Writing Flexible Device Drivers for DSP/BIOS II

Jack Greenbaum

TI Santa Barbara

greenbaum@ti.com

What is a Device Driver?

- Software that isolates an application from I/O hardware

- What's the big deal?
  - Applicability
  - Efficiency
  - Easy of implementation

Overview

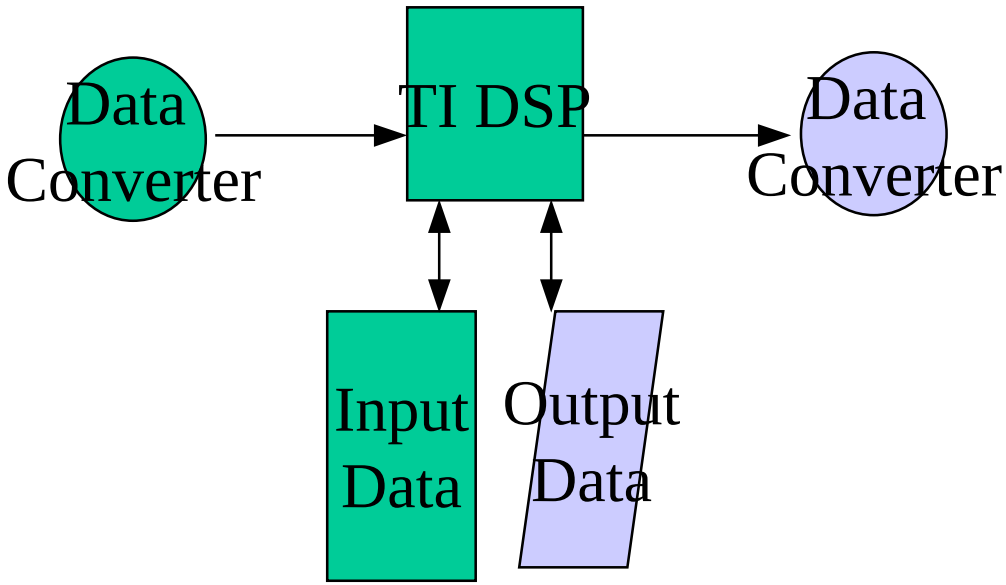
- System model
- Constraints
- LIO API
- Break
- Implementing LIO
- Bridging to DSP/BIOS PIP and SIO
- Multi-channel
- Overhead

Who writes drivers?

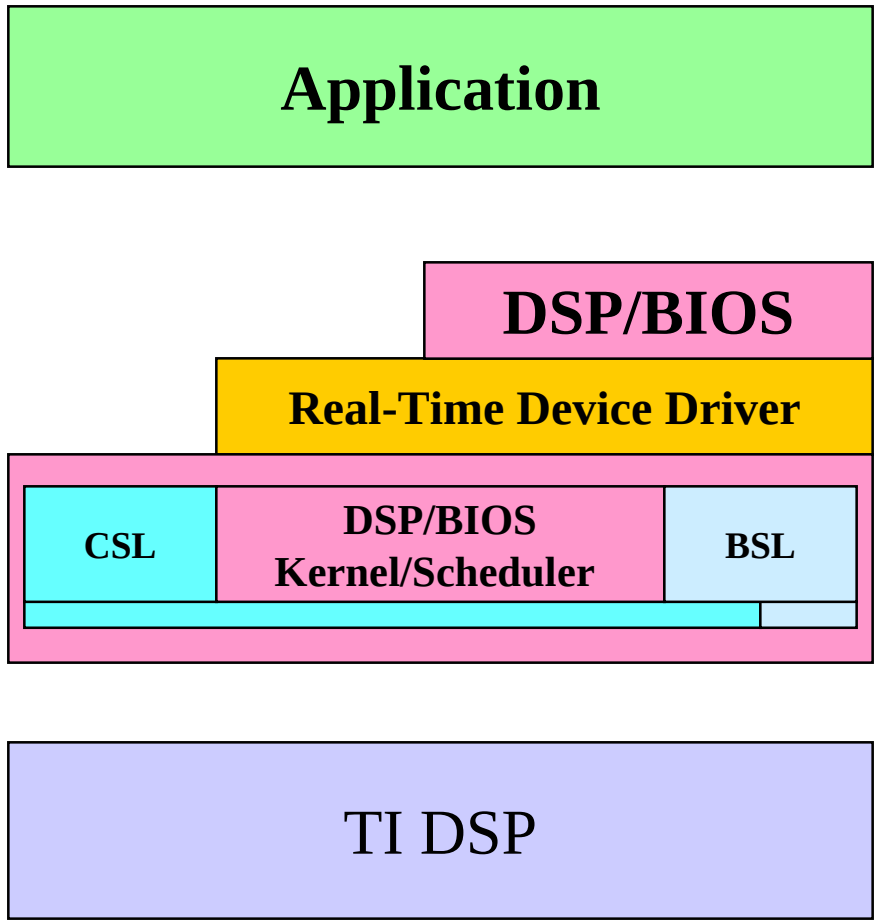
- TI
- End user hardware groups
- 3rd party board vendors

Model System

- Frame-based Streaming I/O



Model Software



Device Constraints

- Expose HW features in a consistent manner
  - Converter parameters
  - Autobuffering/DMA
  - Companding

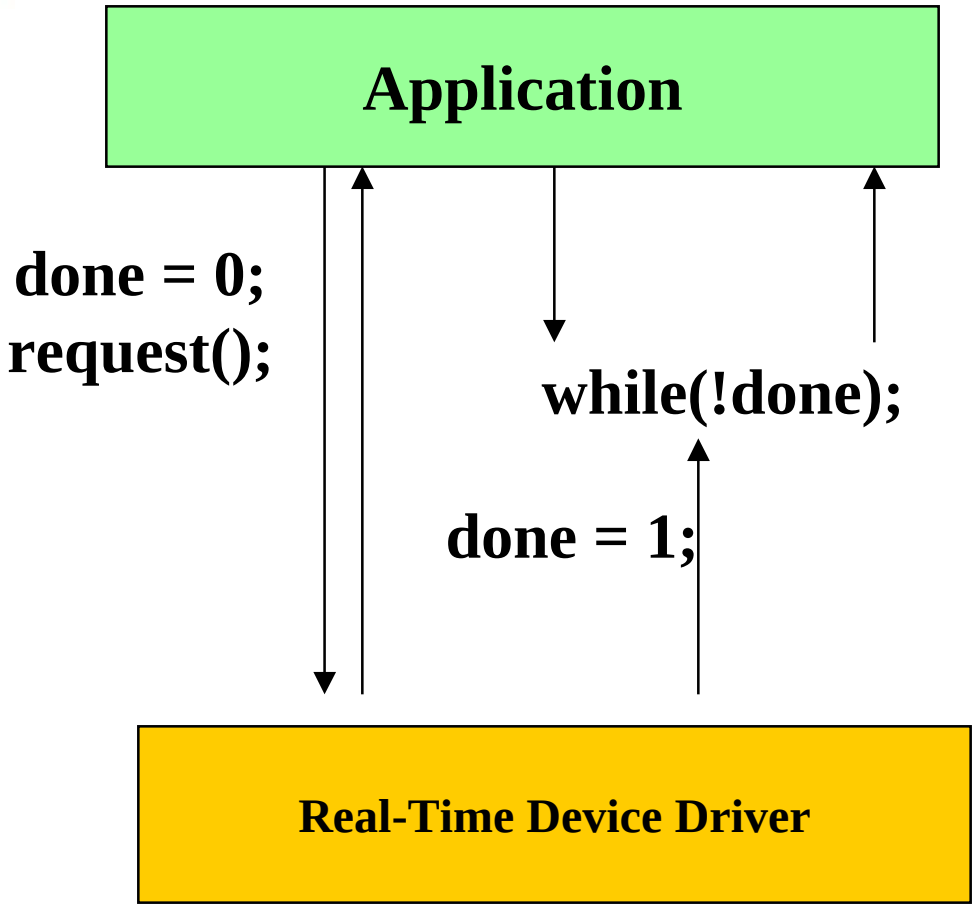
System Constraints

- Memory space and cycle count
- Resource Management
- Namespace Pollution

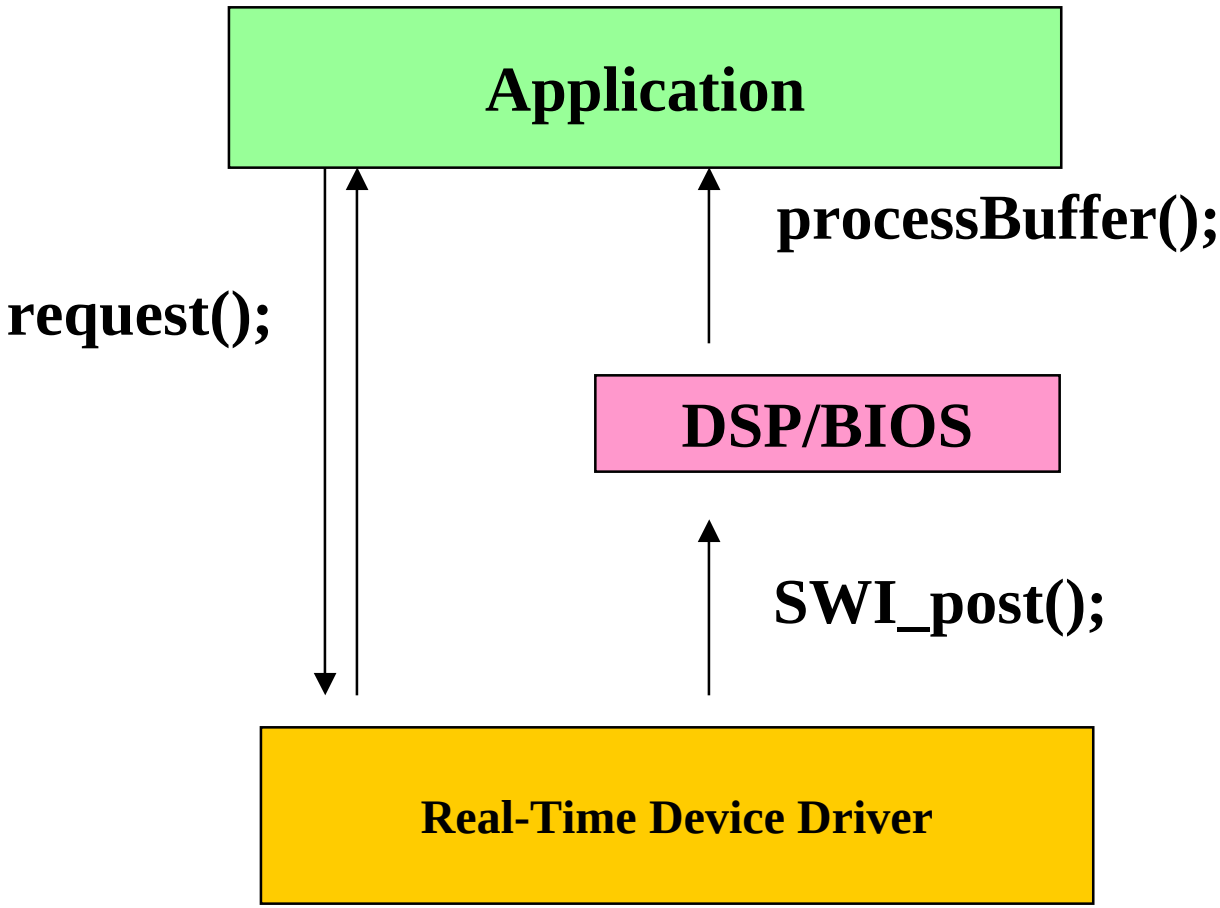
Application Constraints

- Buffer Management
- Signaling

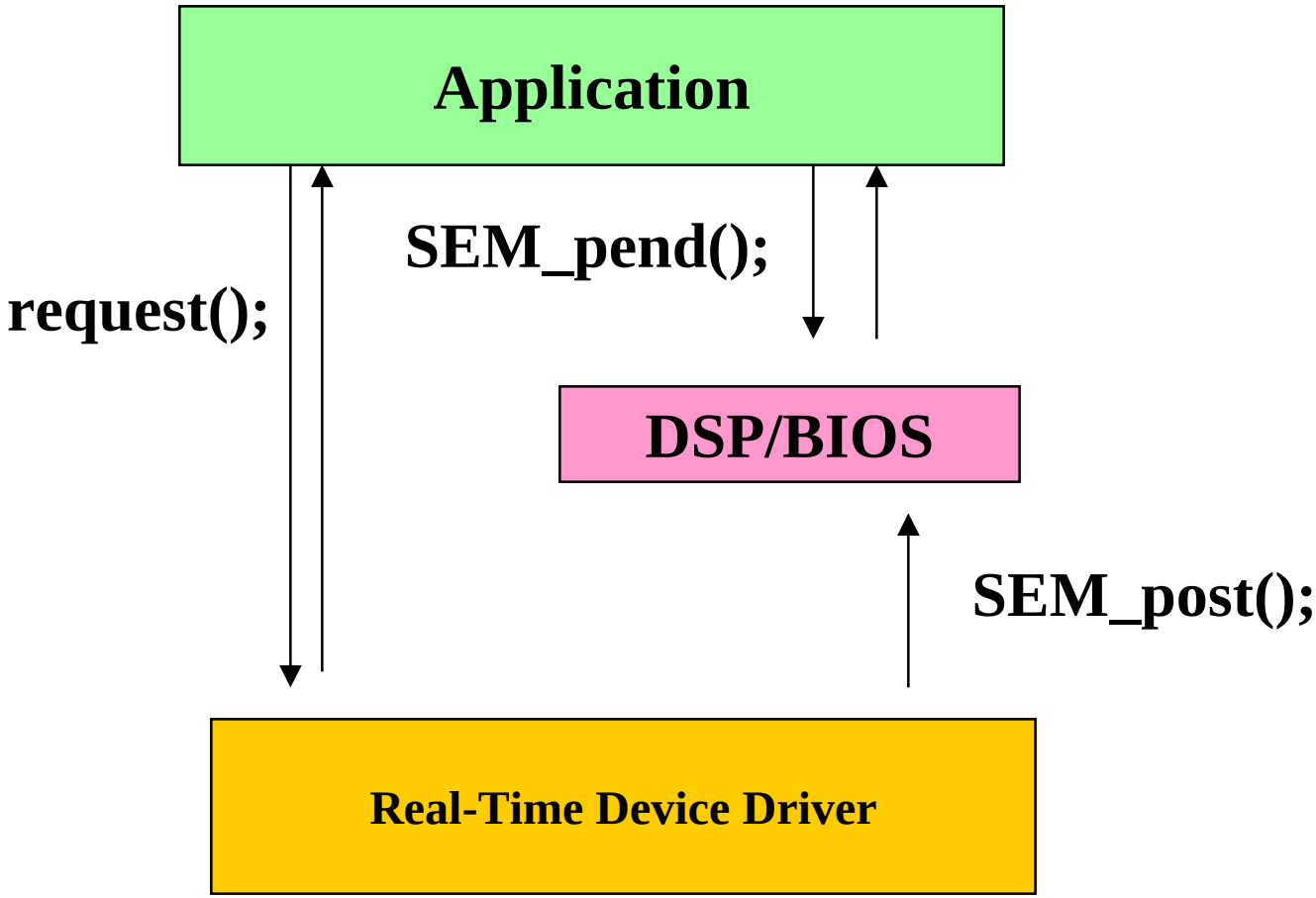
Simple Signaling



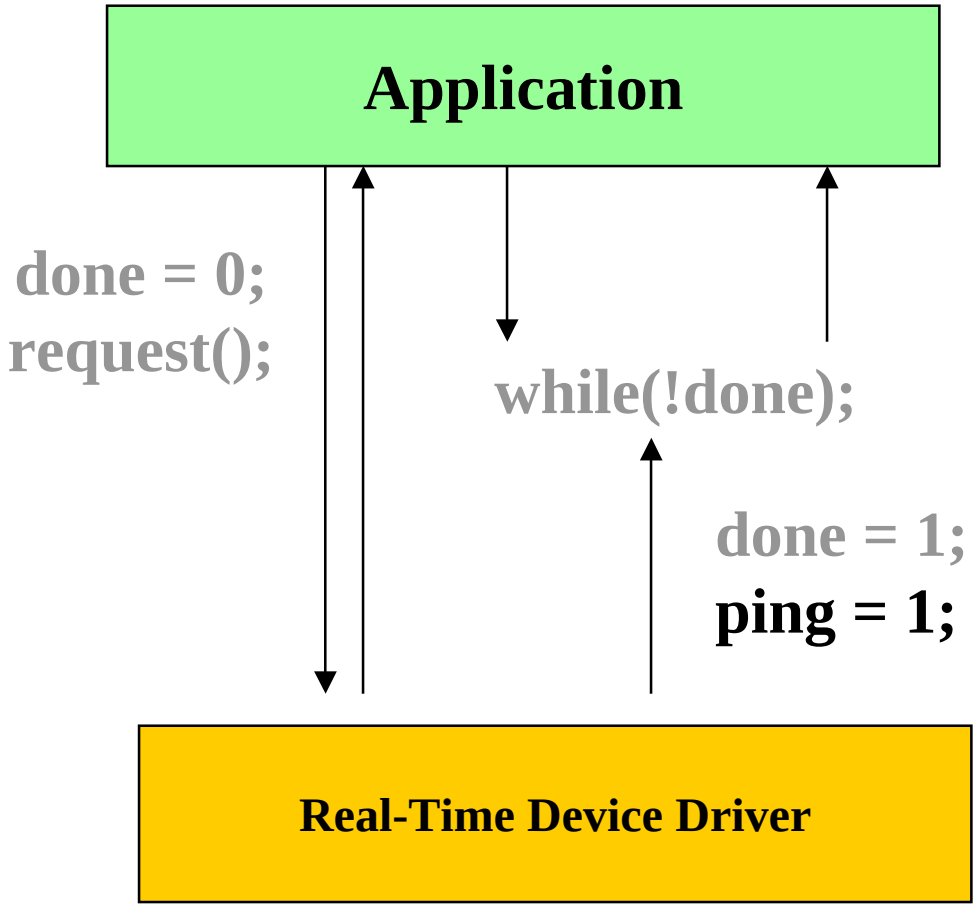
SWI Signaling



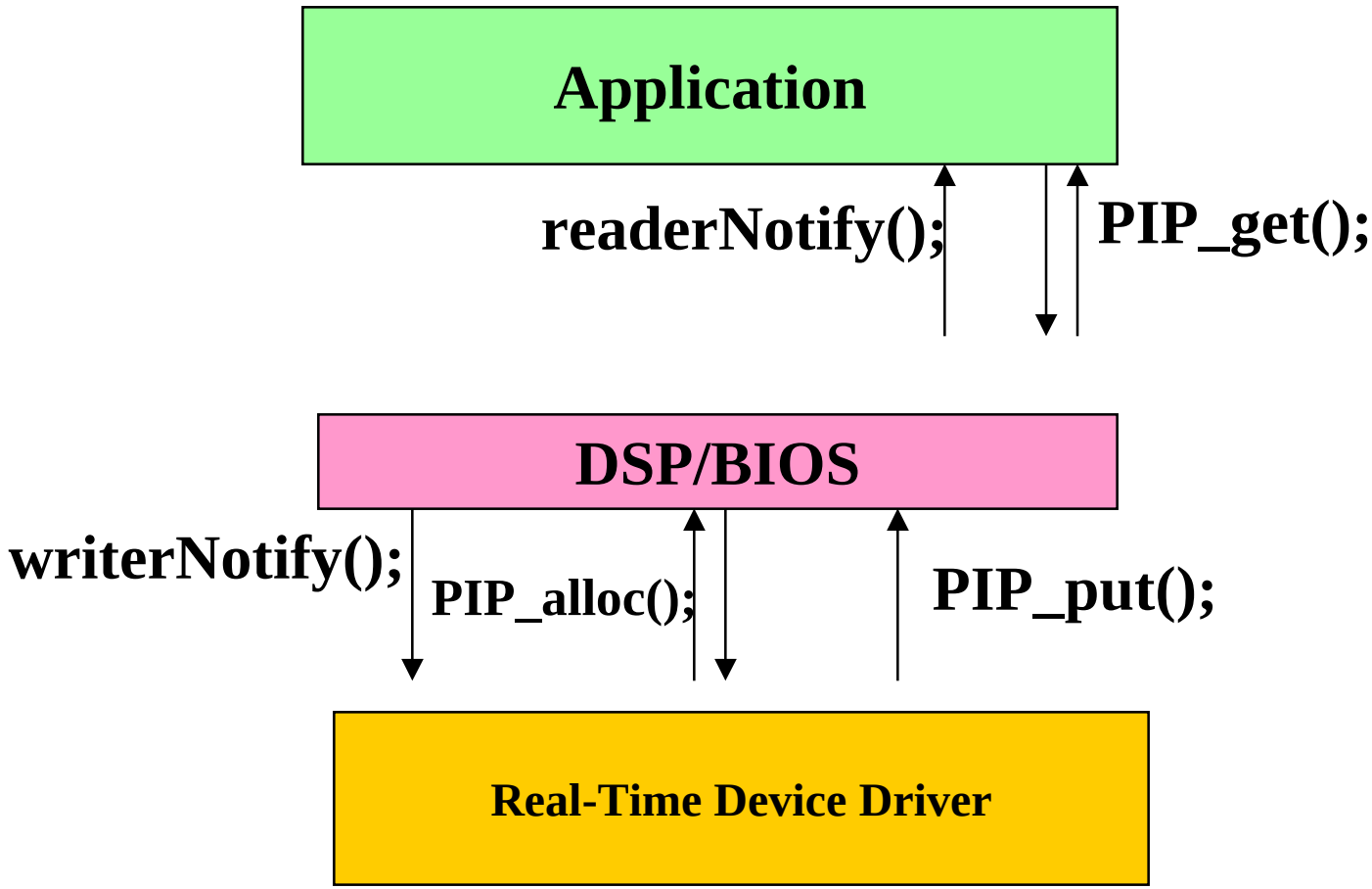
Semaphore Signaling



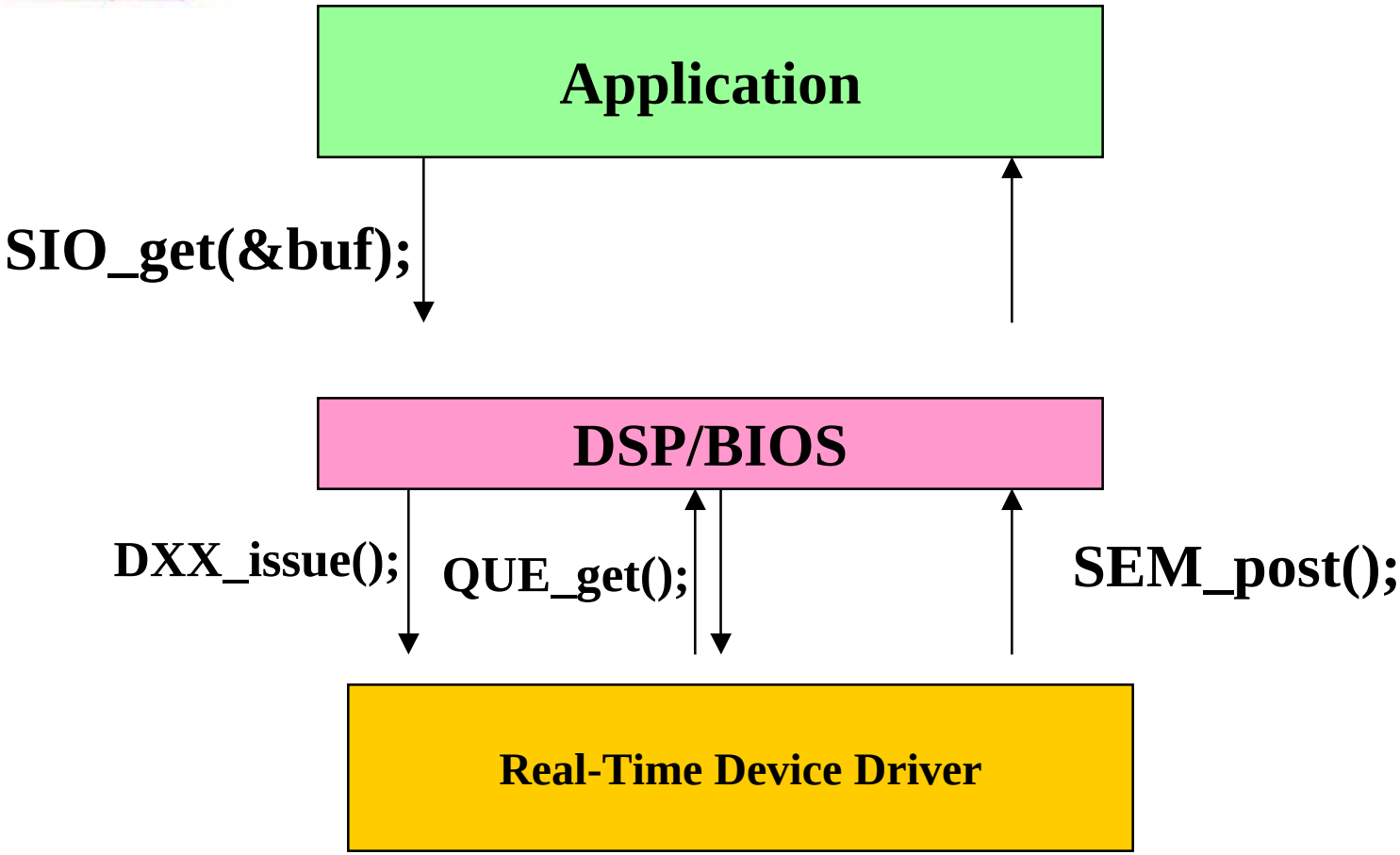
Simple Buffer Management



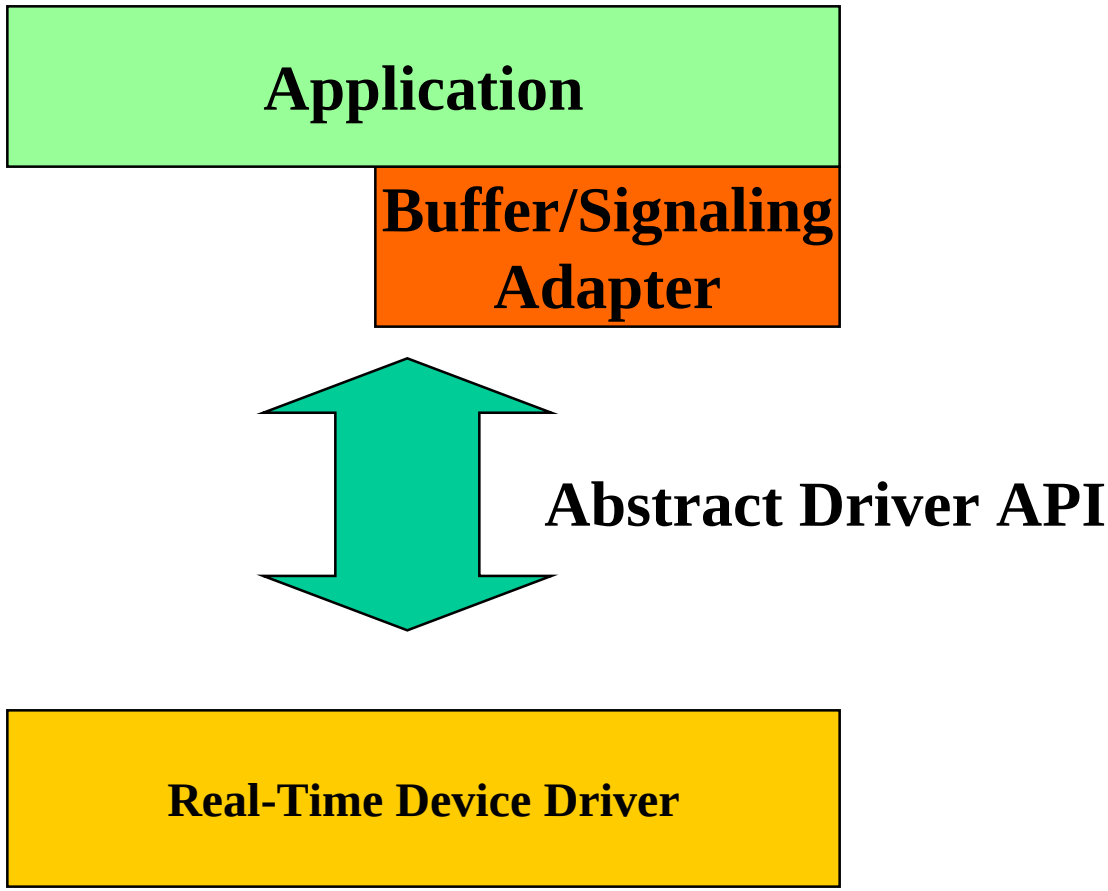
PIP Buffer Management



SIO Buffer Management



Abstract Driver API



Low-Level IO (LIO) API

- 10 functions
  - Control
  - Buffer management
  - Signaling

Control Functions

- init, close
  - Resource allocation
- start, stop
  - Interrupt control
- ctrl
  - Catch-all

Buffer Management

- putBuf, getBuf
  - Hardware queue access
- isFull, isEmpty
  - State of hardware queue

Signaling

- setCallback
  - driver to application signaling

Example: Raw

```
#define RCV_CHAN 0
#define XMT_CHAN 1

LIO_Fxns *driver = LIO_DSK5402_DMA_Fxns;

Void main()
{
      if (!driver->init(RCV_CHAN, NULL)
            || !driver->init(XMT_CHAN, NULL)) {
            LOG_printf(&trace, "init failed");
      } else {
            driver->setCallback(RCV_CHAN, callback, 0);
            driver->setCallback(XMT_CHAN, callback, 0);
      }
}
```

Example: Raw

```
#define BUFCNT 160
#define BUFSIZ BUFCNT*sizeof(short);

unsigned short buf0[BUFCNT], buf1[BUFCNT];
unsigned short buf2[BUFCNT], buf3[BUFCNT];

driver->putBuf(RCV_CHAN, buf0, BUFSIZ);

doubleBuffered
      = driver->putBuf(RCV_CHAN, buf1, BUFSIZ);

if (!doubleBuffered) {
      LOG_printf(&trace,
      "Driver has no hardware queue.");
}
```

Example: Raw

```
while(1) {
        SEM_pend(&testSem, SYS_FOREVER);

        for (i = 0; i < BUFCNT; i++) {
                buf2[i] = buf0[i] & 0xfffe;
        }

        driver->putBuf(XMT_CHAN, buf2, BUFSIZ);

        driver->putBuf(RCV_CHAN, buf0, BUFSIZ);

        if (doubleBuffered) {
                SEM_pend(&testSem, SYS_FOREVER);
                …
}
```

Example: Raw

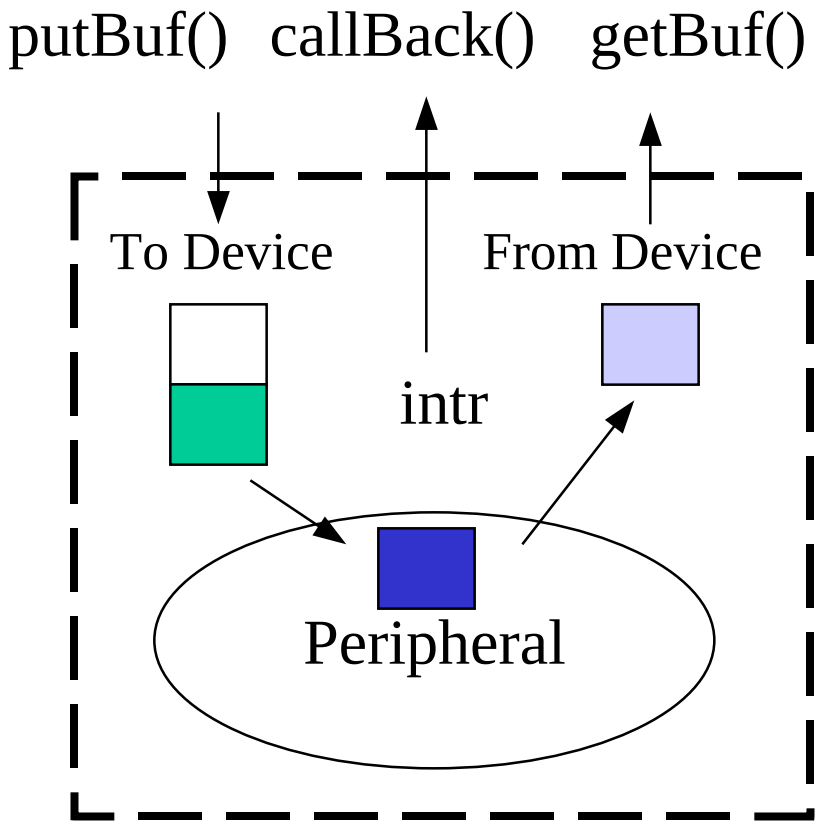
```
static Void callback(Uns chan, Arg ignored)
{
    Ptr buf = driver->getBuf(chan, 0, 0);

    SEM_post(&testSem);
}
```

Implementation Details

- General Approach: State Machine
- Specific examples:
  - Sample-by-Sample
  - 54x DMA
  - 6x11 EDMA

Driver State

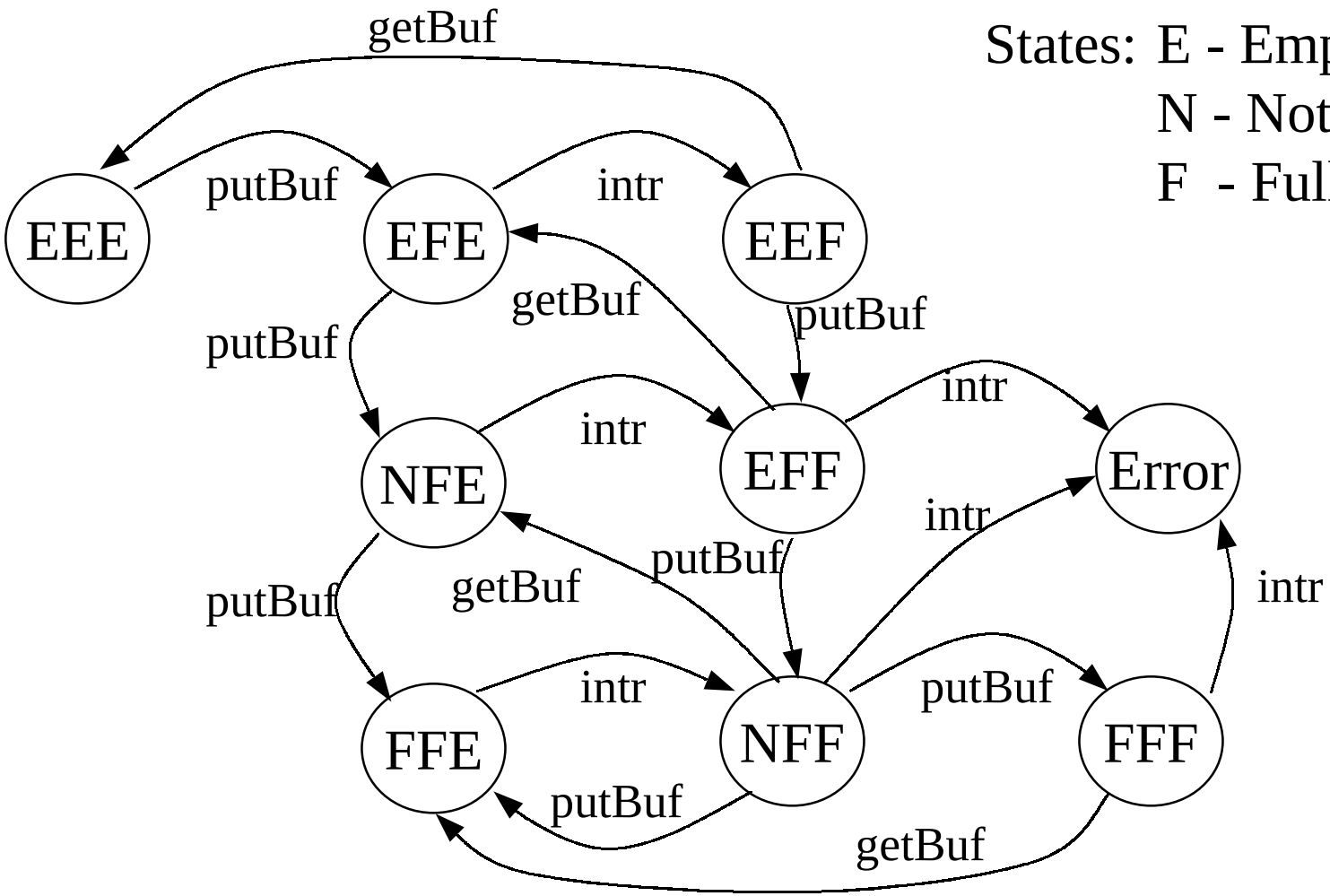
- Device queues
  - empty
  - not full
  - full
- Transitions taken on API calls or interrupt



Driver State

State Vector: To,Dev,From

States: E - Empty
N - Not Full
F - Full



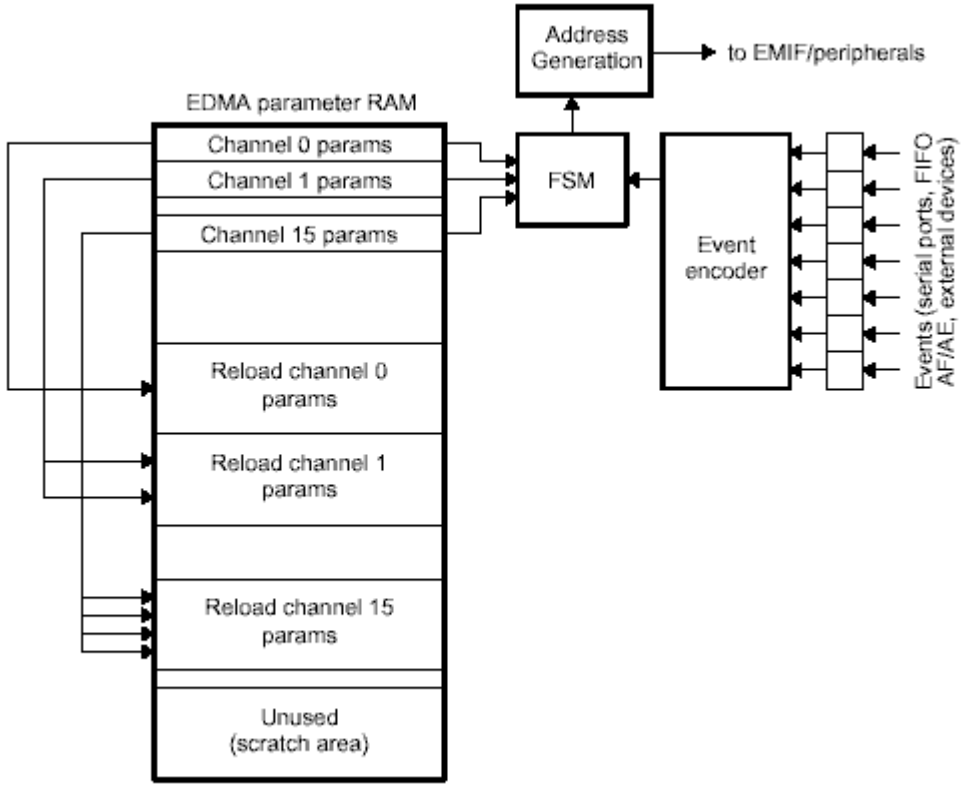
Sample-by-Sample

```
struct state {
 Ptr currentBuffer;
 Uns currentSize;
 Ptr currentPointer;
 Uns currentCount;
 Ptr lastBuffer;
 Uns lastSize;
}
```

DMA

```
struct state {
 Ptr currentBuffer;
 Uns currentSize;
/*
 Ptr currentPointer;
 Uns currentCount;
*/
 Ptr lastBuffer;
 Uns lastSize;
}
```

6x11 EDMA


Address Generation
to EMIF/peripherals
EDMA parameter RAM
Channel 0 params
Channel 1 params
Channel 15 params
Reload channel 0 params
Reload channel 1 params
Reload channel 15 params
Unused (scratch area)
FSM
Event encoder
Events (serial ports, FIFO AF/AE, external devices)

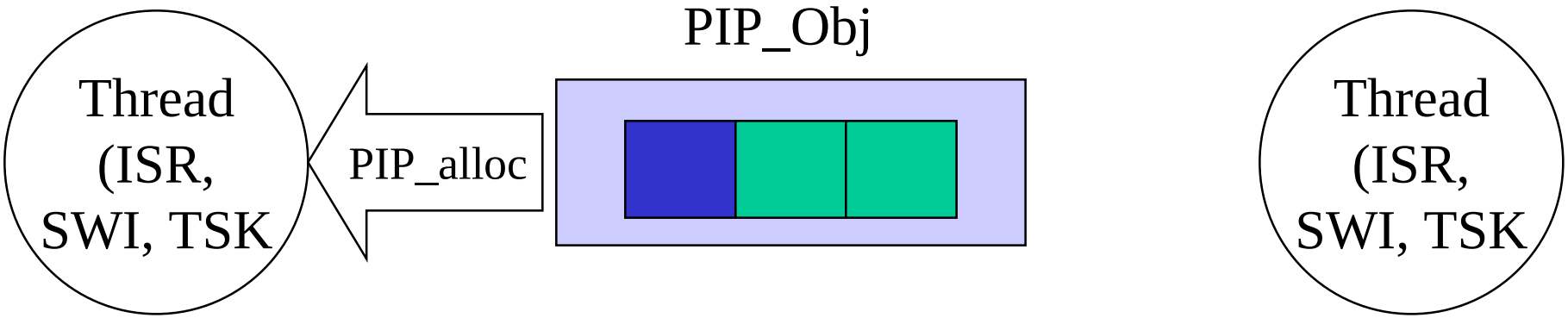
6211 EDMA State Structure

```
struct state {
 Ptr currentBuffer;
 Uns currentSize;
 Ptr linkAddr;
 Ptr lastBuffer;
 Uns lastSize;
}
```

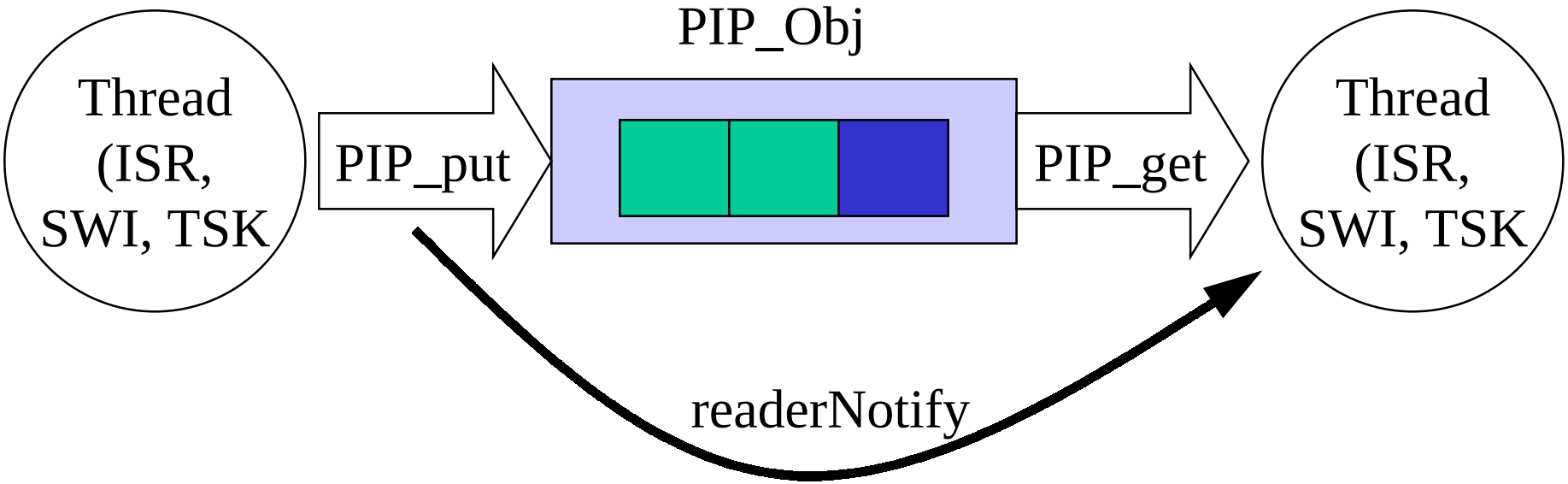
Adapters

- Raw
- DSP/BIOS I/O
  - PIP
  - SIO

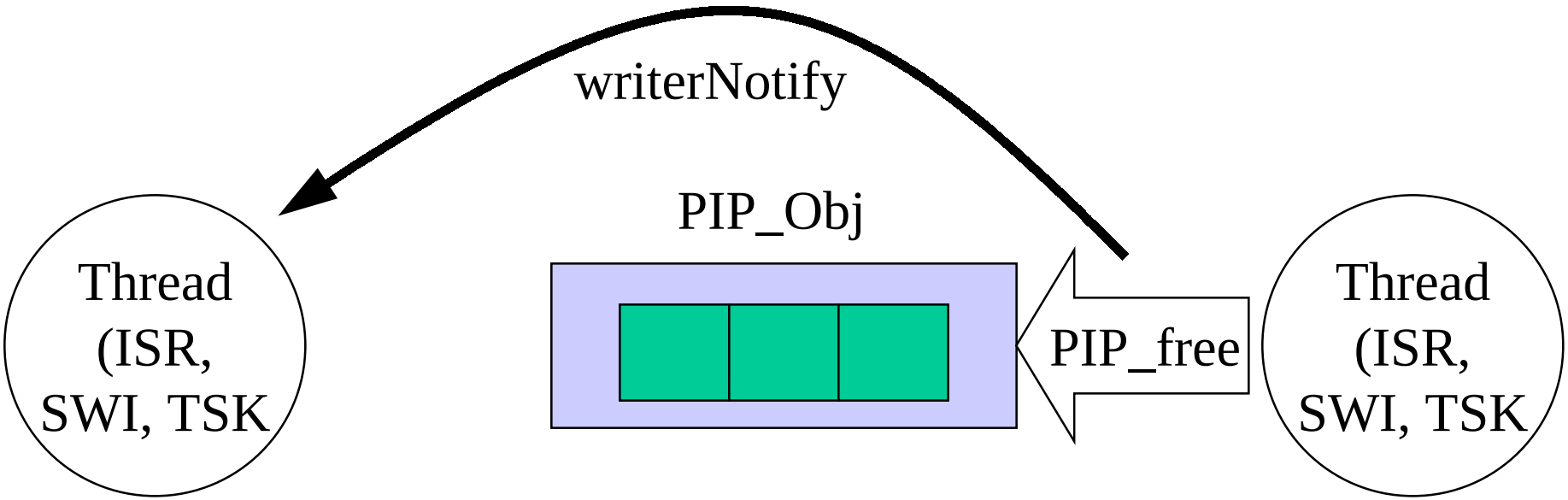
PIP



PIP


PIP_Obj
Thread (ISR, SWI, TSK
PIP_put
PIP_get
Thread (ISR, SWI, TSK
readerNotify

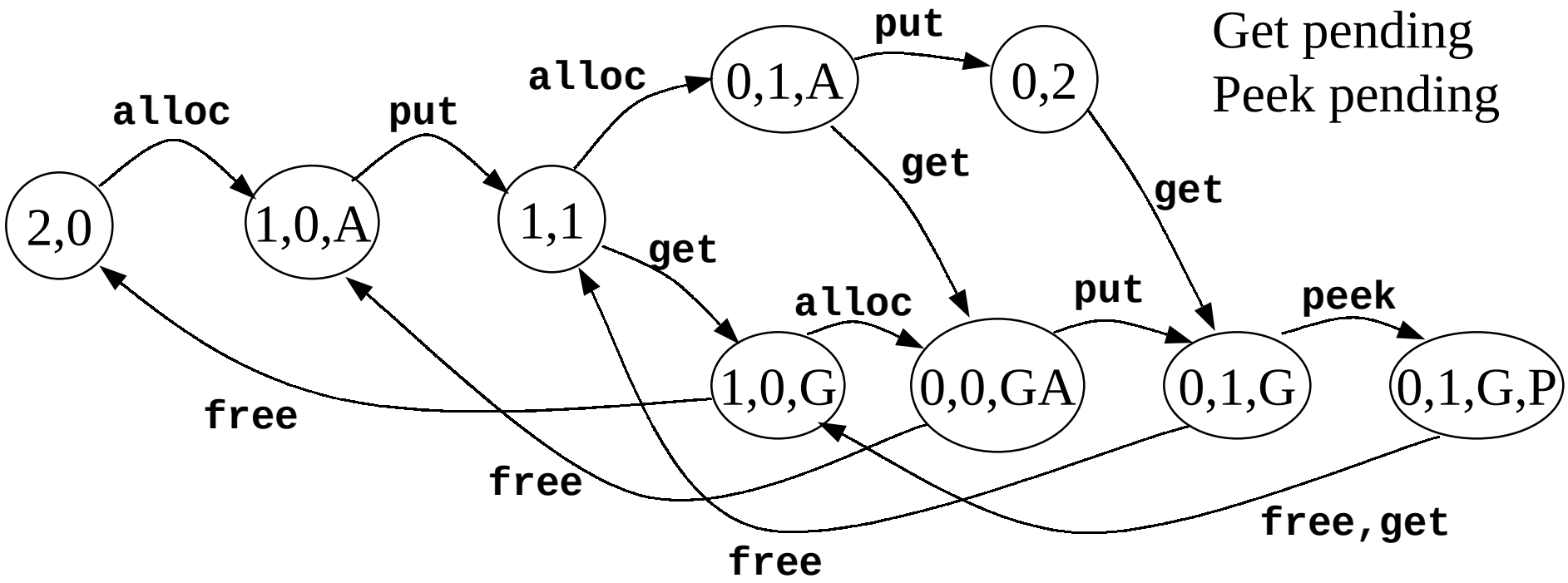
PIP


writerNotify
PIP_Obj
Thread
(ISR,
SWI, TSK
PIP_free
Thread
(ISR,
SWI, TSK

PIP Adapter

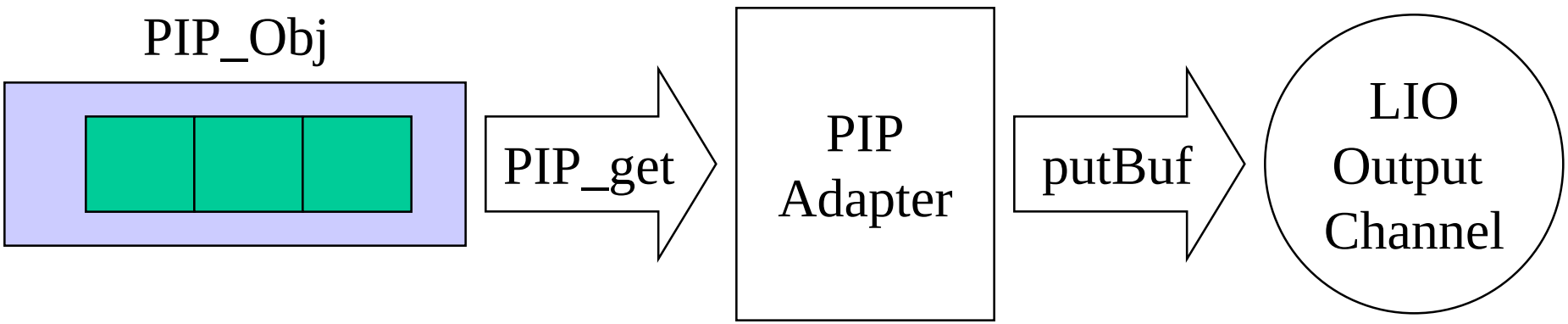
State Vector:

- Reader #Frames
- Writer #Frames
- Alloc pending
- Get pending
- Peek pending



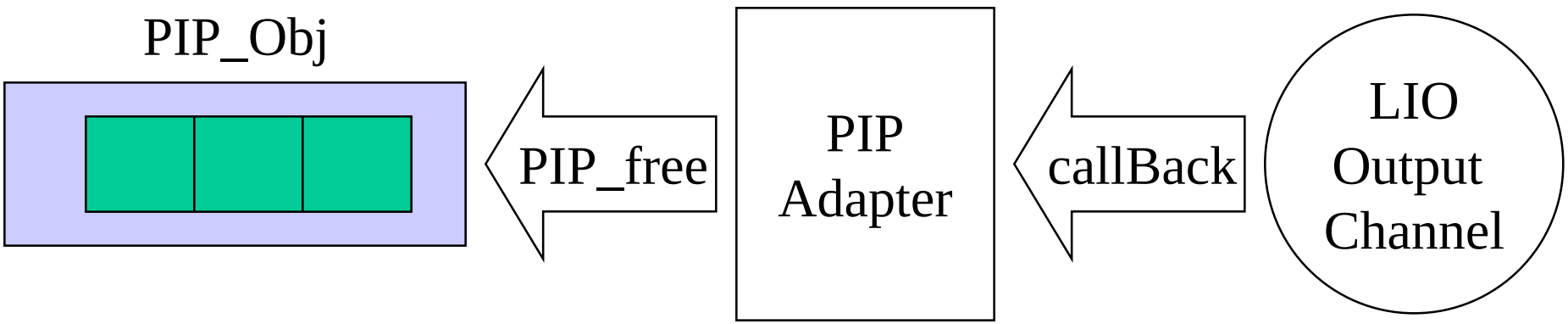
Output PIP

- Reader Notify
  - "prime"
    - PIP_get, PIP_peek
    - LIO_putBuf

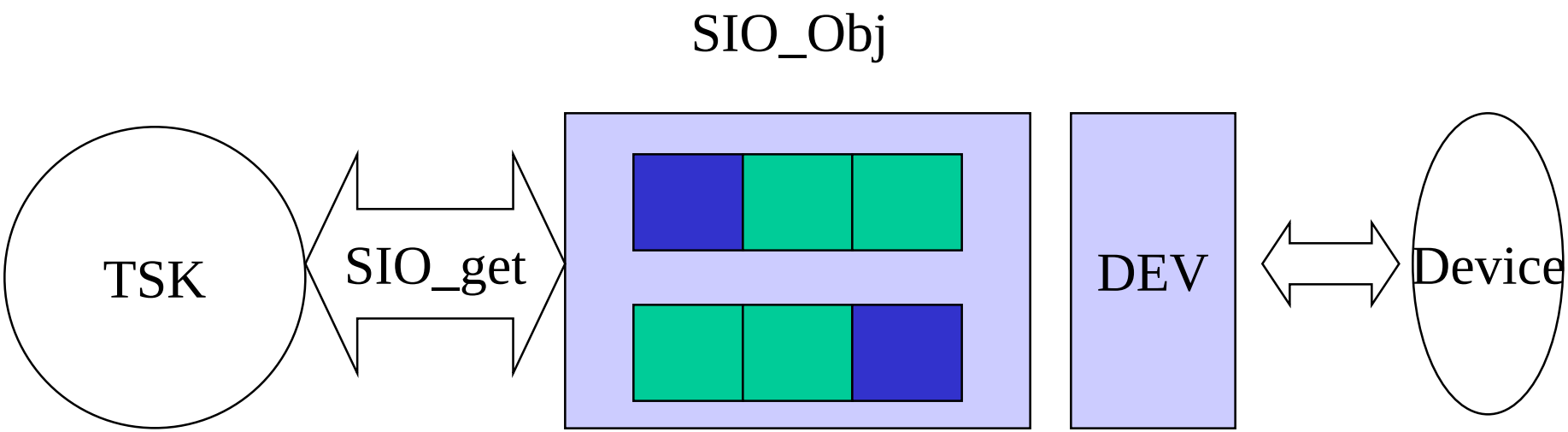


Output PIP

- LIO callback
  - “pump”
    - PIP_free



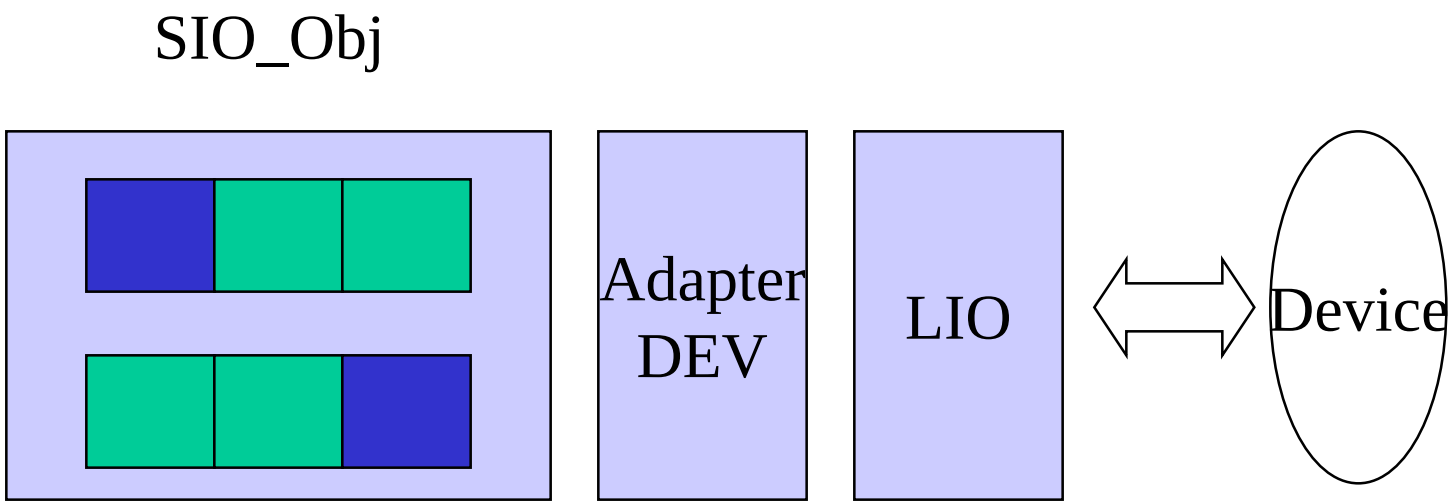
SIO



SIO

- Control
  - SIO_create/delete
  - SIO_cntrl
  - SIO_flush
- Buffer Management
  - SIO_issue/reclaim
- Signaling
  - SIO_get/put
  - SIO_select

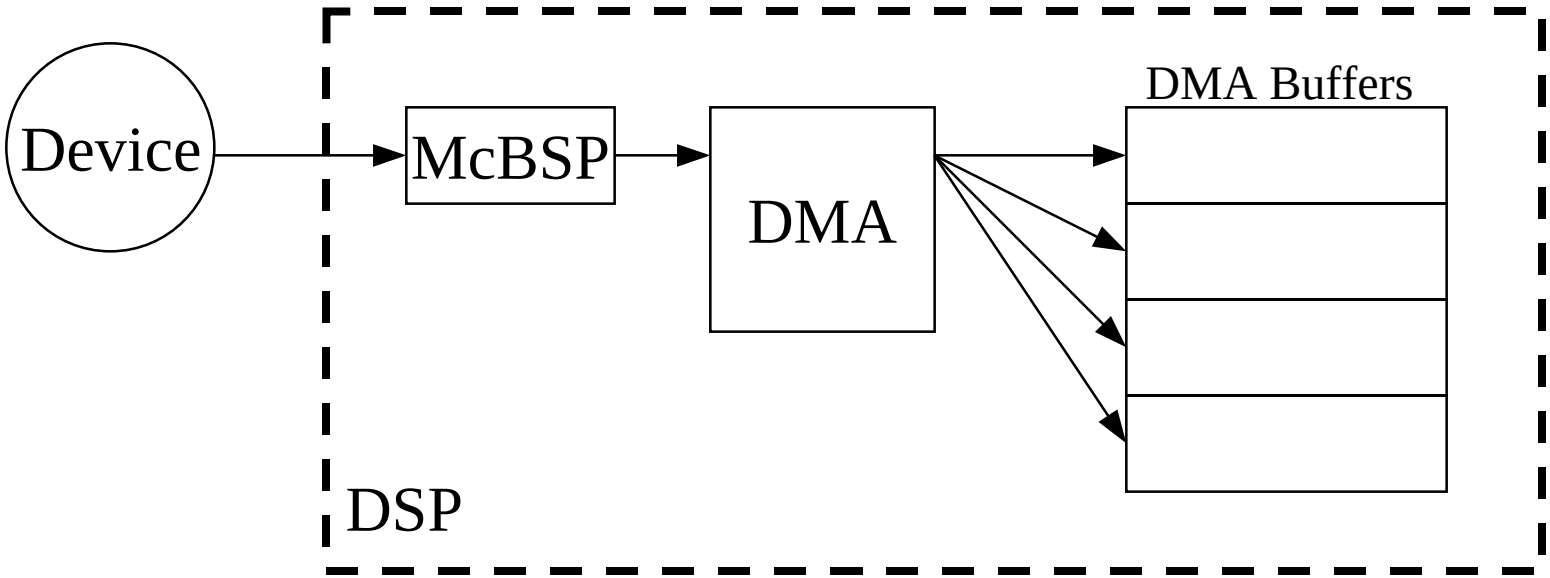
SIO



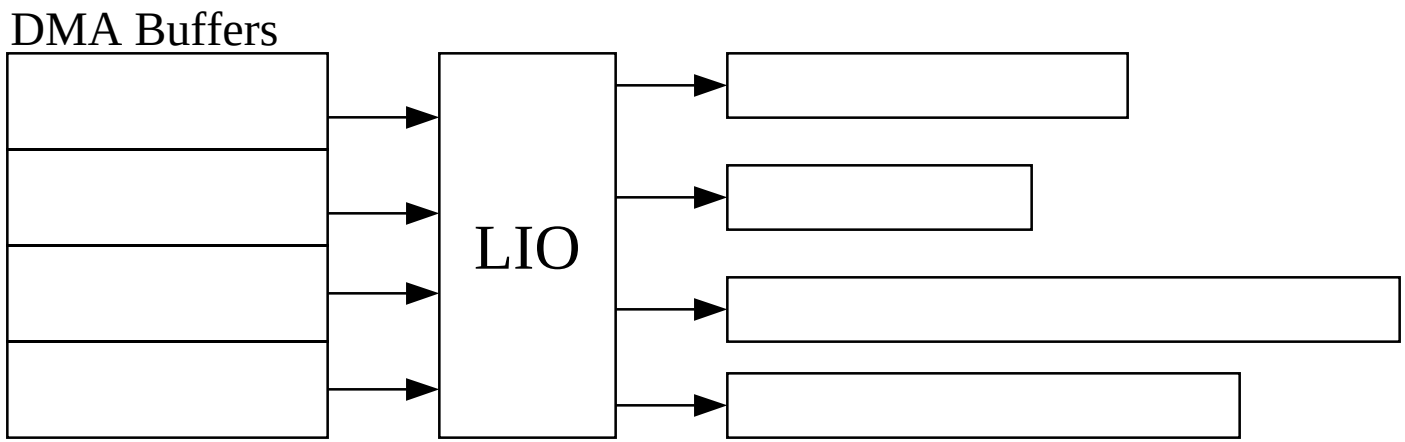
SIO DEV interface

- DEV functions
  - open, close, ctrl
  - issue, reclaim, idle
  - ready

Multi-channel



Multi-channel



Overhead

- Compared to …???

- 62xx
  - PIP LIO adapter: 0.25k words prog, 5 data
  - EDMA LIO driver: 1k words prog, 25 data
  - McBSP driver: 0.5k words prog, 40 data

More Information

- App note
- TISB Apps via Liz Keate (lkeate@ti.com)