Welcome to E-XFL.COM

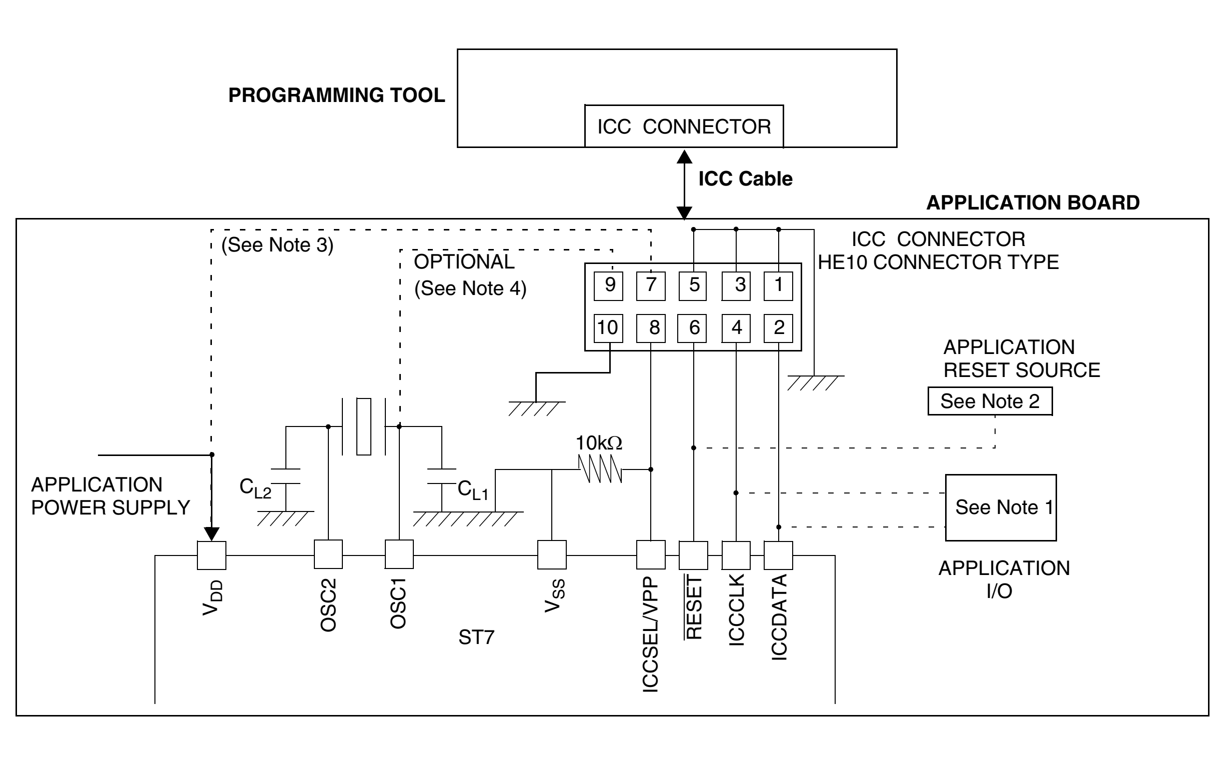
## What is "Embedded - Microcontrollers"?

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

## Applications of "Embedded - Microcontrollers"

| Details | |
|---|---|
| Product Status | Obsolete |
| Core Processor | ST7 |
| Core Size | 8-Bit |
| Speed | 8MHz |
| Connectivity | CANbus, I²C, SCI, SPI |
| Peripherals | LVD, POR, PWM, WDT |
| Number of I/O | 48 |
| Program Memory Size | 60KB (60K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 2K x 8 |
| Voltage - Supply (Vcc/Vdd) | 3.8V ~ 5.5V |
| Data Converters | A/D 16x10b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 125°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 64-LQFP |
| Supplier Device Package | - |
| Purchase URL | https://www.e-xfl.com/product-detail/stmicroelectronics/st72f521ar9tce |

**Figure 6. Typical ICC interface**




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

## 4.5 ICP (in-circuit programming)

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

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

When using an STMicroelectronics or third-party programming tool that supports ICP and the specific microcontroller device, the user needs only to implement the ICP hardware interface on the application board (see *Figure 6*). For more details on the pin locations, refer to the device pinout description.

## 10.9 Register description

### 10.9.1 Control register (WDGCR)

WDGCR Reset value: 0111 1111 (7Fh)

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

**Table 36. WDGCR register description**

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

**Table 37. Watchdog timer register map and reset values**

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

# 12.3 ART registers

## 12.3.1 Control/status register (ARTCSR)

ARTCSR Reset value: 0000 0000 (00h)

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

**Table 45. ARTCSR register description**

| Bit | Name | Function |
|---|---|---|
| 7 | EXCL | *External Clock*<br>This bit is set and cleared by software. It selects the input clock for the 7-bit prescaler.<br>0: CPU clock<br>1: External clock |
| 6:4 | CC[2:0] | *Counter Clock Control*<br>These bits are set and cleared by software. They determine the prescaler division ratio from $f_{INPUT}$ (see *Table 46*). |
| 3 | TCE | *Timer Counter Enable*<br>This bit is set and cleared by software. It puts the timer in the lowest power consumption mode.<br>0: Counter stopped (prescaler and counter frozen)<br>1: Counter running |
| 2 | FCRL | *Force Counter Re-Load*<br>This bit is write-only and any attempt to read it will yield a logical zero. When set, it causes the contents of ARTARR register to be loaded into the counter, and the content of the prescaler register to be cleared in order to initialize the timer before starting to count. |
| 1 | OIE | *Overflow Interrupt Enable*<br>This bit is set and cleared by software. It allows to enable/disable the interrupt which is generated when the OVF bit is set.<br>0: Overflow Interrupt disable<br>1: Overflow Interrupt enable |
| 0 | OVF | *Overflow Flag*<br>This bit is set by hardware and cleared by software reading the ARTCSR register. It indicates the transition of the counter from FFh to the ARTARR value.<br>0: New transition not yet reached<br>1: Transition reached |

**Table 46. Prescaler selection for ART**

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

## 13.4 Low power modes

**Table 56. Effect of low power modes on 16-bit timer**

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

## 13.5 Interrupts

**Table 57. 16-bit timer interrupt control/wake-up capability**

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

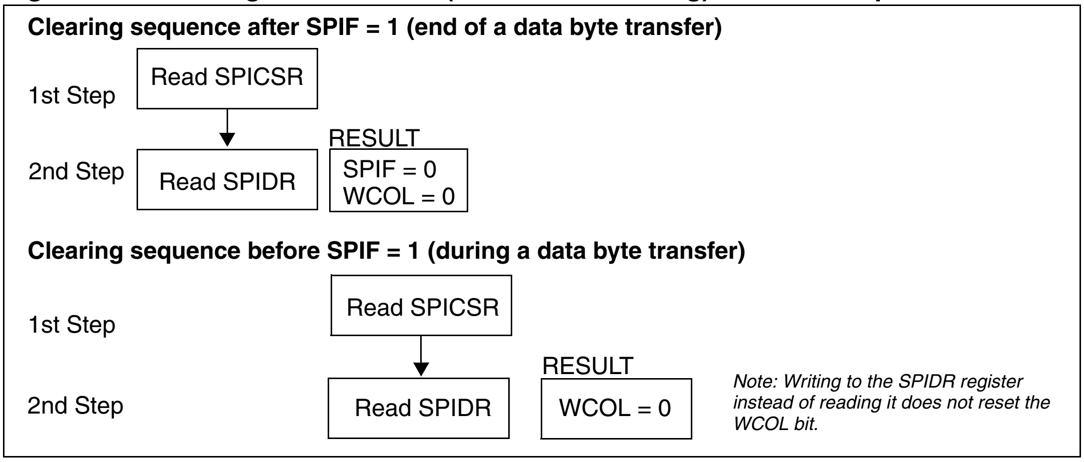
*Note: The 16-bit timer interrupt events are connected to the same interrupt vector (see Chapter 7: Interrupts on page 52). These events generate an interrupt if the corresponding Enable Control Bit is set and the interrupt mask in the CC register is reset (RIM instruction).*

## 13.6 Summary of timer modes

**Table 58. 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) | | | | |

**Figure 60. Clearing the WCOL bit (Write Collision Flag) software sequence**




## 14.5.4 Single master systems

A typical single master system may be configured, using an MCU as the master and four MCUs as slaves (see *Figure 61*).

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

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

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

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

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

**Figure 61. Single master / multiple slave configuration**

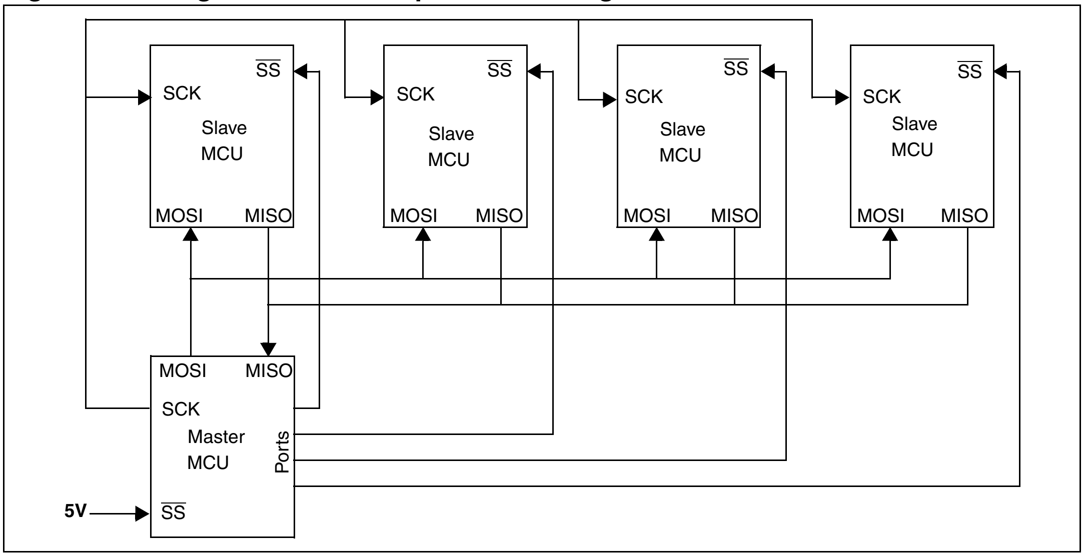


**Table 76. SCIBRR register description**

| Bit | Name | Function |
|---|---|---|
| 7:6 | SCP[1:0] | *First SCI Prescaler*<br>These 2 prescaling bits allow several standard clock division ranges.<br>00: PR prescaling factor = 1<br>01: PR prescaling factor = 3<br>10: PR prescaling factor = 4<br>11: PR prescaling factor = 13 |
| 5:3 | SCT[2:0] | *SCI Transmitter rate divisor*<br>These 3 bits, in conjunction with the SCP1 and SCP0 bits define the total division applied to the bus clock to yield the transmit rate clock in conventional Baud Rate Generator mode.<br>000: TR dividing factor = 1<br>001: TR dividing factor = 2<br>010: TR dividing factor = 4<br>011: TR dividing factor = 8<br>100: TR dividing factor = 16<br>101: TR dividing factor = 32<br>110: TR dividing factor = 64<br>111: TR dividing factor = 128 |
| 2:0 | SCR[2:0] | *SCI Receiver rate divisor*<br>These 3 bits, in conjunction with the SCP[1:0] bits define the total division applied to the bus clock to yield the receive rate clock in conventional Baud Rate Generator mode.<br>000: RR dividing factor = 1<br>001: RR dividing factor = 2<br>010: RR dividing factor = 4<br>011: RR dividing factor = 8<br>100: RR dividing factor = 16<br>101: RR dividing factor = 32<br>110: RR dividing factor = 64<br>111: RR dividing factor = 128 |

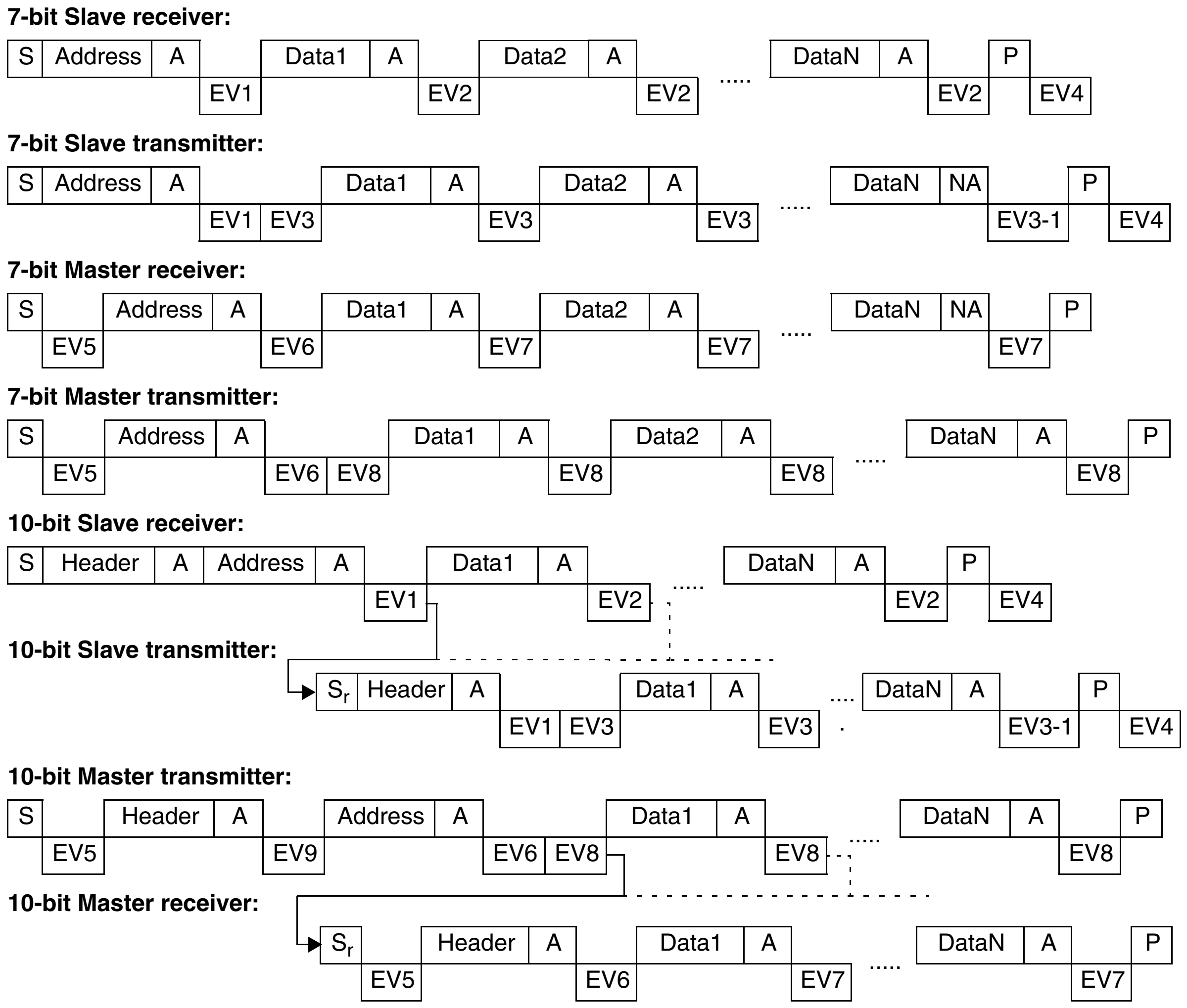
### 15.7.6 Extended receive prescaler division register (SCIERPR)

This register allows setting of the extended prescaler rate division factor for the receive circuit.

SCIERPR Reset value: 0000 0000 (00h)

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

**Figure 68. Transfer sequencing**




**Legend:**

S = Start, Sr = Repeated Start, P = Stop, A = Acknowledge, NA = Non-acknowledge, EVx = Event (with interrupt if ITE = 1)

EV1: EVF = 1, ADSL = 1, cleared by reading SR1 register.

EV2: EVF = 1, BTF = 1, cleared by reading SR1 register followed by reading DR register.

EV3: EVF = 1, BTF = 1, cleared by reading SR1 register followed by writing DR register.

EV3-1: EVF = 1, AF = 1, BTF = 1; AF is cleared by reading SR1 register. BTF is cleared by releasing the lines (STOP = 1, STOP = 0) or by writing DR register (DR = FFh). ***Note:*** *If lines are released by STOP = 1, STOP = 0, the subsequent EV4 is not seen.*

EV4: EVF = 1, STOPF = 1, cleared by reading SR2 register.

EV5: EVF = 1, SB = 1, cleared by reading SR1 register followed by writing DR register.

EV6: EVF = 1, cleared by reading SR1 register followed by writing CR register (for example PE = 1).

EV7: EVF = 1, BTF = 1, cleared by reading SR1 register followed by reading DR register.

EV8: EVF = 1, BTF = 1, cleared by reading SR1 register followed by writing DR register.

EV9: EVF = 1, ADD10 = 1, cleared by reading SR1 register followed by writing DR register.

Table 84. SR1 register description (continued)

| Bit | Name | Function |
|---|---|---|
| 1 | M/SL | *Master/Slave*<br>This bit is set by hardware as soon as the interface is in Master mode (writing START = 1). It is cleared by hardware after detecting a Stop condition on the bus or a loss of arbitration (ARLO = 1). It is also cleared when the interface is disabled (PE = 0).<br>0: Slave mode<br>1: Master mode |
| 0 | SB | *Start bit (Master mode)*<br>This bit is set by hardware as soon as the Start condition is generated (following a write START = 1). An interrupt is generated if ITE = 1. It is cleared by software reading SR1 register followed by writing the address byte in DR register. It is also cleared by hardware when the interface is disabled (PE = 0).<br>0: No Start condition<br>1: Start condition generated |

### 16.7.3 I²C status register 2 (SR2)

SR2 Reset value: 0000 0000 (00h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | | AF | STOPF | ARLO | BERR | GCAL |
| - | | | RO | RO | RO | RO | RO |

Table 85. SR2 register description

| Bit | Name | Function |
|---|---|---|
| 7:5 | - | Reserved. Forced to 0 by hardware. |
| 4 | AF | *Acknowledge failure*<br>This bit is set by hardware when no acknowledge is returned. An interrupt is generated if ITE = 1. It is cleared by software reading SR2 register or by hardware when the interface is disabled (PE = 0).<br>The SCL line is not held low while AF = 1 but by other flags (SB or BTF) that are set at the same time.<br>0: No acknowledge failure<br>1: Acknowledge failure<br>*Note: When an AF event occurs, the SCL line is not held low; however, the SDA line can remain low if the last bits transmitted are all 0. It is then necessary to release both lines by software.* |
| 3 | STOPF | *Stop detection (Slave mode)*<br>This bit is set by hardware when a Stop condition is detected on the bus after an acknowledge (if ACK = 1). An interrupt is generated if ITE = 1. It is cleared by software reading SR2 register or by hardware when the interface is disabled (PE = 0).<br>The SCL line is not held low while STOPF = 1.<br>0: No Stop condition detected<br>1: Stop condition detected |

### Baud rate prescaler register (BRPR)

BRPR Reset value: 00h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| RJW[1:0] | | BRP[5:0] | | | | | |
| R/W_Standby mode | | R/W_Standby mode | | | | | |

**Table 94. BRPR register description**

| Bit | Name | Function |
|---|---|---|
| 7:6 | RJW[1:0] | *Resynchronization Jump Width*<br>These bits determine the maximum number of time quanta by which a bit period may be shortened or lengthened to achieve resynchronization.<br>$t_{RJW} = t_{CAN} * (RJW + 1)$ |
| 5:0 | BRP[5:0] | *Baud rate prescaler*<br>These bits determine the CAN system clock cycle time or time quanta which is used to build up the individual bit timing.<br>$t_{CAN} = t_{CPU} * (BRP + 1)$<br>Where $t_{CPU}$ = time period of the CPU clock. |

The resulting baud rate can be computed by the formula:

$$BR = \frac{1}{t_{CPU} \times (BRP + 1) \times (BS1 + BS2 + 3)}$$

*Note: Writing to this register is allowed only in Standby mode to prevent any accidental CAN protocol violation through programming errors.*

### Bit timing register (BTR)

BTR Reset value: 23h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | BS2[2:0] | | | BS1[3:0] | | | |
| - | R/W_Standby mode | | | R/W_Standby mode | | | |

**Table 95. BTR register description**

| Bit | Name | Function |
|---|---|---|
| 7 | - | Reserved; must be kept at '0' |
| 6:4 | BS2[2:0] | *Bit Segment 2*<br>These bits determine the length of Bit Segment 2.<br>$t_{BS2} = t_{CAN} * (BS2 + 1)$ |
| 3:0 | BS1[3:0] | *Bit Segment 1*<br>These bits determine the length of Bit Segment 1.<br>$t_{BS1} = t_{CAN} * (BS1 + 1)$ |

*Note: Writing to this register is allowed only in Standby mode to prevent any accidental CAN protocol violation through programming errors.*

### Buffer control/status registers (BCSRx)

BCSRx Reset value: 00h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | ACC | RDY | BUSY | LOCK |
| - | | | | RO | RC | RO | RSC |

**Table 104. BCSRx register description**

| Bit | Name | Function |
|---|---|---|
| 7:4 | - | Reserved; must be kept at '0' |
| 3 | ACC | *Acceptance Code*<br>Set by hardware with the message identifier of the highest priority filter which accepted the message stored in the buffer.<br>0: Match for Filter/Mask0. Possible match for Filter/Mask1.<br>1: No match for Filter/Mask0 and match for Filter/Mask1.<br>Reset by hardware when either RDY or RXIF is reset. |
| 2 | RDY | *Message Ready*<br>Set by hardware to signal that a new error-free message is available (LOCK = 0) or that a transmission request is pending (LOCK = 1).<br>Cleared by software when LOCK = 0 to release the buffer and to clear the corresponding RXIF bit in the Interrupt Status Register.<br>Cleared by hardware when LOCK = 1 to indicate that the transmission request has been serviced or cancelled. |
| 1 | BUSY | *Busy Buffer*<br>Set by hardware when the buffer is being filled (LOCK = 0) or emptied (LOCK = 1) and reset after the 2nd intermission bit.<br>Reset by hardware when the buffer is not accessed by the CAN core for transmission nor reception purposes. |
| 0 | LOCK | *Lock Buffer*<br>Set by software to lock a buffer. No more message can be received into the buffer thus preserving its content and making it available for transmission.<br>Cleared by software to make the buffer available for reception. Cancels any pending transmission request.<br>Cleared by hardware once a message has been successfully transmitted provided the early transmit interrupt mode is on. Left untouched otherwise.<br>Note that in order to prevent any message corruption or loss of context, LOCK cannot be set nor reset while BUSY is set. Trying to do so will result in LOCK not changing state. |

### Filter high registers (FHRx)

FHRx Reset value: Undefined

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| FIL[11:4] | | | | | | | |
| R/W | | | | | | | |

**Table 105. FHRx register description**

| Bit | Name | Function |
|---|---|---|
| 7:0 | FIL[11:4] | *Acceptance Filter*<br>These are the most significant 8 bits of a 12-bit message filter. The acceptance filter is compared bit by bit with the identifier and the RTR bit of the incoming message. If there is a match for the set of bits specified by the acceptance mask then the message is stored in a receive buffer. |

### Filter low registers (FLRx)

FLRx Reset value: Undefined

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| FIL[3:0] | | | | Reserved | | | |
| R/W | | | | - | | | |

**Table 106. FLRx register description**

| Bit | Name | Function |
|---|---|---|
| 7:4 | FIL[3:0] | *Acceptance Filter*<br>These are the least significant 4 bits of a 12-bit message filter. |
| 3:0 | - | Reserved; must be kept at ‘0’ |

### Mask high registers (MHRx)

MHRx Reset value: Undefined

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| MSK[11:4] | | | | | | | |
| R/W | | | | | | | |

**Table 107. MHRx register description**

| Bit | Name | Function |
|---|---|---|
| 7:0 | MSK[11:4] | *Acceptance Mask*<br>These are the most significant 8 bits of a 12-bit message mask. The acceptance mask defines which bits of the acceptance filter should match the identifier and the RTR bit of the incoming message.<br>MSK[i] = 0: Don’t care<br>MSK[i] = 1: Match required |

## 20.7 EMC (electromagnetic compatibility) characteristics

Susceptibility tests are performed on a sample basis during product characterization.

### 20.7.1 Functional EMS (electromagnetic susceptibility)

Based on a simple running application on the product (toggling two LEDs through I/O ports), the product is stressed by two electromagnetic events until a failure occurs (indicated by the LEDs).

- **ESD**: Electrostatic discharge (positive and negative) is applied on all pins of the device until a functional disturbance occurs. This test conforms with the IEC 1000-4-2 standard.
- **FTB**: A burst of fast transient voltage (positive and negative) is applied to $V_{DD}$ and $V_{SS}$ through a 100pF capacitor until a functional disturbance occurs. This test conforms with the IEC 1000-4-4 standard.

A device reset allows normal operations to be resumed. The test results given in *Table 142* below are based on the EMS levels and classes defined in application note AN1709.

#### Designing hardened software to avoid noise problems

EMC characterization and optimization are performed at component level with a typical application environment and simplified MCU software. It should be noted that good EMC performance is highly dependent on the user application and the software in particular.

Therefore it is recommended that the user applies EMC software optimization and prequalification tests in relation with the EMC level requested for his application.

**Software recommendations**

The software flowchart must include the management of runaway conditions such as:

- Corrupted program counter
- Unexpected reset
- Critical Data corruption (control registers...)

**Prequalification trials**

Most of the common failures (unexpected reset and program counter corruption) can be reproduced by manually forcing a low state on the $\overline{\text{RESET}}$ pin or the oscillator pins for 1 second.

To complete these trials, ESD stress can be applied directly on the device, over the range of specification values. When unexpected behavior is detected, the software can be hardened to prevent unrecoverable errors occurring (see application note AN1015).

# 20.9 Control pin characteristics

## 20.9.1 Asynchronous $\overline{RESET}$ pin

Subject to general operating conditions for $V_{DD}$, $f_{CPU}$, and $T_A$ unless otherwise specified.

**Table 148. Asynchronous $\overline{RESET}$ pin characteristics**

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $V_{IL}$ | Input low level voltage(1) | | | | 0.16xV$_{DD}$ | V |
| $V_{IH}$ | Input high level voltage(1) | | 0.85xV$_{DD}$ | | | V |
| $V_{hys}$ | Schmitt trigger voltage hysteresis(2) | | | 2.5 | | V |
| $V_{OL}$ | Output low level voltage(3) | $V_{DD}$ = 5V, $I_{IO}$ = +2mA | | 0.2 | 0.5 | V |
| $I_{IO}$ | Input current on $\overline{RESET}$ pin | | | 2 | | mA |
| $R_{ON}$ | Weak pull-up equivalent resistor | | 20 | 30 | 120 | kΩ |
| $t_{w(RSTL)out}$ | Generated reset pulse duration | Stretch applied on external pulse | 0 | | 42(4) | µs |
| | | Internal reset sources | 20 | 30 | 42(4) | 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* and the sum of $I_{IO}$ (I/O ports and control pins) must not exceed $I_{VSS}$.
4. Data guaranteed by design, not tested in production.
5. To guarantee the reset of the device, a minimum pulse has to be applied to the $\overline{RESET}$ pin. All short pulses applied on the $\overline{RESET}$ pin with a duration below $t_{h(RSTL)in}$ can be ignored.
6. The reset network (the resistor and two capacitors) protects the device against parasitic resets, especially in noisy environments.

### 20.11.3 CAN - Controller area network interface

Subject to general operating conditions for $V_{DD}$, $f_{OSC}$, and $T_A$ unless otherwise specified. Refer to *Chapter 9: I/O ports* for more details on the input/output alternate function characteristics (CANTX and CANRX).

**Table 155. CAN characteristics**

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $t_{p(RX:TX)}$ | CAN controller propagation time[1] | | | | 60 | ns |

1. Data based on simulation results, not tested in production

## 20.12 10-bit ADC characteristics

Subject to general operating conditions for $V_{DD}$, $f_{CPU}$, and $T_A$ unless otherwise specified.

**Table 156. 10-bit ADC characteristics**

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $f_{ADC}$ | ADC clock frequency | | 0.4 | | 2 | MHz |
| $V_{AREF}$ | Analog reference voltage | $0.7^*V_{DD} \leq V_{AREF} \leq V_{DD}$ | 3.8 | | $V_{DD}$ | V |
| $V_{AIN}$ | Conversion voltage range | | $V_{SSA}$ | | $V_{AREF}$ | V |
| $I_{lkg}$ | Positive input leakage current for analog input[1] | $-40°C \leq T_A \leq 85°C$ range | | | ±250 | nA |
| | | Other $T_A$ ranges | | | ±1 | µA |
| $R_{AIN}$ | External input impedance[2] | | | | See *Figure 112* and *Figure 113* | kΩ |
| $C_{AIN}$ | External capacitor on analog input | | | | | pF |
| $f_{AIN}$ | Variation frequency of analog input signal | | | | | Hz |
| $C_{ADC}$ | Internal sample and hold capacitor | | | 12 | | pF |
| $t_{ADC}$ | Conversion time (Sample + Hold)<br>$f_{CPU}$ = 8 MHz, speed = 0,<br>$f_{ADC}$ = 2 MHz | | 7.5 | | | µs |
| $t_{ADC}$ | No. of sample capacitor loading cycles | | 4 | | | $1/f_{ADC}$ |
| | No. of hold conversion cycles | | 11 | | | |

1. Injecting negative current on adjacent pins may result in increased leakage currents. Software filtering of the converted analog value is recommended.
2. Any added external serial resistor will downgrade the ADC accuracy (especially for resistance greater than 10kΩ). Data based on characterization results, not tested in production.

**ST72521-Auto Microcontroller FASTROM/ROM Option List**

(Last update: July 2010)

Customer: . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Address: . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Contact: . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Phone No: . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Reference: . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

The FASTROM/ROM code name is assigned by STMicroelectronics.
FASTROM/ROM code must be sent in .S19 format. .Hex extension cannot be processed.

Device Type/Memory Size/Package (check only one option):

| FASTROM DEVICE: | 60K | 32K |
|---|---|---|
| LQFP80 14x14: | [ ] ST72P521(M9)T | |
| LQFP64 14x14: | [ ] ST72P521(R9)T | [ ] ST72P521(R9)T |
| LQFP64 10x10: | [ ] ST72P521(AR9)T | [ ] ST72P521(AR9)T |

| ROM DEVICE: | 60K | 32K |
|---|---|---|
| LQFP80 14x14: | [ ] ST72521B(M9)T | |
| LQFP64 14x14: | [ ] ST72521B(R9)T | [ ] ST72521B(R9)T |
| LQFP64 10x10: | [ ] ST72521B(AR9)T | [ ] ST72521B(AR9)T |

Conditioning (check only one option):
[ ] Tape & Reel [ ] Tray

Temperature range : [ ] A (-40° to +85°C)
[ ] C (-40° to +125°C)

Special Marking: [ ] No [ ] Yes "_ _ _ _ _ _ _ _ _ _" (10 characters max)
Authorized characters are letters, digits, '.', '-', '/' and spaces only.

Clock Source Selection: [ ] Resonator:
[ ] LP: Low power resonator (1 to 2 MHz)
[ ] MP: Medium power resonator (2 to 4 MHz)
[ ] MS: Medium speed resonator (4 to 8 MHz)
[ ] HS: High speed resonator (8 to 16 MHz)
[ ] Internal RC
[ ] External Clock (sets MP Medium Power resonator in Option Byte)

PLL (1)(2) [ ] Disabled [ ] Enabled

LVD Reset [ ] Disabled [ ] High threshold [ ] Med.threshold [ ] Low threshold

Reset Delay [ ] 256 Cycles [ ] 4096 Cycles

Watchdog Selection [ ] Software Activation [ ] Hardware Activation

Halt when Watchdog on [ ] Reset [ ] No reset

Readout Protection [ ] Disabled [ ] Enabled

Date . . . . . . . . . . . . . . . . . Signature . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

Note 1: PLL must be disabled if internal RC Network is selected.
Note 2: The PLL can be enabled only if the resonator is configured to “Medium Power: 2~4 MHz”.

**CAUTION:** The Readout Protection binary value is inverted between ROM and Flash products. The option byte checksum will differ between ROM and Flash.

Please download the latest version of this option list from www.st.com.

To avoid this, a semaphore is set to ‘1’ before checking the level change. The semaphore is changed to level '0' inside the interrupt routine. When a level change is detected, the semaphore status is checked. If it is ‘1’, it means that the last interrupt has been missed. In this case, the interrupt routine is invoked with the call instruction.

There is another possible case, that is, if PxOR or PxDDR are written to with global interrupts disabled (interrupt mask bit set). In this case, the semaphore is changed to ‘1’ when the level change is detected. Detecting a missed interrupt is done after the global interrupts are enabled (interrupt mask bit reset) and by checking the status of the semaphore. If it is ‘1’, it means that the last interrupt was missed and the interrupt routine is invoked with the call instruction.

To implement the workaround, the following software sequence is to be followed for writing into the PxOR/PxDDR registers. The example is for Port PF1 with falling edge interrupt sensitivity. The software sequence is given for both cases (global interrupts disabled / global interrupts enabled):

**Case 1:** Writing to PxOR or PxDDR with global interrupts enabled:

```
LD A,#01

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 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
```