P89LPC900

Microcontroller

Family

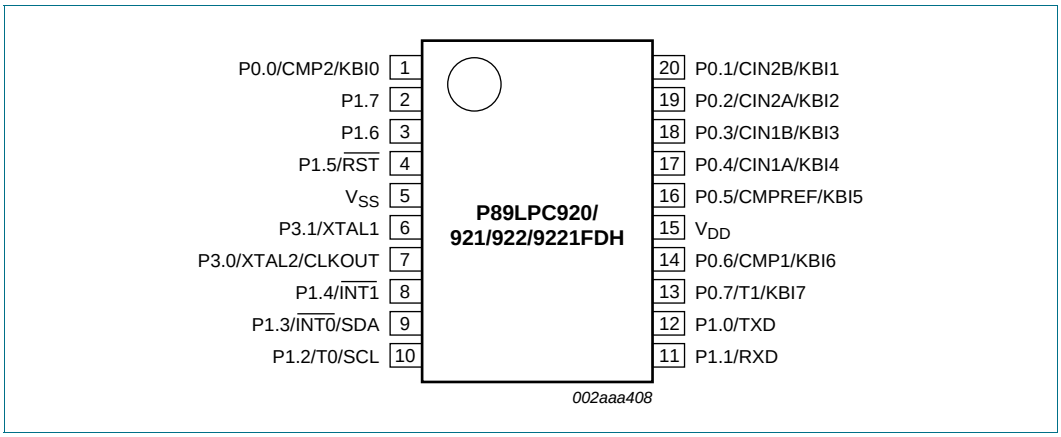
Flash

Programming Specifications

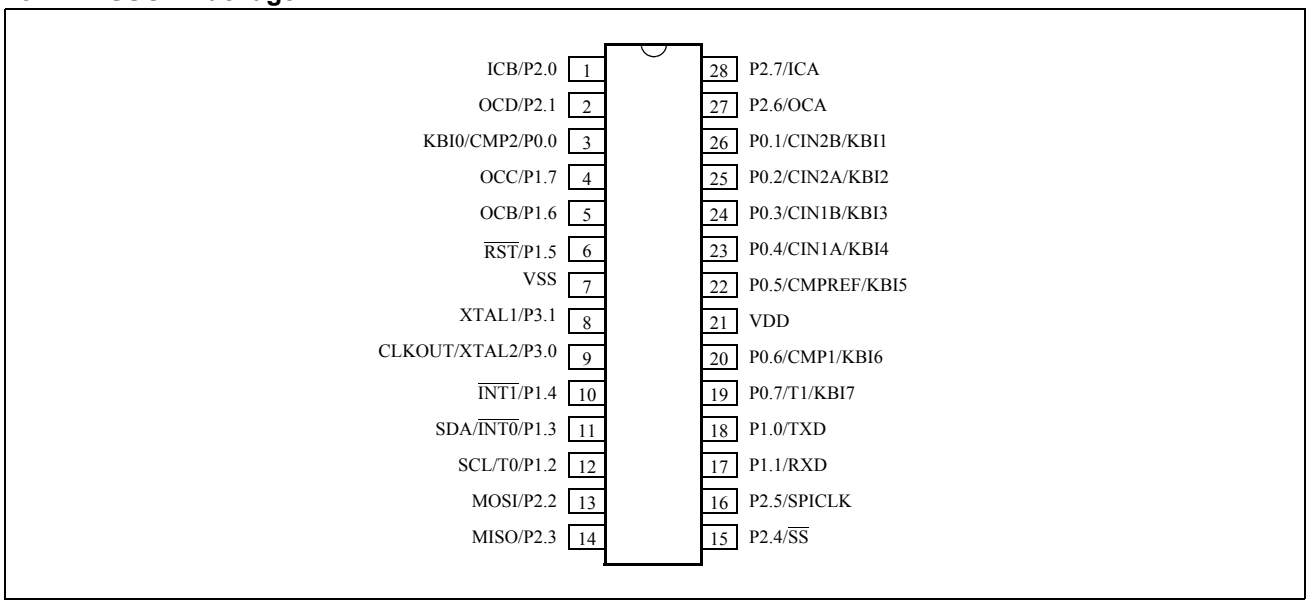
NXP
Semiconductors

PIN CONFIGURATION

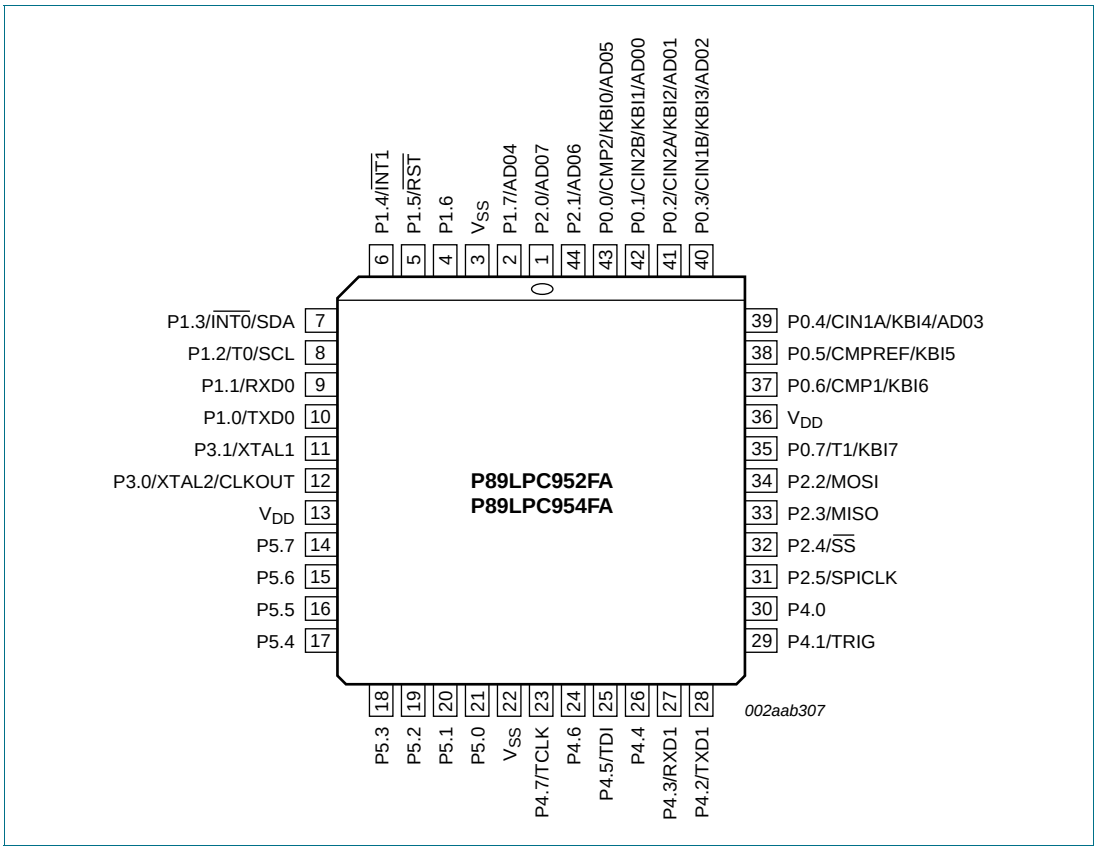
28-Pin TSSOP & DIP Pinout



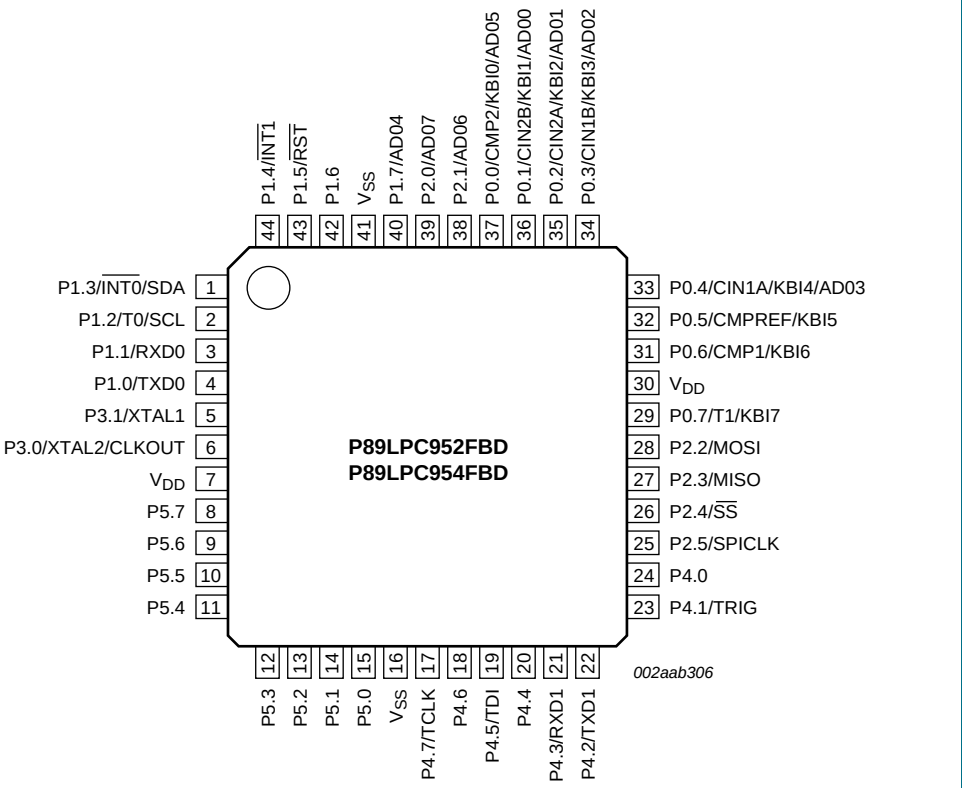
28-Pin TSSOP Package



44-Pin PLCC Package



44-Pin TQFP Package



PIN DESCRIPTION:

PIN NAME	NAME DURING PROG.	I/O	FUNCTION DURING PROGRAMMING
P1.7	$\overline{\text{WRITE}}$/READ	I	Write input
P1.0	SEL0	I	Register select 0
P1.1	SEL1	I	Register select 1
P3.1	CLK	I	External clock input
P1.5	$\overline{\text{RST}}$	P	Reset
P0.7	D7	I/O	Data 7
P0.6	D6	I/O	Data 6
P0.5	D5	I/O	Data 5
P0.4	D4	I/O	Data 4
P0.3	D3	I/O	Data 3
P0.2	D2	I/O	Data 2
P0.1	D1	I/O	Data 1
P0.0	D0	I/O	Data 0
V_{DD}	V_{DD}	P	V_{DD}
Vss	Vss	P	GROUND

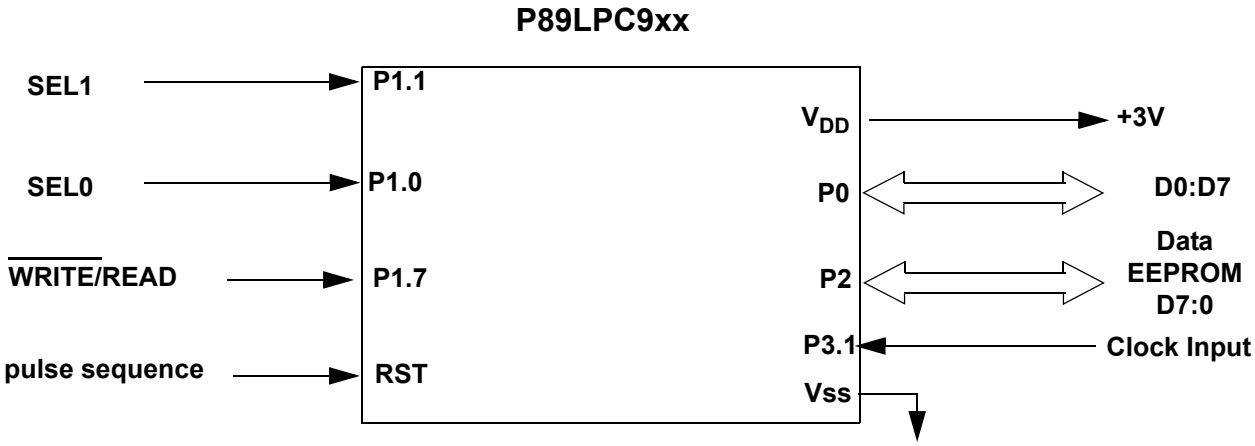


Figure 1: Programming / Verify Pin Connections

PRODUCT SELECTION

PRODUCT	FLASH SIZE	END ADDR	SIGNATURE BYTES MFG ID1 ID2	SECTOR SIZE	PAGE SIZE	PRE-PROGRAMMED SERIAL LOADER	DEFAULT BOOT VECTOR	DATAEE
P89LPC954	16K x 8	3FFF	15H DDH 7AH	1Kx8	64x8	3E00H-3FFFH	3FH	N
P89LPC952	8K x 8	1FFF	15H DDH 28H	1Kx8	64x8	1E00H-1FFFH	1FH	N
P89LPC938	8K x 8	1FFF	15H DDH 25H	1Kx8	64x8	1E00H-1FFFH	1FH	Y
P89LPC9381	4K x 8	0FFF	15H DDH 2AH	1Kx8	64x8	0E00H-0FFFH	0FH	Y
P89LPC936	16K x 8	3FFF	15H DDH 24H	2Kx8	64x8	3E00H-3FFFH	3FH	Y
P89LPC9351	8K x 8	1FFF	15H DDH 2EH	1Kx8	64x8	1E00H-1FFFH	1FH	Y
P89LPC935	8K x 8	1FFF	15H DDH 1EH	1Kx8	64x8	1E00H-1FFFH	1FH	Y
P89LPC934	8K x 8	1FFF	15H DDH 1DH	1Kx8	64x8	1E00H-1FFFH	1FH	N
P89LPC933	4K x 8	0FFF	15H DDH 0AH	1Kx8	64x8	0E00H-0FFFH	0FH	N
P89LPC9321	8K x 8	1FFF	15H DDH 2FH	1Kx8	64x8	1E00H-1FFFH	1FH	Y
P89LPC932A1	8K x 8	1FFF	15H DDH 1FH	1Kx8	64x8	1E00H-1FFFH	1FH	Y
P89LPC932/ CP323x	8K x 8	1FFF	15H DDH 05H	1Kx8	64x8	1E00H-1FFFH	1FH	Y
P89LPC932	8K x 8	1FFF	15H DDH 05H	1Kx8	64x8	1E00H-1FFFH	1EH	Y
P89LPC931	8K x 8	1FFF	15H DDH 09H or 15H DDH 05H	1Kx8	64x8	1E00H-1FFFH	1FH	N
P89LPC930	4K x 8	0FFF	15H DDH 19H or 15H DDH 05H	1Kx8	64x8	0E00H-0FFFH	0FH	N
P89LPC925	8K x 8	1FFF	15H DDH 1CH	1Kx8	64x8	1E00H-1FFFH	1FH	N
P89LPC924	4K x 8	0FFF	15H DDH 1BH	1Kx8	64x8	0E00H-0FFFH	0FH	N
P89LPC922	8K x 8	1FFF	15H DDH 0CH or 15H DDH 05H	1Kx8	64x8	1E00H-1FFFH	1FH	N
P89LPC921	4K x 8	0FFF	15H DDH 0BH or 15H DDH 05H	1Kx8	64x8	0E00H-0FFFH	0FH	N
P89LPC920	2K x 8	07FF	15H DDH 1AH	1Kx8	64x8	0600H-07FFH	07H	N

Revision to previous document

Added devices type P89LPC9351 and P89LPC9321.

Recommended Programmer Features

In order for your customers to obtain the most fexibility in using this device, we recommend that your programmer offer specific features. These include:

- ISP code protection - The upper 512 bytes of last sector of the device (see PRODUCT SELECTION, above) contains factory provided ISP code which will be erased by erasing this sector. We recommend that the user be warned that performing a sector erase on this sector will erase the ISP code. The lower 512 bytes can be erased using the page erase function. A programmer manufacturer might wish to offer an option to "erase all user code except for the ISP code" and "erase all user code including the ISP code"
- User configuration support - Support user erase/programming and reading of Status bit, Boot Vector, UCFG1 (WDT, osc, etc) independent of the user code array. A configuration screen works best.

- Page erase - Erases a single 64-byte page.
- Sector erase - Erases a single sector of 1KB.
- Sector CRC - Provides CRC on a single sector. Compare device CRC with your memory buffer's CRC.
- Global CRC - Provides CRC on entire user code memory. Compare device CRC with your memory buffer's CRC.
- Status Bit- After programming the code memory, the programmer should program the Status bit to a zero. A user option should be available to override this feature.

Programming Interface Architecture

Prior to performing any programmer operations it is necessary to activate the code programming mode of the device. Once in code programming mode, programming operations for this device are accomplished through the use of four registers: FMCON, FMADRH, FMADRL, and FMDATA.

FMCON (flash memory control register) is used to specifiy operation modes and to read status. FMADRH (flash memory address high) and FMADRL (flash memory address low) are used to specifiy the address of the memory location, page, sector, or other resource to be accessed in the flash microcontroller. FMDATA (flash memory data) contains data to be written to or read from the flash memory or other resource.

Once in code programming mode, these four registers can be read or written by using the WRITE/, SEL1, and SEL0 signals in addition to the databus (Port 0) and the clock. The register selected by the binary combinations for the select inputs, SEL1 and SEL0, are shown below:

00	read/write FMADRL
01	read/write FMADRH
10	read/write FMDATA
11	read/write FMCON

FMCON commands (write) include:

LOAD	(00h)	Clear and then load the page register.
PROG	(48h)	Program page with the contents of the page register.
ERS_G	(72h)	Erase global (all sectors and security)
ERS_S	(71h)	Erase sector and security
ERS_P	(70h)	Erase page.
CONF	(6Ch)	Accesses user configuration information addressed by FMADRL.
CRC_G	(1Ah)	Calculate a CRC on the entire user code space.
CRC_S	(19h)	Calculate a CRC on the sector addressed by FMADRH.

FMCON contains status information (read) and is updated by the last operation performed. FMCON

contains the following bits :

Bit	Flag	Description
0	OI	Operation Interrupted.Only used in IAP or ISP modes Should never be set in parallel programming mode. (If observed in parallel programming mode it is likely that status is not being read correctly).
1	SV	Security Violation. Set if operation fails due to security settings. Cycle is aborted.Memory contents are unchanged. CRC output is invalid.
2	HVE	High Voltage Error. Set if error detected in high voltage generation circuits. Cycle is aborted. Memory contents may be corrupted.
3	HVA	High Voltage Abort.Set if high voltage cycle is aborted due to a Vdd brownout condition.Memory contents may be corrupted.
4	-	unused; reads as a '1'
5	-	unused; reads as a '1'
6	-	unused; reads as a '1'
7	BUSY	Set while a program, erase, CRC calculation, or other operation is in progress.

Table 0.1

Code memory programming uses a 64-byte page register. From 1 to 64 bytes may be loaded into the page register. This may be followed by a PROG command causing the new page register contents to be programmed into the flash memory. The page register may not be read. Only page register locations that have been written will be programmed into the flash array, thus it is not necessary to write to all 64 locations in the page register.

Activating Code Memory Programming Mode (see Figure 2)

The microcontroller is placed into code programming mode by performing the following sequence (see Figure 2). (Note that powering the device from Vdd =0V to Vdd = operating voltage is required as indicated in steps 2 & 3)

1. Drive RST pin and P3.1 to the logic zero level.
2. Apply 0V to the VDD pin.
3. Apply VDD to the VDD pin.
4. Wait t_{VR}.
5. Drive the $\overline{\text{Write}}$ pin to the logic high level.
6. Drive RST pin to the logic high level, observing the timing specification, t_{RH}.
7. Drive RST pin to a logic low observing the timing specification, t_{RL}.
8. Repeat steps 6 through 7 four more times for a total of five low-going pulses.
9. Drive RST pin to the logic high level.
10. Wait t_{RP}.
11. The device should now be in programming mode. **Note: Any additional low-going pulses on the RST pin will remove the device from the code programming mode.**
12. Drive the SEL1 and SEL0 pins high. (The combination of SEL1, SEL0, and $\overline{\text{Write}}$ pins high is a read of the FMCON register with the contents of FMCON appearing on Port 0).
13. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low.
14. Continue to pulse P3.1 and read FMCON until the BUSY bit (P0.7) is a zero.

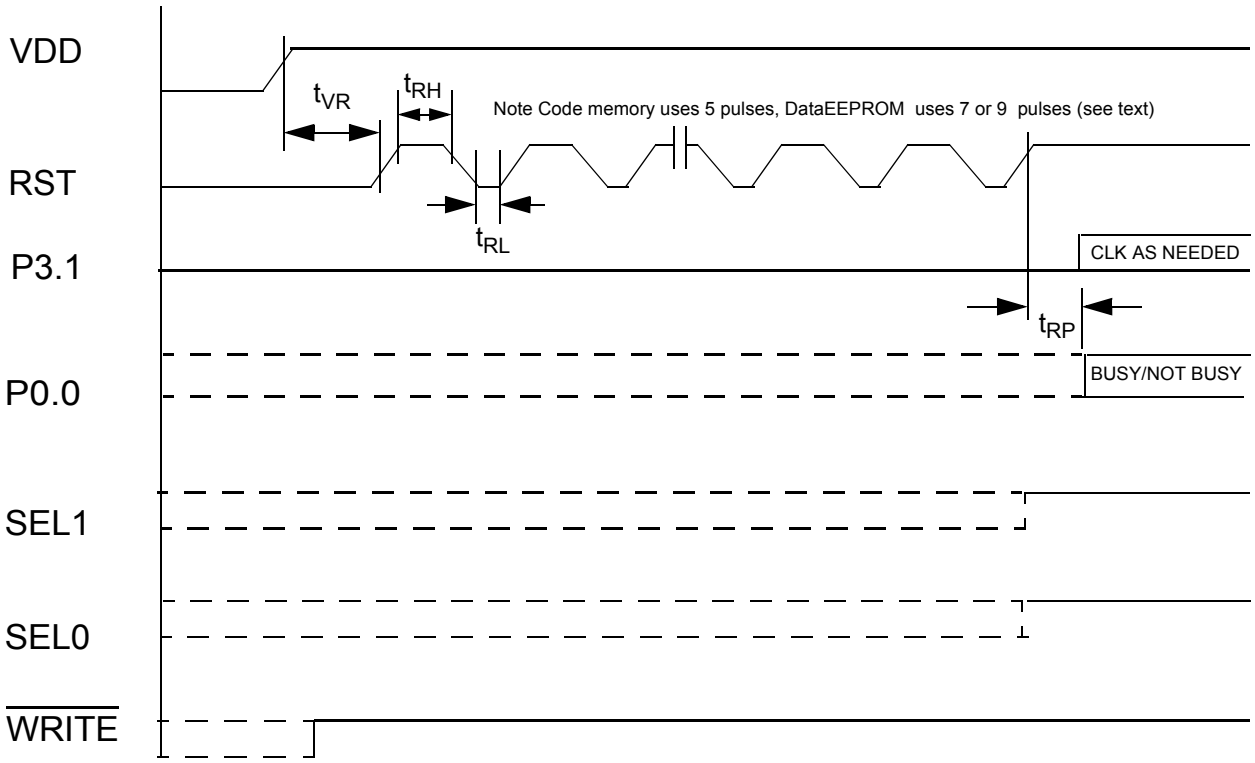


Figure 2: Activation of Code or Data EEPROM Programming Modes

Table 2: AC CHARACTERISTICS, ACTIVATION OF PROGRAMMING MODES, TA = 25 ^{O}C, VDD = 3V ± 5%

PARAMETER	SYMBOL	MIN	MAX	Unit
RST delay from Vdd active	t_{VR}	150		uSec
RST high time	t_{RH}	1	32	uSec
RST low time	t_{RL}	1		uSec
RST high to programming mode active	t_{RP}		150	uSec

Reading Flash Memory Registers (see Figure 3)

Thoughout this document references will be made to reading a value from the flash memory registers (FMCON, FMADRH,FMADRL, FMDATA). To read one of these registers perform the following sequence. (See Figure 3).

1.This sequence assumes that code programming mode is currently activated.
2.Drive the $\overline{\text{Write}}$ pin with a logical one.
3.Drive the SEL1 and SEL0 pins with the binary combination for the desired register.
4.Wait a delay.
5.Read the register data from the data pins (Port 0).

Writing Flash Memory Registers (see Figure 3)

Throughout this document references will be made to writing a value to the flash memory registers (FMCON, FMADRH,FMADRL, FMDATA). To write to one of these registers perform the following sequence. (See Figure 3)

1.This sequence assumes that code programming mode is currently activated.
2.Drive the SEL1 and SEL0 pins with the binary combination for the desired register.
3.Drive the $\overline{\text{Write}}$ pin with a logical zero.
4.Place the data to be written on the databus (Port0).
5.Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low.
6.Drive the $\overline{\text{Write}}$ pin with a logical one.

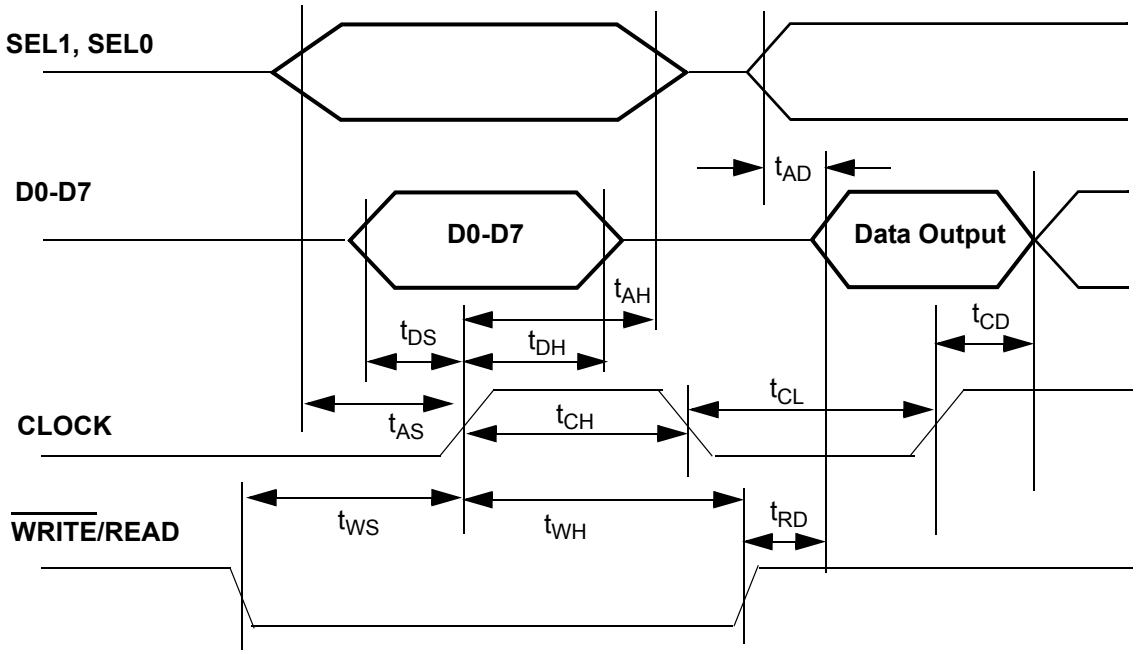


Figure 3. Flash Memory Register Read/Write Cycles

Table 3: AC CHARACTERISTICS, TA = 25 °C, VDD = 3V ± 5%[1]

PARAMETER	SYMBOL	MIN	MAX	Unit
Address setup to clock	t_{AS}	100		nSec
Address hold to clock	t_{AH}	100		nSec
Write setup to clock	t_{WS}	100		nSec
Write hold to clock	t_{WH}	100		nSec
Data input setup to clock high	t_{DS}	100		nSec
Data input hold to clock high	t_{DH}	100		nSec
Address to data valid	t_{AD}		100	nSec
Read to data valid	t_{RD}		100	nSec
Read to Port 0 Low Z		0		nSec
Write to Port 0 High Z		20		nSec
Clock high to next data out valid	t_{CD}	100		nSec
Clock low time	t_{CL}	1		uSec
Clock high time	t_{CH}	1		uSec

Loading the Page Register

The page register is loaded by performing the following sequence:

1. Activate the Code Programming Mode, as previously described, if not already performed.
2. Write the "LOAD" command to FMCON. (Note that this clears the page register of any previously loaded data)
3. Write the lower byte of the first address to be loaded into the page register to FMADRL.
4. Write the data to be loaded to the FMDATA register.
5. Provide three clock pulses to P3.1.
6. The address in FMADRL will auto-increment for the next byte. (Since the page is 64 bytes in length, incrementing past the end of the page will wrap around to the beginning of the same page).
7. Continue writing additional bytes to the page register as desired. You may change to a different address within the page by repeating this process starting with step 3.

Programming User Code Memory

Code memory may only be programmed by using the page register. This may be performed using the following sequence:

1. Load the page register with the data to be programmed as previously described.
2. Write the lower 8-bits of the page address to FMADRL.
3. Write the upper 8-bits of the page address to FMADRH. Note: writing the upper two bits of FMADRL and the upper address byte to FMADRH may be included in step 2 of loading the page register, if desired, since the auto-increment of the page register does not carry beyond the lower 6 bits of FMADRL.
4. Write the PROG command to the FMCON register.
5. Provide a clock pulse to P3.1
6. Read the FMCON register to obtain status.
7. Continue reading, and providing a clock pulse to P3.1, until the interface is either not BUSY or until an error has occured.

Erasing all sectors (global erase)

Sectors and their sector security bits may be erased using the following sequence:

1. Write the ERS_G command to the FMCON register.
2. Read the FMCON register to obtain status. Continue reading until the interface is either not BUSY or until an error has occured.

Erasing a single sector

A single sector and its sector security bits may be erased using the following sequence:

1. Write the upper 8-bits of the sector address to FMADRH. (Note that only FMADRH[4:2] are used here)
2. Write the ERS_S command to the FMCON register.
3. Read the FMCON register to obtain status. Continue reading until the interface is either not BUSY or until an error has occured.

Erasing a single page

A single page may be erased using the following sequence:

1. Write the lower 8-bits of the page register address to FMADRL. (only FMADRL[7:6] are used)
2. Write the upper 8-bits of the page register address to FMADRH.(only FMADRH[4:0] are used)
3. Write the ERS_P command to the FMCON register.
4. Read the FMCON register to obtain status. Continue reading until the interface is either not BUSY or until an error has occured.

CRC Calculation

A 32-bit CRC may be performed on either an individual sector (Sector CRC) or the entire user code memory (Global CRC). Both use the same method for calculating the 32-bit CRC result which is stored in four 8-bit registers. Initially these four 8-bit registers are cleared when the CRC command (CRC_G or CRC_S) is written to FMCON. For each byte of code memory in the intended memory range, the following calculation is performed.

Define a 32-bit CRC result register (CRC)and set its contents = 0.

Define a 32-bit temporary variable (TAP) and set it contents = 0.

Define a single-bit variable (CRC_FLAG).

Starting with the first byte in code memory, and for each byte in the memory, perform the following CRC calculation:

1. Shift the CRC result (CRC) to the left one bit and save the MSB in the CRC_FLAG.
2. Read the byte from code memory and distribute the eight bits of the code-byte into the 32 bits of the TAP variable as shown

in the table, below. Unused bits of the TAP variable must be filled with zeros.

Code-byte bit position	is copied into TAP variable bit position
0	0
1	3
2	5
3	8
4	10
5	13
6	16
7	18

3. XOR the 32-bit TAP variable with the 32-bit CRC variable and save the result in the CRC variable.
4. If the CRC_FLAG (saved in step 1) was a zero, proceed to step 5, else, XOR the CRC variable with 00400007H. Store the result in the CRC variable.
5. The CRC calculation for THIS byte is finished. and the CRC variable holds the current CRC result. Repeat, starting with step 1, for each additional byte of code memory.

If the saved MSB =1, the byte from the code memory is XOR with 00400007H. This result is XOR with the 32-bit CRC result. The result of this operation is stored as the CRC result.

If the saved MSB =0, the byte from the code memory is XOR with the 32-bit CRC result. The result of this operation is stored as the CRC result.

Calculate Global CRC

A 32-bit global CRC of the entire user code memory may be calculated using the following sequence:

1.Write the CRC_G command to the FMCON register.
2.Read the FMCON register to obtain status. Continue reading until the interface is either not BUSY or until an error has occured.
3.Select FMDATA, read mode, then provide a clock pulse to P3.1.
4.Read FMDATA to obtain CRC bits 7:0 (no clock)
5.Provide a clock pulse to P3.1.
6.Read FMDATA again to obtain CRC bits 15:8 (no clock)
7.Provide a clock pulse to P3.1.
8.Read FMDATA again to obtain CRC bits 23:16 (no clock)
9.Provide a clock pulse to P3.1.
10.Read FMDATA again to obtain CRC bits 31:24 (no clock)
11.Read FMCON and provide one clock pulse to P3.1.

Calculate Sector CRC

A 32-bit global CRC of a single sector of user code memory may be calculated using the following sequence:

1.Write the "LOAD" command to FMCON. (Note that this clears the page register of any previously loaded data)
2.Write the upper 8-bits of the sector address to FMADRH.
3.Write the CRC_S command to the FMCON register.
4.Read the FMCON register to obtain status. Continue reading until the interface is either not BUSY or until an error has occured.
5.Select FMDATA, with WRITE\ pin low, then provide a clock pulse to P3.1.
6.Read FMDATA to obtain CRC bits 7:0 (no clock)
7.Provide a clock pulse to P3.1.
8.Read FMDATA again to obtain CRC bits 15:8 (no clock)
9.Provide a clock pulse to P3.1.
10.Read FMDATA again to obtain CRC bits 23:16 (no clock)
11.Provide a clock pulse to P3.1.
12.Read FMDATA again to obtain CRC bits 31:24(no clock)
13.Read FMCON and provide one clock pulse to P3.1

Reading Configuration , Boot Vector, Status Byte, Security Bits, Signature Bytes

Devices parameters such as configuration bytes, status byte, boot vector, security bits , and signature bytes may be read by writing an address of FMADRL and a command to FMCON. These registers have the following addresses:

00h	UCFG1	User Configuration Register 1
01h	UCFG2	User Configuration Register 2
02h	Boot Vector	
03h	Status Byte	
08h	SEC0	Security byte, sector 0
09h	SEC1	Security byte, sector 1
0Ah	SEC2	Security byte, sector 2
0Bh	SEC3	Security byte, sector 3
0Ch	SEC4	Security byte, sector 4
0Dh	SEC5	Security byte, sector 5
0Eh	SEC6	Security byte, sector 6
0Fh	SEC7	Security byte, sector 7
10h	MFGid	Manufacturer id
11h	ID1	Device id 1
12h	ID2	device id 2
18h	SEC8	Security byte, sector 8 (89LPC954)
19h	SEC9	Security byte, sector 9 (89LPC954)
1Ah	SEC10	Security byte, sector 10 (89LPC954)
1Bh	SEC11	Security byte, sector 11 (89LPC954)
1Ch	SEC12	Security byte, sector 12 (89LPC954)

1Dh	SEC13	Security byte, sector 13 (89LPC954)
1Eh	SEC14	Security byte, sector 14 (89LPC954)
1Fh	SEC15	Security byte, sector 15 (89LPC954)

These bytes may be read using the following sequence:

1. Write the CONF command to the FMCON register.
2. Write the address of the register to be read to the FMADRL register.
3. Read the FMDATA register to obtain the desired data. (no clock)
4. An auto-increment mode is provided and may be used if desired. After reading the first byte, provide TWO clock pulses to P3.1 to increment the address to the next location. Thereafter, provide ONE clock pulse to P3.1 to increment to the next location. The auto-increment function will not bypass the unspecified locations 04H through 07H. Thus it will be necessary to provide clock pulses to increment through these locations. To terminate the auto-increment function read FMCON and provide one clock pulse to P3.1

Writing Configuration , Boot Vector, Status Byte, and Security Bits

Device parameters such as configuration bytes, status byte, boot vector, and security bits, made be written by writing a command to FMCON, an address to FMADRL, and then the data to FMDATA. These registers have the following addresses:

00h	UCFG1	User Configuration Register 1
01h	UCFG2	User Configuration Register 2
02h	Boot Vector	Upper byte of PC upon reset, if Status Byte bit 0 = 1.
03h	Status Byte	Use Boot vector as upper byte of reset address
08h	SEC0	Security byte, sector 0
09h	SEC1	Security byte, sector 1
0Ah	SEC2	Security byte, sector 2
0Bh	SEC3	Security byte, sector 3
0Ch	SEC4	Security byte, sector 4
0Dh	SEC5	Security byte, sector 5
0Eh	SEC6	Security byte, sector 6
0Fh	SEC7	Security byte, sector 7
18h	SEC8	Security byte, sector 8 (89LPC954)
19h	SEC9	Security byte, sector 9 (89LPC954)
1Ah	SEC10	Security byte, sector 10 (89LPC954)
1Bh	SEC11	Security byte, sector 11 (89LPC954)
1Ch	SEC12	Security byte, sector 12 (89LPC954)
1Dh	SEC13	Security byte, sector 13 (89LPC954)
1Eh	SEC14	Security byte, sector 14 (89LPC954)
1Fh	SEC15	Security byte, sector 15 (89LPC954)

The first 4 registers are self-erasing when being updated. The security bytes can only be erased by erasing the associated sector using an ERS_S or ERS_G command. These registers may be written using the following sequence:

1. Write the CONF command to the FMCON register.
2. Write the address of the register to be written to the FMADRL register.
3. Write the desired data to the FMDATA register.
4. Read the FMCON register to obtain status. Continue reading until the interface is either not BUSY or until an error has occured.

Note: Some registers, such as signature bytes, may not be erased and reprogrammed by the user. Security bits that are programmed will need to be erased by performing a sector or global erase operation prior to re-programming. (See sector or global erase commands).

Security Bits defintions

This device has three security bits associated with each of its eight sectors, as shown in Figure 3.

SECx

7	6	5	4	3	2	1	0
-	-	-	-	-	EDISx	SPEDISx	MOVCDISx

Address: xxxxh

Unprogrammed value: 00h

BIT	SYMBOL	FUNCTION
SECx.7-3	-	Reserved (should remain unprogrammed at zero).
SECx.2	EDISx	**Erase Disable ISPx**. Disables the ability to perform an erase of sector "x" in ISP or IAP mode.. When programmed, this bit and sector x can only be erased by a 'global' erase command using a commercial programmer . This bit and sector x CANNOT be erased in ISP or IAP modes.
SECx.1	SPEDISx	**Sector Program Erase Disable x**. Disables program or erase of **all or part of sector x**. This bit and sector x are erased by a 'global' erase command(either ISP, IAP, or from commercial programmer.
SECx.0	MOVCDISx	**MOVC Disable**. Disables the MOVC command for sector x. Any MOVC that attempts to read a byte in a MOVC protected sector will return invalid data. This bit can only be erased when sector x is erased.

Figure 3: User sector Security Bytes (SEC0, ..., SEC7)

EDISx	SPEDISx	MOVCDISx	Effects on Parallel Programming
0	0	0	None.
0	0	1	Security violation flag set for sector CRC calculation for the specific sector. Security violation flag set for global CRC calculation if any MOVCDISx bit is set. Cycle aborted. Memory contents unchanged. CRC invalid. Program/erase commands will not result in a security violation.
0	1	0	Security violation flag set for program commands or an erase page command. Cycle aborted. Memory contents unchanged. Sector erase and global erase are allowed.
0	1	1	
1	0	0	Security violation flag set for program or erase commands. Cycle aborted. Memory contents unchanged. Global erase is allowed.
1	0	1	
1	1	0	
1	1	1	

Table 0.4 Effects of Security Bits

Activating Data EEPROM Programming Mode (see Figure 2 - applies to P89LPC932, P89LPC932A1, P89LPC932/CP323x, P89LPC935, P89LPC936, P89LPC938, P89LPC9321 and P89LPC9351 devices).

The microcontroller is placed into Data EEPROM programming mode by performing the following sequence (see Figure 2). (Note that powering the device from Vdd =0V to Vdd = operating voltage is required as indicated in steps 2 & 3)

1.Drive RST pin and P3.1 to the logic zero level.

2.Apply 0V to the VDD pin.

3.Apply VDD to the VDD pin.

4.Wait t_{VR}

5.Drive the $\overline{\text{Write}}$ pin to the logic high level.

6.Drive RST pin to the logic high level, observing the timing specification, t_{RH}.

7.Drive RST pin to a logic low observing the timing specification, t_{RL}.

8.Drive RST pin to the logic high level, observing the timing specification, t_{RH}.

9.If using the P89LPC932, repeat steps 7 through 8 **six** more times for a total of **seven** low-going pulses. If using either the P89LPC932/CP323x, P89LPC932A1, P89LPC935, P89LPC936, P89LPC938, P89LPC9321 or P89LPC9351 device, repeat steps 7 through 8 **eight** more times for a total of **nine** low-going pulses

10.Wait t_{RP}

11.Steps 12 through 15 will be used to clock a 48-bit number into the device, MSB first using P0.7 as the clock pin and P0.0 as the data-in pin. The 48-bit number is : (MSB) 0000 0000 0000 0000 0000 1001 0000 0000 0000 0000 0000 1100.

12.Place the data-bit value on P0.0.

13.Drive P0.7 (clock) high for a minimum of 1 us.

14.Drive P0.7 (clock) low for a minimum of 1 us.

15.Repeat steps 12 through 13 until all 48 bits have been clocked into the device.

16.If using the P89LPC932, repeat steps 13 through 14 **one** more time (data-bit value is a 'don't care'). If using either the P89LPC932/CP323x, P89LPC932A1, P89LPC935, P89LPC936, P89LPC938, P89LPC9321 or P89LPC9351 device, do not perform this step.

17.Drive RST low resetting the device.

18.Drive RST high.

19.Drive the databus (P0) low.

20.Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min.

21.Repeat step 20 until P1.0 goes low.

22.Drive P0 with 75H.

23.Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.

24.Drive P0 with A5H.

25.Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.

26.Drive P0 with FFH.

27.Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.

28.Drive P0 with 75H.

29.Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.

30.Drive P0 with A4H.

31.Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.

32.Drive P0 with 00H.

33.Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1

low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.

34.The device should be in Data EEPROM programming mode ready to for reading or writing of data.

Writing to the Data EEPROM Memory (applies to P89LPC932, P89LPC932A1, P89LPC932/CP323x, P89LPC935, P89LPC936, P89LPC938, P89LPC9321 and P89LPC9351 devices)

Writing data to the Data EEPROM may be performed using the following sequence:

1.This sequence assumes that Data EEPROM programming mode is currently active. If not, activate the Data EEPROM programming mode as described, above.

2.Steps 3 through 8 store the A8 address bit of the 512 byte EEPROM array. This bit does not need to be programmed for each byte that is written into the EEPROM array.

3.Drive P0 with 75H.

4.Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.

5.Drive P0 with F1H.

6.Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.

7.Drive P0 with 00H if the address bit ,A8, for the location to be programmed, is a zero. Drive P0 with 01H if the address bit , A8, for the location to be programmed, is a one.

8.Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.

9.Drive P0 with 75H.

10.Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.

11.Drive P0 with F2H.

12.Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.

13.Drive P0 with the data to be programmed.

14.Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.

15.Drive P0 with 75H.

16.Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.

17.Drive P0 with F3H.

18.Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.

19.Drive P0 with the lower bits of the address (A7:0) of the location to be programmed.

20.Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.

21.Drive P0 with E5H.

22. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.
23. Drive P0 with F1H.
24. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.
25. Drive P0 with F5H.
26. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.
27. Drive P0 with A0H.
28. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.
29. Drive P0 with 00H.
30. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.
31. Drive P0 with 00H.
32. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.
33. Read P2. If P2.7 is low, repeat steps 21through 33 until P2.7 is high. This completes programming of a single byte.
34. Repeat steps 2 through 33 to write additional bytes to the Data EEPROM. If the address bit, A8, does not need to change from its previously stored value, steps 3 through 8 may be omitted.

Reading the Data EEPROM Memory (applies to P89LPC932, P89LPC932A1, P89LPC932/CP323x, P89LPC935, P89LPC936, P89LPC938, P89LPC9321 and P89LPC9351 devices))

Reading the Data EEPROM may be performed using the following sequence:

1. This sequence assumes that Data EEPROM programming mode is currently active. If not, activate the Data EEPROM programming mode as described, above.
2. Steps 3 through 8 store the A8 address bit of the 512 byte EEPROM array. This bit does not need to be programmed for each byte that is read from the EEPROM array.
3. Drive P0 with 75H.
4. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.
5. Drive P0 with F1H.
6. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.
7. Drive P0 with 00H if the address bit ,A8, for the location to be programmed, is a zero. Drive P0 with 01H if the address bit , A8, for the location to be programmed, is a one.
8. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.
9. Drive P0 with 75H.
10. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock

pulses to P3.1 until P1.0 changes state.

11. Drive P0 with F3H.
12. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.
13. Drive P0 with the lower bits of the address (A7:0) of the location to be programmed.
14. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.
15. Drive P0 with 00H.
16. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.
17. Drive P0 with 00H.
18. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.
19. Drive P0 with 00H.
20. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.
21. Drive P0 with E5H.
22. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.
23. Drive P0 with F2H.
24. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.
25. Drive P0 with F5H.
26. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.
27. Drive P0 with A0H.
28. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.
29. Drive P0 with 00H.
30. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.
31. Drive P0 with 00H.
32. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.
33. Read P2 to obtain the DataEEPROM data
34. Repeat steps 2 through 33 to read additional bytes from the Data EEPROM. If the address bit , A8, does not need to change from its previously stored value, steps 3 through 8 may be omitted.

Block or Row Fill of the Data EEPROM Memory (applies to P89LPC932, P89LPC932A1, P89LPC932/CP323x, P89LPC935, P89LPC936, P89LPC938, P89LPC9321 and P89LPC9351 devices))

Filling a row (64 bytes) or a block (512 bytes) of the Data EEPROM may be performed using the following sequence:

1. This sequence assumes that Data EEPROM programming mode is currently active. If not, activate the Data EEPROM programming mode as described, above.
2. Steps 3 through 8 store the A8 address bit of the 512 byte EEPROM array. This bit must be set for the Block Fill. For the Row Fill, it should reflect the A8 bit of the respective Row.
3. Drive P0 with 75H.
4. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.
5. Drive P0 with F1H.
6. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.
7. Drive P0 with 31H for the Block Fill operation. For a Row Fill operation where the address bit , A8, for the Row to be filled is a zero, drive P0 with 20H. For a Row Fill operation where the address bit , A8, for the Row to be filled is a one, drive P0 with 21H.
8. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.
9. Drive P0 with 75H.
10. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.
11. Drive P0 with F2H.
12. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.
13. Drive P0 with the data value to be filled into the Block or Row.
14. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.
15. Drive P0 with 75H.
16. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.
17. Drive P0 with F3H.
18. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.
19. For a Row Fill operation, drive P0 with the lower bits of the address (A7:0) of the Row to be programmed. For the Block Fill operation drive P0 with any value.
20. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.
21. Drive P0 with E5H.
22. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.
23. Drive P0 with F1H.
24. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock

pulses to P3.1 until P1.0 changes state.

25. Drive P0 with F5H.
26. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.
27. Drive P0 with A0H.
28. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.
29. Drive P0 with 00H.
30. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.
31. Drive P0 with 00H.
32. Observe the state of P1.0. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 uS min, and then driving P3.1 low for 1 uS min. If P1.0 has changed state, proceed to the next step. If P1.0 is in the same state, continue to provide clock pulses to P3.1 until P1.0 changes state.
33. Read P2. If P2.7 is low, repeat steps 21through 33 until P2.7 is high. This completes the Block Fill or Row Fill operation..

Unless noted elsewhere, datasheet specification of this device are applicable during programming operations.

Revisions

Jul 08, 2008

Added device type P89LPC9351 and P89LPC9321.

June 15, 2007

Added device type P89LPC954. Added pinouts for 20-pin and 44-pin packages.

May 26, 2004

Added notation to DataEEPROM sections (**Reading the Data EEPROM Memory, Activating Data EEPROM Programming Mode, and Writing to the Data EEPROM Memory, Block Fill, Row Fill**) indicating that the information applies to additional devices.

Added devices types P89LPC924, P89LPC925, P89LPC936, and P89LPC938.

Changed device type P89LPC932/01 to P89LPC932A1 to reflect device name change.

Dec 03, 2003

Fixed bugs in DataEEPROM sections (**Reading the Data EEPROM Memory, Activating Data EEPROM Programming Mode, and Writing to the Data EEPROM Memory).**

Added **Block Fil**l and **Row Fill** operations to DataEEPROM sections.

Changed t_{VR} spec from 50us to 150us.

Nov 12, 2003

Removed the READ command from the FMCON command list.

Nov 7, 2003

Expanded Product Selection table to include new devices (89LPC932/01 and P89LPC932/CP323x).
Corrected default boot vector for P89LPC932.

Aug 29, 2003

Added new devices (89LPC920, 89LPC933, 89LPC934, 89LPC935) to Product Selection table.
Added default boot vector to Product Selection table.
Suggested programmers program the Status bit to zero after programming code memory.

June 26, 2003

Expanded the product selection table. Changed the ISP sector erase warning to cover 4KB and 8KB devices.

Removed references to a READ mode.

Added comment for read/write of DataEEPROM such that it applies only to the P89LPC932.

May 28, 2003

Changed the order of events in performing a CRC_S (sector CRC), specifically changed the sequence from write CRC_S to FMCON , write FMADRH to: write LOAD to FMCON, write FMADRH, write CRC_S to FMCON.

May 27, 2003

Fixed bugs in read of DataEEPROM (**Reading the Data EEPROM Memory).**

May 14, 2003

Fixed bugs in read/write of DataEEPROM (**Activating Data EEPROM Programming Mode).**
Noted that unused bits of FMCON during status read return a logic ‘1’ .

May 5, 2003

Added support for read/write of DataEEPROM.