# Atmel - AT89C51CC01CA-RLTUM Datasheet

**Welcome to E-XFL.COM**

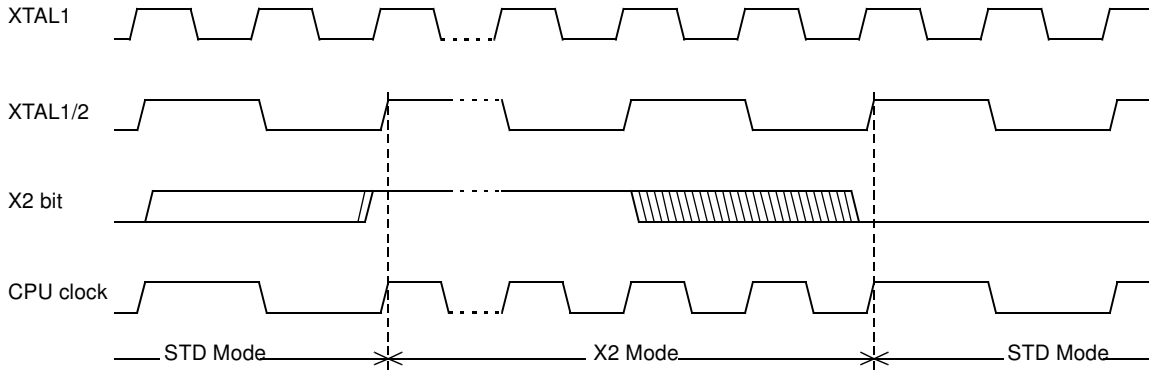
## What is "Embedded - Microcontrollers"?

"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"

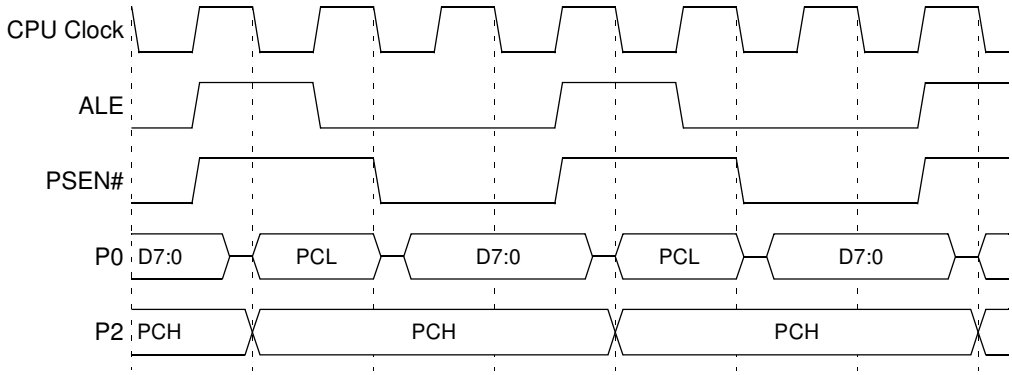
| Details | |
|---|---|
| Product Status | Active |
| Core Processor | 80C51 |
| Core Size | 8-Bit |
| Speed | 40MHz |
| Connectivity | CANbus, UART/USART |
| Peripherals | POR, PWM, WDT |
| Number of I/O | 34 |
| Program Memory Size | 32KB (32K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | 2K x 8 |
| RAM Size | 1.25K x 8 |
| Voltage - Supply (Vcc/Vdd) | 3V ~ 5.5V |
| Data Converters | A/D 8x10b |
| Oscillator Type | External |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 44-LQFP |
| Supplier Device Package | 44-VQFP (10x10) |
| Purchase URL | https://www.e-xfl.com/product-detail/atmel/at89c51cc01ca-rltum |

**Figure 6.** Mode Switching Waveforms




Note: In order to prevent any incorrect operation while operating in the X2 mode, users must be aware that all peripherals using the clock frequency as a time reference (UART, timers...) will have their time reference divided by two. For example a free running timer generating an interrupt every 20 ms will then generate an interrupt every 10 ms. A UART with a 4800 baud rate will have a 9600 baud rate.

**Figure 19.** External Code Fetch Waveforms




## Flash Memory Architecture

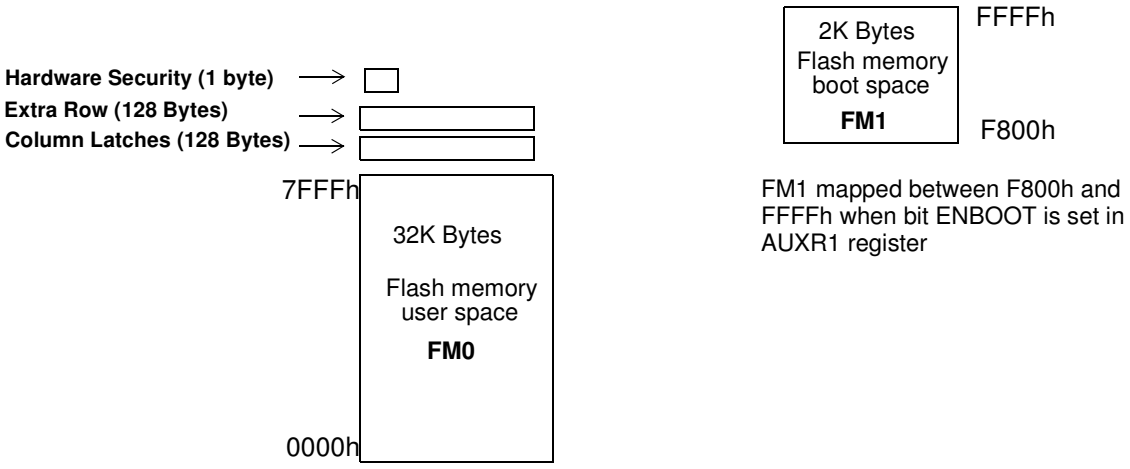
T89C51CC01 features two on-chip Flash memories:

- Flash memory FM0:
  containing 32K Bytes of program memory (user space) organized into 128 byte pages,
- Flash memory FM1:
  2K Bytes for boot loader and Application Programming Interfaces (API).

The FM0 can be program by both parallel programming and Serial In-System-Programming (ISP) whereas FM1 supports only parallel programming by programmers. The ISP mode is detailed in the "In-System-Programming" section.

All Read/Write access operations on Flash Memory by user application are managed by a set of API described in the "In-System-Programming" section.

**Figure 20.** Flash Memory Architecture



## Operation Cross Memory Access

Space addressable in read and write are:

- RAM
- ERAM (Expanded RAM access by movx)
- XRAM (eXternal RAM)
- EEPROM DATA
- FM0 (user flash)
- Hardware byte
- XROW
- Boot Flash
- Flash Column latch

The table below provide the different kind of memory which can be accessed from different code location.

**Table 28.** Cross Memory Access

| | Action | RAM | XRAM ERAM | Boot FLASH | FM0 | E² Data | Hardware Byte | XROW |
|---|---|---|---|---|---|---|---|---|
| boot FLASH | Read | | | OK | OK | OK | OK | - |
| | Write | | | - | OK(1) | OK(1) | OK(1) | OK(1) |
| FM0 | Read | | | - | OK | OK | OK | - |
| | Write | | | - | OK (idle) | OK(1) | - | OK |
| External memory EA = 0 or Code Roll Over | Read | | | - | - | OK | - | - |
| | Write | | | - | - | OK(1) | - | - |

Note: 1. RWW: Read While Write

**Table 32.** Read MOVC A, @DPTR

| Code Execution | FCON Register | | | ENBOOT | DPTR | FM1 | FM0 | XROW | Hardware Byte | External Code |
|---|---|---|---|---|---|---|---|---|---|---|
| | FMOD1 | FMOD0 | FPS | | | | | | | |
| From FM0 | 0 | 0 | X | 0 | 0000h to 7FFFh | | OK | | | |
| | | | | 1 | 0000h to 7FFFh | | OK | | | |
| | | | | | F800h to FFFFh | Do not use this configuration | | | | |
| | 0 | 1 | X | X | 0000 to 007Fh See (1) | | | OK | | |
| | 1 | 0 | X | X | X | | | | OK | |
| | 1 | 1 | X | 0 | 000h to 7FFFh | | OK | | | |
| | | | | 1 | 0000h to 7FFFh | | OK | | | |
| | | | | | F800h to FFFFh | Do not use this configuration | | | | |
| From FM1 (ENBOOT =1 | 0 | 0 | 0 | 1 | 0000h to 7FFF | | OK | | | |
| | | | | | F800h to FFFFh | OK | | | | |
| | | | | 0 | X | NA | | | | |
| | | | 1 | 1 | X | | OK | | | |
| | | | | 0 | X | NA | | | | |
| | 0 | 1 | X | 1 | 0000h to 007h See (2) | | | OK | | |
| | | | | 0 | | NA | | | | |
| | 1 | 0 | X | 1 | X | | | | OK | |
| | | | | 0 | | NA | | | | |
| | 1 | 1 | X | 1 | 000h to 7FFFh | | OK | | | |
| | | | | 0 | | NA | | | | |
| External code: EA=0 or Code Roll Over | X | 0 | X | X | X | | | | | OK |

1. For DPTR higher than 007Fh only lowest 7 bits are decoded, thus the behavior is the same as for addresses from 0000h to 007Fh
2. For DPTR higher than 007Fh only lowest 7 bits are decoded, thus the behavior is the same as for addresses from 0000h to 007Fh

## Boot Process

### Software Boot Process Example

Many algorithms can be used for the software boot process. Below are descriptions of the different flags and Bytes.

Boot Loader Jump Bit (BLJB):
- This bit indicates if on RESET the user wants to jump to this application at address @0000h on FM0 or execute the boot loader at address @F800h on FM1.
- BLJB = 0 (i.e. bootloader FM1 executed after a reset) is the default Atmel factory programming.
- To read or modify this bit, the APIs are used.

Boot Vector Address (SBV):
- This byte contains the MSB of the user boot loader address in FM0.
- The default value of SBV is FCh (no user boot loader in FM0).
- To read or modify this byte, the APIs are used.

Extra Byte (EB) and Boot Status Byte (BSB):
- These Bytes are reserved for customer use.
- To read or modify these Bytes, the APIs are used.

### Hardware Boot Process

At the falling edge of RESET, the bit ENBOOT in AUXR1 register is initialized with the value of Boot Loader Jump Bit (BLJB).

Further at the falling edge of RESET if the following conditions (called Hardware condition) are detected. The FCON register is initialized with the value 00h and the PC is initialized with F800h (FM1 lower byte = Bootloader entry point).

Hardware Conditions:

- PSEN low(1)
- EA high,
- ALE high (or not connected).

The Hardware condition forces the bootloader to be executed, whatever BLJB value is. Then BLBJ will be checked.

If no hardware condition is detected, the FCON register is initialized with the value F0h. Then BLJB value will be checked.

Conditions are:

- If bit BLJB = 1:
  User application in FM0 will be started at @0000h (standard reset).
- If bit BLJB = 0:
  Boot loader will be started at @F800h in FM1.

Note: 1. As PSEN is an output port in normal operating mode (running user applications or bootloader applications) after reset it is recommended to release PSEN after the falling edge of Reset is signaled.
The hardware conditions are sampled at reset signal Falling Edge, thus they can be released at any time when reset input is low.

2. To ensure correct microcontroller startup, the PSEN pin should not be tied to ground during power-on.

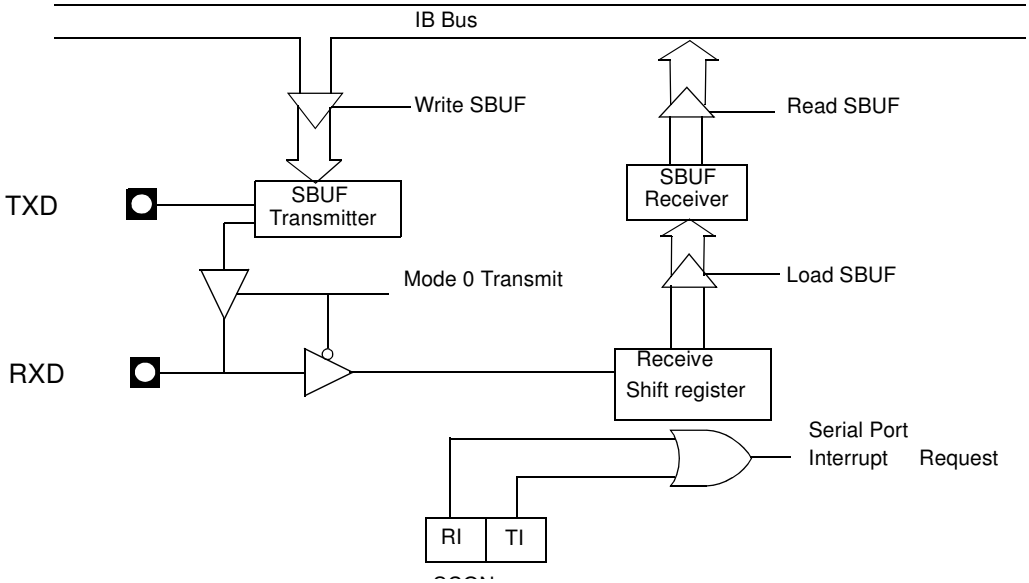
## Serial I/O Port

The T89C51CC01 I/O serial port is compatible with the I/O serial port in the 80C52. It provides both synchronous and asynchronous communication modes. It operates as a Universal Asynchronous Receiver and Transmitter (UART) in three full-duplex modes (Modes 1, 2 and 3). Asynchronous transmission and reception can occur simultaneously and at different baud rates

Serial I/O port includes the following enhancements:

- Framing error detection
- Automatic address recognition

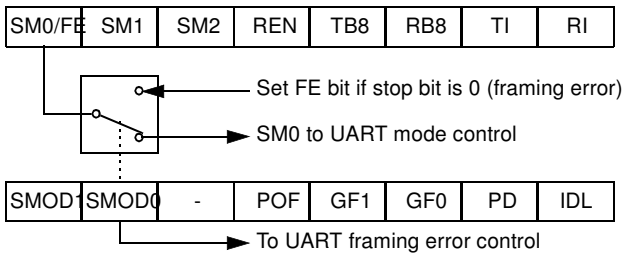
**Figure 27.** Serial I/O Port Block Diagram




### Framing Error Detection

Framing bit error detection is provided for the three asynchronous modes. To enable the framing bit error detection feature, set SMOD0 bit in PCON register.

**Figure 28.** Framing Error Block Diagram




When this feature is enabled, the receiver checks each incoming data frame for a valid stop bit. An invalid stop bit may result from noise on the serial lines or from simultaneous transmission by two CPUs. If a valid stop bit is not found, the Framing Error bit (FE) in SCON register bit is set.

The software may examine the FE bit after each reception to check for data errors. Once set, only software or a reset clears the FE bit. Subsequently received frames with valid stop bits cannot clear the FE bit. When the FE feature is enabled, RI rises on the stop bit instead of the last data bit (See Figure 29. and Figure 30.).

**Table 36.** SADEN Register

SADEN (S:B9h)
Slave Address Mask Register

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

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | | **Mask Data for Slave Individual Address** |

Reset Value = 0000 0000b
Not bit addressable

**Table 37.** SADDR Register

SADDR (S:A9h)
Slave Address Register

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

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | | **Slave Individual Address** |

Reset Value = 0000 0000b
Not bit addressable

**Table 38.** SBUF Register

SBUF (S:99h)
Serial Data Buffer

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

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-0 | | **Data sent/received by Serial I/O Port** |

Reset Value = 0000 0000b
Not bit addressable

## Timers/Counters

The T89C51CC01 implements two general-purpose, 16-bit Timers/Counters. Such are identified as Timer 0 and Timer 1, and can be independently configured to operate in a variety of modes as a Timer or an event Counter. When operating as a Timer, the Timer/Counter runs for a programmed length of time, then issues an interrupt request. When operating as a Counter, the Timer/Counter counts negative transitions on an external pin. After a preset number of counts, the Counter issues an interrupt request. The various operating modes of each Timer/Counter are described in the following sections.

## Timer/Counter Operations

A basic operation is Timer registers THx and TLx (x = 0, 1) connected in cascade to form a 16-bit Timer. Setting the run control bit (TRx) in TCON register (see Figure 40) turns the Timer on by allowing the selected input to increment TLx. When TLx overflows it increments THx; when THx overflows it sets the Timer overflow flag (TFx) in TCON register. Setting the TRx does not clear the THx and TLx Timer registers. Timer registers can be accessed to obtain the current count or to enter preset values. They can be read at any time but TRx bit must be cleared to preset their values, otherwise the behavior of the Timer/Counter is unpredictable.

The C/Tx# control bit selects Timer operation or Counter operation by selecting the divided-down peripheral clock or external pin Tx as the source for the counted signal. TRx bit must be cleared when changing the mode of operation, otherwise the behavior of the Timer/Counter is unpredictable.

For Timer operation (C/Tx# = 0), the Timer register counts the divided-down peripheral clock. The Timer register is incremented once every peripheral cycle (6 peripheral clock periods). The Timer clock rate is $F_{PER}/6$, i.e. $F_{OSC}/12$ in standard mode or $F_{OSC}/6$ in X2 mode.

For Counter operation (C/Tx# = 1), the Timer register counts the negative transitions on the Tx external input pin. The external input is sampled every peripheral cycles. When the sample is high in one cycle and low in the next one, the Counter is incremented. Since it takes 2 cycles (12 peripheral clock periods) to recognize a negative transition, the maximum count rate is $F_{PER}/12$, i.e. $F_{OSC}/24$ in standard mode or $F_{OSC}/12$ in X2 mode. There are no restrictions on the duty cycle of the external input signal, but to ensure that a given level is sampled at least once before it changes, it should be held for at least one full peripheral cycle.

## Timer 0

Timer 0 functions as either a Timer or event Counter in four modes of operation. Figure 31 to Figure 34 show the logical configuration of each mode.

Timer 0 is controlled by the four lower bits of TMOD register (see Figure 41) and bits 0, 1, 4 and 5 of TCON register (see Figure 40). TMOD register selects the method of Timer gating (GATE0), Timer or Counter operation (T/C0#) and mode of operation (M10 and M00). TCON register provides Timer 0 control functions: overflow flag (TF0), run control bit (TR0), interrupt flag (IE0) and interrupt type control bit (IT0).

For normal Timer operation (GATE0 = 0), setting TR0 allows TL0 to be incremented by the selected input. Setting GATE0 and TR0 allows external pin INT0# to control Timer operation.

Timer 0 overflow (count rolls over from all 1s to all 0s) sets TF0 flag generating an interrupt request.

It is important to stop Timer/Counter before changing mode.

**Table 42.** TH0 Register

TH0 (S:8Ch)
Timer 0 High Byte Register

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

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7:0 | | **High Byte of Timer 0.** |

Reset Value = 0000 0000b

**Table 43.** TL0 Register

TL0 (S:8Ah)
Timer 0 Low Byte Register

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

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7:0 | | **Low Byte of Timer 0.** |

Reset Value = 0000 0000b

**Table 44.** TH1 Register

TH1 (S:8Dh)
Timer 1 High Byte Register

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

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7:0 | | **High Byte of Timer 1.** |

Reset Value = 0000 0000b

## Programmable Clock-Output

In clock-out mode, timer 2 operates as a 50%-duty-cycle, programmable clock generator (See Figure 37). The input clock increments TL2 at frequency $F_{OSC}/2$. The timer repeatedly counts to overflow from a loaded value. At overflow, the contents of RCAP2H and RCAP2L registers are loaded into TH2 and TL2. In this mode, timer 2 overflows do not generate interrupts. The formula gives the clock-out frequency depending on the system oscillator frequency and the value in the RCAP2H and RCAP2L registers:

$$Clock-OutFrequency = \frac{FT2clock}{4 \times (65536 - RCAP2H/RCAP2L)}$$

For a 16 MHz system clock in x1 mode, timer 2 has a programmable frequency range of 61 Hz ($F_{OSC}/2^{16}$) to 4 MHz ($F_{OSC}/4$). The generated clock signal is brought out to T2 pin (P1.0).

Timer 2 is programmed for the clock-out mode as follows:

- Set T2OE bit in T2MOD register.
- Clear C/$\overline{T2}$ bit in T2CON register.
- Determine the 16-bit reload value from the formula and enter it in RCAP2H/RCAP2L registers.
- Enter a 16-bit initial value in timer registers TH2/TL2. It can be the same as the reload value or different depending on the application.
- To start the timer, set TR2 run control bit in T2CON register.

It is possible to use timer 2 as a baud rate generator and a clock generator simultaneously. For this configuration, the baud rates and clock frequencies are not independent since both functions use the values in the RCAP2H and RCAP2L registers.

**Figure 37.** Clock-Out Mode

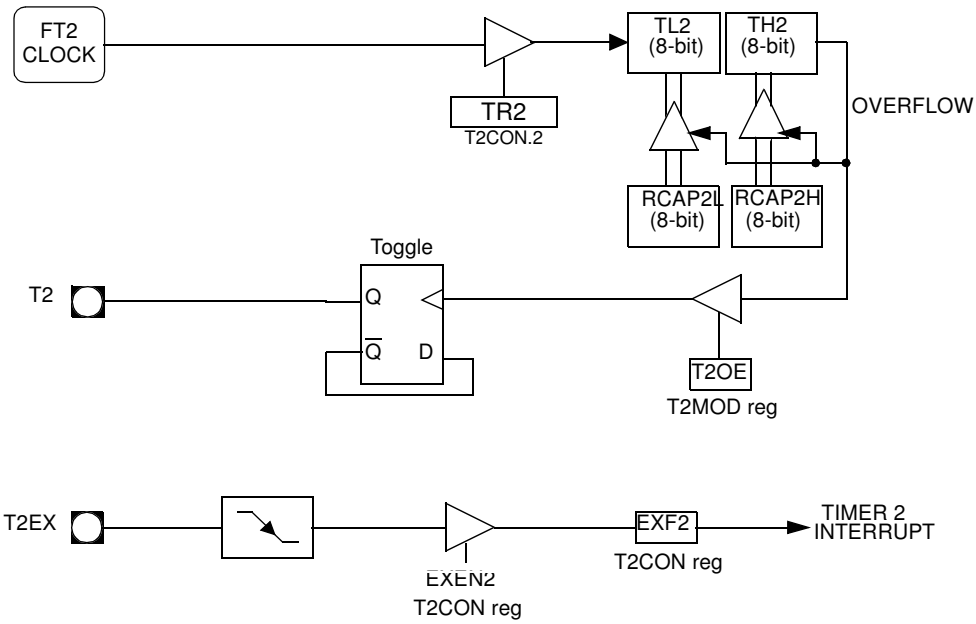


**Table 55.** WDTRST Register

WDTRST (S:A6h Write only)
Watchdog Timer Enable Register

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

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | Watchdog Control Value |

Reset Value = 1111 1111b

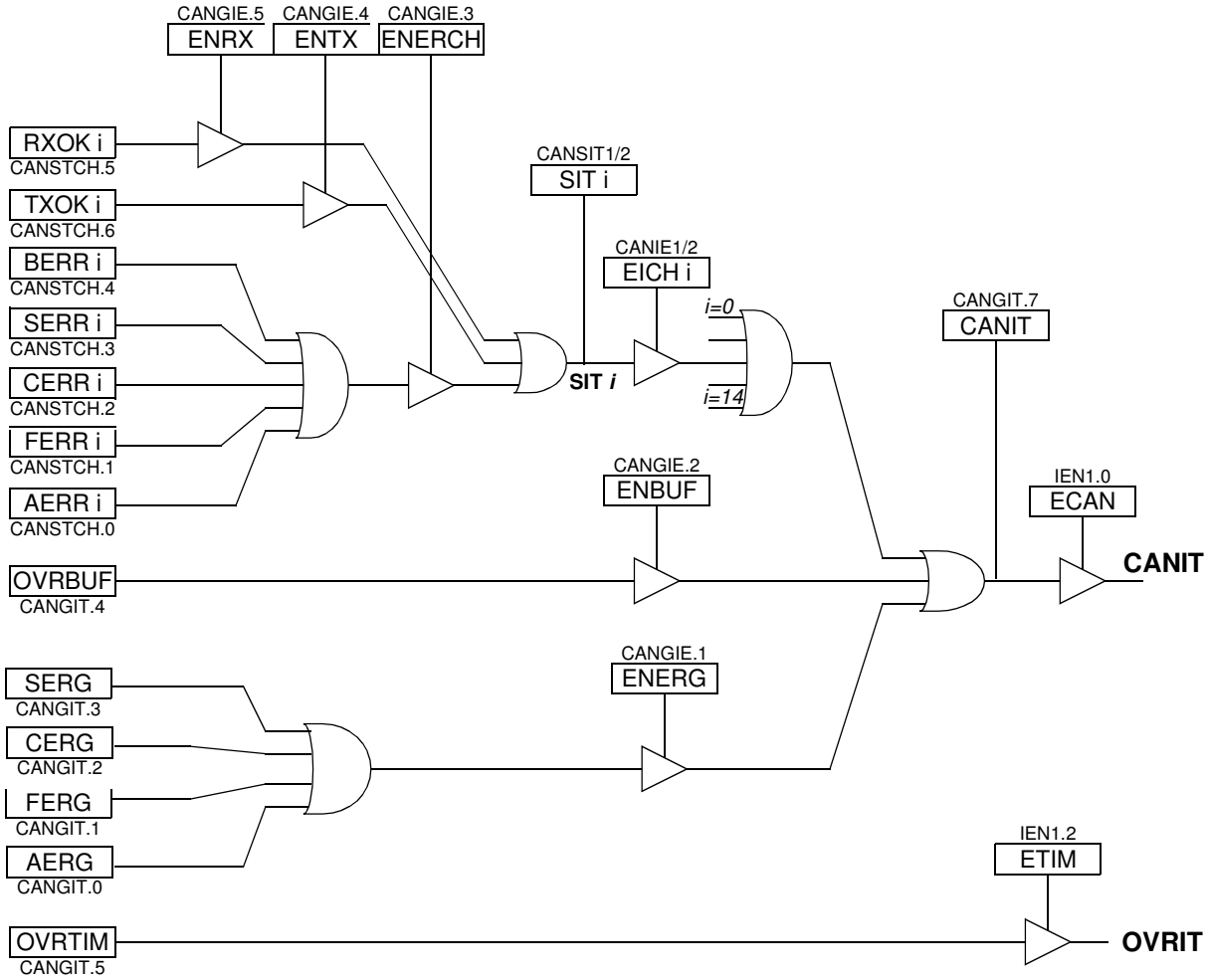
Note: The WDRST register is used to reset/enable the WDT by writing 1EH then E1H in sequence without instruction between these two sequences.

## IT CAN Management

The different interrupts are:

- Transmission interrupt,
- Reception interrupt,
- Interrupt on error (bit error, stuff error, crc error, form error, acknowledge error),
- Interrupt when Buffer receive is full,
- Interrupt on overrun of CAN Timer.

**Figure 46.** CAN Controller Interrupt Structure




To enable a transmission interrupt:

- Enable General CAN IT in the interrupt system register,
- Enable interrupt by message object, EICHi,
- Enable transmission interrupt, ENTX.

To enable a reception interrupt:

- Enable General CAN IT in the interrupt system register,
- Enable interrupt by message object, EICHi,

- Enable reception interrupt, ENRX.

To enable an interrupt on message object error:

- Enable General CAN IT in the interrupt system register,
- Enable interrupt by message object, EICHi,
- Enable interrupt on error, ENERCH.

To enable an interrupt on general error:

- Enable General CAN IT in the interrupt system register,
- Enable interrupt on error, ENERG.

To enable an interrupt on Buffer-full condition:

- Enable General CAN IT in the interrupt system register,
- Enable interrupt on Buffer full, ENBUF.

To enable an interrupt when Timer overruns:

- Enable Overrun IT in the interrupt system register.

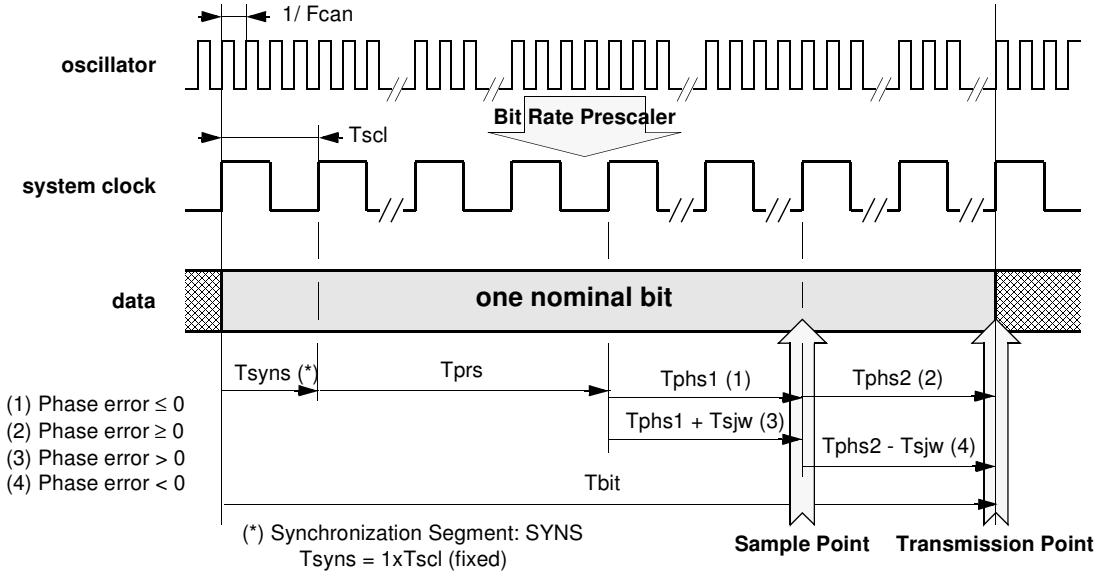
When an interrupt occurs, the corresponding message object bit is set in the SIT register.

To acknowledge an interrupt, the corresponding CANSTCH bits (RXOK, TXOK,...) or CANGIT bits (OVRTIM, OVRBUF,...), must be cleared by the software application.

When the CAN node is in transmission and detects a Form Error in its frame, a bit Error will also be raised. Consequently, two consecutive interrupts can occur, both due to the same error.

When a message object error occurs and is set in CANSTCH register, no general error are set in CANGIE register.

**Figure 48.** General Structure of a Bit Period




Tbit calculation: $Tbit = Tsyns + Tprs + Tphs1 + Tphs2$

**example of bit timing determination for CAN baudrate of 500kbit/s:**
Fosc = 12 MHz in X1 mode => FCAN = 6 MHz

Verify that the CAN baud rate you want is an integer division of FCAN clock.
FCAN/CAN baudrate = 6 MHz/500 kHz = 12
The time quanta TQ must be comprised between 8 and 25: TQ = 12 and **BRP = 0**

Define the various timing parameters: Tbit = Tsyns + Tprs + Tphs1 + Tphs2 = 12TQ
Tsyns = 1TQ and Tsjw =1TQ => **SJW = 0**
If we chose a sample point at 66.6% => Tphs2 = 4TQ => **PHS2 = 3**
Tbit = 12 = 4 + 1 + Tphs1 + Tprs, let us choose Tprs = 3 Tphs1 = 4
**PHS1 = 3** and **PRS = 2**

BRP = 0 so CANBT1 = 00h
SJW = 0 and PRS = 2 so CANBT2 = 04h
PHS2 = 3 and PHS1 = 3 so CANBT3 = 36h

**Table 63.** CANGIE Register

CANGIE (S:C1h)
CAN General Interrupt Enable

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | - | ENRX | ENTX | ENERCH | ENBUF | ENERG | - |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7-6 | - | **Reserved**<br>The values read from these bits are indeterminate. Do not set these bits. |
| 5 | ENRX | **Enable Receive Interrupt**<br>0 - Disable<br>1 - Enable |
| 4 | ENTX | **Enable Transmit Interrupt**<br>0 - Disable<br>1 - Enable |
| 3 | ENERCH | **Enable Message Object Error Interrupt**<br>0 - Disable<br>1 - Enable |
| 2 | ENBUF | **Enable BUF Interrupt**<br>0 - Disable<br>1 - Enable |
| 1 | ENERG | **Enable General Error Interrupt**<br>0 - Disable<br>1 - Enable |
| 0 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |

Note: See Figure 46

Reset Value = xx00 000xb

**Table 64.** CANEN1 Register

CANEN1 (S:CEh Read Only)
CAN Enable Message Object Registers 1

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | ENCH14 | ENCH13 | ENCH12 | ENCH11 | ENCH10 | ENCH9 | ENCH8 |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | **Reserved**<br>The values read from this bit is indeterminate. Do not set this bit. |
| 6-0 | ENCH14:8 | **Enable Message Object**<br>0 - message object is disabled => the message object is free for a new emission or reception.<br>1 - message object is enabled.<br>This bit is resetable by re-writing the CANCONCH of the corresponding message object. |

Reset Value = x000 0000b

**Table 75.** CANSTCH Register

CANSTCH (S:B2h)
CAN Message Object Status Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DLCW | TXOK | RXOK | BERR | SERR | CERR | FERR | AERR |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | DLCW | **Data Length Code Warning**<br>The incoming message does not have the DLC expected. Whatever the frame type, the DLC field of the CANCONCH register is updated by the received DLC. |
| 6 | TXOK | **Transmit OK**<br>The communication enabled by transmission is completed.<br>When the controller is ready to send a frame, if two or more message objects are enabled as producers, the lower index message object (0 to 13) is supplied first.<br>This flag can generate an interrupt and it must be cleared by software. |
| 5 | RXOK | **Receive OK**<br>The communication enabled by reception is completed.<br>In the case of two or more message object reception hits, the lower index message object (0 to 13) is updated first.<br>This flag can generate an interrupt and it must be cleared by software. |
| 4 | BERR | **Bit Error (Only in Transmission)**<br>The bit value monitored is different from the bit value sent.<br>Exceptions:<br>the monitored recessive bit sent as a dominant bit during the arbitration field and the acknowledge slot detecting a dominant bit during the sending of an error frame.<br>This flag can generate an interrupt and it must be cleared by software. |
| 3 | SERR | **Stuff Error**<br>Detection of more than five consecutive bits with the same polarity.<br>This flag can generate an interrupt and it must be cleared by software. |
| 2 | CERR | **CRC Error**<br>The receiver performs a CRC check on each destuffed received message from the start of frame up to the data field.<br>If this checking does not match with the destuffed CRC field, a CRC error is set.<br>This flag can generate an interrupt and it must be cleared by software. |
| 1 | FERR | **Form Error**<br>The form error results from one or more violations of the fixed form in the following bit fields:<br>CRC delimiter<br>acknowledgment delimiter<br>end_of_frame<br>This flag can generate an interrupt. |
| 0 | AERR | **Acknowledgment Error**<br>No detection of the dominant bit in the acknowledge slot.<br>This flag can generate an interrupt and it must be cleared by software. |

Note: See Figure 46.

No default value after reset.

## ADC Converter Operation

A start of single A/D conversion is triggered by setting bit ADSST (ADCON.3).

After completion of the A/D conversion, the ADSST bit is cleared by hardware.

The end-of-conversion flag ADEOC (ADCON.4) is set when the value of conversion is available in ADDH and ADDL, it must be cleared by software. If the bit EADC (IEN1.1) is set, an interrupt occur when flag ADEOC is set (see Figure 62). Clear this flag for re-arming the interrupt.

The bits SCH0 to SCH2 in ADCON register are used for the analog input channel selection.(1)

Note: 1. Always leave Tsetup time before starting a conversion unless ADEN is permanently high. In this case one should wait Tsetup only before the first conversion.

**Table 107.** Selected Analog input

| SCH2 | SCH1 | SCH0 | Selected Analog input |
|---|---|---|---|
| 0 | 0 | 0 | AN0 |
| 0 | 0 | 1 | AN1 |
| 0 | 1 | 0 | AN2 |
| 0 | 1 | 1 | AN3 |
| 1 | 0 | 0 | AN4 |
| 1 | 0 | 1 | AN5 |
| 1 | 1 | 0 | AN6 |
| 1 | 1 | 1 | AN7 |

## Voltage Conversion

When the ADCIN is equals to VAREF the ADC converts the signal to 3FFh (full scale). If the input voltage equals VAGND, the ADC converts it to 000h. Input voltage between VAREF and VAGND are a straight-line linear conversion. All other voltages will result in 3FFh if greater than VAREF and 000h if less than VAGND.

Note: ADCIN should not exceed VAREF absolute maximum range (see “Absolute Maximum Ratings” on page 144)

## Clock Selection

The ADC clock is the same as CPU.

The maximum clock frequency is defined in the DC parmeters for A/D converter. A prescaler is featured (ADCCLK) to generate the ADC clock from the oscillator frequency.

if PRS > 0 then $f_{ADC} = F_{periph} / 2 \times PRS$

if PRS = 0 then $f_{ADC} = F_{periph} / 64$

# Interrupt System

## Introduction

The CAN Controller has a total of 10 interrupt vectors: two external interrupts ($\overline{INT0}$ and $\overline{INT1}$), three timer interrupts (timers 0, 1 and 2), a serial port interrupt, a PCA, a CAN interrupt, a timer overrun interrupt and an ADC. These interrupts are shown below.

**Figure 63.** Interrupt Control System

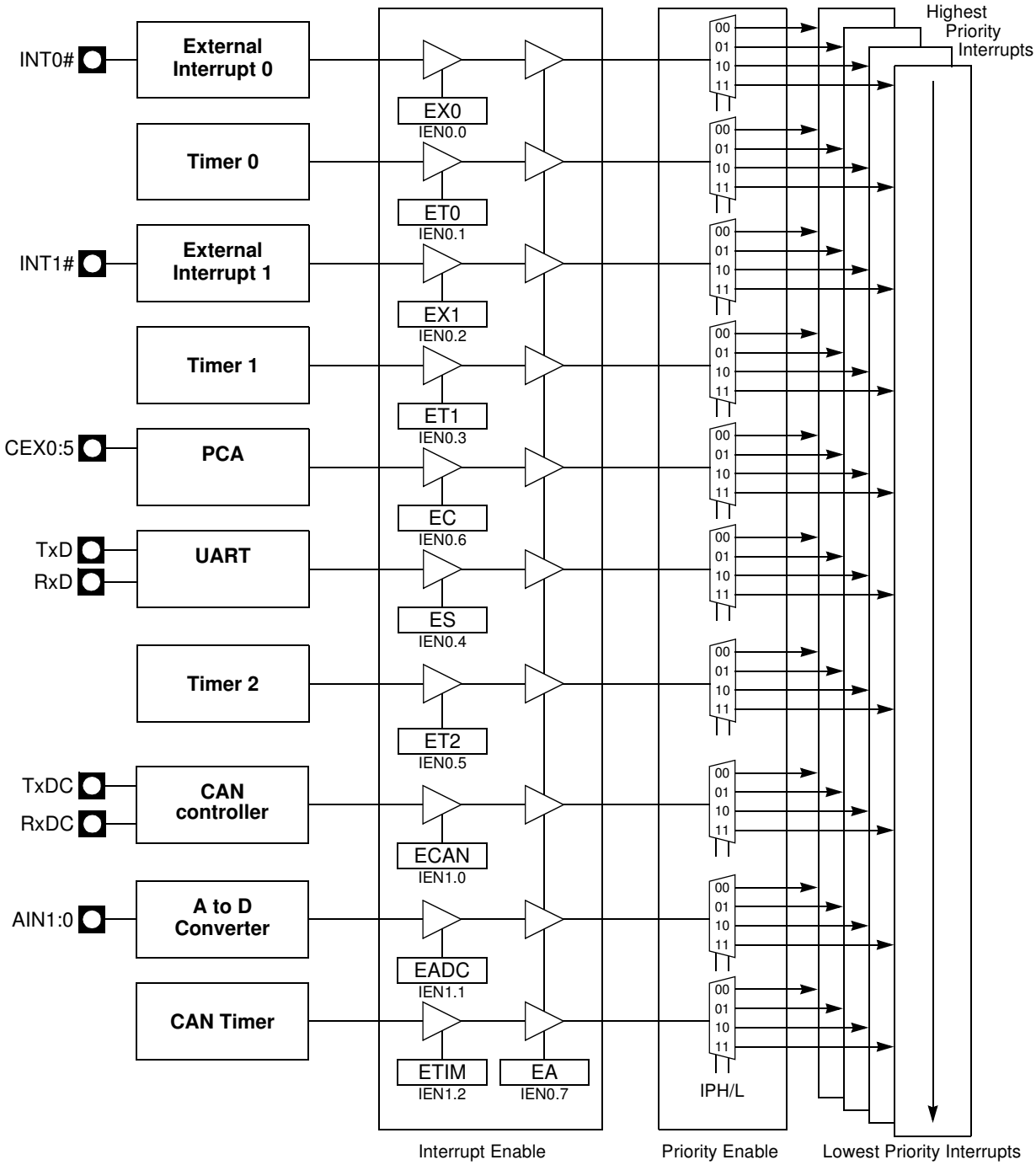


**Table 116.** IEN1 Register

IEN1 (S:E8h)
Interrupt Enable Register

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | ETIM | EADC | ECAN |

| Bit Number | Bit Mnemonic | Description |
|---|---|---|
| 7 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 6 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 5 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 4 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 3 | - | **Reserved**<br>The value read from this bit is indeterminate. Do not set this bit. |
| 2 | ETIM | **TImer Overrun Interrupt Enable bit**<br>Clear to disable the timer overrun interrupt.<br>Set to enable the timer overrun interrupt. |
| 1 | EADC | **ADC Interrupt Enable bit**<br>Clear to disable the ADC interrupt.<br>Set to enable the ADC interrupt. |
| 0 | ECAN | **CAN Interrupt Enable bit**<br>Clear to disable the CAN interrupt.<br>Set to enable the CAN interrupt. |

Reset Value = xxxx x000b
bit addressable