

UDI Tutorial & Driver Walk-Through

Part 1

<http://www.sco.com/forum1999/conference/developfast/f9>

Kurt Gollhardt
SCO Core OS Architect
E-mail: kdg@sco.com



Overview

- **Key Concepts**
- **Description of Sample Driver**
- **Walk-Through**
- **Additional Examples**
- **Hints & Tips**



Key Concepts

- **Static Driver Properties**
- **Initialization**
- **Instance Independence**
- **Regions, Channels & Control Blocks**
- **Channel Operations**
- **Programmed I/O**



Driver Package

- **Driver writer must supply:**
 - Driver source code and/or binary
 - Exported header files (e.g. for custom ops)
 - Static driver properties file (udiprops.txt)
 - » Attached to driver binary for binary distribution
- **Make package with `udimkpkg` command**



Static Driver Properties

udiprops.txt

- **Properties that are known and fixed in advance of compiling the driver**
- **Such as:**
 - driver name
 - supplier name
 - identifying attributes of supported devices
 - message strings



Initialization

udi_init_info

- **Fixed properties of driver code**
- **Such as:**
 - Entry-point function pointers
 - Data structure sizes
 - » Region data, channel context, scratch space
 - Lists of channel operation & control block types



Instance Independence

- **Each H/W device instance (“unit”) identified separately**
- **Driver maintains per-instance state**
 - No writeable memory shared between instances
- **Instances are dynamic (hot plug)**



Regions

- **One or more regions per driver instance**
 - Driver's udi_init_info determines # of regions
 - No writeable memory shared between regions
- **All region data implicitly synchronized**
 - Global and static data is read-only



Channels

- **Channels connect regions**
- **Channel operations pass data over channels**
- **Not all types of objects are transferable between regions**



Control Blocks

- **Maintain per-request state**
- **Required for channel ops & asynchronous service calls**
- **Allocate with `udi_cb_alloc`**
- **Free with `udi_cb_free`**



Channel Operations

- **Function with control block & metalanguage-specific parameters**
- **Target end invoked asynchronously**
- **Target channel stored in control block**
 - Initialized by `udi_cb_alloc`
 - Points to channel over which a channel op is received



Programmed I/O

- **One or more register sets per device**
 - Contiguous block of registers or memory
 - For PCI, register set = BAR
- **All PIO addresses relative to register set**
- **Endianness conversion automatic**
 - Driver simply indicates device endianness
- **Both strong & weak ordering supported**



Description of Sample Driver

- **Simple driver for PC-AT CMOS RAM**
- **Source includes some interrupt handling code even though not used by device**
- **Driver is 95% complete**
 - No constraints propagation
 - No abort handling
 - No tracing/logging



Walk-Through

Listing

- **Complete source code listing in handout**
 - “UDI CMOS Sample Driver Listing”
 - http://www.sco.com/forum1999/conference/developfast/F9/udi_cmos_sample.html
- **Selected excerpts in slides**
 - Excerpts include line numbers
- **Included in SCO UDI Development Kit**
 - <http://www.sco.com/UDI/sco/udidk>



Walk-Through

udiprops.txt line 3

- **First declaration must be properties_version**

```
3  properties_version 0x095
```

- **Comments begin with ‘#’**
- **Line continuation with ‘\’**



Walk-Through

udiprops.txt lines 9-17

- **Identify driver and supplier**

```
9    supplier 1
10   message 1 Project UDI
11   contact  2
12   message 2 http://www.sco.com/udi/participants.html
13   name     3
14   message 3 CMOS RAM Sample UDI Driver
15
16   shortname udi_cmos
17   release  2 alpha1.0rc3
```



Walk-Through

udiprops.txt lines 28-38

- Describe supported devices

```
28 device 10 2 bus_type string system
29 message 10 Motherboard CMOS RAM
30 config_choices 10 ioaddr1 ubit32 0x70 any \
    iolen1 ubit32 2 only
```

- Describe driver modules and regions

```
37 module udi_cmos
38 region 0
```



Walk-Through

udiprops.txt lines 44-57

- **Interface dependencies**

```
44  requires udi          0x095
45  requires udi_physio 0x095
46  requires udi_gio     0x095
47  requires udi_bridge 0x095
```

- **Metalinguage usage**

```
53  meta 1 udi_gio        # Generic I/O Metalinguage
54  meta 2 udi_bridge     # Bus Bridge Metalinguage

56  child_bind_ops 1 0 1  # GIO meta, region 0, ops_idx 1
57  parent_bind_ops 2 0 2 # Bridge meta, region 0, ops_idx 2
```



Walk-Through

udiprops.txt lines 63-64

- **Build rules**

```
63  compile_option -DCMOS_GIO_META=1 -DCMOS_BRIDGE_META=2
64  source_files udi_cmos.c
```



Walk-Through

udi_cmos.c lines 16-20

- **Versions and header files**

```
16  #define UDI_VERSION 0x095
17  #define UDI_PHYSIO_VERSION 0x095
18
19  #include <udi.h>
20  #include <udi_physio.h>
```



Walk-Through

udi_cmos.c lines 25-32

- **Region data structure**

```
25     typedef struct {
26         /* init_context is filled in by
           the environment: */
27         udi_init_context_t      init_context;
28         udi_bus_bind_cb_t       *bus_bind_cb;
29         /* PIO trans handles for reading
           and writing, respectively: */
30         udi_pio_handle_t        trans_read;
31         udi_pio_handle_t        trans_write;
32     } cmos_region_data_t;
```



Walk-Through

udi_cmos.c lines 37-46

- **Scratch structures and offsets**

```
37  typedef struct {
38      udi_ubit8_t    addr;
39      udi_ubit8_t    count;
40  } cmos_gio_xfer_scratch_t;

45  #define SCRATCH_ADDR \
      offsetof(cmos_gio_xfer_scratch_t, addr)
46  #define SCRATCH_COUNT \
      offsetof(cmos_gio_xfer_scratch_t, count)

64  #define GIO_XFER_SCRATCH \
      sizeof(cmos_gio_xfer_scratch_t)
```



Walk-Through

udi_cmos.c lines 51-76

- **Control block indexes**

51	<code>#define GIO_BIND_CB_IDX</code>	1
52	<code>#define GIO_XFER_CB_IDX</code>	2
53	<code>#define GIO_EVENT_CB_IDX</code>	3
54	<code>#define BUS_BIND_CB_IDX</code>	4

- **Channel ops indexes**

75	<code>#define GIO_OPS_IDX</code>	1
76	<code>#define BUS_DEVICE_OPS_IDX</code>	2



Walk-Through

udi_cmos.c lines 87-144

- **Management metalanguage ops vector**

```
87  static const udi_mgmt_ops_t cmos_mgmt_ops = {
88      udi_static_usage,
89      udi_enumerate_no_children,
90      cmos_devmgmt_req,
91      cmos_final_cleanup_req
92  };
```

- **Similarly for GIO & Bridge metas**

- Two bridge ops vectors if using interrupts



Walk-Through

udi_cmos.c lines 149-162

- **Defines for PIO**

```
149  #define CMOS_REGSET      1
      /* first (and only) register set */
150  #define CMOS_BASE       0
      /* base address within this regset */
151  #define CMOS_LENGTH     2
      /* two bytes worth of registers */
152  #define CMOS_PACE       5
      /* wait 5 microseconds btw accesses */

161  #define CMOS_ADDR       0
162  #define CMOS_DATA       1
```



Walk-Through

udi_cmos.c lines 167-168

- **Other defines for CMOS device**

```
167  #define CMOS_DEVSZ      0x40
      /* # data bytes supported by device */
168  #define CMOS_RDONLY_SZ  0x0E
      /* first 14 bytes are read-only */
```



Walk-Through

udi_cmos.c lines 173-236

- **PIO transaction lists**

```
173  static const udi_pio_trans_t cmos_trans_read[] = {  
    ...  
204  };  
205  static const udi_pio_trans_t cmos_trans_write[] = {  
    ...  
236  };
```

- **Transaction list details covered later in Part 2**

