Freescale Semiconductor
Application Note

Document Number: AN3652
Rev. 1, 04/2010

i.MX35 IC Identification Module (IIM) Fusebox

The IC identification module (IIM) provides the primary user-visible mechanism for interfacing with on-chip fuse elements. Among the uses for the fuses are unique chip identifiers, mask revision numbers, cryptographic keys, and various control signals requiring a fixed value.

The purpose of this application note is to describe the i.MX35 Fusebox's electrical characteristics and to provide both a fuse-map definition and a detailed description of the necessary steps to program and read fuse bits.

Contents

freescale semiconductor

1 Fusebox Electrical Specifications

Table 1 describes the operating ranges, supply current parameters, and timing characteristics of the fusebox.

Table 1. Fusebox Supply Current Parameters

Symbol	Parameter	Min	Typ	Max	Units
FUSE_VDD[1]	Fusebox program supply voltage	3.0	3.6	3.6	V
$I_{program}$	eFuse Program Current[2:] Current to program one eFuse bit epm_avdd = 3.6 V	26	35	62	mA
I_{read}	eFuse Read Current[3] Current to read an 8-bit eFuse word vdd_fusebox = 3.3 V	—	12.5	15	mA
$t_{program}$	Program time for eFuse[4]	125	—	—	µs

1 The Fusebox read supply is connected to the supply of the full speed USBPHY. FUSE_VDD is only used for programming. Connect it to ground when not using it for programming.

2 The current $I_{program}$ is during program time ($t_{program}$).

3 The current I_{read} is present for approximately 50 ns of the read access to the 8-bit word.

4 The program length is defined by the value defined in the epm_pgm_length[2:0] bits of the IIM module. The value to program is based on a 32 Hz clock source (4 × 1/32 KHz = 125 µs).

2 IIM Fuse Definitions For Silicon Revision 2.0

Table 2 lists the fuse definitions for Silicon Revision 2.0.

Table 2. IIM Fuse Definitions (Revision 2.0)

IIM Address	IIM Bank	Fuse Name	Number of Fuses	Fuses Function	Setting	Locked by/Is Lock?[1]
0000[7]	0	FBWP	1	Fuse Bank Write Protect. Controls whether the fuse bank may be programmed.	0 Fuse bank may be programmed 1 Fuse bank may not be programmed (it is write-protected)	LOCK
0000[6]	0	FBOP	1	Fuse Bank Override Protect. Controls whether the fuse bank may be overridden.	0 Fuse bank may be overridden 1 Fuse bank may not be overridden (it is override-protected)	LOCK
0000[5]	0	FBRP	1	Fuse Bank Read Protect. Controls whether the fuse bank may be read.	0 Fuse bank may be read by software 1 Fuse bank may not be read by software (it is read-protected)	LOCK

Table 2. IIM Fuse Definitions (Revision 2.0) (continued)

IIM Address	IIM Bank	Fuse Name	Number of Fuses	Fuses Function	Setting	Locked by/Is Lock?[1]
0000[4]	0	FBSP	1	Fuse Bank Scan Protect. Controls whether the fuse bank may be scanned through JTAG (read/program).	0 Fuse bank may be scanned by JTAG 1 Fuse bank may not be scanned by JTAG (it is scan-protected)	LOCK
0000[3]	0	FBESP	1	Fuse Banks Explicit Sense Protect. Controls whether the fuse bank may be explicitly sensed. The state of this fuse controls whether the IIM state machine allows explicit sense cycles (normal, 0-stress, or 1-stress).	0 Fuse bank be explicitly sensed by software 1 Fuse bank may not be explicitly sensed by software (it is sense-protected)	LOCK
0000[2]	0	CRC_LOCK	1	Lock for rows 0018 and 0040. **Note:** Locking for MANU_CRC information is not required. This lock can be useful if MANU_CRC field will be transferred to SW or customer needs.	—	LOCK
0000[1]	0	SIREV_ LOCK	1	Lock for SI_REV[7:0] field (row 001C of fusebank 0)	0 Unlock (The controlled field can be read, sensed, burned, or overridden in the corresponding IIM register) 1 Lock (The controlled field can be read or sensed only)	LOCK
0000[0]	0	RESERVED	1	Reserved	—	LOCK
0004[7]	0	JTAG_LOCK	1	Word lock bit of JTAG related fuses (row 0004 of fusebank 0)	0 Unlock (The controlled field can be read, sensed, burned or overridden in the corresponding IIM register) 1 Lock (The controlled field can be read or sensed only)	LOCK
0004 [6:5]	0	JTAG_ SMODE[1:0]	2	JTAG Security Mode. Controls the security mode of the JTAG debug interface.	00 JTAG enable mode 01 Secure JTAG mode 11 No debug mode	JTAG_LOCK
0004[3]	0	JTAG_HEO	1	JTAG HAB Enable Override. Disallows HAB JTAG enabling. The HAB may normally enable JTAG debugging by means of the HAB_JDE-bit in the IIM SCS0 register. The JTAG_HEO-bit can override this behavior.	0 HAB may enable JTAG debug access 1 HAB JTAG enable is overridden (HAB may not enable JTAG debug access)	JTAG_LOCK

Table 2. IIM Fuse Definitions (Revision 2.0) (continued)

IIM Address	IIM Bank	Fuse Name	Number of Fuses	Fuses Function	Setting	Locked by/Is Lock?[1]
0004[2]	0	KTE	1	Kill Trace Enable. Enables tracing capability on NEXUS, ETM, and other modules.	0 Bus tracing is allowed 1 Bus tracing is allowed in case security state as defined by Secure JTAG allows it (for example, JTAG_ENABLE or NO_DEBUG)	JTAG_LOCK
0004[1]	0	SEC_JTAG_RE	1	Secure JTAG Re-enable. Overrides the Secure JTAG Bypass fuse (JTAG_BP, in this register) to limit JTAG access according to the JTAG_SMODE. The JTAG_RE signal permanently overrides JTAG security bypass, returning the device to “Secure JTAG mode.”	0 Secure JTAG Bypass fuse is not overridden (secure JTAG bypass is allowed) 1 Secure JTAG Bypass fuse is overridden (secure JTAG bypass is not allowed)	JTAG_LOCK
0004[0]	0	JTAG_BP	1	JTAG Debug Security Bypass. Blowing this fuse semi-permanently returns the device to the “JTAG Enable mode.”	0 JTAG Security bypass is not active 1 JTAG Security bypass is active	JTAG_LOCK
000C [7:6]	0	BT_SDMMC_SRC[1:0]	2	Choosing the specific eSDHC controller for booting from.	00 eSDHC-1 01 eSDHC-2 10 eSDHC-3	BOOT_LOCK
000C[5]	0	BT_ECC_SEL	1	Define 4/8-bit ECC. Also used as a fast boot mode indication for eMMC 4.3 protocol.	If the bootable device is NAND then: 0 4-bit ECC 1 8-bit ECC If the bootable device is MMC then: 0 Do not use eMMC fast boot mode. 1 Use eMMC fast boot mode.	BOOT_LOCK

Table 2. IIM Fuse Definitions (Revision 2.0) (continued)

IIM Address	IIM Bank	Fuse Name	Number of Fuses	Fuses Function	Setting	Locked by/Is Lock?[1]
000C [3:2]	0	BT_SPARE_ SIZE[1:0]	2	Specifies the size of spare bytes, for Nand Flash devices, i.e. for BT_MEM_CTL[1:0] = NAND Flash BT_SPARE_SIZE[0] is also used as a fast boot mode indication for eSD 2.10 protocol.	If the bootable device is NAND then: 00 16 bytes spare area size (For 512 B page size device) 01 64 bytes spare area size (For 2 KB page size device) 10 128 bytes spare area size (For 4 KB page size device, Samsung) 11 218 bytes spare area size (For 4 KB page size device, Micron, Toshiba) If the bootable device is SD then: BT_SPARE_SIZE[0]=0 - "FAST_BOOT" bit 29 in ACMD41 argument is 0. BT_SPARE_SIZE[0]=1 - "FAST_BOOT" bit 29 in ACMD41 argument is 1. BT_SPARE_SIZE[1] is reserved in this case.	BOOT_LOCK
000C [1:0]	0	BT_USB_ SRC	2	USB PHY selection	00 UTMI PHY 01 ULPI PHY 10 Serial PHY: ATLAS 11 Serial PHY: PHILIPS 1301	BOOT_LOCK
0010[7]	0	BT_SPARE_ FUSE	1	Spare fuse for easy metal fixes/ECOs. It is routed to SRC and unused there for a while.	—	BOOT_LOCK
0010 [6:5]	0	BT_PAGE_ SIZE[1:0]	2	NAND Flash Page Size. This field is used in conjunction with the BT_MEM_CTL[1:0] setting.	If BT_MEM_CTL = NAND Flash then 00 512 bytes 01 2K bytes 10 4K bytes 11 Reserved	BOOT_LOCK
0010[4]	0	BT_WEIM_ MUXED	1	Selects whether WEIM is in muxed mode or not.	For BT_MEM_CTL[1:0] = WEIM (NOR) 0 Not muxed 1 WEIM in Address muxed mode	BOOT_LOCK
0010[3]	0	GPIO_BT_ SEL	1	GPIO Boot Select. Determines whether certain boot fuse values are controlled from GPIO pins or IIM.	0 The fuse values are determined by GPIO pins 1 The fuse values are determined by fuses	BOOT_LOCK

Table 2. IIM Fuse Definitions (Revision 2.0) (continued)

IIM Address	IIM Bank	Fuse Name	Number of Fuses	Fuses Function	Setting	Locked by/Is Lock?[1]
0010 [2:0]	0	HAB_ TYPE[2:0]	3	Security Type	001 Engineering (allows any code to be flashed and executed, even if it has no valid signature) 100 Security Disabled (For internal/testing use) Others Production (Security On)	BOOT_LOCK
0014[7]	0	BT_SPI_ TYPE	1	Selects between EEPROM and Serial Flash type devices	0 EEPROM 1 Serial Flash (In conjunction with BT_MEM_CTL = 11 and BT_MEM_TYPE[1:0] = 11)	BOOT_LOCK
0014 [6:5]	0	BT_MEM_ TYPE[1:0]	2	Boot Memory Type. Interpreted by boot ROM SW according to BT_MEM_CTL setting. Signals could also be interpreted by HW to alter delays and timing in support of direct boot.	If BT_MEM_CTL = WEIM then 00 NOR 01 Reserved 10 Reserved 11 Reserved If BT_MEM_CTL = NAND Flash 00 3 address cycles 01 4 address cycles 10 5 address cycles 11 6 address cycles BT_MEM_CTL = ATA HDD 00 Reserved 01 P-ATA HDD 10 Reserved 11 Reserved If BT_MEM_CTL = Expansion Card Device 00 SD/MMC 01 Reserved 10 Serial ROM via I2C 11 Serial ROM via SPI	BOOT_LOCK
0014[4]	0	BT_EEPROM _CFG	1	Selects whether EEPROM device is used for load of configuration DCD data, prior to boot from other devices (not applicable when using EEPROM as boot device).	0 Use EEPROM DCD 1 Do not use EEPROM DCD	BOOT_LOCK

Table 2. IIM Fuse Definitions (Revision 2.0) (continued)

IIM Address	IIM Bank	Fuse Name	Number of Fuses	Fuses Function	Setting	Locked by/Is Lock?[1]
0014[3]	0	BT_BUS_ WIDTH	1	NAND Bus Width. This field is used with conjunction with the BT_MEM_CTL[1:0] setting **Note:** NFC - NAND has 16/8 bits WEIM - NOR has 16-bits option only, and available for debugging purpose. (Signals are muxed with display port and NAND and P-ATA)	BT_MEM_CTL[1:0] = NAND Flash 0 8 bit 1 16 bit BT_MEM_CTL[1:0] = WEIM (NOR) 0 16 bit 1 Reserved BT_MEM_CTL[1:0] = Expansion Device (SPI) 0 2-byte address SPI device (16-bit) 1 3-byte address SPI device (24-bit)	BOOT_LOCK
0014 [2:1]	0	BT_MEM_ CTL[1:0]	2	Boot Memory Control Type (memory device)	00 WEIM 01 NAND Flash 10 ATA HDD 11 Expansion Device (SD/MMC, support high storage, EEPROMs. See BT_MEM_TYPE[1:0] settings for details).	BOOT_LOCK
0014[0]	0	DIR_BT_DIS	1	Direct External Memory Boot Disable	0 Direct boot from external memory is allowed 1 Direct boot from external memory is not allowed	BOOT_LOCK
001C	0	SI_REV[7:0]	8	Silicon revision number	0x10 = Rev2.0	SIREV_LOCK
0020–003C	0	RESERVED	64	Reserved	—	LOCK
0044[7]	0	RES0_LOCK	1	Lock for rows 0048–004C of fusebank 0	0 Unlock (The controlled field can be read, sensed, burned, or overridden in the corresponding IIM register) 1 Lock (The controlled field can be read or sensed only)	LOCK
0044[6]	0	TESTER_ LOCK	1	Lock for reserved fuses in rows 0058–007C of fusebank 0.	0 Unlock (The controlled field can be read, sensed, burned, or overridden in the corresponding IIM register) 1 Lock (The controlled field can be read or sensed only)	LOCK

Table 2. IIM Fuse Definitions (Revision 2.0) (continued)

IIM Address	IIM Bank	Fuse Name	Number of Fuses	Fuses Function	Setting	Locked by/Is Lock?[1]
0044[5]	0	SRK0_LOCK	1	Lock for SRK0_HASH[255:248] fuses in row 0050 of fusebank 0	0 Unlock (The controlled field can be read, sensed, burned, or overridden in the corresponding IIM register) 1 Lock (The controlled field can be read or sensed only)	LOCK
0044[4]	0	HAB_CUS_ LOCK	1	Lock for HAB_CUS[7:0] fuses in row 0054 of fusebank 0	0 Unlock (The controlled field can be read, sensed, burned, or overridden in the corresponding IIM register) 1 Lock (The controlled field can be read or sensed only)	LOCK
0044 [3:1]	0	RESERVED	3	Fuses available for software/customers	—	No lock
0044[0]	0	BOOT_LOCK	1	Lock for boot fuses in rows 000C–0018 of fusebank 0	0 Unlock (The controlled field can be read, sensed, burned, or overridden in the corresponding IIM register) 1 Lock (The controlled field can be read or sensed only)	LOCK
0048–004C	0	BI_VER[15:0]	16	Boot Image Version. Indicate the version of the code image authenticated by the High Assurance Boot. The fuse value must match the version of the image stored in non-volatile memory.	—	RES0_LOCK
0050	0	SRK0_HASH [255:248]	8	Most significant byte of 256-bit hash value of AP super root key (SRK0_HASH)	—	SRK0_LOCK
0054	0	HAB_ CUS[7:0]	8	HAB Customer Code—Selects customer code, as input to HAB.	—	HAB_CUS_ LOCK
0000[7]	1	FBWP	1	Fuse Bank Write Protect. Controls whether the fuse bank may be programmed.	0 Fuse bank may be programmed 1 Fuse bank may not be programmed (it is write-protected)	LOCK
0000[6]	1	FBOP	1	Fuse Bank Override Protect. Controls whether the fuse bank may be overridden.	0 Fuse bank may be overridden 1 Fuse bank may not be overridden (it is override-protected)	LOCK

Table 2. IIM Fuse Definitions (Revision 2.0) (continued)

IIM Address	IIM Bank	Fuse Name	Number of Fuses	Fuses Function	Setting	Locked by/Is Lock?[1]
0000[5]	1	FBRP	1	Fuse Bank Read Protect. Controls whether the fuse bank may be read.	0 Fuse bank may be read by software 1 Fuse bank may not be read by software (it is read-protected)	LOCK
0000[4]	1	FBSP	1	Fuse Bank Scan Protect. Controls whether the fuse bank may be scanned through JTAG (read/program). **Note:** Fusebanks with SCC_KEY and SJC_RESP shall be unscannable.	0 Fuse bank may be scanned by JTAG 1 Fuse bank may not be scanned by JTAG (it is scan-protected)	LOCK
0000[3]	1	FBESP	1	Fuse Banks Explicit Sense Protect. Controls whether the fuse bank may be explicitly sensed. The state of this fuse controls whether the IIM state machine allows explicit sense cycles (normal, 0-stress, or 1-stress).	0 Fuse bank be explicitly sensed by software 1 Fuse bank may not be explicitly sensed by software (it is sense-protected)	LOCK
0000[2]	1	L2VAL _LOCK	1	Lock for rows 0078–007C of fusebank 1.	0 Unlock (The controlled field can be read, sensed, burned, or overridden in the corresponding IIM register) 1 Lock (The controlled field can be read or sensed only)	LOCK
0000[1]	1	SJC_RESP_ LOCK	1	Lock for SJC_RESP[55:0] fuses in rows 005C–0074 of fusebank 1. When locked, the fuses cannot be read, sensed, overridden or written.	0 Unlock (SJC_RESP[55:0] can be read, sensed, burned, or overridded in the corresponding IIM register) 1 Lock (SJC_RESP[55:0] cannot be read, sensed, overridden nor written)	LOCK
0000[0]	1	SCC_LOCK	1	Lock for the SCC_KEY[167:0] fuses in rows 0004–0054 of fusebank 1. When unblown, SCC_KEY[167:0] cannot be read or sensed. When blown, SCC_KEY[167:0] cannot be read, sensed, overridden, or written.	0 Unlock (SCC[167:0] cannot be read, sensed, scanned, but can be burned or overridded in the corresponding IIM register) 1 Lock (SCC[167:0] cannot be read, sensed, scanned, overridden nor written)	LOCK
0004–0054	1	SCC_ KEY[167:0]	168	SCC Secret Key. Protected by SCC_LOCK. Neither readable nor explicitly sensible by the default.	Random number for every part	SCC_LOCK

Table 2. IIM Fuse Definitions (Revision 2.0) (continued)

IIM Address	IIM Bank	Fuse Name	Number of Fuses	Fuses Function	Setting	Locked by/Is Lock?[1]
0058	1	RESERVED	8	Fuses available for software/customers	—	No lock
0057–0074	1	SJC_RESP[55:0]	56	Response reference value for the secure JTAG controller	—	SJC_RESP_LOCK (locks also for read, scan and sense)
0000[7]	2	FBWP	1	Fuse Bank Write Protect. Controls whether the fuse bank may be programmed.	0 Fuse bank may be programmed 1 Fuse bank may not be programmed (it is write-protected)	LOCK
0000[6]	2	FBOP	1	Fuse Bank Override Protect. Controls whether the fuse bank may be overridden.	0 Fuse bank may be overridden 1 Fuse bank may not be overridden (it is override-protected)	LOCK
0000[5]	2	FBRP	1	Fuse Bank Read Protect. Controls whether the fuse bank may be read.	0 Fuse bank may be read by software 1 Fuse bank may not be read by software (it is read-protected)	LOCK
0000[4]	2	FBSP	1	Fuse Bank Scan Protect. Controls whether the fuse bank may be scanned using JTAG (read/program).	0 Fuse bank may be scanned by JTAG 1 Fuse bank may not be scanned by JTAG (it is scan-protected)	LOCK
0000[3]	2	FBESP	1	Fuse Banks Explicit Sense Protect. Controls whether the fuse bank may be explicitly sensed. The state of this fuse controls whether the IIM state machine allows explicit sense cycles (normal, 0-stress, or 1-stress).	0 Fuse bank be explicitly sensed by software 1 Fuse bank may not be explicitly sensed by software (it is sense-protected)	LOCK
0000[2]	2	RESERVED	1	Fuse available for software/customers	—	No lock
0000[1]	2	SRK0_LOCK88	1	Lock for SRK0_HASH[247:160] fuses in rows 0004–002C of fusebank 2	0 Unlock (The controlled field can be read, sensed, burned, or overridden in the corresponding IIM register) 1 Lock (The controlled field can be read or sensed only)	LOCK

Table 2. IIM Fuse Definitions (Revision 2.0) (continued)

IIM Address	IIM Bank	Fuse Name	Number of Fuses	Fuses Function	Setting	Locked by/Is Lock?[1]
0000[0]	2	SRK0_ LOCK160	1	Lock for SRK0_HASH[159:0] fuses in rows 0030–007C of fusebank 2	0 Unlock (The controlled field can be read, sensed, burned, or overridden in the corresponding IIM register) 1 Lock (The controlled field can be read or sensed only)	LOCK
0004–002C	2	SRK0_HASH [247:160]	88	88 bits of 256-bit hash value of AP super root key, SRK0_HASH[247:160]	—	SRK0_ LOCK88
0030–007C	2	SRK0_HASH [159:0]	160	160 bits of 256-bit hash value of AP super root key, SRK0_HASH[159:0]	—	SRK0_ LOCK160

1 LOCK means it locks another eFuse, for example, SCC_LOCK locks SCC[167:0]. Another eFuse name means that it is locked by that eFuse, for example, SCC[167:0] are locked by SCC_LOCK.

Table 3 provides a fuse map for Silicon Revision 2.0. Shading indicates burned or locked fuses. The address offset is from the IIM Base Address added to the Fuse Bank Base Address.

Table 3. Fuses Map (Revision 2.0)

Address Offset	7	6	5	4	3	2	1	0	Initially Burned Values[1]
Fuse Bank: 0									
0000	FBWP	FBOP	FBRP	FBSP	FBESP	CRC_LOCK	SIREV_ LOCK	RESERVED	0000 0011
0004	JTAG_ LOCK	JTAG_SMODE [1:0]		RESERVED	JTAG_ HEO	KTE	SEC_JTAG_ RE	JTAG_BP	000x 0000
0008	RESERVED								xxxx xxxx
000C	BT_SDMMC_SRC [1:0]		BT_ECC _SEL	RESERVED	BT_SPARE_SIZE [1:0]		BT_USB_SRC [1:0]		0000 0000
0010	BT_SPARE _FUSE	BT_PAGE_SIZE [1:0]		BT_WEIM_ MUXED	GPIO_ BT_SEL	HAB_TYPE [2]	HAB_TYPE [1]	HAB_TYPE [0]	0000 0001
0014	BT_SPI_ TYPE	BT_MEM_TYPE [1:0]		BT_EEPRO M_CFG	BT_BUS _WIDTH	BT_MEM_CTL [1:0]		DIR_BT_DI S	0000 0000
0018	RESERVED								xxxx xxxx
001C	SI_REV[2]7:0]								0001 0000

Table 3. Fuses Map (Revision 2.0) (continued)

Address Offset	7	6	5	4	3	2	1	0	Initially Burned Values[1]
0020	RESERVED								xxxx xxxx
0024									xxxx xxxx
0028									xxxx xxxx
002C									xxxx xxxx
0030									xxxx xxxx
0034									xxxx xxxx
0038									xxxx xxxx
003C									xxxx xxxx
0040	RESERVED								xxxx xxxx
0044	RES0_ LOCK	TESTER _ LOCK	SRK0_ LOCK	HAB_CUS_ LOCK	—	—	—	BOOT_ LOCK	0100 0000
0048	BI_VER[15:8][3]								0000 0000
004C	BI_VER[7:0][2]								0000 0000
0050	SRK0_HASH[255:248][4]								0000 0000
0054	HAB_CUS[7:0]								0000 0000
0058	RESERVED								xxxx xxxx
005C	RESERVED								xxxx xxxx
0060	RESERVED								xxxx xxxx
0064	RESERVED								xxxx xxxx
0068	RESERVED								xxxx xxxx
006C	RESERVED								xxxx xxxx
0070	RESERVED								xxxx xxxx
0074	RESERVED								xxxx xxxx
0078	RESERVED								xxxx xxxx
007C	RESERVED								xxxx xxxx
Fuse Bank 1									
0000	FBWP	FBOP	FBRP	FBSP	FBESP	L2VAL_ LOCK	SJC_RESP_ LOCK	SCC_LOCK	0101 1101
0004	RESERVED[5]								xxxx xxxx
0008	RESERVED[5]								xxxx xxxx
000C	RESERVED[5]								xxxx xxxx
0010	RESERVED[5]								xxxx xxxx

Table 3. Fuses Map (Revision 2.0) (continued)

Address Offset	7	6	5	4	3	2	1	0	Initially Burned Values[1]
0014	RESERVED[5]								xxxx xxxx
0018	RESERVED[5]								xxxx xxxx
001C	RESERVED[5]								xxxx xxxx
0020	RESERVED[5]								xxxx xxxx
0024	RESERVED[5]								xxxx xxxx
0028	RESERVED[5]								xxxx xxxx
002C	RESERVED[5]								xxxx xxxx
0030	RESERVED[5]								xxxx xxxx
0034	RESERVED[5]								xxxx xxxx
0038	RESERVED[5]								xxxx xxxx
003C	RESERVED[5]								xxxx xxxx
0040	RESERVED[5]								xxxx xxxx
0044	RESERVED[5]								xxxx xxxx
0048	RESERVED[5]								xxxx xxxx
004C	RESERVED[5]								xxxx xxxx
0050	RESERVED[5]								xxxx xxxx
0054	RESERVED[5]								xxxx xxxx
0058	RESERVED								0000 0000
005C	SJC_RESP[55:48][6]								0000 0000
0060	SJC_RESP[47:40][6]								0000 0000
0064	SJC_RESP[39:32][6]								0000 0000
0068	SJC_RESP[31:24][6]								0000 0000
006C	SJC_RESP[23:16][6]								0000 0000
0070	SJC_RESP[15:8][6]								0000 0000
0074	SJC_RESP[7:0][6]								0000 0000
0078	RESERVED								xxxx xxxx
007C	RESERVED								xxxx xxxx
Fuse Bank 2[7]									
0000	FBWP	FBOP	FBRP	FBSP	FBESP	—	SRK0_ LOCK88	SRK0_ LOCK160	0000 0000
0004	SRK0_HASH[247:240]								0000 0000
0008	SRK0_HASH[239:232]								0000 0000

Table 3. Fuses Map (Revision 2.0) (continued)

Address Offset	7	6	5	4	3	2	1	0	Initially Burned Values[1]
000C	SRK0_HASH[231:224]								0000 0000
0010	SRK0_HASH[223:216]								0000 0000
0014	SRK0_HASH[215:208]								0000 0000
0018	SRK0_HASH[207:200]								0000 0000
001C	SRK0_HASH[199:192]								0000 0000
0020	SRK0_HASH[191:184]								0000 0000
0024	SRK0_HASH[183:176]								0000 0000
0028	SRK0_HASH[175:168]								0000 0000
002C	SRK0_HASH[167:160]								0000 0000
0030	SRK0_HASH[159:152]								0000 0000
0034	SRK0_HASH[151:144]								0000 0000
0038	SRK0_HASH[143:136]								0000 0000
003C	SRK0_HASH[135:128]								0000 0000
0040	SRK0_HASH[127:120]								0000 0000
0044	SRK0_HASH[119:112]								0000 0000
0048	SRK0_HASH[111:104]								0000 0000
004C	SRK0_HASH[103:96]								0000 0000
0050	SRK0_HASH[95:88]								0000 0000
0054	SRK0_HASH[87:80]								0000 0000
0058	SRK0_HASH[79:72]								0000 0000
005C	SRK0_HASH[71:64]								0000 0000
0060	SRK0_HASH[63:56]								0000 0000
0064	SRK0_HASH[55:48]								0000 0000
0068	SRK0_HASH[47:40]								0000 0000
006C	SRK0_HASH[39:32]								0000 0000
0070	SRK0_HASH[31:24]								0000 0000
0074	SRK0_HASH[23:16]								0000 0000
0078	SRK0_HASH[15:8]								0000 0000
007C	SRK0_HASH[7:0]								0000 0000

1 Describes the state of the eFuses when the device is manufactured: 0 = Unblown, 1 = Blown
2 According to the chip silicon revision, 0x10 = Rev 2.0
3 Boot Image Version
4 Most significant byte of 256-bit SRKO_HASH

[5] Unreadable SCC Key
[6] Response reference value for the secure JTAG controller. Cannot be read, overridden, or programmed after SJC_REP_LOCK is blown.
[7] Bits 247:0 of Hash of super-root key stored in FLASH

3 Fuse Addressing

The IIM module base address register is 0x53FF_0000. All registers are 8-bit wide, but addressable on 32-bit boundaries.

The IIM contains three fuse banks. Each bank contains 256 8-bit rows for a total of 2048 fuses per bank. Each fuse row is addressable on a 32-bit boundary, meaning that Fuse Row Index 0 is at Fuse Row Offset 0x0, Fuse Row Index 1 is at Fuse Row Offset 0x4, Fuse Row Index 2 is at Fuse Row Offset 0x8, and so on.

Fuse Bank 0 is located at offset 0x0800 from the IIM base address (0x53FF_0000). Fuse Bank 1 is located at offset 0x0C00. Fuse Bank 2 is located at offset 0x1000.

For example, the absolute address of Fuse Row 0xC at Bank 0 is 0x53FF_0000 + 0x0800 + 0xC = 0x53FF_080C. Keep in mind that the Fuse Row Index is 3.

4 Fuse Programming Procedure

Fuse programming is accomplished using the following procedure:

1. Define the fuse bit address by writing to the upper address (UA) and lower address (LA) registers.
   - UA register is located at 0x53FF_0014. LA register is located at 0x53FF_0018.
   - UA[5:3] selects the fuse bank.
   - UA[2:0] provides the most significant portion of the Fuse Row Index within the bank.
   - LA[7:3] provides the least significant portion of the Fuse Row Index within the bank.
   - LA[2:0] selects the bit position within the selected row (or fuse byte).
2. Write 0xAA to the program protection register (PRG_P), which is located at address 0x53FF_0028.

   This register is used to protect against accidental fuse programming. The fuses can be blown only when the value of this register is 0xAA. Software should only program this register to 0xAA while actively blowing fuses. After the program operation is complete, immediately reprogram this register to a different value.
3. Enable and start fuse programming using the fuse control register (FCTL), which is located at address 0x53FF_0010. Writing 0x71 to this register commands the IIM to blow the fuse.
4. Wait until fuse programming is finished by IIM. When bit 1 of the status register (STAT) equals one, the program operation has finished. STAT is located at address 0x53FF_0000.
5. Clear bit 1 of STAT by writing **1** to it.
6. Write any value other than 0xAA to PRG_P register to prevent inadvertent fuse programming.
7. Check if there were errors by reading the content of the module errors register (ERR), which is located at address 0x53FF_0008. If bits[7:1] are all zeros, no error occurred.

5 Fuse Sensing Procedure

While fuse programming blows fuses one bit at a time, fuse sensing or reading always works at a byte boundary. There are two methods for reading fuse values: direct register address dereferencing and explicit fuse sensing.

5.1 Reading Fuses Using Direct Address Dereferencing

Direct register address dereferencing is accomplished by calculating the absolute fuse address and reading the content of the register using standard pointer dereferencing methods. For example, the fuse row located at offset 0x10 from fuse bank 0 can be read with the following C code:

```
char fuse_byte = *(char *) (0x53FF0000 + 0x0800 + 0x10);
```

5.2 Reading Fuses Using Explicit Sensing

Explicit sensing is accomplished using the following procedure:

1. Write something other than 0xAA to PRG_P register to prevent fuses from being blown inadvertently.
2. Define the fuse row (or byte) address by writing to the upper address and lower address registers.
   - UA[5:3] selects the fuse bank.
   - UA[2:0] provides the most significant portion of the Fuse Row Index within the bank.
   - LA[7:3] provides the least significant portion of the Fuse Row Index within the bank.
   - LA [2:0] is disregarded during fuse sensing operations because it always reads all 8 bits within the given fuse row.
3. Set the SENS strength to the fuse control register (FCTL), which will trigger a sense cycle. Only one of bits[3:1] can be set to 1 to start a read cycle.
4. Wait until SNSD bit from the status register (STAT[0]) equals one.
5. Write **1** to SNSD bit from the status register (STAT[0]) to clear it.
6. Check for errors by reading the content of the module errors register (ERR), which is located at address 0x53FF_0008. If bits[7:1] are all zeros, no error occurred.
7. Finally, the fuse row (or byte) value can be retrieved by reading the SDAT register, which is located at address 0x53FF_001C.

6 Sample Code

The following source code describes how to do fuse programming and fuse sensing in C programming language.

```
#define setmem8(address, value) *(volatile unsigned char *)address = (unsigned char)value
#define readmem8(address) (*(volatile unsigned char *)address)

//Important IIM register definitions.
#define IIM_BASE_ADDRESS (0x53FF0000)
```

```
#define IIM_UPPER_ADDRESS_REG       (IIM_BASE_ADDRESS + 0x14)
#define IIM_LOWER_ADDRESS_REG       (IIM_BASE_ADDRESS + 0x18)
#define IIM_PRG_P_REG               (IIM_BASE_ADDRESS + 0x28)
#define IIM_FCTL_REG                (IIM_BASE_ADDRESS + 0x10)
#define IIM_STAT_REG                (IIM_BASE_ADDRESS + 0x00)
#define IIM_ERR_REG                 (IIM_BASE_ADDRESS + 0x08)
#define IIM_SDAT_REG                (IIM_BASE_ADDRESS + 0x1C)

typedef enum
{
        ESNS_1 = 2,
        ESNS_0 = 4,
        ESNS_N = 8 //Normal Sensing
}e_sens_strength;

typedef enum
{
        FUSE_BANK_0,
        FUSE_BANK_1,
        FUSE_BANK_2,
        MAX_FUSE_BANK //MX35 only has 3 fuse banks, 0, 1 and 2.
}e_fuse_banks;

//fuse_bank is the fuse bank index. MX35 only has 3 banks. That's why they've been enumerated.
//fuse_row_addr: represents the address of the row where we are going to blow a fuse.
//       Each Row addresses one byte worth of data. This means one row contains 8 fuses ready
to be blown!
//       Row Index 0 is Row Address 0x0
//       Row Index 1 is Row Address 0x4
//       Row Index 2 is Row Address 0x8
//       Row Index 3 is Row Address 0xC
//               ... And so on

//fuse_bit_addr: A number from 0 to 7 that represent the fuse bit index that we are going to
blow or sense.
static void set_fuse_address(e_fuse_banks fuse_bank, unsigned short fuse_row_addr, unsigned
char fuse_bit_addr)
{
        unsigned char upper_addr;
        unsigned char lower_addr;
        unsigned char fuse_index;

        //A fuse bank contains 256 fuse rows. for a total of 2048 fuse bits.
        //Dividing the fuse_address by 4 will give us the fuse index.
        fuse_index = (unsigned char)(fuse_row_addr >> 2);

        upper_addr = ( (unsigned char)fuse_bank << 3);
        upper_addr |= ((fuse_index >> 5) & 0x7);

        lower_addr = ((fuse_index & 0x1F) << 3);
        lower_addr |= (fuse_bit_addr & 0x7);

        //write address to UA LA register
        setmem8(IIM_UPPER_ADDRESS_REG, upper_addr);
        setmem8(IIM_LOWER_ADDRESS_REG, lower_addr);
}

//Returns 0 when Successful.
```

```
unsigned char fuse_bit_program(e_fuse_banks fuse_bank,
                                    unsigned char fuse_row_addr,
                                    unsigned char fuse_bit_addr)

{
        unsigned char error;

        //Define the fuse bit address that we want to program.
        set_fuse_address(fuse_bank, fuse_row_addr, fuse_bit_addr);

        //write 0xAA to Program Protection Register (PRG_P) register
        //The value 0xAA is an arbitrarily chosen value that needs to be written
        //to the register so the fuse program operation actually works.
        //This helps prevent to blow fuse by mistake.

        setmem8(IIM_PRG_P_REG, 0xAA);
        //Enable and Start Fuse Programming via Fuse Control Register(FCTL)
        setmem8(IIM_FCTL_REG,0x71);

        //Wait until fuse blowing is finished.
        while( (readmem8(IIM_STAT_REG) & 0x2) == 0 );

        //Write 1 to clear PRGD bit
        setmem8(IIM_STAT_REG,0x02);

        //Very good to do too. for safety.
        setmem8(IIM_PRG_P_REG, 0x0);

        //Check Error status.
        error = readmem8(IIM_ERR_REG);
        if (error & 0xFE)

        {

                //Clear Error Status Register. By writing the same value we got.
        //These are Clear-on-Write type of bits.
                setmem8(IIM_ERR_REG, error);
                //Some error occurred.
                return error;
        }

        //No error at all.
        return 0;
}

//Returns the value of all the fuses (8 fuses or 8 bits)
// contained in fuse_row_addr.
unsigned char fuse_byte_read(e_fuse_banks fuse_bank,
                             unsigned char fuse_row_addr,
                             e_sens_strength sens_strength)

{
        unsigned char error;
        unsigned char fuse_byte_value;

        //Just in case.
        // Write something different than 0xAA to IIM_PRG_P_REG to prevent
        //fuses from being blown inadvertently.
        setmem8(IIM_PRG_P_REG, 0x0);

        //Define the fuse row address that we want to read.
        set_fuse_address(fuse_bank, fuse_row_addr, 0);
```

```
        //Set the SENS strength to the Fuse Control Register
        // which will also trigger a sense cycle.

        setmem8(IIM_FCTL_REG, sens_strength);

        // wait for SNSD bit to set. While this is 0,
        // it means the sensing has not finished.
        while ( (readmem8(IIM_STAT_REG) & 0x1) == 0 );

        //Write 1 to SNSD bit in the STAT register. This is the way to clear it.
        setmem8(IIM_STAT_REG,0x01);

        //Was there an error??
        error = readmem8(IIM_ERR_REG);
        if (error & 0xFE)

        {
        //Clear Error Status Register. By writing the same value we got.
        //These are clear on Write type of bits.
                setmem8(IIM_ERR_REG, error);

                while (1); //READ ERROR. LOOP ForEver.
        }

        //Read the byte we wanted.
        fuse_byte_value = readmem8(IIM_SDAT_REG);

        return fuse_byte_value;

}
```

The following source code shows how to blow the BT_SDMMC_SRC[0] fuse, blowing this fuse permanently configures the i.MX35 to boot from e-SDHC2 port when booting from SD/MMC:

```
        //Blow the BT_SDMMC_SRC[0]  Fuse. Bit 6, of Fuse Row 0xC in FuseBank 0
        unsigned char error = fuse_bit_program(FUSE_BANK_0, 0xC, 6);
```

The following source code shows how to explicitly sense the BT_SDMMC_SRC[1:0] fuses using the **fuse_byte_read** function defined above:

```
        unsigned char fuse_byte_value = fuse_byte_read(FUSE_BANK_0, 0xC, ESNS_N);
        unsigned char bt_sdmmc_src = (fuse_byte_value >> 6);
```

As mentioned before, the same fuse row can also be read using direct register address dereferencing, which is shown in the following code:

```
        unsigned char fuse_byte_value = *(unsigned char *)(0x53FF0000 + 0x0800 + 0xC);
        unsigned char bt_sdmmc_src = (fuse_byte_value >> 6);
```

7 Revision History

Table 4. Document Revision History

Rev. Number	Date	Substantive Change(s)
1	04/2010	Updated the Setting for the BT_MEM_CTL = NAND Flash value of the BT_MEM_TYPE[1:0] Fuse (IIM Address = 0014[6:5]) in Table 2
0	07/2009	Initial release

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 10 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: AN3652
Rev. 1
04/2010