Proposed Open Instruments (OI) API Specification

Aaron Cuevas Lopez Goncalo Lopes Jon Newman Jakob Voigts

June 1, 2016

Proposed License

MIT

Participating Projects and Organizations

- Bonsai
- Open Ephys
- RTXI

Potential Adopting Organizations

- Intan
- Spike Gadgets
- NeuroPixels
- NeuroSeeker
- Leaf Labs

Specification

NOTE: This interface specification is a work in progress and will change to support input from the community and to deal with the realities of implementation.

oiCreateContext

Create open instruments hardware context. A context is an opaque handle to a structure which specifies transport type (e.g. PCIe, UDP, etc.), requisite transport configuration, and a small finite state machine to manage port and stream connectivity.

```
int oiCreateContext(const char *spec, void **c)
```

Arguments

- `spec` URI context specification string
- `c` pointer to created context

Returns `int`

- Less than 0: `oiError`

Description

During successful context creation the following actions take place

1. Physical transport is defined (e.g. PCIe, socket, etc)
2. The context state machine enters the INITIALIZED state

Tentatively, context specification is provided via URI string. This may change.

oiDestroyContext

Terminate a open instruments context and frees bound resources.

```
int oiDestroyContext(oiContext c)
```

Arguments

- `c` context

Returns `int`

- Less than 0: `oiError`

Description

Context termination is performed in the following steps:

1. Any blocking operations will return immediately with error code TERMINATE
2. Attached port resources are released
3. The context state machine enters the UNINITIALIZED state

oiOpenPort

Open a physical port within an open instruments context.

```
int oiOpenPort(oiContext c, int port, int flags=OI_DEFAULT_FLAG)
```

Arguments

- `c` context
- `port` physical port number
- `flags` port specification flags

Returns `int`

- Less than 0: `oiError`

oiClosePort

Close a physical port within an open instruments context and free associated resources.

```
int oiClosePort(oiContext c, int port)
```

Arguments

- `c` context
- `port` physical port number

Returns `int`

- Less than 0: `oiError`

oiReadConfig

Read a configuration register from a device on a connected port.

```
int oiReadConfig(const oiContext c, int port, int key, int *value)
```

Arguments

- `c` context
- `port` physical port number
- `key` key of register to read
- `value` currently set register value

Returns `int`

- Less than 0: `oiError`

oiWriteConfig

Set a configuration register on a device on a connected port

```
int oiWriteConfig(const oiContext c, int port, int key, int value, int mask=0xFFFFFFFF)
```

Arguments

- `c` context
- `port` physical port number
- `key` key of register to write to
- `value` value to write to register
- `mask` bit mask applied to value before it is written

Returns `int`

- Less than 0: `oiError`

oiOpenStream

Open a data stream on a connected port. The context's state machine is updated to include the presence and direction specification of the stream.

```
int oiOpenStream(oiContext c, int port, int stream=0)
```

Arguments

- `c` context
- `port` physical port number
- `stream` stream number

Returns `int`

- Less than 0: `oiError`

oiCloseStream

Close data stream on a connected port

```
int oiCloseStream(oiContext c, int port)
```

Arguments

- `c` context
- `port` physical port number

Returns `int`

- Less than 0: `oiError`

oiGetStreamAttributes

Get stream input/output attributes.

```
int oiGetStreamAttributes(const oiContext c, int port, int stream, oiStreamAttributes *a)
```

Arguments

- `c` context
- `port` physical port number
- `stream` stream index on port
- `a` stream attributes structure

Returns `int`

- Less than 0: `oiError`

oiSetStreamAttributes

Set stream input/output attributes

```
int oiSetStreamAttributes(oiContext c, int port, int stream, const oiStreamAttributes *a)
```

Arguments

- `c` context
- `port` physical port number
- `stream` stream index on port
- `a` stream attributes structure

Returns `int`

- Less than 0: `oiError`

oiReadStream

Read data from an open stream

```
int oiReadStream(oiContext c, int port, int stream, int nbytes, void *data)
```

Arguments

- `c` context
- `port` physical port number
- `stream` stream index on port
- `data` buffer to read data into

Returns `int`

- Greater than or equal to 0: number of bytes read
- Less than 0: `oiError`

oiWriteStream

Write data to an open stream

```
int oiWriteStream(oiContext c, int port, int stream, int nbytes, const void *data)
```

Arguments

- `c` context
- `port` physical port number
- `stream` stream index on port
- `data` buffer to write to stream

Returns `int`

- Greater than or equal to 0: number of bytes written
- Less than 0: `oiError`

oiGetNumPorts

Get the number of physical ports associated with a context

```
int oiGetNumPorts(const oiContext c)
```

Arguments

- `c` context

Returns `int`

- Greater than or equal to 0: number of physical ports
- Less than 0: `oiError`

oiGetDeviceType

Query the device type (EEPROM specified) of breakout board attached to a port.

```
int oiGetDeviceType(const oiContext c)
```

Arguments

- `c` context

Returns

- 0 if no device on port
- Positive number indicating `oiDeviceType`
- Negative number indicating `oiError`

Public Types

oiContext

```
typedef void * oiContext
```

oiDeviceType

```
typedef enum device {
    PASSTHROUGH,
    OE_DIO_BOARD,
    OE_AIO_BOARD,
    OE_ADIO_BOARD,
    OE_BASIC_INTAN_BOARD,
    OE_NEUROPIXEL_BOARD,
} oiDeviceType
```

oiError

```
typedef enum error {
    TERMINATE,
    ATTEMPT_WRITE_TO_INPUT_STREAM,
    ATTEMPT_READ_FROM_OUTPUT_STREAM,
    NO_DEVICE_ON_PORT,
    STREAM_DOES_NOT_EXIST,
    PORT_DOES_NOT_EXIST,
    CONTEXT_DOES_NOT_EXIST,
} oiError
```

oiStreamAttributes

```
typedef struct {
    size_t buffer_size;
    size_t timeout_msec;
    int flags;
} oiStreamAttributes
```

Private Types

Two types are used to define an oiContext.

oiContextImpl

```
struct oiContextIml {
    //TODO: members
    oiContextStateMachine *s;
}
```

oiStateMachine

```
struct oiContextStateMachine {
    // TODO: members
}
```