---------|--------------------------|
| 0 | 0 | ISBI 8 bit data ISTBI |
| | 1 | ISBI 7-bit data IPBISTBI |
| 1 | 0 | ISBI 9-bit data ISTBI |
| | 1 | ISBI 8-bit data IPBISTBI |

1. SB = Start Bit, STB = Stop Bit, PB = Parity Bit

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the character made of the 7 or 8 LSB bits (depending on whether M is equal to 0 or 1) and the parity bit.

Example 2: data = 00110101; 4 bits set => parity bit will be 0 if even parity is selected (PS bit = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the character made of the 7 or 8 LSB bits (depending on whether M is equal to 0 or 1) and the parity bit.

Example 3: data = 00110101; 4 bits set => parity bit will be 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 “1s” if even parity is selected (PS = 0) or an odd number of “1s” 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 PCIE is set in the SCICR1 register.

15.6 Low power modes

Table 61. Effect of low power modes on SCI

| Mode | Description |
|------|--|
| WAIT | No effect on SCI. SCI interrupts cause the device to exit from Wait mode. |
| HALT | SCI registers are frozen. In Halt mode, the SCI stops transmitting/receiving until Halt mode is exited. |

15.7 Interrupts

Table 62. SCI interrupt control and 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 | | |
| Received Data Ready to be Read | RDRF | RIE | | |
| Overrun Error or LIN Synch Error Detected | OR/LHE | | | |
| Idle Line Detected | IDLE | ILIE | | |
| Parity Error | PE | PIE | | |
| LIN Header Detection | LHDF | LHIE | | |



The SCI interrupt events are connected to the same interrupt vector (see Interrupts chapter).

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).

15.8 SCI mode register description

15.8.1 Status register (SCISR)

Read only

Reset value: 1100 0000 (C0h)

| | | | | | | | |
|------|----|------|------|----|----|----|----|
| 7 | | | | | | | 0 |
| TDRE | TC | RDRF | IDLE | OR | NF | FE | PE |

Bit 7 = TDRE *Transmit data register empty.*

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 = 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).

- 0: data is not transferred to the shift register
- 1: data is transferred to the shift register

Bit 6 = TC *Transmission complete.*

This bit is set by hardware when transmission of a character 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).

- 0: transmission is not complete
- 1: transmission is complete

Note: TC is not set after the transmission of a Preamble or a Break.

Bit 5 = RDRF *Received data ready flag.*

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).

- 0: data is not received
- 1: received data is ready to be read

Bit 4 = IDLE *Idle line detected.*

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).

- 0: no Idle Line is detected
- 1: idle Line is detected

Note: The IDLE bit will not be set again until the RDRF bit has been set itself (that is, a new idle line occurs).

Bit 3 = **OR** *Overrun error*

The OR bit is set by hardware when the word currently being received in the shift register is ready to be transferred into the RDR register whereas RDRF is still set. 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).

0: no overrun error
1: overrun error detected

Note: When this bit is set, RDR register contents will not be lost but the shift register will be overwritten.

Bit 2 = **NF** Character Noise flag

This bit is set by hardware when noise is detected on a received character. It is cleared by a software sequence (an access to the SCISR register followed by a read to the SCIDR register).

0: no noise
1: noise is detected

Note: This bit does not generate interrupt as it appears at the same time as the RDRF bit which itself generates an interrupt.

Bit 1 = **FE** Framing error.

This bit is set by hardware when a desynchronization, 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).

0: no Framing error
1: framing error or break character detected

Note: This bit does not generate an 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 a frame error and an overrun error, it will be transferred and only the OR bit will be set.

Bit 0 = **PE** Parity error.

This bit is set by hardware when a byte parity error occurs (if the PCE bit is set) 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.

0: no parity error
1: parity error detected

15.8.2 Control register 1 (SCICR1)

Read/ write

Reset value: x000 0000 (x0h)

| | | | | | | | |
|----|----|------|---|------|--------------------|----|-----|
| 7 | | | | | | | 0 |
| R8 | T8 | SCID | M | WAKE | PCE ⁽¹⁾ | PS | PIE |

1. This bit has a different function in LIN mode, please refer to the LIN mode register description.

Bit 7 = **R8** Receive data bit 8.

This bit is used to store the 9th bit of the received word when M = 1.

Bit 6 = **T8** Transmit data bit 8.

This bit is used to store the 9th bit of the transmitted word when M = 1.

Bit 5 = **SCID** *Disabled for low power consumption*

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.

- 0: SCI enabled
- 1: SCI prescaler and outputs disabled

Bit 4 = **M** *Word length.*

This bit determines the word length. It is set or cleared by software.

- 0: 1 start bit, 8 data bits, 1 stop bit
- 1: 1 start bit, 9 data bits, 1 stop bit

Note: The M bit must not be modified during a data transfer (both transmission and reception).

Bit 3 = **WAKE** *Wake-Up method.*

This bit determines the SCI Wake-Up method, it is set or cleared by software.

- 0: idle line
- 1: address mark

Note: If the LINE bit is set, the WAKE bit is deactivated and replaced by the LHDM bit.

Bit 2 = **PCE** *Parity control enable.*

This bit is set and cleared by software. It selects the hardware parity control (generation and detection for byte parity, detection only for LIN parity).

- 0: parity control disabled
- 1: parity control enabled

Bit 1 = **PS** *Parity selection.*

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 will be selected after the current byte.

- 0: even parity
- 1: odd parity

Bit 0 = **PIE** *Parity interrupt enable.*

This bit enables the interrupt capability of the hardware parity control when a parity error is detected (PE bit set). The parity error involved can be a byte parity error (if bit PCE is set and bit LPE is reset) or a LIN parity error (if bit PCE is set and bit LPE is set).

- 0: parity error interrupt disabled
- 1: parity error interrupt enabled

15.8.3 Control register 2 (SCICR2)

Read/ write

Reset value: 0000 0000 (00h)

| | | | | | | | |
|-----|------|-----|------|----|----|--------------------|--------------------|
| 7 | | | | | | | 0 |
| TIE | TCIE | RIE | ILIE | TE | RE | RWU ⁽¹⁾ | SBK ⁽¹⁾ |

1. This bit has a different function in LIN mode, please refer to the LIN mode register description.

Bit 7 = **TIE** *Transmitter interrupt enable.*

This bit is set and cleared by software.

- 0: interrupt is inhibited
- 1: in SCI interrupt is generated whenever TDRE = 1 in the SCISR register

Bit 6 = **TCIE** *Transmission complete interrupt enable*

This bit is set and cleared by software.

0: interrupt is inhibited

1: an SCI interrupt is generated whenever TC = 1 in the SCISR register

Bit 5 = **RIE** *Receiver interrupt enable.*

This bit is set and cleared by software.

0: interrupt is inhibited

1: an SCI interrupt is generated whenever OR = 1 or RDRF = 1 in the SCISR register

Bit 4 = **ILIE** *Idle line interrupt enable.*

This bit is set and cleared by software.

0: interrupt is inhibited

1: an SCI interrupt is generated whenever IDLE = 1 in the SCISR register.

Bit 3 = **TE** *Transmitter enable.*

This bit enables the transmitter. It is set and cleared by software.

0: transmitter is disabled

1: transmitter is enabled

Note: During transmission, a "0" pulse on the TE bit ("0" followed by "1") sends a preamble (idle line) after the current word.

When TE is set there is a 1 bit-time delay before the transmission starts.

Bit 2 = **RE** *Receiver enable.*

This bit enables the receiver. It is set and cleared by software.

0: receiver is disabled in the SCISR register

1: receiver is enabled and begins searching for a start bit

Bit 1 = **RWU** *Receiver wake-up.*

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.

0: receiver in active mode

1: receiver in mute mode

Note: Before selecting Mute mode (by setting the RWU bit) the SCI must first receive a data byte, otherwise it cannot function in Mute mode with wakeup by Idle line detection.

In address mark detection wake-up configuration (WAKE bit = 1) the RWU bit cannot be modified by software while the RDRF bit is set.

Bit 0 = **SBK** *Send break.*

This bit set is used to send break characters. It is set and cleared by software.

0: no break character is transmitted

1: break characters are transmitted

Note: If the SBK bit is set to "1" and then to "0", the transmitter will send a BREAK word at the end of the current word.

15.8.4 Data register (SCIDR)

Read/ write

Reset value: Undefined

Contains the received or transmitted data character, depending on whether it is read from or written to.

| | | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|---|
| 7 | | | | | | | | 0 |
| DR7 | DR6 | DR5 | DR4 | DR3 | DR2 | DR1 | DR0 | |

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 77](#)).

The RDR register provides the parallel interface between the input shift register and the internal bus (see [Figure 77](#)).

15.8.5 Baud rate register (SCIBRR)

Read/ write

Reset value: 0000 0000 (00h)

| | | | | | | | | |
|------|------|------|------|------|------|------|------|---|
| 7 | | | | | | | | 0 |
| SCP1 | SCP0 | SCT2 | SCT1 | SCT0 | SCR2 | SCR1 | SCR0 | |

Note: When LIN slave mode is disabled, the SCIBRR register controls the conventional baud rate generator.

Bits 7:6 = **SCP[1:0]** First SCI Prescaler

These 2 prescaling bits allow several standard clock division ranges

Table 63. PR prescaler

| PR prescaling factor | SCP1 | SCP0 |
|----------------------|------|------|
| 1 | 0 | 0 |
| 3 | | 1 |
| 4 | 1 | 0 |
| 13 | | 1 |

Bits 5:3 = **SCT[2:0]** SCI Transmitter rate divisor

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.

Table 64. Transmitter rate divider

| TR dividing factor | SCT2 | SCT1 | SCT0 |
|--------------------|------|------|------|
| 1 | 0 | 0 | 0 |
| 2 | | | 1 |
| 4 | | 1 | 0 |
| 8 | | | 1 |

Table 64. Transmitter rate divider

| TR dividing factor | SCT2 | SCT1 | SCT0 |
|--------------------|------|------|------|
| 16 | 1 | 0 | 0 |
| 32 | | | 1 |
| 64 | | 1 | 0 |
| 128 | | | 1 |

Bits 2:0 = **SCR[2:0]** *SCI Receiver rate divider*.

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.

Table 65. Receiver rate divider

| RR dividing factor | SCR2 | SCR1 | SCR0 |
|--------------------|------|------|------|
| 1 | 0 | 0 | 0 |
| 2 | | | 1 |
| 4 | | 1 | 0 |
| 8 | | | 1 |
| 16 | 1 | 0 | 0 |
| 32 | | | 1 |
| 64 | | 1 | 0 |
| 128 | | | 1 |

15.8.6 Extended receive prescaler division register (SCI ERPR)

Read/ write

Reset value: 0000 0000 (00h)

| | | | | | | | |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 7 | | | | | | | 0 |
| ERPR7 | ERPR6 | ERPR5 | ERPR4 | ERPR3 | ERPR2 | ERPR1 | ERPR0 |

Bits 7:0 = **ERPR[7:0]** *8-bit Extended Receive Prescaler Register*.

The extended baud rate generator is activated when a value other than 00h is stored in this register. The clock frequency from the 16 divider (see [Figure 79](#)) is divided by the binary factor set in the SCI ERPR register (in the range 1 to 255).

The extended baud rate generator is not active after a reset.

15.8.7 Extended transmit prescaler division register (SCIETPR)

Read/ write

Reset value: 0000 0000 (00h)

| | | | | | | | |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 7 | | | | | | | 0 |
| ETPR7 | ETPR6 | ETPR5 | ETPR4 | ETPR3 | ETPR2 | ETPR1 | ETPR0 |

Bits 7:0 = **ETPR[7:0]** 8-bit Extended Transmit Prescaler Register.

The extended baud rate generator is activated when a value other than 00h is stored in this register. The clock frequency from the 16 divider (see [Figure 79](#)) is divided by the binary factor set in the SCIETPR register (in the range 1 to 255).

The extended baud rate generator is not active after a reset.

Note: In LIN slave mode, the conventional and extended baud rate generators are disabled.

15.9 LIN mode - functional description.

The block diagram of the Serial Control Interface, in LIN slave mode is shown in [Figure 81](#).

It uses six registers:

- 3 control registers: SCICR1, SCICR2 and SCICR3
- 2 status registers: the SCISR register and the LHLR register mapped at the SCIERPR address
- A baud rate register: LPR mapped at the SCIBRR address and an associated fraction register LPFR mapped at the SCIETPR address

The bits dedicated to LIN are located in the SCICR3. Refer to the register descriptions in [Section 15.10: LIN mode register description](#) for the definitions of each bit.

15.9.1 Entering LIN mode

To use the LINSICI in LIN mode the following configuration must be set in SCICR3 register:

- Clear the M bit to configure 8-bit word length.
- Set the LINE bit.

Master

To enter master mode the LSLV bit must be reset. In this case, setting the SBK bit will send 13 low bits.

Then the baud rate can be programmed using the SCIBRR, SCIERPR and SCIETPR registers.

In LIN master mode, the conventional and/ or extended prescaler define the baud rate (as in standard SCI mode)

Slave

Set the LSLV bit in the SCICR3 register to enter LIN slave mode. In this case, setting the SBK bit will have no effect.

In LIN Slave mode the LIN baud rate generator is selected instead of the conventional or extended prescaler. The LIN baud rate generator is common to the transmitter and the receiver.

Then the baud rate can be programmed using LPR and LPRF registers.

Note: It is mandatory to set the LIN configuration first before programming LPR and LPRF, because the LIN configuration uses a different baud rate generator from the standard one.

15.9.2 LIN transmission

In LIN mode the same procedure as in SCI mode has to be applied for a LIN transmission.

To transmit the LIN header the proceed as follows:

- First set the SBK bit in the SCICR2 register to start transmitting a 13-bit LIN synch break
- Reset the SBK bit
- Load the LIN synch field (0x55) in the SCIDR register to request synch field transmission
- Wait until the SCIDR is empty (TDRE bit set in the SCISR register)
- Load the LIN message identifier in the SCIDR register to request Identifier transmission.

Figure 80. LIN characters

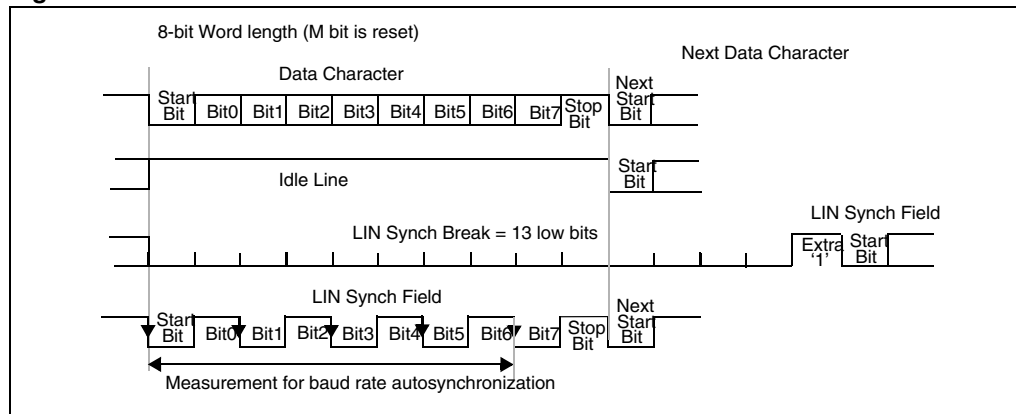
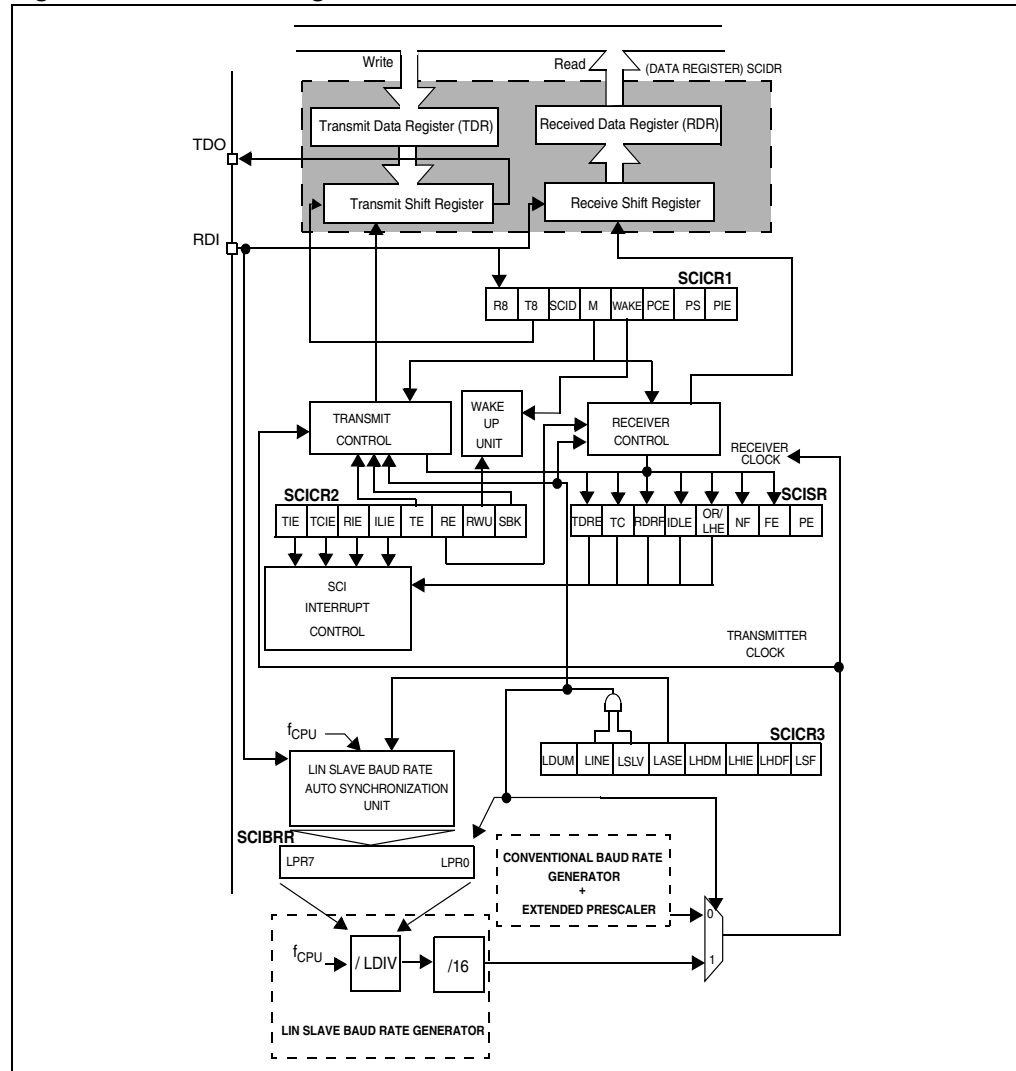


Figure 81. SCI block diagram in LIN slave mode



15.9.3 LIN reception

In LIN mode the reception of a byte is the same as in SCI mode but the LINSICI has features for handling the LIN header automatically (identifier detection) or semiautomatically (synch break detection) depending on the LIN Header detection mode. The detection mode is selected by the LHDM bit in the SCICR3.

Additionally, an automatic resynchronization feature can be activated to compensate for any clock deviation, for more details please refer to [LIN baud rate](#).

LIN header handling by a slave

Depending on the LIN header detection method the LINSICI will signal the detection of a LIN Header after the LIN synch break or after the Identifier has been successfully received.

Note: *It is recommended to combine the header detection function with Mute mode. Putting the LINSPI in mute mode allows the detection of Headers only and prevents the reception of any other characters.*

This mode can be used to wait for the next header without being interrupted by the data bytes of the current message in case this message is not relevant for the application.

Synch break detection (LHDM = 0)

When a LIN synch break is received:

- The RDRF bit in the SCISR register is set. It indicates that the content of the shift register is transferred to the SCIDR register, a value of 0x00 is expected for a break.
- The LHDF flag in the SCICR3 register indicates that a LIN synch break field has been detected.
- An interrupt is generated if the LHIE bit in the SCICR3 register is set and the I[1:0] bits are cleared in the CCR register.
- Then the LIN synch field is received and measured.
 - If automatic resynchronization is enabled (LASE bit = 1), the LIN synch field is not transferred to the shift register: there is no need to clear the RDRF bit.
 - If automatic resynchronization is disabled (LASE bit = 0), the LIN synch field is received as a normal character and transferred to the SCIDR register and RDRF is set.

Note: *In LIN slave mode, the FE bit detects all frame error which does not correspond to a break.*

Identifier detection (LHDM = 1)

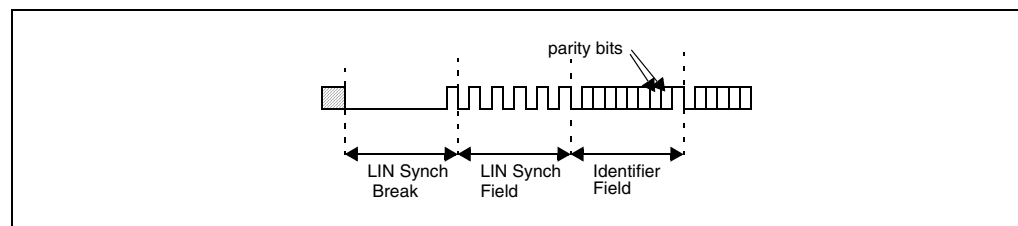
This case is the same as the previous one except that the LHDF and the RDRF flags are set only after the entire header has been received (this is true whether automatic resynchronization is enabled or not). This indicates that the LIN Identifier is available in the SCIDR register.

Note: *During LIN synch field measurement, the SCI state machine is switched off: no characters are transferred to the data register.*

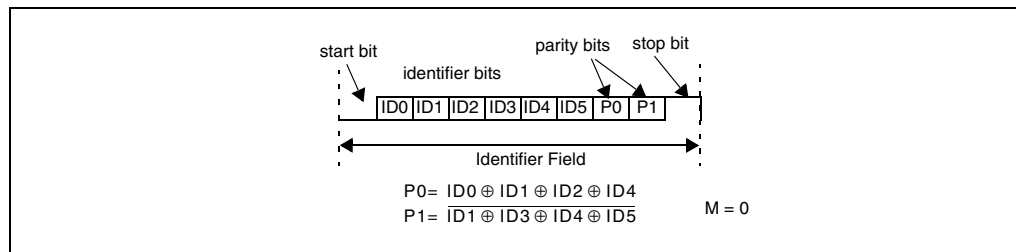
LIN slave parity

In LIN slave mode (LINE and LSLV bits are set) LIN parity checking can be enabled by setting the PCE bit.

In this case, the parity bits of the LIN identifier field are checked. The identifier character is recognized as the 3rd received character after a break character (included):



The bits involved are the two MSB positions (7th and 8th bits if M = 0; 8th and 9th bits if M = 0) of the identifier character. The check is performed as specified by the LIN specification:



15.9.4 LIN error detection

LIN header error flag

The LIN header error flag indicates that an invalid LIN header has been detected.

When a LIN header error occurs:

- The LHE flag is set
- An interrupt is generated if the RIE bit is set and the I[1:0] bits are cleared in the CCR register.

If autosynchronization is enabled (LASE bit = 1), this can mean that the LIN synch field is corrupted, and that the SCI is in a blocked state (LSF bit is set). The only way to recover is to reset the LSF bit and then to clear the LHE bit.

- The LHE bit is reset by an access to the SCISR register followed by a read of the SCIDR register.

LHE/OVR error conditions

When auto resynchronization is disabled (LASE bit = 0), the LHE flag detects:

- That the received LIN synch field is not equal to 55h.
- That an overrun occurred (as in standard SCI mode)
- Furthermore, if LHDM is set it also detects that a LIN header reception timeout occurred (only if LHDM is set).

When the LIN auto-resynchronization is enabled (LASE bit = 1), the LHE flag detects:

- That the deviation error on the synch field is outside the LIN specification which allows up to +/-15.5% of period deviation between the slave and master oscillators.
- A LIN header reception timeout occurred.
If $T_{\text{HEADER}} > T_{\text{HEADER_MAX}}$ then the LHE flag is set. Refer to [Figure 82](#). (only if LHDM is set to 1)
- An overflow during the synch field measurement, which leads to an overflow of the divider registers. If LHE is set due to this error then the SCI goes into a blocked state (LSF bit is set).
- That an overrun occurred on fields other than the synch field (as in standard SCI mode)

Deviation error on the synch field

The deviation error is checking by comparing the current baud rate (relative to the slave oscillator) with the received LIN synch field (relative to the master oscillator). Two checks are performed in parallel:

- The first check is based on a measurement between the first falling edge and the last falling edge of the synch field. Let us refer to this period deviation as D:

If the LHE flag is set, it means that:

$D > 15.625\%$

If LHE flag is not set, it means that:

$D < 16.40625\%$

If $15.625\% \leq D < 16.40625\%$, then the flag can be either set or reset depending on the dephasing between the signal on the RDI line and the CPU clock.

- The second check is based on the measurement of each bit time between both edges of the synch field: this checks that each of these bit times is large enough compared to the bit time of the current baud rate.

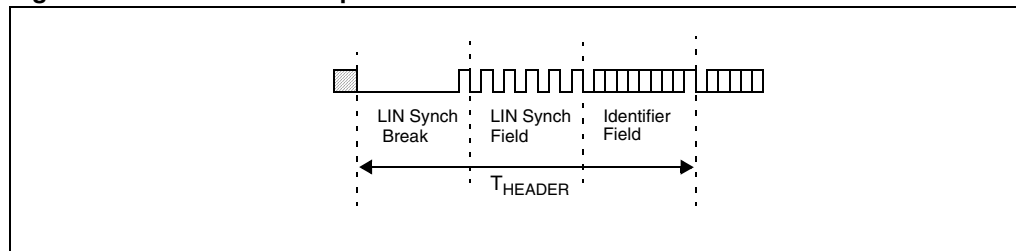
When LHE is set due to this error then the SCI goes into a blocked state (LSF bit is set).

LIN header time-out error

When the LIN Identifier field detection method is used (by configuring LHDM to 1) or when LIN auto-resynchronization is enabled (LASE bit = 1), the LINSCI automatically monitors the $T_{\text{HEADER_MAX}}$ condition given by the LIN protocol.

If the entire Header (up to and including the STOP bit of the LIN identifier field) is not received within the maximum time limit of 57 bit times then a LIN header error is signalled and the LHE bit is set in the SCISR register.

Figure 82. LIN header reception timeout



The time-out counter is enabled at each break detection. It is stopped in the following conditions:

- A LIN identifier field has been received
- An LHE error occurred (other than a timeout error).
- A software reset of LSF bit (transition from high to low) occurred during the analysis of the LIN synch field or

If LHE bit is set due to this error during the LIN synchr field (if LASE bit = 1) then the SCI goes into a blocked state (LSF bit is set).

If LHE bit is set due to this error during fields other than LIN synchr field or if LASE bit is reset then the current received header is discarded and the SCI searches for a new break field.

Note on LIN header time-out limit

According to the LIN specification, the maximum length of a LIN header which does not cause a timeout is equal to $1.4 * (34 + 1) = 49 T_{\text{BIT_MASTER}}$.

$T_{\text{BIT_MASTER}}$ refers to the master baud rate.

When checking this timeout, the slave node is desynchronized for the reception of the LIN break and synch fields. Consequently, a margin must be allowed, taking into account the

worst case: This occurs when the LIN identifier lasts exactly $10 T_{\text{BIT_MASTER}}$ periods. In this case, the LIN break and synch fields last $49 - 10 = 39 T_{\text{BIT_MASTER}}$ periods.

Assuming the slave measures these first 39 bits with a desynchronized clock of 15.5%. This leads to a maximum allowed header length of:

$$39 \times (1/0.845) T_{\text{BIT_MASTER}} + 10 T_{\text{BIT_MASTER}} = 56.15 T_{\text{BIT_SLAVE}}$$

A margin is provided so that the time-out occurs when the header length is greater than $57 T_{\text{BIT_SLAVE}}$ periods. If it is less than or equal to $57 T_{\text{BIT_SLAVE}}$ periods, then no timeout occurs.

LIN header length

Even if no timeout occurs on the LIN header, it is possible to have access to the effective LIN header length (T_{HEADER}) through the LHL register. This allows monitoring at software level the $T_{\text{FRAME_MAX}}$ condition given by the LIN protocol.

This feature is only available when LHDM bit = 1 or when LASE bit = 1.

Mute mode and errors

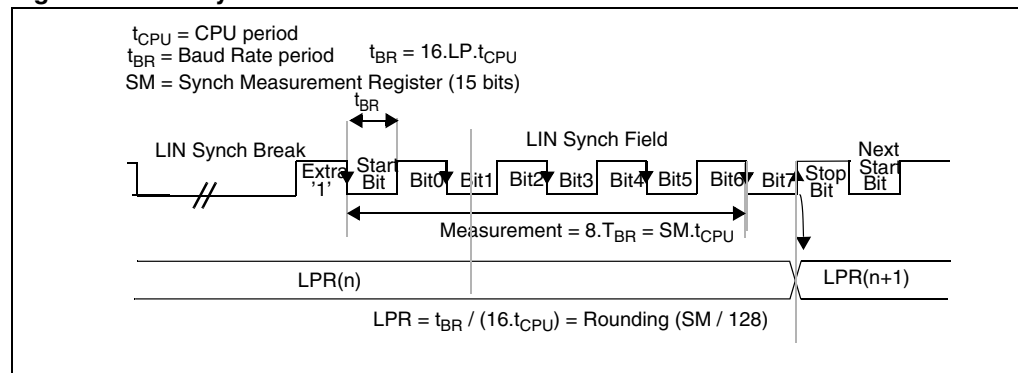
In mute mode when LHDM bit = 1, if an LHE error occurs during the analysis of the LIN synch field or if a LIN header time-out occurs then the LHE bit is set but it does not wake up from mute mode. In this case, the current header analysis is discarded. If needed, the software has to reset LSF bit. Then the SCI searches for a new LIN header.

In mute mode, if a framing error occurs on a data (which is not a break), it is discarded and the FE bit is not set.

When LHDM bit = 1, any LIN header which respects the following conditions causes a wake-up from mute mode:

- A valid LIN break field (at least 11 dominant bits followed by a recessive bit)
- A valid LIN synch field (without deviation error)
- A LIN identifier field without framing error. Note that a LIN parity error on the LIN identifier field does not prevent wake-up from mute mode.
- No LIN header time-out should occur during header reception.

Figure 83. LIN synch field measurement



15.9.5 LIN baud rate

Baud rate programming is done by writing a value in the LPR prescaler or performing an automatic resynchronization as described below.

Automatic resynchronization

To automatically adjust the baud rate based on measurement of the LIN synch field:

- Write the nominal LIN prescaler value (usually depending on the nominal baud rate) in the LPFR / LPR registers.
- Set the LASE bit to enable the auto synchronization unit.

When auto synchronization is enabled, after each LIN synch break, the time duration between five falling edges on RDI is sampled on f_{CPU} and the result of this measurement is stored in an internal 15-bit register called SM (not user accessible) (see [Figure 83](#)). Then the LDIV value (and its associated LPFR and LPR registers) are automatically updated at the end of the fifth falling edge. During LIN synch field measurement, the SCI state machine is stopped and no data is transferred to the data register.

15.9.6 LIN slave baud rate generation

In LIN mode, transmission and reception are driven by the LIN baud rate generator

Note: *LIN master mode uses the extended or conventional prescaler register to generate the baud rate.*

If LINE bit = 1 and LSLV bit = 1 then the conventional and extended baud rate generators are disabled: the baud rate for the receiver and transmitter are both set to the same value, depending on the LIN slave baud rate generator:

$$T_x = R_x = \frac{f_{\text{CPU}}}{(16 \cdot \text{LDIV})}$$

with:

LDIV is an unsigned fixed point number. The mantissa is coded on 8 bits in the LPR register and the fraction is coded on 4 bits in the LPFR register.

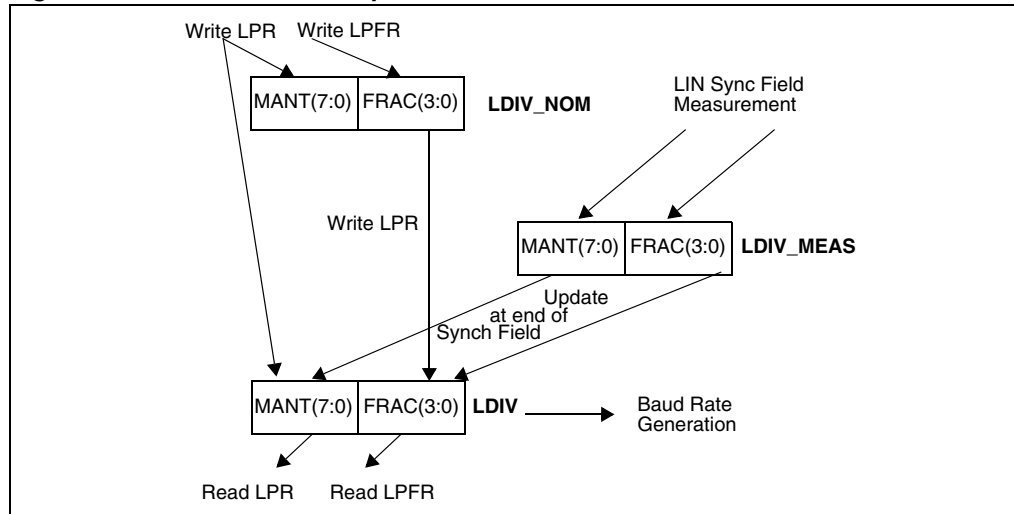
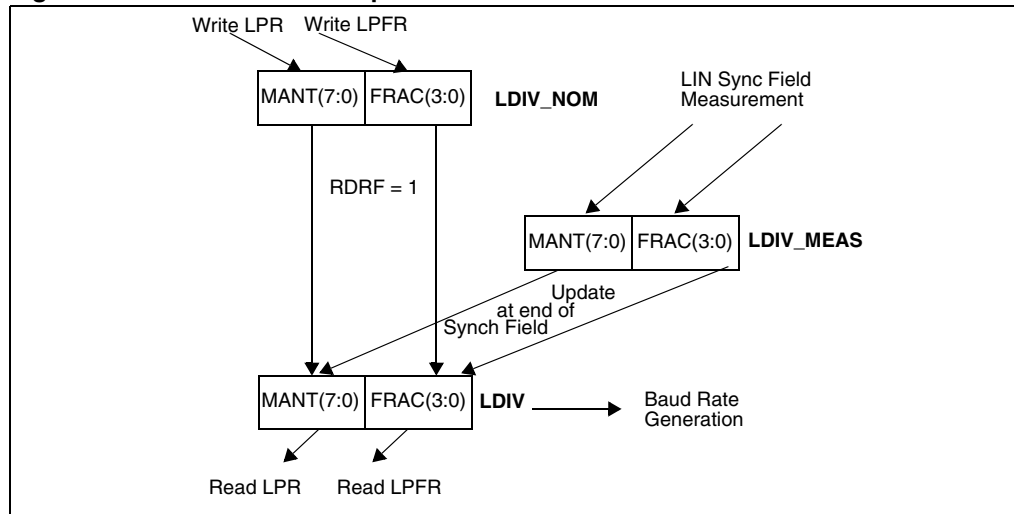
If LASE bit = 1 then LDIV is automatically updated at the end of each LIN synch field.

Three registers are used internally to manage the auto-update of the LIN divider (LDIV):

- LDIV_NOM (nominal value written by software at LPR/LPFR addresses)
- LDIV_MEAS (results of the field synch measurement)
- LDIV (used to generate the local baud rate)

The control and interactions of these registers is explained in [Figure 84](#) and [Figure 85](#). It depends on the LDUM bit setting (LIN divider update method)

Note: *As explained in [Figure 84](#) and [Figure 85](#), LDIV can be updated by two concurrent actions: a transfer from LDIV_MEAS at the end of the LIN sync field and a transfer from LDIV_NOM due to a software write of LPR. If both operations occur at the same time, the transfer from LDIV_NOM has priority.*

Figure 84. LDIV read / write operations when LDUM = 0**Figure 85. LDIV read / write operations when LDUM = 1**

15.9.7 LINSICI clock tolerance

LINSICI clock tolerance when unsynchronized

When LIN slaves are unsynchronized (meaning no characters have been transmitted for a relatively long time), the maximum tolerated deviation of the LINSICI clock is +/-15%.

If the deviation is within this range then the LIN synch break is detected properly when a new reception occurs.

This is made possible by the fact that masters send 13 low bits for the LIN synch break, which can be interpreted as 11 low bits (13 bits -15% = 11.05) by a "fast" slave and then considered as a LIN synch break. According to the LIN specification, a LIN synch break is valid when its duration is greater than $t_{\text{SBRKTS}} = 10$. This means that the LIN synch break must last at least 11 low bits.

Note: If the period desynchronization of the slave is +15% (slave too slow), the character “00h” which represents a sequence of 9 low bits must not be interpreted as a break character (9 bits + 15% = 10.35). Consequently, a valid LIN Synch break must last at least 11 low bits.

LINSCI clock tolerance when synchronized

When synchronization has been performed, following reception of a LIN synch break, the LINSCI, in LIN mode, has the same clock deviation tolerance as in SCI mode, which is explained below:

During reception, each bit is oversampled 16 times. The mean of the 8th, 9th and 10th samples is considered as the bit value.

Consequently, 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 should not exceed 3.75%.

15.9.8 Clock deviation causes

The causes which contribute to the total deviation are:

- D_{TRA}: deviation due to transmitter error.

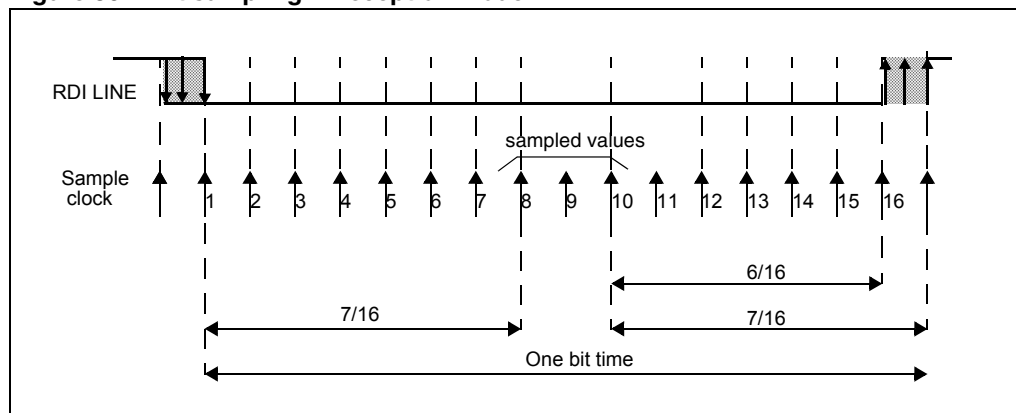
Note: The transmitter can be either a master or a slave (in case of a slave listening to the response of another slave).

- D_{MEAS}: error due to the LIN Synch measurement performed by the receiver.
- 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 LIN 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 LINSCI clock tolerance:

$$D_{TRA} + D_{MEAS} + D_{QUANT} + D_{REC} + D_{TCL} < 3.75\%$$

Figure 86. Bit sampling in reception mode



15.9.9 Error due to LIN synch measurement

The LIN synch field is measured over eight bit times.

This measurement is performed using a counter clocked by the CPU clock. The edge detections are performed using the CPU clock cycle.

This leads to a precision of 2 CPU clock cycles for the measurement which lasts $16 \cdot 8 \cdot \text{LDIV}$ clock cycles.

Consequently, this error (D_{MEAS}) is equal to:

$$2 / (128 \cdot \text{LDIV}_{\text{MIN}}).$$

LDIV_{MIN} corresponds to the minimum LIN prescaler content, leading to the maximum baud rate, taking into account the maximum deviation of +/-15%.

15.9.10 Error due to baud rate quantization

The baud rate can be adjusted in steps of $1 / (16 \cdot \text{LDIV})$. The worst case occurs when the “real” baud rate is in the middle of the step.

This leads to a quantization error (D_{QUANT}) equal to $1 / (2 \cdot 16 \cdot \text{LDIV}_{\text{MIN}})$.

15.9.11 Impact of clock deviation on maximum baud rate

The choice of the nominal baud rate (LDIV_{NOM}) will influence both the quantization error (D_{QUANT}) and the measurement error (D_{MEAS}). The worst case occurs for LDIV_{MIN} .

Consequently, at a given CPU frequency, the maximum possible nominal baud rate (LPR_{MIN}) should be chosen with respect to the maximum tolerated deviation given by the equation:

$$D_{\text{TRA}} + 2 / (128 \cdot \text{LDIV}_{\text{MIN}}) + 1 / (2 \cdot 16 \cdot \text{LDIV}_{\text{MIN}}) + D_{\text{REC}} + D_{\text{TCL}} < 3.75\%$$

Example:

A nominal baud rate of 20Kbits/s at $T_{\text{CPU}} = 125\text{ns}$ (8 MHz) leads to $\text{LDIV}_{\text{NOM}} = 25\text{d}$.

$$\text{LDIV}_{\text{MIN}} = 25 - 0.15 \cdot 25 = 21.25$$

$$D_{\text{MEAS}} = 2 / (128 \cdot \text{LDIV}_{\text{MIN}}) \cdot 100 = 0.00073\%$$

$$D_{\text{QUANT}} = 1 / (2 \cdot 16 \cdot \text{LDIV}_{\text{MIN}}) \cdot 100 = 0.0015\%$$

LIN slave systems

For LIN slave systems (the LINE and LSLV bits are set), receivers wake up by LIN synch break or LIN Identifier detection (depending on the LHDM bit).

Hot plugging feature for LIN slave nodes

In LIN slave mute mode (the LINE, LSLV and RWU bits are set) it is possible to hot plug to a network during an ongoing communication flow. In this case the SCI monitors the bus on the RDI line until 11 consecutive dominant bits have been detected and discards all the other bits received.

15.10 LIN mode register description

15.10.1 Status register (SCISR)

Read only

Reset value: 1100 0000 (C0h)

| | | | | | | | |
|------|----|------|------|-----|----|----|----|
| 7 | | | | | | | 0 |
| TDRE | TC | RDRF | IDLE | LHE | NF | FE | PE |

Bits 7:4 = same function as in SCI mode, please refer to [Section 15.8: SCI mode register description](#).

Bit 3 = LHE *LIN Header Error*.

During LIN header this bit signals three error types:

- The LIN synch field is corrupted and the SCI is blocked in LIN synch state (LSF bit = 1).
- A timeout occurred during LIN Header reception
- An overrun error was detected on one of the header field (see OR bit description in [Section 15.8: SCI mode register description](#)).

An interrupt is generated if RIE = 1 in the SCICR2 register. If blocked in the LIN synch state, the LSF bit must first be reset (to exit LIN synch field state and then to be able to clear LHE flag). Then it is cleared by the following software sequence: An access to the SCISR register followed by a read to the SCIDR register.

0: no LIN header error

1: LIN header error detected

Note: Apart from the LIN header this bit signals an overrun error as in SCI mode, (see description in [Section 15.8: SCI mode register description](#))

Bit 2 = **NF** *Noise flag*

In LIN master mode (LINE bit = 1 and LSLV bit = 0) this bit has the same function as in SCI mode, please refer to [Section 15.8: SCI mode register description](#)

In LIN slave mode (LINE bit = 1 and LSLV bit = 1) this bit has no meaning.

Bit 1 = **FE** *Framing error*.

In LIN slave mode, this bit is set only when a real framing error is detected (if the stop bit is dominant (0) and at least one of the other bits is recessive (1)). It is not set when a break occurs, the LHDF bit is used instead as a break flag (if the LHDM bit = 0). It is cleared by a software sequence (an access to the SCISR register followed by a read to the SCIDR register).

0: no Framing error

1: framing error detected

Bit 0 = **PE** *Parity error*.

This bit is set by hardware when a LIN parity error occurs (if the PCE bit is set) 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.

0: no LIN parity error

1: LIN parity error detected

15.10.2 Control Register 1 (SCICR1)

Read/ write

Reset value: x000 0000 (x0h)

| | | | | | | | |
|----|----|------|---|------|-----|----|-----|
| 7 | | | | | | | 0 |
| R8 | T8 | SCID | M | WAKE | PCE | PS | PIE |

Bits 7:3 = Same function as in SCI mode, please refer to [Section 15.8: SCI mode register description](#).

Bit 2 = **PCE** Parity control enable.

This bit is set and cleared by software. It selects the hardware parity control for LIN identifier parity check.

0: parity control disabled

1: parity control enabled

When a parity error occurs, the PE bit in the SCISR register is set.

Bit 1 = reserved

Bit 0 = same function as in SCI mode, please refer to [Section 15.8: SCI mode register description](#).

15.10.3 Control Register 2 (SCICR2)

Read/ write

Reset value: 0000 0000 (00h)

| | | | | | | | |
|-----|------|-----|------|----|----|-----|-----|
| 7 | | | | | | | 0 |
| TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |

Bits 7:2 = same function as in SCI mode, please refer to [Section 15.8: SCI mode register description](#).

Bit 1 = RWU Receiver wake-up.

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.

0: receiver in active mode

1: receiver in mute mode

Note: Mute mode is recommended for detecting only the header and avoiding the reception of any other characters. For more details please refer to [LIN reception](#).

In LIN slave mode, when RDRF is set, the software can not set or clear the RWU bit.

Bit 0 = **SBK** Send break.

This bit set is used to send break characters. It is set and cleared by software.

0: no break character is transmitted

1: break characters are transmitted

Note: If the SBK bit is set to "1" and then to "0", the transmitter will send a BREAK word at the end of the current word.

15.10.4 Control register 3 (SCICR3)

Read/ write

Reset value: 0000 0000 (00h)

| | |
|------|-----|
| 7 | 0 |
| LDUM | LSF |

Bit 7 = **LDUM** LIN Divider Update Method.

This bit is set and cleared by software and is also cleared by hardware (when RDRF = 1). It is only used in LIN Slave mode. It determines how the LIN Divider can be updated by software.

0: LDIV is updated as soon as LPR is written (if no auto synchronization update occurs at the same time).

1: LDIV is updated at the next received character (when RDRF = 1) after a write to the LPR register

Note: If no write to LPR is performed between the setting of LDUM bit and the reception of the next character, LDIV will be updated with the old value.

After LDUM has been set, it is possible to reset the LDUM bit by software. In this case, LDIV can be modified by writing into LPR / LPFR registers.

Bits 6:5 = **LINE**, **LSLV** LIN Mode Enable Bits.

These bits configure the LIN mode:

Table 66. LIN mode configuration

| LINE | LSLV | Meaning |
|------|------|-------------------|
| 0 | x | LIN mode disabled |
| 1 | 0 | LIN Master Mode |
| | 1 | LIN Slave Mode |

The LIN master configuration enables:

The capability to send LIN synch breaks (13 low bits) using the SBK bit in the SCICR2 register.

The LIN slave configuration enables:

- The LIN slave baud rate generator. The LIN Divider (LDIV) is then represented by the LPR and LPFR registers. The LPR and LPFR registers are read/write accessible at the address of the SCIBRR register and the address of the SCIETPR register
- Management of LIN headers.
- LIN synch break detection (11-bit dominant).
- LIN wake-up method (see LHDM bit) instead of the normal SCI Wake-Up method.
- Inhibition of break transmission capability (SBK has no effect)
- LIN parity checking (in conjunction with the PCE bit)

Bit 4 = **LASE** *LIN Auto Synch Enable*.

This bit enables the Auto Synch Unit (ASU). It is set and cleared by software. It is only usable in LIN Slave mode.

- 0: auto synch unit disabled
- 1: auto synch unit enabled.

Bit 3 = **LHDM** *LIN Header Detection Method*

This bit is set and cleared by software. It is only usable in LIN Slave mode. It enables the Header Detection Method. In addition if the RWU bit in the

SCICR2 register is set, the LHDM bit selects the Wake-Up method (replacing the WAKE bit).

- 0: LIN synch break detection method
- 1: LIN Identifier field detection method

Bit 2 = **LHIE** *LIN Header Interrupt Enable*

This bit is set and cleared by software. It is only usable in LIN Slave mode.

- 0: LIN header interrupt is inhibited.
- 1: An SCI interrupt is generated whenever LHDF = 1.

Bit 1 = **LHDF** *LIN Header Detection Flag*

This bit is set by hardware when a LIN Header is detected and cleared by a software sequence (an access to the SCISR register followed by a read of the SCICR3 register). It is only usable in LIN slave mode.

- 0: no LIN header detected.
- 1: LIN header detected.

Note: The header detection method depends on the LHDM bit:

- If LHDM = 0, a header is detected as a LIN synch break.
- If LHDM = 1, a header is detected as a LIN Identifier, meaning that a LIN synch break field + a LIN synch field + a LIN identifier field have been consecutively received.

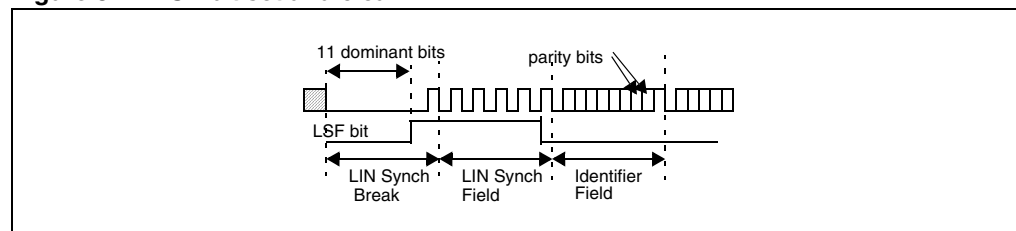
Bit 0 = **LSF** *LIN Synch Field State*

This bit indicates that the LIN synch field is being analyzed. It is only used in LIN slave mode. In auto synchronization mode (LASE bit = 1), when the SCI is in the LIN synch field State it waits or counts the falling edges on the RDI line.

It is set by hardware as soon as a LIN synch break is detected and cleared by hardware when the LIN synch field analysis is finished (see [Figure 87](#)). This bit can also be cleared by software to exit LIN synch state and return to idle mode.

- 0: the current character is not the LIN synch field
- 1: LIN synch field state (LIN synch field undergoing analysis)

Figure 87. LSF bit set and clear



15.10.5 LIN divider registers

LDIV is coded using the two registers LPR and LPFR. In LIN slave mode, the LPR register is accessible at the address of the SCIBRR register and the LPFR register is accessible at the address of the SCIETPR register.

15.10.6 LIN prescaler register (LPR)

Read/ write

Reset value: 0000 0000 (00h)

| | | | | | | | |
|------|------|------|------|------|------|------|------|
| 7 | | | | | | | 0 |
| LPR7 | LPR6 | LPR5 | LPR4 | LPR3 | LPR2 | LPR1 | LPR0 |

LPR[7:0] LIN Prescaler (mantissa of LDIV)

These 8 bits define the value of the mantissa of the LIN Divider (LDIV):

Table 67. LDIV mantissa

| LPR[7:0] | Rounded mantissa (LDIV) |
|----------|-------------------------|
| 00h | SCI clock disabled |
| 01h | 1 |
| ... | ... |
| FEh | 254 |
| FFh | 255 |

Caution: LPR and LPFR registers have different meanings when reading or writing to them. Consequently bit manipulation instructions (BRES or BSET) should never be used to modify the LPR[7:0] bits, or the LPFR[3:0] bits.

15.10.7 LIN prescaler fraction register (LPFR)

Read/ write

Reset value: 0000 0000 (00h)

| | | | | | | | |
|---|---|---|---|-------|-------|-------|-------|
| 7 | | | | | | | 0 |
| 0 | 0 | 0 | 0 | LPFR3 | LPFR2 | LPFR1 | LPFR0 |

Bits 7:4 = Reserved.

Bits 3:0 = **LPFR[3:0]** Fraction of LDIV

These 4 bits define the fraction of the LIN Divider (LDIV):

Table 68. LDIV fraction

| LPFR[3:0] | Fraction (LDIV) |
|-----------|-----------------|
| 0h | 0 |
| 1h | 1/16 |
| ... | ... |
| Eh | 14/16 |
| Fh | 15/16 |

1. When initializing LDIV, the LPFR register must be written first. Then, the write to the LPR register will effectively update LDIV and so the clock generation.
2. In LIN slave mode, if the LPR[7:0] register is equal to 00h, the transceiver and receiver input clocks are switched off.

Examples of LDIV coding:**Example 1:**

LPR = 27d and LPFR = 12d

This leads to:

Mantissa (LDIV) = 27d

Fraction (LDIV) = $12/16 = 0.75d$

Therefore LDIV = 27.75d

Example 2:

LDIV = 25.62d

This leads to:

LPFR = rounded($16 \cdot 0.62d$) = rounded(9.92d) = 10d = Ah

LPR = mantissa (25.620d) = 25d = 1Bh

Example 3:

LDIV = 25.99d

This leads to:

LPFR = rounded($16 \cdot 0.99d$) = rounded(15.84d) = 16d

The carry must be propagated to the mantissa:

LPR = mantissa (25.99) + 1 = 26d = 1Ch

15.10.8 LIN header length register (LHLR)

Read only

Reset value: 0000 0000 (00h)

| | | | | | | | |
|------|------|------|------|------|------|------|------|
| 7 | | | | | | | 0 |
| LHL7 | LHL6 | LHL5 | LHL4 | LHL3 | LHL2 | LHL1 | LHL0 |

Note: In LIN slave mode when $LASE = 1$ or $LHDM = 1$, the LHLR register is accessible at the address of the SCIERPR register.

Otherwise this register is always read as 00h.

Bits 7:0 = **LHL[7:0]** LIN Header Length.

- This is a read-only register, which is updated by hardware if one of the following conditions occurs:
 - After each break detection, it is loaded with “FFh”.
 - If a timeout occurs on T_{HEADER} , it is loaded with 00h.
 - After every successful LIN Header reception (at the same time than the setting of LHDF bit), it is loaded with a value (LHL) which gives access to the number of bit times of the LIN header length (T_{HEADER}). The coding of this value is explained below:

LHL coding

$T_{HEADER_MAX} = 57$

LHL(7:2) represents the mantissa of $(57 - T_{HEADER})$

LHL(1:0) represents the fraction $(57 - T_{HEADER})$

Table 69. LHL mantissa coding

| LHL[7:2] | Mantissa ($57 - T_{HEADER}$) | Mantissa (T_{HEADER}) |
|----------|--------------------------------|---------------------------|
| 0h | 0 | 57 |
| 1h | 1 | 56 |
| ... | ... | ... |
| 39h | 56 | 1 |
| 3Ah | 57 | 0 |
| 3Bh | 58 | Never Occurs |
| ... | ... | ... |
| 3Eh | 62 | Never Occurs |
| 3Fh | 63 | Initial value |

Table 70. LHL fraction coding

| LHL[1:0] | Fraction ($57 - T_{\text{HEADER}}$) |
|----------|---------------------------------------|
| 0h | 0 |
| 1h | 1/4 |
| 2h | 1/2 |
| 3h | 3/4 |

Example of LHL coding**Example 1:**

$$\text{LHL} = 33\text{h} = 001100\ 11\text{b}$$

$$\text{LHL}(7:3) = 1100\text{b} = 12\text{d}$$

$$\text{LHL}(1:0) = 11\text{b} = 3\text{d}$$

This leads to:

$$\text{Mantissa } (57 - T_{\text{HEADER}}) = 12\text{d}$$

$$\text{Fraction } (57 - T_{\text{HEADER}}) = 3/4 = 0.75$$

Therefore:

$$(57 - T_{\text{HEADER}}) = 12.75\text{d}$$

$$\text{and } T_{\text{HEADER}} = 44.25\text{d}$$

Example 2:

$$57 - T_{\text{HEADER}} = 36.21\text{d}$$

$$\text{LHL}(1:0) = \text{rounded}(4 \cdot 0.21\text{d}) = 1\text{d}$$

$$\text{LHL}(7:2) = \text{Mantissa } (36.21\text{d}) = 36\text{d} = 24\text{h}$$

$$\text{Therefore LHL}(7:0) = 10010001 = 91\text{h}$$

Example 3:

$$57 - T_{\text{HEADER}} = 36.90\text{d}$$

$$\text{LHL}(1:0) = \text{rounded}(4 \cdot 0.90\text{d}) = 4\text{d}$$

The carry must be propagated to the mantissa:

$$\text{LHL}(7:2) = \text{Mantissa } (36.90\text{d}) + 1 = 37\text{d} =$$

$$\text{Therefore LHL}(7:0) = 10110000 = \text{A0h}$$

Table 71. LINSCI1 register map and reset values

| Addr. (Hex.) | Register name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----------------|--|--------------------|--------------------|--------------------|--------------------|---------------------|---------------------|---------------------|---------------------|
| 48 | SCI1SR Reset value | TDRE 1 | TC 1 | RDRF 0 | IDLE 0 | OR/LHE 0 | NF 0 | FE 0 | PE 0 |
| 49 | SCI1DR Reset value | DR7 - | DR6 - | DR5 - | DR4 - | DR3 - | DR2 - | DR1 - | DR0 - |
| 4A | SCI1BRR LPR (LIN Slave Mode) Reset value | SCP1 LPR7 0 | SCP0 LPR6 0 | SCT2 LPR5 0 | SCT1 LPR4 0 | SCT0 LPR3 0 | SCR2 LPR2 0 | SCR1 LPR1 0 | SCR0 LPR0 0 |
| 4B | SCI1CR1 Reset value | R8 x | T8 0 | SCID 0 | M 0 | WAKE 0 | PCE 0 | PS 0 | PIE 0 |
| 4C | SCI1CR2 Reset value | TIE 0 | TCIE 0 | RIE 0 | ILIE 0 | TE 0 | RE 0 | RWU 0 | SBK 0 |
| 4D | SCI1CR3 Reset value | LDUM 0 | LINE 0 | LSLV 0 | LASE 0 | LHDM 0 | LHIE 0 | LHDF 0 | LSF 0 |
| 4E | SCI1ERPR LHLR (LIN Slave Mode) Reset value | ERPR7 LHL7 0 | ERPR6 LHL6 0 | ERPR5 LHL5 0 | ERPR4 LHL4 0 | ERPR3 LHL3 0 | ERPR2 LHL2 0 | ERPR1 LHL1 0 | ERPR0 LHL0 0 |
| 4F | SCI1ETPR LPFR (LIN Slave Mode) Reset value | ETPR7 0 0 | ETPR6 0 0 | ETPR5 0 0 | ETPR4 0 0 | ETPR3 LPFR3 0 | ETPR2 LPFR2 0 | ETPR1 LPFR1 0 | ETPR0 LPFR0 0 |

16 LINSCI serial communication interface (LIN master only)

16.1 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.

16.2 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
- Transmitter clock output
- Parity control:
 - Transmits parity bit
 - Checks parity of received data byte
- Reduced power consumption mode
- LIN synch break send capability

16.3 General description

The interface is externally connected to another device by three pins (see [Figure 88: SCI block diagram](#)). Any SCI bidirectional communication requires a minimum of two pins: Receive Data In (RDI) and Transmit Data Out (TDO):

- SCLK: Transmitter Clock Output. This pin outputs the transmitter data clock for synchronous transmission (no clock pulses on start bit and stop bit, and a software option to send a clock pulse on the last data bit). This can be used to control peripherals that have shift registers (e.g. LCD drivers). The clock phase and polarity are software programmable.
- TDO: Transmit Data Output. When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is 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.

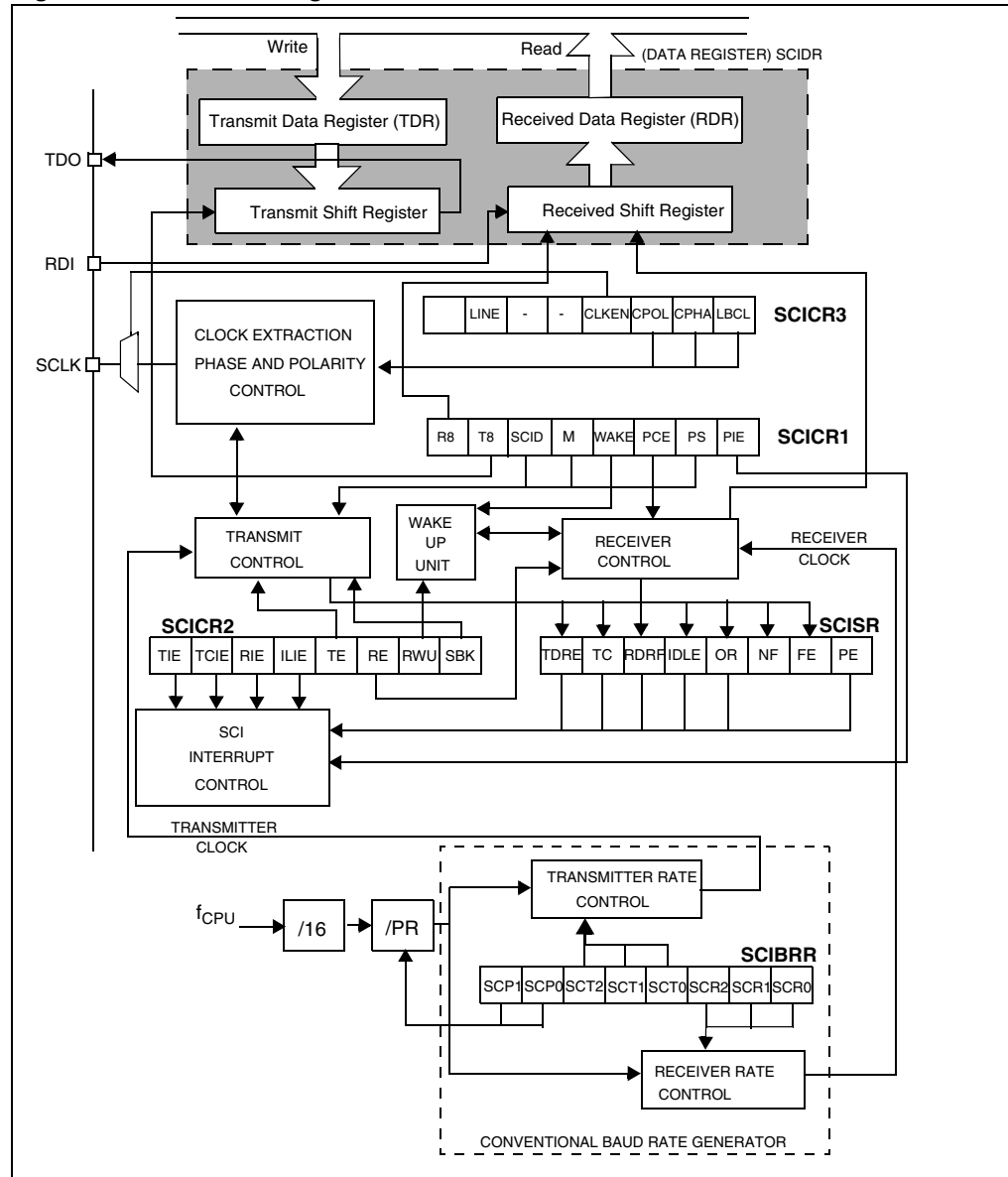
Through these pins, serial data is transmitted and received as frames comprising:

- 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.

Figure 88. SCI block diagram



16.4 Functional description

The block diagram of the serial control interface, is shown in [Figure 88](#). It contains seven dedicated registers:

- Three control registers (SCICR1, SCICR2 and SCICR3)
- A status register (SCISR)
- A baud rate register (SCIBRR)
- An extended prescaler receiver register (SCIERP)
- An extended prescaler transmitter register (SCIETPR)

Refer to the register descriptions in [Section 15.8: SCI mode register description](#) for the definitions of each bit.

16.4.1 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 89](#)).

The TDO pin is in low state during the start bit.

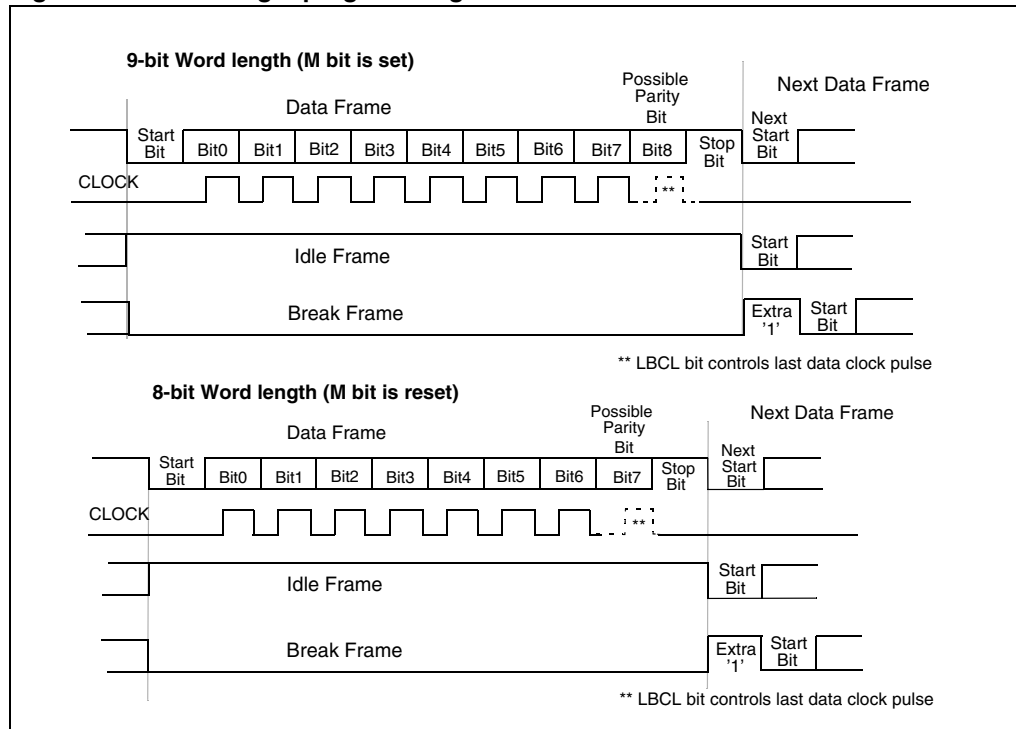
The TDO pin is in high state during the stop bit.

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 89. Word length programming



16.4.2 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.

When the transmit enable bit (TE) is set, the data in the transmit shift register is output on the TDO pin and the corresponding clock pulses are output on the SCLK pin.

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 89](#)).

Procedure

- Select the M bit to define the word length.
- Select the desired baud rate using the SCIBRR and the SCIETPR registers.
- Set the TE bit to send an idle frame as first transmission.
- 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 or after the break frame) 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 89](#)).

As long as the SBK bit is set, the SCI send break frames to the TDO pin. After clearing this 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.*

LIN transmission

The same procedure has to be applied for LIN master transmission with the following differences:

- Clear the M bit to configure 8-bit word length.
- Set the LINE bit to enter LIN master mode. In this case, setting the SBK bit sends 13 low bits.

16.4.3 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 of a buffer (RDR) between the internal bus and the received shift register (see [Figure 88](#)).

Procedure

- Select the M bit to define the word length.
- Select the desired baud rate using the SCIBRR and the SCIERPR registers.
- 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 until the RDRF bit is cleared.

When a 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.

When noise is detected in a frame:

- The NF 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 bit is reset by a SCISR register read operation followed by a SCIDR register read operation.

Framing error

A framing error is detected when:

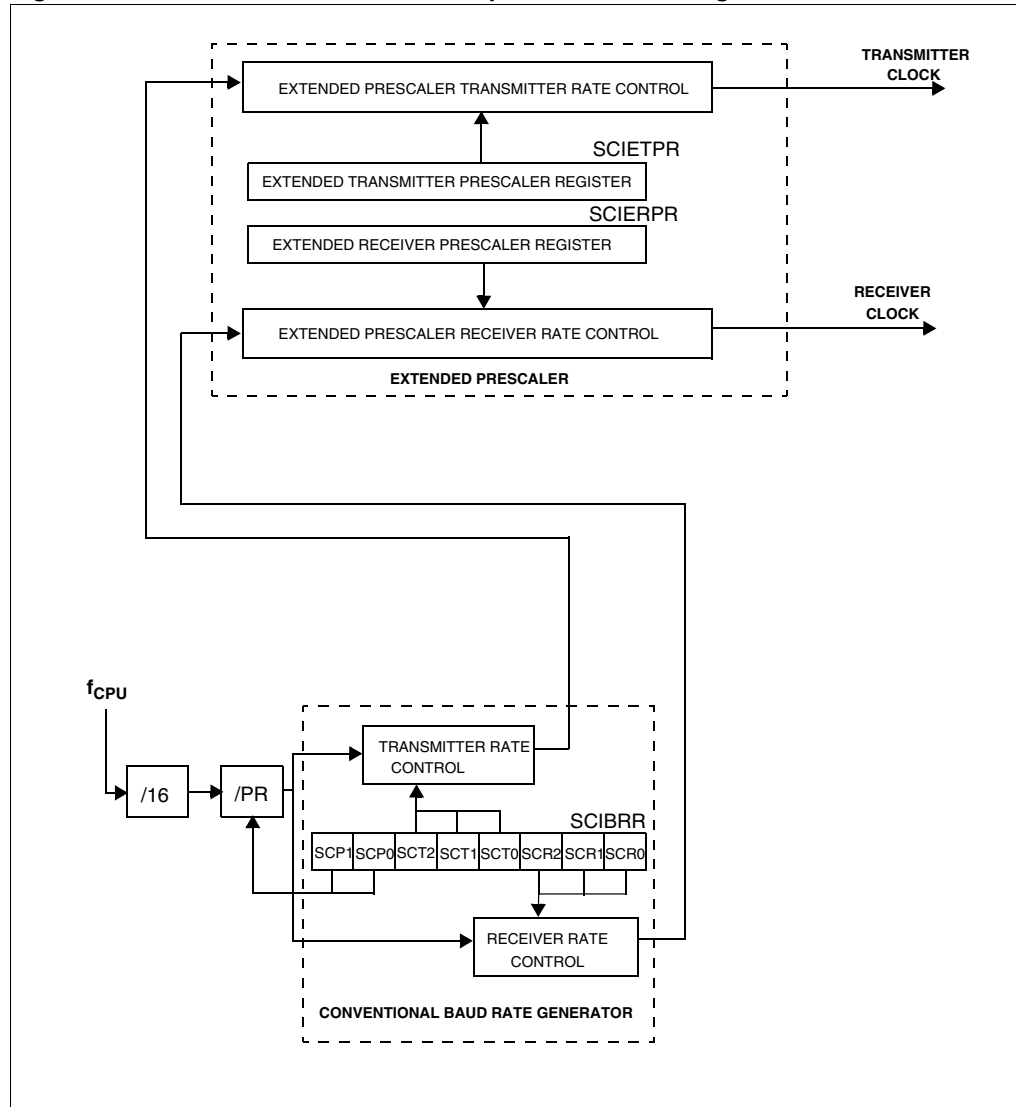
- The stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.
- A break is received.

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.

Figure 90. SCI baud rate and extended prescaler block diagram



16.4.4 Conventional baud rate generation

The baud rates for the receiver and transmitter (Rx and Tx) are set independently and calculated as follows

$$Tx = \frac{f_{CPU}}{(16 \cdot PR) \cdot TR} \quad Rx = \frac{f_{CPU}}{(16 \cdot PR) \cdot RR}$$

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 1: 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.

16.4.5 Extended baud rate generation

The extended prescaler option gives a very fine tuning on 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 90](#).

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:

$$T_x = \frac{f_{\text{CPU}}}{16 \cdot \text{ETPR} \cdot (\text{PR} \cdot \text{TR})} \quad R_x = \frac{f_{\text{CPU}}}{16 \cdot \text{ERPR} \cdot (\text{PR} \cdot \text{RR})}$$

with:

ETPR = 1...255 (see SCIETPR register)

ERPR = 1...255 (see SCIERPR register)

16.4.6 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.

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.

16.4.7 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 72](#).

Table 72. Frame formats

| M bit | PCE bit | SCI frame ⁽¹⁾ |
|-------|---------|----------------------------|
| 0 | 0 | SB 8 bit data STB |
| | 1 | SB 7-bit data PB STB |
| 1 | 0 | SB 9-bit data STB |
| | 1 | SB 8-bit data PB STB |

1. 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 “1s” 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 1:

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 “1s” 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 “1s” if even parity is selected (PS = 0) or an odd number of “1s” 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.

16.5 Low power modes

Table 73. Effect of low power modes on SCI

| Mode | Description |
|------|--|
| WAIT | No effect on SCI. SCI interrupts cause the device to exit from Wait mode. |
| HALT | SCI registers are frozen. In halt mode, the SCI stops transmitting/receiving until Halt mode is exited. |

16.6 Interrupts

Table 74. SCI interrupt control and 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 | | |
| Received data ready to be read | RDRF | RIE | | |
| Overrun error detected | OR | | | |
| Idle line detected | IDLE | ILIE | | |
| Parity error | PE | PIE | | |

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).

16.7 SCI synchronous transmission

The SCI transmitter allows the user to control a one way synchronous serial transmission. The SCLK pin is the output of the SCI transmitter clock. No clock pulses are sent to the SCLK pin during start bit and stop bit. Depending on the state of the LBCL bit in the SCICR3 register, clock pulses are or are not be generated during the last valid data bit (address mark). The CPOL bit in the SCICR3 register allows the user to select the clock polarity, and the CPHA bit in the SCICR3 register allows the user to select the phase of the external clock (see [Figure 91](#), [Figure 92](#) and [Figure 93](#)).

During idle, preamble and send break, the external SCLK clock is not activated.

These options allow the user to serially control peripherals which consist of shift registers, without losing any functions of the SCI transmitter which can still talk to other SCI receivers. These options do not affect the SCI receiver which is independent from the transmitter.

Note: The SCLK pin works in conjunction with the TDO pin. When the SCI transmitter is disabled (TE and RE = 0), the SCLK and TDO pins go into high impedance state.
 The LBCL, CPOL and CPHA bits have to be selected before enabling the transmitter to ensure that the clock pulses function correctly. These bits should not be changed while the transmitter is enabled.

Figure 91. SCI example of synchronous and asynchronous transmission

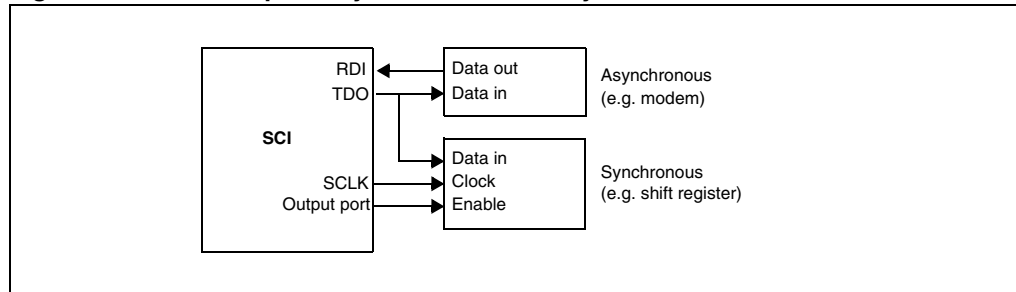


Figure 92. SCI data clock timing diagram (M = 0)

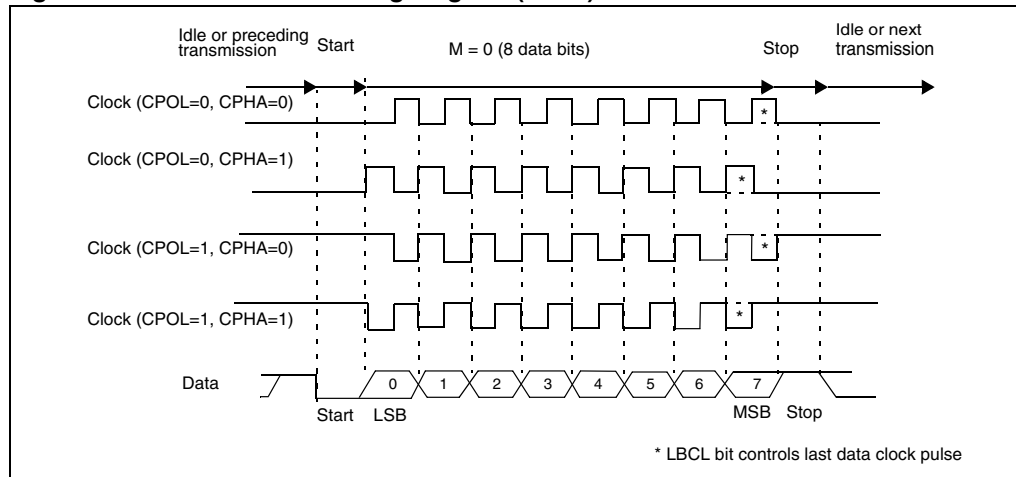
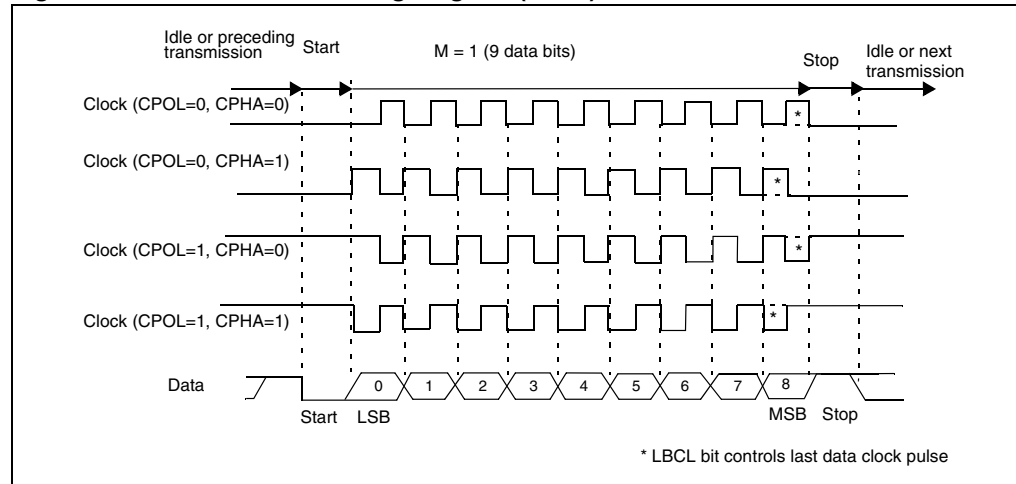


Figure 93. SCI data clock timing diagram (M = 1)

16.8 Register description

16.8.1 Status register (SCISR)

Read only

Reset value: 1100 0000 (C0h)

| | | | | | | | |
|------|----|------|------|----|----|----|----|
| 7 | | | | | | | 0 |
| TDRE | TC | RDRF | IDLE | OR | NF | FE | PE |

Bit 7 = TDRE *Transmit data register empty*.

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).

- 0: data is not transferred to the shift register
- 1: data is transferred to the shift register

Note: Data is not be transferred to the shift register until the TDRE bit is cleared.

Bit 6 = TC *Transmission complete*.

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).

- 0: transmission is not complete
- 1: transmission is complete

Note: TC is not set after the transmission of a Preamble or a Break.

Bit 5 = RDRF *Received data ready flag*.

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).

- 0: data is not received
- 1: received data is ready to be read

Bit 4 = IDLE *Idle line detect*.

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).

- 0: no idle line is detected
- 1: idle line is detected

Note: The IDLE bit is not be set again until the RDRF bit has been set itself (that is, a new idle line occurs).

Bit 3 = OR *Overrun error*.

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).

- 0: no overrun error
- 1: overrun error is detected

Note: When this bit is set, the RDR register content is not lost but the shift register is overwritten.

Bit 2 = NF *Noise flag*.

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).

- 0: no noise is detected
- 1: noise is detected

Note: This bit does not generate interrupt as it appears at the same time as the RDRF bit which itself generates an interrupt.

Bit 1 = FE *Framing error*.

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).

- 0: no framing error is detected
- 1: framing error or break character is detected

Note: This bit does not generate an 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 is transferred and only the OR bit is set.

Bit 0 = PE *Parity error*.

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.

- 0: no parity error
- 1: parity error

16.8.2 Control register 1 (SCICR1)

Read/ write

Reset value: x000 0000 (x0h)

| | | | | | | | |
|----|----|------|---|------|-----|----|-----|
| 7 | | | | | | | 0 |
| R8 | T8 | SCID | M | WAKE | PCE | PS | PIE |

Bit 7 = **R8** *Receive data bit 8.*

This bit is used to store the 9th bit of the received word when M = 1.

Bit 6 = **T8** *Transmit data bit 8.*

This bit is used to store the 9th bit of the transmitted word when M = 1.

Bit 5 = **SCID** *Disabled for low power consumption*

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.

0: SCI enabled

1: SCI prescaler and outputs disabled

Bit 4 = **M** *Word length.*

This bit determines the word length. It is set or cleared by software.

0: 1 start bit, 8 data bits, 1 stop bit

1: 1 start bit, 9 data bits, 1 stop bit

Note: The M bit must not be modified during a data transfer (both transmission and reception).

Bit 3 = **WAKE** *Wake-Up method.*

This bit determines the SCI Wake-Up method, it is set or cleared by software.

0: idle line

1: address mark

Bit 2 = **PCE** *Parity control enable.*

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).

0: parity control disabled

1: parity control enabled

Bit 1 = **PS** *Parity selection.*

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.

0: even parity

1: odd parity

Bit 0 = **PIE** *Parity interrupt enable.*

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.

0: parity error interrupt disabled

1: parity error interrupt enabled

16.8.3 Control register 2 (SCICR2)

Read/ write

Reset value: 0000 0000 (00h)

| | | | | | | | |
|-----|------|-----|------|----|----|-----|-----|
| 7 | | | | | | | 0 |
| TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |

Bit 7 = **TIE** *Transmitter interrupt enable.*

This bit is set and cleared by software.

0: interrupt is inhibited

1: an SCI interrupt is generated whenever TDRE = 1 in the SCISR register

Bit 6 = **TCIE** *Transmission complete interrupt enable*

This bit is set and cleared by software.

0: interrupt is inhibited

1: an SCI interrupt is generated whenever TC = 1 in the SCISR register

Bit 5 = **RIE** *Receiver interrupt enable.*

This bit is set and cleared by software.

0: interrupt is inhibited

1: an SCI interrupt is generated whenever OR = 1 or RDRF = 1 in the SCISR register

Bit 4 = **ILIE** *Idle line interrupt enable.*

This bit is set and cleared by software.

0: interrupt is inhibited

1: an SCI interrupt is generated whenever IDLE = 1 in the SCISR register.

Bit 3 = **TE** *Transmitter enable.*

This bit enables the transmitter. It is set and cleared by software.

0: transmitter is disabled

1: transmitter is enabled

Note: During transmission, a "0" pulse on the TE bit ("0" followed by "1") sends a preamble (idle line) after the current word.

When TE is set there is a 1 bit-time delay before the transmission starts.

Bit 2 = **RE** *Receiver enable.*

This bit enables the receiver. It is set and cleared by software.

0: receiver is disabled

1: receiver is enabled and begins searching for a start bit

Bit 1 = **RWU** *Receiver wake-up.*

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.

0: receiver in active mode

1: receiver in mute mode

Note: Before selecting Mute mode (by setting the RWU bit) the SCI must first receive a data byte, otherwise it cannot function in Mute mode with wakeup by Idle line detection.

In Address Mark Detection Wake-Up configuration (WAKE bit = 1) the RWU bit cannot be modified by software while the RDRF bit is set.

Bit 0 = **SBK** *Send break*.

This bit set is used to send break characters. It is set and cleared by software.

- 0: no break character is transmitted
- 1: break characters are transmitted

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.

16.8.4 Control Register 3 (SCICR3)

Read/ write

Reset value: 0000 0000 (00h)

| | | | | | | | | | |
|---|---|------|---|---|-------|------|------|------|---|
| 7 | - | LINE | - | - | CLKEN | CPOL | CPHA | LBCL | 0 |
|---|---|------|---|---|-------|------|------|------|---|

Bit 7 = Reserved, must be kept cleared.

Bit 6 = **LINE** *LIN Mode Enable*.

This bit is set and cleared by software.

- 0: LIN mode disabled
- 1: LIN master mode enabled

The LIN Master mode enables the capability to send LIN Synch Breaks (13 low bits) using the SBK bit in the SCICR2 register.

In transmission, the LIN synch break low phase duration is shown as below:

Table 75. LIN sync break duration

| LINE | M | Number of low bits sent during a LIN synch break |
|------|---|--|
| 0 | 0 | 10 |
| | 1 | 11 |
| 1 | 0 | 13 |
| | 1 | 14 |

Bits 5:4 = Reserved, forced by hardware to 0.
These bits are not used.

Bit 3 = **CLKEN** *Clock Enable*.

This bit allows the user to enable the SCLK pin.

- 0: SLK pin disabled
- 1: SLK pin enabled

Bit 2 = **CPOL** *Clock Polarity*.

This bit allows the user to select the polarity of the clock output on the SCLK pin. It works in conjunction with the CPHA bit to produce the desired clock/data relationship (see [Figure 92](#) and [Figure 93](#)).

- 0: steady low value on SCLK pin outside transmission window.
- 1: steady high value on SCLK pin outside transmission window.

Bit 1 = **CPHA** *Clock Phase*.

This bit allows the user to select the phase of the clock output on the SCLK pin. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see [Figure 92](#) and [Figure 93](#))

- 0: SCLK clock line activated in middle of data bit.
- 1: SCLK clock line activated at beginning of data bit.

Bit 0 = **LBCL** *Last bit clock pulse*.

This bit allows the user to select whether the clock pulse associated with the last data bit transmitted (MSB) has to be output on the SCLK pin.

- 0: the clock pulse of the last data bit is not output to the SCLK pin.
- 1: the clock pulse of the last data bit is output to the SCLK pin.

Note: The last bit is the 8th or 9th data bit transmitted depending on the 8 or 9 bit format selected by the M bit in the SCICR1 register.

Table 76. SCI clock on SCLK pin

| Data format | M bit | LBCL bit | Number of clock pulses on SCLK |
|-------------|-------|----------|--------------------------------|
| 8 bit | 0 | 0 | 7 |
| | | 1 | 8 |
| 9 bit | 1 | 0 | 8 |
| | | 1 | 9 |

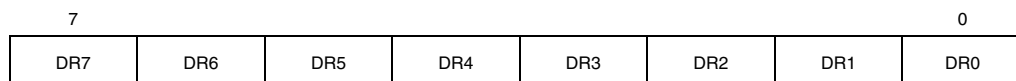
Note: These 3 bits (**CPOL**, **CPHA**, **LBCL**) should not be written while the transmitter is enabled.

16.8.5 Data register (SCIDR)

Read/ write

Reset value: Undefined

Contains the received or transmitted data character, depending on whether it is read from or written to.



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 88](#)).

The RDR register provides the parallel interface between the input shift register and the internal bus (see [Figure 88](#)).

16.8.6 Baud rate register (SCIBRR)

Read/ write

Reset value: 0000 0000 (00h)

| | | | | | | | |
|------|------|------|------|------|------|------|------|
| 7 | | | | | | | 0 |
| SCP1 | SCP0 | SCT2 | SCT1 | SCT0 | SCR2 | SCR1 | SCR0 |

Bits 7:6 = **SCP[1:0]** *First SCI Prescaler*

These 2 prescaling bits allow several standard clock division ranges:

Table 77. PR prescaler

| PR prescaling factor | SCP1 | SCP0 |
|----------------------|------|------|
| 1 | 0 | 0 |
| 3 | | 1 |
| 4 | 1 | 0 |
| 13 | | 1 |

Bits 5:3 = **SCT[2:0]** *SCI Transmitter rate divisor*

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.

Table 78. Transmitter rate divider

| TR dividing factor | SCT2 | SCT1 | SCT0 |
|--------------------|------|------|------|
| 1 | 0 | 0 | 0 |
| 2 | | | 1 |
| 4 | | 1 | 0 |
| 8 | | | 1 |
| 16 | 1 | 0 | 0 |
| 32 | | | 1 |
| 64 | | 1 | 0 |
| 128 | | | 1 |

Note: This TR factor is used only when the ETPR fine tuning factor is equal to 00h; otherwise, TR is replaced by the (TR*ETPR) dividing factor.

Bits 2:0 = **SCR[2:0]** *SCI Receiver rate divisor.*

These 3 bits, in conjunction with the SCP1 and SCP0 bits define the total division applied to the bus clock to yield the receive rate clock in conventional baud rate generator mode.

Table 79. Receiver rate divider

| RR dividing factor | SCR2 | SCR1 | SCR0 |
|--------------------|------|------|------|
| 1 | 0 | 0 | 0 |
| 2 | | | 1 |
| 4 | | 1 | 0 |
| 8 | | | 1 |
| 16 | 1 | 0 | 0 |
| 32 | | | 1 |
| 64 | | 1 | 0 |
| 128 | | | 1 |

Note: This RR factor is used only when the ERPR fine tuning factor is equal to 00h; otherwise, RR is replaced by the (RR*ERPR) dividing factor.

16.8.7 Extended receive prescaler division register (SCIERP)

Read/ write

Reset value: 0000 0000 (00h)

| | | | | | | | |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 7 | | | | | | | 0 |
| ERPR7 | ERPR6 | ERPR5 | ERPR4 | ERPR3 | ERPR2 | ERPR1 | ERPR0 |

Bits 7:0 = **ERPR[7:0]** 8-bit Extended Receive Prescaler Register.

The extended baud rate generator is activated when a value other than 00h is stored in this register. The clock frequency from the 16 divider (see [Figure 90](#)) is divided by the binary factor set in the SCIERP register (in the range 1 to 255).

The extended baud rate generator is not active after a reset.

16.8.8 Extended transmit prescaler division register (SCIETPR)

Read/ write

Reset value: 0000 0000 (00h)

| | | | | | | | |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 7 | | | | | | | 0 |
| ETPR7 | ETPR6 | ETPR5 | ETPR4 | ETPR3 | ETPR2 | ETPR1 | ETPR0 |

Bits 7:0 = **ETPR[7:0]** 8-bit Extended Transmit Prescaler Register.

The extended baud rate generator is activated when a value other than 00h is stored in this register. The clock frequency from the 16 divider (see [Figure 90](#)) is divided by the binary factor set in the SCIETPR register (in the range 1 to 255).

The extended baud rate generator is not active after a reset.

Table 80. Baud rate selection

| Symbol | Parameter | Conditions | | | Standard | Baud rate | Unit |
|------------------------------------|-------------------------|------------------|-----------------------|---|--|--|------|
| | | f _{CPU} | Accuracy vs. standard | Prescaler | | | |
| f _{Tx} f _{Rx} | Communication frequency | 8 MHz | ~0.16% | Conventional Mode TR (or RR) = 128, PR = 13 TR (or RR) = 32, PR = 13 TR (or RR) = 16, PR = 13 TR (or RR) = 8, PR = 13 TR (or RR) = 4, PR = 13 TR (or RR) = 16, PR = 3 TR (or RR) = 2, PR = 13 TR (or RR) = 1, PR = 13 | 300 1200 2400 4800 9600 10400 19200 38400 | ~300.48 ~1201.92 ~2403.84 ~4807.69 ~9615.38 ~10416.67 ~19230.77 ~38461.54 | Hz |
| | | | ~0.79% | Extended Mode ETPR (or ERPR) = 35, TR (or RR) = 1, PR = 1 | 14400 | ~14285.71 | |

Table 81. LINSICI2 register map and reset values

| Address (Hex.) | Register name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------------|-------------------------|------------|------------|------------|------------|------------|------------|------------|------------|
| 60 | SCI2SR Reset value | TDRE 1 | TC 1 | RDRF 0 | IDLE 0 | OR 0 | NF 0 | FE 0 | PE 0 |
| 61 | SCI2DR Reset value | DR7 - | DR6 - | DR5 - | DR4 - | DR3 - | DR2 - | DR1 - | DR0 - |
| 62 | SCI2BRR Reset value | SCP1 0 | SCP0 0 | SCT2 0 | SCT1 0 | SCT0 0 | SCR2 0 | SCR1 0 | SCR0 0 |
| 63 | SCI2CR1 Reset value | R8 - | T8 - | SCID - | M - | WAKE - | PCE | PS | PIE |
| 64 | SCI2CR2 Reset value | TIE 0 | TCIE 0 | RIE 0 | ILIE 0 | TE 0 | RE 0 | RWU 0 | SBK 0 |
| 65 | SCI2CR3 Reset value | | LINE 0 | - 0 | - 0 | CLKEN 0 | CPOL 0 | CPHA 0 | LBCL 0 |
| 66 | SCI2ERPR Reset value | ERPR7 0 | ERPR6 0 | ERPR5 0 | ERPR4 0 | ERPR3 0 | ERPR2 0 | ERPR1 0 | ERPR0 0 |
| 67 | SCI2ETPR Reset value | ETPR7 0 | ETPR6 0 | ETPR5 0 | ETPR4 0 | ETPR3 0 | ETPR2 0 | ETPR1 0 | ETPR0 0 |

17 beCAN controller (beCAN)

The beCAN controller (Basic Enhanced CAN), interfaces the CAN network and supports the CAN protocol version 2.0A and B. It has been designed to manage high number of incoming messages efficiently with a minimum CPU load. It also meets the priority requirements for transmit messages.

17.1 Main features

- Supports CAN protocol version 2.0 A, B Active
- Bit rates up to 1Mbit/s

Transmission

- 2 transmit mailboxes
- Configurable **transmit priority**

Reception

- 1 receive FIFO with three stages
- 6 scalable filter banks
- Identifier list feature
- Configurable FIFO overrun

Management

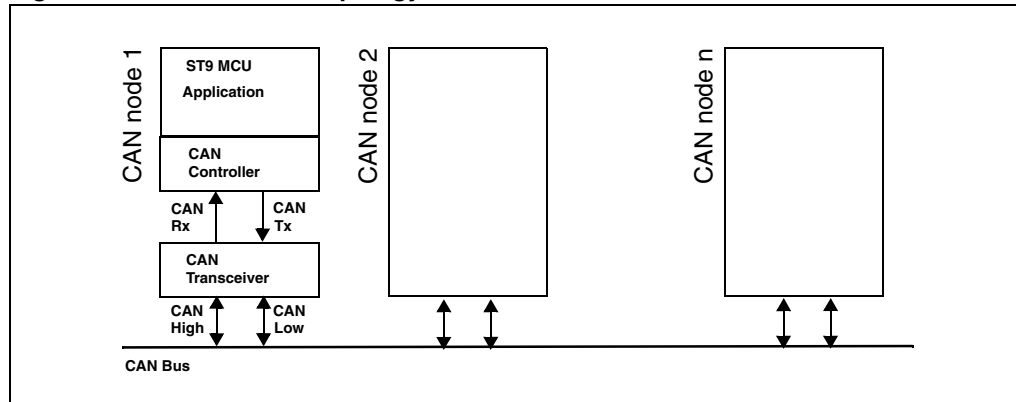
- Maskable interrupts
- Software-efficient mailbox mapping at a unique address space

17.2 General description

In today's CAN applications, the number of nodes in a network is increasing and often several networks are linked together via gateways. Typically the number of messages in the system (and thus to be handled by each node) has significantly increased. In addition to the application messages, network management and diagnostic messages have been introduced.

- An enhanced filtering mechanism is required to handle each type of message. Furthermore, application tasks require more CPU time, therefore real-time constraints caused by message reception have to be reduced.
- A receive FIFO scheme allows the CPU to be dedicated to application tasks for a long time period without losing messages.
The standard HLP (Higher Layer Protocol) based on standard CAN drivers requires an efficient interface to the CAN controller.
- All mailboxes and registers are organized in 16-byte pages mapped at the same address and selected via a page select register.

Figure 94. CAN network topology



CAN 2.0B active core

The beCAN module handles the transmission and the reception of CAN messages fully autonomously. Standard identifiers (11-bit) and extended identifiers (29-bit) are fully supported by hardware.

Control, status and configuration registers

The application uses these registers to:

- Configure CAN parameters, e.g. baud rate
- Request transmissions
- Handle receptions
- Manage interrupts
- Get diagnostic information

Tx mailboxes

Two transmit mailboxes are provided to the software for setting up messages. The Transmission Scheduler decides which mailbox has to be transmitted first.

Acceptance filters

The beCAN provides six scalable/configurable identifier filter banks for selecting the incoming messages the software needs and discarding the others.

Receive FIFO

The receive FIFO is used by the CAN controller to store the incoming messages. Three complete messages can be stored in the FIFO. The software always accesses the next available message at the same address. The FIFO is managed completely by hardware.

Figure 95. CAN block diagram

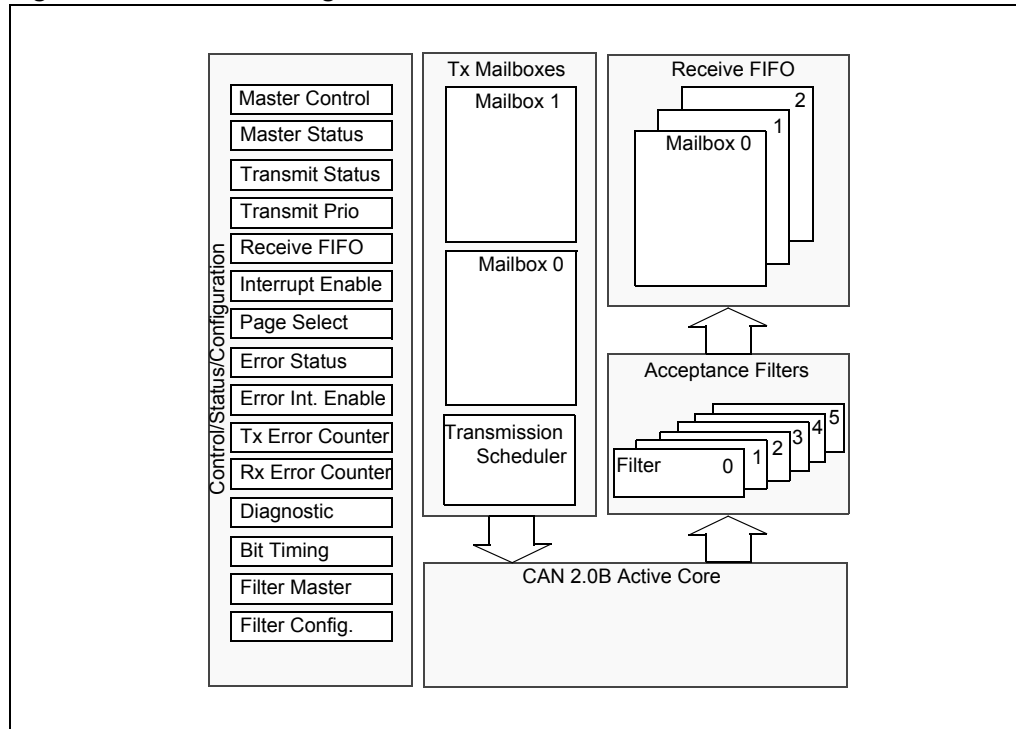
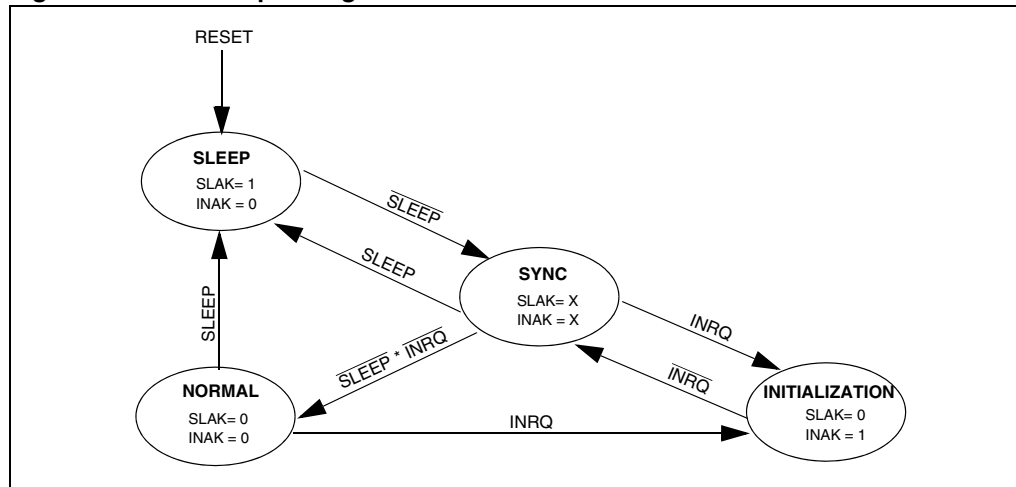


Figure 96. beCAN operating modes



17.3 Operating modes

The beCAN has three main operating modes: Initialization, Normal and Sleep. After a hardware reset, beCAN is in Sleep mode to reduce power consumption. The software requests beCAN to enter Initialization or Sleep mode by setting the INRQ or SLEEP bits in the CMCR register. Once the mode has been entered, beCAN confirms it by setting the INAK or SLAK bits in the CMSR register. When neither INAK nor SLAK are set, beCAN is in

Normal mode. Before entering Normal mode beCAN always has to synchronize on the CAN bus. To synchronize, beCAN waits until the CAN bus is idle, this means 11 consecutive recessive bits have been monitored on CANRX.

Initialization mode

The software initialization can be done while the hardware is in Initialization mode. To enter this mode the software sets the INRQ bit in the CMCR register and waits until the hardware has confirmed the request by setting the INAK bit in the CMSR register.

To leave initialization mode, the software clears the INQR bit. beCAN has left initialization mode once the INAK bit has been cleared by hardware.

While in Initialization mode, all message transfers to and from the CAN bus are stopped and the status of the CAN bus output CANTX is recessive (high).

Entering initialization mode does not change any of the configuration registers.

To initialize the CAN controller, software has to set up the bit timing registers and the filter banks. If a filter bank is not used, it is recommended to leave it non active (leave the corresponding FACT bit cleared).

Normal mode

Once the initialization has been done, the software must request the hardware to enter Normal mode, to synchronize on the CAN bus and start reception and transmission. Entering normal mode is done by clearing the INRQ bit in the CMCR register and waiting until the hardware has confirmed the request by clearing the INAK bit in the CMSR register. Afterwards, the beCAN synchronizes with the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (= bus idle) before it can take part in bus activities and start message transfer.

The initialization of the filter values is independent from Initialization mode but must be done while the filter bank is not active (corresponding FACTx bit cleared). The filter bank scale and mode configuration must be configured in initialization mode.

Low power mode (Sleep)

To reduce power consumption, beCAN has a low power mode called Sleep mode. This mode is entered on software request by setting the SLEEP bit in the CMCR register. In this mode, the beCAN clock is stopped. Consequently, software can still access the beCAN registers and mailboxes but the beCAN will not update the status bits.

Example 1: If software requests entry to initialization mode by setting the INRQ bit while beCAN is in sleep mode, it will not be acknowledged by the hardware, INAK stays cleared.

beCAN can be woken up (exit Sleep mode) either by software clearing the SLEEP bit or on detection of CAN bus activity.

On CAN bus activity detection, hardware automatically performs the wake-up sequence by clearing the SLEEP bit if the AWUM bit in the CMCR register is set. If the AWUM bit is cleared, software has to clear the SLEEP bit when a wake-up interrupt occurs, in order to exit from sleep mode.

Note: If the wake-up interrupt is enabled (WKUIE bit set in CIER register) a wake-up interrupt will be generated on detection of CAN bus activity, even if the beCAN automatically performs the wake-up sequence.

After the SLEEP bit has been cleared, Sleep mode is exited once beCAN has synchronized with the CAN bus, refer to [Figure 96](#). The sleep mode is exited once the SLAK bit has been cleared by hardware.

Test mode

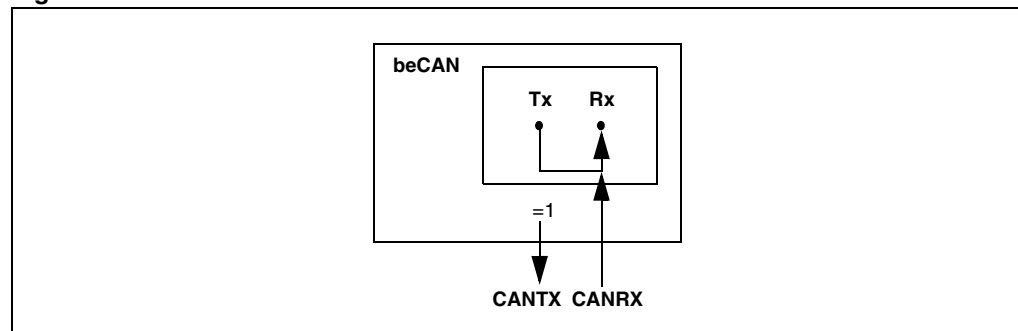
Test mode can be selected by the SILM and LBKM bits in the CDGR register. These bits must be configured while beCAN is in Initialization mode. Once test mode has been selected, beCAN is started in Normal mode.

Silent mode

The beCAN can be put in Silent mode by setting the SILM bit in the CDGR register.

In Silent mode, the beCAN is able to receive valid data frames and valid remote frames, but it sends only recessive bits on the CAN bus and it cannot start a transmission. If the beCAN has to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the CAN core monitors this dominant bit, although the CAN bus may remain in recessive state. Silent mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits (acknowledge bits, error frames).

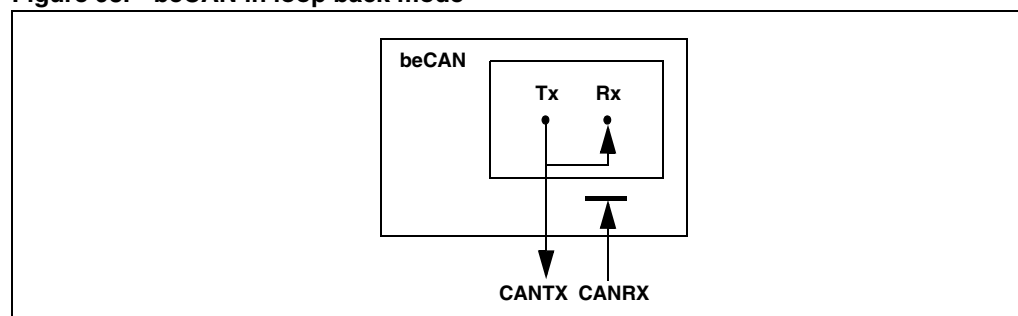
Figure 97. beCAN in silent mode



Loop back mode

The beCAN can be set in loop back mode by setting the LBKM bit in the CDGR register. In loop back mode, the beCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) in the FIFO.

Figure 98. beCAN in loop back mode



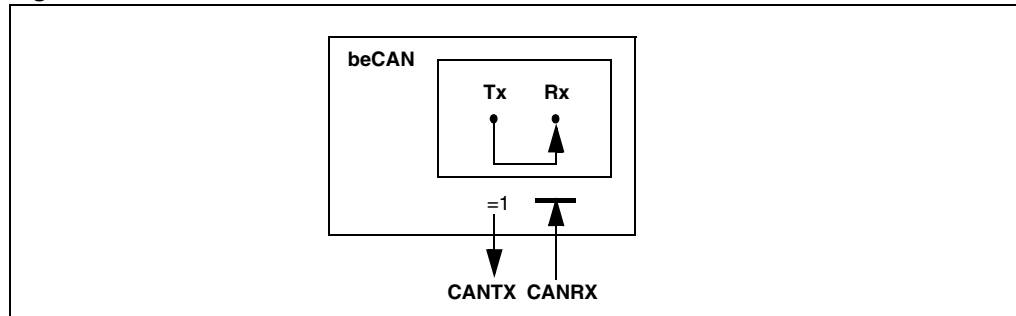
This mode is provided for self-test functions. To be independent of external events, the CAN core ignores acknowledge errors (no dominant bit sampled in the acknowledge slot of a data

/ remote frame) in loop back mode. In this mode, the beCAN performs an internal feedback from its Tx output to its Rx input. The actual value of the CANRX input pin is disregarded by the beCAN. The transmitted messages can be monitored on the CANTX pin.

Loop back combined with silent mode

It is also possible to combine loop back mode and silent mode by setting the LBKM and SILM bits in the CDGR register. This mode can be used for a “Hot Selftest”, meaning the beCAN can be tested like in loop back mode but without affecting a running CAN system connected to the CANTX and CANRX pins. In this mode, the CANRX pin is disconnected from the beCAN and the CANTX pin is held recessive.

Figure 99. beCAN in combined mode



17.4 Functional description

17.4.1 Transmission handling

In order to transmit a message, the application must select one empty transmit mailbox, set up the identifier, the data length code (DLC) and the data before requesting the transmission by setting the corresponding TXRQ bit in the MCSR register. Once the mailbox has left empty state, the software no longer has write access to the mailbox registers. Immediately after the TXRQ bit has been set, the mailbox enters pending state and waits to become the highest priority mailbox, see *Transmit Priority*. As soon as the mailbox has the highest priority it will be scheduled for transmission. The transmission of the message of the scheduled mailbox will start (enter transmit state) when the CAN bus becomes idle. Once the mailbox has been successfully transmitted, it will become empty again. The hardware indicates a successful transmission by setting the RQCP and TXOK bits in the MCSR and CTSR registers.

If the transmission fails, the cause is indicated by the ALST bit in the MCSR register in case of an Arbitration Lost, and/or the TERR bit, in case of transmission error detection.

Transmit priority

By identifier:

when more than one transmit mailbox is pending, the transmission order is given by the identifier of the message stored in the mailbox. The message with the lowest identifier value has the highest priority according to the arbitration of the CAN protocol. If the identifier values are equal, the lower mailbox number will be scheduled first.

By transmit request order:
the transmit mailboxes can be configured as a transmit FIFO by setting the TXFP bit in the CMCR register. In this mode the priority order is given by the transmit request order.

This mode is very useful for segmented transmission.

Abort

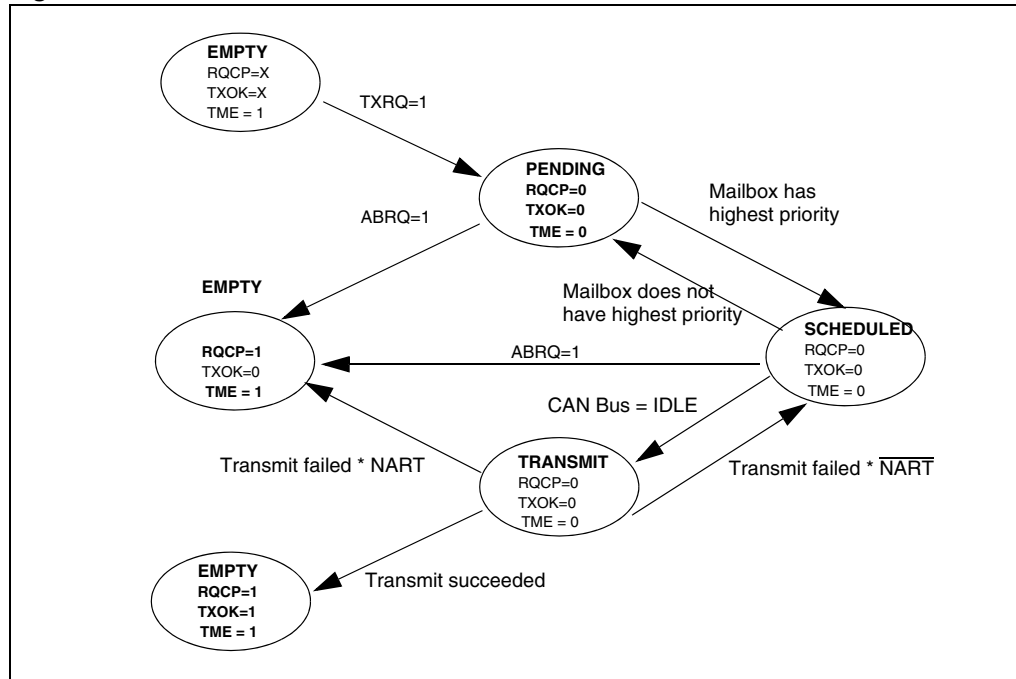
A transmission request can be aborted by the user setting the ABRQ bit in the MCSR register. In pending or scheduled state, the mailbox is aborted immediately. An abort request while the mailbox is in transmit state can have two results. If the mailbox is transmitted successfully the mailbox becomes empty with the TXOK bit set in the MCSR and CTSR registers. If the transmission fails, the mailbox becomes scheduled, the transmission is aborted and becomes empty with TXOK cleared. In all cases the mailbox will become empty again at least at the end of the current transmission.

Non-automatic retransmission mode

To configure the hardware in this mode the NART bit in the CMCR register must be set.

In this mode, each transmission is started only once. If the first attempt fails, due to an arbitration loss or an error, the hardware will not automatically restart the message transmission. At the end of the first transmission attempt, the hardware considers the request as completed and sets the RQCP bit in the MCSR register. The result of the transmission is indicated in the MCSR register by the TXOK, ALST and TERR bits.

Figure 100. Transmit mailbox states



17.4.2 Reception handling

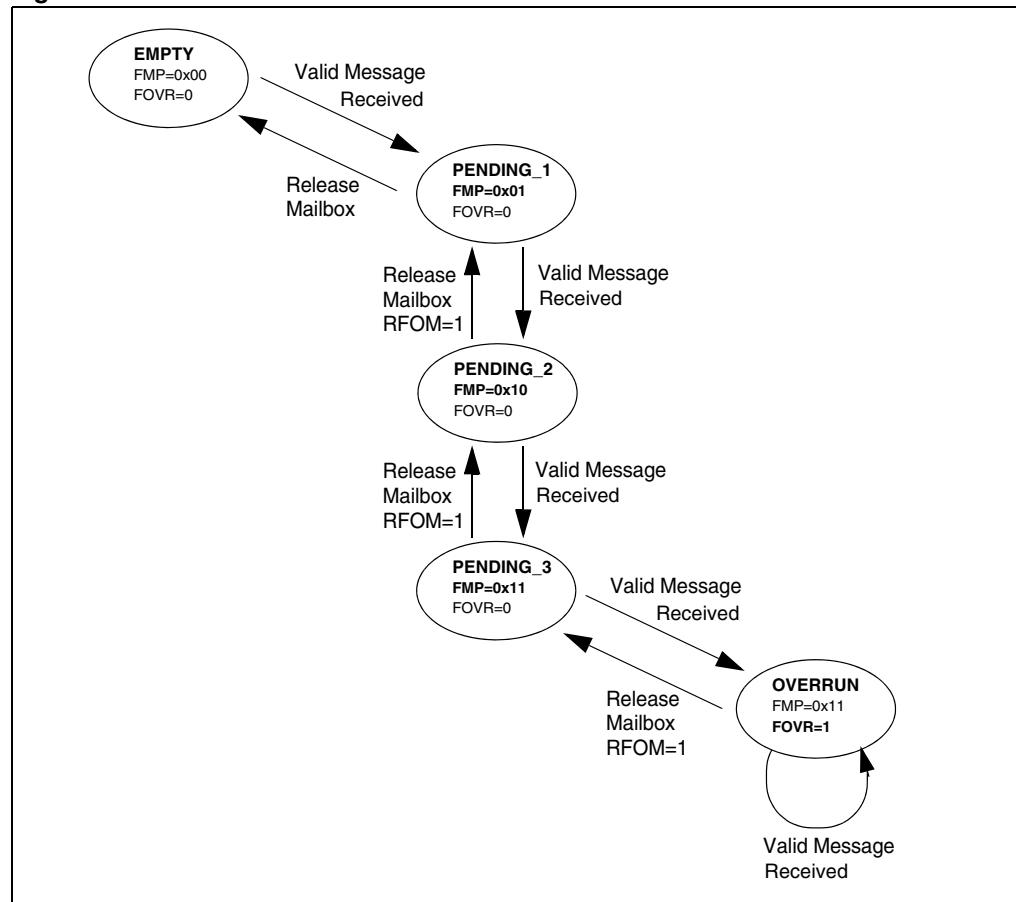
For the reception of CAN messages, three mailboxes organized as a FIFO are provided. In order to save CPU load, simplify the software and guarantee data consistency, the FIFO is

managed completely by hardware. The application accesses the messages stored in the FIFO through the FIFO output mailbox.

Valid message

A received message is considered as valid when it has been received correctly according to the CAN protocol (no error until the last but one bit of the EOF field) and It passed through the identifier filtering successfully, see [Identifier filtering](#).

Figure 101. Receive FIFO states



FIFO management

Starting from the empty state, the first valid message received is stored in the FIFO which becomes pending_1. The hardware signals the event setting the FMP[1:0] bits in the CRFR register to the value 01b. The message is available in the FIFO output mailbox. The software reads out the mailbox content and releases it by setting the RFOM bit in the CRFR register. The FIFO becomes empty again. If a new valid message has been received in the meantime, the FIFO stays in pending_1 state and the new message is available in the output mailbox.

If the application does not release the mailbox, the next valid message will be stored in the FIFO which enters pending_2 state (FMP[1:0] = 10b). The storage process is repeated for the next valid message putting the FIFO into pending_3 state (FMP[1:0] = 11b). At this point,

the software must release the output mailbox by setting the RFOM bit, so that a mailbox is free to store the next valid message. Otherwise the next valid message received will cause a loss of message.

Refer also to [Message storage](#).

Overrun

Once the FIFO is in pending_3 state (that is, the three mailboxes are full) the next valid message reception will lead to an overrun and a message will be lost. The hardware signals the overrun condition by setting the FOVR bit in the CRFR register. Which message is lost depends on the configuration of the FIFO:

- If the FIFO lock function is disabled (RFLM bit in the CMCR register cleared) the last message stored in the FIFO will be overwritten by the new incoming message. In this case the latest messages will be always available to the application.
- If the FIFO lock function is enabled (RFLM bit in the CMCR register set) the most recent message will be discarded and the software will have the three oldest messages in the FIFO available.

Reception related interrupts

On the storage of the first message in the FIFO - FMP[1:0] bits change from 00b to 01b - an interrupt is generated if the FMPIE bit in the CIER register is set.

When the FIFO becomes full (that is, a third message is stored) the FULL bit in the CRFR register is set and an interrupt is generated if the FFIE bit in the CIER register is set.

On overrun condition, the FOVR bit is set and an interrupt is generated if the FOVIE bit in the CIER register is set.

17.4.3 Identifier filtering

In the CAN protocol the identifier of a message is not associated with the address of a node but related to the content of the message. Consequently a transmitter broadcasts its message to all receivers. On message reception a receiver node decides - depending on the identifier value - whether the software needs the message or not. If the message is needed, it is copied into the RAM. If not, the message must be discarded without intervention by the software.

To fulfil this requirement, the beCAN controller provides six configurable and scalable filter banks (0-5) in order to receive only the messages the software needs. This hardware filtering saves CPU resources which would be otherwise needed to perform filtering by software. Each filter bank consists of eight 8-bit registers, CFxR[0:7].

Scalable width

To optimize and adapt the filters to the application needs, each filter bank can be scaled independently. Depending on the filter scale a filter bank provides:

- One 32-bit filter for the STDID[10:0], IDE, EXTID[17:0] and RTR bits.
- Two 16-bit filters for the STDID[10:0], RTR and IDE bits.
- Four 8-bit filters for the STDID[10:3] bits. The other bits are considered as “don't care”.
- One 16-bit filter and two 8-bit filters for filtering the same set of bits as the 16 and 8-bit filters described above.

Refer to [Figure 102](#).

Furthermore, the filters can be configured in mask mode or in identifier list mode.

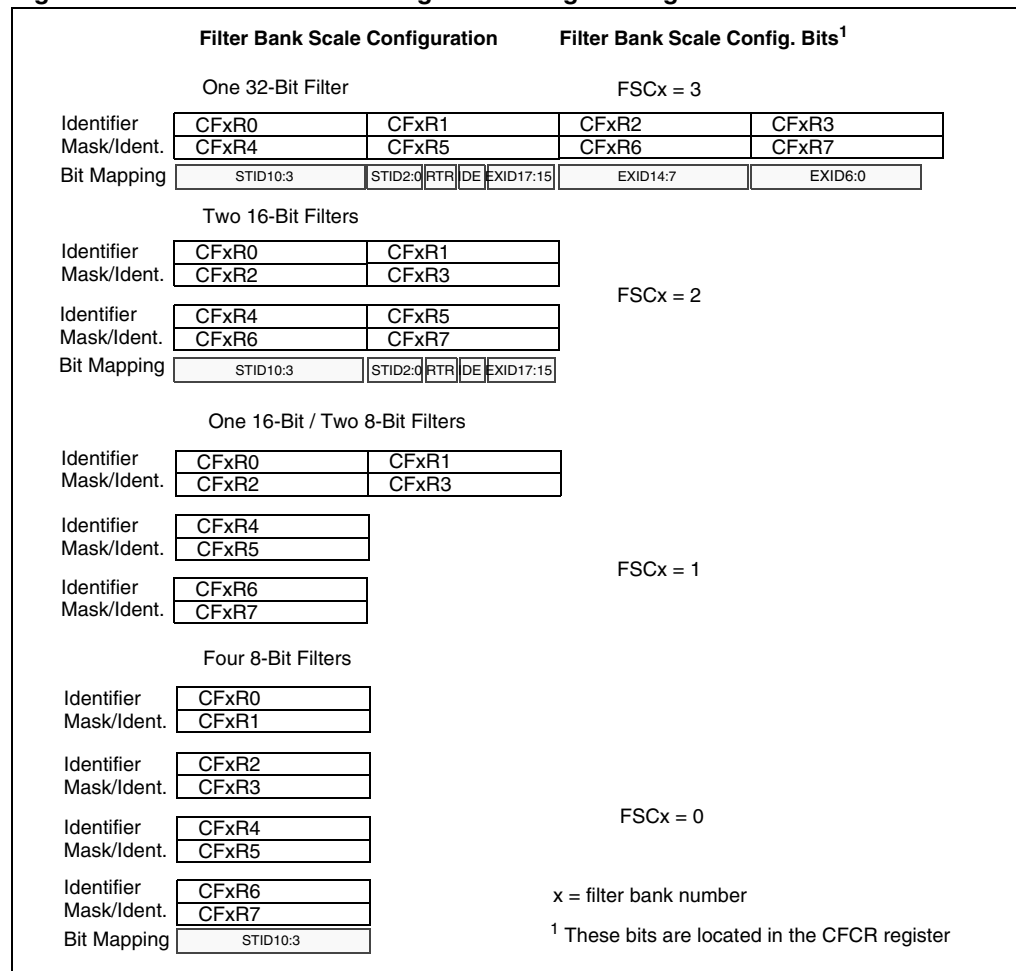
Mask mode

In mask mode the identifier registers are associated with mask registers specifying which bits of the identifier are handled as “must match” or as “don’t care”.

Identifier list mode

In identifier list mode, the mask registers are used as identifier registers. Thus instead of defining an identifier and a mask, two identifiers are specified, doubling the number of single identifiers. All bits of the incoming identifier must match the bits specified in the filter registers.

Figure 102. Filter bank scale configuration - register organization



Filter bank scale and mode configuration

The filter banks are configured by means of the corresponding CFCRx register. To configure a filter bank this must be deactivated by clearing the FACT bit in the CFCR register. The filter scale is configured by means of the FSC[1:0] bits in the corresponding CFCR register, refer

to [Figure 102](#). The identifier list or identifier mask mode for the corresponding Mask/Identifier registers is configured by means of the FMCLx and FMCHx bits in the CFMR register. The FMCLx bit defines the mode for the two least significant bytes, and the FMCHx bit the mode for the two most significant bytes of filter bank x.

Example 1:

- If filter bank 1 is configured as two 16-bit filters, then the FMCL1 bit defines the mode of the CF1R2 and CF1R3 registers and the FMCH1 bit defines the mode of the CF1R6 and CF1R7 registers.
- If filter bank 2 is configured as four 8-bit filters, then the FMCL2 bit defines the mode of the CF2R1 and CF2R3 registers and the FMCH2 bit defines the mode of the CF2R5 and CF2R7 registers.

Note: In 32-bit configuration, the FMCLx and FMCHx bits must have the same value to ensure that the four Mask/Identifier registers are in the same mode.

To filter a group of identifiers, configure the mask/identifier registers in mask mode.

To select single identifiers, configure the mask/identifier registers in identifier list mode.

Filters not used by the application should be left deactivated.

Filter match index

Once a message has been received in the FIFO it is available to the application. Typically application data are copied into RAM locations. To copy the data to the right location the application has to identify the data by means of the identifier. To avoid this and to ease the access to the RAM locations, the CAN controller provides a filter match index.

This index is stored in the mailbox together with the message according to the filter priority rules. Thus each received message has its associated Filter Match Index.

The Filter Match Index can be used in two ways:

- Compare the filter match Index with a list of expected values.
- Use the filter match index as an index on an array to access the data destination location.

For non-masked filters, the software no longer has to compare the identifier.

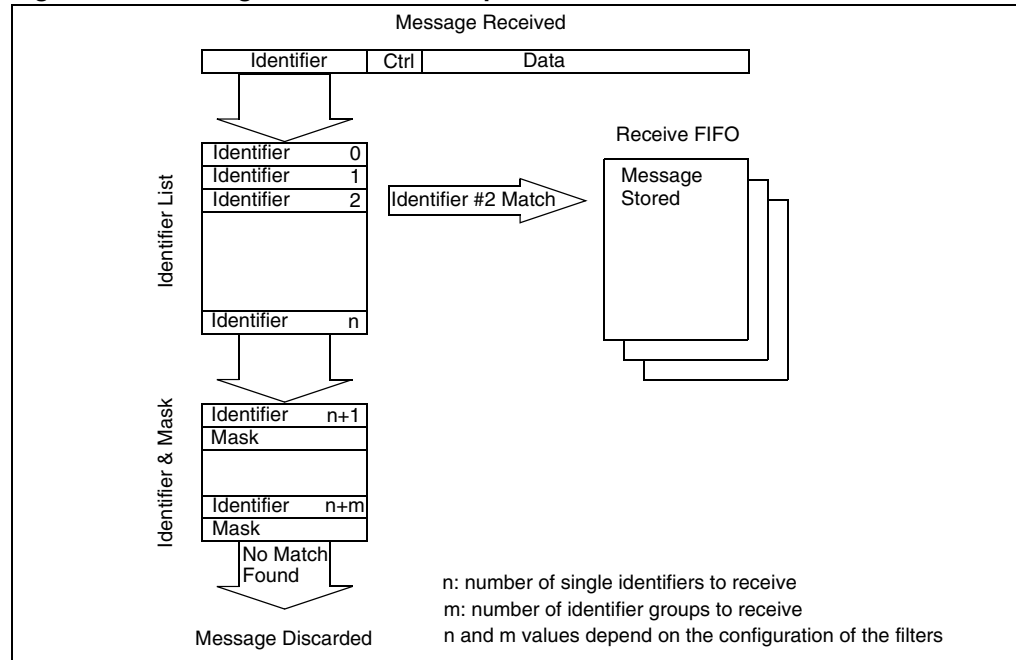
If the filter is masked the software reduces the comparison to the masked bits only.

Filter priority rules

Depending on the filter combination it may occur that an identifier passes successfully through several filters. In this case the filter match value stored in the receive mailbox is chosen according to the following rules:

- A filter in identifier list mode prevails on a filter in mask mode.
- A filter with full identifier coverage prevails over filters covering part of the identifier, e.g. 16-bit filters prevail over 8-bit filters.
- Filters configured in the same mode and with identical coverage are prioritized by filter number and register number. The lower the number the higher the priority.

Figure 103. Filtering mechanism - example



The example above shows the filtering principle of the beCAN. On reception of a message, the identifier is compared first with the filters configured in identifier list mode. If there is a match, the message is stored in the FIFO and the index of the matching filter is stored in the filter match index. As shown in the example, the identifier matches with identifier #2 thus the message content and MFMI 2 is stored in the FIFO.

If there is no match, the incoming identifier is then compared with the filters configured in mask mode.

If the identifier does not match any of the identifiers configured in the filters, the message is discarded by hardware without software intervention.

17.4.4 Message storage

The interface between the software and the hardware for the CAN messages is implemented by means of mailboxes. A mailbox contains all information related to a message; identifier, data, control and status information.

Transmit mailbox

The software sets up the message to be transmitted in an empty transmit mailbox. The status of the transmission is indicated by hardware in the MCSR register.

Table 82. Transmit mailbox mapping

| Offset to transmit mailbox base address (bytes) | Register Name |
|---|---------------|
| 0 | MCSR |
| 1 | MDLC |
| 2 | MIDR0 |
| 3 | MIDR1 |
| 4 | MIDR2 |
| 5 | MIDR3 |
| 6 | MDAR0 |
| 7 | MDAR1 |
| 8 | MDAR2 |
| 9 | MDAR3 |
| 10 | MDAR4 |
| 11 | MDAR5 |
| 12 | MDAR6 |
| 13 | MDAR7 |
| 14 | Reserved |
| 15 | Reserved |

Receive mailbox

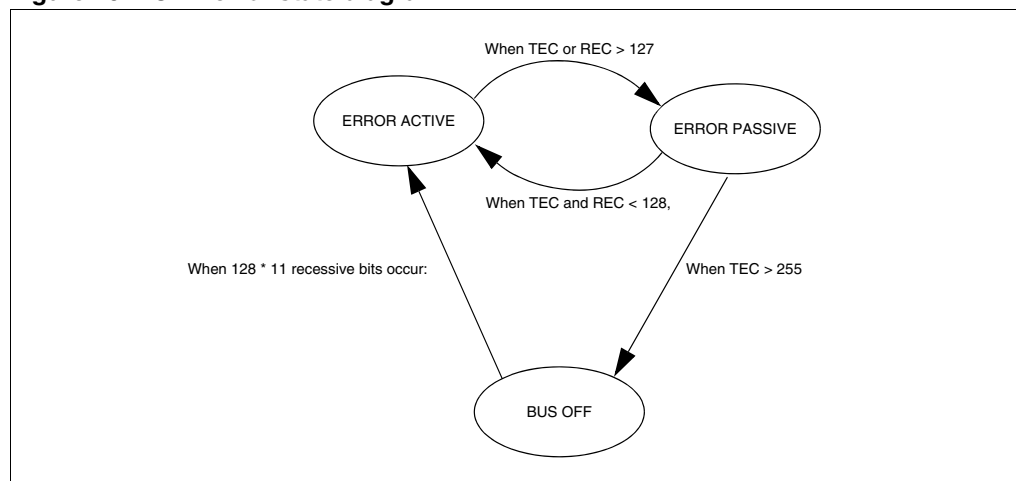
When a message has been received, it is available to the software in the FIFO output mailbox. Once the software has handled the message (e.g. read it) the software must release the FIFO output mailbox by means of the RFOM bit in the CRFR register to make the next incoming message available. The filter match index is stored in the MFMI register.

Table 83. Receive mailbox mapping

| Offset to receive mailbox base address (bytes) | Register Name |
|--|---------------|
| 0 | MFMI |
| 1 | MDLC |
| 2 | MIDR0 |
| 3 | MIDR1 |
| 4 | MIDR2 |
| 5 | MIDR3 |
| 6 | MDAR0 |
| 7 | MDAR1 |
| 8 | MDAR2 |
| 9 | MDAR3 |

Table 83. Receive mailbox mapping (continued)

| Offset to receive mailbox base address (bytes) | Register Name |
|--|---------------|
| 10 | MDAR4 |
| 11 | MDAR5 |
| 12 | MDAR6 |
| 13 | MDAR7 |
| 14 | Reserved |
| 15 | Reserved |

Figure 104. CAN error state diagram

17.4.5 Error management

The error management as described in the CAN protocol is handled entirely by hardware using a Transmit Error Counter (TECR register) and a Receive Error Counter (RECR register), which get incremented or decremented according to the error condition. For detailed information about TEC and REC management, please refer to the CAN standard.

Both of them may be read by software to determine the stability of the network. Furthermore, the CAN hardware provides detailed information on the current error status in CESR register. By means of CEIER register and ERRIE bit in CIER register, the software can configure the interrupt generation on error detection in a very flexible way.

Bus-off recovery

The Bus-Off state is reached when TECR is greater than 255, this state is indicated by BOFF bit in CESR register. In Bus-Off state, the beCAN acts as disconnected from the CAN bus, hence it is no longer able to transmit and receive messages.

Depending on the ABOM bit in the CMCR register beCAN will recover from Bus-Off (become error active again) either automatically or on software request. But in both cases the beCAN has to wait at least for the recovery sequence specified in the CAN standard (128 x 11 consecutive recessive bits monitored on CANRX).

If ABOM is set, the beCAN will start the recovering sequence automatically after it has entered bus-off state.

If ABOM is cleared, the software must initiate the recovering sequence by requesting beCAN to enter initialization mode. Then beCAN starts monitoring the recovery sequence when the beCAN is requested to leave the initialization mode.

Note: In initialization mode, beCAN does not monitor the CANRX signal, therefore it cannot complete the recovery sequence. To recover, beCAN must be in normal mode.

17.4.6 Bit timing

The bit timing logic monitors the serial bus-line and performs sampling and adjustment of the sample point by synchronizing on the start-bit edge and resynchronizing on the following edges.

Its operation may be explained simply by splitting nominal bit time into three segments as follows:

- Synchronization segment (SYNC_SEG):
a bit change is expected to occur within this time segment. It has a fixed length of one time quantum ($1 \times t_{CAN}$).
- Bit segment 1 (BS1):
defines the location of the sample point. It includes the PROP_SEG and PHASE_SEG1 of the CAN standard. Its duration is programmable between 1 and 16 time quanta but may be automatically lengthened to compensate for positive phase drifts due to differences in the frequency of the various nodes of the network.
- Bit segment 2 (BS2):
defines the location of the transmit point. It represents the PHASE_SEG2 of the CAN standard. Its duration is programmable between 1 and 8 time quanta but may also be automatically shortened to compensate for negative phase drifts.
- Resynchronization Jump Width (RJW):
defines an upper bound to the amount of lengthening or shortening of the bit segments. It is programmable between 1 and 4 time quanta.

To guarantee the correct behavior of the CAN controller, SYNC_SEG + BS1 + BS2 must be greater than or equal to 5 time quanta.

For a detailed description of the CAN resynchronization mechanism and other bit timing configuration constraints, please refer to the Bosch CAN standard 2.0.

As a safeguard against programming errors, the configuration of the Bit Timing Registers CBTR1 and CBTR0 is only possible while the device is in Initialization mode.

Figure 105. Bit timing

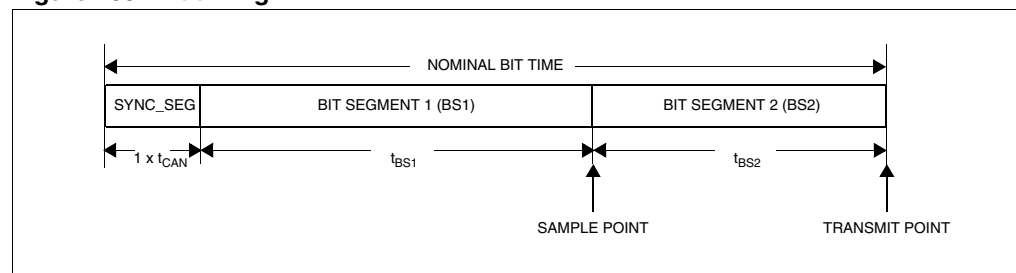


Figure 106. CAN frames (part 1/2)

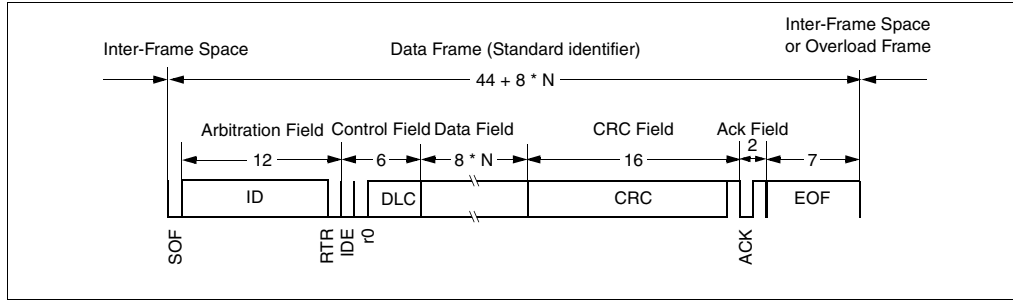
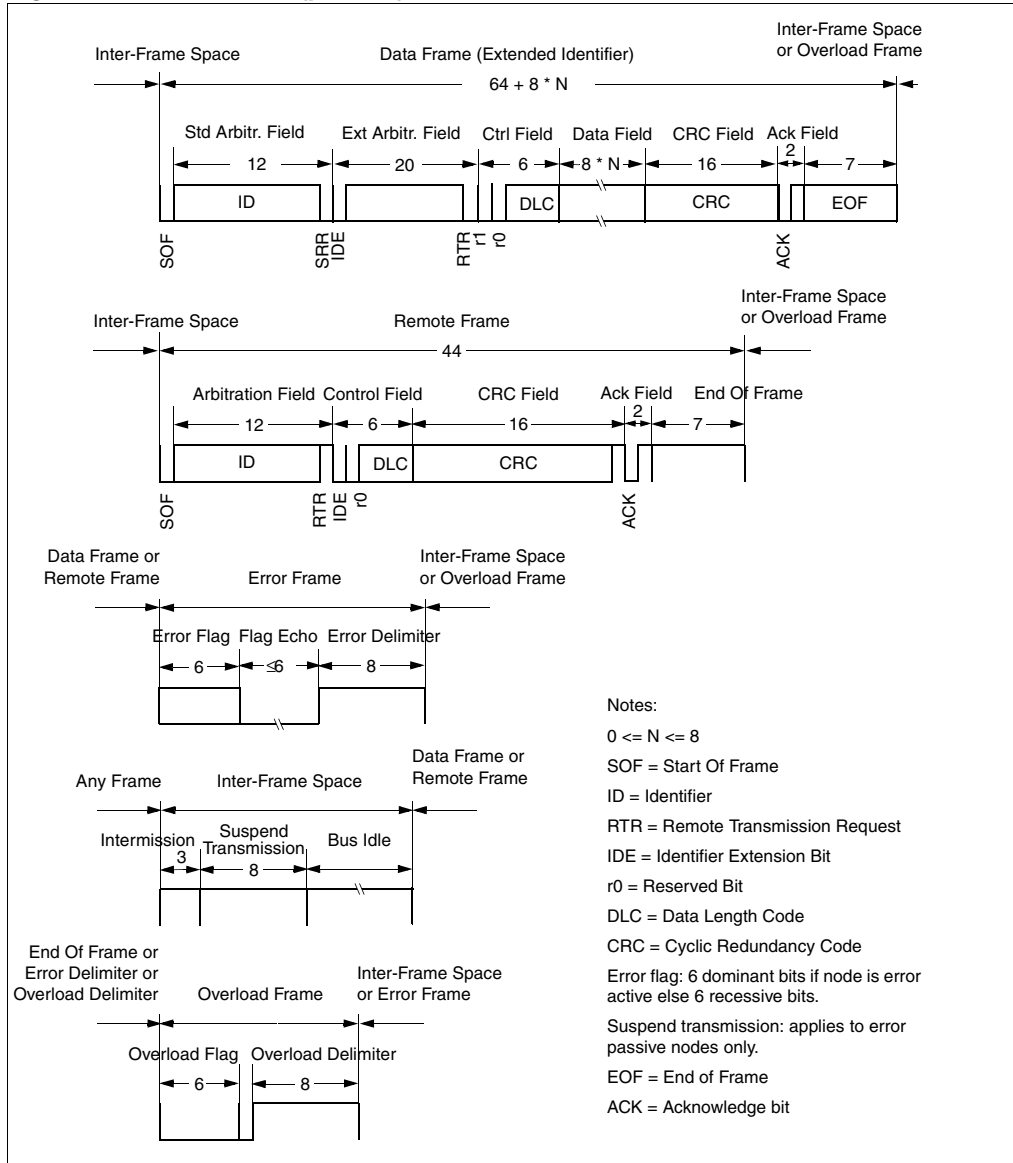


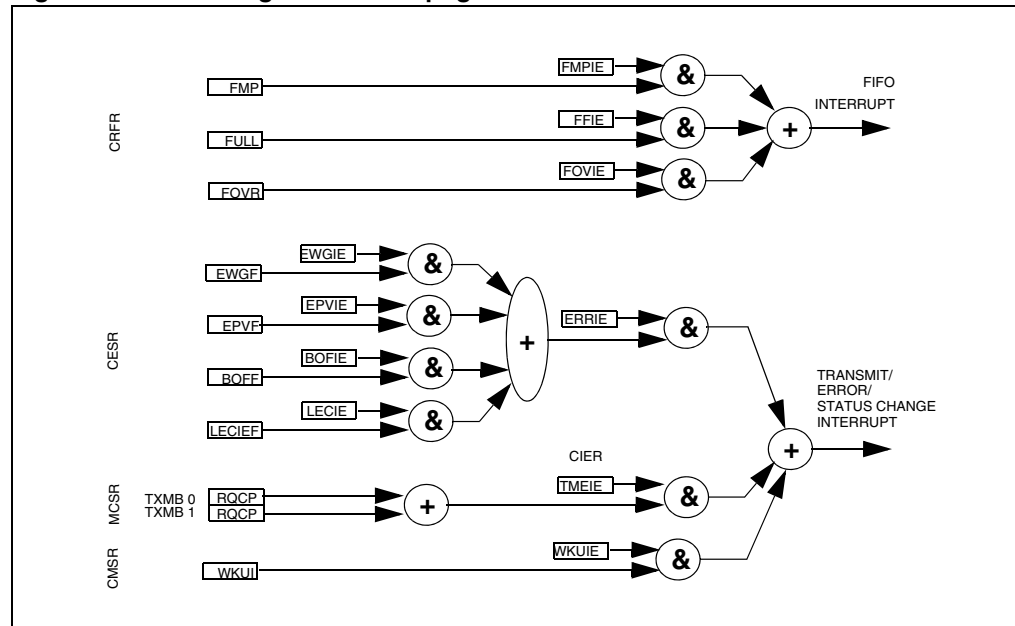
Figure 107. CAN frames (part 2/2)



17.5 Interrupts

Two interrupt vectors are dedicated to beCAN. Each interrupt source can be independently enabled or disabled by means of the CAN Interrupt Enable Register (CIER) and CAN Error Interrupt Enable register (CEIER).

Figure 108. Event flags and interrupt generation



- The FIFO interrupt can be generated by the following events:
 - Reception of a new message, FMP bits in the CRFR0 register incremented.
 - FIFO0 full condition, FULL bit in the CRFR0 register set.
 - FIFO0 overrun condition, FOVR bit in the CRFR0 register set.
- The transmit, error and status change interrupt can be generated by the following events:
 - Transmit mailbox 0 becomes empty, RQCP0 bit in the CTSR register set.
 - Transmit mailbox 1 becomes empty, RQCP1 bit in the CTSR register set.
 - Error condition, for more details on error conditions please refer to the CAN Error Status register (CESR).
 - Wake-up condition, SOF monitored on the CAN Rx signal.

17.6 Register access protection

Erroneous access to certain configuration registers can cause the hardware to temporarily disturb the whole CAN network. Therefore the following registers can be modified by software only while the hardware is in initialization mode:

CBTR0, CBTR1, CFCR0, CFCR1, CFMR and CDGR registers.

Although the transmission of incorrect data will not cause problems at the CAN network level, it can severely disturb the application. A transmit mailbox can be only modified by software while it is in empty state (refer to [Figure 100](#)).

The filters must be deactivated before their value can be modified by software. The modification of the filter configuration (scale or mode) can be done by software only in initialization mode.

17.7 beCAN cell limitations

17.7.1 FIFO corruption

FIFO corruption occurs in the following case:

WHEN the beCAN RX FIFO already holds two messages (that is, $FMP == 2$)

AND the application releases the FIFO (with the instruction $CRFR = B_RFOM$;))

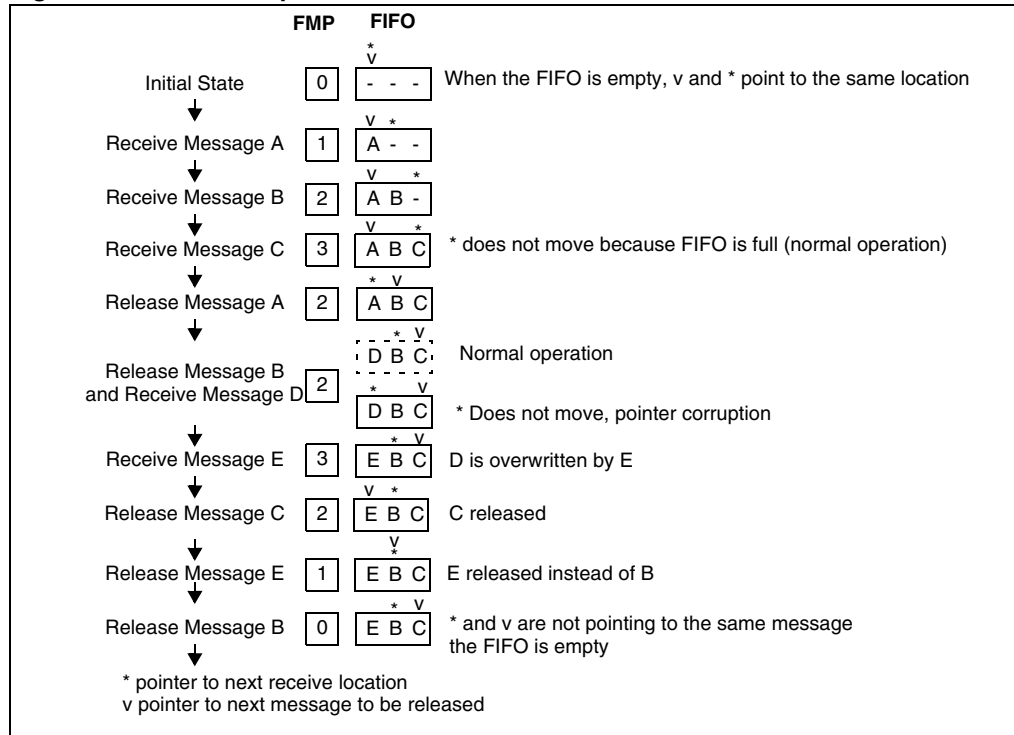
WHILE the beCAN requests the transfer of a new receive message into the FIFO (this lasts one CPU cycle)

THEN the internal FIFO pointer is not updated

BUT the FMP bits are updated correctly

As the FIFO pointer is not updated correctly, this causes the last message received to be overwritten by any incoming message. This means one message is lost as shown in the example in [Figure 109](#). The beCAN will not recover normal operation until a device reset occurs.

Figure 109. FIFO corruption



Workaround

To implement the workaround, use the following sequence to release the CAN receive FIFO. This sequence replaces any occurrence of `CRFR |= B_RFOM;`.

Figure 110. Workaround 1

```
if ((CRFR & 0x03) == 0x02)
    while ((CMSR & 0x20) && (CDGR & 0x08)) { };
CRFR |= B_RFOM;
```

Explanation of workaround 1

First, we need to make sure no interrupt can occur between the test and the release of the FIFO to avoid any added delay.

The workaround checks if the first two FIFO levels are already full (FMP = 2) as the problem happens only in this case.

If $FMP \neq 2$ we release the FIFO immediately, if $FMP = 2$, we monitor the reception status of the cell.

The reception status is available in the CMSR register bit 5 (REC bit).

Note: The REC bit was called RX in older versions of the datasheet.

- If the cell is not receiving, then REC bit in CMSR is at 0, the software can release the FIFO immediately: there is no risk.
- If the cell is receiving, it is important to make sure the release of the mailbox will not happen at the time when the received message is loaded into the FIFO.



We could simply wait for the end of the reception, but this could take a long time (200µs for a 100-bit frame at 500 kHz), so we also monitor the Rx pin of the microcontroller to minimize the time the application may wait in the while loop.

We know the critical window is located at the end of the frame, 6+ CAN bit times after the acknowledge bit (exactly six full bit times plus the time from the beginning of the bit to the sample point). Those bits represent the acknowledge delimiter + the end of frame slot.

We know also that those 6+ bits are in recessive state on the bus, therefore if the CAN Rx pin of the device is at '0', (reflecting a CAN dominant state on the bus), this is early enough to be sure we can release the FIFO before the critical time slot.

Therefore, if the device hardware pin Rx is at 0 and there is a reception on going, its message will be transferred to the FIFO only 6+ CAN bit times later at the earliest (if the dominant bit is the acknowledge) or later if the dominant bit is part of the message.

Compiled with Cosmic C compiler, the workaround generates the following assembly lines:

| | | Cycles | |
|--|------|----------------------|---------------------|
| if ((CRFR & 0x03) == 0x02) | | | |
| | ld | a, CRFR | 3 |
| | and | a, #3 | 2 |
| | cp | a, #2 | 2 |
| | jrne | _RELEASE | 3 test: 10 cycles |
| while ((CMSR & 0x20) && (CDGR & 0x08)) { }; | | | |
| _WHILELOOP: | | | |
| | btjf | CMSR, #5, _RELEASE | 5 |
| | btjt | CDGR, #3, _WHILELOOP | 5 loop: 10 cycles |
| CRFR = B_RFOM; | | | |
| _RELEASE: | | | |
| | bset | CRFR, #5 | 5 release: 5 cycles |

In the worst case configuration, if the CAN cell speed is set to the maximum baud rate, one bit time is 8 CPU cycle. In this case the minimum time between the end of the acknowledge and the critical period is 52 CPU cycles (48 for the 6 bit times + 4 for the (PROP SEG + T_{Seg1})). According to the previous code timing, we need less than 15 cycles from the time we see the dominant state to the time we perform the FIFO release (one full loop + the actual release) therefore the application will never release the FIFO at the critical time when this workaround is implemented.

Timing analysis

- Time spent in the workaround

Inside a CAN frame, the longest period that the Rx pin stays in recessive state is 5 bits. At the end of the frame, the time between the acknowledge dominant bit and the end of reception (signaled by REC bit status) is $8t_{CANbit}$, therefore the maximum time spent in the workaround is: $8t_{CANbit} + t_{loop} + t_{test} + t_{release}$ in this case or $8t_{CANbit} + 25t_{CPU}$.

At low speed, this time could represent a long delay for the application, therefore it makes sense to evaluate how frequently this delay occurs.

In order to reach the critical FMP = 2, the CAN node needs to receive two messages without servicing them. Then in order to reach the critical window, the cell has to receive a third one and the application has to release the mailbox at the same time, at the end of the reception.

In the application, messages are not processed only if either the interrupt are disabled or higher level interrupts are being serviced.

Therefore if: $t_{IT \text{ higher level}} + t_{IT \text{ disable}} + t_{IT \text{ CAN}} < 2 \times t_{CAN \text{ frame}}$

the application will never wait in the workaround

$t_{IT \text{ higher level}}$: This is the sum of the duration of all the interrupts with a level strictly higher than the CAN interrupt level

$t_{IT \text{ disable}}$: This is the longest time the application disables the CAN interrupt (or all interrupts)

$t_{IT \text{ CAN}}$: This is the maximum duration between the beginning of the CAN interrupt and the actual location of the workaround

$t_{CAN \text{ frame}}$: This is minimum CAN frame duration

Figure 111. Critical window timing diagram

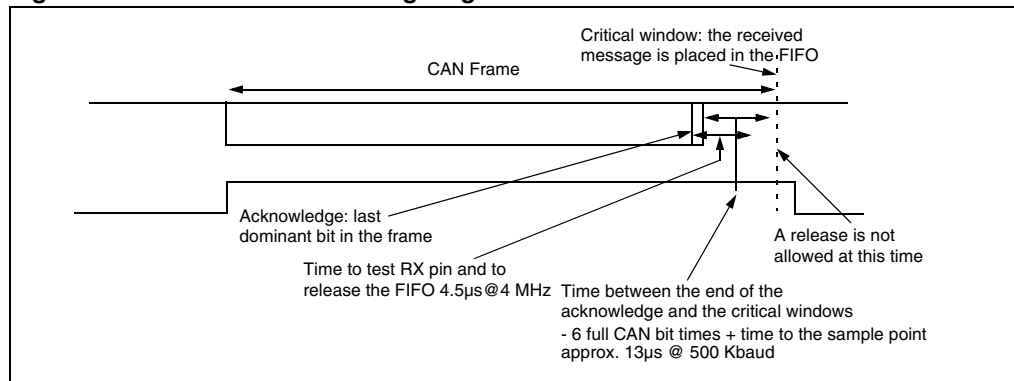
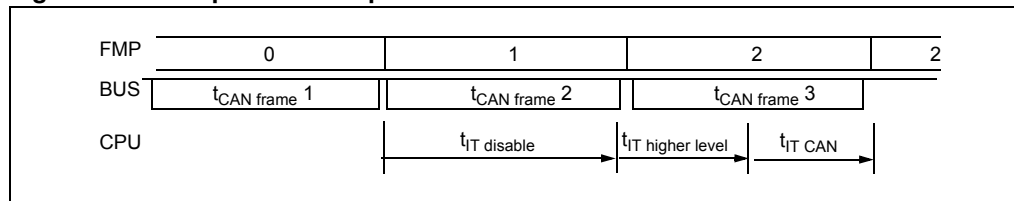


Figure 112. Reception of a sequence of frames



Side-effect of workaround 1

Because the while loop lasts 10 CPU cycles, at high baud rate, it is possible to miss a dominant state on the bus if it lasts just one CAN bit time and the bus speed is high enough (see [Table 84](#)).

Table 84. While loop timing

| f_{CPU} | Software timing: while loop | Minimum baud rate for possible missed dominant bit |
|-----------|-----------------------------|--|
| 8 MHz | 1.25 μ s | 800 Kbaud |
| 4 MHz | 2.5 μ s | 400 Kbaud |
| f_{CPU} | $10/f_{CPU}$ | $f_{CPU}/10$ |

If this happens, we will continue waiting in the while loop instead of releasing the FIFO immediately. The workaround is still valid because we will not release the FIFO during the

critical period. But the application may lose additional time waiting in the while loop as we are no longer able to guarantee a maximum of 6 CAN bit times spent in the workaround.

In this particular case the time the application can spend in the workaround may increase up to a full CAN frame, depending of the frame contents. This case is very rare but happens when a specific sequence is present on in the CAN frame.

The example in [Figure 113](#) shows reception at maximum CAN baud rate: In this case t_{CAN} is $8/f_{CPU}$ and the sampling time is $10/f_{CPU}$.

If the application is using the maximum baud rate and the possible delay caused by the workaround is not acceptable, there is another workaround which reduces the Rx pin sampling time.

Workaround 2 (see [Figure 114](#)) first tests that $FMP = 2$ and the CAN cell is receiving, if not the FIFO can be released immediately. If yes, the program goes through a sequence of test instructions on the RX pin that last longer than the time between the acknowledge dominant bit and the critical time slot. If the Rx pin is in recessive state for more than 8 CAN bit times, it means we are now after the acknowledge and the critical slot. If a dominant bit is read on the bus, we can release the FIFO immediately. This workaround has to be written in assembly language to avoid the compiler optimizing the test sequence.

The implementation shown here is for the CAN bus maximum speed (1 Mbaud @ 8 MHz CPU clock).

Figure 113. Reception at maximum CAN baud rate

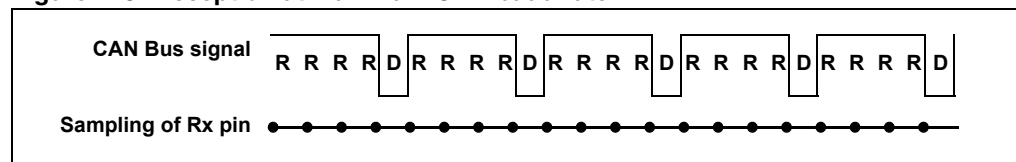


Figure 114. Workaround 2

```

Ld      a, CRFR
And     a, #3
Cp      a, #2          ; test FMP=2 ?
Jrne    _RELEASE      ; if not release

Btjf    CMSR, #5, _RELEASE ; test if reception on going.
                          ; if not release

Btjf    CDGR, #3, _RELEASE ; sample RX pin for 8 CAN bit time
Btjf    CDGR, #3, _RELEASE
Btjf    CDGR, #3, _RELEASE
btjf    CDGR, #3, _RELEASE
btjf    CDGR, #3, _RELEASE
btjf    CDGR, #3, _RELEASE
btjf    CDGR, #3, _RELEASE
btjf    CDGR, #3, _RELEASE
btjf    CDGR, #3, _RELEASE
btjf    CDGR, #3, _RELEASE
btjf    CDGR, #3, _RELEASE
btjf    CDGR, #3, _RELEASE
btjf    CDGR, #3, _RELEASE
btjf    CDGR, #3, _RELEASE
btjf    CDGR, #3, _RELEASE
btjf    CDGR, #3, _RELEASE

_RELEASE:
bset    CRFR, #5

```

17.8 Register description

17.8.1 Control and status registers

CAN master control register (CMCR)

Reset value: 0000 0010 (02h)

| | | | | | | | |
|---|------|------|------|------|------|-------|------|
| 7 | | | | | | | 0 |
| 0 | ABOM | AWUM | NART | RFLM | TXFP | SLEEP | INRQ |

Bit 7 = Reserved, must be kept cleared.

Bit 6 = **ABOM** *Automatic Bus-Off Management*

Read/set/clear

This bit controls the behaviour of the CAN hardware on leaving the bus-off state.

0: the bus-off state is left on software request.

Refer to [Error management](#), bus-off recovery.

1: the bus-off state is left automatically by hardware once 128 x 11 recessive bits have been monitored.

For detailed information on the bus-off state please refer to [Error management](#).

Bit 5 = **AWUM** *Automatic Wake-Up Mode*

Read/set/clear

This bit controls the behaviour of the CAN hardware on message reception during sleep mode.

0: the sleep mode is left on software request by clearing the SLEEP bit of the CMCR register.

1: the sleep mode is left automatically by hardware on CAN message detection. The SLEEP bit of the CMCR register and the SLAK bit of the CMSR register are cleared by hardware.

Bit 4 = **NART** *No Automatic Retransmission*

Read/set/clear

0: the CAN hardware will automatically retransmit the message until it has been successfully transmitted according to the CAN standard.

1: a message will be transmitted only once, independently of the transmission result (successful, error or arbitration lost).

Bit 3 = **RFLM** *Receive FIFO Locked Mode*

Read/set/clear

0: receive FIFO not locked on overrun. Once a receive FIFO is full the next incoming message will overwrite the previous one.

1: receive FIFO locked against overrun. Once a receive FIFO is full the next incoming message will be discarded.

Bit 2 = **TXFP** *Transmit FIFO Priority*

Read/set/clear

This bit controls the transmission order when several mailboxes are pending at the same time.

- 0: priority driven by the identifier of the message
- 1: priority driven by the request order (chronologically)

Bit 1 = **SLEEP** *Sleep Mode Request*

Read/set/clear

This bit is set by software to request the CAN hardware to enter the sleep mode. Sleep mode will be entered as soon as the current CAN activity (transmission or reception of a CAN frame) has been completed.

This bit is cleared by software to exit sleep mode.

This bit is cleared by hardware when the AWUM bit is set and a SOF bit is detected on the CAN Rx signal.

Bit 0 = **INRQ** *Initialization Request*

Read/set/clear

The software clears this bit to switch the hardware into normal mode. Once 11 consecutive recessive bits have been monitored on the Rx signal the CAN hardware is synchronized and ready for transmission and reception. Hardware signals this event by clearing the INAK bit if the CMSR register.

Software sets this bit to request the CAN hardware to enter initialization mode. Once software has set the INRQ bit, the CAN hardware waits until the current CAN activity (transmission or reception) is completed before entering the initialization mode. Hardware signals this event by setting the INAK bit in the CMSR register.

CAN master status register (CMSR)

Reset value: 0000 0010 (02h)

| | | | | | | | |
|---|---|-----|------|------|------|------|------|
| 7 | | | | | | | 0 |
| 0 | 0 | REC | TRAN | WKUI | ERRI | SLAK | INAK |

Note: To clear a bit of this register the software must write this bit with a one.

Bits 7:4 = reserved. Forced to 0 by hardware.

Bit 5 = **REC** *Receive*

Read

The CAN hardware is currently receiver.

Bit 4 = **TRAN** *Transmit*

Read

The CAN hardware is currently transmitter.

Bit 3 = **WKUI** *Wake-Up Interrupt*

Read/clear

This bit is set by hardware to signal that a SOF bit has been detected while the CAN hardware was in sleep mode. Setting this bit generates a status change interrupt if the WKUIE bit in the CIER register is set.

This bit is cleared by software.

Bit 2 = **ERRI** *Error Interrupt*

Read/clear

This bit is set by hardware when a bit of the CESR has been set on error detection and the

corresponding interrupt in the CEIER is enabled. Setting this bit generates a status change interrupt if the ERRIE bit in the CIER register is set.

This bit is cleared by software.

Bit 1 = **SLAK** *Sleep Acknowledge*

Read

This bit is set by hardware and indicates to the software that the CAN hardware is now in sleep mode. This bit acknowledges the sleep mode request from the software (set SLEEP bit in CMCR register).

This bit is cleared by hardware when the CAN hardware has left sleep mode. Sleep mode is left when the SLEEP bit in the CMCR register is cleared. Please refer to the AWUM bit of the CMCR register description for detailed information for clearing SLEEP bit.

Bit 0 = **INAK** *Initialization Acknowledge*

Read

This bit is set by hardware and indicates to the software that the CAN hardware is now in initialization mode. This bit acknowledges the initialization request from the software (set INRQ bit in CMCR register).

This bit is cleared by hardware when the CAN hardware has left the initialization mode and is now synchronized on the CAN bus. To be synchronized the hardware has to monitor a sequence of 11 consecutive recessive bits on the CAN RX signal.

17.8.2 CAN transmit status register (CTSR)

Read/ write

Reset value: 0000 0000 (00h)

| | | | | | | | |
|---|---|-------|-------|---|---|-------|-------|
| 7 | | | | | | | 0 |
| 0 | 0 | TXOK1 | TXOK0 | 0 | 0 | RQCP1 | RQCP0 |

Note: To clear a bit of this register the software must write this bit with a one.

Bits 7:6 = reserved. Forced to 0 by hardware.

Bit 5 = **TXOK1** *Transmission OK for mailbox 1*

Read

This bit is set by hardware when the transmission request on mailbox 1 has been completed successfully. Please refer to [Figure 100](#).

This bit is cleared by hardware when mailbox 1 is requested for transmission or when the software clears the RQCP1 bit.

Bit 4 = **TXOK0** *Transmission OK for mailbox 0*

Read

This bit is set by hardware when the transmission request on mailbox 0 has been completed successfully. Please refer to [Figure 100](#).

This bit is cleared by hardware when mailbox 0 is requested for transmission or when the software clears the RQCP0 bit.

Bits 3:2 = reserved. Forced to 0 by hardware.

Bit 1 = **RQCP1** *Request Completed for Mailbox 1*

Read/ clear

This bit is set by hardware to signal that the last request for mailbox 1 has been completed. The request could be a transmit or an abort request.

This bit is cleared by software.

Bit 0 = **RQCP0** *Request Completed for Mailbox 0*

Read/ clear

This bit is set by hardware to signal that the last request for mailbox 0 has been completed. The request could be a transmit or an abort request.

This bit is cleared by software.

CAN transmit priority register (CTPR)

All bits of this register are read only.

Reset value: 0000 1100 (0Ch)

| | | | | | | | |
|---|------|------|---|------|------|---|------|
| 7 | | | | | | | 0 |
| 0 | LOW1 | LOW0 | 0 | TME1 | TME0 | 0 | CODE |

Bit 7 = reserved. Forced to 0 by hardware.

Bit 6 = **LOW1** *Lowest Priority Flag for Mailbox 1*

Read

This bit is set by hardware when more than one mailbox are pending for transmission and mailbox 1 has the lowest priority.

Bit 5 = **LOW0** *Lowest Priority Flag for Mailbox 0*

Read

This bit is set by hardware when more than one mailbox are pending for transmission and mailbox 0 has the lowest priority.

Note: These bits are set to zero when only one mailbox is pending.

Bit 4 = Reserved. Forced to 0 by hardware.

Bit 3 = **TME1** *Transmit Mailbox 1 Empty*

Read

This bit is set by hardware when no transmit request is pending for mailbox 1.

Bit 2 = **TME0** *Transmit Mailbox 0 Empty*

Read

This bit is set by hardware when no transmit request is pending for mailbox 0.

Bit 1:0 = **CODE** *Mailbox Code*

Read

In case at least one transmit mailbox is free, the code value is equal to the number of the next transmit mailbox free.

In case all transmit mailboxes are pending, the code value is equal to the number of the transmit mailbox with the lowest priority.

CAN receive FIFO registers (CRFR)

Read / write

Reset value: 0000 0000 (00h)

| | | | | | | | |
|---|---|------|------|------|---|------|------|
| 7 | | | | | | | 0 |
| 0 | 0 | RFOM | FOVR | FULL | 0 | FMP1 | FMP0 |

Note: To clear a bit in this register, software must write a "1" to the bit.

Bits 7:6 = reserved. Forced to 0 by hardware.

Bit 5 = **RFOM** Release FIFO Output Mailbox

Read/Set

Set by software to release the output mailbox of the FIFO. The output mailbox can only be released when at least one message is pending in the FIFO. Setting this bit when the FIFO is empty has no effect. If more than one message are pending in the FIFO, the software has to release the output mailbox to access the next message.

Cleared by hardware when the output mailbox has been released.

Bit 4 = **FOVR** FIFO Overrun

Read/ clear

This bit is set by hardware when a new message has been received and passed the filter while the FIFO was full.

This bit is cleared by software.

Bit 3 = **FULL** FIFO Full

Read/ clear

Set by hardware when three messages are stored in the FIFO.

This bit can be cleared by software writing a one to this bit or releasing the FIFO by means of RFOM.

Bit 2 = reserved. Forced to 0 by hardware.

Bits 1:0 = **FMP[1:0]** FIFO Message Pending

Read

These bits indicate how many messages are pending in the receive FIFO.

FMP is increased each time the hardware stores a new message in to the FIFO. FMP is decreased each time the software releases the output mailbox by setting the RFOM bit.

CAN Interrupt Enable Register (CIER)

Read/ write

Reset value: 0000 0000 (00h)

| | | | | | | | |
|-------|---|---|---|--------|-------|--------|-------|
| 7 | | | | | | | 0 |
| WКУIE | 0 | 0 | 0 | FOVIE0 | FFIE0 | FMPIE0 | TMEIE |

Note: All bits of this register are set and cleared by software.

Bit 7 = **WKUIE** *Wake-Up Interrupt Enable*

- 0: no interrupt when WKUI is set.
- 1: interrupt generated when WKUI bit is set.

Bits 6:4 = reserved. Forced to 0 by hardware.

Bit 3 = **FOVIE** *FIFO Overrun Interrupt Enable*

- 0: no interrupt when FOVR bit is set.
- 1: interrupt generated when FOVR bit is set.

Bit 2 = **FFIE** *FIFO Full Interrupt Enable*

- 0: no interrupt when FULL bit is set.
- 1: interrupt generated when FULL bit is set.

Bit 1 = **FMPPIE** *FIFO Message Pending Interrupt Enable*

- 0: no interrupt on FMP[1:0] bits transition from 00b to 01b.
- 1: interrupt generated on FMP[1:0] bits transition from 00b to 01b.

Bit 0 = **TMEIE** *Transmit Mailbox Empty Interrupt Enable*

- 0: no interrupt when RQCPx bit is set.
- 1: interrupt generated when RQCPx bit is set.

CAN error status register (CESR)

Read/ write

Reset value: 0000 0000 (00h)

| | | | | | | | |
|---|------|------|------|---|------|------|------|
| 7 | | | | | | | 0 |
| 0 | LEC2 | LEC1 | LEC0 | 0 | BOFF | EPVF | EWGF |

Bit 7 = reserved. Forced to 0 by hardware.

Bits 6:4 = **LEC[2:0]** *Last Error Code*

Read/ set/ clear

This field holds a code which indicates the type of the last error detected on the CAN bus. If a message has been transferred (reception or transmission) without error, this field will be cleared to '0'. The code 7 is unused and may be written by the CPU to check for update

Table 85. LEC error types

| Code | Error type |
|------|----------------------|
| 0 | No Error |
| 1 | Stuff Error |
| 2 | Form Error |
| 3 | Acknowledgment Error |
| 4 | Bit recessive Error |
| 5 | Bit dominant Error |
| 6 | CRC Error |
| 7 | Set by software |

Bit 3 = reserved. Forced to 0 by hardware.

Bit 2 = **BOFF** *Bus-Off Flag*

Read

This bit is set by hardware when it enters the bus-off state. The bus-off state is entered on TECR overrun, TEC greater than 255, refer to [Error management](#).

Bit 1 = **EPVF** *Error Passive Flag*

Read

This bit is set by hardware when the error passive limit has been reached (receive error counter or transmit error counter greater than 127).

Bit 1 = **EWGF** *Error Warning Flag*

Read

This bit is set by hardware when the warning limit has been reached. Receive error counter or transmit error counter greater than 96.

CAN error interrupt enable register (CEIER)

Read/ write

Reset value: 0000 0000 (00h)

| | | | | | | | |
|-------|---|---|-------|---|-------|-------|-------|
| 7 | | | | | | | 0 |
| ERRIE | 0 | 0 | LECIE | 0 | BOFIE | EPVIE | EWGIE |

Note: All bits of this register are set and clear by software.

Bit 7 = **ERRIE** *Error Interrupt Enable*

- 0: no interrupt will be generated when an error condition is pending in the CESR.
- 1: an interrupt will be generation when an error condition is pending in the CESR.

Bits 6:5 = reserved. Forced to 0 by hardware.

Bit 4 = **LECIE** *Last Error Code Interrupt Enable*

- 0: ERRI bit will not be set when the error code in LEC[2:0] is set by hardware on error detection.
- 1: ERRI bit will be set when the error code in LEC[2:0] is set by hardware on error detection.

Bit 3 = reserved. Forced to 0 by hardware.

Bit 2 = **BOFIE** *Bus-Off Interrupt Enable*

- 0: ERRI bit will not be set when BOFF is set.
- 1: ERRI bit will be set when BOFF is set.

Bit 1 = **EPVIE** *Error Passive Interrupt Enable*

- 0: ERRI bit will not be set when EPVF is set.
- 1: ERRI bit will be set when EPVF is set.

Bit 0 = **EWGIE** *Error Warning Interrupt Enable*

- 0: ERRI bit will not be set when EWGF is set.
- 1: ERRI bit will be set when EWGF is set.

Transmit Error Counter Register (TECR)

Read only

Reset value: 0000 0000 (00h)

| | | | | | | | |
|------|------|------|------|------|------|------|------|
| 7 | | | | | | | 0 |
| TEC7 | TEC6 | TEC5 | TEC4 | TEC3 | TEC2 | TEC1 | TEC0 |

TEC[7:0] is the least significant byte of the 9-bit transmit error counter implementing part of the fault confinement mechanism of the CAN protocol.

Receive Error Counter Register (RECR)

Page: 00h — read only

Reset value: 0000 0000 (00h)

| | | | | | | | |
|------|------|------|------|------|------|------|------|
| 7 | | | | | | | 0 |
| REC7 | REC6 | REC5 | REC4 | REC3 | REC2 | REC1 | REC0 |

REC[7:0] is the receive error counter implementing part of the fault confinement mechanism of the CAN protocol. In case of an error during reception, this counter is incremented by 1 or by 8 depending on the error condition as defined by the CAN standard. After every successful reception the counter is decremented by 1 or reset to 120 if its value was higher than 128. When the counter value exceeds 127, the CAN controller enters the error passive state.

CAN diagnosis register (CDGR)

Read/ write

Reset value: 0000 1100 (0Ch)

| | | | | | | | |
|---|---|---|---|----|------|------|------|
| 7 | | | | | | | 0 |
| 0 | 0 | 0 | 0 | RX | SAMP | SILM | LBKM |

Note: All bits of this register are set and cleared by software.

Bit 3 = **RX** CAN Rx Signal

Read

Monitors the actual value of the **CAN_RX** Pin.

Bit 2 = **SAMP** Last Sample Point

Read

The value of the last sample point.

Bit 1 = **SILM** Silent Mode

Read/Set/Clear

0: normal operation

1: silent mode

Bit 0 = **LBKM** *Loop Back Mode*
Read/Set/Clear

- 0: loop back mode disabled
- 1: loop back mode enabled

CAN bit timing register 0 (CBTR0)

Read/ write

Reset value: 0000 0000 (00h)

| | | | | | | | |
|------|------|------|------|------|------|------|------|
| 7 | | | | | | | 0 |
| SJW1 | SJW0 | BRP5 | BRP4 | BRP3 | BRP2 | BRP1 | BRP0 |

Note: This register can only be accessed by the software when the CAN hardware is in configuration mode.

Bit 7:6 **SJW[1:0]** *Resynchronization Jump Width*

These bits define the maximum number of time quanta the CAN hardware is allowed to lengthen or shorten a bit to perform the resynchronization.

Resynchronization Jump Width = (SJW+1).

Bit 5:0 **BRP[5:0]** *Baud Rate Prescaler*

These bits define the length of a time quantum.

$t_q = (BRP+1)/f_{CPU}$

For more information on bit timing, please refer to [Bit timing](#).

CAN bit timing register 1 (CBTR1)

Read/ write

Reset value: 0001 0011 (23h)

| | | | | | | | |
|---|------|------|------|------|------|------|------|
| 7 | | | | | | | 0 |
| 0 | BS22 | BS21 | BS20 | BS13 | BS12 | BS11 | BS10 |

Bit 7 = reserved. Forced to 0 by hardware.

Bits 6:4 **BS2[2:0]** *Time Segment 2*

These bits define the number of time quanta in time segment 2.

Time segment 2 = (BS2+1)

Bits 3:0 **BS1[3:0]** *Time Segment 1*

These bits define the number of time quanta in time segment 1

Time segment 1 = (BS1+1)

For more information on bit timing, please refer to [Bit timing](#).**CAN filter page select register (CPSR)**

Read/ write

Reset value: 0000 0000 (00h)

| | | | | | | | |
|---|---|---|---|---|------|------|------|
| 7 | | | | | | | 0 |
| 0 | 0 | 0 | 0 | 0 | FPS2 | FPS1 | FPS0 |

Note: All bits of this register are set and cleared by software.

Bits 7:3 = reserved. Forced to 0 by hardware.

Bits 2:0 = **PS[2:0]** *Page Select*

Read/ write

This register contains the page number.

Table 86. Filter page selection

| PS[2:0] | Page selected |
|---------|-------------------------|
| 0 | Tx Mailbox 0 |
| 1 | Tx Mailbox 1 |
| 2 | Acceptance Filter 0:1 |
| 3 | Acceptance Filter 2:3 |
| 4 | Acceptance Filter 4:5 |
| 5 | Reserved |
| 6 | Configuration/Diagnosis |
| 7 | Receive FIFO |

17.8.3 Mailbox registers

This chapter describes the registers of the transmit and receive mailboxes. Refer to [Message storage](#) for detailed register mapping.

Transmit and receive mailboxes have the same registers except:

- MCSR register in a transmit mailbox is replaced by MFMI register in a receive mailbox.
- A receive mailbox is always write protected.
- A transmit mailbox is write enable only while empty, corresponding TME bit in the CTPR register set.

Mailbox control status register (MCSR)

Read/ write

Reset value: 0000 0000 (00h)

| | | | | | | | |
|---|---|------|------|------|------|------|------|
| 7 | | | | | | | 0 |
| 0 | 0 | TERR | ALST | TXOK | RQCP | ABRQ | TXRQ |

Bits 7:6 = reserved. Forced to 0 by hardware.

Bit 5 = **TERR** *Transmission Error*

Read

This bit is updated by hardware after each transmission attempt.

- 0: the previous transmission was successful
- 1: the previous transmission failed due to an error

Bit 4 = **ALST** *Arbitration Lost*

Read

This bit is updated by hardware after each transmission attempt.

- 0: the previous transmission was successful
- 1: the previous transmission failed due to an arbitration lost

Bit 3 = **TXOK** *Transmission OK*

Read

The hardware updates this bit after each transmission attempt.

- 0: the previous transmission failed
- 1: the previous transmission was successful

Note: This bit has the same value as the corresponding TXOKx bit in the CTSR register.

Bit 2 = **RQCP** *Request Completed*

Read/ clear

Set by hardware when the last request (transmit or abort) has been performed.

Cleared by software writing a "1" or by hardware on transmission request.

Note: This bit has the same value as the corresponding RQCPx bit of the CTSR register.

Clearing this bit clears all the status bits (TXOK, ALST and TERR) in the MCSR register and the RQCP and TXOK bits in the CTSR register.

Bit 1 = **ABRQ** *Abort Request for Mailbox*

Read/ set

Set by software to abort the transmission request for the corresponding mailbox.

Cleared by hardware when the mailbox becomes empty.

Setting this bit has no effect when the mailbox is not pending for transmission.

Bit 0 = **TXRQ** *Transmit Mailbox Request*

Read/ set

Set by software to request the transmission for the corresponding mailbox.

Cleared by hardware when the mailbox becomes empty.

Note: This register is implemented only in transmit mailboxes. In receive mailboxes, the MFMI register is mapped at this location.

Mailbox filter match index (MFMI)

This register is read only.

Reset value: 0000 0000 (00h)

| | | | | | | | |
|------|------|------|------|------|------|------|------|
| 7 | | | | | | | 0 |
| FMI7 | FMI6 | FMI5 | FMI4 | FMI3 | FMI2 | FMI1 | FMI0 |

Bits 7:0 = **FMI[7:0]** *Filter Match Index*

This register contains the index of the filter the message stored in the mailbox passed through. For more details on identifier filtering please refer to [Identifier filtering](#) - Filter match index paragraph.

Note: This register is implemented only in receive mailboxes. In transmit mailboxes, the MCSR register is mapped at this location.

Mailbox identifier registers (MIDR[3:0])

Read/ write

Reset value: Undefined

MIDR0

| | | | | | | | |
|---|-----|-----|--------|-------|-------|-------|-------|
| 7 | | | | | | | 0 |
| 0 | IDE | RTR | STID10 | STID9 | STID8 | STID7 | STID6 |

Bit 7 = Reserved. Forced to 0 by hardware.

Bit 6 = **IDE** *Extended Identifier*

This bit defines the identifier type of message in the mailbox.

0: standard identifier.

1: extended identifier.

Bit 5 = **RTR** *Remote Transmission Request*

0: data frame

1: remote frame

Bits 4:0 = **STID[10:6]** *Standard Identifier*

5 most significant bits of the standard part of the identifier.

MIDR1

| | | | | | | | |
|-------|-------|-------|-------|-------|-------|--------|--------|
| 7 | | | | | | | 0 |
| STID5 | STID4 | STID3 | STID2 | STID1 | STID0 | EXID17 | EXID16 |

Bits 7:2 = **STID[5:0]** *Standard Identifier*
6 least significant bits of the standard part of the identifier.

Bits 1:0 = **EXID[17:16]** *Extended Identifier*
2 most significant bits of the extended part of the identifier.

MIDR2

| | | | | | | | |
|--------|--------|--------|--------|--------|--------|-------|-------|
| 7 | | | | | | | 0 |
| EXID15 | EXID14 | EXID13 | EXID12 | EXID11 | EXID10 | EXID9 | EXID8 |

Bits 7:0 = **EXID[15:8]** *Extended Identifier*
Bits 15 to 8 of the extended part of the identifier.

MIDR3

| | | | | | | | |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 7 | | | | | | | 0 |
| EXID7 | EXID6 | EXID5 | EXID4 | EXID3 | EXID2 | EXID1 | EXID0 |

Bits 7:1 = **EXID[6:0]** *Extended Identifier*
6 least significant bits of the extended part of the identifier.

Mailbox data length control register (MDLC)

Read / write

Reset value: xxxx xxxx (xxh)

| | | | | | | | |
|---|---|---|---|------|------|------|------|
| 7 | | | | | | | 0 |
| 0 | 0 | 0 | 0 | DLC3 | DLC2 | DLC1 | DLC0 |

Note: All bits of this register is write protected when the mailbox is not in empty state.

Bit 7 = Reserved, must be kept cleared.

Bits 6:4 = Reserved, forced to 0 by hardware.

Bits 3:0 = **DLC[3:0]** *Data Length Code*
This field defines the number of data bytes a data frame contains or a remote frame request.

Mailbox data registers (MDAR[7:0])

Read / Write

Reset value: Undefined

| | | | | | | | |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 7 | | | | | | | 0 |
| DATA7 | DATA6 | DATA5 | DATA4 | DATA3 | DATA2 | DATA1 | DATA0 |

Note: All bits of this register are write protected when the mailbox is not in empty state.

Bits 7:0 = **DATA[7:0]** *Data*

A data byte of the message. A message can contain from 0 to 8 data bytes.

17.8.4 CAN filter registers

CAN filter configuration register 0 (CFCR0)

Read / write

Reset value: 0000 0000 (00h)

| | | | | | | | |
|---|-------|-------|-------|---|-------|-------|-------|
| 7 | | | | | | | 0 |
| 0 | FSC11 | FSC10 | FACT1 | 0 | FSC01 | FSC00 | FACT0 |

Note:

All bits of this register are set and cleared by software.

To modify the FFAx and FSCx bits, the beCAN must be in INIT mode.

Bit 7 = reserved. Forced to 0 by hardware.

Bits 6:5 = **FSC1[1:0]** *Filter Scale Configuration*

These bits define the scale configuration of filter 1.

Bit 4 = **FACT1** *Filter Active*

The software sets this bit to activate filter 1. To modify the filter 1 registers (CF1R[7:0]), the FACT1 bit must be cleared.

0: filter 1 is not active

1: filter 1 is active

Bit 3 = reserved. Forced to 0 by hardware.

Bits 2:1 = **FSC0[1:0]** *Filter Scale Configuration*

These bits define the scale configuration of filter 0.

Bit 0 = **FACT0** *Filter Active*

The software sets this bit to activate filter 0. To modify the filter 0 registers (CF0R[0:7]), the FACT0 bit must be cleared.

0: filter 0 is not active

1: filter 0 is active

CAN filter configuration register 1 (CFCR1)

Read/ write

Reset value: 0000 0000 (00h)

| | | | | | | | |
|---|-------|-------|-------|---|-------|-------|-------|
| 7 | | | | | | | 0 |
| 0 | FSC31 | FSC30 | FACT3 | 0 | FSC21 | FSC20 | FACT2 |

Note: All bits of this register are set and cleared by software.

Bit 7 = reserved. Forced to 0 by hardware.

Bits 6:5 = **FSC3[1:0]** *Filter Scale Configuration*

These bits define the scale configuration of filter 3.

Bit 4 = **FACT3** *Filter Active*

The software sets this bit to activate filter 3. To modify the filter 3 registers (CF3R[0:7]) the FACT3 bit must be cleared.

- 0: filter 3 is not active
- 1: filter 3 is active

Bit 3 = reserved. Forced to 0 by hardware.

Bits 2:1 = **FSC2[1:0]** *Filter Scale Configuration*

These bits define the scale configuration of Filter 2.

Bit 0 = **FACT2** *Filter Active*

The software sets this bit to activate filter 2. To modify the filter 2 registers (CF2R[0:7]), the FACT2 bit must be cleared.

- 0: filter 2 is not active
- 1: filter 2 is active

CAN filter configuration register 2(CFCR2)

Read/ write

Reset value: 0000 0000 (00h)

| | | | | | | | |
|---|-------|-------|-------|---|-------|-------|-------|
| 7 | | | | | | | 0 |
| 0 | FSC51 | FSC50 | FACT5 | 0 | FSC41 | FSC40 | FACT4 |

Note: All bits of this register are set and cleared by software.

Bit 7 = Reserved. Forced to 0 by hardware.

Bits 6:5 = **FSC5[1:0]** *Filter Scale Configuration*

These bits define the scale configuration of filter 5.

Bit 4 = **FACT5** *Filter Active*

The software sets this bit to activate filter 5. To modify the filter 5 registers (CF5R[0:7]) the FACT5 bit must be cleared.

- 0: filter 5 is not active
- 1: filter 5 is active

Bit 3 = Reserved. Forced to 0 by hardware.

Bits 2:1 = **FSC4[1:0]** *Filter Scale Configuration*

These bits define the scale configuration of filter 4.

Bit 0 = **FACT4** *Filter Active*

The software sets this bit to activate filter 4. To modify the filter 4 registers (CF4R[0:7]), the FACT4 bit must be cleared.

- 0: filter 4 is not active
- 1: filter 4 is active

CAN filter mode register (CFMR0)

Read/ write

Reset value: 0000 0000 (00h)

| | | | | | | | |
|------|------|------|------|------|------|------|------|
| 7 | | | | | | | 0 |
| FMH3 | FML3 | FMH2 | FML2 | FMH1 | FML1 | FMH0 | FML0 |

Note: All bits of this register are set and cleared by software.

Bit 7 = **FMH3** *Filter Mode High*

Mode of the high registers of filter 3.

0: high registers are in mask mode

1: high registers are in identifier list mode

Bit 6 = **FML3** *Filter Mode Low*

Mode of the low registers of filter 3.

0: low registers are in mask mode

1: low registers are in identifier list mode

Bit 5 = **FMH2** *Filter Mode High*

Mode of the high registers of filter 2.

0: high registers are in mask mode

1: high registers are in identifier list mode

Bit 4 = **FML2** *Filter Mode Low*

Mode of the low registers of filter 2.

0: low registers are in mask mode

1: low registers are in identifier list mode

Bit 3 = **FMH1** *Filter Mode High*

Mode of the high registers of filter 1.

0: high registers are in mask mode

1: high registers are in identifier list mode

Bit 2 = **FML1** *Filter Mode Low*

Mode of the low registers of filter 1.

0: low registers are in mask mode

1: low registers are in identifier list mode

Bit 1 = **FMH0** *Filter Mode High*

Mode of the high registers of filter 0.

0: high registers are in mask mode

1: high registers are in identifier list mode

Bit 0 = **FML0** *Filter Mode Low*

Mode of the low registers of filter 0.

0: low registers are in mask mode

1: low registers are in identifier list mode

CAN filter mode register (CFMR1)

Read/ write

Reset value: 0000 0000 (00h)

| | | | | | | | |
|---|---|---|---|------|------|------|------|
| 7 | | | | | | | 0 |
| 0 | 0 | 0 | 0 | FMH5 | FML5 | FMH4 | FML4 |

Note: All bits of this register are set and cleared by software.

Bits 7:4 = Reserved. Forced to 0 by hardware.

Bit 3 = **FMH5** Filter Mode High

Mode of the high registers of filter 5.

0: high registers are in mask mode

1: high registers are in identifier list mode

Bit 2 = **FML5** Filter Mode Low

Mode of the low registers of filter 5.

0: low registers are in mask mode

1: low registers are in identifier list mode

Bit 1 = **FMH4** Filter Mode High

Mode of the high registers of filter 4.

0: high registers are in mask mode

1: high registers are in identifier list mode

Bit 0 = **FML4** Filter Mode Low

Mode of the low registers of filter 4.

0: low registers are in mask mode

1: low registers are in identifier list mode

Filter x register[7:0] (CFxR[7:0])

Read/ write

Reset value: Undefined

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 7 | | | | | | | 0 |
| FB7 | FB6 | FB5 | FB4 | FB3 | FB2 | FB1 | FB0 |

In all configurations:

Bits 7:0 = **FB[7:0]** Filter Bits

Identifier

Each bit of the register specifies the level of the corresponding bit of the expected identifier.

0: dominant bit is expected

1: recessive bit is expected

Mask

Each bit of the register specifies whether the bit of the associated identifier register must match with the corresponding bit of the expected identifier or not.

0: don't care, the bit is not used for the comparison

1: must match, the bit of the incoming identifier must have the same level as specified in the corresponding identifier register of the filter.

Note: Each filter x is composed of 8 registers, $CFxR[7:0]$. Depending on the scale and mode configuration of the filter the function of each register can differ. For the filter mapping, functions description and mask registers association, refer to Section [Identifier filtering](#).

A mask/identifier register in mask mode has the same bit mapping as in identifier list mode.

Note: To modify these registers, the corresponding FACT bit in the CFCR register must be cleared.

Figure 115. CAN register mapping

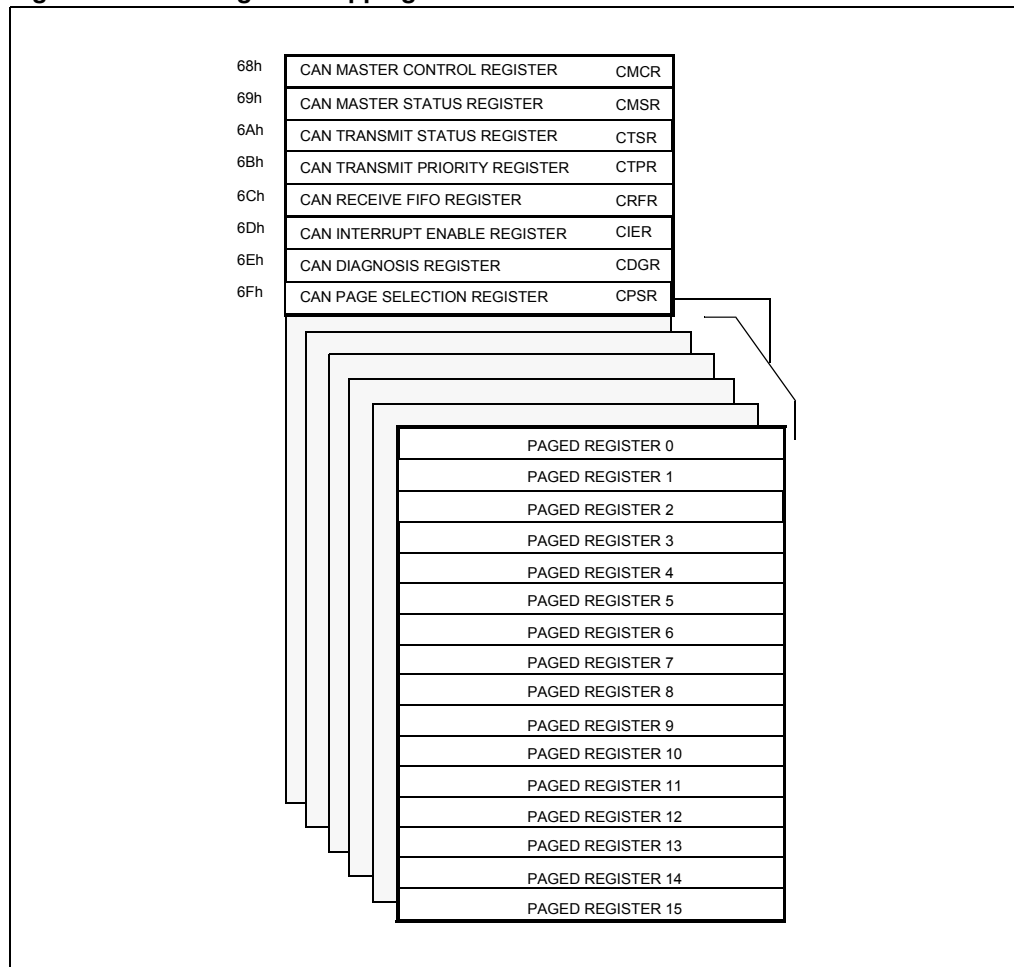


Figure 116. Page mapping for CAN

| | PAGE 0 | PAGE 1 | PAGE 2 | PAGE 3 | PAGE 4 |
|-----|-------------------------|--------------|-----------------------|-----------------------|-----------------------|
| 70h | MCSR | MCSR | CF0R0 | CF2R0 | CF4R0 |
| 71h | MDLC | MDLC | CF0R1 | CF2R1 | CF4R1 |
| 72h | MIDR0 | MIDR0 | CF0R2 | CF2R2 | CF4R2 |
| 73h | MIDR1 | MIDR1 | CF0R3 | CF2R3 | CF4R3 |
| 74h | MIDR2 | MIDR2 | CF0R4 | CF2R4 | CF4R4 |
| 75h | MIDR3 | MIDR3 | CF0R5 | CF2R5 | CF4R5 |
| 76h | MDAR0 | MDAR0 | CF0R6 | CF2R6 | CF4R6 |
| 77h | MDAR1 | MDAR1 | CF0R7 | CF2R7 | CF4R7 |
| 78h | MDAR2 | MDAR2 | CF1R0 | CF3R0 | CF5R0 |
| 79h | MDAR3 | MDAR3 | CF1R1 | CF3R1 | CF5R1 |
| 7Ah | MDAR4 | MDAR4 | CF1R2 | CF3R2 | CF5R2 |
| 7Bh | MDAR5 | MDAR5 | CF1R3 | CF3R3 | CF5R3 |
| 7Ch | MDAR6 | MDAR6 | CF1R4 | CF3R4 | CF5R4 |
| 7Dh | MDAR7 | MDAR7 | CF1R5 | CF3R5 | CF5R5 |
| 7Eh | MTSLR | MTSLR | CF1R6 | CF3R6 | CF5R6 |
| 7Fh | MTSHR | MTSHR | CF1R7 | CF3R7 | CF5R7 |
| | Tx Mailbox 0 | Tx Mailbox 1 | Acceptance Filter 0:1 | Acceptance Filter 2:3 | Acceptance Filter 4:5 |
| | PAGE 6 | PAGE 7 | | | |
| 70h | CESR | MFMI | | | |
| 71h | CEIER | MDLC | | | |
| 72h | TECR | MIDR0 | | | |
| 73h | RECR | MIDR1 | | | |
| 74h | BTCR0 | MIDR2 | | | |
| 75h | BTCR1 | MIDR3 | | | |
| 76h | Reserved | MDAR0 | | | |
| 77h | Reserved | MDAR1 | | | |
| 78h | CFMR0 | MDAR2 | | | |
| 79h | CFMR1 | MDAR3 | | | |
| 7Ah | CFCR0 | MDAR4 | | | |
| 7Bh | CFCR1 | MDAR5 | | | |
| 7Ch | CFCR2 | MDAR6 | | | |
| 7Dh | Reserved | MDAR7 | | | |
| 7Eh | Reserved | MTSLR | | | |
| 7Fh | Reserved | MTSHR | | | |
| | Configuration/Diagnosis | Receive FIFO | | | |

Table 87. beCAN control and status page - register map and reset values

| Address (Hex.) | Register Name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------------|----------------------|------------|-----------|------------|------------|-------------|------------|-------------|------------|
| 68h | CMCR Reset value | 0 | ABOM 0 | AWUM 0 | NART 0 | RFLM 0 | TXFP 0 | SLEEP 1 | INRQ 0 |
| 69h | CMSR Reset value | 0 | 0 | REC 0 | TRAN 0 | WKUI 0 | ERRI 0 | SLAK 1 | INAK 0 |
| 6Ah | CTSR Reset value | 0 | 0 | TXOK1 0 | TXOK0 0 | 0 | 0 | RQCP1 0 | RQCP0 0 |
| 6Bh | CTPR Reset value | 0 | LOW1 0 | LOW0 0 | 1 | TME1 1 | TME0 1 | 0 | CODE0 0 |
| 6Ch | CRFR Reset value | 0 | 0 | RFOM 0 | FOVR 0 | FULL 0 | 0 | FMP1 0 | FMP0 0 |
| 6Dh | CIER Reset value | WКУIE 0 | 0 | 0 | 0 | FOVIE0 0 | FFIE0 0 | FMPIE0 0 | TMEIE 0 |
| 6Eh | CDGR Reset value | 0 | 0 | 0 | 0 | RX 1 | SAMP 1 | SILM 0 | LBKM 0 |
| 6Fh | CFPSR Reset value | 0 | 0 | 0 | 0 | 0 | FPS2 0 | FPS1 0 | FPS0 0 |

Table 88. beCAN mailbox pages - register map and reset values

| Address (Hex.) | Register name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----------------|--------------------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| 70h Receive | MFMI Reset value | FMI7 0 | FMI6 0 | FMI5 0 | FMI4 0 | FMI3 0 | FMI2 0 | FMI1 0 | FMI0 0 |
| 70h Transmit | MCSR Reset value | 0 | 0 | TERR 0 | ALST 0 | TXOK 0 | RQCP 0 | ABRQ 0 | TXRQ 0 |
| 71h | MDLC Reset value | 0 x | x | x | x | DLC3 x | DLC2 x | DLC1 x | DLC0 x |
| 72h | MIDR0 Reset value | x | IDE x | RTR x | STID10 x | STID9 x | STID8 x | STID7 x | STID6 x |
| 73h | MIDR1 Reset value | STID5 x | STID4 x | STID3 x | STID2 x | STID1 x | STID0 x | EXID17 x | EXID16 x |
| 74h | MIDR2 Reset value | EXID15 x | EXID14 x | EXID13 x | EXID12 x | EXID11 x | EXID10 x | EXID9 x | EXID8 x |
| 75h | MIDR3 Reset value | EXID7 x | EXID6 x | EXID5 x | EXID4 x | EXID3 x | EXID2 x | EXID1 x | EXID0 x |
| 76h:7Dh | MDAR[0:7] Reset value | MDAR7 x | MDAR6 x | MDAR5 x | MDAR4 x | MDAR3 x | MDAR2 x | MDAR1 x | MDAR0 x |

Table 88. beCAN mailbox pages - register map and reset values (continued)

| Address (Hex.) | Register name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------------|----------------------|-------------|-------------|-------------|-------------|-------------|-------------|------------|------------|
| 7Eh | MTSLR Reset value | TIME7 x | TIME6 x | TIME5 x | TIME4 x | TIME3 x | TIME2 x | TIME1 x | TIME0 x |
| 7Fh | MTSHR Reset value | TIME15 x | TIME14 x | TIME13 x | TIME12 x | TIME11 x | TIME10 x | TIME9 x | TIME8 x |

Table 89. beCAN filter configuration page - register map and reset values

| Address (Hex.) | Register name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------------|----------------------|------------|------------|------------|------------|-----------|------------|------------|------------|
| 70h | CESR Reset value | 0 | LEC2 0 | LEC1 0 | LEC0 0 | 0 | BOFF 0 | EPVVF 0 | EWGF 0 |
| 71h | CEIER Reset value | ERRIE 0 | 0 | 0 | LECIE 0 | 0 | BOFIE 0 | EPVIE 0 | EWGIE 0 |
| 72h | TECR Reset value | TEC7 0 | TEC6 0 | TEC5 0 | TEC4 0 | TEC3 0 | TEC2 0 | TEC1 0 | TEC0 0 |
| 73h | RECR Reset value | REC7 0 | REC6 0 | REC5 0 | REC4 0 | REC3 0 | REC2 0 | REC1 0 | REC0 0 |
| 74h | CBTR0 Reset value | SJW1 0 | SJW0 0 | BRP5 0 | BRP4 0 | BRP3 0 | BRP2 0 | BRP1 0 | BRP0 0 |
| 75h | CBTR1 Reset value | 0 | BS22 0 | BS21 1 | BS20 0 | BS13 0 | BS12 0 | BS11 1 | BS10 1 |
| 76h | Reserved | x | x | x | x | x | x | x | x |
| 77h | Reserved | x | x | x | x | x | x | x | x |
| 78h | CFMR0 Reset value | FMH3 0 | FML3 0 | FMH2 0 | FML2 0 | FMH1 0 | FML1 0 | FMH0 0 | FML0 0 |
| 79h | CFMR1 Reset value | 0 | 0 | 0 | 0 | FMH5 0 | FML5 0 | FMH4 0 | FML4 0 |
| 7Ah | CFCR0 Reset value | FFA1 0 | FSC11 0 | FSC10 0 | FACT1 0 | FFA0 0 | FSC01 0 | FSC00 0 | FACT0 0 |
| 7Bh | CFCR1 Reset value | FFA3 0 | FSC31 0 | FSC30 0 | FACT3 0 | FFA2 0 | FSC21 0 | FSC20 0 | FACT2 0 |
| 7Ch | CFCR2 Reset value | FFA5 0 | FSC51 0 | FSC50 0 | FACT5 0 | FFA4 0 | FSC41 0 | FSC40 0 | FACT4 0 |

18 10-bit A/D converter (ADC)

18.1 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.

18.2 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 117](#).

18.3 Functional description

18.3.1 Digital A/D conversion result

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.

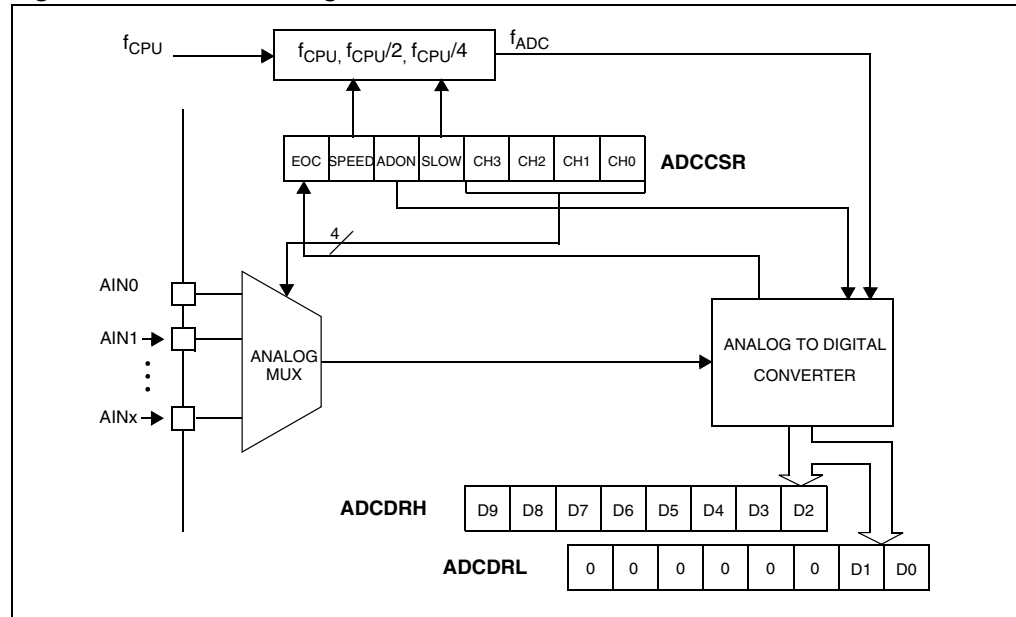
If the input voltage (V_{AIN}) is greater than V_{DDA} (high-level voltage reference) then the conversion result is FFh in the ADCDRH register and 03h in the ADCDRL register (without overflow indication).

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 the electrical characteristics section.

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.

Figure 117. ADC block diagram



18.3.2 A/D conversion

The analog input ports must be configured as input, no pull-up, no interrupt. Refer to the [Chapter 8: 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.

ADC conversion mode:

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 resets the EOC bit.

To read the 10 bits, perform the following steps:

1. Poll EOC bit
2. Read the ADCDRL register
3. Read the ADCDRH register. This clears EOC automatically.

To read only 8 bits, perform the following steps:

1. Poll EOC bit
2. Read the ADCDRH register. This clears EOC automatically.

18.3.3 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.

18.3.4 ADCDR consistency

If an End Of Conversion event occurs after software has read the ADCDRLSB but before it has read the ADCDRMSB, there would be a risk that the two values read would belong to different samples.

To guarantee consistency:

- The ADCDRL and the ADCDRH registers are locked when the ADCCRL is read
- The ADCDRL and the ADCDRH registers are unlocked when the ADCDRH register is read or when ADON is reset.

This is important, as the ADCDR register will not be updated until the ADCDRH register is read.

18.4 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 90. Effect of low power modes on ADC

| Mode | Description |
|------|--|
| WAIT | No effect on A/D converter |
| HALT | A/D converter disabled. After wakeup from Halt mode, the A/D converter requires a stabilization time t_{STAB} (see Electrical Characteristics) before accurate conversions can be performed. |

18.5 Interrupts

None.

18.6 Register description

18.6.1 Control/status register (ADCCSR)

Read/ write (except bit 7 read only)

Reset value: 0000 0000 (00h)

| | | | | | | | |
|-----|-------|------|------|-----|-----|-----|-----|
| 7 | | | | | | | 0 |
| EOC | SPEED | ADON | SLOW | CH3 | CH2 | CH1 | CH0 |

Bit 7 = **EOC** *End of Conversion*

This bit is set by hardware. It is cleared by software reading the ADCDRH register or writing to any bit of the ADCCSR register.

- 0: conversion is not complete
- 1: conversion complete

Bit 6 = **SPEED** *A/D clock selection*

This bit is set and cleared by software.

Table 91. A/D clock selection

| f_{ADC} | Slow | Speed |
|---|------|-------|
| $f_{CPU}/2$ | 0 | 0 |
| f_{CPU} (where $f_{CPU} \leq 4$ MHz) | | 1 |
| $f_{CPU}/4$ | 1 | 0 |
| $f_{CPU}/2$ (same frequency as SLOW = 0, SPEED = 0) | | 1 |

Bit 5 = **ADON** *A/D Converter on*

This bit is set and cleared by software.

- 0: disable ADC and stop conversion
- 1: enable ADC and start conversion

Bit 4 = **SLOW** *A/D Clock Selection*

This bit is set and cleared by software. It works together with the SPEED bit. Refer to [Table 91](#).

Bits 3:0 = **CH[3:0]** *Channel Selection*

These bits are set and cleared by software. They select the analog input to convert.

Table 92. ADC channel selection

| Channel pin ⁽¹⁾ | CH3 | CH2 | CH1 | CH0 |
|----------------------------|-----|-----|-----|-----|
| AIN0 | 0 | 0 | 0 | 0 |
| AIN1 | | | | 1 |
| AIN2 | | | 1 | 0 |
| AIN3 | | | | 1 |
| AIN4 | | 1 | 0 | 0 |
| AIN5 | | | | 1 |
| AIN6 | | | 1 | 0 |
| AIN7 | | | | 1 |

Table 92. ADC channel selection (continued)

| Channel pin ⁽¹⁾ | CH3 | CH2 | CH1 | CH0 |
|----------------------------|-----|-----|-----|-----|
| AIN8 | 1 | 0 | 0 | 0 |
| AIN9 | | | | 1 |
| AIN10 | | | 1 | 0 |
| AIN11 | | | | 1 |
| AIN12 | | 1 | 0 | 0 |
| AIN13 | | | | 1 |
| AIN14 | | | 1 | 0 |
| AIN15 | | | | 1 |

1. The number of channels is device dependent. Refer to the device pinout description.

18.6.2 Data register (ADCDRH)

Read only

Reset value: 0000 0000 (00h)

| | | | | | | | |
|----|----|----|----|----|----|----|----|
| 7 | | | | | | | 0 |
| D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 |

Bits 7:0 = **D[9:2]** MSB of Analog Converted Value

18.6.3 Data register (ADCDRL)

Read only

0000 0000 (00h)

| | | | | | | | |
|---|---|---|---|---|---|----|----|
| 7 | | | | | | | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | D1 | D0 |

Bits 7:2 = Reserved. Forced by hardware to 0.

Bits 1:0 = **D[1:0]** LSB of Analog Converted Value

Table 93. ADC register map and reset values

| Address (Hex.) | Register label | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------------|-----------------------|----------|------------|-----------|-----------|----------|----------|----------|----------|
| 45h | ADCCSR Reset value | EOC 0 | SPEED 0 | ADON 0 | SLOW 0 | CH3 0 | CH2 0 | CH1 0 | CH0 0 |
| 46h | ADCDRH Reset value | D9 0 | D8 0 | D7 0 | D6 0 | D5 0 | D4 0 | D3 0 | D2 0 |
| 47h | ADCDRL Reset value | 0 0 | 0 0 | 0 0 | 0 0 | 0 0 | 0 0 | D1 0 | D0 0 |

19 Instruction set

19.1 CPU addressing modes

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

Table 94. Addressing mode groups

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

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

- Long addressing mode is more powerful because it can use the full 64 Kbyte address space, however it uses more bytes and more CPU cycles.
- Short addressing mode is less powerful because it can generally only access page zero (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 95. 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 |

Table 95. CPU addressing mode overview (continued)

| Mode | | | Syntax | Destination | Pointer address (hex.) | Pointer size (hex.) | Length (bytes) |
|----------|----------|----------|----------------------|-------------|------------------------|---------------------|----------------|
| Short | Indirect | Indexed | ld A, ([\$10],X) | 00..1FE | 00..FF | byte | + 2 |
| 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 |

19.1.1 Inherent

All Inherent instructions consist of a single byte. The opcode fully specifies all the required information for the CPU to process the operation.

| Inherent 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 |

19.1.2 Immediate

Immediate instructions have 2 bytes, the first byte contains the opcode, the second byte contains the operand value.

| Immediate instruction | Function |
|-----------------------|-----------------------|
| LD | Load |
| CP | Compare |
| BCP | Bit compare |
| AND, OR, XOR | Logical operations |
| ADC, ADD, SUB, SBC | Arithmetic operations |

19.1.3 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.

19.1.4 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 indirect 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.

19.1.5 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.

19.1.6 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 96. Instructions supporting direct, indexed, indirect and indirect indexed addressing (part 1)

| Long and short instructions | Function |
|-----------------------------|--|
| LD | Load |
| CP | Compare |
| AND, OR, XOR | Logical Operations |
| ADC, ADD, SUB, SBC | Arithmetic Additions/Subtractions operations |
| BCP | Bit Compare |

Table 97. Instructions supporting direct, indexed, indirect and indirect indexed addressing (part 2)

| Short instructions only | Function |
|-------------------------|-----------------------|
| CLR | Clear |
| INC, DEC | Increment/Decrement |
| TNZ | Test Negative or Zero |
| CPL, NEG | 1 or 2 Complement |

Table 97. Instructions supporting direct, indexed, indirect and indirect indexed addressing (part 2) (continued)

| Short instructions only | Function |
|-------------------------|------------------------------|
| 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 |

19.1.7 Relative mode (direct, indirect)

This addressing mode is used to modify the PC register value, by adding an 8-bit signed offset to it.

| Available relative direct/indirect instructions | 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.

19.2 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 98. Instruction groups

| Description | Instruction | | | | | | | |
|---------------------------------|-------------|------|-----|-----|-----|--|--|--|
| Load and transfer | LD | CLR | | | | | | |
| Stack operation | PUSH | POP | RSP | | | | | |
| Increment/ decrement | INC | DEC | | | | | | |
| Compare and tests | CP | TNZ | BCP | | | | | |
| 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 | | | |

Table 98. Instruction groups

| Description | Instruction | | | | | | | |
|----------------------------------|-------------|-----|------|------|------|-------|-----|-----|
| | SLL | SRL | SRA | RLC | RRC | SWAP | SLA | |
| Shift and rotates | | | | | | | | |
| Unconditional jump or call | JRA | JRT | JRF | JP | CALL | CALLR | NOP | RET |
| Conditional branch | JRxx | | | | | | | |
| Interrupt management | TRAP | WFI | HALT | IRET | | | | |
| Condition code flag modification | SIM | RIM | SCF | RCF | | | | |

19.2.1 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.

| Memo | 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 |

| Memo | Description | Function/example | Dst | Src | I1 | H | I0 | N | Z | 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 >= | | | | | | | | |
| 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 |

| Memo | Description | Function/example | Dst | Src | I1 | H | I0 | N | Z | C |
|------|------------------------|---------------------|--------|------------|----|---|----|---|---|---|
| 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 | Substract 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 |
| LL | 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 | Substraction | 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 | |

20 Electrical characteristics

20.1 Parameter conditions

Unless otherwise specified, all voltages are referred to V_{SS} .

20.1.1 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 $T_A = 25^\circ\text{C}$ and $T_A = T_{A\text{max}}$ (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 ($\text{mean} \pm 3\Sigma$).

20.1.2 Typical values

Unless otherwise specified, typical data is based on $T_A = 25^\circ\text{C}$, $V_{DD} = 5\text{V}$ (for the $4.5\text{V} \leq V_{DD} \leq 5.5\text{V}$ voltage range). They are given only as design guidelines and are not tested.

Typical ADC accuracy values are determined by characterization of a batch of samples from a standard diffusion lot over the full temperature range, where 95% of the devices have an error less than or equal to the value indicated ($\text{mean} \pm 2\Sigma$).

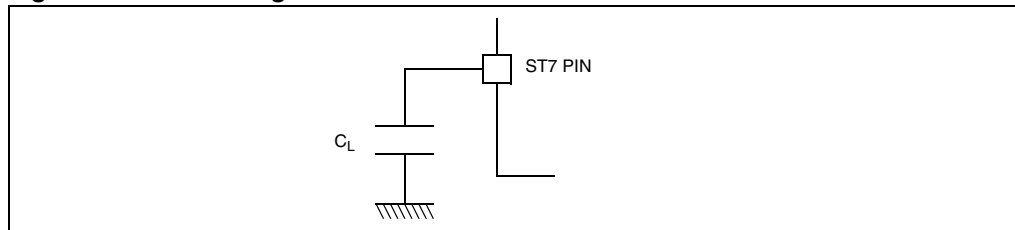
20.1.3 Typical curves

Unless otherwise specified, all typical curves are given only as design guidelines and are not tested.

20.1.4 Loading capacitor

The loading conditions used for pin parameter measurement are shown in [Figure 118](#).

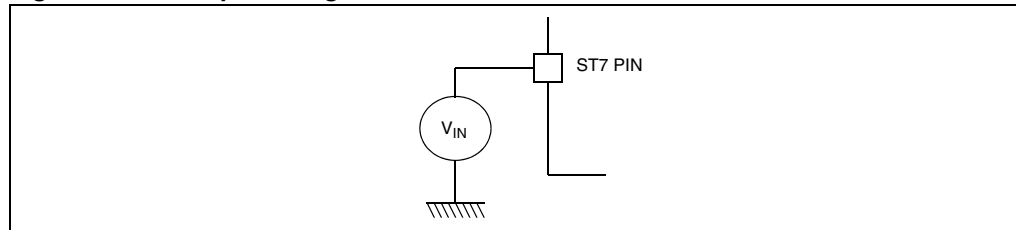
Figure 118. Pin loading conditions



20.1.5 Pin input voltage

The input voltage measurement on a pin of the device is described in [Figure 119](#).

Figure 119. Pin input voltage



20.2 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.

20.2.1 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} | Input voltage on any pin ⁽¹⁾⁽²⁾ | $V_{SS} - 0.3$ to $V_{DD} + 0.3$ | |
| $ \Delta V_{DDx} $ and $ \Delta V_{SSx} $ | Variations between different digital power pins | 50 | mV |
| $ V_{SSA} - V_{SSx} $ | Variations between digital and analog ground pins | 50 | |
| $V_{ESD(HBM)}$ | Electro-static discharge voltage (Human Body Model) | see Section 20.8.3: Absolute maximum ratings (electrical sensitivity) | |
| $V_{ESD(MM)}$ | Electro-static discharge voltage (Machine Model) | | |

1. Directly connecting the \overline{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: 4.7k Ω for \overline{RESET} , 10k Ω for I/Os). Unused I/O pins must be tied in the same way to V_{DD} or V_{SS} according to their reset configuration.

2. $I_{INJ(PIN)}$ must never be exceeded. This is implicitly insured 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.

20.2.2 Current characteristics

| Symbol | Ratings | Maximum value | Unit |
|--------------------------------------|---|---------------|------|
| I_{VDD} | Total current into V_{DD} power lines (source) ⁽¹⁾ | 150 | mA |
| I_{VSS} | Total current out of V_{SS} ground lines (sink) ⁽¹⁾ | | |
| I_{IO} | Output current sunk by any standard I/O and control pin | 25 | |
| | Output current sunk by any high sink I/O pin | 50 | |
| | 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{RESET} pin | | |
| | Injected current on OSC1 and OSC2 pins | | |
| | Injected current on PB3 (on Flash devices) | +5 | |
| | Injected current on any other pin ⁽⁴⁾ | ±5 | |
| $\Sigma I_{INJ(PIN)}$ ⁽²⁾ | Total injected current (sum of all I/O and control pins) ⁽⁴⁾ | ±25 | |

- All power (V_{DD}) and ground (V_{SS}) lines must always be connected to the external supply.
- $I_{INJ(PIN)}$ must never be exceeded. This is implicitly insured 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.
- Negative injection disturbs the analog performance of the device. See note in [Section 20.13: 10-bit ADC characteristics](#).
- 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.

20.2.3 Thermal characteristics

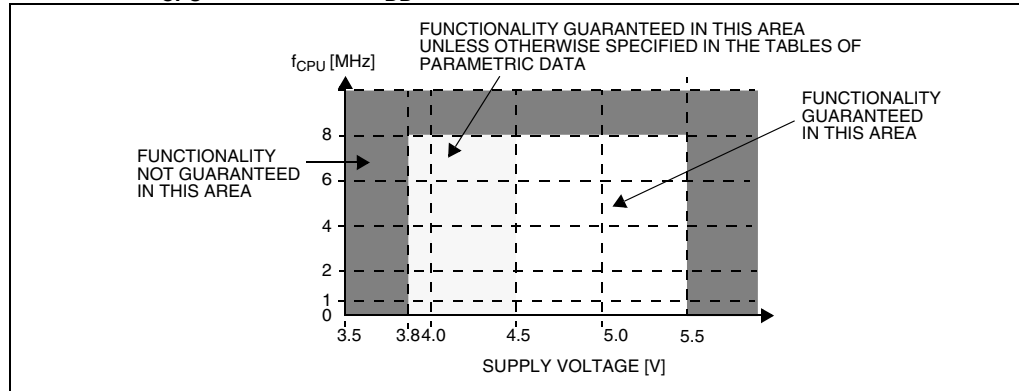
| Symbol | Ratings | Value | Unit |
|-----------|---|-------------|------|
| T_{STG} | Storage temperature range | -65 to +150 | °C |
| T_J | Maximum junction temperature (see Section 21.3: Thermal characteristics) | | |

20.3 Operating conditions

20.3.1 General operating conditions

| Symbol | Parameter | Conditions | Min | Max | Unit |
|------------------|--|---|-----|-----|------|
| f_{CPU} | Internal clock frequency | | 0 | 8 | MHz |
| V_{DD} | Extended operating voltage | No Flash write/ erase. Analog parameters not guaranteed. | 3.8 | 4.5 | V |
| | Standard operating voltage | | 4.5 | 5.5 | |
| | Operating voltage for flash write/ erase | $V_{\text{PP}} = 11.4$ to 12.6V | | | |
| T_{A} | Ambient temperature range | A Suffix version | -40 | 85 | °C |
| | | C Suffix version | | 125 | |

Figure 120. f_{CPU} maximum vs V_{DD}



20.3.2 Operating conditions with low voltage detector (LVD)

Subject to general operating conditions for T_{A} .

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|-----------------------|--|---|--------------------|-----|---------------------|------|
| $V_{\text{IT+}(LVD)}$ | Reset release threshold (V_{DD} rise) | | 4.0 ⁽¹⁾ | 4.2 | 4.5 | V |
| $V_{\text{IT-}(LVD)}$ | Reset generation threshold (V_{DD} fall) | | 3.8 | 4.0 | 4.25 ⁽¹⁾ | |
| $V_{\text{hys}(LVD)}$ | LVD voltage threshold hysteresis ⁽¹⁾ | $V_{\text{IT+}(LVD)} - V_{\text{IT-}(LVD)}$ | 150 | 200 | 250 | mV |
| V_{tPOR} | V_{DD} rise time rate ⁽¹⁾ | | 6 | | | µs/V |
| | | | | | 100 | ms/V |
| $t_{\text{g}(VDD)}$ | V_{DD} glitches filtered (not detected) by LVD ⁽¹⁾ | Measured at $V_{\text{IT-}(LVD)}$ | | | 40 | ns |

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

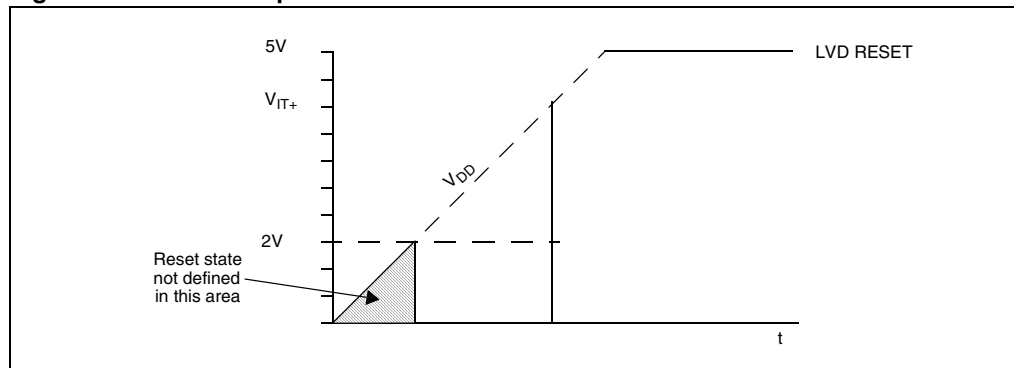
20.3.3 Auxiliary voltage detector (AVD) thresholds

Subject to general operating conditions for T_A .

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|-----------------|---|-------------------------------|--------------------|-----|---------------------|------|
| $V_{IT+(AVD)}$ | 1 \Rightarrow 0 AVDF flag toggle threshold (V_{DD} rise) | | 4.4 ⁽¹⁾ | 4.6 | 4.9 | V |
| $V_{IT-(AVD)}$ | 0 \Rightarrow 1 AVDF flag toggle threshold (V_{DD} fall) | | 4.2 | 4.4 | 4.65 ⁽¹⁾ | |
| $V_{hys(AVD)}$ | AVD voltage threshold hysteresis | $V_{IT+(AVD)} - V_{IT-(AVD)}$ | | 250 | | mV |
| ΔV_{IT} | Voltage drop between AVD flag set and LVD reset activated | $V_{IT-(AVD)} - V_{IT-(LVD)}$ | | 450 | | |

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

Figure 121. LVD startup behavior



Note: When the LVD is enabled, the MCU reaches its authorized operating voltage from a reset state. However, in some devices, the reset signal may be undefined until V_{DD} is approximately 2V. As a consequence, the I/Os may toggle when V_{DD} is below this voltage.

Because Flash write access is impossible below this voltage, the Flash memory contents will not be corrupted.

20.4 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 get the total device consumption, the two current values must be added (except for HALT mode for which the clock is stopped).

Table 99. Supply current consumption

| Symbol | Parameter | Conditions | Flash devices | | ROM devices | | Unit |
|--|---|---|--------------------|--------------------|--------------------|--------------------|------|
| | | | Typ ⁽¹⁾ | Max ⁽²⁾ | Typ ⁽¹⁾ | Max ⁽²⁾ | |
| I _{DD} | Supply current in RUN mode ⁽³⁾ | f _{OSC} = 2 MHz, f _{CPU} = 1 MHz | 1.8 | 3 | 1.1 | 2 | mA |
| | | f _{OSC} = 4 MHz, f _{CPU} = 2 MHz | 3.2 | 5 | 2.2 | 3.5 | |
| | | f _{OSC} = 8 MHz, f _{CPU} = 4 MHz | 6 | 8 | 4.4 | 6 | |
| | | f _{OSC} = 16 MHz, f _{CPU} = 8 MHz | 10 | 15 | 8.9 | 12 | |
| | Supply current in SLOW mode ⁽³⁾ | f _{OSC} = 2 MHz, f _{CPU} = 62.5kHz | 0.5 | 2.7 | 0.1 | 0.2 | |
| | | f _{OSC} = 4 MHz, f _{CPU} = 125 kHz | 0.6 | 3 | 0.2 | 0.4 | |
| | | f _{OSC} = 8 MHz, f _{CPU} = 250 kHz | 0.85 | 3.6 | 0.4 | 0.8 | |
| | | f _{OSC} = 16 MHz, f _{CPU} = 500 kHz | 1.25 | 4 | 0.8 | 1.5 | |
| Supply current in WAIT mode ⁽³⁾ | f _{OSC} = 2 MHz, f _{CPU} = 1 MHz | 1 | 3 | 0.7 | 3 | | |
| | f _{OSC} = 4 MHz, f _{CPU} = 2 MHz | 1.8 | 4 | 1.4 | 4 | | |
| | f _{OSC} = 8 MHz, f _{CPU} = 4 MHz | 3.4 | 5 | 2.9 | 5 | | |
| | f _{OSC} = 16 MHz, f _{CPU} = 8 MHz | 6.4 | 7 | 5.7 | 7 | | |
| Supply current in SLOW WAIT mode ⁽²⁾ | f _{OSC} = 2 MHz, f _{CPU} = 62.5 kHz | 0.4 | 1.2 | 0.07 | 0.12 | | |
| | f _{OSC} = 4 MHz, f _{CPU} = 125 kHz | 0.5 | 1.3 | 0.14 | 0.2 | | |
| | f _{OSC} = 8 MHz, f _{CPU} = 250 kHz | 0.6 | 1.8 | 0.28 | 0.5 | | |
| | f _{OSC} = 16 MHz, f _{CPU} = 500 kHz | 0.8 | 2 | 0.56 | 1 | | |
| Supply current in HALT mode ⁽⁴⁾ | V _{DD} = 5.5V | -40°C ≤ T _A ≤ +85°C | <1 | 10 | <1 | 10 | μA |
| | | -40°C ≤ T _A ≤ +125°C | | 50 | | 50 | |
| Supply current in ACTIVE HALT mode ⁽⁴⁾⁽⁵⁾ | | | 0.5 | 1.2 | 0.18 | 0.25 | mA |
| Supply current in AWUFH mode ⁽⁴⁾⁽⁵⁾ | V _{DD} = 5.5V | -40°C ≤ T _A ≤ +85°C | 25 | 30 | 25 | 30 | μA |
| | | -40°C ≤ T _A ≤ +125°C | | 70 | | 70 | |

1. Typical data are based on T_A = 25°C, V_{DD} = 5V (4.5V ≤ V_{DD} ≤ 5.5V range).
2. Data based on characterization results, tested in production at V_{DD} max., f_{CPU} max. and T_A max.
3. Measurements are done in the following conditions:
 - Program executed from Flash, CPU running with Flash (for flash devices).
 - All I/O pins in input mode with a static value at V_{DD} or V_{SS} (no load)
 - All peripherals in reset state.
 - 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 20.5.1: Crystal and ceramic resonator oscillators](#)) and the peripheral power consumption ([Section 20.4.2: On-chip peripherals](#)).
4. All I/O pins in input mode with a static value at V_{DD} or V_{SS} (no load). Data based on characterization results, tested in production at V_{DD} max., f_{CPU} max. and T_A max.
5. This consumption refers to the Halt period only and not the associated run period which is software dependent.

20.4.1 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 100. Clock source current consumption

| Symbol | Parameter | Conditions | Typ | Max ⁽¹⁾ | Unit |
|----------------------|--|---------------------------------|---|--------------------|------|
| I _{DD(RES)} | Supply current of resonator oscillator ⁽²⁾⁽³⁾ | | See Section 20.5.1: Crystal and ceramic resonator oscillators | | μA |
| I _{DD(PLL)} | PLL supply current | V _{DD} = 5V | 360 | | |
| I _{DD(LVD)} | LVD supply current | HALT mode, V _{DD} = 5V | 150 | 300 | |

1. Data based on characterization results, not tested in production.
2. Data based on characterization results done with the external components specified in [Section 20.5.1: Crystal and ceramic resonator oscillators](#), not tested in production.
3. As the oscillator is based on a current source, the consumption does not depend on the voltage.

20.4.2 On-chip peripherals

T_A = 25°C, f_{CPU} = 8 MHz

Table 101. Peripheral consumption

| Symbol | Parameter | Conditions | Typ | Unit |
|-----------------------|---|------------------------|-----|------|
| I _{DD(TIM)} | 16-bit timer supply current ⁽¹⁾ | V _{DD} = 5.0V | 50 | μA |
| I _{DD(TIM8)} | 8-bit timer supply current ⁽¹⁾ | | | |
| I _{DD(ART)} | ART PWM supply current ⁽²⁾ | | 75 | |
| I _{DD(SPI)} | SPI supply current ⁽³⁾ | | 400 | |
| I _{DD(SCI)} | SCI supply current ⁽⁴⁾ | | | |
| I _{DD(ADC)} | ADC supply current when converting ⁽⁵⁾ | | 800 | |
| I _{DD(CAN)} | CAN supply current ⁽⁶⁾ | | | |

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. Data valid for one SCI.
5. Data based on a differential I_{DD} measurement between reset configuration and continuous A/D conversions.
6. Data based on a differential I_{DD} measurement between reset configuration (CAN disabled) and a permanent CAN data transmit sequence with RX and TX connected together. This measurement include the pad toggling consumption.

20.5 Clock and timing characteristics

Subject to general operating conditions for V_{DD} , f_{OSC} , and T_A .

Table 102. General timings

| Symbol | Parameter | Conditions | Min | Typ ⁽¹⁾ | Max | Unit |
|---------------|---|---------------------------|------|--------------------|------|---------------|
| $t_{c(INST)}$ | Instruction cycle time | | 2 | 3 | 12 | t_{CPU} |
| | | $f_{CPU} = 8 \text{ MHz}$ | 250 | 375 | 1500 | ns |
| $t_{v(IT)}$ | Interrupt reaction time ⁽²⁾ $t_{v(IT)} = \Delta t_{c(INST)} + 10$ | | 10 | | 22 | t_{CPU} |
| | | $f_{CPU} = 8 \text{ MHz}$ | 1.25 | | 2.75 | μs |

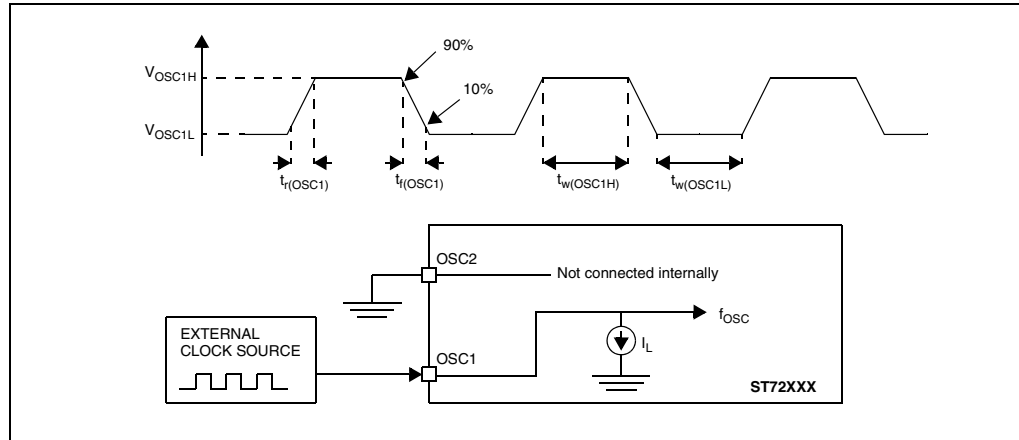
1. Data based on typical application software.
2. Time measured between interrupt event and interrupt vector fetch. $Dt_{c(INST)}$ is the number of t_{CPU} cycles needed to finish the current instruction execution.

Table 103. External clock source

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|----------------------------------|---------------------------------------|----------------------------------|---------------------|-----|---------------------|---------------|
| V_{OSC1H} | OSC1 input pin high level voltage | see Figure 122 | $0.7 \times V_{DD}$ | - | V_{DD} | V |
| V_{OSC1L} | OSC1 input pin low level voltage | | V_{SS} | - | $0.3 \times V_{DD}$ | |
| $t_{w(OSC1H)}$ $t_{w(OSC1L)}$ | OSC1 high or low time ⁽¹⁾ | | 25 | - | | ns |
| $t_r(OSC1)$ $t_f(OSC1)$ | OSC1 rise or fall time ⁽¹⁾ | | | - | 5 | |
| I_L | OSCx Input leakage current | $V_{SS} \leq V_{IN} \leq V_{DD}$ | | - | ± 1 | μA |

1. Data based on design simulation and/or technology characteristics, not tested in production.

Figure 122. Typical application with an external clock source



20.5.1 Crystal and ceramic resonator oscillators

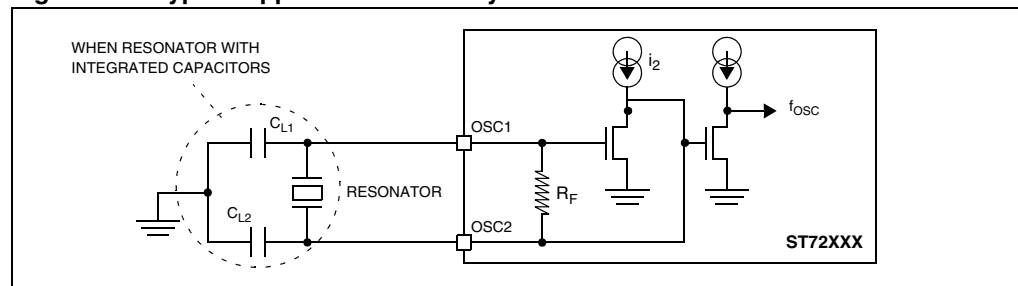
The ST7 internal clock can be supplied with four different crystal/ ceramic resonator oscillators. All the information given in this paragraph is based on characterization results with specified typical external components. In the application, the resonator and the load capacitors have to be placed as 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 (frequency, package, accuracy...)^{(a)(b)}.

Table 104. Oscillator characteristics

| Symbol | Parameter | Conditions | Min | Max | Unit |
|----------------------|--|--|-------------------------|--------------------------|------------|
| f_{OSC} | Oscillator Frequency ⁽¹⁾ | LP: Low power oscillator MP: Medium power oscillator MS: Medium speed oscillator HS: High speed oscillator | 1 >2 >4 >8 | 2 4 8 16 | MHz |
| R_F | Feedback resistor | | 20 | 40 | k Ω |
| C_{L1} C_{L2} | Recommended load capacitance versus equivalent serial resistance of the crystal or ceramic resonator (R_S) | $R_S = 200\Omega$ LP oscillator $R_S = 200\Omega$ MP oscillator $R_S = 200\Omega$ MS oscillator $R_S = 100\Omega$ HS oscillator | 22 22 18 15 | 56 46 33 33 | pF |
| i_2 | OSC2 driving current | $V_{DD} = 5V$ LP oscillator $V_{IN} = V_{SS}$ MP oscillator MS oscillator HS oscillator | 80 160 310 610 | 150 250 460 910 | μA |

1. The oscillator selection can be optimized in terms of supply current using an high quality resonator with small R_S value. Refer to crystal/ceramic resonator manufacturer for more details.

Figure 123. Typical application with a crystal or ceramic resonator



- Resonator characteristics given by the crystal/ceramic resonator manufacturer.
- $t_{SU(OSC)}$ is the typical oscillator start-up time measured between $V_{DD} = 2.8V$ and the fetch of the first instruction (with a quick V_{DD} ramp-up from 0 to 5V (< 50 μs)).

20.5.2 PLL characteristics

Operating conditions: V_{DD} 3.8 to 5.5V @ T_A 0 to 70°C^(a) or V_{DD} 4.5 to 5.5V @ T_A -40 to 125°C

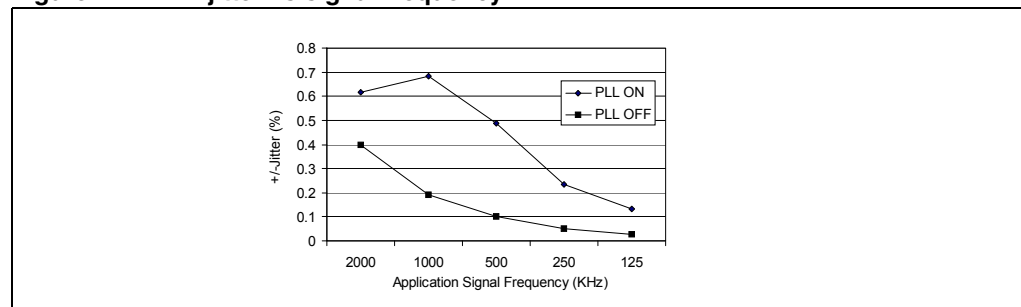
Table 105. PLL characteristics

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|--------------------------|---------------------------|---|-----|-------------------------|-----|------|
| $V_{DD(PLL)}$ | PLL Voltage Range | $T_A = 0$ to +70°C | 3.8 | | 5.5 | |
| | | $T_A = -40$ to +125°C | 4.5 | | | |
| f_{OSC} | PLL input frequency range | | 2 | | 4 | MHz |
| $\Delta f_{CPU}/f_{CPU}$ | PLL jitter ⁽¹⁾ | $f_{OSC} = 4$ MHz, $V_{DD} = 4.5$ to 5.5V | | See note ⁽²⁾ | | % |
| | | $f_{OSC} = 2$ MHz, $V_{DD} = 4.5$ to 5.5V | | | | |

1. Data characterized but not tested.

2. Under characterization.

Figure 124. PLL jitter vs signal frequency⁽¹⁾



1. Measurement conditions: $f_{CPU} = 4$ MHz, $T_A = 25^\circ\text{C}$

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 124 shows the PLL jitter integrated on application signals in the range 125 kHz to 2 MHz. At frequencies of less than 125 kHz, the jitter is negligible.

a. Data characterized but not tested

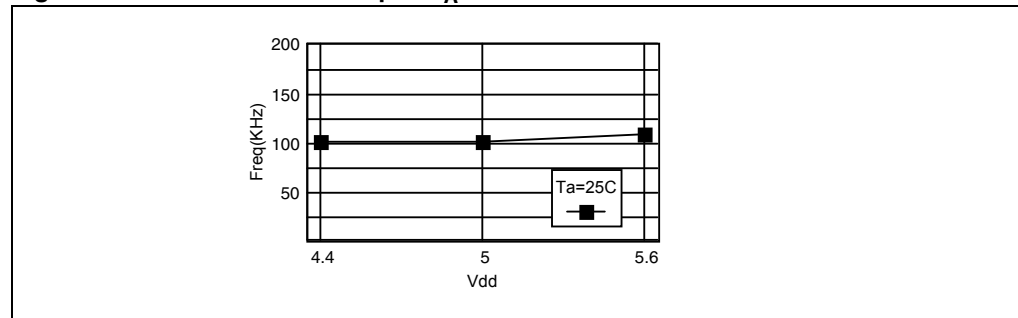
20.6 Auto wakeup from halt oscillator (AWU)

Table 106. AWU oscillator characteristics

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|-------------|---|------------|-----|-----|-----|---------|
| f_{AWU} | AWU oscillator frequency ⁽¹⁾ | | 50 | 100 | 250 | kHz |
| t_{RCSRT} | AWU oscillator startup time | | | 10 | | μ s |

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

Figure 125. AWU oscillator freq. @ T_A 25°C



20.7 Memory characteristics

20.7.1 RAM and hardware registers

Table 107. 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.

20.7.2 Flash memory

Table 108. 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⁾ | $4.5V \leq V_{DD} \leq 5.5V$ | 11.4 | | 12.6 | V |
| I_{PP} | V_{PP} current ⁽³⁾⁽⁴⁾ | Read ($V_{PP} = 12V$) | | | 200 | μ A |
| | | Write / erase | | | 30 | mA |
| t_{VPP} | Internal V_{PP} stabilization time | | | $10^{(3)}$ | | μ s |

Table 108. Dual voltage HDFlash memory (continued)

| Symbol | Parameter | Conditions | Min ⁽¹⁾ | Typ | Max ⁽¹⁾ | Unit |
|---------------------------|--|--------------------------|--------------------|-----|--------------------|------------------|
| t_{RET} | Data retention | $T_A = 55^\circ\text{C}$ | 20 | | | years |
| N_{RW} | Write erase cycles | $T_A = 85^\circ\text{C}$ | 100 | | | cycles |
| T_{PROG} T_{ERASE} | Programming or erasing temperature range | | -40 | 25 | 85 | $^\circ\text{C}$ |

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.
4. In Write / erase mode the IDD supply current consumption is the same as in Run mode (see [Section 20.2.2](#))

20.8 EMC characteristics

Susceptibility tests are performed on a sample basis during product characterization.

20.8.1 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:** Electro-Static 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 are given in the table below 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 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).

Table 109. 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 | $V_{DD} = 5V$, $T_A = +25^\circ C$, $f_{OSC} = 8\text{ MHz}$ 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 | $V_{DD} = 5V$, $T_A = +25^\circ C$, $f_{OSC} = 8\text{ MHz}$ conforms to IEC 1000-4-4 | |

20.8.2 Electromagnetic interference (EMI)

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 110. EMI emissions

| Symbol | Parameter | Conditions | Monitored frequency band | Max vs. $[f_{OSC}/f_{CPU}]^{(1)}$ | | Unit |
|-----------|------------|---|--------------------------|-----------------------------------|----------|------------|
| | | | | 8/4 MHz | 16/8 MHz | |
| S_{EMI} | Peak level | Flash devices: $V_{DD} = 5V$, $T_A = +25^\circ C$, LQFP64 package conforming to SAE J 1752/3 | 0.1 MHz to 30 MHz | 31 | 32 | dB μ V |
| | | | 30 MHz to 130 MHz | 32 | 37 | |
| | | | 130 MHz to 1 GHz | 11 | 16 | |
| | | | SAE EMI Level | 3.0 | 3.5 | |
| | | ROM devices: $V_{DD} = 5V$, $T_A = +25^\circ C$, LQFP64 package conforming to SAE J 1752/3 | 0.1 MHz to 30 MHz | 10 | 18 | dB μ V |
| | | | 30 MHz to 130 MHz | 15 | 25 | |
| | | | 130 MHz to 1 GHz | -3 | 1 | |
| | | | SAE EMI Level | 2.0 | 2.5 | |

1. Not tested in production.

20.8.3 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 (see [Table 111](#) and [Table 112](#) below). For more details, refer to 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.

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 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 application note AN1181.

Table 111. Absolute maximum ratings

| Symbol | Ratings | Conditions | Maximum 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 | |
| V _{ESD(CDM)} | Electro-tatic discharge voltage (charge device model) | | 750 on corner pins, 500 on others | |

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

Table 112. Electrical sensitivities

| Symbol | Parameter | Conditions | Class ⁽¹⁾ |
|--------|------------------------|---|----------------------|
| LU | Static latch-up class | T _A = +25°C T _A = +85°C T _A = +125°C | A |
| DLU | Dynamic latch-up class | V _{DD} = 5.5V, f _{OSC} = 4 MHz, T _A = +25°C | |

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).

20.9 I/O port pin characteristics

20.9.1 General characteristics

Subject to general operating conditions for V_{DD} , f_{OSC} , and T_A unless otherwise specified.

Table 113. I/O characteristics

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit | |
|--------------------------------------|---|---|---------------------|-----|---------------------|-----------|----|
| V_{IL} | Input low level voltage ⁽¹⁾ | CMOS ports | | | $0.3 \times V_{DD}$ | V | |
| V_{IH} | Input high level voltage ⁽¹⁾ | | $0.7 \times V_{DD}$ | | | | |
| V_{hys} | Schmitt trigger voltage hysteresis ⁽²⁾ | | | 1 | | | |
| V_{IL} | Input low level voltage ⁽¹⁾ | TTL ports | | | 0.8 | mV | |
| V_{IH} | Input high level voltage ⁽¹⁾ | | 2 | | | | |
| V_{hys} | Schmitt trigger voltage hysteresis ⁽²⁾ | | | 400 | | | |
| $I_{INJ(PIN)}$ | Injected Current on PB3 | $V_{DD} = 5V$ | Flash devices | 0 | | +4 | mA |
| | Injected Current on any other I/O pin | | ROM devices | | | ±4 | |
| $\Sigma I_{INJ(PIN)}$ ⁽³⁾ | Total injected current (sum of all I/O and control pins) ⁽⁴⁾ | | | | | ±25 | |
| I_{lkg} | Input leakage current on robust pins | See Section 20.13: 10-bit ADC characteristics | | | | | |
| | Input leakage current ⁽⁵⁾ | $V_{SS} \leq V_{IN} \leq V_{DD}$ | | | ±1 | µA | |
| I_S | Static current consumption ⁽⁶⁾ | Floating input mode | | 200 | | | |
| R_{PU} | Weak pull-up equivalent resistor ⁽⁷⁾ | $V_{IN} = V_S$ $V_{DD} = 5V$ | 50 | 90 | 250 | kΩ | |
| C_{IO} | I/O pin capacitance | | | 5 | | pF | |
| $t_{f(I/O)out}$ | Output high to low level fall time | $C_L = 50pF$ Between 10% and 90% | | 25 | | ns | |
| $t_{r(I/O)out}$ | Output low to high level rise time | | | | | | |
| $t_{w(IT)in}$ | External interrupt pulse time ⁽⁴⁾ | | 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} absolute maximum rating 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 20.2: Absolute maximum ratings](#) for more details.
4. To generate an external interrupt, a minimum pulse width must be applied on an I/O port pin configured as an external interrupt source.
5. Leakage could be higher than max. if negative current is injected on adjacent pins.
6. Configuration not recommended, all unused pins must be kept at a fixed voltage: Using the output mode of the I/O, for example, or an external pull-up or pull-down resistor (see [Figure 126](#)). Data based on design simulation and/or technology characteristics, not tested in production.
7. The R_{PU} pull-up equivalent resistor is based on a resistive transistor (corresponding I_{PU} current characteristics described in [Figure 127](#)).

Figure 126. Connecting unused I/O pins

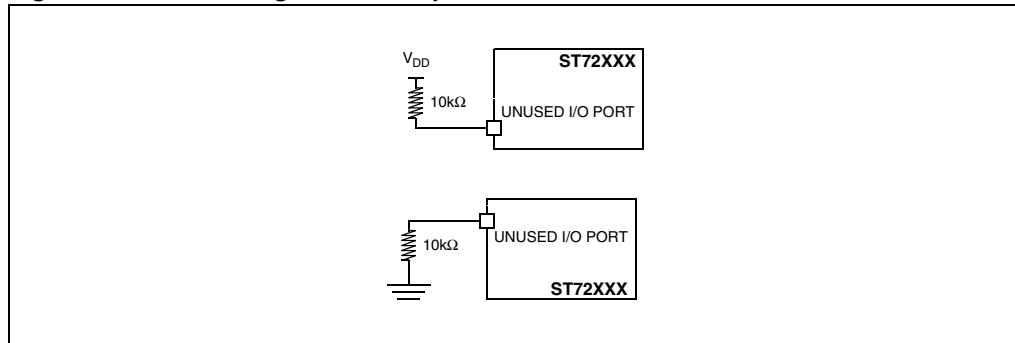


Figure 127. R_{PU} vs V_{DD} with $V_{IN} = V_{SS}$

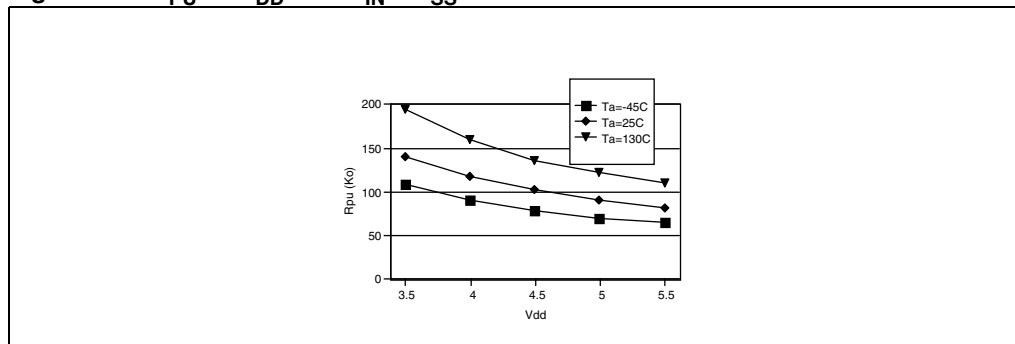
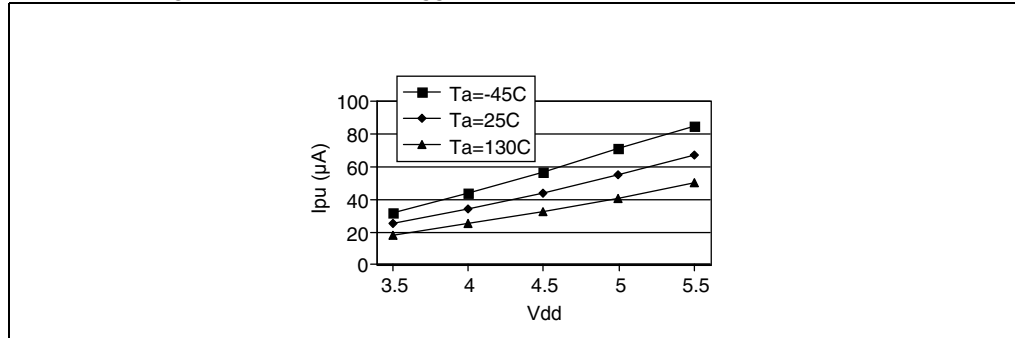


Figure 128. I_{PU} vs V_{DD} with $V_{IN} = V_{SS}$



20.9.2 Output driving current

Subject to general operating conditions for V_{DD} , f_{OSC} , and T_A unless otherwise specified.

Table 114. Output driving current

| Symbol | Parameter | Conditions | Min | Max | Unit |
|----------------|--|---|------------------------------|------------|------|
| $V_{OL}^{(1)}$ | Output low level voltage for a standard I/O pin when eight pins are sunk at same time (see Figure 129) | $I_{IO} = +5 \text{ mA}$ | | 1.2 | V |
| | | $I_{IO} = +2 \text{ mA}$ | | 0.5 | |
| | Output low level voltage for a high sink I/O pin when four pins are sunk at same time (see Figure 130 and Figure 133) | $I_{IO} = +20 \text{ mA}$, $T_A \leq 85 \text{ }^\circ\text{C}$ $T_A \geq 85 \text{ }^\circ\text{C}$ | | 1.3 1.5 | |
| | | $I_{IO} = +8 \text{ mA}$ | | 0.6 | |
| $V_{OH}^{(2)}$ | Output high level voltage for an I/O pin when four pins are sourced at same time (see Figure 131 and Figure 134) | $I_{IO} = -5 \text{ mA}$, $T_A \leq 85 \text{ }^\circ\text{C}$ $T_A \geq 85 \text{ }^\circ\text{C}$ | $V_{DD}-1.4$ $V_{DD}-1.6$ | | |
| | | $I_{IO} = -2 \text{ mA}$ | $V_{DD}-0.7$ | | |

1. The I_{IO} current sunk must always respect the absolute maximum rating specified in [Section 20.2.2: Current characteristics](#) 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 20.2.2: Current characteristics](#) and the sum of I_{IO} (I/O ports and control pins) must not exceed I_{VDD} . True open drain I/O pins does not have V_{OH} .

Figure 129. Typical V_{OL} at $V_{DD} = 5V$ (standard)

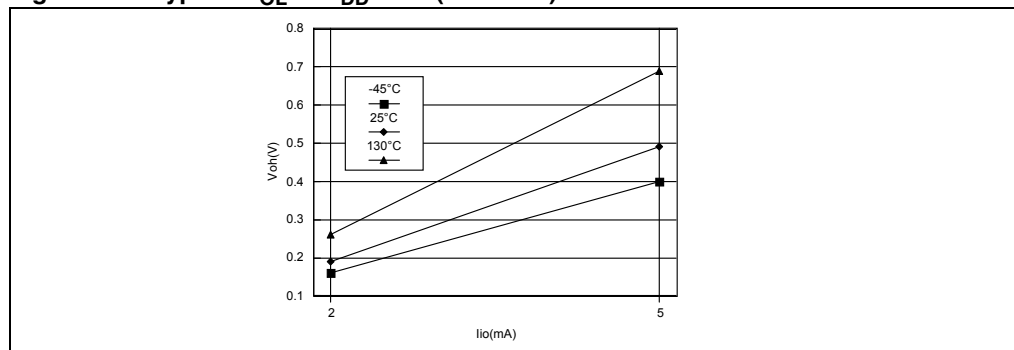


Figure 130. Typical V_{OL} at $V_{DD} = 5V$ (high-sink)

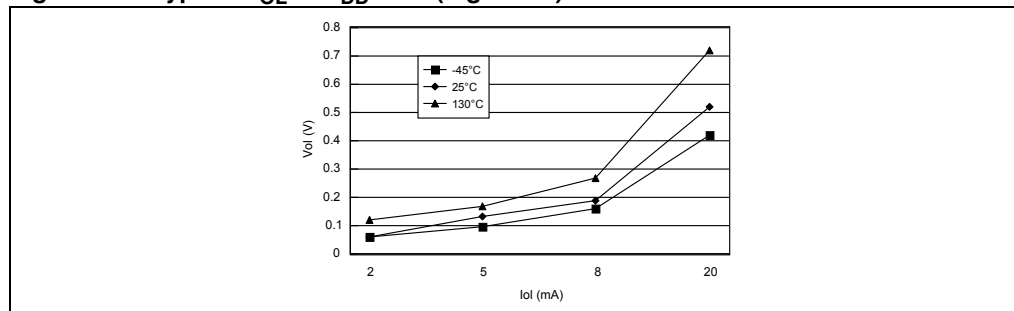


Figure 131. Typical V_{OH} at $V_{DD} = 5V$

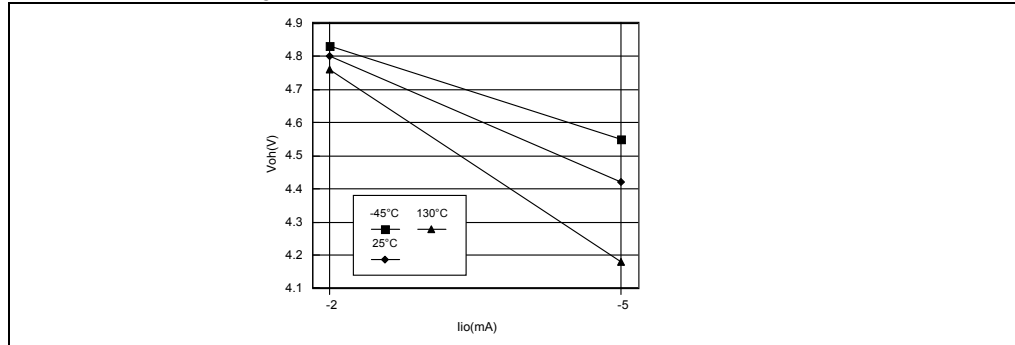


Figure 132. Typical V_{OL} vs V_{DD} (standard I/Os)

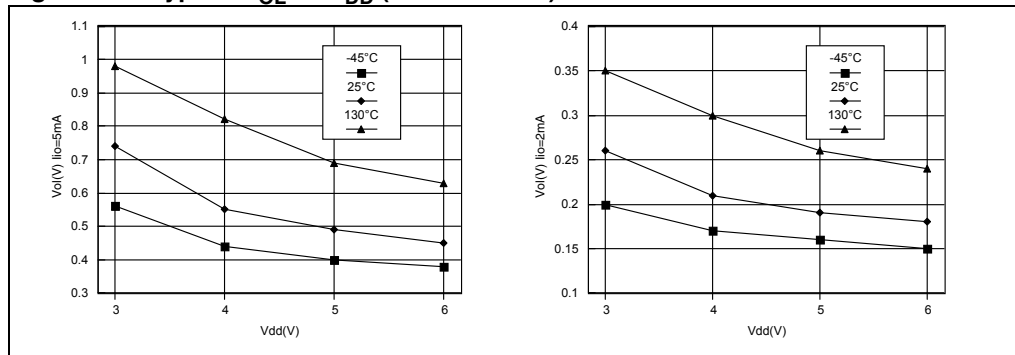


Figure 133. Typical V_{OL} vs V_{DD} (high-sink I/Os)

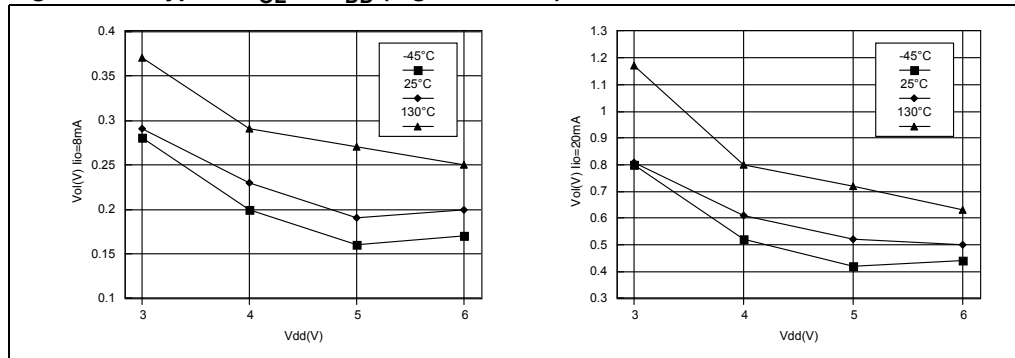
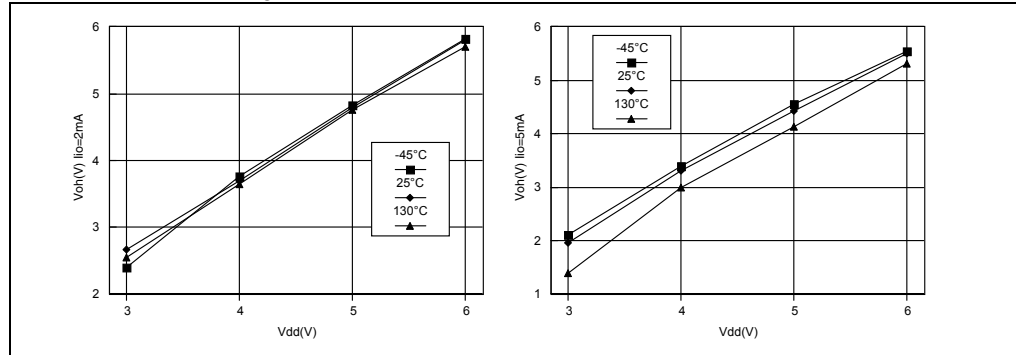


Figure 134. Typical V_{OH} vs V_{DD} 

20.10 Control pin characteristics

20.10.1 Asynchronous $\overline{\text{RESET}}$ pin

Subject to general operating conditions for V_{DD} , f_{OSC} , and T_A unless otherwise specified.

Table 115. $\overline{\text{RESET}}$ pin characteristics

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|------------------|---|-----------------------|------------------------|------|---------------------|---------------|
| V_{IL} | Input low level voltage ⁽¹⁾ | | | | $0.3 \times V_{DD}$ | V |
| V_{IH} | Input high level voltage ⁽¹⁾ | | $0.7 \times V_{DD}$ | | | |
| V_{hys} | Schmitt trigger voltage hysteresis ⁽²⁾ | $V_{DD} = 5V$ | | 1.5 | | |
| V_{OL} | Output low level voltage ⁽³⁾ | $V_{DD} = 5V$ | $I_{IO} = +5\text{mA}$ | 0.68 | 0.95 | |
| | | | $I_{IO} = +2\text{mA}$ | 0.28 | 0.45 | |
| R_{ON} | Weak pull-up equivalent resistor ⁽⁴⁾ | $V_{IN} = V_{SS}$ | 20 | 40 | 80 | $k\Omega$ |
| $t_{w(RSTL)out}$ | Generated reset pulse duration | Internal reset source | | 30 | | μs |
| $t_{h(RSTL)in}$ | External reset pulse hold time ⁽⁵⁾ | | 2.5 | | | μs |
| $t_{g(RSTL)in}$ | 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 20.2.2: Current characteristics](#) and the sum of I_{IO} (I/O ports and control pins) must not exceed I_{VSS} .
4. 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.
5. The reset network (the resistor and two capacitors) protects the device against parasitic resets, especially in noisy environments.
6. Data guaranteed by design, not tested in production.

$\overline{\text{RESET}}$ circuit design recommendations

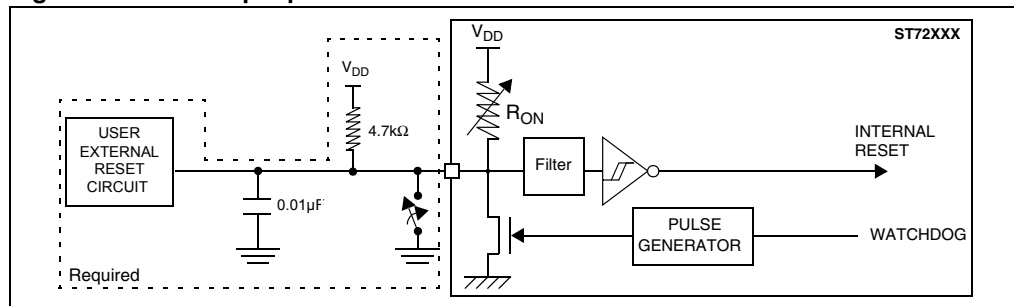
The reset network protects the device against parasitic resets. The output of the external reset circuit must have an open-drain output to drive the ST7 reset pad. Otherwise the device can be damaged when the ST7 generates an internal reset (LVD or watchdog).

Whatever 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} max. level specified in [Section 20.10.1: Asynchronous \$\overline{\text{RESET}}\$ pin](#). 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 (by an external pull-up for example) is less than the absolute maximum value specified for $I_{\text{INJ}}(\overline{\text{RESET}})$ in [Section 20.2.2: Current characteristics](#).

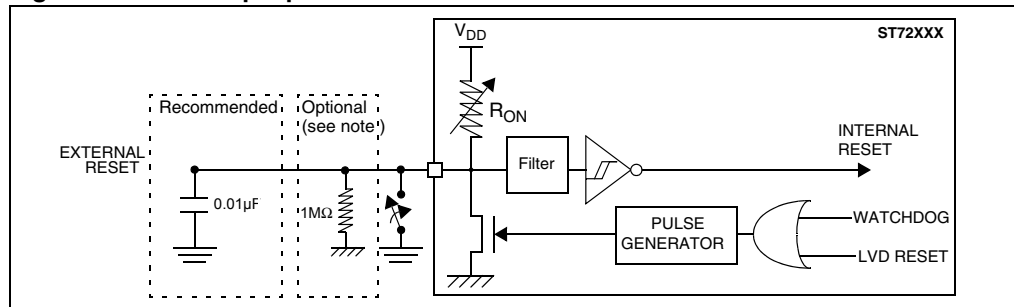
$\overline{\text{RESET}}$ pin protection when LVD is disabled

Figure 135. $\overline{\text{RESET}}$ pin protection when LVD is disabled



$\overline{\text{RESET}}$ pin protection when LVD is enabled

Figure 136. $\overline{\text{RESET}}$ pin protection when LVD is enabled

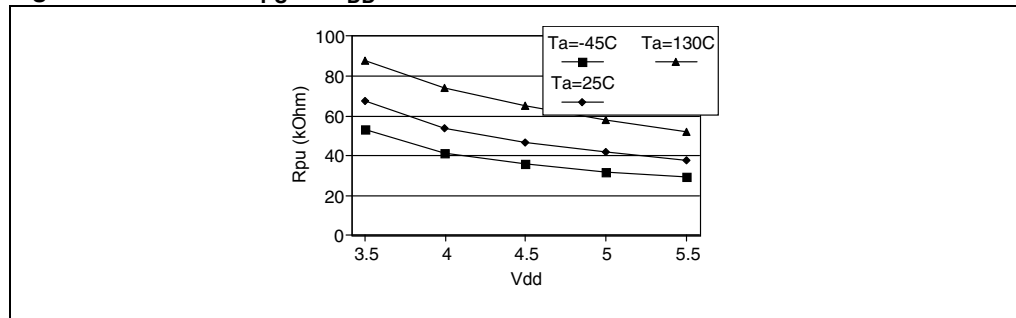


Note: When the LVD is enabled, it is mandatory not to connect a pull-up resistor. A 10nF pull-down capacitor is recommended to filter noise on the reset line. In case a capacitive power supply is used, it is recommended to connect a 1 MΩ pull-down resistor to the $\overline{\text{RESET}}$ pin to discharge any residual voltage induced by this capacitive power supply (this will add 5 μA to the power consumption of the MCU).

Tips when using the LVD

1. Check that all recommendations related to reset circuit have been applied (see [RESET circuit design recommendations](#))
2. Check that the power supply is properly decoupled (100nF + 10µF close to the MCU). Refer to AN1709. If this cannot be done, it is recommended to put a 100 nF + 1MΩ pull-down on the RESET pin.
3. The capacitors connected on the $\overline{\text{RESET}}$ pin and also the power supply are key to avoiding any start-up marginality. In most cases, steps 1 and 2 above are sufficient for a robust solution. Otherwise: replace 10nF pull-down on the $\overline{\text{RESET}}$ pin with a 5 µF to 20 µF capacitor.

Figure 137. $\overline{\text{RESET}}$ R_{PU} vs V_{DD}



20.10.2 ICCSEL/ V_{PP} pin

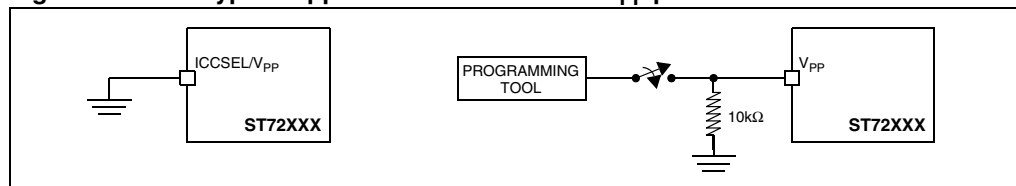
Subject to general operating conditions for V_{DD}, f_{OSC}, and T_A unless otherwise specified.

Table 116. ICCSEL/V_{PP} pin characteristics

| Symbol | Parameter | Conditions | Min | Max | Unit |
|-----------------|---|-----------------------------------|----------------------|------|------|
| V _{IL} | Input low level voltage ⁽¹⁾ | | V _{SS} | 0.2 | V |
| V _{IH} | Input high level voltage ⁽¹⁾ | | V _{DD} -0.1 | 12.6 | |
| I _L | Input leakage current | V _{IN} = V _{SS} | | ±1 | µA |

1. Data based on design simulation and/or technology characteristics, not tested in production.

Figure 138. Two typical applications with ICCSEL/V_{PP} pin



1. When ICC mode is not required by the application ICCSEL/V_{PP} pin must be tied to V_{SS}.

20.11 Timer peripheral characteristics

Subject to general operating conditions for V_{DD} , f_{OSC} , and T_A unless otherwise specified.

Refer to I/O port characteristics for more details on the input/output alternate function characteristics (output compare, input capture, external clock, PWM output...).

Table 117. 8-bit PWM-ART auto reload timer characteristics

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|----------------|--|------------------------------|-------|-----|-------------|-----------|
| $t_{res(PWM)}$ | PWM resolution time | | 1 | | | t_{CPU} |
| | | $f_{CPU} = 8 \text{ MHz}$ | 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 | $V_{DD} = 5V$, Res = 8-bits | | 20 | | mV |
| $t_{COUNTER}$ | Timer clock period when internal clock is selected | $f_{CPU} = 8 \text{ MHz}$ | 1 | | 128 | t_{CPU} |
| | | | 0.125 | | 16 | μs |

Table 118. 8-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 | | | |
| | | $f_{CPU} = 8 \text{ MHz}$ | 250 | | | ns |
| f_{PWM} | PWM repetition rate | | 0 | | $f_{CPU}/4$ | MHz |
| Res_{PWM} | PWM resolution | | | | 8 | bit |
| $t_{COUNTER}$ | Timer clock period | $f_{CPU} = 8 \text{ MHz}$ | 2 | | 8000 | t_{CPU} |
| | | | 0.250 | | 1000 | μs |

Table 119. 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 | | | |
| | | $f_{CPU} = 8 \text{ MHz}$ | 250 | | | ns |
| f_{EXT} | Timer external clock frequency | | 0 | | $f_{CPU}/4$ | MHz |
| f_{PWM} | PWM repetition rate | | | | | |
| Res_{PWM} | PWM resolution | | | | 16 | bit |

Table 119. 16-bit timer characteristics (continued)

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|----------------------|--|----------------------------------|-------|-----|-----|------------------|
| t_{COUNTER} | Timer clock period when internal clock is selected | $f_{\text{CPU}} = 8 \text{ MHz}$ | 2 | | 8 | t_{CPU} |
| | | | 0.250 | | 1 | μs |

20.12 Communication interface characteristics

20.12.1 SPI - serial peripheral interface

Subject to general operating conditions for V_{DD} , f_{OSC} , and T_A unless otherwise specified.

Refer to I/O port characteristics for more details on the input/output alternate function characteristics (\overline{SS} , SCK, MOSI, MISO).

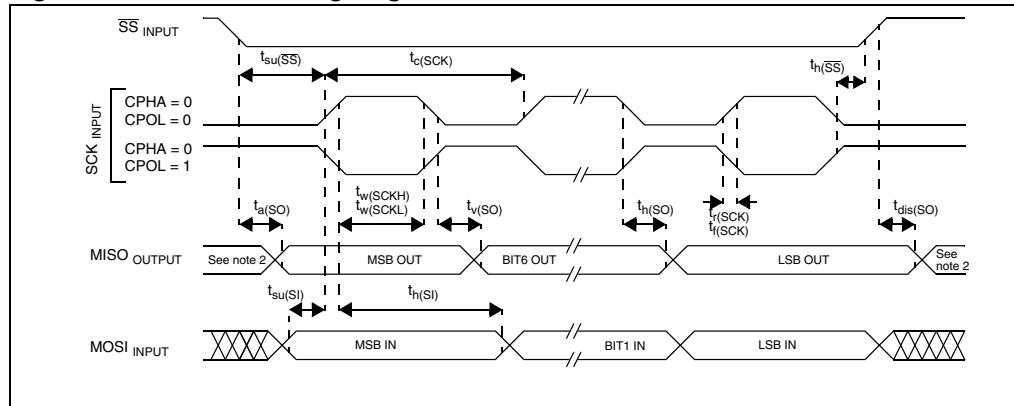
Table 120. SPI characteristics

| Symbol | Parameter | Conditions | Min | Max | Unit | |
|-------------------------------|---|----------------------------|------------------------------|-------------------|------|-----|
| $f_{SCK} = 1 / t_c(SCK)$ | SPI clock frequency | Master, $f_{CPU} = 8$ MHz | $f_{CPU} / 128 = 0.0625$ | $f_{CPU} / 4 = 2$ | MHz | |
| | | Slave, $f_{CPU} = 8$ MHz | 0 | $f_{CPU} / 2 = 4$ | | |
| $t_r(SCK)$ | SPI clock rise and fall time | | See I/O port pin description | | | |
| $t_f(SCK)$ | | | | | | |
| $t_{su}(\overline{SS})^{(1)}$ | \overline{SS} setup time ⁽²⁾ | Slave | $(4 \times T_{CPU}) + 50$ | | ns | |
| $t_h(\overline{SS})^{(1)}$ | | | \overline{SS} hold time | 120 | | |
| $t_w(SCKH)^{(1)}$ | SCK high and low time | Master | 100 | | | |
| $t_w(SCKL)^{(1)}$ | | Slave | 90 | | | |
| $t_{su}(MI)^{(1)}$ | Data input setup time | Master | 100 | | | |
| $t_{su}(SI)^{(1)}$ | | Slave | | | | |
| $t_h(MI)^{(1)}$ | Data input hold time | Master | | | | |
| $t_h(SI)^{(1)}$ | | Slave | | | | |
| $t_a(SO)^{(1)}$ | Data output access time | Slave | 0 | 120 | | |
| $t_{dis}(SO)^{(1)}$ | | | Data output disable time | | | 240 |
| $t_v(SO)^{(1)}$ | Data output valid time | Slave (after enable edge) | | 90 | | |
| $t_h(SO)^{(1)}$ | Data output hold time | | 0 | | | |
| $t_v(MO)^{(1)}$ | Data output valid time | Master (after enable edge) | | 120 | | |
| $t_h(MO)^{(1)}$ | Data output hold time | | 0 | | | |

1. Data based on design simulation and/or characterization results, not tested in production.

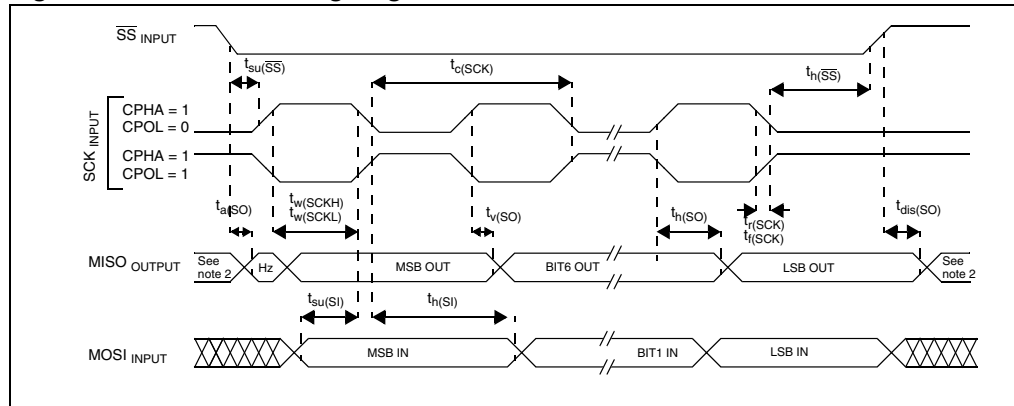
2. Depends on f_{CPU} . For example, if $f_{CPU} = 8$ MHz, then $T_{CPU} = 1 / f_{CPU} = 125$ ns and $t_{su}(\overline{SS}) = 550$ ns.

Figure 139. SPI slave timing diagram with CPHA = 0



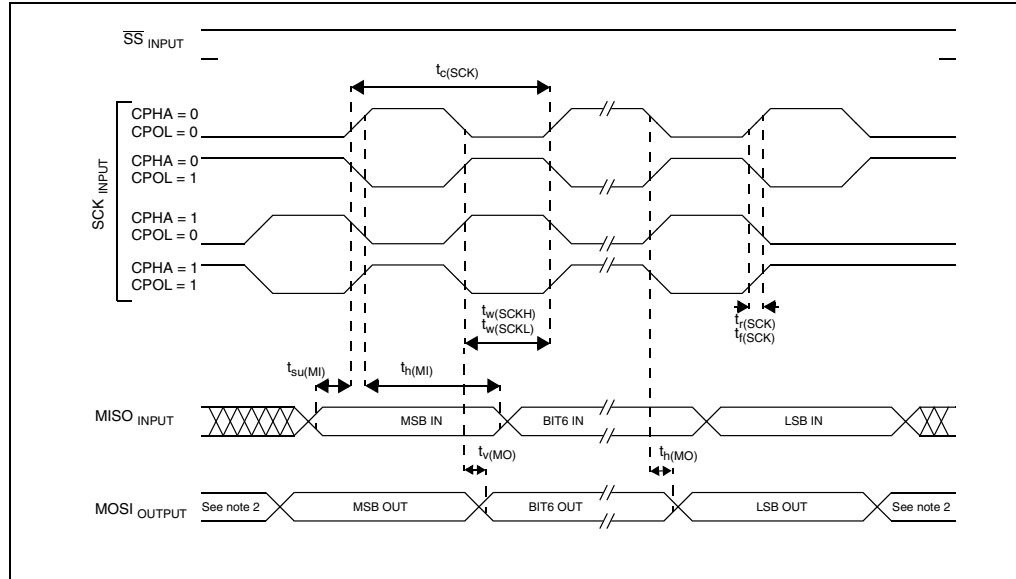
1. Measurement points are done at CMOS levels: $0.3 \times V_{DD}$ and $0.7 \times V_{DD}$
2. When no communication is on-going the data output line of the SPI (MOSI in master mode, MISO in slave mode) has its alternate function capability released. In this case, the pin status depends on the I/O port configuration.

Figure 140. SPI slave timing diagram with CPHA = 1



1. Measurement points are done at CMOS levels: $0.3 \times V_{DD}$ and $0.7 \times V_{DD}$.
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 141. SPI master timing diagram



1. Measurement points are done at CMOS levels: $0.3 \times V_{DD}$ and $0.7 \times V_{DD}$.
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.

20.12.2 CAN - controller area network interface

Subject to general operating condition for V_{DD} , f_{OSC} , and T_A unless otherwise specified. Refer to I/O port characteristics for more details on the input/output alternate function characteristics (CANTX and CANRX).

Table 121. CAN characteristics

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|--------------|--|------------|-----|-----|-----|------|
| $t_p(RX:TX)$ | CAN controller propagation time ⁽¹⁾ | | - | - | 60 | ns |

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

20.13 10-bit ADC characteristics

Subject to general operating conditions for V_{DD} , f_{CPU} , and T_A unless otherwise specified.

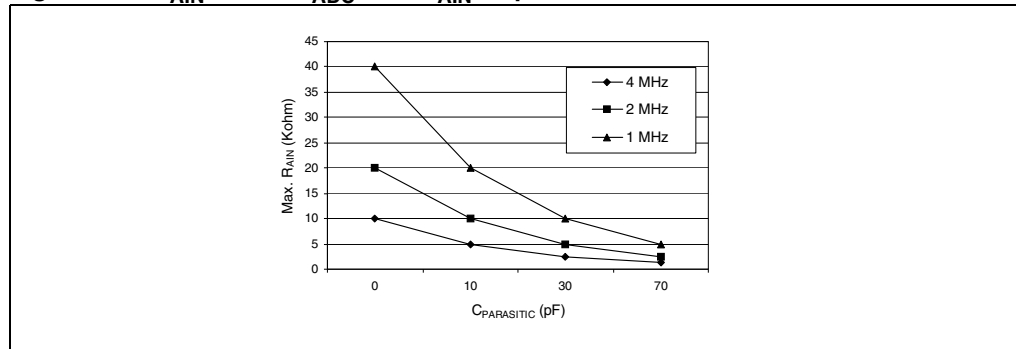
Table 122. ADC characteristics

| Symbol | Parameter | Conditions | Min ⁽¹⁾ | Typ | Max ⁽¹⁾ | Unit |
|-----------|---|------------|--------------------|-----|--------------------|------|
| f_{ADC} | ADC clock frequency | | 0.4 | | 4 | MHz |
| V_{AIN} | Conversion voltage range ⁽²⁾ | | V_{SSA} | | V_{DDA} | V |

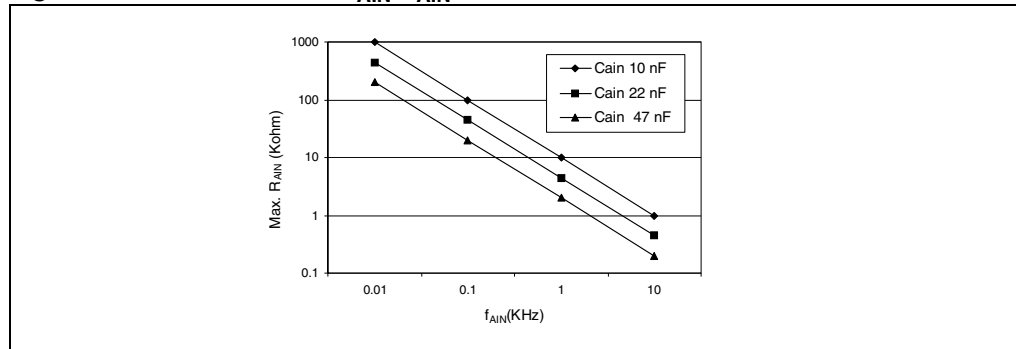
Table 122. ADC characteristics (continued)

| Symbol | Parameter | Conditions | Min ⁽¹⁾ | Typ | Max ⁽¹⁾ | Unit |
|------------|---|---|--------------------|-----|--|-------------|
| R_{AIN} | External input impedance | | | | see Figure 142 and Figure 143 | k Ω |
| C_{AIN} | External capacitor on analog input | | | | | pF |
| f_{AIN} | Variation frequency of analog input signal | | | | | Hz |
| I_{Ikg} | Negative input leakage current on robust analog pins (refer to Table 3) | $V_{IN} < V_{SS}$, $ I_{IN} < 400\mu A$ on adjacent robust analog pin | | 5 | 6 | μA |
| C_{ADC} | Internal sample and hold capacitor | | | 6 | | pF |
| t_{CONV} | Conversion time | $f_{ADC} = 4 \text{ MHz}$ | | 3.5 | | μs |
| | | | | 14 | | $1/f_{ADC}$ |
| I_{ADC} | Analog part | Sunk on V_{DDA} ²⁾ | | | 3.6 | mA |
| | Digital part | Sunk on V_{DD} | | | 0.2 | |

1. Data based on characterization results, not tested in production.
2. When V_{DDA} and V_{SSA} pins are not available on the pinout, the ADC refers to V_{DD} and V_{SS} .

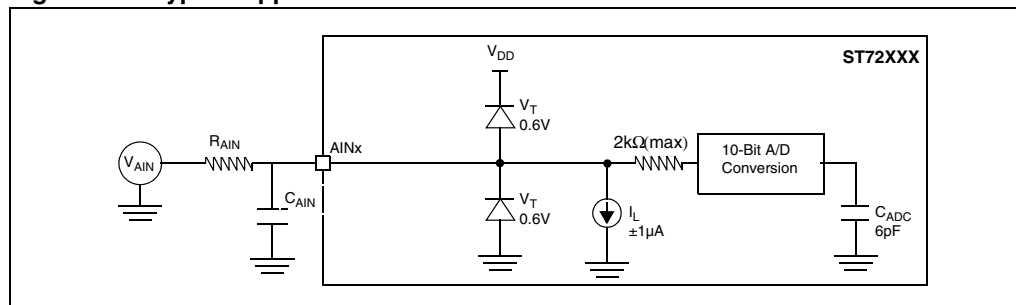
Figure 142. R_{AIN} max vs f_{ADC} with $C_{AIN} = 0pF$ 

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.
2. Any added external serial resistor will downgrade the ADC accuracy (especially for resistance greater than 10k Ω). Data based on characterization results, not tested in production.

Figure 143. Recommended C_{AIN}/R_{AIN} values

- Figure 143 shows that depending on the input signal variation (f_{AIN}), C_{AIN} can be increased for stabilization time and reduced to allow the use of a larger serial resistor (R_{AIN}). It is valid for all f_{ADC} frequencies ≤ 4 MHz.

Figure 144. Typical application with ADC



Analog power supply and reference pins

Depending on the MCU pin count, the package may feature separate V_{DDA} 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. In smaller packages V_{DDA} and V_{SSA} pins are not available and the analog supply and reference pads are internally bonded to the V_{DD} and V_{SS} pins.

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 [General PCB design guidelines](#)).

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

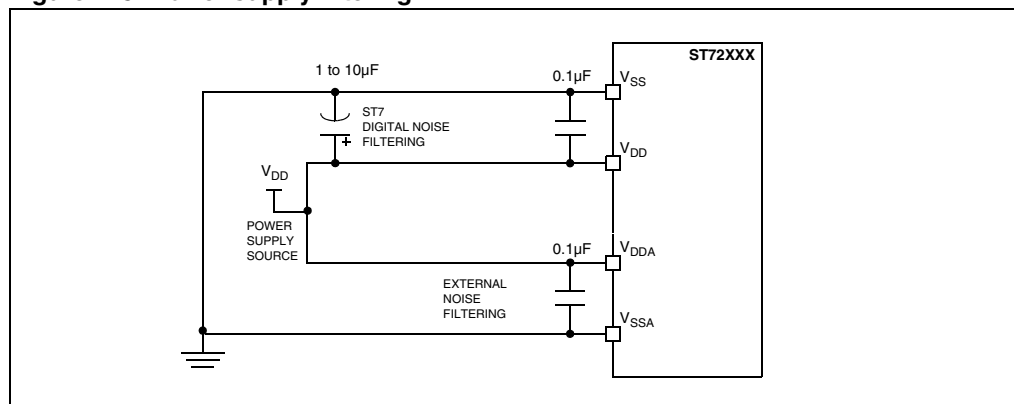
0.1 μF and optionally, if needed 10pF capacitors as close as possible to the ST7 power supply pins and a 1 to 10 μF capacitor close to the power source (see [Figure 145](#)).

- The analog and digital power supplies should be connected in a star network. Do not use a resistor, as V_{DDA} 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.

Software filtering of spurious conversion results

For EMC performance reasons, it is recommended to filter A/D conversion outliers using software filtering techniques.

Figure 145. Power supply filtering



ADC accuracy

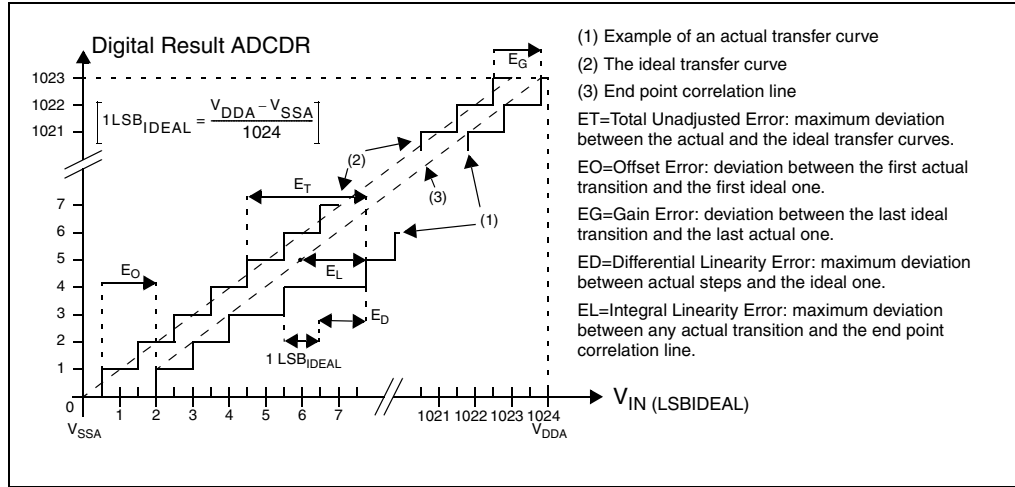
Table 123. ADC accuracy with $f_{\text{CPU}} = 8 \text{ MHz}$, $f_{\text{ADC}} = 4 \text{ MHz}$, $R_{\text{AIN}} < 10 \text{ kW}$, $V_{\text{DD}} = 5 \text{ V}$

| Symbol | Parameter | Conditions | Typ | Max | Unit |
|------------------|---|------------|-----|---------------------|------|
| $ E_{\text{T}} $ | Total unadjusted error ⁽¹⁾ | | 4 | Note ⁽²⁾ | LSB |
| $ E_{\text{O}} $ | Offset error ⁽¹⁾ | | 2.5 | 4 | |
| $ E_{\text{G}} $ | Gain error ⁽¹⁾ | | 3 | | |
| $ E_{\text{D}} $ | Differential linearity error ⁽¹⁾ | | 1.5 | 2 | |
| $ E_{\text{L}} $ | Integral linearity error ⁽¹⁾ | | | | |

1. Data based on characterization results, not tested in production. ADC accuracy vs. negative injection current: injecting negative current on any of the standard (non-robust) analog input pins should be avoided as this significantly reduces the accuracy of the conversion being performed on another analog input. It is recommended to add a Schottky diode (pin to ground) to standard analog pins which may potentially inject negative current. The effect of negative injection current on robust pins is specified in [Section 20.9: I/O port pin characteristics](#). Any positive injection current within the limits specified for $I_{\text{INJ(PIN)}}$ and $\Sigma I_{\text{INJ(PIN)}}$ in [Section 20.9: I/O port pin characteristics](#) does not affect the ADC accuracy

2. Under characterization.

Figure 146. ADC accuracy



21 Package characteristics

21.1 ECOPACK®

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.

21.2 Package mechanical data

Figure 147. 64-pin low profile quad flat package (14x14)

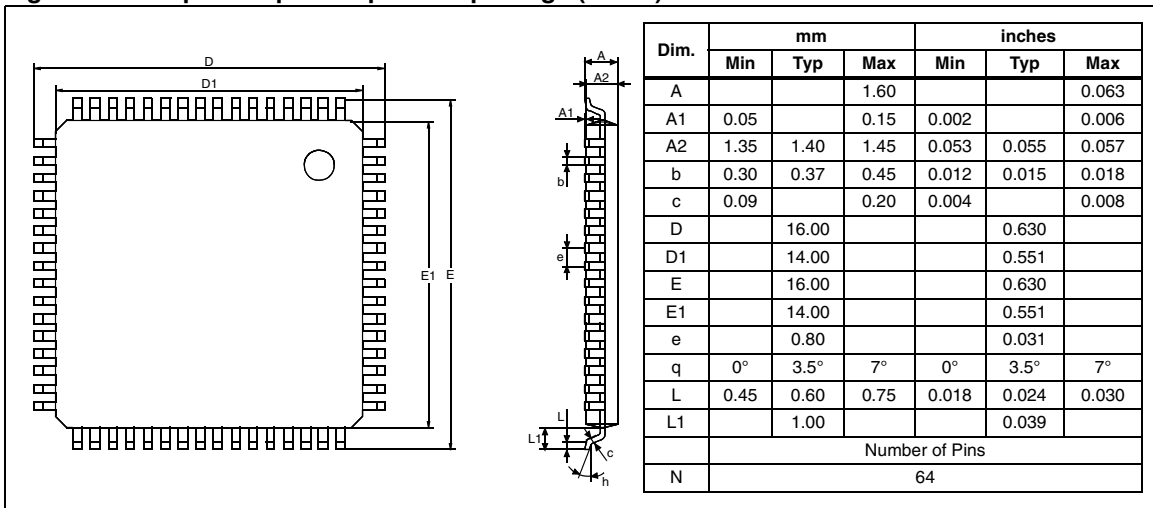


Figure 148. 32-pin low profile quad flat package (7x7)

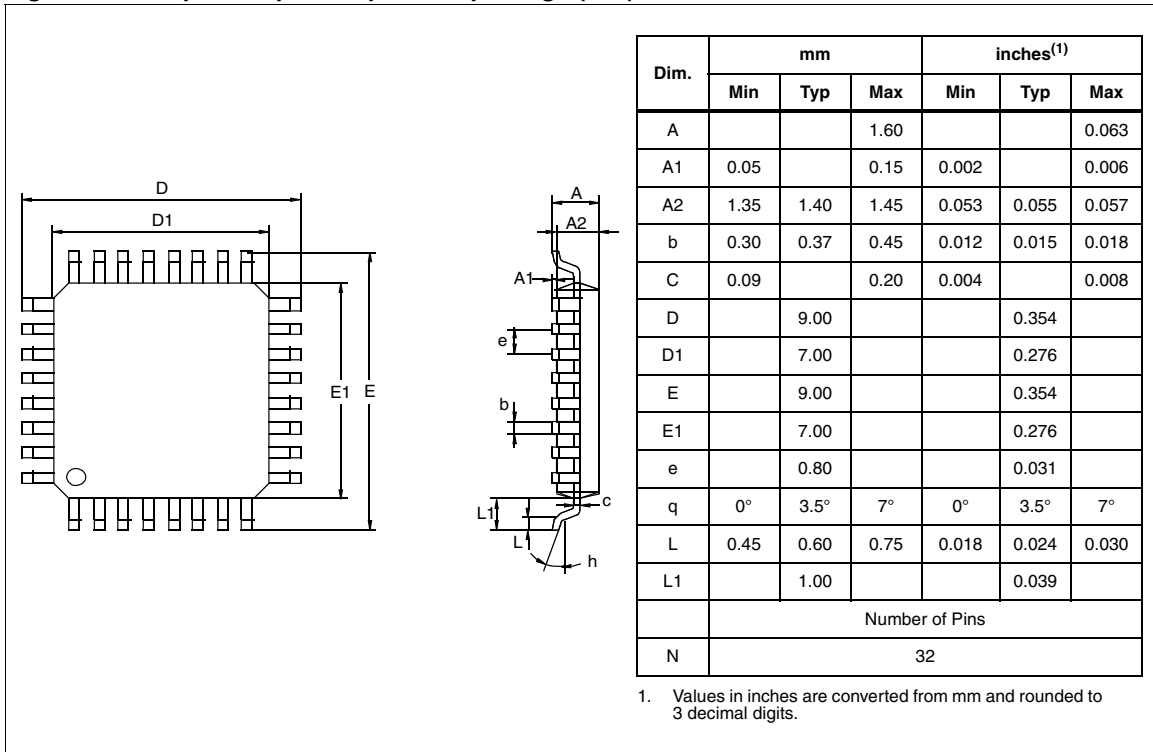


Figure 149. 44-pin low profile quad flat package (10x10)

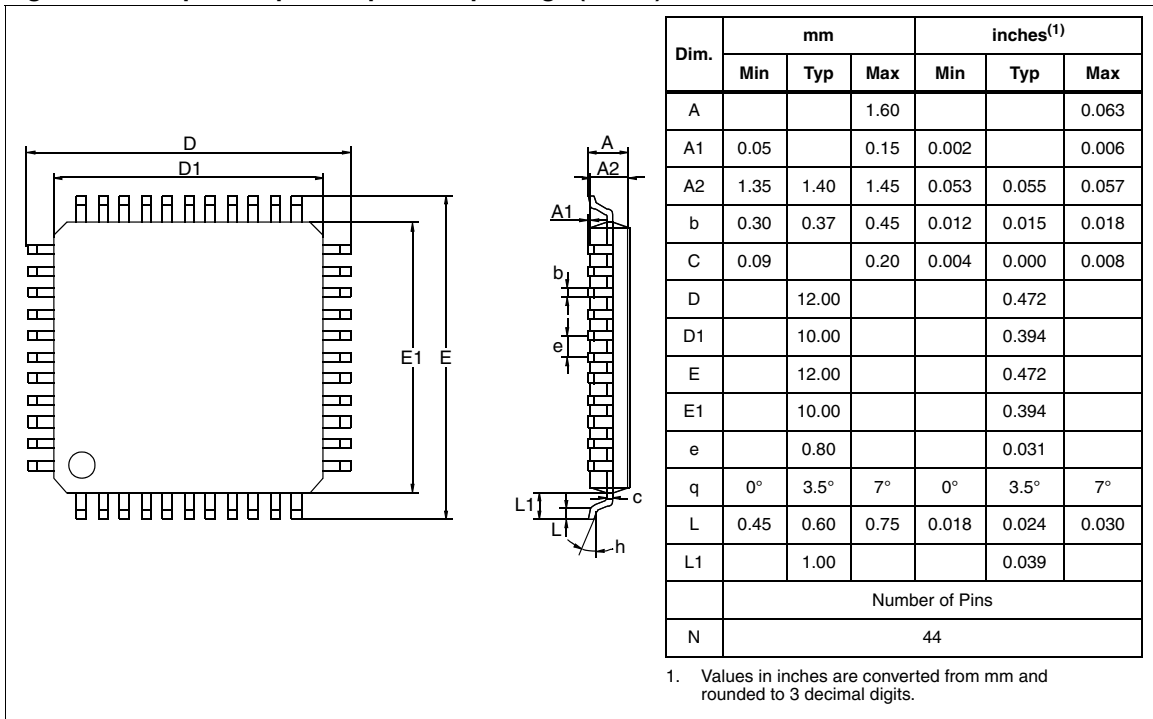
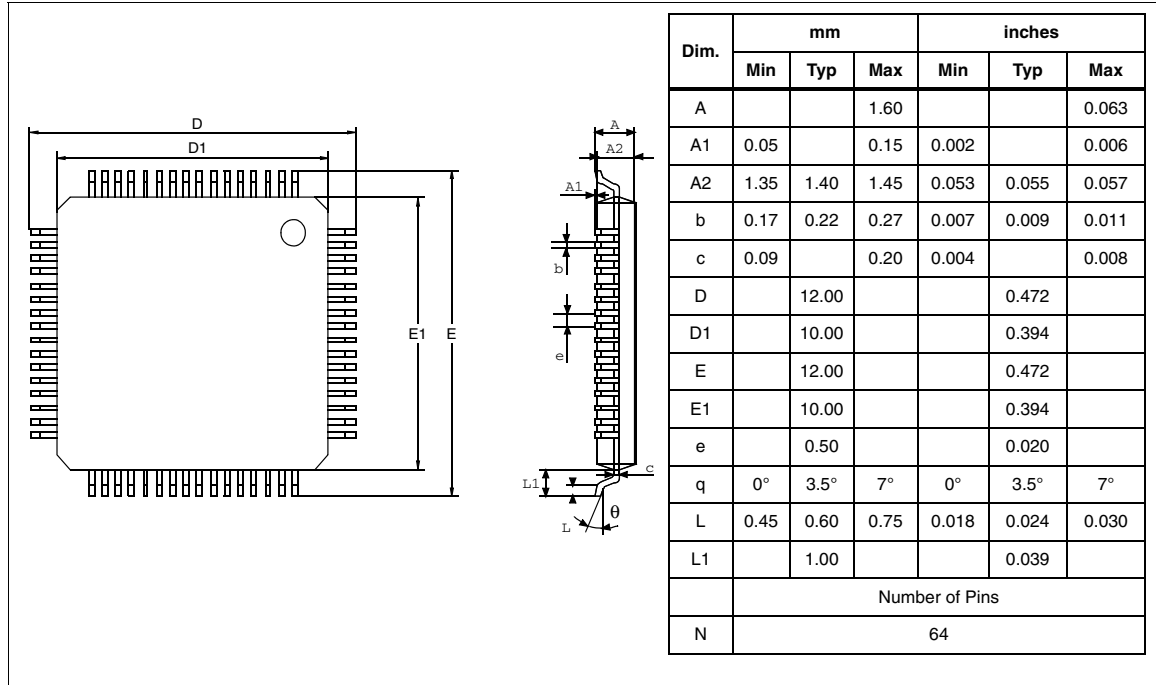


Figure 150. 64-pin low profile quad flat package (10 x10)



21.3 Thermal characteristics

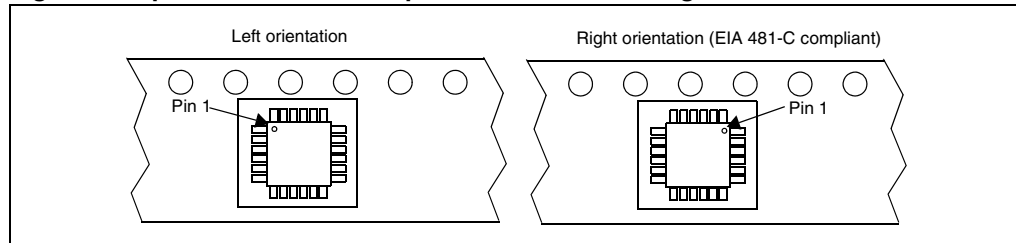
| Symbol | Ratings | Value | Unit |
|------------|--|----------------|------|
| R_{thJA} | Package thermal resistance (junction to ambient) LQFP64 LQFP44 LQFP32 | 60 52 70 | °C/W |
| P_D | Power dissipation ⁽¹⁾ | 500 | mW |
| T_{Jmax} | Maximum junction temperature ⁽²⁾ | 150 | °C |

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.

21.4 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 151](#).

Figure 151. pin 1 orientation in tape and reel conditioning

See also [Figure 152: ST72F561xx-Auto Flash commercial product structure on page 310](#) and [Figure 153: ST72P561xxx-Auto FastROM commercial product structure on page 311](#).

22 Device configuration and ordering information

22.1 Introduction

Each device is available for production in user programmable versions (Flash) as well as in factory coded versions (ROM/FASTROM).

ST72561-Auto devices are ROM versions. ST72P561-Auto devices are Factory Advanced Service Technique ROM (FASTROM) versions: They are factory-programmed HDFlash devices.

ST72F561-Auto Flash devices are shipped to customers with a default content (FFh), while ROM 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 devices are factory-configured.

22.2 Flash devices

22.2.1 Flash configuration

The option bytes allows 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 directly the Flash devices using ICP, Flash devices are shipped to customers with a reserved internal clock source enabled. In masked ROM devices, the option bytes are fixed in hardware by the ROM code (see option list).

Option byte 0

OPT7 = **WDGHALT** *Watchdog reset on HALT*

This option bit determines if a RESET is generated when entering HALT mode while the Watchdog is active.

- 0: no reset generation when entering halt mode
- 1: reset generation when entering halt mode

OPT6 = **WDGSW** *Hardware or software watchdog*

This option bit selects the watchdog type.

- 0: hardware (watchdog always enabled)
- 1: software (watchdog to be enabled by software)

OPT5 = reserved, must be kept at default value.

OPT4 = **LVD** *Voltage detection*

This option bit enables the voltage detection block (LVD).

- 0: LVD on
- 1: LVD off

OPT3 = **PLL OFF** *PLL activation*

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.

- 0: PLL x2 enabled
- 1: PLL x2 disabled

Caution: The PLL can be enabled only if the “OSC RANGE” (OPT11:10) bits are configured to “MP - 2~4 MHz”. Otherwise, the device functionality is not guaranteed.

| | Static option byte 0 | | | | | | | | Static option byte 1 | | | | | | | |
|------------------------|----------------------|----|----------|-----|--------|-----|---|-------|----------------------|---|---------|---|----------|---|----------|------|
| | 7 | | | | | | | 0 | 7 | | | | | | 0 | |
| | WDG | | Reserved | LVD | PLLOFF | PKG | | FMP_R | AFI_MAP | | OSCTYPE | | OSCRANGE | | Reserved | RSTC |
| | HALT | SW | | | | 1 | 0 | | 1 | 0 | 1 | 0 | 1 | 0 | | |
| Default ⁽¹⁾ | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | |

1. Option bit values programmed by ST.

OPT2:1 = **PKG[1:0]** *Package selection*

These option bits select the device package.

Table 124. Package selection

| Selected package | PKG | |
|------------------|-----|---|
| | 1 | 0 |
| LQFP 64 | 1 | x |
| LQFP 44 | 0 | 1 |
| LQFP 32 | | 0 |

Note: *Pads that are not bonded to external pins are in input pull-up configuration when the package selection option bits have been properly programmed. The configuration of these pads must be kept in reset state to avoid added current consumption.*

OPT0 = **FMP_R** *Flash memory read-out protection*

Read-out protection, when selected provides a protection against program memory content extraction and against write access to Flash memory. Erasing the option bytes when the FMP_R option is selected causes the whole user memory to be erased first, and the device can be reprogrammed. Refer to [Section 3.3.1: Read-out protection](#) and the *ST7 Flash Programming Reference Manual* for more details.

0: read-out protection enabled

1: read-out protection disabled

Option byte 1

OPT7:6 = **AFI_MAP[1:0]** *AFI Mapping*

These option bits allow the mapping of some of the Alternate Functions to be changed.

Table 125. Alternate function remapping 1

| AFI mapping 1 | AFI_MAP(1) |
|--|------------|
| T16_OCMP1 on PD3 T16_OCMP2 on PD5 T16_ICAP1 on PD4 LINSI2_SCK not available LINSI2_TDO not available LINSI2_RDI not available | 0 |
| T16_OCMP1 on PB6 T16_OCMP2 on PB7 T16_ICAP1 on PC0 LINSI2_SCK on PD3 LINSI2_TDO on PD5 LINSI2_RDI on PD4 | 1 |

Table 126. Alternate function remapping 0

| AFI mapping 0 | AFI_MAP(0) |
|----------------------------|------------|
| T16_ICAP2 is mapped on PD1 | 0 |
| T16_ICAP2 is mapped on PC1 | 1 |

OPT5:4 = **OSCTYPE[1:0]** *Oscillator Type*

These option bits select the ST7 main clock source type.

Table 127. OSCTYPE selection

| Clock source | OSCTYPE | |
|--|---------|---|
| | 1 | 0 |
| Resonator oscillator | 0 | 0 |
| Reserved | | 1 |
| Reserved internal clock source (used only in ICC mode) | 1 | 0 |
| External source | | 1 |

OPT3:2 = **OSCRANGE[1:0]** *Oscillator range*

If the resonator oscillator type is selected, these option bits select the resonator oscillator. This selection corresponds to the frequency range of the resonator used. If external source is selected with the OSCTYPE option, then the OSCRANGE option must be selected with the corresponding range.

Table 128. OSCRANGE selection

| Typical frequency range | | OSCRANGE | |
|-------------------------|---------|----------|---|
| | | 1 | 0 |
| LP | 1~2 MHz | 0 | 0 |
| MP | 2~4 MHz | | 1 |

Table 128. OSCRANGE selection (continued)

| Typical frequency range | | OSCRANGE | |
|-------------------------|----------|----------|---|
| | | 1 | 0 |
| MS | 4~8 MHz | 1 | 0 |
| HS | 8~16 MHz | | 1 |

OPT1 = reserved

OPT0 = **RSTC** *RESET* clock cycle selection

This option bit selects the number of CPU cycles inserted 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.

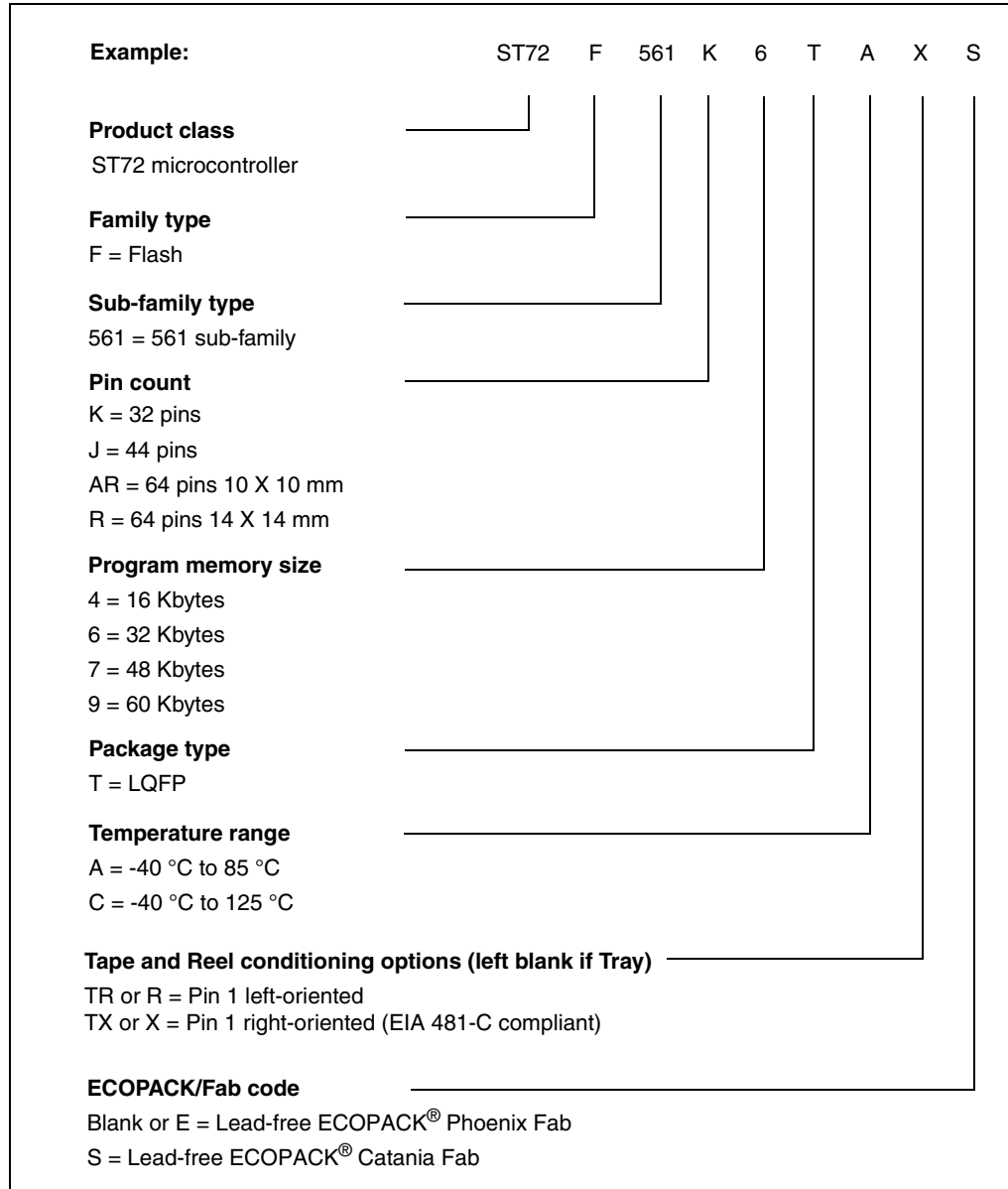
0: reset phase with 4096 CPU cycles

1: reset phase with 256 CPU cycles

22.2.2 Flash ordering information

The following [Figure 152](#) serves as a guide for ordering.

Figure 152. ST72F561xx-Auto Flash commercial product structure



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.

22.3 Transfer of customer code

Customer code is made up of the ROM/FASTROM contents and the list of the selected options (if any). The ROM/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.

The selected options are communicated to STMicroelectronics using the correctly completed OPTION LIST appended.

Refer to application note AN1635 for information on the counter listing returned by ST after code has been transferred.

The STMicroelectronics sales organization is pleased to provide detailed information on contractual points.

The following *Figure 153* serves as a guide for ordering.

Figure 153. ST72P561xxx-Auto FastROM commercial product structure

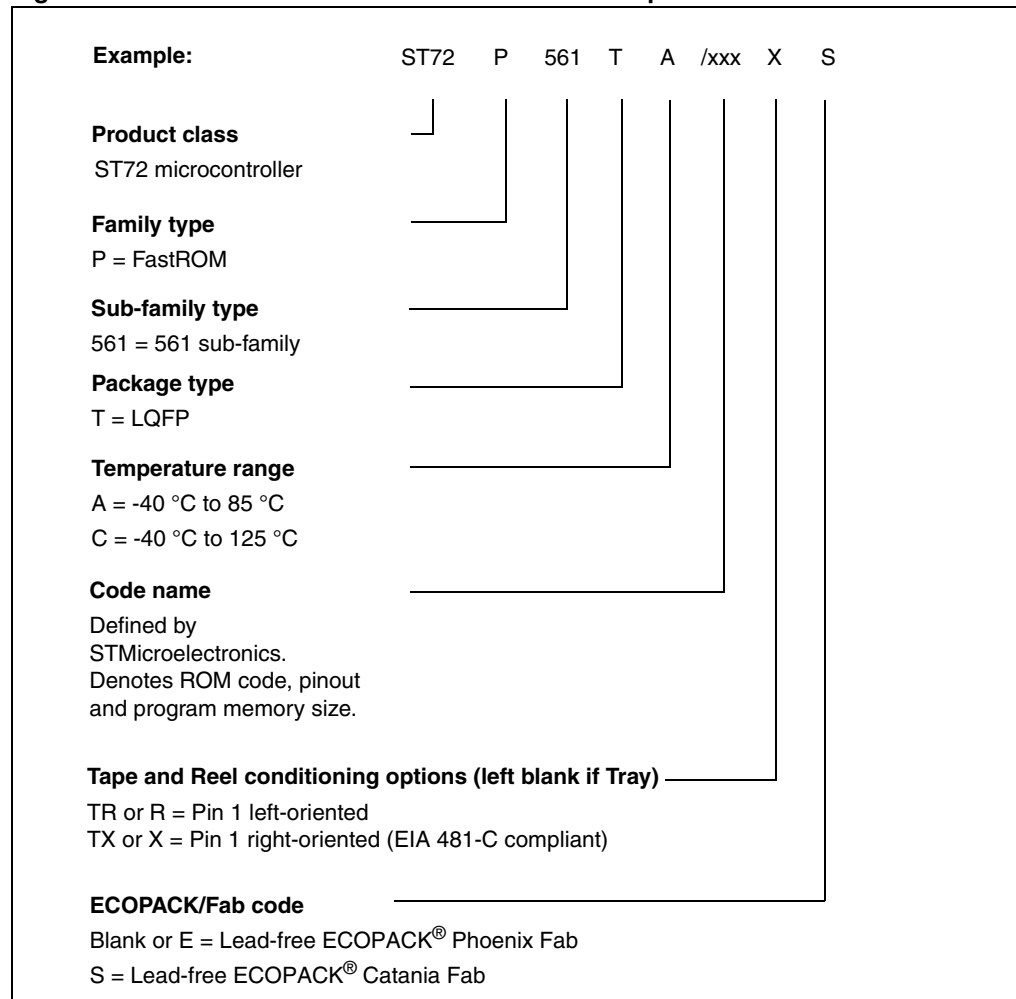
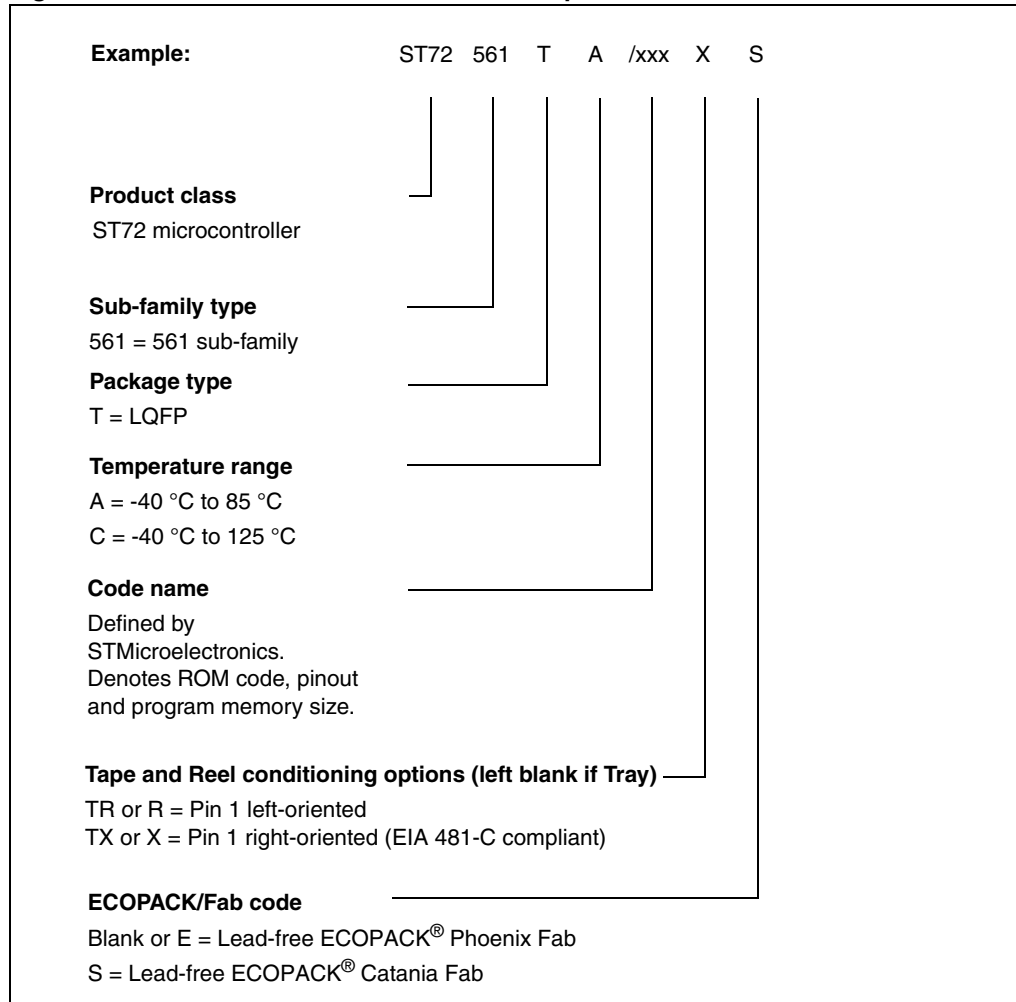


Figure 154. ST72561xx-Auto ROM commercial product structure



| ST72561-Auto MICROCONTROLLER OPTION LIST | | | | | |
|---|--|---|---|--------------------------------------|--------------------------------------|
| (Last update: March 2008) | | | | | |
| Customer Address | | | | | |
| Contact | | | | | |
| Phone No | | | | | |
| Reference/ROM Code* | | | | | |
| * The ROM/FASTROM code name is assigned by STMicroelectronics. | | | | | |
| ROM/FASTROM code must be sent in .S19 format. .Hex extension cannot be processed. | | | | | |
| Device Type/Memory Size/Package (check only one option) | | | | | |
| ROM: | Package | 60K | 48K | 32K | 16K |
| | ILQFP64 10x10: | <input type="checkbox"/> ST72561AR9 | <input type="checkbox"/> ST72561AR7 | <input type="checkbox"/> ST72561AR6 | <input type="checkbox"/> ST72561AR4 |
| | ILQFP64 14x14: ² | <input type="checkbox"/> ST72561R9 | <input type="checkbox"/> ST72561R7 | <input type="checkbox"/> ST72561R6 | <input type="checkbox"/> ST72561R4 |
| | ILQFP44: | <input type="checkbox"/> ST72561J9 | <input type="checkbox"/> ST72561J7 | <input type="checkbox"/> ST72561J6 | <input type="checkbox"/> ST72561J4 |
| | ILQFP32: | <input type="checkbox"/> ST72561K9 | <input type="checkbox"/> ST72561K7 | <input type="checkbox"/> ST72561K6 | <input type="checkbox"/> ST72561K4 |
| FASTROM: | Package | 60K | 48K | 32K | 16K16K |
| | ILQFP64 10x10: | <input type="checkbox"/> ST72P561AR9 | <input type="checkbox"/> ST72P561AR7 | <input type="checkbox"/> ST72P561AR6 | <input type="checkbox"/> ST72P561AR4 |
| | ILQFP64 14x14: ² | <input type="checkbox"/> ST72P561R9 | <input type="checkbox"/> ST72P561R7 | <input type="checkbox"/> ST72P561R6 | <input type="checkbox"/> ST72P561R4 |
| | ILQFP44: | <input type="checkbox"/> ST72P561J9 | <input type="checkbox"/> ST72P561J7 | <input type="checkbox"/> ST72P561J6 | <input type="checkbox"/> ST72P561J4 |
| | ILQFP32: | <input type="checkbox"/> ST72P561K9 | <input type="checkbox"/> ST72P561K7 | <input type="checkbox"/> ST72P561K6 | <input type="checkbox"/> ST72P561K4 |
| Conditioning: | <input type="checkbox"/> Tray | <input type="checkbox"/> Tape & Reel | | | |
| Special Marking: | <input type="checkbox"/> No | <input type="checkbox"/> Yes " _____ " (LQFP32 7 characters, other packages 10 characters max.) | | | |
| Authorized characters are letters, digits, '.', '-', '/' and spaces only. | | | | | |
| Temperature Range: | <input type="checkbox"/> A (-40°C to +85°C) | | | | |
| | <input type="checkbox"/> C (-40°C to +125°C) | | | | |
| Clock Source Selection: | <input type="checkbox"/> Resonator | | | | |
| | <input type="checkbox"/> External Source | | | | |
| Oscillator/External source range: | <input type="checkbox"/> LP: Low power (1 to 2 MHz) | | | | |
| | <input type="checkbox"/> MP: Medium power (2 to 4 MHz) | | | | |
| | <input type="checkbox"/> MS: Medium speed (4 to 8 MHz) | | | | |
| | <input type="checkbox"/> HS: High speed (8 to 16 MHz) | | | | |
| LVD | <input type="checkbox"/> Disabled | | <input type="checkbox"/> Enabled | | |
| PLL1 | <input type="checkbox"/> Disabled | | <input type="checkbox"/> Enabled | | |
| Watchdog Selection | <input type="checkbox"/> Software Activation | | <input type="checkbox"/> Hardware Activation | | |
| Watchdog Reset on Halt | <input type="checkbox"/> Reset | | <input type="checkbox"/> No Reset | | |
| Read-out Protection | <input type="checkbox"/> Disabled | | <input type="checkbox"/> Enabled | | |
| Reset Delay | <input type="checkbox"/> 256 Cycles | | <input type="checkbox"/> 4096 Cycles | | |
| LINSCI2 Mapping | <input type="checkbox"/> Not available (AFIMAP[1] = 0) | | <input type="checkbox"/> Mapped (AFIMAP[1] = 1) | | |
| T16_ICAP2 Mapping | <input type="checkbox"/> On PD1 (AFIMAP[0] = 0) | | <input type="checkbox"/> On PC1 (AFIMAP[0] = 1) | | |
| Comments: | | | | | |
| Supply Operating Range in the application: | | | | | |
| Notes | | | | | |
| Signature | | | | | |
| Date | | | | | |
| 1 If PLL is enabled, medium power (2 to 4 MHz range) has to be selected (MP) | | | | | |
| 2 LQFP64 14x14 available only in automotive version. | | | | | |
| Please download the latest version of this option list from: www.st.com | | | | | |

23 Development tools

Full details of tools available for the ST7 from third party manufacturers can be obtained from the STMicroelectronics Internet site: www.st.com.

Tools from isystem and hitex include C compilers, emulators and gang programmers.

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.

ST programming tools

- ST7MDT25-EPB: for in-socket or ICC programming
- ST7-STICK: for ICC programming

24 Important notes

24.1 All devices

24.1.1 RESET pin protection with LVD enabled

As mentioned in note 2 below [Figure 135](#), 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.

24.1.2 Clearing active interrupts outside interrupt routine

When an active interrupt request occurs at the same time as the related flag or interrupt mask is being cleared, the CC register may be corrupted.

Concurrent interrupt context

The symptom does not occur when the interrupts are handled normally, that is, when:

- The interrupt request is cleared (flag reset or interrupt mask) within its own interrupt routine
- The interrupt request is cleared (flag reset or interrupt mask) within any interrupt routine
- The interrupt request is cleared (flag reset or interrupt mask) 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 1:

```
SIM
reset flag or interrupt mask
RIM
```

Nested interrupt context

The symptom does not occur when the interrupts are handled normally, that is, when:

- The interrupt request is cleared (flag reset or interrupt mask) within its own interrupt routine
- The interrupt request is cleared (flag reset or interrupt mask) within any interrupt routine with higher or identical priority level
- The interrupt request is cleared (flag reset or interrupt mask) 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 flag or interrupt mask
POP CC
```

24.1.3 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 ensure 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 and if it is '1' this 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 writing to PxOR or PxDDR is done 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' this 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 interrupt disabled/enabled).

Case 1: writing to PxOR or PxDDR with global interrupts enabled:

```
LD A, #01
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
.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
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
```

24.1.4 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.

24.1.5 Header time-out does not prevent wake-up from mute mode

Normally, when LINSICI is configured in LIN slave mode, if a header time-out occurs during a LIN header reception (that is, header length > 57 bits), the LIN Header Error bit (LHE) is set, an interrupt occurs to inform the application but the LINSICI should stay in mute mode, waiting for the next header reception.

Problem description

The LINSICI sampling period is $T_{bit} / 16$. If a LIN Header time-out occurs between the 9th and the 15th sample of the Identifier Field Stop Bit (refer to [Figure 155](#)), the LINSICI wakes up from mute mode. Nevertheless, LHE is set and LIN Header Detection Flag (LHDF) is kept cleared.

In addition, if LHE is reset by software before this 15th sample (by accessing the SCISR register and reading the SCIDR register in the LINSICI interrupt routine), the LINSICI will generate another LINSICI interrupt (due to the RDRF flag setting).

Impact on application

Software may execute the interrupt routine twice after header reception.

Moreover, in reception mode, as the receiver is no longer in mute mode, an interrupt will be generated on each data byte reception.

Workaround

The problem can be detected in the LINSICI interrupt routine. In case of timeout error (LHE is set and LHLR is loaded with 00h), the software can check the RWU bit in the SCICR2 register. If RWU is cleared, it can be set by software. Refer to [Figure 156](#). Workaround is shown in bold characters.

Figure 155. Header reception event sequence

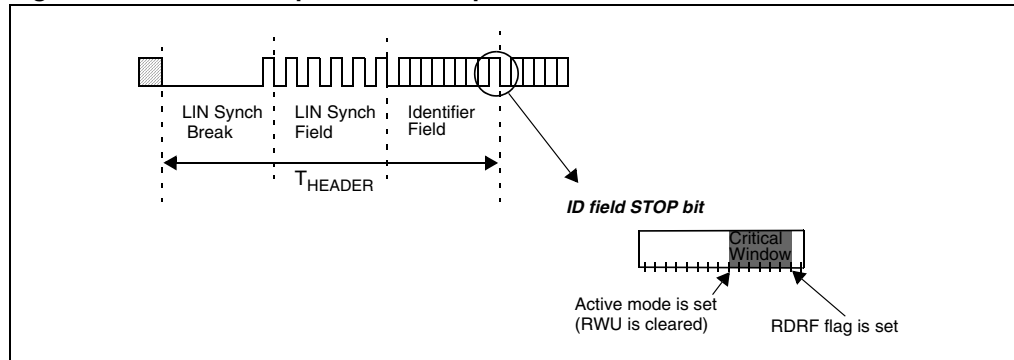


Figure 156. LINSICI interrupt routine

```

@interrupt void LINSICI_IT ( void ) /* LINSICI interrupt routine */
{
    /* clear flags */
    SCISR_buffer = SCISR;
    SCIDR_buffer = SCIDR;

    if ( SCISR_buffer & LHE ) /* header error ? */
    {
        if (!LHLR) /* header time-out? */
        {
            if ( !(SCICR2 & RWU) ) /* active mode ? */
            {
                _asm("sim"); /* disable interrupts */
                SCISR;
                SCIDR; /* Clear RDRF flag */
                SCICR2 |= RWU; /* set mute mode */
                SCISR;
                SCIDR; /* Clear RDRF flag */
                SCICR2 |= RWU; /* set mute mode */
                _asm("rim"); /* enable interrupts */
            }
        }
    }
}

```

Example using Cosmic compiler syntax

24.1.6 CAN FIFO corruption

The beCAN FIFO gets corrupted when a message is received and simultaneously a message is released while FMP = 2. For details and a description of the workaround refer to [FIFO corruption](#).

24.2 Flash/FastROM devices only

24.2.1 LINSICI wrong break duration

SCI mode

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 generate 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 ($f_{\text{CPU}} = 8 \text{ MHz}$ and $\text{SCIBRR} = 0xC9$), 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

LIN mode

If the LINE bit in the SCICR3 is set and the M bit in the SCICR1 register is reset, the LINSICI is in LIN master mode. 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:

- 24 bits instead of 13 bits

Occurrence

The occurrence of the problem is random and proportional to the baud rate. With a transmit frequency of 19200 baud ($f_{\text{CPU}} = 8 \text{ MHz}$ and $\text{SCIBRR} = 0xC9$), the wrong break duration occurrence is around 1%.

Analysis

The LIN protocol specifies a minimum of 13 bits for the break duration, but there is no maximum value. Nevertheless, the maximum length of the header is specified as $(14 + 10 + 10 + 1) \times 1.4 = 49$ bits. This is composed of:

- the synch break field (14 bits)
- the synch field (10 bits)
- the identifier field (10 bits)

Every LIN frame starts with a break character. Adding an idle character increases the length of each header by 10 bits. When the problem occurs, the header length is increased by 11 bits and becomes $((14 + 11) + 10 + 10 + 1) = 45$ bits.

To conclude, the problem is not always critical for LIN communication if the software keeps the time between the sync field and the ID smaller than 4 bits, that is, 208 μ s at 19200 baud.

The workaround is the same as for SCI mode but considering the low probability of occurrence (1%), it may be better to keep the break generation sequence as it is.

24.2.2 16-bit and 8-bit timer PWM mode

In PWM mode, the first PWM pulse is missed after writing the value FFFCh in the OC1R or OC2R register.

24.3 ROM devices only

24.3.1 16-bit timer PWM mode buffering feature change

In all devices, the frequency and period of the PWM signal are controlled by comparing the counter with a 16-bit buffer updated by the OCiHR and OCiLR registers. In ROM devices, contrary to the description in *Pulse width modulation mode*, the output compare function is not inhibited after a write instruction to the OCiHR register. Instead the buffer update at the end of the PWM period is inhibited until OCiLR is written. This improved buffer handling is fully compatible with applications written for Flash devices.

25 Revision history

Table 129. Document revision history

| Date | Revision | Changes |
|-------------|----------|---|
| 03-May-2004 | 1 | <p>Added TQFP 10x10 package</p> <p>Removed internal RC</p> <p>Updated Figure 11: Clock, reset and supply block diagram</p> <p>Added note on monotonous V_{DD} ramp on Section 5.6.1: Low voltage detector (LVD)</p> <p>Added caution ART Ext clock not available in HALT see Section 11: PWM auto-reload timer (ART)</p> <p>Added note "Once the OCIE bit is set both output compare features may trigger..." and "Once the ICIE bit is set both input capture features may trigger..." in 8-bit timer Section 13: 8-bit timer (TIM8).</p> <p>Changed clock from fcpu/8000 to fosc2/8000 in Section 13: 8-bit timer (TIM8)</p> <p>Changed description of CSR register to read only except bit 2 R/W Section 13: 8-bit timer (TIM8)</p> <p>Added note to SPI slave freq. and updated Master mode procedure in Section 14: Serial peripheral interface (SPI)</p> <p>Changed description of NF bit in Section 15.10: LIN mode register description</p> <p>Removed "Configurable timer resolution" under "Time triggered communication option" from Section 17: beCAN controller (beCAN)</p> <p>Added Clearing interrupts limitation and SCI wrong break duration to Chapter 24: Important notes</p> <p>Removed beCAN Time triggered mode feature from Section 17: beCAN controller (beCAN)</p> <p>Renamed CMSR RX and TX bits to REC and TRAN in Section 17: beCAN controller (beCAN)</p> <p>Added beCAN FIFO corruption limitation FIFO corruption</p> <p>Modified I_{INJ} for Port B3 in Section 20.9.1: General characteristics</p> |
| 11-May-2004 | 2 | <p>Modified Clearing interrupts limitation in Chapter 24: Important notes</p> |
| 12-May-2005 | 3 | <p>LINSCI™ changed to LINSICI™ throughout document.</p> <p>Changed name of WWDGR register to WDGWR in Table 4: Hardware register map</p> <p>Changed Static power consumption to 200 uA typ in Section 20.9.1: General characteristics</p> <p>Modified Readout Protection description in Section 3.3.1: Read-out protection</p> <p>Added Section 24.1: All devices</p> <p>Modified Figure 135: RESET pin protection when LVD is enabled and Figure 136: RESET pin protection when LVD is disabled and related notes</p> <p>Added 48K ROM version in Table 2: Product overview, Figure 5: Memory map and Figure 154: ST72561xxx-Auto ROM commercial product structure</p> |
| 24-Oct-2005 | 4 | <p>Added standard version 16K ROM/Flash devices</p> <p>Modified data retention in Section 20.7: Memory characteristics</p> <p>Added "6" and "3" standard version device type coding to Figure 152: ST72F561xxx-Auto Flash commercial product structure and Figure 154: ST72561xxx-Auto ROM commercial product structure</p> <p>Modified power consumption Section 20.4: Supply current characteristics</p> <p>Added CDM in Electrostatic discharge (ESD)</p> <p>Added "External interrupt missed" Section 24.1.3: External interrupt missed</p> |

Table 129. Document revision history (continued)

| Date | Revision | Changes |
|-------------|----------|---|
| 23-Jun-2006 | 5 | <p>Replaced TQFP with LQFP packages throughout document</p> <p>Changed device summary on Table 2: Product overview</p> <p>Changed Section 8.2.1: Input modes</p> <p>Changed title of Section 8.4: I/O port register configurations from “I/O Port Implementation” to “I/O Port Register Configurations”</p> <p>Changed Master mode operation</p> <p>Corrected name of bit 5 in SPICSR register in Table 59: SPI register map and reset values</p> <p>Changed Section 20.5.2: PLL characteristics</p> <p>Added links to Table 111: Absolute maximum ratings and Table 112: Electrical sensitivities in Section 20.8.3: Absolute maximum ratings (electrical sensitivity)</p> <p>Removed EMC protection circuitry in Figure 136: RESET pin protection when LVD is disabled (device works correctly without these components)</p> <p>Changed Section 20.12.1: SPI - serial peripheral interface</p> <p>Changed title of Figure 147: 64-pin low profile quad flat package (14x14)</p> <p>Changed title of Figure 148: 32-pin low profile quad flat package (7x7)</p> <p>Changed title of Figure 149: 44-pin low profile quad flat package (10x10)</p> <p>Changed notes in Table 21.3: Thermal characteristics</p> <p>Changed AFI mapping for Option byte 1</p> <p>Changed Figure 154: ST72561xxx-Auto ROM commercial product structure, Figure 153: ST72P561xxx-Auto FastROM commercial product structure and Figure 152: ST72F561xxx-Auto Flash commercial product structure</p> <p>Changed Section 20.12.1: SPI - serial peripheral interface</p> <p>Deleted Section 15.1.5 “Clearing active interrupts outside interrupt routine” (text already exists in Section 24.1.2: Clearing active interrupts outside interrupt routine)</p> <p>Added Section 24.1.5: Header time-out does not prevent wake-up from mute mode</p> |
| 11-Mar-2008 | 6 | <p>Removed ‘F’ from root part numbers and removed references to ‘automotive only’ devices in the Table 2: Product overview</p> <p>Removed ‘mcu’ from the URL reference in ST72561-Auto MICROCONTROLLER OPTION LIST and Chapter 23: Development tools</p> <p>Removed section on “SOLDERING AND GLUEABILITY INFORMATION” and replaced with Section 21.4: Packaging for automatic handling</p> <p>Removed references to standard device temperature codes from Figure 154: ST72561xxx-Auto ROM commercial product structure, Figure 153: ST72P561xxx-Auto FastROM commercial product structure and Figure 152: ST72F561xxx-Auto Flash commercial product structure</p> <p>Made small editing changes throughout document to improve readability</p> |
| 02-Aug-2010 | 7 | <p>Updated following figures:</p> <ul style="list-style-type: none"> – Figure 152: ST72F561xxx-Auto Flash commercial product structure – Figure 153: ST72P561xxx-Auto FastROM commercial product structure – Figure 154: ST72561xxx-Auto ROM commercial product structure <p>Added Chapter 21.4: Packaging for automatic handling</p> |
| 03-Jan-2011 | 8 | <p>Reinstated : Chapter 17: beCAN controller (beCAN)</p> |

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.

© 2011 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