United Electronic Industries

The High-Performance Alternative

PowerDNA DNA-QUAD-604 Quadrature Encoder Input Layer User Manual

32-bit 4-channel Quadrature Encoder Input Layer for PowerDNA, UEILogger, UEIPAC, and UEIModbus Cubes

Version 1.2

August 2007 Edition
PN Man-DNA-QUAD-604-0807

Contacting United Electronic Industries

Mailing Address:

27 Renmar Avenue

Walpole, MA 02081

U.S.A.

For a list of our distributors and partners in the US and around the world, please see http://www.ueidaq.com/partners/

Support:

Telephone: (508) 921-4600
Fax: (508) 668-2350

Also see the FAQs and online "Live Help" feature on our web site.

Internet Support:

Support	support@ueidaq.com
Web-Site	www.ueidaq.com
FTP Site	ftp://ftp.ueidaq.com

Product Disclaimer:

WARNING!

DO NOT USE PRODUCTS SOLD BY UNITED ELECTRONIC INDUSTRIES, INC. AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS.

Products sold by United Electronic Industries, Inc. are not authorized for use as critical components in life support devices or systems. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness. Any attempt to purchase any United Electronic Industries, Inc. product for that purpose is null and void and United Electronic Industries Inc. accepts no liability whatsoever in contract, tort, or otherwise whether or not resulting from our or our employees' negligence or failure to detect an improper purchase.

Table of Contents

Table of Figures

Chapter 1 Introduction

This document outlines the feature-set and use of the DNA-QUAD-604 layer. This layer is a 4-channel quadrature encoder input module for the PowerDNA I/O Cube.

1.1 Organization of this manual

This PowerDNA DNA-QUAD-604 User Manual is organized as follows:

Introduction

This section provides an overview of PowerDNA Quadrature Encoder Input board features and what you need to get started.

The DNA-QUAD-604 layer

This chapter provides an overview of the device architecture, connectivity, and logic of the DNA-QUAD-604 layer.

Programming with the UeiDaq Framework High-Level API

This chapter provides an overview of the how to create a session, configure the session for encoder inputs, and interpret results on the DNA-QUAD-604 layer.

Programming with the Low-Level API

This chapter describes low-level API commands for configuring and using the DNA-QUAD-604 layer.

Appendices – Accessories

This appendix provides a list of accessories available for DNA-QUAD-604 layer.

Index

This is an alphabetical listing of the topics covered in this manual.

Manual Conventions

To help you get the most out of this manual and our products, please note that we use the following conventions:

Tips are designed to highlight quick ways to get the job done, or reveal good ideas you might not discover on your own.

NOTE: Notes alert you to important information.

Caution advises you of precautions to take to avoid injury, data loss, and damage to your boards or a system crash.

Text formatted in **bold** typeface generally represents text that should be entered verbatim. For instance, it can represent a command, as in the following example: “You can instruct users how to run setup using a command such as **setup.exe**.”

1.2 The DNA-QUAD-604 Layer

The DNA-QUAD-604 has the following features:

- 4 independent quadrature encoder input units (QDU) with simultaneous start/pause/stop option and independent or shared clock capability
- A, B, and Z (index) inputs for each encoder with edge detection on each
- Z input has programmable polarity, edge sensitivity, and other functions (reset/load counter immediate, reset/load counter on A/B states, generate interrupt or trigger pulse, etc.)
- All decoder units share same ground, which is electrically isolated from the sytsem with 350V isolation level
- Six I/O pins per decoder unit
  - Input: A
  - Input: B
  - Input: Z (index)
  - Input: TRIGIN/DIn0 (may be used as general purpose input)
  - Output: TRIGOUT/DOut0 (may be used as general purpose output)
  - Output: CLKOUT/DOut1 (may be used as general purpose output)
- Direct access to all pins as general purpose digital I/O
  - All input pins combined into dedicated 16-bit register
  - All GPIO mode outputs accessible via dedicated 8-bit register
  - GPIO accessible in buffered mode via 1024-word I/O FIFOs
  - Clock for buffered operation derived from SYNC bus
- 1x, 2x, 4x counting modes for quadrature encoder
- Four modes of operation –

Operating Mode	Max. Input Frequency (MHz)		
	1x	2x	4x
Quadrature Encoder	4.125	8.25	16.5
Direction of count	8.25	NA	NA
Count up			
Count down			

- Can capture rotational data at specific time increments
- Can capture time data at specific angular positions
- Optional timestamp with every data sample
- Position compare outputs on DOut0 and DOut1
- Phase error reporting on quadrature encoder failure
- 1024 x 32 bit input FIFO on every quadrature encoder plus programmable watermark level
- Uses 66 MHz internal or up to 16.5 MHz external timebase
- 512 x 32-bit input and output FIFOs on each channel
- 24-bit debouncer on all inputs running from 16.5 MHz clock
- 16 interrupt sources on every counter (see QDU_IER register for detail)
- SYNCx interface support (start/stop trigger and timebase)
- Global enable/disable via SYNC interface
- Simultaneous updates on all four channels

1.2.1 Specifications

The DNA-QUAD-604 has the Technical Specifications listed in **Table 1-1**.

Table 1-1. Technical Specifications

Technical Specifications:

Number of inputs	4
Counter depth	32 bits
Input encoder modes	x1, x2 and x4
Maximum input frequency	16.5 MHz at x4, 8.25 at x2, 4.125 at x1
Minimum input rise/fall time	1 µsec
Minimum frequency	no low limits
Minimum pulse width / period	15.15 nsec / 30.30 nsec
On-board FIFOs, per input	1024 counts
Debounce interval	60.6 nS to 1 S (user programmable in 256 nS steps)
Protection	7 kV ESD, ±40 VDC (80 mA per pin, max), 350V isolation
Input Low voltage	0.0-0.8V
Input High voltage	2.0-5.0V
Output Low voltage	0.0-0.8V
Output High voltage	2.0-5.0V at ±12 mA
Power consumption	2W
Operating range	Tested -40 to +85 °C
Humidity range	90%, noncondensing
Vibration	5 g (10 - 500 Hz)
Shock	50 g, 18 shocks at 6 orientations
Alltitude	to 70,000 feet

Figure 1-1 is a photo of the DNA-QUAD-604 Layer Board.

Figure 1-1. Photo of DNA-QUAD-604 Board

1.3 Definition of Terms

This section is a glossary of terms used specifically with the DNA-QUAD-604 layer.

QDU — Quadrature decoder unit

CR — Counter register. This is the main register that holds the current value of the up/down counter.

CTU — Counter/Timer Unit

EM — End mode. This is a special term used to describe a condition that ends the current operation performed by a **QDU**. An end condition acts as a tigger that copies the current value of the CR register and optional timestamp into the input FIFO. An end mode can be defined as a “clock source for the input FIFO.” When detected by the QDU, it may halt or stop and reload/restart operation, if reload mode is enabled. Once stopped, the counter may be re-enabled by a software or hardware trigger. As an option, an interrupt may be generated when a CM end-mode condition is detected.

The following end-mode conditions are defined:

EM_CR0 — CR<CR0. This may be used for frequency measurement of input **A**.

EM_CR1 — CR>CR1. This may be used for frequency measurement of input **A**.

EM_CR01 — CR<CR0 or CR>CR1. This may be used for wraparound counter/ encoder.

EM_IE — An event (rising or falling edge on the index or TRIGIN input.

EM_TBR — Timebase register count is down to 0.

EM_INC — Increments/decrements counter by pre-defined number of counts. Used with timestamps to measure time interval between beginning and end of N counts and also to generate an external pulse every N counts.

EM_INF — Indefinite wraparound counter (used in non-buffered mode)

CM — Counter mode. Defines mode of operation for main counter, such as

CM_CDU — Count up for input A
CM_CDA — Count down for input A
CM_DC — Direction counter (uses A as souce and B as direction)
CM_QE — Quadrature decoder mode (A, B, and Z inputs)
All four modes may have a hardware or software trigger, one-time or retriggerable

CRM — Counter reload mode. Defines the value loaded into **CR** after **EM** is detected, as follows:
CRM_LR — LR Load Register
CRM_CR01 — CR0 for the down count and CR1 for the up count
CRM_CR10 — CR0 for the down count and CR1 for the up count
CRM_NR — No reload (counter keeps its value)
CRM_OTR — One time reload. Reloads once after every start trigger and after an EM event.

EVT — Index or user event, positive or negative edge on **Z** input, or **TRIGIN** pin on any of the following **A/B** transitions: low/low, low/high, high/low, high/high. (Some counting modes are not compatible with some end modes. See **CCR** register description for details.)

1x, 2, 4x Counting — Figure 1-2 illustrates the pulse trains generated when you use 1x, 2x, or 4x counting modes.

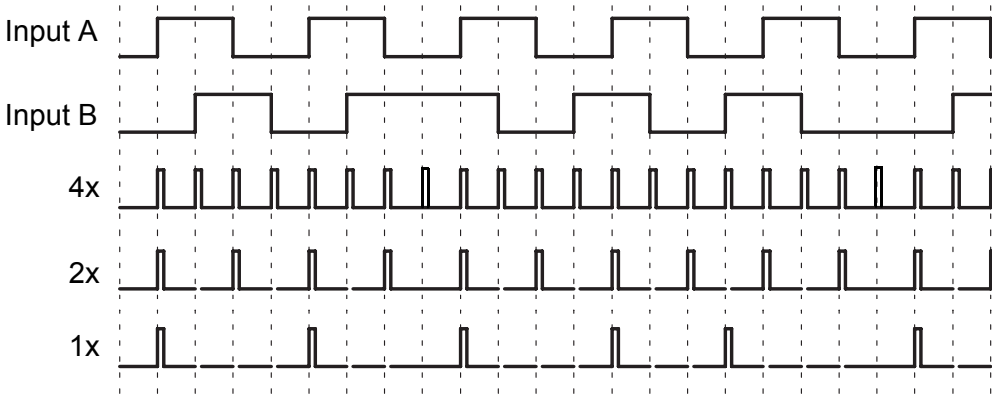


Figure 1-2. Pulse Trains in 1x, 2x, and 4x Counting Modes

The 1x signal has a pulse at each rising edge of Input A signal.

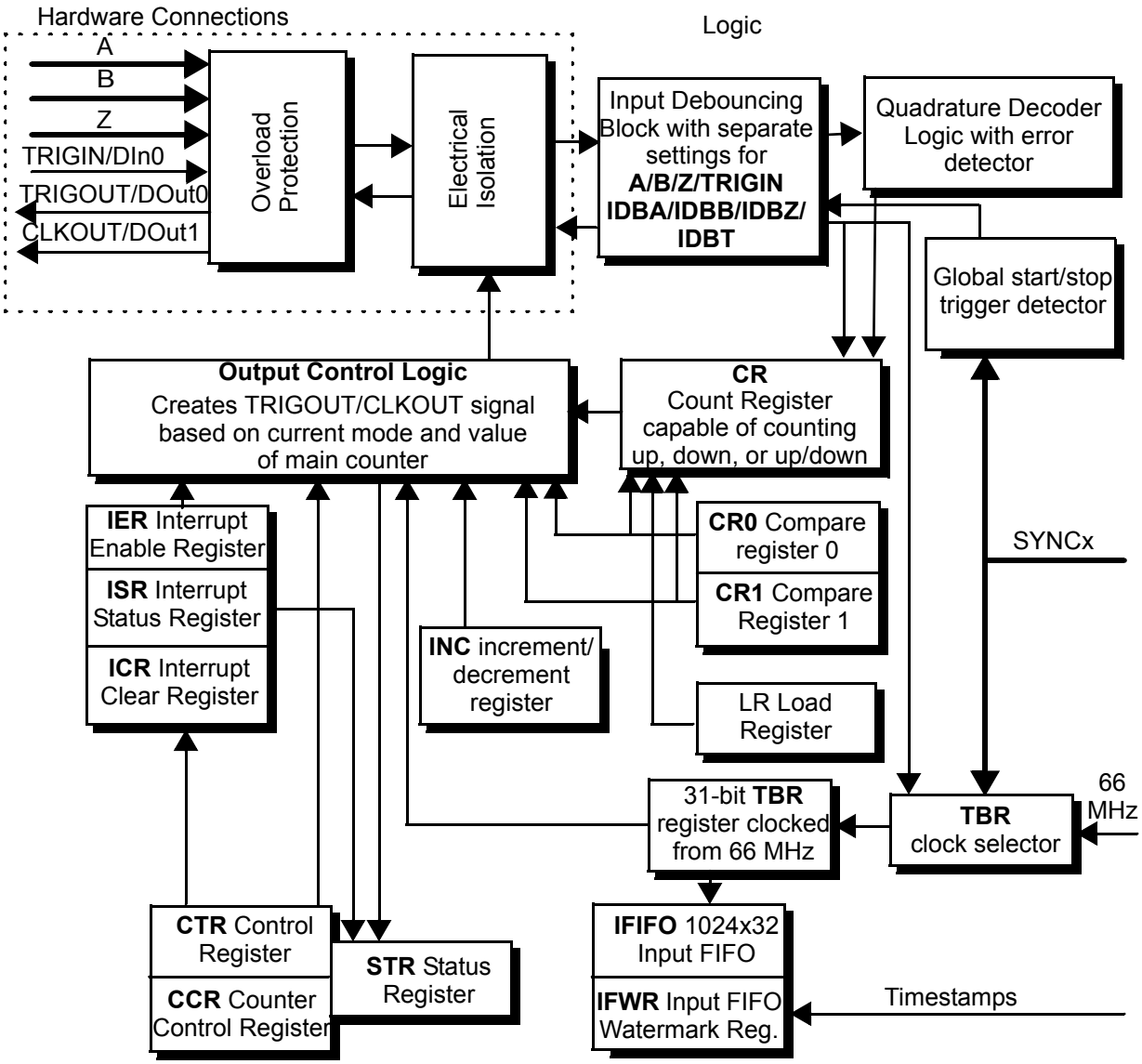
The 2x signal has a pulse at each rising and falliing edge of Input A signal.

The 4x signal has a pulse at each rising or falling edge of both A and B input signals.

NOTE: For Register Maps and detailed bit descriptions not included in this chapter, refer to *Chapter 3 – Programming with the Low Level API.*

1.4 Functional Description

Figure 1-3 is a functional block diagram of a single Counter/Timer Unit (**CTU**) of a DNA-QUAD-604 Layer board.



Note 1: Every CTU supports two additional types of global start/stop triggers, both of which work as gating functions on the A/B/Z debouncers. They can enable/disable all counters on the layer via the LCR_IOEN bit in the LCR register (0x0) or the standard start/stop trigger form the SYNC interface, which may be enabled/disabled for every counter. SYNCx lines may, as an option, be used to provide a clock input to the time base registers.

Note 2: The diagram above does not show all registers of the layer.

Figure 1-3. Functional Block Diagram of Counter/Timer Unit (CTU)

Each of the four counter/timer units (CTUs) is connected to the external world by a 6-wire interface that also provides ESD and overvoltage protection. The six inputs and outputs are:

- Quadrature Encoder signals, A, B, and Z (index)
- An additional TRIGIN line that may be used to reset the Quadrature Counter, to pace the input FIFO, or as a general purpose digital input
- An output clock line, CLKOUT, which can be used as a quadrature clock output corresponding to the 1x, 2x, or 4x time base, a position above CR1, an index event, or a general purpose digital output
- An output trigger line, TRIGOUT, which can be used as a position below CR0, a start/stop trigger, an end mode output, or as a general purpose digital output

All input lines are connected to the debouncing block, which eliminates unwanted spikes from the incoming signals. The debouncing block is controlled by the 24-bit **IDBA.IDBB,IDBZ** and **IDBT** registers, which are loaded with the number of 16.5 MHz clock cycles during which the input signal must be stable before being accepted for further processing.

The main counter register, **CR**, may use signals from Input A or outputs from the Quadrature Decoder Unit (QDU) to count up, down, or up and down.

The load register, **LR**, provides the initial value for starting the count. In some operating modes, a strobe on the Z input is used to reset the CR register with the LR stored value.

In some modes, the time base register (**TBR**) is used to define the pace of counter captures. The TBR may use the internal clock, the debounced TRIGIN signal, or one of the SYNCx lines as a clock source. This permits you to use the on-board PLL as a clock at a frequency other than 66 MHz.

Two compare registers, **CR0** and **CR1**, are used to specify maximum and minimum expected values for the quadrature encoder counter. In some modes, a pulse on the the Z index pin resets **CR** to the **CR0** or **CR1** value, depending on the direction of counting.

The **CTR** control register can be used to enable/disable the counter, to access the counter/timer pins when in GPIO mode, and to enable/disable the inversion mode for I/O pins and buffered FIFO operation.

The **CCR** counter control register defines the mode of operation of the counter and the **STR** status register reports the current status of the quadrature decoder unit operation.

The **IER** interrupt enable register can be used to enable/disable interrupt generation for various conditions. The **ISR/ICR** Interrupt Status/Clear Registers report the status of enabled interrupts and clear the interrupt conditions after they are processed by the host computer.

The Input FIFO enables implementation of buffered I/O mode and the Input FIFO Watermark Register **IFWR** may be used to set the desired watermark level for the input FIFO.

The quadrature encoder delay (**QED**) register specifies the minimum acceptable delay, stated as a number of 16.5 MHz clock intervals, between edges of the A and B inputs.

1.4.1 Counter Register (CR) Counting Modes

The **CR** can perform up, down, and up/down counting operations. Depending on the reload option selected, it can be reloaded with the value stored in **LR**, **CR0**, or **CR1**. It can operate in any of four modes:

- **Count Up Counter for Input A.**
  In this mode, the counter increments its value with every pulse from the debounced A input. Polarity and edge sensiitivity are selectable. When combined with EM = CR>CR1 and input FIFO with timestamps, this mode adds the current timestamp to every CR1 count as it is stored in the FIFO, facilitating frequency calculations.
- **Count Down Counter for Input A.**
  Same as count up counter except direction of count is down.
- **Direction Counter**
  Uses A as source and B as a direction. This is simplest version of the quadrature encoder. It does not detect quadrature encoder errors and is identical to the CT-601 quadrature mode.
- **Quadrature Decode Mode**
  Uses A, B, and Z inputs and Quadrature Detector. In ths mode, the counter is driven by the quadrature detector unit, which tracks the A, B, and Z inputs and sends pulses to the **CR** register. 1x, 2x, and 4x modes are available for the quadrature clock.

All four modes may be marked as triggerable by the trigger event (pulse on the TRIGIN input) or re-triggerable (stops on EM detection and restarts on trigger), which effectively creates 12 different modes of operation for the CR register.

1.4.2 Input Modes

In addition to its application as a quadrature decoder, the DNA-QUAD-604 can also be used for counting events and for measuring pulse periods. As an option, an interrupt can be generated when **CR** matches **CR0** or **CR1**.

1.4.3 Output Pins

Each of the four Quadrature Detector Units of the DNA-QUAD-604 has two dedicated output pins that may be used as general purpose digital outputs or as an clock line (CLKOUT) output or an output trigger (TRIGOUT) line.

1.5 Device Architecture

The DNA-QUAD-604 layer consists of two PCBs: the main board and a power and isolation board. All inputs and outputs of these layer/boards are electrically isolated and overvoltage protected.

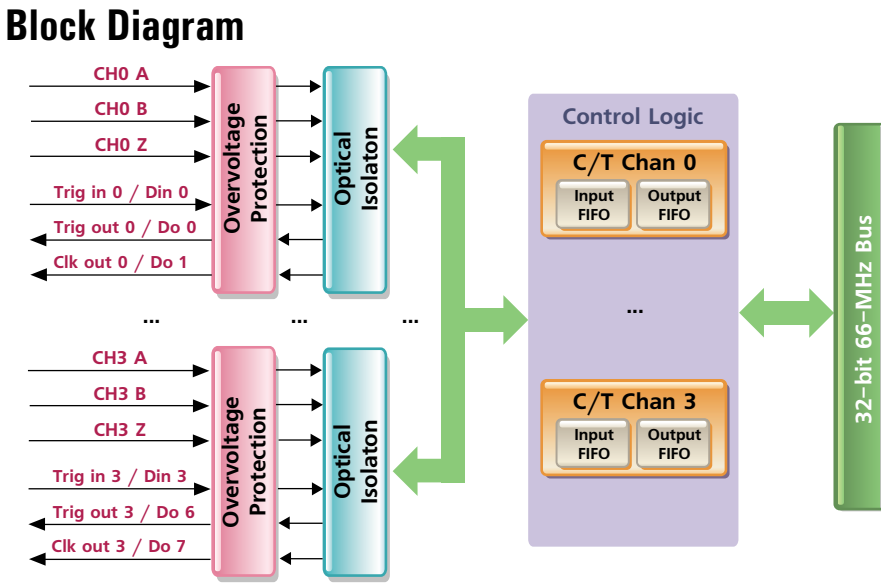


Figure 1-4. DNA-QUAD-604 Block Diagram

As shown in the block diagram, each encoder sensor produces three signal inputs to the 604 layer – A, B, and Z. The A and B signals are trains of pulses that are displaced from each other by 90° and vary in frequency with the speed of encoder rotation. The phase difference between the A and B signals indicates in which direction the encoder is moving. The number of pulses received per unit time indicates the velocity of the encoder; the rate of change gives the acceleration. The Z signal is an index pulse that gives a specific position of the encoder shaft as a reference.

Pulses received from each encoder are accumulated in a 32-bit counter/timer unit. Each reading is stored in the FIFO buffer for transfer to the bus at fixed (programmable) time intervals.

The digital inputs can be used to accept external event, alarm, trigger, and clock inputs. The digital outputs can generate similar types of signals for external devices.

1.6 Layer Connectors and Wiring

As standard with other PowerDNA Layers, the QUAD-604 uses a B-size 37-pin D-sub connector with pinout as shown in **Figure 1-5**. All signals are referenced relative to isolated ground (IGND).

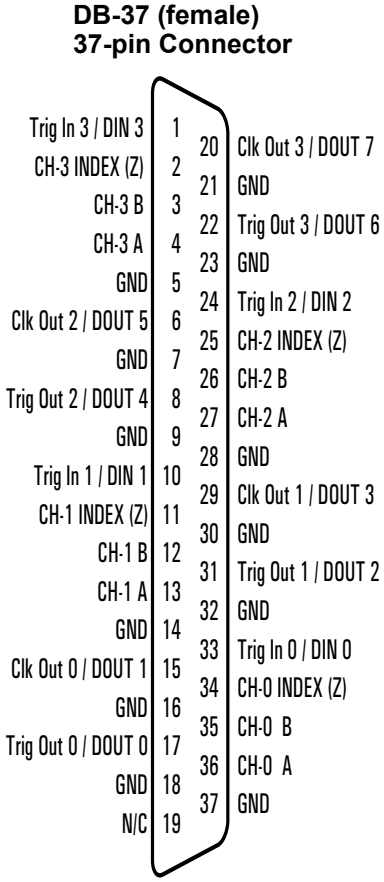


Figure 1-5. DNA-QAD-604 Pinout Diagram

The following inputs and outputs are connected to the QUAD-604 through the DB-37 connector, as follows:

CH-x A – Encoder pulse train “A” output signal for Channels 0 to 3.

CH-x B – Encoder pulse train “B” output signal for Channels 0 to 3, shifted 90° from the A signal. (Phase determines rotational direction of encoder.)

CH-x INDEX – Encoder index signal “Z” for Channels 0 to 3.

DIN x – Digital Input Channels 0 to 3.

DIOUT x – Digital Output Channels 0 to 7.

GND – Layer ground, isolated from system ground

N/C – Not connected

Chapter 2 Programming with the High Level API

This section describes how to program the PowerDNA DNA-QUAD-604 using the UeiDaq Framework (High Level) API.

Since UeiDaq Framework is object oriented, its objects can be manipulated in the same manner in various development environments such as Visual C++, Visual Basic or LabVIEW.

Although the following section focuses on the C++ API, the concept is the same no matter what programming language you use.

Please refer to the “UeiDaq Framework User Manual” for more information about using other programming languages.

Refer to the examples that come with the UeiDaq Framework. The examples contain detailed and commented code that compiles and executes.

2.1 Creating a Session

The Session object controls all operations on your PowerDNA device. Therefore, the first task is to create a session object, as follows:

```
CUeiSession session;
```

2.2 Configuring Channels

Framework uses resource strings to select which device, subsystem and channels to use within a session. The resource string syntax is similar to a web URL:

```
<device class>://<IP address>/<Device Id>/<Subsystem><Channel list>
```

For PowerDNA, the device class is **pdna**.

For example, the following resource string selects counter input line 0 on device 1 at IP address 192.168.100.2:

```
“pdna://192.168.100.2/Dev1/Ci0”
```

Use the Session object’s method “CreateCIChannel” to configure the counter/timers you wish to use in input mode.

You can select the following mode for input operations:

```
UeiCounterModeQuadratureEncoder: Measure the position of a quadrature
encoder.
```

The source of the input signal can be configured to come from the A, B, and Z (index) outputs of the quadrature encoder.

You can specify when the counting operation starts and stops using an external or a software gate.

You can invert and/or divide the source signal before performing the counting operation.

```
// Configure counter 0 to measure quadrature encoder position.
// Don’t divide or invert the input signal.
session.CreateCIChannel("pdna://192.168.100.2/Dev0/ci0",
                        UeiCounterSourceInput,
                        UeiCounterModeQuadratureEncoder,
                        UeiCounterGateInternal,
                        1, false);
```

2.3 Configuring the Timing

You can configure the DNA-QUAD-604 to run in simple mode (point by point) or buffered mode. In simple mode, the delay between samples is determined by software on the host computer. In buffered mode, the delay between samples is determined by the on-board clock.

The following sample shows how to configure the simple mode. Please refer to the “UeiDaq Framework User’s Manual” to learn how to use the other timing modes.

```
session.ConfigureTimingForSimpleIO();
```

2.4 Reading Data

Reading data from the DNA-QUAD-604 is done using a reader object. There is a reader object to read raw data coming straight from the counter line.

The following sample code shows how to create a scaled reader object and read samples.

```
// Create a reader and link it to the session's stream
  CUeiCounterReader reader(session.GetDataStream());
// read one scan
 unsigned short countedEvents;
 reader.ReadSingleScan(&countedEvents);
```

2.5 Cleaning-up the Session

The session object cleans itself up when it goes out of scope or when it is destroyed. However, you can manually clean up the session (to reuse the object with a different set of channels or parameters).

```
  session.CleanUp();
```

Chapter 3 Programming with the Low-Level API

This section describes how to program the PowerDNA cube using the low-level API. The low-level API offers direct access to PowerDNA DAQBios protocol and also allows you to access device registers directly.

We recommend, however, that you use the UeiDaq Framework High Level API (*see Chapter 2*), because it is easier to use. You should need to use the low-level API only if you are using an operating system other than Windows.

3.1 Data Representation

Counter data is represented as 32-bit words.

3.2 Configuration Settings

Configuration settings are passed in `DqCmdSetCfg()`.

Note that not all configuration bits apply to the QUAD-604 layer.

The following bits are used:

```
#define DQ_LN_MAPPED     (1L<<15)  // For WRRD (DMAP) devices
                                      //(automatically selected)
#define DQ_LN_ACTIVE     (1L<<1)   // "STS" LED status
#define DQ_LN_ENABLED    (1L<<0)   // enable operations
```

The DQ_LN_ACTIVE flag is needed to switch on "STS" LED on CPU layer.

The DQ_LN_ENABLE flag enables all operations with the layer.

3.3 Channel List Settings

Channel list settings are not currently supported.

3.4 Layer-specific Commands and Parameters

The DNA-QUAD-604 layer API provides an extensive set of functions to control quadrature encoder counters, as listed below. All these functions are designed to set up specific single-point modes of operation. Most of them work directly with DNA-QUAD-604 registers. Refer to the register definitions for details. See the PowerDNA API Reference Manual for a complete description of these functions.

- `DqAdv604SetRegister()`
  This function writes a `value` to the selected `reg` register. `reg` specifies the relative offset of the register of interest.
- `DqAdv604GetRegister()`
  This function reads `value` from the selected `reg` register. `reg` specifies the relative offset of the register of interest.
- `DqAdv604EnableAll()`
  This function enables all encoder counters on the layer.
- `DqAdv604DisableAll()`
  This function disables all encoder counters on the layer.

- `DqAdv604StartCounter()`
  This function starts the specified encoder counter with the given configuration.
- `DqAdv604StopCounter()`
  This function stops the specified encoder counter with the given configuration.
- `DqAdv604ClearCounter()`
  This function disables the specified encoder counter and clears its configuration.
- `DqAdv604ReadRegisterValue()`
  This function reads the value from a register `reg` of the encoder `counter` relative to layer `devn`.
- `DqAdv604ConfigCounter()`
  This function sets up an encoder counter configuration.
- `DqAdv604CfgForBinCounter()`
  This function sets up the configuration for an encoder counter in a 604 layer for getting the number of counts during a timeframe. Read register for an immediate measurement.
- `DqAdv604CfgForQuadrature()`
  This function sets up the configuration for an encoder counter in a 604 layer.

Appendices

A - Accessories

The following cables and STP boards are available for the DNA-QUAD-604 layer.

DNA-CBL-37S
3ft, 37-way round extender cable with thumbscrew connectors on both ends; connects DNA-QUAD-604 to screw termination panels and other devices.

DNA-STP-37
37-way screw terminal panel.

B. EEPROM Structure

The following structure represents content of the layer EEPROM:

```
/* combined structure to be allocated after CMNDEVS */
typedef struct {
    DQEECMNDEVS ee;
    DQOPMODEPRM_604_ opmodeprm;
    DQCNAMES_604_ cname;
} DEVEEPROM_604_, *pDEVEEPROM_604_;
```

`DQEECMNDEVS` is a standard EEPROM header. See the API Reference Manual for details.

`DQOPMODEPRM_604_` contains data for operating mode. This data is pre-loaded into the working array on switching to configuration mode and can be overwritten before going into operating mode.

```
typedef struct {
    uint32 conf;       // control word - layer API flags
    uint32 cvclk;      // CV clock
    uint32 clclk;      // CL clock
    uint32 trig;       // trigger conditions
    int clperint;      // number of channel lists per interrupt; ignored if
<1 or invalid
} DQOPMODEPRM_604_, *pDQOPMODEPRM_604_;
```

Channel names are stored in the following structure (up to 16 characters long):

```
typedef struct {
    char cname[DQ_PL_604_CHAN][DQ_PL_604_NAMELEN];
} DQCNAMES_604_, *pDQCNAMES_604_;
```

User can set and store these parameters using `DqCmdSetParameters()`. See the API Reference Manual for details.

PowerDNA Explorer provides graphical interface for program startup and shutdown states as well as names and operation mode parameters.

```
typedef struct {
    uint32 conf;       // control word - layer API flags
    uint32 cvclk;      // CV clock
    uint32 clclk;      // CL clock
    uint32 trig;       // trigger conditions
```

```
    int clperint;      // number of channel lists per interrupt; ignored if
<1 or invalid
} DQOPMODEPRM_604_, *pDQOPMODEPRM_604_;
```

Channel names are stored in the following structure (up to 16 characters long):

```
typedef struct {
    char cname[DQ_PL_604_CHAN][DQ_PL_604_NAMELEN];
} DQCNAMES_604_, *pDQCNAMES_604_;
```

User can set and store these parameters using `DqCmdSetParameters()`. See the API Reference Manual for details.

PowerDNA Explorer provides graphical interface for program startup and shutdown states as well as names and operation mode parameters.

Index