Concurrent Real-Time/esd esdcan-pcie402 CAN Driver/API Release 3.10.4 for the CAN-PCIe/402
Installation on RedHawk Linux

WCS-ES-CAN-402

July 25, 2018

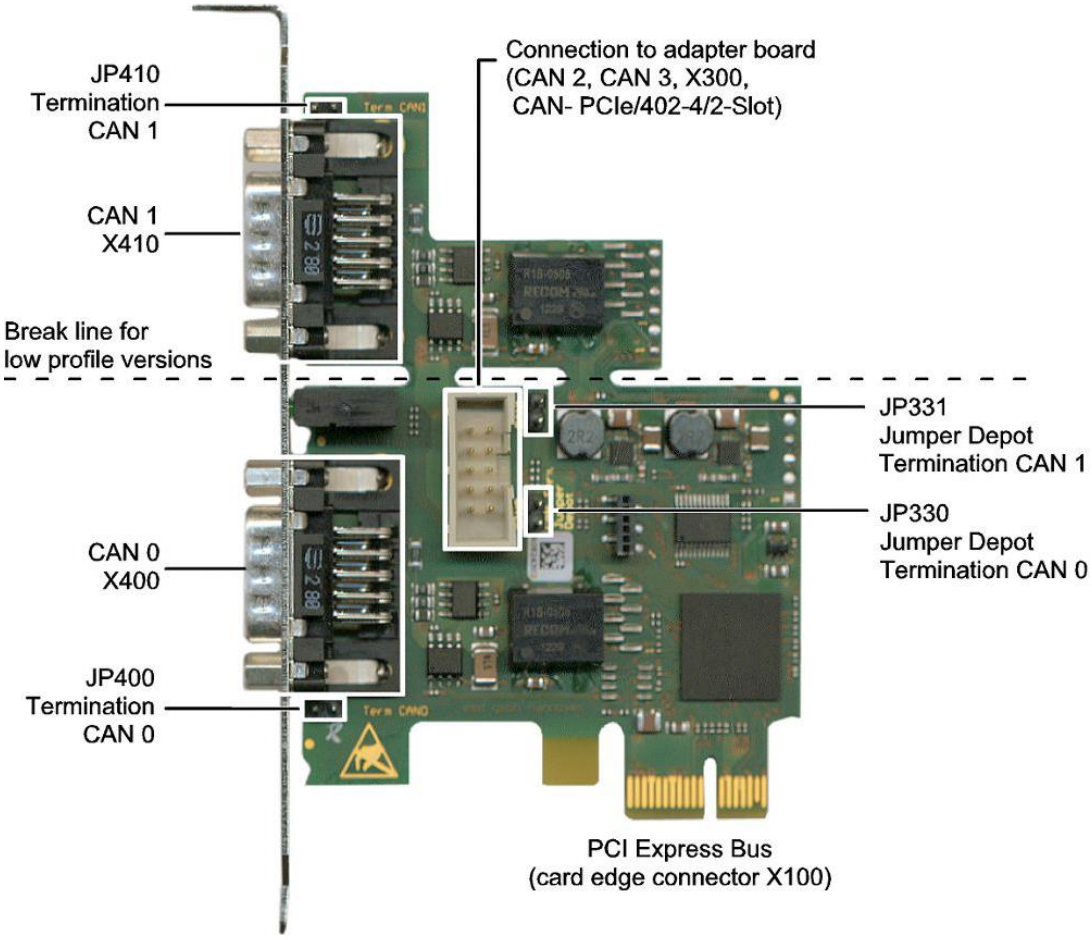
concurrent
REAL-TIME

This page intentionally blank.

1. Introduction

This document assists the user in installing the esd electronic system design gmbh ***esdcan-pcie402*** device driver and CAN-API version 3.10.4 software along with **cantest** test program on a RedHawk Linux based system for use with the PCIe-CAN/402 CAN bus card. This software will be referred to as the ***esdcan-pcie402*** driver in this document.

The PCIe-CAN/402 card provides two (optionally four) CAN (ISO 11898-2) net interfaces.



CAN-PCIe/402

The vendor releases distinct 32-bit and 64-bit versions of this software; Concurrent Real-Time bundles both versions into a single RPM package which will select the correct version matching your system OS when the ***esdcan-pcie402*** software is installed.

2. Hardware Installation

Versions with four CAN ports may require a second slot adjacent to the main card slot for the daughter card containing the electronics for the third and fourth ports. These two cards are interconnected with a small ribbon cable.

Normally the PCIe-CAN/402 hardware will have been installed by Concurrent Real-Time as part of system integration. If this product is being installed in the field, please refer to page 12 of the **CAN-PCIe402_Hardware_en_10.pdf** document included with this package.

<u>Caution</u>: when installing the CAN-PCIe/402 card ensure the computer is powered off and the machine's power cord is disconnected. Please observe electrostatic discharge precautions such as the use of a grounding strap.

After installation, you may verify the board is properly installed by issuing the commands below, which if the board is present will provide output *similar* to that shown below – the *nn:nn.n* bus enumeration numbers (09:00.0 in the example below) may differ, for example:

```
=== as user or root ===
# lspci -v -d 12fe:0402

09:00.0 CANBUS: ESD Electronic System Design GmbH Device 0402 (rev 01)
        Subsystem: ESD Electronic System Design GmbH Device 0402
        Flags: bus master, fast devsel, latency 0, IRQ 88
        Memory at c0100000 (32-bit, non-prefetchable) [size=128K]
        Capabilities: <access denied>
```

If output like this is not displayed by the **lspci** command, the device has not been properly installed in the system.

3. Software Installation

Before extracting the source code and building the driver and other related products, be sure the kernel on which you wish to use this software is the one currently running. If the kernel you wish to use is not currently running, reboot to the kernel you desire to use before proceeding.

The ***esdcan-pcie402*** driver is supplied in RPM format on CDROM. Please read the following carefully before proceeding with the installation.

Caution:

Before installing the software, the kernel build environment **must** be set up and match the current OS kernel you are using. If you are running one of the preconfigured kernels supplied by Concurrent Real-Time and have not previously done so, run the following commands while logged in as the root user before installing the driver software:

```
# cd /lib/modules/`uname -r`/build
# ./ccur-config -c -n
```

If you have built and are running a customized kernel configuration the kernel build environment should already have been set up when that custom kernel was built.

This requirement is enforced by the RPM package. If **ccur-config** has not been previously run on your system, the RPM will generate a detailed error message and the installation will fail.

To install the ***esdcan-pcie402*** package, load the CD-ROM installation media and issue the following commands as the root user. The system should auto-mount the CD-ROM to a mount point in the **/media** or **/run/media** directory based on the CD-ROM's volume label – in this case, ***esdcan-pcie402***. The example's ***[user_name]*** may be ***root*** or the logged-in user. Then enter the following commands from a shell window:

```
=== as root ===
      --- on RedHawk 6.5 and below ---
# cd /media/esdcan-pcie402
      --- or on RedHawk 7.0 and above ---
# cd /run/media/[user_name]/esdcan-pcie402
      --- or on Ubuntu RedHawk ---
# cd /media/[user_name]/esdcan-pcie402

# rpm -ivh esdcan-pcie402_RedHawk_driver*.rpm   (on an RPM based system)
      --- or ---
# dpkg -i esdcan-pcie402_RedHawk_driver*.deb   (on a Debian based system)

# cd /
# eject
```

The ***esdcan-pcie402*** driver files will be copied into the **/usr/local/CCRT/drivers/esdcan-pcie402** directory from the CDROM drive, and the driver and test applications will all be built from source. The driver and driver startup scripts will be installed. These scripts automatically load the driver during system bootup and create device nodes. These scripts can also be invoked manually via the **service/systemctl** commands. Though this dynamically loadable driver can be loaded manually with the ***insmod*** or ***modprobe*** command once it has been built, and can be unloaded manually by issuing the ***rmmod*** command, it is recommended that the startup scripts be used instead.

The source tree for this product contains an enhanced Makefile with additional targets to permit building and installing the driver and other components in an easier, more automated fashion. Manual intervention should not be necessary in most cases. The top-level CCRT_ReadMe file explains the new targets added to Concurrent Real-Time's version of the top-level Makefile.

Building the *esdcan-pcie402* driver for other kernels

RedHawk Linux is provided with three different kernels: the 'standard, 'trace', and 'debug' variants. The ***esdcan-pcie402.ko*** kernel driver is initially built for the kernel variant or 'flavor' that was booted when the RPM was installed. To build the driver for a different kernel variant:

First, ***reboot the system*** to the desired kernel variant. To rebuild and install just the loadable kernel module component of the ***esdcan-pcie402*** package:

```
== as root ==
# cd /lib/modules/`uname -r`/build
# ./ccur-config -c -n          [set up the proper kernel build environment]
# cd /usr/local/CCRT/drivers/esdcan-pcie402
# make clean
# make driverinstall           [compile and install loadable kernel module]
# make load                    [load the newly installed driver]
```

To recompile and reinstall the entire package, perform the following steps:

```
== as root ==
# cd /usr/local/CCRT/drivers/esdcan-pcie402
# make clean
# make                         [build objects and executables]
# make install                 [install loadable kernel module and API files]
```

Instructions on removing the software may be found in Section 6. Removing the *esdcan-pcie402* package.

4. Loading the *esdcan-pcie402* Driver

The ***esdcan-pcie402*** driver module is dynamically loadable and obtains its major number from the kernel during initialization. A driver initialization file installed as **/etc/rc.d/init.d/esdcan-pcie402** or **/usr/lib/systemd/system/esdcan-pcie402.service** will load the driver module and create the appropriate device node files when the system is booted to init levels 3 or 5.

To disable this feature, issue the following command as the root user:

```
      --- on RedHawk 6.5 and below ---
# /sbin/chkconfig --del esdcan-pcie402
      --- or on RedHawk 7.0 and above ---
# /bin/systemctl disable esdcan-pcie402
```

To re-enable auto-loading of the ***esdcan-pcie402*** driver, issue the following command:

```
      --- on RedHawk 6.5 and below ---
# /sbin/chkconfig --add esdcan-pcie402
      --- or on RedHawk 7.0 and above ---
# /bin/systemctl enable esdcan-pcie402
```

The **chkconfig/systemctl** command references special lines in the **init/service** file to control these actions.

An entry to /etc/blacklist.conf was added for **chkconfig**/**systemctl** commands to work properly. This entry disables the automatic loading of the driver when the kernel scans the PCI bus.

The ***esdcan-pcie402*** driver may also be manually loaded and unloaded via the **service/systemctl** script:

```
=== as root ===
      --- on RedHawk 6.5 and below ---
# /sbin/service esdcan-pcie402 start        - load the driver and create /dev/ files
# /sbin/service esdcan-pcie402 stop   - unload the driver and remove /dev/ files
      --- or on RedHawk 7.0 and above ---
# /bin/systemctl start esdcan-pcie402  - load the driver and create /dev/ files
# /bin/systemctl stop esdcan-pcie402        - unload the driver and remove /dev/ files
```

The **service/systemctl** command will execute the same script that is run when the system is booted.

```
=== as root ===
# lsmod                          (this command will display the esdcan-pcie402 driver)
      --- or on RedHawk 6.5 and below ---
# /sbin/service esdcan-pcie402 status
      --- or on RedHawk 7.0 and above ---
# /bin/systemctl status esdcan-pcie402
```

The driver supports the installation of up to eight CAN-PCIe/402 boards. The Concurrent Real-Time authored startup scripts will use a ***esdcan-pcie402*** driver generated script - **/proc/bus/can/CAN_PCIe402/inodes** – to create only the required **/dev/can*** files

Messages generated by the driver when it is loaded should appear **in /var/log/messages** or in the output of the **dmesg** command. There will be other unrelated messages – the driver messages of interest will look like this:

```
# dmesg
…
esd CAN driver: Firmware-version = 0.0.42 (hex)
esd CAN driver: Hardware-version = 1.0.16 (hex)
esd CAN driver: Card = 0 Minor(s) = 0, 1
esd CAN driver: CAN_PCIe402
esd CAN driver: Baudrate not set
esd CAN driver: mode=0x00000000, major=237, verbose=0x00000001
esd CAN driver: Version 3.10.4 (11:04:04, Oct 13 2017)
esd CAN driver: successfully loaded
```

5. Testing and Usage

Communications API

Included with the driver is the **NTCAN-API** communications layer software. This software is documented in the included **documentation/can-api_part1_function_manual_45.pdf** file, and is copied to the system's **/usr/include** and **/usr/local/lib*** directories during installation.

The **ntcan.h** file defines the library interfaces that are implemented by the API software. Static and dynamically linked versions of the NTCAN-API library are provided. The **libntcan.a** library contains the interface routines described in **ntcan.h**, and is used to link applications statically. However, it is strongly recommended by esd that applications be dynamically linked using the provided **libntcan.so** library.

Both 64-bit and 32-bit versions of the **libntcan** library are supplied with this release. The 32-bit versions are installed in the **/usr/local/lib** directory, on 64-bit systems the 64-bit libraries are installed in **/usr/local/lib64** directory.

Cantest

The **cantest** sample program included with the ***esdcan-pcie402*** package is documented starting on page 170 of the **can-api_part1_function_manual_45.pdf** document, which may be found in the **documentation** directory. Source to this program is provided in the **example** directory. When invoked without command line arguments **cantest** will list the available CAN devices and a summary of available tests:

```
# cantest
CAN Test Rev 2.12.6  -- (c) 1997-2015 esd electronic system design gmbh

Available CAN-Devices:
Net   0: ID=CAN_PCIE402 (2 ports) Serial no.: GN001652
         Versions (hex): Lib=4.0.01 Drv=3.A.04 HW=1.0.16 FW=0.0.42 (0.0.00)
         Baudrate=7fffffff (Not set) Status=0000 Features=00000ffa
         Ctrl=esd Advanced CAN Core @ 80 MHz (Error Active / REC:0 / TEC:0)
         Transceiver=TI SN65HVD265
         TimestampFreq=80.000000 MHz Timestamp=00000000B72CEFE6
Net   1: ID=CAN_PCIE402 (2 ports) Serial no.: GN001652
         Versions (hex): Lib=4.0.01 Drv=3.A.04 HW=1.0.16 FW=0.0.42 (0.0.00)
         Baudrate=7fffffff (Not set) Status=0000 Features=00000ffa
         Ctrl=esd Advanced CAN Core @ 80 MHz (Error Active / REC:0 / TEC:0)
         Transceiver=TI SN65HVD265
         TimestampFreq=80.000000 MHz Timestamp=00000000B72D0236

Syntax: cantest test-Nr [net id-1st id-last count
        txbuf rxbuf txtout rxtout baud testcount data0 data1 ...]
Test   0:  canSend()
Test  20:  canSendT()
Test  50:  canSend() with incrementing ids
Test   1:  canWrite()
```

```
Test  21:  canWriteT()
Test  51:  canWrite() with incrementing ids
Test   2:  canTake()
Test  12:  canTake() with time-measurement for 10000 can-frames
Test  22:  canTakeT()
Test  32:  canTake() in Object-Mode
Test  42:  canTakeT() in Object-Mode
Test   3:  canRead()
Test  13:  canRead() with time-measurement for 10000 can-frames
Test  23:  canReadT()
Test   4:  canReadEvent()
Test  64:  Retrieve bus statistics (every tx timeout)
Test  74:  Reset bus statistics
Test  84:  Retrieve bitrate details (every tx timeout)
Test   5:  canSendEvent()
Test   8:  Create auto RTR object
Test   9:  Wait for RTR reply
Test  19:  Wait for RTR reply without text-output
Test 100:  Object Scheduling test
Test  -2:  Overview without syntax help
Test  -3:  Overview without syntax help but with feature flags details
```

The **cantest** application can identify the CAN nets and has a menu of tests that can verify correct operation of the ***esdcan-pcie402*** driver and **CAN-PCIe/402** hardware. A simple example would be to connect nets 0 and 1 with a loopback cable, and then issue the following two commands:

```
# cantest 3 0 &
# cantest 1 1
```

The first line starts a reader process on port (net) 0 and the second a sender process on port 1. The application should show the content of messages successfully sent/received, with an incremented field in each successive message. Note: the receiving process will run forever and must be manually terminated. It is probably best to put the above commands in an executable shell script to reduce error messages resulting from reading timeouts; a script will minimize the time between commands so the window for such timeouts is minimized.

The messages output when running this sequence will look something like this:

```
test=3 net=0 id-1st=0 id-last=0 count=1
test=1 net=1 id-1st=0 id-last=0 count=1
txbuf=10 rxbuf=100 txtout=1000 rxtout=5000 baudrate=2 (500000 baud)
txbuf=10 rxbuf=100 txtout=1000 rxtout=5000 baudrate=2 (500000 baud)
testcount=10
Duration=     0 msec Can-Messages=1
Duration=        0 msec Can-Messages=1
RX-ID=        0 (0x00000000) len=8 data= 00 00 00 00 01 00 00 00   [........]
Duration=     0 msec Can-Messages=1
Duration=        1 msec Can-Messages=1
RX-ID=        0 (0x00000000) len=8 data= 01 00 00 00 01 00 00 00   [........]
Duration=     0 msec Can-Messages=1
Duration=        0 msec Can-Messages=1
RX-ID=        0 (0x00000000) len=8 data= 02 00 00 00 01 00 00 00   [........]
Duration=     1 msec Can-Messages=1
Duration=        1 msec Can-Messages=1
RX-ID=        0 (0x00000000) len=8 data= 03 00 00 00 01 00 00 00   [........]
Duration=     0 msec Can-Messages=1
Duration=        0 msec Can-Messages=1
RX-ID=        0 (0x00000000) len=8 data= 04 00 00 00 01 00 00 00   [........]
Duration=     0 msec Can-Messages=1
Duration=        0 msec Can-Messages=1
RX-ID=        0 (0x00000000) len=8 data= 05 00 00 00 01 00 00 00   [........]
Duration=     1 msec Can-Messages=1
Duration=        1 msec Can-Messages=1
RX-ID=        0 (0x00000000) len=8 data= 06 00 00 00 01 00 00 00   [........]
```

```
Duration=     0 msec Can-Messages=1
Duration=        0 msec Can-Messages=1
RX-ID=        0 (0x00000000) len=8 data= 07 00 00 00 01 00 00 00   [........]
Duration=     0 msec Can-Messages=1
Duration=        0 msec Can-Messages=1
RX-ID=        0 (0x00000000) len=8 data= 08 00 00 00 01 00 00 00   [........]
Duration=     0 msec Can-Messages=1
Duration=        1 msec Can-Messages=1
RX-ID=        0 (0x00000000) len=8 data= 09 00 00 00 01 00 00 00   [........]
Test-Duration=4 msec
```

Note that the first byte of each message contains an incremented value from that of the preceding message. By default, the receiving copy of **cantest** reads forever and will begin to issue timeout messages after a few seconds after the sending process completes. You will have to manually kill this process.

A more intensive test sequence might be:

```
cantest 3 0 0 0 1 10 100 1000 5000 0 50000 &
cantest 1 1 0 0 1 10 100 1000 5000 0 50000
wait
sleep 1
cantest 3 1 0 0 1 10 100 1000 5000 0 50000 &
cantest 1 0 0 0 1 10 100 1000 5000 0 50000
wait
```

This sequence differs from the simple example by changing the baud rate from the default 500 Kbits/sec to 1 Mbit/second, changing the test count from the default 10 iterations to 50,000, and then repeating the test sequence reversing the transmit and receive ports.

CCRT_canio

The **CCRT_canio** is a CCRT supplied sample program. This program is used along with **cantest** in **Run_canio** script to test the board's functionality. For this script and test to function properly two ports need to be connected to each other with a loopback cable (i.e. nets 0 and 1).
The CCRT_canio test can be run as follows:

```
=== as root ===
# cd /usr/local/CCRT/drivers/esdcan-pcie402/CCRT
# make
# ./Run_canio PORTA PORTB #_OF_PASSES
```

Where PORTA and PORTB are the numbers of the ports connected (i.e. 0 and 1) and the #_OF_PASSES is the number of times to run the test.

6. Removing the *esdcan-pcie402* package

The ***esdcan-pcie402*** driver is a dynamically loadable driver that can be unloaded (and its device nodes removed) as follows:

```
=== as root ===
# cd /usr/local/CCRT/drivers/esdcan-pcie402
# make unload        (unload the driver from the kernel)
```

Alternately, one can use the **service** script:

```
=== as root ===
# /bin/systemctl stop esdcan-pcie402
```

To uninstall the ***esdcan-pcie402*** driver and its accessory files from the system directories:

```
=== as root ===
# cd /usr/local/CCRT/drivers/esdcan-pcie402
# make uninstall (remove the driver and related files from the system)
```

To completely remove the ***esdcan-pcie402*** package, do the following after the driver has been uninstalled:

```
=== as root ===
# rpm -e esdcan-pcie402
```

Warning: Any local changes made to files belonging to the ***esdcan-pcie402*** package, either those in the **/usr/local/CCRT/drivers/esdcan-pcie402** directory or elsewhere on the system, will be lost when the rpm -e command is executed.

---

7. Notes and Errata

The following changes and additions were made by Concurrent Real-Time to the original esd 3.10.4 release:

- The 32-bit and 64-bit releases were bundled into one RPM file; the proper version is selected and unpacked during the installation process.
- The top-level Makefile was heavily modified to add install, uninstall, driverinstall, load, and unload targets.
- A 'documentation' directory was created and populated.
- A top-level CCRT_ReadMe file was created.
- The driver source was modified to optionally use dynamically assigned device major numbers; the original version used a hard-coded value of 52.
- Driver init scripts were created; these load the driver requesting the device major number be dynamically assigned, create the **/dev** device nodes, pin the device interrupt level to a specified processor, and pin the driver's ***kesdcan_pcie402*** kernel thread to the same processor and raise its priority to use the real-time scheduler.

Original versions of any files modified by Concurrent Real-Time have been retained; these files have an '.ORIG' suffix appended to their names.

The warning message "could not find …/driver/.nucleus.o.cmd" generated when building the kernel driver module may safely be ignored. This is an artifact of the Linux kernel build mechanism's treatment of *object.o_shipped* binary files.

Included with this release of the **esdcan-pcie402** driver are several manuals in Adobe PDF format. These may be found in the **/usr/local/CCRT/drivers/esdcan-pcie402/documentation** directory and include:

- esdCAN-PCIe402_ReleaseNotes_r*.pdf *(this document)*
- can-api_part1_function_manual_45.pdf *(describes the programming API)*
- CAN-API_Part2_Installation_Manual_40.pdf *(installation on other platforms)*
- CAN-PCIe402_Datasheet_en_4.pdf *(hardware datasheet)*
- CAN-PCIe402_Hardware_en_10.pdf *(hardware installation, technical data)*

- CAN-Wiring.pdf *(whitepaper on CANbus wiring)*
- can2spec.pdf *(CANbus specification, from 1991)*
- intro-e.pdf *(CAN tutorial)*

It is ***strongly*** recommended by esd to link applications using the dynamic library.

The industry standard pin assignments of the CAN-PCIe/402's DSub9 male connectors are documented on page 20 of the **CAN-PCIe402_Hardware_en_10.pdf** document. The CAN-PCIe/402 provides optional internal termination to the CAN bus, by default it is not terminated, look at page 9 of the **CAN-PCIe402_Hardware_en_10.pdf** document for more information. Please note that a CAN bus is terminated at each end with a 120-ohm resistor. See the provided **CAN-Wiring.pdf** whitepaper for more details.

The ***esdcan-pcie402*** kernel driver starts a ***kesdcan_pcie402*** kernel thread to handle interrupt completion tasks. The design of the driver requires this thread be run on the same CPU as the card's interrupt routine. The Concurrent Real-Time authored driver startup script will pin any interrupt levels used by the PCIe/402 cards to a specific processor; the default is CPU 0. The processor used may be changed by editing the **/usr/lib/config/esdcan-pcie402_start** script and changing the value of the DPCCPU variable. By default, the ***kesdcan_pcie402*** process runs at interactive priorities; the **esdcan-pcie402_start** script will raise the priority of this process to use the real-time scheduler.

There are some known issues that you may encounter:

1. When loading the module, you may see a warning about a tainted kernel: "esdcan_pcie402: module license 'Proprietary' taints kernel". This is **not** an error message and does not affect the functionality of the CAN driver.
2. Depending on your kernel configuration a compile-time warning "cast to pointer from integer of different size" may be issued when the ***esdcan-pcie402*** driver is compiled.
3. It takes several seconds for the kernel driver to configure and initialize the CAN-PCIe/402 board when the driver is loaded.

Please contact Concurrent Real-Time if you encounter problems with this product.