Discussion:

[etherlab-users] WARNINGS: "Working counter changed" and "datagram UNMATCHED"

r*@tecodrive.com** 9 years ago

Hello everybody!

I am trying to build an etherlab-application based on the user-example. I modified the source code for my bus configuration. After compiling and running the program I received the following output:

Configuring PDOs...
Activating master...
Starting timer...
Started.
5 slave(s).
AL states: 0x02.
Link is up.
Domain1: WC 1.
Domain1: State 1.
Domain1: WC 4.
Domain1: WC 5.
Domain1: WC 6.
Domain1: State 2.
AL states: 0x08.
Domain1: WC 0.
Domain1: State 0.
Domain1: WC 6.
Domain1: State 2.
Domain1: WC 0.
Domain1: State 0.
Domain1: WC 6.
Domain1: State 2.

The program works, the digital outputs are blinking/toggleing. I paid attention to the domain working counter- and state changes. This messages appear when the program is running a few seconds. The kernel logs showed the following messages:

***@terminal:~$ sudo dmesg -c
[29842.462777] EtherCAT 0: Domain 0: Working counter changed to 0/6.
[29842.752788] EtherCAT WARNING: Datagram f73b59c0 (domain0-0) was SKIPPED 2 times.
[29843.076180] EtherCAT WARNING 0: 1 datagram UNMATCHED!
[29843.472786] EtherCAT 0: Domain 0: Working counter changed to 6/6.
***@terminal:~$ sudo dmesg -c
[29855.462831] EtherCAT 0: Domain 0: Working counter changed to 0/6.
[29855.882832] EtherCAT WARNING: Datagram f73b59c0 (domain0-0) was SKIPPED 2 times.
[29856.084222] EtherCAT WARNING 0: 1 datagram UNMATCHED!
[29856.472779] EtherCAT 0: Domain 0: Working counter changed to 6/6.

Thread Navigation
r***@tecodrive.com 9 years ago
matthieu bec 9 years ago
r***@tecodrive.com 9 years ago
Matthieu Bec 9 years ago

Here some infos about the system: Ubuntu 10.04 LTS with Kernel 2.6.31.6 and PREEMPT patch but same warnings also appear on a system without RT PREEMPT.
etherlab version info:
***@terminal:~$ sudo /opt/etherlab/bin/ethercat version
IgH EtherCAT master 1.5.1 0f7a243b03e4

I can imagine there are timing problems because program is working and the messages appear irregular. On the other side, EtherCAT is a real-time capable system but does not require real-time, isn't it? Can you tell me what's wrong in the program or what is the reason for warnings?

Below you can see the program.

Best Regards,
Marcel

```
#include <errno.h>
#include <signal.h>
#include <stdio.h>
#include <string.h>
#include <sys/resource.h>
#include <sys/time.h>
#include <sys/types.h>
#include <unistd.h>

/****************************************************************************/

#include "ecrt.h"

/****************************************************************************/

// Application parameters
#define FREQUENCY 100
#define PRIORITY 1

// Optional features
#define CONFIGURE_PDOS 1
#define SDO_ACCESS 0

/****************************************************************************/

// EtherCAT
static ec_master_t *master = NULL;
static ec_master_state_t master_state = {};

static ec_domain_t *domain1 = NULL;
static ec_domain_state_t domain1_state = {};

static ec_slave_config_t *sc_ana_in = NULL;
```

```
static ec_slave_config_state_t sc_ana_in_state = {};

// Timer
static unsigned int sig_alarms = 0;
static unsigned int user_alarms = 0;

/****************************************************************************/

// process data
static uint8_t *domain1_pd = NULL;

#define BusCouplerPos 0, 0
#define PotentialPos 0, 1
#define DigOutSlavePos 0, 2
#define DigInSlavePos 0, 2
#define AnaInSlavePos 0, 3
#define ThermoSlavePos 0, 4

#define Beckhoff_EK1100 0x00000002, 0x044c2c52
#define Beckhoff_EL9110 0x00000002, 0x23963052
#define Beckhoff_EL1859 0x00000002, 0x07433052
#define Beckhoff_EL3111 0x00000002, 0x0c273052
#define Beckhoff_EL3311 0x00000002, 0x0cef3052


// offsets for PDO entries
static unsigned int off_potential_status;
static unsigned int off_dig_out;
static unsigned int off_dig_in;
static unsigned int off_ana_in;
static unsigned int off_thermo_in;

const static ec_pdo_entry_reg_t domain1_regs[] = {
{PotentialPos, Beckhoff_EL9110, 0x6000, 1, &off_potential_status},
{DigOutSlavePos, Beckhoff_EL1859, 0x7080, 1, &off_dig_out},
{DigInSlavePos, Beckhoff_EL1859, 0x6000, 1, &off_dig_in},
{AnaInSlavePos, Beckhoff_EL3111, 0x6000, 0x11, &off_ana_in},
{ThermoSlavePos, Beckhoff_EL3311, 0x6000, 0x11, &off_thermo_in},
{}
};

static unsigned int counter = 0;
static unsigned int blink = 0;

/****************************************************************************/

#if CONFIGURE_PDOS

/* Master 0, Slave 1, "EL9110"
* Vendor ID: 0x00000002
* Product code: 0x23963052
* Revision number: 0x00100000
*/
```

```
ec_pdo_entry_info_t slave_1_pdo_entries[] = {
{0x6000, 0x01, 1}, /* Input */
};

ec_pdo_info_t slave_1_pdos[] = {
{0x1a00, 1, slave_1_pdo_entries + 0}, /* PowerOK */
};

ec_sync_info_t slave_1_syncs[] = {
{0, EC_DIR_INPUT, 1, slave_1_pdos + 0, EC_WD_DISABLE},
{0xff}
};

/* Master 0, Slave 2, "EL1859"
* Vendor ID: 0x00000002
* Product code: 0x07433052
* Revision number: 0x00100000
*/

ec_pdo_entry_info_t slave_2_pdo_entries[] = {
{0x7080, 0x01, 1}, /* Output */
{0x7090, 0x01, 1}, /* Output */
{0x70a0, 0x01, 1}, /* Output */
{0x70b0, 0x01, 1}, /* Output */
{0x70c0, 0x01, 1}, /* Output */
{0x70d0, 0x01, 1}, /* Output */
{0x70e0, 0x01, 1}, /* Output */
{0x70f0, 0x01, 1}, /* Output */
{0x6000, 0x01, 1}, /* Input */
{0x6010, 0x01, 1}, /* Input */
{0x6020, 0x01, 1}, /* Input */
{0x6030, 0x01, 1}, /* Input */
{0x6040, 0x01, 1}, /* Input */
{0x6050, 0x01, 1}, /* Input */
{0x6060, 0x01, 1}, /* Input */
{0x6070, 0x01, 1}, /* Input */
};

ec_pdo_info_t slave_2_pdos[] = {
{0x1608, 1, slave_2_pdo_entries + 0}, /* Channel 9 */
{0x1609, 1, slave_2_pdo_entries + 1}, /* Channel 10 */
{0x160a, 1, slave_2_pdo_entries + 2}, /* Channel 11 */
{0x160b, 1, slave_2_pdo_entries + 3}, /* Channel 12 */
{0x160c, 1, slave_2_pdo_entries + 4}, /* Channel 13 */
{0x160d, 1, slave_2_pdo_entries + 5}, /* Channel 14 */
{0x160e, 1, slave_2_pdo_entries + 6}, /* Channel 15 */
{0x160f, 1, slave_2_pdo_entries + 7}, /* Channel 16 */
{0x1a00, 1, slave_2_pdo_entries + 8}, /* Channel 1 */
{0x1a01, 1, slave_2_pdo_entries + 9}, /* Channel 2 */
{0x1a02, 1, slave_2_pdo_entries + 10}, /* Channel 3 */
{0x1a03, 1, slave_2_pdo_entries + 11}, /* Channel 4 */
{0x1a04, 1, slave_2_pdo_entries + 12}, /* Channel 5 */
{0x1a05, 1, slave_2_pdo_entries + 13}, /* Channel 6 */
{0x1a06, 1, slave_2_pdo_entries + 14}, /* Channel 7 */
```

```
{0x1a07, 1, slave_2_pdo_entries + 15}, /* Channel 8 */
};

ec_sync_info_t el1859_syncs[] = {
{0, EC_DIR_OUTPUT, 8, slave_2_pdos + 0, EC_WD_ENABLE},
{1, EC_DIR_INPUT, 8, slave_2_pdos + 8, EC_WD_DISABLE},
{0xff}
};

/* Master 0, Slave 3, "EL3111"
* Vendor ID: 0x00000002
* Product code: 0x0c273052
* Revision number: 0x00100000
*/

ec_pdo_entry_info_t slave_3_pdo_entries[] = {
{0x6000, 0x01, 1}, /* Underrange */
{0x6000, 0x02, 1}, /* Overrange */
{0x6000, 0x03, 2}, /* Limit 1 */
{0x6000, 0x05, 2}, /* Limit 2 */
{0x6000, 0x07, 1}, /* Error */
{0x0000, 0x00, 1}, /* Gap */
{0x0000, 0x00, 5}, /* Gap */
{0x1c32, 0x20, 1},
{0x1800, 0x07, 1}, /* TxPDO State */
{0x1800, 0x09, 1}, /* TxPDO Toggle */
{0x6000, 0x11, 16}, /* Value */
};

ec_pdo_info_t slave_3_pdos[] = {
{0x1a00, 11, slave_3_pdo_entries + 0}, /* AI TxPDO-Map Standard Ch.1 */
};

ec_sync_info_t el3111_syncs[] = {
{0, EC_DIR_OUTPUT, 0, NULL, EC_WD_DISABLE},
{1, EC_DIR_INPUT, 0, NULL, EC_WD_DISABLE},
{2, EC_DIR_OUTPUT, 0, NULL, EC_WD_DISABLE},
{3, EC_DIR_INPUT, 1, slave_3_pdos + 0, EC_WD_DISABLE},
{0xff}
};

/* Master 0, Slave 4, "EL3311"
* Vendor ID: 0x00000002
* Product code: 0x0cef3052
* Revision number: 0x00120000
*/

ec_pdo_entry_info_t slave_4_pdo_entries[] = {
{0x6000, 0x01, 1}, /* Underrange */
{0x6000, 0x02, 1}, /* Overrange */
{0x6000, 0x03, 2}, /* Limit 1 */
{0x6000, 0x05, 2}, /* Limit 2 */
{0x6000, 0x07, 1}, /* Error */
{0x0000, 0x00, 7}, /* Gap */
```

```
{0x6000, 0x0f, 1}, /* TxPDO State */
{0x1800, 0x09, 1},
{0x6000, 0x11, 16}, /* Value */
};

ec_pdo_info_t slave_4_pdos[] = {
{0x1a00, 9, slave_4_pdo_entries + 0}, /* TC TxPDO-MapCh.1 */
};

ec_sync_info_t slave_4_syncs[] = {
{0, EC_DIR_OUTPUT, 0, NULL, EC_WD_DISABLE},
{1, EC_DIR_INPUT, 0, NULL, EC_WD_DISABLE},
{2, EC_DIR_OUTPUT, 0, NULL, EC_WD_DISABLE},
{3, EC_DIR_INPUT, 1, slave_4_pdos + 0, EC_WD_DISABLE},
{0xff}
};

#endif

/****************************************************************************/

#if SDO_ACCESS
static ec_sdo_request_t *sdo;
#endif

/****************************************************************************/

void check_domain1_state(void)
{
ec_domain_state_t ds;

ecrt_domain_state(domain1, &ds);

if (ds.working_counter != domain1_state.working_counter)
printf("Domain1: WC %u.\n", ds.working_counter);
if (ds.wc_state != domain1_state.wc_state)
printf("Domain1: State %u.\n", ds.wc_state);

domain1_state = ds;
}

/****************************************************************************/

void check_master_state(void)
{
ec_master_state_t ms;

ecrt_master_state(master, &ms);

if (ms.slaves_responding != master_state.slaves_responding)
printf("%u slave(s).\n", ms.slaves_responding);
if (ms.al_states != master_state.al_states)
printf("AL states: 0x%02X.\n", ms.al_states);
if (ms.link_up != master_state.link_up)
```

```
printf("Link is %s.\n", ms.link_up ? "up" : "down");

master_state = ms;
}

/*****************************************************************************/

void check_slave_config_states(void)
{
ec_slave_config_state_t s;

ecrt_slave_config_state(sc_ana_in, &s);

if (s.al_state != sc_ana_in_state.al_state)
printf("AnaIn: State 0x%02X.\n", s.al_state);
if (s.online != sc_ana_in_state.online)
printf("AnaIn: %s.\n", s.online ? "online" : "offline");
if (s.operational != sc_ana_in_state.operational)
printf("AnaIn: %soperational.\n",
s.operational ? "" : "Not ");

sc_ana_in_state = s;
}

/*****************************************************************************/

#if SDO_ACCESS
void read_sdo(void)
{
switch (ecrt_sdo_request_state(sdo)) {
case EC_REQUEST_UNUSED: // request was not used yet
ecrt_sdo_request_read(sdo); // trigger first read
break;
case EC_REQUEST_BUSY:
fprintf(stderr, "Still busy...\n");
break;
case EC_REQUEST_SUCCESS:
fprintf(stderr, "SDO value: 0x%04X\n",
EC_READ_U16(ecrt_sdo_request_data(sdo)));
ecrt_sdo_request_read(sdo); // trigger next read
break;
case EC_REQUEST_ERROR:
fprintf(stderr, "Failed to read SDO!\n");
ecrt_sdo_request_read(sdo); // retry reading
break;
}
}
#endif

/*****************************************************************************/

void cyclic_task()
{
int i;
```

```
// receive process data
ecrt_master_receive(master);
ecrt_domain_process(domain1);

// check process data state (optional)
check_domain1_state();

if (counter) {
counter--;
} else { // do this at 1 Hz
counter = FREQUENCY;

// calculate new process data
blink = !blink;

// check for master state (optional)
check_master_state();

// check for islave configuration state(s) (optional)
//check_slave_config_states();

#if SDO_ACCESS
// read process data SDO
read_sdo();
#endif

}

#if 0
// read process data
printf("AnaIn: state %u value %u\n",
EC_READ_U8(domain1_pd + off_ana_in_status),
EC_READ_U16(domain1_pd + off_ana_in_value));
#endif

#if 1
// write process data
EC_WRITE_U8(domain1_pd + off_dig_out, blink ? 0x06 : 0x09);
#endif

// send process data
ecrt_domain_queue(domain1);
ecrt_master_send(master);
}

/****************************************************************************/

void signal_handler(int signum) {
switch (signum) {
case SIGALRM:
sig_alarms++;
break;
}
```

```
}

/****************************************************************************/

int main(int argc, char **argv)
{
ec_slave_config_t *sc;
struct sigaction sa;
struct itimerval tv;

master = ecrt_request_master(0);
if (!master)
return -1;

domain1 = ecrt_master_create_domain(master);
if (!domain1)
return -1;

if (!(sc_ana_in = ecrt_master_slave_config(
master, AnaInSlavePos, Beckhoff_EL3111))) {
fprintf(stderr, "Failed to get slave configuration.\n");
return -1;
}

#if SDO_ACCESS
fprintf(stderr, "Creating SDO requests...\n");
if (!(sdo = ecrt_slave_config_create_sdo_request(sc_ana_in,
0x3102, 2, 2))) {
fprintf(stderr, "Failed to create SDO request.\n");
return -1;
}
ecrt_sdo_request_timeout(sdo, 500); // ms
#endif

#if CONFIGURE_PDOS
printf("Configuring PDOs...\n");
if (ecrt_slave_config_pdos(sc_ana_in, EC_END, el3111_syncs)) {
fprintf(stderr, "Failed to configure PDOs.\n");
return -1;
}

if (!(sc = ecrt_master_slave_config(
master, DigOutSlavePos, Beckhoff_EL1859))) {
fprintf(stderr, "Failed to get slave configuration.\n");
return -1;
}

if (ecrt_slave_config_pdos(sc, EC_END, el1859_syncs)) {
fprintf(stderr, "Failed to configure PDOs.\n");
return -1;
}
#endif

// Create configuration for bus coupler
```

```
sc = ecrt_master_slave_config(master, BusCouplerPos, Beckhoff_EK1100);
if (!sc)
return -1;

if (ecrt_domain_reg_pdo_entry_list(domain1, domain1_regs)) {
fprintf(stderr, "PDO entry registration failed!\n");
return -1;
}

printf("Activating master...\n");
if (ecrt_master_activate(master))
return -1;

if (!(domain1_pd = ecrt_domain_data(domain1))) {
return -1;
}

#if PRIORITY
pid_t pid = getpid();
if (setpriority(PRIO_PROCESS, pid, -19))
fprintf(stderr, "Warning: Failed to set priority: %s\n",
strerror(errno));
#endif

sa.sa_handler = signal_handler;
sigemptyset(&sa.sa_mask);
sa.sa_flags = 0;
if (sigaction(SIGALRM, &sa, 0)) {
fprintf(stderr, "Failed to install signal handler!\n");
return -1;
}

printf("Starting timer...\n");
tv.it_interval.tv_sec = 0;
tv.it_interval.tv_usec = 1000000 / FREQUENCY;
tv.it_value.tv_sec = 0;
tv.it_value.tv_usec = 1000;
if (setitimer(ITIMER_REAL, &tv, NULL)) {
fprintf(stderr, "Failed to start timer: %s\n", strerror(errno));
return 1;
}

printf("Started.\n");
while (1) {
pause();

#if 0
struct timeval t;
gettimeofday(&t, NULL);
printf("%u.%06u\n", t.tv_sec, t.tv_usec);
#endif

while (sig_alarms != user_alarms) {
cyclic_task();
```

```
user_alarms++;
}
}

return 0;
}

/*****************************************************************************/
```

matthieu bec 9 years ago

Hello Marcel,

Two things you might want to check:

- have a look at the 'dc_user' example (mlockall, nanosleep cycle). The 'user' example works with setitimer, that might be less deterministic somehow

- are you using the 'generic' driver? have a look at patch drivers if your NIC is supported

Regards,
Matthieu

--
Matthieu Bec GMTO Corp.
cell : +1 626 354 9367 P.O. Box 90933
phone: +1 626 204 0527 Pasadena, CA 91109-0933

r*@tecodrive.com** 9 years ago

Hello Matthieu,

thank you very much for your answer. I displayed the NIC information with terminal command "sudo lshw -C network":
*-network:1
description: Ethernet interface
product: 82540EM Gigabit Ethernet Controller
vendor: Intel Corporation
physical id: 8

bus info: ***@0000:00:08.0
logical name: eth2
version: 02
serial: 00:08:54:53:fa:42
size: 1GB/s
capacity: 1GB/s
width: 32 bits
clock: 66MHz
capabilities: pm pcix bus_master cap_list ethernet physical tp
10bt 10bt-fd 100bt 100bt-fd 1000bt-fd autonegotiation
configuration: autonegotiation=on broadcast=yes driver=e1000
driverversion=7.3.21-k3-NAPI duplex=full firmware=N/A latency=64
link=yes mingnt=255 multicast=yes port=twisted pair speed=1GB/s
resources: irq:9 memory:f0820000-f083ffff ioport:d240(size=8)

Yes, I am using the generic driver because of information from
etherlab.org: "Since version 1.5, there is a generic Ethernet driver
among the native ones, that spans all Ethernet devices supported by
the Linux kernel. Although it is not usable with realtime patches like
RTAI (because it uses the lower network stack layers), it runs
perfectly with realtime preemption."

Regards,
Marcel
...

----- Ende der weitergeleiteten Nachricht -----

Matthieu Bec 9 years ago

Hello Marcel,

The generic driver should handle 1kHz frame rate just fine. Your card
use the e1000 driver and IgH patch would work, but may not make a lot of
difference for your relatively slow use case.
The intel NIC are somewhat optimized for bandwidth rather than latency,
adjusting the e1000 driver params might help: InterruptThrottleRate,
Rx/TxIntDelay etc. - modinfo e1000 will list them, look for more
information in this mail archive some people reported good improvements.

other than that, nothing strikes me as out of the ordinary in your app,
sometimes the issue lies elsewhere: bad slave, faulty cable, etc

Matthieu
...

--
Matthieu Bec GMTO Corp.
cell : +1 626 354 9367 P.O. Box 90933
phone: +1 626 204 0527 Pasadena, CA 91109-0933