Приложения

Приложение А (Листинг DMA для современных устройств)

```
1.#include <linux/module.h>                          //Динамическая загрузка
   модулей в ядро.
2.#include <linux/pci.h>                             //определение PCI и
   прототипы функций
3.#include <linux/slab.h>                            //Распределение памяти
4.#include <linux/dma-mapping.h>                    //Список возможных атрибутов,
   связанных с отображением DMA.
5.#include <linux/dmapool.h>
6.#include "out.c"
7.MODULE_LICENSE( "GPL" );
8.#define pool_size 1024
9.#define pool_align 8
10.
11.     /* обмен данными будет осуществляться в обоих направлениях */
12.     static int direction = PCI_DMA_BIDIRECTIONAL;
13.     // int direction = PCI_DMA_TODEVICE ;       //Обмен данными в
   направлении устройства
14.     // int direction = PCI_DMA_FROMDEVICE ;     //Обмен данными в
   направлении от устройства
15.     //int direction = PCI_DMA_NONE;            //Блокировка обмена
   данными
16.
17.     static int __init my_init( void )
18.     {
19.         char *kbuf;                             //буфер DMA
20.
21.         dma_addr_t handle;                      /* Dma_addr_t может
   содержать любой действительный адрес DMA или шины для платформы. Оно
   может
22.                                                  * передаваться
   устройству для использования в качестве источника или цели DMA. Это
   характерно для
23.                                                  * данное устройство и
   может быть передано между физическим адресом ЦП и адресным пространством
   шины
24.
```

```
25.                                                     dma-
   mapping.h :#define DMA_MAPPING_ERROR        (~(dma_addr_t)0)
26.                                                 */
27.
28.         size_t size = ( 10 * PAGE_SIZE );       //Размер страницы
29.         struct dma_pool *mypool;                //devic.h :
   dma_pools:  Dma pools (if dma'ble device).
30.
31.         /* использование метода dma_alloc_coherent */
32.         kbuf = dma_alloc_coherent( NULL, size, &handle, GFP_KERNEL );
33.         output( kbuf, handle, size, "This is the dma_alloc_coherent()
   string" );
34.         dma_free_coherent( NULL, size, kbuf, handle );
35.
36.         /* использование метода dma_map/unmap_single */
37.         kbuf = kmalloc( size,
   GFP_KERNEL );                                   // GFP_KERNEL (__GFP__WAIT
   | __GFP__IO | __GFP__FS) - выделение производится от имени процесса,
38.
       //который выполняет системный запрос в пространстве ядра — такой
   запрос может быть временно переводиться
39.
       //в пассивное состояние (блокирован).
40.
       //slab.h
41.         handle = dma_map_single( NULL, kbuf, size, direction );
42.         output( kbuf, handle, size, "This is the dma_map_single()
   string" );
43.         dma_unmap_single( NULL, handle, size, direction );
44.         kfree( kbuf );
45.
46.         /* использование метода dma_pool */
47.         mypool = dma_pool_create( "mypool", NULL, pool_size, pool_align,
   0 );
48.         kbuf = dma_pool_alloc( mypool, GFP_KERNEL, &handle );
49.         output( kbuf, handle, size, "This is the dma_pool_alloc()
   string" );
50.         dma_pool_free( mypool, kbuf, handle );
51.         dma_pool_destroy( mypool );
52.      return -1;
```

```
53.     }
```

Приложение Б (Листинг DMA для устаревших устройств)

```
1.#include <linux/module.h>
2.#include <linux/pci.h>
3.#include <linux/slab.h>
4.MODULE_LICENSE( "GPL" );
5.
6.#include "out.c"
7.
8./* обмен данными будет осуществляться в обоих направлениях */
9.static int direction = PCI_DMA_BIDIRECTIONAL;
10.     // int direction = PCI_DMA_TODEVICE ;
11.     // int direction = PCI_DMA_FROMDEVICE ;
12.     //int direction = PCI_DMA_NONE;
13.
14.
15.     static int __init my_init( void ) {
16.      char *kbuf;
17.      dma_addr_t handle;
18.      size_t size = ( 10 * PAGE_SIZE );
19.      /* использование метода pci_alloc_consistent */
20.      kbuf = pci_alloc_consistent( NULL, size, &handle );
21.      output( kbuf, handle, size, "This is the pci_alloc_consistent()
   string" );
22.      pci_free_consistent( NULL, size, kbuf, handle );
23.      /* использование метода pci_map/unmap_single */
24.      kbuf = kmalloc( size, GFP_KERNEL );
25.      handle = pci_map_single( NULL, kbuf, size, direction );
26.      output( kbuf, handle, size, "This is the pci_map_single()
   string" );
27.      pci_unmap_single( NULL, handle, size, direction );
28.      kfree( kbuf );
29.      return -1;
30.     }
```

Приложение В (Листинг: Общая часть тестов)

```
1.static int __init my_init( void );
2.module_init( my_init );
3.
4.MODULE_AUTHOR( "Sergey Mirzoyan" );
5.
6.#define MARK "=> "
7.
8.static void output( char *kbuf, dma_addr_t handle, size_t size, char
   *string ) {
9.   unsigned long diff;
10.        diff = (unsigned long)kbuf - handle;
11.        printk( KERN_INFO MARK "kbuf=%12p, handle=%12p, size = %d\n",
12.                kbuf, (void*)(unsigned long)handle, (int)size );
13.        printk( KERN_INFO MARK "(kbuf-handle)= %12p, %12lu,
   PAGE_OFFSET=%12lu, compare=%lu\n",
14.                (void*)diff, diff, PAGE_OFFSET, diff - PAGE_OFFSET );
15.        strcpy( kbuf, string );
16.        printk( KERN_INFO MARK "string written was, %s\n", kbuf );
17.     }
```