



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

菜鸟笔记---startx执行后显示Only console users are allowed to run the X server

3219

Ethercat系列---IGH ethercat master在linux4.x环境下的编译安装

2596

菜鸟笔记---linux系统移植之[FAILED] Failed to start Network Time Synchronization. 问题解决

2215

菜鸟笔记---fatal error: openssl/bio.h: No such file or directory 解决方案

1962

菜鸟笔记---debian根文件系统启动时[FAILED] Failed to start Create Volatile Files and Directories.

1786

最新评论

菜鸟笔记---startx执行后显示Only consol...
电子厂 在造螺丝工: 想请教作者这个问题有没有什么办法解决哇~

Ethercat系列---IGH DC模式测试电机程...
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...

Ethercat系列---IGH DC模式测试电机程...
nanshanaa: 博主，在哪里可以获取到当前电机是否处于使能状态啊

QT笔记---QT程序打包发布 (linuxdeploy...
中华田园巨龙 啊，这就不清楚了，我的方法对你不适用的话，需要你自己在摸索

QT笔记---QT程序打包发布 (linuxdeploy...
Oliver_zrz: 请问没有libqgtk2.so和libqgtk2st yle.so，按上面的操作并没有解决问题是...

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

uboot笔记---SF: unrecognized JEDEC id bytes: 20, ba, 20 解决

Nor Flash笔记---看一图看懂并学会计算Flash大小

QT编译BUG解决---error: invalid conversion from 'Window' [aka 'long unsigned int'] to 'EGLNativeWindowType

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

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)

```
1 #include <errno.h>
2 #include <signal.h>
3 #include <stdio.h>
4 #include <string.h>
5 #include <sys/resource.h>
6 #include <sys/time.h>
7 #include <sys/types.h>
8 #include <unistd.h>
9 #include <time.h>
10 #include <sys/mman.h>
11 #include <malloc.h>
12 #include <sched.h> /* sched_setscheduler() */
13
14 /******
15
16 #include "ecrt.h"
17
18 /******
19
20 // Application parameters
21 #define FREQUENCY 1000
22 #define CLOCK_TO_USE CLOCK_MONOTONIC
23 #define MEASURE_TIMING
24
25 #define TARGET_POSITION 0 /*target position*/
26 #define CYCLIC_POSITION 8 /*Operation mode for 0x6060:0*/
27
28 /******
29
30 #define NSEC_PER_SEC (1000000000L)
31 #define PERIOD_NS (NSEC_PER_SEC / FREQUENCY) /*本次设置周期PERIOD_NS为1ms*/
32
33 #define DIFF_NS(A, B) (((B).tv_sec - (A).tv_sec) * NSEC_PER_SEC + \
34 (B).tv_nsec - (A).tv_nsec)
35
36 #define TIMESPEC2NS(T) ((uint64_t) (T).tv_sec * NSEC_PER_SEC + (T).tv_nsec)
37
38 /******
39
40 // EtherCAT
41 static ec_master_t *master = NULL;
42 static ec_master_state_t master_state = {};
43
44 static ec_domain_t *domain1 = NULL;
45 static ec_domain_state_t domain1_state = {};
46
47 static ec_slave_config_t *sc = NULL;
48 static ec_slave_config_state_t sc_state = {};
49
50 /******
51
52 // process data
53 static uint8_t *domain1_pd = NULL;
54
55 #define PANASONIC 0,0 /*EtherCAT address on the bus*/
56
57 #define VID_PID 0x000066F,0x515050a1 /*Vendor ID, product code*/
58
59 /*Offsets for PDO entries*/
60 static struct{
61 unsigned int ctrl_word;
62 unsigned int operation_mode;
63 unsigned int target_position;
64 unsigned int touch_probe_function;
65 unsigned int error_code;
66 unsigned int status_word;
67 unsigned int modes_of_operation_display;
68 unsigned int position_actual_value;
69 unsigned int touch_probe_status;
70 unsigned int touch_probe_pos1_pos_value;
71 unsigned int following_error_actual_value;
72 unsigned int digital_inputs;
73 }offset;
74
75 const static ec_pdo_entry_reg_t domain1_regs[] = {
76 {PANASONIC, VID_PID, 0x6040, 0, 0,offset.ctrl_word},
77 {PANASONIC, VID_PID, 0x6060, 0, 0,offset.operation_mode },
78 {PANASONIC, VID_PID, 0x607A, 0, 0,offset.target_position},
79 {PANASONIC, VID_PID, 0x60B8, 0, 0,offset.touch_probe_function},
80 {PANASONIC, VID_PID, 0x603F, 0, 0,offset.error_code},
81 {PANASONIC, VID_PID, 0x6041, 0, 0,offset.status_word},
82 {PANASONIC, VID_PID, 0x6061, 0, 0,offset.modes_of_operation_display},
83 {PANASONIC, VID_PID, 0x6064, 0, 0,offset.position_actual_value},
84 {PANASONIC, VID_PID, 0x60B9, 0, 0,offset.touch_probe_status},
85 {PANASONIC, VID_PID, 0x60BA, 0, 0,offset.touch_probe_pos1_pos_value},
86 {PANASONIC, VID_PID, 0x60F4, 0, 0,offset.following_error_actual_value},
87 {PANASONIC, VID_PID, 0x60FD, 0, 0,offset.digital_inputs},
88 {}
89 };
90
91 /******
92 /*Config PD0s*/
93 static ec_pdo_entry_info_t device_pdo_entries[] = {
94 /*RxPdo 0x1600*/
95 {0x6040, 0x00, 16},
96 {0x6060, 0x00, 8 },
97 {0x607A, 0x00, 32},
98 {0x60B8, 0x00, 16},
99 /*TxPdo 0x1A00*/
100 {0x603F, 0x00, 16},
101 {0x6041, 0x00, 16},
102 {0x6061, 0x00, 8},
103 {0x6064, 0x00, 32},
104 {0x60B9, 0x00, 16},
105 {0x60BA, 0x00, 32},
106 {0x60F4, 0x00, 32},
107 {0x60FD, 0x00, 32},
108 };
109
110 static ec_pdo_info_t device_pdos[] = {
111 /*RxPdo
112 {0x1600, 4, device_pdo_entries + 0 },
113 /*TxPdo
114 {0x1A00, 8, device_pdo_entries + 4 }
115 };
```

分类专栏

1 订阅

5 篇文章

订阅专栏

```

116 static ec_sync_info_t device_syncs[] = {
117     { 0, EC_DIR_OUTPUT, 0, NULL, EC_WD_DISABLE },
118     { 1, EC_DIR_INPUT, 0, NULL, EC_WD_DISABLE },
119     { 2, EC_DIR_OUTPUT, 1, device_pdos + 0, EC_WD_DISABLE },
120     { 3, EC_DIR_INPUT, 1, device_pdos + 1, EC_WD_DISABLE },
121     { 0xFF }
122 };
123
124 static unsigned int counter = 0;
125 static unsigned int blink = 0;
126 static unsigned int sync_ref_counter = 0;
127 const struct timespec cyclotime = { 0, PERIOD_NS };
128
129
130 /*****
131
132 struct timespec timespec_add(struct timespec time1, struct timespec time2)
133 {
134     struct timespec result;
135
136     if ((time1.tv_nsec + time2.tv_nsec) >= NSEC_PER_SEC) {
137         result.tv_sec = time1.tv_sec + time2.tv_sec + 1;
138         result.tv_nsec = time1.tv_nsec + time2.tv_nsec - NSEC_PER_SEC;
139     } else {
140         result.tv_sec = time1.tv_sec + time2.tv_sec;
141         result.tv_nsec = time1.tv_nsec + time2.tv_nsec;
142     }
143
144     return result;
145 }
146
147 *****/
148 void check_domain1_state(void)
149 {
150     ec_domain_state_t ds;
151
152     ecrt_domain_state(domain1, &ds);
153
154     if (ds.working_counter != domain1_state.working_counter)
155         printf("Domain1: WC %u.\n", ds.working_counter);
156     if (ds.wc_state != domain1_state.wc_state)
157         printf("Domain1: State %u.\n", ds.wc_state);
158
159     domain1_state = ds;
160 }
161
162 /*****
163
164 void check_master_state(void)
165 {
166     ec_master_state_t ms;
167
168     ecrt_master_state(master, &ms);
169
170     if (ms.slaves_responding != master_state.slaves_responding)
171         printf("%u slave(s).\n", ms.slaves_responding);
172     if (ms.al_states != master_state.al_states)
173         printf("AL states: 0x%02X.\n", ms.al_states);
174     if (ms.link_up != master_state.link_up)
175         printf("Link is %s.\n", ms.link_up ? "up" : "down");
176
177     master_state = ms;
178 }
179
180 *****/
181 void check_slave_config_states(void)
182 {
183     ec_slave_config_state_t s;
184     ecrt_slave_config_state(sc, &s);
185     if (s.al_state != sc_state.al_state)
186     {
187         printf("slave: State 0x%02X.\n", s.al_state);
188     }
189     if (s.online != sc_state.online)
190     {
191         printf("slave: %s.\n", s.online ? "online" : "offline");
192     }
193     if (s.operational != sc_state.operational)
194     {
195         printf("slave: %soperational.\n", s.operational ? "" : "Not ");
196     }
197     sc_state = s;
198 }
199
200 /*****
201
202 void cyclic_task()
203 {
204     static uint16_t command=0x004F; //用来帮助判断状态字值的
205     uint16_t status; //读取伺服状态
206
207     struct timespec wakeupTime, time;
208     #ifdef MEASURE_TIMING
209     struct timespec startTime, endTime, lastStartTime = {};
210     uint32_t period_ns = 0, exec_ns = 0, latency_ns = 0,
211         latency_min_ns = 0, latency_max_ns = 0,
212         period_min_ns = 0, period_max_ns = 0,
213         exec_min_ns = 0, exec_max_ns = 0;
214     #endif
215
216     // get current time
217     clock_gettime(CLOCK_TO_USE, &wakeupTime);
218
219     while(1) {
220         wakeupTime = timespec_add(wakeupTime, cyclotime);
221         clock_nanosleep(CLOCK_TO_USE, TIMER_ABSTIME, &wakeupTime, NULL);
222
223         // Write application time to master
224         //
225         // It is a good idea to use the target time (not the measured time) as
226         // application time, because it is more stable.
227         //
228         ecrt_master_application_time(master, TIMESPEC2NS(wakeupTime));
229
230     #ifdef MEASURE_TIMING
231         clock_gettime(CLOCK_TO_USE, &startTime);
232         latency_ns = DIFF_NS(wakeupTime, startTime);
233         period_ns = DIFF_NS(lastStartTime, startTime);
234         exec_ns = DIFF_NS(lastStartTime, endTime);
235         lastStartTime = startTime;
236
237         if (latency_ns > latency_max_ns) {
238             latency_max_ns = latency_ns;
239         }
240         if (latency_ns < latency_min_ns) {
241             latency_min_ns = latency_ns;
242         }
243         if (period_ns > period_max_ns) {
244             period_max_ns = period_ns;
245         }
246         if (period_ns < period_min_ns) {
247             period_min_ns = period_ns;
248         }
249     }
250 }

```

```

251     if (exec_ns > exec_max_ns) {
252         exec_max_ns = exec_ns;
253     }
254     if (exec_ns < exec_min_ns) {
255         exec_min_ns = exec_ns;
256     }
257 #endif
258
259     // receive process data
260     ecrt_master_receive(master);
261     ecrt_domain_process(domain1);
262
263     // check process data state (optional)
264     check_domain1_state();
265
266     if (counter) {
267         counter--;
268     } else { // do this at 1 Hz
269         counter = FREQUENCY;
270     }
271
272     // check for master state (optional)
273     check_master_state();
274     // check for slave configuration state(s)
275     check_slave_config_states();
276
277 #ifdef MEASURE_TIMING
278     // output timing stats
279     printf("period    %10u ... %10u\n",
280           period_min_ns, period_max_ns);
281     printf("exec      %10u ... %10u\n",
282           exec_min_ns, exec_max_ns);
283     printf("latency   %10u ... %10u\n",
284           latency_min_ns, latency_max_ns);
285     period_max_ns = 0;
286     period_min_ns = 0xffffffff;
287     exec_max_ns = 0;
288     exec_min_ns = 0xffffffff;
289     latency_max_ns = 0;
290     latency_min_ns = 0xffffffff;
291 #endif
292
293     // calculate new process data
294     blink = !blink;
295 }
296 /*Read state*/
297 status = EC_READ_U16(domain1_pd + offset.status_word); //读取状态字
298
299 // write process data
300 //DS402 CANOpen over EtherCAT status machine
301 if( (status & command) == 0x0040 )
302 {
303     EC_WRITE_U16(domain1_pd + offset.ctrl_word, 0x0006 );
304     EC_WRITE_S8(domain1_pd + offset.operation_mode, CYCLIC_POSITION);
305     //set control mode
306     command = 0x006F;
307 }
308
309 else if( (status & command) == 0x0021)
310 {
311     EC_WRITE_U16(domain1_pd + offset.ctrl_word, 0x0007 );
312     command = 0x006F;
313 }
314
315 else if( (status & command) == 0x0023)
316 {
317     EC_WRITE_U16(domain1_pd + offset.ctrl_word, 0x000f );
318     command = 0x006F;
319 }
320 //operation enabled
321
322 else if( (status & command) == 0x0027)
323 {
324     EC_WRITE_U16(domain1_pd + offset.ctrl_word, 0x001f );
325 }
326
327 if (sync_ref_counter) {
328     sync_ref_counter--;
329 } else {
330     sync_ref_counter = 1; // sync every cycle
331
332     clock_gettime(CLOCK_TO_USE, &time);
333     ecrt_master_sync_reference_clock_to(master, TIMESPEC2NS(time));
334 }
335 ecrt_master_sync_slave_clocks(master);
336
337 // send process data
338 ecrt_domain_queue(domain1);
339 ecrt_master_send(master);
340
341 #ifdef MEASURE_TIMING
342     clock_gettime(CLOCK_TO_USE, &endTime);
343 #endif
344 }
345
346 //*****
347
348 int main(int argc, char **argv)
349 {
350     if (mlockall(MCL_CURRENT | MCL_FUTURE) == -1) {
351         perror("mlockall failed");
352         return -1;
353     }
354
355     master = ecrt_request_master(0);
356     if (!master)
357         return -1;
358
359     domain1 = ecrt_master_create_domain(master);
360     if (!domain1)
361         return -1;
362
363     // Create configuration for bus coupler
364     sc = ecrt_master_slave_config(master, PANASONIC, VID_PID);
365     if (!sc)
366         return -1;
367
368     printf("Configuring PDOs...\n");
369     if (ecrt_slave_config_pdos(sc, EC_END, device_syncs))
370     {
371         fprintf(stderr, "Failed to configure slave PDOs!\n");
372         exit(EXIT_FAILURE);
373     }
374     else
375     {
376         printf("**Success to configuring slave PDOs*\n");
377     }
378
379     if (ecrt_domain_reg_pdo_entry_list(domain1, domain1_regs))
380     {
381         fprintf(stderr, "PDO entry registration failed!\n");
382         exit(EXIT_FAILURE);
383     }
384
385     // configure SYNC signals for this slave

```

```

387 ecrt_slave_config_dc(sc, 0x0300, PERIOD_NS, 4400000, 0, 0);
388 printf("Activating master...\n");
389 if (ecrt_master_activate(master))
390     return -1;
391
392 if (!(domain1_pd = ecrt_domain_data(domain1))) {
393     return -1;
394 }
395
396 /* Set priority */
397
398 struct sched_param param = {};
399 param.sched_priority = sched_get_priority_max(SCHED_FIFO);
400
401 printf("Using priority %i.", param.sched_priority);
402 if (sched_setscheduler(0, SCHED_FIFO, &param) == -1) {
403     perror("sched_setscheduler failed");
404 }
405
406 printf("Starting cyclic function.\n");
407 cyclic_task();
408
409 return 0;
410 }
411
412 /******

```

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KGH主站

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

2. 查看电机的信息

```
1 | sudo ethercat cstruct -a 0
```

记下电机VendorID，ProductID，PDO信息

3. 自定义代码

(1) 周期修改

```

/*****
// Application-parameters
#define FREQUENCY 1000 //频率 单位HZ
#define CLOCK_TO_USE CLOCK_MONOTONIC
#define MEASURE_TIMING
#define TARGET_POSITION 0 /*target position*/
#define CYCLIC_POSITION 8 /*Operation mode for 0x6060:0*/

```

(2) VendorID，ProductID修改

```

#define PANASONIC 0,0 /*EtherCAT address on the bus*/
#define VID_PID 0x0000060F,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, 0x6070, 0, &offset.target_position},
    {PANASONIC, VID_PID, 0x6080, 0, &offset.touch_probe_function},
    {PANASONIC, VID_PID, 0x608F, 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, 0x6089, 0, &offset.touch_probe_status},
    {PANASONIC, VID_PID, 0x608A, 0, &offset.touch_probe_pos1_pos_value},
    {PANASONIC, VID_PID, 0x60F4, 0, &offset.following_error_actual_value},
    {PANASONIC, VID_PID, 0x60F0, 0, &offset.digital_inputs},
};

/*****Config PDOs*/
static ec_pdo_entry_info_t device_pdo_entries[] = {
    /*RxPdo 0x1600*/
    {0x6040, 0x00, 16},
    {0x6060, 0x00, 8},
    {0x6070, 0x00, 32},
    {0x6080, 0x00, 16},
    /*TxPdo 0x1800*/
    {0x603F, 0x00, 16},
    {0x6041, 0x00, 16},
    {0x6061, 0x00, 8},
    {0x6064, 0x00, 32},
    {0x6089, 0x00, 16},
    {0x608A, 0x00, 32},
    {0x60F4, 0x00, 32},
    {0x60F0, 0x00, 32},
};

static ec_pdo_info_t device_pdos[] = {
    /*RxPdo
    {0x1600, 4, device_pdo_entries + 0},
    //TxPdo
    {0x1800, 8, device_pdo_entries + 4}
    */
};

```

4. 编译，运行

```

1 | make
2 | 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

```

1 | int32_t actual_position =0;
2 | int32_t target_position =0;
3 |
4 | ...
5 | ...
6 |
7 | if (sync_ref_counter) {
8 |     sync_ref_counter--;
9 | } else {
10 |     sync_ref_counter = 1; // sync every cycle
11 |
12 |     clock_gettime(CLOCK_TO_USE, &time);
13 |     ecrt_master_sync_reference_clock_to(master, TIMESPEC2NS(time));
14 |
15 |     if( (status & command) == 0x0027) //确认为OP状态，电机使能
16 |     {
17 |         actual_position = EC_READ_S32(domain1_pd + offset.position_actual_value); //读取当前位置
18 |         target_position = actual_position + 10; //把当前位置+10 赋值给下一目标位置
19 |         EC_WRITE_S32(domain1_pd + offset.target_position, target_position); //把目标位置写入对象字典
20 |     }
21 |
22 | }

```

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,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限





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 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 条评论



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/scynk/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的博文 5274
发现SYNC0信号的周期跳变量大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。	chn512的博文 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_ioct! 基于ioct!接口的sanyo电机测试例子 mytest/test_torque_tec_ioct! 基于ioct!...	
IGH EtherCAT应用层控制电机代码	qn_43530144的博文 1828
在主站配置好之后,连接从站,我用的是雷赛伺服,对于大部分来说改个pid, vid, 应该就可以用,这个是用的是pvl模式,应该还是容易懂的,我把一些用...	
EtherCAT使用与解析_关于DC	lswdoyy的博文 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的博文	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)	chn512的博文 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)	chn512的博文 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, 过段时间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的博文 1333
需要对应打了实时补丁的内核,可以看我前一篇博文 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	PI_sunyang的博文 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 8157
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ethercat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...	
©2020 CSDN 皮肤主题: 精致技术 设计师:CSDN官方博客 返回首页	



中华田园巨龙

关注



1



34



18



专栏目录