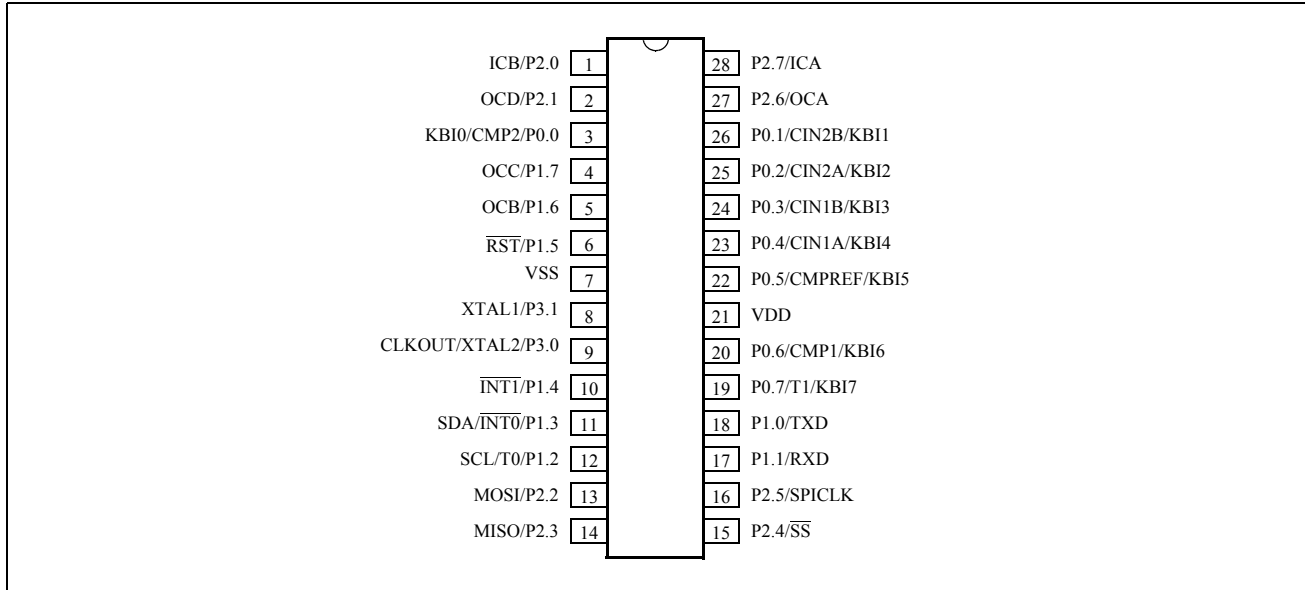


P89LPC900
Microcontroller
Family
Flash
Programming
Specifications

P89LPC900 Family

PIN CONFIGURATION

28-Pin TSSOP Package



P89LPC900 Family

PIN DESCRIPTION:

PIN NAME	NAME DURING PROG.	I/O	FUNCTION DURING PROGRAMMING
P1.7	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	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}
V _{SS}	V _{SS}	P	GROUND

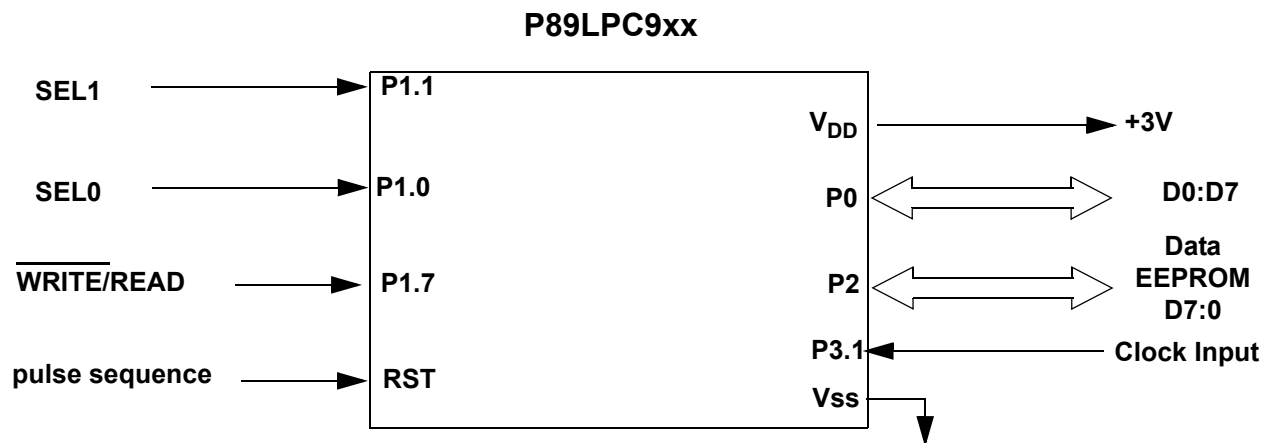


Figure 1: Programming / Verify Pin Connections

P89LPC900 Family

PRODUCT SELECTION

PRODUCT	FLASH SIZE	END ADDRESS	SIGNATURE BYTES			SECTOR SIZE	PAGE SIZE	PRE-PROGRAMMED SERIAL LOADER
			MFG	ID1	ID2			
P89LPC932	8K x 8	1FFF	15H	DDH	05H	1Kx8	64x8	1E00H-1FFFFH
P89LPC931	8K x 8	1FFF	15H	DDH	09H or 15H DDH 05H	1Kx8	64x8	1E00H-1FFFFH
P89LPC930	4K x 8	0FFF	15H	DDH	19H or 15H DDH 05H	1Kx8	64x8	0E00H-0FFFFH
P89LPC922	8K x 8	1FFF	15H	DDH	0CH or 15H DDH 05H	1Kx8	64x8	1E00H-1FFFFH
P89LPC921	4K x 8	0FFF	15H	DDH	0BH or 15H DDH 05H	1Kx8	64x8	0E00H-0FFFFH

Revision to previous document

Expanded Product Selection table to include new devices.

Recommended Programmer Features

In order for your customers to obtain the most flexibility 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 (sector 7 for 8K x 8 devices, sector 3 for 4K x 8 devices) 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.

P89LPC900 Family

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 specify operation modes and to read status. FMADRH (flash memory address high) and FMADRL (flash memory address low) are used to specify 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.
READ (80h)	Read main memory addressed by FMADRH and FMADRL. This command

is not enabled in production devices.

P89LPC900 Family

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 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 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.

P89LPC900 Family

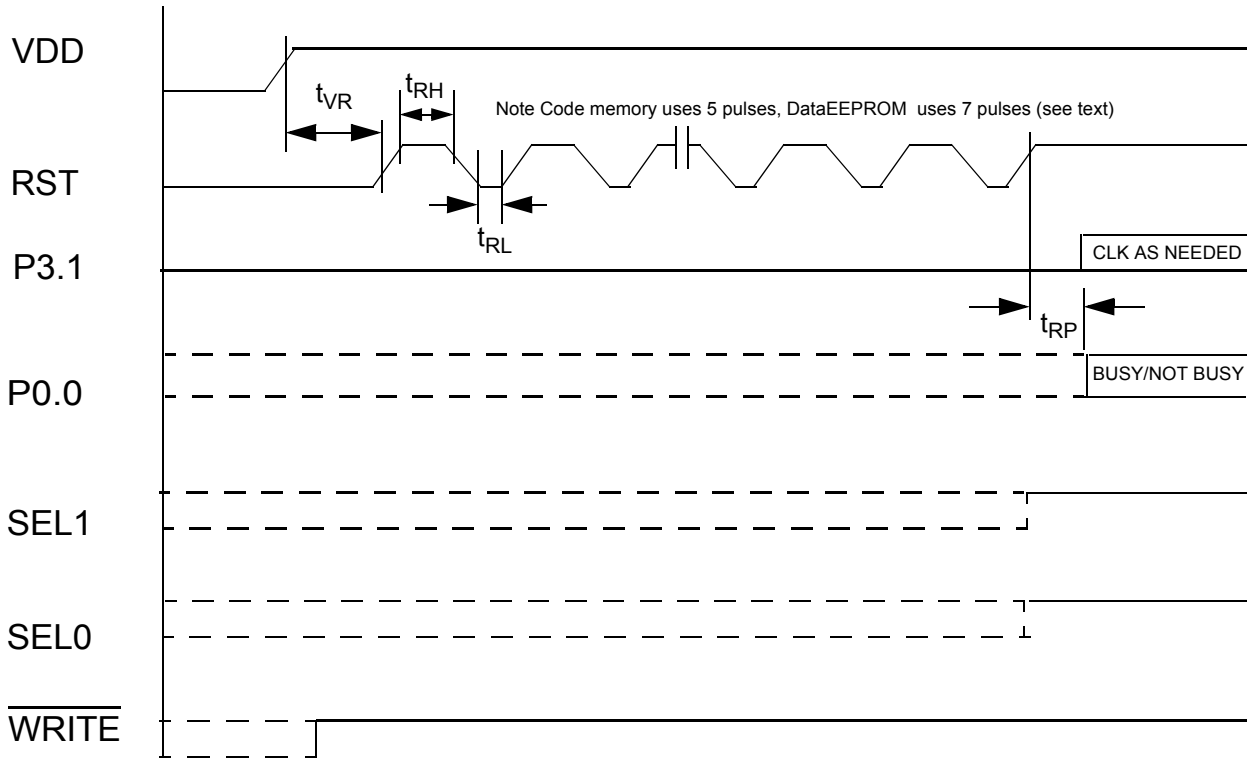


Figure 2: Activation of Code or Data EEPROM Programming Modes

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

PARAMETER	SYMBOL	MIN	MAX	Unit
RST delay from Vdd active	t_{VR}	50		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

P89LPC900 Family

Reading Flash Memory Registers (see Figure 3)

Throughout 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.

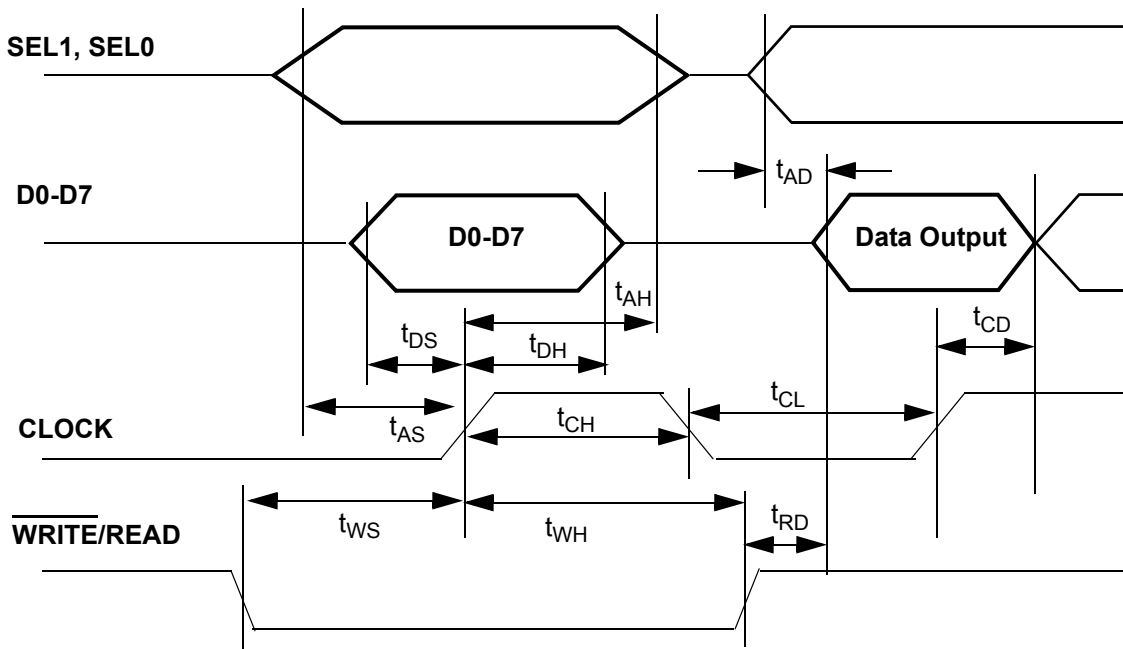


Figure 3. Flash Memory Register Read/Write Cycles

P89LPC900 Family**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 occurred.

P89LPC900 Family

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 occurred.

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 occurred.

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 occurred.

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 its 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

P89LPC900 Family

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

- XOR the 32-bit TAP variable with the 32-bit CRC variable and save the result in the CRC variable.
- 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.
- 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:

- Write the CRC_G command to the FMCON register.
- Read the FMCON register to obtain status. Continue reading until the interface is either not BUSY or until an error has occurred.
- Select FMDATA, read mode, then provide a clock pulse to P3.1.
- Read FMDATA to obtain CRC bits 7:0 (no clock)
- Provide a clock pulse to P3.1.
- Read FMDATA again to obtain CRC bits 15:8 (no clock)
- Provide a clock pulse to P3.1.
- Read FMDATA again to obtain CRC bits 23:16 (no clock)
- Provide a clock pulse to P3.1.
- Read FMDATA again to obtain CRC bits 31:24 (no clock)
- 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:

P89LPC900 Family

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 occurred.
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

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

P89LPC900 Family

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

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 occurred.

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).

P89LPC900 Family

Security Bits definitions

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
Address: xxxh	-	-	-	-	-	EDISx	SPEDISx	MOVCDISx
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 - P89LPC932 only)

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)

P89LPC900 Family

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. Repeat steps 7 through 8 **six** more times for a total of **seven** 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 0001 0000 0000 0000 0000 0000 1100.
12. Place the data-bit value on P0.0.
13. Drive P0.7 (clock) high.
14. Drive P0.7 (clock) low.
15. Repeat steps 12 & 13 until all 48 bits have been clocked into the device.
16. Drive RST low resetting the device.
17. Drive RST high.
18. Drive the databus (P0) low.
19. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 μs min, and then driving P3.1 low.
20. Repeat step 19, 255 times, for a total of 256 clock pulses on P3.1.
21. The device should be in Data EEPROM programming mode ready to for reading or writing of data.

Writing to the Data EEPROM Memory (P89LPC932 only)

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. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 μs min, and then driving P3.1 low.
5. Drive P0 with F1H.
6. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 μs min, and then driving P3.1 low.
7. If the Data EEPROM address bit, A8, is a one, drive P0 with 01H. If the Data EEPROM address bit, A8, is a zero, drive P0 with 00H.
8. Provide two clock pulses to P3.1.
9. Drive P0 with 75H.
10. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 μs min, and then driving P3.1 low.
11. Drive P0 with F2H.
12. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 μs min, and then driving P3.1 low.
13. Drive P0 with data to be programmed
14. Provide two clock pulses to P3.1.
15. Drive P0 with 75H.
16. Provide a clock pulse to P3.1 by driving P3.1 to a logic high for 1 μs min, and then driving P3.1 low.

P89LPC900 Family

17. Drive P0 with F3H.
18. 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.
19. Drive P0 with address bits A7:A0.
20. Provide two clock pulses to P3.1.
21. Drive P0 with 00H.
22. Continuously clock P3.1 while waiting for 4mS. P3.1 should be clocked at least 256 times.
23. Repeat steps 3 through 21 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 (P89LPC932 only)

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. Drive P0 with 75H.
3. 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.
4. Drive P0 with A5H.
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 P0 with FFH.
7. Provide two clock pulses to P3.1.
8. Drive P0 with 75H.
9. 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.
10. Drive P0 with A4H.
11. 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.
12. Drive P0 with 00H.
13. Provide two clock pulses to P3.1.
14. Steps 15 through 20 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.
15. Drive P0 with 75H.
16. 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.
17. Drive P0 with F1H.
18. 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.
19. If the Data EEPROM address bit, A8, is a one, drive P0 with 01H. If the Data EEPROM address bit, A8, is a zero, drive P0 with 00H.
20. Provide two clock pulses to P3.1.
21. Drive P0 with 75H.
22. 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.
23. Drive P0 with F3H.
24. 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.
25. Drive P0 with address bits A7:A0
26. Provide two clock pulses to P3.1.
27. Drive P0 with 00H.
28. Provide 10 clock pulses to P3.1.
29. Drive P0 with E5H.

P89LPC900 Family

30. 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.
31. Drive P0 with F2H.
32. 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.
33. Drive P0 with F5H.
34. 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.
35. Drive P0 with A0H.
36. 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.
37. Sample P2 to obtain the Data EEPROM contents.
38. Repeat steps 15 through 37 to read additional bytes from the Data EEPROM. If the address bit A8 does not need to change from its previously stored value, steps 15 through 20 may be omitted.

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

Revisions

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.