中华田园巨龙
码龄4年 暂无认证
78
原创
1221
积分
私信 关注

搜博主文章

热门文章

最新评论

您愿意向朋友推荐“博客详情页”吗？

强烈不推荐 不推荐 一般般 推荐 强烈推荐

最新文章

2021年 29篇 2020年 58篇

Ethercat系列---IGH DC模式测试电机程序（Preempt-RT）

原创 中华田园巨龙 2020-09-21 16:20:35 1557 收藏 18 版权

分类专栏： EtherCAT系列 文章标签： IGH Ethercat linux 嵌入式

EtherCAT系列 专栏收录该内容

先上代码

linux内核版本：4.19-rt
Ethercat: DC模式
伺服运行模式: CSP（周期位置同步）

```c
#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 <time.h>
#include <sys/mman.h>
#include <malloc.h>
#include <sched.h> /* sched_setscheduler() */

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

#include "ecrt.h"

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

// Application parameters
#define FREQUENCY 1000
#define CLOCK_TO_USE CLOCK_MONOTONIC
#define MEASURE_TIMING

#define TARGET_POSITION         0 /*target position*/
#define CYCLIC_POSITION           8   /*Operation mode for 0x6060:0*/

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

#define NSEC_PER_SEC (1000000000L)
#define PERIOD_NS (NSEC_PER_SEC / FREQUENCY)    /*本次设置周期PERIOD_NS为1ms*/

#define DIFF_NS(A, B) (((B).tv_sec - (A).tv_sec) * NSEC_PER_SEC + \
        (B).tv_nsec - (A).tv_nsec)

#define TIMESPEC2NS(T) ((uint64_t) (T).tv_sec * NSEC_PER_SEC + (T).tv_nsec)

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

// 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 = NULL;
static ec_slave_config_state_t sc_state = {};

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

// process data
static uint8_t *domain1_pd = NULL;

#define PANASONIC        0,0                    /*EtherCAT address on the bus*/

#define VID_PID          0x0000066F,0x515050a1   /*Vendor ID, product code*/

/*Offsets for PDO entries*/
static struct{
    unsigned int ctrl_word;
    unsigned int operation_mode;
    unsigned int target_position;
    unsigned int touch_probe_function;
    unsigned int error_code;
    unsigned int status_word;
    unsigned int modes_of_operation_display;
    unsigned int position_actual_value;
    unsigned int touch_probe_status;
    unsigned int touch_probe_pos1_pos_value;
    unsigned int following_error_actual_value;
    unsigned int digital_inputs;
}offset;

const static ec_pdo_entry_reg_t domain1_regs[] = {
    {PANASONIC, VID_PID, 0x6040, 0, &offset.ctrl_word},
    {PANASONIC, VID_PID, 0x6060, 0, &offset.operation_mode },
    {PANASONIC, VID_PID, 0x607A, 0, &offset.target_position},
    {PANASONIC, VID_PID, 0x60B8, 0, &offset.touch_probe_function},
    {PANASONIC, VID_PID, 0x603F, 0, &offset.error_code},
    {PANASONIC, VID_PID, 0x6041, 0, &offset.status_word},
    {PANASONIC, VID_PID, 0x6061, 0, &offset.modes_of_operation_display},
    {PANASONIC, VID_PID, 0x6064, 0, &offset.position_actual_value},
    {PANASONIC, VID_PID, 0x60B9, 0, &offset.touch_probe_status},
    {PANASONIC, VID_PID, 0x60BA, 0, &offset.touch_probe_pos1_pos_value},
    {PANASONIC, VID_PID, 0x60F4, 0, &offset.following_error_actual_value},
    {PANASONIC, VID_PID, 0x60FD, 0, &offset.digital_inputs},
    {}
};

/****************************************************************************/
/*Config PDOs*/
static ec_pdo_entry_info_t device_pdo_entries[] = {
    /*RxPdo 0x1600*/
    {0x6040, 0x00, 16},
    {0x6060, 0x00, 8 },
    {0x607A, 0x00, 32},
    {0x60B8, 0x00, 16},
    /*TxPdo 0x1A00*/
    {0x603F, 0x00, 16},
    {0x6041, 0x00, 16},
    {0x6061, 0x00, 8},
    {0x6064, 0x00, 32},
    {0x60B9, 0x00, 16},
    {0x60BA, 0x00, 32},
    {0x60F4, 0x00, 32},
    {0x60FD, 0x00, 32},
};

static ec_pdo_info_t device_pdos[] = {
    //RxPdo
    {0x1600, 4, device_pdo_entries + 0 },
    //TxPdo
    {0x1A00, 8, device_pdo_entries + 4 }
};
```

```c

static ec_sync_info_t device_syncs[] = {
    { 0, EC_DIR_OUTPUT, 0, NULL, EC_WD_DISABLE },
    { 1, EC_DIR_INPUT, 0, NULL, EC_WD_DISABLE },
    { 2, EC_DIR_OUTPUT, 1, device_pdos + 0, EC_WD_DISABLE },
    { 3, EC_DIR_INPUT, 1, device_pdos + 1, EC_WD_DISABLE },
    { 0xFF}
};

static unsigned int counter = 0;
static unsigned int blink = 0;
static unsigned int sync_ref_counter = 0;
const struct timespec cycletime = {0, PERIOD_NS};

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

struct timespec timespec_add(struct timespec time1, struct timespec time2)
{
    struct timespec result;

    if ((time1.tv_nsec + time2.tv_nsec) >= NSEC_PER_SEC) {
        result.tv_sec = time1.tv_sec + time2.tv_sec + 1;
        result.tv_nsec = time1.tv_nsec + time2.tv_nsec - NSEC_PER_SEC;
    } else {
        result.tv_sec = time1.tv_sec + time2.tv_sec;
        result.tv_nsec = time1.tv_nsec + time2.tv_nsec;
    }

    return result;
}

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

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, &s);
    if (s.al_state != sc_state.al_state)
    {
        printf("slave: State 0x%02X.\n", s.al_state);
    }
    if (s.online != sc_state.online)
    {
        printf("slave: %s.\n", s.online ? "online" : "offline");
    }
    if (s.operational != sc_state.operational)
    {
        printf("slave: %soperational.\n", s.operational ? "" : "Not ");
    }
    sc_state = s;
}

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

void cyclic_task()
{
    static uint16_t command=0x004F;//用来帮助判断状态字的值
    uint16_t    status; //读取伺服状态

    struct timespec wakeupTime, time;
#ifdef MEASURE_TIMING
    struct timespec startTime, endTime, lastStartTime = {};
    uint32_t period_ns = 0, exec_ns = 0, latency_ns = 0,
             latency_min_ns = 0, latency_max_ns = 0,
             period_min_ns = 0, period_max_ns = 0,
             exec_min_ns = 0, exec_max_ns = 0;
#endif

    // get current time
    clock_gettime(CLOCK_TO_USE, &wakeupTime);

    while(1) {
        wakeupTime = timespec_add(wakeupTime, cycletime);
        clock_nanosleep(CLOCK_TO_USE, TIMER_ABSTIME, &wakeupTime, NULL);

        // Write application time to master
        //
        // It is a good idea to use the target time (not the measured time) as
        // application time, because it is more stable.
        //
        ecrt_master_application_time(master, TIMESPEC2NS(wakeupTime));

#ifdef MEASURE_TIMING
        clock_gettime(CLOCK_TO_USE, &startTime);
        latency_ns = DIFF_NS(wakeupTime, startTime);
        period_ns = DIFF_NS(lastStartTime, startTime);
        exec_ns = DIFF_NS(lastStartTime, endTime);
        lastStartTime = startTime;

        if (latency_ns > latency_max_ns) {
            latency_max_ns = latency_ns;
        }
        if (latency_ns < latency_min_ns) {
            latency_min_ns = latency_ns;
        }
        if (period_ns > period_max_ns) {
            period_max_ns = period_ns;
        }
        if (period_ns < period_min_ns) {
            period_min_ns = period_ns;
        }
```

```
        if (exec_ns > exec_max_ns) {
            exec_max_ns = exec_ns;
        }
        if (exec_ns < exec_min_ns) {
            exec_min_ns = exec_ns;
        }
#endif

        // 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 slave configuration state(s)
        check_slave_config_states();

#ifdef MEASURE_TIMING
            // output timing stats
            printf("period     %10u ... %10u\n",
                    period_min_ns, period_max_ns);
            printf("exec       %10u ... %10u\n",
                    exec_min_ns, exec_max_ns);
            printf("latency    %10u ... %10u\n",
                    latency_min_ns, latency_max_ns);
            period_max_ns = 0;
            period_min_ns = 0xffffffff;
            exec_max_ns = 0;
            exec_min_ns = 0xffffffff;
            latency_max_ns = 0;
            latency_min_ns = 0xffffffff;
#endif

            // calculate new process data
            blink = !blink;
        }
        /*Read state*/
        status = EC_READ_U16(domain1_pd + offset.status_word);//读取状态字

        // write process data
        //DS402 CANOpen over EtherCAT status machine
        if( (status & command) == 0x0040 )
        {
            EC_WRITE_U16(domain1_pd + offset.ctrl_word, 0x0006 );
            EC_WRITE_S8(domain1_pd + offset.operation_mode, CYCLIC_POSITION);
        //set control mode
            command = 0x006F;
        }

        else if( (status & command) == 0x0021)
        {
            EC_WRITE_U16(domain1_pd + offset.ctrl_word, 0x0007 );
            command = 0x006F;
        }

        else if( (status & command) == 0x0023)
        {
            EC_WRITE_U16(domain1_pd + offset.ctrl_word, 0x000f );
            command = 0x006F;
        }
        //operation enabled

        else if( (status & command) == 0x0027)
        {
            EC_WRITE_U16(domain1_pd + offset.ctrl_word, 0x001f );
        }

        if (sync_ref_counter) {
            sync_ref_counter--;
        } else {
            sync_ref_counter = 1; // sync every cycle

            clock_gettime(CLOCK_TO_USE, &time);
            ecrt_master_sync_reference_clock_to(master, TIMESPEC2NS(time));
        }
        ecrt_master_sync_slave_clocks(master);

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

#ifdef MEASURE_TIMING
        clock_gettime(CLOCK_TO_USE, &endTime);
#endif
    }
}

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

int main(int argc, char **argv)
{
    if (mlockall(MCL_CURRENT | MCL_FUTURE) == -1) {
        perror("mlockall failed");
        return -1;
    }

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

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

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

    printf("Configuring PDOs...\n");
    if (ecrt_slave_config_pdos(sc, EC_END, device_syncs))
    {
        fprintf(stderr, "Failed to configure slave PDOs!\n");
        exit(EXIT_FAILURE);
    }
    else
    {
        printf("*Success to configuring slave PDOs*\n");
    }

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

    // configure SYNC signals for this slave
```

```
    ecrt_slave_config_dc(sc, 0x0300, PERIOD_NS, 4400000, 0, 0);

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

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

    /* Set priority */

    struct sched_param param = {};
    param.sched_priority = sched_get_priority_max(SCHED_FIFO);

    printf("Using priority %i.", param.sched_priority);
    if (sched_setscheduler(0, SCHED_FIFO, &param) == -1) {
        perror("sched_setscheduler failed");
    }

    printf("Starting cyclic function.\n");
    cyclic_task();

    return 0;
}

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

操作步骤

实验效果：电机成功切换OP并使能

自定义步骤：

1、启动IGH主站

```
sudo /etc/init.d/ethercat restart
```

2、查看电机的信息

```
sudo ethercat cstruct -a 0
```

记下电机VenderID，ProductID，PDO信息

3、自定义代码

（1）周期修改

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

// Application parameters
#define FREQUENCY 1000
#define CLOCK_TO_USE CLOCK_MONOTONIC
#define MEASURE_TIMING

#define TARGET_POSITION         0 /*target position*/
#define CYCLIC_POSITION           8   /*Operation mode for 0x6060:0*/
```

频率 单位HZ

（2）VenderID，ProductID修改

```
#define PANASONIC       0,0                     /*EtherCAT address on the bus*/
#define VID_PID         0x0000066F,0x515050a1   /*Vendor ID, product code*/
```

（3）PDO修改

```
/*Offsets for PDO entries*/
static struct{
    unsigned int ctrl_word;
    unsigned int operation_mode;
    unsigned int target_position;
    unsigned int touch_probe_function;
    unsigned int error_code;
    unsigned int status_word;
    unsigned int modes_of_operation_display;
    unsigned int position_actual_value;
    unsigned int touch_probe_status;
    unsigned int touch_probe_pos1_pos_value;
    unsigned int following_error_actual_value;
    unsigned int digital_inputs;
}offset;

const static ec_pdo_entry_reg_t domain1_regs[] = {
    {PANASONIC, VID_PID, 0x6040, 0, &offset.ctrl_word},
    {PANASONIC, VID_PID, 0x6060, 0, &offset.operation_mode },
    {PANASONIC, VID_PID, 0x607A, 0, &offset.target_position},
    {PANASONIC, VID_PID, 0x60B8, 0, &offset.touch_probe_function},
    {PANASONIC, VID_PID, 0x603F, 0, &offset.error_code},
    {PANASONIC, VID_PID, 0x6041, 0, &offset.status_word},
    {PANASONIC, VID_PID, 0x6061, 0, &offset.modes_of_operation_display},
    {PANASONIC, VID_PID, 0x6064, 0, &offset.position_actual_value},
    {PANASONIC, VID_PID, 0x60B9, 0, &offset.touch_probe_status},
    {PANASONIC, VID_PID, 0x60BA, 0, &offset.touch_probe_pos1_pos_value},
    {PANASONIC, VID_PID, 0x60F4, 0, &offset.following_error_actual_value},
    {PANASONIC, VID_PID, 0x60FD, 0, &offset.digital_inputs},
    {}
};

/*****************************************************************************/
/*Config PDOs*/
static ec_pdo_entry_info_t device_pdo_entries[] = {
    /*RxPdo 0x1600*/
    {0x6040, 0x00, 16},
    {0x6060, 0x00, 8 },
    {0x607A, 0x00, 32},
    {0x60B8, 0x00, 16},
    /*TxPdo 0x1A00*/
    {0x603F, 0x00, 16},
    {0x6041, 0x00, 16},
    {0x6061, 0x00, 8},
    {0x6064, 0x00, 32},
    {0x60B9, 0x00, 16},
    {0x60BA, 0x00, 32},
    {0x60F4, 0x00, 32},
    {0x60FD, 0x00, 32},
};

static ec_pdo_info_t device_pdos[] = {
    //RxPdo
    {0x1600, 4, device_pdo_entries + 0 },
    //TxPdo
    {0x1A00, 8, device_pdo_entries + 4 }
};
```

对应PDO名称

PDO地址

PDO数据类型

PDO个数：例 4个RXPDO，8个TXPDO

4、编译，运行

```
make
sudo ./ec_dc_user_example
```

上面的程序在执行后，电机会使能但不会运动，如果想要让电机运动的话，可以参考以下代码

```
        int32_t actual_position =0;
        int32_t target_position =0;

        ...
        ...

        if (sync_ref_counter) {
            sync_ref_counter--;
        } else {
            sync_ref_counter = 1; // sync every cycle

            clock_gettime(CLOCK_TO_USE, &time);
            ecrt_master_sync_reference_clock_to(master, TIMESPEC2NS(time));

            if( (status & command) == 0x0027)   //确认为OP状态，电机使能
            {
            actual_position = EC_READ_S32(domain1_pd + offset.position_actual_value);//读取当前位置
            target_position = actual_position + 10;   //把当前位置+10 赋值为下一目标位置
            EC_WRITE_S32(domain1_pd + offset.target_position, target_position ); //把目标位置写入对象字典
            }

        }
```

说明：定义一个当前位置actual_position 和目标位置target_position ，在周期任务里，判断当电机切换为使能状态后，读取电机当前位置，并把这个数值加10作为目标位置写入电机。

ethercat电机mfc简单连接程序 12-28
ethercat电机mfc简单连接程序，在windows操作系统上，建立控制电机工程的程序，可以自主的设计控制程序，代码只提供了连接电机部分代码，简单清晰

优质评论可以帮助作者获得更高权重 评论

nanshanaa: 博主，在哪里可以获取到当前电机是否处于使能状态啊 13 天前 回复

中华田园巨龙 博主 回复： 一般可以通过伺服面板查看状态。或者在安全状态下，用手转动电机轴。转不动说明已经使能。也可以通过读6041状态字。 13 天前 回复

推着我的小摩托: 大佬，这个是主站作为参考时钟的demo，如果是从站作为参考时钟怎么运行呢？ 28 天前 回复

all in the time !: 大佬，你好！请问你在这里只给电机了一个目标位置，那驱动电机的底层在哪里呢？比如说电机的转速、加速度等这些。谢谢！ 3 月前

登录 查看 34 条热评

AM5728 Preempt-RT Igh Ethercat安装与测试_YEYUANGEN... 9-1
IgH EtherCAT Master 源码编译安装 https://blog.csdn.net/scyhk/article/details/51672136 Ethercat系列---IGH DC模式测试电机程序(Preempt-RT) https://...

Ethercat系列--IGH Ethercat Master 安装时./configure参数一览 9-18
Ethercat系列---IGH DC模式测试电机程序(Preempt-RT) 中华田园巨龙: 一般可以通过伺服面板查看状态。或者在安全状态下,用手转动电机轴。转不动说明...

关于Igh-EthercatMaster DC时钟的同步方式和性能 exat500g的博客 5274
发现SYNC0信号的周期跳变最大7us,平均2us,而且有随机周期160ms的7us跳变尖峰,比起官方宣称的100ns时间同步精度相差甚远. 原来是Igh-EthercatMa...

EtherCAT igh源码的ecrt_slave_config_dc（ ）函数的理解。 cln512的博客 1721
总结一下自己对igh的ecrt_slave_config_dc（ ）函数的理解。参考了igh的example里的"dc_user例程"。 例程里有这样一处代码： // configure SYNC signa...

EtherCAT总线通信Freerun、SM、DC三种同步模式分析_a41... 8-29
1、一般而言,如果EtherCAT总线通讯时的同步模式不是DC模式,那么就是SM同步模式了, 2、SM(Sync Manager同步管理器)指的是同步管理器的同步,它的...

Ethercat-IghMaster 1.5.2调试笔记_wllw7176的专栏 9-14
5. 标准Igh Master接口的使用例子 mytest目录 mytest/test_torque_sanyo_ioctl 基于ioctl接口的sanyo电机测试例子 mytest/test_torque_tec_ioctl 基于ioctl...

IGH EtherCAT应用层控制电机代码 qq_43530144的博客 1828
在主站配置好之后，连接从站，我用的是雷赛的伺服，对于大部分来说改个pid，vid，应该就可以用，这个是用的pv模式 应该还是容易懂的，我把一些用...

EtherCAT使用与解析-关于DC lswdcyy的博客 1316
DC，distributed clock(不是Direct current...)，分布式时钟；分布式时钟的意义在于所有EtherCAT设备使用相同的时间，控制各个设备同步执行，尤其体...

倍福提供的EtherCAT从站代码包解析 - 进阶的Kaiser@ZJU... 12-26
在EtherCAT实现SoE功能 这个文件显示了一个简短的例子 38,440testappl.c 29,615 testappl.h Testappl.c测试应用程序是一个特定的从站堆栈提供大部分...

EtherCAT 同步模式_chuhan0732的博客 9-14
EtherCAT三种同步方式 自由运行模式 (Free Run:非同步运行) 自由运行模式通过应用程序控制器的本地计时器中断启动。本地周期从通信周期或主站周期...

EtherCAT igh主站控制松下伺服（csp模式） cln512的博客 1749
完整代码 #include <errno.h> #include <signal.h> #include <stdio.h> #include <string.h> #include <sys/resource.h> #include <sys/time.h> #include <sy...

EtherCAT igh主站控制埃斯顿伺服（csp模式） cln512的博客 2686
完整代码 算了、干脆直接贴代码了。最近都在搞EtherCAT主站、从站的应用，过段日子再分享一些项目中基础的东西。驱动伺服主要还是参考厂商提供的...

基于EtherCAT实时通信的电机驱动控制 最新发布 10-16
实时工业以太网EtherCAT凭借着高性能、低成本、应用简易等优点在现代控制领域得到了广泛的应用和迅速的发展。为了将EtherCAT快速应用到电机驱...

EtherCAT igh主站控制3个台达asdaa2伺服转圈圈 12-03
EtherCAT igh主站控制3个台达asdaa2伺服转圈圈。包括伺服使能、控制3个电机转圈圈、每秒读取电机的实际位置、运行90秒自动退出程序。。。需要...

ethercat环境搭建（igh安装） weixin_42186805的博客 1333
需要对应打了实时补丁的内核，可以看我前一篇博客 环境准备 ：和之前搭建内核的环境一样， gcc修改一下，改成4.8.5，(把/usr/bin 下的链接文件gcc指...

Etherlab IGH DC问题 Pl_sunyang的博客 1332
同步时钟有两种方式：1.使用主站时钟作为整个从机时钟的DC同步方式（同步时间长，误差大）2.使用第一个带DC的从站作为参考时钟,然后将主站时钟...

IgH配置EtherCAT流程 EtherCAT 6157
IgH设置EtherCAT数据流程 在启动IgH协议栈后，协议栈会自动的把EtherCAT从站初始化到PREOP模式，从PREOP状态迁移到SAFEOP状态，需要设置P...

Ethercat igh 文档 11-25
Ethercat igh 文档

IGH EtherCAT manual 10-28
linux开源ethercat主站igh，官网用户手册。

EtherCAT IGH 安装手册 10-28
IGH安装手册，beckhoff 数字量输入输出控制实例解析。

1 IgH EtherCAT 主站 Queen B is Mine 8129
1 IgH EtherCAT 主站本章涵盖了有关EtherCAT主站的一些常规信息。1.1 功能摘要下面的列表给出了主要功能的简短摘要。 Linux 2.6 / 3.x的内核模块设...

EtherCAT主站配置过程分析.xlsx 10-15
本文档分析了国内某ehtercat主站的启动过程。对启动过程中的每一个报文的作用进行了标注。部分标注如下： "DC过程： 1.主站写900，发一帧写的数据...