QUICC Engine Microcode RAM Packages Format and Software Loader

This document summarizes the format for all RAM-based microcode packages used by the software drivers and the conventions which are used to integrate each package with the unified drivers environment which is not dependent on device version.

1 Input for Microcode Loader

Each microcode package provided by Freescale is in the format of a header file ".h" loaded by the software drivers into the appropriate RAM location in the appropriate setup, for example either a common RAM for all the RISC engines or a distributed RAM for each RISC. This '.h' file should be included in the Use-Case (application) file only. It contains some handshake information used by the software drivers dynamically. The name of the file suggests its functionality, the device name, and the revision of the device which the package is applicable to.

1.1 83xx Devices (ROM and RAM Based)

1.1.1 Compilation Flags

The loader contains the following structures and compilation flags ('defines'):

1. `MPC83xx_Rxx_UC_PATCH_COMMON_IRAM`. For example for the MPC8360 rev 2.0 device it should look like this:
   `MPC8360_R2_0_UC_PATCH_COMMON_IRAM`. This define controls, for each of the QUICC Engine RISCs, whether the IRAM is shared between the RISCs or one half is dedicated per RISC.

When the code size is small enough so that it can fit on half of the available IRAM, it could be instantiated twice using this flag. This method also enables loading asymmetrical code which is dedicated to each RISC.

The values that can be used here are the following:

e_QE_COMMON_IRAM—Single and common microcode for all the RISCs

e_QE_COMMON_PATCH—Single microcode for all the RISCs at different offsets per RISC

e_QE_MULTIPLE_PATCHES—Different microcode and traps for each of the RISCs

2. MPC83xx_Rxx_UC_PATCH or. Following this define is the actual listing of the opcode of the microcode package itself. The microcode must be in one single array and contiguous. If different packages are used for different RISCs (for example, e_QE_MULTIPLE_PATCHES) the file will contain numerous definitions and names for each of the packages which is included.
3. MPC83xx_Rxx_UC_PATCH_OFFSETS. This is an array of up to 2 entries (for a 2 RISCs machine) that notify the start address of the microcode opcodes for each RISC. If common IRAM mode is used, for example e_QE_COMMON_IRAM, there is a single entry in this array.
4. MPC83xx_Rxx_UC_PATCH_TRAPS. Defines the trap registers content. These are the locations in ROM where the code is trapped. It contains the content of up to 16 trap registers. In case there is a dedicated package per RISC (for example e_QE_MULTIPLE_PATCHES) there will be multiple instantiation for this define, each containing up to 16 trap registers for each RISC.
5. MPC83xx_Rxx_UC_PATCH_VIRTUAL_TRAPS. Defines the conditional branches in the microcode that are currently being used by this microcode package. This structure contains 5 words that the application should write to a specific location. For detailed description see Section 1.1.2, “Virtual Microcode Traps”.
6. MPC83xx_Rxx_UC_PATCH_EXT_MODES. Defines extended modes that can affect the software driver operation. These modes are defined specifically for each package with some conventions and with agreement with the software drivers. Examples are the impact on the driver host commands on some of the microcode packages. For a more detailed description see Section 1.1.3, “Extended Modes”.
7. MPC83xx_Rxx_UC_PATCH_INFO. Brief description of the patch, includes the patch name, patch revision and chip revision.

1.1.2 Virtual Microcode Traps

These virtual traps are conditional branches in the microcode. These are “soft” provisional traps that were introduced in the ROM code to enable higher flexibility and save hardware traps if they are being used. If new features are activated or an issue is being fixed in the RAM package using these virtual traps, they should be activated. This data structure signals to the microcode which of these virtual traps is active.

This structure contains 6 words that the application should copy to some specific locations which have been defined. Table 1 describes this structure.

Table 1. Virtual Patch Traps

Offset in Array	Protocol	Destination Offset in the PRAM	Size of Operand
0x0	Ethernet interworking global PRAM	0xF8	32 bits
0x4	ATM interworking global PRAM	0xF8	32 bits
0x8	PPP interworking global PRAM	0xF8	32 bits
0xC	Ethernet Rx global parameter RAM. Only 8 LSB are valid.	0x22	8 bits
0x10	ATM global parameters table. Only 8 LSB are valid.	0x28	8 bits
0x14	Insert frame global PRAM	0xF8	32 bits

1.1.3 Extended Modes

This is a double-word bit array (64 bits) that defines special functionality which has an impact on the software drivers. Each bit has its own impact and has special instructions for the software associated with it. This structure is described in Table 2.

Table 2. Extended Mode

Bit	Name	Description
0	General push command	Indicates that **prior** to each host command given by the application the software **must** assert a special host command (push command). Push command description: **CECDR** = 0x0080_0000 **CECR** = 0x01C1_000F
1	UCC ATM Rx-Init push command	Indicates that **after** issuing ATM Rx Init command, the host must issue another special command (push command) and immediately following that re-issue the ATM Rx Init command. (This makes the sequence of initializing the ATM receiver a sequence of three host commands) Push command description: **CECDR** = 0x0080_0000 **CECR** = 0x01C1_000F
2	Add/Remove command validation	Indicates that **following** the specific host command: "Add/Remove entry in Hash Lookup Table" used in Interworking setup, the user **must** issue another command. The additional command description: **CECDR** = 0xCE00_0003 **CECR** = 0x01C1_0F58

Table 2. Extended Mode (continued)

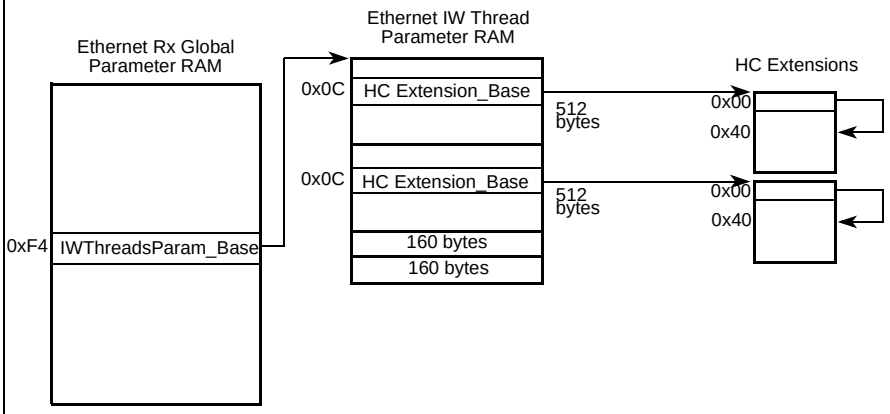
Bit	Name	Description
3	HC for Ethernet memory allocation	Indicates that the software has to initialize some pointers in the Ethernet IW thread pages which are used when Header Compression is activated. Follow these instructions: 1. Do not increase the IW Thread page by 128 bytes as indicated in HC_en bit in InitEnet Command Parameter Table For Ethernet Init Host Command. 2. A pointer to HC Extension of 128 bytes (in multiuser RAM memory space) must be programmed in each Ethernet IW thread Parameter RAM. The pointer is programmed at offset 0xC from IWThreadsParam_Base (programmed in Ethernet Global Parameter RAM). This pointer is aligned to 128 bytes. Note that IW Thread Parameter RAM is partitioned in 512 and 160 byte chunks as depicted in the figure below. 3. In each HC Extension at offset 0x0 write a pointer of value HC_Entension_Base + 0x40 
4	Push command for Tx Enet Thread	Indicates that after issuing Eth Init Tx command, user must issue this command for each SNUM of Enet Tx thread. Push command description: **CECDR** = 0x0080_0003 **CECR** = 0x7'b{0}, 8'b{Enet Tx thread SNUM}, 1'b{1}, 12'b{0}, 4'b{1}

Table 2. Extended Mode (continued)

Bit	Name	Description
5	QE_ENET22	Indicates that QE_ENET22 bug is fixed. 0 The bug is not fixed in IRAM based package or no IRAM-based package is downloaded in IRAM. In this case the SW workaround must be applied. 1 The bug is fixed in IRAM based package. No software workaround should be applied. The software workaround is: To avoid the situation described above, at least one of these measures needs to be taken: 1. Only one Rx thread is enabled. 2. The Rx VFIFO size should be equal to 0.5 kbytes. 3. If the Rx VFIFO size is between 0.5 to 1.1 Kbytes, at least 4 threads must be enabled. 4. If the Rx VFIFO size is between 1.1 to 1.6 Kbytes, at least 6 threads must be enabled. 5. If the Rx VFIFO size is between 1.6 to 2.2 Kbytes, at least 8 threads must be enabled. 6. If the Rx VFIFO size is between 2.2 to 4.5 Kbytes, the user must allocate 512 bytes (must be initialized to zero) in the Multiuser RAM. The base address for this area must be 512 bytes aligned. Immediately after the Ethernet Rx INIT command was ended (and before the UCC receiver is enabled), this address (24 bits) has to be written to the Rx GPRAM in offset 0x09–0x0B. To offset 0x08, the user has to write the value "0x3F". For example, if the base address for this new structure is 0x123400, the Rx GPRAM[0x08–0x0B] is equal to 0x3F123400. **Note:** In this case there is no limitation to the number of threads that have to be enabled. 7. For other Rx VFIFO sizes please Use a microcode IRAM image. For detailed description of the Errata see http://www.freescale.com/files/32bit/doc/errata/MPC8360ECE.pdf
6–31	—	Not Valid. Cleared to zero by the microcode loader routine.

1.2 MPC8569 and P1021 (RAM Based)

1.2.1 Compilation Flags

The loader contains the following structures and compilation flags (‘defines’):

1. `MPC85xx_Rx_x_UC_PATCH or` P1021_TYPE_x_UC_PATCH. Following this define is the actual listing of the opcode of the microcode package itself. The microcode must be in one single array and contiguous. If different packages are used for different RISCs (for example, `e_QE_MULTIPLE_PATCHES`) the file will contain numerous definitions and names for each of the packages which is included.
2. `MPC85xx_Rxx_UC_PATCH_EXT_MODES or` P1021_TYPE_x_UC_PATCH_EXT_MODES. This define does not change for IRAM based devices.
3. `MPC85xx_Rxx_UC_PATCH_INFO or` P1021_TYPE_B_UC_PATCH_INFO. Brief description of the patch, includes the patch name, patch revision and chip revision.

2 Revision History

Table 3 provides a revision history for this document.

Table 3. Document Revision History

Revision	Date	Substantive Change(s)
6	Jan 2009	Added information about RAM based devices
5	Dec 2009	Added Doc ID ”QEUCODELOADER” and updated backpage info.
4	10/27/2008	Revised bit 5 in Table 2.
3	10/6/2008	Added more information for bit 5 in Table 2.
2	09/26/2008	Enlarged figure in description of bit 3 row in Table 2 Modified MPC83xx_Rxx_UC_PATCH_COMMON_IRAM bullet in Section 1, “Input for Microcode Loader.”
1	09/22/2008	Added bit 5 row in Table 2. Modified `MPC83xx_Rxx_UC_PATCH_COMMON_IRAM` bullet in Section 1, “Input for Microcode Loader.”
0	02/2008	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

QEUCODELOADER
Jan 2010
Rev. 6