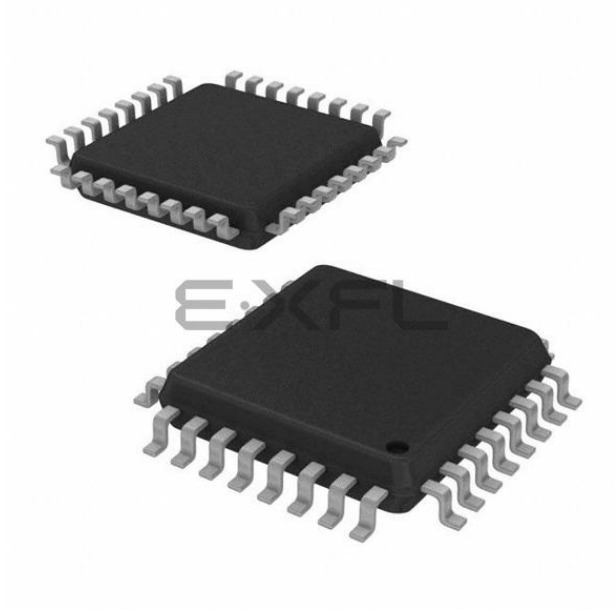


Welcome to E-XFL.COM

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	24
Program Memory Size	32KB (32K 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 6x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	32-LQFP
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/stmicroelectronics/st72f561k6t6

SYSTEM INTEGRITY MANAGEMENT (Cont'd)

6.4.4 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 Section 6.4.2.1 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.

RESET Sources	LVDRF	WDGRF
External $\overline{\text{RESET}}$ pin	0	0
Watchdog	0	1
LVD	1	X

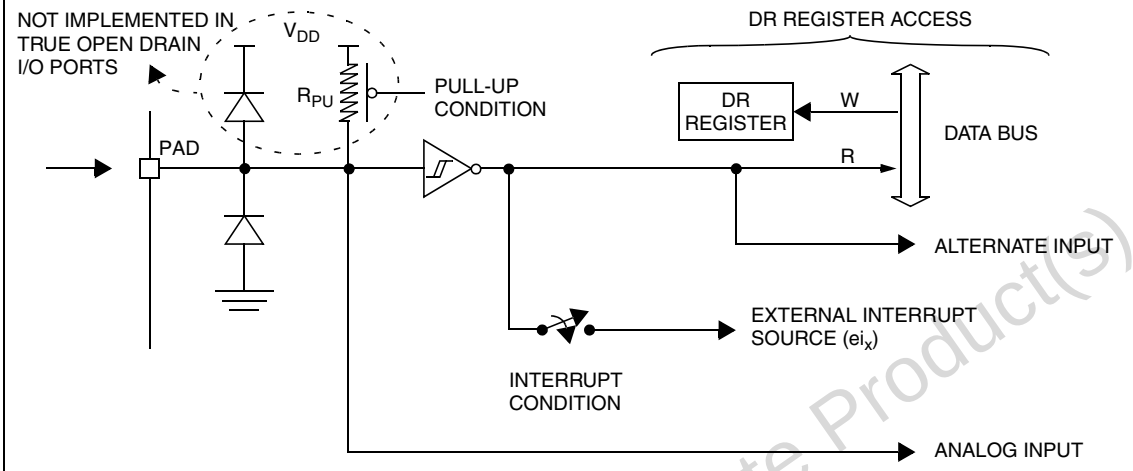
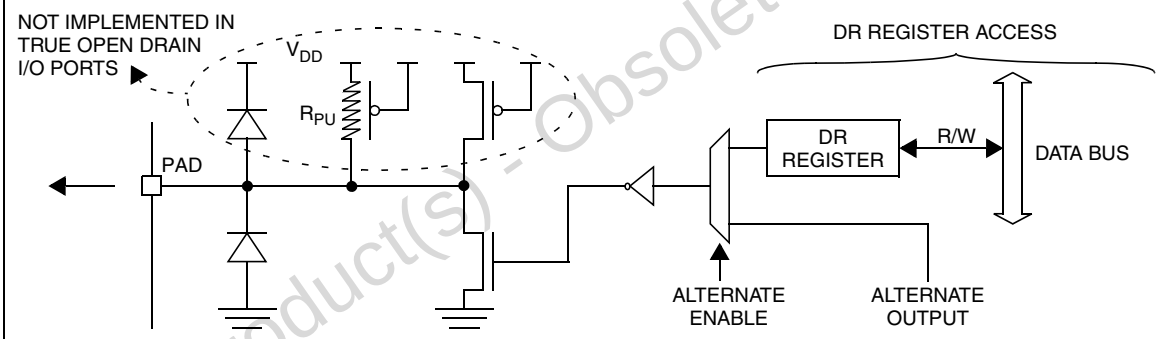
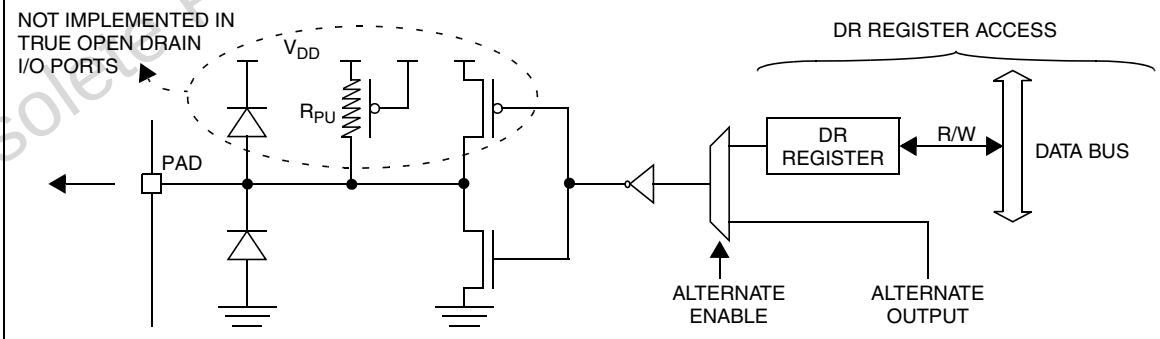
Application notes

The LVDRF flag is not cleared when another RESET type occurs (external or watchdog), the LVDRF flag remains set to keep trace of the original failure.
In this case, 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 can not be used in the application.

I/O PORTS (Cont'd)

Table 13. I/O Port Configurations

	Hardware Configuration
INPUT [1]	
OPEN-DRAIN OUTPUT [2]	
PUSH-PULL OUTPUT [2]	

Notes:

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.

WINDOW WATCHDOG (Cont'd)

Figure 36. Exact Timeout Duration (t_{min} and t_{max})

WHERE:

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

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

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

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

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

TB1 Bit (MCCSR Reg.)	TB0 Bit (MCCSR Reg.)	Selected MCCSR Timebase	MSB	LSB
0	0	2ms	4	59
0	1	4ms	8	53
1	0	10ms	20	35
1	1	25ms	49	54

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

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

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

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

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

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

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

Example:

With 2ms timeout selected in MCCSR register

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

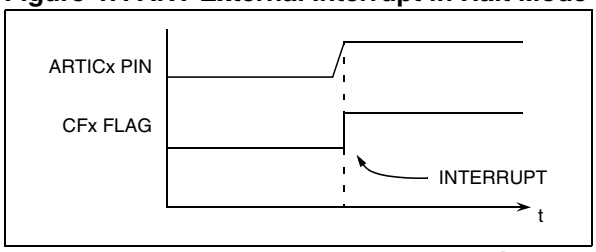
External Interrupt Capability

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 47).

Figure 47. ART External Interrupt in Halt Mode



8-BIT TIMER (Cont'd)

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.

Notes: 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).

8-BIT TIMER (Cont'd)

10.5.3.4 One Pulse Mode

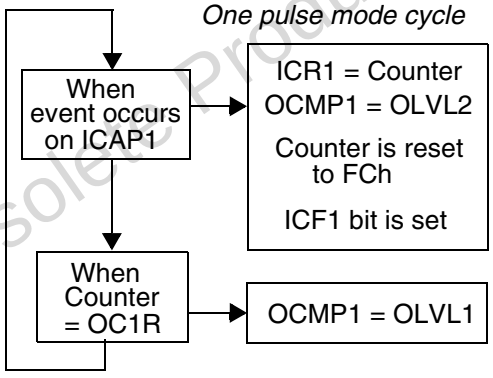
One Pulse mode enables the generation of a pulse when an external event occurs. This mode is selected via the OPM bit in the CR2 register.

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 19 Clock Control Bits).



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 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:

$$OCiR\ Value = \frac{t * f_{CPU}}{PRESC} - 5$$

Where:

t = 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 the CC[1:0] bits, see Table 19 Clock Control Bits)

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).

Notes:

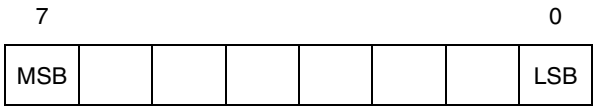
1. The OCF1 bit cannot be set by hardware in one pulse mode but the OCF2 bit can generate an Output Compare interrupt.
2. When the Pulse Width Modulation (PWM) and One Pulse Mode (OPM) bits are both set, the PWM mode is the only active one.
3. If OLVL1=OLVL2 a continuous signal will be seen on the OCMP1 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.

8-BIT TIMER (Cont'd)

INPUT CAPTURE 1 REGISTER (IC1R)

Read Only
Reset Value: Undefined

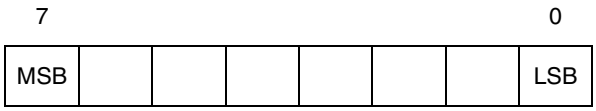
This is an 8-bit read only register that contains the counter value (transferred by the input capture 1 event).



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.



OUTPUT COMPARE 2 REGISTER (OC2R)

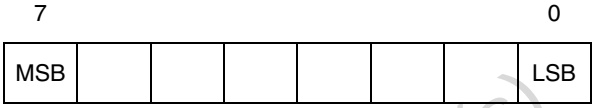
Read/Write
Reset Value: 0000 0000 (00h)

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

COUNTER REGISTER (CTR)

Read Only
Reset Value: 1111 1100 (FCh)

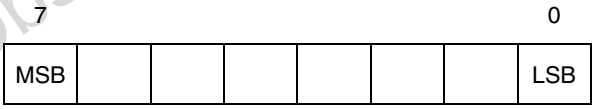
This is an 8-bit register that contains the counter value. A write to this register resets the counter. An access to this register after accessing the CSR register clears the TOF bit.



ALTERNATE COUNTER REGISTER (ACTR)

Read Only
Reset Value: 1111 1100 (FCh)

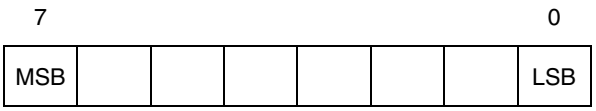
This is an 8-bit register that contains the counter value. A write to this register resets the counter. An access to this register after an access to CSR register does not clear the TOF bit in the CSR register.



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).



SERIAL PERIPHERAL INTERFACE (cont'd)

10.6.6 Low Power Modes

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.

10.6.6.1 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 Section 10.6.3.2), make sure the master drives a low level on the $\overline{SS}$ pin when the slave enters HALT mode.

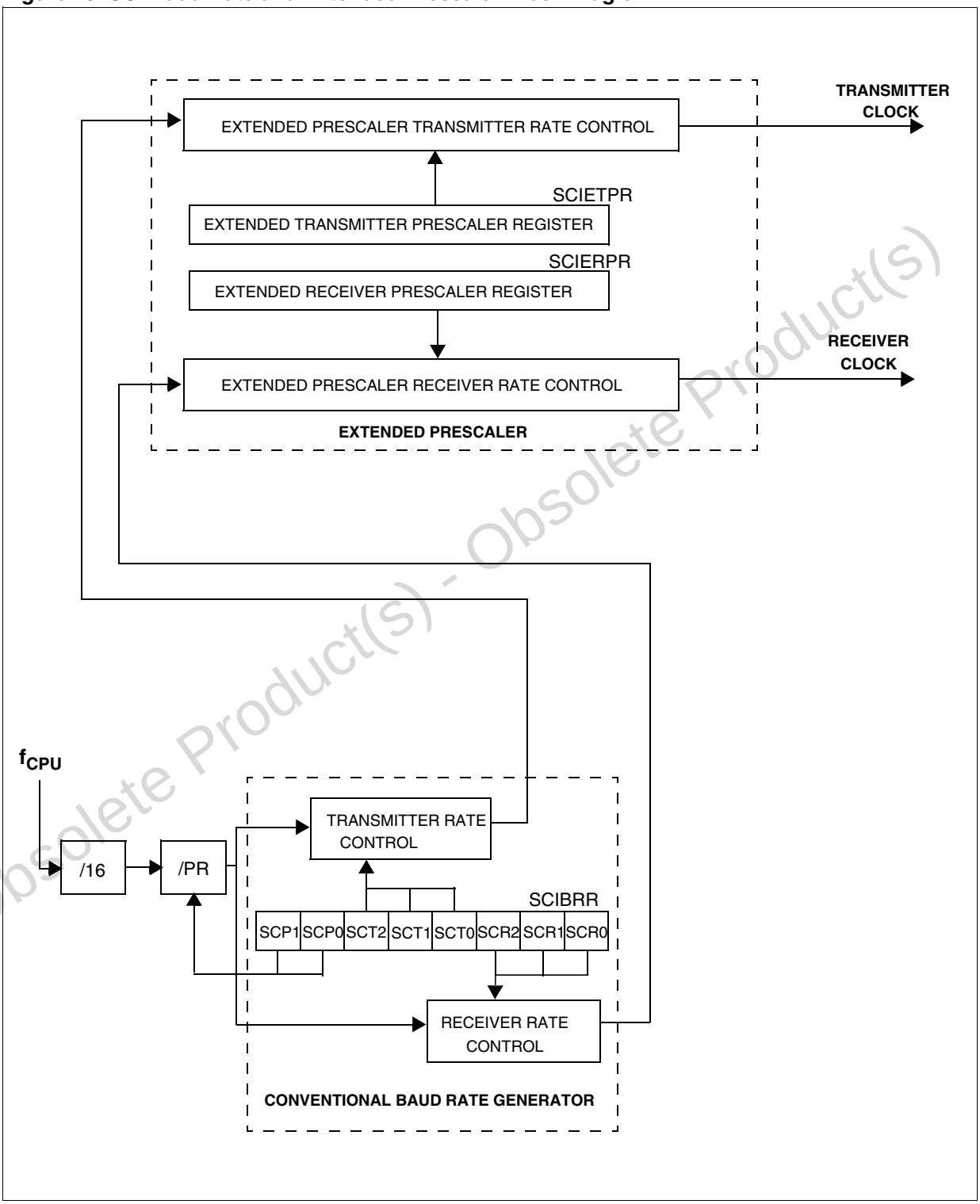
10.6.7 Interrupts

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).

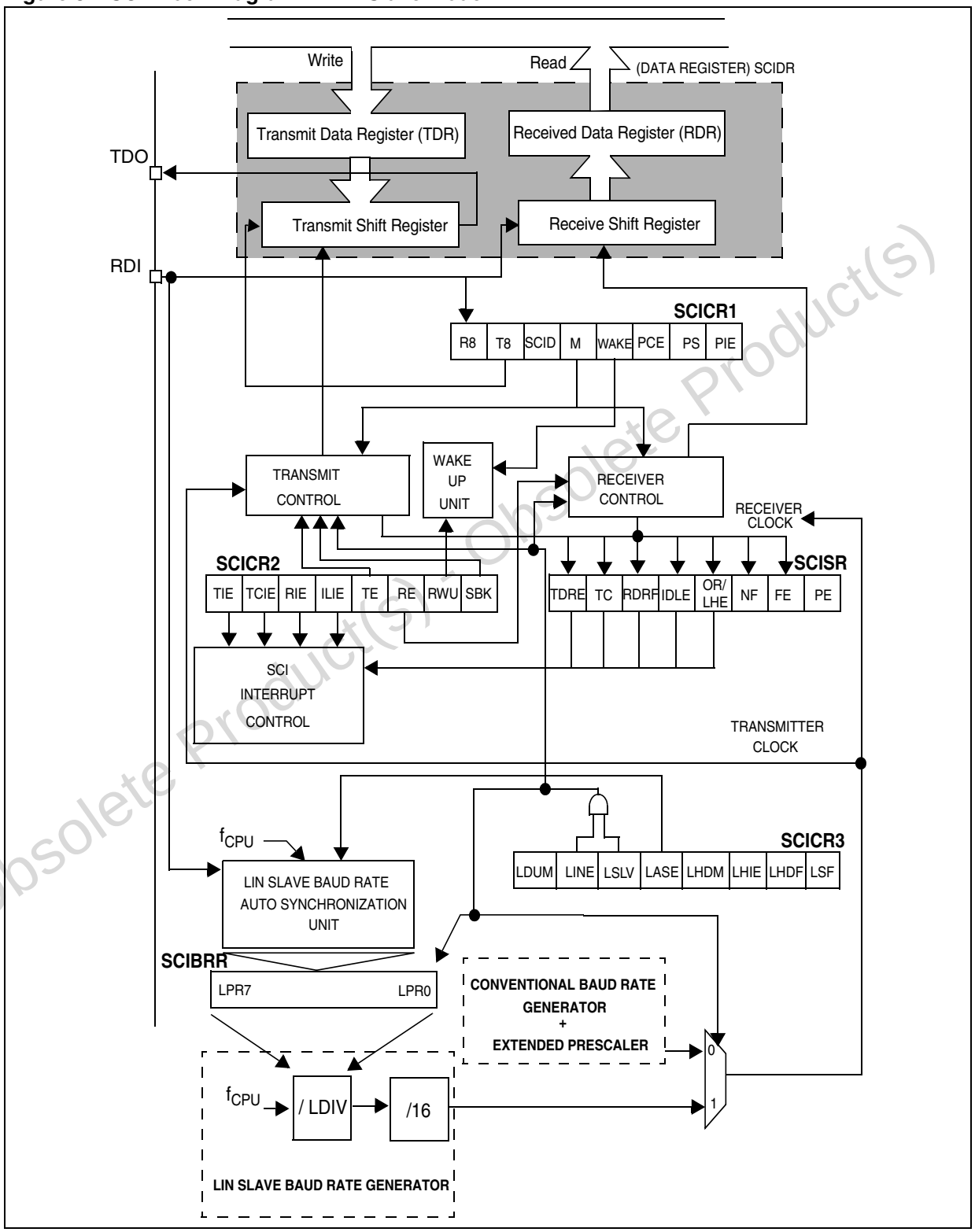
LINSCI™ SERIAL COMMUNICATION INTERFACE (SCI Mode) (cont'd)

Figure 79. SCI Baud Rate and Extended Prescaler Block Diagram



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

Figure 81. SCI Block Diagram in LIN Slave Mode



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

10.7.10 LIN Mode Register Description

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 0.1.8 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 0.1.8 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 0.1.8 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 0.1.8 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

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 0.1.8 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 0.1.8 SCI Mode Register Description.

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

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 0.1.8 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

Notes:

– Mute mode is recommended for detecting only the Header and avoiding the reception of any other characters. For more details, please refer to Section 0.1.9.3 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.

CONTROL REGISTER 3 (SCICR3)
Read/Write
Reset Value: 0000 0000 (00h)

7							0
LDUM	LINE	LSLV	LASE	LHDM	LHIE	LHDF	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

Notes:

- 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:

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

LINSCI™ SERIAL COMMUNICATIONS INTERFACE (LIN Master) (Cont'd)

Table 28. LINSCI2 Register Map and Reset Values

Address (Hex.)	Register Name	7	6	5	4	3	2	1	0
60	**SCI2SR**	TDRE	TC	RDRF	IDLE	OR	NF	FE	PE
	Reset Value	1	1	0	0	0	0	0	0
61	**SCI2DR**	DR7	DR6	DR5	DR4	DR3	DR2	DR1	DR0
	Reset Value	-	-	-	-	-	-	-	-
62	**SCI2BRR**	SCP1	SCP0	SCT2	SCT1	SCT0	SCR2	SCR1	SCR0
	Reset Value	0	0	0	0	0	0	0	0
63	**SCI2CR1**	R8	T8	SCID	M	WAKE	PCE	PS	PIE
	Reset Value	-	-	-	-	-			
64	**SCI2CR2**	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
	Reset Value	0	0	0	0	0	0	0	0
65	**SCI2CR3**		LINE	-	-	CLKEN	CPOL	CPHA	LBCL
	Reset Value	0	0	0	0	0	0	0	0
66	**SCI2ERPR**	ERPR7	ERPR6	ERPR5	ERPR4	ERPR3	ERPR2	ERPR1	ERPR0
	Reset Value	0	0	0	0	0	0	0	0
67	**SCI2ETPR**	ETPR7	ETPR6	ETPR5	ETPR4	ETPR3	ETPR2	ETPR1	ETPR0
	Reset Value	0	0	0	0	0	0	0	0

beCAN CONTROLLER (Cont'd)

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 Section 0.1.4.4 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.

10.9.4.3 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 9. Filter Bank Scale Configuration - Register Organisation.

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.

beCAN CONTROLLER (Cont'd)

10.9.4.6 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 behaviour 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

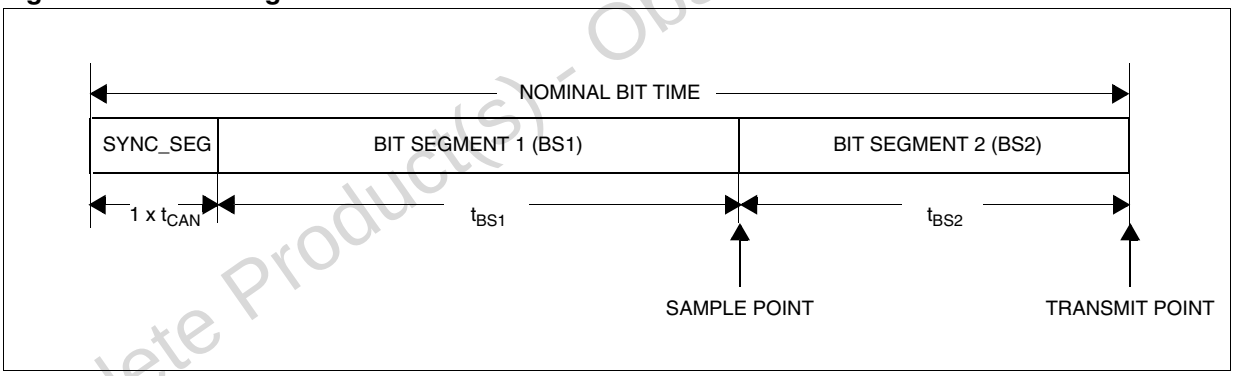
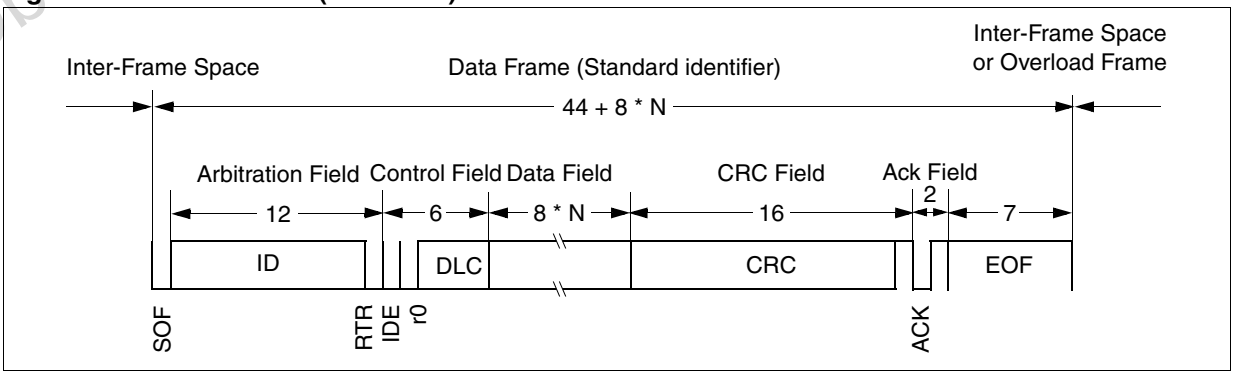


Figure 106. CAN Frames (Part 1of 2)

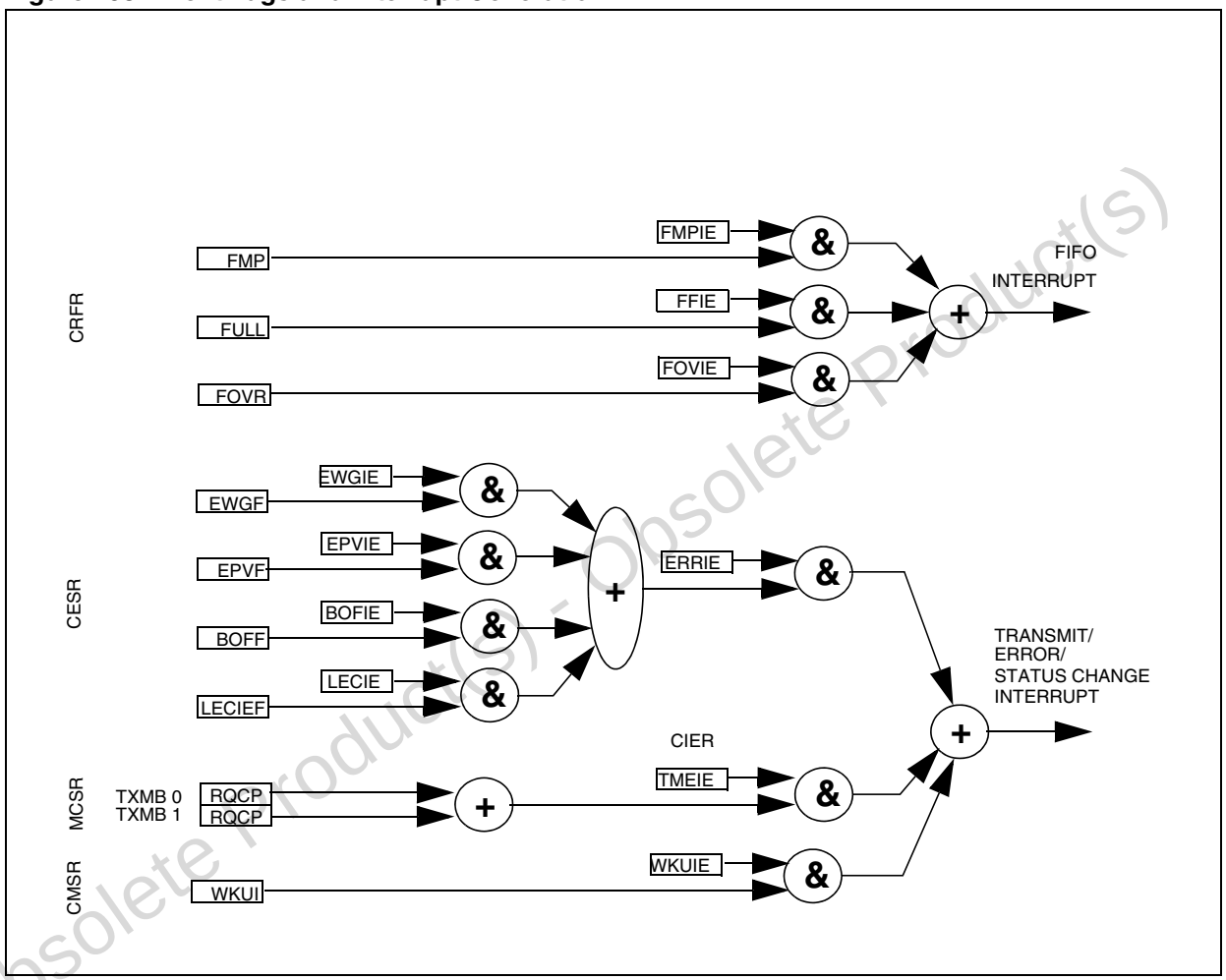


beCAN CONTROLLER (Cont'd)

10.9.5 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



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
```

10.10 10-BIT A/D CONVERTER (ADC)

10.10.1 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.

10.10.2 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 116.

10.10.3 Functional Description

10.10.3.1 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.

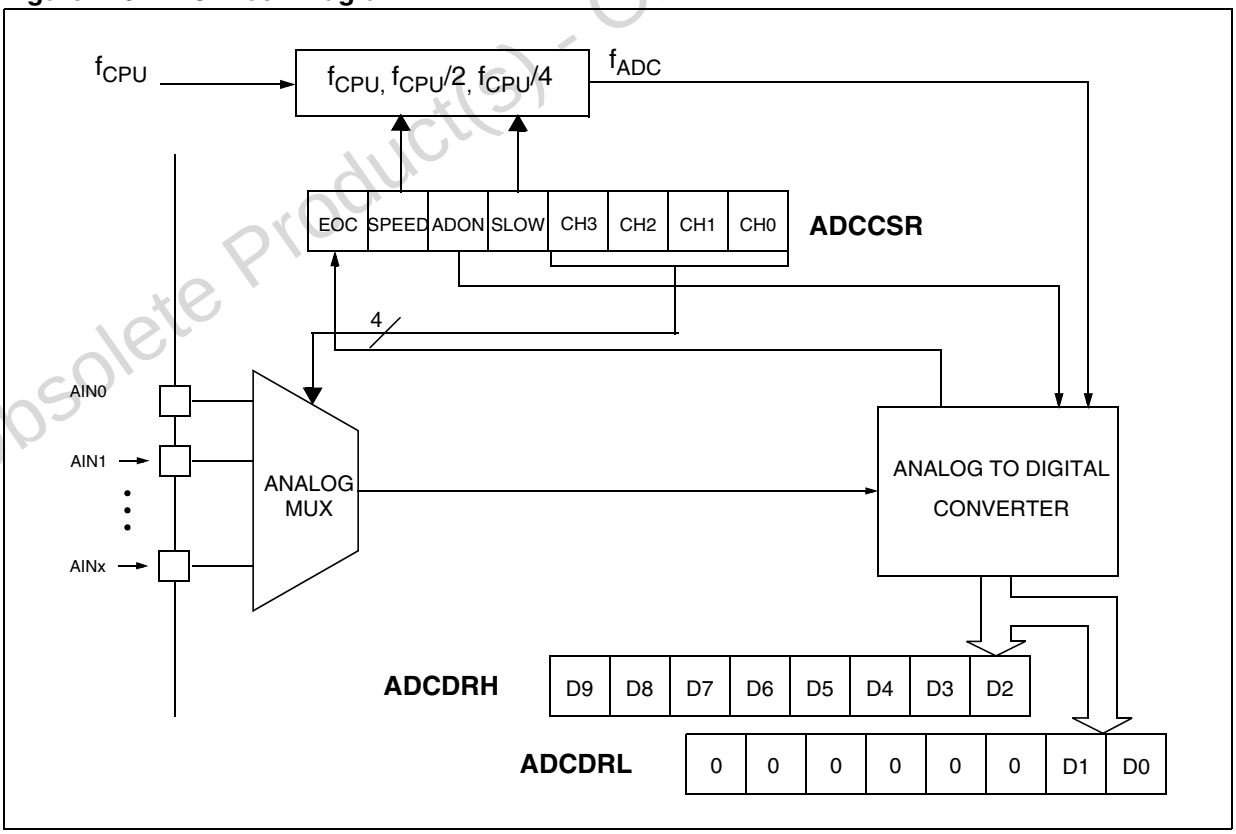
If the input voltage (V_{AIN}) is greater than V_{DDA} (high-level voltage reference) then the conversion result is FFh in the ADCDRH register and 03h in the ADCDRL register (without overflow indication).

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 116. ADC Block Diagram



12.3.2 Operating Conditions with Low Voltage Detector (LVD)

Subject to general operating conditions for T_A.

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$V_{IT+(LVD)}$	Reset release threshold (V_{DD} rise)		4.0[1)]	4.2	4.5	V
$V_{IT-(LVD)}$	Reset generation threshold (V_{DD} fall)		3.8	4.0	4.25[1)]	
$V_{hys(LVD)}$	LVD voltage threshold hysteresis[1)]	$V_{IT+(LVD)}$-$V_{IT-(LVD)}$	150	200	250	mV
Vt_{POR}	V_{DD} rise time rate[1)]		6			µs/V
					100	ms/V
$t_{g(VDD)}$	V_{DD} glitches filtered (not detected) by LVD[1)]	Measured at $V_{IT-(LVD)}$			40	ns

Notes:

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

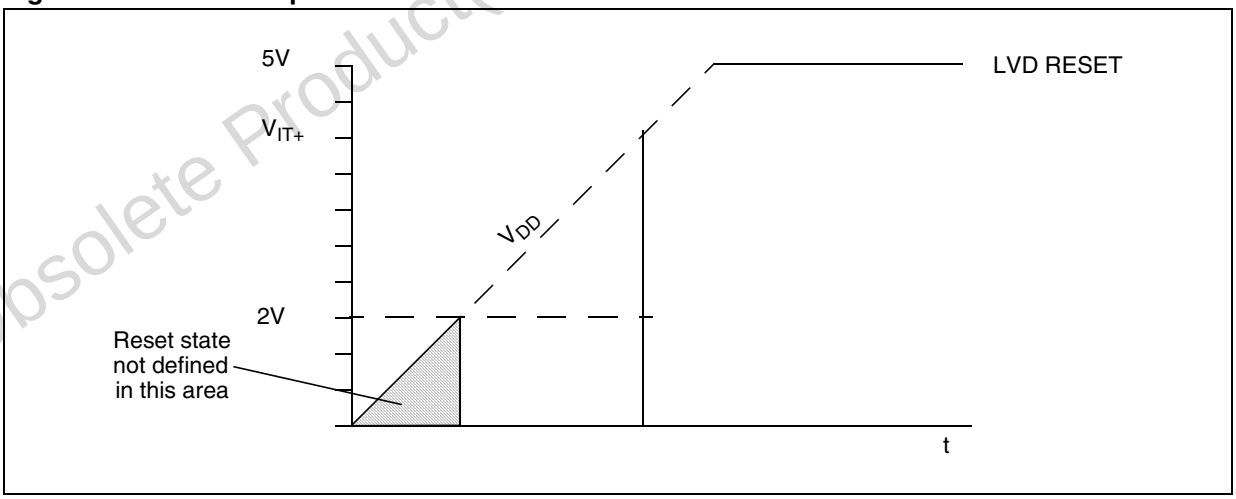
12.3.3 Auxiliary Voltage Detector (AVD) Thresholds

Subject to general operating conditions for T_A.

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$V_{IT+(AVD)}$	1⇒0 AVDF flag toggle threshold (V_{DD} rise)		4.4[1)]	4.6	4.9	V
$V_{IT-(AVD)}$	0⇒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 120. 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.

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