

Prima Logic Language

Your CDQPrima has a powerful micro-processor which is made available to the user with Prima Logic Language. Using Prima Logic Language, the user can program automatic functions for configuration and remote control.

7. PLL Basics

Prima Logic Language makes the **CDQPrima's** powerful micro-processor available to the user for automatic control of internal and external functions. Using Prima Logic Language, it is possible to configure the **CDQPrima** to automatically dial, configure, send alarms, start or stop external devices, etc., depending on certain internal or external conditions.

Prima Logic Language is based on the premise that certain user defined 'events' should be translated into user defined actions. The conversion of events, such as silence detection, into actions, such as relay closures, is handled by Prima Logic Language (PLL). PLL is a simple but powerful language based on this 'event-to-action' logic designed specifically for **CDQPrima** monitor and control.

At the **CDQPrima**, there are various inputs called *events*. One class of events are switch closures, which can be either internal 'virtual' switches, or external switches. Other events are based on detectors, such as silence detection, error detection, and status detection. These events are all binary in nature; i.e., on or off (high or low).

7.1 PLL Events

One of the more powerful features of all **CDQPrima** models is the ability to map events to actions. PLL is a rich language for mapping these

events to actions. A complex event can also be created using Boolean operators to link several simple events.

Examples of internal events include:

- Detected Bit errors or BER threshold exceeded
- Detected out of frame, or OOF threshold exceeded
- Silence detected on encoder or decoder, left, right, or stereo
- Internal timers started or stopped
- Encoder or decoder PLL locked
- Encoder or decoder audio PLL locked
- Failure of any of the four on-board DSP's
- DIF state

Examples of external events include:

- Input from up to eight (optional) opto-isolators
- Actions sent from far-end **CDQPrima**

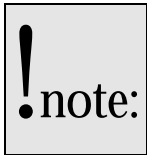
The following sections discuss the available 'events' that can be used as stimuli for PLL applications

7.1.1 Bit Error Rate Detection

Although formally not a part of Prima Logic Language, all **CDQPrima** models come standard with Bit Error Rate (BER) detection and display capabilities. This BER detection can be used as a stand-alone feature for monitoring line condition, but can also be used as an event input for Prima Logic Language.

<Maint><BER Det><Dsply Cnt>	MBC	DC	Display BER counter
<Maint><BER Det><Down Cnt>	MBD	DC	Set BER down count rate
<Maint><BER Det><Set Thrsh>	MBL	DC	Set BER count threshold (limit)
<Maint><BER Det><Set Peak>	MBP	DC	Set BER count maximum
<Maint><BER Det><Reset Cnt>	MBR	DC	Reset BER counter
<Maint><BER Det><Up Cnt>	MBU	DC	Set BER up count rate

The bit error rate detector provides a method of monitoring the number of errors in the digital transmission path. The bit error rate detector is used when ISO frame protection is enabled. When each ISO/MPEG frame is received (every 24 milliseconds for 48 kHz sampling), the header CRC is checked for validity. If the frame header has an invalid



CRC, then the BER counter is incremented by a BER up count (any number from 0 to 9). If the frame is valid, then the BER counter is decremented by the BER down count (any number from 0 to 9). The BER detector can only monitor the frame header, bit allocation and scale factor bits. Since there is no advanced knowledge of the actual audio data, the BER counter cannot count errors in the audio. Therefore, the BER detector tends to underestimate the actual errors.

If the BER up count is set to 1 (by the **MBU** command) and the down count is set to 0 (by the **MBD** command), then the BER counter counts the total number of frames in error. If the up counter is set to 2 and the down count is set to 1, then the BER counter is sensitive to burst errors but not random errors.

The BER counter is compared to the BER threshold (set by the **MBL** command) to see if the counter is above or below the threshold. The bit error threshold detector results in either a 1 or a 0 output. This output is called the BER event which can be used as a trigger for a Prima Logic Language action (see the **CEV** command). The 1 output signifies that the content of the bit error counter exceeds the threshold set by the **MBL** command. A 0 signifies that the counter content is below the threshold value. When a BER event occurs, you can program your **CDQPrima** to take action, such as closing a relay, dialing a phone number, lighting an LED or displaying a scrolling message.

The BER counter can be reset to 0 by the **MBR** command. The current contents of the BER counter can be displayed by the **MBC** command, and the maximum count can be set by the **MBP** command.

Figures 7-1 and 7-2 show the contents of the bit error counter verses' time for various up and down count rates. The occurrence of the bit errors is identical, even the error threshold is the same. The resulting bit error event is radically different for the two cases.

The out of frame detector provides a method of monitoring the number of framing errors that occurred in the digital transmission path. The OOF detector is used when ISO/MPEG types of frames are enabled by the **EAL** command. When each ISO/MPEG frame is received (every 24 milliseconds for 48 kHz sampling), the header CRC is checked for validity. If the frame header has invalid framing bits, then the OOF counter is incremented by a OOF up count (any number from 0 to 9). If the frame header bits are valid, then the OOF counter is decremented by the OOF down count (any number from 0 to 9). The OOF detection function is similar to the Bit Error Detector described in the previous section.

If the OOF up count is set to 1 (by the **MOU** command) and the down count is set to 0 (by the **MOD** command), the OOF counter counts the total number of frames in error. If the up counter is set to 2 and the down count is set to 1, then the OOF counter is sensitive to burst errors but not random errors.

The OOF counter is compared to the OOF threshold (set by the **MOL** command) to see if the counter is above or below the threshold. An out of frame event is generated whenever the threshold is exceeded. Actions such as closing a relay, dialing a phone number, lighting an LED or displaying a scrolling message can be taken based on this event.

The OOF counter can be reset to 0 by the **MOR** command. The current contents of the OOF counter can be displayed by the **MOC** command, and the maximum count can be set using the **MOP** command.

7.1.3 Silence / Audio Detection

<Maint><Quiet Det><Time left>	MQC	DC	Display silence detector level time remaining
<Maint><Quiet Det><Read lvl>	MQD	DC	Display silence detector level
<Maint><Quiet Det><Set lvl>	MQL	DC	Set silence detector level
<Maint><Quiet Det><Set time>	MQT	DC	Set silence time duration

One of the most powerful features of the **CDQPrima** is the availability of six silence detectors. The silence detectors, coupled with Prima Logic Language make automatic, unattended operation possible for applications such as STL backup and others.

There are six silence detectors. These are:

- Encoder left channel input
- Encoder right channel input
- Encoder stereo input
- Decoder left channel output
- Decoder right channel output
 - Decoder stereo output


A stereo silence detector uses the greater of the left or right channel signal for the determination of silence.

At 0.1 second intervals, the audio levels of the encoder and decoder left and right channels are measured. If the level is below the value set by the **MQL** command for a period of time as set by the **MQT** command, then the channel is said to be silent. When a channel is silent, the silent event input is set to true. The value of the silence event may be used as an input for the Event-to-Action logic interpreter and can initiate a PLL action or virtual action.

The current value of any of the six silence detectors can be displayed by the **MQD** command. The time remaining before a silence detection event can be displayed by the **MQC** command.

Remembering that any event can be mapped into other events using Boolean operators, by simply NOTing a silence detector, you create an audio detector. Using this 'audio detector' is easy to make the **CDQPrima** execute any command, including dialing into a remote unit, on the presence of audio.

7.1.4 Timers

<Common><PLL><Stop timr>	CCT		Cancel timer
<Common><PLL><Set timr>	CTM		Set timer time-out duration

There are four internal timers available to the PLL programmer. Although the timers are stopped and started using PLL 'actions', discussed later, the state of the timers are PLL 'Events'. There are eight 'events' associated with these four timers:

TI0 = timer 0 running
 TI1 = timer 1 running

- TI2 = timer 2 running
- TI3 = timer 3 running
- TS0 = timer 0 just expired
- TS1 = timer 1 just expired
- TS2 = timer 2 just expired
- TS3 = timer 3 just expired

7.1.5 Optically Isolated Inputs (option)


Up to eight (4 on 100 Series and 8 on 200 Series Models) opto-isolator inputs are optionally available on the **CDQPrima**. These inputs require standard TTL level logic for operation. The optically isolated inputs can therefore be triggered from any TTL level source, or can be triggered from external switches or relays using a pull-up configuration as shown in Appendix D. The voltages needed for pull-up are available right on the rear panel connector. This feature can assist in the monitoring of peripheral equipment at unmanned locations, and make the **CDQPrima** ideal for unattended and automatic backup of an STL.

Any change in state of any optical input can be used as an event. In addition, the same optical input can be used as two different event stimuli depending on the swing of the input: positive to negative is a different event than negative to positive.

- OI0 = optical isolator input 0
- OI1 = optical isolator input 1
- OI2 = optical isolator input 2
- OI3 = optical isolator input 3
- OI4 = optical isolator input 4
- OI5 = optical isolator input 5
- OI6 = optical isolator input 6
- OI7 = optical isolator input 7

Installation of the optional opto-isolator board is detailed in the 'Options Installation' chapter of this manual. Connector pin-out information can be found in Appendix D.

7.1.6 Simulated Switches

<Encoder><More><Contacts> <Set Swtch>	ESW		Set a simulated switch
--	-----	---	------------------------

CI0 through CI7 can be thought of as imaginary switches and can be classified as both ‘events’ and ‘actions’. These are software switches, capable of triggering events, that are controlled by other events, such as virtual actions, or remote commands.

CI1 and CI2 have been pre-assigned by the factory, but can be re-defined. Pressing the front panel ON and OFF CUE buttons sets and clears Event CI1 and CI2. Front panel CUE button 1 corresponds to Event CI1 while button 2 corresponds to Event CI2

7.1.7 Status Event

Status events are an important class of events that can be used to link the status of the various sections of the **CDQPrima** to PLL actions. Status events include alarms and ‘normal’ conditions:

DDAPL	decoder digital audio Phase Lock Loop
EDAPL	encoder digital audio Phase Lock Loop

These two events are defined by the state of the digital audio Phase Lock Loop. Phase lock is required for the digital audio circuits to function. Unlike silence detection, which may not activate for a user defined time interval, if digital audio phase lock is lost, which would result in silence, the event is immediate.

EPL	encoder PLL locked
DPL	decoder PLL locked

In order for the **CDQPrima** to operate, both encoder and decoder phase lock must be established. Events can be triggered if this lock is not established or is lost.

DSPD	decoder DSP dead
DSPE	encoder DSP dead
DSPR	Reed-Solomon DSP dead
DSPV	VU meter DSP dead

These four events monitor the state of various digital signal processors. If any of these processors malfunction, audio is lost. Therefore, these events can be used for sanity checking.

FRAMED	decoder framed
--------	----------------

This event is true as long as the decoder is framed.

CD1	carrier detect for DIF 1
CD2	carrier detect for DIF 2

CD3	carrier detect for DIF 3
CD4	carrier detect for DIF 4
CD5	carrier detect for DIF 5
CD6	carrier detect for DIF 6

The truth of these events mimics the state of the internal Terminal Adapter TA101, TA201 or TA301. If a call is received, the carrier detect goes high, and if a line is dropped, the carrier detect goes low.

7.1.8 Link Event


LN0	imported action from far end CDQPrima on link 0
LN1	imported action from far end CDQPrima on link 1
LN2	imported action from far end CDQPrima on link 2
LN3	imported action from far end CDQPrima on link 3
LN4	imported action from far end CDQPrima on link 4
LN5	imported action from far end CDQPrima on link 5
LN6	imported action from far end CDQPrima on link 6
LN7	imported action from far end CDQPrima on link 7
LN8	imported action from far end CDQPrima on link 8
LN9	imported action from far end CDQPrima on link 9
LN10	imported action from far end CDQPrima on link 10
LN11	imported action from far end CDQPrima on link 11

There are 12 virtual links to/from the far end **CDQPrima**. An event on any one of these links is caused by an action on the far end **CDQPrima**. It is therefore these 12 links that make far-end PLL command execution possible. The factory 'actions' associated with the link events are as follows:

LN0...LN7	Relays RL0...RL7
LN8	Receive Cue 1 (RCUE1) LED
LN9...LN11	Undefined

and can be re-defined at any time by the user.

7.2 PLL Actions

<Common><Status><LS Clear>	CAR		Clear the latched value of the action word
<Common><Status><LS Read>	CLA		Print latched value of the action word
<Common><Status><RS Read>	CRA		Print real-time value of the action word

The **CDQPrima**'s ability to control not only itself, but peripheral equipment is based on three different classes of PLL 'actions'. Physical actions, those actions which actually do something, such as closing a relay or illuminating an alarm, link actions and 'Virtual Actions'. Virtual actions are the automatic execution of any valid **CDQPrima** command or groups of commands, and are discussed later.

Actions are also binary in nature, and thus are either true or false (high or low, on or off). This means that the output Action will do something, such as open or close a relay, light or extinguish an LED, or execute some command. A real-time snapshot of the Actions can be seen by executing the **CRA** command. The Actions are also latched. The latched values are read by the **CLA** command and are cleared by the **CAR** command. The purpose of the latched Actions concept is to see if an Action occurred anytime in the past. This allows the detection of transient Actions (Actions which occur and then disappear). One feature of the PLL is the ability to convert a binary action into a command, or *Virtual Action*.

Action can also be the result of transitions of an Event or Event Expression from low to high or high to low. For example, the LED display might scroll a text a message when the silence detector goes from audio present (not silent) to no audio present (silent).

7.2.1 Physical Actions

Possible physical actions available with Prima Logic Language are listed below:

- RL0 = relay 0 contact closure
- RL1 = relay 1 contact closure
- RL2 = relay 2 contact closure
- RL3 = relay 3 contact closure
- RL4 = relay 4 contact closure
- RL5 = relay 5 contact closure
- RL6 = relay 6 contact closure
- RL7 = relay 7 contact closure

Up to eight relay outputs are optionally available on **CDQPrima** models. These are 'dry contact' relays, which can be used to control external events, alarms, and equipment. For example, a relay can be connected to a tape recorder, which can then be turned on or off from the far end of some local condition. Pull-up and pull-down voltages are available

right on the rear panel connector. The connector pin-outs and pull-up resistor configurations are shown in Appendix D. The factory default 'event' trigger for the optional relays are Link Event LN0 through LN7.

SC1 = send cue LED
 RC1 = receive cue LED

The send and receive cue LEDs are front panel indicators that can be lit or extinguished locally or from the far end. These LEDs are useful for operator signaling. The front panel LEDs labeled

SCUE1	(SC1)
RCUE1	(RC1)
ESUM	(ESM)
DSUM	(DSM)

are actually defined by Prima Logic Language. They are illuminated based on the Action shown in parentheses in the above list. This means that the state of these LEDs plus all the relays are completely user definable and can be changed to meet the needs of different applications.

RLS = summary alarm relay

All **CDQPrima** models have a Summary Alarm relay that can be user defined. The factory default definition of the summary alarm is: encoder and decoder phase lock, no errors or out of frame conditions. This relay is user re-definable and can be activated either locally or from the far end **CDQPrima**.

There are four output conditions associated with this single relay:

Normal 1	— power on AND no summary alarm
Alarm 1	— power off OR summary alarm
Normal 2	— power on AND no summary alarm
Alarm 2	— power off OR summary alarm

Pin-outs for the summary alarm connector can be found in Appendix B.

ESM	encoder summary alarm
DSM	decoder summary alarm

The encoder and decoder summary alarms, although only 'simulated' alarms, i.e. there is no physical relay associated with them, can be defined as PLL actions, and mapped into other events or actions. The factory defaults of these alarms are listed in Appendix C.

Another physical action is the starting of any one of four timers (see the **CTM** command). The expiration of a timer is an input event. Timers can be canceled by the **CCT** command.

7.2.2 Link Actions

N/A	ELU	DC	Set link message update rate
-----	-----	----	------------------------------


Actions can be exported to a far end **CDQPrima**. This exported action appears as an input event to the far end **CDQPrima**.

- LN0 = action exported to far end **CDQPrima** on link 0
- LN1 = action exported to far end **CDQPrima** on link 1
- LN2 = action exported to far end **CDQPrima** on link 2
- LN3 = action exported to far end **CDQPrima** on link 3
- LN4 = action exported to far end **CDQPrima** on link 4
- LN5 = action exported to far end **CDQPrima** on link 5
- LN6 = action exported to far end **CDQPrima** on link 6
- LN7 = action exported to far end **CDQPrima** on link 7
- LN8 = action exported to far end **CDQPrima** on link 8
- LN9 = action exported to far end **CDQPrima** on link 9
- LN10 = action exported to far end **CDQPrima** on link 10
- LN11 = action exported to far end **CDQPrima** on link 11

LN0...LN11 are actions exported to the far end **CDQPrima** on virtual links, in contrast to physical actions which are performed locally.

The exported actions are sent to the far end whenever the event mapped into the link changes state or whenever a link timer has expired. This link timer interval is set by the **ELU** command. The result is that the exported actions are repeatedly sent to the far end even if no change has occurred. This is an attempt to communicate with the far end even in the presence of bit errors on the transmission line.

7.2.3 Virtual Actions

<Common><Virt Act>	CVA		Define virtual action
--------------------	-----	---	-----------------------

Virtual Actions can be considered command macros, or scripts, that are executed when an event, defined by the user, becomes true. A Virtual Action can be almost any remote control command, or even combinations of several commands. In addition, several commands can be combined into one Virtual Action using Boolean operators. Remote control commands can also be sent to the far end **CDQPrima**. Examples of Virtual Actions will be shown later in this chapter. With software revision 21 and later, Virtual Actions can now be redefined recursively.


This means that you can now embed or redefine a Virtual Action within a Virtual Action.

There are 4 Virtual Actions available. Unlike Physical actions and link actions, which close relays or alarms on the local or far-end **CDQPrima**, Virtual Actions execute pre-defined programs on either the local or far-end **CDQPrima**. Programs that can be executed include reconfiguration, dialing or re-dialing, switching input source, etc.

Inside a Virtual Action, a single semi-colon (;) is used as a command separator, that is, two commands can be executed by Virtual Actions. However, if you are defining a Virtual Action to execute two commands *inside* of a Virtual Action, then you use two semi-colons (;;) as a command separator as follows:

Virtual action 1: (DO THIS; and DO THAT)
 Virtual action 2: (reassign virtual action 1 to (DO THIS;; and DO THAT))

7.3 Event To Action Logic

<Common><PLL><Program>	CEA		Set event-to-action logic
<Common><PLL><Prt evnt>	CEV		Print event inputs

The **CDQPrima** contains a rich language for mapping input events or remote signaling into actions that can either execute internal commands or send signals to external devices.

Actions and events can be combined to form complex conditions using a full set of Boolean operators.

The current state of all available events are displayed by the **CEV** command. These input events may be physical inputs from external devices, such as input optical isolators, or logical inputs, such as computer generated switch closures (see the **ESW** command).

The mapping of input events into output actions is controlled by the **CEA** command.

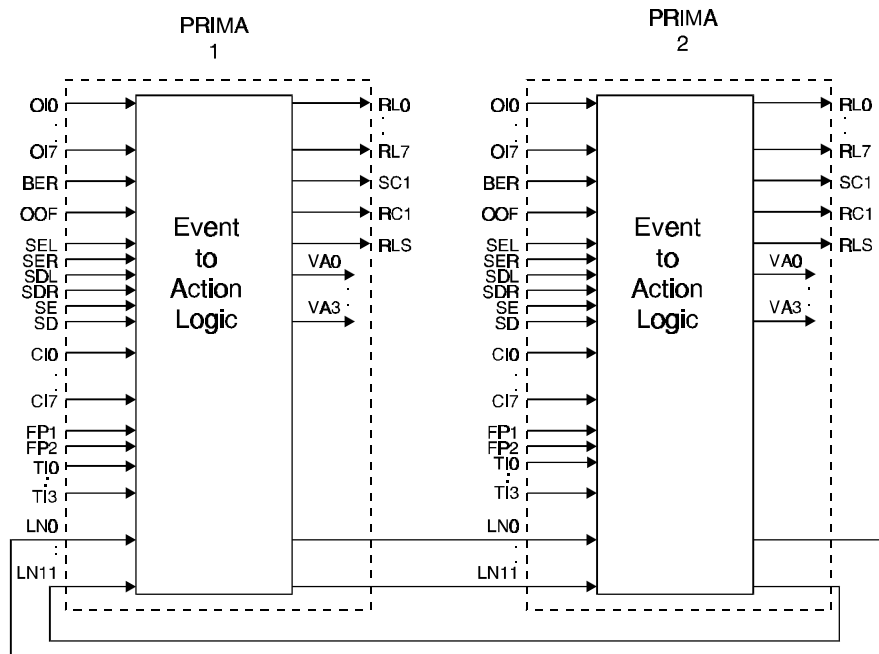
Action Words are the output of the Event-to-Action logic that is controlled by the Prima Logic Language (PLL). The real-time value of the Action Word is displayed by the **CRA** command while the latched

value of the Action Word is displayed by the **CLA** command. The latched Action Word values are reset by way of the **CAR** command.

Events are mapped into actions by Boolean logic which includes AND, OR and NOT operators. The group of events joined by the Boolean operators is called an *Event Expression*. The real-time values of the various events can be displayed by executing the **CEV** command, discussed in Chapter 5.

Approximately every 0.01 second, each Event Expression associated with an Action is evaluated and the corresponding Action is set true or false.

The Figure below shows the complete interconnection of two **CDQPrimas** units and shows all the possible events and actions.



Events

- OI0..OI7 - Optical Isolated / TTL inputs.
- BER - Bit error rate detector
- OOF - Out of frame detector
- SEL - Encoder left channel silence detector
- SER - Encoder right channel silence detector
- SDL - Decoder left channel silence detector
- SDR - Decoder right channel silence detector
- SE - Encoder stereo silence detector
- SD - Decoder stereo silence detector
- CI0..CI7 - Computer simulated switch closures
- FP1..FP2 - Front panel cue push buttons 1 and 2
- TI0..TI1 - Timers 0 and 1
- LN0..LN11 - Links from far end PRIMA

Actions

- RL0..RL7 - Relay closures
- SC1 - Send cue LED
- RC1 - Receive cue LED
- RLS - Summary relay closure
- VA0..VA3 - Virtual actions
- LN0..LN11 - Link to far end PRIMA

Figure 7-1 Event/Action interconnections

As this figure illustrates, any event, either internal or external, can be mapped into any action. In addition, actions and events can be combined into complex actions and events using logical operators. The logical operators are, in order of decreasing precedence:

- ! = NOT
- & = AND
- # = OR

An expression can have a maximum of 4 OR terms. If more than 4 OR terms are needed, a simple technique called DeMorgan's Theorem can be used to change ORs to ANDs. There is no limit on AND terms.

DeMorgan's Theorem states:

$$A = B \# C$$

is equivalent to:

$$A = !(B \& C)$$

Thus the equation which contains five OR terms, which is not allowed in PLL:

$$RLS = CI0 \# CI1 \# CI2 \# CI3 \# CI4 \# CI5$$

can be rewritten as:

$$RLS = !(CI0 \& CI1 \& CI2 \& CI3 \& CI4 \& CI5)$$

using DeMorgan's Theorem. The current PLL implementation only allows parentheses around an entire expression. The full use of the AND operators will be allowed in future releases.

The +, - and @ (positive edge, negative edge or any edge) introduce the concept of actions based on transitions (edges). For example, if an LED message should occur when the front panel CUE 1 ON button is pressed, then the following PLL commands can be used:

Example 1:

```
CVA 0 CLM 10 HELLO WORLD
CEA VA0 +(CI1)
```

The first statement (**CVA 0**) sets Virtual Action 0 to execute the '**CLM 10 HELLO WORLD**' (Display the message HELLO WORLD for 10 seconds) command. The second line states (**CEA** - Set Event to Action) that when computer input 1 (the CUE 1 button) changes from low to high (a + edge), then Virtual Action 0 is set to a 1 (executed).

Please note, however, that the +, - and @ modifiers are valid only for an entire expression, and cannot be used as individual term modifiers in a complex expression. For example, the complex event:

```
+(CD1 & !FRAMED)    is valid
while
+CD1 & !FRAMED      is not valid.
```

The LED message programmed in Example 1 can be immediately suppressed when the CUE 1 OFF button is depressed if the following additional PLL statements are executed:

Example 2:

CVA 1 CLM 0	(Assign Virtual Action 1 to turn off message)
CEA VA1 -(CI1)	(When CI1 changes from high to low, execute Virtual Action 1)

The first PLL statement associates the CLM 0 command with Virtual Action 1. The second statement activates VA1 on the high-to-low transition (- edge) of the CUE 1 indication.

See Appendix C for the default settings of the CEA commands. Note that RL4 .. RL7 are not available on the 100 Series codecs, but they are allowed in the PLL even though they have no effect.

The ! operator can be used in conjunction with the +, - and @ operator. For example:

RL0 = OI0	Relay 0 follows the level of optical input 0
RL0 = !OI0	Relay 0 follows the inverted level of optical input 0
RL0 = +OI0	Relay 0 closes when OI0 changes from a low to a high (note: there is no way to open relay 0 when closed this way)
RL0 = -OI0	Relay 0 closes when OI0 changes from a high to a low (note: there is no way to open relay 0 when closed this way)
RL0 = +!OI0	Relay 0 closes when OI0 changes from a high to a low (note: there is no way to open relay 0 when closed this way)

Let us look at another simple example. Optically isolated input 2 is connected to link 5 by the command:

Example 3:

CEA LN5 OI2

This means that when optically isolated input 2 becomes a 1, then link 5 becomes a 1 and when OI2 becomes a 0, then link 5 is 0. Remember that the Link 5 action is exported to the far end **CDQPrima** and becomes an

input event. More on this below. For now, we will just concentrate on the language.

The inverted opto-isolator input 2 is connected to link 5 by the command:

Example 4: **CEA LN5 !OI2**

This means that when optically isolated input 2 becomes a 0, then link 5 becomes a 1 and when OI2 becomes a 1, then link 5 is 0.

Since Virtual Actions are evaluated every 0.01 second, the following PLL statement produces an unexpected result.

Example 5: **CEV VA0 CI0** execute VA0 when CI is high

As long as CI0 is high, then once every .01 seconds, Virtual Action 0 is executed, resulting in continuous execution. What is probably meant by this expression is that when CI0 *changes* from a low to a high, then Virtual Action 0 should be executed. To do this, you must specify the edge transition:

CEV VA0 +CI0

The next example is slightly more complicated:

Example 6: **CEA LN7 !(OI0 # CI1)**

This states that link 7 will be low when either OI0 is high or CI1 is high.

Another example is using the front panel receive cue RCUE1 LED to indicate the real-time state of the encoder stereo silence. This is accomplished by the following command:

Example 7: **CEA RC1 SE** set receive cue led to follow encoder silence detector

The above command is useful when trying to debug the correct settings for the silence detector. A similar trick can be used for the BER or OOF detectors.

To reset the received cue led back to its original definition, type:

CEA RC1 LN8

An interesting example of the power of the PLL is shown below. The audio output can be muted when the BER detector raises to a high level, even if the decoder is framed.

Example 8:

```

CVA 0 DMU BOTH
CEA VA0 +BER
CVA 1 DMU NORM
CEA VA1 -BER

```

The first command attaches the **DMU BOTH** (mute both decoder output channels) command to Virtual Action 0. Virtual Action 0 is executed when the BER detector transitions from low to high. The third line attaches the "restore DA output to normal" command to Virtual Action 1. The fourth line states that when the BER detector goes from a high BER count to a low BER count, then Virtual Action 1 is executed.

A further PLL example is shown below:

Example 9:

```

CEA LN10 OI2 & SD # !CI3

```

In this example, link 10 is set high if *either* **OI2** AND **SD** are high OR if **CI3** is low. Remember that **&** is higher precedence than **#** so the above expression could be written:

```

CEA LN10 (OI2 & SD) # !CI3

```

Parenthesis are currently not allowed except around the entire expression. Although logically correct, the above example would be much clearer if parenthesis were allowed.

7.4 PLL Examples

The examples that follow have been implemented by many users and have shown to be quite useful for day-to-day operation. Remember, these are just examples, and can be tailored for particular applications.

7.4.1 Auto Detection of Incoming Call Format

Using the power of PLL coupled with the built-in 'Quick-configurations', it is possible to create a small PLL program to enable the **CDQPrima** to automatically determine the line format, algorithm, and bit rate of any device which dials into it. The example shown here can detect G.722 or MPEG Layer II algorithms, Single line, 2 line or multiple line H.221 line formats. The next example shows how to switch between MPEG Layer II and MPEG Layer III.

CEA VA0 +(CD1 & CI1)

Set virtual action 0 (VA0) to execute when the condition 'carrier detect 1' *and* SCUE 1 is high (true)

CEA VA1 +(CD1 & CI1 & TS1)

Execute VA1 when carrier detect 1 *and* SCUE 1 is on *and* timer 1 just expired.

CEA VA2 +(CD2 & FRAMED & TS0 & CI1)

Execute VA2 when the call on line 1 framed *and* a call came in on line 2 *and* timer 0 just expired *and* SCUE 1 is on.

CEA VA3 +(CD1 & !FRAMED & TS0 & CI1)

Execute VA3 when a call on line 1 does not frame *and* timer 0 just expired *and* SCUE 1 is on.

CVA 0 CSD 4;EAM JS;CTM 1 20

Define VA0 to:

- Load quick configuration 4 (H.221 BONDING, 6 lines)
- Set algorithm mode to joint stereo
- Start a 20 second timer (timer 1)

CVA 1 CSD 8;EBR A;CTM 0 5

Define VA1 to:

- Load quick configuration 8 (1 line, line 1, 48 kHz sampling), automatic bit rate select
- Start a 5 second timer (timer 0)

CVA 2 CSD 1;EBR A

Define VA2 to:

- Load quick configuration 1 (2 lines line 1 & 2), automatic bit rate select

CVA 3 CSD 7;EBR A

Define VA3 to:

- Load G.722 quick configuration, automatic bit rate select

How these 8 lines work is as follows:

- To activate the auto-detect function, enable CUE 1. To disable the auto-detect function, turn off CUE 1. On models that do not have CUE buttons (110 and 210) you must use the keypad

sequence <Encoder> <More> <Contacts> <Set Swtch> <CI1> <ON/OFF> to turn the auto detect function on or off.

- As soon as a call comes in, virtual action 0 sets the **CDQPrima** to H.221 BONDING line format, up to 6 lines, and starts a 20 second timer. H.221 bonding allows up to six 64 kb/s 'B' channels to be used simultaneously, and will rate adapt to the number of lines used, and defaults to stereo at more than 2 connected lines. H.221 BONDING does not support 56 kb/s connections.
- If the **CDQPrima** does not frame after 20 seconds using H.221 BONDING, then the single line quick configuration is loaded and a 5 second timer is started.
- If the **CDQPrima** frames to the above condition and a call does not come in on the second line, the program stops. If, however, a call comes in on the second line within 5 seconds, the **CDQPrima** reconfigures to 2 line mode.
- If only one line connects, and the 5 second timer times out before the **CDQPrima** frames, then the **CDQPrima** reconfigures to G.722 encoding.

This auto-detect function is very powerful, especially when you do not know in advance what type of device is calling you, or the called **CDQPrima** is in an unattended location. You can modify this function to suit your own needs, for example, G.722 or the H.221 configuration can be replaced by an MPEG Layer III configuration. There are some restrictions however:

- This will only work with Version 19 or later software on both the *calling* and *called CDQPrima*.
- On all **CDQPrima** models except the 230, you must use a terminal to enter the virtual actions. The larger display on Model 230 will allow you to use the keypad to enter the virtual actions. If you cannot get access to a terminal form programming, contact MUSICAM USA and we will program it for you.

- H.221 BONDING takes about 17 seconds to frame, single line and 2 line modes take almost 25 seconds to frame, and G.722 takes almost 30 seconds to frame.
- Due to the 5 second timer used, 2 line calls must use an auto dialer or Speed Dial entry. If the second line does not connect within 5 seconds of the first line, the called **CDQPrima** loads G.722.
- **You must turn the auto-detect function off to place a call.**
- If the calling codec requires the codec to be in the decoder independent mode, replace Quick Configuration 1 with Quick Configuration 27 and replace Quick Configuration 8 with Quick Configuration 25.

Note that in this last example, the semi-colon (;) was used to separate two or more commands to be executed within a virtual action. If, however, I was to define this virtual action inside another virtual action, I would need a double semi-colon.

7.4.2 Auto Detection of Incoming Call Algorithm - Layer II or Layer III

Using the power of PLL coupled with the built-in 'Quick-configurations', it is possible to create a small PLL program to enable the **CDQPrima** to automatically determine the line format, algorithm and bit rate of any device which dials into it. The example shown here can detect MPEG Layer II and MPEG Layer III. As in the previous example, CI1 (CUE 1 ON) is used to turn the function on or off.

CEA VA0 +(CD1 & CI1)

Set virtual action 0 (VA0) to execute when the condition 'carrier detect 1' *and* SCUE 1 is high (true)

CEA VA1 +(CD1 & CD2 & CI1)

Execute VA1 when carrier detect 1 *and* carrier detect 2 os on *and* SCUE 1 is on.

CEA VA2 +(CD1 & !FRAMED & TS0 & CI1)

Execute VA2 when the call on line 1 did not frame *and* timer 0 just expired *and* SCUE 1 is on.

CEA VA3 +(CD2 & !FRAMED & TS0 & CI1)

Execute VA3 when a call on line 2 does not frame *and* timer 0 just expired *and* SCUE 1 is on (This assumes that line 1 has already connected).

CVA 0 CSD 8; EBR A; CTM 0 5

Define VA0 to:

- Load Quick Configuration 8 (MPEG Layer II, Single line)
- Set bit rate to automatic
- Start a 5 second timer (timer 0)

CVA 1 CSD 1; EBR A; CTM 1 5

Define VA1 to:

- Load Quick Configuration 1 (MPEG Layer II, 2 line 48 kHz sampling), automatic bit rate select
- Start a 5 second timer (timer 1)

CVA 2 CSD 29; EBR A

Define VA2 to:

- Load Quick Configuration 29 (MPEG Layer III, single line)
- Set bit rate to automatic

CVA 3 CSD xx; EBR A

Define VA3 to:

- Load a user defined MPEG Layer III 2 line Quick Configuration
- Set bit rate to automatic

How these 8 lines work is as follows:

- To activate the auto-detect function, enable CUE 1. To disable the auto-detect function, turn off CUE 1. On models that do not have CUE buttons (110 and 210) you must use the keypad sequence <Encoder> <More> <Contacts> <Set Swtch> <CI1> <ON/OFF> to turn the auto detect function on or off.
- As soon as a call comes in, virtual action 0 sets the **CDQPrima** to Layer II, single line, automatic bit rate selection.
- If a second call does not come in, and the **CDQPrima** does not frame within 5 seconds, load a single line Layer III configuration, automatic bit rate selection.
- If 2 lines connect, load a 2 line MPEG Layer II configuration, automatic bit rate selection.

- If 2 lines connect but the **CDQPrima** does not frame within 5 seconds, load a 2 line MPEG Layer III configuration, set to automatic bit rate select.

This auto-detect function is very powerful, especially when you do not know in advance what algorithm or line format will be used by the calling **CDQPrima**, or the called **CDQPrima** is in an unattended location. There are some restrictions however:

- On all **CDQPrima** models except the 230, you must use a terminal to enter the virtual actions. The larger display on Model 230 will allow you to use the keypad to enter the virtual actions. If you cannot get access to a terminal for programming, contact MUSICAM USA and we will program it for you.
- Due to the 5 second timer used, 2 line calls must use an auto dialer or Speed Dial entry. If the second line does not connect within 5 seconds of the first line, the called **CDQPrima** may not frame.
- **You must turn the auto-detect function off to place a call.**
- You will need to use decoder independent Quick Configurations if the calling codec is a Telos Zephyr.

7.4.3 Dial On Audio - Disconnect On Silence

Using the power of PLL coupled with the silence detectors built into all **CDQPrima** models, it is possible to program your **CDQPrima** to automatically establish a connection simply by activating an audio source. An example is shown here:

- Set up the encoder silence detector by defining a silence event as audio at -70 dB down for a defined number of seconds:

```
MQL E -70
MQT E xxx      where xxx is the number of 1/10
                 second clock ticks.
```

- Create two virtual actions to dial a far end codec when audio is detected and to hang up when silence is detected:

CVA 0 CSD yy where yy is a speed dial ID number
CVA 1 CHU ALL

- Define the controlling event-to-action logic, remembering that the NOT of a silence detector is an audio detector:

CEA VA0 +(!SE & CI1)
CEA VA1 +(SE & CI1)

When set up this way, just run a test tone for a minute before your program material starts, and the **CDQPrima** will do the rest. Please note that if the program source is classical music, or other materials with long quiet periods, than the silence detector may need to be re-defined.

In this example, as in all previous examples, we are using the CUE 1 ON and OFF buttons to turn the function on and off.

7.4.4 Automatic STL Backup

Suppose you are using a **CDQPrima** over dedicated lines for your main STL. You can use the same **CDQPrima** using ISDN lines in the event that your dedicated lines fail. The **CDQPrima** can even switch over automatically. Just equip your 200 Series **CDQPrima** with an internal terminal adapter in addition to the dedicated interface. The PLL example shown here can be used to automatically back up the dedicated line with ISDN.

CVA 0 CSD xx where xx is a speed dial ID number
CEA VA0 +SE

In this example, the CVA command assigns the operation of speed dialing entry xx to the virtual action 0. The CEA command states that when the stereo encoder silence detector detects silence, indicating problems, then it sets virtual action 0 high. As just defined, the virtual action 0 would then perform the speed dial. The ability to establish an alternate connection when a connection fails for any reason is a powerful feature of PLL. At the studio side, you can use the carrier detect indications to signal the studio **CDQPrima** to switch to the TA lines. You can even use PLL to switch back to the main feed when the studio drops the ISDN lines.

You can also use a **CDQPrima** to back up your main STL, even if it is not a **CDQPrima**.

Most STL systems incorporate external alarm monitoring capabilities in case of failures, and a **CDQPrima** equipped with optional opto-isolated inputs and relays can take advantage of these external inputs.

- Create two virtual actions to dial a far end codec when an external alarm is detected and to hang up when the alarm clears:

```
CVA 0 CSD xx    where xx is a speed dial ID number
CVA 1 CHU ALL
```

- Define the controlling event-to-action logic, monitoring the state of the alarm leads connected to opto-input 0:

```
CEA VA0 +OI0
CEA VA1 -OI0
```

In this case, the external alarm, which is normally low, activates virtual action 0, which is programmed to dial the studio, as soon as the state changes from normal (low) to alarm (high). When the STL alarm condition clears (returns to low), the back up connection is terminated.

7.4.5 Automatic Layer III Line and Bit Rate Fallback

Some Layer II codecs, including all MUSICAM USA and CCS Audio Products codecs, automatically sense when there is a line failure when using a 2 line configuration. This allows automatic falling back to a single line configuration, without loss of audio.

When using Layer III with 2 lines, when a single line is lost, audio is lost permanently. No Layer III codec on the market has the capability to automatically fall back to a single line Layer III configuration, until now.

Using the power of Prima Logic Language, you can program any **CDQPrima** model to automatically fall back to a single ISDN 'B' channel and a lower bit rate should a line fail when using Layer 3. In addition, this function is automatic, requiring no intervention once programmed. The example shown here illustrates just how easy it is.

- Create three Quick Configurations for the algorithm setups — 2-line, line 1 only and line 2 only.

- Define virtual actions to load the selected configurations.
- Define event-to-action statements to determine the number of connected lines and execute the appropriate Quick Configuration.

1. Create three Quick Configurations as follows:

Encoder/decoder algorithm: MPEGL3
 Encoder/decoder bit rate: 112 or 128 kb/s (use the highest available for your connection)
 Sample rate: 32 or 48 kHz
 Encoder algorithm mode: Joint stereo
 Encoder/decoder line format: CCSL12
 Decoder independent: YES

The CDQPrima returns ID number xx when this configuration is entered

Encoder/decoder algorithm: MPEGL3
 Encoder/decoder bit rate: 56 or 64 kb/s (use the highest available for your connection)
 Sample rate: 32 or 48 kHz
 Encoder algorithm mode: Mono
 Encoder/decoder line format: 1 line, line 1
 Decoder independent: YES

The CDQPrima returns ID number yy when this configuration is entered

Encoder/decoder algorithm: MPEGL3
 Encoder/decoder bit rate: 56 or 64 kb/s (use the highest available for your connection)
 Sample rate: 32 or 48 kHz
 Encoder algorithm mode: Mono
 Encoder/decoder line format: 1 line, line 2
 Decoder independent: YES

The CDQPrima returns ID number zz when this configuration is entered

2. Set up 4 Virtual Actions as follows:

CVA 0 CSD xx	Load Quick Configuration xx, 2-line, lines 1 & 2.
CVA 1 CSD yy; DAL MPEGL3	Load Quick Configuration yy, 1 line, line 1. Set decoder algorithm to Layer 3.
CVA 2 CSD zz; DAL MPEGL3	Load Quick Configuration zz, 1 line, line 2. Set decoder algorithm to Layer 3.
CVA 3 CTM 0 3	Start a 3 second timer for delayed start.

2. Set the following Event-to-Action Logic to determine which Virtual Action to execute:

CEA VA0 +(CD1 & CD2)	Execute VA0 when connected on 2 lines.
CEA VA1 +(CD1 & !CD2 & !TI0)	Execute VA1 when only line 2 is connected and the timer is not running.
CEA VA2 +(!CD1 & CD2 & !TI0)	Execute VA2 when only line 1 is connected and the timer is not running.
CEA VA3 +(CD1 & CD2 & FRAMED)	Execute VA3 when both lines connect and framed.

In the above Event-to-Action logic statements you can add the condition **&CI1** to allow you to turn the function on and off using the CUE 1 button on your **CDQPrima** 120, 220 and 230 models.

Here's how these 8 lines work:

- If 2 lines are connected, load the 2-line configuration.
- If only one line connects, or if any line drops, load the appropriate single line configuration..

- A timer is used to delay the start of the routine until the *CDQPrima* is framed.



Unlike Layer 2, which will ‘fall back’ instantly when a line drops, there will be several seconds of muted audio as Layer 3 falls back. This, however, is better than no audio at all. **This function will only work if programmed into BOTH codecs, and will not work with Telos Zephyr.**