```
**********************************************************
   28034 Thu Jan 31 12:56:20 2013
new/usr/src/uts/common/io/usb/usba/hcdi.c
3503 usba doesn't initialize cc_no_resources, causes gibberish output in kstat
Reviewed by: Garrett D'Amore <garrett@damore.org>
Reviewed by: Richard Lowe <richlowe@richlowe.net>
**********************************************************
______unchanged_portion_omitted_


 266 /*
 267  * Allocate the hotplug kstats structure
 268  */
 269 void
 270 usba_hcdi_create_stats(usba_hcdi_t *hcdi, int instance)
 271 {
 272         char                    kstatname[KSTAT_STRLEN];
 273         const char              *dname = ddi_driver_name(hcdi->hcdi_dip);
 274         hcdi_hotplug_stats_t    *hsp;
 275         hcdi_error_stats_t      *esp;

 277         if (HCDI_HOTPLUG_STATS(hcdi) == NULL) {
 278                 (void) snprintf(kstatname, KSTAT_STRLEN, "%s%d,hotplug",
 279                     dname, instance);
 280                 HCDI_HOTPLUG_STATS(hcdi) = kstat_create("usba", instance,
 281                     kstatname, "usb_hotplug", KSTAT_TYPE_NAMED,
 282                     sizeof (hcdi_hotplug_stats_t) / sizeof (kstat_named_t),
 283                     KSTAT_FLAG_PERSISTENT);

 285                 if (HCDI_HOTPLUG_STATS(hcdi) == NULL) {

 287                         return;
 288                 }

 290                 hsp = HCDI_HOTPLUG_STATS_DATA(hcdi);
 291                 kstat_named_init(&hsp->hcdi_hotplug_total_success,
 292                     "Total Hotplug Successes", KSTAT_DATA_UINT64);
 293                 kstat_named_init(&hsp->hcdi_hotplug_success,
 294                     "Hotplug Successes", KSTAT_DATA_UINT64);
 295                 kstat_named_init(&hsp->hcdi_hotplug_total_failure,
 296                     "Hotplug Total Failures", KSTAT_DATA_UINT64);
 297                 kstat_named_init(&hsp->hcdi_hotplug_failure,
 298                     "Hotplug Failures", KSTAT_DATA_UINT64);
 299                 kstat_named_init(&hsp->hcdi_device_count,
 300                     "Device Count", KSTAT_DATA_UINT64);

 302                 HCDI_HOTPLUG_STATS(hcdi)->ks_private = hcdi;
 303                 HCDI_HOTPLUG_STATS(hcdi)->ks_update = nulldev;
 304                 kstat_install(HCDI_HOTPLUG_STATS(hcdi));
 305         }

 307         if (HCDI_ERROR_STATS(hcdi) == NULL) {
 308                 (void) snprintf(kstatname, KSTAT_STRLEN, "%s%d,error",
 309                     dname, instance);
 310                 HCDI_ERROR_STATS(hcdi) = kstat_create("usba", instance,
 311                     kstatname, "usb_errors", KSTAT_TYPE_NAMED,
 312                     sizeof (hcdi_error_stats_t) / sizeof (kstat_named_t),
 313                     KSTAT_FLAG_PERSISTENT);

 315                 if (HCDI_ERROR_STATS(hcdi) == NULL) {

 317                         return;
 318                 }

 320                 esp = HCDI_ERROR_STATS_DATA(hcdi);
 321                 kstat_named_init(&esp->cc_crc, "CRC Errors", KSTAT_DATA_UINT64);
```



```
 322                 kstat_named_init(&esp->cc_bitstuffing,
 323                     "Bit Stuffing Violations", KSTAT_DATA_UINT64);
 324                 kstat_named_init(&esp->cc_data_toggle_mm,
 325                     "Data Toggle PID Errors", KSTAT_DATA_UINT64);
 326                 kstat_named_init(&esp->cc_stall,
 327                     "Endpoint Stalls", KSTAT_DATA_UINT64);
 328                 kstat_named_init(&esp->cc_dev_not_resp,
 329                     "Device Not Responding", KSTAT_DATA_UINT64);
 330                 kstat_named_init(&esp->cc_pid_checkfailure,
 331                     "PID Check Bit Errors", KSTAT_DATA_UINT64);
 332                 kstat_named_init(&esp->cc_unexp_pid,
 333                     "Invalid PID Errors", KSTAT_DATA_UINT64);
 334                 kstat_named_init(&esp->cc_data_overrun,
 335                     "Data Overruns", KSTAT_DATA_UINT64);
 336                 kstat_named_init(&esp->cc_data_underrun,
 337                     "Data Underruns", KSTAT_DATA_UINT64);
 338                 kstat_named_init(&esp->cc_buffer_overrun,
 339                     "Buffer Overruns", KSTAT_DATA_UINT64);
 340                 kstat_named_init(&esp->cc_buffer_underrun,
 341                     "Buffer Underruns", KSTAT_DATA_UINT64);
 342                 kstat_named_init(&esp->cc_timeout,
 343                     "Command Timed Out", KSTAT_DATA_UINT64);
 344                 kstat_named_init(&esp->cc_not_accessed,
 345                     "Not Accessed By Hardware", KSTAT_DATA_UINT64);
 346                 kstat_named_init(&esp->cc_no_resources,
 347                     "No Resources", KSTAT_DATA_UINT64);
 348 #endif /* ! codereview */
 349                 kstat_named_init(&esp->cc_unspecified_err,
 350                     "Unspecified Error", KSTAT_DATA_UINT64);
 351                 kstat_named_init(&esp->cc_stopped_polling,
 352                     "Stopped Polling", KSTAT_DATA_UINT64);
 353                 kstat_named_init(&esp->cc_pipe_closing,
 354                     "Pipe Closing", KSTAT_DATA_UINT64);
 355                 kstat_named_init(&esp->cc_pipe_reset,
 356                     "Pipe Reset", KSTAT_DATA_UINT64);
 357                 kstat_named_init(&esp->cc_not_supported,
 358                     "Command Not Supported", KSTAT_DATA_UINT64);
 359                 kstat_named_init(&esp->cc_flushed,
 360                     "Request Flushed", KSTAT_DATA_UINT64);
 346 #ifdef  NOTYETNEEDED
 347                 kstat_named_init(&esp->hcdi_usb_failure,
 348                     "USB Failure", KSTAT_DATA_UINT64);
 349                 kstat_named_init(&esp->hcdi_usb_no_resources,
 350                     "No Resources", KSTAT_DATA_UINT64);
 351                 kstat_named_init(&esp->hcdi_usb_no_bandwidth,
 352                     "No Bandwidth", KSTAT_DATA_UINT64);
 353                 kstat_named_init(&esp->hcdi_usb_pipe_reserved,
 354                     "Pipe Reserved", KSTAT_DATA_UINT64);
 355                 kstat_named_init(&esp->hcdi_usb_pipe_unshareable,
 356                     "Pipe Unshareable", KSTAT_DATA_UINT64);
 357                 kstat_named_init(&esp->hcdi_usb_not_supported,
 358                     "Function Not Supported", KSTAT_DATA_UINT64);
 359                 kstat_named_init(&esp->hcdi_usb_pipe_error,
 360                     "Pipe Error", KSTAT_DATA_UINT64);
 361                 kstat_named_init(&esp->hcdi_usb_pipe_busy,
 362                     "Pipe Busy", KSTAT_DATA_UINT64);
 363 #endif

 362                 HCDI_ERROR_STATS(hcdi)->ks_private = hcdi;
 363                 HCDI_ERROR_STATS(hcdi)->ks_update = nulldev;
 364                 kstat_install(HCDI_ERROR_STATS(hcdi));
 365         }
 366 }
______unchanged_portion_omitted_
```

```
**********************************************************
    9123 Thu Jan 31 12:56:20 2013
new/usr/src/uts/common/sys/usb/usba/hcdi.h
3503 usba doesn't initialize cc_no_resources, causes gibberish output in kstat
Reviewed by: Garrett D'Amore <garrett@damore.org>
Reviewed by: Richard Lowe <richlowe@richlowe.net>
**********************************************************
______unchanged_portion_omitted_

 303 /*
 304  * USB error kstats named structure
 305  */
 306 typedef struct hcdi_error_stats {
 307         /* transport completion codes */
 308         struct kstat_named      cc_crc;
 309         struct kstat_named      cc_bitstuffing;
 310         struct kstat_named      cc_data_toggle_mm;
 311         struct kstat_named      cc_stall;
 312         struct kstat_named      cc_dev_not_resp;
 313         struct kstat_named      cc_pid_checkfailure;
 314         struct kstat_named      cc_unexp_pid;
 315         struct kstat_named      cc_data_overrun;
 316         struct kstat_named      cc_data_underrun;
 317         struct kstat_named      cc_buffer_overrun;
 318         struct kstat_named      cc_buffer_underrun;
 319         struct kstat_named      cc_timeout;
 320         struct kstat_named      cc_not_accessed;
 321         struct kstat_named      cc_no_resources;
 322         struct kstat_named      cc_unspecified_err;
 323         struct kstat_named      cc_stopped_polling;
 324         struct kstat_named      cc_pipe_closing;
 325         struct kstat_named      cc_pipe_reset;
 326         struct kstat_named      cc_not_supported;
 327         struct kstat_named      cc_flushed;

 329 #ifdef  NOTYETNEEDED
 330         /* USBA function return values */
 331         struct kstat_named      hcdi_usb_failure;
 332         struct kstat_named      hcdi_usb_no_resources;
 333         struct kstat_named      hcdi_usb_no_bandwidth;
 334         struct kstat_named      hcdi_usb_pipe_reserved;
 335         struct kstat_named      hcdi_usb_pipe_unshareable;
 336         struct kstat_named      hcdi_usb_not_supported;
 337         struct kstat_named      hcdi_usb_pipe_error;
 338         struct kstat_named      hcdi_usb_pipe_busy;
 339 #endif
 328 } hcdi_error_stats_t;
______unchanged_portion_omitted_
```