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, LINbusSCI, SPI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	48
Program Memory Size	16KB (16K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	3.8V ~ 5.5V
Data Converters	A/D 16x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	64-LQFP
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/stmicroelectronics/st72f561ar4t6

5 CENTRAL PROCESSING UNIT

5.1 INTRODUCTION

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

5.2 MAIN FEATURES

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

5.3 CPU REGISTERS

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

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.

Index Registers (X and Y)

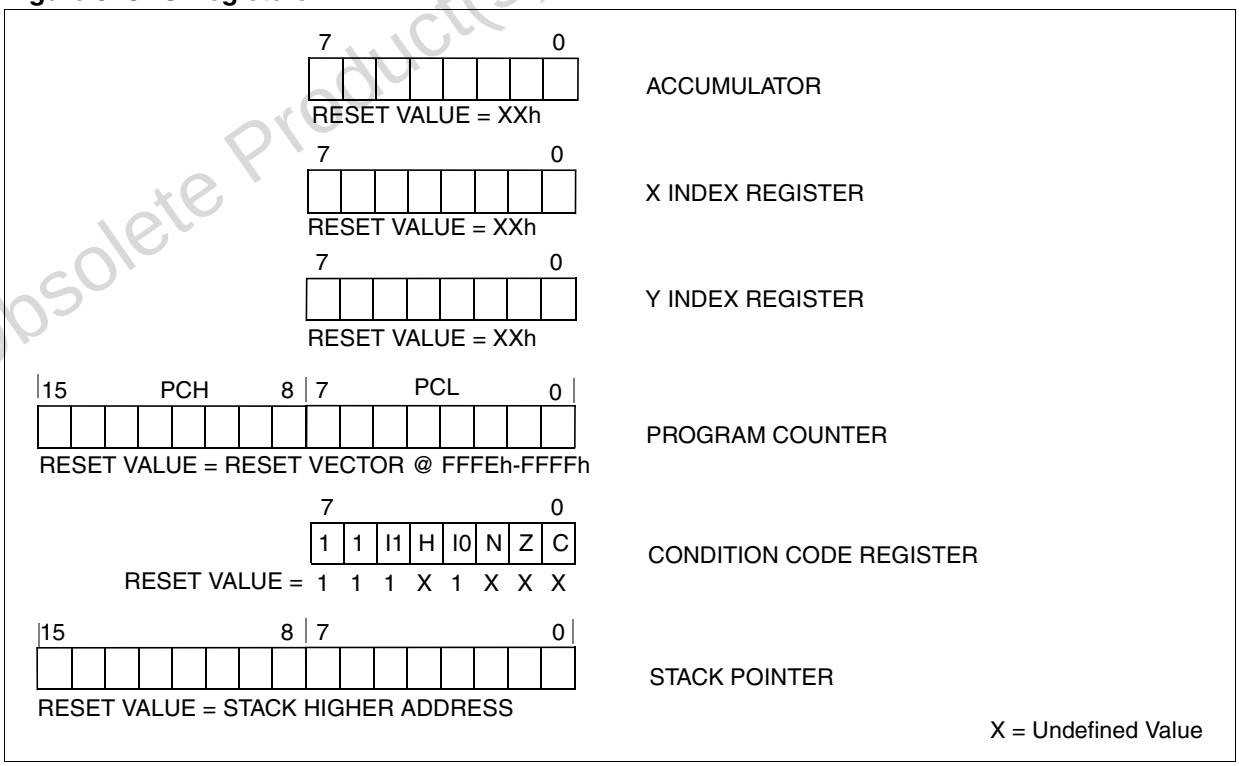
These 8-bit registers are used to create effective addresses or as temporary storage areas for data manipulation. (The Cross-Assembler generates a precede instruction (PRE) to indicate that the following instruction refers to the Y register.)

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

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



CENTRAL PROCESSING UNIT (Cont'd)

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.

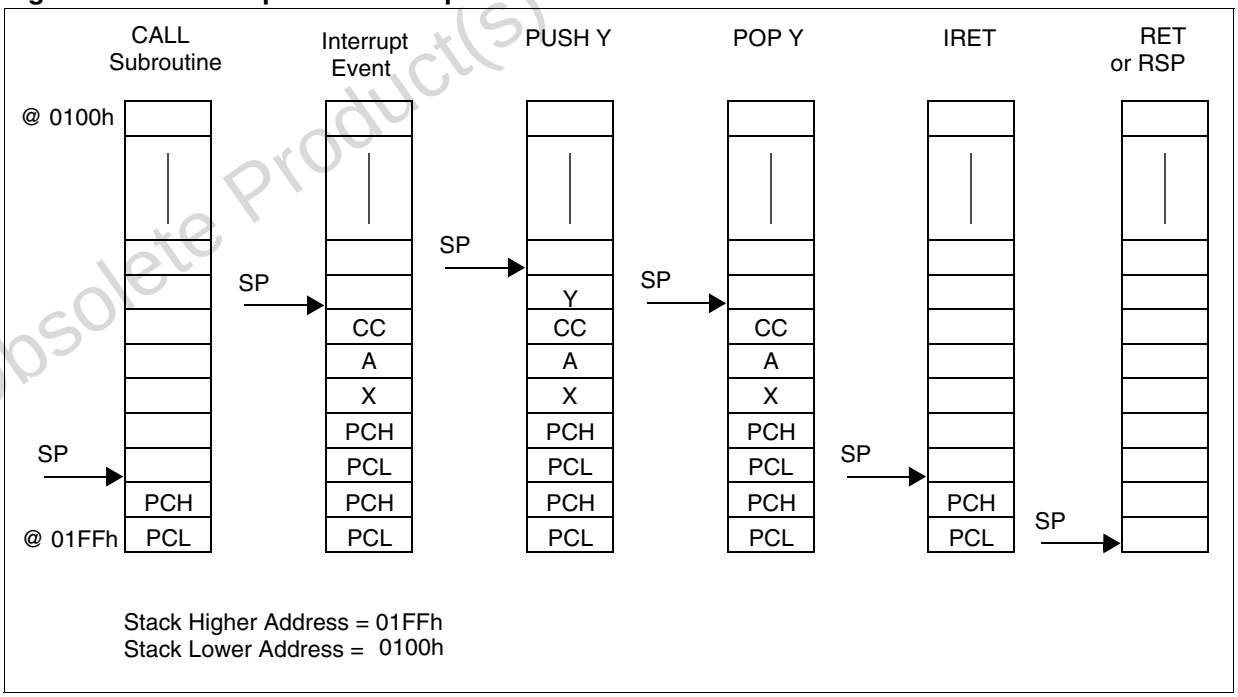
The stack is used to save the return address during a subroutine call and the CPU context during an interrupt. The user may also directly manipulate the stack by means of the PUSH and POP instructions. In the case of an interrupt, the PCL is stored at the first location pointed to by the SP. Then the 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



6.3 RESET SEQUENCE MANAGER (RSM)

6.3.1 Introduction

The reset sequence manager includes three RESET sources as shown in Figure 2:

- 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.

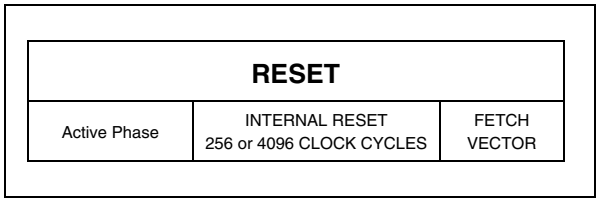
The basic RESET sequence consists of three phases as shown in Figure 1:

- 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



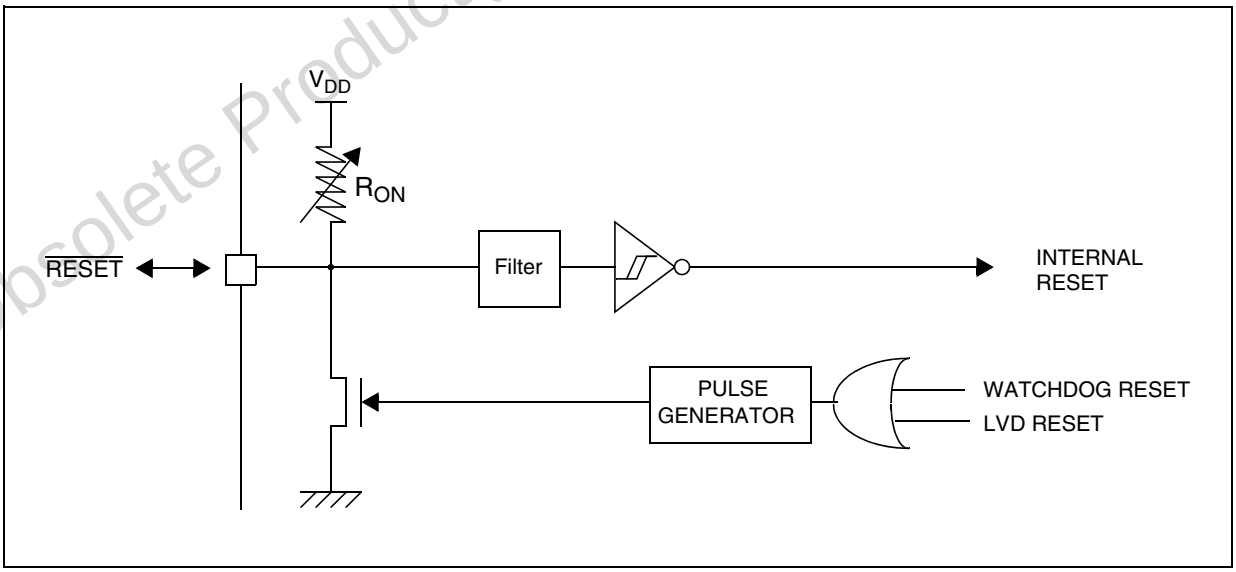
Caution: When the ST7 is unprogrammed or fully erased, the Flash is blank and the RESET vector is not programmed. For this reason, it is recommended to keep the RESET pin in low state until programming mode is entered, in order to avoid unwanted behavior.

6.3.2 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 Electrical Characteristic section for more details.

A RESET signal originating from an external source must have a duration of at least $t_{h(RSTL)in}$ in order to be recognized (see Figure 3). This detection is asynchronous and therefore the MCU can enter reset state even in HALT mode.

Figure 13. Reset Block Diagram



WINDOW WATCHDOG (Cont'd)

10.1.9 Interrupts

None.

10.1.10 Register Description

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).

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.

Figure 38. Watchdog Timer Register Map and Reset Values

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

8-BIT TIMER (Cont'd)

Figure 60. Counter Timing Diagram, Internal Clock Divided by 2

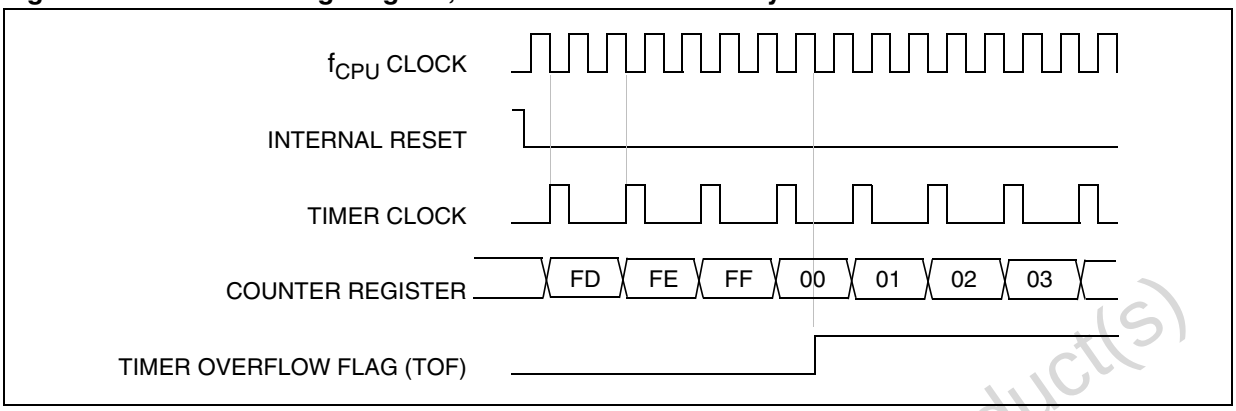


Figure 61. Counter Timing Diagram, Internal Clock Divided by 4

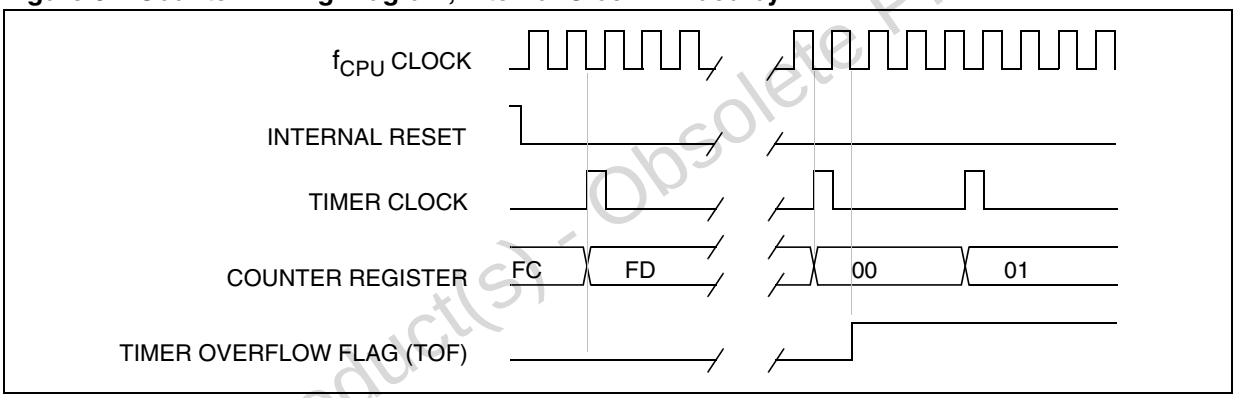
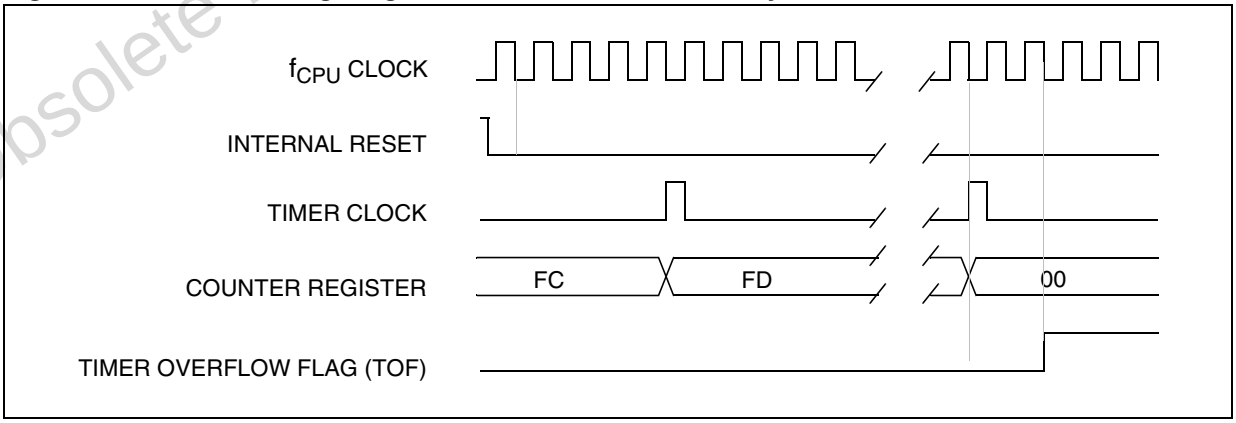


Figure 62. Counter Timing Diagram, Internal Clock Divided by 8



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

SERIAL PERIPHERAL INTERFACE (cont'd)

10.6.3.2 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 Section 10.6.5.3).

Figure 72. Generic $\overline{SS}$ Timing Diagram

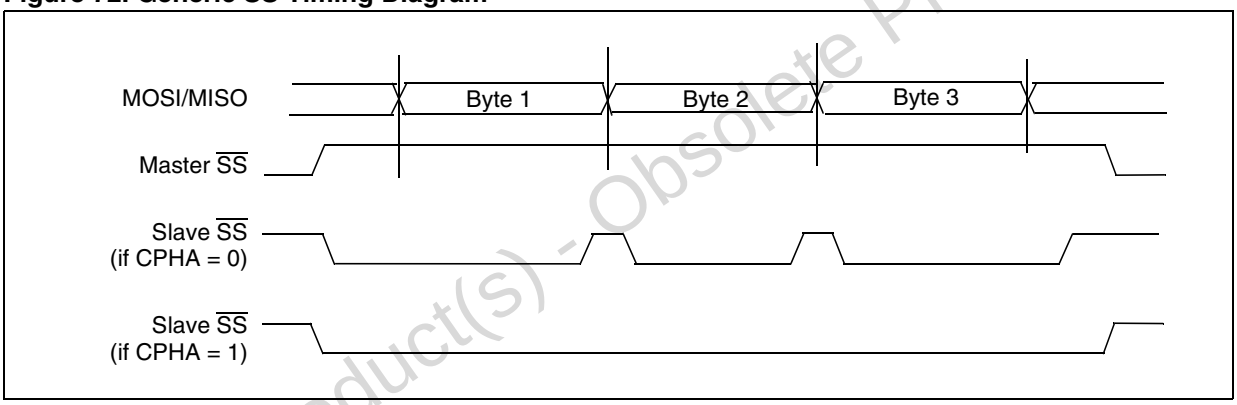
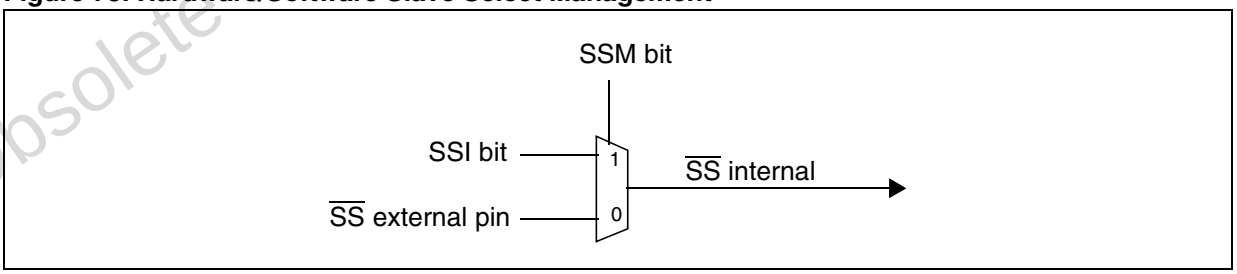


Figure 73. Hardware/Software Slave Select Management



LINSCI™ SERIAL COMMUNICATION INTERFACE (SCI Mode) (cont'd)

10.7.5 SCI Mode - Functional Description

Conventional Baud Rate Generator Mode

The block diagram of the Serial Control Interface in conventional baud rate generator mode is shown in Figure 1.

It uses four registers:

– 2 control registers (SCICR1 and SCICR2)

– A status register (SCISR)

– A baud rate register (SCIBRR)

Extended Prescaler Mode

Two additional prescalers are available in extended prescaler mode. They are shown in Figure 3.

– An extended prescaler receiver register (SCIERPR)

– An extended prescaler transmitter register (SCIETPR)

10.7.5.1 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 2).

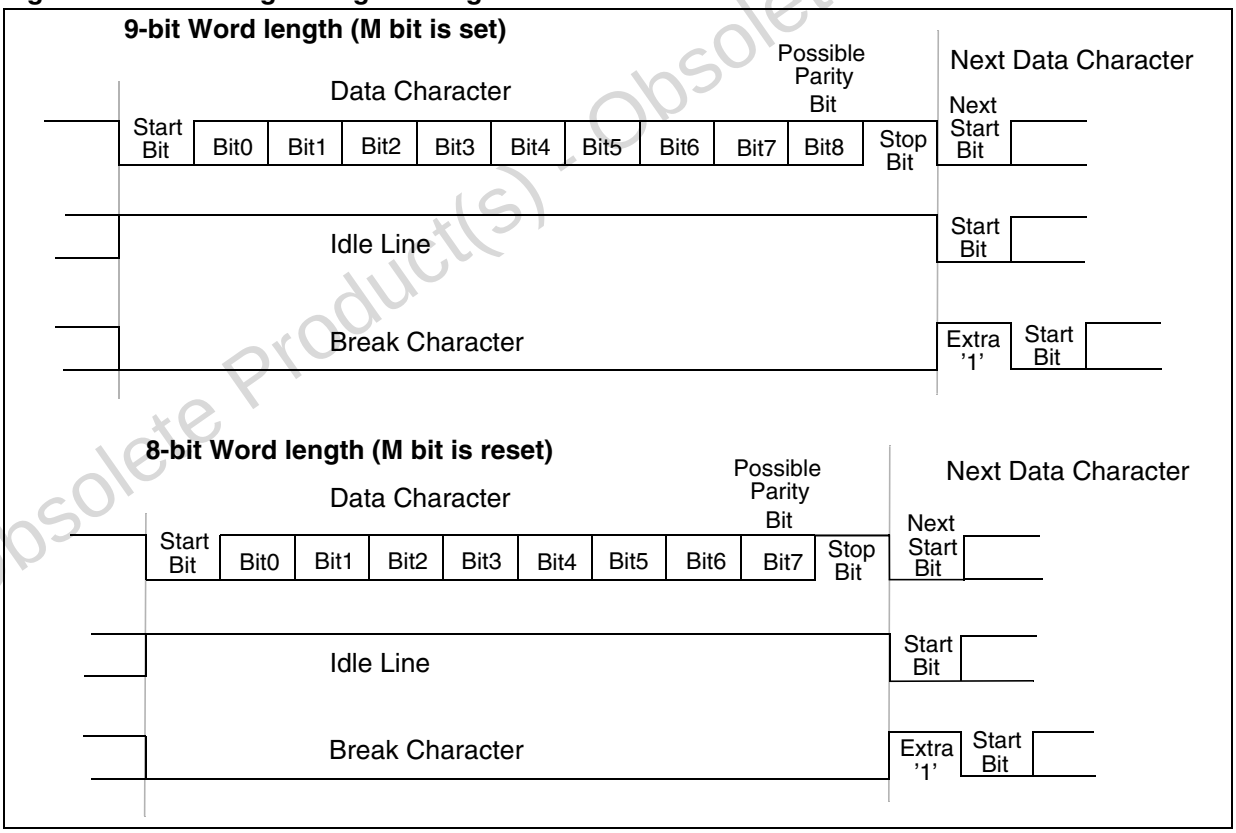
The TDO pin is in low state during the start bit.

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.

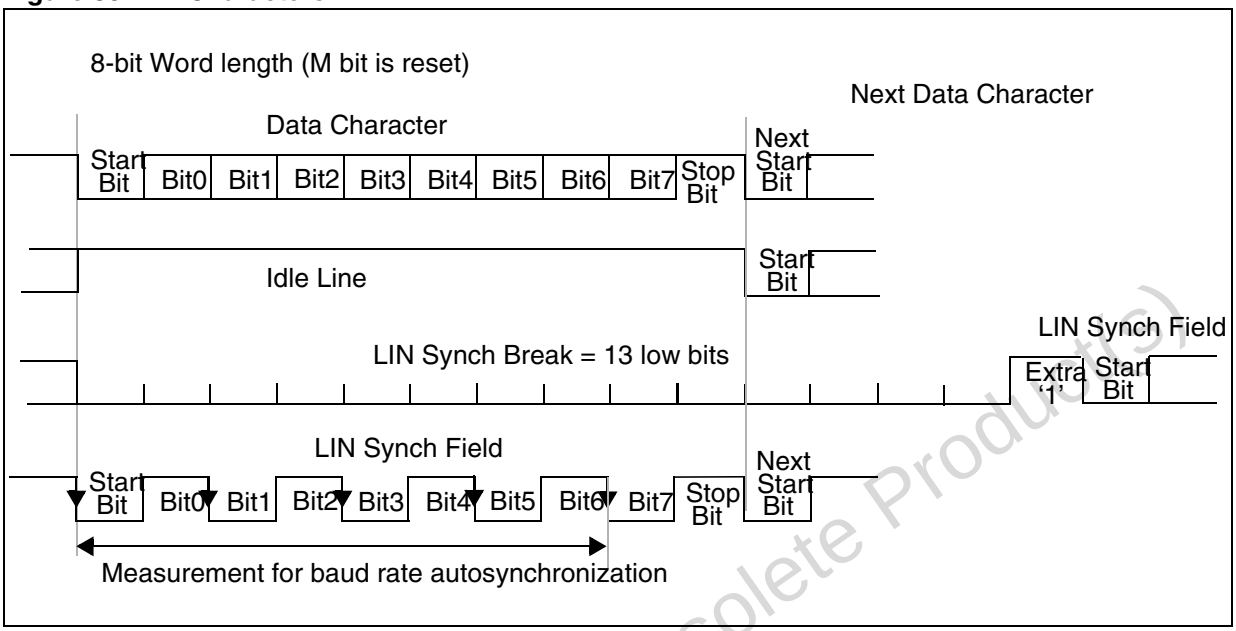
A Break character is a character with a sufficient number of low level bits to break the normal data format followed by an extra "1" bit to acknowledge the start bit.

Figure 78. Word Length Programming



LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Mode) (cont'd)

Figure 80. LIN Characters



LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Master Only) (Cont'd)

10.8.4.2 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.

LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Master Only) (Cont'd)

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

Notes:

– 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

Notes:

– 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 wake-up 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.

beCAN CONTROLLER (Cont'd)

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.

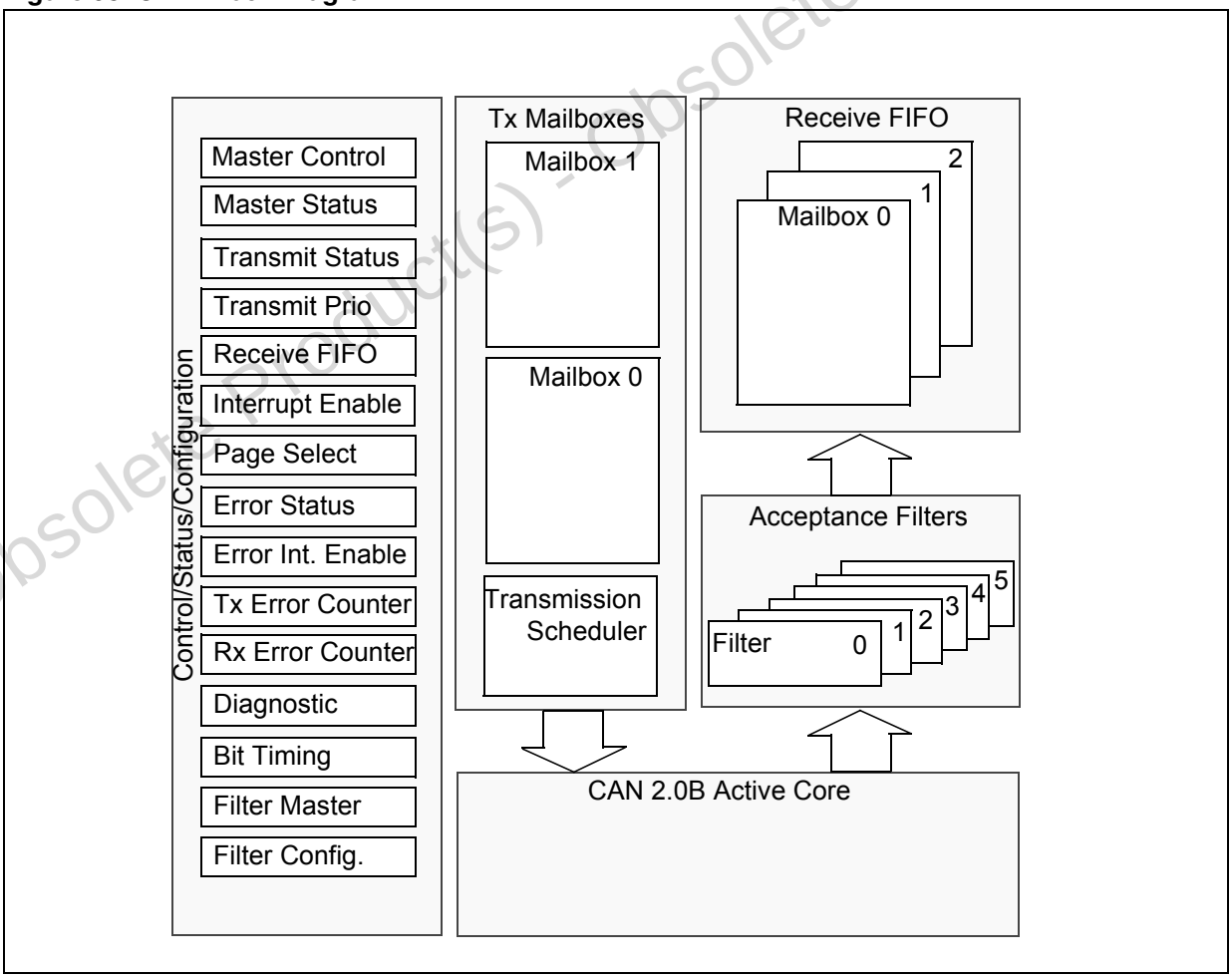
Acceptance Filters

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



beCAN CONTROLLER (Cont'd)

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
```

beCAN CONTROLLER (Cont'd)

Side-effect of Workround 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 1).

Table 29. While Loop Timing

f_{CPU}	Software timing: While loop	Minimum baud rate for possible missed dominant bit
8 MHz	1.25 µs	800 Kbaud
4 MHz	2.5 µs	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.

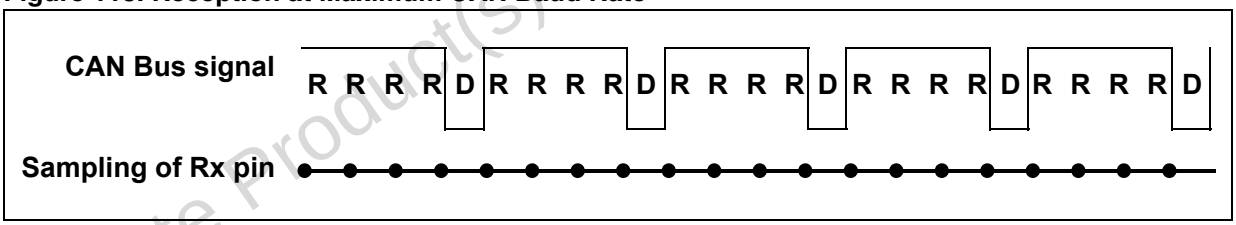
The example in Figure 20 shows reception at maximum CAN baud rate: In this case t_{CAN} is $8/f_{CPU}$ and the sampling time is $10/f_{CPU}$.

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 21) 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



beCAN CONTROLLER (Cont'd)

TRANSMIT ERROR COUNTER REG. (TECR)

Read Only

Reset Value: 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 REG. (RECR)

Page: 00h — Read Only

Reset Value: 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)

All bits of this register are set and clear by software.
Read / Write
Reset Value: 0000 1100 (0Ch)

7							0
0	0	0	0	RX	SAMP	SILM	LBKM

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)

This register can only be accessed by the software when the CAN hardware is in configuration mode.
Read / Write
Reset Value: 0000 0000 (00h)

7							0
SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0

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.
$tq = (BRP+1)/f_{CPU}$

For more information on bit timing, please refer to Section 0.1.4.6 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)

12.4 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).

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 f_{OSC} = 4 MHz, f_{CPU} = 2 MHz f_{OSC} = 8 MHz, f_{CPU} = 4 MHz f_{OSC} = 16 MHz, f_{CPU} = 8 MHz		1.8 3.2 6 10	3 5 8 15	1.1 2.2 4.4 8.9	2 3.5 6 12	mA
	Supply current in SLOW mode [3)]	f_{OSC} = 2 MHz, f_{CPU} = 62.5kHz f_{OSC} = 4 MHz, f_{CPU} = 125 kHz f_{OSC} = 8 MHz, f_{CPU} = 250 kHz f_{OSC} = 16 MHz, f_{CPU} = 500 kHz		0.5 0.6 0.85 1.25	2.7 3 3.6 4	0.1 0.2 0.4 0.8	0.2 0.4 0.8 1.5	
	Supply current in WAIT mode [3)]	f_{OSC} = 2 MHz, f_{CPU} = 1 MHz f_{OSC} = 4 MHz, f_{CPU} = 2 MHz f_{OSC} = 8 MHz, f_{CPU} = 4 MHz f_{OSC} = 16 MHz, f_{CPU} = 8 MHz		1 1.8 3.4 6.4	3 4 5 7	0.7 1.4 2.9 5.7	3 4 5 7	
	Supply current in SLOW WAIT mode [2)]	f_{OSC} = 2 MHz, f_{CPU} = 62.5 kHz f_{OSC} = 4 MHz, f_{CPU} = 125 kHz f_{OSC} = 8 MHz, f_{CPU} = 250 kHz f_{OSC} = 16 MHz, f_{CPU} = 500 kHz		0.4 0.5 0.6 0.8	1.2 1.3 1.8 2	0.07 0.14 0.28 0.56	0.12 0.25 0.5 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	

Notes:

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 12.5.3) and the peripheral power consumption (Section 12.4.2).
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.

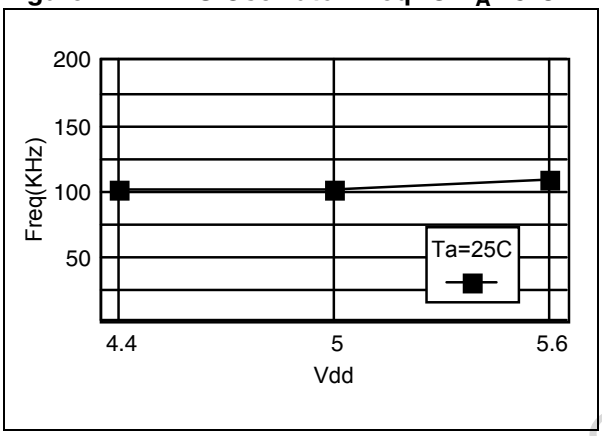
CLOCK CHARACTERISTICS (Cont'd)

12.6 Auto Wakeup from Halt Oscillator (AWU)

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{AWU}	AWU oscillator frequency[1)]		50	100	250	kHz
t_{RCSRT}	AWU oscillator startup time			10		µs

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

Figure 124. AWU Oscillator Freq. @ T_A 25°C



12.10 CONTROL PIN CHARACTERISTICS

12.10.1 Asynchronous $\overline{\text{RESET}}$ Pin

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

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
V_{IL}	Input low level voltage[1)]					0.3 x V_{DD}	V
V_{IH}	Input high level voltage[1)]			0.7 x 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} = +5mA		0.68	0.95	
			I_{IO} = +2mA		0.28	0.45	
R_{ON}	Weak pull-up equivalent resistor[4)]	$V_{IN} = V_{SS}$		20	40	80	kΩ
$t_{w(RSTL)out}$	Generated reset pulse duration	Internal reset source			30		µs
$t_{h(RSTL)in}$	External reset pulse hold time[5)]			2.5			
$t_{g(RSTL)in}$	Filtered glitch duration[6)]				200		ns

Notes:

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 12.2.2 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 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.

12.11 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...).

12.11.1 8-Bit PWM-ART Autoreload Timer

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$t_{res(PWM)}$	PWM resolution time		1			t_{CPU}
		f_{CPU} = 8 MHz	125			ns
f_{EXT}	ART external clock frequency		0		$f_{CPU}/2$	MHz
f_{PWM}	PWM repetition rate					
Res_{PWM}	PWM resolution				8	bit
V_{OS}	PWM/DAC output step voltage	V_{DD} = 5V, Res = 8-bits		20		mV
$t_{COUNTER}$	Timer clock period when internal clock is selected	f_{CPU} = 8 MHz	1		128	t_{CPU}
			0.125		16	µs

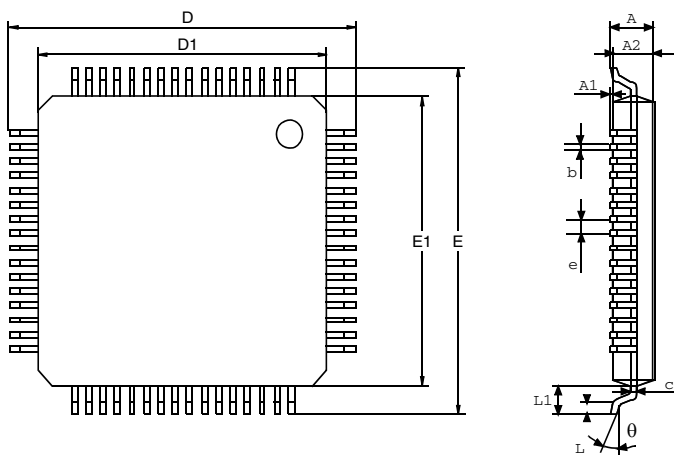
12.11.2 8-Bit Timer

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 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 MHz	2		8000	t_{CPU}
			0.250		1000	µs

12.11.3 16-Bit Timer

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 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
$t_{COUNTER}$	Timer clock period when internal clock is selected	f_{CPU} = 8 MHz	2		8	t_{CPU}
			0.250		1	µs

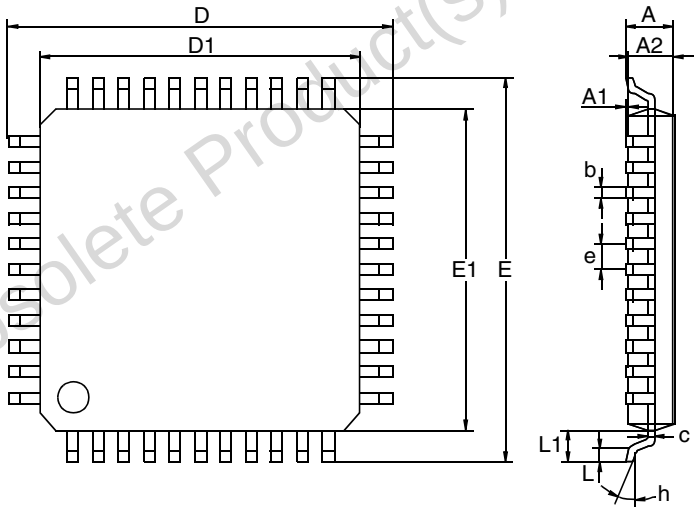
Figure 147. 64-Pin Low Profile Quad Flat Package (10 x10)



Dim.	mm			inches[1]		
	Min	Typ	Max	Min	Typ	Max
A			1.60			0.0630
A1	0.05		0.15	0.0020		0.0059
A2	1.35	1.40	1.45	0.0531	0.0551	0.0571
b	0.17	0.22	0.27	0.0067	0.0087	0.0106
c	0.09		0.20	0.0035		0.0079
D		12.00			0.4724	
D1		10.00			0.3937	
E		12.00			0.4724	
E1		10.00			0.3937	
e		0.50			0.0197	
θ	0°	3.5°	7°	0°	3.5°	7°
L	0.45	0.60	0.75	0.0177	0.0236	0.0295
L1		1.00			0.0394	
	Number of Pins					
N	64					

Note 1. Values in inches are converted from mm and rounded to 4 decimal digits.

Figure 148. 44-Pin Low Profile Quad Flat Package (10x10)



Dim.	mm			inches[1]		
	Min	Typ	Max	Min	Typ	Max
A			1.60			0.0630
A1	0.05		0.15	0.0020		0.0059
A2	1.35	1.40	1.45	0.0531	0.0551	0.0571
b	0.30	0.37	0.45	0.0118	0.0146	0.0177
C	0.09		0.20	0.0035		0.0079
D		12.00			0.4724	
D1		10.00			0.3937	
E		12.00			0.4724	
E1		10.00			0.3937	
e		0.80			0.0315	
θ	0°	3.5°	7°	0°	3.5°	7°
L	0.45	0.60	0.75	0.0177	0.0236	0.0295
L1		1.00			0.0394	
	Number of Pins					
N	44					

Note 1. Values in inches are converted from mm and rounded to 4 decimal digits.

IMPORTANT NOTES (Cont'd)

Occurrence

The occurrence of the problem is random and proportional to the baud rate. With a transmit frequency of 19200 baud (f_{CPU} = 8 MHz and SCIBRR = 0xC9), the wrong break duration occurrence is around 1%.

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) x 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, 208µs 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.

16.2.2 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.

16.3 ROM DEVICES ONLY

16.3.1 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 Section 10.5.3.5 on page 103, 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.