Description

The 80S32 is a high performance microcontroller. It is fully compatible with the well-known 80C32/80C52 device and the technology combines high execution speed with high level of integration. Some features have been added to the 80C32 device and the performance has been increased by a factor of 3 that means a processing power of about 2.3 MIPS at 15 MHz. This device is powerful enough to handle the requirements of embedded on board applications.

Specific Features

- 512 bytes on-chip memory ;
- Full 64 Kbytes addressing range for program and data, with expansion up to 16 Mbytes for data and 8 Mbytes for program;
- De-multiplexed Address/Data bus ;
- Memory protection for both internal and external memory;
- Program downloading and program execution from RAM capabilities;
- Fives serial interfaces:
  - One RS232 UART;
  - Four extra configurable USARTs (RS232, PacketWire and TTC-B-01 compatible);
  - two 64 bytes FIFO that can be associated to one of the extra USART and used as emission or reception buffer;
- Three 16-bit counters/timers with extended time count duration;
- Five external interrupts with two priority levels;
- A CRC calculation acceleration unit compatible with CCSDS TM and TC packets;

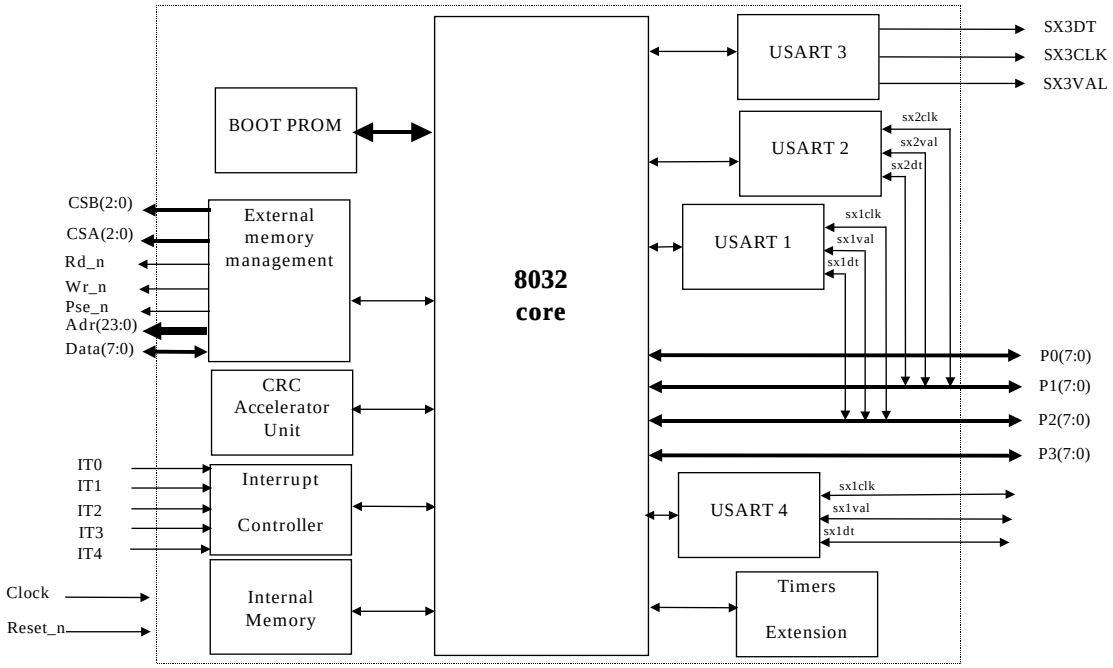


Table of contents

1. INTRODUCTION

1.1 Scope

This data sheet presents the technical information concerning the ADV 80S32 microcontroller which is an 8-bit microcontroller based on the well-known 8052/51 device and improved for space application.

The 80S32 microcontroller is 100% code compatible with the original 8052/51 and provides the following extra features:

- 256 bytes of additional on-chip memory;
- On-chip EDAC protecting external and internal memory;
- 4 extra USARTs supporting RS232, Packet Wire and TTC-B-01 protocols;
- Extended interrupt control capabilities with 5 instead of 2 external interrupts;
- CRC accelerator unit processing 8 bit calculation in 2 clock cycles;
- Extension of the time duration of the 3 counters;
- Program downloading capabilities via PacketWire interface;

This data sheet only contains the description of all the specific features of the 80S32 microcontroller and is not duplicating generic information on the 8052/51 that is available in public documentation. RD1 is the reference document that has been used during the development.

1.2 Reference documents

RD-1 'MCS51 MICROCONTROLLER FAMILY USER'S MANUAL' - INTEL -
(can be downloaded from http://developper.intel.com/design/mcs51/manuals).

RD-2 Packet Telecommand Decoder (PTD) Data Sheet, MA28140, GPS SOS Radiation Hard Handbook, July 1994,

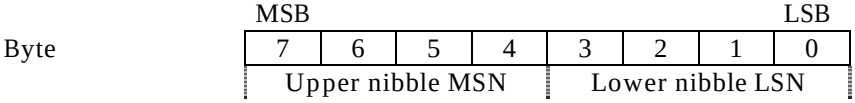
RD-3 Telecommand Decoder Specification, ESA PSS-04-151, Issue 1, September 1993

RD-4 Virtual Channel Assembler (VCA) Preliminary Data Sheet, MS12399, Mitel Semiconductor, January 1994

RD-5 Spacecraft data Handling Interface Standards - TTC-B-01, Issue 1, September 1979

1.3 Convention

The following convention to present byte data is used in this document:



2. SPECIAL FUNCTION REGISTERS

All the special function registers of the original 8051/52 are present in the 80S32 device and are repeated in §2.1. Extra SFRs have also been added for the implementation of the new functions and are presented in §2.2.

2.1 Original Special Function Registers

All the special function registers of the original 8051/52 are present in the ADV 80S32 devices. More detailed information on their use can be obtained from an 805x data sheet.

SFR	Function	Address	Reset Value
ACC	Accumulator	$E0_H$	00
B	B Register	$F0_H$	00
PSW	Program Status Word	$D0_H$	00
SP	Stack Pointer	81_H	07_H
DPL	Data Pointer low byte	82_H	00
DPH	Data Pointer high byte	83_H	00
P0	Port0	80_H	FF_H
P1	Port1	90_H	FF_H
P2	Port2	$A0_H$	FF_H
P3	Port3	$B0_H$	FF_H
IP	Interrupt Priority Control	$B8_H$	00
IE	Interrupt Enable Control	$A8_H$	00
TMOD	Timer/Counter Mode Control	89_H	00
TCON	Timer/Counter Control	88_H	00
T2CON	Timer/Counter 2 Control	$C8_H$	00
TH0	Timer/Counter 0 High Byte	$8C_H$	00
TL0	Timer/Counter 0 Low Byte	$8A_H$	00
TH1	Timer/Counter 1 High Byte	$8D_H$	00
TL1	Timer/Counter 1 Low Byte	$8B_H$	00
TH2	Timer/Counter 2 High Byte	CD_H	00
TL2	Timer/Counter 2 Low Byte	CC_H	00
RCAP2H	T/C 2 Capture Register High Byte	CB_H	00
RCAP2L	T/C 2 Capture Register Low Byte	CA_H	00
SCON	Serial Control	98_H	00
SBUF	Serial Data Buffer	99_H	00
PCON	Power Control	87_H	00

2.1.1 PSW: PROGRAM STATUS WORD REGISTER

Bit	Name	Description
7	CY	Carry flag
6	AC	Auxiliary Carry flag
5	F0	General purpose status flag.
4	RS1	Register bank select bit 1
3	RS0	Register bank select bit 0
2	OV	Overflow flag
1	-	User definable flag
0	P	Parity flag. Set when there are an odd number of bits in ACC.

RS1	RS0	Register Bank	Address
0	0	0	00_H-07_H
0	1	1	08_H-$0F_H$
1	0	2	10_H-17_H
1	1	3	18_H-$1F_H$

2.1.2 PCON: POWER CONTROL REGISTER

Bit	Name	Description
7	SMOD	Doubles the baud rate when set in serial mode 2 or when Timer1 is used to generate the baud rate in modes 1 or 3.
3	GF1	General-purpose flag bit.
2	GF0	General-purpose flag bit.
1	PD	Set to enter power down mode.
0	IDL	Set to enter idle mode.

NOTE: PD takes precedence over IDL.

2.1.3 IE: INTERRUPT ENABLE REGISTER

Bit	Name	Description
7	EA	Clear to globally disable all interrupt sources.
5	ET2	Set to enable Timer2 interrupt.
4	ES	Set to enable the Serial port interrupt
3	ET1	Set to enable Timer1 interrupt.
2	EX1	Set to enable External interrupt1.
1	ET0	Set to enable Timer0 interrupt.
0	EX0	Set to enable External interrupt0.

2.1.4 IP: INTERRUPT PRIORITY REGISTER

Bit	Name	Description
5	PT2	Timer2 interrupt priority level
4	PS	Serial Port interrupt priority level
3	PT1	Timer1 interrupt priority level
2	PX1	External interrupt 1 priority level
1	PT0	Timer0 interrupt priority level
0	PX0	External interrupt priority level

Priority levels are (from high to low):

IE0
TF0
IE1
TF1
RI or TI
TF2 or EXF2

2.1.5 TCON: TIMER/COUNTER CONTROL REGISTER

Bit	Name	Description
7	TF1	Timer1 overflow flag. Hardware controlled. Set when Timer/Counter1 overflows. Cleared when processor vectors to interrupt service routine.
6	TR1	When set, starts Timer/Counter1 running.
5	TF0	Timer0 overflow flag. Hardware controlled. Set when Timer/Counter0 overflows. Cleared when processor vectors to interrupt service routine.
4	TR0	When set, starts Timer/Counter0 running.
3	IE1	Set when External Interrupt1 detected. Cleared when interrupt is processed. Hardware controlled.
2	IT1	Set to specify External Interrupt1 as falling edge triggered. Clear to specify External Interrupt1 as low level triggered.
1	IE0	Set when External Interrupt0 detected. Cleared when interrupt is processed. Hardware controlled.
0	IT0	Set to specify External Interrupt0 as falling edge triggered. Clear to specify External Interrupt0 as low level triggered.

2.1.6 TMOD: TIMER/COUNTER MODE CONTROL REGISTER

Bit	Name	Description
7	GATE1	Set to specify Timer/Counter1 is gated when *Timer0Gate*=1.
6	CT1	Set to specify Timer/Counter1 as a counter (counting *Timer1*=1). Clear to specify input from internal clock.
5	T1M1	Timer/Counter1 mode select bit 1.
4	T1M0	Timer/Counter1 mode select bit 0.
3	GATE0	Set to specify Timer/Counter0 is gated when *Timer1Gate*=1.
2	CT0	Set to specify Timer/Counter0 as a counter (counting *Timer0*=1). Clear to specify input from internal clock.
1	T0M1	Timer/Counter0 mode select bit 1.
0	T0M0	Timer/Counter0 mode select bit 0.

T0M1	T0M0	Mode	Operation
0	0	0	13-bit timer/counter
0	1	1	16-bit timer/counter
1	0	2	8-bit auto-reload timer/counter
1	1	3	TL0 is an 8-bit timer/counter controlled by the standard Timer0 control bits. TH0 is an 8-bit timer controlled by Timer1 control bits. Timer 1 is stopped.

T1M1	T1M0	Mode	Operation
0	0	0	13-bit timer/counter
0	1	1	16-bit timer/counter
1	0	2	8-bit auto-reload timer/counter
1	1	3	Timer1 is stopped.

2.1.7 T2CON: TIMER/COUNTER 2 CONTROL REGISTER

Bit	Name	Description
7	TF2	Set by hardware when Timer2 overflows, cleared by software. Is not set when either RCLK=1 or TCLK=1.
6	EXF2	Timer2 External interrupt flag. Set whenever EXEN2=1 and a negative transition on *Timer2Capt* occurs. Causing a reload or capture, and a vector to the Timer2 interrupt service routine. Cleared by software.
5	RCLK	When set, the serial port uses Timer2 overflows for its receive clock, otherwise it uses Timer1 overflows.
4	TCLK	When set, the serial port uses Timer2 overflows for its transmit clock, otherwise it uses Timer1 overflows.
3	EXEN2	When set allows a capture or reload to occur because of a negative transition on *Timer2Capt*, if Timer2 is not being used to clock the serial port (i.e. RCLK=0 and TCLK=0).
2	TR2	Set to start Timer2 running.
1	CT2	Set to select Timer2 as falling edge triggered External Event Counter.
0	CPRL2	When set, Timer2 captures if EXEN2=1. When cleared, auto-reloads occur. If Timer2 is used to generate, the serial clock Timer2 is forced to auto-reload.

Mode	CPRL2	TCLK	RCLK	Comment
Capture	1	0	0	
Auto Reload	0	0	0	
Baud Generator	X	1	1	receive and transmit same Baud
Baud Generator	X	0	1	set receive Baud only
Baud Generator	X	1	0	set transmit Baud only

'X' denotes don't care.

2.1.8 SCON: SERIAL PORT CONTROL REGISTER

Bit	Name	Description
7	SM0	Serial port mode bit 0
6	SM1	Serial port mode bit 1
5	SM2	If set in serial modes 2 or 3, RI is only activated if the 9th received data bit (RB8) is 1. If set in serial mode 1, then RI is only activated if a valid stop bit is received.
4	REN	Set to enable serial reception.
3	TB8	In serial modes 2 and 3, the 9th data bit transmitted.
2	RB8	In serial modes 2 and 3, the 9th data bit received.
1	TI	Transmit interrupt flag, set at the beginning of the stop bit, or at the end of the 8th bit time in mode 0. Cleared by software.
0	RI	Receive interrupt flag, set halfway through the stop bit, or at the end of the 8th bit time in mode 0. Cleared by software.

SM0	SM1	Serial Mode	Description	Baud Rate
0	0	0	Shift Register	*Clock* / 12
0	1	1	8-bit UART	Variable
1	0	2	9-bit UART	*Clock* / 64 or *Clock* / 32
1	1	3	9-bit UART	Variable

2.2 Additional Special Function Registers

To support the additional functions of the 80S32 the following registers have been added to the SFR space at locations that are not used in the original 8052/51 device.

SYMBOL	DESCRIPTION	DIRECT ADDR (Hex)	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB							LSB	ACCESS
DTBANK	Data memory bank	84	8-bit word								R/W
PGBANK	Program memory bank	85	0	7-bit word							R/W
DPSEL	Data Pointer Select	92	-							DPSEL	R/W
SYSCON	XRAM configuration	93	-	-	-		-		XMAP		R/W
MPCON	Internal memory protection configuration	94					IPCON		XPCON		R/W
MPSTAT	Internal and external memory protection status	95	PG_ER_FLAG0		DT_ER_FLAG1		XMER		IMER		R/W
EMCON	External memory protection configuration	96	-	-	PMCON0		DMCON		EMA		R/W
CSCON	CS configuration	9E	-	-	CSPGCON0		-		CSDTCON		R/W
IXERAD	Internal memory error address (LSB)	9A	8-bit word								R/W
PDERADL	External (program/data) memory error address (LSB)	A1	8-bit word								R/W
PDERADH	External (program/data) memory error address (MSB)	A2	8-bit word								R/W
P0DIR	Port 0 direction control bits	A4	8-bit word								R/W
P1DIR	Port 1 direction control bits	A5	8-bit word								R/W
P2DIR	Port 2 direction control bits	A6	8-bit word								R/W
P3DIR	Port 3 direction control bits	A7	8-bit word								R/W
CRCIN	CRC calculation input	A9	8-bit word								R/W
CRCL	CRC calculation result and initialization register (LSB)	AA	8-bit word								R/W
CRCH	CRC calculation result and initialization register (MSB)	AB	8-bit word								R/W
IXEP1	Interrupt enable + priority register 1	AC	-		-		EEXM		EINM		R/W
IXEP2	Interrupt enable + priority register 2	AD	ESX1		ESX2		ESX3		ESX4		R/W
IXEP3	Interrupt enable + priority register 3	AE	-		EX4		EX3		EX2		R/W
TDIVCON	Timer input selection	B1	TDIVIN		T0IN		T1IN		T2IN		R/W
TDIVL	Timer input frequency division ratio (LSB)	B2	8-bit word								R/W
TDIVH	Timer input frequency division ratio (MSB)	B3	8-bit word								R/W
EXTBUS	Address bus extension configuratiobon	B5	EXTAD23	EXTAD22	EXTAD21	EXTAD20	EXTAD19	EXTAD18	EXTAD17	EXTAD16	R/W

SYMBOL	DESCRIPTION	DIRECT ADDR (Hex)	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB					LSB	ACCESS
WAITMEM	WAIT states configuration	**0C3**	**WSPA**					**WSDA**	R/W*
P1CON	Port 1 configuration bits	D8	8-bit word						R/W*
P2CON	Port 2 configuration bits	E8	8-bit word						R/W*
P3CON	Port 3 configuration bits	F8	8-bit word						R/W*
TXCON1	External interrupt control	BA	-		IE4		IE3	IE2	R/W
TXCON2	External interrupt control	BB	AL4		AL3		AL2	AL1	R/W
TXCON3	External interrupt control	BC	AL0		IT4		IT3	IT2	R/W
FIFOCON	FIFO configuration	C7	TFIFO				RFIFO		R/W
SX1CON1	Extra USART1 configuration 1	D2	MODE1	MODE0	ME2	ME1	ME0		R/W
SX1CON2	Extra USART1 configuration 2	D3			RB8/EOR/BOT		TI	RI	R/W
SX1BUFL	Extra USART1 receive/transmit register (LSB)	D4	8-bit word						R/W
SX1BUFH	Extra USART1 receive/transmit register (MSB)	D5	8-bit word						R/W
SX1FREQL	Extra USART1 frequency divider ratio (LSB)	D6	8-bit word						R/W
SX1FREQH	Extra USART1 frequency divider ratio (MSB)	D7	8-bit word						R/W
SX2CON1	Extra USART2 configuration 1	D9	MODE1	MODE0	ME2	ME1	ME0		R/W
SX2CON2	Extra USART2 configuration 2	DA			RB8/EOR/BOT		TI	RI	R/W
SX2BUFL	Extra USART2 receive/transmit register (LSB)	DB	8-bit word						R/W
SX2BUFH	Extra USART2 receive/transmit register (MSB)	DC	8-bit word						R/W
SX2FREQL	Extra USART2 frequency divider ratio (LSB)	DD	8-bit word						R/W
SX2FREQH	Extra USART2 frequency divider ratio (MSB)	DE	8-bit word						R/W
SX3CON1	Extra USART3 configuration 1	E2	MODE1	MODE0	ME2	ME1	ME0		R/W
SX3CON2	Extra USART3 configuration 2	E3			RB8/EOR/BOT		TI	RI	R/W
SX3BUFL	Extra USART3 receive/transmit register (LSB)	E4	8-bit word						R/W
SX3BUFH	Extra USART3 receive/transmit register (MSB)	E5	8-bit word						R/W
SX3FREQL	Extra USART3 frequency divider ratio (LSB)	E6	8-bit word						R/W
SX3FREQH	Extra USART3 frequency divider ratio (MSB)	E7	8-bit word						R/W
SX4CON1	Extra USART4 configuration 1	E9	MODE1	MODE0	ME2	ME1	ME0		R/W
SX4CON2	Extra USART4 configuration 2	EA			RB8/EOR/BOT		TI	RI	R/W
SX4BUFL	Extra USART4	EB	8-bit word						R/W

SYMBOL	DESCRIPTION	DIRECT ADDR (Hex)	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB LSB	ACCESS
	receive/transmit register (LSB)			
SX4BUFH	Extra USART4 receive/transmit register (MSB)	EC	8-bit word	R/W
SX4FREQL	Extra USART4 frequency divider ratio (LSB)	ED	8-bit word	R/W
SX4FREQH	Extra USART4 frequency divider ratio (MSB)	EE	8-bit word	R/W

* Bit addressable

The 80S32 implements two data pointers. At any time only one of them is selected and can be use as the DPTR register. DPSEL SFR is the data pointer selection register. It works in a similar way to the Philips and Siemens device, e.g. the least significant bit is used to determine which DPTR is selected. The fastest way to swap between data pointer is to use the INC DPSEL instruction. The double DPTR implementation enables faster block data transfers in the external memory, the first register being loaded with the source address and the second one with the destination address. The transfer is then a loop that reads data from the source address, increments DPTR, changes DPTR selection, writes data to the destination address, increments DPTR and changes DPTR selection. DPSEL also acts on the DTBANK register (cf. 3.4.5).

3. MEMORY ORGANIZATION

3.1 Introduction

All 8052/51 devices have separate address spaces for program and data memory.

The program memory space is external and can only be read and not written. Up to 64KBytes of the program memory can be accessed by using the PSEN (Program store enable) signal as a read strobe signal.

The data memory space occupies a separate address space from the program memory. The lowest 256 bytes of the data memory space (the internal block) are on chip while up to 64 Kbytes of the external data memory can be accessed for read operation by asserting the Rd_n signal or write operation by asserting the Wr_n signal.

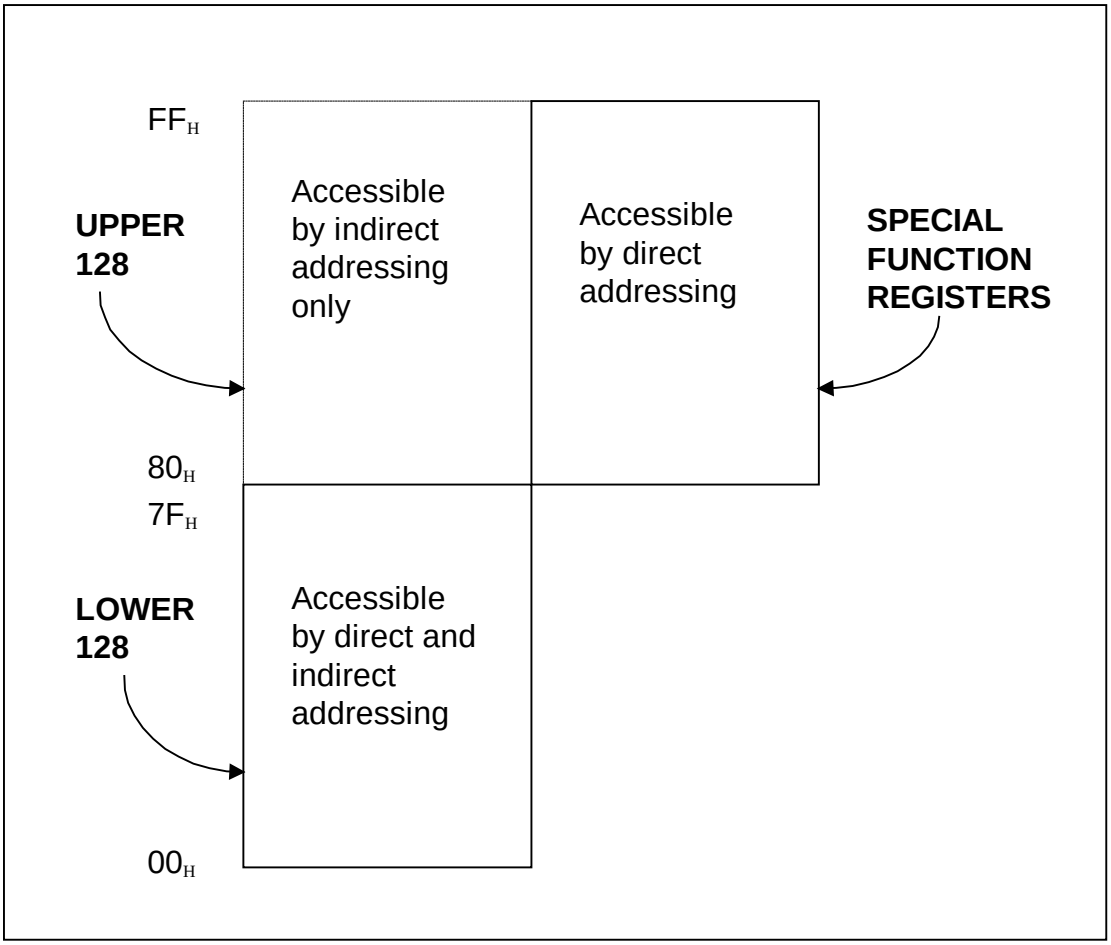
The 80S32 device is fully compatible with this organization and adds the possibilities:

- to download a program in RAM and to execute it from this RAM,
- to extend the data memory space up to 16 Mbytes accessible by page of 64 Kbytes,
- to extend the program memory space up to 8 Mbytes arranged in page of 32 Kbytes,
- to share one RAM device between program and data,
- to protect with an EDAC the accesses to the program and data memory spaces,
- to remap addresses 0000_H to $00FF_H$ of the external memory to an on chip 256 bytes RAM block, which is separated from the internal block. This eXtra block can be used in small applications to avoid the use of an external memory.

In addition the 80S32 microcontroller implements a CS generator unit that will avoid the need to add an external decoder in many cases.

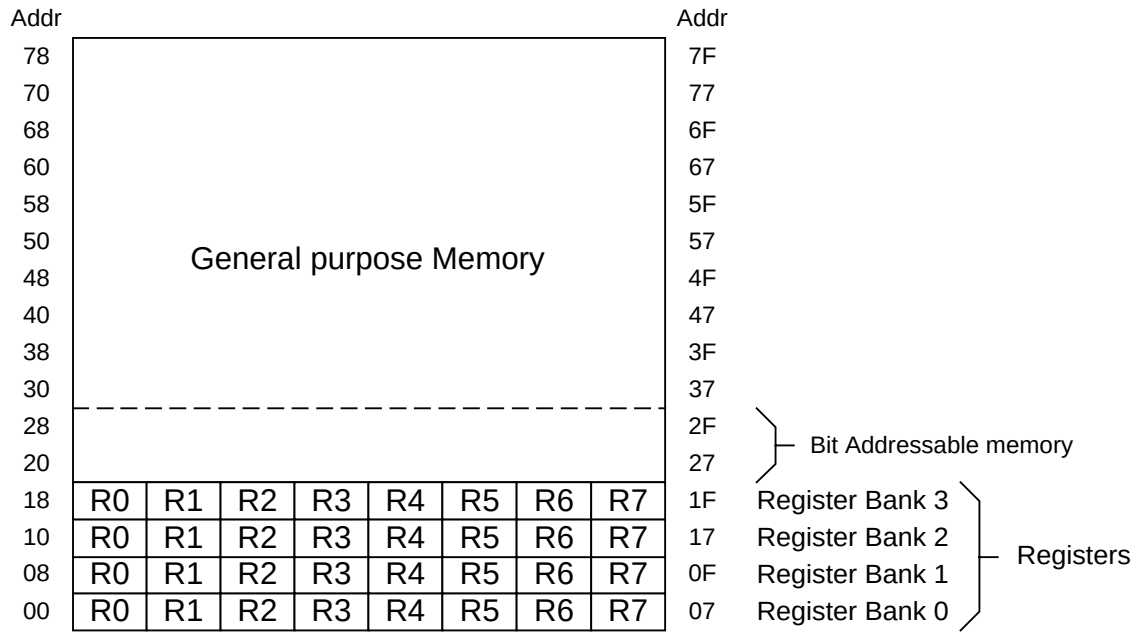
3.2 Internal Memory

The original 8052/51 has 256 bytes of register RAM and a number of SFRs that are collectively known as internal data memory. The internal data memory is divided into three blocks, which are referred to as the Lower 128, Upper 128 and SFR space. Although the internal data memory address space is only 8-bits wide, it can accommodate more than 256 registers by processing differently direct access and indirect access instructions. Direct access instructions with addresses higher than $7F_H$ always access the SFR space, while indirect accesses with addresses higher than $7F_H$ access the upper 128 space.



All the SFR have been presented in the beginning of the document (cf. § 2). In the rest of the document the term internal memory will apply to the upper 128 and lower 128 address spaces.

3.2.1 Lower 128 Bytes: Address 00_H - $7F_H$



The structure of the lower 128 bytes space is presented in the figure here-above:

The lowest 32 bytes of the space are used for the Register Banks. The ADV 80S32 implements four Register Banks. Only one Register Bank can be used at a time and the selection of Register Bank is controlled by bits RS1 and RS0 in the Processor Status Word SFR (PSW).

Addresses 20_H to $2F_H$ implement a general-purpose memory that is bit addressable. The address of a bit is defined by its position in this area. Bit address 0 is located in bit position 0 (least significant position) of the byte address 20_H, bit address 1 is located in bit position 1 of the byte address 20_H, and so on up to bit $7F_H$ which is located in bit position 7 (most significant bit) of the byte address $2F_H$. Bit manipulation instruction are presented in §12.5 (more details will be found in RD-1).

Addresses 30_H to $7F_H$ is a general-purpose memory that can be accessed either by direct and indirect addressing instructions.

3.2.2 Upper 128 Bytes: Address 80_H - FF_H

Addresses 80_H to FF_H form a general purpose memory that can only be accessed by indirect addressing instructions.

3.2.3 Internal Memory protection

The 80S32 has the possibility to protect the internal memory with an EDAC (cf. §3.6).

The EDAC function is enabled by setting the IPCON bit in the MPCON Special Function Register (cf. Figure 3-14). When the internal memory protection is enabled, memory locations 00_H to $7F_H$ are used to store data while memory location 80_H to FF_H are used to store protection code. Addresses 80_H to FF_H are in this case not accessible by software and each access to this part of the memory is re-directed to the requested address modulo 80_H. For example, an access to address $0AB_H$ is re-directed to address $2B_H$.

When enabled, the internal memory protection is transparent to the user. When data is written to an address X, its protection code is calculated by the EDAC and written at address X + 80_H. When data is read from address X, its protection code is read from address X+ 80_H and is used to correct the data in real time before being used by the 80S32.

The detection of a recoverable error or a unrecoverable error is flagged in the IMER bits of the MPSTAT register (cf. Figure 3-12) and an interruption is generated if previously enabled (cf. Figure 6-1). The address of the corrupted data is then available in the IXERAD register and has, in the case of a recoverable error, to be used to read and write back the corrupted data in order to refresh it. After error processing the IMER bits shall be reset by software (cf. Figure 3-12).

Enabling the EDAC does not slow down the microprocessor accesses and thus does not degrade the microcontroller performance.

3.3 eXtended RAM (XRAM)

The XRAM extends the internal memory space of 256 bytes but is mapped into the external data memory space at addresses 0000_H to $00FF_H$ and can therefore only be accessed by using MOVX instructions (cf. RD1). The XRAM can only be used to store data and cannot contain instruction codes.

The use of the XRAM is enabled via the XMAP bit in the SYSCON register (cf. Figure 3-13). When this is done all « MOVX @DPTR,A » and « MOVX A,@DPTR » instructions when 0000H < DPTR < 00FFH and DTBANK = "00", and all « MOVX @Ri,A » and « MOVX A,@Ri » instructions addresses the XRAM instead of the external memory. As a consequence, access to the external memory at address between 0000_H and $00FF_H$ in page 0 is not possible when the XRAM is enabled. Considering that XRAM cannot contain instruction code, the XRAM shall always be disabled during downloading of program instructions with an address between 0000_H and $00FF_H$.

The XRAM can also be protected by EDAC similarly to the internal memory. The EDAC is switched on by setting the XPCON bits in the MPCON register (cf. Figure 3-14). When the XRAM protection is enabled, memory location 00_H to $7F_H$ are used to store data while memory location 80_H to FF_H are used to store protection code.

Addresses 80_H to FF_H are not accessible by software and each access to this part of the memory is re-directed to the requested address modulo 80_H. For example, an access to address AB_H is re-directed to address $2B_H$.

When enabled, the XRAM protection is transparent to the user. When data is written to an address X, its protection code is calculated by the EDAC and written to address X + 80_H. When data is read from address X, its protection code is read from address X+ 80_H and is used to correct the data in real time before being used by the 80S32 core.

The detection of a recoverable error or a unrecoverable error is flagged in the IMER bits of the MPSTAT register (cf. Figure 3-12) and an interruption is generated if previously enabled (cf. §6). The address of the corrupted data is then available in the IXERAD register and has, in the case of a recoverable error, to be used to read and write back the corrupted data in order to refresh it. After error processing the IMER shall be reset by software (cf. Figure 3-12).

Enabling the EDAC does not slow down the microprocessor accesses and thus does not degrade the microcontroller performance.

For programs which do not need much data storage capacity, the use of the XRAM can avoid the use of an external RAM.

3.4 External memory

3.4.1 Introduction

As presented in § 3.1, the 80S32 can access two different memory spaces by means of different control signals. PSE_n is used as the read strobe signal for the program memory space while Rd_n and Wr_n are used as the read and write strobe signal of the data memory space (cf. RD 1). In the original 8052/51 PSE_n is always used for accessing the program memory space while Rd_n and Wr_n are always used for accessing the data memory space. We will call this configuration the « ROM program execution».

The 80S32 adds a new possibility that is to use Rd_n instead of Pse_n to access program memory. We will call this configuration the « RAM program execution».

After reset, the 80S32 is in the « ROM program execution» mode and can execute a program from an external non-volatile memory (EA = '0') or the internal bootstrap program (EA = '1').

The configuration can then be changed by software at any time via the EMP bit in the EMCON register (cf. Figure 3-11). Each time the configuration is changed, the microcontroller is reset (except the EMCON register) which enables to restart the program execution from address 0 of the RAM when changing from the « ROM program execution» mode to the « RAM program execution» mode or from address 0 of the ROM when changing from the « RAM program execution» mode to the « ROM program execution» mode.

For external memory accesses, the 80S32 uses an 8 bit bi-directional data bus Data(7:0), a 16 bit address bus Add(15:0) and up to 8 bits ExtAd16 to ExtAd23 for extending the address bus up to 24 bits. ExtAd23 has different meaning depending on the fact that the memory protection is enabled or disabled.

When in the « ROM program execution» mode, the differentiation between program and data accesses is performed by the assertion of the PSE_n or the Rd_n signal.

In the « RAM Program Execution» all accesses are performed with the Rd_n signal and the Pse_n signal is always de-asserted. In this mode, the program and data can be located in the same memory space, generally the same device, or in separated memory spaces. These two sub-configurations are commanded via the EMA bits in the EMCON register (cf. Figure 3-11). In fact, these sub-configurations only impact the on-chip address decoder unit behavior (cf. §3.4.6). When program and data are in separated spaces, for a same address value the differentiation is performed by the CS signal that is asserted. If a CSA signal is asserted then the access addresses the program memory space, while if a CSB signal is asserted then the access addresses the data memory space.

When program and data are in the same memory space then only CSA signals are used both for program and data accesses. In this case the developer can arrange the memory space as he wants provided that:

- the first program instruction shall be located at address 0 ;
- the first instructions of vectoring programs 0 to 12 shall be located at address x*8+3 where x represents the number of the interrupt ;
- if more than 8 Mbytes of RAM are implemented then program data shall not be located at an address higher than $7FFFFF_H$.

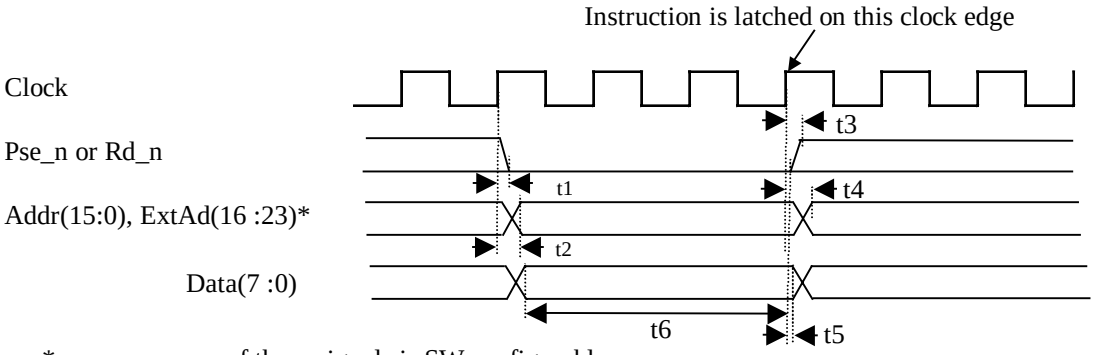
Provided that all these constraints are respected, the distribution of program and data is user defined.

3.4.2 External memory access

The external memory can be protected by an EDAC. The protection is selectively switched on or off for program and data accesses via the PMCON and DMCON bits in the EMCON register (Figure 3-11).

The figures hereafter present how accesses to the program memory are handled in the case where memory protection is disabled (Figure 3-1) and enabled (Figure 3-2). In the two cases, the access is performed in 3 clock cycles minimum but can be longer if the microcontroller is executing an instruction that need more than 3 clock cycles to be processed. When the memory protection is on the ExtAd23 signal is used to indicate if the microcontroller accesses the instruction (ExtAd23 = '0') or the protection code (ExtAd23 = '1'). ExtAd23 is at logic level '0' for 1.5 clock cycle at the beginning of the access and then at logic level '1' for the rest of the access. The data is latched in an intermediate register on the same falling clock edge which sets the ExtAd23 signal to '1' and is corrected in real time with the code present on the bus in the second part of the access before it is used. When the memory protection is off ExtAd23 makes the 24th bit of the address bus.

To speed-up the access, the Pse_n or the Rd_n signal is permanently asserted for program accesses as long as no external memory data access is needed.



* use or non-use of these signals is SW configurable

Figure 3-1: Program access with memory protection off

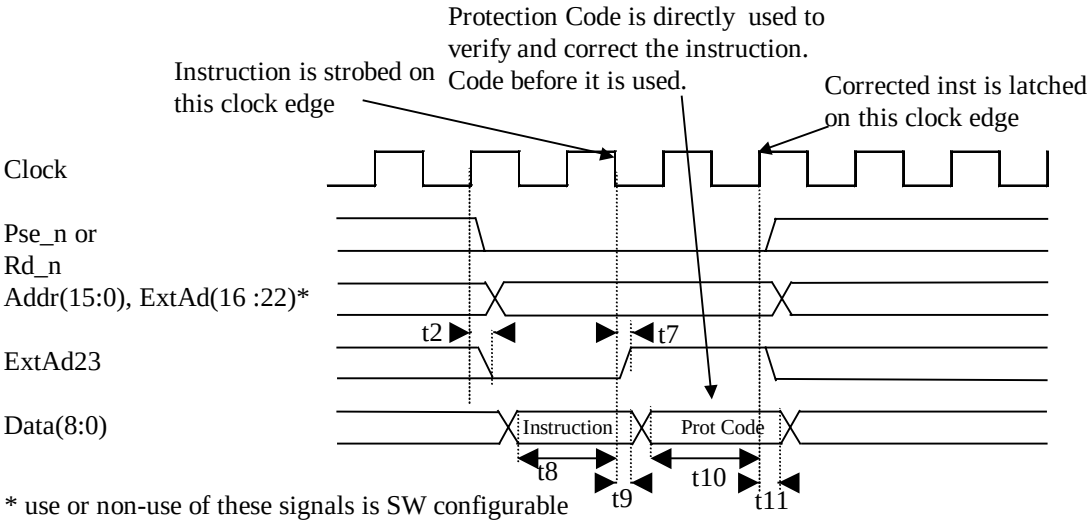


Figure 3-2: Program access with memory protection on

The minimum access time of the memory devices used to store the program shall be equal to $3T_{Clock}$– 2 ns when the memory protection is off, and equal to 1,4 T_{Clock}– 8ns when the memory protection is on. However to accommodate slow devices it is possible to insert between 1 and 15 wait states of 1 T_{Clock} in all program accesses, which increase the minimum access time between 1 and 15 T_{Clock} (cf. §3.4.3).

The figures hereafter present how read accesses to the data memory are handled in the case when memory protection is off (Figure 3-3) or on (Figure 3-4).

In both cases, the access is performed in 4 clock cycles. When the memory protection is on then the ExtAd23 signal is used to indicate if the microcontroller accesses the instruction (ExtAd23 = '0') or the protection code (ExtAd23 = '1'). ExtAd23 is at logic level '0' for 1.5 clock cycles at the beginning of the access and then at logic level '1' for the rest of the access. The data is latched in an intermediate register 1.5 clock cycles after the beginning and it is corrected in real time with the data present on the bus in the second part of the access before it is used.

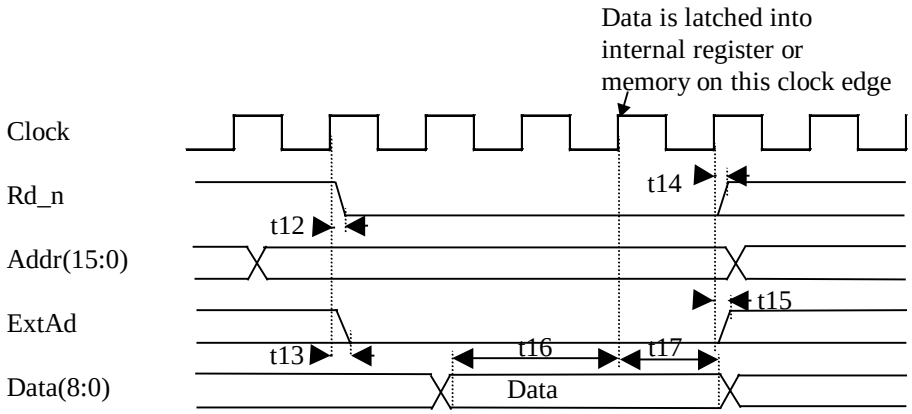


Figure 3-3: Read data access with memory protection off

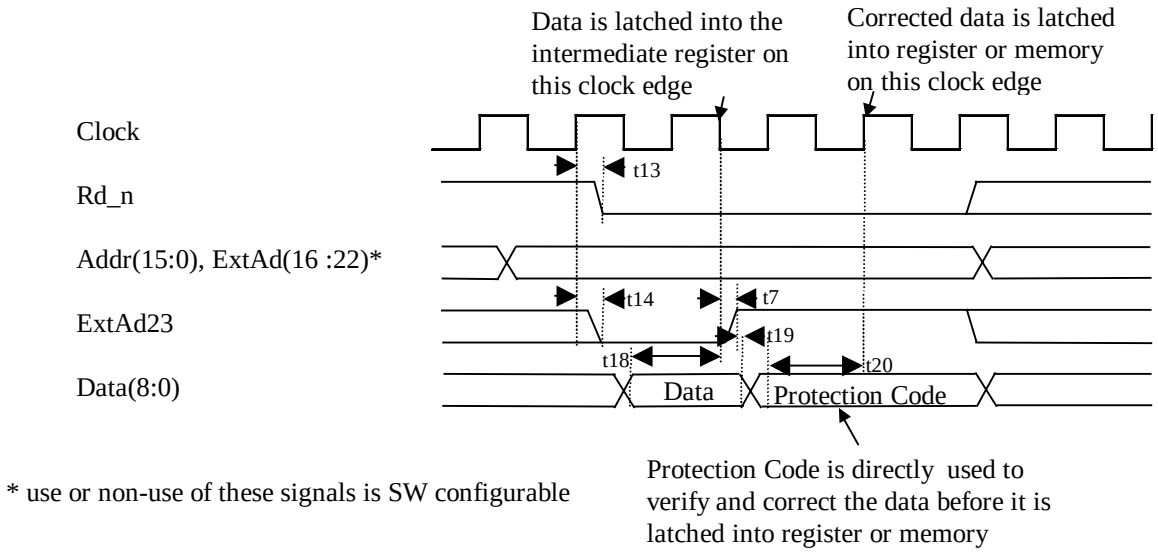
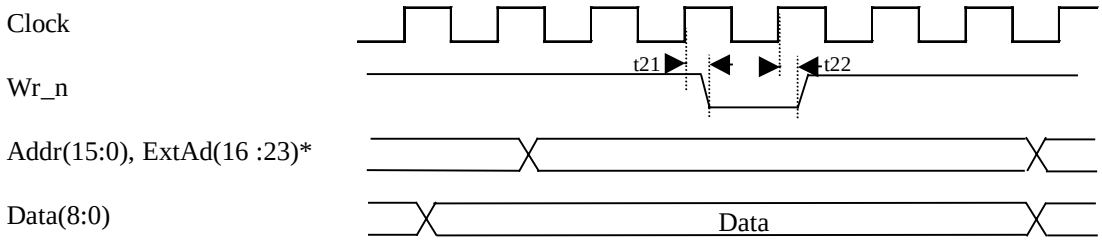


Figure 3-4: Read data access with memory protection on

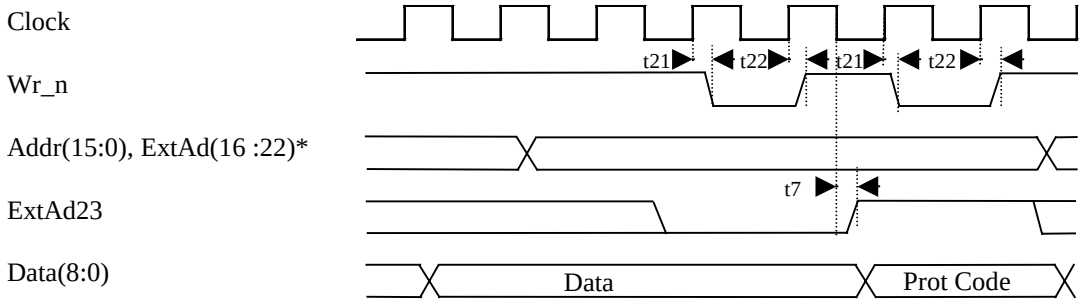
The figures hereafter present how write accesses to the data memory are handled in the case when memory protection is off (Figure 3-5) or on (Figure 3-6).

In both cases, the access is performed in 7 clock cycles. When the memory protection is on then the ExtAd23 signal is used to indicate if the microcontroller outputs data on the data bus (ExtAd23 = '0') or a protection code (ExtAd23 = '1'). The design guarantees that when Wr_n is asserted then all the signals ExtAd(23-16), Add and Data are stable. In addition, these signals change only half a clock cycle after the Wr_n rising edge.



* use or non-use of these signals is SW configurable

Figure 3-5: Write data access with memory protection off



* use or non-use of these signals is SW configurable

Figure 3-6: Write data access with memory protection on

The minimum access time of the memory devices used to store the program shall be equal to $3T_{Clock}$ – 2 ns when the memory protection is off, and equal to 1,4 T_{Clock} – 8 ns when the memory protection is on. In addition the minimum write pulse duration shall be less than 1 T_{clock}. However to accommodate slow devices it is possible to insert between 1 and 15 wait states of 1 T_{Clock} in all program accesses, which increase the minimum access time between 1 and 15 T_{Clock} (cf. §3.4.3).

3.4.3 Wait states insertion

The ADV80S32 gives the possibility to include between 1 and 15 Wait States into program and data accesses. Wait state insertions is compatible with both the ROM program execution and the RAM program execution mode, and also with memory protection scheme. The number of Wait States to be inserted is defined into the WaitMem SFR which is presented in the figure hereafter.

SFR @	(MSB)		(LSB)
$0C3_H$	WSPA	WSDA	

Symbol	Significance in write mode	Significance in read mode
WSPA	Wait states for program accesses. This field shall be used to program the number of wait states to insert during program accesses.	Indicates the number of Wait States inserted during program accesses.
WSDA	Wait states for data accesses. This field shall be used to program the number of wait states to insert during program accesses.	Indicates the number of Wait States inserted during data accesses.

Figure 3-7: WAITMEM register

WAITMEM register acts on the length of the Pse_n, Rd_n and Wr_n pulses by extending their duration of WSPA T_{clock} for a program memory access and WSDA T_{clock} for a data memory access. During read operations and when the memory protection is on, the 80S32 reads two data in the same cycle (cf. Figure 3-2 and Figure 3-4) WSPA or WSDA waits states are inserted both for the acquisition of the data and for the acquisition of the protection code (e.g. before and after the ExtAd23 rising edge).

3.4.4 External Memory Protection

The EMCON register allows to activate memory protection for both the program memory accesses and data memory accesses. EDAC for program accesses can be switched on or off with the PMCON bits of the EMCON register, while the EDAC for data accesses can be switched on or off with the DMCON bit in the EMCON register.

When the external memory protection is used, the EXTAD23 output shall be enabled in the EXTBUS register (cf. 3.4.5). It shall be used externally to differentiate between data accesses (EXTAD23 = ‘0’) and protection code accesses (EXTAD23 = ‘1’).

When an error is detected during a program access, it is flagged in the PG_ER_FLAG bits in the MPSTAT register and an interrupt is generated if it was previously enabled (cf. Figure 3-12). The location address of the error can be found in the PDERADH (MSB) and PDERADL (LSB) registers. After the error has been processed the PG_ER_FLAG has to be reset by SW (by writing in the MPSTAT register, cf. Figure 3-12).

The 80S32 process in the same way errors detected during data accesses, except that an error is in this case flagged in the DT_ER_FLAG bits of the MPSTAT register.

3.4.5 Memory banks

The ADV80S32 enable to extend the external memory spaces up to 8 Mbytes for program and 16 Mbytes for the code by using page memory schemes.

To support memory banks the address bus can be extended from 16 bits to 24 bits via the EXTBUS register presented in the figure hereafter:

SFR @	(MSB)							(LSB)
$0B5_H$	EXTAD23	EXTAD22	EXTAD21	EXTAD20	EXTAD19	EXTAD18	EXTAD17	EXTAD16

Symbol	Significance in write mode	Significance in read mode
EXTAD16	Program memory configuration: • '0': configure P0.7 as an IO port • '1': configure P0.7 is the 17th address bit	Program memory configuration: • '0': P0.7 as an IO port • '1': P0.7 is the 17th address bit
EXTAD17	Program memory configuration: • '0': configure P0.6 as an IO port • '1': configure P0.6 is the 18th address bit	Program memory configuration: • '0': P0.6 as an IO port • '1': P0.6 is the 18th address bit
EXTAD18	Program memory configuration: • '0': configure P0.5 as an IO port or CSB_2* • '1': configure P0.5 is the 19th address bit	Program memory configuration: • '0': P0.5 as an IO port or CSB_2 • '1': P0.5 is the 19th address bit
EXTAD19	Program memory configuration: • '0': configure P0.4 as an IO port or CSB_1* • '1': configure P0.4 is the 20th address bit	Program memory configuration: • '0': P0.4 as an IO port or CSB_1 • '1': P0.4 is the 20th address bit
EXTAD20	Program memory configuration: • '0': configure P0.3 as an IO port or CSB_0* • '1': configure P0.3 is the 21st address bit	Program memory configuration: • '0': P0.3 as an IO port or CSB_0 • '1': P0.3 is the 21st address bit
EXTAD21	Program memory configuration: • '0': configure P0.2 as an IO port or CSA_2* • '1': configure P0.2 is the 22nd address bit	Program memory configuration: • '0': P0.2 as an IO port or CSA_2 • '1': P0.2 is the 22nd address bit

EXTAD22	Program memory configuration: • '0': configure P0.1 as an IO port or CSA_1* • '1': configure P0.1 is the 23rd address bit	Program memory configuration: • '0': P0.1 as an IO port or CSA_1 • '1': P0.1 is the 23rd address bit
EXTAD23	Program memory configuration: • '0': configure P1.2 as an IO port • '1': configure P1.2 is the 24th address bit or used for memory protection.	Program memory configuration: • '0': P1.2 as an IO port • '1': P1.2 is the 24th address bit or used for memory protection.

*** Configuration between port and CS function is selected with the P0DIR SFR register**

Figure 3-8: EXTBUS register

Each bit can be individually configured. For program memory space, paging mechanism takes care that the interrupt vector table located in the program memory between address 0 and 73_H shall be accessible at any time and that the instruction flow shall not be disturbed during pages switching. To do so, the program memory between addresses 0000_H and $7FFF_H$ is used as a common memory that does not support paging and the program memory between addresses 8000_H and $FFFF_H$ is used as a paged memory where the accessed page can be selected via the CODBANK 8-bit register.

Starting from the 8032 16-bit address, the chip elaborates the 23-bit code address as follow:

- if 0 <= 8032 address < 8000_H then the 23-bit address is a concatenation of 7-bit to '0' with the 8032 address,
- if 8000_H <= 8032 address < $FFFF_H$ then the 23-bit address is a concatenation of the CODBANK register with the 15 least significant bit of the 8032 address.

Common memory will be used to host the interrupt vector table, the page switching routines and others user defined routines which can be accessed independently from the selected page. The paged memory will be used to host the rest of the program that shall be broken down into blocks of 32 Kbytes and distributed into the available pages.

Many compilers and linkers support bank switching, and many of them include library functions for page switching. The documentation accompanying your compiler will provide information concerning its expanded memory support.

For data memory space the paging is implemented on the whole address range and a 24-bit address is elaborated by concatenating the DTBANK register with the 16-bits of the 8032 address. However, when the external memory protection is on, the most significant bit of DTBANK is not used and ExtAD23 is used as an indicator of the access ('0' = data access, '1' = code access).

As for the DPTR register, the DTBANK register is doubled. At any time one of them is selected and used to elaborate the address for all data memory access. Selection between the two registers is performed with the DPSEL register. This mean that to each DPTR register is associated a DTBANK register that enable faster block data transfers between banks.

Access to the XRAM with MOVX A,@DPTR and MOVX @DPTR,A instructions is only possible when DTBANK = 0. When DTBANK is different from 0, MOVX A,@Ri and MOVX @Ri,A shall be used for all accesses to XRAM.

3.4.6 On-chip address decoder unit

For applications not needing more than 128 Kbytes of protected or unprotected memories, the 80S32 provides an on-chip address decoder unit that can avoid the use of extra glue logic between the microcontroller and the memory. The microcontroller can manage up to 6 chip select signals, 3 for the program memory space and 3 for the data memory space.

The CS generation is performed in two stages. The first stage performs the address decoding and generates 6 CS signals, 3 for program accesses and 3 for data accesses.

The decoding function depends on the size of the memory devices used to implement the program memory space and data memory space (which is defined via the CSCON register cf. Figure 3-10) and whether the memory accesses are protected by EDAC or not (which is defined in the EMCON register cf. Figure 3-11). The table hereafter summarizes the possible configurations and the values of the Add and ExtAd signals for which CS signals are asserted.

Device size	**Memory protection**	**CS0 and CS3***	**CS1 and CS4***	**CS2 and CS5***
8K	protected	0 < Add < 4K	4K < Add < 8K	8K < Add < 12K
8K	unprotected	0 < Add < 8K	8K < Add < 16K	16K < Add < 24K
32K	protected	0 < Add < 16K	16K < Add < 32K	32K < Add < 48K
32K	unprotected	0 < Add < 32K and EXTAD16=0	32K < Add < 64K and EXTAD16=0	0K < Add < 32K and EXTAD16=1
128 K	protected	always set		-
128 K	unprotected	always set	-	-

* CS0, CS1 and CS2 are unset during data memory accesses; CS3, CS4 and CS5 are unset during program memory accesses.

The second stage is used to route the program access CS signals (CS0, CS1, CS2) and data access CS signals (CS3, CS4, CS5) onto the microcontroller interface signals CSA_0, CSA_1, CSA_2 and CSB_0, CSB_1, CSB_2 (all signals are low level active).

The connection between these signals is depending on the External Memory Access type defined in the EMA bits (cf. Figure 3-11) and are presented in the table hereafter:

Configuration	CSA_0	CSA_1	CSA_2	CSB_0	CSB_1	CSB_2
EMA = '01'	CS3	CS4	CS5	N.U.	N.U.	N.U.
EMA = '11'	CS0	CS1	CS2	CS3	CS4	CS5
EMA = '10'	CS0 or CS3	CS1 or CS4	CS2 or CS5	N.U.	N.U.	N.U.

Note: EMA = '00' not used

As defined, the address decoder function is compatible with the program-downloading feature. The downloading program can either be the internal bootstrap program or a program stored in an external ROM accessed with the Pse_n signal. If data and program share the same memory space, the memory devices used to store data and memory shall be connected to the CSA signals. If not the memory devices used to store the program shall be connected to the CSA signals while the memory devices used to store data shall be connected to the CSB signals.

After reset the microcontroller is in the « ROM program execution» mode and executes the on-chip downloading routine (EA = '1') or the program stored in an external PROM (EA = '0'). In this configuration, all accesses to the data memory space are addressing devices connected to the CSA signal. After program downloading, the last action of the bootstrap program shall then be to enter the « RAM Program Execution» mode that causes the device to be reset and to start the execution of the program from RAM address 0.

In the « RAM Program Execution» mode all accesses to the data program space are addressing devices connected to the CSA signals that corresponds to the RAM where the program has been downloaded. Concerning data, depending whether the « separated RAM configuration » or « shared RAM configuration » is in use then all accesses will address devices connected to the CSB signals or the CSA signals.

3.4.7 Memory organization summary

The table hereafter summarizes the different physical memory spaces that are used by the 80S32 and their use depending on the selected configuration mode.

	ROM Space	RAM Space	RAM Extension Space
Memory	ROM	RAM	RAM
Maximum size	8M*8	16M*8	16M*8
control signal	Pse_n	Rd_n and Wr_n	Rd_n and Wr_n
Selection signal	Pse_n	CS_A	CS_B
use of addresses 0 to $7FFFFF_H$			
'ROM Program Access' mode	Program	Data,	unused,
'Separated RAM program Access' mode	unused	program	data
'Shared RAM program Access' mode	unused	Program and data	unused
use of addresses 800000_H to $FFFFFF_H$			
'ROM Program Access' mode	protection code	data or protect. Code	unused
'Separated RAM program Access' mode	unused	Unused or Protect. Code	data or protect. Code
'Shared RAM program Access' mode	unused	data or protect. Code	unused

Memory of the 80S32 is split into 3 different physical spaces, the ROM space, the RAM space and the RAM extension space. The ROM space corresponds to the program space of the original 8051 and contains the code to be executed after a reset when EA = 0. It can be the application program or a bootstrap to download the application program into the RAM. Access to the ROM space is always indicated with Pse_n at logic level '0' and Rd_n and Wr_n are never asserted for ROM accesses.

The RAM space corresponds to the data space of the original 8051. In the 'ROM Program Access mode' it is used by the 80S32 to store program variables. In the 'Separated RAM Program access' mode it shall contain the application program since the 80S32 reads all code information from this memory area. In the 'Shared RAM Program Access' mode it shall contain the application program but it is also used by the 80S32 to store program variables. In this last case this is the programmer responsibility to verify code and data will not overlap. Access to RAM space is always indicated with the CSA, Rd_n and Wr_n signals. Pse_n is never asserted for RAM accesses..

The RAM extension space is specific to the 80S32 and is only used in the 'Separated RAM Program Access' mode to store program variable. Access to RAM space is always indicated with CSB, Rd_n and Wr_n signals. Pse_n is never asserted for RAM Extension accesses

Program downloading is performed either by executing the internal bootstrap program (EA=1) or an external bootstrap program located in ROM (EA=0) with the 80S32 in the 'ROM program access mode'. In this mode the bootstrap program can download the program into the RAM space that is considered as the data space. During downloading, XRAM shall be disabled in order to be able to download the program between address 0 and $0FF_H$. After downloading it is just needed to switch to one of the 'RAM Program Access' mode in order to start the execution of the downloaded program. It is then possible to use the XRAM.

3.5 On chip bootstrap program

The 80S32 includes a bootstrap program that is activated after reset when the EA interface signal has been pulled to VCC through a 10 K resistor. The program configures extra USART1 as a PacketWire receiver and extra USART2 as a PacketWire transmitter.

After configuration, the bootstrap program waits for the reception of a frame onto extra USART1 interface. A frame contains between 11 and 128 bytes and is delimited with the SXVAL signal. A frame starts with setting of SXVAL and ends with resetting of this same signal. The 80S32 process CCSDS formatted packet. It accepts both TC and TM packet but all its reports are formatted according to TM packet structure. In the rest of this paragraph, a packet received by the 80S32 will be called a command and a packet sent by the 80S32 will be called a report. The structure of a packet is presented in the figure hereafter :

PACKET HEADER							**PACKET DATA FIELD**		
PACKET ID				Packet sequence control		Packet length	Data Field Header	Application data	Packet error control
Version number = 0 or 4	Type = 0 or 1	Data Field Header Flag = 1	Application Process ID	Sequence Flag = 3	Sequence count				
3 bits	1 bit	1 bit	11 bits	2 bits	14 bits				
16 bits				16 bits		16 bits	8 bits	8 to 952 bits	16 bits

The version number indicates the packet version. For a TC packet this version number shall be ‘000’, for a TM packet the version number shall be ‘000’ or ‘100’.

The type bit indicates if the packet is a TC (Type = ‘1’) or a TM (Type = ‘0’). This field is checked with the previous one and an ID error is indicated in an Acknowledge Report(cf. Figure 3-9) when type = ‘1’ and version is different from 0 or when type = ‘0’ and version is different from 4 or 0.

The Application Process identifies the application to which the packet is sent. The APID field of a command is not checked by the device. The APID field value to be used in report can be downloaded via a command.,

The sequence flag is always ‘11’ indicating that the application data are not segmented. This field is checked by the device that indicates and ID error in an Acknowledge Report (cf. Figure 3-9) when its value is different from ‘11’.

The sequence count is an incremental value which number the packet. It is stored by the device and used in the sequence count field of the report generated for the response. The 80S32 does not verifies the increment of this field.

The Packet Length indicates the number of bytes-1 of the Packet Data Field. When a complete command has been received, the 80S32 verifies that the packet length is coherent with the length indicated in this field.

The Data Field Header is a 1 byte value which is used to indicate the type of the command (received packets) or the type of the report (transmitted packets).

The Application Data field contains all parameters of the command,

The Packet Error Control is a 16-bit code that protect the Packet Data Field. It contains the result of the CRC calculation performed on the Data Field Header and Application Data fields. The 80S32 verifies this field and do not process the packet if this check fail. In case of Checksum error an Acknowledge report reporting the error is generated.

The 80S32 downloading program processes 5 different commands and generates 3 different reports which are presented in the figure hereafter :

Name	**Data Field Header (8 Bits)**	**Data field length in Bytes**	**Data field definition**
Memory Write command	0	4 to 119	This packet is used to load data into the program memory. The data field contains between 4 and 119 bytes defined as follow: Byte 1: Program page address, Byte 2: Program Start Address MSB, Byte 3: Program Start Address LSB Byte 4 to 119 : program data The 80S32 responds to a correct load program with a positive acknowledge packet.
Memory Read command	1	4	This frame is used to read data from the program memory. The data field contains 3 bytes as follow: Byte 1: Program Page Address, Byte 2: Program Start Address MSB, Byte 3: Program Start Address LSB, Byte 4: Number of bytes to transfer (maximum 119, any value greater than the limit is interpreted as 119). The 80S32 responds to a correct read program with a Memory content report.
Memory Content report	2	1 to 119	This packet is used in response to a memory read. The data field contains between 1 and 119 bytes read from the memory.

Configure micro command	3	7	This command is used to configure the external memory, and directly acts on the MPSTAT, EMCON and CSCON and EXTBUS registers. It contains 3 bytes as follow: Byte 1: Value to be written into MPSTAT, Byte 2: value to be written into EMCON, Byte 3: value to be written into CSCON, Byte 4: value to be written into EXTBUS, Byte 5 : value to be written into WAITMEM, Byte 6 : value to be written into P0CON, Byte 5 : value to be written into P1CON, The 80S32 responds to a correct configure external memory with a memory configuration report.
Micro configuration report	4	7	This report is used in response to Configure External Memory command and indicates the value in the different registers as follow: Byte 1: Value of the MPSTAT register, Byte 2: value of the EMCON register, Byte 3: value of the CSCON register, Byte 4 : value of the EXTBUS register. Byte 5 : value of the WAITMEM register, Byte 6 : value of the P0CON register, Byte 5 : value of the P1CON register,
Start Execution command	5	0	This command is used to exit the download mode and to start program execution from external RAM. The 80S32 responds to a Start Execution with a positive acknowledge report

Acknowledge report	6	1	This report is used in response to a correct Load Program or Start Execution command, or in response to an incorrect command. The data byte is coded as follow: 00: No error 01: ID error, 02: Length Error, 03: Checksum error, 04: Command error
Set Packet Type command	7	3	This command is used to define the type of packet to be generated by the 80S32 and the APID value to be used for the responses. Octet 1 : code if packet shall be CCSDS compliant when 0 or PSS compliant when 80_H; Octet 2 : 5 most significant bit of the APID right aligned; Octet 3 : 8 least significant bit of the APID,

Figure 3-9 : Packet Data Field definition

After reset, the bootstrap program enters a loop where it is waiting for the reception of a command packet on the extra serial line 1. When a complete packet has been received, the program checks the command validity before to execute it. Finally, the program prepares and send the appropriate report on the extra serial line 2.

On reception of the Start Execution frame, the 80S32 responds with an acknowledge frame before to exit the loop and to start program execution from the RAM. The EA pin is then configured as an output initialized to '0'.

3.6 Error Detection and Correction (EDAC)

The memory protection is performed by an EDAC that calculates for each data byte an 8-bit protection code. This protection code is made of 8 bits and enables to correct single and double error.

The 8-bit protection code c7-c0 of a nibble i7-i0 is calculated as follows:

c7 = not(i0⊕ i2⊕ i5⊕ i7)
c6 = not(i1⊕ i2⊕ i3⊕ i5⊕ i6⊕ i7),
c5 = i3⊕ i4⊕ i5⊕ i6,
c4 = i0⊕ i4⊕ i5⊕ i6⊕ i7,
c3 = i0⊕ i1⊕ i2⊕ i6,
c2 = i0⊕ i1⊕ i2⊕ i3⊕ i7,
c1 = i0⊕ i1⊕ i3⊕ i4⊕ i5⊕ i7,
c0 = i1⊕ i4⊕ i6⊕ i7,

Error detection on an 8-bit data i7-i0 protected with the 8-bit code c7-c0 is performed by calculating the 8 following syndromes:

s7 = c7 ⊕ not(i0⊕ i2⊕ i5⊕ i7)
s6 = c6 ⊕ not(i1⊕ i2⊕ i3⊕ i5⊕ i6⊕ i7),
s5 = c5 ⊕ i3⊕ i4⊕ i5⊕ i6,
s4 = c4 ⊕ i0⊕ i4⊕ i5⊕ i6⊕ i7,
s3 = c3 ⊕ i0⊕ i1⊕ i2⊕ i6,
s2 = c2 ⊕ i0⊕ i1⊕ i2⊕ i3⊕ i7,
s1 = c1 ⊕ i0⊕ i1⊕ i3⊕ i4⊕ i5⊕ i7,
s0 = c0 ⊕ i1⊕ i4⊕ i6⊕ i7,

When S0 = S1 = S2 = S3 = S4 = S5 = S6 = S7 0 then the data is considered as correct. When one of the syndromes is different from 0 then the data has been corrupted and a decoding performed on the 8 syndromes bit enable to correct single and double error.

.

3.7 Special Function Register used for memory management

CSCON register is used for configuring the CS generation. It is presented in the figure hereafter:

SFR @	(MSB)				(LSB)
$09E_H$	-	-	CSPGCON	-	CSDTCON

Symbol	Significance in write mode	Significance in read mode
CSPGCON	Program memory configuration: • '00': no change • '01': configure for 8K devices • '10': configure for 32K devices • '11': configure for 128K devices	Program memory configuration: • '00': not used • '01': 8K devices configuration • '10': 32K devices configuration • '11': 128K devices configuration (default)
CSDTCON	Data memory configuration: • '00': no change • '01': configure for 8K devices • '10': configure for 32K devices • '11': configure for 128K devices	Data memory configuration: • '00': no change • '01': 8K devices configuration • '10': 32K devices configuration • '11': 128K devices configuration (default)

Figure 3-10: CSCON register

The External Memory Configuration register is used for the external memory configuration. It is presented in the figure hereafter.

SFR @	(MSB)				(LSB)
096_H	-	-	PMCON	DMCON	EMA

Symbol	Significance in write mode	Significance in read mode
PMCON	Program memory configuration: • '0X': no change • '10': disable memory protection • '11': enable memory protection	Program memory configuration: • '00': memory protection disabled (default) • '01': memory protection enabled • '1X': not used
DMCON*	Data memory configuration: • '0X': no change • '10': disable memory protection • '11': enable memory protection	Data memory configuration: • '00': memory protection disabled (default) • '01': memory protection enabled • '1X': not used •
EMA	External Memory Access: • '0X': no change • '01': set « ROM program execution» configuration • '10': set « RAM program execution» configuration (program and data shares the same address space) • '11': set « RAM program execution» configuration (data and program are in separate address spaces)	External Memory Access: • '01': « ROM program execution» configuration • '10': « RAM program execution» configuration(program and data shares the same address space) • '11': « RAM program execution» configuration(program and data are in separate address spaces) • '00': not used

* when '10' or '11' memory protection is disabled

Figure 3-11: EMCON (External Memory CONfiguration) register.

The MPSTAT register flags the detection of internal and external errors. It is presented in the figure hereafter. When an error has been processed, the corresponding flag shall be reset by software.

SFR @	(MSB)			(LSB)
95_H	PG_ER_FLAG	DT_ER_FLAG	XMER	IMER

Symbol	Significance in write mode	Significance in read mode
PG_ER_FLAG	Error in program memory: • '00': no change • '01': flags an unrecoverable error flag • '10': flags a recoverable error flag • '11': resets flag error	Error in program memory: • '00': no error • '01': unrecoverable error • '10': recoverable error • '11': not used
DT_ER_FLAG	Error in data memory: • '00': no error • '01': flags an unrecoverable error • '10': flags an recoverable error • '11': resets flag error	Error in data memory: • '00': no error • '01': unrecoverable error • '10': recoverable error • '11': not used
XMER	Error in XRAM memory: • '00': no change • '01': flags an unrecoverable error • '10': flags a recoverable error • '11': resets flag error	Error in XRAM memory: • '00': no error • '01': unrecoverable error • '10': recoverable error • '11': not used
IMER	Error in internal memory: • '00': no change • '01': flags an unrecoverable error • '10': flags a recoverable error • '11': resets flag error	Error in internal memory: • '00': no error • '01': unrecoverable error • '10': recoverable error • '11': not used

Figure 3-12: MPSTAT register

The system configuration register is used for configuring the eXtra memory. It is presented in the figure hereafter:

SFR @	(MSB)				(LSB)
93_H	-	-	-	XMAP	

Symbol	Significance in write mode	Significance in read mode
XMAP	XRAM function: • '00': no change • '01': not used • '10': disables XRAM • '11': enables XRAM	XRAM function: • '00': XRAM disabled (default) • '01': XRAM enabled • '1X': not used

Figure 3-13: SYSCON register

The MPCON register is used for configuring the Internal Memory Protection. It is presented in the figure hereafter.

SFR @	(MSB)					(LSB)
94_H	-	-	-	-	IPCON	XPCON

Symbol	Significance in write mode	Significance in read mode
XPCON	XRAM protection configuration: • '0X': no change • '10': disable XRAM protection • '11': enable XRAM protection	XRAM protection configuration: • '00': XRAM protection disabled (default) • '01': XRAM protection enabled • '1X': not used
IPCON	Internal memory protection configuration: • '0X': no change • '10' disable internal memory protection • '11': enable internal memory protection	Internal memory protection configuration: • '00': internal memory protection disabled (default) • '01': internal memory protection enabled • '1X': not used

Figure 3-14: MPCON register

4. TIMER/COUNTERS

4.1 8032 Timer operation summary

This section summarizes the operation of the three timers provided in the original 8032/52 devices and which are also implemented in the 80S32. More information can be found in RD1.

4.1.1 Functional modes

The three timers can be configured in a timer mode and count in this case on the clock/12 (to maintain compatibility with the 80x1) or in a counter mode and count in this case the falling edges occurring its corresponding external input pin T0Ex, T1Ex, or T2Ex.

In counter mode the 80S32 provides some extra functions which allows to generate time durations which are longer than what was possible in the original 80x2 (cf. §4.1.3)

The selection of Timer/Counter mode is controlled by the C/T bit in the TMOD SFR.

The usage of Timer0 and Timer1 are almost the same (except in mode 3). They both have four operational modes:

Mode 0	13-bit timer/counter
Mode 1	16 bit timer/counter
Mode 2	Auto-reload 8 bit timer/counter
Mode 3	(Timer 0) Two 8-bit timer/counter (Timer 1) Stopped

Timer2 has three functioning modes as defined hereafter:

Mode 0	Auto-Reload
Mode 1	Capture-Mode
Mode 2	Baud-Rate generator

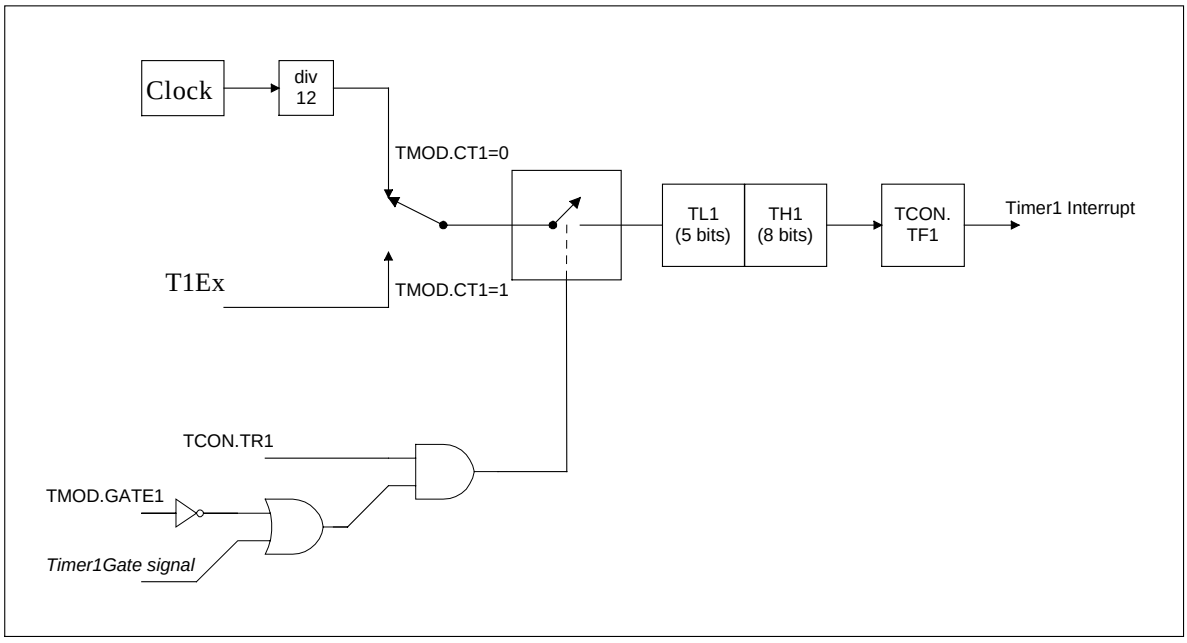
4.1.2 Timer/Counter Configuration

In timer mode, the registers are incremented at a maximum rate of clock / 12 while in the counter mode the registers are incremented in response to a falling edge on one of the external *T0Ex* or *T1Ex* or T2Ex signals. The external signals are only sampled once every 12 clock cycles, so the pulse should be low and high for at least 12 cycles to ensure a correct count. The maximum increment rate for the counter mode is then clock / 24.

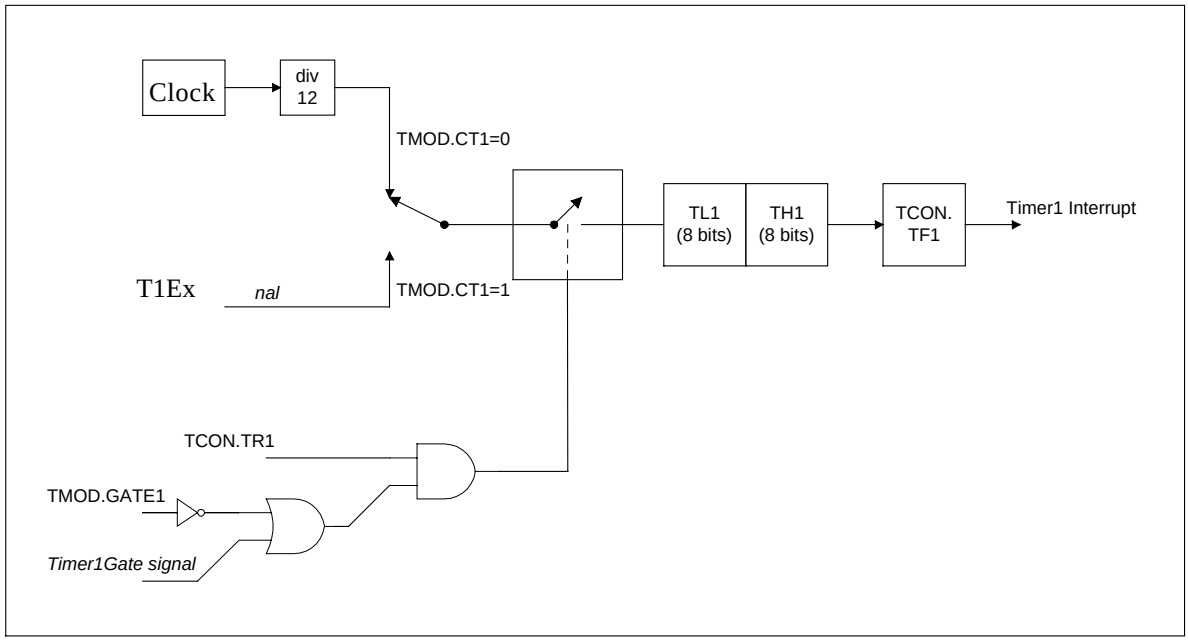
The Timer / Counters set the relevant interrupt flag, when the register overflows, rolling over to zero. In the auto-reload modes the overflow also reloads the incrementing register.

For Timer / Counters 0 and 1, mode 3 is special. In this mode Timer1 holds its count and Timer0 takes over its interrupt.

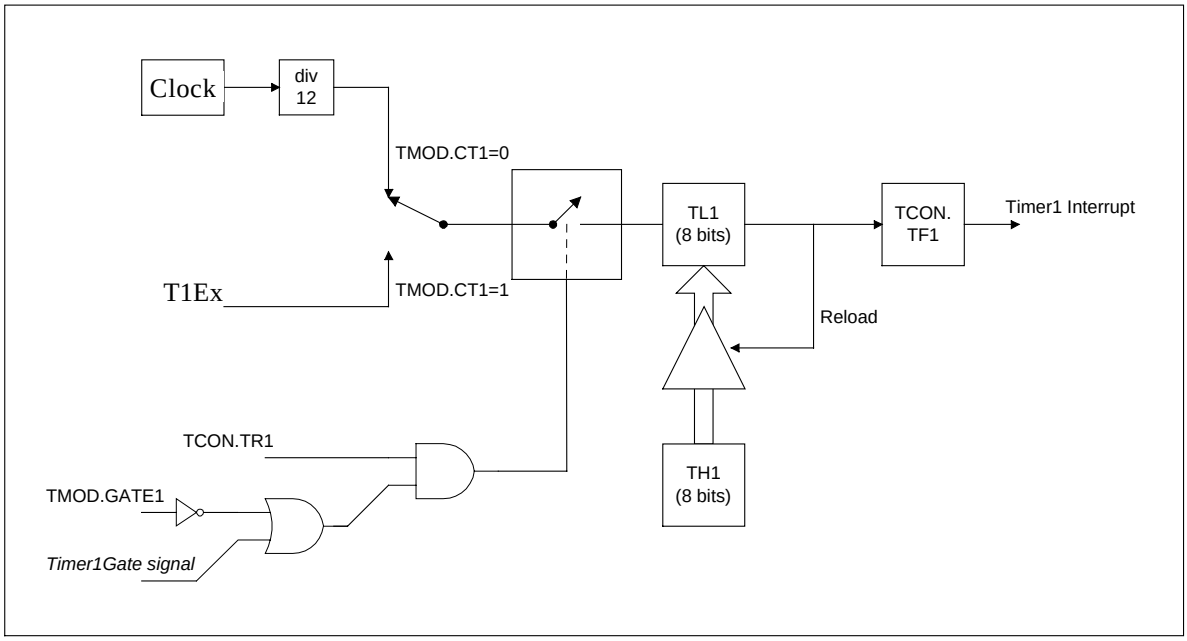
The following connection diagrams show the registers that are incremented in response to the external signals and set-up register bits.



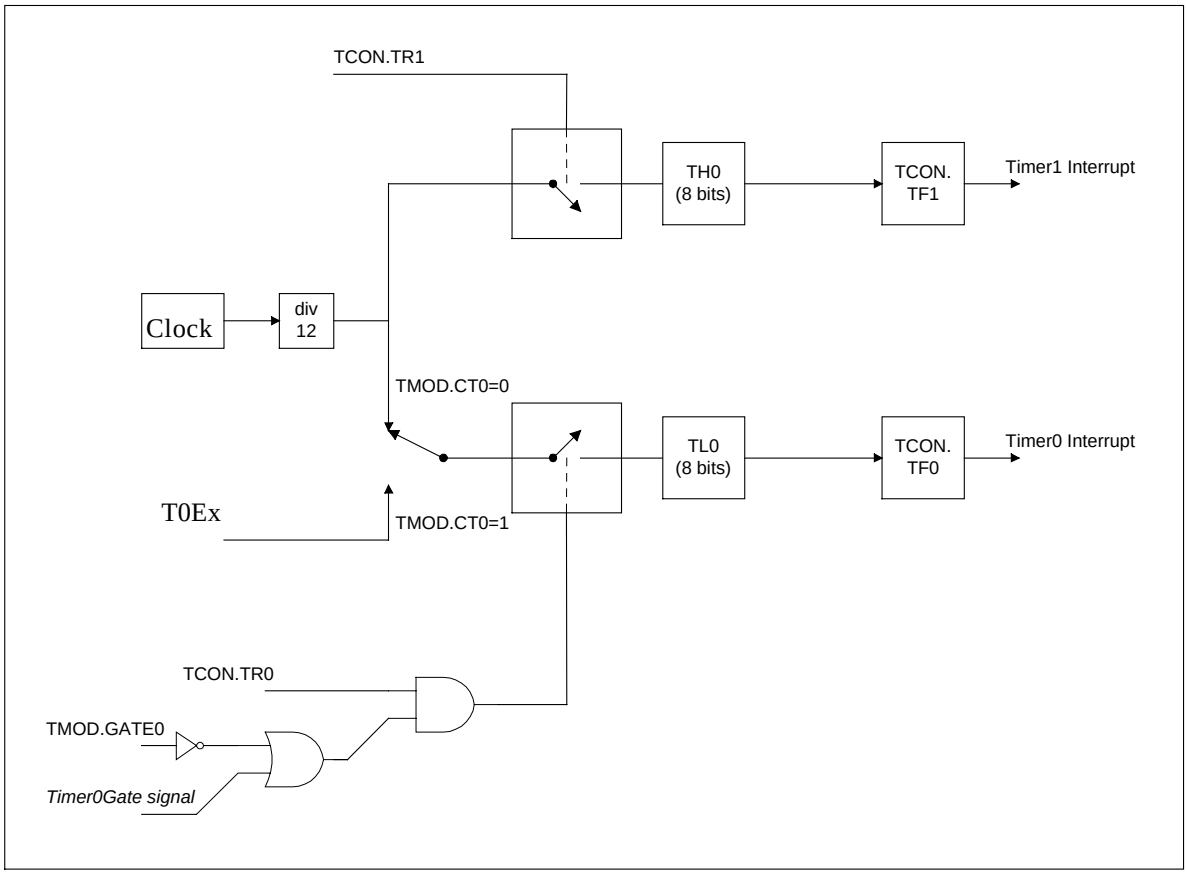
Timer/Counter1 Mode 0: 13-bit counter



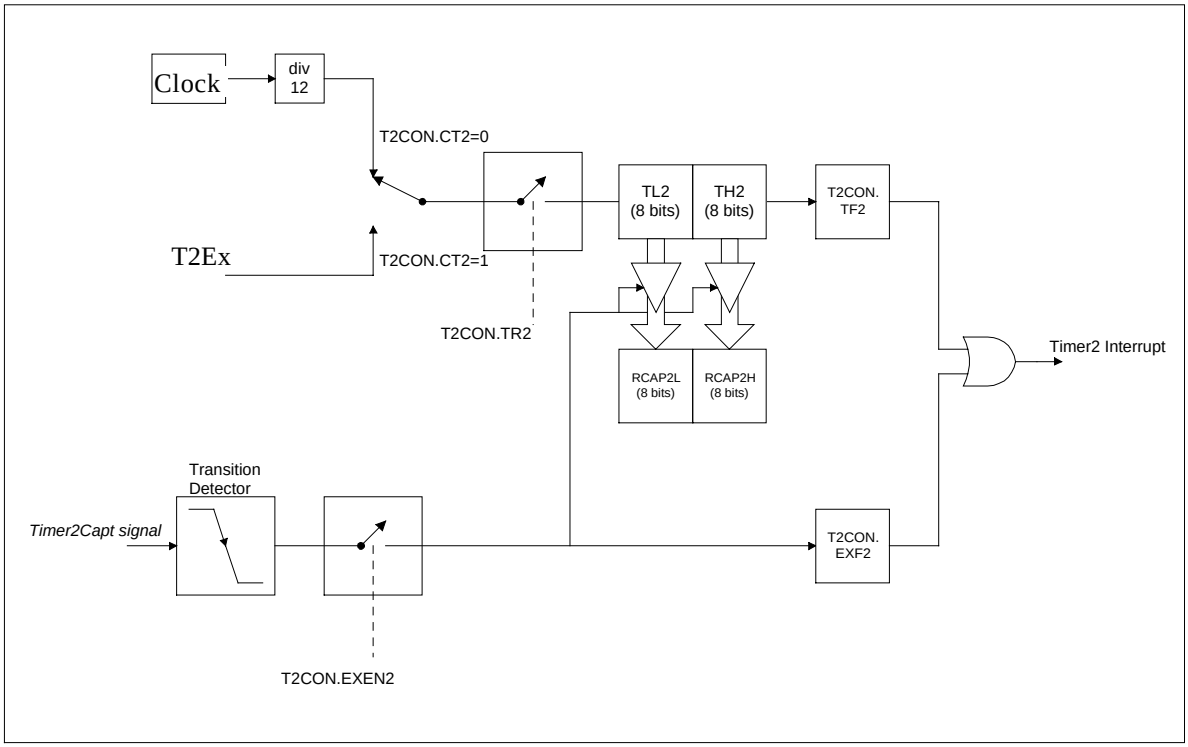
Timer/Counter1 Mode 1: 16-bit counter



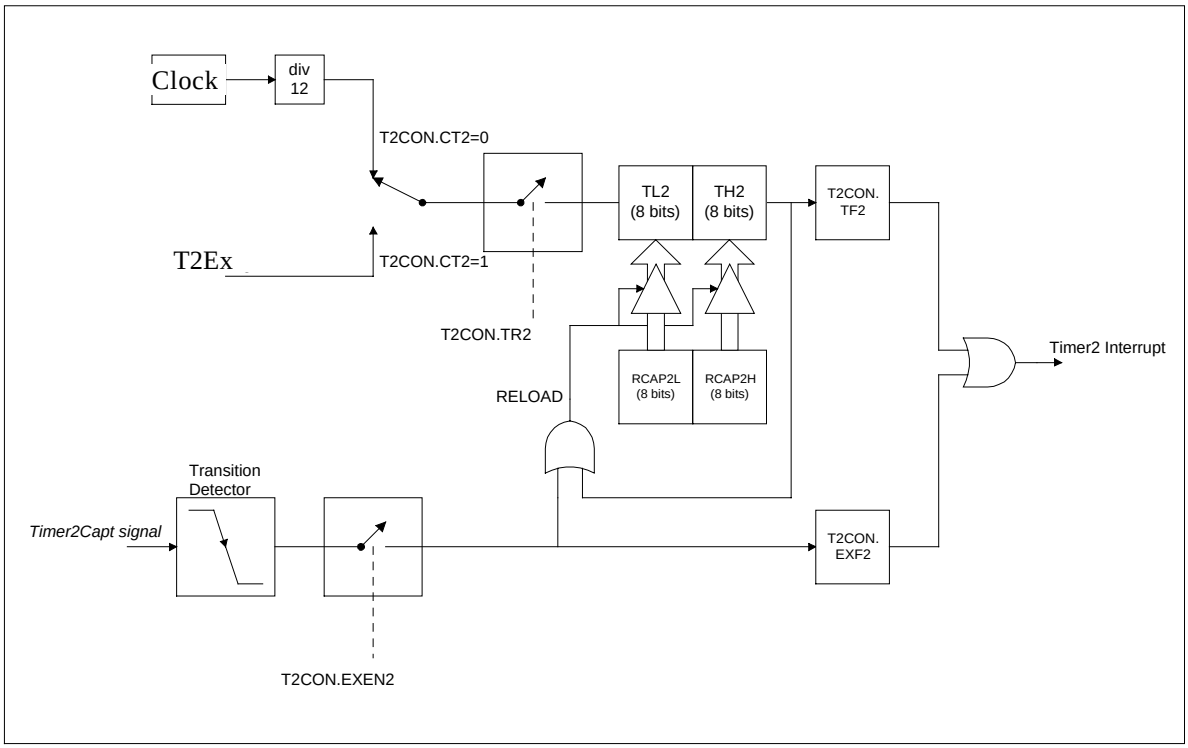
Timer/Counter1 Mode 2: 8-bit auto-reload counter



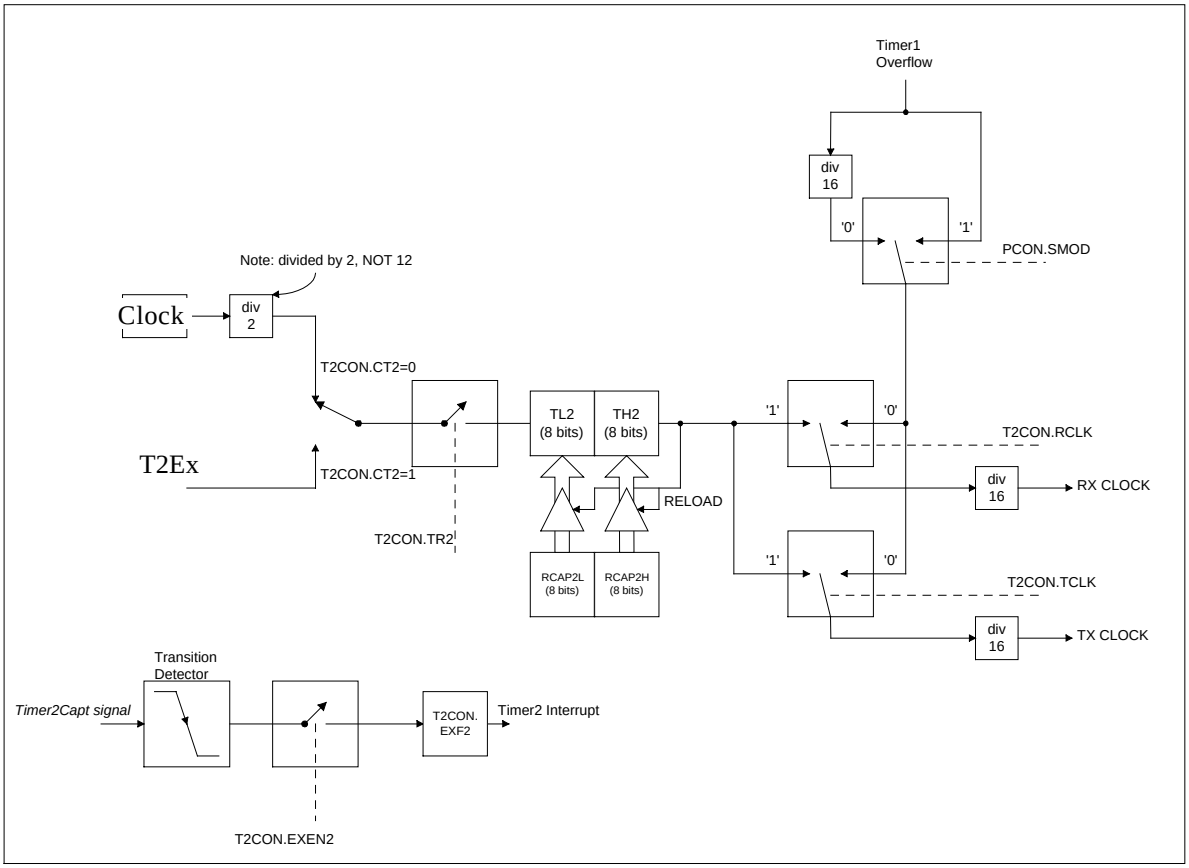
Timer/Counter0 Mode 3: Two 8-bit counters



Timer 2 in Capture Mode



Timer2 in Auto-Reload Mode



Timer2 in Baud Rate Generator Mode

4.1.3 Extended timer functions

In counter mode, when the C/T bits in the TMOD or TCON SFR is set, the counting source of the three timers can be selected between the external timer inputs T0Ex, T1Ex and T2Ex (as in the original 80x2) and the output of an internal frequency divider (specific to the ADV 80S32).

The frequency divider is common to the three timers and divides the clock (or the clock divided by 2) by a ratio programmable between 32 and 65535. The ratio (16 bits) is written MSB first in the TDIVH SFR and LSB last in the TDIVL SFR, while the input of the divider is selected in the TDIVCON register.

The counting source of the three timers is individually selectable in the TDIVCON register presented hereafter:

SFR @	(MSB)			(LSB)
$0B1_H$	TDIVIN0	T0IN0	T1IN0	T2IN0

Symbol	Significance in write mode	Significance in read mode
TDIVIN	Frequency divider input selection: • '10': select clock • '11': select clock / 2 • '0X': no change	Frequency divider input selection: • '00': clock selected (default) • '01': clock / 2 selected • '1X': not used
T0IN	Timer0 external input selection: • '10': select external input signal • '11': select frequency divider output • '0X': no change	Timer0 external input selection: • '00': external input signal selected (default) • '01': frequency divider output selected • '1X': not used
T1IN	Timer1 external input selection: • '10': select external input signal • '11': select frequency divider output • '0X': no change	Timer1 external input selection: • '00': external input signal selected (default) • '01': frequency divider output selected • '1X': not used
T2IN	Timer2 external input selection: • '10': select external input signal • '11': select frequency divider output • '0X': no change	Timer2 external input selection: • '00': external input signal selected (default mode) • '01': frequency divider output selected • '1X': not used

Figure 4-1: TDIVCON register

The maximum time durations which can be obtained for the different timers are given hereafter for a 12 MHz clock:

Timer	Maximum time duration
Timer0 and Timer1 (mode 0)	> 42 s
Timer0 and Timer1 (mode 1)	> 340 s
Timer0 and Timer1 (mode 2)	> 1 s
Timer2 (mode 0)	> 340 s

5. SERIAL PORTS

5.1 Introduction

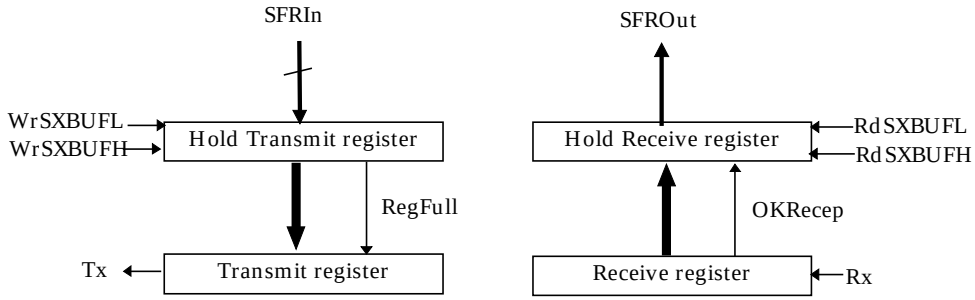
In addition to the standard 80C52 UART, the 80S32 provides four extra USARTs (Universal Synchronous Asynchronous Receiver Transmitter). Each extra USART can operate in three modes:

- RS232 asynchronous type mode which allows asynchronous full duplex transfer of 8 bit data at a programmable rate ;
- Synchronous PacketWire mode which allows half duplex transfer of byte packets ;
- Synchronous TTC-B-01 type mode that allows half duplex transfer of 8 or 16 bit words.

Each USART uses a 16 bit transmit hold register to be written with data to transmit and a receive hold register which contains the last received data. These two registers are both accessed via the Special Function Registers SXBUFL for the least significant byte and SXBUFH for the most significant byte.

In all modes where 8 bit data are transferred, Special Function Register SXBUFH is not used and the accesses to it have no impact on the transmission or reception.

In all modes where 16 bit data words are transferred, SXBUFH shall be used to write the most significant byte of the data to be sent or to read the most significant byte of the last received data while SXBUFL shall be used to write the least significant byte of the data to be sent or to read the least significant byte of the last received data. Since all transfers are initiated by an access to SXBUFL, then when the SW needs to send or wants to recuperate a 16 bit data used, it is mandatory to access SXBUFH prior to SXBUFL.



USART block diagram

The serial interface has 3 signals SXDT, SXVAL and SXCLK. The function of each of these signals is not fixed and depends on the current transfer mode of the USART as defined in the rest of this chapter.

Each extra USART includes a frequency divider that can be used to program the transmission rate. The frequency division ratio is programmable by steps of 16 between 16 and 2^{20} clocks in the SXFREQ register, and corresponds to the transmission time of 1 bit. The transmission Baud rate equals f_{Clock} / 16*SXFREQ (where SXFREQ = 0 ⇔ SXFREQ = 2^{16}).

The most significant byte of the division ratio shall always be loaded first into SXFREQH before to load the least significant byte into SXFREQL.

5.2 Configuration

The SXCON1 and SXCON2 special function registers allow to configure the serial interface associated to each extra USART. Structures of the SXCON registers are presented in Figure 5-1 and Figure 5-2. After reset the extra USARTs are disabled.

(MSB) (LSB)

MODE1	MODE0	ME2	ME1	ME0	N.U

SFR address = * $0D2_H$ for USART1, $0D9_H$ for USART2, $0E2_H$ for USART3, $0E9_H$ for USART4.

Symbol	Significance in write mode	Significance in read mode
MODE1 MODE0	Serial interface type : • '00': disable USART • '01': set RS232 mode • '10': set PacketWire interface mode • '11': set TTC-B-01 mode	Serial interface type : • '00': USART disabled (default) • '01': RS232 mode • '10': PacketWire interface mode • '11': TTC-B-01 interface mode
ME2 ME1 ME0	Extension mode	Extension mode

Figure 5-1: SXCON1 register

(MSB) (LSB)

OVW	RB8/EOR/BOT	TI	RI

SFR address = $0D3_H$ for USART1, $0DA_H$ for USART2, $0E3_H$ for USART3, $0EA_H$ for USART4.

Symbol	Significance in write mode	Significance in read mode
OVW	OVW register state : • '0X': no change • '10': reset OVW bit • '11': set OVW bit	Overwrite bit : Sets to indicate when a data has been received but that the previous one has not been read from SXBUF or that the FIFO is full. The new data has then overwritten the previous one.
RB8 or EOR or BOT	In RS232 mode: • ignored bit In PacketWire interface mode, EOR has a different meaning in transmission and In reception mode • in transmission mode control of the validation signal (SXVAL) and of the EOR1 bit: ⇒ '00' : no change ⇒ ''01' : reset EOR1 bit, ⇒ '10': Reset SXVAL, ⇒ '11': Set SXVAL. • in reception mode control of the EOR1 bit: ⇒ '01': reset EOR1, ⇒ '00' or '1X': no change In TTC-B-01 mode: • BOT = Transmission/reception starting bit in master mode: ⇒ '0X': no change, ⇒ 'X0': no change, ⇒ '11': Start transmission/reception ignored bit in slave mode	In RS232 mode: • RB8 = 9^{th} data bit received = parity bit received In PacketWire interface mode. EOR is a 2-bit structure : • The least significant bit EOR0 indicates the logical state of SXVAL. EOR0 inverts the value of SXVAL • The most significant bit EOR1 is set on the falling edge of SXVAL, indicating the end of a transfer. It shall be reset by software In TTC-B-01 mode: • BOT = ignored bit
TI	Hold Transmit register state: • '0X': no change • '10': reset TI bit • '11': set TI bit	Hold transmit register state: • '01': Hold Transmit register empty • '00': Hold Transmit register full • '1X': not used
RI	Hold Receive register state: • '0X': no change	Hold Receive register state: • '01': Hold Receive register full

Symbol	Significance in write mode	Significance in read mode
OVW	OVW register state : • '0X': no change • '10': reset OVW bit • '11': set OVW bit	Overwrite bit : Sets to indicate when a data has been received but that the previous one has not been read from SXBUF or that the FIFO is full. The new data has then overwritten the previous one.
	• '10': reset RI bit • '11': set RI bit	• '00': Hold Receive register empty • '1X' not used

Figure 5-2: SXCON2 register

Table hereafter presents the possible combinations for the different serial modes supported by the extra USARTs. They are presented more in detail in the following sections.

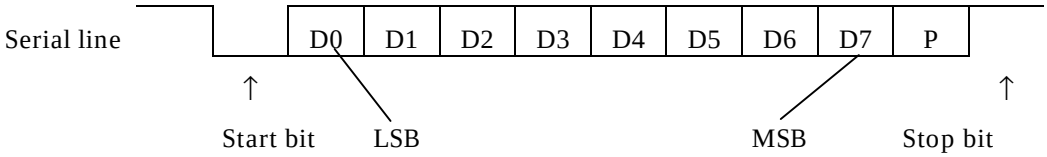
MODE	ME2	ME1	ME0	Significance
00	X	X	X	USART disabled
01	X	0	0	RS232 mode, no parity, 2 stop bits
01	X	0	1	RS232 mode, even parity, 1 stop bit
01	X	1	0	RS232 mode, odd parity, 1 stop bit
01	P	1	1	RS232 mode, programmable parity, 1 stop bit • ME2: Parity bit
10	X	X	0	PacketWire reception mode
10	X	X	1	PacketWire transmission mode • ME1 = 1: continuous transmission. Data in SXBUF is continuously transmitted
11	0	0	0	TTC-B-01 mode: 8 bits, slave input mode
11	0	0	1	TTC-B-01 mode: 8 bits, slave output mode
11	0	1	0	TTC-B-01 mode: 16 bits, slave input mode, master
11	0	1	1	TTC-B-01 mode:16 bits, slave output mode
11	1	0	0	TTC-B-01 mode: 8 bits, master input mode
11	1	0	1	TTC-B-01 mode: 8 bits, master output mode
11	1	1	0	TTC-B-01 mode: 16 bits, master input mode, master
11	1	1	1	TTC-B-01 mode: 16 bits, master output mode

Figure 5-3: Mode Extension bits interpretation

5.3 RS232 asynchronous type mode

The 80S32 is configured into this mode when the MODE bits in SXCON register are set to '01'. In this mode, each USART operates in full duplex meaning it can receive and transmit data simultaneously in an asynchronous manner.

Data are transmitted on the SXVAL signal, which is used as an output, and received on the SXDT signal, which is used as an input. The serial line format is:



The baud rate is programmable in the SXFREQ register as defined in 5.1.

5.3.1 Transmission

In the transmission mode, 11 bits are transmitted onto the serial line, which are:

- 1 start bit ;
- the 8 bits data which have been written in the SXBUF registers (the least significant bit is transmitted first) ;
- a programmable ninth bit which can be an odd parity bit, an even parity bit, a programmable bit (the bit value shall then be set in ME2) or the first stop bit (two stop bits configuration) ;
- one stop bit (which is the second one in the two stop bits configuration).

The TI bit in the SXCON register is set when the data contained in the Hold Transmit register is passed to the Transmit register for transmission on the line. It informs the user that the Hold Transmit register is empty and that it can write new data to the register. When the TI bit is set, an interrupt is generated if not masked (cf. § 6).

The TI bit is reset either when new data is written to the Hold Transmit register or directly by writing to this bit in the register. If new data is written to the Hold Transmit register while previous data has not been loaded to the Transmit register, the data present in the Hold Transmit register is overwritten.

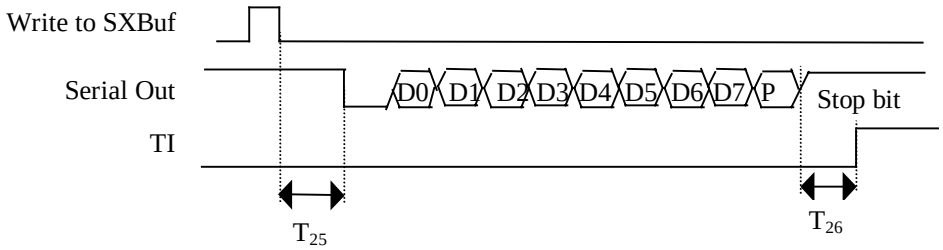


Figure 5-4 : Asynchronous transmission mode

5.3.2 Reception

In the reception mode, data are received by sampling the SXDT serial input line. The sampling frequency is 16 times faster than the bit transmission frequency programmed in the SXFREQ register(cf. §5.1).

The start bit is detected when the 1st, 7th, 8th and 9th of the 16 last samples are all '0'. Then the bits constituting the transmission are calculated each 16 sampling clock period, by performing a majority voting on the 7th, 8th and 9th samples of the bit period.

The 8 bits data words are shifted into the Receive register as soon as they are received and the first received bit shall be the least significant one.

Depending on the Mode Extension configuration, the 10th is processed as an odd or even parity bit, or as a user-defined bit or as the first stop bit. For all cases this bit is stored into the RB8 bit of the SXCON register.

The receiver performs some verification on the parity bit and number of bits. If all tests are correct, the data is transferred from the Receive register to the Hold Receive register after reception of the 11th bit.

If the parity bit is incorrect or if the stop bit is not found where expected, the reception is invalidated and the data is not written in the SXBUF registers.

The RI bit of the SXCON register is as soon as the received data is transferred into the Hold Receive register. It is used for informing the user that data has been received and can be read into the SXBUF register. The setting of the RI bit can generate an interrupt (cf. § 6). The RI bit is reset when the Hold Receive register has been read or by writing directly the RI bit in the SXCON register. If new data is received while the Hold Receive register is full, the old data is overwritten.

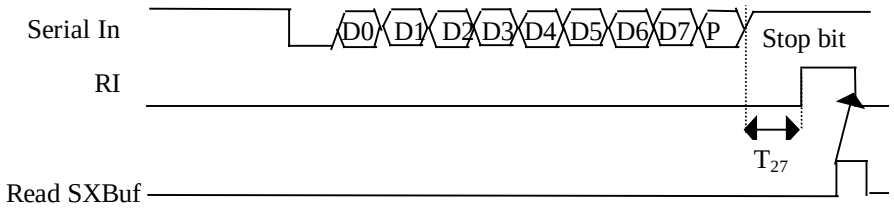


Figure 5-5: Asynchronous reception

5.4 Synchronous Packet Wire mode

The USART is configured in synchronous Packet Wire mode, also known as MAP in VCA/VCM, by setting the MODE bits to '10' in SXCON register.

In the Packet Wire mode the USART can be configured either as receiver or transmitter but it cannot be receiver and transmitter simultaneously.

5.4.1 Reception

The Packet Wire reception mode provided by the USART is compatible with the serial transmission protocol implemented in the PTD (cf. RD-2) and the VCA (cf. RD-3) devices. In this configuration SXDT is an input that corresponds to the data signal, SXVAL is an input signal that corresponds to the validation signal and SXCLK is an input signal that corresponds to the serial clock signal.

While the SXVAL signal is not asserted, data on the SXDT serial line are ignored. When the validation signal is set, data on the SXDT serial line are shifted, most significant bit first, into the Receive register on the rising SXCLK edge. When 8 bits have been shifted to the Receive register they are written to the Hold Receive where they can be read by software.

The RI bit of the SXCON register is set as soon as the received data is transferred to the Hold Receive register. It is used for informing the user that data has been received and can be read from the SXBUF register. The setting of the RI bit can generate an interrupt (cf. § 6). The RI bit is reset when the Hold Receive register has been read or by writing directly to the RI bit in the SXCON register. If new data is received while the Hold Receive register is full, the old data is overwritten.

The end of a block reception is detected when the SXVAL signals goes to logic level '0'. The EOR1 bit in the SXCON register is set and an interrupt can be generated (cf. § 6). This bit is reset by writing directly the EOR bits in the SXCON register (cf. §5.2).

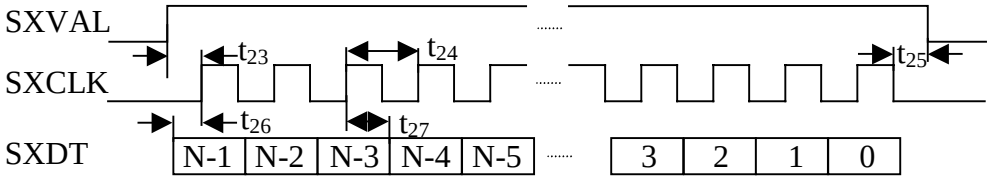


Figure 5-6: Synchronous Packet Wire reception

T_{Off}

in

5.4.2 Transmission

In transmission mode, SXDT is an output that corresponds to the data signal, SXVAL is an output that corresponds to the validation signal and SXCLK is an output that corresponds to the serial clock signal.

The data transmission is validated with the SXVAL signal that is controlled via the EOR bit in the SXCON register. By setting this bit to '1', the software indicates the beginning of a block transfer and enables the transmission. By setting this bit to '0', the software indicates the end of a block transfer and disables the transmission.

When the transmission is enabled, the SW has to write the successive data to be sent on the serial line in the SXBUF register. In Packet Wire mode, the USART works on 8 bit data, so only the least significant byte of SXBUF (e.g. SXBUFL) is used. The writing rate is controlled by the TI bit that indicates the availability of the Transmit Hold register and works in the same manner than for RS232 type transmission.

The data to be transmitted is shifted on the SXDT line, most significant bit first, in conjunction with the SXCLK signal at a frequency which can be programmed with the SXFREQ register(cf §5.1) and which is equal to the Clock/16*SXFREQ (where SXFREQ = 0 ⇔ SXFREQ = 2^{16}). During a bit period, SXCLK is first at logic level '0' for the first half of the period and then at logic level '1' for the second half of the period. The receiver can then use the rising SXCLK edges to store the data received onto SXDT into its register.

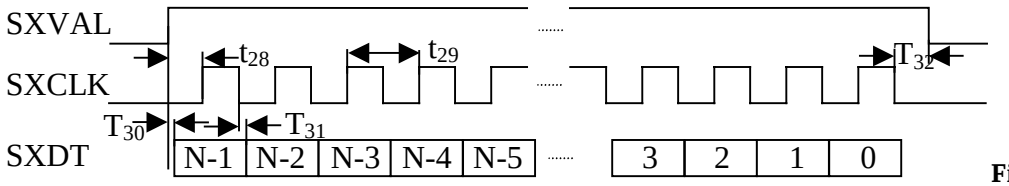


Figure 5-7: Synchronous Packet Wire transmission

T_{On} T_{Off}

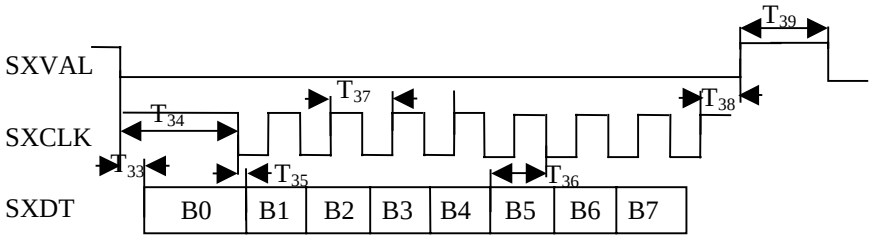
5.5 Synchronous TTC-B-01 mode

The USART is configured in synchronous TTC-B-01 mode, by setting the MODE bits to '11' in SXCON register.

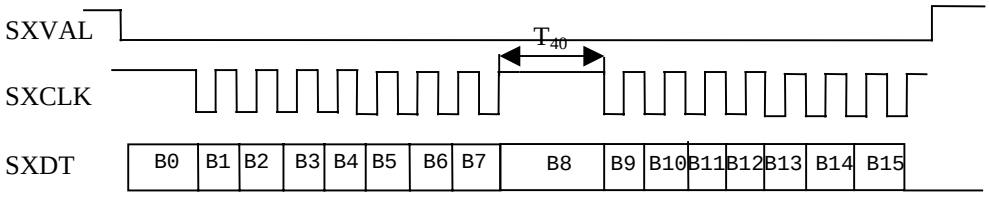
In this mode the USART TTC-B-01 is compatible with the Digital Serial and Memory Load command transmissions described in the TTC-B-01 standard with the restriction that the clock is only active during a transmission / reception but is maintained at logic level '0' or '1' outside this region.

In the TTC-B-01 mode the USART can be configured either as receiver or transmitter but it cannot be a receiver and a transmitter simultaneously. In addition the mode extension bit enables to program the USART as master for the transfer (e.g. the USART generates the clock and validation signals) or slave, and the size (8 bits or 16 bits) of the data word to be transferred.

TTC-B-01 like transmission is presented in the figure hereafter:



8bit data transfer



16 bit data transfer

Figure 5-8: TTC-B-01 transfers

5.5.1 Transmission

In transmission mode, the USART can be configured as communication master or slave. The SXDT signal is an output that correspond to the data signals while SXCLK corresponds to the transmission clock signal and SXVAL corresponds to the validity signal. When the USART is configured as the communication master, the transmission clock and validity signals are generated by the USART and SXCLK and SXVAL are output signals. In the other case, the USART shall use an external clock and validity signal to rhythm the transmission and SXCLK and SXVAL are input signals.

As in the two other modes, the data to be transmitted shall be loaded by SW into the SXBUF register. For 8 bit transfers (ME1 = '0') the data has to be written in the SXBUFL Special Function Register. For 16-bit transfer (ME1 = '1'), the most significant byte shall be first written into SXBUFH and the least significant byte shall then be written into SXBUFL.

In master mode, the transmission is started as soon as the data has been written into the register and if the transmission of the previous data has been terminated. By design a delay corresponding to half the bit transmission period is inserted between two transmissions. The data transmission rate is programmable in the SXFREQ register(cf. §5.1) according to the formula: transmission rate = F_{Clock}/16*SXFREQ bit/s where F_{Clock} represent the clock frequency (SXFREQ = 0 ⇔ SXFREQ = 2^{16}). The most significant bit is output first at the same time where SXVAL is set to '0'. Then transfer of the 7/15 remaining bit is performed as presented in Figure 5-8..

In slave mode, the transmission is under control of external signals. The most significant bit of the data to be transmitted is shifted onto the SXDT line as soon as the SXVAL line has been detected at logic level '0'. Then the USART detects the falling SXCLK edges to shift the data to be transmitted onto the SXDT line.

The TI bit in the SXCON register is managed in the same way as for RS232 and Packet Wire transmission. TI is set when the data contained in the Hold Transmit register is passed to the Transmit register for transmission on the line. It informs the user that the Hold Transmit register is empty and that it can write new data in the register. The setting of the TI bit can generate an interrupt (cf. § 6).

The TI bit is reset either when new data is written in the Hold Transmit register or directly by writing this bit in the register. If new data is written in the Hold Transmit register while the previous data has not already been loaded in the Transmit register, the old data present into the Hold Transmit register is overwritten.

5.5.2 Reception

In reception mode, the USART can be configured as communication master or slave. The SXDT signal is an input that corresponds to the data signals while SXCLK corresponds to the transmission clock signal and SXVAL corresponds to the validity signal. When the USART is configured as the communication master, the transmission clock and validity signals are generated by the USART and SXCLK and SXVAL are output signals. In the other case, the USART shall use an external clock and validity signal to rhythm the transmission and SXCLK and SXVAL are input signals.

In master mode, the beginning of the transmission is commanded by setting the BOT bit in the SXCON register which asserts the SXVAL signal to logic level '0' for 8,5 bit (or 16,5 bit when 16 bits have to be received) transmission period. Then the SXCLK signal generation is performed as presented in Figure 5-8. SXCLK is only generated for the duration of the reception and it is maintained at logic level '1' outside transmission or reception periods. Data received on the SXDT signal are shifted to the internal Receive register on the falling SXCLK edge.

In slave mode, the USART is waiting for the SXVAL assertion to start the data reception. The data on the SXDT signal are shifted in the internal Receive register on the falling SXCLK edge.

The RI bit in the SXCON register is set as soon as the received data is transferred into the Hold Receive register. It is used to inform the user that data has been received and can be read from the SXBUF register. The setting of the RI bit can generate an interrupt(cf. § 6). The RI bit is reset when the Hold Receive register has been read or by writing directly the RI bit in the SXCON register. If new data is received while the Hold Receive register is full, old data is overwritten.

5.6 Transmission and reception FIFOs.

The 80S32 implements two 64 bytes FIFO blocks which can be used in conjunction with one of the four extra USARTS during transmission and reception. The reception FIFO is used as a buffer to store received data from a selected USART and the transmission FIFO is used as a buffer to store data to be transferred by a selected USART. The selection of the USART that shall be used with the FIFOs is performed via the FIFOCON register presented in the figure hereafter.

SFR @	(MSB)		(LSB)
$0C7_H$	TFIFO	RFIFO	

Symbol	Significance in write mode	Significance in read mode
TFIFO	Transmitting FIFO configuration: • '0XXX': no change • '100X': disable transmitting FIFO, • '101X' : FIFOs in test mode[1] • '1100': connect transmitting FIFO to extra USART 1 • '1101': connect transmitting FIFO to extra USART 2 • '1110': connect transmitting FIFO to extra USART 3 • '1111': connect transmitting FIFO to extra USART 4	Transmitting FIFO configuration: • '1XXX': not used • '001X': not used, • '00X1': not used • '0000': FIFO not used, • '0100': transmitting FIFO connected to extra USART 1 • '0101': transmitting FIFO connected to extra USART 2 • '0110': transmitting FIFO connected to extra USART 3 • '0111': transmitting FIFO connected to extra USART 4
RFIFO	Receiving FIFO configuration: • '0XXX': no change • '10XX': disable receiving FIFO, • '1100': connect receiving FIFO to extra USART 1 • '1101': connect receiving FIFO to extra USART 2 • '1110': connect receiving FIFO to extra USART 3 • '1111': connect receiving FIFO to extra USART 4	Receiving FIFO configuration: • '1XXX': not used • 'X1XX': not used, • '0000': FIFO not used, • '0100': Receiving FIFO connected to extra USART 1 • '0101': Receiving FIFO connected to extra USART 2 • '0110': Receiving FIFO connected to extra USART 3 • '0111': Receiving FIFO connected to extra USART 4

[1] Test mode activation automatically disable receiving and transmitting FIFO

Figure 5-9: FIFOCON register

When the receiving FIFO is used, the RI bit of the extra USART connected to this FIFO is set when the FIFO is not empty and indicates that at least one data is available.

When the transmitting FIFO is used, the TI bit of the extra USART connected to this FIFO is set when the FIFO is not full and indicates that data can be loaded into the FIFO.

Data are read from the receiving FIFO and loaded into the transmitting FIFO via the SXBUFL and SXBUFH registers in the same way than in normal mode (see 5.1). In particular, for 16 bits transfer the constraint to always access SXBUFH prior to SXBUFL shall be respected in order not to loose data or to send wrong data.

6. INTERRUPT CONTROLLER

The 80S32 microcontroller manages the interrupt as the original 8052/51 but has extended capabilities to process 15 interrupts, the 6 of the original 8052/51 plus 9 related to its specific needs. The 15 interrupts are presented in the table hereafter:

Name	Description	IT N°
IE0	External interrupt 0	0
TF0	Timer 0 interrupt	1
IE1	External interrupt 1	2
TF1	Timer 1 interrupt	3
RI+TI	Serial interrupt	4
TF2	Timer 2 interrupt	5
EXMEMER	External memory error interrupt	6
SX1	Extra UART1 interrupt	7
INMEMER	Internal memory error interrupt	8
SX2	Extra UART2 interrupt	9
IE2	External interrupt 2	10
SX3	Extra UART3 interrupt	11
IE3	External interrupt 3	12
SX4	Extra UART4 interrupt	13
IE4	External interrupt 4	14

In the 80S32, as in the original 8052/51, one or several flags are associated to each interrupt and are polled each clock cycle (instead of each 12 clock cycles in the original component) in the order given in the table here-below by the interrupt controller.

At each clock cycle the interrupt controller determines if an interrupt has to be served or not according to the following procedure:

- is the general interrupt flag IE set? If no, then no interrupt can be served ;
- is there any high level interrupt currently in process? If yes, then no other interrupt can be served ;
- polls all interrupt flags relative to the interrupt enabled with a high priority level in the order given in the table. When one of the interrupt flag is found active, then the corresponding interrupt is served and the polling process is exited ;
- is there any low level interrupt currently in process? If yes, then no other interrupt can be served ;
- polls all interrupt flags relative to the interrupt enabled with a low priority level in the order given in the table. When one of the interrupt flag is found active, then the corresponding interrupt is served and the polling process is exited.

To serve an interrupt, the controller generates an LCALL to the appropriate location in Program Memory. The location is calculated with the formula (8*IT N°) + 3. The beginning of the interrupt processing procedure shall be stored at this address. The interrupt is considered in process as long as the interrupt processing procedure is not terminated which is indicated by the execution of a RETI instruction.

The authorization/inhibition as well as the priority of the 9 added interrupt is performed via the IXEP1, IXEP2 and IXEP3 registers which are presented in the figures hereafter:

Sfr @	(MSB)				(LSB)
$0AC_H$	-	-	EEXM	EINM	

Symbol	Significance in write mode	Significance in read mode
EINM	Internal memory error interrupt authorization command: • '10': enables with low priority • '11': enables with high priority • '01': disables interrupt • '00': no change	Internal memory error interrupt authorization status: • '10': enabled with low priority • '11': enabled with high priority • '0X': interrupt disabled
EEXM	External memory error interrupt authorization command: • '10': enables with low priority • '11': enables with high priority • '01': disables interrupt • '00': no change	External memory error interrupt authorization status: • '10': enabled with low priority • '11': enabled with high priority • '0X': interrupt disabled

Figure 6-1: IXEP1 register

Sfr @	(MSB)			(LSB)
$0AD_H$	ESX1	ESX2	ESX3	ESX4

Symbol	Significance in write mode	Significance in read mode
ESX1	Extra UART1 interrupt authorization command: • '10': enables with low priority • '11': enables with high priority • '01': disables interrupt • '00': no change	Extra UART1 interrupt authorization status: • '10': enabled with low priority • '11': enabled with high priority • '0X': interrupt disabled
ESX2	Extra UART2 interrupt authorization command: • '10': enables with low priority • '11': enables with high priority • '01': disables interrupt • '00': no change	Extra UART2 interrupt authorization status: • '10': enabled with low priority • '11': enabled with high priority • '0X': interrupt disabled
ESX3	Extra UART3 interrupt authorization command: • '10': enables with low priority • '11': enables with high priority • '01': disables interrupt • '00': no change	Extra UART3 interrupt authorization status: • '10': enabled with low priority • '11': enabled with high priority • '0X': interrupt disabled
ESX4	Extra UART4 interrupt authorization command: • '10': enables with low priority • '11': enables with high priority • '01': disables interrupt • '00': no change	Extra UART4 interrupt authorization status: • '10': enabled with low priority • '11': enabled with high priority • '0X': interrupt disabled

Figure 6-2: IXEP2 register

Sfr @	(MSB)			(LSB)
$0AE_H$	-	EX4	EX3	EX2

Symbol	Significance in write mode	Significance in read mode
EX2	External interrupt2 authorization command: • '10': enables with low priority • '11': enables with high priority • '01': disables interrupt • '00': no change	External interrupt2 authorization status: • '10': enabled with low priority • '11': enabled with high priority • '0X': interrupt disabled
EX 3	External interrupt3 authorization command: • '10': enables with low priority • '11': enables with high priority • '01': disables interrupt • '00': no change	External interrupt3 authorization status: • '10': enabled with low priority • '11': enabled with high priority • '0X': interrupt disabled
EX 4	External interrupt4 authorization command: • '10': enables with low priority • '11': enables with high priority • '01': disables interrupt • '00': no change	External interrupt4 authorization status: • '10': enabled with low priority • '11': enabled with high priority • '0X': interrupt disabled

Figure 6-3: IXEP3 register

The 3 additional external interrupts can be programmed to be level activated or transition activated by setting or clearing bit IT4, IT3 or IT2 in the TXCON register. If Itx = 0, external interrupt x is triggered by a detected low at the INTx pin. If ITx = 1, external interrupt x is edge triggered and set on a high to low transition at the INTx pin.

External interrupt 2 and 3 are flagged into bits IE2, IE3 and IE4 of the TXCON register. A '1' indicates than a valid interrupt condition has been detected. As for the original 8052, the 3 bits IE2, IE3 and IE4 can be set/reset by software that gives the possibility to generate interrupts by software.

In addition the active level of the 5 external interrupts can also be programmed via the AL bits in the TXCON registers.

External interrupt timings are internally double-resynchronized. An external interrupt shall have a minimum duration of one system clock period + 10 (TBC) ns.

The TXCON registers are presented in the figure hereafter:

Sfr @	(MSB)				(LSB)
$0BA_H$	AL0	IT4	IT3	IT2	

Symbol	Significance in write mode	Significance in read mode
IT2	External interrupt 2 type selection: • '10': level active • '11': edge active • '0X': no change	External interrupt 2 type: • '00': level active • '01': edge active • '1X': not used
IT3	External interrupt 3 type selection: • '10': level active • '11': edge active • '0X': no change	External interrupt 3 type: • '00': level active • '01': edge active • '1X': not used
IT4	External interrupt 4 type selection: • '10': level active • '11': edge active • '0X': no change	External interrupt 4 type: • '00': level active • '01': edge active • '1X': not used
AL0	Interrupt 0 active level selection: • '10': low level/falling edge active • '11': high level/rising edge active • '0X': no change	Interrupt 0 active level: • '10': low level/falling edge active • '11': high level/rising edge active • '1X': not used

Figure 6-4: TXCON3 register

Sfr @	(MSB)			(LSB)
$0BB_H$	AL4	AL3	AL2	AL1

Symbol	Significance in write mode	Significance in read mode
AL1	Interrupt 1 active level selection: • '10': low level/falling edge active • '11': high level/rising edge active • '0X': no change	Interrupt 1 active level: • '10': low level/falling edge active • '11': high level/rising edge active • '0X': not used
AL2	Interrupt 2 active level selection: • '10': low level/falling edge active • '11': high level/rising edge active • '0X': no change	Interrupt 2 active level: • '10': low level/falling edge active • '11': high level/rising edge active • '0X': not used
AL3	Interrupt 3 active level selection: • '10': low level/falling edge active • '11': high level/rising edge active • '0X': no change	Interrupt 3 active level: • '10': low level/falling edge active • '11': high level/rising edge active • '0X': not used
AL4	Interrupt 4 active level selection: • '10': low level/falling edge active • '11': high level/rising edge active • '0X': no change	Interrupt 4 active level: • '10': low level/falling edge active • '11': high level/rising edge active • '0X': not used

Figure 6-5: TXCON2 register

Sfr @	(MSB)			(LSB)
$0BC_H$	-	IE4	IE3	IE2

Symbol	Significance in write mode	Significance in read mode
IE2	External interrupt 2 flag setting: • '10': reset flag, • '11': set flag, • '0X': no change.	External interrupt 2 flag: • '00': interrupt not active • '01': interrupt active • '1X': not used
IE3	External interrupt 3 flag setting: • '10': reset flag, • '11': set flag, • '0X': no change.	External interrupt 3 flag: • '00': interrupt not active • '01': interrupt active • '1X': not used
IE4	External interrupt 4 flag setting: • '10': reset flag, • '11': set flag, • '0X': no change.	External interrupt 4 flag: • '00': interrupt not active • '01': interrupt active • '1X': not used

Figure 6-6: TXCON1 register

The External Memory Error interrupts is used to indicate that an erroneous data has been read during an external program and data access. The interrupt flags relating to this interruption are the PG_ER_FLAG and DT_ER_FLAG bits in the MPSTAT register. These bit are automatically set when an error condition is detected during an access or can also be set by software. The resetting of these flags has to be performed in the interrupt processing routine. If this is not done then a new interrupt will be generated after the end of the interrupt processing routine execution.

The Internal Memory Error interrupts is used to indicate that an erroneous data has been read during a data access to the internal or the eXtra memory. The interrupt flags that are relating to this interruption are the XMER and IMER bits in the MPSTAT register. These bit are automatically set when an error condition is detected during an access or can also be set by software. The resetting of these flags has to be performed in the interrupt processing routine. If this is not done then a new interrupt will be generated after the end of the interrupt processing routine execution.

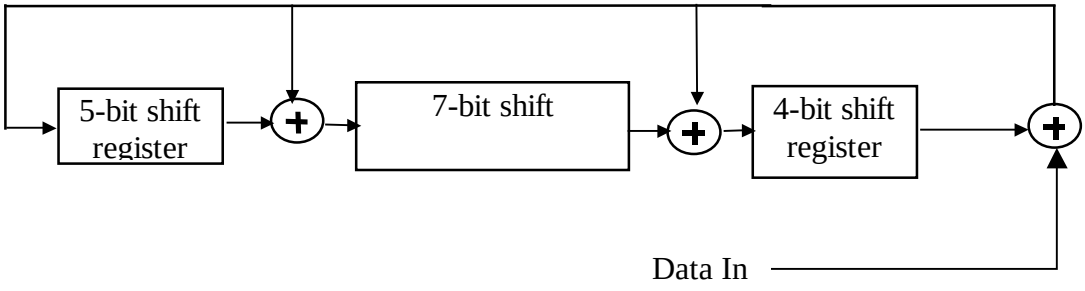
The 4 extra USART interrupts are used to indicate for each USART that it is ready to send a data or that it has received a valid data that can be retrieved from the internal register. The interrupt flags that are relating to these interruptions are the RI, TI and optionally EOR bits in the SXxCON registers. These bit are automatically set when an error condition is detected during an access or can also be set by software. The resetting of these flags has to be performed in the interrupt processing routine. If this is not done then a new interrupt will be generated after the end of the interrupt processing routine execution.

7. CRC ACCELERATOR UNIT

The 80S32 provides a Cyclic Redundancy Code (CRC) acceleration unit, which accelerates the CRC calculation with the polynomial:

$$G(X) = X^{16} + X^{12} + X^5 + 1.$$

The calculation is implemented according to the galois structure presented hereafter :



The unit uses three Special Function Registers:

- CRCIN shall be used by the application to load the element of the message for which the CRC has to be calculated / verified ;
- CRCH contains the most significant byte of the calculation result ;
- CRCL contains the least significant byte of the calculation result.

Calculation of the CRC of a message B1 B2.... Bx where B represents a byte is as follow:

- CRCH and CRCL shall be first initialized with the initial calculation value (this value is 'FF' for most ESA applications) ;
- the successive bytes of the message are written, starting from the most significant one, into the CRCIN register ;
- finally the CRC result is obtained by reading the CRCH and CRCL registers.

For example, it is also possible to verify the validity of a message B1 B2.... Bx protected by the code 'C0' 'C1' as follow:

- CRCH and CRCL shall be first initialized with the initial calculation value (this value is 'FF' for most ESA applications) ;
- the successive bytes of the message are written, starting from the most significant one, into the CRCIN register ;
- 'C0' and 'C1' are finally written into CRCIN. If the message corresponds to the code then CRCH and CRCL register shall be 0.

8. I/O PORTS

The 80S32 contains four 8-bit parallel input/output ports. Each port bit is configurable individually as an input, an output or to be used as an alternate function.

Four special function registers EXTAD, P1CON, P2CON and P3CON allow the user to configure each port bit as a bi-directional port or as an alternate function. Each bit of the ExtAD/PxCON register configures the corresponding bit on Px port with the following correspondence:

- 0: port bit configured as a bi-directional port ;
- 1: port bit configured as an alternate function (default configuration).

Four special function registers P0DIR, P1DIR, P2DIR and P3DIR allow to configure each port bit as an input or an output when it has been configured as a bi-directional port. Each bit of the PxDIR register configures the corresponding bit on Px port with the following correspondence:

- 0: port bit configured as an input ;
- 1: port bit configured as an output (default value).

For example, the bit 2 in the P3 port is configured as an input if P3CON = 'XXXXX1XX' and P3DIR = 'XXXXX0XX'.

Table of the alternate functions is given in § 9.2.

9. INTERFACE AND SIGNAL DESCRIPTION

The 80S32 has 65 interface signals that comprise 2 input signals, 19 output signals and 44 bi-directional signals.

9.1 Interface signals

Name	Input/Output	Description
Clock	I	Main system clock
Reset_n	I	Active low reset
Adr[15:0]	O	Address bus for external memory access
Data[7:0]	I/O	Bi-directional data bus for external memory access
PSE_n	O	External ROM read signal (active low)
Rd_n	O	External RAM read signal (active low)
Wr_n	O	External RAM write signal (active low)
EA	I/O	External Access : after reset EA is used to determine if program shall be executed from the internal bootstrap (EA = '1') or from external ROM (EA = '0'). After program downloading EA is configured as an output at logic level '0'. EA shall be pull to VSS or VCC through a 10 Kohms resistor.
Sx3dt	I/O	Serial port 3 data line
Sx3val	I/O	Serial port 3 data or validity line
Sx3clk	I/O	Serial port 3 clock line
P0[7:0]	I/O	Parallel port or alternative function (cf. following table)
P1[7:0]	I/O	Parallel port or alternative function (cf. following table)
P2[7:0]	I/O	Parallel port or alternative function (cf. following table)
P3[7:0]	I/O	Parallel port or alternative function (cf. following table)

9.2 Alternative port functions

Most of the parallel port pins are shared with an alternative function as listed hereafter :

Name	Port	Description
CSA_0	P0[0]	Chip select 0 for RAM1 space.
EXTAD22	P0[1]	Chip select 1 for RAM1 space or bit 22 of the address bus in case of address expansion
EXTAD21	P0[2]	Chip select 2 for RAM1 space or bit 21 of the address bus in case of address expansion
EXTAD20	P0[3]	Chip select 0 for RAM2 space or bit 20 of the address bus in case of address expansion
EXTAD19	P0[4]	Chip select 1 for RAM2 space or bit 19 of the address bus in case of address expansion
EXTAD18	P0[5]	Chip select 2 for RAM2 space or bit 18 of the address bus in case of address expansion
EXTAD17	P0[6]	bit 17 of the address bus for address expansion
EXTAD16	P0[7]	bit 16 of the address bus for address expansion
T2EX	P1[0]	Timer 2 external input
T2CAPT	P1[1]	Timer 2 capture signal
SX2DT	P1[3]	Serial port 2 data line
SX2VAL	P1[4]	Serial port 2 data or validity line
SX2CLK	P1[5]	Serial port 2 clock line
TIMER0_IRQ	P1[6]	Timer0 interrupt(cf. RD1)
TIMER1_IRQ	P1[7]	Timer1 interrupt(cf. RD1)
TIMER2_IRQ	P2[0]	Timer2 interrupt(cf. RD1)
SX1DT	P2[5]	Serial port 1 data line
SX1VAL	P2[6]	Serial port 1 data or validity line
SX1CLK	P2[7]	Serial port 1 clock line
Rxd	P3[0]	Serial port 0 data input
Txd	P3[1]	Serial port 0 data output
T0EX	P3[4]	Timer 0 external input
T1EX	P3[5]	Timer 1 external input
INT4	P3[7]	External interrupt 4 input, active low, configurable as rising/falling edge sensitive or high/low level sensitive

Alternate functions on ports 1, 2 and 3 are selected in registers P1CON, P2CON and P3CON. Alternate functions on port 0 are selected in register ExtMem.

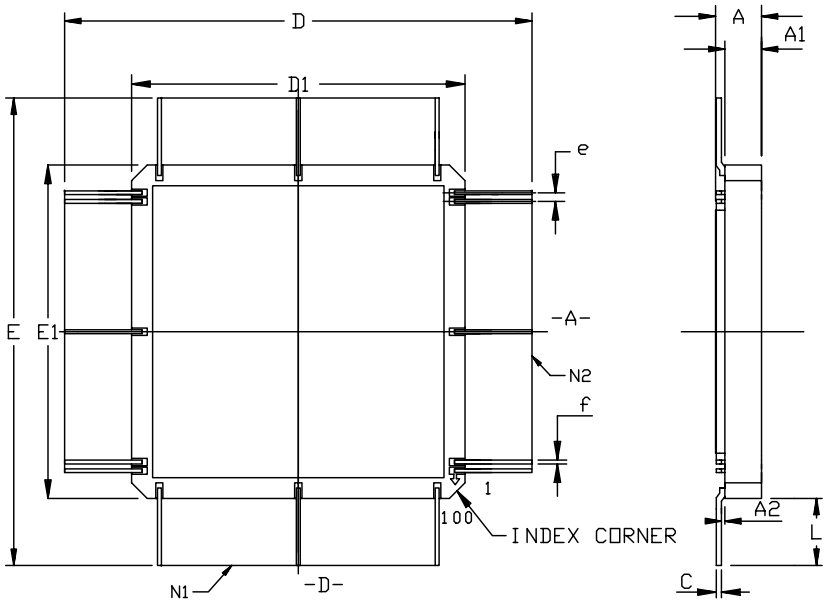
10. PACKAGES DETAILS

10.1 Dimensions

TEMIC Semiconductors

Mechanical Data

100 LEAD MQFP FLAT



	Min	Max	Min	Max
A	2.21	2.67	.087	.105
C	0.15	0.20	.006	.008
D	33.80	35.30	1.331	1.390
D1	18.80	19.30	.740	.760
E	33.80	35.30	1.331	1.390
E1	18.80	19.30	.740	.760
e	0.635 BSC		.025 BSC	
f	0.254 REF		.010 REF	
A1	1.83	2.24	.072	.088
A2	0.203 REF		.008 REF	
L	7.50	8.00	.295	.315
N1	25		25	
N2	25		25	

PACKAGE CODE : U68
INTERNAL CODE : KU
MHS S.A.

REV : A DATE : 01-03-00

10.2 Pin Assignement

signal	pin	Type	signal	pin	type	signal	pin	type
VSSA1	1	VSSA	P02_EXTAD21	35	BIOT3	SX3DT	69	BIOT3UP
VSSB1	2	VSSB	P03_EXTAD20	36	BIOT3	SX3VAL	70	BIOT3
ADR0	3	BOUT3	P04_EXTAD19	37	BIOT3	P31_TXD	71	BIOT3UP
ADR1	4	BOUT3	P05_EXTAD18	38	BIOT3	P32	72	BIOT3UP
ADR2	5	BOUT3	P06_EXTAD17	39	BIOT3	P33	73	BIOT3UP
ADR3	6	BOUT3	P07_EXTAD16	40	BIOT3	P34_T0EX	74	BIOT3UP
ADR4	7	BOUT3	VCCB2	41	VCCB	P35_T1EX	75	BIOT3UP
ADR5	8	BOUT3	VSSB3	42	VSSB	P36	76	BIOT3UP
ADR6	9	BOUT3	P10_T2EX	43	BIOT3UP	P37_INT4	77	BIOT3UP
ADR7	10	BOUT3	P11_T2CAPT	44	BIOT3UP	VCCB4	78	VCCB
ADR8	11	BOUT3	P12	45	BIOT3UP	VCCA2	79	VCCA
ADR9	12	BOUT3	P13_SX2DT	46	BIOT3UP	VSSB5	80	VSSB
ADR10	13	BOUT3	P14_SX2VAL	47	BIOT3	CLK	81	BINTTL
ADR11	14	BOUT3	P15_SX2CLK	48	BIOT3UP	SX3CLK	82	BIOT3UP
ADR12	15	BOUT3	P16_TIMER0_IRQ	49	BIOT3UP	INT2	83	BINTTL
ADR13	16	BOUT3	P17_TIMER1_IRQ	50	BIOT3UP	DATA0	84	BIOT3UP
ADR14	17	BOUT3	VCCB3	51	VCCB	DATA1	85	BIOT3UP
ADR15	18	BOUT3	VSSA2	52	VSSA	DATA2	86	BIOT3UP
EXTAD_23	19	BOUT3	VSSB4	53	VSSB	DATA3	87	BIOT3UP
CSA_0	20	BOUT3	INT0	54	BINTTL	DATA4	88	BIOT3UP
CSA_1	21	BOUT3	P20_TIMER2_IRQ	55	BIOT3UP	DATA5	89	BIOT3UP
CSA_2	22	BOUT3	P21	56	BIOT3UP	DATA6	90	BIOT3UP
VCCB1	23	VCCB	P22	57	BIOT3UP	DATA7	91	BIOT3UP
VSSB2	24	VSSB	P23	58	BIOT3UP	INT3	92	BINTTL
PSE_N	25	BOUT3	P24	59	BIOT3UP	VCCB5	93	VCCB
RD_N	26	BOUT3	P25_SX1DT	60	BIOT3UP	VCCA3	94	VCCA
WR_N	27	BOUT3	P26_SX1VAL	61	BIOT3	INT1	95	BINTTL
EA	28	BIOT3UP	P27_SX1CLK	62	BIOT3UP	MODE_TEST[1]	96	BINTTL
VCCA1	29	VCCA	SX4DT	63	BIOT3UP	CODE0_BUSCTL0[2]	97	BIOT3UP
CSB_0	30	BOUT3	SX4VAL	64	BIOT3	CODE1_BUSCTL1[2]	98	BIOT3UP
CSB_1	31	BOUT3	SX4CLK	65	BIOT3UP	SCANEN_BUSCTL2[2]	99	BIOT3UP

CSB_2	32	BOUT3	P30_RXD	66	BIOT3UP	FALLCLK_BUSCTL3[2]	100	BIOT3UP
P00	33	BIOT3UP	VSSA3	67	VSSA			
P01_EXTAD22	34	BIOT3	RST	68	BINTTL			

[1] Test input. Shall be pull to VSS, [2] output to be left floating.

11. ELECTRICAL CHARACTERISTICS

11.1 Maximum ratings

Parameter	Min	Max	Unit
Supply Voltage	-0.5	7.0	V
Input Voltage	Vss - 0.3	Vcc + 0.3	V
Operating Temperature	-55	+125	°C
Storage Temperature	-65	+150	°C

11.2 Recommended operating conditions

Symbol	Parameter	Min	Typ.	Max	Unit
Vcc	Supply Voltage	4.5	5.0	5.5	V
VIH	Input High Voltage	2.2	-	Vcc	V
VIL	Input Low Voltage	-0.3	0	0.8	V
T_{Clock}	Clock period	50			ns
	Duty cycle			45	%

11.3 DC Parameters (min,max values).

Symbol	Parameter	Condition	Min	Typ.	Max	Unit
VOH	Output high voltage	Vcc = 4.5 v, I_{OH} = 6 or 3 mA	3.9	-	-	V
VOL	Output low voltage	Vcc = 4.5 v, Iol = -6 or 3 mA	-	-	0.4	V
IIX	Input leakage current		-10.0		+10.0	µA

11.4 AC Characteristics

Symbol	Parameter	Min.	Typ.	Max	Unit
t1	Clock rising edge to Pse_n/Rd_n low time for program memory accesses			19	ns
t2	Clock rising edge to Address valid time for program memory accesses			28	ns
t3	Clock rising edge to Pse_n/Rd_n high time for program memory accesses			18	ns
t4	Clock rising edge to Address invalid time for program memory accesses			28	ns
t5	Data setup time for non-protected program memory accesses	2			ns
t6	Data hold time for non-protected program memory accesses	1			ns
t7	Clock falling edge to ExtAd23 high time			16	ns
t8	Data setup time for protected program memory accesses	8			ns
t9	Data hold time for protected program memory accesses	0			ns
t10	Protection code setup time for protected program memory accesses	1			ns
t11	Protection code hold time for protected program memory accesses	0			ns
t12	Clock rising edge to Rd_n low time for data memory accesses			19	ns
t13	Clock rising edge to Address valid time for data memory accesses			27	ns
t14	Clock rising edge to Rd_n high time for data memory accesses			18	ns
t15	Clock rising edge to Address invalid time for data memory accesses			27	ns
t16	Data setup time for non-protected data memory accesses	2			ns
t17	Data hold time for non-protected data memory accesses	0			ns
t18	Data setup time for protected date memory accesses	6			ns

t19	Data hold time for protected data memory accesses	0			ns
t20	Protection code setup time for protected data memory accesses	1			ns
t21	Clock rising edge to Wr_n low time for data memory accesses			19	ns
t22	Clock rising edge to Wr_n high time for data memory accesses			19	ns

PacketWire transfer timings :

T23	SXVAL assertion to SXCLK rising (reception mode)	1			T_{Clock}
T24	SXCLK period (reception mode)	4			T_{Clock}
T25	SXCLK falling to SXVAL deassertion (reception mode)	0			T_{Clock}
T26	SXDT Setup Time (reception mode)	-1			ns
T27	SXDT hold time (reception mode)	2			ns
t28	SXVAL assertion to SXCLK rising (transmission mode)		0.5 T_{trans} + 1 T_{Clock}		T_{trans}
t29 T_{trans}	SXCLK period (transmission mode)	16			T_{Clock}
t30	SXVAL assertion to SXDT valid (transmission mode)		0.5 T_{trans} + 1 T_{Clock}		
t31	SXCLK falling to SXDT valid (transmission mode)		0		T_{Clock}
t32	SXCLK falling to SXVAL deassertion (transmission mode)		0		T_{trans}

T_{trans} is the transmission period defined with the SXFREQ registers

TTC-B-01 transfer timings :

t33	SXVAL falling to SXDT valid : ➢ transmission – master mode ➢ transmission – slave mode		 1 3		T_{Clock}
t34	SXVAL falling to SXCLK falling : ➢ master mode, ➢ slave mode	 2	 7		 T_{trans} T_{Clock}
t35	SXDT setup time ➢ transmission mode ➢ reception mode	 3			 T_{Clock}

t36	SXDT hold time ➢ transmission mode ➢ reception mode	 1 1		4	 T_{Clock} T_{Clock}
t37 = Ttrans	SXCLK period ➢ master mode, ➢ slave mode	 4			 T_{Clock}
t38	SXCLK rising to SXVAL rising : ➢ master mode, ➢ slave mode	 2	 0.5		 T_{trans} T_{Clock}
t39	Minimum time between 2 transfers: ➢ master mode, ➢ slave mode	 2	 1		 T_{trans} T_{Clock}
t40	Intermedaite gap : ➢ master mode, ➢ slave mode	 2	 8,5		 T_{trans} T_{Clock}

T_{trans} is the transmission period defined with the SXFREQ registers

12. ADV 80S32 INSTRUCTION LIST

The ADV 80S32 instruction set is 100% compatible with the 805x instruction set but has an execution cycle time that is between 4 and 2 times faster than this device. In a typical application the speedup will be approximately x3.

NOTE: 8051/52 programmers may be more used to thinking in machine cycles where one machine cycle equals 12 clock cycles of the original 805x. However, for accurate timing comparison real clock cycles are used in the tables below.

12.1 Key to addressing modes

Rn	Register R0-R7 of the currently selected register bank.
Direct	8-bit internal data location address. This could be an internal, data location (0-127) or an SFR. I.E. I/O port, control register, status register etc. (128-255)).
@Ri	8-bit internal data ram location (0-255) addressed indirectly through register R0 or R1.
#data	8-bit constant included in instruction.
#data16	16-bit constant included in instruction.
addr16	16-bit destination address. Used by LCALL and LJMP. A branch can be anywhere within the 64 Kbytes address range.
addr11	11-bit destination address. Used by ACALL and AJMP. The branch will be within the same 2 Kbytes page of program memory as the first byte of the following instruction.
rel	Signed 8-bit offset byte. Used by SJMP and all conditional jumps.
bit	Direct addressed bit in internal data ram of special function register.
(n)	Clock cycle count for taken branches.

12.2 Arithmetic Operations

Opcode	Bytes	805x clocks	80S32 clocks
ADD A, Rn	1	12	4
ADD A, direct	2	12	6
ADD A, @Ri	1	12	5
ADD A, #data	2	12	6
ADDC A, Rn	1	12	4
ADDC A, direct	2	12	6
ADDC A, @Ri	1	12	5
ADDC A, #data	2	12	6
SUBB A, Rn	1	12	4
SUBB A, direct	2	12	6
SUBB A, @Ri	1	12	5
SUBB A, #data	2	12	6
INC A	1	12	3
INC Rn	1	12	4
INC direct	2	12	7
INC @Ri	1	12	5
DEC A	1	12	3
DEC Rn	1	12	4
DEC direct	2	12	7
DEC @Ri	1	12	5
INC DPTR	1	24	6
MUL AB	1	48	15
DIV AB	1	48	18
DA A	1	12	12

12.3 Logical Instructions

Opcode	Bytes	805x clocks	80S32 clocks
CLR A	1	12	3
CPL A	1	12	3
ANL A, Rn	1	12	4
ANL A, direct	2	12	6
ANL A, @Ri	1	12	5
ANL A, #data	2	12	6
ANL direct, a	2	12	7
ANL direct, #data	3	24	10
ORL A, Rn	1	12	4
ORL A, direct	2	12	6
ORL A, @Ri	1	12	5
ORL A, #data	2	12	6
ORL direct, a	2	12	7
ORL direct, #data	3	24	10
XRL A, Rn	1	12	4
XRL A, direct	2	12	6
XRL A, @Ri	1	12	5
XRL A, #data	2	12	6
XRL direct, a	2	12	7
XRL direct, #data	3	24	10
RL A	1	12	3
RLC A	1	12	3
RR A	1	12	3
RRC A	1	12	3
SWAP A	1	12	3

12.4 Data Transfer

Opcode	Bytes	805x clocks	80S32 clocks
MOV A, Rn	1	12	3
MOV A, direct	2	12	6
MOV A, @Ri	1	12	4
MOV A, #data	2	12	6
MOV Rn, a	1	12	4
MOV Rn, direct	2	24	7
MOV Rn, #data	2	12	6
MOV direct, a	2	12	6
MOV direct, Rn	2	24	6
MOV direct, direct	3	24	9
MOV direct, @Ri	2	24	6
MOV direct, #data	3	24	10
MOV @Ri, A	1	12	5
MOV @Ri, direct	2	24	6
MOV @Ri, #data	2	12	6
MOV DPTR, #data16	3	24	9
MOVC A, @A+DPTR	1	24	8
MOVC A, @A+PC	1	24	8
MOVX A, @Ri	1	24	10
MOVX A, @DPTR	1	24	10
MOVX @Ri, A	1	24	12
MOVX @DPTR, A	1	24	12
PUSH direct	2	24	7
POP direct	2	24	7
XCH A, Rn	1	12	7
XCH A, direct	2	12	7
XCH A, @Ri	1	12	7
XCHD A, @Ri	1	12	10

12.5 Boolean Variable Manipulation

Opcode	Bytes	805x clocks	80S32 clocks
CLR C	1	12	3
CLR bit	2	12	9
SETB C	1	12	3
SETB bit	2	12	9
CPL C	1	12	3
CPL bit	2	12	9
ANL C, bit	2	24	7
ANL C, /bit	2	24	7
ORL C, bit	2	24	7
ORL C, /bit	2	24	7
MOV C, bit	2	12	7
MOV bit, C	2	24	9

12.6 Program Branching

Opcode	Bytes	805x clocks	80S32 clocks
JC rel	2	24	7
JNC rel	2	24	7
JB bit, rel	3	24	12
JNB bit, rel	3	24	12
JBC bit, rel	3	24	12
JZ rel	2	24	7
JNZ rel	2	24	7
SJMP rel	2	24	7
ACALL addr11	2	24	13
LCALL addr16	3	24	15
RET	1	24	10
RETI	1	24	10
AJMP addr11	2	24	7
LJMP addr16	3	24	10
JMP @A+DPTR	1	24	7
CJNE A, direct, rel	3	24	13
CJNE A, #data, rel	3	24	13
CJNE Rn, #data, rel	3	24	9(12)
CJNE @Ri, #data, rel	3	24	9(12)
DJNZ Rn, rel	2	24	6(9)
DJNZ direct, rel	3	24	13
NOP	1	12	12*

NOTE: A 'NOP' instruction is still 12 cycles, because of its uses in software delays. Keeping the same number of cycles ensures that code running at the same clock rate on the original 805x will produce the same delay when running on the 80S32. Note that this method is not recommended, and all cases must be checked carefully.

12.7 Performance Comparison

The following table illustrates the speed improvement of the enhanced 80S32 over the original 805x, running at the same clock frequency.

Number of Opcodes	Speed Improvement
41	4.0 X
28	3.4 X
1	3.2 X
76	3.0 X
14	2.7 X
26	2.4 X
10	2.2 X
32	2.0 X
5	1.8 X
14	1.7 X
1	1.6 X
3	1.3 X
2	1.2 X
2	1.0 X
TOTAL	**WEIGHTED AVERAGE**
255	**3X**

NOTE: comparison is for devices running at the same clock frequency.