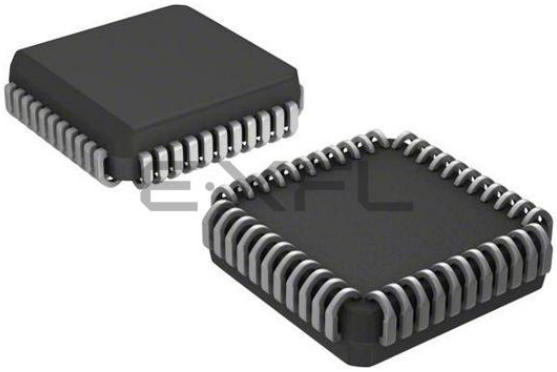


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	80C51
Core Size	8-Bit
Speed	40MHz
Connectivity	SPI, UART/USART
Peripherals	POR, PWM, WDT
Number of I/O	32
Program Memory Size	16KB (16K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	1.25K x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 5.5V
Data Converters	-
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	44-LCC (J-Lead)
Supplier Device Package	44-PLCC (16.6x16.6)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/t89c51rb2-slsim

Table 3. Pin Description for 40 - 44 Pin Packages

Mnemonic	Pin Number DIL	LCC	VQFP44 1.4	Type	Name and Function
V_{SS}	20	22	16	I	**Ground**: 0V reference
V_{CC}	40	44	38	I	**Power Supply**: This is the power supply voltage for normal, idle and power - down operation
P0.0 - P0.7	39 - 32	43 - 36	37 - 30	I/O	**Port 0**: Port 0 is an open-drain, bidirectional I/O port. Port 0 pins that have 1s written to them float and can be used as high impedance inputs. Port 0 must be polarized to V_{CC} or V_{SS} in order to prevent any parasitic current consumption. Port 0 is also the multiplexed low - order address and data bus during access to external program and data memory. In this application, it uses strong internal pull - up when emitting 1s. Port 0 also inputs the code bytes during FLASH programming. External pull - ups are required during program verification during which P0 outputs the code bytes.
P1.0 - P1.7	1 - 8	2 - 9	40 - 44 1 - 3	I/O	**Port 1**: Port 1 is an 8 - bit bidirectional I/O port with internal pull - ups. Port 1 pins that have 1s written to them are pulled high by the internal pull - ups and can be used as inputs. As inputs, Port 1 pins that are externally pulled low will source current because of the internal pull - ups. Port 1 also receives the low - order address byte during memory programming and verification. Alternate functions for T89C51RB2/RC2 Port 1 include:
	1	2	40	I/O	**P1.0**: Input / Output
				I/O	**T2 (P1.0):** Timer/Counter 2 external count input/Clockout
	2	3	41	I/O	**P1.1:** Input / Output
				I	**T2EX:** Timer/Counter 2 Reload/Capture/Direction Control
				I	$\overline{\textbf{SS}}$**:** SPI Slave Select
	3	4	42	I/O	**P1.2:** Input / Output
				I	**ECI:** External Clock for the PCA
	4	5	43	I/O	**P1.3:** Input / Output
				I/O	**CEX0:** Capture/Compare External I/O for PCA module 0
	5	6	44	I/O	**P1.4:** Input / Output
				I/O	**CEX1:** Capture/Compare External I/O for PCA module 1
	6	7	1	I/O	**P1.5:** Input / Output
				I/O	**CEX2:** Capture/Compare External I/O for PCA module 2
				I/O	**MISO:** SPI Master Input Slave Output line When SPI is in master mode, MISO receives data from the slave peripheral. When SPI is in slave mode, MISO outputs data to the master controller.
	7	8	2	I/O	**P1.6**: Input / Output
				I/O	**CEX3:** Capture/Compare External I/O for PCA module 3
				I/O	**SCK:** SPI Serial Clock SCK outputs clock to the slave peripheral
	8	9	3	I/O	**P1.7**: Input / Output:

Figure 10. PCA Timer/Counter

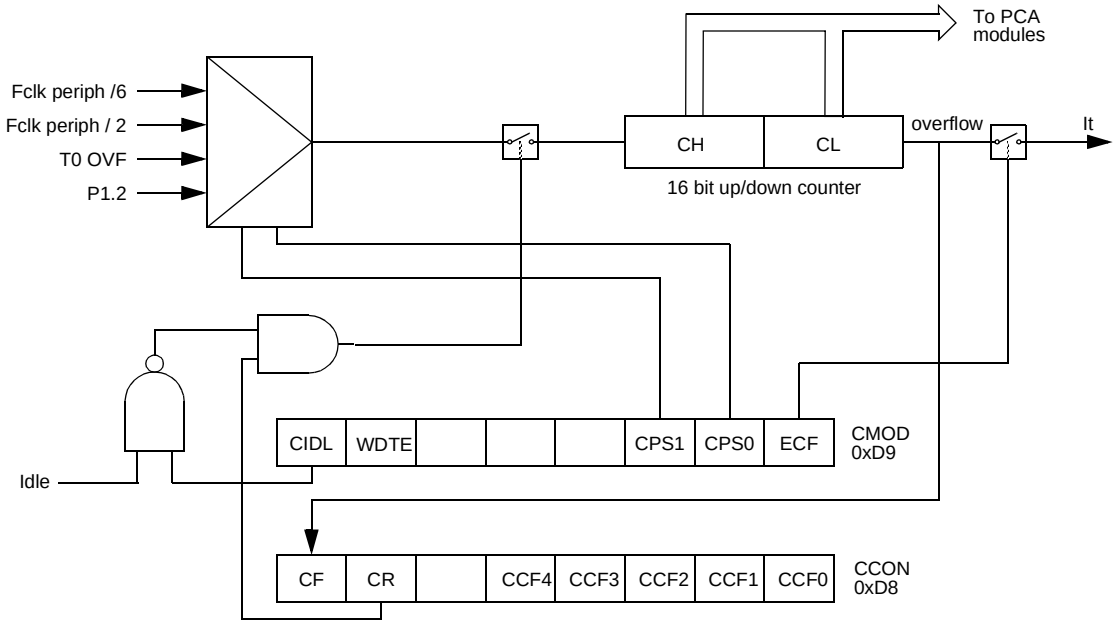


Table 14. CCON Register

CCON - PCA Counter Control Register (D8h)

7	6	5	4	3	2	1	0
CF	CR	-	CCF4	CCF3	CCF2	CCF1	CCF0

Bit Number	Bit Mnemonic	Description
7	CF	**PCA Counter Overflow flag** Set by hardware when the counter rolls over. CF flags an interrupt if bit ECF in CMOD is set. CF may be set by either hardware or software but can only be cleared by software.
6	CR	**PCA Counter Run control bit** Must be cleared by software to turn the PCA counter off. Set by software to turn the PCA counter on.
5	-	**Reserved** The value read from this bit is indeterminate. Do not set this bit.
4	CCF4	**PCA Module 4 interrupt flag** Must be cleared by software. Set by hardware when a match or capture occurs.
3	CCF3	**PCA Module 3 interrupt flag** Must be cleared by software. Set by hardware when a match or capture occurs.
2	CCF2	**PCA Module 2 interrupt flag** Must be cleared by software. Set by hardware when a match or capture occurs.
1	CCF1	**PCA Module 1 interrupt flag** Must be cleared by software. Set by hardware when a match or capture occurs.
0	CCF0	**PCA Module 0 interrupt flag** Must be cleared by software. Set by hardware when a match or capture occurs.

Reset Value = 000X 0000b
Not bit addressable

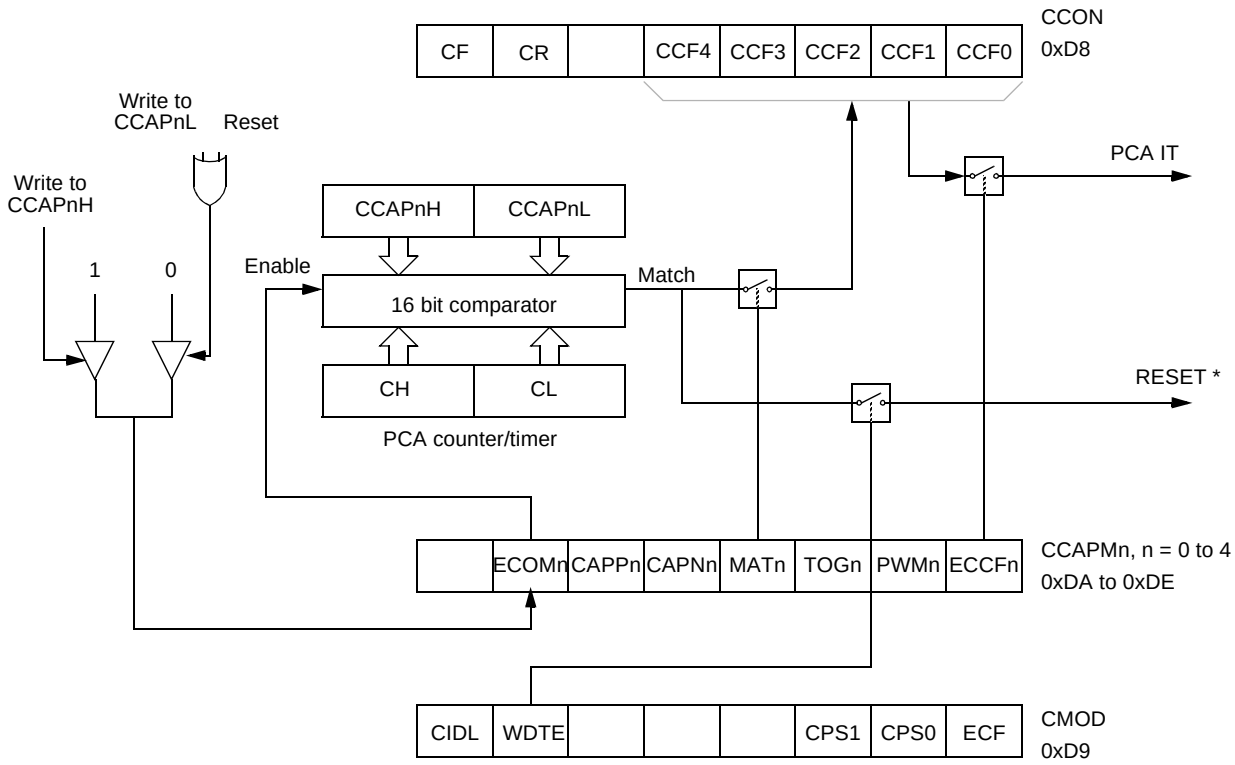
The watchdog timer function is implemented in module 4 (See Figure 13).

The PCA interrupt system is shown in Figure 11.

16-bit Software Timer/ Compare Mode

The PCA modules can be used as software timers by setting both the ECOM and MAT bits in the modules CCAPMn register. The PCA timer will be compared to the module's capture registers and when a match occurs, an interrupt will occur if the CCFn (CCON SFR) and the ECCFn (CCAPMn SFR) bits for the module are both set (See Figure 13).

Figure 13. PCA Compare Mode and PCA Watchdog Timer



* Only for Module 4

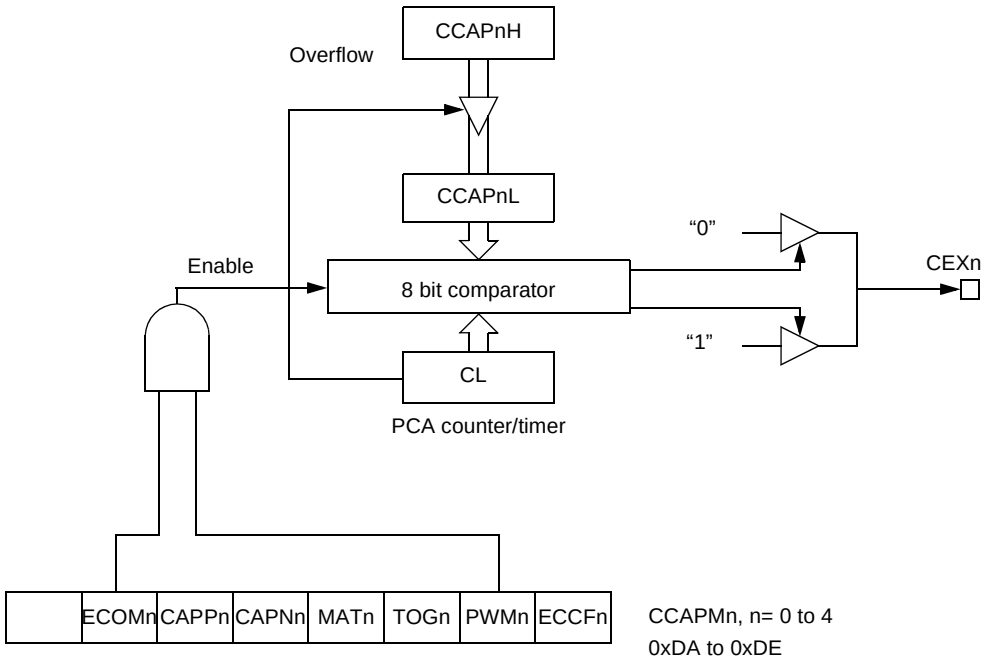
Before enabling ECOM bit, CCAPnL and CCAPnH should be set with a non zero value, otherwise an unwanted match could happen. Writing to CCAPnH will set the ECOM bit.

Once ECOM set, writing CCAPnL will clear ECOM so that an unwanted match doesn't occur while modifying the compare value. Writing to CCAPnH will set ECOM. For this reason, user software should write CCAPnL first, and then CCAPnH. Of course, the ECOM bit can still be controlled by accessing to CCAPMn register.

Pulse Width Modulator Mode

All of the PCA modules can be used as PWM outputs. Figure 15 shows the PWM function. The frequency of the output depends on the source for the PCA timer. All of the modules will have the same frequency of output because they all share the PCA timer. The duty cycle of each module is independently variable using the module's capture register CCAPLn. When the value of the PCA CL SFR is less than the value in the module's CCAPLn SFR the output will be low, when it is equal to or greater than the output will be high. When CL overflows from FF to 00, CCAPLn is reloaded with the value in CCAPHn. This allows updating the PWM without glitches. The PWM and ECOM bits in the module's CCAPMn register must be set to enable the PWM mode.

Figure 15. PCA PWM Mode



PCA Watchdog Timer

An on-board watchdog timer is available with the PCA to improve the reliability of the system without increasing chip count. Watchdog timers are useful for systems that are susceptible to noise, power glitches, or electrostatic discharge. Module 4 is the only PCA module that can be programmed as a watchdog. However, this module can still be used for other modes if the watchdog is not needed. Figure 13 shows a diagram of how the watchdog works. The user pre-loads a 16-bit value in the compare registers. Just like the other compare modes, this 16-bit value is compared to the PCA timer value. If a match is allowed to occur, an internal reset will be generated. This will not cause the RST pin to be driven high.

In order to hold off the reset, the user has three options:

- periodically change the compare value so it will never match the PCA timer,
- periodically change the PCA timer value so it will never match the compare values, or
- disable the watchdog by clearing the WDTE bit before a match occurs and then re-enable it.

The first two options are more reliable because the watchdog timer is never disabled as in option #3.If the program counter ever goes astray, a match will eventually occur and

The SADEN byte is selected so that each slave may be addressed separately.
For slave A, bit 0 (the LSB) is a don't-care bit; for slaves B and C, bit 0 is a 1.To communicate with slave A only, the master must send an address where bit 0 is clear (e. g. 1111 0000b).
For slave A, bit 1 is a 1; for slaves B and C, bit 1 is a don't care bit. To communicate with slaves B and C, but not slave A, the master must send an address with bits 0 and 1 both set (e. g. 1111 0011b).
To communicate with slaves A, B and C, the master must send an address with bit 0 set, bit 1 clear, and bit 2 clear (e. g. 1111 0001b).

Broadcast Address

A broadcast address is formed from the logical OR of the SADDR and SADEN registers with zeros defined as don't-care bits, e. g. :

SADDR0101 0110b
SADEN1111 1100b

Broadcast =SADDR OR SADEN1111 111Xb

The use of don't-care bits provides flexibility in defining the broadcast address, however in most applications, a broadcast address is FFh. The following is an example of using broadcast addresses:

Slave A:SADDR1111 0001b
SADEN1111 1010b
Broadcast1111 1X11b,

Slave B:SADDR1111 0011b
SADEN1111 1001b
Broadcast1111 1X11B,

Slave C:SADDR=1111 0010b
SADEN1111 1101b
Broadcast1111 1111b

For slaves A and B, bit 2 is a don't care bit; for slave C, bit 2 is set. To communicate with all of the slaves, the master must send an address FFh. To communicate with slaves A and B, but not slave C, the master can send and address FBh.

Reset Addresses

On reset, the SADDR and SADEN registers are initialized to 00h, i. e. the given and broadcast addresses are XXXX XXXXb (all don't-care bits). This ensures that the serial port will reply to any address, and so, that it is backwards compatible with the 80C51 microcontrollers that do not support automatic address recognition.

Table 33. BDRCON Register

BDRCON - Baud Rate Control Register (9Bh)

7	6	5	4	3	2	1	0
-	-	-	BRR	TBCK	RBCK	SPD	SRC

Bit Number	Bit Mnemonic	Description
7	-	**Reserved** The value read from this bit is indeterminate. Do not set this bit
6	-	**Reserved** The value read from this bit is indeterminate. Do not set this bit
5	-	**Reserved** The value read from this bit is indeterminate. Do not set this bit.
4	BRR	**Baud Rate Run Control bit** Cleared to stop the internal Baud Rate Generator. Set to start the internal Baud Rate Generator.
3	TBCK	**Transmission Baud rate Generator Selection bit for UART** Cleared to select Timer 1 or Timer 2 for the Baud Rate Generator. Set to select internal Baud Rate Generator.
2	RBCK	**Reception Baud Rate Generator Selection bit for UART** Cleared to select Timer 1 or Timer 2 for the Baud Rate Generator. Set to select internal Baud Rate Generator.
1	SPD	**Baud Rate Speed Control bit for UART** Cleared to select the SLOW Baud Rate Generator. Set to select the FAST Baud Rate Generator.
0	SRC	**Baud Rate Source select bit in Mode 0 for UART** Cleared to select $F_{OSC}/12$ as the Baud Rate Generator ($F_{CLK\ PERIPH}/6$ in X2 mode). Set to select the internal Baud Rate Generator for UARTs in mode 0.

Reset Value = XXX0 0000b
Not bit addressablef

Table 38. IE1 Register

IE1 - Interrupt Enable Register (B1h)

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

Bit Number	Bit Mnemonic	Description
7	-	**Reserved**
6	-	**Reserved**
5	-	**Reserved**
4	-	**Reserved**
3	-	**Reserved**
2	SPI	**SPI interrupt Enable bit** Cleared to disable SPI interrupt. Set to enable SPI interrupt.
1	-	**Reserved**
0	KBD	**Keyboard interrupt Enable bit** Cleared to disable keyboard interrupt. Set to enable keyboard interrupt.

Reset Value = XXXX X000b
Bit addressable

Table 39. IPL1 Register

IPL1 - Interrupt Priority Register (B2h)

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

Bit Number	Bit Mnemonic	Description
7	-	**Reserved** The value read from this bit is indeterminate. Do not set this bit.
6	-	**Reserved** The value read from this bit is indeterminate. Do not set this bit.
5	-	**Reserved** The value read from this bit is indeterminate. Do not set this bit.
4	-	**Reserved** The value read from this bit is indeterminate. Do not set this bit.
3	-	**Reserved** The value read from this bit is indeterminate. Do not set this bit.
2	SPIL	**SPI interrupt Priority bit** Refer to SPIH for priority level.
1	-	**Reserved** The value read from this bit is indeterminate. Do not set this bit.
0	KBDL	**Keyboard interrupt Priority bit** Refer to KBDH for priority level.

Reset Value = XXXX X000b
Bit addressable

Figure 27. Data Transmission Format (CPHA = 0)

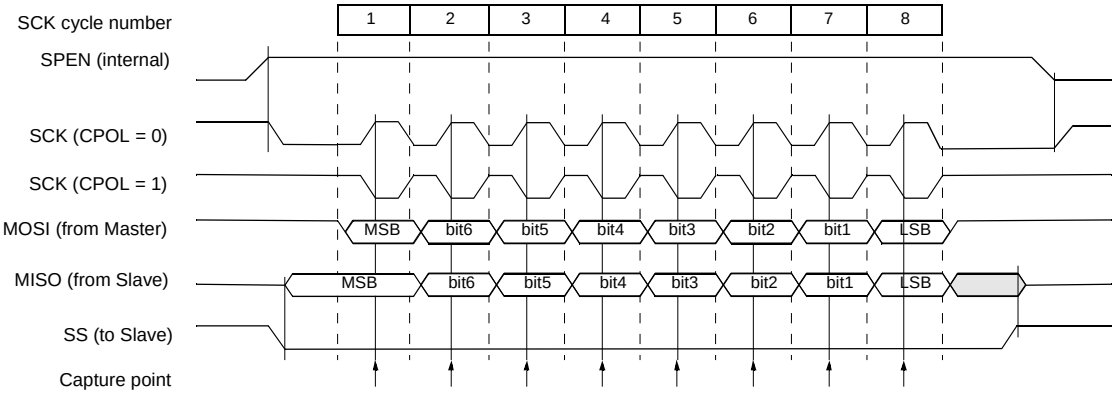


Figure 28. Data Transmission Format (CPHA = 1)

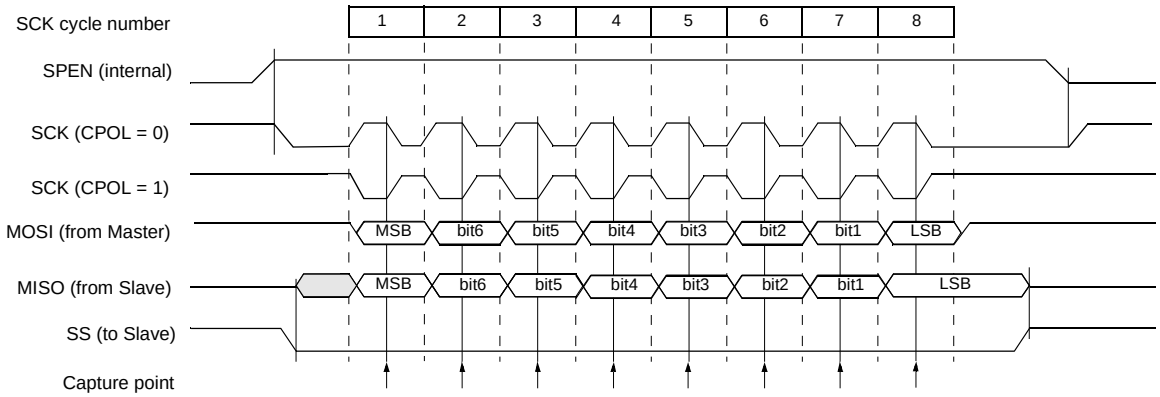
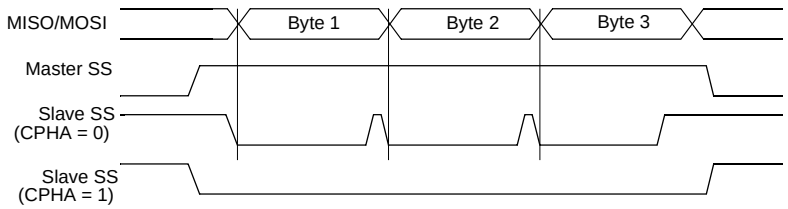


Figure 29. CPHA/$\overline{SS}$ Timing



As shown in Figure 28, the first SCK edge is the MSB capture strobe. Therefore the Slave must begin driving its data before the first SCK edge, and a falling edge on the $\overline{SS}$ pin is used to start the transmission. The $\overline{SS}$ pin must be toggled high and then low between each byte transmitted (Figure 25).

Figure 29 shows an SPI transmission in which CPHA is'1'. In this case, the Master begins driving its MOSI pin on the first SCK edge. Therefore the Slave uses the first SCK edge as a start transmission signal. The $\overline{SS}$ pin can remain low between transmissions (Figure 24). This format may be preferable in systems having only one Master and only one Slave driving the MISO data line.

Error conditions

The following flags in the SPSTA signal SPI error conditions:

Mode Fault (MODF)

Mode Fault error in Master mode SPI indicates that the level on the Slave Select ($\overline{SS}$) pin is inconsistent with the actual mode of the device. MODF is set to warn that there may have a multi-master conflict for system control. In this case, the SPI system is affected in the following ways:

- An SPI receiver/error CPU interrupt request is generated,
- The SPEN bit in SPCON is cleared. This disable the SPI,
- The MSTR bit in SPCON is cleared

When $\overline{SS}$ DISable (SSDIS) bit in the SPCON register is cleared, the MODF flag is set when the $\overline{SS}$ signal becomes'0'.

However, as stated before, for a system with one Master, if the $\overline{SS}$ pin of the Master device is pulled low, there is no way that another Master attempt to drive the network. In this case, to prevent the MODF flag from being set, software can set the SSDIS bit in the SPCON register and therefore making the $\overline{SS}$ pin as a general purpose I/O pin.

Clearing the MODF bit is accomplished by a read of SPSTA register with MODF bit set, followed by a write to the SPCON register. SPEN Control bit may be restored to its original set state after the MODF bit has been cleared.

Write Collision (WCOL)

A Write Collision (WCOL) flag in the SPSTA is set when a write to the SPDAT register is done during a transmit sequence.

WCOL does not cause an interruption, and the transfer continues uninterrupted.

Clearing the WCOL bit is done through a software sequence of an access to SPSTA and an access to SPDAT.

Overrun Condition

An overrun condition occurs when the Master device tries to send several data bytes and the Slave devise has not cleared the SPIF bit issuing from the previous data byte transmitted. In this case, the receiver buffer contains the byte sent after the SPIF bit was last cleared. A read of the SPDAT returns this byte. All others bytes are lost.

This condition is not detected by the SPI peripheral.

Interrupts

Two SPI status flags can generate a CPU interrupt requests:

Table 46. SPI Interrupts

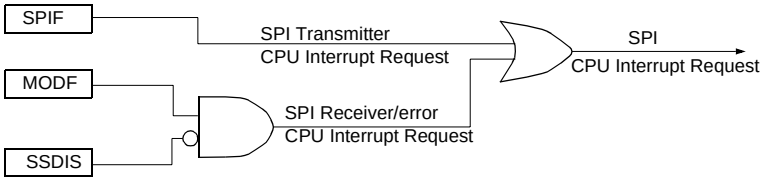
Flag	Request
SPIF (SP data transfer)	SPI Transmitter Interrupt request
MODF (Mode Fault)	SPI Receiver/Error Interrupt Request (if SSDIS ='0')

Serial Peripheral data transfer flag, SPIF: This bit is set by hardware when a transfer has been completed. SPIF bit generates transmitter CPU interrupt requests.

Mode Fault flag, MODF: This bit becomes set to indicate that the level on the SS is inconsistent with the mode of the SPI. MODF with SSDIS reset, generates receiver/error CPU interrupt requests.

Figure 30 gives a logical view of the above statements.

Figure 30. SPI Interrupt Requests Generation



Registers

There are three registers in the module that provide control, status and data storage functions. These registers are describes in the following paragraphs.

Serial Peripheral Control register (SPCON)

- The Serial Peripheral Control Register does the following:
- Selects one of the Master clock rates,
- Configure the SPI module as Master or Slave,
- Selects serial clock polarity and phase,
- Enables the SPI module,
- Frees the SS pin for a general purpose

Table 47 describes this register and explains the use of each bit.

Table 47. SPCON Register

SPCON - Serial Peripheral Control Register (0C3H)

7	6	5	4	3	2	1	0
SPR2	**SPEN**	**SSDIS**	**MSTR**	**CPOL**	**CPHA**	**SPR1**	**SPR0**

Bit Number	Bit Mnemonic	Description
7	SPR2	**Serial Peripheral Rate 2** Bit with SPR1 and SPR0 define the clock rate.
6	SPEN	**Serial Peripheral Enable** Cleared to disable the SPI interface. Set to enable the SPI interface.
5	SSDIS	**$\overline{SS}$ Disable** Cleared to enable SS# in both Master and Slave modes. Set to disable SS# in both Master and Slave modes. In Slave mode, this bit has no effect if CPHA ='0'.
5	MSTR	**Serial Peripheral Master** Cleared to configure the SPI as a Slave. Set to configure the SPI as a Master.
4	CPOL	**Clock Polarity** Cleared to have the SCK set to'0' in idle state. Set to have the SCK set to'1' in idle low.
3	CPHA	**Clock Phase** Cleared to have the data sampled when the SCK leaves the idle state (see CPOL). Set to have the data sampled when the SCK returns to idle state (see CPOL).

Bit Number	Bit Mnemonic	Description			
2	SPR1	**SPR2**	**SPR1**	**SPR0**	**Serial Peripheral Rate**
		0	0	0	$F_{CLK\ PERIPH}$/2
		0	0	1	$F_{CLK\ PERIPH}$/4
		0	1	0	$F_{CLK\ PERIPH}$/8
1	SPR0	0	1	1	$F_{CLK\ PERIPH}$/16
		1	0	0	$F_{CLK\ PERIPH}$/32
		1	0	1	$F_{CLK\ PERIPH}$/64
		1	1	0	$F_{CLK\ PERIPH}$/128
		1	1	1	Invalid

Reset Value= 0001 0100b

Not bit addressable

Serial Peripheral Status Register (SPSTA)

The Serial Peripheral Status Register contains flags to signal the following conditions:

- Data transfer complete
- Write collision
- Inconsistent logic level on $\overline{SS}$ pin (mode fault error)

Table 48 describes the SPSTA register and explains the use of every bit in the register.

Table 48. SPSTA Register

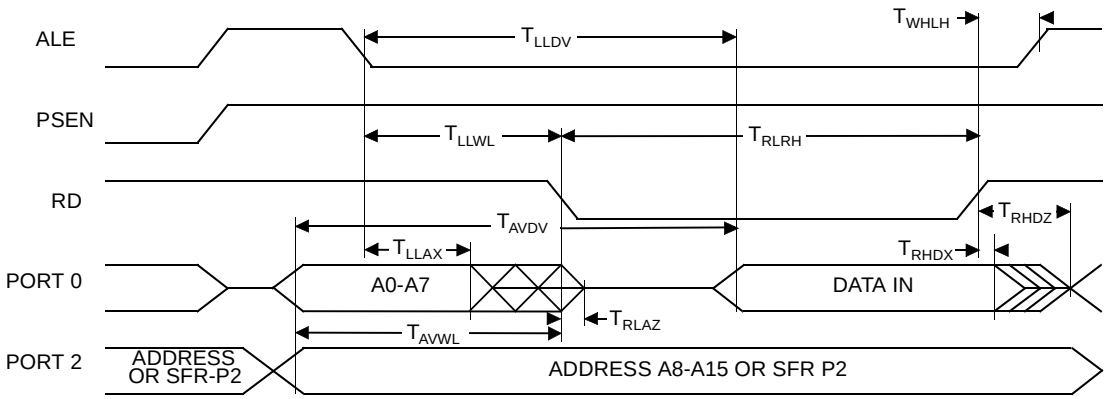
SPSTA - Serial Peripheral Status and Control register (0C4H)

7	6	5	4	3	2	1	0
SPIF	**WCOL**	**-**	**MODF**	**-**	**-**	**-**	**-**

Bit Number	Bit Mnemonic	Description
7	SPIF	**Serial Peripheral data transfer flag** Cleared by hardware to indicate data transfer is in progress or has been approved by a clearing sequence. Set by hardware to indicate that the data transfer has been completed.
6	WCOL	**Write Collision flag** Cleared by hardware to indicate that no collision has occurred or has been approved by a clearing sequence. Set by hardware to indicate that a collision has been detected.
5	-	**Reserved** The value read from this bit is indeterminate. Do not set this bit.
4	MODF	**Mode Fault** Cleared by hardware to indicate that the $\overline{SS}$ pin is at appropriate logic level, or has been approved by a clearing sequence. Set by hardware to indicate that the $\overline{SS}$ pin is at inappropriate logic level.
3	-	**Reserved** The value read from this bit is indeterminate. Do not set this bit
2	-	**Reserved** The value read from this bit is indeterminate. Do not set this bit

External Data Memory Read Cycle

Figure 38. External Data Memory Read Cycle



Serial Port Timing – Shift Register Mode

Table 64. Symbol Description

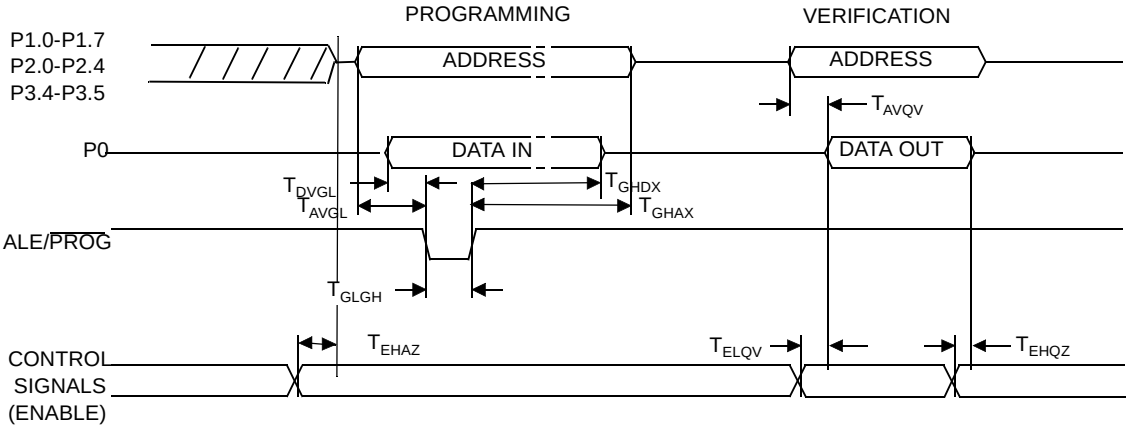
Symbol	Parameter
T_{XLXL}	Serial port clock cycle time
T_{QVHX}	Output data set-up to clock rising edge
T_{XHQX}	Output data hold after clock rising edge
T_{XHDX}	Input data hold after clock rising edge
T_{XHDV}	Clock rising edge to input data valid

Table 65. AC Parameters for a Fix Clock

	-M		-L		
Symbol	Min	Max	Min	Max	Units
T_{XLXL}	300		300		ns
T_{QVHX}	200		200		ns
T_{XHQX}	30		30		ns
T_{XHDX}	0		0		ns
T_{XHDV}		117		117	ns

Flash EEPROM Programming and Verification Waveforms

Figure 40. Flash EEPROM Programming and Verification Waveforms



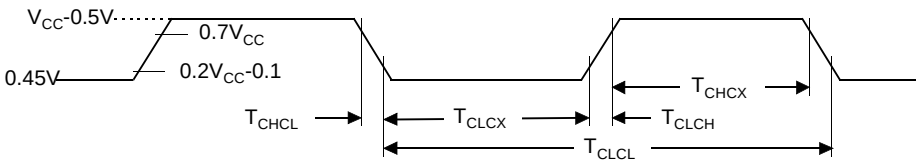
External Clock Drive Characteristics (XTAL1)

Table 68. External Clock Drive Characteristics (XTAL1)

Symbol	Parameter	Min	Max	Units
T_{CLCL}	Oscillator Period	25		ns
T_{CHCX}	High Time	3		ns
T_{CLCX}	Low Time	3		ns
T_{CLCH}	Rise Time		3	ns
T_{CHCL}	Fall Time		3	ns
T_{CHCX}/T_{CLCX}	Cyclic ratio in X2 mode	40	60	%

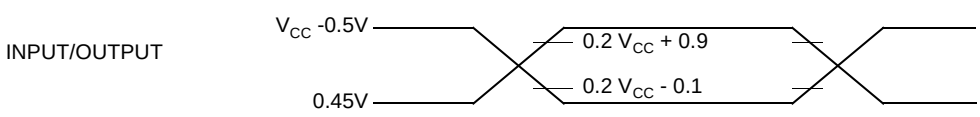
External Clock Drive Waveforms

Figure 41. External Clock Drive Waveforms



AC Testing Input/Output Waveforms

Figure 42. AC Testing Input/Output Waveforms



AC inputs during testing are driven at V_{CC} - 0.5 for a logic “1” and 0.45V for a logic “0”. Timing measurement are made at V_{IH} min. for a logic “1” and V_{IL} max for a logic “0”.

Flash EEPROM Memory

The Flash memory increases EPROM and ROM functionality with in-circuit electrical erasure and programming. It contains 16K or 32K bytes of program memory organized respectively in 128 or 256 pages of 128 bytes. This memory is both parallel and serial In-System Programmable (ISP). ISP allows devices to alter their own program memory in the actual end product under software control. A default serial loader (bootloader) program allows ISP of the Flash.

The programming **does not require 12V** external programming voltage. The necessary high programming voltage is generated on-chip using the standard V_{CC} pins of the microcontroller.

Features

- Flash E^2PROM internal program memory.
- Boot vector allows user provided Flash loader code to reside anywhere in the Flash memory space. This configuration provides flexibility to the user.
- Default loader in Boot ROM allows programming via the serial port without the need of a user provided loader.
- Up to 64K byte external program memory if the internal program memory is disabled (EA = 0).
- Programming and erase voltage with standard 5V or 3V V_{CC} supply.
- Read/Programming/Erase:
- Byte-wise read without wait state
- Byte or page erase and programming (10 ms)
- Typical programming time (32K bytes) in 10s
- Parallel programming with 87C51 compatible hardware interface to programmer
- Programmable security for the code in the Flash
- 10k write cycles
- 10 years data retention

Flash Programming and Erasure

The 16K or 32K bytes Flash is programmed by bytes or by pages of 128 bytes. It is not necessary to erase a byte or a page before programming. The programming of a byte or a page includes a self erase before programming.

There are three methods of programming the Flash memory:

- First, the on-chip ISP bootloader may be invoked which will use low level routines to program the pages. The interface used for serial downloading of Flash is the UART.
- Second, the Flash may be programmed or erased in the end-user application by calling low-level routines through a common entry point in the Boot ROM.
- Third, the Flash may be programmed using the parallel method by using a conventional EPROM programmer. The parallel programming method used by these devices is similar to that used by EPROM 87C51 but it is not identical and the commercially available programmers need to have support for the T89C51RB2/RC2. The bootloader and the Application Programming Interface (API) routines are located in the BOOT ROM.

Table 70. Program Lock Bits

Program Lock Bits				
Security level	**LB0**	**LB1**	**LB2**	**Protection description**
1	U	U	U	No program lock features enabled.
2	P	U	U	MOVC instruction executed from external program memory is disabled from fetching code bytes from internal memory, $\overline{EA}$ is sampled and latched on reset, and further parallel programming of the Flash is disabled. ISP and software programming with API are still allowed.
3	X	P	U	Same as 2, also verify through parallel programming interface is disabled.
4	X	X	P	Same as 3, also external execution is disabled. (Default)

Note: U: unprogrammed or "one" level.
P: programmed or "zero" level.
X: do not care
WARNING: Security level 2 and 3 should only be programmed after Flash and code verification.

These security bits protect the code access through the parallel programming interface. They are set by default to level 4. The code access through the ISP is still possible and is controlled by the "software security bits" which are stored in the extra Flash memory accessed by the ISP firmware.

To load a new application with the parallel programmer, a chip erase must first be done. This will set the HSB in its inactive state and will erase the Flash memory. The part reference can always be read using Flash parallel programming modes.

Default Values

The default value of the HSB provides parts ready to be programmed with ISP:

- BLJB: Programmed force ISP operation.
- X2: Unprogrammed to force X1 mode (Standard Mode).
- XRAM: Unprogrammed to valid XRAM
- LB2-0: Security level four to protect the code from a parallel access with maximum security.

Software Registers

Several registers are used, in factory and by parallel programmers, to make copies of hardware registers contents. These values are used by Atmel ISP (see Section "In-System Programming (ISP)", page 100).

These registers are in the "Extra Flash Memory" part of the Flash memory. This block is also called "XAF" or eXtra Array Flash. They are accessed in the following ways:

- Commands issued by the parallel memory programmer.
- Commands issued by the ISP software.
- Calls of API issued by the application software.

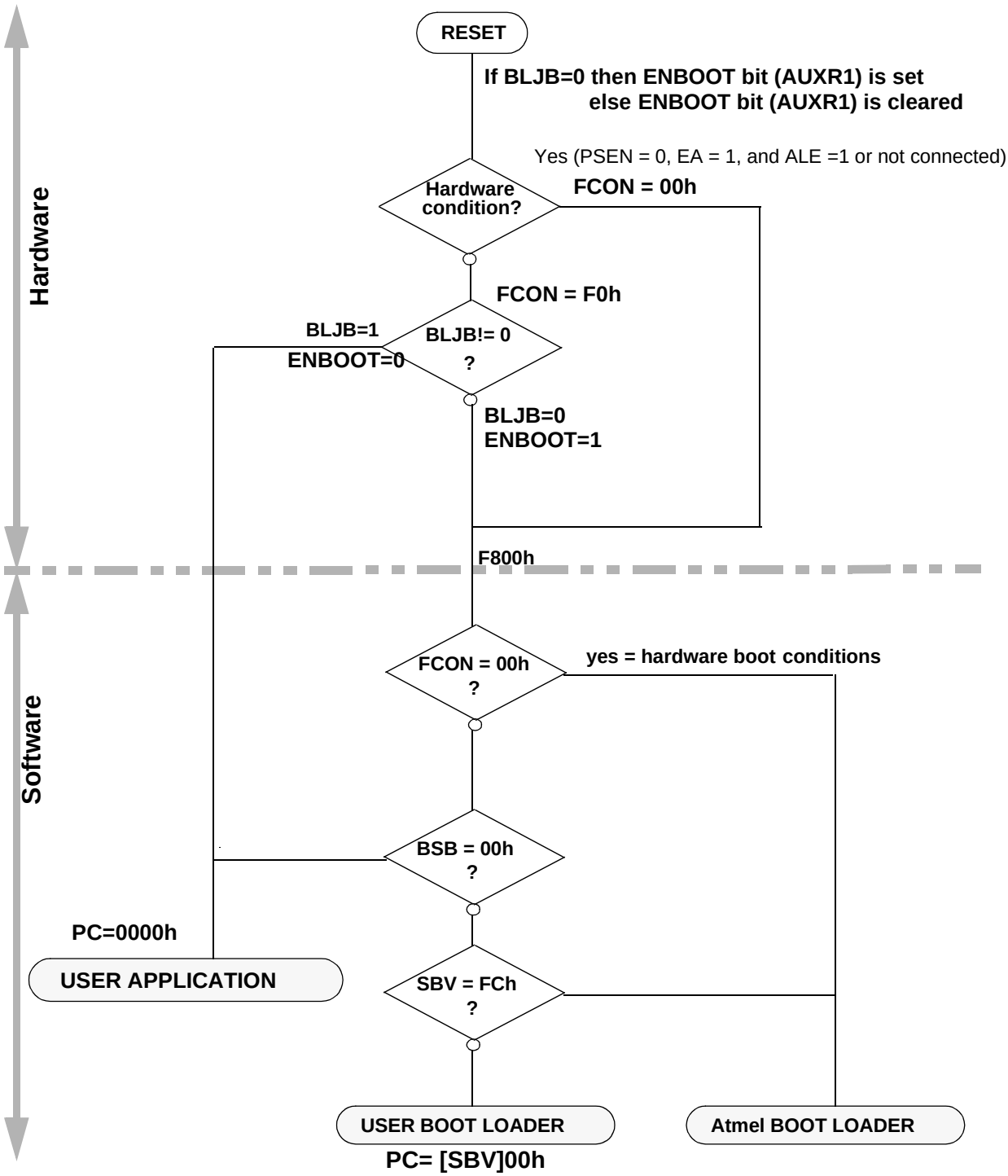
Several software registers described in Table 71.

After programming the Flash, the status byte should be programmed to zero in order to allow execution of the user's application code beginning at address 0000h.

Boot Process Summary

The boot process is summarized on the following flowchart:

Figure 47. Boot process flowchart



RECORD TYPE	COMMAND/DATA FUNCTION
04	Display Device Data or Blank Check Record type 04 causes the contents of the entire Flash array to be sent out the serial port in a formatted display. This display consists of an address and the contents of 16 bytes starting with that address. No display of the device contents will occur if security bit 2 has been programmed. The dumping of the device data to the serial port is terminated by the reception of any character. General Format of Function 04 :05xxxx04ssssseeeeffcc Where: 05 = number of bytes (hex) in record xxxx = required field, but value is a “don’t care” 04 = “Display Device Data or Blank Check” function code ssss = starting address eeee = ending address ff = subfunction 00 = display data 01 = blank check cc = checksum Example: :0500000440004FFF0069 (display 4000–4FFF)
05	Miscellaneous Read Functions General Format of Function 05 :02xxxx05ffsscc Where: 02 = number of bytes (hex) in record xxxx = required field, but value is a “don’t care” 05= “Miscellaneous Read” function code ffss = subfunction and selection code 0000 = read copy of the signature byte – manufacturer id (58H) 0001 = read copy of the signature byte – device ID# 1 (Family code) 0002 = read copy of the signature byte – device ID # 2 (Memories size and type) 0003 = read copy of the signature byte – device ID # 3 (Product name and revision) 0700 = read the software security bits 0701 = read BSB 0702 = read SBV 0704 = read HSB cc = checksum Example: :020000050001F0 read copy of the signature byte – device id # 1

In-application Programming Method

Several Application Program Interface (API) calls are available for use by an application program to permit selective erasing and programming of Flash pages. All calls are made through a common interface, PGM_MTP. The programming functions are selected by setting up the microcontroller’s registers before making a call to PGM_MTP at FFF0h. Results are returned in the registers. The API calls are shown in Table .

A set of Philips® compatible API calls is provided.

When several bytes have to be programmed, it is highly recommended to use the Atmel API “PROGRAM DATA PAGE” call. Indeed, this API call writes up to 128 bytes in a single command.

Table 75. API Calls (Continued)

API Call	Parameter
READ copy of the MANUFACTURER ID	Input Parameters: R0 = osc freq (integer Not required, left for Philips compatibility) R1 = 00h DPH = 00h DPL = 00h (manufacturer ID) Return Parameter ACC = value of byte read
READ copy of the device ID # 1	Input Parameters: R0 = osc freq (integer Not required, left for Philips compatibility) R1 = 00h DPH = 00h DPL = 01h (device ID # 1) Return Parameter ACC = value of byte read
READ copy of the device ID # 2	Input Parameters: R0 = osc freq (integer Not required, left for Philips compatibility) R1 = 00h DPH = 00h DPL = 02h (device ID # 2) Return Parameter ACC = value of byte read
READ copy of the device ID # 3	Input Parameters: R0 = osc freq (integer Not required, left for Philips compatibility) R1 = 00h DPH = 00h DPL = 03h (device ID # 2) Return Parameter ACC = value of byte read
READ SOFTWARE SECURITY BITS	Input Parameters: R0 = osc freq (integer Not required, left for Philips compatibility) R1 = 07h DPH = 00h DPL = 00h (Software security bits) Return Parameter ACC = value of byte read
READ HARDWARE SECURITY BITS	Input Parameters: R0 = osc freq (integer Not required, left for Philips compatibility) R1 = 07h -> OBh DPH = 00h DPL = 04h (Hardware security bits) Return Parameter ACC = value of byte read