Freescale Semiconductor
Application Note

AN3428
Rev. 0, 07/2007

MSC8144 QUICC Engine™ Ethernet Extended Frame Filtering using Parse Command Descriptors (PCDs)

by *Tomer Cohen*
NCSD Applications
Freescale Semiconductor, Inc.
Austin, Texas

The MSC8144[1] QUICC Engine™ block inherently supports the concept of frame filtering. This document describes how to configure the MSC8144 QUICC Engine block for extended frame filtering mode using programmable data structures called parse command descriptors (PCDs).

Contents

1 Frame Filtering Basics

When a receiver detects a frame, the UCC Ethernet controller begins filtering, which is a frame recognition function. The two filtering modes—MPC82xx backward-compatible frame filtering and extended frame filtering—are described in this document.

Extended parsing mode allows enhanced frame filtering based on fields extracted from the L2, L3, and L4 headers of a frame. The choice of the fields is user-programmable.

The core initializes the PCDs. In the PCDs, the core selects the header fields to be extracted from the frame headers and generates a LookupKey and the type of lookup table (to be used for the lookup). The core programs the PCDs during initialization.

After a frame arrives, the Ethernet controller uses the PCDs as directives to parse the frame headers, generate the

1. Other than MSC8144 QUICC Engine block, this application note is also applicable to the MSC8144E and MSC8144EC QUICC Engine blocks.

LookupKey (one or more), and perform table lookups. The LookupKey is generated by concatenating header fields.

Figure 1 shows the PCD data structures.

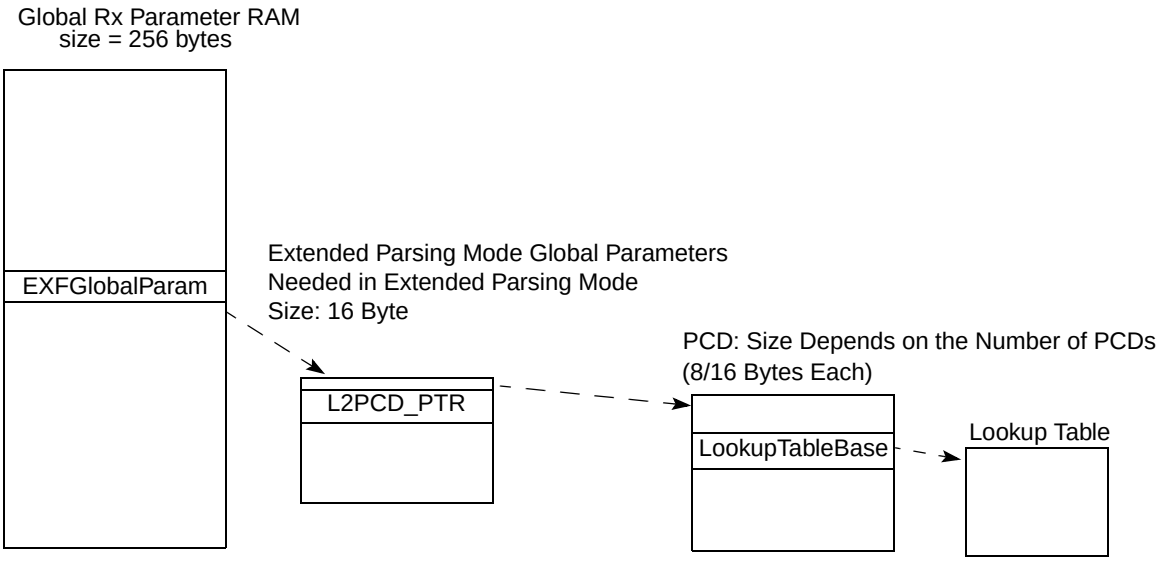

Figure 1. Receiver Parameter RAM – PCD Data Structures

In the protocol-specific mode, the QUICC Engine block parses the frames according to a predefined protocol stack. The protocol-specific mode is optimized for performance. The parser can extract the following fields (if present in the frame) in the protocol-specific mode to form a LookupKey:

- L2: MAC source address, MAC destination address, VLAN tag(s), Etype field, and so on.
- L3: IPv4 source address, IPv4 destination address, IP protocol type, and so on.
- L4: TCP/UDP source port, TCP/UDP destination port, TCP flags, and so on.

A user can program the fields to be extracted for a LookupKey up to 24 bytes. To use the extended parsing mode, initialize the following elements in the order listed:

1. Rx global parameter RAM
2. Extended parsing mode global parameters
3. PCDs
4. Hash lookup table
5. Interworking global parameters (only for L3/L4 filtering)

The following sections describe how to configure the extended parsing mode data structures. The rest of the QUICC Engine block configurations for the examples are available in the software that accompanies this application note.

2 L2 PCD Frame Filtering Configuration Example

In the example presented here, the Ethernet controller filters the frames by MAC destination address. Figure 2 shows the VTagged Ethernet encapsulation.

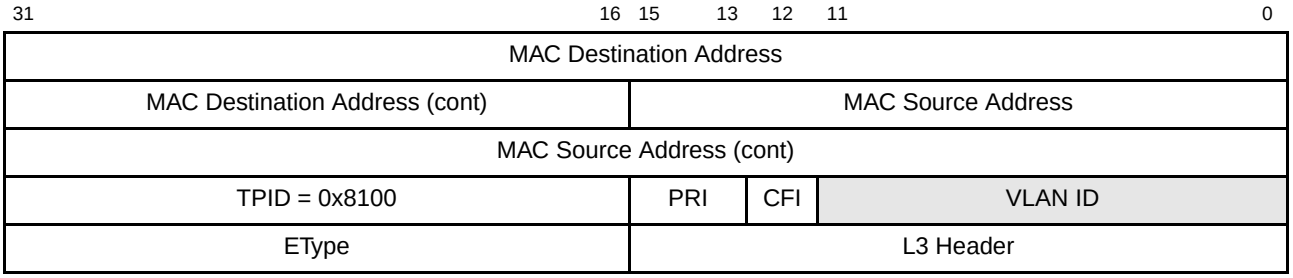

Figure 2. VTagged Ethernet Encapsulation

2.1 Rx Global Parameter RAM

To enable extended parsing mode, set the Rx global parameter, RAM REMODER[EXP] bit and initialize the EXPGlobalParam parameter in Rx global parameter RAM. The base address for the extended parsing global parameters is a 16-byte data structure. Table 1 describes the Rx global parameter RAM. For more information, see the Ethernet chapter of *MSC8144 Reference Manual*.

Table 1. Rx Global Parameter RAM

Offset	Bits	Name	Description	Configuration Example
0x00	31–0	REMODER	Receive Ethernet Mode Register. Defines the extended modes for the UCC Ethernet port.	0x80000800
0xC0		EXPGlobalParam	Base Address for Extended Parsing Global Parameters. Allocate 16 bytes for this data structure.	0x0001B500

2.2 Extended Parsing Mode Global Parameters

The extended parsing mode parameters are located at the base address programmed in the EXPGlobalParam entry in the Rx Global Parameter RAM. Table 2 describes the extended parsing global parameters.

Table 2. Extended Parsing Global Parameters Description

base_addr = 0xFEE1B500	Bits	Name	Description	Configuration Example
base_addr +0x0	7–0	Reserved	Should be cleared.	0x00
base_addr +0x0	15–8	L2PCDPTR	Pointer to the first PCD base address. The user must allocate space in the multiuser RAM for the PCDs.	0x01
base_addr +0x2	15–0			0xB600
base_addr + 0x4-7		Reserved	Should be cleared.	0x00000000
base_addr + 0x8-F		Reserved	Should be cleared.	0x00000000

2.3 Parsing Command Descriptor (PCD)

The PCD is used to parse the frame headers, generate the LookupKey (one or more), and perform table lookups. Multiple PCDs can be used on a given frame. The pointer to the first PCD is located in the multiuser RAM at the base address L2PCDPTR, which is programmed in the extended parsing mode parameter RAM data structure. Each PCD is 8–16 bytes long. PCD commands execute sequentially until a match occurs or a last PCD is encountered.

The PCD chain configured in the example contains the following PCDs:

- *GenerateLookupKey_EthFast PCD*. The first PCD in the chain, which is used to generate a LookupKey from the L2Frame header field. Subsequent PCDs use the LookupKey to perform a table lookup. In this example, a 48-bit MAC destination address is extracted from the L2 header for the LookupKey. Figure 3 shows the GenerateLookupKey_EthFast PCD configuration in the example.

base_addr = 0xFEE1_B600	15	14	13	12	11	8	7	2	1	0	**Configuration Example**
base_addr + 0	PCD OPCODE=0x00						PCDIDValue				0x0000
base_addr + 2	MAC dst	MAC src	TCI1	TCI2	Reserved				Src Port	PCDID	0x8000
base_addr+ 4	Reserved										0x0000
base_addr + 6	Reserved										0x0000

Figure 3. GenerateLookupKey_EthFast PCD

- *Four Way Hash Lookup PCD*. The second PCD in the chain, which is used to perform a lookup using the LookupKey generated by the ‘Generate LookupKey’ PCD. The example uses an 8-byte LookupKeySize and a 2-bit hash key (4 sets in the lookup table). The base address of the lookup table is 0xFEE1B000. Figure 4 shows the four-way hash lookup PCD configuration in the example.

base_addr = 0XFEE1B608	15	8	7	5	4	3	0	**Configuration Example**
base_addr + 0	PCD OPCODE = 0x20		Reserved					0x2000
base_addr + 2	LookupKeySize		Reserved		EXT	HashKeySize		0x3F01
base_addr + 4	LookupTableBase							0xFEE1
base_addr + 6								0xB000

Figure 4. Four-Way Hash Lookup PCD

- *Last PCD*. The third and last PCD in the chain, which is used at the end of PCD flow. It specifies the actions taken in case of a lookup miss. Figure 5 shows the last PCD configuration in the example.

base_addr = 0xFEE1_B610	15 14 10 9 8 7 0	Configuration Example
base_addr + 0	PCD OPCODE=0x3F / Reserved	0x3F00
base_addr + 2	DBM / Reserved / BDM / Reserved	0x0000
base_addr + 4	Reserved	0x0000
base_addr + 6	Reserved	0x0000

Figure 5. Last PCD Fields

2.4 Hash Lookup Table Initialization

The hash lookup table is initialized using the Ethernet command, Add/Remove Entry in Hash Lookup Table. This command is used to add or remove an entry in the hash lookup table. The command is issued by writing to the QUICC Engine command register (CECR) and to the QUICC Engine command data registers (CECDR). Figure 6 shows the configuration of "Set entry in Hash Lookup Table Command Parameters" in the example. For more information on these registers and parameters, see *MSC8144 Reference Manual*.

base_addr = 0XFEE1A000	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	Configuration Example
base_addr + 0	ADDE / EXLT / SLTE / HTFI / Reserved / Reserved	0x8800
base_addr + 2	LookupKeySize / Reserved / HashKeySize	0x3F01
base_addr + 4	Lookup Table Base Address	0xFEE1
base_addr + 6	Lookup Table Base Address	0xB000
base_addr+8	Secondary Lookup Table Base Address	0x0000
base_addr+A	Secondary Lookup Table Base Address	0x0000
base_addr+C	Reserved	0x0000
base_addr+E	Reserved	0x0000
base_addr+10 (TAD)	EXF / V / Rej / IWE = 0 / Res / Res / VTagOP / VNonVTagOP / Res / RQoS	0x4000
base_addr+12 (TAD)	VPriority / CFI / VID	0x0000
base_addr+14 (TAD)	Reserved / Reserved	0x0000
base_addr+16 (TAD)	Reserved	0x0000
base_addr+18	Reserved	0x0000
base_addr+1A	Reserved	0x0000
base_addr+1C	Reserved	0x0000

Figure 6. Set entry in Hash Lookup Table Command Parameters

base_addr = 0XFEE1A000	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	Configuration Example
base_addr+1E	Reserved	0x0000
base_addr+20	LookupKey (up to 16 bytes)	0x0000
base_addr+22		0x0000
base_addr+24		0x6365
base_addr+26		0x0000
base_addr+28		0x0000
base_addr+2A		0x0000
base_addr+2C		0x0000
base_addr+2E		0x0000
base_addr + 30-3F	Reserved	0x0000
base_addr + 0x40-0xBF or 0x40-0x9F or 0x40-0x7F	Reserved (size depends on LookupKeySize: 24, 16, or 8 bytes respectively)	0x0000

Figure 6. Set entry in Hash Lookup Table Command Parameters (continued)

In this example, the base address is 0xFEE1A000, the lookup table base address is 0xFEE1B000, and the LookupKey is 0x0000000063650000. Therefore, the Ethernet controller accepts frames with a destination MAC address of 0x000000006365.

After initializing the data structure as shown in Figure 6, the user must issue the command, Add/Remove Entry in Hash Lookup Table, to build the hash table. CECDR contains the base address in multiuser RAM where the data structure in Figure 6 resides. Therefore, to issue this command, write the values to the registers, as specified in Table 3.

Table 3. Issuing Command Add/Remove Entry in Hash Lookup Table

Register	Access Type	Value	Description
CECDR	Write	0xFEE1A000	Pointer to “Set entry in Hash Lookup Table Command Parameters”
CECR	Write	0x03C10013	Issue the command, Add/Remove Entry in Hash Lookup Table

After issuing the command, wait until the CECR[FLG] = 0. At this stage, another entry can be added to the hash table. For example, the second entry is set when LookupKey is 0x0000000063690000 so that the Ethernet controller accepts frames with a destination MAC address of 0x000000006365 or 0x000000006369. To add another entry, change the value of LookupKey in Figure 6 to 0x0000000063690000 and then issue the command, Add/Remove Entry in Hash Lookup Table, as specified in Table 3.

2.5 Software Support

This section lists the C code to configure the extended parsing mode data structures. The complete QUICC Engine configuration code for this example can be found in the software contained in AN3428SW.zip.

```
typedef unsigned long           uint32; /* Unsigned 32-bit integer */
#define READ_UINT32(arg, data)      data = (uint32)(arg)
#define WRITE_UINT32(arg, data)     arg = (uint32)(data)
#define CE_PRAM                             0xfee10000

void ce_ucc3_L2_PCD_init ()
{
        uint32 i;

                        // UCC3 Rx Global Parameter RAM. Base address of Rx Global Parameter
                         //RAM is “CE_PRAM + 0x5700”
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0x5700),0x80000800);  // REMODER.
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0x57c0),0x0001b500); // EXPGlobalParam

        //Extended Parsing Global Parameters table
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb500),0x0001b600); //Pointer to the First PCD
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb504),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb508),0x00000000);

        //Parsing Command Descriptor (PCD)
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb600),0x00008000); //GenerateLookupKey_EthFast
                                                                  //PCD, Filtering by MACdst
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb604),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb608),0x20003f01); //Four Way Hash Lookup
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb60c),CE_PRAM + 0xb000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb610),0x3f000000); //Last PCD
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb614),0x00000000);

        //Set entry in Hash Lookup Table Command Parameters
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa000),0x88003f01);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa004),CE_PRAM + 0xb000); //LookupTableBaseAddr
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa008),0x00000000); //Secondary LookupTableBase
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa00c),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa010),0x40000000); //V=1
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa014),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa018),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa01c),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa020),0x00000000); //LookupKey =
                                                                  // 0x0000000063650000
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa024),0x63650000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa028),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa02c),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa030),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa034),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa038),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa03c),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa040),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa044),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa048),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa04c),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa050),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa054),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa058),0x00000000);
```

```
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa05c),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa060),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa064),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa068),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa06c),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa070),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa074),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa078),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa07c),0x00000000);

        //clear LookupTable area
        for (i = 0; i < 128; i++)
        {
                WRITE_UINT32(*(uint32 *)((CE_PRAM + 0xb000) + i * 4), 0x00000000);
        }
        // Command register - Build the Hash table
        WRITE_UINT32(*(uint32 *)(CECDR), CE_PRAM + 0xa000);
        WRITE_UINT32(*(uint32 *)(CECR), 0x03c10013);
        wait_for_reg_negate (CECR,0x00010000); //wait until CECR[FLG] = 0

        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa024), 0x63690000); //LookupKey =
                                                                 //0x0000000063690000
        // Command register - Build the Hash table
        WRITE_UINT32(*(uint32 *)(CECDR), CE_PRAM + 0xa000);
        WRITE_UINT32(*(uint32 *)(CECR), 0x03c10013);
        wait_for_reg_negate (CECR,0x00010000); //wait until CECR[FLG] = 0
} //end of func ce_ucc3_L2_PCD_init

void wait_for_reg_pos (addr,data_check)
{

     uint32 data = 0x0;
          READ_UINT32(*(uint32 *)addr,data);
while ((data & data_check) != data_check)
                         READ_UINT32(*(uint32 *)addr,data);

}
```

3 L3 PCD Frame Filtering Configuration Example

In the example presented here, the Ethernet controller filters the frames by IP destination address. Figure 7 shows the IPv4 (RFC791) header.

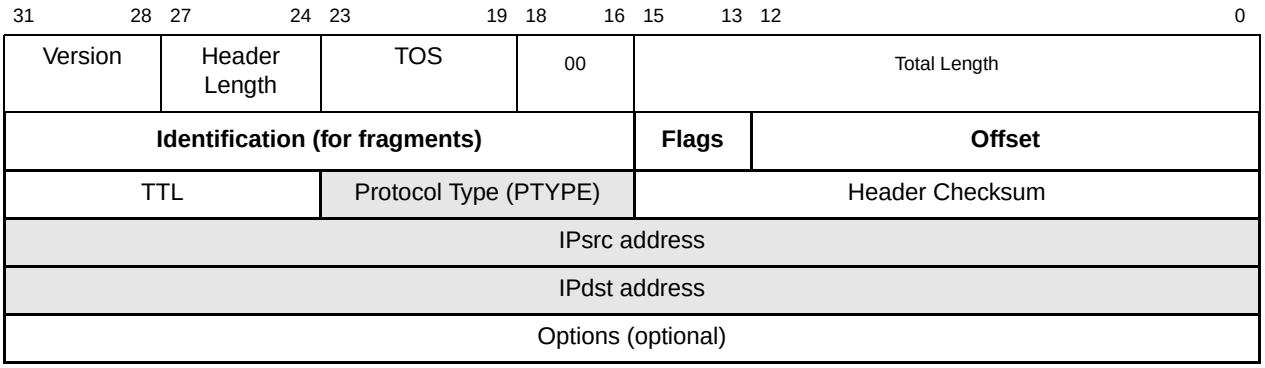

Figure 7. IPv4 (RFC791)

3.1 Rx Global Parameter RAM

In addition to the initialization instructions provided in Section 2.1, “Rx Global Parameter RAM,” the parameters—REMODER[IWEn] = 1, IWGlobalParam_Base, and IWThreadsParam_Base—must be initialized in Rx Global Parameter RAM. Table 4 describes the Rx global parameter RAM. For more information, see *MSC8144 Reference Manual*.

Table 4. Rx Global Parameter RAM

Offset	Bits	Name	Description	Configuration Example
0x00	31–0	REMODER	Receive Ethernet Mode Register. Defines the extended modes for the UCC Ethernet port.	0x90000800
0xC0		EXPGlobalParam	Base Address for Extended Parsing Global Parameters. Allocate 16 bytes for this data structure.	0x0001B500
0xF0		IWGlobalParam_ Base	Base Address for interworking global parameters. Allocate 256 bytes for this data structure if the Ethernet controller is used in interworking mode.	0x0001B700
0xF4		IWThreadsParam _Base	Base Address for interworking thread parameters. Allocate N × 576 or N × 672 bytes for IW temporary parameters if the Ethernet controller is used in interworking mode. N equals the number of Rx threads as initialized by the user in the INIT RX Host Command. N × 576 bytes are allocated if it is guaranteed that all incoming frames are shorter than MRBLR or if (MAXD1 and MAXD2) ≤ MRBLR (that is, 1BD per frame). The pointer should be aligned to 512 bytes. N × 672 bytes are allocated if the incoming frames are longer than MRBLR and if (MAXD1 and MAXD2) > MRBLR.	0x00011000

3.2 Extended Parsing Mode Global Parameters

For information on extended parsing mode global parameters, see Section 2.2, “Extended Parsing Mode Global Parameters.”

3.3 Parsing Command Descriptor (PCD)

The L3 PCD filtering is same as L2 PCD filtering discussed in Section 2.3, “Parsing Command Descriptor (PCD).” The only difference is the first PCD in the chain, GenerateLookupKey_Eth PCD.

- *GenerateLookupKey_Eth PCD*. The PCD is used to generate a LookupKey from the L2, L3, and L4 frame header fields on the Ethernet receive port. Subsequent PCDs use the LookupKey to perform a table Lookup. The bits in offset + 8 of the PCD describe the expected protocol stack in the incoming frame. These bits are used to determine ParseHit or ParseMiss on the frame. For each set bit, the parser searches the corresponding header in the frame. If the frame is not found, a ParseMiss occurs. The PCD contains a pointer to the next PCD used if a ParseMiss occurs.

  In this example, a 32-bit IP destination address for LookupKey is extracted from the L3 header. The pointer to the next PCD is used if a ParseMiss occurs, unless it is programmed to point to Last PCD. Figure 8 shows the GenerateLookupKey_EthFast PCD configuration in the example.

base_addr = 0XFEE1B600	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Configuration Example
base_addr + 0	PCD OPCODE=0x01								PCDIDValue								0x0100
base_addr + 2	TCI1NE	TCI2NE	TCI3NE	TCI4NE	TCI5NE	TCI6NE	0	0	Reserved								0x0000
base_addr+ 4	UIE	Reserved	FF rg	TTL En	Reserved				MissPCDPTR								0x0001
base_addr + 6	MissPCDPTR																0xB618
base_addr + 8	TCI1	TCI2	TCI3	TCI4	TCI5	TCI6	no IPOpt	no IPFrg	PPPoE Session	IP	TCP	UDP	SCTP	Reserved			0x0000
base_addr+ A	Reserved						no ICMP	no IGMP	no ARP	no DHCP	no TCPCtl	Reserved					0x0000
base_addr + C	MAC dst (6)	MAC src (6)	TCI1 (2)	TCI2 (2)	EType (2)	PPP SID (2)	PPP PID (2)q	IPsrc (4)	IPdst (4)	PTYPE (1)	IP TOS (1)	TUSP src (2)	TUSP dst (2)	TFlg (1)	Source Port (1)	PCDID (1)	0x0080
base_addr + E	Reserved																0x0000

Figure 8. Generate LookupKey_Eth PCD

- *Four-Way Hash Lookup PCD*. For information on this PCD, see Section 2.3, "Parsing Command Descriptor (PCD)." Figure 9 shows the four-way hash lookup PCD configuration in the example.

base_addr = 0XFEE1B610	15	8	7	5	4	3	0	Configuration Example
base_addr + 0	PCD OPCODE=0x20		Reserved					0x2000
base_addr + 2	LookupKeySize		Reserved		EXT	HaskKeySize		0x3F01
base_addr + 4	LookupTableBase							0xFEE1
base_addr + 6								0xB000

Figure 9. Four Way Hash Lookup PCD

- *Last PCD*. For information on this PCD, see Section 2.3, "Parsing Command Descriptor (PCD)." Figure 10 shows the last PCD configuration in the example.

base_addr = 0XFEE1B618	15	14	10	9	8	7	0	Configuration Example
base_addr + 0	PCD OPCODE=0x3F					Reserved		0x3F00
base_addr + 2	DBM	Reserved		BDM		Reserved		0x0000
base_addr + 4	Reserved							0x0000
base_addr + 6								0x0000

Figure 10. Last PCD Fields

3.4 Hash Lookup Table Initialization

Only the value of the LookupKey field in “Set entry in Hash Lookup Table Command Parameters” is different from the initialization instructions presented in Section 2.4, “Hash Lookup Table Initialization.” In this example, the LookupKey is 0xDF895F0900000000. Therefore, the Ethernet controller accepts frames with an IP destination address of 0xDF895F09 (223.137.95.9).

3.5 Interworking Global Parameters Initialization

The interworking global parameters are located at the base address programmed in the IWGlobalParam_Base entry in the Rx global parameter RAM. The field IW_EthLenType must be initialized in the “Interworking Global Parameters” table. Table 5 describes the interworking global parameters.

Table 5. Interworking Global Parameters

base_addr = 0XFEE1B700	Size (Bits)	Name	Description	Configuration Example
base_addr + 0x00-0x4C		—	Bits should be cleared.	0x0000
base_addr + 0x4E	16	IW_EthLenType	Value compared to Ethernet Length/Type field to distinguish Ethernet and IEEE 802.3 type of frames. Most common value is 0x0600.	0x0600
base_addr + 0x50-0x7B		—	Bits should be cleared.	0x0000

This example does not use IW mode. Therefore, the IW global parameters data structure is initialized to zero (except for the field IW_EthLenType, which is used).

3.6 Software Support

This section lists the C code to configure the extended parsing mode data structures. The complete QUICC Engine configuration code for this example can be found in the software contained in AN3428SW.zip.

```
typedef unsigned long           uint32; /* Unsigned 32-bit integer */
#define READ_UINT32(arg, data)      data = (uint32)(arg)
```

```
#define WRITE_UINT32(arg, data)     arg = (uint32)(data)
#define CE_PRAM                              0xfee10000

void ce_ucc3_L3_PCD_init ()

{

        uint32 i;
        // UCC3 Rx Global Parameter RAM
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0x5700),0x90000800);  // REMODER.
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0x57c0),0x0001b500);  // EXPGlobalParam
         WRITE_UINT32(*(uint32 *)(CE_PRAM + 0x57f0),0x0001b700);  // IWGlobalParam_Base
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0x57f4),0x00011000);  // IWThreadsParam_Base

        //Extended Parsing Global Parameters table
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb500),0x0001b600);  //Pointer to the First PCD
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb504),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb508),0x00000000);

        //Parsing Command Descriptor (PCD)
        WRITE_UINT32(*(uint32 *)0xfee1b600,0x01000000); //GenerateLookupKey_Eth PCD ,
                                                        //Filtering by IP dst addr
        WRITE_UINT32(*(uint32 *)0xfee1b604,0x0001b618);
        WRITE_UINT32(*(uint32 *)0xfee1b608,0x00000000);
        WRITE_UINT32(*(uint32 *)0xfee1b60c,0x00800000);

        WRITE_UINT32(*(uint32 *)0xfee1b610,0x20003f01); //Four Way Hash Lookup
        WRITE_UINT32(*(uint32 *)0xfee1b614,0x0001b000);

        WRITE_UINT32(*(uint32 *)0xfee1b618,0x3f000000); //Last PCD
        WRITE_UINT32(*(uint32 *)0xfee1b61c,0x00000000);

        //Set entry in Hash Lookup Table Command Parameters
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa000),0x88003f01);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa004),CE_PRAM + 0xb000); // LookupTableBase
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa008),0x00000000); // Secondary LookupTableBase
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa00c),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa010),0x40000000); // V=1
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa014),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa018),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa01c),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa020),0xdf895f09);  //LookupKey =
                                                                 //0xdf895f0900000000
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa024),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa028),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa02c),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa030),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa034),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa038),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa03c),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa040),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa044),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa048),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa04c),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa050),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa054),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa058),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa05c),0x00000000);

        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa060),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa064),0x00000000);
```

```
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa068),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa06c),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa070),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa074),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa078),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa07c),0x00000000);

        //clear LookupTable area
        for (i = 0 ; i < 128 ; i++)

       {
                WRITE_UINT32(*(uint32 *)((CE_PRAM + 0xb000) + i * 4), 0x00000000);
        }

        // Command register - Build the Hash table
        WRITE_UINT32(*(uint32 *)(CECDR), CE_PRAM + 0xa000);
        WRITE_UINT32(*(uint32 *)(CECR), 0x03c10013);
        wait_for_reg_negate (CECR ,0x00010000);

        //INIT IWGlobalParam
             for (i=0x0;i < 0x100;i= i + 0x4)
        {
                WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb700 + i),0x0);
        }
    WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb700 + 0x4c),0x00000600); // IW_EthLenType
} //end of func ce_ucc3_L3_PCD_init
void wait_for_reg_pos (addr,data_check) {
     uint32 data = 0x0;
          READ_UINT32(*(uint32 *)addr,data);
     while ((data & data_check) != data_check)
                          READ_UINT32(*(uint32 *)addr,data);
}
```

4 L4 PCD Frame Filtering Configuration Example

In the example presented here, the Ethernet controller filters the frames by UDP destination port. Figure 11 shows the UDP (RFC768) header.

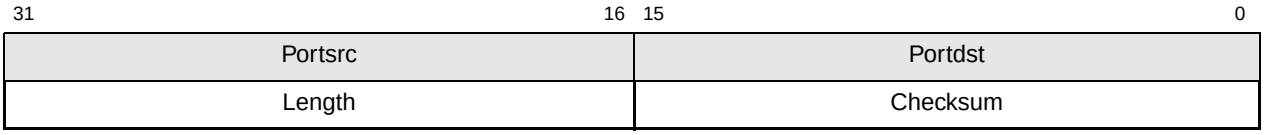

Figure 11. UDP (RFC768)

4.1 Rx Global Parameter RAM

For information on Rx global parameter RAM, see Section 3.1, “Rx Global Parameter RAM.”

4.2 Extended Parsing Mode Global Parameters

For information on extended parsing mode global parameters, see Section 2.2, “Extended Parsing Mode Global Parameters.”

4.3 Parsing Command Descriptor (PCD)

The L4 PCD filtering is same as L3 PCD filtering discussed in Section 3.3, “Parsing Command Descriptor (PCD).” The only difference is the first PCD in the chain, GenerateLookupKey_Eth PCD. Instead of extracting the 32-bit IP destination address, this PCD extracts the 16-bit UDP destination port. Figure 12 shows the GenerateLookupKey_EthFast PCD configuration in the example.

base_addr = 0XFEE1B600	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Configuration Example
base_addr + 0	PCD OPCODE=0x01								PCDIDValue								0x0100
base_addr + 2	TCI1NE	TCI2NE	TCI3NE	TCI4NE	TCI5NE	TCI6NE	0	0	Reserved								0x0000
base_addr+ 4	UIE	Reserved	FF rg	TTL En	Reserved				MissPCDPTR								0x0001
base_addr + 6	MissPCDPTR																0xB618
base_addr + 8	TCI1	TCI2	TCI3	TCI4	TCI5	TCI6	no IP Opt	no IP Frg	PPPoE Session	IP	TCP	UDP	SCTP	Reserved			0x0000
base_addr+ A	Reserved						no ICMP	no IGMP	no ARP	no DHCP	no TCPCtl	Reserved					0x0000
base_addr + C	MAC dst (6)	MAC src (6)	TCI1 (2)	TCI2 (2)	EType (2)	PPP SID (2)	PPP PID (2)q	IPsrc (4)	IPdst (4)	PTYPE (1)	IP TOS (1)	TUSP src (2)	TUSP dst (2)	TFlg (1)	Source Port (1)	PCDID (1)	0x0008
base_addr + E	Reserved																0x0000

Figure 12. Generate LookupKey_Eth PCD

Figure 13 shows the four-way hash lookup PCD configuration in the example.

base_addr = 0XFEE1B610	15 … 8	7 … 5	4	3 … 0	Configuration Example
base_addr + 0	PCD OPCODE=0x20	Reserved			0x2000
base_addr + 2	LookupKeySize	Reserved	EXT	HaskKeySize	0x3F01
base_addr + 4	LookupTableBase				0xFEE1
base_addr + 6					0xB000

Figure 13. Four Way Hash Lookup PCD

Figure 14 shows the last PCD configuration in the example.

base_addr = 0XFEE1B618	15	14 … 10	9	8	7 … 0	Configuration Example
base_addr + 0	PCD OPCODE = 0x3F				Reserved	0x3F00
base_addr + 2	DBM	Reserved		BDM	Reserved	0x0000
base_addr + 4	Reserved					0x0000
base_addr + 6						0x0000

Figure 14. Last PCD Field Descriptions

4.4 Hash Lookup Table Initialization

Only the value of the LookupKey field in “Set entry in Hash Lookup Table Command Parameters” is different from the initialization instructions in Section 2.4, “Hash Lookup Table Initialization.” In this example, the LookupKey is 0x1111000000000000. Therefore, the Ethernet controller accepts frames with a UDP destination port of 0x1111.

4.5 Interworking Global Parameters Initialization

For initialization instructions, see Section 3.5, “Interworking Global Parameters Initialization.”

4.6 Software support

This section lists the C code to configure the extended parsing mode data structures. The complete QUICC Engine configuration code for this example can be found in the software contained in AN3428SW.zip..

```
typedef unsigned long           uint32; /* Unsigned 32-bit integer */
#define READ_UINT32(arg, data)      data = (uint32)(arg)
#define WRITE_UINT32(arg, data)     arg = (uint32)(data)
#define CE_PRAM                             0xfee10000

void ce_ucc3_L4_PCD_init ()
{
        uint32 i;
```

```
// UCC3 Rx Global Parameter RAM
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0x5700),0x90000800);  // REMODER.
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0x57c0),0x0001b500);  // EXPGlobalParam
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0x57f0),0x0001b700);  // IWGlobalParam_Base
 WRITE_UINT32(*(uint32 *)(CE_PRAM + 0x57f4),0x00011000);  // IWThreadsParam_Base

 //Extended Parsing Global Parameters table
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb500),0x0001b600);  //Pointer to the First PCD
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb504),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb508),0x00000000);

//Parsing Command Descriptor (PCD)
WRITE_UINT32(*(uint32 *)0xfee1b600,0x01000000); //GenerateLookupKey_Eth PCD ,
                                                //Filtering by UDP dest port
WRITE_UINT32(*(uint32 *)0xfee1b604,0x0001b618);
WRITE_UINT32(*(uint32 *)0xfee1b608,0x00000000);
WRITE_UINT32(*(uint32 *)0xfee1b60c,0x00080000);

WRITE_UINT32(*(uint32 *)0xfee1b610,0x20003f01); //FourWayHashLookup
WRITE_UINT32(*(uint32 *)0xfee1b614,0x0001b000);

WRITE_UINT32(*(uint32 *)0xfee1b618,0x3f000000); //Last PCD
WRITE_UINT32(*(uint32 *)0xfee1b61c,0x00000000);

//Set entry in Hash Lookup Table Command Parameters
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa000),0x88003f01);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa004),CE_PRAM + 0xb000); // LookupTableBase
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa008),0x00000000); // Secondary LookupTableBase
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa00c),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa010),0x40000000); // V=1
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa014),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa018),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa01c),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa020),0x11110000);  //LookupKey =
                                                         //0x1111000000000000
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa024),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa028),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa02c),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa030),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa034),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa038),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa03c),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa040),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa044),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa048),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa04c),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa050),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa054),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa058),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa05c),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa060),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa064),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa068),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa06c),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa070),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa074),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa078),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa07c),0x00000000);
```

```
        //clear LookupTable area
        for (i = 0 ; i < 128 ; i++) {
                WRITE_UINT32(*(uint32 *)((CE_PRAM + 0xb000) + i * 4), 0x00000000);
        }

        // Command register - Build the Hash table
        WRITE_UINT32(*(uint32 *)(CECDR), CE_PRAM + 0xa000);
        WRITE_UINT32(*(uint32 *)(CECR), 0x03c10013);
        wait_for_reg_negate (CECR ,0x00010000);

        //INIT IWGlobalParam
              for (i=0x0;i < 0x100;i= i + 0x4)
                  {
                  WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb700 + i),0x0);
                   }

        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb700 + 0x4c),0x00000600); // IW_EthLenType
} //end of func ce_ucc3_L4_PCD_init

void wait_for_reg_pos (addr,data_check) {
        uint32 data = 0x0;
                READ_UINT32(*(uint32 *)addr,data);
        while ((data & data_check) != data_check)
                        READ_UINT32(*(uint32 *)addr,data);

}
```

5 L4 PCD Frame Filtering: 4-Channel Configuration Example

In the example presented here, the configuration filters the UDP packets to four receive channels, one channel for each core. Incoming UDP packets are filtered on the basis of UDP destination port, and the QUICC Engine block forwards them into the appropriate receive channel. Table 6 describes the mapping between UDP destination port and channel number (core number).

Table 6. Mapping between UDP Destination Port and Channel Number

UDP Destination Port	Channel Number (core Number)
0x1110	0
0x1111	1
0x1112	2
0x1113	3

5.1 Rx Global Parameter RAM

For information on Rx global parameter RAM, see Section 3.1, “Rx Global Parameter RAM.”

5.2 Extended Parsing Mode Global Parameters

For information on extended parsing mode global parameters, see Section 2.2, “Extended Parsing Mode Global Parameters.”

5.3 Parsing Command Descriptor (PCD)

For information on PCD, see Section 4.3, “Parsing Command Descriptor (PCD).”

5.4 Hash Lookup Table Initialization

To filter and forward incoming UDP packets on the basis of UDP destination port to four receive channels, at least four entries must be set in the hash table. Each entry has a different LookupKey and a different termination action descriptor (TAD).

To set four entries in the table, issue the command, Add/Remove Entry in Hash Lookup Table, four times with a different LookupKey and a different TAD[VPriority] each time. The value in TAD[VPriority] determines the Rx channel of the incoming frames.

Table 7 lists the configuration values of the LookupKey and TAD[VPriority] fields in “Set entry in Hash Lookup Table Command Parameters” for each of the four entries in the lookup table.

Table 7. Configuration Value of the Fields “LookupKey” and “TAD[VPriority]”

Entry Number	UDP Dest Port	Channel Number (Core Number)	LookupKey	TAD[VPriority]
1	0x1110	0	0x1110000000000000	0
2	0x1111	1	0x1111000000000000	1
3	0x1112	2	0x1112000000000000	2
4	0x1113	3	0x1113000000000000	3

In this example, the “Set entry in Hash Lookup Table Command Parameters” base address is 0xFEE1A000 and the lookup table base address is 0xFEE1B000. Figure 15 shows the configuration of “Set entry in Hash Lookup Table Command Parameters” in this example.

base_addr = 0XFEE1A000	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Configuration Example
base_addr + 0	ADDE		EXLT	SLTE		HTFI	Reserved		Reserved								0x8800
base_addr + 2	LookupKeySize								Reserved				HashKeySize				0x3f01
base_addr + 4	Lookup Table Base Address																0xFEE1
base_addr + 6																	0xB000
base_addr+8	Secondary Lookup Table Base Address																0x0000
base_addr+A																	0x0000
base_addr+C	Reserved																0x0000
base_addr+E	Reserved																0x0000

Figure 15. Set entry in Hash Lookup Table Command Parameters

base_addr = 0XFEE1A000	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Configuration Example
base_addr+10 (TAD)	EXF	V	Rej	IWE=0	Res\	Res	VTagOP				VNonVTagOP	Res			RQoS		0x4000
base_addr+12 (TAD)	VPriority			CFI	VID												0x0000 OR 0x2000 OR 0x4000 OR 0x6000
base_addr+14 (TAD)	Reserved								Reserved								0x0000
base_addr+16 (TAD)	Reserved																0x0000
base_addr+18	Reserved																0x0000
base_addr+1A	Reserved																0x0000
base_addr+1C	Reserved																0x0000
base_addr+1E	Reserved																0x0000
base_addr+20	LookupKey (up to 16 bytes)																0x1110 OR 0x1111 OR 0x1112 OR 0x1113
base_addr+22																	0x0000
base_addr+24																	0x0000
base_addr+26																	0x0000
base_addr+28																	0x0000
base_addr+2A																	0x0000
base_addr+2C																	0x0000
base_addr+2E																	0x0000
base_addr 30-3F	Reserved																0x0000
base_addr + 0x40-0xBF or 0x40-0x9F or 0x40-0x7F	Reserved (size depends on LookupKeySize: 24, 16, or 8 bytes respectively)																0x0000

Figure 15. Set entry in Hash Lookup Table Command Parameters (continued)

After initialization of each data structure in Figure 15, issue the command “Add/Remove Entry in Hash Lookup Table” (as described in Table 3) to add an entry to the hash table.

5.5 Interworking Global Parameters Initialization

For initialization instructions, see Section 3.5, “Interworking Global Parameters Initialization.”

5.6 Software Support

This section lists the C code to configure the extended parsing mode data structures. The complete QUICC Engine configuration code for this example can be found in the software contained in AN3428SW.zip.

```
typedef unsigned long           uint32; /* Unsigned 32-bit integer */
#define READ_UINT32(arg, data)      data = (uint32)(arg)
#define WRITE_UINT32(arg, data)     arg = (uint32)(data)
#define CE_PRAM                             0xfee10000

void ce_ucc3_4channels_L4_PCD_init ()
{

        uint32 i;

        // UCC3 Rx Global Parameter RAM
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0x5700),0x90000800);  // REMODER.
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0x57c0),0x0001b500);  // EXPGlobalParam
         WRITE_UINT32(*(uint32 *)(CE_PRAM + 0x57f0),0x0001b700);  // IWGlobalParam_Base
         WRITE_UINT32(*(uint32 *)(CE_PRAM + 0x57f4),0x00011000);  // IWThreadsParam_Base

         //Extended Parsing Global Parameters table
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb500),0x0001b600);  //Pointer to the First PCD
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb504),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb508),0x00000000);

        //Parsing Command Descriptor (PCD)
        WRITE_UINT32(*(uint32 *)0xfee1b600,0x01000000); //GenerateLookupKey_Eth PCD ,
                                                        //Filtering by UDP dest port
        WRITE_UINT32(*(uint32 *)0xfee1b604,0x0001b618);
        WRITE_UINT32(*(uint32 *)0xfee1b608,0x00000000);
        WRITE_UINT32(*(uint32 *)0xfee1b60c,0x00080000);

        WRITE_UINT32(*(uint32 *)0xfee1b610,0x20003f01); //FourWayHashLookup
        WRITE_UINT32(*(uint32 *)0xfee1b614,0x0001b000);

        WRITE_UINT32(*(uint32 *)0xfee1b618,0x3f000000); //Last PCD
        WRITE_UINT32(*(uint32 *)0xfee1b61c,0x00000000);

        //Set entry in Hash Lookup Table Command Parameters
         WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa000),0x80003f01);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa004),CE_PRAM + 0xb000); // LookupTableBase
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa008),0x00000000); // Secondary LookupTableBase
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa00c),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa010),0x40000000); // V=1 , VPRi = i;
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa014),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa018),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa01c),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa020),0x11100000);  //LookupKey =
                                                                 //0x111i000000000000
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa024),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa028),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa02c),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa030),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa034),0x00000000);
```

```
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa038),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa03c),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa040),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa044),0x00000000);

        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa048),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa04c),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa050),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa054),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa058),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa05c),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa060),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa064),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa068),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa06c),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa070),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa074),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa078),0x00000000);
        WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa07c),0x00000000);
        //clear LookupTable area

        for (i = 0 ; i < 128 ; i++) {
                WRITE_UINT32(*(uint32 *)((CE_PRAM + 0xb000) + i * 4), 0x00000000);
        }

               for (i = 0 ; i < 4 ; i++)
         {

                WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa020),0x11100000 | i<<16 ); //LookupKey
                                                                  //= 0x111i000000000000
                WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa010),0x40000000 | i<<13 ); //V=1 , VPRi
                                                                        //= i;

                // Command register - Build the Hash table
                WRITE_UINT32(*(uint32 *)(CECDR), CE_PRAM + 0xa000);
                WRITE_UINT32(*(uint32 *)(CECR), 0x03c10013);
                wait_for_reg_negate (CECR ,0x00010000);
      }

        //INIT IWGlobalParam
             for (i=0x0;i < 0x100;i= i + 0x4)

                 {

                WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb700 + i),0x0);

                 }

       WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb700 + 0x4c),0x00000600); // IW_EthLenType

}//end of func ce_ucc3_4channels_L4_PCD_init

void wait_for_reg_pos (addr,data_check) {
uint32 data = 0x0;
         READ_UINT32(*(uint32 *)addr,data);
                while ((data & data_check) != data_check)
                         READ_UINT32(*(uint32 *)addr,data);

}
```

THIS PAGE INTENTIONALLY LEFT BLANK

THIS PAGE INTENTIONALLY LEFT BLANK

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 Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor
Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
+1-800 441-2447 or
+1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com

Document Number: AN3428
Rev. 0
07/2007

freescale™
semiconductor