Release Notes
CCURUEGO (WC-UEGO)

READ ME BEFORE INSTALLING THIS PRODUCT

Driver	ccuruego (WC-UEGO)	v 23.0.1
OS	RedHawk (CentOS or Ubuntu based)	5.4, 6.0, 6.3, 6.5, 7.0, 7.2 and 7.3
Vendor	Concurrent Real-Time	
Hardware	PCIe 4-Channel Universal Exhaust Gas Oxygen Sensor Simulator Card (UEGO)	
Date	July 12th, 2018	

Concurrent REAL-TIME

Table of Contents

1. Introduction

This document assists the user in installing the CCUR-PCIe-UEGO Linux **ccuruego** driver and related software on the RedHawk OS for use with the CCUR-PCIe 4-Channel Universal Exhaust Gas Oxygen Sensor Simulator Card (**UEGO**). The directions in this document supersede all others – they are specific to installing the software on Concurrent Real-Time's RedHawk systems. Other information provided as part of this release, when it may contradict these directions, should be ignored and these directions should prevail.

For additional information on this driver and usage refer to the **ccuruego** man page.

The UEGO is a 4-Channel Universal Exhaust Gas Oxygen Sensor Simulator card with a PCI express interface. In addition to the sensor simulator, the card incorporates a CJ135 module that controls linear oxygen sensors. Normally, the CJ135 is designed to be used inside an engine control unit (ECU) for Gasoline or Diesel engines. It is being used in this card to test and perform on-board diagnostics.

Features and Characteristics of the **UEGO** are:

- 4-channel O2 Sensor Simulation
- Wideband or Narrowband Sensors
- Heater PCM Monitoring
- Open, Ground and V+ Fault Insertion
- Galvanic Isolation Per Channel Pair
- Input Protection +/-50 VDC
- On-Board Test ASIC
- Industry Standard SCSI 68-pin Connector for I/O
- PCI Express x1 Revision 1.0a
- Non-volatile Storage of Calibration & Configuration Data
- NIST Traceable Calibration Standard

2. Requirements

- CCUR-UEGO PCIe board physically installed in the system.
- Selected versions of RedHawk Revision 5.4. 6.0, 6.3, 6.5, 7.0, 7.2 and 7.3. Actual supported versions depend on the driver being installed.

3. Documentation

- PCIe 4-Channel Universal Exhaust Gas Oxygen Sensor Simulator Card (UEGO) Software Interface by Concurrent Real-Time.
- PCIe 4-Channel Universal Exhaust Gas Oxygen Sensor Simulator Card (UEGO) Design Specification by Concurrent Real-Time.

4. Installation and Removal

4.1.   Hardware Installation

The CCUR-UEGO card is a Gen 1 PCI Express product and is compatible with any PCI Express slot. The board must be installed in the system before attempting to use the driver.

The **ccuruego** driver is designed to support IRQ sharing. If this device's IRQ is being shared by another device then this driver's performance could be compromised. Hence, as far as possible, move this board into a PCI slot who's IRQ is not being shared with other devices.

<table>
<tr>
<td>

⚠

</td>
<td>

Caution: **when installing the card insure 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.**

</td>
</tr>
</table>

An *'lspci -v'* or the *'lsirq'* command can be used to determine the IRQs of various devices in the system.

```
# lspci -v -d1542:9300
```

02:04.0 System peripheral: Concurrent Real-Time Device 9300 (rev 01)
    Subsystem: PLX Technology, Inc. Device 9056
    Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV+ VGASnoop- ParErr- Stepping-SERR- FastB2B- DisINTx-
    Status: Cap+ 66MHz+ UDF- FastB2B+ ParErr- DEVSEL=medium >TAbort-
        <TAbort- <MAbort- >SERR- <PERR- INTx-
    Latency: 96, Cache Line Size: 32 bytes
    Interrupt: pin A routed to IRQ 88
    Region 0: Memory at c0100800 (32-bit, non-prefetchable) [size=512]
    Region 2: Memory at c0100000 (32-bit, non-prefetchable) [size=2K]
    Capabilities: <access denied>

```
# lsirq
```

    88   02:04.0 Concurrent Real-Time Unknown device (rev 01)

After installing the card, reboot the system and verify the hardware has been recognized by the operating system by executing the following command:

```
# lspci –d 1542:9300
```

For each CCUR-UEGO PCIe board installed, a line similar to one of the following will be printed, depending on the revision of the system's */usr/share/hwdata/pci.ids* file:

    02:04.0 Unclassified device [0008]: Concurrent Real-Time Device 9300 (rev 01)

If a line similar to the above is not displayed by the **lspci** command, the board has not been properly installed in the system. Make sure that the device has been correctly installed prior to attempting to use the software. One similar line should be found for each installed card.

The default driver configuration uses MSI interrupts. If the kernel supports MSI interrupts, then sharing of interrupts will not occur, in which case the board placement will not be an issue.

4.2.   Software Installation

Concurrent Real-Time™ port of the **ccuruego** software is distributed in RPM format on a CD-ROM. Source for the API library and kernel loadable driver are not included, however, source for example test programs as well as documentation is provided in PDF format.

The software is installed in the **/usr/local/CCRT/drivers/ccuruego** directory. This directory will be referred to as the "top-level" directory by this document.

<table>
<tr>
<td>

⚠

</td>
<td>

Warning: 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:

</td>
</tr>
</table>

```
    # 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.

To install the **ccuruego** 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 **ccuruego_driver**. 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/ccuruego_driver
      --- or on RedHawk 7.0 and above ---
# cd /run/media/[user_name]/ccuruego_driver

# rpm –ivh ccuruego_RedHawk_driver*.rpm     (on an RPM based system)
   --or--
# dpkg –i ccuruego_RedHawk_driver*.deb      (on an Debian based system)

# cd /
# eject
```

On successful installation the source tree for the **ccuruego** package, including the loadable kernel module, API libraries, and test programs is extracted into the **/usr/local/CCRT/drivers/ccuruego** directory by the rpm installation process, which will then compile and install the various software components.

The loadable kernel module is installed in the **/lib/modules/`uname –r`/misc** directory.

Issue the command below to view the boards found by the driver:

```
# cat /proc/ccuruego

Version          : 22.8-1
Built            : Feb 9 2018, 11:02:52
Boards           : 1
  card=0: [02:04.0] bus=2, slot=4, func=0, irq=88, msi=1, ID=4294967295,
                    BoardInfo=0x93000102
```

Once the package is installed, the driver needs to be loaded with one of the following commands:

```
== as root ==
# cd /usr/local/CCRT/drivers/ccuruego
# make load
      --- or on RedHawk 6.5 and below ---
# /sbin/service ccuruego start
      --- or on RedHawk 7.0 and above ---
# /usr/bin/systemctl start ccuruego
```

4.3.   Software Removal

The **ccuruego** driver is a dynamically loadable driver that can be unloaded, uninstalled and removed. Once removed, the only way to recover the driver is to re-install the **rpm** from the installation CDROM:

If any changes have been made to the driver package installed in */usr/local/CCRT/drivers/ccuruego* directory, they need to be backed up prior to invoking the removal; otherwise, all changes will be lost.

```
== as root ==
# rpm -e ccuruego     (driver unloaded, uninstalled, and deleted – on an RPM based system)
    --or--
# dpkg -P ccuruego     (driver unloaded, uninstalled, and deleted – on an Debian based
                        system)
```

If, for any reason, the user wishes to un-load and uninstall the driver and not remove it, they can perform the following:

```
== as root ==
# cd /usr/local/CCRT/drivers/ccuruego
# make unload          (unload the driver from the kernel)
      --- or on RedHawk 6.5 and below ---
# /sbin/service ccuruego stop
      --- or on RedHawk 7.0 and above ---
# /usr/bin/systemctl stop ccuruego
```

To uninstall the **ccuruego** driver, do the following after it has been unloaded:

```
=== as root ===
# cd /usr/local/CCRT/drivers/ccuruego
# make uninstall       (uninstall the driver and library)
```

In this way, the user can simply issue the *'make install'* and *'make load'* in the */usr/local/CCRT/drivers/ccuruego* directory at a later date to re-install and re-load the driver.

5. Auto-loading the Driver

The **ccuruego** driver is a dynamically loadable driver. Once you install the package or perform the *'make install'*, appropriate installation files are placed in the /etc/rc.d/rc*.d or /usr/lib/system/systemd directories so that the driver is automatically loaded and unloaded when Linux is booted and shutdown. If, for any reason, you do not wish to automatically load and unload the driver when Linux is booted or shutdown, you will need to manually issue the following command to enable/disable the automatic loading of the driver:

```
=== as root ===
      --- on RedHawk 6.5 and below ---
# /sbin/chkconfig --add ccuruego       (enable auto-loading of the driver)
# /sbin/chkconfig --del ccuruego       (disable auto-loading of the driver)
      --- or on RedHawk 7.0 and above ---
# /usr/bin/systemctl enable ccuruego   (enable auto-loading of the driver)
# /usr/bin/systemctl disable ccuruego  (disable auto-loading of the driver)
```

6. Testing and Usage

Build and run the driver test programs, if you have not already done so:

```
# cd /usr/local/CCRT/drivers/ccuruego
# make test                            (build the test programs)
```

Several tests have been provided in the */usr/local/CCRT/drivers/ccuruego/test* directory and can be run to test the driver and board.

```
=== as root ===
```

```
# cd /usr/local/CCRT/drivers/ccuruego
# make test                          (build the test programs)
# ./test/ccuruego_dump                (dump all board resisters)
# ./test/ccuruego_rdreg               (display board resisters)
# ./test/ccuruego_reg                 (Display board resisters)
# ./test/ccuruego_regedit             (Interactive board register editor test)
# ./test/ccuruego_tst                 (Interactive test to test driver and board)
# ./test/ccuruego_wreg                (edit board resisters)
# ./test/Eeprom/ccuruego_eeprom       (Eeprom: Burn Eeprom)

# ./test/Flash/ccuruego_flash         (Flash: Flash firmware)
# ./test/Flash/ccuruego_fwreload      (Flash: Firmware reload)

# ./test/lib/ccuruego_disp            (library: display board registers)
# ./test/lib/ccuruego_identify        (library: identify cards in the system)
# ./test/lib/ccuruego_info            (library: provide information of all boards)
# ./test/lib/ccuruego_pwm             (library: display and test PWM functionality of
                                        board)
# ./test/lib/ccuruego_tst_lib         (library: Interactive test to test driver and board)
```

7. Re-building the Driver, Library and Tests

If for any reason the user needs to manually rebuild and load an *installed **rpm*** package, they can go to the installed directory and perform the necessary build.

Warning: 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.

To build the driver and tests:

```
=== as root ===
# cd /usr/local/CCRT/drivers/ccuruego
# make clobber      (perform cleanup)
# make              (make package and build the driver, library and tests)
```

*(Note: if you only wish to build the driver, you can enter the '**make driver**' command instead)*

After the driver is built, you will need to install the driver. This install process should only be necessary if the driver is re-built with changes.

```
=== as root ===
# cd /usr/local/CCRT/drivers/ccuruego
# make install      (install the driver software, library and man page)
```

Once the driver and the board are installed, you will need to **load** the driver into the running kernel prior to any access to the CCUR UEGO board.

```
=== as root ===
# cd /usr/local/CCRT/drivers/ccuruego
```

```
# make load          (load the driver)
```

8. Software Support

- This driver package includes extensive software support and test programs to assist the user in communicating with the board. Refer to the PCIe 4-Channel Universal Exhaust Gas Oxygen Sensor Simulator Card (UEGO) Design Specification by Concurrent Real-Time for more information on the product.

8.1.   Device Configuration

After the driver is successfully loaded, the device to card association file **ccuruego_devs** will be created in the ***/usr/local/CCRT/drivers/ccuruego/driver*** directory, if it did not exist. Additionally, there is a symbolic link to this file in the ***/usr/lib/config/ccuruego*** directory as well. If the user wishes to keep the default one-to-one device to card association, no further action is required. If the device to card association needs to be changed, this file can be edited by the user to associate a particular device number with a card number that was found by the driver. The commented portion on the top of the **ccuruego_devs** file is automatically generated every time the user issues the *'make load'* or *'/sbin/service ccuruego start' (on RedHawk 6.5 and below)* or *'/usr/bin/systemctl start ccuruego' (on RedHawk 7.0 and above)* command with the current detected cards, information. Any device to card association edited and placed in this file by the user is retained and used during the next *'make load'* or *'/sbin/service ccuruego load'* or *'/usr/bin/systemctl start ccuruego'* process.

If the user deletes the **ccuruego_devs** file and recreates it as an empty file and performs a *'make load'* or if the user does not associate any device number with card number, the driver will provide a one to one association of device number and card number. For more information on available commands, view the commented section of the **ccuruego_devs** configuration file.

> **Warning:** If you edit the **ccuruego_devs** file to associate a device to a card, you will need to re-issue the *'make load'* or *'/sbin/service ccuruego start'* or *'/usr/bin/systemctl start ccuruego'* command to generate the necessary device to card association. This device to card association will be retained until the user changes or deletes the association. **If any invalid association is detected, the loading of the driver will fail**.

8.2.   Library Interface

There is an extensive software library that is provided with this package. For more information on the library interface, please refer to the PCIe 4-Channel Universal Exhaust Gas Oxygen Sensor Simulator Card (UEGO) Software Interface by Concurrent Real-Time for more information.

8.3.   Debugging

This driver has some debugging capability and should only be enabled while trying to trouble-shoot a problem. Once resolved, debugging should be disabled otherwise it could adversely affect the performance and behavior of the driver.

To enable debugging, the **Makefile** file in ***/usr/local/CCRT/drivers/ccuruego/driver*** should be edited to un-comment the statement (*remove the preceding '#'*):

```
# BUILD_TYPE=debug
```

Next, use and install the debug driver

```
# cd /usr/local/CCRT/drivers/ccuruego/driver
# make
# make install
```

Next, edit the **ccuruego_config** file in **/usr/local/CCRT/drivers/ccuruego/driver** to un-comment the statement (remove the preceding '#'):

```
# ccuruego_debug_mask=0x00002040
```

Additionally, the value of the debug mask can be changed to suite the problem investigated. Once the file has been edited, the user can load the driver by issuing the following:

```
# cd /usr/local/CCRT/drivers/ccuruego/driver
# make load
```

The user can also change the debug flags after the driver is loaded by passing the above debug statement directly to the driver as follows:

```
# echo "ccuruego_debug_mask=0x00082047" > /proc/driver/ccuruego
```

Following are the supported flags for the debug mask as shown in the **ccuruego_config** file.

```
##############################################################################
#                                                                            #
#          D_ENTER          0x00000001  /* enter routine */                  #
#          D_EXIT           0x00000002  /* exit routine */                   #
#                                                                            #
#          D_L1             0x00000004  /* level 1 */                        #
#          D_L2             0x00000008  /* level 2 */                        #
#          D_L3             0x00000010  /* level 3 */                        #
#          D_L4             0x00000020  /* level 4 */                        #
#                                                                            #
#          D_ERR            0x00000040  /* level error */                    #
#          D_WAIT           0x00000080  /* level wait */                     #
#                                                                            #
#          D_INT0           0x00000100  /* interrupt level 0 */              #
#          D_INT1           0x00000200  /* interrupt level 1 */              #
#          D_INT2           0x00000400  /* interrupt level 2 */              #
#          D_INT3           0x00000800  /* interrupt level 3 */              #
#          D_INTW           0x00001000  /* interrupt wakeup level */         #
#          D_INTE           0x00002000  /* interrupt error */                #
#                                                                            #
#          D_RTIME          0x00010000  /* display read times */             #
#          D_WTIME          0x00020000  /* display write times */            #
#          D_REGS           0x00040000  /* dump registers */                 #
#          D_IOCTL          0x00080000  /* ioctl call */                     #
#                                                                            #
#          D_DATA           0x00100000  /* data level */                     #
#          D_DMA            0x00200000  /* DMA level */                      #
#          D_DBUFF          0x00800000  /* DMA buffer allocation */          #
#                                                                            #
#          D_NEVER          0x00000000  /* never print this debug message */ #
#          D_ALWAYS         0xffffffff  /* always print this debug message */ #
#          D_TEMP           D_ALWAYS    /* Only use for temporary debug code */ #
##############################################################################
```

Another variable **ccuruego_debug_ctrl** is also supplied in the **ccuruego_config** that the driver developer can use to control the behavior of the driver. The user can also change the debug flags after the driver is loaded by passing the above debug statement directly to the driver as follows:

```
# echo "ccuruego_debug_ctrl=0x00001234" > /proc/driver/ccuruego
```

In order to make use of this variable, the driver must be coded to interrogate the bits in the **ccuruego_debug_ctrl** variable and alter its behavior accordingly.

9. Notes and Errata

- Driver and board support MSI interrupts.
- It is possible that *lspci* calls may still display the device with the old name of **"Concurrent Computer Corporation"** instead of **"Concurrent Real-Time"** if the OS has not been updated.

Appendix A: External Connections and Pin-outs

The input/output signals from the UEGO are connected via an industry standard 68-pin SCSI type connector with the following pin-out:
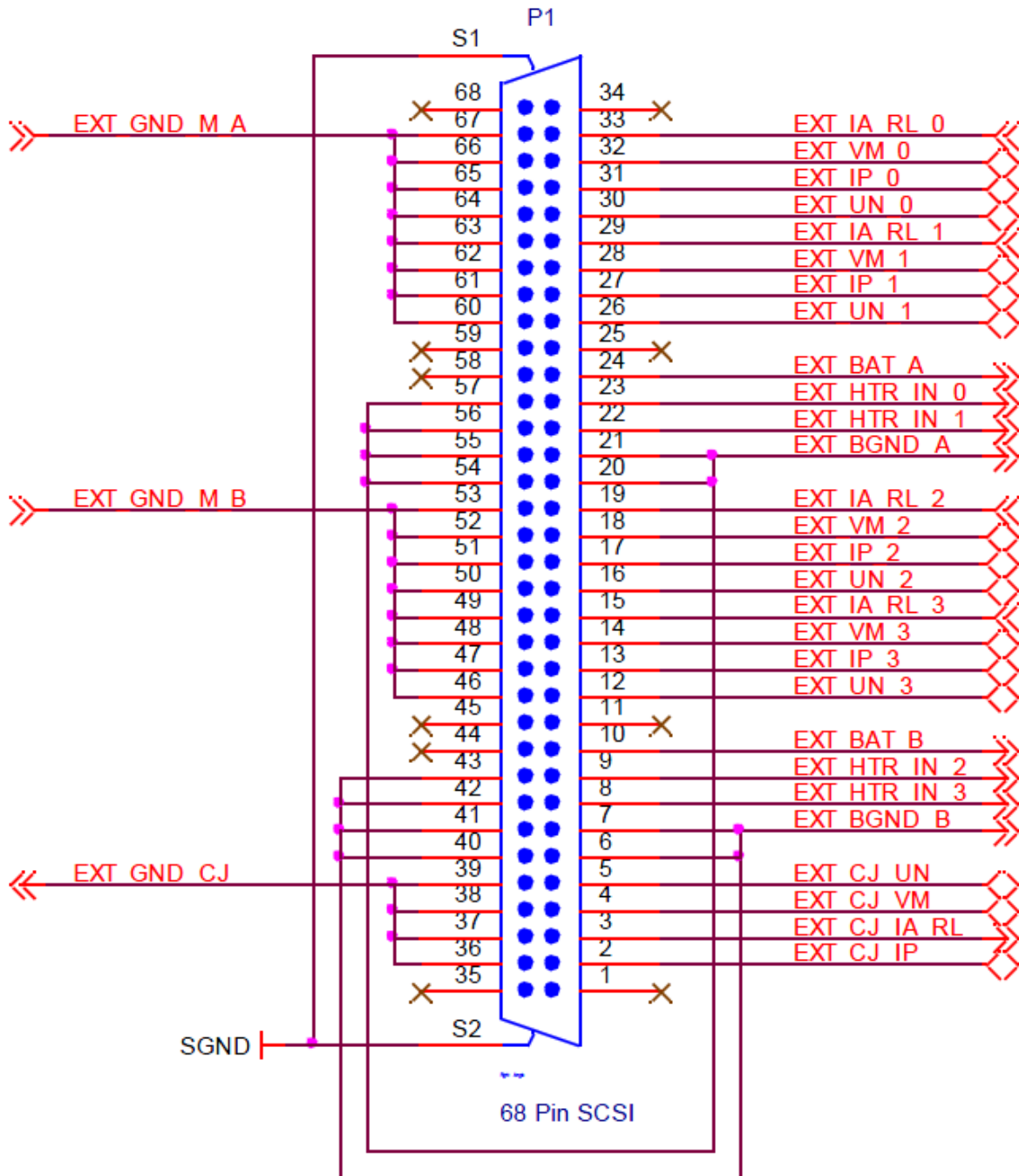
Appendix B: External Connections and Pin-outs

The UEGO has a single multicolor LED indicator located at the top front edge of the board visible via a hole in the front panel. If the board is in a reset state the indicator will be solid Red. After reset is complete, the indicator will cycle through Red, Green and Blue for approximately 1 second each as a lamp test. If the indicator remains Red after reset is complete it would indicate a board malfunction. Other states of the board during operation are indicated as follows:

Color	Description	Input/Outputs
Red	Board in Reset	Not Active
Green	Board Operational	Not Active
Blue	Board Operational	Active

The Green & Blue indication will *flash* at about a one second rate if the Identify Board bit is set.

Appendix C: The 4-Channel Universal Exhaust Gas Oxygen Sensor Simulator Card
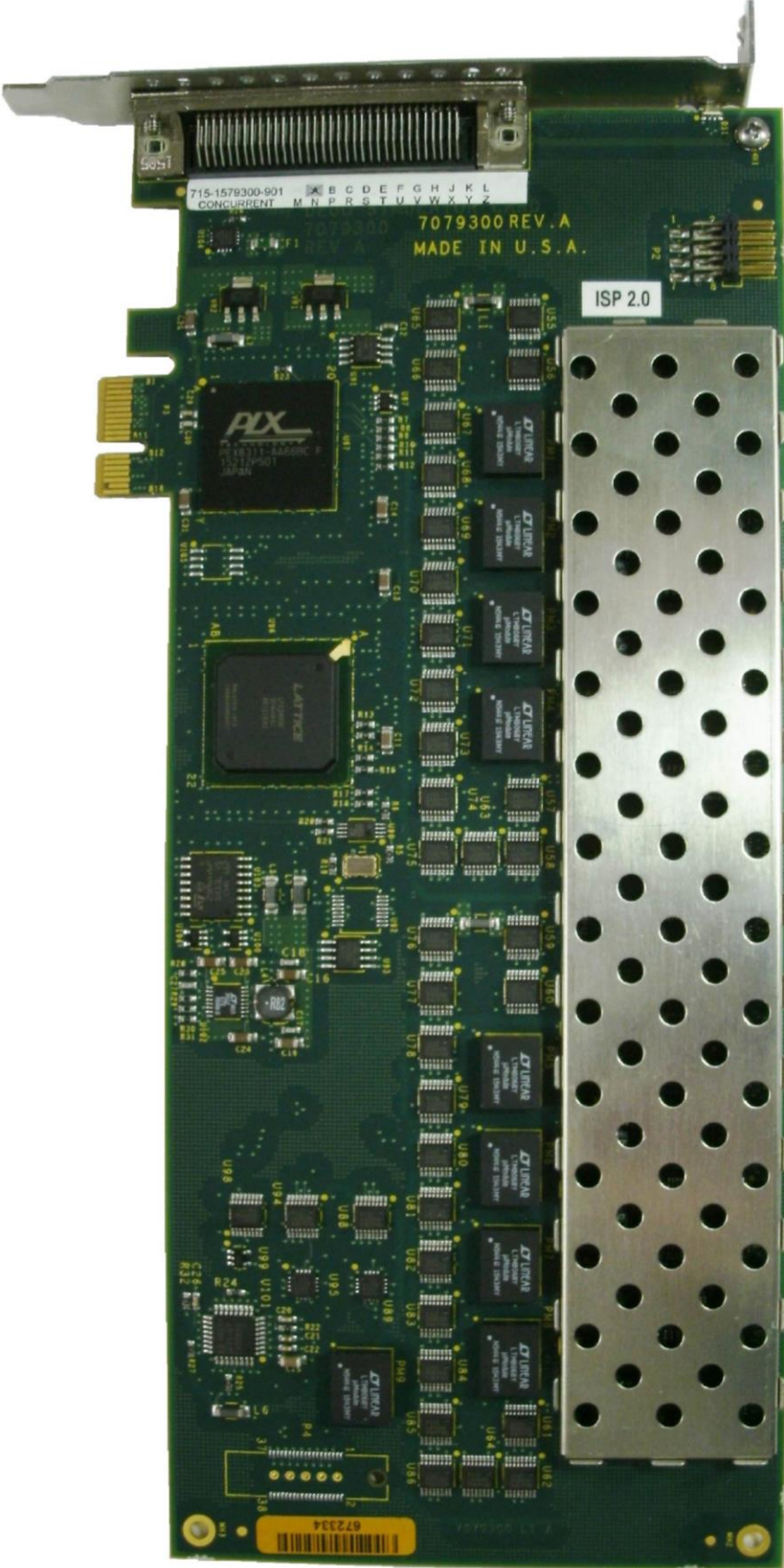
This page intentionally left blank