Welcome to E-XFL.COM

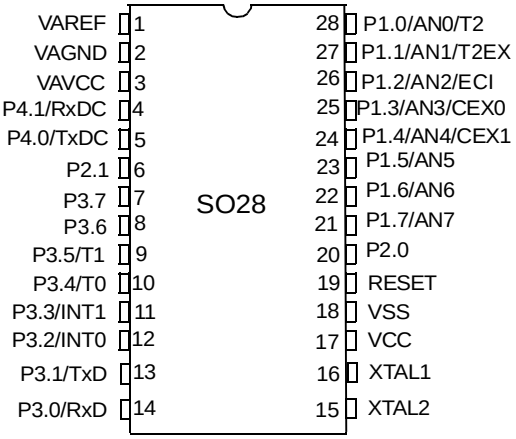
What is "Embedded - Microcontrollers"?

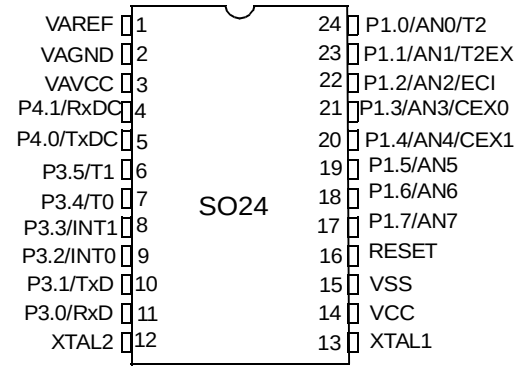
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" 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.

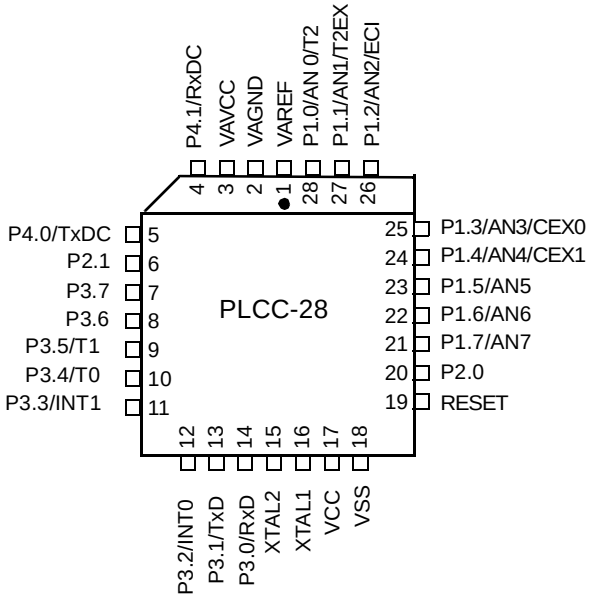
Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	80C51
Core Size	8-Bit
Speed	40MHz
Connectivity	CANbus, UART/USART
Peripherals	POR, PWM, WDT
Number of I/O	20
Program Memory Size	16KB (16K x 8)
Program Memory Type	FLASH
EEPROM Size	2K x 8
RAM Size	512 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	28-SOIC (0.295", 7.50mm Width)
Supplier Device Package	28-SO
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/t89c51cc02ua-tisim

Pin Configurations







Read-Modify-Write Instructions

Some instructions read the latch data rather than the pin data. The latch based instructions read the data, modify the data and then rewrite the latch. These are called 'Read-Modify-Write' instructions. Below is a complete list of these special instructions (See Table 1). When the destination operand is a Port or a Port bit, these instructions read the latch rather than the pin:

Table 1. Read/Modify/Write Instructions

Instruction	Description	Example
ANL	Logical AND	ANL P1, A
ORL	Logical OR	ORL P2, A
XRL	Logical EX-OR	XRL P3, A
JBC	Jump if bit = 1 and clear bit	JBC P1.1, LABEL
CPL	Complement bit	CPL P3.0
INC	Increment	INC P2
DEC	Decrement	DEC P2
DJNZ	Decrement and jump if not zero	DJNZ P3, LABEL
MOV Px.y, C	Move carry bit to bit y of Port x	MOV P1.5, C
CLR Px.y	Clear bit y of Port x	CLR P2.4
SET Px.y	Set bit y of Port x	SET P3.3

It is not obvious that the last three instructions in this list are Read-Modify-Write instructions. These instructions read the port (all 8 bits), modify the specifically addressed bit and write the new byte back to the latch. These Read-Modify-Write instructions are directed to the latch rather than the pin in order to avoid possible misinterpretation of voltage (and therefore, logic) levels at the pin. For example, a Port bit used to drive the base of an external bipolar transistor cannot rise above the transistor's base-emitter junction voltage (a value lower than VIL). With a logic one written to the bit, attempts by the CPU to read the Port at the pin are misinterpreted as logic zero. A read of the latch rather than the pins returns the correct logic one value.

Quasi Bi-directional Port Operation

Port 1, Port 3 and Port 4 have fixed internal pull-ups and are referred to as 'quasi-bidirectional' Ports. When configured as an input, the pin impedance appears as logic one and sources current in response to an external logic zero condition. Resets write logic one to all Port latches. If logical zero is subsequently written to a Port latch, it can be returned to input conditions by a logic one written to the latch.

Note: Port latch values change near the end of Read-Modify-Write insruction cycles. Output buffers (and therefore the pin state) are updated early in the instruction after Read-Modify-Write instruction cycle.

Logical zero-to-one transitions in Port 1, Port 3 and Port 4 use an additional pull-up (p1) to aid this logic transition See Figure 2. This increases switch speed. This extra pull-up sources 100 times normal internal circuit current during 2 oscillator clock periods. The internal pull-ups are field-effect transistors rather than linear resistors. Pull-ups consist of three p-channel FET (pFET) devices. A pFET is on when the gate senses logic zero and off when the gate senses logic one. pFET #1 is turned on for two oscillator periods immediately after a zero-to-one transition in the Port latch. A logic one at the Port pin turns on pFET #3 (a weak pull-up) through the inverter. This inverter and pFET pair form a latch to drive logic one. pFET #2 is a very weak pull-up switched on whenever the

SFR Mapping

Tables 3 through Table 11 show the Special Function Registers (SFRs) of the T89C51CC02.

Table 2. C51 Core SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
ACC	E0h	Accumulator								
B	F0h	B Register								
PSW	D0h	Program Status Word	CY	AC	F0	RS1	RS0	OV	F1	P
SP	81h	Stack Pointer								
DPL	82h	Data Pointer Low byte LSB of DPTR								
DPH	83h	Data Pointer High byte MSB of DPTR								

Table 3. I/O Port SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
P1	90h	Port 1								
P2	A0h	Port 2 (x2)								
P3	B0h	Port 3								
P4	C0h	Port 4 (x2)								

Table 4. Timers SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
TH0	8Ch	Timer/Counter 0 High byte								
TL0	8Ah	Timer/Counter 0 Low byte								
TH1	8Dh	Timer/Counter 1 High byte								
TL1	8Bh	Timer/Counter 1 Low byte								
TH2	CDh	Timer/Counter 2 High byte								
TL2	CCh	Timer/Counter 2 Low byte								
TCON	88h	Timer/Counter 0 and 1 control	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
TMOD	89h	Timer/Counter 0 and 1 Modes	GATE1	C/T1#	M11	M01	GATE0	C/T0#	M10	M00

Power Management

Two power reduction modes are implemented in the T89C51CC02: the Idle mode and the Power-down mode. These modes are detailed in the following sections. In addition to these power reduction modes, the clocks of the core and peripherals can be dynamically divided by 2 using the X2 Mode detailed in Section “Clock”.

Reset Pin

In order to start-up (cold reset) or to restart (warm reset) properly the microcontroller, a high level has to be applied on the RST pin. A bad level leads to a wrong initialisation of the internal registers like SFRs, PC, etc. and to unpredictable behavior of the microcontroller. A warm reset can be applied either directly on the RST pin or indirectly by an internal reset source such as a watchdog, PCA, timer, etc.

At Power-up (cold reset)

Two conditions are required before enabling a CPU start-up:

- VDD must reach the specified VDD range,
- The level on xtal1 input must be outside the specification (VIH, VIL).

If one of these two conditions are not met, the microcontroller does not start correctly and can execute an instruction fetch from anywhere in the program space. An active level applied on the RST pin must be maintained until both of the above conditions are met. A reset is active when the level VIH1 is reached and when the pulse width covers the period of time where VDD and the oscillator are not stabilized. Two parameters have to be taken into account to determine the reset pulse width:

- VDD rise time (vddrst),
- Oscillator startup time (oscrst).

To determine the capacitor the highest value of these two parameters has to be chosen. The reset circuitry is shown in Figure 5.

Figure 5. Reset Circuitry

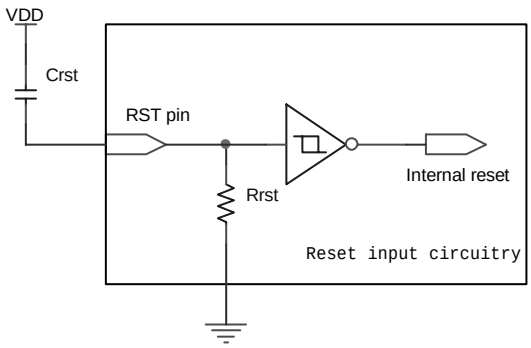


Table 13 and Table 14 give some typical examples for three values of VDD rise times, two values of oscillator start-up time and two pull-down resistor values.

Table 13. Minimum Reset Capacitor for a 50K Pull-down Resistor

oscrst/vddrst	1ms	10ms	100ms
5ms	820nF	1.2µF	12µF
20ms	2.7µF	3.9µF	12µF

Table 14. Minimum Reset Capacitor for a 15k Pull-down Resistor

oscrst/vddrst	1ms	10ms	100ms
5ms	2.7µF	4.7µF	47µF
20ms	10µF	15µF	47µF

Note: These values assume VDD starts from 0v to the nominal value. If the time between two on/off sequences is too fast, the power-supply decoupling capacitors may not be fully discharged, leading to a bad reset sequence.

During a Normal Operation (Warm Reset)

Reset pin must be maintained for at least 2 machine cycles (24 oscillator clock periods) to apply a reset sequence during normal operation. The number of clock periods is mode independent (X2 or X1).

Watchdog Reset

A 1K resistor must be added in series with the capacitor to allow the use of watchdog reset pulse output on the RST pin or when an external power-supply supervisor is used. Figure 6 shows the reset circuitry when a capacitor is used.

Figure 6. Reset Circuitry for a Watchdog Configuration

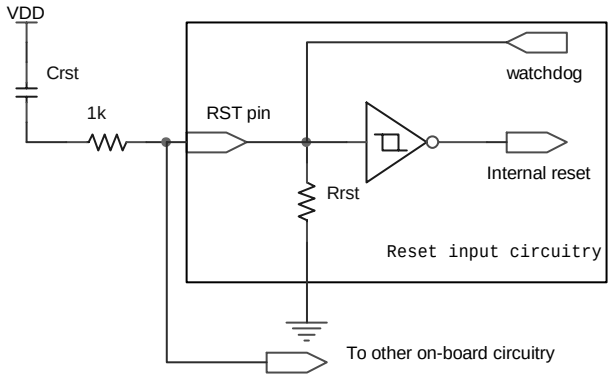
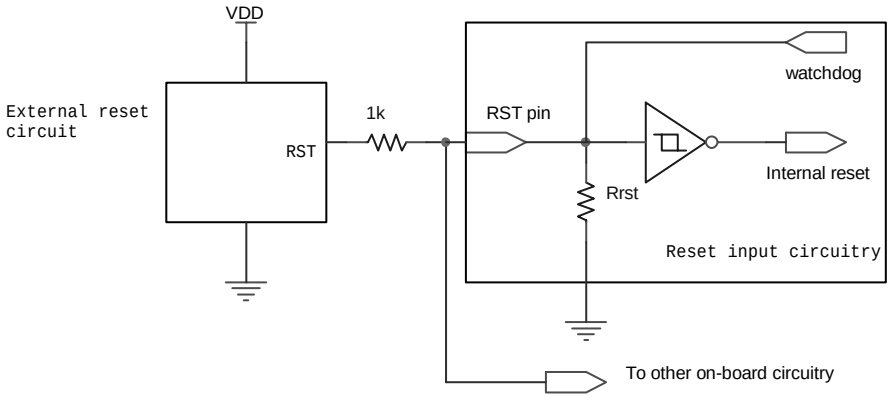


Figure 7 shows the reset circuitry when an external reset circuit is used.

Figure 7. Reset Circuitry Example Using an External Reset Circuit



Data Memory

The T89C51CC02 provides data memory access in two different spaces:

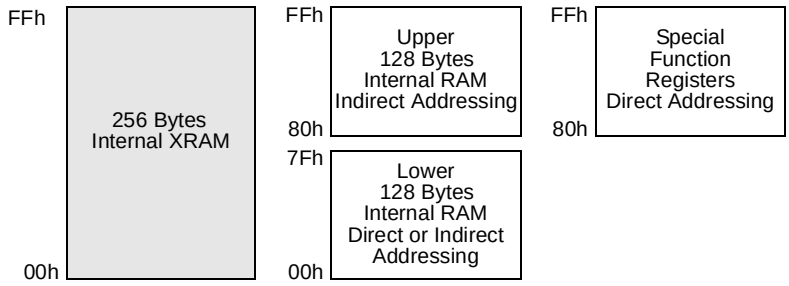
The internal space mapped in three separate segments:

- The lower 128 Bytes RAM segment.
- The upper 128 Bytes RAM segment.
- The expanded 256 Bytes RAM segment (XRAM).

A fourth internal segment is available but dedicated to Special Function Registers, SFRs, (addresses 80h to FFh) accessible by direct addressing mode.

Figure 9 shows the internal data memory spaces organization.

Figure 9. Internal memory - RAM



Internal Space

Lower 128 Bytes RAM

The lower 128 Bytes of RAM (See Figure 10) are accessible from address 00h to 7Fh using direct or indirect addressing modes. The lowest 32 Bytes are grouped into 4 banks of 8 registers (R0 to R7). Two bits RS0 and RS1 in PSW register (See Table 17) select which bank is in use according to Table 16. This allows more efficient use of code space, since register instructions are shorter than instructions that use direct addressing, and can be used for context switching in interrupt service routines.

Table 16. Register Bank Selection

RS1	RS0	Description
0	0	Register bank 0 from 00h to 07h
0	1	Register bank 0 from 08h to 0Fh
1	0	Register bank 0 from 10h to 17h
1	1	Register bank 0 from 18h to 1Fh

The next 16 Bytes above the register banks form a block of bit-addressable memory space. The C51 instruction set includes a wide selection of singlebit instructions, and the 128 bits in this area can be directly addressed by these instructions. The bit addresses in this area are 00h to 7Fh.

Figure 22. UART Timing in Mode 1

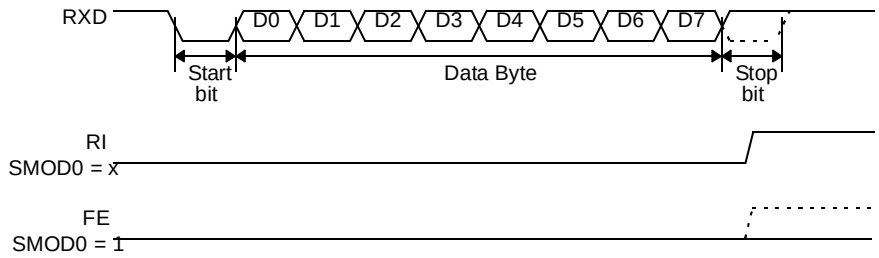
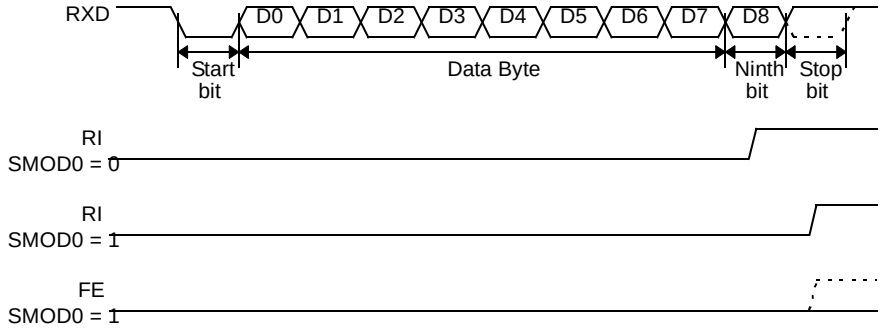


Figure 23. UART Timing in Modes 2 and 3



Automatic Address Recognition

The automatic address recognition feature is enabled when the multiprocessor communication feature is enabled (SM2 bit in SCON register is set).

Implemented in the hardware, automatic address recognition enhances the multiprocessor communication feature by allowing the serial port to examine the address of each incoming command frame. Only when the serial port recognizes its own address will the receiver set the RI bit in the SCON register to generate an interrupt. This ensures that the CPU is not interrupted by command frames addressed to other devices.
If necessary, the user can enable the automatic address recognition feature in mode 1. In this configuration, the stop bit takes the place of the ninth data bit. bit RI is set only when the received command frame address matches the device's address and is terminated by a valid stop bit.

To support automatic address recognition, a device is identified by a given address and a broadcast address.

Note: The multiprocessor communication and automatic address recognition features cannot be enabled in mode 0 (i.e. setting SM2 bit in SCON register in mode 0 has no effect).

Given Address

Each device has an individual address that is specified in the SADDR register; the SADEN register is a mask byte that contains don't-care bits (defined by zeros) to form the device's given address. The don't-care bits provide the flexibility to address one or more slaves at a time. The following example illustrates how a given address is formed. To address a device by its individual address, the SADEN mask byte must be 1111 1111b.
For example:

```
SADDR0101 0110b
SADEN1111 1100b
Given0101 01XXb
```

Timers/Counters

The T89C51CC02 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 37) 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 24 through Figure 27 show the logical configuration of each mode.

Timer 0 is controlled by the four lower bits of TMOD register (See Figure 38) and bits 0, 1, 4 and 5 of TCON register (See Figure 37). 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.

- For normal Timer operation (GATE1= 0), setting TR1 allows TL1 to be incremented by the selected input. Setting GATE1 and TR1 allows external pin INT1# to control Timer operation.
- Timer 1 overflow (count rolls over from all 1s to all 0s) sets the TF1 flag generating an interrupt request.
- When Timer 0 is in mode 3, it uses Timer 1's overflow flag (TF1) and run control bit (TR1). For this situation, use Timer 1 only for applications that do not require an interrupt (such as a Baud Rate Generator for the Serial Port) and switch Timer 1 in and out of mode 3 to turn it off and on.
- It is important to stop Timer/Counter before changing mode.

Mode 0 (13-bit Timer)

Mode 0 configures Timer 1 as a 13-bit Timer, which is set up as an 8-bit Timer (TH1 register) with a modulo-32 prescaler implemented with the lower 5 bits of the TL1 register (See Figure 24). The upper 3 bits of TL1 register are ignored. Prescaler overflow increments TH1 register.

Mode 1 (16-bit Timer)

Mode 1 configures Timer 1 as a 16-bit Timer with TH1 and TL1 registers connected in cascade (See Figure 25). The selected input increments TL1 register.

Mode 2 (8-bit Timer with Auto-Reload)

Mode 2 configures Timer 1 as an 8-bit Timer (TL1 register) with automatic reload from TH1 register on overflow (See Figure 26). TL1 overflow sets TF1 flag in TCON register and reloads TL1 with the contents of TH1, which is preset by software. The reload leaves TH1 unchanged.

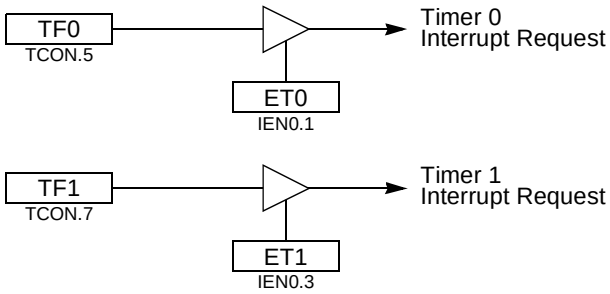
Mode 3 (Halt)

Placing Timer 1 in mode 3 causes it to halt and hold its count. This can be used to halt Timer 1 when TR1 run control bit is not available i.e. when Timer 0 is in mode 3.

Interrupt

Each Timer handles one interrupt source that is the timer overflow flag TF0 or TF1. This flag is set every time an overflow occurs. Flags are cleared when vectoring to the Timer interrupt routine. Interrupts are enabled by setting ETx bit in IEN0 register. This assumes interrupts are globally enabled by setting EA bit in IEN0 register.

Figure 28. Timer Interrupt System



Registers

Table 37. TCON Register
TCON (S:88h)
Timer/Counter Control Register

7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Bit Number	Bit Mnemonic	Description
7	TF1	**Timer 1 Overflow Flag** Cleared by hardware when processor vectors to interrupt routine. Set by hardware on Timer/Counter overflow, when Timer 1 register overflows.
6	TR1	**Timer 1 Run Control bit** Clear to turn off Timer/Counter 1. Set to turn on Timer/Counter 1.
5	TF0	**Timer 0 Overflow Flag** Cleared by hardware when processor vectors to interrupt routine. Set by hardware on Timer/Counter overflow, when Timer 0 register overflows.
4	TR0	**Timer 0 Run Control bit** Clear to turn off Timer/Counter 0. Set to turn on Timer/Counter 0.
3	IE1	**Interrupt 1 Edge Flag** Cleared by hardware when interrupt is processed if edge-triggered (See IT1). Set by hardware when external interrupt is detected on INT1# pin.
2	IT1	**Interrupt 1 Type Control bit** Clear to select low level active (level triggered) for external interrupt 1 (INT1#). Set to select falling edge active (edge triggered) for external interrupt 1.
1	IE0	**Interrupt 0 Edge Flag** Cleared by hardware when interrupt is processed if edge-triggered (See IT0). Set by hardware when external interrupt is detected on INT0# pin.
0	IT0	**Interrupt 0 Type Control bit** Clear to select low level active (level triggered) for external interrupt 0 (INT0#). Set to select falling edge active (edge triggered) for external interrupt 0.

Reset Value = 0000 0000b

Watchdog Programming

The three lower bits (S0, S1, S2) located into WDTPRG register permit to program the WDT duration.

Table 49. Machine Cycle Count

S2	S1	S0	Machine Cycle Count
0	0	0	$2^{14} - 1$
0	0	1	$2^{15} - 1$
0	1	0	$2^{16} - 1$
0	1	1	$2^{17} - 1$
1	0	0	$2^{18} - 1$
1	0	1	$2^{19} - 1$
1	1	0	$2^{20} - 1$
1	1	1	$2^{21} - 1$

To compute WD Timeout, the following formula is applied:

$$FTime-Out = \frac{F_{wd}}{12 \times ((2^{14} \times 2^{Svalue}) - 1)}$$

Note: Svalue represents the decimal value of (S2 S1 S0)

Find Hereafter computed Timeout values for f_{OSC}XTAL = 12 MHz in X1 mode

Table 50. Timeout Computation

S2	S1	S0	f_{osc}=12 MHz	f_{osc}=16MHz	f_{osc}=20 MHz
0	0	0	16.38 ms	12.28 ms	9.82 ms
0	0	1	32.77 ms	24.57 ms	19.66 ms
0	1	0	65.54 ms	49.14 ms	39.32 ms
0	1	1	131.07 ms	98.28 ms	78.64 ms
1	0	0	262.14 ms	196.56 ms	157.28 ms
1	0	1	524.29 ms	393.12 ms	314.56 ms
1	1	0	1.05 s	786.24 ms	629.12 ms
1	1	1	2.10 s	1.57 s	1.25 ms

CAN Controller

The CAN Controller provides all the features required to implement the serial communication protocol CAN as defined by BOSCH GmbH. The CAN specification as referred to by ISO/11898 (2.0A & 2.0B) for high speed and ISO/11519-2 for low speed. The CAN Controller is able to handle all types of frames (Data, Remote, Error and Overload) and achieves a bitrate of 1-Mbit/s at 8 MHz[1] Crystal frequency in X2 Mode.

Note: 1. At BRP = 1 sampling point will be fixed.

CAN Controller Description

The CAN controller accesses are made through SFR.
Several operations are possible by SFR:

- arithmetic and logic operations, transfers and program control (SFR is accessible by direct addressing).
- 4 independent message objects are implemented, a pagination system manages their accesses.

Any message object can be programmed in a reception buffer block (even non-consecutive buffers). For the reception of defined messages one or several receiver message objects can be masked without participating in the buffer feature. An IT is generated when the buffer is full. The frames following the buffer-full interrupt will not be taken into account until at least one of the buffer message objects is re-enabled in reception. Higher priority of a message object for reception or transmission is given to the lower message object number.

The programmable 16-bit Timer (CANTIMER) is used to stamp each received and sent message in the CANSTMP register. This timer starts counting as soon as the CAN controller is enabled by the ENA bit in the CANGCON register.

The Time Trigger Communication (TTC) protocol is supported by the T89C51CC02.

Figure 32. CAN Controller Block Diagram

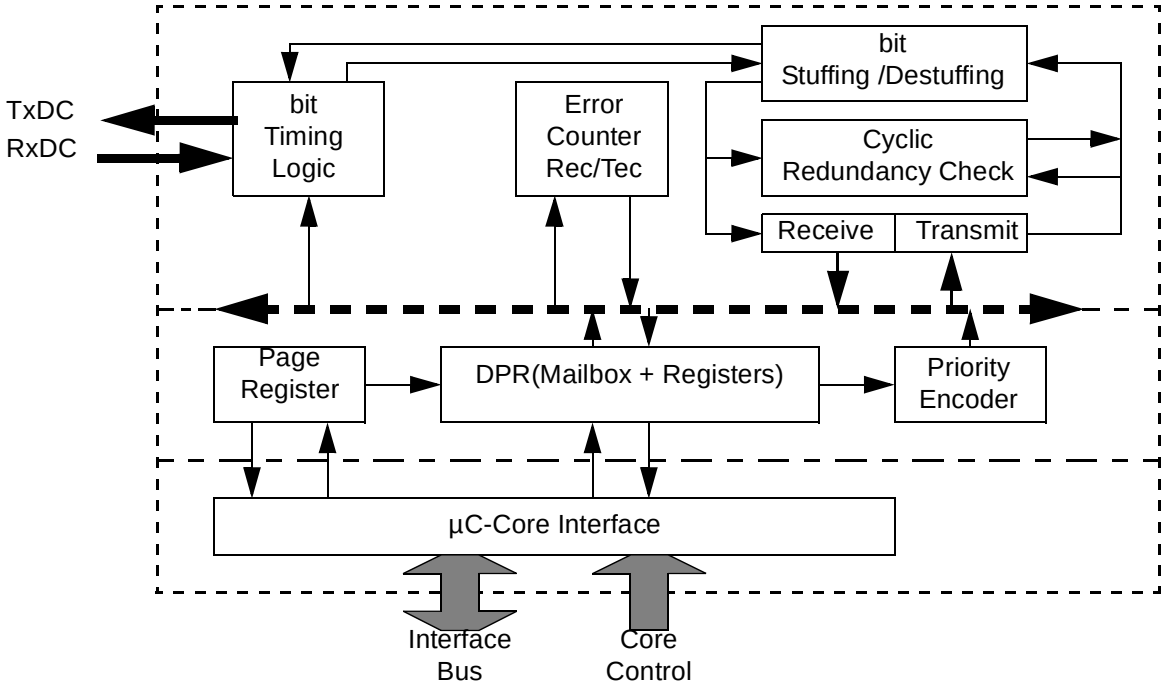
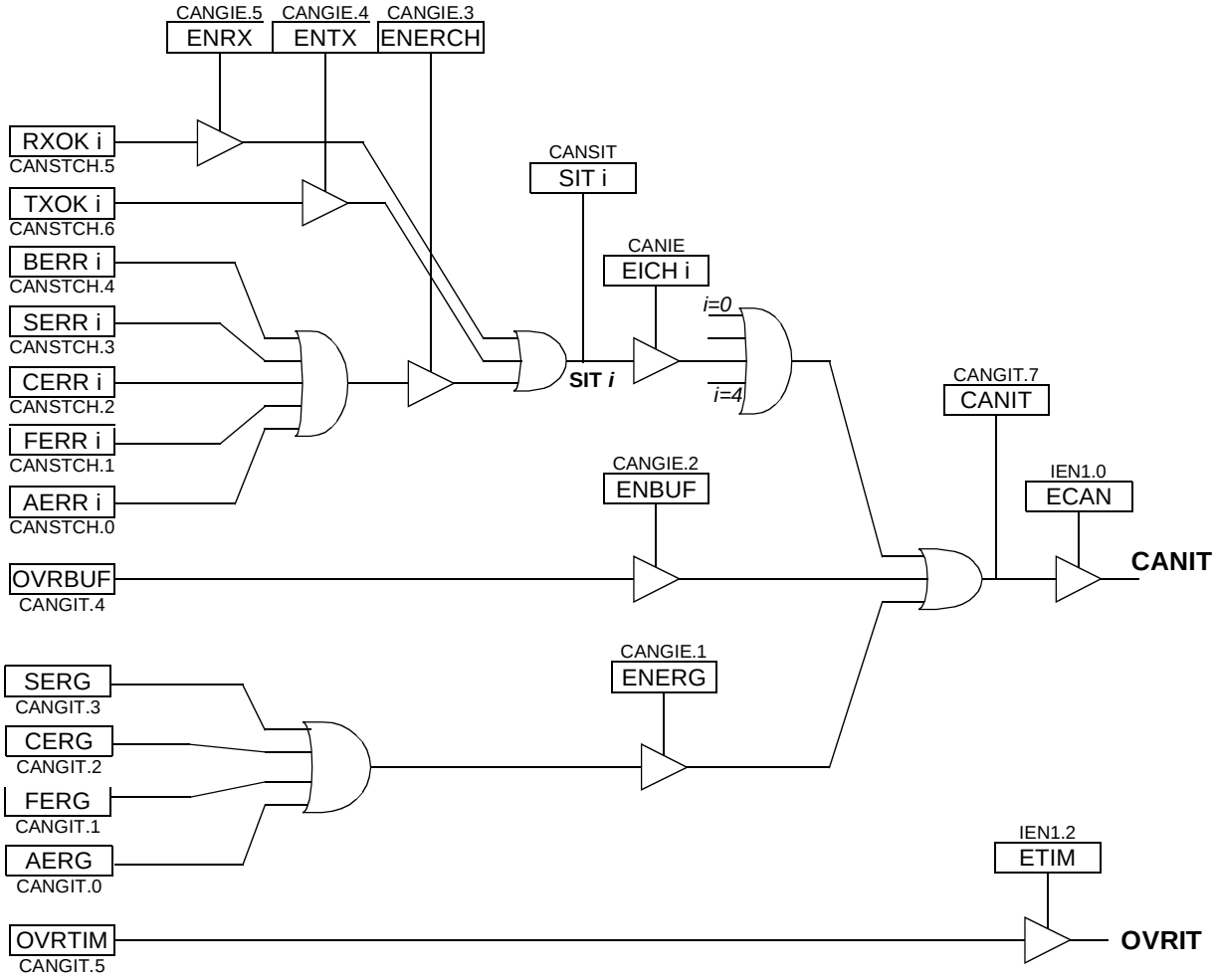


Figure 35. CAN Controller Interrupt Structure



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

```
// Enable the CAN macro
 CANGCON = 02h
2. Configure message object 3 in reception to receive only standard (11bit
identifier) message 100h
// Select the message object 3
 CANPAGE = 30h
// Enable the interrupt on this message object
 CANIE = 08h
// Clear the status and control register
 CANSTCH = 00h
 CANCONCH= 00h
// Init the acceptance filter to accept only message 100h in standard mode
 CANIDT1 = 20h
 CANIDT2 = 00h
 CANIDT3 = 00h
 CANIDT4 = 00h
 CANIDM1 = FFh
 CANIDM2 = FFh
 CANIDM3 = FFh
 CANIDM4 = FFh
// Enable channel in reception
 CANCONCH = 88h // enable reception

Note: to enable the CAN interrupt in reception:
EA = 1
ECAN = 1
CANGIE = 20h
3. Send a message on the message object 0
// Select the message object 0
 CANPAGE = 00h
// Enable the interrupt on this message object
 CANIE = 01h
// Clear the Status register
 CANSTCH = 00h;
// load the identifier to send (ex: 555h)
 CANIDT1 = AAh;
 CANIDT2 = A0h;
// load data to send
 CANMSG = 00h
 CANMSG = 01h
 CANMSG = 02h
 CANMSG = 03h
 CANMSG = 04h
 CANMSG = 05h
 CANMSG = 06h
 CANMSG = 07h
// configure the control register
 CANCONCH = 18h
4. Interrupt routine
// Save the current CANPAGE
```

CAN SFRs

Table 54. SFR Mapping

	0/8[1]	1/9	2/A	3/B	4/C	5/D	6/E	7/F	
F8h	**IPL1** **xxxx x000**	CH 0000 0000	CCAP0H 0000 0000	CCAP1H 0000 0000					FFh
F0h	**B** **0000 0000**		ADCLK xxx0 0000	ADCON x000 0000	ADDL 0000 0000	ADDH 0000 0000	ADCF 0000 0000	IPH1 xxxx x000	F7h
E8h	**IEN1** **xxxx x000**	CL 0000 0000	CCAP0L 0000 0000	CCAP1L 0000 0000					EFh
E0h	**ACC** **0000 0000**								E7h
D8h	**CCON** **0000 0000**	CMOD 0xxx x000	CCAPM0 x000 0000	CCAPM1 x000 0000					DFh
D0h	**PSW** **0000 0000**	FCON 0000 0000	EECON xxxx xx00						D7h
C8h	**T2CON** 0000 0000	T2MOD xxxx xx00	RCAP2L 0000 0000	RCAP2H 0000 0000	TL2 0000 0000	TH2 0000 0000		CANEN xxxx 0000	CFh
C0h	**P4** **xxxx xx11**	CANGIE 1100 0000		CANIE 1111 0000	CANIDM1 xxxx xxxx	CANIDM2 xxxx xxxx	CANIDM3 xxxx xxxx	CANIDM4 xxxx xxxx	C7h
B8h	**IPL0** **x000 0000**	SADEN 0000 0000		CANSIT xxxx 0000	CANIDT1 xxxx xxxx	CANIDT2 xxxx xxxx	CANIDT3 xxxx xxxx	CANIDT4 xxxx xxxx	BFh
B0h	**P3** **1111 1111**	CANPAGE 1100 0000	CANSTCH xxxx xxxx	CANCONCH xxxx xxxx	CANBT1 xxxx xxxx	CANBT2 xxxx xxxx	CANBT3 xxxx xxxx	IPH0 x000 0000	B7h
A8h	**IEN0** **0000 0000**	SADDR 0000 0000	CANGSTA 1010 0000	CANGCON 0000 0000	CANTIML 0000 0000	CANTIMH 0000 0000	CANSTMPL xxxx xxxx	CANSTMPH xxxx xxxx	AFh
A0h	**P2** **xxxx xx11**	CANTCON 0000 0000	AUXR1[2] xxxx 00x0	CANMSG xxxx xxxx	CANTTCL 0000 0000	CANTTCH 0000 0000	WDTRST 1111 1111	WDTPRG xxxx x000	A7h
98h	**SCON** **0000 0000**	SBUF 0000 0000		CANGIT 0x00 0000	CANTEC 0000 0000	CANREC 0000 0000			9Fh
90h	**P1** **1111 1111**								97h
88h	**TCON** **0000 0000**	TMOD 0000 0000	TL0 0000 0000	TL1 0000 0000	TH0 0000 0000	TH1 0000 0000		CKCON 0000 0000	8Fh
80h		SP 0000 0111	DPL 0000 0000	DPH 0000 0000				PCON 00x1 0000	87h
	0/8[1]	1/9	2/A	3/B	4/C	5/D	6/E	7/F	

Table 63. CANIE Register

CANIE (S:C3h) – CAN Enable Interrupt message object Registers 2

7	6	5	4	3	2	1	0
-	-	-	-	IECH 3	IECH 2	IECH 1	IECH 0

Bit Number	Bit Mnemonic	Description
7 - 4	-	**Reserved** The values read from these bits are indeterminate. Do not set these bits.
3 - 0	IECH3:0	**Enable Interrupt by Message Object** 0 - disable IT. 1 - enable IT. IECH3:0 = 0b 0000 1100 -> Enable IT's of message objects 3 & 2.

Reset Value = xxxx 0000b

Table 64. CANBT1 Register

CANBT1 (S:B4h) – CAN bit Timing Registers 1

7	6	5	4	3	2	1	0
-	BRP 5	BRP 4	BRP 3	BRP 2	BRP 1	BRP 0	-

Bit Number	Bit Mnemonic	Description
7	-	**Reserved** The value read from this bit is indeterminate. Do not set this bit.
6 - 1	BRP5:0	**Baud Rate Prescaler** **The period of the CAN** controller system clock Tscl is programmable and determines the individual bit timing.(1) $Tscl = \frac{BRP[5..0] + 1}{F_{CAN}}$
0	-	**Reserved** The value read from this bit is indeterminate. Do not set this bit.

Note: 1. The CAN controller bit timing registers must be accessed only if the CAN controller is disabled with the ENA bit of the CANGCON register set to 0.
See Figure 37.

No default value after reset.

Table 70. CANIDT1 Register for V2.0 part A
CANIDT1 for V2.0 part A (S:BCh) – CAN Identifier Tag Registers 1

7	6	5	4	3	2	1	0
IDT 10	IDT 9	IDT 8	IDT 7	IDT 6	IDT 5	IDT 4	IDT 3

Bit Number	Bit Mnemonic	Description
7 - 0	IDT10:3	**IDentifier Tag Value** See Figure 39.

No default value after reset.

Table 71. CANIDT2 Register for V2.0 part A
CANIDT2 for V2.0 part A (S:BDh) – CAN Identifier Tag Registers 2

7	6	5	4	3	2	1	0
IDT 2	IDT 1	IDT 0	-	-	-	-	-

Bit Number	Bit Mnemonic	Description
7 - 5	IDT2:0	**IDentifier Tag Value** See Figure 39.
4-0	-	**Reserved** The values read from these bits are indeterminate. Do not set these bits.

No default value after reset.

Table 72. CANIDT3 Register for V2.0 part A
CANIDT3 for V2.0 part A (S:BEh) –CAN Identifier Tag Registers 3

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

Bit Number	Bit Mnemonic	Description
7 - 0	-	**Reserved** The values read from these bits are indeterminate. Do not set these bits.

No default value after reset.

Programmable Counter Array (PCA)

The PCA provides more timing capabilities with less CPU intervention than the standard timer/counters. Its advantages include reduced software overhead and improved accuracy. The PCA consists of a dedicated timer/counter which serves as the time base for an array of two compare/capture modules. Its clock input can be programmed to count any of the following signals:

- PCA clock frequency/6 (See “clock” section)
- PCA clock frequency/2
- Timer 0 overflow
- External input on ECI (P1.2)

Each compare/capture modules can be programmed in any one of the following modes:

- Rising and/or falling edge capture,
- Software timer
- High-speed output
- Pulse width modulator

When the compare/capture modules are programmed in capture mode, software timer, or high speed output mode, an interrupt can be generated when the module executes its function. Both modules and the PCA timer overflow share one interrupt vector.

The PCA timer/counter and compare/capture modules share Port 1 for external I/Os. These pins are listed below. If the pin is not used for the PCA, it can still be used for standard I/O.

PCA Component	External I/O Pin
16-bit Counter	P1.2/ECI
16-bit Module 0	P1.3/CEX0
16-bit Module 1	P1.4/CEX1

PCA Timer

The PCA timer is a common time base for both modules (See Figure 9). The timer count source is determined from the CPS1 and CPS0 bits in the **CMOD SFR** (See Table 8) and can be programmed to run at:

- 1/6 the PCA clock frequency.
- 1/2 the PCA clock frequency.
- The Timer 0 overflow.
- The input on the ECI pin (P1.2).

Table 95. CCON Register

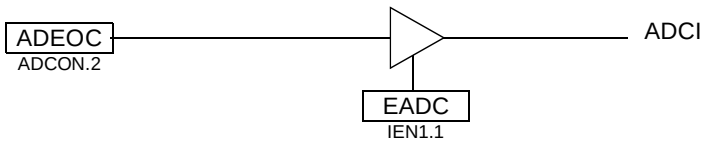
CCON (S:D8h)
PCA Counter Control Register

7	6	5	4	3	2	1	0
CF	CR	-	-	-	-	CCF1	CCF0

Bit Number	Bit Mnemonic	Description
7	CF	**PCA Timer/Counter Overflow flag** Set by hardware when the PCA Timer/Counter rolls over. This generates a PCA interrupt request if the ECF bit in CMOD register is set. Must be cleared by software.
6	CR	**PCA Timer/Counter Run Control bit** Clear to turn the PCA Timer/Counter off. Set to turn the PCA Timer/Counter on.
5-2	-	**Reserved** The value read from these bist are indeterminate. Do not set these bits.
1	CCF1	**PCA Module 1 Compare/Capture Flag** Set by hardware when a match or capture occurs. This generates a PCA interrupt request if the ECCF 1 bit in CCAPM 1 register is set. Must be cleared by software.
0	CCF0	**PCA Module 0 Compare/Capture Flag** Set by hardware when a match or capture occurs. This generates a PCA interrupt request if the ECCF 0 bit in CCAPM 0 register is set. Must be cleared by software.

Reset Value = 00xx xx00b

Figure 51. ADC interrupt structure



Routine Examples

```
1. Configure P1.2 and P1.3 in ADC channels
// configure channel P1.2 and P1.3 for ADC
 ADCF = 0Ch

// Enable the ADC
 ADCON = 20h
2. Start a standard conversion
// The variable 'channel' contains the channel to convert
// The variable 'value_converted' is an unsigned int
// Clear the field SCH[2:0]
 ADCON &= F8h
// Select channel
 ADCON |= channel
// Start conversion in standard mode
 ADCON |= 08h
// Wait flag End of conversion
 while((ADCON & 01h)!= 01h)
// Clear the End of conversion flag
 ADCON &= EFh
// read the value
 value_converted = (ADDH << 2)+(ADDL)

3. Start a precision conversion (need interrupt ADC)
// The variable 'channel' contains the channel to convert
// Enable ADC
 EADC = 1
// clear the field SCH[2:0]
 ADCON &= F8h
// Select the channel
 ADCON |= channel
// Start conversion in precision mode
 ADCON |= 48h
```

Note: To enable the ADC interrupt: EA = 1