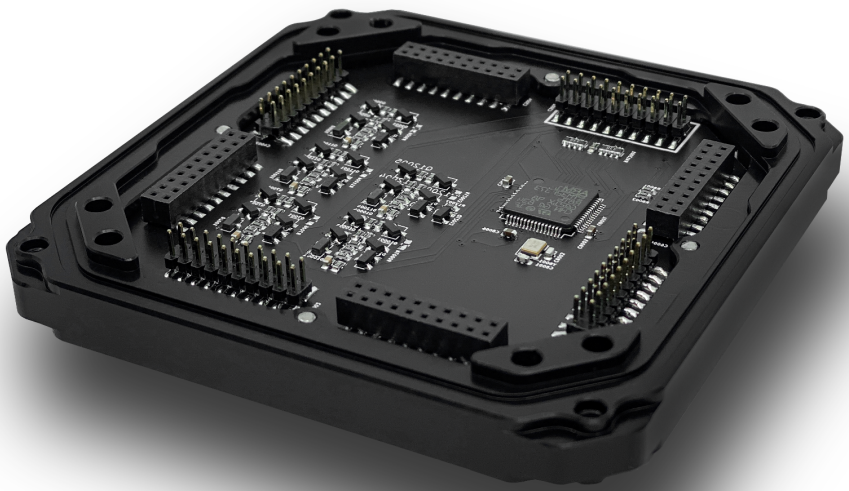
Input Peripheral Manual

NODEUDESIGN



https://www.nodeudesign.com
sales@nodeudesign.com

Table of Contents

1.1. Pinouts

Pin	Name	Description	
X1	Channel 1	Input Channel 1	Encoder 1 A
X2	Channel 2	Input Channel 2	Encoder 1 B
X3	Channel 3	Input Channel 3	Encoder 2 A
X4	Channel 4	Input Channel 4	Encoder 2 B
X5	Channel 4	Input Channel 5	-

1.1. Electrical Characteristics

Absolute Maximum Ratings	
	Max
Voltage at any Input terminal	36V
V(ESD) Electrostatic discharge	16000V
Temperature (Storage & Operating)	-40℃ - 80℃
Operating Temperature	-20℃ - 80℃

Recommended Ratings			
	Min	**Typical**	**Max**
Operating Voltage	7.5V	-	36V
Continuous Current per Channel	-	1.0A	2.0A[1]
PWM Frequency Input	500Hz	-	100KHz
Digital High Logic Level	2.5V	-	3.3V
Digital Low Logic Level	0V	-	2.5

[1] Depends on output connector interface.

2.1. Hardware

The Input Peripheral is designed to capture a wide range of signals, including: analog, digital, PWM frequency, pwm duty cycle, and encoder values. The input peripheral's abundance of five input modes are possible through multiplexing capabilities and use of dedicated hardware systems provided by the NUD architecture. Each input features ESD protection diodes for withstanding environments where static discharge may otherwise disable input circuitry. Furthermore, the Input Peripheral's voltage and current modes feature integrated hardware filters that limit the analog input frequency to 1500Hz.

When in current input mode, the Input Peripheral engages a 390Ω precision resistor for voltage conversion. Current mode and Voltage mode are similar in operation and can be utilized in identical fashion.

2.2. Software

Channel Modes:

The Input Peripheral features multiple modes for each channel that can be selected and changed on the fly for any application or as sensors are changed. These modes are digital, voltage/current, input counter, timed(frequency and duty cycle), and encoder.

Digital mode allows the Input Peripheral to detect simple digital signals such as local proximity sensors or limit switches. Digital modes allow simple high or low signals to be read. Values are sent to the Brain as "1" for a logic high, or "0" as a logic low.

The Input Peripheral offers two analog modes for each channel, voltage sensing and current sensing. These analog modes allow common 0-10v or 4-20mA sensor signals to be received. The analog signals are measured with a sigma-delta analog to digital converter. This offers high-performance and low-power measurement. A benefit of the value measurement being offloaded is that the peripheral features maximum and minimum value capture as well as out-of-bound trigger ranges. This increases value measurement speed and offloads input processing from the Brain logic. Finally, filters can be implemented as described in the Software Defined Filters[2] section.

The Input Peripheral also offers complex digital modes including frequency and duty cycle measurement. Both modes feature rising or falling edge detection for proper edge synchronization.

Encoder interfaces allow for motor control applications. Channels 1 & 2 or 3 & 4 can be configured in coupled mode for standard incremental encoder. The encoder interface allows tracking of absolute position changes since power-up or last reset.

[2] Coming Soon

Event Triggers:

Many systems utilize a read-modify-write style control loop, but the NUD architecture goes beyond this with its Rising and Falling Event Trigger system. The Event Trigger system, configurable in all available Input Modes, allows customizable callback functions to promptly execute upon measured input levels crossing configurable thresholds. Event Triggers allow system control at the edge of your NUD Stack, ensuring reactions to inputs as fast as possible.

Digital modes may be initialized to trigger on the rising or falling edge of the signal input. This may be useful for creating a software flag, or disabling an output drive when a limit switch has been triggered.

Analog and PWM modes may be configured to generate triggers when a measured frequency or duty cycle crosses configurable setpoints. This includes high, low, or high-low boundaries. Furthermore, configurable hysteresis settings are available to prevent inadvertent triggers.

Event Trigger Data Types:

TriggerConfig(select, level, hysteresis, mode)

TriggerConfig is the data type and parameter used in the TriggerConfig property. TriggerConfig packs all necessary trigger parameters into one modifiable data type. TriggerConfig's members may be accessed individually, or reinitialized to new values.

Parameter 1: select: FALLING : 0 RISING : 1	Parameter 2: level: DIGITAL : (0 - 1) VOLTAGE : (0 - 10000) (mV] CURRENT : (0 - 24) (mA] PWM : (0 - 100000) [Hz] DUTYCYCLE : (0 - 1000) [0.1%] ENCODER : (int32) COUNTER : (int32)	Parameter 3: hysteresis: DIGITAL : (0 - 1) VOLTAGE : (0 - 10000) (mV] CURRENT : (0 - 24) (mA] PWM : (0 - 100000) [Hz] DUTYCYCLE : (0 - 1000) [0.1%] ENCODER : (int32) COUNTER : (int32)	Parameter 4: mode: DISABLED : 0 PIN : 1 MESSAGE : 2

TriggerValue(channel, select, level)

TriggerValue is the data type returned by a Trigger or Trigger Clear event. TriggerValue and its members will be automatically passed into any function assigned to a trigger callback, thereby providing the system with information about the events channel, rising or falling edge, and the channel's input level.

Parameter 1: channel: 1 - 5	Parameter 2: select: FALLING : 0 RISING : 1	Parameter 3: level: DIGITAL : (0 - 1) VOLTAGE : (0 - 10000) (mV] CURRENT : (0 - 24) (mA] PWM : (0 - 100000) [Hz] DUTYCYCLE : (0 - 1000) [0.1%] ENCODER : (int32) COUNTER : (int32)

Event Trigger Callback Functions:

Event Trigger Callback Functions create the link between the event being triggered and the function which is called subsequently. When a function is defined as the callback to an event, the function will execute as soon as possible when the associated event is triggered.

ChannelTriggered(callbackFunction)

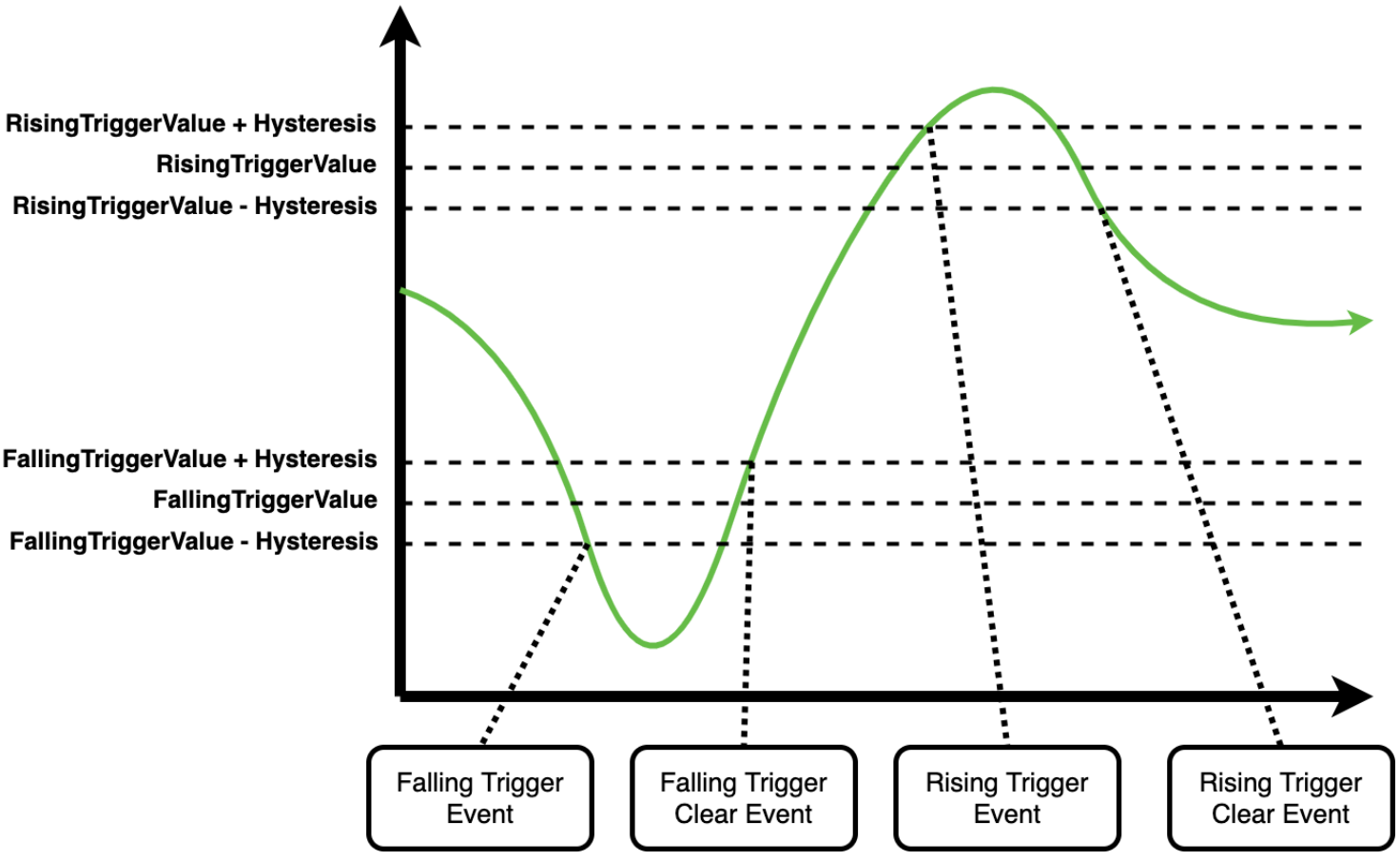
ChannelTriggered is the event called upon when the input signal level crosses the channel's Trigger Value + Hysteresis. When this event is triggered, the function assigned to it's callback will execute.

```
def triggerFunction( channel, select, value ):
   print("do something")

InputPeripheral.events.ChannelTriggered.setCallback( self.triggeredFunction )
```

ChannelCleared(callbackFunction)

ChannelCleared is the event called upon when the input signal level crosses the channel's Trigger Value - Hysteresis. When this event is triggered, the function assigned to it's callback will execute.

```
def triggerClearFunction( channel, select, value ):
   print("do something else")

InputPeripheral.events.ChannelTriggered.setCallback( self.triggerClearFunction )
```



3.1. ChannelMode(Fetch, Store)

Description:

ChannelMode configures selected channel's input mode.

`ChannelMode.fetch()`: Queues the selected channel input mode value to be transmitted to the brain. The Input Peripheral will then reply back to the Brain with the corresponding value. This function returns nothing.
`ChannelMode.value()`: Reads the most recently updated selected channel mode value directly from memory. This function must be called after a fetch(), and returns the selected channel input mode value.
`ChannelMode.store()`: Configures the selected channel input mode between Digital sensing, Voltage sensing, Current sensing, Timed, and Encoder modes. This function returns nothing.

Fetch:	**Value(Response):**	**Store:**	
Parameter 1: Channel: (0 - 4)	Response 1: Channel Input Mode: 0 : DIGITAL 1 : VOLTAGE 2 : CURRENT 3 : TIMED 4 : ENCODER	Parameter 1: Channel: (0 - 4)	Parameter 2: Channel Input Mode: 0 : DIGITAL 1 : VOLTAGE 2 : CURRENT 3 : TIMED 4 : ENCODER

Example:

```
InputPeripheral.properties.ChannelMode.store( 1, InputModes.ENCODER )
InputPeripheral.properties.ChannelMode.fetch( 1 )
inputMode = InputPeripheral.properties.ChannelMode.value( 1 )
print( inputMode )
```

```
Output: 4
```

The code first configures Channel 2's input mode to measure an encoder signal by calling `ChannelMode.store( 1, InputModes.ENCODER )`. The code requests the input peripheral to reply back with Channel 2's present input mode by calling `ChannelMode.fetch( 1 )`. Next, the value is saved to the variable "`inputMode`" by calling `ChannelMode.value( 1 )`. Finally, Channel 2's `ChannelMode` value is printed in the terminal.

3.2. CoupledMode(Fetch, Store)

Description:

`CoupledMode` enables channel pairs 1 & 2 or 3 & 4 to be coupled for encoder use.

`CoupledMode.fetch()`: Queues the selected pair coupled mode enable value to be transmitted to the brain. The Input Peripheral will then reply back to the Brain with the corresponding boolean value. This function returns nothing.
`CoupledMode.value()`: Reads the most recently updated selected pair Coupled Mode enable value directly from memory. This function must be called after a fetch(), and returns a boolean value.
`CoupledMode.store()`: Sets the selected channel pair's coupled mode enable value.

Fetch:	**Value(Response):**	**Store:**	
Parameter 1: Channel Pair: (0 - 1)	Response 1: Coupled Mode Enable: 0 : DISABLED 1 : ENABLED	Parameter 1: Channel Pair: (0 - 1)	Parameter 2: Coupled Mode Enable: 0 : DISABLE 1 : ENABLE

Example:

```
InputPeripheral.properties.ChannelMode.store( 0, InputModes.ENCODER )
InputPeripheral.properties.ChannelMode.store( 1, InputModes.ENCODER )
InputPeripheral.properties.CoupledMode.store( 0 , ENABLE )
InputPeripheral.properties.CoupledMode.fetch( 0 )
coupledMode1 = InputPeripheral.properties.CoupledMode.value( 0 )
print( coupledMode1 )
```

```
Output: 1
```

The code first configures Channels 1 and 2's channel modes as encoders by calling `ChannelMode.store( 0, InputModes.ENCODER )`, and `ChannelMode.store( 1, InputModes.ENCODER )`. The code then couples Channel 1 and 2 in an encoder configuration by calling `CoupledMode.store( 0, ENABLE )`. The code requests the input peripheral to reply back with Channel 1 and Channel 2's coupled mode enable value by calling `CoupledMode.fetch( 0 )`. Next, the value is saved to the variable "`coupledMode1`" by calling `CoupledMode.value( 0 )`. Finally, Channel 1 and Channel 2's coupled mode enable value is then printed in the terminal.

3.3. SingleValue(Fetch)

Description:

SingleValue returns the selected input channel digital state, voltage value, current value, PWM frequency, PWM Duty Cycle, or Encoder Value. The returned value type depends on the selected input channels Channel Mode.

SingleValue.fetch(): Queues the selected channel value to be transmitted to the brain. The Input Peripheral will then reply back to the Brain with the corresponding value. This function returns nothing.
SingleValue.value(): Reads the most recently updated selected channel input value. This function must be called after a fetch(), and returns the selected input channels measured value.

Fetch:	**Value(Response):**					
Parameter 1: Channel Pair: (0 - 4)	Digital	Voltage	Current	Frequency	Duty Cycle	Encoder
	(0, 1) [OFF/ON]	(0 - 36,000) [mV]	(0 - 24,000) [UA]	(1 - 100,000) [Hz]	(0 - 1000) [0.1%]	(-2,147,483,648 - 2,147,483,647) [Counts]

Example:

```
InputPeripheral.properties.ChannelMode.store( 2, InputModes.CURRENT )
InputPeripheral.properties.SingleValue.fetch( 2 )
singleVal = InputPeripheral.properties.SingleValue.value( 2 )
print( singleVal )
```

```
Output: 1550
```

The code first configures Channel 3's measurement mode to current sensing by calling ChannelMode.store(2, InputModes.CURRENT). The code requests the Input Peripheral to reply back the current through the load by calling SingleValue.fetch(2). Next, the value is saved to the variable "singleVal" by calling SingleValue.value(2). Finally, Channel 3's measured current value is printed in the terminal.

3.4. SingleMaxValue(Fetch, Reset)

Description:

SingleMaxValue is the selected channel maximum value since the channel was configured or reset. The values type depends on the selected input channels Channel Mode.

SingleMaxValue.fetch(): Queues the selected channel maximum value to be transmitted to the brain. The Input Peripheral will then reply back to the Brain with the corresponding value. This function returns nothing.
SingleMaxValue.value(): Reads the most recently updated selected channel maximum value. This function must be called after a fetch(), and returns the selected input channels measured maximum value since the last Reset.
SingleMaxValue.reset(): Resets the selected channel maximum value to the channel current input value. The Input Peripheral will begin to monitor and update the channel maximum input value immediately after the reset command has been issued.

Fetch:	**Value(Response):**					**Reset:**
Parameter 1: Channel: (0 - 4)	Digital	Voltage	Current	Frequency	Duty Cycle	Parameter 1: Channel: (0 - 4)
	(0, 1) [OFF/ON]	(0 - 36,000) [mV]	(0 - 24,000) [UA]	(1 - 100,000) [Hz]	(0 - 1000) [0.1%]	

Example:

```
InputPeripheral.properties.ChannelMode.store( 1, InputModes.VOLTAGE )
InputPeripheral.properties.SingleMaxValue.fetch( 1 )
singleMaxVal = InputPeripheral.properties.SingleMaxValue.value( 1 )
print( singleMaxVal )
InputPeripheral.properties.SingleMaxValue.reset( 1 )
```

```
Output: 5000
```

The code first configures Channel 2's input measurement mode to voltage sensing by calling ChannelMode.store(1, InputModes.VOLTAGE). The code then requests the Input Peripheral to reply back this value by calling SingleMaxValue.fetch(1). Next, the value is saved to the variable "singleMaxVal" by calling SingleMaxValue.value(1). The value is then printed in the terminal. Channel 2's maximum voltage value is then reset by calling SingleMaxValue.reset(1).

3.5. SingleMinValue(Fetch, Reset)

Description:

SingleMinValue is the selected channel minimum value since the channel was configured, or reset. The values type depends on the selected input channels Channel Mode.

SingleMinValue.fetch(): Queues the present selected channel minimum value to be transmitted to the brain. The Input Peripheral will then reply back to the Brain with the corresponding value. This function returns nothing.
SingleMinValue.value(): Reads the most recently updated selected channel minimum value. This function must be called after a fetch(), and returns the selected input channels measured minimum value since the last Reset.
SingleMinValue.reset(): Resets the selected channel minimum value to the channel current value. The Input Peripheral will begin to monitor and update the channel minimum input value immediately after the reset command has been issued.

Fetch:	**Value(Response):**					
Parameter 1: Channel Pair: (0 - 4)	Digital	Voltage	Current	Frequency	Duty Cycle	Encoder
	(0, 1) [OFF/ON]	(0 - 36,000) [mV]	(0 - 24,000) [UA]	(1 - 100,000) [Hz]	(0 - 1000) [0.1%]	(-2,147,483,648 - 2,147,483,647) [Counts]

Reset:
Parameter 1: Channel: (0 - 4)

Example:

```
InputPeripheral.properties.ChannelMode.store( 0, InputModes.FREQUENCY )
InputPeripheral.properties.SingleMinValue.fetch( 0 )
singleMinVal = InputPeripheral.properties.SingleMinValue.value( 0 )
print( singleMinVal )
InputPeripheral.properties.SingleMinValue.fetch( 0 )
```

```
Output: 20000
```

The code first configures Channel 1's input measurement mode to frequency sensing by calling ChannelMode.store(0, InputModes.FREQUENCY). The code then requests the Input Peripheral to reply back with Channel 1's minimum frequency by calling SingleMinValue.fetch(0). Next, the value is saved to the variable "singleMinVal" by calling SingleMinValue.value(0). The value is then printed in the terminal. Channel 1's minimum frequency is reset by calling SingleMinValue.reset(0).

3.6. AllValues(Fetch)

Description:

`AllValues` is the array of all Input Peripheral channel values since the channels were configured or reset. Each returned value type depends on each channel Channel Mode.

`AllValues.fetch()`: Queues all Input Peripheral channel values to be transmitted to the brain. The Input Peripheral will then reply back to the Brain with the corresponding values. This function returns nothing.
`AllValues.value()`: Reads the array of the most recently updated Input Peripherals input channel values. This function must be called after a fetch(), and returns an array of all Input Peripherals channel values.

Fetch:	**Value(Response):**					
Parameter 1: Channel Pair: (0 - 4)	Digital	Voltage	Current	Frequency	Duty Cycle	Encoder
	(0, 1) [OFF/ON]	(0 - 36,000) [mV]	(0 - 24,000) [UA]	(1 - 100,000) [Hz]	(0 - 1000) [0.1%]	(-2,147,483,648 - 2,147,483,647) [Counts]

Example:

```
InputPeripheral.properties.AllValues.fetch()
allVals = InputPeripheral.properties.AllValues.value()
print( allVals )
```

```
Output: 1, 4800, 125, 1200
```

The above code assumes that the Input Peripheral's channels have been configured as: Ch1: Digital, Ch2: Voltage, Ch3+4(Coupled): Encoder, Ch5: Current. The code then requests the Input Peripheral to reply an array of all Input Peripheral channels values by calling `AllValues.fetch()`. Next, the array is saved to the variable "`allVals`" by calling `AllValues.value()`. Finally, the array is then printed in the terminal, where 1 corresponds to a digital high signal, 4800 corresponds to 4.8V, 125 corresponds to an encoder position value, and 1200 corresponds to 1.2A.

3.7. AllMaxValues(Fetch, Reset)

Description:

`AllMaxValues` is the array of all Input Peripherals channels maximum values since the channels were configured or reset. Each returned value type depends on each input channel's Channel Mode.

`AllMaxValues.fetch()`: Queues all input channel's maximum values to be transmitted to the brain. The Input Peripheral will then reply back to the Brain with the corresponding values. This function returns nothing.
`AllMaxValues.value()`: Reads the array of the most recently updated Input Peripherals input channels maximum values. This function must be called after a fetch(), and returns an array of all Input Peripherals channel values.
`AllMaxValues.reset()`: Resets all Input Peripheral channel maximum values to each respective channel's present input value. The Input Peripheral will begin to monitor and update each channel's maximum input value immediately after the reset command has been issued.

Fetch:	**Value(Response):**					
Parameter 1: Channel Pair: (0 - 4)	Digital	Voltage	Current	Frequency	Duty Cycle	Encoder
	(0, 1) [OFF/ON]	(0 - 36,000) [mV]	(0 - 24,000) [uA]	(1 - 100,000) [Hz]	(0 - 1000) [0.1%]	(-2,147,483,648 - 2,147,483,647) [Counts]

Reset:
Parameter 1: Channel: (0 - 4)

Example:

```
InputPeripheral.properties.AllMaxValues.fetch()
allMaxVals = InputPeripheral.properties.AllMaxValues.value()
print( allMaxVals )
InputPeripheral.properties.AllMaxValues.reset()
```

```
Output: 250, 25000, 1, 1800
```

The above code assumes that the Input Peripheral's channels have been configured as: Ch1+2(Coupled): Encoder, Ch3: Voltage, Ch4: Digital, Ch5: Current. The code requests the Input Peripheral to reply an array of all Input Peripherals channel maximum values by calling `AllMaxValues.fetch()`. Next, the array is saved to the variable "`allMaxVals`" by calling `AllMaxValues.value()`. The array is then printed in the terminal, where 250 corresponds to an encoder position maximum value, 25000 corresponds to 25V, 1 corresponds to a digital high signal, and 1800 corresponds to 1.8A. Finally, the Input Peripherals maximum input channel values are reset by calling `AllMaxValues.reset()`.

3.8. AllMinValues(Fetch, Reset)

Description:

`AllMinValues` is the array of all Input Peripherals channels minimum values since the channels were configured or reset. Each returned value type depends on each input channel's Channel Mode.

`AllMinValues.fetch()`: Queues all input channel's minimum values to be transmitted to the brain. The Input Peripheral will then reply back to the Brain with the corresponding values. This function returns nothing.
`AllMinValues.value()`: Reads the array of the most recently updated Input Peripherals input channels minimum values. This function must be called after a fetch(), and returns an array of all Input Peripherals channel values.
`AllMinValues.reset()`: Resets all Input Peripheral channel minimum values to each respective channel's present input value. The Input Peripheral will begin to monitor and update each channel's minimum input value immediately after the reset command has been issued.

Fetch:	**Value(Response):**					
Parameter 1: Channel Pair: (0 - 4)	Digital	Voltage	Current	Frequency	Duty Cycle	Encoder
	(0, 1) [OFF/ON]	(0 - 36,000) [mV]	(0 - 24,000) [UA]	(1 - 100,000) [Hz]	(0 - 1000) [0.1%]	(-2,147,483,648 - 2,147,483,647) [Counts]

Reset:
Parameter 1: Channel: (0 - 4)

Example:

```
InputPeripheral.properties.AllMinValues.fetch()
allMinVals = InputPeripheral.properties.AllMinValues.value()
print( allMinVals )
InputPeripheral.properties.AllMinValues.reset()
```

```
Output: 250, 300, 200
```

The above code assumes that the Input Peripheral's channels have been configured as: Ch1+2(Coupled): Encoder, Ch3+4(Coupled): Encoder, Ch5: Voltage. The code requests the Input Peripheral to reply an array of all Input Peripherals channel minimum values by calling `AllMinValues.fetch()`. Next, the array is saved to the variable "`allMinVals`" by calling `AllMinValues.value()`. The array is then printed in the terminal, where 250 corresponds to channel 1 and 2's encoder position minimum value, 300 corresponds to channel 3 and 4's encoder position minimum value, and 200 corresponds to 200mV. Finally, the Input Peripherals minimum input channel values are reset by calling `AllMinValues.reset()`.

3.9. TriggerConfig(Fetch, Store)

Description:

`TriggerConfig` configures the Input Peripherals selected channels trigger values. TriggerConfig utilizes the data type `TriggerConfig`. When an event trigger occurs, the corresponding callback function will automatically pass in the `TriggerValue()` data type containing the channel number, trigger select, and channel value level.

`TriggerConfig.fetch()`: Queues the selected input channel's event trigger values to be transmitted to the brain. The Input Peripheral will then reply back to the Brain with the corresponding values. This function returns nothing.
`TriggerConfig.value()`: Reads the most recently updated Input Peripheral events trigger values. This function must be called after a fetch(), and returns a `TriggerConfig` data type containing the Input Peripherals selected channels event trigger values.

Parameter 1: select: FALLING : 0 RISING : 1	Parameter 2: level: DIGITAL : (0 - 1) VOLTAGE : (0 - 10000) (mV] CURRENT : (0 - 24) (mA] PWM : (0 - 100000) [Hz] DUTYCYCLE : (0 - 1000) [0.1%] ENCODER : (int32) COUNTER : (int32)	Parameter 3: hysteresis: DIGITAL : (0 - 1) VOLTAGE : (0 - 10000) (mV] CURRENT : (0 - 24) (mA] PWM : (0 - 100000) [Hz] DUTYCYCLE : (0 - 1000) [0.1%] ENCODER : (int32) COUNTER : (int32)	Parameter 4: mode: DISABLED : 0 PIN : 1 MESSAGE : 2

Example:

```
InputPeripheral.properties.ChannelMode.store( 1, InputModes.DIGITAL )
triggerConfiguration = TriggerConfig( TriggerSelect.FALLING, 1, 0, TriggerMode.MESSAGE )
InputPeripheral.properties.TriggerConfig.store( 1, triggerConfiguration )
InputPeripheral.properties.TriggerConfig.fetch( 1 )
newTriggerConfiguration = InputPeripheral.properties.TriggerConfig.value( 1 )
print( newTriggerConfiguration )
```

```
Output: 1, 1, 0, 2
```

The code first configures the Input Peripheral's channel mode to Digital by calling `ChannelMode.store( InputModes.DIGITAL )`. Next, the variable containing the TriggerConfig data type is initialized. The trigger is configured as a Rising Edge, High Logic Level, 0 hysteresis, Message type trigger. Then, channel 1's rising trigger is set by calling `TriggerConfig.store( 1, triggerConfiguration )`. Next, the code requests the Input Peripheral to reply back with channel 1's trigger value by calling `TriggerConfig.fetch( 1 )`. Channel 1's trigger values are then saved to the variable "`newTriggerConfiguration`" by calling `TriggerConfig.value( 1 )`. Finally, Channel 1's trigger values are printed in the terminal.

1.2. Importing Input Peripheral

```
import NUD.System as s

import NUD.Peripherals.Input.Input as i
import NUD.Peripherals.Input.InputValues as icv
import NUD.Peripherals.Input.InputTypes as it

periph = i.Input()

system = s.System()
# Attach the CAN peripheral to Side 1 Slot A
system.attachPeripheral( periph, s.SideSlot.SIDE_1_SLOT_A )
```

1.3. Taking a Voltage Measurement

```
InputPeripheral.properties.ChannelMode.store( 1, it.InputModes.VOLTAGE )
InputPeripheral.properties.Voltage.fetch( 1 )
channel1Voltage = InputPeripheral.properties.Voltage.value( 1 )
print( channel1Voltage )
```

```
Output: 4998
```

The code first configures the Input Peripheral's channel mode to voltage mode by calling `ChannelMode.store( 1, InputModes.VOLTAGE )`. Next, the code requests the Input Peripheral to reply back with channel 1's voltage measurement value by calling `Voltage.fetch( 1 )`. The Input Peripheral's channel 1 voltage measurement value is then saved to the variable "`channel1Voltage`" by calling `Voltage.value( 1 )`. Finally, the Input Peripheral's channel 1 voltage measurement value is printed in the terminal.

1.4. Configuring and Reading Event Triggers

```
# Initialize Trigger Variables
triggerRisingConfiguration = it.TriggerConfig( it.TriggerSelect.RISING, 5000, 250,
it.TriggerMode.MESSAGE )
triggerFallingConfiguration = it.TriggerConfig( it.TriggerSelect.Falling, 4200, 250,
it.TriggerMode.MESSAGE )

# Trigger Callback Functions
def InputTrigger( self, channel, select, value ):
   print("Event Trigger:", channel, select, value)
def InputTriggerClear( self, channel, select, value):
   print("Event Clear:", channel, select, value)

# Configure Triggers
InputPeripheral.properties.ChannelMode.store( 2, it.InputModes.VOLTAGE )
InputPeripheral.properties.TriggerConfig.store( 2, triggerConfiguration )

# Configure Trigger Callbacks
InputPeripheral.events.ChannelTriggered.setCallback( self.InputTriggered )
InputPeripheral.events.TriggerCleared.setCallback( self.InputTriggerCleared )
```

```
Event Trigger: 2, 0, 3950
Event Clear: 2, 0, 4450
Event Trigger: 2, 1, 5250
Event Clear: 2, 1, 4750
```