Freescale Semiconductor
Application Note

Document Number: AN4196
Rev. 0, 8/2010

Ethernet Jumbo Frames in SmartDSP OS Applications

The maximum Ethernet frame size is defined as 1518 bytes in the **IEEE** 802.3 Ethernet standard [1]. A specially defined Ethernet jumbo frame can be used to increase the Ethernet throughput and to reduce the CPU/DSP core load. Note that Ethernet jumbo frames are not defined in the **IEEE** 802.3 standard, and only can be used in systems where both the sender and receiver support it. The definition of the Ethernet jumbo frame size is vendor-dependent [2].

This application note describes how to enable Ethernet jumbo frames in a SmartDSP OS application. It reviews the modifications necessary to enable the jumbo frames and provides a sample application running on the MSC8156ADS.

NOTE

Although this note refers to the MSC8156, the process used to enable jumbo frames applies to all members of that DSP family, including the MSC8151, MSC8152, MSC8154, MSC8154E, MSC8156, MSC8156E, MSC8251, MSC8252, MSC8254, and MSC8256 DSP devices, and the MSC8144 DSP family, including the MSC8144 and MSC8144E.

Contents

© 2010 Freescale Semiconductor, Inc.

freescale semiconductor

1 Application Physical Connections

This application note describes how to build a SmartDSP OS application that sends an Ethernet jumbo frame from the MSC8156ADS Ethernet Controller 1 (UEC0) and receive it using the MSC8156ADS Ethernet Controller 2 (UEC1). Figure 1 shows the connection diagram for this system.

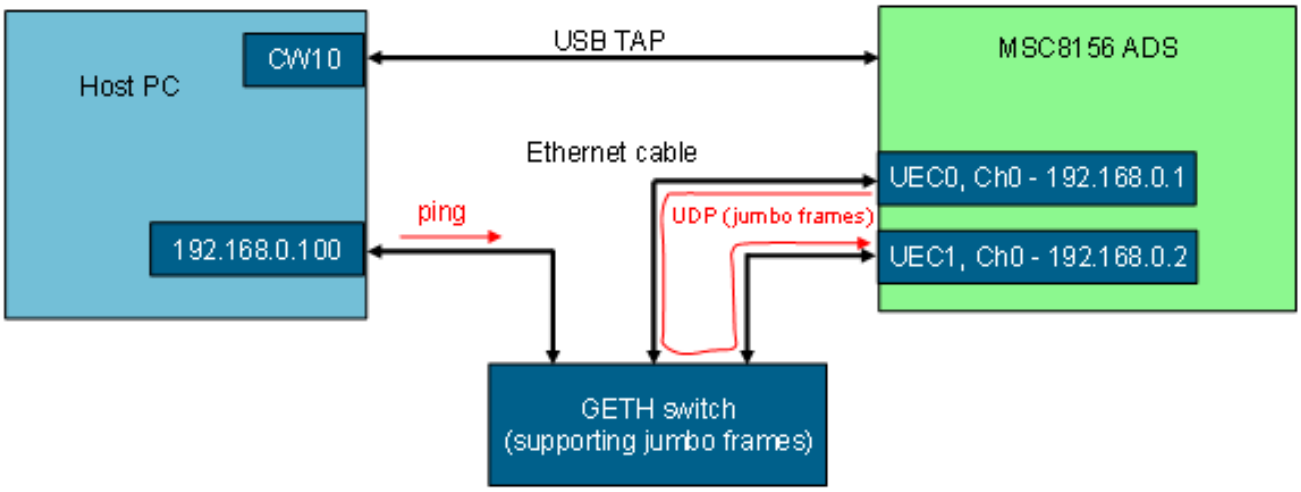

Figure 1. Example Application Connection Diagram

2 Application Configuration

The SmartDSP OS uses the Buffered I/O (BIO) module for sending to and receiving from buffered interfaces such as the Ethernet port. This module provides a high-level application programmer interface (API) for application development and low level drivers to send/receive data packets [3][4]. The BIO module uses an OS buffer pool and an OS frame data structure `os_frame_t` to transfer the data between application and the low level drivers. The OS buffer pool stores fixed length buffers. The buffer size and the number of the buffers are specified when the OS buffer pool is constructed by the application.

The OS frame data structure handles data buffers (analogous to file handles) and also can take and release the buffers in the OS buffer pool. There are two modes for the frame structure, one is single buffer frame, and the other is the multi buffer frame. We have to choose one of these modes for the application at compile time.

A single buffer frame uses a single buffer to store an entire Ethernet frame. For example, if the jumbo frame size is 9018 bytes, the buffer size must be defined as 9018 bytes or larger, and the OS frame data structure has a pointer to the buffer. The advantage of this mode is that the overhead is small because the data is located continuously. This disadvantage of this mode is that the memory efficiency is not good. To use the single buffer frame, define `FRAME_SINGLE_BUFFER_ONLY` prior to the first inclusion of `os_frame.h` (typically included hierarchically in `smartdsp_os.h`) for whole application sources. Figure 2 shows a conceptual diagram of the single buffer frame operation.

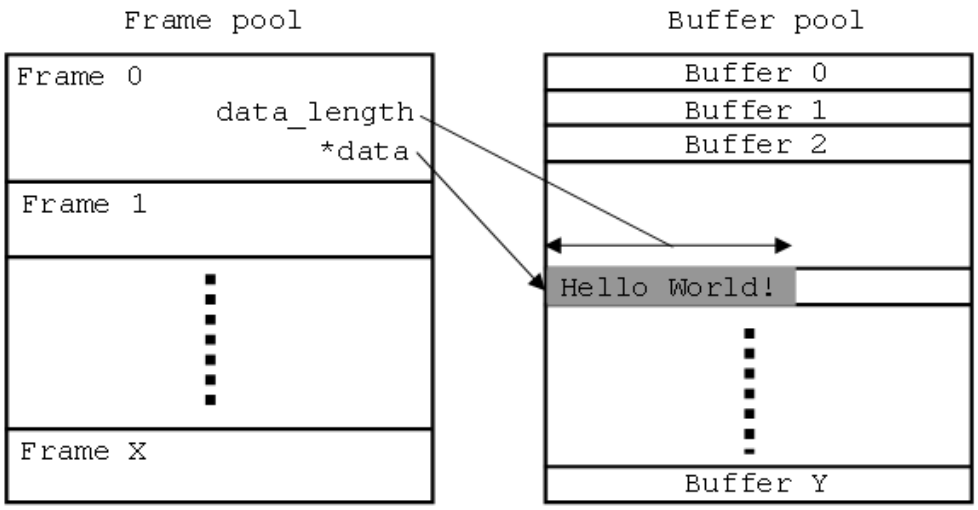

Figure 2. Single Buffer Frame Concept

Multi buffer frames manage multiple buffers to store a single Ethernet frame. Assuming the jumbo frame size is 9018 bytes, if we define the buffer size as 512 bytes, the OS frame data structure must manage up to 18 pointers for up to 18 buffers in the OS buffer pool to store the entire Ethernet frame (512 bytes * 18 = 9216 bytes). The advantage of this mode is that memory efficiency is better than the single buffer mode. The disadvantage is the overhead used to parse whole data stored in the multiple buffers. If we use the multi buffer frame, we must not define `FRAME_SINGLE_BUFFER_ONLY` in the application. Figure 3 show a conceptual diagram of the multi buffer frame operation.

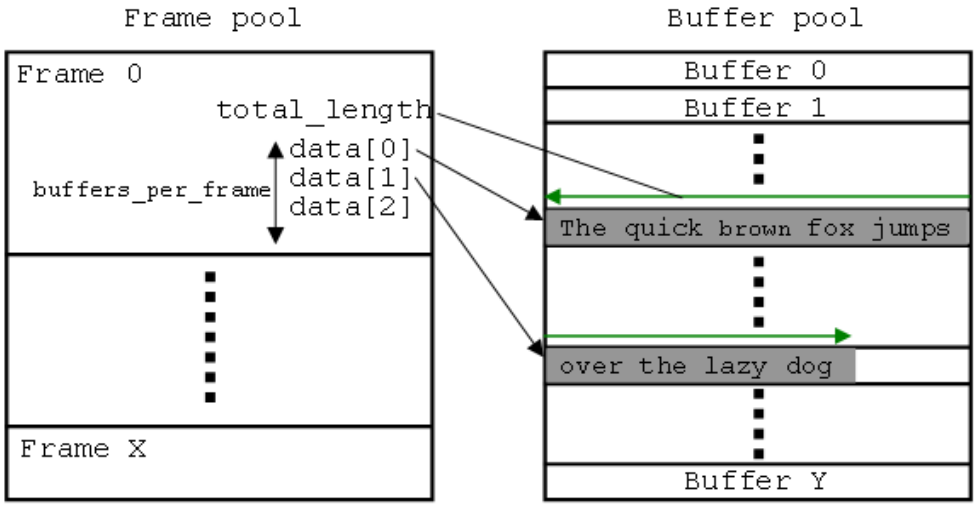

Figure 3. Multi-Buffer Frame Concept

In the sample program used in this application note, the frame mode is predefined in preprocessor macro settings, and the buffer size and the number of buffers in a frame are defined in `app_config.h` as shown in Figure 4.

```
#ifdef FRAME_SINGLE_BUFFER_ONLY
  /* Single buffer frame */
  #define TEST_BUF_SIZE             0x2400 //9216 bytes
  #define TEST_BUFS_IN_FRAME        1
#else
  /* Multi buffer frame */
  #define TEST_BUF_SIZE             0x200 //512 bytes
  #define TEST_BUFS_IN_FRAME        18 //512*18=9216 bytes
#endif
```

Figure 4. Example `app_config.h` File

To construct the OS buffer pool, we can call the `osMemPartCreate()` function during the application initialization stage.

3 UEC Driver Configurations

Turn on `MSC815X_UEC0` or `MSC815X_UEC1` to enable the UEC driver. Also, configure the frame mode for the UEC driver. For multi buffer frames, define `UEC0_MULTI_BUFFERED_FRAME` or `UEC1_MULTI_BUFFERED_FRAME` in `os_config.h`. Figure 5 shows an example of `os_config.h`.

```
#define MSC815X_UEC0                                    ON
#if MSC815X_UEC0==ON
  #ifndef FRAME_SINGLE_BUFFER_ONLY
#define UEC0_MULTI_BUFFERED_FRAMEON
  #endif
#endif

#define MSC815X_UEC1                                   ON
#if MSC815X_UEC1==ON
  #ifndef FRAME_SINGLE_BUFFER_ONLY
    #define UEC1_MULTI_BUFFERED_FRAME  ON
  #endif
#endif
```

Figure 5. Example of `os_config.h`

To enable Ethernet jumbo frames in the MSC8156, use the three fields in RX Global Parameter RAM listed in Table 1 to set the length so that it is larger than maximum jumbo frame size when initializing the application [5][6].

Table 1. Fields in the Rx Global Parameter RAM that Enable Ethernet Jumbo Frames

Offset	Bits	Name	Description	Initialized by
0x4C	0–15	MFLR	*Maximum frame length register (typically 1518 decimal).* If the Ethernet controller detects an incoming frame exceeding MFLR, it sets RxBD[LG] (frame too long) in the last RxBD but does not discard the rest of that frame. The controller also reports the frame status and length of the received frame. MFLR includes all in-frame bytes between the start frame delimiter and the end of the frame.	CPU

Table 1. Fields in the Rx Global Parameter RAM that Enable Ethernet Jumbo Frames (continued)

Offset	Bits	Name	Description	Initialized by
0x50	0–15	MAXD1	*Maximum DMA1 length register (typically 1520 decimal)*. Permits the user to prevent system bus writes after a frame exceeds a certain size. The MAXD1 value is valid only if an address match is detected. If the Ethernet controller detects an incoming Ethernet frame larger than the user-defined value in MAXD1, the rest of the frame is discarded. The Ethernet controller waits for the end of the frame and reports the frame status and the frame length in the last RxBD. This value must be greater than block size. The frame length includes the discarded bytes.	CPU
0x52	0–15	MAXD2	*Maximum DMA2 length register (typically 1520 decimal)*. Permits the user to prevent system bus writes after a frame exceeds a certain size. The value of MAXD2 is valid in promiscuous or Extended Parsing mode when no address match is detected and the Last PCD is reached. If the Ethernet controller detects an incoming Ethernet frame larger than the value in MAXD2, the rest of the frame is discarded. The Ethernet controller waits for the end of the frame and reports frame status and length in the last RxBD. In a monitor station, MAXD2 can be much less than MAXD1 to receive entire frames addressed to this station, but receive only the headers of all other frames. This value must be lower than MAXD1. The frame length includes the discarded bytes.	CPU

In SmartDSP OS, the three fields described in Table 1 are defined in the UEC structure as shown in Figure 6.

```
uec->uecRxGlobalPram->mflr
uec->uecRxGlobalPram->maxd1
uec->uecRxGlobalPram->maxd2
```

Figure 6. Fields in the UEC Structure Used to Enable Ethernet Jumbo Frames

These three fields are initialized using the `initRxGlobalParameterRam()` function in `msc815x_uec_init.c`, as listed in Figure 7.

```
static os_status initRxGlobalParameterRam(
        msc815x_uec_t                       *uec,
        msc815x_uec_init_params_t           *init_params,
        uec_eth_params_t                    *eth_params)
{
  ...
  WRITE_UINT16(uec->uecRxGlobalPram->mflr,
    (eth_params->max_frame_length & MAXFRM_MASK));

  ...
  WRITE_UINT16(uec->uecRxGlobalPram->maxd1,
   ((eth_params->max_frame_length > MAX_DMA1_LENGTH) ?
    eth_params->max_frame_length : MAX_DMA1_LENGTH));
  ...
  WRITE_UINT16(uec->uecRxGlobalPram->maxd2,
   ((eth_params->max_frame_length > MAX_DMA2_LENGTH) ?
    eth_params->max_frame_length : MAX_DMA2_LENGTH));
  ...
}
```

Figure 7. `initRxGlobalParameterRam()` function

In the context discussed, the value `eth_params->max_frame_length` initializes the three fields. We can define the initial value of `eth_params->max_frame_length` in `msc815x_config.c`.

The first thing to do is to define an initial value of `structure uec_eth_params_t` and set the `max_frame_length` field to the maximum Ethernet jumbo frame size. Figure 8 shows the example of the initial value defined in `msc815x_config.c`.

```
/* Ethernet initialize parameters for jumbo frames */
uec_eth_params_t my_eth_params =
{
  (TEST_BUF_SIZE * TEST_BUFS_IN_FRAME),
   /* max_frame_length must be equal to or larger
      than ETHER_MAX_LEN. */
  UEC_DEF_MIN_FRAME_LEN,
  UEC_DEF_PREAMBLE_LEN,
  UEC_DEF_NON_BTB_CS_IPG,
  UEC_DEF_NON_BTB_IPG,
  UEC_DEF_BTB_IPG,
  UEC_DEF_MIN_IFG,
  UEC_DEF_ALT_BEB_TRUNCATION,
  UEC_DEF_MAX_RETRANSMISSIONS,
  UEC_DEF_COLLISION_WINDOW,
  UEC_DEF_ETH_FLAGS
};
```

Figure 8. Initial Value of `uec_eth_params_t` Structure

Secondly, define an initial value of the structure `msc815x_uec_init_params_t`, and set the `eth_param` field to the pointer of the `uec_eth_params_t` structure shown in Figure 8. The UEC driver initialization done by the SmartDSP OS uses this structure. Figure 9 shows the example of defining the initial value in `msc815x_config.c`.

```
msc815x_uec_init_params_t msc815x_uec_init_params_0 =
{
  UEC_ID0,
  0xFEE02000,
  UEC_IF_RGMII,
  (UEC_DEF_FLAGS | UEC_CFG_ENABLE_STATISTICS |
   UEC_CFG_ENABLE_ADDITIONAL_STATISTICS),
  OS_HWI_PRIORITY0,
  TEST_TX_BD_RING_LEN,
  TEST_RX_BD_RING_LEN,
  TEST_BUF_SIZE,
  OS_NUM_OF_CORES*TEST_NUM_OF_CHANNELS,
  OS_NUM_OF_CORES*TEST_NUM_OF_CHANNELS,
  UEC_DEF_NUM_OF_THREADS,
  UEC_DEF_NUM_OF_THREADS,
  TRUE,
  NULL,
  &my_eth_params
    /* Set Ethernet initialization parameters. */
};
```

Figure 9. Initial value of msc815x_uec_init_params_t structure

4 SmartDSP OS Net Libraries Modifications

The maximum Ethernet frame length is hard-coded as a constant definition `ETHER_MAX_LEN` (= 1518) in the SmartDSP net library private header (see `SmartDSP/source/net/include/net_if_ethernet_.h0)`, as shown in Figure 10.

```
#define ETHER_MAX_LEN 1518 /* maximum frame length, including CRC */
```

Figure 10. Original definition of ETHER_MAX_LEN

We must modify the constant definition `ETHER_MAX_LEN` to enable the Ethernet jumbo frame, as shown in Figure 11.

```
#define ETHER_MAX_LEN 9018 /* maximum frame length, including CRC */
```

Figure 11. Modified definition of ETHER_MAX_LEN

You must rebuild the SmartDSP net libraries after modifying the definition.

5 Behavior of the Sample Application

The sample project contains the following four targets:

- `SBF_Debug`. Single buffer mode, Non-optimized
- `SBF_Release`. Single buffer mode, Optimized
- `MBF_Debug`. Multi buffer mode, Non-optimized
- `MBF_Release`. Multi buffer mode, Optimized

The application is configured as a single core application, defined by `OS_NUM_OF_CORES==1` in `os_config.h`. If we run it from CodeWarrior debugger, it should work using the following sequence:

1. Initializes UEC0 channel0 and UEC1 channel0
2. Binds UDP socket to each channel
3. Starts software timer with 1 second interval
4. Sends UDP jumbo frame from UEC0 channel0 at the timer interrupt handler
5. Receives UDP jumbo frames from UEC1 channel0, then calls the Rx callback function
6. In the Rx callback function, prints received bytes count into CodeWarrior debugger console view, as shown in Figure 12.

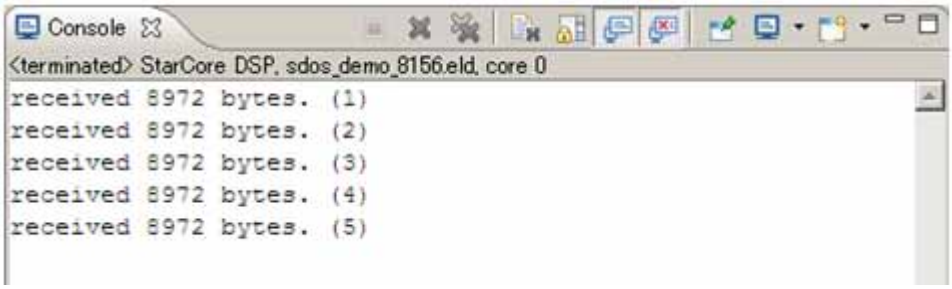

Figure 12. CodeWarrior Debugger Console View

6 Summary

The following list summarizes the main actions required to enable Ethernet jumbo frames in a SmartDSP OS application.

- Decide the maximum frame length in the application
- Decide the buffer size and number of buffers to store an Ethernet frame in order to configure the buffer mode and construct the buffer pool.
- Define structure `uec_eth_params_t` and `structure msc815x_uec_init_params_t` to initialize the UEC driver
- Modify the definition `ETHER_MAX_LEN` in SmartDSP OS net library header to the maximum frame length, then rebuild the SmartDSP OS net libraries

7 References

[1]. **IEEE** Std 802.3-2008: *Carrier Sense Multiple Access With Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*, **IEEE**, Dec. 2008

[2] *Jumbo/Giant Frame Support on Catalyst Switches Configuration Example* (http://www.cisco.com/en/US/products/hw/switches/ps700/products_configuration_example09186a008010edab.shtml), Cisco Systems, Inc., May-2005

[3] *SmartDSP OS User Guide*, Rev. 22, Freescale Semiconductor, Inc., Mar. 2010

[4] *SmartDSP OS API Reference Manual*, Rev.22, Freescale Semiconductor, Inc., Mar. 2010

[5] *MSC8156 Reference Manual*, Rev. I, Freescale Semiconductor, Inc., Dec. 2009

[6] *QUICC Engine Block Reference Manual with Protocol Interworking*, Rev. 2 draft1, Freescale Semiconductor, Inc., Apr. 2009.

NOTE

For the latest version of the Freescale documentation, consult you local sales office or representative.

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
+86 010 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

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters which may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals” must be validated for each customer application by customer’s technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale, the Freescale logo, and StarCore are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. QUICC Engine is a trademark of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© 2010 Freescale Semiconductor, Inc.,

Document Number: AN4196
Rev. 0
8/2010

freescale™
semiconductor