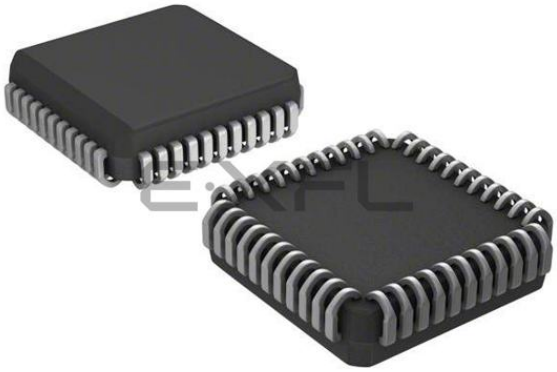


Welcome to E-XFL.COM

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-LCC (J-Lead)
Supplier Device Package	44-PLCC (16.6x16.6)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/at89c51cc01ua-slsum

Table 4. Timers SFRs (Continued)

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
T2CON	C8h	Timer/Counter 2 control	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2#	CP/RL2#
T2MOD	C9h	Timer/Counter 2 Mode	–	–	–	–	–	–	T2OE	DCEN
RCAP2H	CBh	Timer/Counter 2 Reload/Capture High byte	–	–	–	–	–	–	–	–
RCAP2L	CAh	Timer/Counter 2 Reload/Capture Low byte	–	–	–	–	–	–	–	–
WDTRST	A6h	Watchdog Timer Reset	–	–	–	–	–	–	–	–
WDTPRG	A7h	Watchdog Timer Program	–	–	–	–	–	S2	S1	S0

Table 5. Serial I/O Port SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
SCON	98h	Serial Control	FE/SM0	SM1	SM2	REN	TB8	RB8	TI	RI
SBUF	99h	Serial Data Buffer	–	–	–	–	–	–	–	–
SADEN	B9h	Slave Address Mask	–	–	–	–	–	–	–	–
SADDR	A9h	Slave Address	–	–	–	–	–	–	–	–

Table 6. PCA SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
CCON	D8h	PCA Timer/Counter Control	CF	CR	–	CCF4	CCF3	CCF2	CCF1	CCF0
CMOD	D9h	PCA Timer/Counter Mode	CIDL	WDTE	–	–	–	CPS1	CPS0	ECF
CL	E9h	PCA Timer/Counter Low byte	–	–	–	–	–	–	–	–
CH	F9h	PCA Timer/Counter High byte	–	–	–	–	–	–	–	–
CCAPM0	DAh	PCA Timer/Counter Mode 0	–	ECOM0	CAPP0	CAPN0	MAT0	TOG0	PWM0	ECCF0
CCAPM1	DBh	PCA Timer/Counter Mode 1		ECOM1	CAPP1	CAPN1	MAT1	TOG1	PWM1	ECCF1
CCAPM2	DCh	PCA Timer/Counter Mode 2		ECOM2	CAPP2	CAPN2	MAT2	TOG2	PWM2	ECCF2
CCAPM3	DDh	PCA Timer/Counter Mode 3		ECOM3	CAPP3	CAPN3	MAT3	TOG3	PWM3	ECCF3
CCAPM4	DEh	PCA Timer/Counter Mode 4		ECOM4	CAPP4	CAPN4	MAT4	TOG4	PWM4	ECCF4
CCAP0H	FAh	PCA Compare Capture Module 0 H	CCAP0H7	CCAP0H6	CCAP0H5	CCAP0H4	CCAP0H3	CCAP0H2	CCAP0H1	CCAP0H0
CCAP1H	FBh	PCA Compare Capture Module 1 H	CCAP1H7	CCAP1H6	CCAP1H5	CCAP1H4	CCAP1H3	CCAP1H2	CCAP1H1	CCAP1H0
CCAP2H	FCh	PCA Compare Capture Module 2 H	CCAP2H7	CCAP2H6	CCAP2H5	CCAP2H4	CCAP2H3	CCAP2H2	CCAP2H1	CCAP2H0
CCAP3H	FDh	PCA Compare Capture Module 3 H	CCAP3H7	CCAP3H6	CCAP3H5	CCAP3H4	CCAP3H3	CCAP3H2	CCAP3H1	CCAP3H0
CCAP4H	FEh	PCA Compare Capture Module 4 H	CCAP4H7	CCAP4H6	CCAP4H5	CCAP4H4	CCAP4H3	CCAP4H2	CCAP4H1	CCAP4H0

Table 9. CAN SFRs (Continued)

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
CANIDT1	BCh	CAN Identifier Tag byte 1(Part A) CAN Identifier Tag byte 1(PartB)	IDT10 IDT28	IDT9 IDT27	IDT8 IDT26	IDT7 IDT25	IDT6 IDT24	IDT5 IDT23	IDT4 IDT22	IDT3 IDT21
CANIDT2	BDh	CAN Identifier Tag byte 2 (PartA) CAN Identifier Tag byte 2 (PartB)	IDT2 IDT20	IDT1 IDT19	IDT0 IDT18	– IDT17	– IDT16	– IDT15	– IDT14	– IDT13
CANIDT3	BEh	CAN Identifier Tag byte 3(PartA) CAN Identifier Tag byte 3(PartB)	– IDT12	– IDT11	– IDT10	– IDT9	– IDT8	– IDT7	– IDT6	– IDT5
CANIDT4	BFh	CAN Identifier Tag byte 4(PartA) CAN Identifier Tag byte 4(PartB)	– IDT4	– IDT3	– IDT2	– IDT1	– IDT0	RTRTAG –	– RB1TAG	RB0TAG –
CANIDM1	C4h	CAN Identifier Mask byte 1(PartA) CAN Identifier Mask byte 1(PartB)	IDMSK10 IDMSK28	IDMSK9 IDMSK27	IDMSK8 IDMSK26	IDMSK7 IDMSK25	IDMSK6 IDMSK24	IDMSK5 IDMSK23	IDMSK4 IDMSK22	IDMSK3 IDMSK21
CANIDM2	C5h	CAN Identifier Mask byte 2(PartA) CAN Identifier Mask byte 2(PartB)	IDMSK2 IDMSK20	IDMSK1 IDMSK19	IDMSK0 IDMSK18	– IDMSK17	– IDMSK16	– IDMSK15	– IDMSK14	– IDMSK13
CANIDM3	C6h	CAN Identifier Mask byte 3(PartA) CAN Identifier Mask byte 3(PartB)	– IDMSK12	– IDMSK11	– IDMSK10	– IDMSK9	– IDMSK8	– IDMSK7	– IDMSK6	– IDMSK5
CANIDM4	C7h	CAN Identifier Mask byte 4(PartA) CAN Identifier Mask byte 4(PartB)	– IDMSK4	– IDMSK3	– IDMSK2	– IDMSK1	– IDMSK0	RTRMSK –	– 	IDEMSK –

Table 10. Other SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
PCON	87h	Power Control	SMOD1	SMOD0	–	POF	GF1	GF0	PD	IDL
AUXR	8Eh	Auxiliary Register 0	–	–	M0	–	XRS1	XRS2	EXTRAM	A0
AUXR1	A2h	Auxiliary Register 1	–	–	ENBOOT	–	GF3	0	–	DPS
CKCON	8Fh	Clock Control	CANX2	WDX2	PCAX2	SIX2	T2X2	T1X2	T0X2	X2

Register

Table 12. CKCON Register

CKCON (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	**Timer 2 clock** (1) Clear to select 6 clock periods per peripheral clock cycle. Set to select 12 clock periods per peripheral clock cycle.
2	T1X2	**Timer 1 clock** (1) Clear to select 6 clock periods per peripheral clock cycle. Set to select 12 clock periods per peripheral clock cycle.
1	T0X2	**Timer 0 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

Registers

Table 15. PCON Register

PCON (S:87h) – Power configuration Register

7	6	5	4	3	2	1	0
SMOD1	**SMOD0**	**-**	**POF**	**GF1**	**GF0**	**PD**	**IDL**

Bit Number	Bit Mnemonic	Description
7	SMOD1	**Serial port Mode bit 1** Set to select double baud rate in mode 1, 2 or 3
6	SMOD0	**Serial port Mode bit 0** Clear to select SM0 bit in SCON register. Set to select FE bit in SCON register.
5	-	**Reserved** The value read from this bit is indeterminate. Do not set this bit.
4	POF	**Power-Off Flag** Clear to recognize next reset type. Set by hardware when V_{cc} rises from 0 to its nominal voltage. Can also be set by software.
3	GF1	**General-purpose flag 1** One use is to indicate whether an interrupt occurred during normal operation or during Idle mode.
2	GF0	**General-purpose flag 0** One use is to indicate whether an interrupt occurred during normal operation or during Idle mode.
1	PD	**Power-down Mode bit** Cleared by hardware when an interrupt or reset occurs. Set to activate the Power-down mode. If IDL and PD are both set, PD takes precedence.
0	IDL	**Idle Mode bit** Cleared by hardware when an interrupt or reset occurs. Set to activate the Idle mode. If IDL and PD are both set, PD takes precedence.

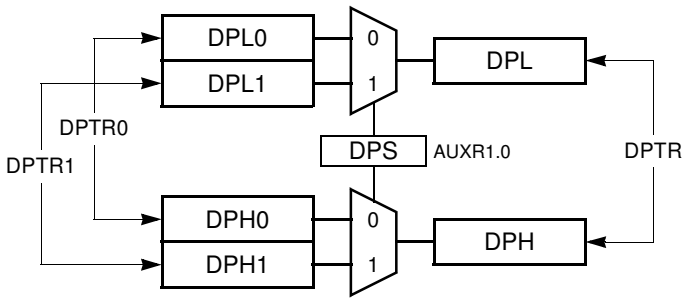
Reset Value = 00X1 0000b

Dual Data Pointer

Description

The T89C51CC01 implements a second data pointer for speeding up code execution and reducing code size in case of intensive usage of external memory accesses.
DPTR 0 and DPTR 1 are seen by the CPU as DPTR and are accessed using the SFR addresses 83h and 84h that are the DPH and DPL addresses. The DPS bit in AUXR1 register (see Figure 20) is used to select whether DPTR is the data pointer 0 or the data pointer 1 (see Figure 16).

Figure 16. Dual Data Pointer Implementation



Application

Software can take advantage of the additional data pointers to both increase speed and reduce code size, for example, block operations (copy, compare…) are well served by using one data pointer as a “source” pointer and the other one as a “destination” pointer. Hereafter is an example of block move implementation using the two pointers and coded in assembler. The latest C compiler takes also advantage of this feature by providing enhanced algorithm libraries.

The INC instruction is a short (2 Bytes) and fast (6 machine cycle) way to manipulate the DPS bit in the AUXR1 register. However, note that the INC instruction does not directly force the DPS bit to a particular state, but simply toggles it. In simple routines, such as the block move example, only the fact that DPS is toggled in the proper sequence matters, not its actual value. In other words, the block move routine works the same whether DPS is '0' or '1' on entry.

```
; ASCII block move using dual data pointers
; Modifies DPTR0, DPTR1, A and PSW
; Ends when encountering NULL character
; Note: DPS exits opposite to the entry state unless an extra INC AUXR1 is
added

AUXR1EQU0A2h

move:movDPTR,#SOURCE ; address of SOURCE
  incAUXR1 ; switch data pointers
  movDPTR,#DEST ; address of DEST
mv_loop:incAUXR1; switch data pointers
  movxA,@DPTR; get a byte from SOURCE
  incDPTR; increment SOURCE address
  incAUXR1; switch data pointers
  movx@DPTR,A; write the byte to DEST
  incDPTR; increment DEST address
  jnzmv_loop; check for NULL terminator
end_move:
```

Table 26. Programming Spaces

	Write to FCON				
	FPL3:0	**FPS**	**FMOD1**	**FMOD0**	**Operation**
User	5	X	0	0	No action
	A	X	0	0	Write the column latches in user space
Extra Row	5	X	0	1	No action
	A	X	0	1	Write the column latches in extra row space
Hardware Security Byte	5	X	1	0	No action
	A	X	1	0	Write the fuse bits space
Reserved	5	X	1	1	No action
	A	X	1	1	No action

Notes: 1. The sequence 5xh and Axh must be executing without instructions between them otherwise the programming is aborted.
2. Interrupts that may occur during programming time must be disabled to avoid any spurious exit of the programming mode.

Status of the Flash Memory

The bit FBUSY in FCON register is used to indicate the status of programming.

FBUSY is set when programming is in progress.

Selecting FM1

The bit ENBOOT in AUXR1 register is used to map FM1 from F800h to FFFFh.

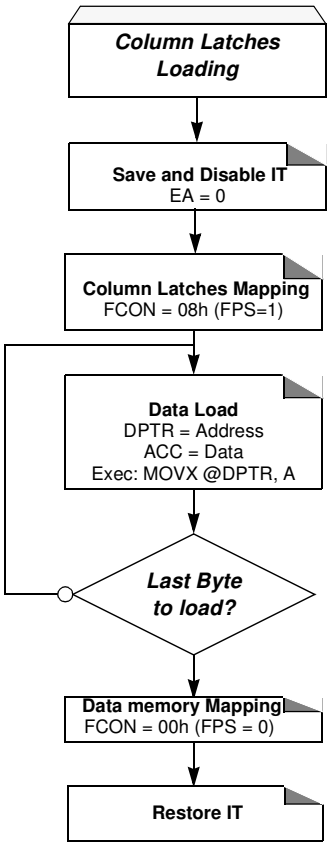
Loading the Column Latches

Any number of data from 1 Byte to 128 Bytes can be loaded in the column latches. This provides the capability to program the whole memory by byte, by page or by any number of Bytes in a page.
When programming is launched, an automatic erase of the locations loaded in the column latches is first performed, then programming is effectively done. Thus no page or block erase is needed and only the loaded data are programmed in the corresponding page.

The following procedure is used to load the column latches and is summarized in Figure 21:

- Save then disable interrupt and map the column latch space by setting FPS bit.
- Load the DPTR with the address to load.
- Load Accumulator register with the data to load.
- Execute the MOVX @DPTR, A instruction.
- If needed loop the three last instructions until the page is completely loaded.
- Unmap the column latch and Restore Interrupt

Figure 21. Column Latches Loading Procedure



Note: The last page address used when loading the column latch is the one used to select the page programming address.

Programming the Flash Spaces

User

The following procedure is used to program the User space and is summarized in Figure 22:

- Load up to one page of data in the column latches from address 0000h to 7FFFh.
- Save then disable the interrupts.
- Launch the programming by writing the data sequence 50h followed by A0h in FCON register (only from FM1).
  The end of the programming indicated by the FBUSY flag cleared.
- Restore the interrupts.

Extra Row

The following procedure is used to program the Extra Row space and is summarized in Figure 22:

- Load data in the column latches from address FF80h to FFFFh.
- Save then disable the interrupts.
- Launch the programming by writing the data sequence 52h followed by A2h in FCON register. This step of the procedure must be executed from FM1. The end of the programming indicated by the FBUSY flag cleared.
  The end of the programming indicated by the FBUSY flag cleared.
- Restore the interrupts.

Table 32. Read MOVC A, @DPTR

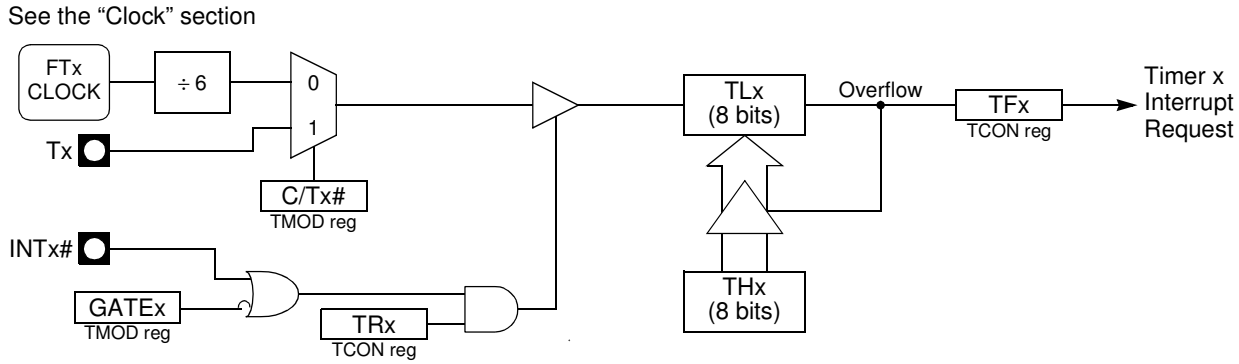
Code Execution	FCON Register			ENBOOT	DPTR	FM1	FM0	XROW	Hardware Byte	External Code
	FMOD1	FMOD0	FPS							
From FM0	0	0	X	0	0000h to 7FFFh		OK			
				1	0000h to 7FFFh		OK			
					F800h to FFFFh	Do not use this configuration				
	0	1	X	X	0000 to 007Fh See (1)			OK		
	1	0	X	X	X				OK	
	1	1	X	0	000h to 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

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

Mode 2 configures Timer 0 as an 8-bit Timer (TL0 register) that automatically reloads from TH0 register (see Figure 33). TL0 overflow sets TF0 flag in TCON register and reloads TL0 with the contents of TH0, which is preset by software. When the interrupt request is serviced, hardware clears TF0. The reload leaves TH0 unchanged. The next reload value may be changed at any time by writing it to TH0 register.

Figure 33. Timer/Counter x (x = 0 or 1) in Mode 2



Mode 3 (Two 8-bit Timers)

Mode 3 configures Timer 0 such that registers TL0 and TH0 operate as separate 8-bit Timers (see Figure 34). This mode is provided for applications requiring an additional 8-bit Timer or Counter. TL0 uses the Timer 0 control bits C/T0# and GATE0 in TMOD register, and TR0 and TF0 in TCON register in the normal manner. TH0 is locked into a Timer function (counting F_{PER} /6) and takes over use of the Timer 1 interrupt (TF1) and run control (TR1) bits. Thus, operation of Timer 1 is restricted when Timer 0 is in mode 3.

Figure 34. Timer/Counter 0 in Mode 3: Two 8-bit Counters

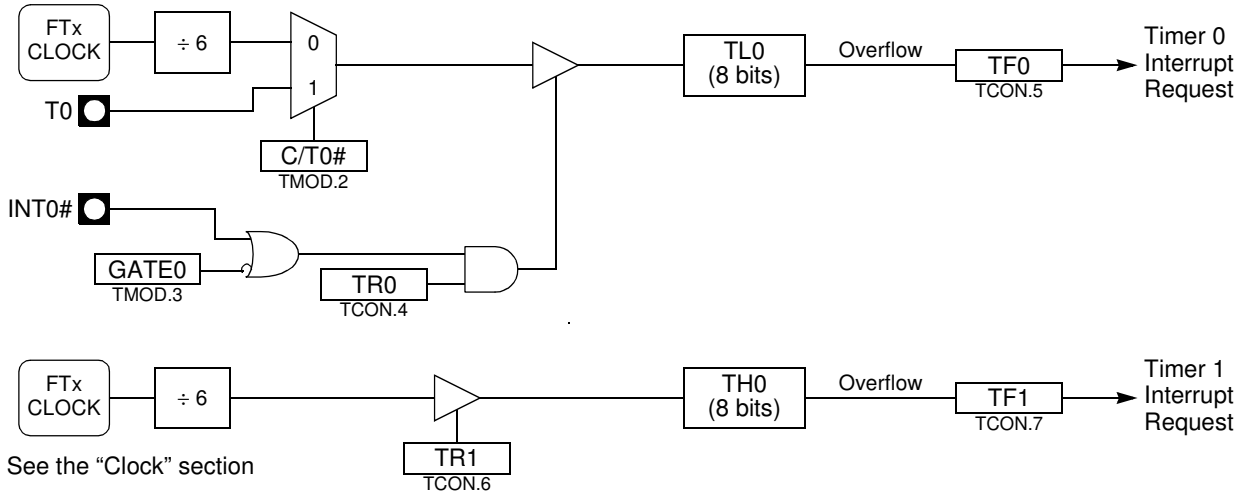


Table 42. TH0 Register

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

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

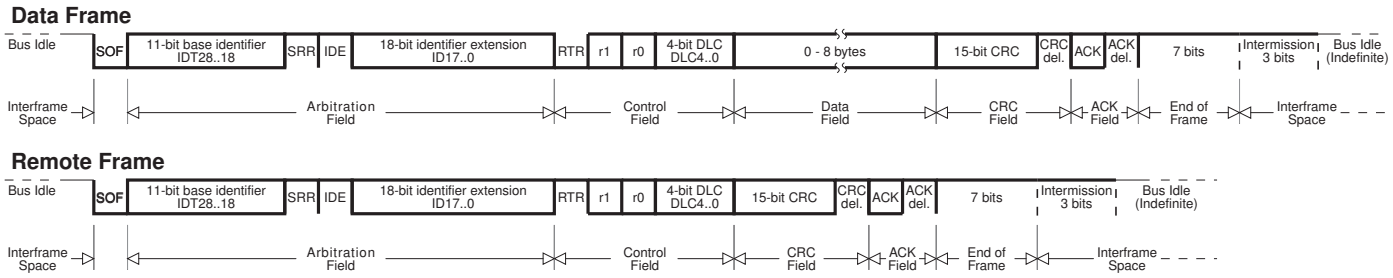
Reset Value = 1111 1111b

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

number of following data bytes in the "Data field". In a remote frame, the DLC contains the number of requested data bytes. The "Data field" that follows can hold up to 8 data bytes. The frame integrity is guaranteed by the following "Cyclic Redundant Check (CRC)" sum. The "ACKnowledge (ACK) field" compromises the ACK slot and the ACK delimiter. The bit in the ACK slot is sent as a recessive bit and is overwritten as a dominant bit by the receivers which have at this time received the data correctly. Correct messages are acknowledged by the receivers regardless of the result of the acceptance test. The end of the message is indicated by "End Of Frame (EOF)". The "Intermission Frame Space (IFS)" is the minimum number of bits separating consecutive messages. If there is no following bus access by any node, the bus remains idle.

CAN Extended Frame

Figure 40. CAN Extended Frames



A message in the CAN extended frame format is likely the same as a message in CAN standard frame format. The difference is the length of the identifier used. The identifier is made up of the existing 11-bit identifier (base identifier) and an 18-bit extension (identifier extension). The distinction between CAN standard frame format and CAN extended frame format is made by using the IDE bit which is transmitted as dominant in case of a frame in CAN standard frame format, and transmitted as recessive in the other case.

Format Co-existence

As the two formats have to co-exist on one bus, it is laid down which message has higher priority on the bus in the case of bus access collision with different formats and the same identifier / base identifier: The message in CAN standard frame format always has priority over the message in extended format.

There are three different types of CAN modules available:

- 2.0A - Considers 29 bit ID as an error
- 2.0B Passive - Ignores 29 bit ID messages
- 2.0B Active - Handles both 11 and 29 bit ID Messages

Bit Timing

To ensure correct sampling up to the last bit, a CAN node needs to re-synchronize throughout the entire frame. This is done at the beginning of each message with the falling edge SOF and on each recessive to dominant edge.

Bit Construction

One CAN bit time is specified as four non-overlapping time segments. Each segment is constructed from an integer multiple of the Time Quantum. The Time Quantum or TQ is the smallest discrete timing resolution used by a CAN node.

- ACK Errors
  As already mentioned frames received are acknowledged by all receivers through positive acknowledgement. If no acknowledgement is received by the transmitter of the message an ACK error is indicated.

Error at Bit Level

- Monitoring
  The ability of the transmitter to detect errors is based on the monitoring of bus signals. Each node which transmits also observes the bus level and thus detects differences between the bit sent and the bit received. This permits reliable detection of global errors and errors local to the transmitter.
- Bit Stuffing
  The coding of the individual bits is tested at bit level. The bit representation used by CAN is "Non Return to Zero (NRZ)" coding, which guarantees maximum efficiency in bit coding. The synchronization edges are generated by means of bit stuffing.

Error Signalling

If one or more errors are discovered by at least one node using the above mechanisms, the current transmission is aborted by sending an "error flag". This prevents other nodes accepting the message and thus ensures the consistency of data throughout the network. After transmission of an erroneous message that has been aborted, the sender automatically re-attempts transmission.

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).
- 15 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 T89C51CC01.

Fault Confinement

With respect to fault confinement, a unit may be in one of the three following status:

- error active
- error passive
- bus off

An error active unit takes part in bus communication and can send an active error frame when the CAN macro detects an error.

An error passive unit cannot send an active error frame. It takes part in bus communication, but when an error is detected, a passive error frame is sent. Also, after a transmission, an error passive unit will wait before initiating further transmission.

A bus off unit is not allowed to have any influence on the bus.

For fault confinement, two error counters (TEC and REC) are implemented.

See CAN Specification for details on Fault confinement.

Figure 49. Line Error Mode

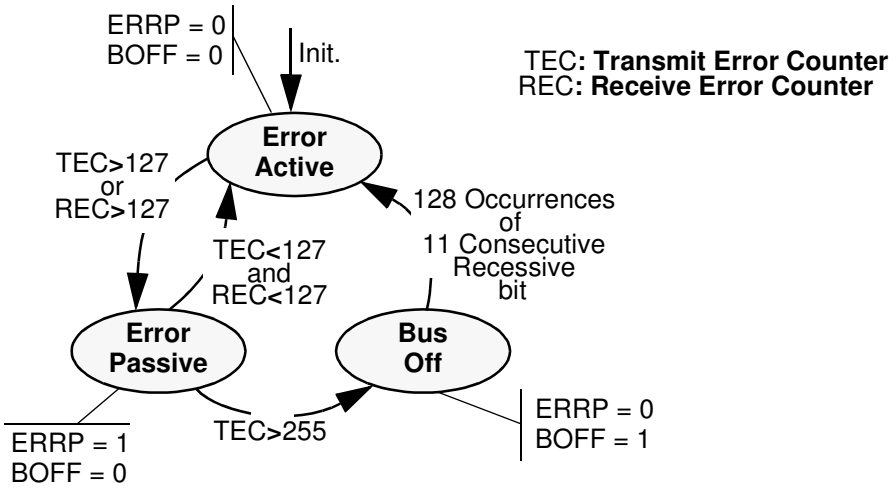


Table 65. CANEN2 Register

CANEN2 (S:CFh Read Only)
CAN Enable Message Object Registers 2

7	6	5	4	3	2	1	0
ENCH7	ENCH6	ENCH5	ENCH4	ENCH3	ENCH2	ENCH1	ENCH0

Bit Number	Bit Mnemonic	Description
7-0	ENCH7:0	**Enable Message Object** 0 - message object is disabled => the message object is free for a new emission or reception. 1 - message object is enabled. This bit is resetable by re-writing the CANCONCH of the corresponding message object.

Reset Value = 0000 0000b

Table 66. CANSIT1 Register

CANSIT1 (S:BAh Read Only)
CAN Status Interrupt Message Object Registers 1

7	6	5	4	3	2	1	0
-	SIT14	SIT13	SIT12	SIT11	SIT10	SIT9	SIT8

Bit Number	Bit Mnemonic	Description
7	-	**Reserved** The values read from this bit is indeterminate. Do not set this bit.
6-0	SIT14:8	**Status of Interrupt by Message Object** 0 - no interrupt. 1 - IT turned on. Reset when interrupt condition is cleared by user. SIT14:8 = 0b 0000 1001 -> IT's on message objects 11 and 8. see Figure 46.

Reset Value = x000 0000b

Table 76. 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 50.

No default value after reset.

Table 77. 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 50.
4-0	-	**Reserved** The values read from these bits are indeterminate. Do not set these bits.

No default value after reset.

Table 78. CANIDT3 Register for V2.0 part A

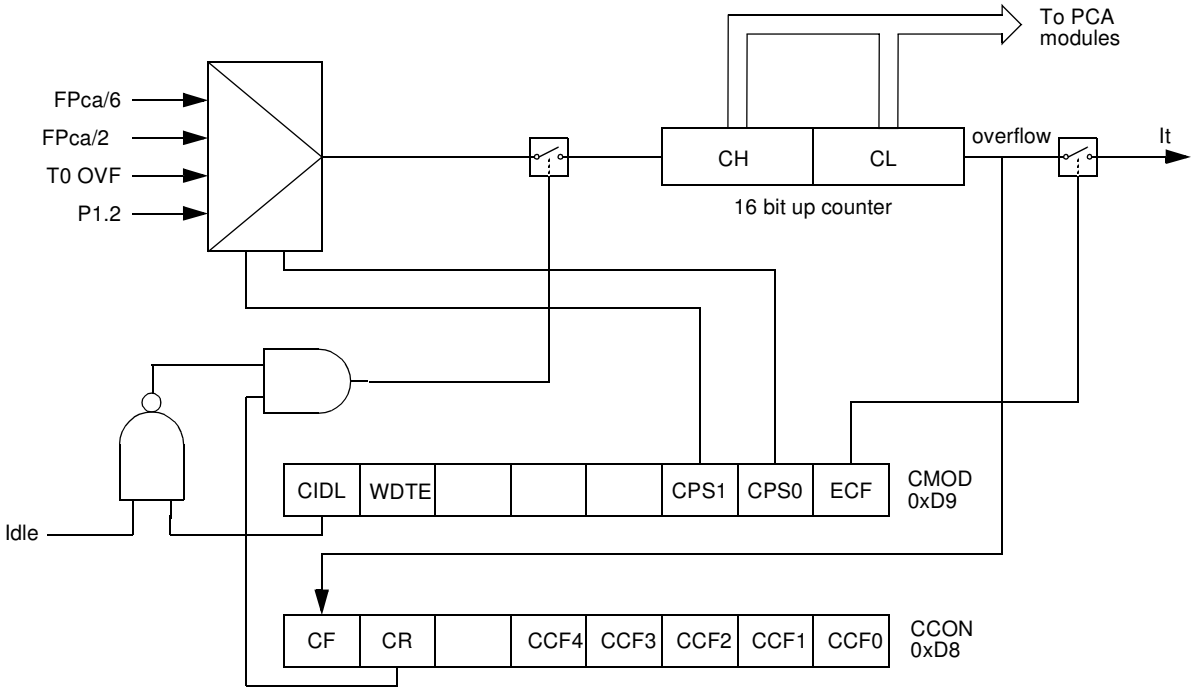
CANIDT3 for V2.0 part A (S:BEh)
CAN Identifier Tag Registers 3

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.

Figure 53. PCA Timer/Counter



The CMOD register includes three additional bits associated with the PCA.

- The CIDL bit which allows the PCA to stop during idle mode.
- The WDTE bit which enables or disables the Watchdog function on module 4.
- The ECF bit which when set causes an interrupt and the PCA overflow flag CF in CCON register to be set when the PCA timer overflows.

The CCON register contains the run control bit for the PCA and the flags for the PCA timer and each module.

- The CR bit must be set to run the PCA. The PCA is shut off by clearing this bit.
- The CF bit is set when the PCA counter overflows and an interrupt will be generated if the ECF bit in CMOD register is set. The CF bit can only be cleared by software.
- The CCF0:4 bits are the flags for the modules (CCF0 for module0...) and are set by hardware when either a match or a capture occurs. These flags also can be cleared by software.

PCA Modules

Each one of the five compare/capture modules has six possible functions. It can perform:

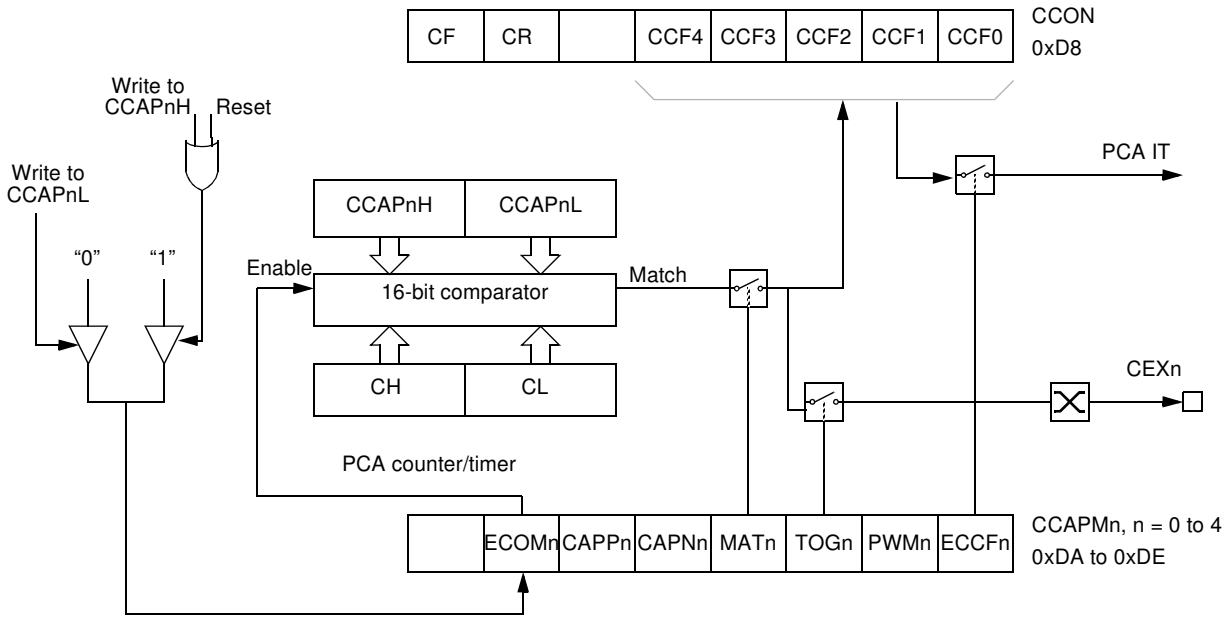
- 16-bit Capture, positive-edge triggered
- 16-bit Capture, negative-edge triggered
- 16-bit Capture, both positive and negative-edge triggered
- 16-bit Software Timer
- 16-bit High Speed Output
- 8-bit Pulse Width Modulator.

In addition module 4 can be used as a Watchdog Timer.

High Speed Output Mode

In this mode the CEX output (on port 1) associated with the PCA module will toggle each time a match occurs between the PCA counter and the module's capture registers. To activate this mode the TOG, MAT, and ECOM bits in the module's CCAPMn SFR must be set.

Figure 57. PCA High Speed Output Mode



Pulse Width Modulator Mode

All the PCA modules can be used as PWM outputs. The output frequency depends on the source for the PCA timer. All the modules will have the same output frequency because they all share the PCA timer. The duty cycle of each module is independently variable using the module's capture register CCAPLn. When the value of the PCA CL SFR is less than the value in the module's CCAPLn SFR the output will be low, when it is equal to or greater than it, the output will be high. When CL overflows from FF to 00, CCAPLn is reloaded with the value in CCAPHn. the allows the PWM to be updated without glitches. The PWM and ECOM bits in the module's CCAPMn register must be set to enable the PWM mode.

Registers

Table 108. ADCF Register

ADCF (S:F6h)
ADC Configuration

7	6	5	4	3	2	1	0
CH 7	**CH 6**	**CH 5**	**CH 4**	**CH 3**	**CH 2**	**CH 1**	**CH 0**

Bit Number	Bit Mnemonic	Description
7-0	CH 0:7	**Channel Configuration** Set to use P1.x as ADC input. Clear to use P1.x as standart I/O port.

Reset Value = 0000 0000b

Table 109. ADCON Register

ADCON (S:F3h)
ADC Control Register

7	6	5	4	3	2	1	0
-	**PSIDLE**	**ADEN**	**ADEOC**	**ADSST**	**SCH2**	**SCH1**	**SCH0**

Bit Number	Bit Mnemonic	Description
7	-	
6	PSIDLE	**Pseudo Idle Mode (Best Precision)** Set to put in idle mode during conversion Clear to convert without idle mode.
5	ADEN	**Enable/Standby Mode** Set to enable ADC Clear for Standby mode (power dissipation 1 uW).
4	ADEOC	**End Of Conversion** Set by hardware when ADC result is ready to be read. This flag can generate an interrupt. Must be cleared by software.
3	ADSST	**Start and Status** Set to start an A/D conversion. Cleared by hardware after completion of the conversion
2-0	SCH2:0	**Selection of Channel to Convert** see Table 107

Reset Value = X000 0000b

Ordering Information

Table 136. Possible Order Entries

Part Number	Boot Loader	Temperature Range	Package	Packing	Product Marking
T89C51CC01UA-7CTIM	OBSOLETE				
T89C51CC01UA-RLTIM					
T89C51CC01UA-SLSIM					
T89C51CC01CA-7CTIM					
T89C51CC01CA-RLTIM					
T89C51CC01CA-SLSIM					
AT89C51CC01UA-RLTUM	UART	Industrial & Green	VQFP44	Tray	89C51CC01UA-UM
AT89C51CC01UA-SLSUM	UART	Industrial & Green	PLCC44	Stick	89C51CC01UA-UM
AT89C51CC01CA-RLTUM	CAN	Industrial & Green	VQFP44	Tray	89C51CC01CA-UM
AT89C51CC01CA-SLSUM	CAN	Industrial & Green	PLCC44	Stick	89C51CC01CA-UM

- Factory default programming for T89C51CC01CA-xxxx is bootloader CAN and HSB=BBh
  - . X1 mode
  - . BLJB = 0; jump to Bootloader
  - . LB2 = 0 Security Level 4
- Factory default programming for T89C51CC01UA-xxxx is bootloader UART and HSB=BBh
  - . X1 mode
  - . BLJB = 0; jump to Bootloader
  - . LB2 = 0 Security Level 4

Note: Customer can change these modes by re-programming with a parallel programmer, this can be done by an Atmel distributor.