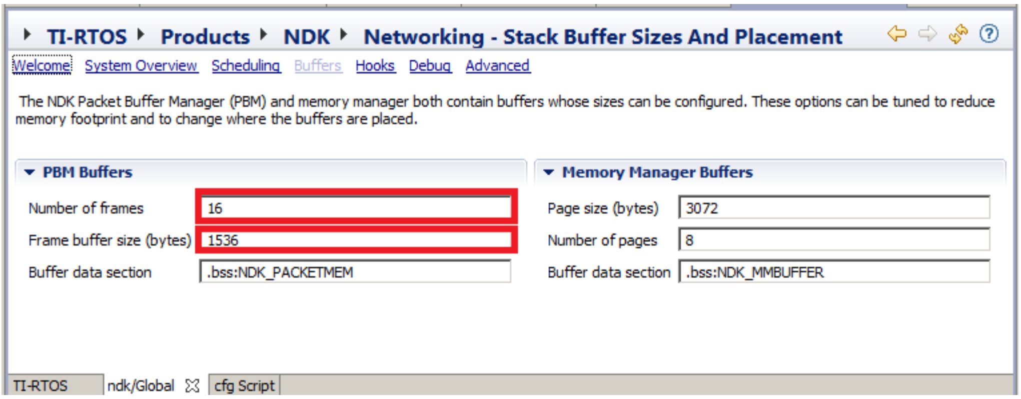
NDK Static Network Buffers

Contents

The NDK statically defines the network (packet) buffers. The Packet Buffer Manager (PBM) component is responsible for managing these buffers. There are two configuration parameters that dictate the amount of RAM that is used:

- **Global.pktNumFrameBufs**: Determines the number of network buffers are that present in the system
- **Global.pktSizeFrameBuf**: Determines the size of each buffer

To set these parameters, first open the NDK settings in your application's configuration file (*.cfg) as described in Configuring NDK Memory Use. Once you see the **TI-RTOS > Products > NDK > Networking - Welcome** configuration panel, choose the **Buffers** link.



The NDK Packet Buffer Manager (PBM) and memory manager both contain buffers whose sizes can be configured. These options can be tuned to reduce memory footprint and to change where the buffers are placed.

PBM Buffers

Number of frames: 16

Frame buffer size (bytes): 1536

Buffer data section: .bss:NDK_PACKETMEM

Memory Manager Buffers

Page size (bytes): 3072

Number of pages: 8

Buffer data section: .bss:NDK_MMBUFFER

About Ethernet frame buffers

Each frame buffer is used to store a single Ethernet frame.

Typically the size needed for an Ethernet frame is 1514 octets (bytes). This size allows for the 1500-byte Ethernet payload plus the 14-byte Ethernet header (6-byte destination MAC address + 6-byte source MAC address + 2-byte protocol type field). On devices with the data cache enabled, the value of 1536 bytes is generally used to allow for proper cache line alignment.

There are two main arrays: <syntaxhighlight lang='c'> INT8 ti_ndk_config_Global_pHdrMem[PKT_NUM_FRAMEBUF * sizeof(PBM_Pkt)] // PKT_NUM_FRAMEBUF = Global.pktNumFrameBufs UINT8 ti_ndk_config_Global_pBufMem[PKT_NUM_FRAMEBUF * PKT_SIZE_FRAMEBUF] // PKT_SIZE_FRAMEBUF = Global.pktSizeFrameBuf </syntaxhighlight> During startup, each of the `PBM_Pkt` entries in the `ti_ndk_config_Global_pHdrMem` array is initialized. Part of that initialization is to assign it a single packet buffer from `ti_ndk_config_Global_pBufMem`. The size of the buffer is `PKT_SIZE_FRAMEBUF`.

Note: Each packet is the same size. The NDK does not support chaining multiple packet buffers to hold a single Ethernet packet.

When a `PBM_alloc()` occurs, a `PBM_Pkt` entry is dequeued from a linked list and given to the requester. `PBM_alloc()` is called for the following reasons:

- NIMU Ethernet driver to receive incoming packets.
- Stack to transmit an application or service (such as ICMP) packet.

Note: The NDK does support zero-copy and jumbo packets. These topics are not part of this discussion.

How to reduce RAM usage

1. **Reduce the number of packets**: The number of packets (`Global.pktNumFrameBufs`) can be reduced to gain a significant RAM savings. However, if you reduce the number of packets too much, packets may be dropped when an incoming burst occurs. If this occurs with a TCP packet, the protocol will correct this via retransmission. For UDP, the packet is lost.
2. **Decrease the size of a packet**: The size of a packet (`Global.pktSizeFrameBuf`) can be reduced and significant RAM savings can occur. If you know there is a maximum size for the packets that will be received and transmitted, you can reduce the size. There are two risks to consider if you reduce the size of the packet:
    - An incoming Ethernet packet may be larger than `Global.pktSizeFrameBuf`.
    - An outgoing packet may be larger than `Global.pktSizeFrameBuf`.

   In both cases, the `PBM_alloc()` uses the Memory Manager (`mmAlloc()`) to allocate a `PBM_Pkt` entry and a large enough buffer to hold the packet. When `PBM_free()` is called, the memory is returned to the Memory Manager (`mmFree()`). If no memory is available in the Memory Manager, the allocation fails. For most driver implementations, the incoming packet will be dropped as a result.

Unfortunately, there is currently no way to query usage of the packet buffers (for example, to see the high-water mark) or the maximum size requested during the current run. We are looking into the best way to add this enhancement.

{{ 1. switchcategory:MultiCore=	Keystone=	C2000=*For technical support on*	DaVinci=*For technical support on*	MSP430=*For technical support on*	OMAP35x=*For technical support on*	OMAPL1=*For technical support on*	MAVRK=*For technical support on*	*For technical s please post you questions at*

- For technical support on MultiCore devices, please post your questions in the C6000 MultiCore Forum
- For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum

Please post only comments related to the article **NDK Static Network Buffers** here.

- For technical support on MultiCore devices, please post your questions in the C6000 MultiCore Forum
- For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum

Please post only comments related to the article **NDK Static Network Buffers** here.

*the C2000 please post your questions on The C2000 Forum. Please post only comments about the article **NDK Static Network Buffers** here.*

*DaVincoplease post your questions on The DaVinci Forum. Please post only comments about the article **NDK Static Network Buffers** here.*

*MSP430 please post your questions on The MSP430 Forum. Please post only comments about the article **NDK Static Network Buffers** here.*

*OMAP please post your questions on The OMAP Forum. Please post only comments about the article **NDK Static Network Buffers** here.*

*OMAP please post your questions on The OMAP Forum. Please post only comments about the article **NDK Static Network Buffers** here.*

*MAVRK please post your questions on The MAVRK Toolbox Forum. Please post only comments about the article **NDK Static Network Buffers** here.*

*http://e2e.ti.cor Please post on comments abo article **NDK Sta Network Buffe** here.*

}}

Links

Amplifiers & Linear
Audio
Broadband RF/IF & Digital Radio
Clocks & Timers
Data Converters

DLP & MEMS
High-Reliability
Interface
Logic
Power Management

Processors

- ARM Processors
- Digital Signal Processors (DSP)
- Microcontrollers (MCU)
- OMAP Applications Processors

Switches & Multiplexers
Temperature Sensors & Control ICs
Wireless Connectivity

Retrieved from "https://processors.wiki.ti.com/index.php?title=NDK_Static_Network_Buffers&oldid=182538"

This page was last edited on 28 July 2014, at 06:41.