Application Note

programming of the parallel EEPROM MMEE08001808SCC

Purpose

This application note describes the hardware and software required to program 3DPLUS parallel EEPROMs.

Introduction

The MMEE08001808SCC 8Mbit EEPROMs are organized as 1Mx8. The device use eight 128Kbit EEPROM: the address, data I/O and control signal lines of the eight devices are parallel, while the #CEi (chip enables) are separated. Figure 1 shows the block diagram of the module.

One of the very interesting feature of the EEPROM (Electrically Erasable and Programmable CMOS ROM) components is that they can be electrically erased and Programmed.
It brings them the capability to be programmed before or/and after assembly on the PCB :

Prior mounting on the PCB :

- The EEPROM module can be programmed with any standard PROM programmer.

After mounting on the PCB :

- The EEPROM module can be programmed by connection to the PROM programmer thanks to a program connector on the PCB.
- It can also be programmed directly by the application (FPGA or Jtag port of the FPGA for instance).

There are three special "Enable" pins on the EEPROM:

- CE (Chip Enable) Activates the chip (if CE is not enabled, the EEPROM will do nothing)
- OE (Output Enable) Makes the chip output a byte
- WE (Write Enable) Makes the chip program a byte into itself

When you are programming an EEPROM, you want to keep CE enabled the whole time, so tie CE low. (Because all three of these Enable signals are active-low.) When programming, keep OE disabled the entire time, because you are not having the EEPROM output any bytes: Since it is active-low, tie it high to disable it. Leave WE disabled for now.

RDY/#BUSY pin needs an external pull up resistor.

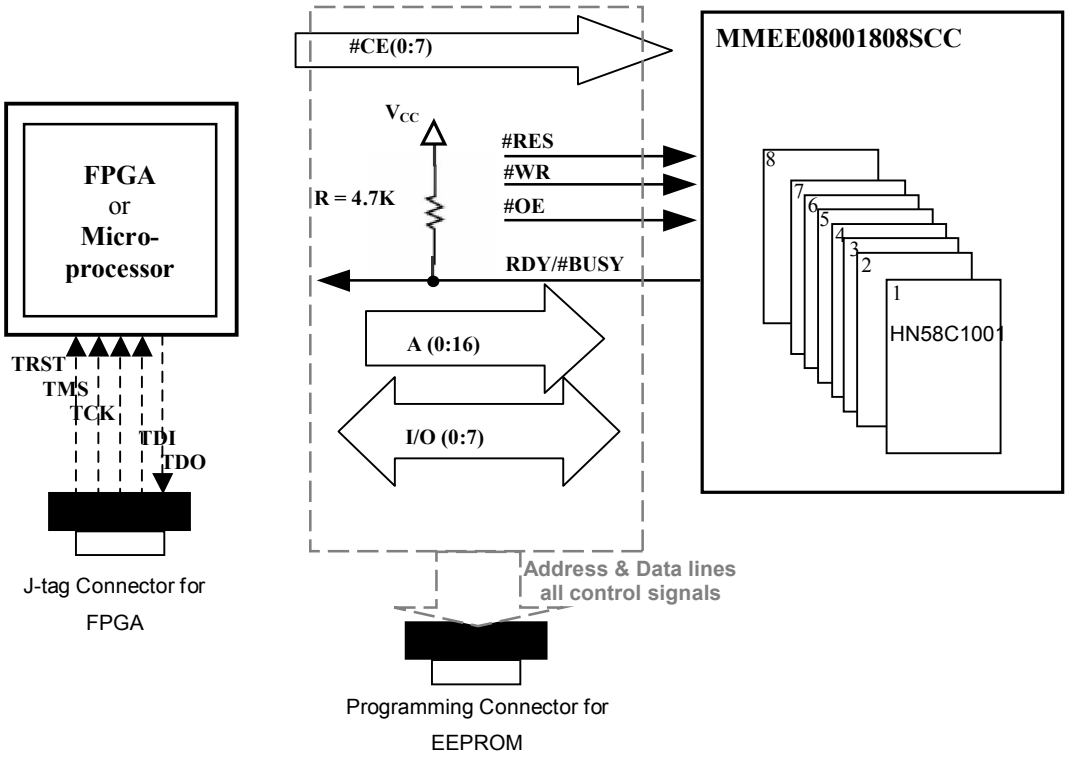

Figure 1 MMEE08001808SCC Block diagram after board assembly

Programming

Program the module with a programmer, PXI card socket, or on board FPGAs, note that any commercial programmer supporting Hitachi HN58C1001 EEPROM can be used to program 3D Plus MMEE08001808SCC EEPROMs

1) Turn the EEPROM on. Connect its power pins to a power supply. (that would be pins 10)
2) Configure the EEPROM's address and data buses to indicate the byte you would like to store, and the address you want to store it in. For example, to store the value 80 in the very first byte of the EEPROM (which is address 0), tie all the address bus pins low, so the address is set to 0. Then set the data bus pins to the binary representation of 80, which is 01010000. Remember that bit numbers are read from right to left, not from left to right, so in this example, the first four data bus pins should be 0.
3) Take the CE and WE pin low to write the byte. Respect the timing in figure 2. Send WE low for a moment, then put it high again. If the EEPROM is working properly, you have now stored your byte, and you can later examine it by taking OE high and to respect figure 3 for reading operation.
4) The READY/BUSY pin, which is an output from the EEPROM which indicates when the chip has finished writing to itself and is ready to accept more input. If you are programming the chip using an automated machine, the speed of the EEPROM may be a factor, but when programming it by

hand, this is not likely to matter much, since the EEPROM stores new values into itself in quite a bit less than one second.

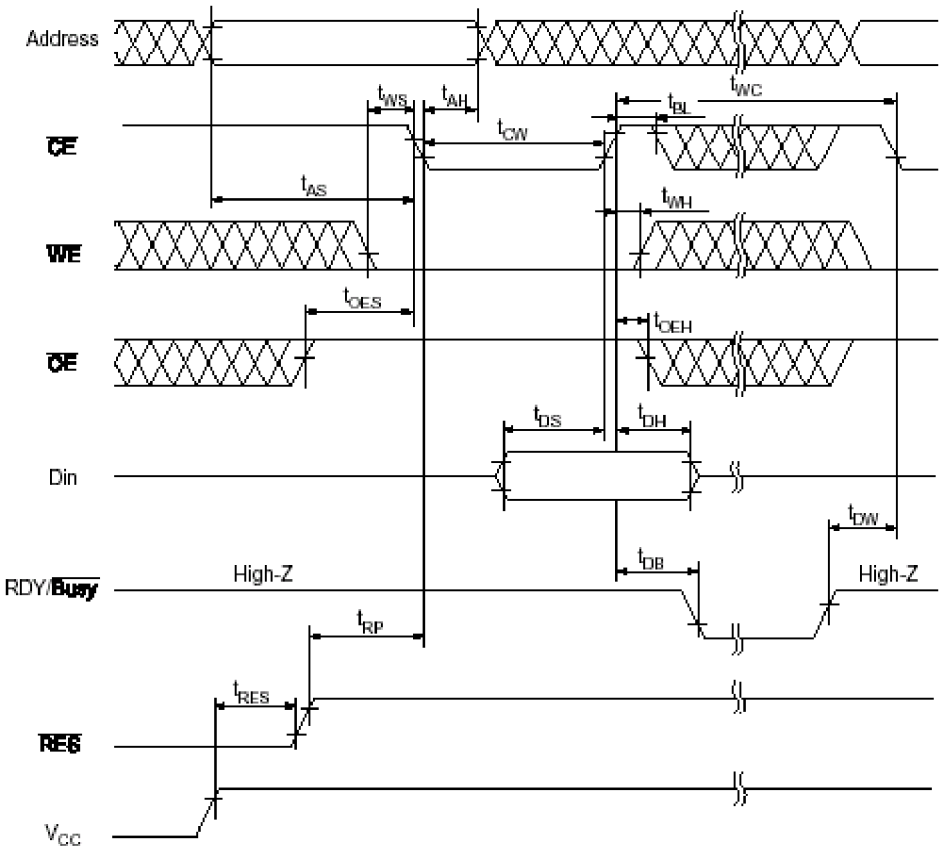

Figure 2 MMEE08001808SCC Write cycle timing (#CE control)

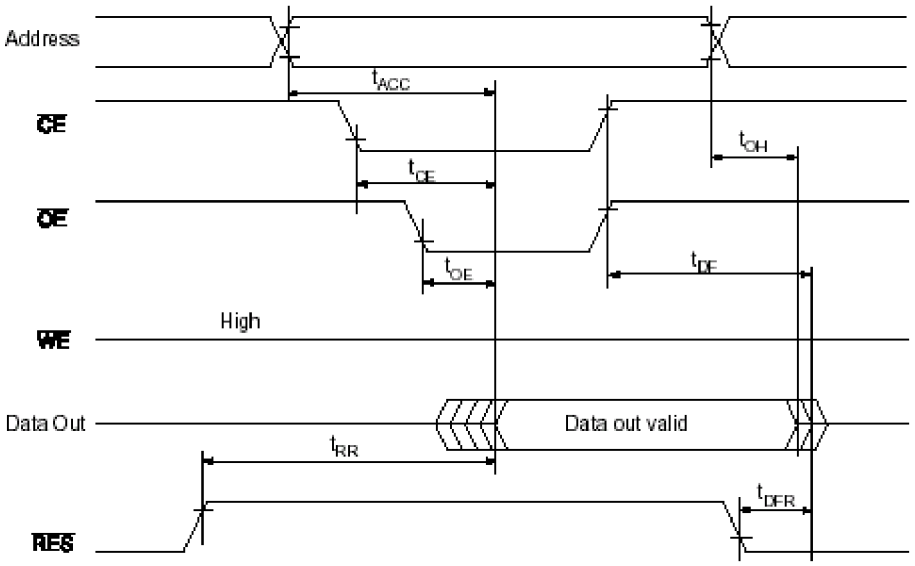

Figure 3 MMEE08001808SCC Read cycle timing

Write Cycle

Parameter	Symbol	Min*[2]	Typ	Max	Unit	Test conditions
Address setup time	t_{AS}	0	—	—	ns	
Address hold time	t_{AH}	150	—	—	ns	
$\overline{CE}$ to write setup time ($\overline{WE}$ controlled)	t_{CS}	0	—	—	ns	
$\overline{CE}$ hold time ($\overline{WE}$ controlled)	t_{CH}	0	—	—	ns	
$\overline{WE}$ to write setup time ($\overline{CE}$ controlled)	t_{WS}	0	—	—	ns	
$\overline{WE}$ hold time ($\overline{CE}$ controlled)	t_{WH}	0	—	—	ns	
$\overline{OE}$ to write setup time	t_{OES}	0	—	—	ns	
$\overline{OE}$ hold time	t_{OEH}	0	—	—	ns	
Data setup time	t_{DS}	100	—	—	ns	
Data hold time	t_{DH}	10	—	—	ns	
$\overline{WE}$ pulse width ($\overline{WE}$ controlled)	t_{WP}	250	—	—	ns	
$\overline{CE}$ pulse width ($\overline{CE}$ controlled)	t_{CW}	250	—	—	ns	
Data latch time	t_{DL}	300	—	—	ns	
Byte load cycle	t_{BLC}	0.55	—	30	µs	
Byte load window	t_{BL}	100	—	—	µs	
Write cycle time	t_{WC}	—	—	10*[3]	ms	
Time to device busy	t_{DB}	120	—	—	ns	
Write start time	t_{DW}	150*[4]	—	—	ns	
Reset protect time	t_{RP}	100	—	—	µs	
Reset high time*[5]	t_{RES}	1	—	—	µs	

Notes: 1. t_{DF} and t_{DFR} are defined as the time at which the outputs achieve the open circuit conditions and are no longer driven.
2. Use this device in longer cycle than this value.
3. t_{WC} must be longer than this value unless polling techniques or RDY/$\overline{Busy}$ are used. This device automatically completes the internal write operation within this value.
4. Next read or write operation can be initiated after t_{DW} if polling techniques or RDY/$\overline{Busy}$ are used.
5. This parameter is sampled and not 100% tested.
6. A7 to A16 are page addresses and must be same within the page write operation.
7. See AC read characteristics.

Table 1 MMEE08001808SCC write cycle parameters

Parameter	Symbol	Min	Max	Unit	Test conditions
Address to output delay	t_{ACC}	—	150	ns	$\overline{CE}$ = $\overline{OE}$ = V_{IL}, $\overline{WE}$ = V_{IH}
$\overline{CE}$ to output delay	t_{CE}	—	150	ns	$\overline{OE}$ = V_{IL}, $\overline{WE}$ = V_{IH}
$\overline{OE}$ to output delay	t_{OE}	10	75	ns	$\overline{CE}$ = V_{IL}, $\overline{WE}$ = V_{IH}
Address to output hold	t_{OH}	0	—	ns	$\overline{CE}$ = $\overline{OE}$ = V_{IL}, $\overline{WE}$ = V_{IH}
$\overline{OE}$ ($\overline{CE}$) high to output float*[1]	t_{DF}	0	50	ns	$\overline{CE}$ = V_{IL}, $\overline{WE}$ = V_{IH}
$\overline{RES}$ low to output float*[1]	t_{DFR}	0	350	ns	$\overline{CE}$ = $\overline{OE}$ = V_{IL}, $\overline{WE}$ = V_{IH}
$\overline{RES}$ to output delay	t_{RR}	0	450	ns	$\overline{CE}$ = $\overline{OE}$ = V_{IL}, $\overline{WE}$ = V_{IH}

Table 2 MMEE08001808SCC read cycle parameters

Data Protection

Unintentional programming may occur under certain circumstances, the data protection function of this device can prevent this.

Protection by #RES

When power on/off, noise on the control pins generated by external circuits may act as a trigger and turn the EEPROM to program mode by mistake. Device's data protection at V_{CC} ON/Off can be achieved by keeping EEPROM's #RES signal at V_{SS} level.

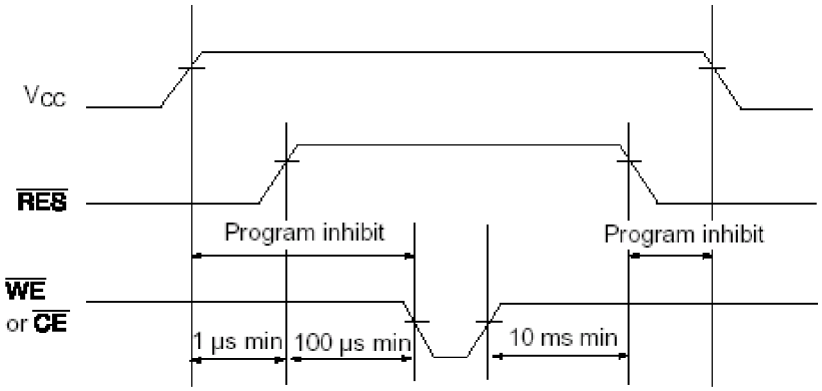

Figure 4 Data protection by #RES

Software protection

To prevent unintentional programming, the device has the software data protection (SDP) mode. The SDP mode is enabled by inputting the following 3 bytes code and write data.

Address	Data	
5555	AA	
↓	↓	
AAAA or 2AAA	55	
↓	↓	
5555	A0	
↓	↓	
Write address	Write data }	Normal data input

The SDP mode is disabled by inputting the following 6 bytes code:

Address	Data
5555	AA
↓	↓
AAAA or 2AAA	55
↓	↓
5555	80
↓	↓
5555	AA
↓	↓
AAAA or 2AAA	55
↓	↓
5555	20

Note that data can not be written during software data protection (SDP) mode.