Freescale Semiconductor
Application Note

AN4205
Rev. 0, 11/2010

Using an I²C EEPROM During MSC8157 Initialization

The MSC8157 family allows you to use an I²C EEPROM to to initialize the DSP during the reset and boot phases and run-time. During the initialization stages, the EEPROM can be used for multiple purposes.

This document provides guidance for setting up a system in which a (shared) EEPROM is used. It describes the situations in which an EEPROM can be used and examines the EEPROM contents and how the boot program parses and executes based on these contents.

NOTE

MSC8157 family refers to the MSC8157, MSC8157E, MSC8158, and MSC8158E DSPs.

This document assists system engineers to use a shared I²C EEPROM to initialize multiple MSC8157 DSPs.

Contents

1 Reset Configuration Word (RCW) Basics

The MSC8157 can read multiple RCWs from a shared EEPROM. During the power-on reset ($\overline{\text{PORESET}}$) sequence, the MSC8157 DSP reads its RCW. The RCW consists of two 32-bit values stored in the Reset Configuration Word Low Register (RCWLR) and the Reset Configuration Word High Register (RCWHWR). The RCW defines the boot and system information that configures the system, selects the boot port, and stores other values sampled during reset.

The MSC8157 allows multiple sources to provide the RCW, including:

- *Default values*. You can use one of two default RCW values as appropriate for the system. While providing the least flexibility, this option releases the user from providing an I^2C EEPROM or external logic, such as a field programmable gate array (FPGA) or complex programmable logic device (CPLD) to drive various pins with specific values during the $\overline{\text{PORESET}}$ sequence.
- *External inputs*. These pins can be used in two ways:
  - Reduced mode. The MSC8157 has 22 pins that can drive values for a subset of the 64-bit RCW. When these values are used, all the other bits have predefined values. This option, while more flexible than using the default values, still releases the user from providing an I^2C EEPROM. However, it does not allow you to set all 64 bits in the RCW.
  - Multiplexed mode. This mode uses 16 of the RC pins and the $\overline{\text{RCS\_LSEL}}$ pins to enable loading of the RCW bits in four lanes of 16-bits each. This mode allows you to set all 64 bits in the RCW without using and EEPROM.
- *I^2C EEPROM.* The MSC8157 has dedicated hardware that can access an I^2C EEPROM at a predefined address to read the RCW. This option allows you to set all 64 bits in the RCW.

To choose among these three options, you must set the RCW_SRC[2–0] inputs as indicated in **Table 1**. The RCW_SRC value is sampled into the reset status register (RSR).

Table 1. RCW_SRC Values

RCW_SRC[2–0]	Description
000	RCW_SRC[0:2] 000 Multiplexed external RCW loading. The RCW is driven by external logic on RC[15:0]. The $\overline{\text{RCS\_LSEL}}$ signals select which bits should be driven on RC[15:0]. See **Section 5.2.5.2** in the device Reference Manual for details.
001	Reserved
010	Reset configuration word is loaded from an I^2C EEPROM in 16-bit addressing mode.
011	Some bits of the reset configuration word are loaded from external pins and others by default.

When RCW_SRC is configured as 010, the RCW is read from an I^2C EEPROM. To support this feature, two more concepts are introduced:

- *Reset master.* The first MSC8157 to be configured out of PORESET is the reset master. During its PORESET interval, it reads its RCW from the EEPROM at address 0x50. Then, it starts executing its boot code and discovers that it is the reset master and starts executing as described in Section 1.2, “Reset Master” on page 4.
- *Reset slave.* The reset slave is taken out of PORESET by the reset master and then accesses the reset master, which emulates an EEPROM at address 0x57.

These two entities are required in a system because the MSC8157 reset hardware assumes that it is the sole master on the I^2C bus. As a result, as soon as the MSC8157 reset hardware finite state machine (FSM) recognizes that it should access the EEPROM, it does so without verifying that the bus is free. The reset master controls the access timing of the reset slaves so that each slave has the sole ownership of the I^2C while it reads its individual RCW.

The reset master is identified by the combination of two elements:

- During the PORESET sequence, its STOP_BS signal is pulled low.
- The RCWHR[RM] bit is set to 1, which indicates that the device is the initiator/master.

1.1 EEPROM Initialization Requirements

The following items are required for EEPROM initialization:

- For each EEPROM in the system, there must be at least one EEPROM master, which is also the reset master (RCWHR[RM] = 1).
- For each EEPROM, there can be 0 or more reset slaves, each of which reads its RCW only from the EEPROM but does not read data from it during the boot sequence. The number of reset slaves is written as a single byte in address 0x18 of the EEPROM.
- For each EEPROM, there can be 0 or more EEPROM slaves, each of which reads its RCW from the EEPROM and uses data only during the boot sequence. The number of EEPROM slaves is written as a single byte in address 0x96 of the EEPROM.
- Every EEPROM slave must also be a reset slave.
- There may be up to 15 reset slaves per EEPROM. The following equation defines the limitations on the number of slaves:

$$0 \le \text{number of EEPROM slaves} \le \text{number of reset slaves} \le 15 \qquad \textbf{Eqn. 1}$$

- The lowest-numbered reset slave must have a higher number than the highest-numbered EEPROM slave (that is, EEPROM slaves are slaves 0–4 and reset slaves are slaves 5–12).
- EEPROM slaves must be numbered sequentially starting with 0.
- All devices connected to the same EEPROM must receive the $\overline{\text{PORESET}}$ signal together (that is, no single device can proceed through the PORESET sequence without the others).
- ***For Multi-Device RCW only***. The EEPROM master can have its $\overline{\text{HRESET}}$ asserted without the reset slaves being in reset at the same time. Each reset slave can have its $\overline{\text{HRESET}}$ asserted without the master being reset.
- ***For Multi-Device RCW and Boot Using I^2C***. If the EEPROM master has its $\overline{\text{HRESET}}$ asserted, the EEPROM slaves must have their $\overline{\text{HRESET}}$ signals asserted as well. The EEPROM slaves can have $\overline{\text{HRESET}}$ asserted without the master being reset. However, there must be external logic that performs the actions that the master performs during its boot sequence. The logic can be implemented using an FPGA or other implementation.
- During the entire system initialization process using a shared EEPROM, there should be no unrelated I^2C traffic on the bus.

1.2 Reset Master

When $\overline{\text{PORESET}}$ is deasserted, the reset hardware reads RCW_SRC and STOP_BS. If the former indicates that RCW is to be read using an EEPROM and the latter equals 0, the hardware accesses an EEPROM at address 0x50. **Table 2** shows the data to be read and the address at which it is read.

Table 2. Reset Master in EEPROM

Address	Value	Use
0x00	0xAA	Training sequence for the reset hardware
0x01	0x55	
0x02	0xAA	
0x03	0xFF	Header expected by the reset hardware
0x04	0xFF	
0x05	0xFF	
0x06	RCWLR	
0x07		
0x08		
0x09		
0x0A	0xFF	Header expected by the reset hardware
0x0B	0xFF	
0x0C	0xFF	
0x0D	RCWHR	
0x0E		
0x0F		
0x10		
0x11	0x00	Header expected by the reset hardware.
0x12	0x00	Header expected by the reset hardware.
0x13	0x00	Header expected by the reset hardware.
0x14	0x00	Header expected by the reset hardware.
0x15	0x00	Header expected by the reset hardware.
0x16	0x00	Header expected by the reset hardware.
0x17	0xFF	Header expected by the reset hardware.

After the $\overline{\text{PORESET}}$ and $\overline{\text{HRESET}}$ signals are all deasserted, the cores start executing the boot code. Core 0, which is the core that performs the actual booting (see the device reference manual), reads both RSR[RCW_SRC] and RCWHR[RM] to see if it is the reset master.

NOTE

See the device specific reference manual **Section 6.1.5** for details on the reset master flow.

The reset master connects to the reset slaves STOP_BS signals using dedicated general-purpose input/output (GPIO) pins. The number of pins actually driven during the boot process is a function of the number of reset slaves (see **Table 3**). The GPIO signals are selected in the following order:

1. GPIO0—always used
2. GPIO1
3. GPIO2
4. GPIO3
5. GPIO21

Table 3. Number of Reset Slaves versus the Number of GPIO Signals

Number of GPIOs	Number of slaves	
	Direct Connect	External Logic
1	1	1
2	2	2
3	3	3, 6, 7
4	4	4, 8 9, 10,1 1, 12, 13, 14, 15
5	5	5

When up to five slaves are connected directly, the reset master deasserts the GPIO signals in order, thus directly pulling each of the slave STOP_BS signals low in sequence. After all the slaves finish reading their RCWs, the reset master drives all the GPIO signals high and then releases them together.

When there are more than five slaves, the reset master drives the value of the reset slave to be released for each slave (for example, for slave 0 of 12, GPIO[3–0] equals 0000) and external logic drives the STOP_BS signal low for the specified slave. After all slaves finish reading their RCWs, the reset master drives all the GPIO pins high and then releases them together. If there are eight slaves, even though they can be coded using only 3 GPIO lines (000 to 111), the requirement for “release all slaves” code requires the use of four GPIO lines. The external logic decodes the value on the GPIO pins and drives the slave STOP_BS signals accordingly.

1.3 Reset Slave

The reset hardware samples the value of STOP_BS and RCW_SRC during the PORESET sequence. If the RCW source is an EEPROM and STOP_BS is high, the MSC8157 is a reset slave.

The hardware waits until STOP_BS is pulled low and then accesses an EEPROM at address 0x57. This address actually accesses the reset master. The reset slave expects exactly the same data flow as the reset master from the EEPROM (see **Table 2**). When the reset slave finishes its reset phase and starts executing the boot code, it reads the boot port (BPRT) from RCWHR. If both the BPRT and the RCW source indicate an EEPROM, the reset slave waits until the reset master pull its STOP_BS high before trying to access the EEPROM to load the boot code.

2 Other Boot Ports

When booting across the Ethernet or serial RapidIO interface, the EEPROM provides support for system configuration. Addresses 0x97–0x216 of the EEPROM are dedicated to store these configurations.

2.1 Support for Boot Over the Serial RapidIO Interface

During a boot over the serial RapidIO interface, it may be necessary to program the serial RapidIO interface registers to configure the MSC8157 according to the system-specific analog parameters. A dedicated BPRT (0x02 in the serial RapidIO interface with I^2C) specifies whether the EEPROM configuration space should be parsed before enabling the serial RapidIO lanes.

If the EEPROM must be parsed, the boot code checks for pairs of 32-bit address-data values. Each 32-bit data set is written to the preceding address value. The boot code stops reading the EEPROM when the address-data pair equals (0xFFFFFFFF, 0xFFFFFFFF). Because the configuration space has 384 bytes and each address-data pair uses 8 bytes, the maximum number of such pairs is 47 (8 bytes are reserved for the end flag).

2.2 Support for Boot Over Ethernet

During a boot over Ethernet, there are two options for configuring the MAC address:

- Predefined MAC address + Device ID. The predefined address is: 0x1E-F7-D5-00-**00**-00. The device ID replaces the fifth byte of the predefined address (bolded in this example).
- MAC address is read from the EEPROM.

To support reading the MAC addresses from the EEPROM, the configuration space is divided into 6-byte chunks, which yields 64 possible MAC addresses. When the BPRT is one of the Ethernet with I^2C support options (SGMII1, RGMII1, SGMII2 and RGMII2), the boot code reads the MAC address from the EEPROM before configuring the QUICC Engine™ subsystem and accessing the dynamic host configuration protocol (DHCP) server. The MAC address is selected based on the device ID. Thus, each device reads from its specified address in the EEPROM as defined by 0x97 + (device ID × 6).

NOTE

The boot sequence resets the QUICC Engine subsystem at the end of the boot program. Therefore, the last MAC address that was read is no longer configured in the device. The user can reread the MAC address during the system initialization and thereby maintain a single MAC address for boot and run-time operations.

3 Boot Over I^2C

One of the boot port selections (BPRT = 0x0) uses an I^2C EEPROM. The boot code expects to read structures of data from the EEPROM starting at address 0x218. Each structure consists of a header, a payload, and a checksum. As the boot code reads and parses each header, it determines whether the structure targets the device. If it does, the program reads the payload into its location and then compares the checksum with the one that the boot code calculated on the fly. If the structure is not intended for the MSC8157, the boot code skips to the next structure in the EEPROM. The structure is described in **Table 4**:

Table 4. I^2C EEPROM Data Structure

Region	Size	Description
Control	1 byte	Bit 7. Check checksum. Bit 6. Reserved. Bit 5–0. Device ID or 3F (broadcast).
Payload Size	3 bytes	Size of the payload in bytes
Next Block Address	4 bytes	Address of the next structure in the EEPROM. • 0x00000000. Concatenated to this structure. • 0xFFFFFFFF. Current structure is the last one. • Other. Absolute address in EEPROM.

Table 4. I²C EEPROM Data Structure (continued) (continued)

Region	Size	Description
Destination Address	4 bytes	Address that the boot code should copy the payload to. This address is as seen by the SC3580 core.
Payload	0 to 2^{16} bytes	Variable in length.
Checksum	2 bytes	XOR of all bytes in the structure. Includes the header and payload.
$\overline{\text{Checksum}}$	2 bytes	$\overline{\text{XOR}}$ of all bytes in the structure. Includes the header and payload.

The boot code uses the control byte to determine whether the boot code should read the payload. The options are:

- Control[5–0] = RCWHR[DEVID]. The data structure is for this specific MSC8157 only.
- Control[5–0] = 0x3F. The data structure is for all MSC8157 devices connected to this EEPROM.
- Other. The data structure is not for this device. The boot code reads the value of the “Payload Size” and “Next Block Address” to calculate the location of the next data structure in the EEPROM.

Because you can specify which MSC8157 should read which data structure, the EEPROM can contain different programs in it for different devices. In such a case, use the first data structures as pointers to each of the devices codes and include no payload data.

4 Boot Patch

Patch mode is enabled if RCWHR[BP] is set. When enabled, the boot program loads the patch code from I²C EEPROM and executes it in the same way as boot-over-I²C. After the patch executes, the boot code continues to load boot code from the boot port defined by RCWHR[BPRT].

NOTE

Boot patch cannot be used if the boot port is I²C. If the boot port is defined as I²C and RCW[BP] is set, the boot code generates an error and the core goes into a Debug state.

5 I²C Bus Arbitration

The EEPROM master arbitrates the use of the I²C bus during the system initialization by having the boot code poll the value on STOP_BS at specific points in the boot process. The EEPROM master toggles the STOP_BS signals for the slaves to ensure that only one device accesses the I²C bus at any given time.

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or
+1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku
Tokyo 153-0064
Japan
0120 191014 or
+81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 010 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor
Literature Distribution Center
+1-800 441-2447 or
+1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com

Document Number: AN4205
Rev. 0
11/2010

freescale™
semiconductor