Release Notes
CCURPWMIN (WC-PWM-1112 Input)

Driver	ccurpwmin (WC-PWM-1112)	
OS	RedHawk	
Vendor	Concurrent Real-Time, Inc.	
Hardware	PCIe 12-Channel PWM Input Card (CP-PWM-1112)	
Author	Darius Dubash	
Date	February 16th, 2022	Rev 2022.1

Table of Contents

1. Introduction

This document assists the user in installing the CP-PWM-1112-Linux ***ccurpwmin*** driver and related software on the RedHawk OS for use with the CP-PWM-1112 PWM input board. 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 Concurrent Real-Time's directions, should be ignored and Concurrent Real-Time's directions should prevail.

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

Brief Description of the CP-PWM-1112 Input Board
The CP-PWM-1112 is an FPGA-based Pulse Width Modulation (PWM) Input card from Concurrent Real-Time. The CP-PWM-1112 input card is designed for capturing pulse width modulated signals with high accuracy. With a timing resolution of 15.15 nanoseconds and the ability to measure the frequency and duty cycle, the CP-PWM-1112 is ideal for use in hardware-in-the-loop (HIL) applications. The CP-PWM-1112 comes in PCIe form factor. Multiple CP-PWM-1112 cards can be placed in one system. A Molex LFH-60 connector is mounted on each card for connection to external devices.

This driver is not supported with 32-bit applications running on RedHawk 7.0 and above.

CP-PWM-1112 Features:

- FPGA based PWM board
- PCIe form factor
- TTL inputs
- 66 MHz PWM measurement base frequency
- Minimum pulse width of 15.15 nsec
- Minimum period of 30.30 nsec
- Pulse-width/period accuracy of 2 internal clock cycles (30.30 nsec)
- Programmable pulse width averaging (maximum limit is 127 pulses)
- Measurement frequency range of 0.05 Hz to 660KHz
- Measurement duty cycle of 0-100%
- Programmable digital debouncing filters on every channel
- External Connectors: Molex LFH-60
- Power Consumption: ~5 watts

2. Requirements

- CP-PWM-1112 PCIe input board physically installed in the system.
- This driver supports various versions of RedHawk. Actual supported versions depend on the driver being installed.

3. Documentation

- PCIe 12-Channel PWM Input (PWMIN) Software Interface by Concurrent Real-Time.

4. Installation and Removal

4.1. Hardware Installation

The CCUR-PWMIN card is a x1 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.

***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.*

The ***ccurpwmin*** driver is designed to support IRQ sharing, however, since it does not use interrupts, it should not matter which PCIe slot is placed in.

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

```
# lspci –v
   05:04.0 System peripheral: Concurrent Computer Corporation Device 9272 (rev 01)
           Subsystem: PLX Technology, Inc. Device 9056
           Flags: bus master, 66MHz, medium devsel, latency 96, IRQ 19
           Memory at c4b08000 (32-bit, non-prefetchable) [size=512]
           Memory at c4b00000 (32-bit, non-prefetchable) [size=32K]
           Capabilities: <access denied>

# lsirq
   19      05:04.0 Concurrent Computer Corporation 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 1542:9272
```

For each CCURPWM-1012 (Output) or CCURPWMIN-1112 (Input) 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:

```
   0b:04.0 System peripheral: Concurrent Computer Corporation Unknown device 9272 (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 individual driver detects whether the board is a CCURPWM-1012 (Output) or CCURPWMIN-1112 (Input) based on the firmware register. If a ***ccurpwm*** driver is installed and only *input* cards are present, or if a ***ccurpwmin*** driver is installed and only *output* cards are present, then the driver loading will fail and appropriate message will be generated in the kernel log file that can be viewed by the ***dmesg*** command.

4.2. Software Installation

Concurrent Real-Time™ port of the ***ccurpwmin*** software is distributed in RPM and DEB format on a CD-ROM. Source for the API library, example test programs, and kernel loadable driver are included, as is documentation in PDF format.

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

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 install the ***ccurpwmin*** 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 **ccurpwmin_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/ccurpwmin_driver
      --- or on RedHawk 7.0 and above ---
# cd /run/media/[user_name]/ccurpwmin_driver
      --- or on Ubuntu RedHawk ---
# cd /media/[user_name]/ccurpwmin_driver

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

# cd /
# eject
```

On successful installation of the ***rpm*** the ***ccurpwmin*** driver files will be extracted into the ***/usr/local/CCRT/drivers/ccurpwmin*** directory from the CDROM drive and compiled. Next, the driver will need to be loaded on the machine once the hardware has been installed.

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

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

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

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

```
# cat /proc/ccurpwmin

version: 24.1.0
built: Thu May  6 11:03:25 EDT 2021
boards: 1
  card=0: [07:04.0] bus=7, slot=4, func=0, irq=32, firmware=0x00020002
```

Note: With RedHawk 7.5 you may see a cautionary message similar to the following when the **ccurpwmin** driver is loaded on the system console or via *dmesg* command:

```
CHRDEV "ccurpwmin" major number 233 goes below the dynamic allocation range
```

As documented in the kernel driver **Documentation/devices.txt** file a range of character device numbers from 234 to 254 are officially available for dynamic assignment. Dynamic assignments start at 254 and grow downward. This range is sometimes exceeded as additional kernel drivers are loaded. Note that this was also the case with earlier kernels – the newer 7.5 kernel has added a runtime check to produce this warning message that the lower bound has been exceeded, not reduced the range of numbers officially available for dynamic assignment. If you see this message please verify the assigned number(s) isn't being used by a device installed on your system.

On RedHawk 8.x kernels, you may see cautionary messages on the system console or via *dmesg* command similar to the following when the **ccurpwmin** driver is loaded, as this is a proprietary driver:

```
ccurpwmin: module verification failed: signature and/or required key missing - tainting kernel
```

4.3. Software Removal

The ***ccurpwmin*** 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/ccurpwmin*** directory, they need to be backed up prior to invoking the removal; otherwise, all changes will be lost.

```
=== as root ===
# rpm –e ccurpwmin         (driver uninstalled, and deleted – on an RPM based system)
--- or ---
# dpkg -P ccurpwmin        (driver uninstalled, and deleted – on a 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/ccurpwmin
# make unload           (unload the driver from the kernel)
      --- or on RedHawk 6.5 and below ---
# /sbin/service ccurpwmin stop
      --- or on RedHawk 7.0 and above ---
# /usr/bin/systemctl stop ccurpwmin
      --- or on Ubuntu RedHawk ---
# /bin/systemctl stop ccurpwmin
```

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

```
=== as root ===
# cd /usr/local/CCRT/drivers/ccurpwmin
# 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/ccurpwmin*** directory later to re-install and re-load the driver.

5. Auto-load the Driver

The ***ccurpwmin*** driver is a dynamically loadable driver. Once you install the package or perform the ***'make install'***, appropriate installation files are placed in the /usr/lib/system/systemd directory 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 ccurpwmin          (enable auto-loading of the driver)
# /sbin/chkconfig --del ccurpwmin          (disable auto-loading of the driver)
      --- or on RedHawk 7.0 and above ---
# /usr/bin/systemctl enable ccurpwmin      (enable auto-loading of the driver)
# /usr/bin/systemctl disable ccurpwmin     (disable auto-loading of the driver)
      --- or on Ubuntu RedHawk ---
# /bin/systemctl enable ccurpwmin          (enable auto-loading of the driver)
# /bin/systemctl disable ccurpwmin         (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/ccurpwmin
# make test                           (build the test programs)
```

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

```
=== as root ===
# cd /usr/local/CCRT/drivers/ccurpwmin
# make test                            (build the test programs)
# ./test/ccurpwmin_disp                (display channel data)
# ./test/ccurpwm_dump                  (Display board registers including pci config and
                                       bridge register and main control registers)
# ./test/ccurpwmin_reg                 (display board resisters)
# ./test/ccurpwmin_tst                 (Interactive test to test driver and board)
# ./test/ccurpwmin_tst_lib             (library: Interactive test to test driver and board)
# ./test/ccurpwm_rdreg                 (display board resisters by offset)
# ./test/ccurpwm_wreg                  (write board resisters by offset)
```

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/ccurpwmin
# 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/ccurpwmin
# 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 CCURPWMIN board.

```
=== as root ===
# cd /usr/local/CCRT/drivers/ccurpwmin
# 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 *Concurrent Real-Time PCIe 12-Channel PWM Input Card (PWMIN) Software Interface* document for more information on the product.

8.1. Device Configuration

After the driver is successfully loaded, the device to card association file ***ccurpwmin_devs*** will be created in the ***/usr/local/CCRT/drivers/ccurpwmin/driver*** directory, if it did not exist. Additionally, there is a symbolic link to this file in the ***/usr/lib/config/ccurpwmin*** 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 ***ccurpwmin_devs*** file is automatically generated every time the user issues the ***'make load'*** 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'*** process.

If the user deletes the ***ccurpwmin_devs*** 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 ***ccurpwmin_devs*** configuration file.

Warning: If you edit the ***ccurpwmin_devs*** file to associate a device to a card, you will need to re-issue the ***'make load'*** 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 12-Channel PWM Input Card (PWMIN) Software Interface by Concurrent Real-Time* document.

8.3. Firmware Updates

This board is capable of being re-programmed in the field as new firmware updates are made available by *Concurrent Real-Time™.* The procedure for re-programming the firmware will be supplied to the user at the time when a firmware update is necessary.

8.4. 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/ccurpwmin/driver*** should be edited to un-comment the statement (*remove the preceding '#'*):

```
# EXTRA_CFLAGS += -DCCURPWMIN_DEBUG
```

Next, compile and install the driver

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

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

```
# ccurpwmin_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/ccurpwmin/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 "ccurpwmin_debug_mask=0x00082047" > /proc/driver/ccurpwmin
```

Following are the supported flags for the debug mask as shown in the ***ccurpwmin_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_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 ***ccurpwmin_debug_ctrl*** is also supplied in the ***ccurpwmin_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 "ccurpwmin_debug_ctrl=0x00001234" > /proc/driver/ccurpwmin
```

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

9. Notes and Errata

- If a kernel is configured with the CONFIG_DEBUG_LOCK_ALLOC define, the driver will fail to compile due to mutex_lock_nested() call being included with GPL requirement. If you want to successfully compile the driver, you will need to remove the CONFIG_DEBUG_LOCK_ALLOC define and rebuild the kernel.
- Ubuntu kernels RH8.0 onwards may have the default ***systemd-timesyncd*** daemon installed which does not accurately adjust the system.You may want to replace the default with the ***chrony*** package for a more accurate time asjustment.
- This board does not use interrupts for DMA operation.
- You should not supply the board with less than 0 volts or greater than 5 volts, otherwise, you could damage it.

Appendix A: Features

1. Frequency /Period measurement

Period is measured as the sum of On Time and Off Time.

$$Period = On\ Time + Off\ Time$$

2. Duty cycle measurement

Duty cycle is measured as the ratio of On Time and Period.

$$Duty\ cycle\ (\%) = (On\ Time\ /\ Period\ )\ \ x\ 100$$

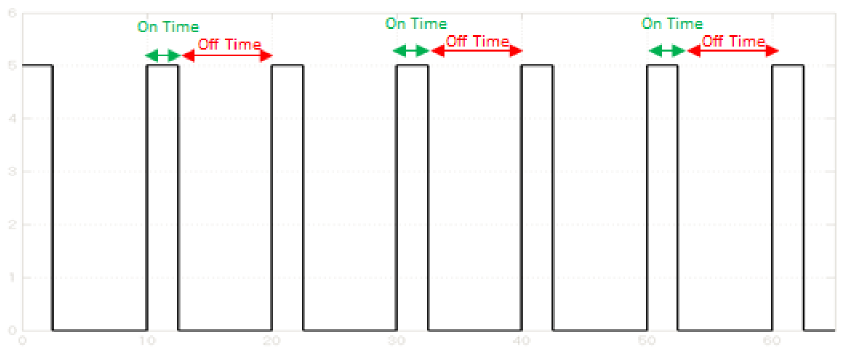

Figure 1. Typical PWM input

3. Rising edge count

Count of the number of rising edges between user resets of the rising edge counter.

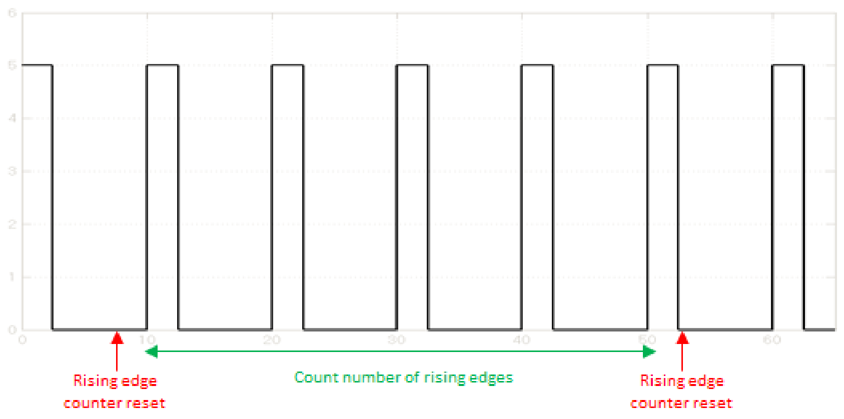

Figure 2. Rising edge count between user reset of the rising edge counter

4. Average period measurement based on a programmable window

Average period is measured as the ratio of the sum of the periods over a window and the size of the programmable window.

$$Average\ period\ \ = \frac{\sum_{i=1}^{j} Period}{j}$$

where *j* is the size of the programmable window with a maximum window size of 127. The periods are stored in a FIFO and each edge that initiates a measurement causes a transfer of the periods in and out of the FIFO. The sum of the periods is supplied to the user in a 32-bit register. Users need to ensure that the window size of average selection times the period width count must not exceed the 32-bit register, otherwise, incorrect averaging will result. This is only true when the input pulse is of a very low frequency.(less than 0.52Hz) with the maximum window size of 127. As the frequency is reduced, the user needs to reduce the window size accordingly.

5. Programmable digital debouncing filter

The debouncing filter can be set up with a count of 0 to a maximum of 127. A count of 0 means no filtering. A count greater than 0 means, filter the input for glitches less than $count \times 30.30ns$.

If count is 1, glitches/frequencies above 33MHz will be filtered out.
If count is 127, glitches/frequencies above 250KHz will be filtered out.

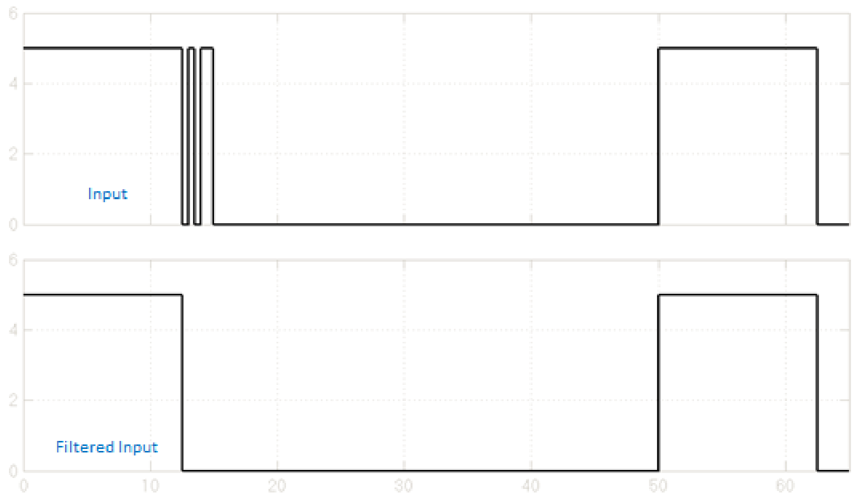

Figure 3. Debouncing and glitch removal

Appendix B: External Connections and Pin-outs

LFH60 Connector

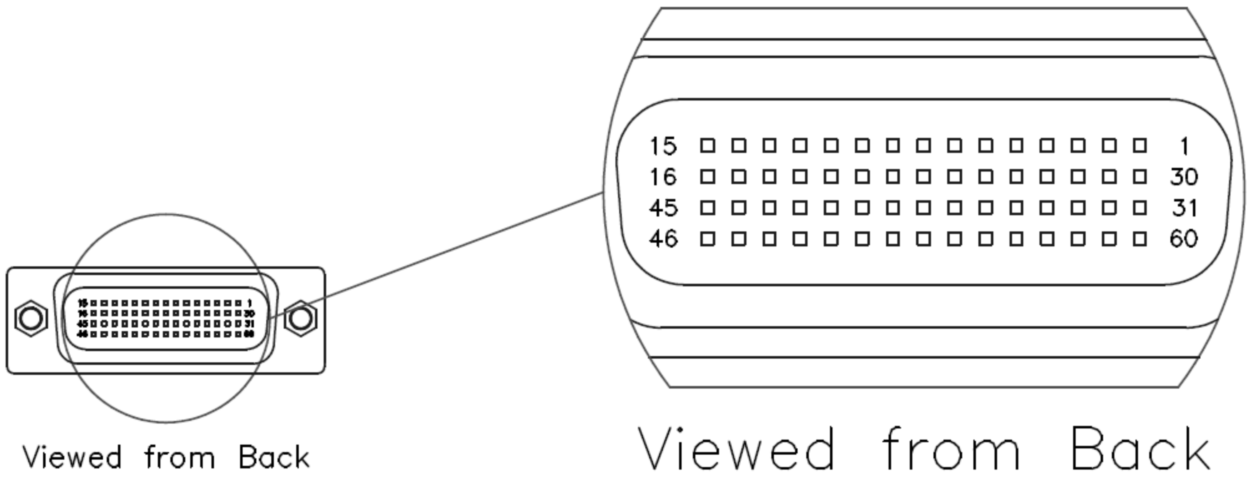

Figure 4. LFH60 connector

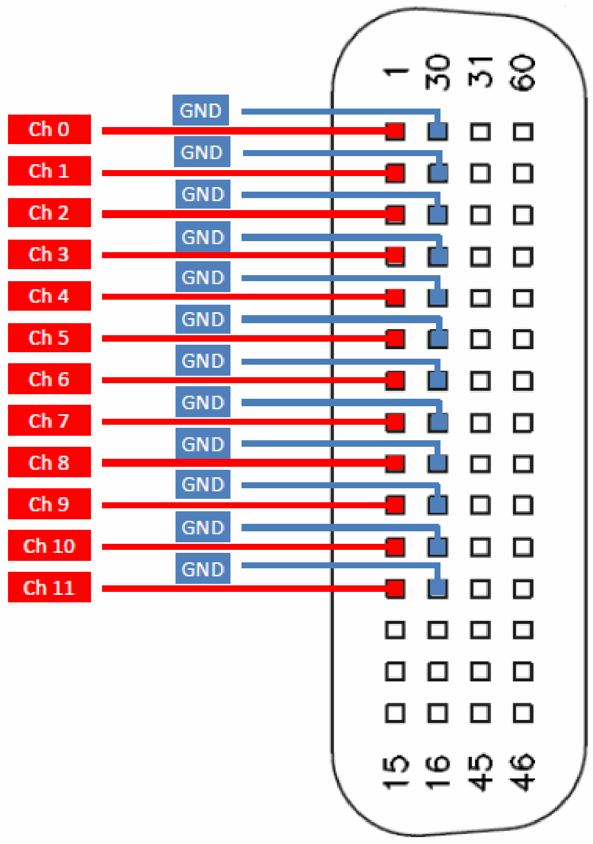

Figure 5. LFH60 PWM Input Pin-out

Appendix C: The Board

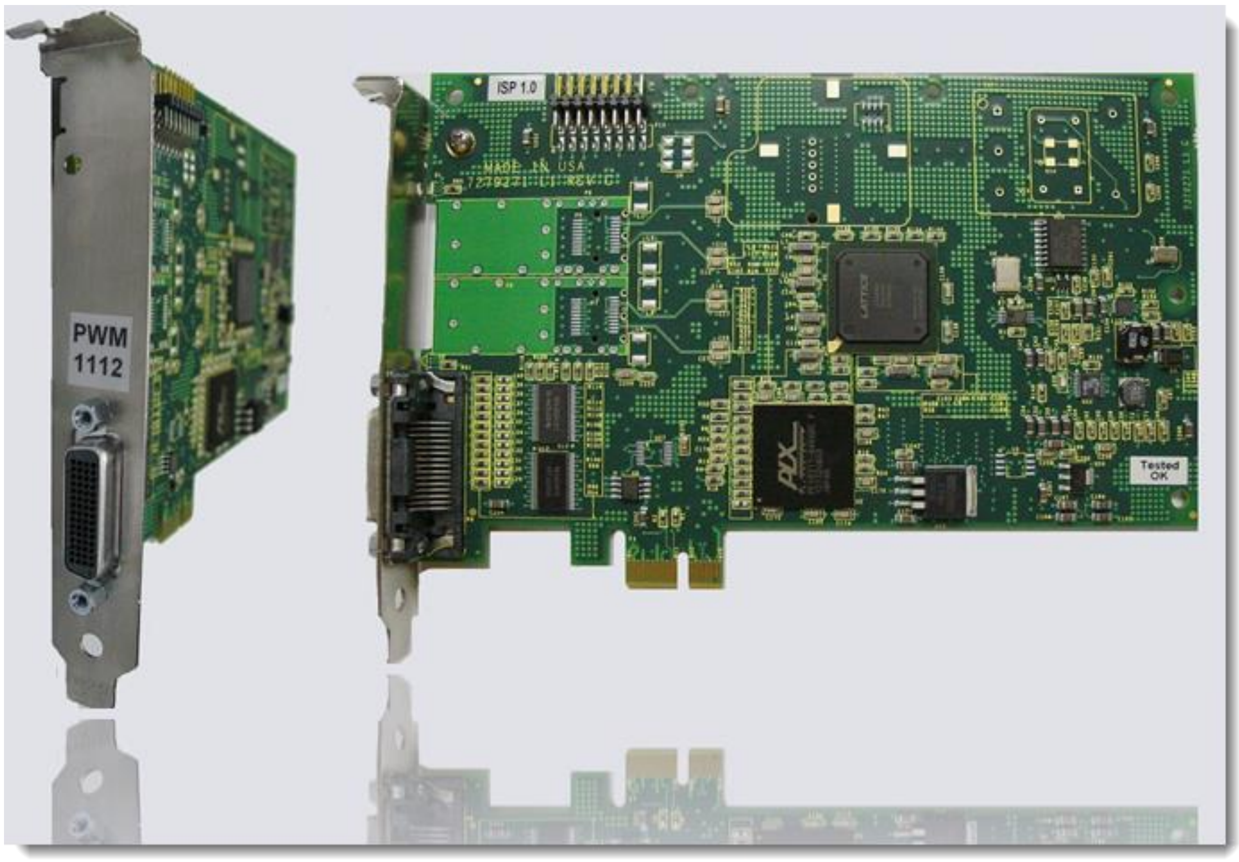

Figure 6. CP-PWM-1112 Card

This page intentionally left blank