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).

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.
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>
<thead>
<tr>
<th>MAC Destination Address</th>
<th>MAC Destination Address (cont)</th>
<th>MAC Source Address</th>
<th>MAC Source Address (cont)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPID = 0x8100</td>
<td>VPRI</td>
<td>CFI</td>
<td>VLAN ID</td>
</tr>
<tr>
<td>EType</td>
<td>L3 Header</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 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>
<thead>
<tr>
<th>Offset Bits Name</th>
<th>Description</th>
<th>Configuration Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>REMODER 0x00</td>
<td>Ethernet Mode Register</td>
<td>0x80000800</td>
</tr>
<tr>
<td>EXPGlobalParam</td>
<td>Base Address for Extended Parsing Global Parameters. The user needs to allocate 16 bytes for this data structure.</td>
<td>0x0000b500</td>
</tr>
</tbody>
</table>

### 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>
<thead>
<tr>
<th>base_addr = 0xfe1b500</th>
<th>Bits</th>
<th>Name</th>
<th>Description</th>
<th>Configuration Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>base_addr +0x0</td>
<td>0–7</td>
<td>Reserved</td>
<td>Set to zero</td>
<td>All zeros</td>
</tr>
<tr>
<td>base_addr +0x0</td>
<td>8–15</td>
<td>L2PCDPRTR</td>
<td>Pointer to first parse command descriptor base address. The user is responsible for allocation of space in the multi-user RAM for the PCDs.</td>
<td>0x01</td>
</tr>
<tr>
<td>base_addr +0x2</td>
<td>0–15</td>
<td>—</td>
<td>Reserved</td>
<td>All zeros</td>
</tr>
<tr>
<td>base_addr +0x4-7</td>
<td>—</td>
<td>Reserved</td>
<td>Set to zero</td>
<td>All zeros</td>
</tr>
<tr>
<td>base_addr +0x8-F</td>
<td>—</td>
<td>Reserved</td>
<td>Set to zero</td>
<td>All zeros</td>
</tr>
</tbody>
</table>
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 LookupKeyIdSize 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 Cds t | MA Csr t | 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          | LookupKeyIdSize | Reserved | EXT HashKeyIdSize | 0x3f01 |
| base_addr + 4          | LookupTableBase | 0xfee1 |
| base_addr + 6          | All zeros |

**Figure 3. Four Way Hash Lookup PCD**
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.
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

<table>
<thead>
<tr>
<th>Register</th>
<th>Access Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CECDR</td>
<td>Write</td>
<td>0xfee1a000</td>
<td>Pointer to “Set entry in Hash Lookup Table Command Parameters”</td>
</tr>
<tr>
<td>CECR</td>
<td>Write</td>
<td>0x03c10013</td>
<td>issue ADD/REMOVE ENTRY IN HASH LOOKUP TABLE.</td>
</tr>
</tbody>
</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.

```c
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, 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),0xa00c);
    WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa010),0x0a000000); //N=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);
}
```

Ethernet Extended Frame Filtering (PCD), Rev. 0
L2 PCDs Frame Filtering Configuration Example

```
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

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 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)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Bits</th>
<th>Name</th>
<th>Description</th>
<th>Configuration Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>0–31</td>
<td></td>
<td>REMODER</td>
<td>Ethernet mode register.</td>
<td>0x90000800</td>
</tr>
<tr>
<td>0xC0</td>
<td></td>
<td>EXPGlobalParam</td>
<td>Base address for extended parsing global parameters. The user must allocate 16 bytes for this data structure.</td>
<td>0x0000b500</td>
</tr>
<tr>
<td>0xF0</td>
<td></td>
<td>IWGlobalParam_Base</td>
<td>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.</td>
<td>0x0000b700</td>
</tr>
<tr>
<td>0xF4</td>
<td></td>
<td>IWThreadsParam_Base</td>
<td>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<em>576 bytes are allocated if it is guaranteed that all incoming frames are shorter than MRBLR or if (MAXD1 and MAXD2) &lt;= MRBLR (that is, 1 BD per frame). The pointer must be aligned to 512 bytes. N</em>672 bytes are allocated if the incoming frames may be longer than MRBLR and if (MAXD1 and MAXD2) &gt; MRBLR. For additional information on MRBLR values, see the [</td>
<td>0x00011000</td>
</tr>
</tbody>
</table>

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 programmed 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        |
| base_addr + 2          | TCI 1N E | TCI 2N E | TCI 3N E | TCI 4N E | TCI 5N E | TCI 6N E | 0 | 0 |   |   |     |     |     |     |     | Reserved               |
| base_addr + 4          | UIE | Reserved | FF rg | TTL En | Reserved | MissPCDPIFRG | 0x0000 |
| base_addr + 6          | MissPCDPTR | 0xb618 |
| base_addr + 8          | TCI 1 | TCI 2 | TCI 3 | TCI 4 | TCI 5 | TCI 6 | nol IP Opt | no IP Frag | PP PoE Session | IP | TC P | UD P | SC TP | Reserved | All zeros               |
| base_addr + A          | Reserved | no CM P | no G MP | no ARP | no DHCP | no TCP | Reserved | All zeros               |
| base_addr + C          | MAC dst (6) | MAC src (6) | TCI 1 (2) | TCI 2 (2) | EType (2) | PP P SID (2) | PP P PID (2)q | IP Src cm (4) | IP Dst (4) | PT YP E (1) | IP TOS (1) | TU SP src (2) | TU SP dst (2) | TF I (1) | Source Port (1) | PC DID (1) | 0x0080 |
| base_addr + E          | Reserved | All zeros |

**Figure 6. Generate LookupKey_Eth PCD**
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>
<thead>
<tr>
<th>base_addr = 0xfeef1b618</th>
<th>Size (Bits)</th>
<th>Name</th>
<th>Description</th>
<th>Configuration Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>base_addr + 0</td>
<td>16</td>
<td>IW_EthLenType</td>
<td>Value compared to Ethernet Length/Type field to distinguish Ethernet and 802.3 type of frames. Most common value is 0x0600.</td>
<td>0x0600</td>
</tr>
<tr>
<td>base_addr + 4</td>
<td>—</td>
<td>—</td>
<td>Bits must be cleared.</td>
<td>All zeros</td>
</tr>
<tr>
<td>base_addr + 6</td>
<td>—</td>
<td>—</td>
<td>Bits must be cleared.</td>
<td>All zeros</td>
</tr>
</tbody>
</table>
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.

```c
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),0x00000000);

    //Set entry in Hash Lookup Table Command Parameters
    WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa000),0x88003f01);
    WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa004),0xe00a04);
    WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa008),0xe000a8);
    WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa00c),0x80000000);
    WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa010),0x40000000);
    WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa014),0x00000000);
    WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa018),0x00000000);
}
```
L3 PCDs Frame Filtering Configuration Example

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 + 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 + 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),0);
}
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;
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)](image)

| 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.

<table>
<thead>
<tr>
<th>base_addr = 0xfee1b600</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>
</thead>
<tbody>
<tr>
<td>base_addr + 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PCD OPCODE=0x01</td>
</tr>
<tr>
<td>base_addr + 2</td>
<td>TCI</td>
<td>TCI</td>
<td>TCI</td>
<td>TCI</td>
<td>TCI</td>
<td>TCI</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>base_addr + 4</td>
<td>UIE</td>
<td>FF</td>
<td>TTL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MissPCDPTR</td>
</tr>
<tr>
<td>base_addr + 6</td>
<td>MissPCDPTR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x0001</td>
</tr>
<tr>
<td>base_addr + 8</td>
<td>TCI</td>
<td>TCI</td>
<td>TCI</td>
<td>TCI</td>
<td>TCI</td>
<td>TCI</td>
<td>nol Opt</td>
<td>nol ICMP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>base_addr + A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>All zeros</td>
</tr>
<tr>
<td>base_addr + C</td>
<td>MAC dst (6)</td>
<td>MAC src (6)</td>
<td>TCI</td>
<td>TCI</td>
<td>TCI</td>
<td>TCI</td>
<td>TCI</td>
<td>PP P P ID (2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>base_addr + E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>All zeros</td>
</tr>
</tbody>
</table>

**Figure 9. Generate L2 LookupKey PCD**

<table>
<thead>
<tr>
<th>base_addr = 0xfee1b610</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>
</thead>
<tbody>
<tr>
<td>base_addr + 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PCD OPCODE=0x20</td>
</tr>
<tr>
<td>base_addr + 2</td>
<td>LookupKeySize</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>base_addr + 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EX T HashKeySize</td>
</tr>
<tr>
<td>base_addr + 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0xfee1</td>
</tr>
</tbody>
</table>

**Figure 10. Four Way Hash Lookup PCD**
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 provides “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)),0x0b00000000);
    WRITE_UINT32(*((uint32 *)(CE_PRAM + 0xb508)),0x00000000);

    // 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)),0x0b00000000);
    WRITE_UINT32(*((uint32 *)(CE_PRAM + 0xb508)),0x00000000);

    // 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)),0x0b00000000);
    WRITE_UINT32(*((uint32 *)(CE_PRAM + 0xb508)),0x00000000);
L4 PCDs Frame Filtering Configuration Example

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 *)(0xfee1b618,0x30000000)); //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 + 0xb0000000); //LookupTableBase
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa008,0x00000000)); //Secondary LookupTableBase
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa00c,0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa010,0x00000000)); //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 + 0xa03c),0x00000000);
WRITE_UINT32(*(uint32 *)(CE_PRAM + 0xa03e),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>
<thead>
<tr>
<th>UDP Destination Port</th>
<th>Channel Number (Core Number)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1110</td>
<td>0</td>
</tr>
<tr>
<td>0x1111</td>
<td>1</td>
</tr>
<tr>
<td>0x1112</td>
<td>2</td>
</tr>
<tr>
<td>0x1113</td>
<td>3</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Entry Number</th>
<th>UDP Dest Port</th>
<th>Channel Number (Core Number)</th>
<th>LookupKey</th>
<th>TAD[VPriority]</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Entry</td>
<td>0x1110</td>
<td>0</td>
<td>0x1110000000000000</td>
<td>0</td>
</tr>
<tr>
<td>Second Entry</td>
<td>0x1111</td>
<td>1</td>
<td>0x1111000000000000</td>
<td>1</td>
</tr>
<tr>
<td>Third Entry</td>
<td>0x1112</td>
<td>2</td>
<td>0x1112000000000000</td>
<td>2</td>
</tr>
<tr>
<td>Forth Entry</td>
<td>0x1113</td>
<td>3</td>
<td>0x1113000000000000</td>
<td>3</td>
</tr>
</tbody>
</table>

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 10. Configuration Value**

**Figure 12. Set entry in Hash Lookup Table Command Parameters**
<table>
<thead>
<tr>
<th>base_addr = 0xffe1a000</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>
</thead>
<tbody>
<tr>
<td>base_addr+8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>All zeros</td>
<td></td>
</tr>
<tr>
<td>base_addr+A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>All zeros</td>
<td></td>
</tr>
<tr>
<td>base_addr+C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>base_addr+E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>All zeros</td>
<td></td>
</tr>
<tr>
<td>base_addr+10 (TAD)</td>
<td>EX</td>
<td>V</td>
<td>Rej</td>
<td>IW</td>
<td>Res</td>
<td>Res</td>
<td>VTagOP</td>
<td>VN onV Tag OP</td>
<td>Res</td>
<td>RQoS</td>
<td>0x4000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>base_addr+12 (TAD)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>All zeros OR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x2000 OR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x4000 OR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x6000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>base_addr+14 (TAD)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>All zeros</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>base_addr+16 (TAD)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>All zeros</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>base_addr+18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>All zeros</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>base_addr+1A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>All zeros</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>base_addr+1C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>All zeros</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>base_addr+1E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>All zeros</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>base_addr+20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x1110 OR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x1111 OR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x1112 OR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x1113</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>base_addr+22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>All zeros</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>base_addr+24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>All zeros</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>base_addr+26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>All zeros</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>base_addr+28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>All zeros</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>base_addr+2A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>All zeros</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>base_addr+2C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>All zeros</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>base_addr+2E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>All zeros</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>base_addr 30-3F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>base_addr + 0x40-0x8F or</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved (size depends on LookupKeySize: 24, 16 or 8 bytes respectively)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x40-0x9F or</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>All zeros</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x40-0x7F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>All zeros</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.

```c
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 *)0xfee1b614,0x0001b600);

    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
```

Ethernet Extended Frame Filtering (PCD), Rev. 0
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)),0x111i000000000000); //LookupKey = 0x11i00000000000000
  WRITE_UINT32(*((uint32 *)(CE_PRAM + 0xa010)),0x40000000); //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)
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>
<thead>
<tr>
<th>Rev. Number</th>
<th>Date</th>
<th>Substantive Change(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10/2009</td>
<td>Initial public release.</td>
</tr>
</tbody>
</table>