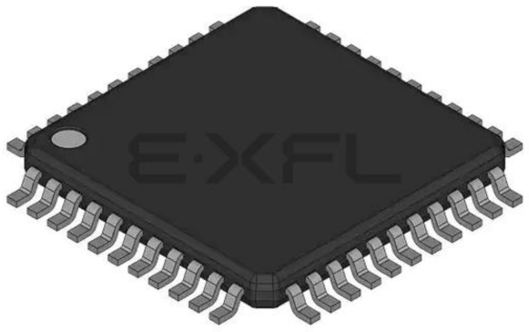


Welcome to E-XFL.COM

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.

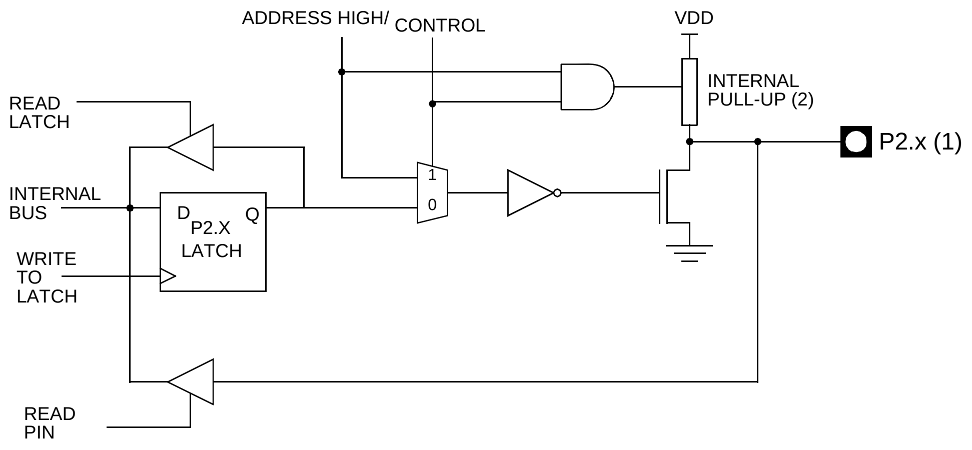
Applications of "Embedded - Microcontrollers"

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	36
Program Memory Size	64KB (64K x 8)
Program Memory Type	FLASH
EEPROM Size	2K x 8
RAM Size	2.25K 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/at89c51cc03ca-rltum

Pin Name	Type	Description
VSS	GND	**Circuit ground**
TESTI	I	**Must be connected to VSS**
VCC		**Supply Voltage**
VAREF		Reference Voltage for ADC
VAGND		Reference Ground for ADC
P0.0:7	I/O	**Port 0:** Is an 8-bit open drain bi-directional I/O port. Port 0 pins that have 1's written to them float, and in this state can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external Program and Data Memory. In this application it uses strong internal pull-ups when emitting 1's. Port 0 also outputs the code Bytes during program validation. External pull-ups are required during program verification.
P1.0:7	I/O	**Port 1:** Is an 8-bit bi-directional I/O port with internal pull-ups. Port 1 pins can be used for digital input/output or as analog inputs for the Analog Digital Converter (ADC). Port 1 pins that have 1's written to them are pulled high by the internal pull-up transistors and can be used as inputs in this state. As inputs, Port 1 pins that are being pulled low externally will be the source of current (I_{IL}, see section "Electrical Characteristic") because of the internal pull-ups. Port 1 pins are assigned to be used as analog inputs via the ADCCF register (in this case the internal pull-ups are disconnected). As a secondary digital function, port 1 contains the Timer 2 external trigger and clock input; the PCA external clock input and the PCA module I/O. P1.0/AN0/T2 Analog input channel 0, External clock input for Timer/counter2. P1.1/AN1/T2EX Analog input channel 1, Trigger input for Timer/counter2. P1.2/AN2/ECI Analog input channel 2, PCA external clock input. P1.3/AN3/CEX0 Analog input channel 3, PCA module 0 Entry of input/PWM output. P1.4/AN4/CEX1 Analog input channel 4, PCA module 1 Entry of input/PWM output. P1.5/AN5/CEX2 Analog input channel 5, PCA module 2 Entry of input/PWM output. P1.6/AN6/CEX3 Analog input channel 6, PCA module 3 Entry of input/PWM output. P1.7/AN7/CEX4 Analog input channel 7, PCA module 4 Entry ot input/PWM output. Port 1 receives the low-order address byte during EPROM programming and program verification. It can drive CMOS inputs without external pull-ups.
P2.0:7	I/O	**Port 2:** Is an 8-bit bi-directional I/O port with internal pull-ups. Port 2 pins that have 1's written to them are pulled high by the internal pull-ups and can be used as inputs in this state. As inputs, Port 2 pins that are being pulled low externally will be a source of current (I_{IL}, see section "Electrical Characteristic") because of the internal pull-ups. Port 2 emits the high-order address byte during accesses to the external Program Memory and during accesses to external Data Memory that uses 16-bit addresses (MOVX @DPTR). In this application, it uses strong internal pull-ups when emitting 1's. During accesses to external Data Memory that use 8 bit addresses (MOVX @Ri), Port 2 transmits the contents of the P2 special function register. It also receives high-order addresses and control signals during program validation. It can drive CMOS inputs without external pull-ups.

Pin Name	Type	Description
P3.0:7	I/O	**Port 3:** Is an 8-bit bi-directional I/O port with internal pull-ups. Port 3 pins that have 1's written to them are pulled high by the internal pull-up transistors and can be used as inputs in this state. As inputs, Port 3 pins that are being pulled low externally will be a source of current (I_{IL}, see section "Electrical Characteristic") because of the internal pull-ups. The output latch corresponding to a secondary function must be programmed to one for that function to operate (except for TxD and $\overline{WR}$). The secondary functions are assigned to the pins of port 3 as follows: P3.0/RxD: Receiver data input (asynchronous) or data input/output (synchronous) of the serial interface P3.1/TxD: Transmitter data output (asynchronous) or clock output (synchronous) of the serial interface P3.2/$\overline{INT0}$: External interrupt 0 input/timer 0 gate control input P3.3/$\overline{INT1}$: External interrupt 1 input/timer 1 gate control input P3.4/T0: Timer 0 counter input P3.5/T1/$\overline{SS}$: Timer 1 counter input SPI Slave Select P3.6/$\overline{WR}$: External Data Memory write strobe; latches the data byte from port 0 into the external data memory P3.7/$\overline{RD}$: External Data Memory read strobe; Enables the external data memory. It can drive CMOS inputs without external pull-ups.
P4.0:4	I/O	**Port 4:** Is an 2-bit bi-directional I/O port with internal pull-ups. Port 4 pins that have 1's written to them are pulled high by the internal pull-ups and can be used as inputs in this state. As inputs, Port 4 pins that are being pulled low externally will be a source of current (IIL, on the datasheet) because of the internal pull-up transistor. The output latch corresponding to a secondary function RxDC must be programmed to one for that function to operate. The secondary functions are assigned to the two pins of port 4 as follows: P4.0/TxDC: Transmitter output of CAN controller P4.1/RxDC: Receiver input of CAN controller. P4.2/MISO: Master Input Slave Output of SPI controller P4.3/SCK: Serial Clock of SPI controller P4.4/MOSI: Master Ouput Slave Input of SPI controller It can drive CMOS inputs without external pull-ups.

Figure 3. Port 2 Structure



Notes:
1. Port 2 is precluded from use as general-purpose I/O Ports when as address/data bus drivers.
2. Port 2 internal strong pull-ups FET (P1 in FiGURE) assist the logic-one output for memory bus cycle.

When Port 0 and Port 2 are used for an external memory cycle, an internal control signal switches the output-driver input from the latch output to the internal address/data line.

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). When the destination operand is a Port or a Port bit, these instructions read the latch rather than the pin:

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 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

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
IEN0	A8h	Interrupt Enable Control 0	EA	EC	ET2	ES	ET1	EX1	ET0	EX0
IEN1	E8h	Interrupt Enable Control 1	–	–	–	–	ESPI	ETIM	EADC	ECAN
IPL0	B8h	Interrupt Priority Control Low 0	–	PPC	PT2	PS	PT1	PX1	PT0	PX0
IPH0	B7h	Interrupt Priority Control High 0	–	PPCH	PT2H	PSH	PT1H	PX1H	PT0H	PX0H
IPL1	F8h	Interrupt Priority Control Low 1	–	–	–	–	SPIL	POVRL	PADCL	PCANL
IPH1	F7h	Interrupt Priority Control High1	–	–	–	–	SPIH	POVRH	PADCH	PCANH

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
ADCON	F3h	ADC Control	–	PSIDLE	ADEN	ADEOC	ADSST	SCH2	SCH1	SCH0
ADCF	F6h	ADC Configuration	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
ADCLK	F2h	ADC Clock	–	–	–	PRS4	PRS3	PRS2	PRS1	PRS0
ADDH	F5h	ADC Data High byte	ADAT9	ADAT8	ADAT7	ADAT6	ADAT5	ADAT4	ADAT3	ADAT2
ADDL	F4h	ADC Data Low byte	–	–	–	–	–	–	ADAT1	ADAT0

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
CANGCON	ABh	CAN General Control	ABRQ	OVRQ	TTC	SYNCTTC	AUT–BAUD	TEST	ENA	GRES
CANGSTA	AAh	CAN General Status	–	OVFG	–	TBSY	RBSY	ENFG	BOFF	ERRP
CANGIT	9Bh	CAN General Interrupt	CANIT	–	OVRTIM	OVRBUF	SERG	CERG	FERG	AERG
CANBT1	B4h	CAN Bit Timing 1	–	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	–
CANBT2	B5h	CAN Bit Timing 2	–	SJW1	SJW0	–	PRS2	PRS1	PRS0	–
CANBT3	B6h	CAN Bit Timing 3	–	PHS22	PHS21	PHS20	PHS12	PHS11	PHS10	SMP
CANEN1	CEh	CAN Enable Channel byte 1	–	ENCH14	ENCH13	ENCH12	ENCH11	ENCH10	ENCH9	ENCH8
CANEN2	CFh	CAN Enable Channel byte 2	ENCH7	ENCH6	ENCH5	ENCH4	ENCH3	ENCH2	ENCH1	ENCH0
CANGIE	C1h	CAN General Interrupt Enable	–	–	ENRX	ENTX	ENERCH	ENBUF	ENERG	–
CANIE1	C2h	CAN Interrupt Enable Channel byte 1	–	IECH14	IECH13	IECH12	IECH11	IECH10	IECH9	IECH8

Registers

Table 2. CKCON0 Register

CKCON0 (S:8Fh)
Clock Control Register

7	6	5	4	3	2	1	0
CANX2	WDX2	PCAX2	SIX2	T2X2	T1X2	T0X2	X2

Bit Number	Bit Mnemonic	Description
7	CANX2	**CAN clock** [(1)] Clear to select 6 clock periods per peripheral clock cycle. Set to select 12 clock periods per peripheral clock cycle.
6	WDX2	**WatchDog clock** [(1)] Clear to select 6 clock periods per peripheral clock cycle. Set to select 12 clock periods per peripheral clock cycle.
5	PCAX2	**Programmable Counter Array clock** [(1)] Clear to select 6 clock periods per peripheral clock cycle. Set to select 12 clock periods per peripheral clock cycle.
4	SIX2	**Enhanced UART clock (MODE 0 and 2)** [(1)] Clear to select 6 clock periods per peripheral clock cycle. Set to select 12 clock periods per peripheral clock cycle.
3	T2X2	**Timer2 clock** [(1)] Clear to select 6 clock periods per peripheral clock cycle. Set to select 12 clock periods per peripheral clock cycle.
2	T1X2	**Timer1 clock** [(1)] Clear to select 6 clock periods per peripheral clock cycle. Set to select 12 clock periods per peripheral clock cycle.
1	T0X2	**Timer0 clock** [(1)] Clear to select 6 clock periods per peripheral clock cycle. Set to select 12 clock periods per peripheral clock cycle.
0	X2	**CPU clock** Clear to select 12 clock periods per machine cycle (STD mode) for CPU and all the peripherals. Set to select 6 clock periods per machine cycle (X2 mode) and to enable the individual peripherals "X2"bits.

Note: 1. This control bit is validated when the CPU clock bit X2 is set; when X2 is low, this bit has no effect.

Reset Value = 0000 0000b

Bit Number	Bit Mnemonic	Description
4-2	XRS1-0	**ERAM size:** Accessible size of the ERAM **XRS 2:0 ERAM size** 000 256 Bytes 001 512 Bytes 010 768 Bytes 011 1024 Bytes 100 1792 Bytes 101 2048 Bytes (default configuration after reset) 110 Reserved 111 Reserved
1	EXTRAM	**Internal/External RAM (00h - FFh)** access using MOVX @ Ri/@ DPTR 0 - Internal ERAM access using MOVX @ Ri/@ DPTR. 1 - External data memory access.
0	A0	**Disable/Enable ALE)** 0 - ALE is emitted at a constant rate of 1/6 the oscillator frequency (or 1/3 if X2 mode is used) 1 - ALE is active only during a MOVX or MOVC instruction.

Reset Value = X001 0100b
Not bit addressable

Table 8. AUXR1 Register

AUXR1 (S:A2h)
Auxiliary Control Register 1

7	6	5	4	3	2	1	0
-	-	ENBOOT	-	GF3	0	-	DPS

Bit Number	Bit Mnemonic	Description
7-6	-	**Reserved** The value read from these bits is indeterminate. Do not set these bits.
5	ENBOOT	**Enable Boot Flash** Set this bit for map the boot Flash between F800h -FFFFh Clear this bit for disable boot Flash.
4	-	**Reserved** The value read from this bit is indeterminate. Do not set this bit.
3	GF3	**General-purpose Flag 3**
2	0	**Always Zero** This bit is stuck to logic 0 to allow INC AUXR1 instruction without affecting GF3 flag.
1	-	**Reserved for Data Pointer Extension.**
0	DPS	**Data Pointer Select Bit** Set to select second dual data pointer: DPTR1. Clear to select first dual data pointer: DPTR0.

Reset Value = XXXX 00X0b

FSTA Register

Table 14. FSTA Register

FSTA Register (S:D3h)
Flash Status Register

7	6	5	4	3	2	1	0
						SEQERR	FLOAD

Bit Number	Bit Mnemonic	Description
7-2		**unusesd**
1	SEQERR	**Flash activation sequence error** Set by hardware when the flash activation sequence(MOV FCON 5X and MOV FCON AX)is not correct (See Error Repport Section) Clear by software or clear by hardware if the last activation sequence was correct (previous error are canceled)
0	FLOAD	**Flash Colums latch loaded** Set by hardware when the first data is loaded in the column latches. Clear by hardware when the activation sequence suceed (flash write sucess, or reset column latch success)

Reset Value= 0000 0000b

Mapping of the Memory Space

By default, the user space is accessed by MOVC A, @DPTR instruction for read only. The column latches space is made accessible by setting the FPS bit in FCON register. Writing is possible from 0000h to FFFFh, address bits 6 to 0 are used to select an address within a page while bits 15 to 7 are used to select the programming address of the page.
Setting FPS bit takes precedence on the EXTRAM bit in AUXR register.

The other memory spaces (user, extra row, hardware security) are made accessible in the code segment by programming bits FMOD0 and FMOD1 in FCON register in accordance with Table 15. A MOVC instruction is then used for reading these spaces.

Table 15. FM0 Blocks Select Bits

FMOD1	FMOD0	FM0 Adressable space
0	0	User (0000h-FFFFh)
0	1	Extra Row(FF80h-FFFFh)
1	0	Hardware Security Byte (0000h)
1	1	Column latches reset (note1)

Notes: 1. The column latches reset is a new option introduced in the AT89C51CC03, and is not available in T89C51CC01/2

Launching Programming

FPL3:0 bits in FCON register are used to secure the launch of programming. A specific sequence must be written in these bits to unlock the write protection and to launch the programming. This sequence is 5xh followed by Axh. Table 16 summarizes the memory spaces to program according to FMOD1:0 bits.

In-System Programming (ISP)

With the implementation of the User Space (FM0) and the Boot Space (FM1) in Flash technology the AT89C51CC03 allows the system engineer the development of applications with a very high level of flexibility. This flexibility is based on the possibility to alter the customer program at any stages of a product's life:

- Before assembly the 1st personalization of the product by programming in the FM0 and if needed also a customized Boot loader in the FM1.
  Atmel provide also a standard Boot loader by default UART or CAN.
- After assembling on the PCB in its final embedded position by serial mode via the CAN bus or UART.

This In-System Programming (ISP) allows code modification over the total lifetime of the product.

Besides the default Boot loader Atmel provide to the customer also all the needed Application-Programming-Interfaces (API) which are needed for the ISP. The API are located also in the Boot memory.

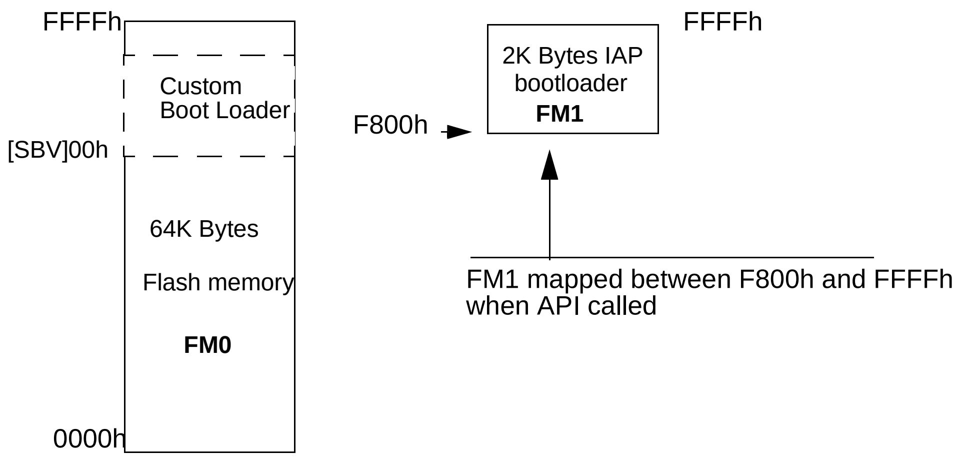
This allow the customer to have a full use of the 64-Kbyte user memory.

Flash Programming and Erasure

There are three methods of programming the Flash memory:

- The Atmel bootloader located in FM1 is activated by the application. Low level API routines (located in FM1)will be used to program FM0. The interface used for serial downloading to FM0 is the UART or the CAN. API can be called also by the user's bootloader located in FM0 at [SBV]00h.
- A further method exists in activating the Atmel boot loader by hardware activation.
- The FM0 can be programmed also by the parallel mode using a programmer.

Figure 29. Flash Memory Mapping



Boot Process

Software Boot Process Example

Many algorithms can be used for the software boot process. Before describing them,

The description of the different flags and Bytes is given below:

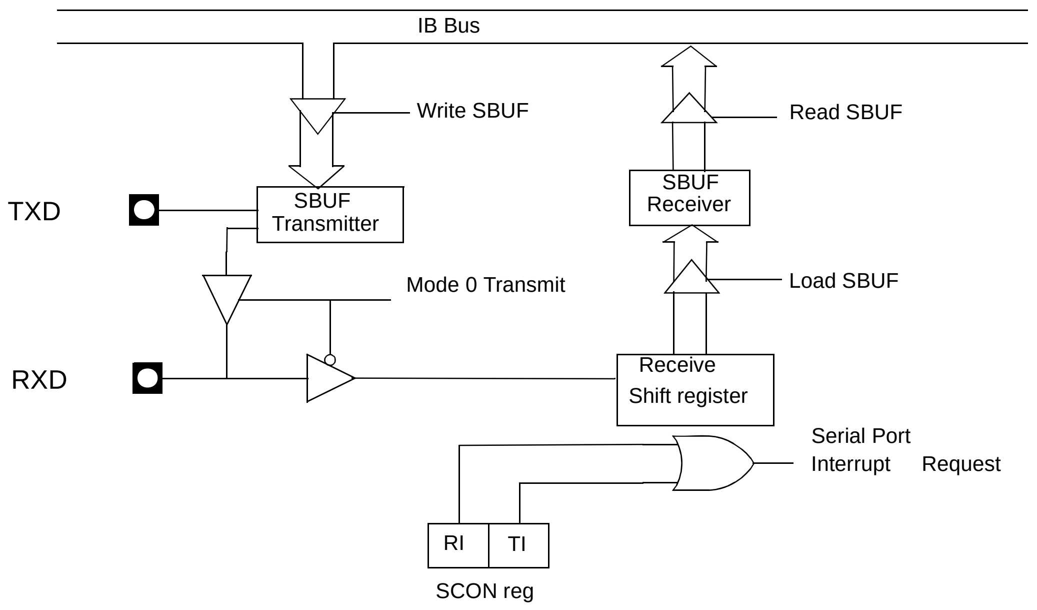
Serial I/O Port

The AT89C51CC03 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

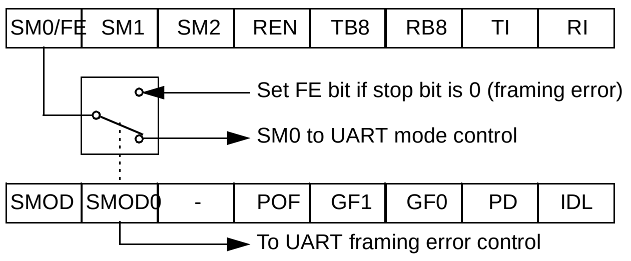
Figure 31. Serial I/O Port Block Diagram



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 32. 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

Programmable Clock-Output

In clock-out mode, timer 2 operates as a 50%-duty-cycle, programmable clock generator (See Figure 41). 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:

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

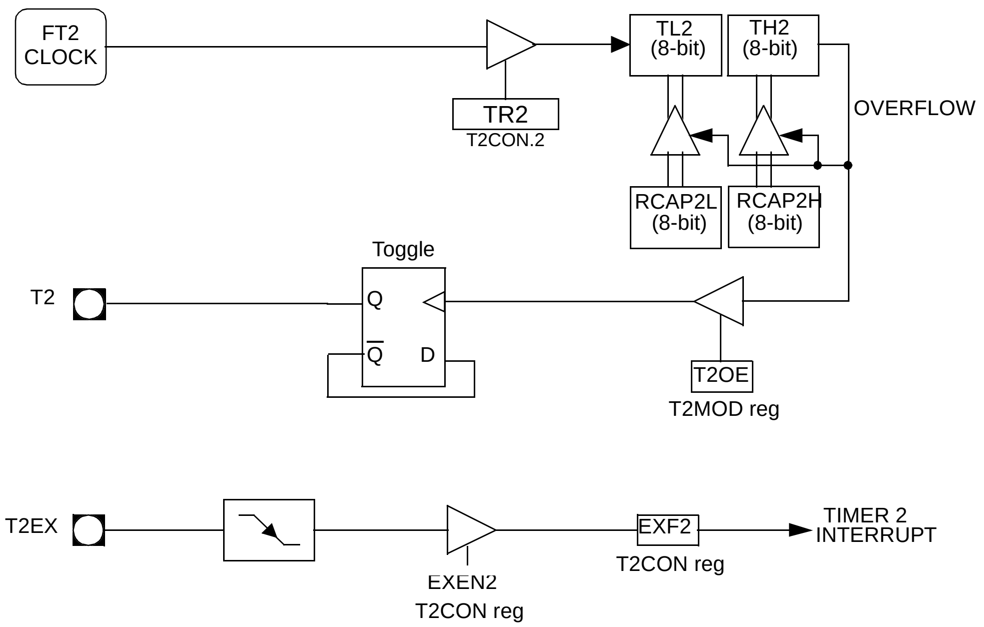
For a 16 MHz system clock in x1 mode, timer 2 has a programmable frequency range of 61 Hz ($F_{OSC}/2^{16)}$ to 4 MHz ($F_{OSC}/4$). The generated clock signal is brought out to T2 pin (P1.0).

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

- Set T2OE bit in T2MOD register.
- Clear C/$\overline{\text{T2}}$ 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 41. Clock-Out Mode



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 and 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/sec at 8 MHz[1] Crystal frequency in X2 mode.

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

CAN Protocol

The CAN protocol is an international standard defined in the ISO 11898 for high speed and ISO 11519-2 for low speed.

Principles

CAN is based on a broadcast communication mechanism. This broadcast communication is achieved by using a message oriented transmission protocol. These messages are identified by using a message identifier. Such a message identifier has to be unique within the whole network and it defines not only the content but also the priority of the message.

The priority at which a message is transmitted compared to another less urgent message is specified by the identifier of each message. The priorities are laid down during system design in the form of corresponding binary values and cannot be changed dynamically. The identifier with the lowest binary number has the highest priority.

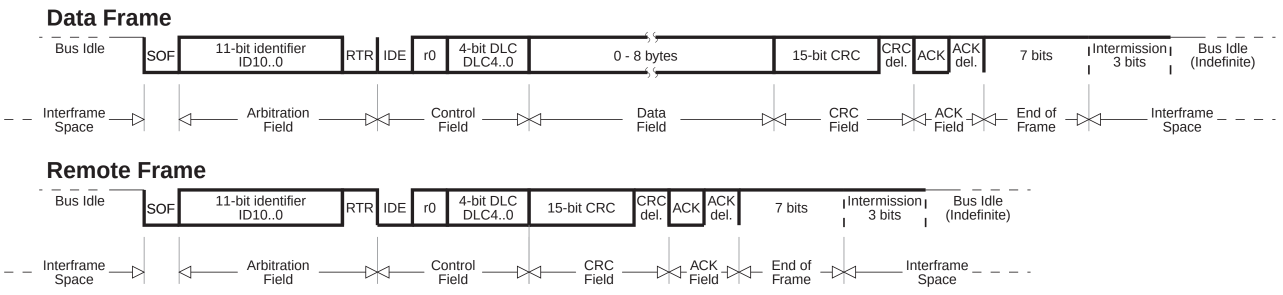
Bus access conflicts are resolved by bit-wise arbitration on the identifiers involved by each node observing the bus level bit for bit. This happens in accordance with the "wired and" mechanism, by which the dominant state overwrites the recessive state. The competition for bus allocation is lost by all nodes with recessive transmission and dominant observation. All the "losers" automatically become receivers of the message with the highest priority and do not re-attempt transmission until the bus is available again.

Message Formats

The CAN protocol supports two message frame formats, the only essential difference being in the length of the identifier. The CAN standard frame, also known as CAN 2.0 A, supports a length of 11 bits for the identifier, and the CAN extended frame, also known as CAN 2.0 B, supports a length of 29 bits for the identifier.

Can Standard Frame

Figure 43. CAN Standard Frames



A message in the CAN standard frame format begins with the "Start Of Frame (SOF)", this is followed by the "Arbitration field" which consist of the identifier and the "Remote Transmission Request (RTR)" bit used to distinguish between the data frame and the data request frame called remote frame. The following "Control field" contains the "IDentifier Extension (IDE)" bit and the "Data Length Code (DLC)" used to indicate the

Working on Message Objects

The Page message object register (CANPAGE) is used to select one of the 15 message objects. Then, message object Control (CANCONCH) and message object Status (CANSTCH) are available for this selected message object number in the corresponding SFRs. A single register (CANMSG) is used for the message. The mailbox pointer is managed by the Page message object register with an auto-incrementation at the end of each access. The range of this counter is 8.
Note that the maibox is a pure RAM, dedicated to one message object, without overlap. In most cases, it is not necessary to transfer the received message into the standard memory. The message to be transmitted can be built directly in the maibox. Most calculations or tests can be executed in the mailbox area which provide quicker access.

CAN Controller Management

In order to enable the CAN Controller correctly the following registers have to be initialized:

- General Control (CANGCON),
- Bit Timing (CANBT 1, 2 and 3),
- And for each page of 15 message objects
  - message object Control (CANCONCH),
  - message object Status (CANSTCH).

During operation, the CAN Enable message object registers 1 and 2 (CANEN 1 and 2) gives a fast overview of the message objects availability.

The CAN messages can be handled by interrupt or polling modes.

A message object can be configured as follows:

- Transmit message object,
- Receive message object,
- Receive buffer message object.
- Disable

This configuration is made in the CONCH1:2 field of the CANCONCH register (see Table 46).

When a message object is configured, the corresponding ENCH bit of CANEN 1 and 2 register is set.

Table 46. Configuration for CONCH1:2

CONCH 1	CONCH 2	Type of Message Object
0	0	Disable
0	1	Transmitter
1	0	Receiver
1	1	Receiver buffer

When a Transmitter or Receiver action of a message object is completed, the corresponding ENCH bit of the CANEN 1 and 2 register is cleared. In order to re-enable the message object, it is necessary to re-write the configuration in CANCONCH register.

Non-consecutive message objects can be used for all three types of message objects (Transmitter, Receiver and Receiver buffer),

- 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.

```
// Enable the CAN macro
 CANGCON = 02h
```

2. Configure message object 3 in reception to receive only standard (11-bit identifier) message 100h

```
// Select the message object 3
 CANPAGE = 30h
// Enable the interrupt on this message object
 CANIE2 = 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 12

```
// Select the message object 12
 CANPAGE = C0h
// Enable the interrupt on this message object
 CANIE1 = 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
```

Table 49. CANGSTA Register

CANGSTA (S:AAh Read Only)
CAN General Status Register

7	6	5	4	3	2	1	0
-	OVFG	-	TBSY	RBSY	ENFG	BOFF	ERRP

Bit Number	Bit Mnemonic	Description
7	-	**Reserved** The values read from this bit is indeterminate. Do not set this bit.
6	OVFG	**Overload Frame Flag** This status bit is set by the hardware as long as the produced overload frame is sent. This flag does not generate an interrupt
5	-	**Reserved** The values read from this bit is indeterminate. Do not set this bit.
4	TBSY	**Transmitter Busy** This status bit is set by the hardware as long as the CAN transmitter generates a frame (remote, data, overload or error frame) or an ack field. This bit is also active during an InterFrame Spacing if a frame must be sent. This flag does not generate an interrupt.
3	RBSY	**Receiver Busy** This status bit is set by the hardware as long as the CAN receiver acquires or monitors a frame. This flag does not generate an interrupt.
2	ENFG	**Enable On-chip CAN Controller Flag** Because an enable/disable command is not effective immediately, this status bit gives the true state of a chosen mode. This flag does not generate an interrupt.
1	BOFF	**Bus Off Mode** **see Figure 53**
0	ERRP	**Error Passive Mode** **see Figure 53**

Reset Value = x0x0 0000b

Table 50. CANGIT Register

CANGIT (S:9Bh)
CAN General Interrupt

7	6	5	4	3	2	1	0
CANIT	-	OVRTIM	OVRBUF	SERG	CERG	FERG	AERG

Bit Number	Bit Mnemonic	Description
7	CANIT	**General Interrupt Flag**[1] This status bit is the image of all the CAN controller interrupts sent to the interrupt controller. It can be used in the case of the polling method.
6	-	**Reserved** The values read from this bit is indeterminate. Do not set this bit.
5	OVRTIM	**Overrun CAN Timer** This status bit is set when the CAN timer switches 0xFFFF to 0x0000. If the bit ETIM in the IE1 register is set, an interrupt is generated. Clear this bit in order to reset the interrupt.
4	OVRBUF	**Overrun BUFFER** 0 - no interrupt. 1 - IT turned on This bit is set when the buffer is full. Bit resetable by user. see Figure 50.
3	SERG	**Stuff Error General** Detection of more than five consecutive bits with the same polarity. This flag can generate an interrupt. resetable by user.
2	CERG	**CRC Error General** The receiver performs a CRC check on each destuffed received message from the start of frame up to the data field. If this checking does not match with the destuffed CRC field, a CRC error is set. This flag can generate an interrupt. resetable by user.
1	FERG	**Form Error General** The form error results from one or more violations of the fixed form in the following bit fields: CRC delimiter acknowledgment delimiter end_of_frame This flag can generate an interrupt. resetable by user.
0	AERG	**Acknowledgment Error General** No detection of the dominant bit in the acknowledge slot. This flag can generate an interrupt. resetable by user.

Note: 1. This field is Read Only.

Reset Value = 0x00 0000b

Table 62. CANBT3 Register

CANBT3 (S:B6h)
CAN Bit Timing Registers 3

7	6	5	4	3	2	1	0
-	**PHS2 2**	**PHS2 1**	**PHS2 0**	**PHS1 2**	**PHS1 1**	**PHS1 0**	**SMP**

Bit Number	Bit Mnemonic	Description
7	-	**Reserved** The value read from this bit is indeterminate. Do not set this bit.
6-4	PHS2 2:0	**Phase Segment 2** This phase is used to compensate for phase edge errors. This segment can be shortened by the re-synchronization jump width. Tphs2 = Tscl x (PHS2[2..0] + 1) Phase segment 2 is the maximum of Phase segment 1 and the Information Processing Time (= 2TQ).
3-1	PHS1 2:0	**Phase Segment 1** This phase is used to compensate for phase edge errors. This segment can be lengthened by the re-synchronization jump width. Tphs1 = Tscl x (PHS1[2..0] + 1)
0	SMP	**Sample Type** 0 - once, at the sample point. 1 - three times, the threefold sampling of the bus is the sample point and twice over a distance of a 1/2 period of the Tscl. The result corresponds to the majority decision of the three values.

Note: 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 52.

No default value after reset.

Table 69. CANIDT4 Register for V2.0 part A

CANIDT4 for V2.0 part A (S:BFh)
CAN Identifier Tag Registers 4

7	6	5	4	3	2	1	0
-	-	-	-	-	RTRTAG	-	RB0TAG

Bit Number	Bit Mnemonic	Description
7-3	-	**Reserved** The values read from these bits are indeterminate. Do not set these bits.
2	RTRTAG	**Remote Transmission Request Tag Value**.
1	-	**Reserved** The values read from this bit are indeterminate. Do not set these bit.
0	RB0TAG	**Reserved Bit 0 Tag Value.**

No default value after reset.

Table 70. CANIDT4 Register for V2.0 part A

CANIDT1 for V2.0 part B (S:BCh)
CAN Identifier Tag Registers 1

7	6	5	4	3	2	1	0
IDT 28	IDT 27	IDT 26	IDT 25	IDT 24	IDT 23	IDT 22	IDT 21

Bit Number	Bit Mnemonic	Description
7-0	IDT28:21	**IDentifier Tag Value** See Figure 54.

No default value after reset.

Table 71. CANIDT2 Register for V2.0 part B

CANIDT2 for V2.0 part B (S:BDh)
CAN Identifier Tag Registers 2

7	6	5	4	3	2	1	0
IDT 20	IDT 19	IDT 18	IDT 17	IDT 16	IDT 15	IDT 14	IDT 13

Bit Number	Bit Mnemonic	Description
7-0	IDT20:13	**IDentifier Tag Value** See Figure 54.

No default value after reset.

In a Master configuration, the $\overline{SS}$ line can be used in conjunction with the MODF flag in the SPI Status register (SPSCR) to prevent multiple masters from driving MOSI and SCK (see Error conditions).

A high level on the $\overline{SS}$ pin puts the MISO line of a Slave SPI in a high-impedance state.

The $\overline{SS}$ pin could be used as a general-purpose if the following conditions are met:

- The device is configured as a Master and the SSDIS control bit in SPCON is set. This kind of configuration can be found when only one Master is driving the network and there is no way that the $\overline{SS}$ pin could be pulled low. Therefore, the MODF flag in the SPSCR will never be set[1].
- The Device is configured as a Slave with CPHA and SSDIS control bits set[2]. This kind of configuration can happen when the system includes one Master and one Slave only. Therefore, the device should always be selected and there is no reason that the Master uses the $\overline{SS}$ pin to select the communicating Slave device.

Note: 1. Clearing SSDIS control bit does not clear MODF.

2. Special care should be taken not to set SSDIS control bit when CPHA ='0' because in this mode, the $\overline{SS}$ is used to start the transmission.

Baud Rate

In Master mode, the baud rate can be selected from a baud rate generator which is controlled by three bits in the SPCON register: SPR2, SPR1 and SPR0.The Master clock is selected from one of seven clock rates resulting from the division of the internal clock by 4, 8, 16, 32, 64 or 128.

Table 90 gives the different clock rates selected by SPR2:SPR1:SPR0.

In Slave mode, the maximum baud rate allowed on the SCK input is limited to $F_{sys}/4$

Table 90. SPI Master Baud Rate Selection

SPR2	SPR1	SPR0	Clock Rate	Baud Rate Divisor (BD)
0	0	0	Don't Use	No BRG
0	0	1	$F_{CLK\ PERIPH}$ /4	4
0	1	0	$F_{CLK\ PERIPH}$/8	8
0	1	1	$F_{CLK\ PERIPH}$ /16	16
1	0	0	$F_{CLK\ PERIPH}$ /32	32
1	0	1	$F_{CLK\ PERIPH}$ /64	64
1	1	0	$F_{CLK\ PERIPH}$ /128	128
1	1	1	Don't Use	No BRG

External Data Memory Characteristics

Table 121. Symbol Description

Symbol	Parameter
T_{RLRH}	$\overline{RD}$ Pulse Width
T_{WLWH}	$\overline{WR}$ Pulse Width
T_{RLDV}	$\overline{RD}$ to Valid Data In
T_{RHDX}	Data Hold After $\overline{RD}$
T_{RHDZ}	Data Float After $\overline{RD}$
T_{LLDV}	ALE to Valid Data In
T_{AVDV}	Address to Valid Data In
T_{LLWL}	ALE to $\overline{WR}$ or $\overline{RD}$
T_{AVWL}	Address to $\overline{WR}$ or $\overline{RD}$
T_{QVWX}	Data Valid to $\overline{WR}$ Transition
T_{QVWH}	Data set-up to $\overline{WR}$ High
T_{WHQX}	Data Hold After $\overline{WR}$
T_{RLAZ}	$\overline{RD}$ Low to Address Float
T_{WHLH}	$\overline{RD}$ or $\overline{WR}$ High to ALE high

Table 122. AC Parameters for a Fix Clock (F=40MHz)

Symbol	Min	Max	Units
T_{RLRH}	130		ns
T_{WLWH}	130		ns
T_{RLDV}		100	ns
T_{RHDX}	0		ns
T_{RHDZ}		30	ns
T_{LLDV}		160	ns
T_{AVDV}		165	ns
T_{LLWL}	50	100	ns
T_{AVWL}	75		ns
T_{QVWX}	10		ns
T_{QVWH}	160		ns
T_{WHQX}	15		ns
T_{RLAZ}		0	ns
T_{WHLH}	10	40	ns