Freescale Semiconductor
Application Note

AN3880
Rev. 0, 10/2009

Ethernet Extended Frame Filtering (PCD)

by *Networking and Multimedia Group*
Freescale Semiconductor, Austin, TX

The QE inherently supports the concept of frame filtering. This document describes how to configure the QE for extended frame filtering mode using programmable parse command descriptor (PCD).

Contents

1 Introduction

When the receiver detects a frame, the UCC Ethernet controller begins to perform the frame recognition function (that is, filtering). Two modes are supported: MPC82xx backward compatible frame filtering, and extended frame filtering mode.

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

The core initializes data structures called parse command descriptors (PCDs). In the PCDs, the core selects which header fields are to be extracted from the frame headers to generate a LookupKey and the type of LookupTable to be used for the lookup. The PCDs are programmed by the core at initialization.

Upon arrival of a frame, the Ethernet controller uses the PCDs as directives to parse the frame headers, generate a LookupKey (one or more), and perform table lookups. The LookupKey is generated by concatenating header fields.

freescale semiconductor

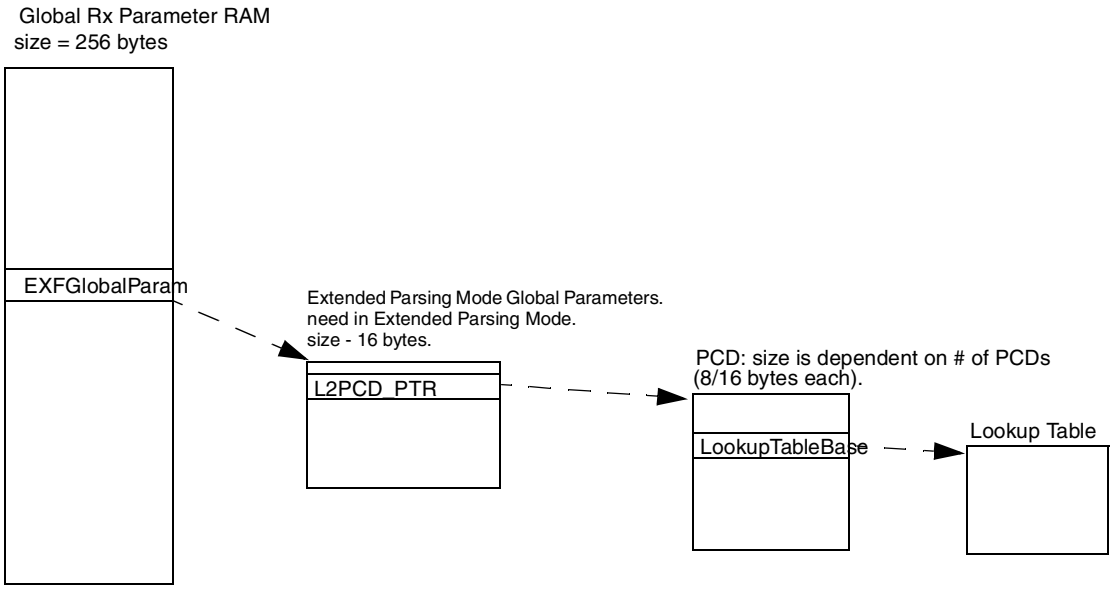

Figure 1. Receiver Parameter RAM—PCD Data Structures

In the protocol-specific mode, the QUICC Engine parses the frames according to predefined protocol stack.

If present in the frame, the following fields can be extracted by the parser in the protocol specific mode to form a LookupKey. The user may program which fields are actually extracted for a LookupKey of up to 24 bytes:

- L2: MAC source address, MAC destination address, VLAN Tag(s), Etype field, etc.
- L3: IPv4 source address, IPv4 destination address, IP Protocol Type, etc.
- L4: TCP/UDP source port, TCP/UDP destination port, TCP Flags. etc.

To use extended parsing mode, the user must perform the following configuration steps:

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

NOTE—Important:

The following sections describe how to configure the extended parsing mode data structures only. The rest of the QE configurations of these examples can be found in the attached software to this application note.

2 L2 PCDs Frame Filtering Configuration Example

This section describes configuration example for L2 PCD filtering. In this example, The Ethernet controller performs filtering by MAC destination address. Table 1 describes VTagged Ethernet encapsulation.

Table 1. VTagged Ethernet Encapsulation

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MAC Destination Address																															
MAC Destination Address (cont)																MAC Source Address															
MAC Source Address (cont)																															
TPID = 0x8100																VPRI			C FI	VLAN ID											
EType																L3 Header															

2.1 Rx Global Parameter RAM

The extended parsing mode is enabled by the user by programming in Rx global parameter RAM REMODER[EXP] = 1.

The user must initialize EXPGlobalParam in Rx global parameter RAM—the base address for extended parsing global parameters, which is a 16-byte data structure.

Table 2. Rx Global Parameter RAM

Offset	Bits	Name	Description	Configuration Example
0x00	0–31	REMODER	Ethernet Mode Register.	0x80000800
0xC0	—	EXPGlobalParam	Base Address for Extended Parsing Global Parameters. The user needs to allocate 16 bytes for this data structure.	0x0000b500

2.2 Extended Parsing Mode Global Parameters

The extended parsing mode parameters, shown in Table 3, are located at base address programmed in EXPGlobalParam entry in the Rx global parameter RAM.

Table 3. Extended Parsing Global Parameters Description

base_addr = 0xfee1b500	Bits	Name	Description	Configuration Example
base_addr +0x0	0–7	Reserved	Set to zero	All zeros
base_addr +0x0	8–15	L2PCDPTR	Pointer to first parse command descriptor base address. The user is responsible for allocation of space in the multi-user RAM for the PCDs.	0x01
base_addr +0x2	0–15			0xb600
base_addr + 0x4-7	—	Reserved	Set to zero	All zeros
base_addr + 0x8-F	—	Reserved	Set to zero	All zeros

2.3 Parsing Command Descriptor (PCD)

The parsing command descriptor (PCD) is a data structure programmed by the user that is used for parsing of the frame, generation of LookupKey, and lookup in lookup tables. Multiple PCDs may be used on a given frame. The pointer to the first PCD is located in the multi-user RAM at base address L2PCDPTR, which is programmed in the extended parsing mode parameter RAM data structure. Each PCD is 8/16 bytes long. PCD commands are executed sequentially until a match occurs or a last PCD is encountered.

The PCD chain that is configured in this example contains the following PCDs:

- **GenerateLookupKey_EthFast PCD**—this is the first PCD in the chain. This type of PCD is used to generate a LookupKey from the L2Frame header fields. The LookupKey is used by subsequent PCDs to perform a table lookup. In this example, we extract from the L2 header 48-bit MAC destination address for LookupKey.
- **Four Way Hash Lookup PCD**—this is the second PCD in the chain. This type of PCD is used to perform a lookup using the LookupKey that was generated by the ‘Generate LookupKey’ PCD. In this example, we use 8-byte LookupKeySize and 2-bit Hash Key (4 sets in the lookup table) and base address of the lookup table is 0xfee1b000.
- **Last PCD**—this is the third and last PCD in the chain. This PCD is used at the end of PCD flow. It specifies the actions taken in case of lookup miss.

Figure 2 through Figure 4 describe the PCD configuration in this example.

base_addr = 0xfee1b600	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Configuration Example
base_addr + 0	PCD OPCODE=0x00								PCDIDValue								All zeros
base_addr + 2	MA Cdst	MA Csrc	TCI 1	TCI 2	Reserved										Src Port	PC DID	0x8000
base_addr+ 4	Reserved																All zeros
base_addr + 6	Reserved																All zeros

Figure 2. GenerateLookupKey_EthFast PCD

base_addr = 0xfee1b608	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	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																	All zeros

Figure 3. Four Way Hash Lookup PCD

base_addr = 0xfee1b610	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Configuration Example
base_addr + 0	PCD OPCODE=0x3F								Reserved								0x3f00
base_addr + 2	DBM	Reserved						BDM	Reserved								All zeros
base_addr + 4	Reserved																All zeros
base_addr + 6																	All zeros

Figure 4. Last PCD Field Descriptions

2.4 Hash Lookup Table Initialization

The hash lookup table initialization is done using a command from the Ethernet command set—Add/Remove Entry in the hash lookup table. The command is issued by writing to the CECR register, and to CECDR registers. The CECDR register contains the base address in multi-user RAM where the data structure in Figure 5 resides.

In this example, the “Set entry in Hash lookup table Command Parameters” base address is 0xfee1a000, the lookup table base adders is 0xfee1b000, and the LookupKey is 0x0000000063650000. This means that frames with destination mac address of 0x000000006365 are accepted by the Ethernet controller.

Figure 5 describes the configuration of “Set entry in Hash lookup table Command Parameters” in this example.

base_addr = 0xfee1a000	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	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																	All zeros
base_addr+8	Secondary Lookup Table Base Address																All zeros
base_addr+A																	All zeros
base_addr+C	Reserved																All zeros
base_addr+E	Reserved																All zeros
base_addr+10 (TAD)	EXF	V	Rej	IWE=0	Res	Res	VTagOP				VNonVTagOP		Res		RQoS		0x4000
base_addr+12 (TAD)	VPriority			CFI	VID												All zeros
base_addr+14 (TAD)	Reserved								Reserved								All zeros

Figure 5. Set entry in Hash Lookup Table Command Parameters

base_addr = 0xfee1a000	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Configuration Example
base_addr+16 (TAD)	Reserved																All zeros
base_addr+18	Reserved																All zeros
base_addr+1A	Reserved																All zeros
base_addr+1C	Reserved																All zeros
base_addr+1E	Reserved																All zeros
base_addr+20	LookupKey (up to 16 bytes)																All zeros
base_addr+22																	All zeros
base_addr+24																	0x6365
base_addr+26																	All zeros
base_addr+28																	All zeros
base_addr+2A																	All zeros
base_addr+2C																	All zeros
base_addr+2E																	All zeros
base_addr + 30-3F	Reserved																All zeros
base_addr + 0x40-0xBF or 0x40-0x9F or 0x40-0x7F	Reserved (size depends on LookupKeySize: 24, 16 or 8 bytes respectively)																All zeros

Figure 5. Set entry in Hash Lookup Table Command Parameters (continued)

After initializing the data structure in Figure 5, the user must issue the ADD/REMOVE ENTRY IN HASH LOOKUP TABLE to build the hash table.

The CECDR register contains the base address in multi-user RAM where the data structure in Figure 5 resides. Therefore, to issue this command, the user must write the values in Table 4 to the registers.

Table 4. Issue 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 ADD/REMOVE ENTRY IN HASH LOOKUP TABLE.

After issuing the command, the user must wait until CECR[FLG] = 0.

At this stage, the user can choose to add another entry to the hash table. For example, a second entry is set when LookupKey is 0x0000000063690000. This means that frames with destination mac address of 0x000000006365 OR 0x000000006369 are accepted by the Ethernet controller. The user must change the value of LookupKey in Figure 5 to 0x0000000063690000 and to issue the “Add/Remove Entry in hash lookup table” command as described in Figure 4.

2.5 Software Support

This section provides “C” code that configures the extended parsing mode data structures only. The complete QE configuration Code of this example can be found in the attached software L2_PCD_Filtering.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),0x0000b500); // EXPGlobalParam

   //Extended Parsing Global Parameters table
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb500),0x0000b600); //Pointer to the First PCD
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb504),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb508),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb50c),0x00000000);

   //Parsing Command Descriptor (PCD)
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb600),0x00008000); //GenerateLookupKey_EthFast PCD, Filterring 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 PCDs Frame Filtering Configuration Example

This section describes the configuration example for L3 PCD filtering. In this example, the Ethernet controller performs filtering by IP Destination Address. Table 5 describes IPv4 (RFC791) header.

Table 5. IPv4 (RFC791)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Version				Header Length				TOS					00			Total Length															
Identification (for fragments)																Flags			Offset											
TTL								Protocol Type (PTYPE)								Header Checksum															
IPsrc address																															
IPdst address																															
Options (optional)																															

3.1 Rx Global Parameter RAM

In addition to the initialization instructions that are provided in Section 2.1, “Rx Global Parameter RAM,” the user must initialize REMODER[IWEn]=1, IWGlobalParam_Base and IWThreadsParam_Base in Rx global parameter RAM.

Table 6. Rx Global Parameter RAM

Offset	Bits	Name	Description	Configuration Example
All zeros	0–31	REMODER	Ethernet mode register.	0x90000800
0xC0	—	EXPGlobalParam	Base address for extended parsing global parameters. The user must allocate 16 bytes for this data structure.	0x0000b500
0xF0	—	IWGlobalParam_Base	Base address for interworking global parameters. The user must allocate 256 bytes for this data structure if the Ethernet controller is used in interworking mode.	0x0000b700
0xF4	—	IWThreadsParam_Base	Base address for interworking thread parameters. The user must 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, 1 BD per frame). The pointer must be aligned to 512 bytes. N*672 bytes are allocated if the incoming frames may be longer than MRBLR and if (MAXD1 and MAXD2) > MRBLR. For additional information on MRBLR values, see the [	0x00011000

3.2 Extended Parsing Mode Global Parameters

Initialization instructions provided in Section 2.2, “Extended Parsing Mode Global Parameters.”

3.3 Parsing Command Descriptor (PCD)

Only the first PCD in the chain is different from the chain that was configured in Section 2.3, “Parsing Command Descriptor (PCD)”:

1. **GenerateLookupKey_Eth PCD**—This type of PCD is used to generate a LookupKey from the L2, L3, and L4 Frame header fields on the Ethernet receive port. The LookupKey is used by subsequent PCDs 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 bit that is set the parser searches the corresponding header in the frame, and if not found, a “ParseMiss” occurs. This PCD contains a Pointer to next PCD used if a Parse Miss has occurred.

In this example, extract 32-bit IP destination address for LookupKey from the L3 header. The pointer to next PCD used if a “Parse Miss” occurred have been programed to point on “Last PCD”.

2. **Four Way Hash Lookup PCD.**
   As described in Section 2.3, “Parsing Command Descriptor (PCD)”
3. **Last PCD.**
   As described in Section 2.3, “Parsing Command Descriptor (PCD)”

Figure 6 through Figure 8 describe the PCD's configuration in this example.

base_addr = 0xfee1b600	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Configuration Example
base_addr + 0	PCD OPCODE=0x01								PCDIDValue								0x0100
base_addr + 2	TCI 1NE	TCI 2NE	TCI 3NE	TCI 4NE	TCI 5NE	TCI 6NE	0	0	Reserved								All zeros
base_addr+ 4	UIE	Reserved	FFrg	TTL En	Reserved				MissPCDPTR								0x0001
base_addr + 6	MissPCDPTR																0xb618
base_addr + 8	TCI 1	TCI 2	TCI 3	TCI 4	TCI 5	TCI 6	noIP Opt	noIPFrg	PPPoE Session	IP	TCP	UDP	SCTP	Reserved			All zeros
base_addr+ A	Reserved						noICMP	noIGMP	noARP	noDHCP	noTCPCtl	Reserved					All zeros
base_addr + C	MAC dst (6)	MAC src (6)	TCI 1 (2)	TCI 2 (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																All zeros

Figure 6. Generate LookupKey_Eth PCD

base_addr = 0xfee1b610	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	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 7. Four Way Hash Lookup PCD

base_addr = 0xfee1b618	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Configuration Example
base_addr + 0	PCD OPCODE=0x3F								Reserved								0x3f00
base_addr + 2	DBM	Reserved					BDM		Reserved								All zeros
base_addr + 4	Reserved																All zeros
base_addr + 6																	All zeros

Figure 8. Last PCD Field Descriptions

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 that given in Section 2.4, “Hash Lookup Table Initialization.”

In this example the LookupKey is 0xdf895f0900000000. This means that frames with IP destination address of 0xdf895f09 (223.137.95.9) is accepted by the Ethernet controller.

3.5 Interworking Global Parameters Initialization

The interworking global parameters are located at base address programmed in IWGlobalParam_Base entry in the Rx global parameter RAM.

The user must initialize the field “IW_EthLenType” in the “Interworking Global Parameters” table:

Table 7. Interworking Global Parameters Description

base_addr = 0xfee1b700	Size (Bits)	Name	Description	Configuration Example
base_addr + 0x00-0x4c	—	—	Bits must be cleared.	All zeros
base_addr + 0x4e	16	IW_EthLenType	Value compared to Ethernet Length/Type field to distinguish Ethernet and 802.3 type of frames. Most common value is 0x0600.	0x0600
base_addr + 0x50-0x7B	—	—	Bits must be cleared.	All zeros

In this example, we are NOT using IW mode. Therefore, the IW Global Parameters Data structure is initialized to zero (except of the field IW_EthLenType which is being used).

3.6 Software Support

This section provide “C” code which configures the Extended parsing mode data structures only. The complete QE configuration Code of this example can be found in the attached software L3_PCD_Filtering.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),0x0000b500);  // EXPGlobalParam
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0x57f0),0x0000b700);  // IWGlobalParam_Base
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0x57f4),0x00011000);  // IWThreadsParam_Base

   //Extended Parsing Global Parameters table
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb500),0x0000b600);  //Pointer to the First PCD
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb504),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb508),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb50c),0x00000000);

   //Parsing Command Descriptor (PCD)
WRITE_UINT32(*(uint32 *)0xfee1b600,0x01000000); //GenerateLookupKey_Eth PCD, Flittering 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 PCDs Frame Filtering Configuration Example

This section describes the configuration example for L4 PCD filtering. In this example, the Ethernet controller performs filtering by the UDP destination port. Table 8 describes the UDP (RFC768) header.

Table 8. UDP (RFC768)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Portsrc																Portdst															
Length																Checksum															

4.1 Rx Global Parameter RAM

Initialization instructions are provided in Section 3.1, “Rx Global Parameter RAM.”

4.2 Extended Parsing Mode Global Parameters

Initialization instructions are provided in Section 2.2, “Extended Parsing Mode Global Parameters.”

4.3 Parsing Command Descriptor (PCD)

The difference between the PCD chain in this example and the chain that is configured in Section 3.3, “Parsing Command Descriptor (PCD),” is in the first PCD “GenerateLookupKey_Eth.” Instead of the 32-bit IP destination address being extracted, the 16-bit UDP destination port is extracted.

Figure 9 through Figure 11 describe the PCD's configuration in this example.

base_addr = 0xfee1b600	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Configuration Example
base_addr + 0	PCD OPCODE=0x01								PCDIDValue								0x0100
base_addr + 2	TCI 1NE	TCI 2NE	TCI 3NE	TCI 4NE	TCI 5NE	TCI 6NE	0	0	Reserved								All zeros
base_addr+ 4	UIE	Reserved	FFrg	TTL En	Reserved				MissPCDPTR								0x0001
base_addr + 6	MissPCDPTR																0xb618
base_addr + 8	TCI 1	TCI 2	TCI 3	TCI 4	TCI 5	TCI 6	noIP Opt	noIPFrg	PPPoE Session	IP	TCP	UDP	SCTP	Reserved			All zeros
base_addr+ A	Reserved						noICMP	noIGMP	noARP	noDHCP	noTCPCtl	Reserved					All zeros
base_addr + C	MAC dst (6)	MAC src (6)	TCI 1 (2)	TCI 2 (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																All zeros

Figure 9. Generate L2 LookupKey PCD

base_addr = 0xfee1b610	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	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 10. Four Way Hash Lookup PCD

base_addr = 0xfee1b618	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Configuration Example
base_addr + 0	PCD OPCODE=0x3F								Reserved								0x3f00
base_addr + 2	DBM	Reserved						BDM	Reserved								All zeros
base_addr + 4	Reserved																All zeros
base_addr + 6																	All zeros

Figure 11. 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 given in Section 2.4, “Hash Lookup Table Initialization.”

In this example, the LookupKey is 0x1111000000000000, which means that only frames with UDP destination port of 0x1111 are accepted by the Ethernet controller.

4.5 Interworking Global Parameters Initialization

Initialization instructions provided in Section 3.5, “Interworking Global Parameters Initialization.”

4.6 Software Support

This section provide “C” code which configures the Extended parsing mode data structures only. The complete QE configuration Code of this example can be found in the attached software L4_PCD_Filtering.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),0x0000b500);  // EXPGlobalParam
   WRITE_UINT32(*(uint32 *)(CE_PRAM + 0x57f0),0x0000b700);  // IWGlobalParam_Base
   WRITE_UINT32(*(uint32 *)(CE_PRAM + 0x57f4),0x00011000);  // IWThreadsParam_Base

   //Extended Parsing Global Parameters table
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb500),0x0000b600);  //Pointer to the First PCD
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb504),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb508),0x00000000);
```

```
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb50c),0x00000000);

   //Parsing Command Descriptor (PCD)
WRITE_UINT32(*(uint32 *)0xfee1b600,0x01000000); //GenerateLookupKey_Eth PCD, Flittering 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 PCDs Frame Filtering—4 Channels Configuration Example

The configuration described in this section filters UDP Packets to four receive channels (one channel for each core). Incoming UDP packets are filtered on UDP destination port bases and forwarded by QE into the appropriate receive channel.

Table 9 describes the mapping between UDP destination port and channel number (core number) in this example.

Table 9. 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

Initialization instructions are provided in Section 3.1, “Rx Global Parameter RAM.”

5.2 Extended Parsing Mode Global Parameters

Initialization instructions are provided in Section 2.2, "Extended Parsing Mode Global Parameters."

5.3 Parsing Command Descriptor (PCD)

Initialization instructions are provided in Section 4.3, "Parsing Command Descriptor (PCD)."

5.4 Hash Lookup Table Initialization

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

To set four entries in the table, the user must issue the command "ADD/REMOVE ENTRY IN HASH LOOKUP TABLE" four times, each time with a different LookupKey and a different TAD[VPriority]. The value in TAD[VPriority] determines the Rx channel of the incoming frames.

Table 10 describes the configuration value of the fields "LookupKey" and "TAD[VPriority]" in "Set entry in Hash Lookup Table Command Parameters" for each one of the four entries in the lookup table.

Table 10. Configuration Value

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

In this example, the "Set entry in Hash Lookup Table Command Parameters" base address is 0xfee1a000, the lookup table base address is 0xfee1b000, the LookupKey and TAD[VPriority] are described in Table 8-73 in Section 8.6.2.6.5 of the *QUICC Engine™ Block Reference Manual with Protocol Interworking*.

Figure 12 describes the configuration of "Set entry in Hash Lookup Table Command Parameters" in this example.

<table>
<tr><th>base_addr = 0xfee1a000</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>Configuration Example</th></tr>
<tr><td>base_addr + 0</td><td colspan="2">ADDE</td><td>EX LT</td><td>SLT E</td><td>HT FI</td><td colspan="3">Reserved</td><td colspan="8">Reserved</td><td>0x8800</td></tr>
<tr><td>base_addr + 2</td><td colspan="8">LookupKeySize</td><td colspan="4">Reserved</td><td colspan="4">HashKeySize</td><td>0x3f01</td></tr>
<tr><td>base_addr + 4</td><td colspan="16" rowspan="2">Lookup Table Base Address</td><td>0xfee1</td></tr>
<tr><td>base_addr + 6</td><td>0xb000</td></tr>
</table>

Figure 12. Set entry in Hash Lookup Table Command Parameters

base_addr = 0xfee1a000	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Configuration Example
base_addr+8	Secondary Lookup Table Base Address																All zeros
base_addr+A																	All zeros
base_addr+C	Reserved																All zeros
base_addr+E	Reserved																All zeros
base_addr+10 (TAD)	EXF	V	Rej	IWE=0	Res	Res	VTagOP				VNonVTagOP	Res			RQoS		0x4000
base_addr+12 (TAD)	VPriority			CFI	VID												All zeros OR 0x2000 OR 0x4000 OR 0x6000
base_addr+14 (TAD)	Reserved								Reserved								All zeros
base_addr+16 (TAD)	Reserved																All zeros
base_addr+18	Reserved																All zeros
base_addr+1A	Reserved																All zeros
base_addr+1C	Reserved																All zeros
base_addr+1E	Reserved																All zeros
base_addr+20	LookupKey (up to 16 bytes)																0x1110 OR 0x1111 OR 0x1112 OR 0x1113
base_addr+22																	All zeros
base_addr+24																	All zeros
base_addr+26																	All zeros
base_addr+28																	All zeros
base_addr+2A																	All zeros
base_addr+2C																	All zeros
base_addr+2E																	All zeros
base_addr 30-3F	Reserved																All zeros
base_addr + 0x40-0xBF or 0x40-0x9F or 0x40-0x7F	Reserved (size depends on LookupKeySize: 24, 16 or 8 bytes respectively)																All zeros

Figure 12. Set entry in Hash Lookup Table Command Parameters (continued)

After each initialization of the data structure in Figure 12, the user must issue the command "ADD/REMOVE ENTRY IN HASH LOOKUP TABLE" (as described in Table 4) to add an entry to the hash table.

5.5 Interworking Global Parameters Initialization

Initialization instructions are provided in Section 3.5, "Interworking Global Parameters Initialization."

5.6 Software Support

This section provides "C" code that configures the extended parsing mode data structures only. The complete QE configuration code of this example can be found in the attached software. L4_4channels_PCD_Filtering.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),0x0000b500);  // EXPGlobalParam
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0x57f0),0x0000b700);  // IWGlobalParam_Base
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0x57f4),0x00011000);  // IWThreadsParam_Base

   //Extended Parsing Global Parameters table
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb500),0x0000b600);  //Pointer to the First PCD
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb504),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb508),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xb50c),0x00000000);

   //Parsing Command Descriptor (PCD)
WRITE_UINT32(*(uint32 *)0xfee1b600,0x01000000); //GenerateLookupKey_Eth PCD, Flittering 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);
}
```

6 Revision History

Table 11 provides a revision history for this application note.

Table 11. Document Revision History

Rev. Number	Date	Substantive Change(s)
0	10/2009	Initial public release.

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 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: AN3880
Rev. 0
10/2009