igh ethercat应用层代码分析

时间 2020-05-31

标签 igh ethercat 应用层 代码 分析 繁體版

原文 https://blog.csdn.net/qq_43530144/article/details/105857464

igh ethercat应用层代码分析

igh ethercat能够本身编写应用层代码与从站进行数据交互，自己主站给了例子，在example文件夹下，根据本身平台选择看某个例子，通常刚开始能够看user文件夹下的main.c文件，下面是我进行了一些修改，也带了一些解释。其实也就是控制io作流水灯操做。

```
//gcc test.c -o test.out -I/opt/etherlab/include -L/opt/etherlab/lib/ -lethercat
#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//控制数据发送频率
#define PRIORITY 1

// Optional features
#define CONFIGURE_PDOS 1

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

// 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_dig_out = NULL;
static ec_slave_config_state_t sc_dig_out_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 //alias,position
#define DigOutSlavePos 0, 1

#define Beckhoff_EK9100 0x0000001a, 0x09139100 //vendor ID,product code
#define Beckhoff_EL2016 0x0000001a, 0x09132016

static unsigned int Dig_out=1;

// offsets for PDO entries
```


8. Improved open set domain adaptation

相关文章

1. iGH EtherCAT初始化流程分析（二）
2. IgH EtherCAT Master 源码编译安装
3. Linux下IGH Ethercat Master安装
4. EtherCAT 应用层协议
5. IgH(IgH EtherCAT Master for Linux)编译之linux Debian篇
6. IGH (EtherCAT Master Linux on PC)编译教程
7. IgH Master 1.5.2 Documentation 中文版
8. Java应用代码分层实践
9. Igh Etherlab-master安装
10. EtherCAT 应用层协议的部分理解（一）

>>更多相关文章<<

```c
static unsigned int off_dig_out;

const static ec_pdo_entry_reg_t domain1_regs[] = {
    {DigOutSlavePos, Beckhoff_EL2016, 0x3001, 1, &off_dig_out},
    {}
};

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

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

#if CONFIGURE_PDOS

// Digital out ------------------------

static ec_pdo_entry_info_t el2016_channels[] = {
    {0x3001, 1, 1}, // Value 1
    {0x3001, 2, 1}, // Value 2
    {0x3001, 3, 1}, // Value 3
    {0x3001, 4, 1}, // Value 4
    {0x3001, 5, 1}, // Value 5
    {0x3001, 6, 1}, // Value 6
    {0x3001, 7, 1}, // Value 7
    {0x3001, 8, 1}  // Value 8
};

static ec_pdo_info_t el2016_pdos[] = {
    {0x1600, 8, el2016_channels},
};

static ec_sync_info_t el2016_syncs[] = {
    {0, EC_DIR_OUTPUT, 1, el2016_pdos},
    {0xff}
};
#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_dig_out, &s);

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

    sc_dig_out_state = s;
}

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

void cyclic_task()
{
    // 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;

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

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

    }

    // write process data
  b l i n k++;
if(blink==100){
        if(Dig_out<0x100){
     D i g _ o u t=Dig_out*2;
        }
        else{
     D i g _ o u t=1;
        }
       b l i n k=0;
}
    EC_WRITE_U8(domain1_pd + off_dig_out, Dig_out);

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

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

void signal_handler(int signum) {//进程收到定时信号所作的操做
    switch (signum) {
        case SIGALRM:
            sig_alarms++;
            break;
    }
}
```

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

int main(int argc, char **argv)
{
    ec_slave_config_t *sc;// 存放从站配置的结构体
    struct sigaction sa;// 描述信号到达时要采起的操做的结构体
    struct itimerval tv;//计算时间，用于定时的结构体

    master = ecrt_request_master(0);// 请求EtherCAT主机进行实时操做。
    if (!master)
        return -1;

    domain1 = ecrt_master_create_domain(master);// 建立新的进程数据域
    if (!domain1)
        return -1;

    printf("Configuring PDOs...\n");

    if (!(sc_dig_out = ecrt_master_slave_config(//获取从站配置
                    master, DigOutSlavePos, Beckhoff_EL2016))) {//第一个参数是所请求的主站实例，第二个包括主站的别名和位置，
第三个包括供应商码和产品码
        fprintf(stderr, "Failed to get slave configuration.\n");
        return -1;
    }

    if (ecrt_slave_config_pdos(sc_dig_out, EC_END, el2016_syncs)) {//指定完整的PDO配置。第一个参数是获取的从站配置，第二个参
数表示同步管理器配置数，EC_END=~u0,第三个参数表示同步管理器配置数组
        fprintf(stderr, "Failed to configure PDOs.\n");
        return -1;
    }

    // Create configuration for bus coupler
    sc = ecrt_master_slave_config(master, BusCouplerPos, Beckhoff_EK9100);
    if (!sc)
        return -1;

    if (ecrt_domain_reg_pdo_entry_list(domain1, domain1_regs)) {//为进程数据域注册一组PDO项。参数一:建立的进程数据域，参数二
:pdo注册数组
        fprintf(stderr, "PDO entry registration failed!\n");
        return -1;
    }

    printf("Activating master...\n");
    if (ecrt_master_activate(master))// 激活主站
        return -1;

    if (!(domain1_pd = ecrt_domain_data(domain1))) {//返回域的进程数据
        return -1;
    }

#if PRIORITY
    pid_t pid = getpid();//得到进程PID
    if (setpriority(PRIO_PROCESS, pid, -19))//设置进程优先级
        fprintf(stderr, "Warning: Failed to set priority: %s\n",
                strerror(errno));
#endif

    sa.sa_handler = signal_handler;//指定信号关联函数
    sigemptyset(&sa.sa_mask);// 初始化给出的信号集为空
    sa.sa_flags = 0;
    if (sigaction(SIGALRM, &sa, 0)) {//系统调用，用于更改进程在收到特定信号时采起的操做
        fprintf(stderr, "Failed to install signal handler!\n");
        return -1;
    }

    printf("Starting timer...\n");
    tv.it_interval.tv_sec = 0;//next time
    tv.it_interval.tv_usec = 1000000 / FREQUENCY;
    tv.it_value.tv_sec = 0;//current time
    tv.it_value.tv_usec = 1000;
    if (setitimer(ITIMER_REAL, &tv, NULL)) {//定时功能，以系统真实的时间来计算，每隔一段时间送出SIGALRM信号。
```

```
        fprintf(stderr, "Failed to start timer: %s\n", strerror(errno));
        return 1;
    }

    printf("Started.\n");
    while (1) {
        pause();//挂起，等待cpu唤醒

        while (sig_alarms != user_alarms) {//意思是每过一段时间才会执行一次，而不是一直死循环
            cyclic_task();
            user_alarms++;
        }
    }

    return 0;
}

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


/** 先定义一些结构体，其中有存放从站配置的结构体，类型为ec_slave_config_t{}，用在接收函数ecrt_master_slave_config()的返回值；用在描述信号到达时要采起的操做的结构体struct sigaction{}；计算时间，用于定时的结构体struct itimerval{}。 下面是关于主站和从站的一些配置 1. 请求EtherCAT主机进行操做，我感受能够认为是建立一个主站实例对象。2. 建立进程数据域。 3. 获取从站配置 4. 指定完整的pdo配置 5. 为进程数据域注册一组PDO项。6. 激活主站 7. 返回域的进程数据 8. 可选选项：设置进程优先级 9. 可选选项：指定进程收到某种信号采起的操做 10. 可选选项：设置定时发送信号 11. 进入循环，开始循环任务 其中对于几个可选选项，设置优先级，是为了让实时性更好；另外两个定时发送的信号做为进程采起操做的条件，而操做内容又做为循环任务的周期时间间隔，不对，这两个并再也不是可选项，应该必需要设置的，周期交换数据的时间间隔要大于等于从站处理 数据和传输等时间总和。 下面是在循环任务中作的事情：1. 从硬件获取接收的帧并处理数据报。2. 肯定域数据报的状态。3. 检查域的状态（可选项）4. 周期性检查主站状态和从站配置状态（可选项）5. 计算将要发送流水灯数据和数据改变周期大小 6. 将数据放入数据域 7. 将主数据报队列中的全部域数据报排队 8. 发送队列中的全部数据报。 其中对于配置PDO所涉及到的几个结构体 typedef struct { uint16_t index; /**< PDO entry index. */
    uint8_t subindex; /**< PDO entry subindex. */
    uint8_t bit_length; /**< Size of the PDO entry in bit. */
} ec_pdo_entry_info_t;


typedef struct {
    uint16_t index; /**< PDO index. */
    unsigned int n_entries; /**< Number of PDO entries in entries to map. Zero means, that the default mapping shall be
 used (this can only be done if the slave is present at bus configuration time). */
    ec_pdo_entry_info_t *entries; /**< Array of PDO entries to map. Can either be NULL, or must contain at least n_entr
ies values. */
} ec_pdo_info_t;


typedef struct {
    uint8_t index; /**< Sync manager index. Must be less than #EC_MAX_SYNC_MANAGERS for a valid sync manager, but can a
lso be 0xff to mark the end of the list. */
    ec_direction_t dir; /**< Sync manager direction. */
    unsigned int n_pdos; /**< Number of PDOs in pdos. */
    ec_pdo_info_t *pdos; /**< Array with PDOs to assign. This must contain at least n_pdos PDOs. */
    ec_watchdog_mode_t watchdog_mode; /**< Watchdog mode. */
} ec_sync_info_t;


typedef struct {
    uint16_t alias; /**< Slave alias address. */
    uint16_t position; /**< Slave position. */
    uint32_t vendor_id; /**< Slave vendor ID. */
    uint32_t product_code; /**< Slave product code. */
    uint16_t index; /**< PDO entry index. */
    uint8_t subindex; /**< PDO entry subindex. */
    unsigned int *offset; /**< Pointer to a variable to store the PDO entry's(byte-)offset in the process data. */
    unsigned int *bit_position; /**< Pointer to a variable to store a bit position (0-7) within the offset. Can be NULL
, in which case an error is raised if the PDO entry does not byte-align. */
} ec_pdo_entry_reg_t;


#define EC_WRITE_U8(DATA, VAL) \ do { \ *((uint8_t *)(DATA)) = ((uint8_t) (VAL)); \ } while (0)

*/
```

相关文章

1. iGH EtherCAT初始化流程分析（二）
2. IgH EtherCAT Master 源码编译安装
3. Linux下IGH Ethercat Master安装
4. EtherCAT 应用层协议
5. IgH(IgH EtherCAT Master for Linux)编译之linux Debian篇
6. IGH (EtherCAT Master Linux on PC)编译教程
7. IgH Master 1.5.2 Documentation 中文版
8. Java应用代码分层实践
9. Igh Etherlab-master安装
10. EtherCAT 应用层协议的部分理解（一）

更多相关文章...

相关标签/搜索

igh ethercat 应用层 源码分析 分层 代码分析工具 Caffe代码解析 代码 应用 分析