

## FCC Ethernet UDP Forwarding using Microcode

This is a limited-functionality demo version

The demo microcode module only recognizes and passes UDP frames, and suppresses any none-UDP frames (including ping).

The full version enables the reception of all protocol types, phrases them to 8 Rx queues, and forwards selected UDP frames, based on IP and port number.

If you wish to obtain a copy of the full version then please contact DoGav Systems.

### 1. PRODUCT DESCRIPTION

This microcode module is an evaluation copy that illustrates the throughput boost of implementing microcode at the CPM level. It shows the speed and cost-effectiveness of microcode compared to the traditional methods of high-level code or costly, dedicated hardware. This module is applicable to any PQII or PQIII FCC channel that is configured to operate in an Ethernet mode.

The microcode analyzes each incoming frame. If it's a UDP frame, the microcode then checks to see if the destination port number is listed as a "looped back" port in the port table.

If so, the frame is forwarded by inserting it into the next available TxBD; otherwise it is discarded. If the frame is not UDP or is not assigned as loop-backed in the port table, it is discarded.

In addition, the user can replace the DA MAC with user-selected MAC address. Any combination of the 64K ports can be dynamically selected as looped back.

This microcode module is applicable to a single BD frame. The application must allocate long enough buffers to meet this limit and also adjust the MRBLR parameter accordingly.

While running under this microcode neither RxBD rings nor TxBD rings are available to the application. The microcode swaps exhausted buffers from the TxBD to the RxBD when forwarding a frame (if TxBD is not in Ready state).

It can be tailored to specific requests. For example, the following enhancements can be implemented:

- Multiple RxBD rings. Each ring represents user-defined pattern (Protocols, VLAN, Ports, etc.).
- Multiple TxBD rings, to support user-defined priority.
- QoS (WFQ algorithm).
- Frames which span over a number of BD.
- Checksum check (IP & TCP/UDP header) of the incoming frames.
- Checksum calculation (IP & TCP/UDP header) of the outgoing frames.
- Changing the contents of any field in the outgoing frame's header.
- Answering ARP and PING automatically.
- Recognizing exact MAC addresses for more than a single address.
- Filtering out frames based on Ethernet type, IP type, port number, IP address, etc.
- Smart forwarding (queuing transmit requests and inserting them into the TxBD ring automatically).
- Extend forwarding to other protocols, such as TCP, RTP, etc.

## 2. MICROCODE STRUCTURE

For this microcode module, the user must initialize a 64-byte structure. The structure must be located on the DPRAM with a 2-byte address that's divisible by 64. The address (the offset from DPRAM start) should be written to offset 0xFC of the relevant FCC parameter RAM.

The following is the definition of the microcode structure:

Offset	Name	Width	Description
0x00	tmp_reg1	Word	Microcode managed parameter; initialize to zero.
0x04	tmp_reg2	Word	Microcode managed parameter; initialize to zero.
<b>0x08</b>	<b>udp_table_addr</b>	<b>Word</b>	<b>Initialize to the address of the UDP forwarding table. The address must be aligned to 4 (divisible by 4).</b>
0x0C	udp_record	Word	Microcode managed parameter; initialize to zero.
<b>0x10</b>	<b>tbase</b>	<b>Word</b>	<b>Initialize to the base address of the TxBD ring.</b>

<b>0x14</b>	<b>tbptr</b>	<b>Word</b>	<b>Holds the address of the current BD in the TxBD ring. Must be initialized to tbase.</b>
0x18	tbd_stat	Hword	Microcode managed parameter; initialize to zero.
0x1A	tbd_len	Hword	Microcode managed parameter; initialize to zero.
0x1C	tbd_ptr	Word	Microcode managed parameter; initialize to zero.
<b>0x20</b>	<b>mac_d</b>	<b>Word</b>	<b>Initialize this to the first 4 bytes of the MAC address that is inserted to the forwarded frame as destination address. The address should be written here in <u>little-endian</u> form.</b>
<b>0x24</b>	<b>mac_d2</b>	<b>Hword</b>	<b>Initialize to the last 2 bytes of the MAC address that is inserted to the forwarded frame as destination address. The address should be written here in <u>little-endian</u> form.</b>
0x26	discard_cnt	Hword	Counter of discarded frames due to lack of Tx BDs. Initialize to zero.
0x28	frdptr_org	Word	Microcode managed parameter; initialize to zero.
0x2C	not_ip_frame_cnt	Hword	Counter of discarded frames due to not being IP frames. Initialize to zero.
0x2E	not_udp_frame_cnt	Hword	Counter of discarded frames due to not being UDP frames. Initialize to zero.
0x30	not_port_frwd_cnt	Hword	Counter of discarded frames which hold a port that is not enabled for forwarding. Initialize to zero.
0x32	forward_cnt	Hword	Counter of frames forwarded by microcode. Initialize to zero.
0x34	reserved1	Word	Reserved; initialize to zero.
0x38	reserved2	Word	Reserved; initialize to zero.
0x3C	reserved3	Word	Reserved; initialize to zero.

### 3. UDP PORT TABLE

The received UDP frame's destination port number determines where frames will be forwarded. The user must initialize an 8-Kbytes table that resides in the external memory (typically the SDRAM). The table base address should be divisible by 4. Each bit in the table represents a UDP port. For example, bit 0 (the most significant bit) of the first byte represents port 0, and the most significant bit of the following byte represents port 8.

## 4. FCC BUFFER DESCRIPTORS

This microcode requires that the user allocate buffers for **all** of the BD, both Rx and Tx, before enabling the FCC channel. After enabling the channel, the user should not alter the BD contents.

## 5. FCC REGISTERS

The user can enable or disable the microcode. This is done with bit 7 in the FPSMR register of the relevant FCC channel. To enable this microcode, the user needs to turn on this bit (write a 1 to it). (In the manual, this bit is listed as reserved.) Once the microcode is installed and enabled, it is activated on all of the FCC channels that work in Fast Ethernet mode.

## 6. INSTALLATION

### 6.1. Package Contents

This package comes as a ZIP compressed file which contains the following files:

1. *InstallUcode.c* – This file contains two c functions, one for installation of the uCode and the other for uninstalling it.
2. *InstallUcode.h* – This file contains function prototypes.
3. *Eth\_UDP\_Forward\_Freeware.h* – This file contains the c-array representing the uCode and defines of the trap addresses.
4. *BasicDef.h* – This file contains definitions and typedefs required to compile the installation function.
5. *UdpForward.h* – This file contains definitions of the structures which are mentioned in this document.
6. FCC Ethernet UDP Forwarding.*pdf* – This document.
7. FCC Ethernet UDP Forwarding Evaluation License.*pdf*

### 6.2. Trap Usage

This microcode module occupies three traps.



### 6.3. Installation Procedure

The user should call the InstallUcode function at the **earliest** stage of the application. It MUST be called before initializing the FCC channels. The call should be made only once.

## 7. EXAMPLE

This example shows how to initialize the structure and set some ports to loop-backed mode.

```
void InitUdpForwarding()
{
    SUDP_Forward *pUdpForward;
    BD *pBD;
    int i;
    INT32 i32;
    // clean udp port table
    memset((void *) &Ports_Table, 0, sizeof(SUDP_Ports_Table));
    // enable port 20000
    Ports_Table.Port_Entry[20000 / 8] = (0x80 >> 20000 % 8);
    // enable port 34955
    Ports_Table.Port_Entry[34955 / 8] = (0x80 >> 34955 % 8);
    // set pointer to udp structure for microcode module
    pUdpForward = (SUDP_Forward *) (IMMR_BASE + 0x1000);
    // clear structure
    memset(pUdpForward, 0, sizeof(SUDP_Forward));
    // set base address of TxBD ring
    pUdpForward->tbase = (UINT32) TBASE_ADDR;
    pUdpForward->tbptr = pUdpForward->tbase;
    // set udp table address
    pUdpForward->udp_table_addr = (UINT32) &Ports_Table;
    // set destination MAC address to swap, in little-endian form
    pUdpForward->mac_d = 0x456E0C00;
    pUdpForward->mac_d2 = 0xF2A6;
    // init TxBDs
    for (i = 0, pBD = (BD *) TBASE_ADDR ; i < FCCFAST_NUM_TXBD ; i++,
pBD++)
    {
        // alloc buffer
        if ((i32 = (INT32) ALLOC_BUFF(FccFastEthPoolID, 1)) ==
enNoBuff)
        {
            while (TRUE) i++;
        }
        pBD->ptr = (UINT8 *) i32;
    }
    // need here to turn on bit 7 of FPSMR register in relevant FCC
}
```

### **ABOUT DOGAV SYSTEMS**

DoGav Systems is a leading provider of software and hardware consultancy and training services. It specializes in Freescale's processors, in particular the PowerQUICC family of communication processors. It has a proven track record of over 20 years supporting Freescale customers in developing market-leading products for the communications equipment market.

DoGav Systems is Freescale's most experienced and active microcode developer. Since receiving its license in 2000, it has developed numerous customized microcode packages for both small and large Freescale customers. These packages are now successfully deployed in commercial products. In addition, DoGav Systems also offers more than 30 off-the-shelf microcode products for the PowerQUICC I, PowerQUICC II, PowerQUICC III and PowerQUICC II Pro processors.

