Discussion:

How to "upload/download" SDOs in application(*.c file)?

(too old to reply)

jerry itnet 2018-06-18 16:15:23 UTC

I am confused on how to "upload/download" SDOs in an application.
For example, am I correct by writing codes like:

---------- application.c

....
uint16_t sdo_index = 0xf880; uint8_t sdo_subindex = 1;
uint32_t value = 0;
ecrt_slave_config_sdo32(sc, sdo_index, sdo_subindex, value);

if (ecrt_master_activate(master)) return -1;

size_t *result_size; uint8_t *target; const uint8_t *data;
uint32_t *abort_code;
ecrt_master_sdo_upload(master, 5, 0xf880, 1, target, 4, result_size,
abort_code);
...
cycle_task();
...

Thank in advance for any hint!
Jerry

Gavin Lambert 2018-06-18 23:25:38 UTC

There are a few different kinds of API to do SDO transfers, depending on how and when you want to do them. (But in short: no, that code is probably not correct.)

ecrt_master_sdo_download/upload are blocking APIs; they are intended to be called before activating the master. It is still possible to call them on a separate thread while the master is active (though be aware that this will cause lock contention so is not recommended if you have a fast cycle time). You must not call it from your realtime thread. These are intended for network discovery or other one-off or infrequent actions.

ecrt_slave_config_create_sdo_request and the companion ecrt_sdo_request_* methods are intended for the case where you want to do transfers (either one-off or periodically) from the realtime thread while the master is active. You create the request first (before activating the master) and then use the other methods to kick off a specific action and then poll it for completion from inside your realtime loop. These are non-blocking.

ecrt_slave_config_sdo* is for slave configuration settings that need to be written to the slave each time it reboots. You call these once before activating the master and then the master itself takes care of sending these on first configuration and also if the slave needs to be reconfigured for any reason (eg. network disruption, rebooting).

It's all explained in the documentation, the examples, and the header files themselves.

From: jerry itnet
Sent: Tuesday, 19 June 2018 04:15
Subject: [etherlab-users] How to "upload/download" SDOs in application(*.c file)?

I am confused on how to "upload/download" SDOs in an application.
For example, am I correct by writing codes like:

---------- application.c ----------
....
uint16_t sdo_index = 0xf880; uint8_t sdo_subindex = 1; uint32_t value = 0;
ecrt_slave_config_sdo32(sc, sdo_index, sdo_subindex, value);

if (ecrt_master_activate(master)) return -1;

size_t *result_size; uint8_t *target; const uint8_t *data; uint32_t *abort_code;
ecrt_master_sdo_upload(master, 5, 0xf880, 1, target, 4, result_size, abort_code);
...
cycle_task();
...

Thank in advance for any hint!
Jerry

jerry itnet 2018-06-20 16:07:56 UTC

Hi, Gavin:

Thanks for your guide, now it can upload SDO packet (seen with Wireshark).
But I got an error message on console:
* "Failed to execute SDO upload: Bad Address"*

---------- my code

...
size_t *result_size; uint8_t *target;

5 Replies
54 Views

Permalink to this page
Disable enhanced parsing

Thread Navigation

jerry itnet 2018-06-18 16:15:23 UTC
Gavin Lambert 2018-06-18 23:25:38
jerry itnet 2018-06-20 16:07:56 UTC
Gavin Lambert 2018-06-21 00:02:02
jerry itnet 2018-06-21 15:25:36
jerry itnet 2018-06-21 17:54:01

const uint8_t *data; uint32_t *abort_code;
ecrt_master_sdo_upload(master, 5, 0xf880, 1, target, 4, result_size,
abort_code);

...

if (ecrt_master_activate(master)) return -1;

cycle_task();

...

What could be wrong?
Thanks again!

Jerry

> *Post by Gavin Lambert*
> There are a few different kinds of API to do SDO transfers, depending on
> how and when you want to do them. (But in short: no, that code is probably
> not correct.)
> ecrt_master_sdo_download/upload are blocking APIs; they are intended to
> be called before activating the master. It is still possible to call them *on
> a separate thread* while the master is active (though be aware that this
> will cause lock contention so is not recommended if you have a fast cycle
> time). You *must not* call it from your realtime thread. These are
> intended for network discovery or other one-off or infrequent actions.
> ecrt_slave_config_create_sdo_request and the companion ecrt_sdo_request_*
> methods are intended for the case where you want to do transfers (either
> one-off or periodically) from the realtime thread while the master is
> active. You create the request first (before activating the master) and
> then use the other methods to kick off a specific action and then poll it
> for completion from inside your realtime loop. These are non-blocking.
> ecrt_slave_config_sdo* is for slave configuration settings that need to be
> written to the slave each time it reboots. You call these once before
> activating the master and then the master itself takes care of sending
> these on first configuration and also if the slave needs to be reconfigured
> for any reason (eg. network disruption, rebooting).
> It's all explained in the documentation, the examples, and the header files themselves.
> *From:* jerry itnet
> *Sent:* Tuesday, 19 June 2018 04:15
> *Subject:* [etherlab-users] How to "upload/download" SDOs in
> application(*.c file)?
> I am confused on how to "upload/download" SDOs in an application.
> ---------- application.c ----------
> ----------
> ....
> uint16_t sdo_index = 0xf880; uint8_t sdo_subindex = 1;
> uint32_t value = 0;
> ecrt_slave_config_sdo32(sc, sdo_index, sdo_subindex, value);
> if (ecrt_master_activate(master)) return -1;
> size_t *result_size; uint8_t *target; const uint8_t *data;
> uint32_t *abort_code;
> ecrt_master_sdo_upload(master, 5, 0xf880, 1, target, 4, result_size, abort_code);
> ...
> cycle_task();
> ...
> ----------
> ----------
> Thank in advance for any hint!
> Jerry

Gavin Lambert 2018-06-21 00:02:02 UTC

You are passing a garbage pointer as the abort_code. This should be a pointer to a local variable that will receive the output value. See the examples.

From: jerry itnet [mailto:***@gmail.com]
Sent: Thursday, 21 June 2018 04:08
Subject: Re: [etherlab-users] How to "upload/download" SDOs in application(*.c file)?

Hi, Gavin:

Thanks for your guide, now it can upload SDO packet (seen with Wireshark).
But I got an error message on console:
"Failed to execute SDO upload: Bad Address"

---------- my code ----------
...
size_t *result_size; uint8_t *target;
const uint8_t *data; uint32_t *abort_code;
ecrt_master_sdo_upload(master, 5, 0xf880, 1, target, 4, result_size, abort_code);

...
if (ecrt_master_activate(master)) return -1;
cycle_task();
...

What could be wrong?
Thanks again!

Jerry

On Mon, Jun 18, 2018 at 7:25 PM, Gavin Lambert
<***@tomra.com<mailto:***@tomra.com>> wrote:
There are a few different kinds of API to do SDO transfers, depending on how and when you want to do them. (But in short: no, that code is probably not correct.)

ecrt_master_sdo_download/upload are blocking APIs; they are intended to be called before activating the master. It is still possible to call them on a separate thread while the master is active (though be aware that this will cause lock contention so is not recommended if you have a fast cycle time). You must not call it from your realtime thread. These are intended for network discovery or other one-off or infrequent actions.

ecrt_slave_config_create_sdo_request and the companion ecrt_sdo_request_* methods are intended for the case where you want to do transfers (either one-off or periodically) from the realtime thread while the master is active. You create the request first (before activating the master) and then use the other methods to kick off a specific action and then poll it for completion from inside your realtime loop.

These are non-blocking.

ecrt_slave_config_sdo* is for slave configuration settings that need to be written to the slave each time it reboots. You call these once before activating the master and then the master itself takes care of sending these on first configuration and also if the slave needs to be reconfigured for any reason (eg. network disruption, rebooting).

It's all explained in the documentation, the examples, and the header files themselves.

From: jerry itnet
Sent: Tuesday, 19 June 2018 04:15
Subject: [etherlab-users] How to "upload/download" SDOs in application(*.c file)?

I am confused on how to "upload/download" SDOs in an application.
For example, am I correct by writing codes like:

---------- application.c ----------
....
uint16_t sdo_index = 0xf880; uint8_t sdo_subindex = 1; uint32_t value = 0;
ecrt_slave_config_sdo32(sc, sdo_index, sdo_subindex, value);

if (ecrt_master_activate(master)) return -1;

size_t *result_size; uint8_t *target; const uint8_t *data; uint32_t *abort_code;
ecrt_master_sdo_upload(master, 5, 0xf880, 1, target, 4, result_size, abort_code);
...
cycle_task();
...

Thank in advance for any hint!
Jerry

jerry itnet 2018-06-21 15:25:36 UTC

Actually I had been looking for examples before I posted my questions:
* But I could not find any example with usage of "ecrt_master_sdo_upload".
* Do I have to check a specific branch?

Thanks!

> *Post by Gavin Lambert*
> You are passing a garbage pointer as the abort_code. This should be a pointer to a local variable that will receive the output value. See the examples.
> *Sent:* Thursday, 21 June 2018 04:08
> *Subject:* Re: [etherlab-users] How to "upload/download" SDOs in application(*.c file)?
> Thanks for your guide, now it can upload SDO packet (seen with Wireshark).
> * "Failed to execute SDO upload: Bad Address"*
> ---------- my code ----------
> ...
> size_t *result_size; uint8_t *target;
> const uint8_t *data; uint32_t *abort_code;
> ecrt_master_sdo_upload(master, 5, 0xf880, 1, target, 4, result_size, abort_code);
> ...
> if (ecrt_master_activate(master)) return -1;
> cycle_task();
> ...
> ----------
> ----------
> What could be wrong?
> Thanks again!
> Jerry
> There are a few different kinds of API to do SDO transfers, depending on how and when you want to do them. (But in short: no, that code is probably not correct.)
> ecrt_master_sdo_download/upload are blocking APIs; they are intended to be called before activating the master. It is still possible to call them *on a separate thread* while the master is active (though be aware that this will cause lock contention so is not recommended if you have a fast cycle time). You *must not* call it from your realtime thread. These are intended for network discovery or other one-off or infrequent actions.
> ecrt_slave_config_create_sdo_request and the companion ecrt_sdo_request_* methods are intended for the case where you want to do transfers (either one-off or periodically) from the realtime thread while the master is active. You create the request first (before activating the master) and then use the other methods to kick off a specific action and then poll it for completion from inside your realtime loop. These are non-blocking.
> ecrt_slave_config_sdo* is for slave configuration settings that need to be written to the slave each time it reboots. You call these once before activating the master and then the master itself takes care of sending these on first configuration and also if the slave needs to be reconfigured for any reason (eg. network disruption, rebooting).
> It's all explained in the documentation, the examples, and the header files themselves.
> *From:* jerry itnet
> *Sent:* Tuesday, 19 June 2018 04:15
> *Subject:* [etherlab-users] How to "upload/download" SDOs in application(*.c file)?
> I am confused on how to "upload/download" SDOs in an application.
> ---------- application.c ----------
> ....
> uint16_t sdo_index = 0xf880; uint8_t sdo_subindex = 1;
> uint32_t value = 0;
> ecrt_slave_config_sdo32(sc, sdo_index, sdo_subindex, value);
> if (ecrt_master_activate(master)) return -1;
> size_t *result_size; uint8_t *target; const uint8_t *data;
> uint32_t *abort_code;
> ecrt_master_sdo_upload(master, 5, 0xf880, 1, target, 4, result_size, abort_code);
> ...
> cycle_task();
> ...
> ----------
> ----------
> Thank in advance for any hint!
> Jerry

jerry itnet 2018-06-21 17:54:01 UTC

I do the following, but still got error of "Bad address":
---
typedef struct {
// inputs
uint16_t slave_position;
uint16_t sdo_index;
uint8_t sdo_entry_subindex;
size_t target_size;
uint8_t *target;

```
// outputs
size_t data_size;
uint32_t abort_code;
} ec_ioctl_slave_sdo_upload_t;

ec_ioctl_slave_sdo_upload_t data;

data.slave_position = 5;
data.sdo_index = 0xf880;
data.sdo_entry_subindex = 1;
data.target_size = 4;

ecrt_master_sdo_upload(master, data.slave_position, data.sdo_index,
data.sdo_entry_subindex, data.target,
data.target_size,
&data.data_size, &data.abort_code);
```


Thanks!

> *Post by Gavin Lambert*
> You are passing a garbage pointer as the abort_code. This should be a
> pointer to a local variable that will receive the output value. See the
> examples.
> *Sent:* Thursday, 21 June 2018 04:08
> *Subject:* Re: [etherlab-users] How to "upload/download" SDOs in
> application(*.c file)?
> Thanks for your guide, now it can upload SDO packet (seen with Wireshark).
> * "Failed to execute SDO upload: Bad Address"*
> ---------- my code ----------
> ---
> ...
> size_t *result_size; uint8_t *target;
> const uint8_t *data; uint32_t *abort_code;
> ecrt_master_sdo_upload(master, 5, 0xf880, 1, target, 4, result_size, abort_code);
> ...
> if (ecrt_master_activate(master)) return -1;
> cycle_task();
> ...
> ---
> ---
> What could be wrong?
> Thanks again!
> Jerry
> There are a few different kinds of API to do SDO transfers, depending on
> how and when you want to do them. (But in short: no, that code is probably
> not correct.)
> ecrt_master_sdo_download/upload are blocking APIs; they are intended to
> be called before activating the master. It is still possible to call them *on
> a separate thread* while the master is active (though be aware that this
> will cause lock contention so is not recommended if you have a fast cycle
> time). You *must not* call it from your realtime thread. These are
> intended for network discovery or other one-off or infrequent actions.
> ecrt_slave_config_create_sdo_request and the companion ecrt_sdo_request_*
> methods are intended for the case where you want to do transfers (either
> one-off or periodically) from the realtime thread while the master is
> active. You create the request first (before activating the master) and
> then use the other methods to kick off a specific action and then poll it
> for completion from inside your realtime loop. These are non-blocking.
> ecrt_slave_config_sdo* is for slave configuration settings that need to be
> written to the slave each time it reboots. You call these once before
> activating the master and then the master itself takes care of sending
> these on first configuration and also if the slave needs to be reconfigured
> for any reason (eg. network disruption, rebooting).
> It's all explained in the documentation, the examples, and the header files themselves.
> *From:* jerry itnet
> *Sent:* Tuesday, 19 June 2018 04:15
> *Subject:* [etherlab-users] How to "upload/download" SDOs in
> application(*.c file)?
> I am confused on how to "upload/download" SDOs in an application.
> ---------- application.c ----------
> ---
> ....
> uint16_t sdo_index = 0xf880; uint8_t sdo_subindex = 1;
> uint32_t value = 0;
> ecrt_slave_config_sdo32(sc, sdo_index, sdo_subindex, value);
> if (ecrt_master_activate(master)) return -1;
> size_t *result_size; uint8_t *target; const uint8_t *data;
> uint32_t *abort_code;
> ecrt_master_sdo_upload(master, 5, 0xf880, 1, target, 4, result_size, abort_code);
> ...
> cycle_task();
> ...
> ---
> ---
> Thank in advance for any hint!
> Jerry