EPMUX

1.0.0

Generated by Doxygen 1.8.13

Contents

1 Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

epmux_channel_for_line_a_t	**2**
epmux_channel_for_line_b_t	**3**

2 File Index

2.1 File List

Here is a list of all files with brief descriptions:

3 Data Structure Documentation

3.1 epmux_channel_for_line_a_t Struct Reference

```
#include <epmux.h>
```

Data Fields

- uint8_t ModuleNumber
- uint8_t ChannelNumber
- uint8_t Reserved [14]

3.1.1 Field Documentation

3.1.1.1 ChannelNumber

```
uint8_t ChannelNumber
```

Channel channel number within the module (connector pin number). Numbering starts from 1. 0 – disconnect all channels. Valid range:0 – 64.

3.1.1.2 ModuleNumber

```
uint8_t ModuleNumber
```

Module chain position. Numbering starts from 1. 0 – disconnect all channels.

3.1.1.3 Reserved

```
uint8_t Reserved[14]
```

Software should not rely on the value of this field. To provide compatibility with future products the value of this field shouldn't be modified.

The documentation for this struct was generated from the following file:

- epmux.h

3.2 epmux_channel_for_line_b_t Struct Reference

```
#include <epmux.h>
```

Data Fields

- uint8_t ModuleNumber
- uint8_t ChannelNumber
- uint8_t Reserved [14]

3.2.1 Field Documentation

3.2.1.1 ChannelNumber

```
uint8_t ChannelNumber
```

Channel channel number within the module (connector pin number). Numbering starts from 1. 0 – disconnect all channels. Valid range:0 – 64.

3.2.1.2 ModuleNumber

```
uint8_t ModuleNumber
```

Module chain position. Numbering starts from 1. 0 – disconnect all channels.

3.2.1.3 Reserved

```
uint8_t Reserved[14]
```

Software should not rely on the value of this field. To provide compatibility with future products the value of this field shouldn't be modified.

The documentation for this struct was generated from the following file:

- epmux.h

3.3 epmux_get_chain_structure_t Struct Reference

```
#include <epmux.h>
```

Data Fields

- uint8_t ChainLength
- uint8_t ChainStructure [16]
- uint8_t Reserved [16]

3.3.1 Field Documentation

3.3.1.1 ChainLength

```
uint8_t ChainLength
```

Device chain length. Equals to the number of connected modules.

3.3.1.2 ChainStructure

```
uint8_t ChainStructure[16]
```

Device chain structure. Shows arrangement of modules of different types. Module types: 1 – module type A, supports channel connection to line A only; 2 – module type AB, support channel connection to both line A and line B; 0 – placeholder for the remaining part of the array, does not correspond to any real modules.

3.3.1.3 Reserved

```
uint8_t Reserved[16]
```

Software should not rely on the value of this field. To provide compatibility with future products the value of this field shouldn't be modified.

The documentation for this struct was generated from the following file:

- epmux.h

3.4 epmux_get_identity_information_t Struct Reference

```
#include <epmux.h>
```

Data Fields

- uint8_t Manufacturer [16]
- uint8_t ProductName [16]
- uint8_t ControllerName [16]
- uint8_t HardwareMajor
- uint8_t HardwareMinor
- uint16_t HardwareBugfix
- uint8_t BootloaderMajor
- uint8_t BootloaderMinor
- uint16_t BootloaderBugfix
- uint8_t FirmwareMajor
- uint8_t FirmwareMinor
- uint16_t FirmwareBugfix
- uint32_t SerialNumber
- uint8_t Reserved [8]

3.4.1 Field Documentation

3.4.1.1 BootloaderBugfix

```
uint16_t BootloaderBugfix
```

Bootloader release version number.

3.4.1.2 BootloaderMajor

```
uint8_t BootloaderMajor
```

Bootloader major version number.

3.4.1.3 BootloaderMinor

```
uint8_t BootloaderMinor
```

Bootloader minor version number.

3.4.1.4 ControllerName

```
uint8_t ControllerName[16]
```

User controller name. This name can be set by user via additional command.

3.4.1.5 FirmwareBugfix

```
uint16_t FirmwareBugfix
```

Firmware release version number.

3.4.1.6 FirmwareMajor

```
uint8_t FirmwareMajor
```

Firmware major version number.

3.4.1.7 FirmwareMinor

```
uint8_t FirmwareMinor
```

Firmware minor version number.

3.4.1.8 HardwareBugfix

```
uint16_t HardwareBugfix
```

Number of edits for this release of hardware.

3.4.1.9 HardwareMajor

```
uint8_t HardwareMajor
```

The major number of the hardware version.

3.4.1.10 HardwareMinor

```
uint8_t HardwareMinor
```

Minor number of the hardware version.

3.4.1.11 Manufacturer

```
uint8_t Manufacturer[16]
```

Manufacturer name. The name is set by the manufacturer.

3.4.1.12 ProductName

```
uint8_t ProductName[16]
```

Product name. The name is set by the manufacturer.

3.4.1.13 Reserved

```
uint8_t Reserved[8]
```

3.4.1.14 SerialNumber

```
uint32_t SerialNumber
```

Device serial number.

The documentation for this struct was generated from the following file:

- epmux.h

4 File Documentation

4.1 epmux.h File Reference

epmux API

```
#include <stdint.h>
#include <wchar.h>
```

Data Structures

- struct epmux_get_identity_information_t
- struct epmux_get_chain_structure_t
- struct epmux_channel_for_line_a_t
- struct epmux_channel_for_line_b_t

Macros

- #define EPMUX_BUILDER_VERSION_MAJOR 0
- #define EPMUX_BUILDER_VERSION_MINOR 10
- #define EPMUX_BUILDER_VERSION_BUGFIX 12
- #define EPMUX_BUILDER_VERSION_SUFFIX ""
- #define EPMUX_BUILDER_VERSION "0.10.12"
- #define EPMUX_URPC_API_EXPORT __attribute__((visibility("default")))
- #define EPMUX_URPC_CALLING_CONVENTION
- #define device_undefined (-1)
- #define result_ok 0
- #define result_error (-1)
- #define result_not_implemented (-2)
- #define result_value_error (-3)
- #define result_nodevice (-4)
- #define EPMUX_MAX_CHAIN_LENGTH 0x10
- #define EPMUX_MODULE_TYPE_A 0x1
- #define EPMUX_MODULE_TYPE_AB 0x2
- #define EPMUX_NO_MODULE 0x0

Logging level

- #define LOGLEVEL_ERROR 0x01
- #define LOGLEVEL_WARNING 0x02
- #define LOGLEVEL_INFO 0x03
- #define LOGLEVEL_DEBUG 0x04

Typedefs

- typedef int device_t
- typedef int result_t
- typedef void(EPMUX_URPC_CALLING_CONVENTION ∗ epmux_logging_callback_t) (int loglevel, const wchar_t ∗message, void ∗user_data)

Functions

- EPMUX_URPC_API_EXPORT void EPMUX_URPC_CALLING_CONVENTION epmux_logging_callback_↩ stderr_wide (int loglevel, const wchar_t ∗message, void ∗user_data)
- EPMUX_URPC_API_EXPORT void EPMUX_URPC_CALLING_CONVENTION epmux_logging_callback_↩ stderr_narrow (int loglevel, const wchar_t ∗message, void ∗user_data)
- EPMUX_URPC_API_EXPORT void EPMUX_URPC_CALLING_CONVENTION epmux_set_logging_↩ callback (epmux_logging_callback_t cb, void ∗data)
- EPMUX_URPC_API_EXPORT device_t EPMUX_URPC_CALLING_CONVENTION epmux_open_device (const char ∗uri)
- EPMUX_URPC_API_EXPORT result_t EPMUX_URPC_CALLING_CONVENTION epmux_libversion (char ∗lib_version)
- EPMUX_URPC_API_EXPORT result_t EPMUX_URPC_CALLING_CONVENTION epmux_save_settings (device_t handle)
- EPMUX_URPC_API_EXPORT result_t EPMUX_URPC_CALLING_CONVENTION epmux_read_settings (device_t handle)
- EPMUX_URPC_API_EXPORT result_t EPMUX_URPC_CALLING_CONVENTION epmux_get_identity_↩ information (device_t handle, epmux_get_identity_information_t ∗output)
- EPMUX_URPC_API_EXPORT result_t EPMUX_URPC_CALLING_CONVENTION epmux_reset (device_↩ t handle)
- EPMUX_URPC_API_EXPORT result_t EPMUX_URPC_CALLING_CONVENTION epmux_get_chain_↩ structure (device_t handle, epmux_get_chain_structure_t ∗output)
- EPMUX_URPC_API_EXPORT result_t EPMUX_URPC_CALLING_CONVENTION epmux_all_channels_off (device_t handle)
- EPMUX_URPC_API_EXPORT result_t EPMUX_URPC_CALLING_CONVENTION epmux_get_channel_↩ for_line_a (device_t handle, epmux_channel_for_line_a_t ∗output)
- EPMUX_URPC_API_EXPORT result_t EPMUX_URPC_CALLING_CONVENTION epmux_set_channel_↩ for_line_a (device_t handle, epmux_channel_for_line_a_t ∗input)
- EPMUX_URPC_API_EXPORT result_t EPMUX_URPC_CALLING_CONVENTION epmux_get_channel_↩ for_line_b (device_t handle, epmux_channel_for_line_b_t ∗output)
- EPMUX_URPC_API_EXPORT result_t EPMUX_URPC_CALLING_CONVENTION epmux_set_channel_↩ for_line_b (device_t handle, epmux_channel_for_line_b_t ∗input)
- EPMUX_URPC_API_EXPORT result_t EPMUX_URPC_CALLING_CONVENTION epmux_close_device (device_t ∗handle_ptr)
- EPMUX_URPC_API_EXPORT result_t EPMUX_URPC_CALLING_CONVENTION epmux_get_profile (device_t handle, char ∗∗buffer, void ∗(∗allocate)(size_t))
- EPMUX_URPC_API_EXPORT result_t EPMUX_URPC_CALLING_CONVENTION epmux_set_profile (device_t handle, char ∗buffer)

4.1.1 Detailed Description

epmux API

4.1.2 Macro Definition Documentation

4.1.2.1 device_undefined

```
#define device_undefined (-1)
```

4.1.2.2 EPMUX_BUILDER_VERSION

```
#define EPMUX_BUILDER_VERSION "0.10.12"
```

4.1.2.3 EPMUX_BUILDER_VERSION_BUGFIX

```
#define EPMUX_BUILDER_VERSION_BUGFIX 12
```

4.1.2.4 EPMUX_BUILDER_VERSION_MAJOR

```
#define EPMUX_BUILDER_VERSION_MAJOR 0
```

4.1.2.5 EPMUX_BUILDER_VERSION_MINOR

```
#define EPMUX_BUILDER_VERSION_MINOR 10
```

4.1.2.6 EPMUX_BUILDER_VERSION_SUFFIX

```
#define EPMUX_BUILDER_VERSION_SUFFIX ""
```

4.1.2.7 EPMUX_MAX_CHAIN_LENGTH

```
#define EPMUX_MAX_CHAIN_LENGTH 0x10
```

4.1.2.8 EPMUX_MODULE_TYPE_A

```
#define EPMUX_MODULE_TYPE_A 0x1
```

4.1.2.9 EPMUX_MODULE_TYPE_AB

```
#define EPMUX_MODULE_TYPE_AB 0x2
```

4.1.2.10 EPMUX_NO_MODULE

```
#define EPMUX_NO_MODULE 0x0
```

4.1.2.11 EPMUX_URPC_API_EXPORT

```
#define EPMUX_URPC_API_EXPORT __attribute__((visibility("default")))
```

4.1.2.12 EPMUX_URPC_CALLING_CONVENTION

```
#define EPMUX_URPC_CALLING_CONVENTION
```

4.1.2.13 LOGLEVEL_DEBUG

```
#define LOGLEVEL_DEBUG 0x04
```

Logging level - debug

4.1.2.14 LOGLEVEL_ERROR

```
#define LOGLEVEL_ERROR 0x01
```

Logging level - error

4.1.2.15 LOGLEVEL_INFO

```
#define LOGLEVEL_INFO 0x03
```

Logging level - info

4.1.2.16 LOGLEVEL_WARNING

```
#define LOGLEVEL_WARNING 0x02
```

Logging level - warning

4.1.2.17 result_error

```
#define result_error (-1)
```

4.1.2.18 result_nodevice

```
#define result_nodevice (-4)
```

4.1.2.19 result_not_implemented

```
#define result_not_implemented (-2)
```

4.1.2.20 result_ok

```
#define result_ok 0
```

4.1.2.21 result_value_error

```
#define result_value_error (-3)
```

4.1.3 Typedef Documentation

4.1.3.1 device_t

```
typedef int device_t
```

4.1.3.2 epmux_logging_callback_t

```
typedef void(EPMUX_URPC_CALLING_CONVENTION * epmux_logging_callback_t) (int loglevel, const
wchar_t *message, void *user_data)
```

Logging callback prototype.

Parameters

loglevel	- A logging level.
message	- A message.

4.1.3.3 result_t

```
typedef int result_t
```

4.1.4 Function Documentation

4.1.4.1 epmux_all_channels_off()

```
EPMUX_URPC_API_EXPORT result_t EPMUX_URPC_CALLING_CONVENTION epmux_all_channels_off (
             device_t handle )
```

Turns off all channels of all modules.

Parameters

in	*handle*	- Device ID, obtained by epmux_open_device() function.

4.1.4.2 epmux_close_device()

```
EPMUX_URPC_API_EXPORT result_t EPMUX_URPC_CALLING_CONVENTION epmux_close_device (
             device_t * handle_ptr )
```

Close specified device.

Parameters

handle_ptr	- An identifier of device.

4.1.4.3 epmux_get_chain_structure()

```
EPMUX_URPC_API_EXPORT result_t EPMUX_URPC_CALLING_CONVENTION epmux_get_chain_structure (
             device_t handle,
             epmux_get_chain_structure_t * output )
```

Automatic module chain structure discovery / update. By this command you can discover the number of connected modules, their types and arrangement in device chain. This command is not only for device discovery, but also for internal automatic configuration of the multiplexor. Therefore this command should be called in case of «hot-plug» hardware changes (module connection / disconnection). Otherwise you can face errors related to incorrect module numbering.

Parameters

in	*handle*	- Device ID, obtained by epmux_open_device() function.
out	*output*	- Device out data.

4.1.4.4 epmux_get_channel_for_line_a()

```
EPMUX_URPC_API_EXPORT result_t EPMUX_URPC_CALLING_CONVENTION epmux_get_channel_for_line_a (
             device_t handle,
             epmux_channel_for_line_a_t * output )
```

Active channel for line A (main line for signal output). The channel is determined by the module chain position and the channel number within the module (connector pin number). Only one channel can be connected to the line at the same time. In case of new channel connection the previous channel will be disconnected automatically. In case of «hot-plug» hardware changes (module connection / disconnection) hardware chain configuration should be updated by the get_chain_structure command before new channel connection.

Parameters

in	*handle*	- Device ID, obtained by epmux_open_device() function.
out	*output*	- Device out data.

4.1.4.5 epmux_get_channel_for_line_b()

```
EPMUX_URPC_API_EXPORT result_t EPMUX_URPC_CALLING_CONVENTION epmux_get_channel_for_line_b (
            device_t handle,
            epmux_channel_for_line_b_t * output )
```

Active channel for line B. The channel is determined by the module chain position and the channel number within the module (connector pin number). Only one channel can be connected to the line at the same time. In case of new channel connection the previous channel will be disconnected automatically. Some modules don't support channel connection to line B. You can find out the modules with line B connection support in current hardware chain configuration by the get_chain_structure command. In case of «hot-plug» hardware changes (module connection / disconnection) hardware chain configuration should be updated by the get_chain_structure command before new channel connection.

Parameters

in	*handle*	- Device ID, obtained by epmux_open_device() function.
out	*output*	- Device out data.

4.1.4.6 epmux_get_identity_information()

```
EPMUX_URPC_API_EXPORT result_t EPMUX_URPC_CALLING_CONVENTION epmux_get_identity_information (
            device_t handle,
            epmux_get_identity_information_t * output )
```

Return device identity information such as firmware version and serial number. It is useful to find your device in a list of available devices. It can be called from the firmware and bootloader.

Parameters

in	*handle*	- Device ID, obtained by epmux_open_device() function.
out	*output*	- Device out data.

4.1.4.7 epmux_get_profile()

```
EPMUX_URPC_API_EXPORT result_t EPMUX_URPC_CALLING_CONVENTION epmux_get_profile (
            device_t handle,
            char ** buffer,
            void *(*)(size_t) allocate )
```

Load profile from device.

Parameters

in	*handle*	- Device id.
out	*buffer*	- Pointer to output char* buffer. Memory for char* pointer must be allocated.
out	*allocate*	- Function for memory allocate.

4.1.4.8 epmux_libversion()

```
EPMUX_URPC_API_EXPORT result_t EPMUX_URPC_CALLING_CONVENTION epmux_libversion (
            char * lib_version )
```

Get library version.

Parameters

out	*lib_version*	- Library version.

4.1.4.9 epmux_logging_callback_stderr_narrow()

```
EPMUX_URPC_API_EXPORT void EPMUX_URPC_CALLING_CONVENTION epmux_logging_callback_stderr_narrow
(
            int loglevel,
            const wchar_t * message,
            void * user_data )
```

Simple callback for logging to stderr in narrow (single byte) chars.

Parameters

loglevel	- A logging level.
message	- A message.

4.1.4.10 epmux_logging_callback_stderr_wide()

```
EPMUX_URPC_API_EXPORT void EPMUX_URPC_CALLING_CONVENTION epmux_logging_callback_stderr_wide (
            int loglevel,
            const wchar_t * message,
            void * user_data )
```

Simple callback for logging to stderr in wide chars.

Parameters

loglevel	- A logging level.
message	- A message.

4.1.4.11 epmux_open_device()

```
EPMUX_URPC_API_EXPORT device_t EPMUX_URPC_CALLING_CONVENTION epmux_open_device (
            const char * uri )
```

Open a device by name *name* and return identifier of the device which can be used in calls.

Parameters

in	*name*	- A device name. Device name has form "com:port" or "xi-net://host/serial" or "udp://host:port". In case of USB-COM port the "port" is the OS device uri. For example "com:\\.\COM3" in Windows or "com:///dev/ttyACM34" in Linux/Mac. In case of network device the "host" is an IPv4 address or fully qualified domain uri (FQDN), "serial" is the device serial number in hexadecimal system. For example "xi-net://192.168.0.1/00001234" or "xi-net://hostname.com/89ABCDEF". In case of ethernet udp-com adapter the "host" is an IPv4 address, "port" is network port For example: "udp://192.168.0.2:1024" Note: only one program may use COM-device in same time. If errors occur when opening device, you need to make sure that the COM port is in the system and device is not currently used by other programs.

4.1.4.12 epmux_read_settings()

```
EPMUX_URPC_API_EXPORT result_t EPMUX_URPC_CALLING_CONVENTION epmux_read_settings (
            device_t handle )
```

Read all settings from controller's flash memory to controller's RAM, replacing previous data in controller's RAM.

Parameters

in	*handle*	- Device ID, obtained by epmux_open_device() function.

4.1.4.13 epmux_reset()

```
EPMUX_URPC_API_EXPORT result_t EPMUX_URPC_CALLING_CONVENTION epmux_reset (
            device_t handle )
```

Resets controller equivalently to the power switch reset. Shouldn't be used in normal practice.

Parameters

in	*handle*	- Device ID, obtained by epmux_open_device() function.

4.1.4.14 epmux_save_settings()

```
EPMUX_URPC_API_EXPORT result_t EPMUX_URPC_CALLING_CONVENTION epmux_save_settings (
            device_t handle )
```

Save all settings from controller's RAM to controller's flash memory, replacing previous data in controller's flash memory.

Parameters

in	*handle*	- Device ID, obtained by epmux_open_device() function.

4.1.4.15 epmux_set_channel_for_line_a()

```
EPMUX_URPC_API_EXPORT result_t EPMUX_URPC_CALLING_CONVENTION epmux_set_channel_for_line_a (
            device_t handle,
            epmux_channel_for_line_a_t * input )
```

Active channel for line A (main line for signal output). The channel is determined by the module chain position and the channel number within the module (connector pin number). Only one channel can be connected to the line at the same time. In case of new channel connection the previous channel will be disconnected automatically. In case of «hot-plug» hardware changes (module connection / disconnection) hardware chain configuration should be updated by the get_chain_structure command before new channel connection.

Parameters

in	*handle*	- Device ID, obtained by epmux_open_device() function.
in	*input*	- Device in data.

4.1.4.16 epmux_set_channel_for_line_b()

```
EPMUX_URPC_API_EXPORT result_t EPMUX_URPC_CALLING_CONVENTION epmux_set_channel_for_line_b (
            device_t handle,
            epmux_channel_for_line_b_t * input )
```

Active channel for line B. The channel is determined by the module chain position and the channel number within the module (connector pin number). Only one channel can be connected to the line at the same time. In case of new channel connection the previous channel will be disconnected automatically. Some modules don't support channel connection to line B. You can find out the modules with line B connection support in current hardware chain configuration by the get_chain_structure command. In case of «hot-plug» hardware changes (module connection / disconnection) hardware chain configuration should be updated by the get_chain_structure command before new channel connection.

Parameters

in	*handle*	- Device ID, obtained by epmux_open_device() function.
in	*input*	- Device in data.

4.1.4.17 epmux_set_logging_callback()

```
EPMUX_URPC_API_EXPORT void EPMUX_URPC_CALLING_CONVENTION epmux_set_logging_callback (
            epmux_logging_callback_t cb,
            void * data )
```

Sets a logging callback. Passing NULL disables logging.

Parameters

logging_callback	a callback for log messages

4.1.4.18 epmux_set_profile()

```
EPMUX_URPC_API_EXPORT result_t EPMUX_URPC_CALLING_CONVENTION epmux_set_profile (
            device_t handle,
            char * buffer )
```

Save profile to device

Parameters

in	*handle*	- Device id.
in	*buffer*	- Input char∗ buffer.

Index