PYRAMID
www.pyramid.tech

IGX

IGX - User Manual

Document ID: 4113694732

Version: v10

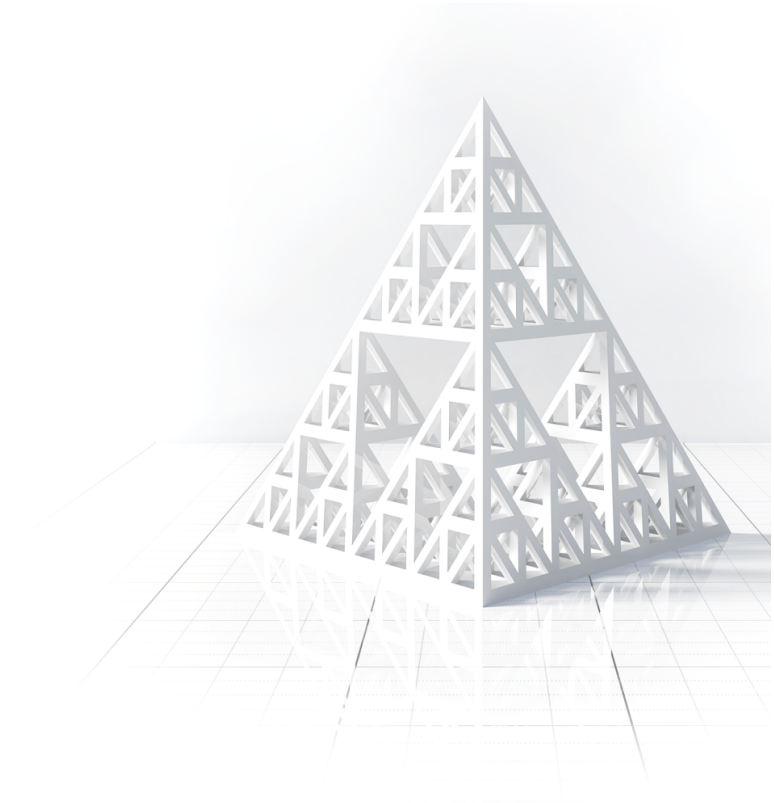


Table of Contents

1 Introduction

Document ID: 2649784673

Author	Matthew Nichols
Owner	Project Lead
Purpose	Provide clear, concise, and comprehensive information for the safe and effective use, maintenance, and troubleshooting of IGX based products.
Scope	The use of an IGX product by an end user.
Intended Audience	IGX end-users
Process	Standard Manual Creation Process
Training	NOT APPLICABLE

1.1 IGX Control System Framework

IGX is Pyramid's latest software control system framework. It's designed from the ground up for high-reliability applications including medical and light industrial settings. The framework is monolithic and modular, meaning that it comes with many features by default and that the features are all encapsulated such that they can be reused and extended easily. Some key features are:

- Battle proven real-time QNX operating system with highly reliable microkernel.
- Custom web server with low latency and high bandwidth.
- Harmonized IO data available over multiple Ethernet protocols.
- Powerful data acquisition and processing tools.
- Built-in and user configurable safety interlocking system, allows for automatic safety actions.
- User definable expression language for simple scripting.

2 Document Control

2.1 Version History

Version	Description	Saved by	Saved on
v10	Added PWM Module information	Matthew Nichols	Jul 7, 2026 9:11 PM
v9	Added network and browser troubleshooting information	Matthew Nichols	Jun 30, 2026 5:09 PM
v8	Added PID expression documentation	Matthew Nichols	May 12, 2026 10:16 PM
v7	Added more information for dose controller states and improved the state machine diagram.	Harvey Jules Nett	May 7, 2025 2:21 PM
v6	Add new network guidelines and more information about policy hashes	Matthew Nichols	Mar 14, 2025 6:23 PM
v5	Added detail to network and admin features. Added new sections for dose controller and notification systems.	Matthew Nichols	Mar 13, 2025 10:13 PM
v4	Added chapters for NTP servers and permission policies.	Matthew Nichols	Jun 3, 2024 9:32 PM
v3	Fixed conditional expression documentation.	Matthew Nichols	Apr 19, 2024 9:31 PM
v2	SD card imaging instructions and expression language documentation.	Harvey Jules Nett	Apr 1, 2024 3:51 PM
v1	Initial release	Harvey Jules Nett	Dec 1, 2023 5:51 PM

2.2 Approvals

This document has been reviewed and approved as follows.

Document Control Not Reviewed

Current document version: v.1

No reviewers assigned.

2.2.1 Signatures

Tuesday, Jul 7, 2026, 09:12 PM UTC, (v. 1)
Matthew Nichols signed ; meaning: **Review**

3 IGX Networking Guide

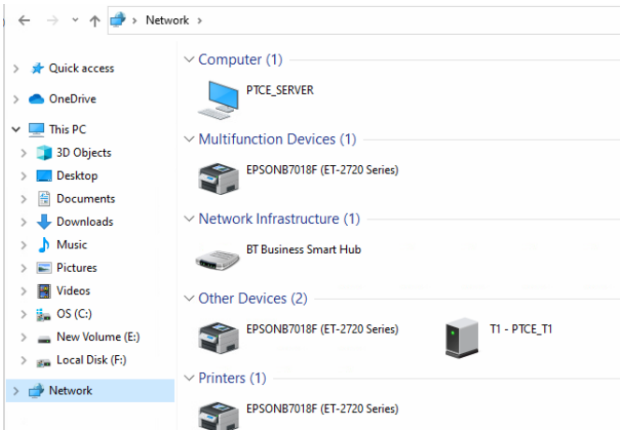
3.1 Network Introduction

The IGX framework uses Ethernet as its primary communication medium. It supports GUIs, control systems, databases, scripting, and more. IGX primarily communicates via HTTP, which serves the embedded web GUI.

To connect with an IGX device, you need a basic understanding of Ethernet network concepts. This guide walks you through connecting to IGX devices and managing larger networks with multiple Pyramid and third-party products.

3.1.1 Quick Start

If you know network concepts, you can start using IGX devices immediately. Our products have DHCP enabled and will try to obtain an IP address automatically. If no DHCP response occurs, the device defaults to the static IP address `192.168.100.20` with a netmask of `255.255.255.0`. Once the device has an IP address, you can discover it on your network using UPnP. Windows 10 and 11 include a discovery tool in File Explorer.



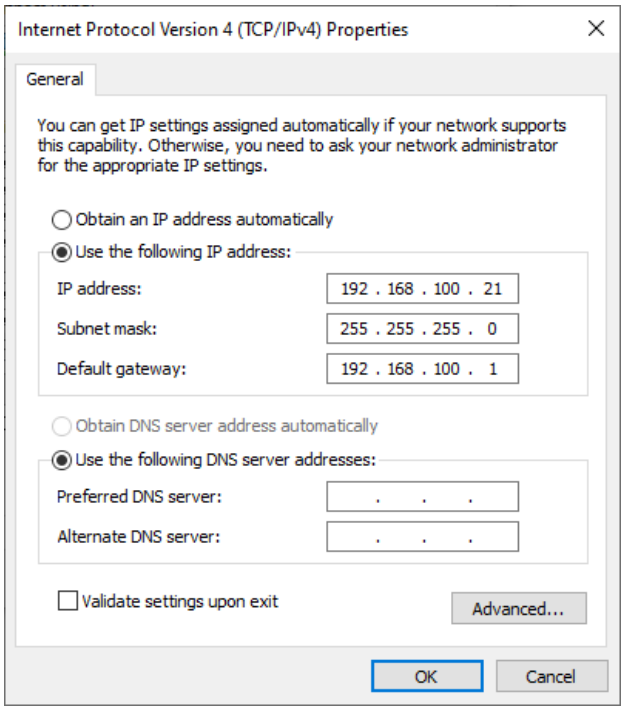
1 : Pyramid devices are found under "Other Devices". The device name is customizable.

Ensure network discovery is on. Press refresh if the device was added recently.

When the device icon appears on the Network page, double-click it to open its URL in your web browser.

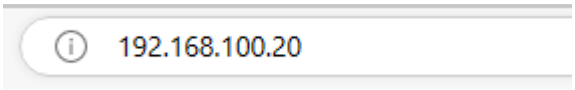
Network settings can be changed in the web browser after the page loads.

If the previous steps don't work, or you want to explicitly connect directly to the IGX device, you'll need to configure your computer's IP address settings to cooperate with the device's settings.



2 : Windows Ethernet Adaptor settings, note that the computer's assigned IP address is different from the device's IP address.

Refer to your operating system's documentation for specific instructions on how to do this. Once you've set your computer's Ethernet adapter to the right subnet, type the device's IP address directly into the address bar of a new tab on your web browser to open the web interface. You can then log in and start controlling the device's settings and collecting data.



3 : Browsers let you directly connect to the device's web GUI.

Depending on your device and network, additional configuration may be needed. Refer to your device documentation for details.

3.2 Web Browser Recommendations and Troubleshooting

Use a modern web browser when connecting to IGX device web interfaces. Pyramid supports the latest versions of Google Chrome, Microsoft Edge, and Mozilla Firefox. Older browser versions may work but are not guaranteed.

Google Chrome

Recommended modern browser for accessing IGX device web interfaces.

Google Chrome[1]

Microsoft Edge

Supported modern browser included with current Windows systems.

Microsoft Edge[2]

Mozilla Firefox

Supported modern browser for Windows, macOS, and Linux.

Mozilla Firefox[3]

3.2.1 Hard refresh the browser cache

If the device web interface does not load correctly, shows stale information, or behaves unexpectedly, first perform a hard refresh. A hard refresh forces the browser to reload the page files from the IGX device instead of using cached files stored on your computer.

- **Chrome, Edge, or Firefox on Windows/Linux:** press `Ctrl` + `F5`. If that does not work, press `Ctrl` + `Shift` + `R`.
- **Chrome, Edge, or Firefox on macOS:** press `Command` + `Shift` + `R`.
- **Alternative method:** hold `Shift` and click the browser Reload button.

After the hard refresh completes, wait for the page to finish loading and try the same action again. If the problem continues, open the browser developer tools and check the console for errors.

1. https://www.google.com/chrome/
2. https://www.microsoft.com/edge
3. https://www.mozilla.org/firefox/new/

3.2.2 Try a different browser or disable extensions

If a hard refresh does not resolve the issue, try opening the IGX device web interface in another supported browser, such as Chrome, Edge, or Firefox. This helps determine whether the issue is specific to one browser.

Browser extensions can also interfere with embedded web interfaces. Ad blockers, privacy extensions, script blockers, security plug-ins, or extensions that force HTTPS may block scripts or network requests needed by the IGX GUI. As a quick test, open the device in a private or incognito window, or temporarily disable extensions and reload the page.

If the browser shows an HTTPS or certificate warning, confirm you are using the correct device address. IGX interfaces are typically served over HTTP, and browser settings or extensions that force HTTPS can prevent the page from loading.

3.2.3 Check the browser debug console

The debug console records browser-side errors that can help identify whether the issue is caused by the web interface. Open the console while viewing the IGX device web page, then repeat the action that caused the problem.

- **Chrome or Edge:** press `F12`, or press `Ctrl` + `Shift` + `I` on Windows/Linux, then select the **Console** tab.
- **Firefox:** press `F12`, or press `Ctrl` + `Shift` + `K` on Windows/Linux, to open the Web Console.
- **macOS browsers:** press `Command` + `Option` + `I`, then select the **Console** tab. In Firefox, `Command` + `Option` + `K` opens the Web Console directly.

Look for red errors, failed network requests, or repeated warnings that appear when the problem occurs. When contacting Pyramid, include the error text or a screenshot plus the browser name/version, device IP address or hostname, and the steps that caused the issue.

3.2.4 Check network connectivity

If the browser cannot reach the IGX device at all, verify the network connection before assuming the web interface is at fault.

- **Ping the device:** From a command prompt or terminal, run `ping <device IP address>`, such as `ping 192.168.100.20`. A successful reply confirms that the computer can reach the device at the IP network level.
- **Check the Ethernet cable and link lights:** Confirm that the Ethernet cable is fully seated and that the link/activity LEDs are lit on the computer, network switch, and IGX device where applicable. Try another cable or switch port if the link light is off.
- **Verify the IP address and subnet:** Confirm that the computer and IGX device are on the same subnet for direct connections. For example, if the IGX device is at `192.168.100.20` with netmask `255.255.255.0`, the computer should use a different address in the same range, such as `192.168.100.21`.
- **Disable VPN or proxy connections:** VPN clients, proxy settings, and some corporate security tools can redirect or block traffic to local network devices. Temporarily disconnect from the VPN or bypass the proxy for local addresses if permitted by your organization.
- **Check for IP address conflicts:** If another device is using the same IP address as the IGX device or your computer, communication may be intermittent or fail completely. Windows may display a duplicate IP address warning, but conflicts can also appear as inconsistent ping results or a web page that loads only sometimes.

3.2.5 Report console errors to Pyramid

If the console shows errors while using the IGX device web interface, treat that as a software issue and report it to Pyramid Technical Consultants. Console errors are especially useful because they usually point directly to the part of the web interface that failed.

When reporting the issue, include as much of the following information as possible:

- The IGX device model and firmware/software version, if known.
- The browser name and version, such as the latest version of Chrome, Edge, or Firefox.
- The operating system you are using, such as Windows 10, Windows 11, macOS, or Linux.
- The exact steps required to reproduce the problem.
- A screenshot or copied text of any red console errors or failed network requests.

With the error details and reproduction steps, Pyramid can usually identify and fix these issues quickly.

3.2.6 Power cycle the IGX device

If the network connection is present but the embedded web interface remains unresponsive, power cycle the IGX device. Turn the device off, wait a few seconds, and then turn it back on. Allow the device enough time to finish booting before refreshing the browser or reconnecting.

Power cycling can recover the embedded web server if it has become temporarily unresponsive. Do not remove power while a firmware update or configuration write is in progress unless instructed by Pyramid support.

3.2.7 Restart the browser or Operating system

In some extreme cases, unexpected or odd behavior may be caused by a web browser bug or an operating system issue rather than the IGX device itself. If a hard refresh and console check do not resolve the problem, completely close the web browser, including all open tabs and browser windows, and then restart it before trying again.

If the behavior continues after restarting the browser, reboot the operating system. Restarting the browser or rebooting the computer is worth trying when the web interface behaves unexpectedly, especially if the issue appears after an operating system or browser update.

3.3 IP Networking Basics

Internet Protocol (IP) networks form the basis of modern digital communication, enabling devices to exchange data over a shared network. They connect devices like computers, smartphones, printers, and IGX devices, allowing resource and information sharing.

Here are the basics of IP networks users need to know:

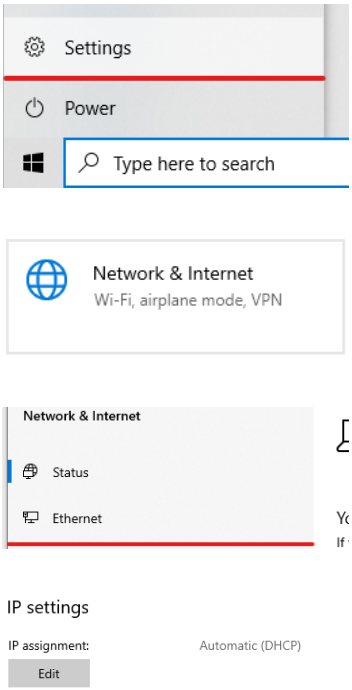
1. **IP Address**: Each device on an IP network has a unique identifier called an IP address, used to identify and locate it.
   IPv4 addresses consist of four sets of numbers separated by periods (e.g., 192.168.100.20). IPv6 addresses use eight sets of four hexadecimal characters separated by colons (e.g., 2001:0db8:85a3:0000:0000:8a2e:0370:7334).
2. **Subnet Mask**: A subnet mask defines the range of IP addresses in a network. It helps devices determine if they are on the same network. Subnet masks use the same format as IP addresses. For two computers to communicate without advanced routing, they must share the same subnet mask.
3. **Routers**: Routers connect multiple networks and direct data packets between them. They use IP addresses and subnet masks to find the best path. Home routers often serve as Wi-Fi access points, enabling wireless connections.
4. **DHCP**: Dynamic Host Configuration Protocol (DHCP) automatically assigns IP addresses and network settings to devices joining the network, simplifying connections and ensuring correct settings.
5. **DNS**: Domain Name System (DNS) translates human-readable domain names (e.g., www.example.com[4]) into IP addresses, making it easier to access websites without remembering numerical addresses.

4. http://www.example.com/

3.4 Setting your Windows 10/11 IP Address

This guide covers configuring Ethernet IP settings on Windows 10 and Windows 11 for both Dynamic Host Configuration Protocol (DHCP) and manual static IP configurations.

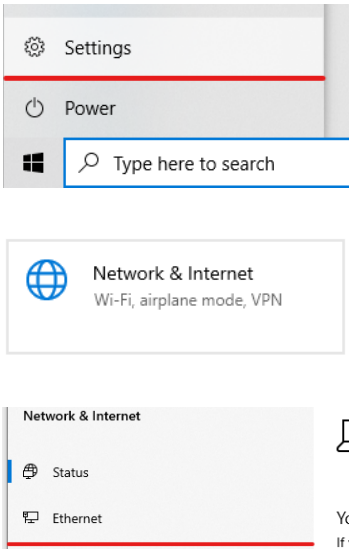
3.4.1 Using DHCP (Automatic IP Configuration)



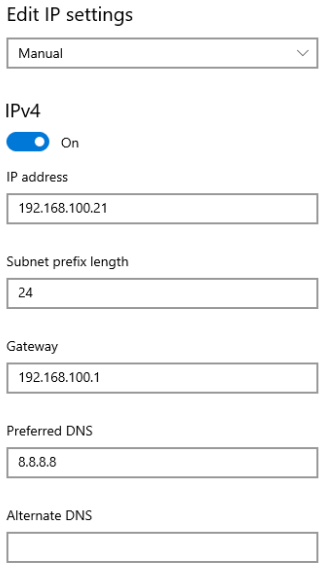
1. Open Settings: **Press Windows+I or click Start > Settings.**
2. Go to Network & Internet.
3. Select your adapter:
   - Windows 10: Ethernet (left pane) > click your Ethernet connection.
   - Windows 11: Network & Internet > Advanced network settings > Network adapters > select your Ethernet adapter.
4. Find IP settings and choose Edit:
   - Windows 10: Scroll to IP settings > Edit.
   - Windows 11: Adapter settings page > IP assignment > Edit.
5. In Edit IP settings, select Automatic (DHCP) and Save.

Windows 10 will now automatically obtain an IP address, subnet mask, default gateway, and DNS server addresses from your network's DHCP server. This is frequently a centralized router managed by an IT department or office administrator.

3.4.2 Using Manual Static IP Configuration



1. Open Settings: **Press Windows+I or click Start > Settings.**
2. Go to Network & Internet.
3. Select your adapter:
   - Windows 10: Ethernet (left pane) > click your Ethernet connection.
   - Windows 11: Network & Internet > Advanced network settings > Network adapters > select your Ethernet adapter.
4. Open IP assignment Edit:
   - Windows 10: Scroll to IP settings > Edit.
   - Windows 11: On the adapter page, under IP assignment, choose Edit.
5. In Edit IP settings, set to Manual.
6. Enable IPv4 (toggle On).
7. Enter the following values:
   - IP address: 192.168.100.21 (for direct connection to an IGX at 192.168.100.20).
   - Subnet mask/Prefix length: 255.255.255.0 or prefix length 24 (/24).



4 : New Windows 10 interface

    - Gateway (optional): Only if internet access or routing is needed.
8. Note: Ensure the IP you assign is unique on the network.
9. DNS servers (if needed for internet): Preferred 8.8.8.8, Alternate 8.8.4.4, or use your IT-provided DNS.
10. Click Save to apply.

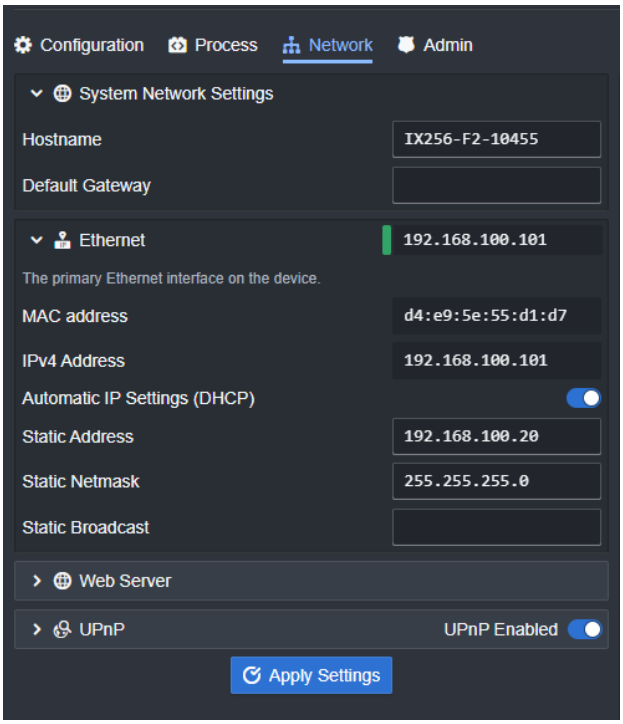
Windows will now use the static IP address, subnet mask, default gateway, and DNS server addresses you entered for your Ethernet connection.

Remember to verify your settings with your network administrator or ISP to ensure proper network connectivity. If you encounter any issues or need to revert to DHCP, follow the steps for "Using DHCP (Automatic IP Configuration)" to switch back.

3.5 IGX Network Settings

When you first connect to your IGX device, configure various network settings. These control the device's network identity, IP configuration, and routing.

- **Hostname**: The device's name on the local network for identification.
- **Default Gateway**: The router IP address that connects the device beyond the local network.
- **MAC Address**: The unique identifier of the device's network hardware.
- **IPv4 Address**: The device's assigned IP address on the network.
- **Automatic IP Settings (DHCP)**: Enable or disable DHCP.
- **Static Address**: Assign a fixed IP address when DHCP is off.
- **Static Netmask**: Set the subnet mask for the static address.
- **Static Broadcast**: Specify the broadcast address for the static IP.



5 : IGX Network Settings GUI

3.5.1 Applying Network Settings

Press the Apply Settings  button to command the IGX device to accept and apply new network settings.

Unplugging and reinserting the Ethernet cable prompts the IGX device to refresh its network settings, re-establishing its connection and applying configuration changes.

3.5.2 Hostname

Hostnames are human-readable names assigned to devices on a network, replacing numerical IP addresses. This helps people identify and connect to devices without memorizing IPs. Hostnames are essential in network infrastructure across industries, providing a consistent label for each device even if its IP changes (e.g., with DHCP).

Hostname requirements and best practices

- **Uniqueness**: Each device must have a unique hostname to avoid conflicts and ensure communication.
- **Length and characters**: Hostnames can be up to 253 characters, typically limited to 63 per label. They may contain letters (a-z), digits (0-9), and hyphens (-) but must not start or end with a hyphen.
- **DNS compliance**: For domain name systems (DNS), hostnames should follow the **RFC 1035** standard and be case-insensitive.

3.5.3 Default Gateway

The default gateway enables the IGX to communicate with devices outside its local network. In most setups, the default gateway is the IP address of a router. Devices on the same subnet communicate directly; traffic destined for a different subnet is forwarded through the gateway.

When using DHCP, the default gateway is configured automatically. You can override this via the IGX IO. Leaving the field blank lets the system use the DHCP-assigned gateway.

3.5.4 MAC Address

The MAC address (Media Access Control address) is a unique hardware identifier assigned to the IGX device network interface at the factory. It operates at the data link layer (Layer 2) and is used for communication within the local network. IP addresses operate at the network layer (Layer 3) and are used for routing data between networks.

Security Considerations

While the MAC address is a reliable way to identify a device on a local network, it should not be used for security purposes. This is because **MAC addresses can be easily spoofed** (faked) by attackers. For example, an attacker can change the MAC address of their device to mimic another device on the network, potentially allowing them to intercept or disrupt communications.

In security-sensitive applications, relying solely on the MAC address for device identification is not sufficient. Instead, stronger authentication mechanisms should be implemented, such as encryption, certificates, and secure network protocols.

3.5.5 IPv4 Address

The IPv4 address is a unique numerical identifier assigned to your IGX device on an IPv4 network. This address is used to identify and communicate with the device over the network. Just like how a home address allows postal mail to reach your house, an IPv4 address enables data to be routed to the correct device in a network.

IPv4 Address Basics

An **IPv4 address** is composed of four octets, each ranging from 0 to 255, separated by dots. For example, `192.168.100.20` is a typical IPv4 address. Each of these octets represents an 8-bit binary number, giving the address a total of 32 bits.

The address is divided into two parts:

- **Network portion**: Identifies the specific network the device belongs to. (`192.168.100`)
- **Host portion**: Identifies the specific device within that network. (`.20`)

An IPv4 address enables communication between devices on the same network or across different networks. When your IGX device wants to communicate with another device, the IPv4 address is used to direct the data to the correct destination.

Considerations for IPv4 Address Assignment

- **IP Conflicts:** If two devices on the network are accidentally assigned the same IPv4 address, it will lead to IP conflicts, causing communication issues. This can happen if you use static IPs without carefully managing the address assignments.
- **Subnetting:** In larger networks, the subnet mask is used along with the IPv4 address to determine which devices are on the same subnet and which require routing through a default gateway. It is important to ensure that the subnet mask is properly configured.

3.5.6 Automatic IP Settings (DHCP)

Automatic IP Settings, also known as DHCP (Dynamic Host Configuration Protocol), is a network protocol that allows devices, like your IGX device, to automatically obtain an IPv4 address from a DHCP server on the network. This feature simplifies network configuration by eliminating the need for manually assigning a static IP address to each device on the network.

DHCP Basics

When you enable DHCP on your IGX device, it sends a request (called a DHCP Discover message) to the DHCP server. The DHCP server then responds by assigning an available IPv4 address to the device. This address is usually provided along with other network settings, such as the default gateway, DNS servers, and subnet mask.

Advantages of Using DHCP

- **Simplifies Configuration:** With DHCP, devices do not require manual configuration. The device will automatically obtain an IP address and other network settings, making it much easier to set up and manage devices on the network.
- **Reduces IP Management:** DHCP eliminates the need for network administrators to manually track and assign IP addresses, especially in large networks with many devices.
- **Automatic Renewal:** DHCP handles address lease renewal automatically, so devices maintain connectivity without manual intervention.

DHCP and Network Requirements

To use DHCP, your network must have a DHCP server. In most home or office networks, this server is typically built into the router. The router assigns IP addresses to devices on the network as they join. If there is no DHCP server, devices will not be able to automatically obtain an IP address and must be manually configured with a static IP.

Limitations of DHCP

- **Temporary IP Assignment:** IP addresses assigned by DHCP are temporary. The DHCP server leases the address to the device for a set period, after which the device must renew the lease. This usually happens automatically, but it can cause temporary disruptions if the lease is not renewed properly.
- **IP Address Conflicts:** If the DHCP server assigns the same IP address to two devices at the same time (due to network errors or misconfiguration), it can cause an IP conflict. This can disrupt communication between the devices. DHCP usually prevents this by tracking IP address assignments, but it's still a risk to consider.

3.5.7 Static Address

If DHCP (Dynamic Host Configuration Protocol) is not enabled or not available on the network, the static address option lets you manually assign a fixed IPv4 address to the IGX device. The device always uses the same address, so it can be reliably reached.

Static Address Advantages

- **Predictability:** A static IP address ensures that the device always has the same address, making it easier to configure other systems or software to communicate with it. For example, automation systems that communicate with the IGX need a predictable address.
- **Fallback:** Static addresses are also useful as a fallback when DHCP fails or is unavailable. If your network uses DHCP, but the server is down, the IGX device can still function by using a manually assigned static IP, preventing network disruption.
- **Controlled Network Setup:** In some situations, you may want more control over your network. Using static IPs allows network administrators to carefully plan and allocate IP addresses to each device in the network, avoiding potential issues that can arise from dynamic address assignments.

3.5.8 Static Netmask

The static netmask defines the size and scope of the local network (or subnet). It determines how the IPv4 address is divided between the network portion (which identifies the network itself) and the host portion (which identifies the specific device within that network).

Note: The static netmask is only used when you configure a static IP address. It won't be applied if the device is using DHCP.

Netmask Basics

- **Subnetting**: The netmask is used to separate the network and host portions of an IP address. For example, in a common class C subnet configuration, the netmask `255.255.255.0` means that the

first three octets of the IPv4 address (e.g., `192.168.1.x`) identify the network, while the last octet (the `x` in `192.168.1.x`) is used to uniquely identify each device on that network.

- **Routing**: The netmask also helps devices determine whether another address is on the same local network or if it needs to be routed through a default gateway to reach another network. For instance, if your IGX device has the address `192.168.1.10` and the **netmask** is `255.255.255.0`, it will assume any address in the range `192.168.1.1` to `192.168.1.255` is local, and any address outside of this range will be routed through the default gateway.

3.5.9 Static Broadcast

The static broadcast address is used for sending broadcast messages to all devices on the local network (or subnet). A broadcast message is one that is sent to all devices within a specific network range, rather than targeting a single device.

Note: The static broadcast setting is only honored when using a static IP address. If DHCP is used, the broadcast address is typically assigned automatically based on the subnet configuration.

Broadcast Address Basics

- **Communication with Multiple Devices:** The broadcast address allows a device to send messages to all devices on the local subnet. It is used by network protocols such as ARP, DHCP discovery, and mDNS for device discovery and address resolution.
- **Device Discovery:** Broadcast addresses are often used in protocols that involve automatic discovery of devices on the network, such as DHCP discovery or mDNS (multicast DNS) for finding devices on a local network.
- **Subnet Scope**: Just like the static netmask, the broadcast address works within a particular subnet. For example, if the static netmask is `255.255.255.0` and the static IP address is `192.168.1.10`, the broadcast address for the network would be `192.168.1.255`. This address allows messages to be sent to every device in the `192.168.1.x` network.

3.6 Universal Plug-and-Play (UPnP)

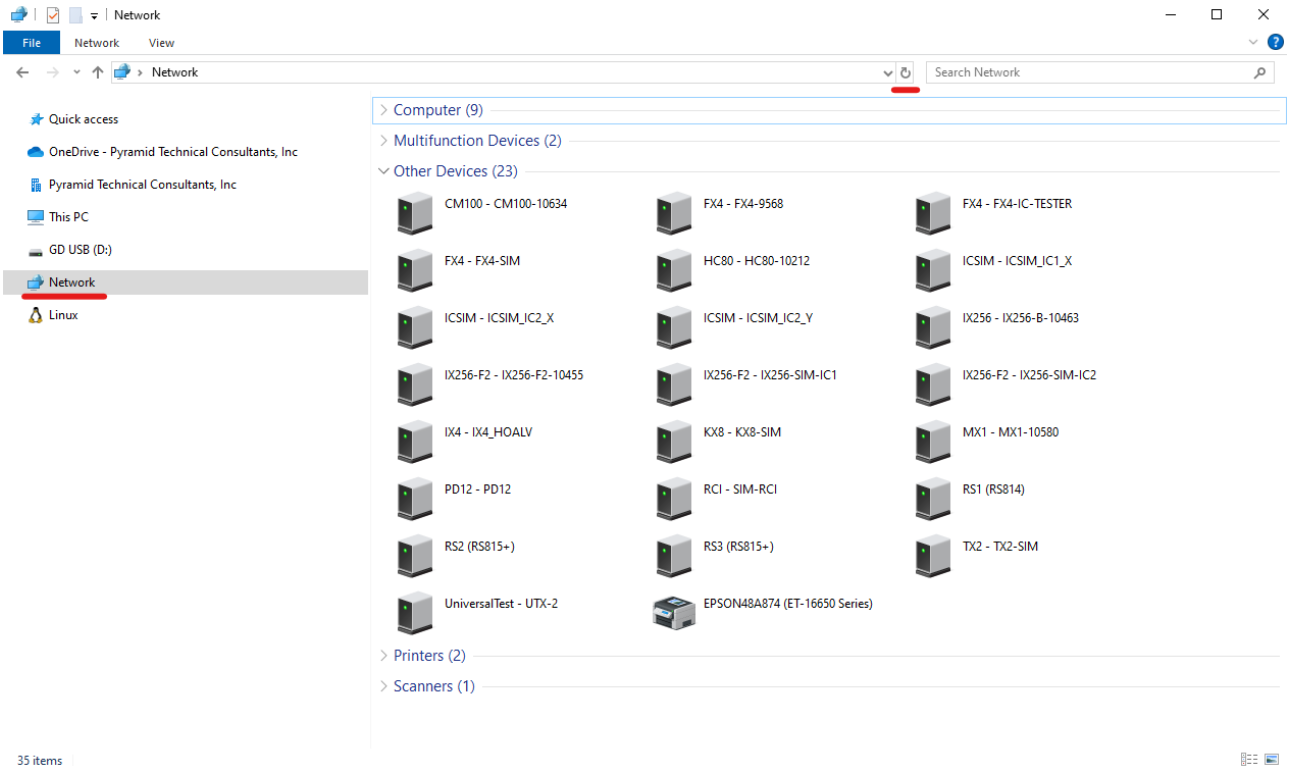
The IGX device comes with a UPnP server enabled by default. UPnP is a protocol that allows devices on a local network to automatically discover and communicate with each other. This can greatly simplify the setup process, as devices can be automatically recognized and configured without the need for manual intervention.

For example, when you connect your IGX device to the network, other devices or software on the network can easily detect the IGX device and establish communication with it. This is useful for scenarios where you want to quickly integrate the IGX into an existing network or control system without complex configurations.

3.6.1 Windows Network Explorer

In Windows, the File Explorer provides a simple way to discover and access IGX devices that have UPnP enabled. This eliminates the need to manually search for IP addresses.

1. Open File Explorer (`Win + E`).
2. Click on "Network" in the left-hand sidebar.
3. Press the refresh button at the top right to rescan for devices.
4. Locate the IGX device in the Network section.
   - Double-click on the device to automatically open its web GUI in your default browser.
   - Right-click on the device see device properties like IP address and serial number.



6 : Some of The Pyramid Office IGX Devices

This method is especially useful when working in networks where DHCP assigns dynamic IP addresses, making it easier to locate and interact with IGX devices.

3.6.2 Security Considerations with UPnP

While UPnP provides ease of use and convenience, it is also a controversial protocol. Some network administrators consider it a security risk because it not only allows devices to be discovered but also allows remote configuration of devices. This can make the network vulnerable to attacks, especially when devices are exposed to the internet or poorly secured networks.

The main concern is that UPnP enables external devices to request changes to the configuration of other devices on the network. There are known malware programs that specifically target UPnP-enabled devices in order to modify settings and potentially compromise the network's security.

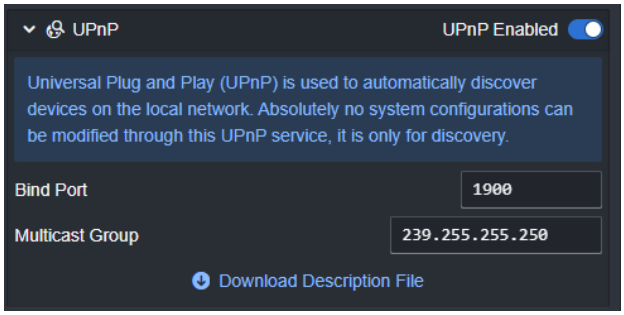
How UPnP Is Safeguarded on IGX

While UPnP discovery is enabled by default on the IGX device, it's important to note that IGX does not support any modification of device parameters through UPnP. This means that even though the device can be discovered on the network, no settings or parameters can be altered remotely via the UPnP protocol.

The only possible action an external device can take through UPnP is the discovery of the IGX device’s presence on the network. This discovery can be achieved through other methods, such as network scans, so UPnP does not provide any additional exposure beyond what’s already possible.

Disabling or Configuring UPnP

For users who are concerned about the security implications of UPnP, the IGX UPnP server can be easily disabled. You can simply click the switch next to UPnP in the network settings to turn off the UPnP server. This ensures that the device will no longer automatically be discoverable on the network via UPnP.



If UPnP is left enabled, the bind port and multicast group can be configured to suit specialized applications or network environments. For more advanced setups, the ability to control these settings provides flexibility in managing how devices are discovered and accessed on the network.

Additionally, if you need to retrieve more detailed information about the IGX device's UPnP configuration, you can use the XML description file available in the UPnP section. By clicking the button at the bottom of the UPnP settings, you can download this file, which contains detailed information about the device's network settings and services exposed via UPnP.

3.7 Network Topology Guidelines

IGX is a flexible platform that can scale from a single "control system in a box" to being one component of a much larger distributed system. Here we outline example network architectures for small, medium, and large-scale IGX deployments, along with tips for troubleshooting common networking issues:

There are many different ways to create perfectly valid networks using IGX devices. This intention of this guide is to create a set of standardized recommendations and best practices to help inform users of how Pyramid intended IGX to be used.

3.7.1 Network Layer Basics

In order to understand topology, you need to understand how the layers of a network function. Users familiar with layers 1, 2, and 3 networking can skip to the next section.

Layer 1: The Physical Layer

Layer 1 is responsible for the actual transmission of raw data over a network. It defines how bits (1s and 0s) are converted into signals and sent across a physical medium. If there is a problem with layer 1, none of the higher layers will work correctly.

Key characteristics:

- **Medium:** Copper cables (Ethernet), fiber optics, or wireless signals.
- **Transmission type:** Electrical pulses (Ethernet), light pulses (fiber optics), or radio waves (Wi-Fi).
- **Hardware involved:** Network interface cards (NICs), cables, connectors, and repeaters.

Why it matters for IGX:

- Poor cabling, interference, or weak signals can cause data loss or slow communication.
- High-performance IGX networks should use shielded Ethernet (Cat6 or better) or fiber to reduce noise and ensure low latency.

Layer 2: The Data Link Layer

Layer 2 handles local network communication by organizing data into structured frames and ensuring it reaches the correct device within the same network. This is where the concept of "packets" of data is first introduced.

Key characteristics:

- **MAC addressing:** Each network device has a unique MAC address, which is used to direct traffic within the LAN.
- **Switching:** Network switches direct traffic based on MAC addresses, ensuring that data reaches only the intended device.
- **Error detection:** Basic mechanisms like frame checksums help detect data corruption.

Why it matters for IGX:

- Fast, low-latency communication happens at Layer 2 because switches forward data directly without complex routing decisions.
- IGX devices that require real-time control (WebSockets, EPICS, etc.) perform best on a well-optimized Layer 2 network.
- VLANs (Virtual LANs) work at Layer 2 and can segment network traffic without requiring routing. However, the packets must be inspected one by one to determine their VLAN.

Layer 3: The Network Layer

Layer 3 enables communication between different networks by using IP addressing and routing. It determines how data moves across multiple subnets or larger infrastructures.

Key characteristics:

- **IP addressing:** Every device is assigned an IP address to identify it on a network.
- **Routing:** Routers (or Layer 3 switches) forward packets between different networks based on IP addresses.
- **Traffic control:** Layer 3 can enforce security policies, prioritize certain types of traffic, and optimize network paths.

Why it matters for IGX:

- When IGX devices are spread across multiple subnets, Layer 3 routing is required to allow them to communicate.
- Layer 3 adds security and control by allowing network segmentation and firewall rules.
- However, routing introduces additional latency, so critical real-time IGX communications should be kept within Layer 2 whenever possible.

Layer 2 vs. Layer 3: When to Use Each in IGX Networks

	Layer 2 (Switching)	Layer 3 (Routing)
Function	Local communication within the same network	Communication between different networks
Addressing	MAC addresses	IP addresses
Performance	Very low latency	Slightly higher latency due to routing
Security & Control	Basic (VLANs for segmentation)	Advanced (firewalls, traffic policies)
Best for IGX	Real-time control, low-latency communication	Managing multiple networks, security enforcement

For IGX real-time performance, prioritize Layer 2 networking with VLANs. Use Layer 3 routing when multiple independent networks need to communicate while maintaining security and manageability.

3.7.2 Network Topology Basics

When setting up a network, the way everything connects together affects speed, reliability, and security. The best approach for IGX networks is a star topology, where all devices connect to a central network switch. This design is simple, efficient, and easy to troubleshoot.

But how many stars should your network have? Should all devices be connected to a single switch, or should they be spread across multiple switches? And should those switches be physically separate or grouped into virtual networks (VLANs)? Let's break it down in simple terms.

Single-Star vs. Multi-Star Networks

A star topology has one central switch with IGX devices, computers, and other equipment connected to it. The number of stars in your network depends on how many devices you need to connect and how much separation is needed for performance and security.

1. Single-Star Network (One Central Switch)

- **Best for:** Small setups with a few IGX devices.
- **How it works:** Everything connects to one switch, making communication fast and simple.
- **Downsides:** If the switch fails, everything goes offline. Too many devices can slow things down.

2. Multi-Star Network (Multiple Switches)

- **Best for:** Larger setups where different systems need separation or redundancy.
- **How it works:** Each group of devices gets its own switch, and the switches connect to a core switch (a "network hub" for all the stars).
- **Advantages:**
  - Improves performance by reducing network traffic in each group.
  - Provides backup paths in case one switch has issues.
  - Allows different departments or machine groups to have their own dedicated networks.

Using VLANs vs. Separate Switches

When setting up multiple stars, you can separate traffic in two ways:

1. VLANs (Virtual Networks on One Switch)
2. Physically Separate Switches

Both methods create independent networks, but they work differently:

	VLANs (Same Switch)	Separate Switches
Cost	Cheaper (one switch)	More expensive (multiple switches)
Complexity	Requires configuration	Simple plug-and-play setup
Security	Strong, but depends on setup	Maximum physical separation
Performance	Good for most cases	Best for very high-speed needs
Failure Impact	If the switch fails, all VLANs go down	Each switch is independent

When to Use VLANs

- If you want separation but only have one switch.
- If you need flexibility, as VLANs can be reconfigured easily.
- If security is important, but not absolutely critical.

When to Use Separate Switches

- If different systems must never share traffic (e.g., medical safety systems).
- If performance is the top priority, and you don't want any congestion.
- If you need hardware redundancy, one switch failing won't affect others.

Questions to Ask Yourself

- **How many IGX devices and clients do I have?** → If many, consider multiple stars.
- **Do I need strict security between systems?** → Separate switches are safer.
- **Do I want to keep costs down?** → VLANs allow separation without extra hardware.

- **Do I need the absolute fastest communication?** → Separate switches prevent traffic slowdowns.

3.7.3 Understanding Network Congestion

Network congestion occurs when too much data is trying to move through a network at once, overwhelming the available bandwidth and causing delays, dropped packets, or complete communication failures. In high-speed, low-latency IGX networks, congestion can severely impact real-time control and data transmission.

At a basic level, a network has a finite amount of *bandwidth*, the total capacity it can handle at a given time. Congestion occurs when the *demand* for data transfer exceeds this bandwidth, leading to bottlenecks. If a network is running at or near 100% utilization, latency increases and packets start getting dropped. This can happen for several reasons:

1. **Too Many Devices Sending Data at Once**
   - When multiple devices (IGX controllers, clients, logging systems) send large amounts of data simultaneously, the switch or router may struggle to keep up.
2. **Overloaded Switch Ports or Links**
   - A single switch port handling too much traffic can become a bottleneck, even if the switch itself has plenty of capacity.
   - Uplinks between switches can also become congested if they aren't fast enough to handle all the connected devices.
3. **Broadcast and Multicast Storms**
   - Some network traffic, like discovery protocols, EPICS PV searches, or poorly configured multicast streaming, can flood the entire network.
   - If every device has to process unnecessary traffic, overall performance degrades.
4. **Quality of Service (QoS) Misconfiguration**
   - If priority settings are not used properly, critical IGX traffic might get delayed behind lower-priority data (like large file transfers or non-essential telemetry).
5. **Routing and Processing Delays**
   - In Layer 3 networks, routers must inspect and forward packets, adding processing time.
   - If a router or firewall is overloaded, it can create a bottleneck that slows communication.

To relive network congestion, you can take a number of steps including:

1. **Upgrade to Gigabit or 10G Switches**
   - Replace older 100 Mbps switches with 1 Gbps or 10 Gbps models, especially on high-traffic links.
   - Ensure that uplinks between switches are at least as fast as the busiest device connections to prevent bottlenecks.
2. **Add Dedicated Switches for High-Priority Traffic**
   - Instead of connecting all devices to a single switch, distribute traffic by adding dedicated switches for different functions (e.g., one for IGX control traffic, another for logging and monitoring).
   - Use a core switch to connect these smaller switches efficiently.
3. **Use Managed Layer 2 and Layer 3 Switches**
   - Layer 2 switches forward data within the same network using MAC addresses, ensuring low-latency direct communication.
   - Layer 3 switches handle routing between different networks but with higher efficiency than traditional routers.
   - Use Layer 2 switches for IGX control networks and Layer 3 switches only where routing is required.
4. **Use VLANs to Isolate Traffic**
   - VLANs allow different types of traffic to share a single switch while remaining logically separated. This prevents non-critical traffic from interfering with real-time control.
     For Example:

- VLAN 10: IGX control traffic (real-time, high priority)
- VLAN 20: Monitoring and logging (lower priority)
- VLAN 30: Administrative access (management and SSH)

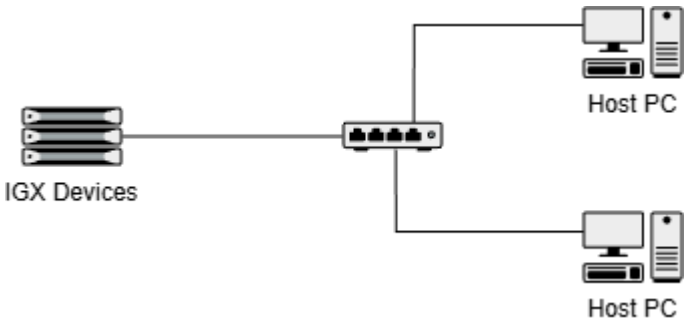
5. **Prioritize Traffic with QoS**
   - Configure Quality of Service (QoS) settings on managed switches to prioritize IGX control data over bulk transfers and background traffic.
   - Assign higher priority to WebSockets and EPICS control messages to ensure low-latency communication.

3.7.4 Recommended Network Architectures

This list is not exhaustive and should only be considered a starting point for network design. If you or your team require assistance in specific network setups, please feel free to contact Pyramid directly for personalized recommendations.

Small-Scale Deployment

For a small lab setup or a standalone medical device, you might have one IGX device connected to a couple of operator PCs. A simple star topology is typically used, e.g., a single network switch to which the IGX device and the HMI computer are connected. All devices reside on a single IP subnet (perhaps a private range like 192.168.x.x). This flat network makes it easy to set up and is low latency.



IGX can broadcast and be discovered by the PC easily. Use DHCP if you have a router; if not, IGX will fall back to a default IP (192.168.100.20) which you can set your PC to match for initial access. Security in this scenario may rely on an airgap or the fact that it's a closed network, but it's still wise to put the switch behind a firewall.

Troubleshooting is straightforward, check that the IGX and PC have IP addresses in the same range and that link lights are on.

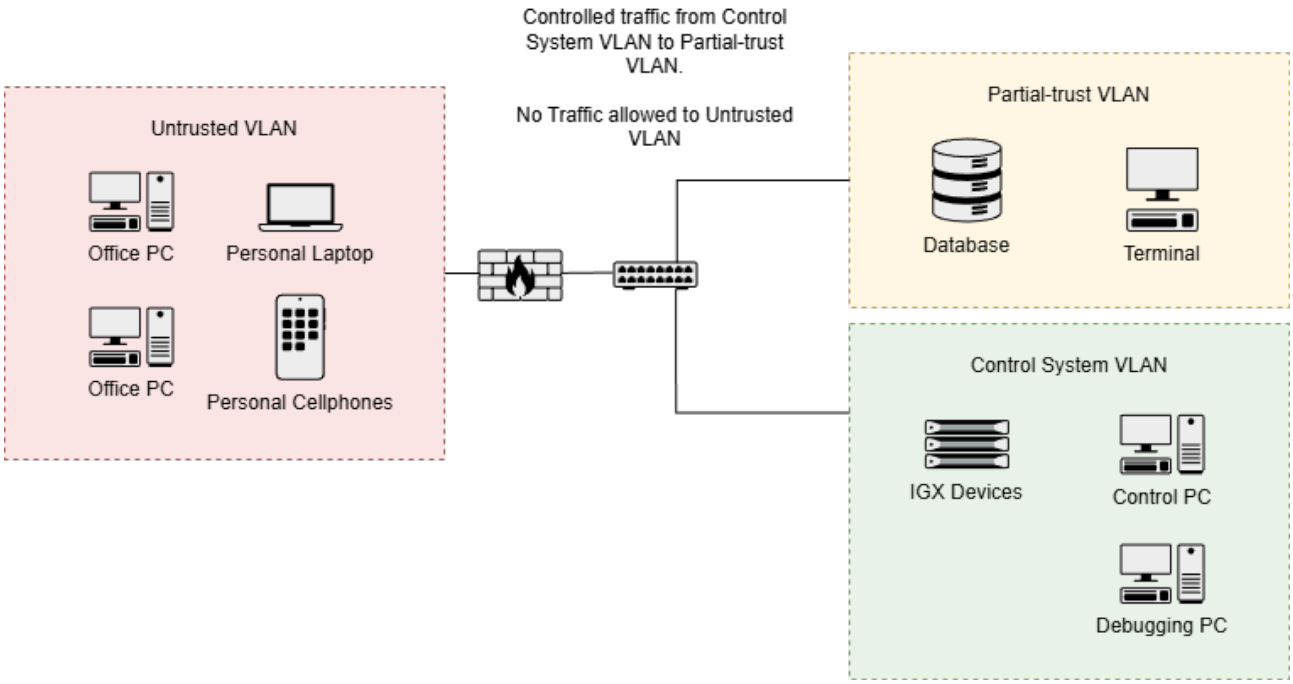
With only a few devices, latency will be negligible, any noticeable delay is likely due to a misconfiguration, using a gigabit switch can help ensure bandwidth isn't a bottleneck (IGX's data, even streaming, is usually not huge, but gigabit gives more headroom).

Medium-Scale Deployment

In a medium deployment, you might have several IGX controllers (for different machines or therapy delivery systems, for example) and multiple operator stations, data servers, etc.

Here a segmented star topology is a good approach. For instance, use a core Layer 3 switch and multiple VLANs: one VLAN per machine or per critical subsystem. Each VLAN can host one or more IGX devices and their direct clients. Within a VLAN, it's L2 switching for speed, but VLANs are isolated from each other to contain broadcasts and enhance security.

The Layer 3 core routes between VLANs where needed, e.g., an engineering workstation on VLAN 10 can reach an IGX on VLAN 20 for maintenance, but perhaps normal office devices on VLAN 30 cannot reach VLAN 20 at all (enforced by ACL).



This design could include a redundant core switch for failover, and maybe each VLAN spans two switches (with MSTP/RSTP to block one path until needed).

Implement firewall rules at the router/L3 switch for any inter-VLAN traffic. A common practice is to have a dedicated “control network” VLAN (or set of VLANs) and only allow specific services to/from the “business network.” Perhaps an IGX device needs to push data to a central database in the IT network, you’d open only that path (DB server IP/port) rather than full access.

In a medium network, enabling QoS on the switches becomes important, as multiple devices could generate traffic. Give IGX control protocols the highest priority across the switching infrastructure.

Ensure uplinks (inter-switch links) are sized appropriately (e.g., use gigabit or 10Gb uplinks if multiple 100Mb devices are sending data simultaneously).

Common issues in this scenario include VLAN misconfiguration, e.g., IGX on VLAN 20 cannot talk to a client on VLAN 10 because the routing or ACL is mis-set. Always verify the VLAN tagging and IP addressing. If EPICS PVs aren’t visible across VLANs, remember that you might need an EPICS gateway or to configure the EPICS CA address list (since broadcasts won’t cross VLANs without help). If latency spikes or jitter is observed, check if some non-critical traffic is saturating an uplink (a traffic storm or large file transfer), that’s where QoS or adding bandwidth can fix it.

Large-Scale Deployment

In a large deployment, dozens of IGX devices might be distributed across a facility (for example, many control units in a factory or multiple therapy rooms in a hospital). This calls for a hierarchical network architecture. Typically, you’d have an Access layer (industrial switches in each area connecting local IGX devices), Distribution/Aggregation layer (regional switches that gather those access switches, possibly with routing), and a Core layer (high-speed backbone connecting to data centers or campus network).

At this scale, following the Purdue Model or similar OT/IT separation is strongly recommended i.e., the IGX and other control devices reside in an *Industrial Zone* (Levels 0-2 for devices, Level 3 for control center/SCADA servers) which is separated by firewalls from the *Enterprise Zone* (Level 4 office/ business network). Concretely, you might have all IGX networks aggregated into an OT core switch, which connects to an enterprise firewall; on the other side of the firewall is the corporate core. This ensures any traffic between IGX systems and corporate systems is mediated and inspected. Within the industrial network, use multiple VLANs or even separate physical networks for different lines/ departments (to limit the blast radius of any issue).

Large networks must tolerate multiple failures. Deploy redundant core switches (in a failover pair), redundant distribution links (use link aggregation or dual links with one blocking via STP for backup). Mesh topologies might be used at the core/distribution level for fast failover – e.g., a ring of distribution switches connecting different production areas. Also consider redundant devices: if an IGX is controlling a critical process, you might have a hot standby unit or at least a strategy to quickly swap in a spare (this is more of a system design than pure network, but it interacts with network addressing/DNS – you could use a virtual IP that a backup controller takes over if the primary fails).

With many devices, monitor network utilization. Ensure the core and distribution have high throughput (10GbE or more) so they don't become bottlenecks when many IGXs are streaming data or during peak periods. Implement network time protocol (NTP) or IEEE 1588 PTP if precise time synchronization is needed across devices, sometimes necessary for correlating data in industrial/medical systems, though IGX's use cases will determine that.

Use a centralized network management system to keep track of switch configurations, VLAN assignments, and device connections. Document each IGX device's switch port and IP. This aids in troubleshooting.

As networks scale, issues like IP address conflicts, DHCP quirks, or routing misconfigurations can crop up. It's often useful to statically assign IPs to critical IGX devices (or use DHCP reservations) to have predictable addresses. If an IGX device is unreachable, check if it might have gotten the wrong IP or if a duplicate IP on the network is knocking it offline. Also verify route tables, e.g., the PC trying to reach the IGX might be on a different subnet and if the route or firewall isn't set up, it will fail. Tools like ping and traceroute are still your first go-tos.

For latency issues, a large network might introduce routing delays, ensure that any routers/switches handling control traffic have appropriate performance (industrial switches with fast throughput). If using any WLAN or VPN in the mix (perhaps for remote monitoring tablets), be mindful that wireless can add latency and occasional packet loss and stick to wired for primary control paths and use wireless only for non-critical monitoring or redundant paths.

3.8 Running a Local NTP Time Server

3.8.1 Overview

Sometimes when you are running a control system on a closed network, for example in industrial or medical systems, it can be challenging to propagate accurate times across the network without any access to the internet. One solution to this problem is using a Windows computer as a gateway between the internet and the closed control system. The Windows computer gets an accurate time from the internet and passes it along to all the devices downstream in the closed network.

Setting up an NTP (Network Time Protocol) server on Windows 10 or 11 involves configuring the Windows Time service. This is a built-in, but not enabled by default, service that can be configured to run in the background.

Important Notes Before Starting

- Make sure your Windows machine's time is synchronized with a reliable time source to maintain accurate time on your NTP server. This means the Windows machine must have an internet connection or some other source of accurate time information, for example a GPS transceiver.
- Ensure that your network and security policies allow NTP traffic. NTP traffic going to a Windows computer is not typical, your IT department may have a policy against this.

3.8.2 Steps to Configure Windows as an NTP Server

By following these steps, you should be able to configure your Windows 10 or 11 machine to act as an NTP server that other devices can use for time synchronization.

1. **Open Command Prompt as Administrator**
   - Press `Win + X` and select "Command Prompt (Admin)" or "Windows PowerShell (Admin)".
2. **Stop the Windows Time Service**

- Execute the following command to stop the Windows Time service:

```
net stop w32time
```

3. **Configure the Windows Time Service to Use an Internal Time Source**
   - Execute the following command to configure the Windows Time service:

```
w32tm /config /manualpeerlist:"0.pool.ntp.org 1.pool.ntp.org 2.pool.ntp.org
3.pool.ntp.org" /syncfromflags:manual /reliable:YES /update
```

4. **Set the Announce Flags to 5**
   - Execute the following command to configure the Announce Flags registry entry to 5:

```
w32tm /config /reliable:YES /update
```

5. **Enable NTP Server**
   - Execute the following command to set the server as an NTP server:

```
reg add HKLM\\SYSTEM\\CurrentControlSet\\Services\\W32Time\\TimeProviders\
\NtpServer /v Enabled /t REG_DWORD /d 1 /f
```

6. **Start the Windows Time Service**
   - Execute the following command to start the Windows Time service:

```
net start w32time
```

7. **Configure the Windows Firewall to Allow NTP Traffic**
   - Execute the following command to open the NTP port (UDP 123) in the firewall:

```
netsh advfirewall firewall add rule name="NTP" dir=in action=allow protocol=UDP
localport=123
```

Verify Configuration

To verify that your configuration is correct, you can use the following command:

```
w32tm /query /status
```

This command provides status information about the Windows Time service.

Additional Configuration for NTP Clients

Ensure that the client devices are configured to use your Windows NTP server as their time source. This usually involves specifying the IP address of your Windows NTP server in the NTP settings of each client device. If your NTP client is an IGX device, these settings can be found under the “Admin” settings in the system clock configuration section.

3.9 Security Best Practices for IGX Networks

Industrial and medical networks face significant cybersecurity risks, so network design must incorporate strong security without compromising the real-time performance of IGX devices. Key best practices include network segmentation, traffic filtering/prioritization, and secure access controls.

3.9.1 VLAN Segmentation

Use VLANs to isolate critical devices and traffic. For example, put IGX controllers and associated HMIs on a dedicated VLAN separate from the corporate office network or other lab equipment. This containment limits exposure - an office PC malware outbreak won’t directly hit the IGX control VLAN.

VLANs create separate broadcast domains, which also improves performance by limiting unnecessary traffic to IGX devices. It’s recommended to create VLANs by function (e.g. control data, vision cameras, admin access, etc.) and then use Layer 3 rules to tightly control what crosses between

them. In an industrial context, VLANs (or deeper segmentation like separate subnets and firewalls) implement the “zones” of the ICS Purdue model, keeping the control layer (Level 1/2) isolated from general enterprise traffic. Ensure your VLAN design is documented and that only the necessary VLANs are trunked to switches where needed (to avoid VLAN hopping issues).

3.9.2 Quality of Service (QoS)

Implement QoS rules to prioritize IGX traffic on the network. Industrial protocols and control commands should take precedence over miscellaneous traffic. Managed switches can classify and prioritize packets (for example, by tagging IGX’s HTTP/WS traffic or giving high priority to EPICS or other control UDP packets).

By assigning higher priority to time-sensitive data, switches ensure minimal delay and jitter for those packets. For instance, you might set IGX’s control VLAN to have a guaranteed bandwidth or highest priority queue, whereas a camera streaming VLAN might be lower priority.

QoS is especially important if the network is converged (carrying both control and non-control data), it prevents large file transfers or video streams from introducing latency to critical IGX communications.

Classify by protocols or VLAN and avoid over-complicating (too many priority levels can be hard to manage). Also, verify end-to-end support. If traffic goes through routers, they should preserve or reapply the QoS markings.

3.9.3 Access Control & Firewalls

Lock down network access to IGX devices. Only allow the necessary protocols/ports and only from authorized IP ranges. For example, if IGX should be accessed by an engineering workstation and a SCADA server, then configure an ACL or firewall rules so that only those systems (and perhaps a jump host for remote access) can reach the IGX’s IP.

Block internet access to IGX entirely unless absolutely required. Place a firewall between the IGX network (OT network) and any IT network or internet connection, this is a core tenet of ICS security. The firewall can permit required services (e.g. allow HTTPS/SSH from IT for maintenance or allow database connections from IGX out to an IT historian server) but should default-deny anything else.

Within the control network, use switch ACLs or port security to prevent unauthorized devices from plugging in (e.g., only allow known MAC addresses or implement 802.1X authentication if devices support it, note many medical devices might not, in which case rely on network infrastructure). Regularly review these rules to adapt to any new threats.

Also consider internal firewalls or “cell firewalls” between segments of the OT network for defense-in-depth. For instance, if you have multiple IGX devices across different lab areas, firewall rules can prevent one area’s IGX from talking to another’s unless needed, reducing the impact if one is compromised.

3.9.4 Cyber Hygiene & Monitoring

Even with segmentation, assume that attacks can happen. Implement intrusion detection or at least monitoring on the IGX network. Many industrial IPS/IDS tools can watch for anomalous traffic (like strange SSH attempts or unfamiliar protocol commands). Ensure all IGX devices have up-to-date firmware security patches (apply the Pyramid-provided updates in a timely manner).

Disable any unused services on the IGX if possible (minimize open ports). Monitor network logs, for example, keep an eye on login attempts on IGX (via its syslog or auth log accessible through SSH).

In medical settings, also ensure compliance with any standards (like HIPAA, if patient data is involved, or FDA guidance on networked medical device security).

3.9.5 Secure Remote Access

If IGX devices need remote access (for vendor support or remote engineers), do not expose them directly. Instead, use a VPN into the network or a secure jump server. Remote access should be

provided by secure solution with zero-trust access controls and monitoring. That means enforce multi-factor authentication for VPN or remote desktop logins, restrict remote users to only the necessary VLAN or host, and log all remote sessions.

Consider using an encrypted VPN tunnel that terminates in the corporate DMZ or an OT DMZ, from which the IGX network is reachable. Pyramid's IGX uses SSH, so one approach is to require users to VPN in, then SSH to the IGX device (with key authentication). Another approach is setting up a dedicated bastion host on the IGX network that authorized users can RDP/SSH into, and from there access IGX devices. In any case, never leave IGX management ports accessible from the public internet.

Finally, when remote access is not in use, keep it disabled.

4 IGX Web GUI Guide

IGX includes a built-in web-based GUI for configuring and monitoring devices. This interface functions similarly to a standard website:

- Users enter the IGX device's address in a web browser.
- The browser automatically communicates with the IGX device, acting as a web server.
- Once connected, the browser retrieves and displays the GUI, allowing users to interact with device settings and status information.

Since the IGX web GUI follows familiar web-based interaction patterns, no special software is needed. Any modern web browser can access it, making device management straightforward and intuitive.

4.1 Multiple Client Support

The IGX web server is designed to support multiple clients simultaneously, ensuring that all connected users see synchronized, real-time data. However, each additional client increases CPU load on the device. If you notice performance issues, try closing extra connections, such as multiple browser tabs or instances of the interface running on different computers. This can help improve responsiveness.

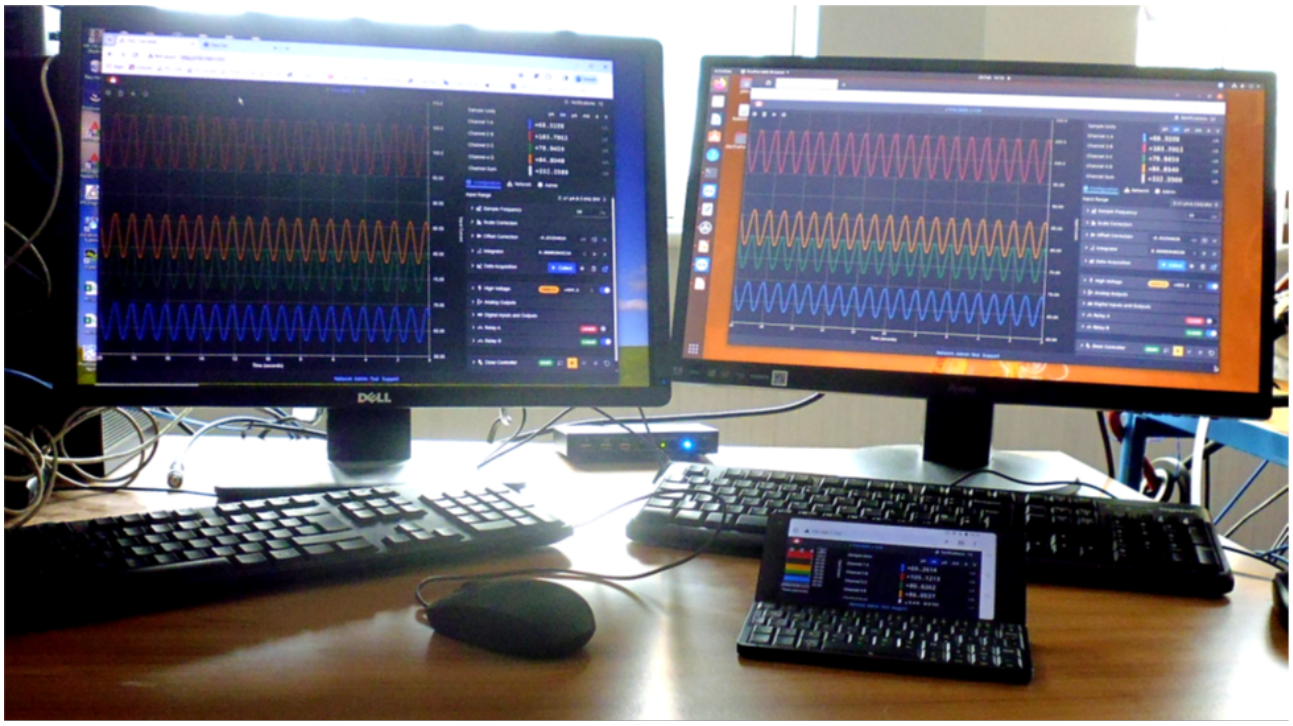
Additionally, users should exercise caution when modifying device settings while multiple users are connected to the same IGX device. Since settings are synchronized across all clients, changes made by one user will immediately update for all others, potentially leading to unintended misconfigurations if multiple users attempt to configure the system at the same time.

4.2 Browser Compatibility

IGX is primarily tested on Google Chrome and Microsoft Edge, making these the recommended browsers for optimal performance. If users encounter issues while using the web GUI, switching to Chrome or Edge may resolve them.

Firefox and Safari are also supported; however, certain features may not function correctly in some cases. If you experience any problems while using these browsers, please report the issue to Pyramid for further investigation and potential fixes.

The picture below shows web pages served by one FX4 to Windows, Linux and Android machines via wired Ethernet and WiFi.



4.3 Browser Cache

When you access the IGX web GUI, your browser caches (stores) certain files, such as JavaScript, CSS, and HTML, to improve loading speed for future visits. This means that instead of downloading everything again, the browser reuses stored copies of these files.

However, caching can sometimes cause issues when the IGX software is updated. If the browser continues using old, cached files, the interface may not display correctly, or new features may not appear. Clearing the cache forces the browser to fetch the latest version of the web GUI from IGX.

If the IGX web GUI is not displaying correctly after an update, performing a **hard refresh** forces the browser to reload the page without using cached files.

Use the following key combinations to perform a hard refresh in your browser:

- **Windows/Linux:** Press **Ctrl + Shift + R**
- **Mac:** Press **Cmd + Shift + R**

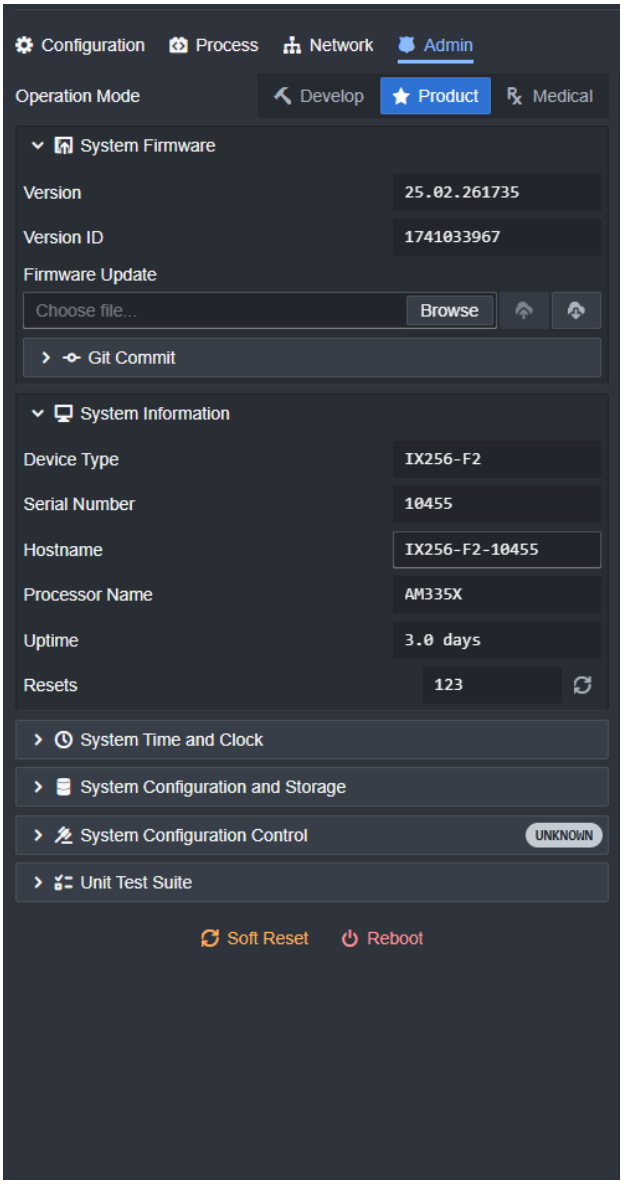
This ensures that the latest version of the IGX web interface is loaded, resolving potential display issues caused by outdated cached files.

5 IGX Admin Guide

IGX devices come with a standardized set of administration tools and configurations that simplify the management of the system. These tools help maintain consistency across devices and streamline the tasks for system administrators. The Admin Settings provide a centralized interface to control and monitor various aspects of the device's operation, making it easier to configure, maintain, and troubleshoot.

5.1 Key Functions of Admin Settings

- **Operation Mode**: Switch between different operational modes depending on the specific requirements of your setup.
- **System Firmware**: You can update or manage the firmware of the IGX device through the admin settings.
- **System Information**: Detailed system information, including the hardware details, and status of key system components.
- **System Time and Clock**: You can configure the clock and system time settings in the admin section.
- **System Configuration and Storage**: Control and monitor how parameters (IO), configuration files, and data are stored.
- **System Configuration Control**: Controls how the device prevents or permits modification of controlled parameters (IO).
- **Unit Test Procedures**: Some IGX devices come with built-in unit test procedures that can be executed via the admin settings.
- **Reset Buttons:** Allows the user to reset the software remotely.



7 : IGX Admin Settings

5.2 Operation Modes

IGX supports three modes of operation, each with specific rules and restrictions governing what can and cannot be modified on the device. These modes affect the device's functionality and access to certain parameters, particularly Input/Output (IO) settings. Some features and parameters of the device may only be available or accessible in a specific mode, while others may be restricted to ensure proper usage and prevent accidental modifications.

5.2.1 Develop Mode

This mode is primarily intended for Pyramid employees and engineers. It allows the modification of factory-level parameters and access to settings that are generally restricted in other modes. Developers and technical staff can use this mode to adjust internal settings, test new software, and modify calibration parameters.

Use Cases

- **Calibration**: It can be used by anyone calibrating a device to modify calibration parameters that normal users should not change.
- **Development:** Access to low level IO that lets developers create and debug software.

5.2.2 Product Mode

This is the default operational mode for most IGX devices. Product mode is for the regular daily use of the device and allows normal functionality. It provides flexibility in modifying controlled parameters (such as measurement settings) for general operations. This mode is for typical laboratory or industrial usage, with no need for restricted access.

Use Cases

- **Normal Device Use**: Ideal for end-users who need to configure the device for their needs but still want to prevent accidental low-level system changes that could affect calibrations.

5.2.3 Medical Mode

Medical Mode is designed to enhance safety and regulatory compliance, particularly in medical environments. It restricts access to protected settings, preventing *accidental* modifications and ensuring the device operates consistently and safely. This is especially important in medical applications, where incorrect configurations could have serious consequences.

However, Medical Mode alone is not a complete safety solution, it is intended to be one component of a broader safety system. It should be used alongside other safety measures, such as:

- Hash verification to ensure software configurations.
- Network-level security to protect against unauthorized access.
- Operator training processes to ensure proper device use.

By integrating Medical Mode with these additional safeguards, the system achieves a higher level of reliability and compliance in medical applications.

Use Cases

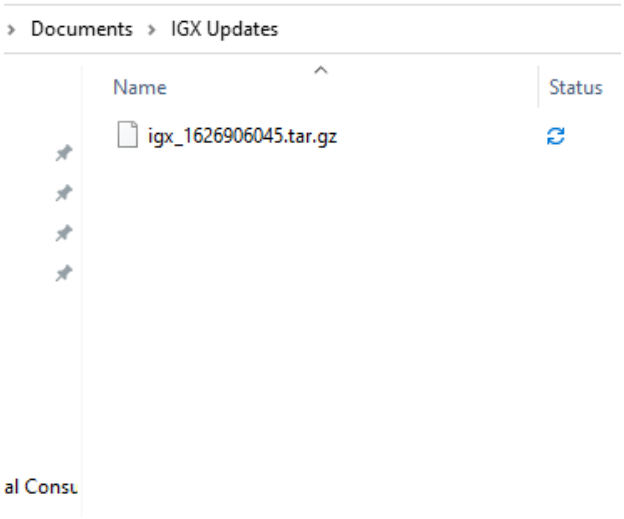
- **Preventive Safety**: Medical professionals or administrators can use this mode to ensure the device is protected from accidental changes to sensitive parameters, providing an extra layer of safety when using the device in high-stakes environments.

5.3 IGX System Firmware Update Guide

5.3.1 Identify the Firmware File

An IGX firmware file is usually stored in a `.tar.gz` format, with the name "igx" followed by the version number. This file contains all the binaries and configurations required for updating the device. The operating system (QNX) and supporting libraries are stored separately in a different file system.

Do not decompress the file before uploading it to the device, as the device expects a compressed file and will fail to update without one.



8 : A typical IGX firmware file

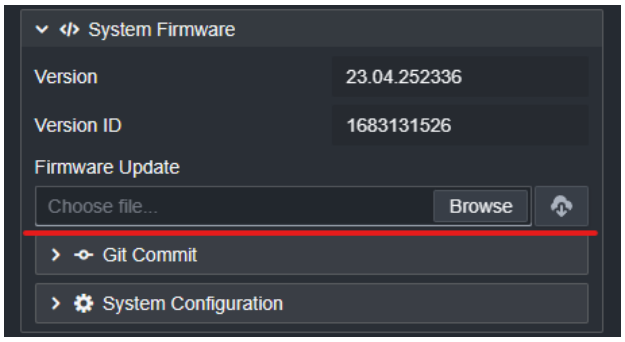
5.3.2 Uploading the Firmware

The process for accessing the update screen may vary between devices and software versions. Typically, there will be an "Admin" link that directs you to the relevant page.

If unsure, you can enter the direct URL into your browser to access the admin page, such as `http://<Device IP Address>/io/admin` (e.g., http://192.168.100.20/io/admin). Most modern browsers do not require the "http://" prefix, so you can omit it.

Upon accessing the admin page, you will see an interface similar to the images shown in the original documentation. The procedure for updating the firmware remains the same, regardless of the interface's appearance.

On the latest versions of IGX, the GUI should look like the following:



9 : IGX firmware upload interface.

Optionally you can click the download button to download a copy of the last uploaded firmware. This is useful for recovering back to an older version.

Click on the “Browse” button to select your update file from your local computer.

Then the download button should change into an upload button. Click the upload button to install the selected update.

5.3.3 Post-Firmware Update

The device will automatically restart after the firmware update. You will temporarily lose connection to the device, which is normal.

Do not manually refresh the page during the update or power down the device until it has restarted. If the device has not restarted after 3 minutes or you cannot reconnect, try power cycling the device to

resolve any unexpected issues. If the device still malfunctions, contact Pyramid's support team for assistance.

5.3.4 Update Validation and Troubleshooting

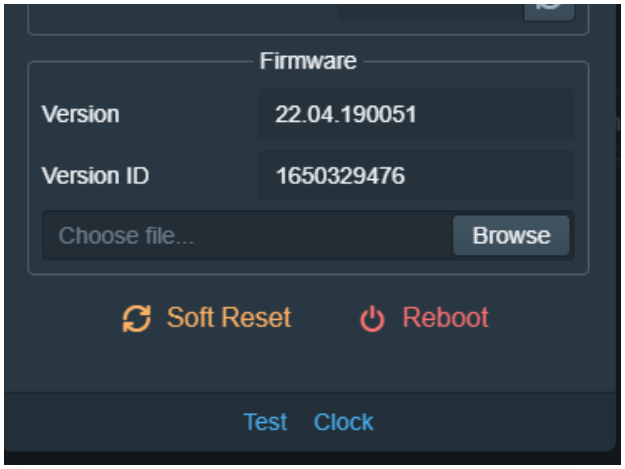
Verify that the device is functioning as expected. If necessary, power cycle the device to resolve any issues that may have arisen after the update.

> If you are upgrading from a significantly older firmware version, manually power cycling the device might be required to complete the update.

Contact Pyramid's support team if you require further assistance.

5.3.5 IGX Historic System Firmware Updating

In older versions of IGX, you may find the admin page looking like this. While the interface itself is different, the procedure of updating will be the same.

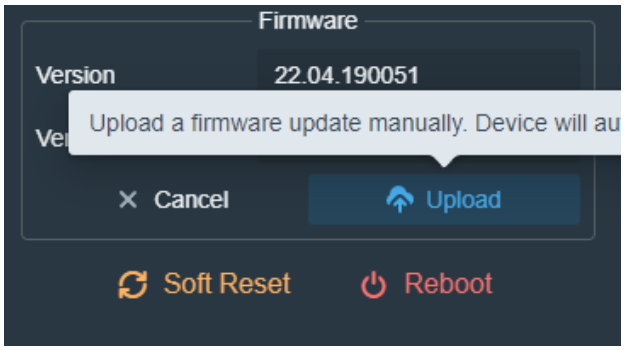


10 : Before a file is selected

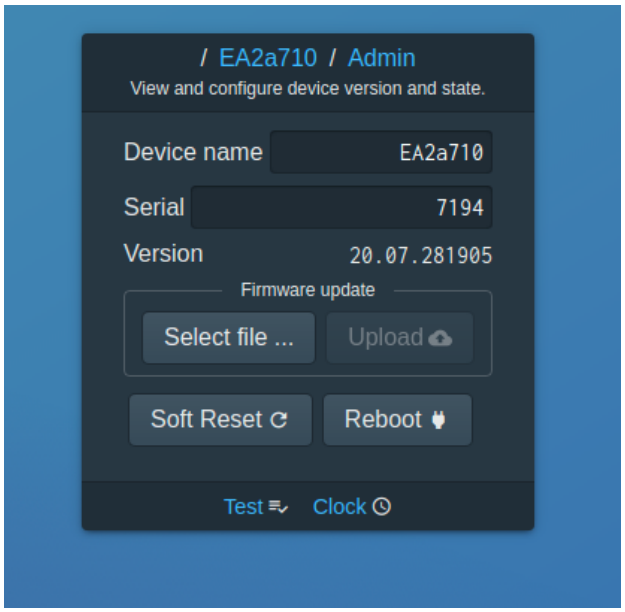
Click on “Browse” and select the update file from your local machine.

After selecting your file, the “Upload” button should appear.

Click the “Upload” button and the update process will begin.



11 : After the file is selected



12 : Historic file upload interface

Press the “Select file ...” button and select the compressed update file on your system.

Then press the “Upload” button. The update will automatically start and then reset the device.

5.4 IGX Permissions and Configuration Control

Within the IGX system, each IO may be optionally linked to a distinct permission policy. This policy is designated by a unique identifier in the form of a simple string. It includes a range of critical features that bolster the management and safeguarding of the system's IO activities.

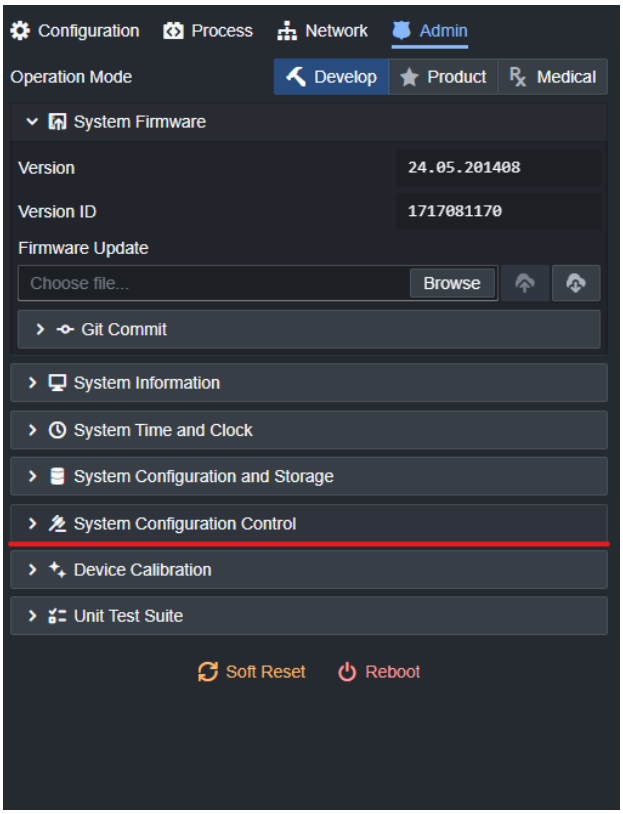
Key Features of IO Permission Policies

- **Write Permit Control**: Each permission policy includes a write permit. This write permit dictates whether the IO operations governed by the policy are writable. By controlling the write permissions systematically, the system ensures consistent and secure write operations across all IO.
- **Systematic Write Control**: The inclusion of write permits in the permission policies provides a structured approach to manage write access. This systematic control mechanism ensures that write operations are only performed when authorized, thereby maintaining the integrity and security of the control system.
- **IO State Monitoring**: Beyond controlling write operations, the permission policy maintains a hash of all IO assigned to it. This hash acts as a comprehensive record, enabling both internal and external systems to monitor the state of all associated IO.
- **Configuration Consistency**: The hash maintained by the permission policy ensures that the system configuration remains consistent and in the expected state. This eliminates the need for monitoring each IO individually, thereby simplifying the monitoring process and enhancing system reliability.
- **External and Internal System Integration**: The hashed record of IO states facilitates seamless integration with both internal and external monitoring systems. This integration ensures that any discrepancies or unauthorized changes in the system configuration can be promptly detected and addressed.

The GUI for monitoring and controlling permission policies and configuration control can be found under the admin tab under the “System Configuration Control” section.

Here you will find a list of all available permission policies on the system and their current status.

This GUI is mostly helpful for users wishing to observe the current state of a particular policy and gain a better understanding of the interlocks that limit it.



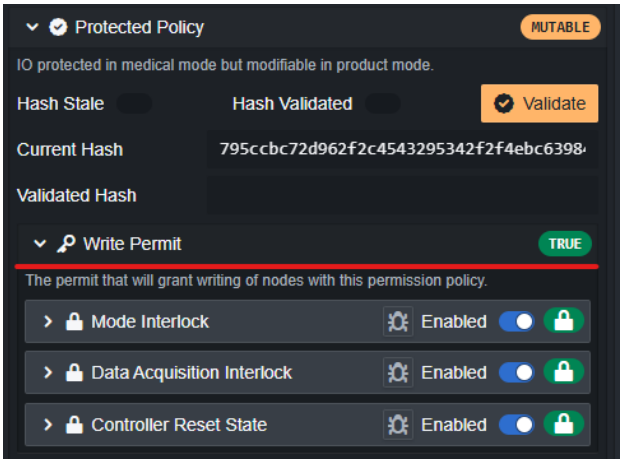
13 : Location of Permission Policy Control GUI

5.4.1 Permission Policy Write Permit

Each permission policy has a **write permit** associated with it. When this permit is granted (true) then the IO are writable, assuming nothing else is preventing it. If the permit is revoked (false) then the IO can only be read.

Different policies will have different sets of configurable interlocks that drive this behavior. These interlocks will observe the system state and react to changes.

See the IGX Using Interlocks and Permits (see page 65) for more information on how permits and interlocks work.

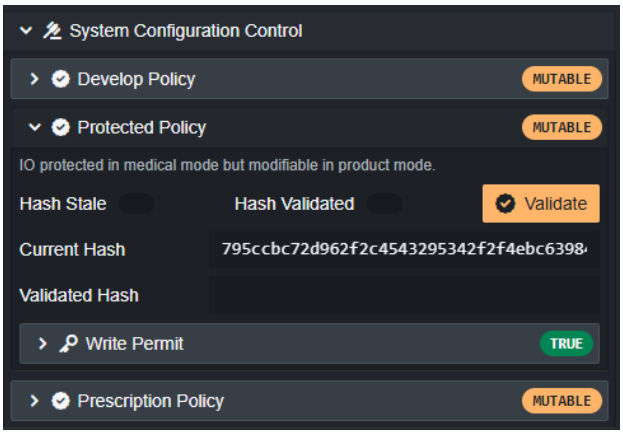


14 : Permission Policy Write Permit

5.4.2 Permission Policy Hash

Each permission policy has a SHA-256 hash of all the current IO values under that policy. To ensure smooth performance, the hash is not re-calculated every time any IO changes. Instead, the hash is immediately marked as “stale” and later re-calculated. This ensures that even with heavy IO modifications the CPU will not be burdened with unnecessary hash calculations.

Optionally a user can “validate” a hash by pressing the Validate button. This will save the current hash in the Validated Hash IO and set the Hash Validated IO to true. If the current hash every diverges from the validated hash, Hash Validated will be set back to false.



15 : Permission Policy Hash GUI

5.4.3 Default Permission Policies

IGX systems come with a default set of permission policies. Some more advanced control systems will deploy with additional policies. See your specific product user manual in these cases.

Develop Policy

The develop policy is intended to be used on IO that are only mutable during development of the product. These IO are obscure IGX parameters that are difficult to understand and potentially dangerous to change but must still be changeable under strict control. Typically, only Pyramid employees change these IO, however sometimes users may change these IO to help repair broken devices in the field under Pyramid’s supervision. Some examples are:

- Physical device calibration parameters
- Device serial numbers and identifiers
- Safety limits for integral safety components
- Settings related to manufacturing testing

This policy is only writable if the system is in the “develop” operation mode.

Protected Policy

The protected policy is the most commonly used policy. It prevents unwanted modifications to the majority of user configurable parameters on a device or control system. This policy write permit has an interlock which will make the policy unwritable if the system is in “medical” mode. All interlock parameter IO will use this policy by default.

In medical or high-risk industrial systems, these are the parameters that are desirable to “lock down” after they have been tuned to a desired configuration. Some examples are:

- The input range of an electrometer or programmable gain amplifier
- Configurable offsets or user scale factors
- Conversion and sampling frequencies for data acquisition systems
- Expression models for converting dose to charge
- High voltage power supply configuration settings
- The majority of interlock configuration systems
- Process controller configurations

Prescription Policy

The prescription policy is a less used, but still vital policy. It is used for prescriptive IO that will change from session to session of a data acquisition or process controller and so should not be restricted from being written when the operational mode is set to “medical”.

These IO will not be writable after the session starts and only become writable again once the session has been reset. If a process controller is not in the ready state or there is an active data acquisition session, the policy's write permit will be revoked. Some examples are:

- The dose or charge prescription on a dose controller
- The control point table in a dose or motion controller
- Some interlock parameters that have been configured to be prescriptive

Process Policy

The process policy have same purpose with prescription policy, IO will not writable after session starts from user perspective (through network protocol) but IGX process may change it during session, typical usage is with control point table.

5.4.4 Policy Hashes for Configuration Control

IGX devices use policy hashes as a way to verify and monitor system configuration. Each IO permission policy generates a unique hash value that includes data from all IO settings within that policy. If any IO within the policy changes, the hash will update accordingly.

This mechanism allows system designers to:

- Monitor IGX configurations to ensure they match expected parameters.
- Prevent high-risk operations from running with incorrect settings.
- Detect unauthorized or accidental modifications to critical IO configurations.

External supervisory software can be used to ensure configuration integrity by actively monitoring the policy hash generated by IGX. This software functions as a verification layer, keeping track of the latest known valid hash and continuously comparing it to the current hash reported by IGX. If a mismatch is detected, it can signal a potential configuration change, either intentional or accidental, which may require intervention before allowing a critical process to proceed. This method ensures that IGX devices operate within predefined parameters and helps prevent errors caused by unnoticed or unauthorized modifications.

Alternatively, instead of maintaining a separate database of valid hashes, the supervisory software can delegate this responsibility to IGX itself. This is done by programmatically sending a validation command, which instructs IGX to store the current hash as the new valid reference. Once this is set, IGX can internally enforce configuration consistency by blocking execution if any parameter within the policy hash changes. This is achieved through an internal interlock, which automatically prevents system actions if the stored hash no longer matches the expected value. This method reduces dependency on external tracking and allows IGX to handle configuration enforcement autonomously.

Both approaches are equally valid, and the choice between them depends on system architecture and operational requirements. An external monitoring approach may be preferable for centralized verification, allowing cross-checking of multiple devices and integration with broader security policies. On the other hand, using IGX's internal validation mechanism simplifies enforcement at the device level, reducing reliance on networked software while maintaining a high level of configuration control. A system designer may even choose to use a combination of the two techniques to create a layer of redundancy. The decision ultimately hinges on factors such as system complexity, security policies, redundancy requirements, and real-time enforcement needs.

5.5 IGX SSH and SFTP Guide

5.5.1 Introduction

IGX devices run the QNX operating system and can be accessed via standard SSH and SFTP protocols. The default login credentials for IGX devices are username: `root` and password: `pyramid`, on some older versions of IGX the default root password is `root`. This guide provides step-by-step instructions on how to access IGX devices via SSH and SFTP using the command prompt in Windows 10 and WinSCP. Additionally, it covers basic QNX console commands and how to start and stop the IGX application.

Secure Shell (SSH) Protocol

The SSH protocol is a secure network protocol used for remote access to computers. It is designed to provide secure communication between two untrusted networks by using encryption to protect the contents of the communication. SSH is widely used by network administrators to manage remote systems and by developers to access remote development environments.

The SSH protocol uses the TCP/IP protocol suite for communication, and by default, uses port 22. The protocol supports several authentication methods, including password-based authentication, public key authentication, and host-based authentication. SSH also supports tunneling of TCP/IP connections, allowing applications to securely communicate over the SSH connection.

The SSH protocol is defined in several RFCs, including RFC 4250 (The Secure Shell (SSH) Protocol Assigned Numbers), RFC 4251 (The Secure Shell (SSH) Protocol Architecture), and RFC 4253 (The Secure Shell (SSH) Transport Layer Protocol).

Secure File Transfer Protocol (SFTP)

The Secure File Transfer Protocol (SFTP) is a secure alternative to the File Transfer Protocol (FTP) for transferring files between computers. SFTP is based on the SSH protocol and uses the same encryption and authentication mechanisms to protect the contents of the communication.

SFTP uses the TCP/IP protocol suite for communication and typically uses port 22, the same port as SSH. SFTP supports several operations, including file upload, file download, and directory listing. SFTP also supports resume of interrupted transfers and can preserve file attributes such as modification time, permissions, and ownership.

The SFTP protocol is defined in several RFCs, including RFC 913 (FTP server extension), RFC 913 (FTP client extension), and RFC 4253 (The Secure Shell (SSH) Transport Layer Protocol).

Further Reading

If you're interested in learning more about the SSH and SFTP protocols, here are some resources to get you started:

- SSH Protocol Specification: IETF RFC2450[5]
- SFTP Protocol Specification: IETF RFC913[6]
- OpenSSH: https://www.openssh.com/
- PuTTY: https://www.chiark.greenend.org.uk/~sgtatham/putty/
- WinSCP: https://winscp.net/eng/docs/start

5.5.2 Accessing IGX Devices via SSH

To access an IGX device via SSH, follow these steps:

1. Open the Command Prompt on Windows 10 by typing `cmd` into the Start menu.
2. In the Command Prompt, type the following command to connect to the IGX device via SSH:

5. https://www.ietf.org/rfc/rfc4250.txt
6. https://www.ietf.org/rfc/rfc913.txt

```
ssh root@<ip_address>
```

Replace `<ip_address>` with the IP address of the IGX device you wish to connect to.

3. When prompted, enter the password for the root user (default password is `pyramid` or `root`).

You should now be connected to the IGX device via SSH and can execute QNX commands.

5.5.3 Accessing IGX Devices via SFTP

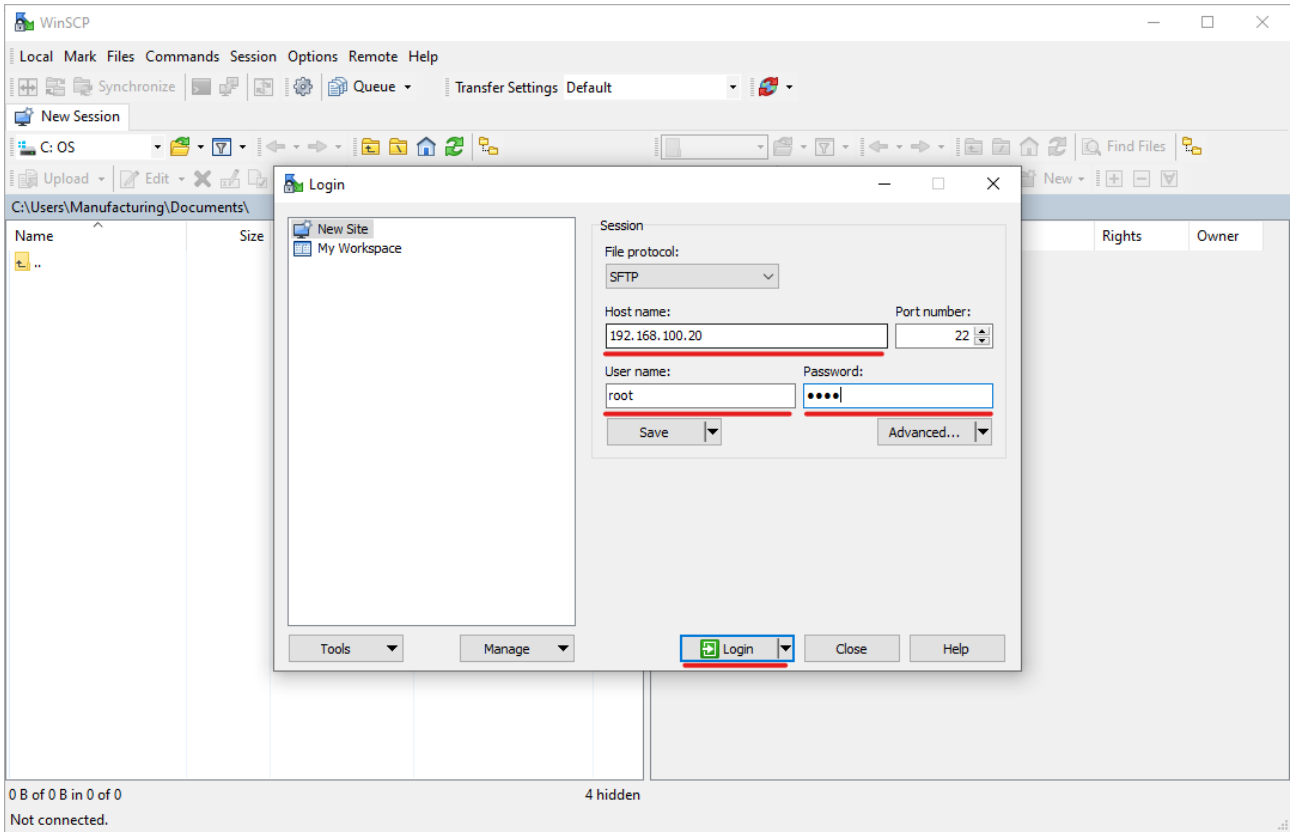
To access an IGX device via SFTP, follow these steps:

1. Download and install WinSCP from the official website: https://winscp.net/eng/download.php
2. Open WinSCP and click on the "New Site" button.
3. In the "New Site" window, enter the following information:

```
File protocol: SFTP
Host name: <ip_address>
Port number: 22
User name: root
Password: pyramid
```

Replace `<ip_address>` with the IP address of the IGX device you wish to connect to.

4. Click the "Save" button and then "Login" to connect to the IGX device via SFTP.
5. If prompted that the SSH key of the device has changed, you may click “Yes” if:
   a. The IP address was recently used by a different device, and you are confident that this is correct.
   b. You do trust that the device at the given IP address is in fact the device you wish to connect to.



16 : WinSCP Example Settings

You should now be able to browse and transfer files to/from the IGX device via WinSCP.

5.5.4 Basic QNX Console Commands

QNX operates in a command-line interface (CLI) environment, and being familiar with these basic commands will allow you to interact with the system effectively. The QNX Console (also called the QNX Shell) is the primary tool for system administration, diagnostics, and troubleshooting. While some tasks, such as navigating files and directories, are similar to other UNIX-like systems.

Command	Description	Example
`ls`	Lists the contents of the current directory.	`ls /root`
`cd <directory>`	Changes the current directory to the specified directory.	`cd /root`
`cat <file>`	Concatenates and prints the content of files to the terminal.	`cat /root/config/system.xml`
`pwd`	Prints the current working directory (the directory you're currently in).	`pwd`
`ifconfig`	Displays network interface configuration information, including IP addresses, network interfaces, etc.	`ifconfig`
`ping <ip>`	Tests network connectivity by sending ICMP echo requests to the specified IP address.	`ping 192.168.0.1`
`netstat`	Displays network statistics, including active network connections, routing tables, and interface stats.	`netstat`
`kill <pid>`	Terminates the process with the specified process ID (pid).	`kill 1234`
`slay <process>`	Terminates a process by name, much like kill, but allows you to use the process name instead of the ID.	`slay igx`
`ps`	Displays a list of currently running processes and their statuses.	`ps`
`tar`	Compresses and decompresses files and directories using the tar archiver.	`tar -cvf archive.tar file`
`top`	Displays real-time system resource usage, such as CPU, memory, and process information.	`top`
`shutdown`	Immediately restarts the operating system. The IGX environment should automatically start if using the default configuration.	`shutdown`

Filesystem Navigation

`ls` and `cd` are fundamental commands for exploring and navigating the filesystem. The `ls` command lists files in the current directory, and `cd` changes the working directory to a specified location. Use the `pwd` command to always know your current location within the file system.

Process Management

`ps` and top provide information about running processes. `ps` lists all active processes and their IDs, while top provides a real-time view of resource usage, which is helpful for monitoring system performance. The kill and slay commands are used to terminate processes by process ID or name, respectively.

Networking Tools

ping and `ifconfig` are essential for network diagnostics. ping helps test network connectivity, and `ifconfig` displays the configuration of network interfaces. `netstat` is another useful tool for reviewing network statistics and active connections.

QNX Documentation

- ifconfig[7] - Manually retrieve and configure IP settings
- route[8] - See the routing table and modify it
- dhclient[9] - Use DHCP to assign an IP address automatically
- netstat[10] - See various network related statistics to debug issues

Identify Your Network Interfaces

1. `dm0` is the primary Ethernet interface on all IGX devices.
2. `dm1` is used for a secondary Ethernet port, if available on the device.

Setting a Temporary Static IP Address

A new IP address can be manually assigned to an IGX device independent of its internal configuration using the `ifconfig` utility. This method only changes the IP address temporarily, it will reset upon reboot, or if IGX restarts and applies its own network settings again.

Run the following command via SSH:

```
ifconfig dm0 192.168.100.20 netmask 255.255.255.0
```

- Replace `192.168.100.20` with your desired IP address.
- The netmask defaults to `255.255.255.0` if unspecified.

Setting a Temporary Dynamic IP Address

You can configure a temporary dynamic IP address using DHCP, allowing the device to request an IP address from a DHCP server (such as a router) on the network. This method is useful when a static IP is not needed or when operating in a network-managed environment.

Run the following command via SSH:

```
dhclient dm0
```

This command broadcasts a DHCP request, and if a DHCP server is available, it will respond with an assigned IP address. However, the IP will not persist after a reboot, and if no DHCP server is detected, the command will fail, leaving the device without an assigned IP.

File Compression

`tar` is the standard tool for working with file archives in QNX. It can be used to compress or extract files and directories, which is especially useful for creating backups or transferring files between systems.

Shutdown and Restart

`shutdown` is used to restart the system immediately. It's important to note that when using IGX, the environment will typically restart automatically if using the default configuration after the operating system has rebooted.

---

7. https://www.qnx.com/developers/docs/7.1/#com.qnx.doc.neutrino.utilities/topic/i/ifconfig.html
8. https://www.qnx.com/developers/docs/7.1/#com.qnx.doc.neutrino.utilities/topic/r/route.html
9. https://www.qnx.com/developers/docs/7.1/#com.qnx.doc.neutrino.utilities/topic/d/dhclient.html
10. https://www.qnx.com/developers/docs/7.1/#com.qnx.doc.neutrino.utilities/topic/n/netstat.html

Additional QNX Resources

To learn more about these commands and explore additional options, please refer to the official QNX documentation at Official QNX Documentation[11]. The documentation covers all available commands, their arguments, and detailed examples, making it an essential resource for anyone working with QNX-based systems.

5.5.5 Starting and Stopping the IGX Application

The IGX application is located at `/root/igx` and can be started and stopped using the following commands:

- To start the IGX application, use the following command:

  ```
  /root/igx
  ```

  This will automatically kill any other instance of IGX that may be running.
- To stop the IGX application, use the following command:

  ```
  hamctrl -stop
  slay igx
  ```

  This will gracefully stop the IGX application and release all resources.

Note: Only one instance of the IGX application can run at a time, so it is important to stop the currently running instance before starting a new one.

5.5.6 Writable System Configuration Files in QNX

IGX devices use a QNX-based filesystem architecture that differs from standard Linux environments in one important way: many core system configuration files are stored on a read-only filesystem image. This design improves system reliability and recoverability, as it ensures that a known-good baseline configuration is always preserved, even in the event of filesystem corruption.

However, this behavior also affects how system files can be modified.

Modifying Protected System Files

Files such as `/etc/passwd`, `/etc/shadow`, and `/etc/ssh/sshd_config` exist on a read-only layer by default. As a result, standard tools (such as `passwd`) will fail if you attempt to modify these files directly.

To make these files editable, you must first create a writable copy at the same path. This is done using the `cp` command:

```
cp /etc/passwd /etc/passwd
cp /etc/shadow /etc/shadow
```

This operation creates a writable overlay version of the file. Once copied, the system will use the writable version instead of the read-only original, allowing normal modification using standard tools.

For example, after copying `/etc/passwd` and `/etc/shadow`, you can safely use:

```
passwd
```

to change user credentials.

Reverting to Default Configuration

If you need to restore the original system defaults, simply remove the writable versions of the files:

```
rm /etc/passwd
```

11. https://www.qnx.com/developers/docs/7.1/#com.qnx.doc.neutrino.user_guide/topic/cmdline.html

```
rm /etc/shadow
```

Once removed, the system will fall back to the original read-only versions.

SSH Configuration

The same mechanism applies to SSH configuration. The SSH daemon configuration file is located at:

```
/etc/ssh/sshd_config
```

To modify it, first create a writable copy:

```
cp /etc/ssh/sshd_config /etc/ssh/sshd_config
```

You can then edit the file as needed, for example to disable password authentication or configure other SSH settings.

5.5.7 Installing IGX Versions with SSH

While the preferred method for installing a new version of IGX is through the web GUI, you can also install it manually using SSH if needed. This can be useful when the web interface is unavailable or if you simply prefer using command-line tools.

Step 1: Transfer the Update File to the Device

Move the IGX update file (`igx_123456789.tar.gz` for example) to the `/root` directory on the device. You can do this using:

- A file transfer tool like WinSCP (for Windows users see above).
- The `scp` command from a local terminal:

  ```
  scp <path-to-update-file.tar.gz> root@device-ip:/root/
  ```

Step 2: Stop IGX Instance

Normally, IGX manages its own update process and is aware when files are being modified. However, since this method updates files without IGX's awareness, it can lead to unexpected behavior. To prevent issues, it's best to stop IGX before proceeding.

Run the following commands via SSH:

```
hamctrl -stop
kill -9 <igx-pid>
```

Explanation

- `hamctrl -stop` disables the safety system that would otherwise automatically restart IGX.
- `kill -9 <igx-pid>` immediately stops the IGX process.
  - This method is not graceful, but it ensures IGX is completely stopped.
  - Since you may be dealing with a faulty version of IGX, relying on a normal shutdown may not be safe or effective.
  - To find the IGX process ID (PID), you can use:

    ```
    ps | grep igx
    ```

Step 3: Unpack and Install IGX

Once the file is on the device, run the following command via SSH to extract and install the update:

```
tar -mxzf <path-to-update-file.tar.gz> -C /root
```

Explanation

- The `-mxzf` arguments are important, they tell the command exactly how to unpack the file.
- The `-C /root` option ensures that the files are extracted directly into the `/root` directory rather than the current working directory.
- The `tar` command will automatically place all the new IGX files in their correct locations.

After extraction, IGX should be ready to run with the newly installed version, you can do this manually or just by power cycling the device.

5.6 IGX System XML File

IGX uses an XML file[12] named `system.xml` to determine its configuration upon startup. This file defines device behavior, settings, and overrides when necessary. Unlike previous generations of Pyramid system XML files, the IGX `system.xml` is minimalist by default. This means:

- Default settings are automatically applied without needing to be explicitly defined in the XML file.
- Modifications are only required if your configuration deviates from the standard setup.
- Unnecessary entries do not need to be included, keeping the file lean and manageable.

The `system.xml` file is stored on the device at

```
/root/config/system.xml
```

You should only edit `system.xml` if you need to:

- Change the device type (e.g., FX4, CM100).
- Enable or disable specific software features.
- Want to add your own custom functionality to the device.

Modifications should be made carefully, as incorrect configurations may cause unexpected behavior or prevent the device from starting properly.

5.6.1 Using the Web GUI to Modify the System XML File

12. https://en.wikipedia.org/wiki/XML

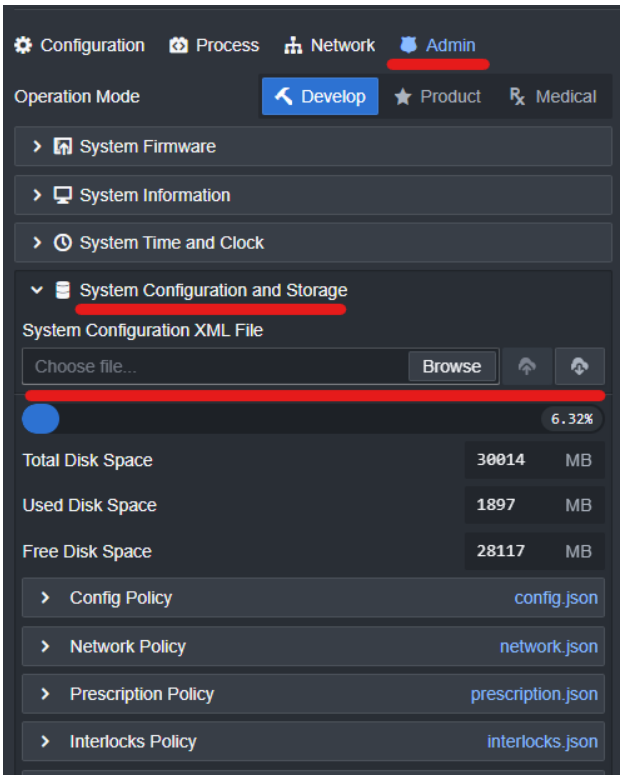
Navigate to the Admin tab, then open the System Configuration and Storage section.

In the System Configuration XML File area, you can:

- Download the current configuration file by clicking the button on the right.
- Upload a modified XML file from your computer to the device.

Typically, you'll want to download the existing XML file first, make your edits locally, and then upload the updated version.

After uploading a new XML file, the device will automatically restart, and your changes will take effect.

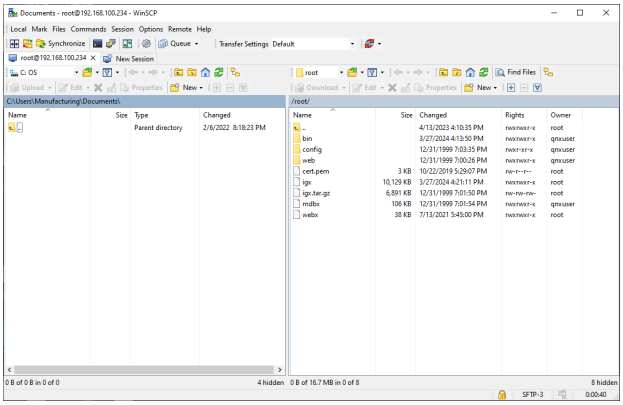


17 : System Configuration and Storage

5.6.2 Using WinSCP to Modify the System XML File

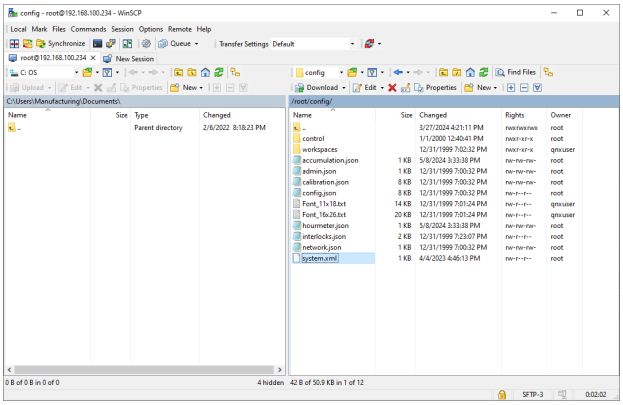
Follow the instructions in the previous chapter for how to start using WinSCP for the first time.

- Login to the IGX device using WinSCP, username `root` and password `root`.
- Upon logging in, you will start in the `/root` directory.
- Navigate to `/root/config` by double-clicking the `config` folder in the file list on the right.



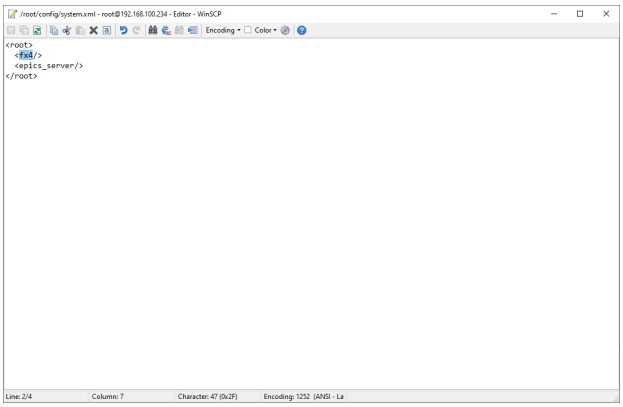
18 : Starting in the /root Directory

- Inside the `/root/config` directory, locate the file named `system.xml`.



19 : The config Directory Files

- Double-click `system.xml` to open it for editing.
- Modify the file as needed.
  - For example, you can change the `<fx4/>` node to specify a different device name.
  - Refer to the device node types in the following section for available options.
- Save the file after making modifications.
- Power cycle the device (turn it off and back on) to apply the changes.



20 : The system XML file.

5.6.3 Recovering from a Misconfiguration

- If the device fails to start due to a misconfigured file, reconnect with WinSCP and revert the changes to restore the original configuration.
- This ensures that you can recover the device without needing reprogram the SD card.

5.6.4 The System XML File Format

The format of the XML file follows standard XML formatting rules. The top-level node must be `<root>`.

```
<root>
  <node name="example" />
</root>
```

The parent child relationship in the XML mimicked in the runtime IGX structure. So, if you define a new node underneath another node, IGX will make that node a child of the parent node.

```
<root>
  <node name="parent" >
    <node name="child" />
  </node>
</root>
```

Fields can be assigned through XML node attributes. Fields allow you to define properties for nodes and tell the system and GUI how it should handle the given node.

```
<root>
  <node name="parent" detail="My custom node that contains a custom XML IO" >
    <analog_io name="my_xml_io" label="My XML IO" units="pA" value="0.1" />
  </node>
</root>
```

Field Types (XML Attributes)

The following is all the possible field types you can define on a node. The most commonly used fields are, name, label, value, readonly, units, and format.

Field	Type	Required	Notes
name	string	Yes	The name of the node must be unique among sibling nodes. Must not be the same as any field name. For example, name cannot equal “name” or “label”.
alias	string	No	A unique identifier for the IO, used by EPICS server for PV name.
label	string	No	Human friendly name for the node, used by GUIs.
detail	string	No	Brief description of the node, used by the GUI and automated report generation.
hidden	bool	No	True if this node is too complicated for the average user and should be hidden by the GUI.
color	string	No	Name of a color to assign to this node. Possible values are “red”, “blue”, “green”, “orange”, and “gray”.
icon	string	No	Name of an icon to use for this node.
value	any	No	The value for this node, if it is an I/O type.
readonly	bool	No	True if this node’s value field should only be read only. This will effect the API permissions and also the GUI’s display widget.
units	string	No	The unit of measure for the value field. (“V”, “A”, “mA”, etc.) Used by the GUI and APIs to help clarify to the user what numerical value means.
format	string	No	The formatting specifier of how to format the value number.
store	string	No	Set to “hourmeter” to save a rapidly changing readonly IO. Set to “config” to save and restore a seldom changed writable IO.

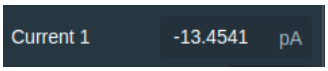
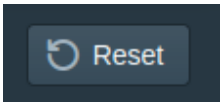
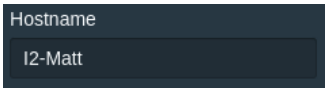
Generic Node Types

These nodes can be used anywhere in the XML file, as parents or children of any other node.

Name	Description
`<node/>`	Just a simple node by itself, no special behavior or functions. Useful for organizing other nodes into groups in a logical way.

Generic IO Node Types

These nodes allow the user to add new IO to the device. This can be very helpful when creating your own control systems programmatically or when using the IGX expression engine.

Name	Description	GUI Element
`<analog_io/>`	A double floating-point IO, used for any numerical type data you may need. If readonly is false, the IO will be an input field in the GUI. If true, it will be displayed as a readback field.	
`<digital_io/>`	A Boolean IO, used for any digital signals, flags, checkboxes, or status LEDs. Setting it to readonly will make it an LED.	
`<button_io/>`	Same as `<digital_io/>` except, the GUI element is now a button that will set the digital to true when the button is pressed.	
`<string_io/>`	A string IO, for any length string. Useful for configuration values, states, IDs, and file names.	

Root Node Types

These optional nodes can be added or removed from the root node to extend the functionality of the device or remove unused functionality.

XML Tag Node	Description
`<epics_server/>`	An EPICS server which is able to publish device IO over the network.

For example, you can create an XML file like the following to the EPICS server node.

```
<root>
  <device/>
  <epics_server/>
</root>
```

Replace the `<device/>` node with whatever device node is used by the specific device. See below for more information on device nodes.

Device Node Types

The system.xml file in IGX determines what kind of device the system should operate as when it starts up. This configuration is critical for ensuring the correct software is loaded for the corresponding hardware.

Important Considerations:

- Mismatching hardware and software can lead to unexpected behavior and system instability.
- Incorrect settings may result in **undefined behavior**, potentially affecting device functionality.
- Changes to this file should be made with extreme caution and only by users who understand the system requirements.

Device	XML Tag Node	Description
FX4	`<fx4/>`	4-channel electrometer
IX256-F2	`<ix256/>` or `<ix256_f2/>`	256-channel electrometer with a dual channel mezzanine option

Device	XML Tag Node	Description
IX256-F1	`<ix256_f1/>`	256-channel electrometer with a single channel mezzanine option
MX1	`<mx1/>`	8-channel data acquisition system with relays
M1	`<m1/>`	8-channel data acquisition system
TX2	`<tx2/>`	2-channel gaussmeter
T1	`<t1/>`	Single channel gaussmeter
KX8	`<kx8/>`	8-channel safety relay controller
RCI	`<rci/>`	Nozzle room controller
ACI	`<aci/>`	Accelerator controller interface
ICI	`<ici/>`	Interlock controller interface
HC80	`<hc80/>`	Helium gas flow controller.

6 IGX Expression Language Guide

6.1 Introduction to ExprTk Language

ExprTk is a powerful and flexible mathematical expression parsing and evaluation library for C++. It allows users to create, manipulate, and evaluate complex mathematical expressions through a simple and intuitive language. In this guide, we'll provide an overview of how you can use the ExprTk language to create expressions, incorporate variables, and use built-in functions and operators.

The ExprTk website has lots of simple and complex examples to help insipre and teach you how the language works and can be found here: C++ Mathematical Expression Library (ExprTk) - By Arash Partow[13]

The ExprTk source code can be helpful for some users and can be found on GitHub under the MIT license: Arash Partow ExprTk (github.com)[14].

6.2 Arithmetic Operators

Expressions can be built using a variety of built in operators. The following is a list of the mathematical and logical operators. These operators evaluate into a single numerical value.

Basic Arithmetic Operators

Operator	Description
`+`	Addition: Adds two values together.
`-`	Subtraction: Subtracts the second value from the first value.
`*`	Multiplication: Multiplies two values together.
`/`	Division: Divides the first value by the second value.
`%`	Modulus: Returns the remainder of the division of the first value by the second value.
`^`	Exponentiation: Raises the first value to the power of the second value.

Equalities and Inequalities Operators

The following are operators for checking equalities or inequalities. The results of these operators will always be 0 for false and 1 for true.

Operator	Description
`==`	Equality: Checks if two values are equal.
`!=` or `<>`	Inequality: Checks if two values are not equal.
`<`	Less Than: Checks if the first value is less than the second value.
`<=`	Less Than or Equal To: Checks if the first value is less than or equal to the second value.
`>`	Greater Than: Checks if the first value is greater than the second value.
`>=`	Greater Than or Equal To: Checks if the first value is greater than or equal to the second value.

13. https://www.partow.net/programming/exprtk/
14. https://github.com/ArashPartow/exprtk/tree/master

Logical Operators

The following are logical operators. The results of these operators will always be 0 for false and 1 for true.

Operator	Description
`true`	True state or any value other than zero, typically 1.
`false`	False state, value of exactly zero.
`and`	Logical AND, True only if x and y are both true. Example: `x and y`
`&`	Similar to AND but with left to right expression short circuiting optimization.
`nand`	Logical NAND, True only if either x or y is false. Example: `x nand y`
`mand(x, y, ...)`	Multi-input logical AND, True only if all inputs are true. Left to right short-circuiting of expressions. Example: `mand(x > y, z < w, u or v, w and x)`
`or`	Logical OR, True if either x or y is true. Example: `x or y`
`|`	Similar to OR but with left to right expression short circuiting optimization.
`nor`	Logical NOR, True only if the result of x or y is false. Example: `x nor y`
`mor(x, y, ...)`	Multi-input logical OR, True if at least of the inputs are true. Left to right short-circuiting of expressions. Example: `mor(x > y, z < w, u or v, w and x)`
`xor`	Logical XOR, True only if the logical states of x and y differ. Example: `x xor y`
`xnor`	Logical XNOR, True iff the biconditional of x and y is satisfied. Example: `x xnor y`
`not(x)`	Logical NOT, Negate the logical sense of the input.

Variable Assignment Operators

In addition to these operators, there are also a selection of assignment operators to pick from. Unlike the operators above, these will assign the evaluated value to a variable on the lefthand side of the operator.

Operator	Description
`:=`	Assignment: Assigns the value on the right to the variable on the left.
`+=`	Add and Assign: Adds the value on the right to the variable on the left, and assigns the result to the variable on the left.
`-=`	Subtract and Assign: Subtracts the value on the right from the variable on the left, and assigns the result to the variable on the left.
`*=`	Multiply and Assign: Multiplies the value on the right with the variable on the left and assigns the result to the variable on the left.
`/=`	Divide and Assign: Divides the variable on the left by the value on the right and assigns the result to the variable on the left.
`%=`	Modulus and Assign: Takes the modulus of the variable on the left by the value on the right and assigns the result to the variable on the left.

6.3 Built-in Functions

ExprTk comes with a wide range of built-in functions and operators to help you create more complex expressions. These include trigonometric functions (e.g., `sin`, `cos`, `tan`), logarithmic functions (e.g., `log`, `log10`, `exp`), and many others. To use a function, simply write its name followed by its arguments in parentheses. For example: `sin(x) * cos(y)`. Some functions accept a variable number of arguments. These functions will be defined with an argument list ending in `...` to show that more arguments can be added.

ExprTk library supports a wide range of built-in functions for various mathematical operations and calculations. Here's a table listing some of the commonly used built-in functions in ExprTk along with a brief description of their behavior:

General Purpose Functions

Function	Description
`equal(x, y)`	Equality test between x and y using normalized epsilon.
`not_equal(x, y)`	Not-equal test between x and y using normalized epsilon.
`abs(x)`	Returns the absolute value of x.
`sgn(x)`	Sign of x, -1 where x < 0, +1 where x > 0, else zero.
`sqrt(x)`	Returns the square root of x.
`root(x, n)`	Nth-Root of x. where n is a positive integer.
`pow(x, y)`	Raises x to the power of y (x^y).
`exp(x)`	Returns the exponential function of x (e^x).
`expm1(x)`	Returns e to the power of x minus 1, where x is very small.
`log(x)`	Returns the natural logarithm (base e) of x.
`log10(x)`	Returns the logarithm (base 10) of x.
`log2(x)`	Returns the logarithm (base 2) of x.
`logn(x, n)`	Base N logarithm of x. where n is a positive integer.
`ceil(x)`	Smallest integer that is greater than or equal to x.
`floor(x)`	Largest integer that is less than or equal to x.
`round(x)`	Round x to the nearest integer.
`roundn(x, n)`	Round x to n decimal places where n > 0 and is an integer. Example: `roundn(1.2345678, 4) == 1.2346`
`frac(x)`	Fractional portion of x.
`trunc(x)`	Integer portion of x.
`max(x, y, ...)`	Largest value of all the inputs.
`min(x, y, ...)`	Smallest value of all the inputs.
`clamp(r0, x, r1)`	Clamp x in range between r0 and r1, where r0 < r1.

Function	Description
`sum(x, y, ...)`	Sum of all the inputs.
`mul(x, y, ...)`	Product of all the inputs.
`avg(x, y, ...)`	Average of all the inputs.
`hypot(x, y)`	Hypotenuse of x and y. `hypot(x, y) = sqrt(x*x + y*y)`
`erf(x)`	Error function of x.
`erfc(x)`	Complimentary error function of x.
`ncdf(x)`	Normal cumulative distribution function.
`swap(x, y)`	Swap the values of the variables x and y and return the current value of y.

Trigonometry Functions

Function	Description
`acos(x)`	Returns the arc cosine of x in radians.
`asin(x)`	Returns the arc sine of x in radians.
`atan(x)`	Returns the arc tangent of x in radians.
`atan2(x, y)`	Returns the arc tangent of y/x in radians.
`cos(x)`	Returns the cosine of x, where x is in radians.
`cosh(x)`	Returns the hyperbolic cosine of x.
`sin(x)`	Returns the sine of x, where x is in radians.
`sinh(x)`	Returns the hyperbolic sine of x.
`tan(x)`	Returns the tangent of x, where x is in radians.
`tanh(x)`	Returns the hyperbolic tangent of x.

6.4 Conditional Expressions

If statements have multiple possible syntax that can be used.

The functional form `if (x, y, z)`, uses three arguments, `x`, `y`, and `z`, where if `x` is true, `y` is returned and otherwise `z` is returned. For example:

```
if (x > 10, 5, 6)
```

If x is greater than 10, the expression resolves to 5 and otherwise it resolves to 6.

A more conventional form, where the branching expressions are defined after the if statement, with optional curly brackets, can also be used. For example:

```
if (x) z;

if (x) { z; }
```

If, else, and else if statements can also be used. For example:

```
if (condition ...)
{
  ...
}
else if (alternate condition ...)
{
  ...
}
else
{
  ...
}
```

6.5 Common Variables

IGX adds its own special variables to all expressions in order to enhance the base functionality. These special variables are always available and update automatically every time the expression is evaluated.

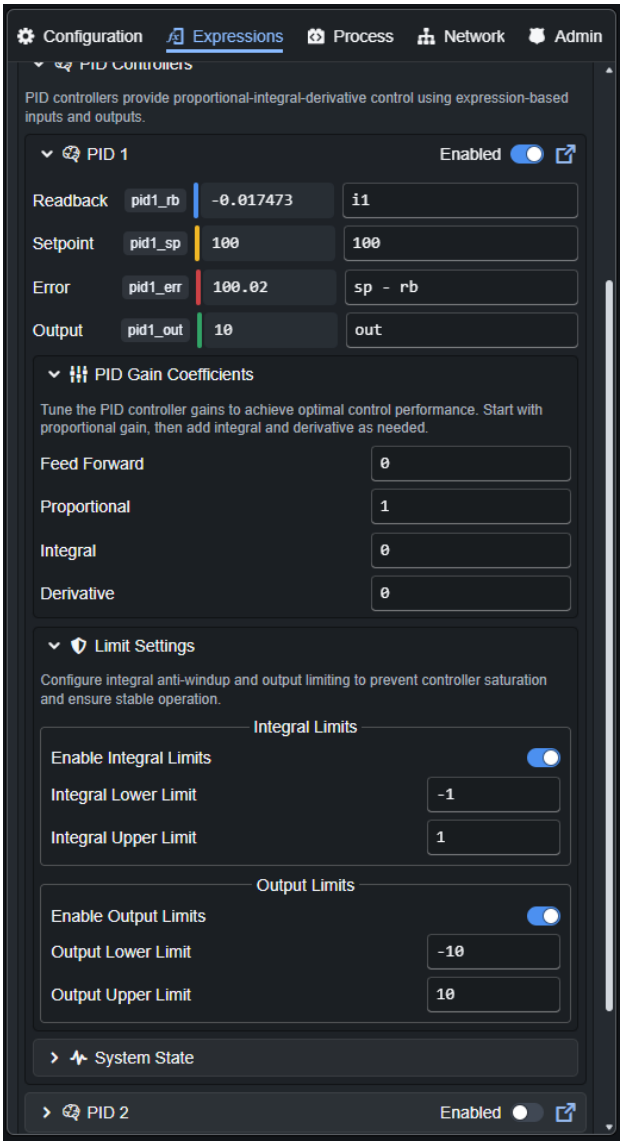
Variable	Description
`rt`	Running timer in seconds for how long the device has been running.
`prev`	The previous value of the expression. Allows for simple recursive expressions or simple IIR filters.

7 IGX PID Expression Controllers

The User PID Controller is a closed-loop controller available on all IGX devices. It compares a live process variable IO (readback) to a desired setpoint IO, computes the error IO, and drives a control output IO using proportional–integral–derivative (PID) behavior with optional feed-forward. Inputs and the final output path use the same expression system as the rest of IGX, so you can reference device signals, math, and calibrations without custom firmware.

At its core, the controller:

- Evaluates readback and setpoint expressions each control cycle (1kHz by default).
- Computes error from an expression (default: setpoint minus readback).
- Builds a raw PID value from feed-forward, proportional, integral, and derivative terms.
- Applies optional integral anti-windup and output limiting.
- Publishes the result through an output expression (default: pass-through of the computed PID output).



21 : IGX PID Controllers

This design is suited for temperature, pressure, flow, beam parameters, or any loop where the process variable and actuator command can be represented as numeric expressions.

Where to Find It

Each device exposes at least four independent controllers under the Expressions tab, typically named PID 1 through PID 4. Together they appear alongside user expressions and shared runtime symbols in the web interface. A dedicated PID Plots tab provides time-series views of readback vs setpoint, controller output, error, and integral state.

7.1 Enabling and Disabling

Enabled turns the entire PID calculation on or off.

- When enabled, the controller runs every update: expressions are refreshed, PID terms are updated, and the output is driven accordingly.

- When disabled, the controller does not advance its internal calculation for that cycle; the output effectively holds at its last computed state. Use this to freeze the loop during maintenance or when downstream equipment must not be driven.

Configuration fields marked as protected may require appropriate operator permissions.

7.2 Expression-Based Signals

Four expression-driven values define the loop. Each can use the full IGX expression language and can reference other symbols on the device.

Readback

Readback is the process variable: the quantity you want to track toward the setpoint (e.g. a sensor, filtered value, or derived quantity).

Setpoint

Setpoint is the desired target for the readback, in the same units as the readback when you use the default error definition.

Error

Error is computed from an expression. The default is: `sp - rb` where, inside this expression only:

Symbol	Meaning
`sp`	Current setpoint (numeric value after the setpoint expression is evaluated)
`rb`	Current readback (numeric value after the readback expression is evaluated)

You can replace the default with custom error definitions (for example signed vs magnitude-only behavior), or combine multiple inputs, as long as the expression produces a single numeric error signal the PID acts on.

Output

Output is how the final command leaves the block. The PID core first computes an internal raw value (feedforward + P + I + D terms, with optional clipping). That raw value is exposed to the output expression as out.

Symbols available in the output expression:

Symbol	Meaning
`sp`	Current setpoint
`rb`	Current readback
`out`	Raw PID result after gains, integral, derivative, and output limiting

The default output expression is `out`, meaning the raw PID result is published unchanged. You may scale, offset, or otherwise modify this expression if your actuator or downstream logic requires it.

Each PID instance also registers short symbols for use in other expressions (for example readback and setpoint symbols derived from the block name such as `pid1_rb`, `pid1_sp`, and similarly for error and output, exact prefixes match the vertex name configured for that controller).

7.3 PID Gain Coefficients

The controller implements a discrete-time PID with the following configuration parameters:

Parameter	Role
Feed Forward (kff)	Adds `kff * setpoint` to the output. Useful when the setpoint predicts a known steady-state command (e.g. open-loop schedule plus PID trim).
Proportional (kP)	Adds `kP * error`. Primary “stiffness” of the loop.
Integral (kI)	Accumulates `kI * error * Δt` and adds the accumulated integral state to the output. Removes steady state offset when tuned appropriately.
Derivative (kD)	Adds `kD *` a smoothed rate of change of error (`d(error)/dt`), to damp oscillations. The implementation applies filtering to reduce sensitivity to noise.

Practical tuning (typical workflow):

1. Start with **integral and derivative at zero**, modest **proportional** gain, and **feed-forward** off unless you already have a model.
2. Increase **kP** until the response is acceptably fast without sustained oscillation.
3. Add **kI** in small steps to remove offset; watch for overshoot or slow ringing.
4. Add **kD** only if you need extra damping; too much **kD** can amplify noise.

7.4 Limit Settings

Integral limits (anti-windup)

When Enable Integral Limits is on, the internal integrator only accumulates between Integral Lower Limit and Integral Upper Limit. This prevents integral windup when the actuator saturates or when the loop cannot reach the setpoint for a long time.

When off, the integral accumulates without clipping (use with care).

Output limits

When Enable Output Limits is on, the raw PID sum is clamped to Output Lower Limit and Output Upper Limit before being assigned to out for the output expression.

When off, no clamping is applied at this stage (the output expression may still enforce its own bounds if desired).

7.5 Wiring PID Output to Hardware

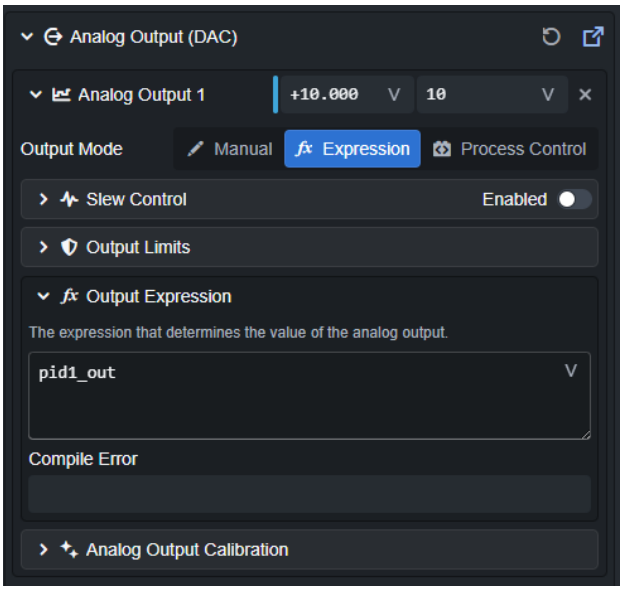
To calculate the PID output and expose the IO to the public API, configure only the PID controller.

To link the PID output to another part of the IGX control system, such as a DAC output, configure the hardware to be driven by an expression that uses the PID output symbol.

For example, the screenshot on the right shows a DAC analog output channel set to expression-driven with the expression pid1_out.

Here, I use the PID output directly, but you can apply scale factors or other control elements.

This same thing can be done in any of IGX's expression driven outputs. Including analog, digital, and PWM outputs.



22 : Simple DAC Expression with PID Output

7.6 System State (Monitoring and Diagnostics)

These read-only values help verify timing and behavior while tuning:

IO	Purpose
Delta Time	Effective time step between PID updates (seconds), used for integral and derivative. Displayed value is a smoothed representation of the measured interval.
Error (previous)	Error from the prior cycle (internal storage for derivative computation).
Derivative	Filtered derivative term driving the **kD** contribution.
Integral	Accumulated integral state driving the **kI** contribution (subject to integral limits when enabled).

The PID Plots tab overlays readback and setpoint, shows output, error, and integral over time for intuitive loop debugging.

7.7 Reset State Button

The Reset State button clears the PID internal memory:

- Integral and derivative states are zeroed.
- Previous-error storage is cleared.
- The raw PID output and the published output value are forced to zero (after reset, the next enabled cycle will rebuild state from fresh expressions).

Use Reset State after large setpoint changes, after taking a process offline, or when you need a known-zero integrator before restarting a run. It does not by itself change your readback / setpoint / error / output expressions, only the controller's internal state and immediate output values.

7.8 Practical Considerations

- **Generic Across Devices:** The same controller block exists on every IGX platform; wiring is done entirely through expressions and downstream IO, not per-product firmware variants.

- **Unit Consistency:** Readback, setpoint, and error should use compatible units unless you deliberately scale in the error or output expression.
- **Disable before reconfiguration:** When changing gains or limits on a live process, consider disabling the controller momentarily or resetting integrator state after changes to avoid transients from stale integral values.
- **Output expression:** Treat `out` as the post-limit PID command; use the output expression for final shaping so hardware never sees an unclamped value if you rely on output limits.

8 IGX Notification System

The IGX system includes a built-in notification system that provides real-time updates to users about important events, status changes, or issues within the system. Notifications help keep users informed and allow for quick responses to potential problems or system events.

Each notification includes the following components:

- **Timestamp**: The time the notification occurred, based on the device's system time.
- **Message**: A string of text that provides details about the event. This message can include optional Markdown[15] formatting for better readability and organization.
- **Category**: This helps categorize the notification and indicates its level of importance. The categories are:
  - **Info**: General information about system events.
  - **Fault**: Alerts related to system errors or failures.
  - **Warning**: Notifications about potential issues that should be addressed.
  - **Success**: Confirms successful operations or actions.
  - **Important**: Highlights critical notifications that require immediate attention.
- **Read/Unread Flag**: A Boolean value indicating whether the notification has been acknowledged (read) or is still pending (unread).

8.1 Notification GUI

Notifications can always be accessed through the web GUI by clicking the notification button located in the top right corner of the screen. This button also displays the number of unread notifications.

In the notifications panel, you can:

- **Mark all as read**: This will update all unread notifications to read status.
- **Clear all**: This will remove all notifications from the panel and delete them from the database.

8.2 Interlock Notifications

Each interlock on an IGX device can be configured to generate a notification whenever it enters a fault state. These notifications provide immediate feedback on system issues, helping users quickly identify and address faults, but can also be disabled when no longer useful.

When an interlock notification is triggered, it will:

- Include a direct link to the interlock in the notification message for easy access.
- Provide details on what conditions caused the fault, helping with troubleshooting.
- Always be categorized as "fault", ensuring it is clearly identified as a critical issue.

These notifications ensure that users are promptly alerted to interlock faults, allowing for faster diagnosis and resolution of potential safety or operational concerns.

15. https://en.wikipedia.org/wiki/Markdown

9 IGX Using Interlocks and Permits

This guide provides an overview of the IGX interlock and permit system, a critical safety feature designed to ensure the correct and safe operation of the IGX system.

Software-based interlocks are safety mechanisms used in industrial and controlled environments to prevent accidents, ensure proper operation, and maintain the integrity of systems and processes. These interlocks are programmed into the IGX system to control the flow of operations based on predefined safety conditions.

- **How They Work**: Unlike hardware-based interlocks, which rely on physical components like switches or relays, software-based interlocks are managed by a microprocessor running programmable logic. This software evaluates inputs from sensors or devices and makes decisions based on the conditions you define.
- **Advantages**: Software interlocks are often more cost-effective and easier to maintain than hardware-based systems. However, they typically complement rather than replace hardware-based interlocks, as certain critical safety functions still require physical components to meet stringent safety standards.

In real-world operations, interlocks ensure that specific actions or conditions occur only when it is safe to do so. For example, an interlock might prevent the operation of a high-voltage system unless all safety checks have been satisfied. These safety mechanisms help prevent accidents and maintain the overall integrity of the system.

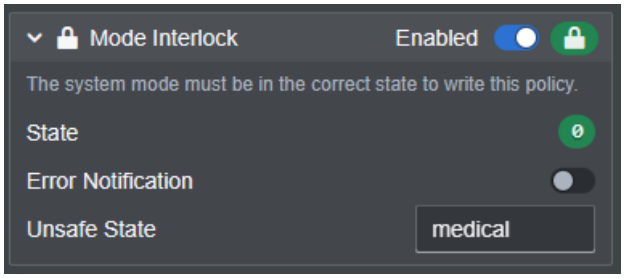
Key Features of the IGX Interlock and Permit System

- **Configurable Interlock Parameters**: You can easily configure and adjust interlock parameters through a user-friendly graphical user interface (GUI). The interface includes descriptive text, making it easier for users to understand and adjust the settings without needing extensive technical knowledge.
- **Adjustable Condition Parameters**: The system provides the ability to adjust the parameters or limits that define when interlocks are triggered. This ensures that the system can be fine-tuned to meet the specific needs and safety requirements of different operational environments.
- **Enable/Disable Interlocks**: The system allows you to enable or disable interlocks, providing the flexibility to override interlocks temporarily or permanently when needed. This feature is useful when performing maintenance, testing, or troubleshooting, where temporary bypassing of certain safety checks may be required.
- **Latching Interlocks**: Some interlocks can be set to latch, meaning they will remain in an error state until explicitly disabled and re-enabled. This ensures that once an interlock condition is triggered, the system will not resume operation until the issue has been addressed and verified, preventing unintentional or unsafe restarts.
- **Permit Creation**: Multiple interlocks can be grouped together into a single permit. This enables you to control access to protected outputs (such as enabling high-voltage systems or radiation sources) only when all required conditions are met.

9.1 Interlock IO

IGX interlocks are IO safety mechanisms that monitor specific conditions within the system. They do not take action on their own; instead, they simply report their state as an integer value. Interlocks ensure that certain conditions are met before any actions can be performed, but they require a permit or additional software to prevent actions or trigger a response.

Interlocks feed into the permit system, which is responsible for controlling the system's actions. Only when all interlock conditions are satisfied can the corresponding permit be *granted*. Then the system allows or prevents actions, such as enabling radiation sources or activating sensitive processes.



This interlock detects when the IGX system is in medical mode and prevents otherwise unsafe actions from occurring.

Through the GUI, you can enable, disable, and configure interlocks to define the conditions that must be met before a permit is granted.

Each IGX interlock is represented by an integer value that corresponds to a specific state or condition of the interlock. The values are as follows:

Value	State	Meaning
`0`		The interlock is in a normal state, with no errors or warnings.
`1`		The interlock is in a faulted state, indicating that the fault condition has been triggered.
`2`		The interlock is in a warning state, meaning that a warning condition has been met. Not all interlocks have a warning condition enabled.

9.1.1 Global Interlock Parameter IO

These interlock parameters can be applied to any interlock in IGX. Some of these parameters may be hidden to the user if they are inappropriate for a given application.

- **Enabled**: Turns the interlock on or off. When disabled, the interlock is always in the **OK (0)** state, meaning it won't trigger any faults or warnings.
- **Latching**: When enabled, the interlock will remain in the faulted (1) state until it is manually disabled. This ensures that once a fault occurs, it cannot be cleared automatically, helping to prevent unintentional resets during critical situations.
- **Condition Time**: Defines the amount of time the faulting condition must persist before triggering a fault. This parameter introduces a time-based leniency or hysteresis, preventing immediate or transient changes from causing unnecessary faults.
- **Enable Notification**: Allows notifications to be posted when the interlock faults. This feature helps operators stay informed about when specific interlocks trigger, providing details about the fault conditions to improve monitoring and response time.

9.1.2 Standardized Interlock Types

The IGX framework includes several pre-made interlock types, each designed to ensure reliable and predictable condition checking. These interlocks are robust, well-tested, and help maintain safety by monitoring various operational parameters.

These interlock parameters will be automatically converted into the same units as the IO being checked against, ensuring the interlock is relevant to the specific measurement.

Range Interlocks

Range interlocks check if a given numeric IO is within predefined limits. These limits are optional, and if a parameter is omitted, it is not considered in the condition check.

- **Upper Limit**: The value above which an error is triggered.
- **Lower Limit**: The value below which an error is triggered.

- **Upper Warning**: The value above which a warning is triggered.
- **Lower Warning**: The value below which a warning is triggered.

For example, if you're monitoring a temperature sensor with a range of 0°C to 100°C, you would set the following range interlock parameters:

- **Upper Limit**: 100°C, if exceeded, trigger a fault.
- **Lower Limit**: 0°C, if below, trigger a fault.

Tolerance Interlock

Tolerance interlocks compare a given numeric IO to a command IO to see if they fall within an acceptable tolerance range. The tolerance is defined as a percentage of the command value, allowing the system to detect discrepancies between expected and actual readings.

- **Minimum**: The minimum value required for the command IO to enable the check. Typically, in the same units of the IO being monitored.
- **Tolerance Limit**: The maximum acceptable percentage error before triggering an error.
- **Tolerance Warning**: The percentage error that, if exceeded, will trigger a warning.
- **Absolute:** When enabled the absolute value of the IO will be used in the condition checks.

For example, if you're checking the actual output of a power supply against a commanded value, you might set the tolerance parameters as follows:

- **Minimum**: 1 Volt, the commanded voltage must be at least 1 Volt before checking tolerance.
- **Tolerance Limit**: 5%, if the actual output deviates from the commanded value by more than 5%, trigger a fault.

Digital State Interlock

Digital State Interlocks are straightforward interlocks that monitor the state of a corresponding digital IO (either true or false). These interlocks trigger a fault based on the state of the IO, ensuring that the system behaves according to specific on/off or true/false conditions.

- **Fault Polarity:** When enabled (true), the interlock will trigger a fault when the corresponding IO is in the true state. If the fault polarity is set to false, the interlock will trigger a fault when the IO is in the false state.

Imagine a digital state interlock that monitors the "Power On" status of a system.

- **Fault Polarity**: True, the interlock will fault when the "Power On" IO is true (indicating the system is powered on). This could be used to prevent certain actions or processes when the system is powered on, ensuring that no actions are taken until the system is safely powered off.

Digital Match Interlock

The digital match interlock, sometime also referred to as an “agreement” interlock, compares two different digital IO values and ensures they either match or are always inverted from each other, based on how the interlock is configured. This interlock is commonly used to verify the consistency between a command signal and a readback signal, ensuring the system behaves as expected.

Some examples of use cases would be:

1. Ensuring that when a relay is commanded to open, a corresponding readback of the relay state matches the command. If the relay command and readback do not match, a fault is triggered.
2. Checking that when a solenoid is energized, a corresponding limit switch is triggered as expected. This helps verify that the mechanical system responds correctly to electrical commands.

You can configure the condition time parameter to introduce some lenience, especially in systems involving mechanical components that may take time to respond to commands. This helps prevent unnecessary faults due to delayed mechanical responses, providing enough time for systems like relays or limit switches to react.

The digital match interlock includes one user-definable parameter:

- **Inverted**: When set (true), the two digital IO must always be inverted (if one is true, the other must be false) to avoid triggering a fault. When unset (false), the two digital IO must match in value (both true or both false) to avoid triggering a fault.

State Interlock

State interlocks are used to compare a string-based state IO against predefined safe and unsafe states. These interlocks are typically used at a higher level to integrate and manage lower-level interlocks, ensuring that the system remains within a defined operational state.

- **Safe State**: If this value is set (not empty), the interlock will fault if the state IO is in any value other than the defined safe state.
- **Unsafe State**: If this value is set (not empty), the interlock will fault if the state IO matches the defined unsafe state.

Both safe and unsafe states can be simultaneously defined, but this is typically redundant and unnecessary.

Imagine you have a device state IO that tracks the operational status of a machine. The safe state might be “operational,” and an unsafe state could be “error.”

If you wanted to define the interlock using the safe sate:

- **Safe State**: "operational", the interlock will not trigger any faults as long as the device is in the "operational" state.

If instead you wanted to just define the interlock to fault in the “error” state.

- **Unsafe State**: "error", if the device enters the "error" state, the interlock will immediately trigger a fault, indicating that the machine is in an unsafe condition and should not continue operation.

Watchdog Interlock

The Watchdog Interlock is designed to detect faults caused by a lack of activity over a certain period of time. It monitors the system for inactivity, and if no activity is detected within a specified time frame, it triggers a fault.

This interlock uses a monotonically stable timer (which does not account for time zone changes or daylight savings) to track the last recorded activity. The timer resets when the interlock is "tickled" (i.e., it is refreshed by a specified action). The exact condition that triggers the reset is determined by the specific implementation of the interlock and cannot be modified by the user.

However, the user can define two important parameters for the Watchdog Interlock:

- **Time Limit**: The maximum time period allowed for activity. If this time is exceeded without activity, a fault will be triggered.
- **Time Warning**: The maximum time period allowed for activity before a warning is issued. If the time limit is approached, the system will issue a warning before triggering a fault.

Watchdog Counter Interlock

The Watchdog Counter Interlock is a specialized version of the Watchdog Interlock, intended specifically for use with external software systems. It is designed to monitor a counter that is regularly incremented by external or supervisory software. This interlock ensures that the external software is running correctly and that the network can handle timely communication.

Due to the explicit software-oriented nature of the Watchdog Counter Interlock, it is typically only enabled by users who intend to write or develop their own custom software to interact with the IGX device. This interlock is particularly useful in environments where external software is responsible for maintaining the health and communication of the system, such as in industrial automation or custom control systems.

Like the Watchdog Interlock, the Watchdog Counter Interlock has the same parameters for Time Limit and Time Warning. Additionally, it includes:

- **Counter**: This is a value that the external software increments by 1 each time an activity check is passed. Each time the counter value is successfully incremented, the watchdog is "tickled," and the timer resets, ensuring that the system continues to function safely.
- **Overflow Limit**: The counter has an upper limit, defined by the Overflow Limit. Once the counter reaches this value, it is expected to reset to 0. The external system should not increment the counter beyond this limit. If the counter reaches the Overflow Limit, it should rollover back to 0 rather than continue to increase beyond the predefined threshold.

9.2 Permit IO

Permits in the IGX system are responsible for checking the status of their associated child interlocks and reporting their combined state. Each permit will have a short description in its corresponding GUI section, outlining the specific action it enables or prevents. Permits themselves do not have any user configurable parameters; their behavior is entirely defined by their interlocks and those interlock parameters.

A permit can be in one of two states: *granted* or *revoked*.

- **Granted Permit:** This means that all associated interlocks are in a safe or warning state, and the system is allowed to proceed with the intended operation. A granted permit has a Boolean value of true.
- **Revoked Permit:** This means that at least one interlock has entered an error state, and the system should not proceed with the operation. A revoked permit has a Boolean value of false.

Value	State	Meaning
`true`		All interlocks are OK or in warning, allowing the operation to proceed.
`false`		At least one interlock is in error, preventing the operation from continuing.

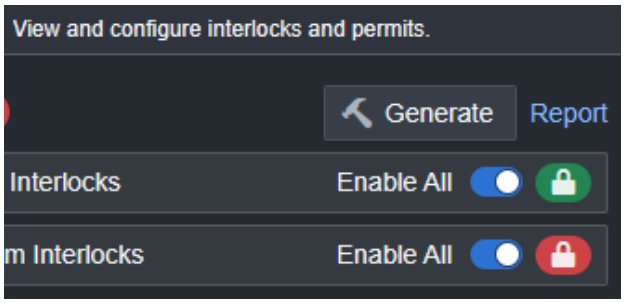
Imagine a permit that controls the operation of a high-voltage system. This permit would be granted if all interlocks related to safety (e.g., "temperature", "voltage", and "safety status") are in an acceptable state (OK or warning). If any of these interlocks enter an error state (e.g., "temperature" exceeds safe limits), the permit would be revoked, stopping the operation of the high-voltage system to ensure safety.

9.3 Reporting and Auditing

It is critical to system safety to regularly conduct interlock parameter audits to ensure that the configuration is still meeting the requirements. IGX includes a system for generating interlock reports in the form of a CSV file containing all the relevant fields.

IGX will have a nested system of interlock managers which each have their own scope that they can report on. In order to generate a report of the global interlock scope, navigate to the following URL `https://<ip>/io/interlocks`.

To generate an interlock report, use the following steps.



23 : Interlock report interface

First click the “Generate” button to create a new report.

Then click the report link to download the newly generated report.

Once the report is on your system you can open it using Excel to see the values.

10 IGX Environmental Sensors

IGX includes built in support for reading a wide range of environmental sensors, such as pressure, temperature, humidity, oxygen concentration, and accelerometers. These sensors allow IGX devices to understand the conditions around them and make appropriate adjustments in real time. In many detector, beamline, and facility control applications, environmental context is just as important as the raw measurement itself. Small shifts in temperature, pressure, or humidity can influence detector behavior, change gas density, or introduce subtle drifts that accumulate over time. Integrating these sensors directly into the IGX framework allows users to validate conditions, apply corrections, and maintain stable performance without relying on external systems.

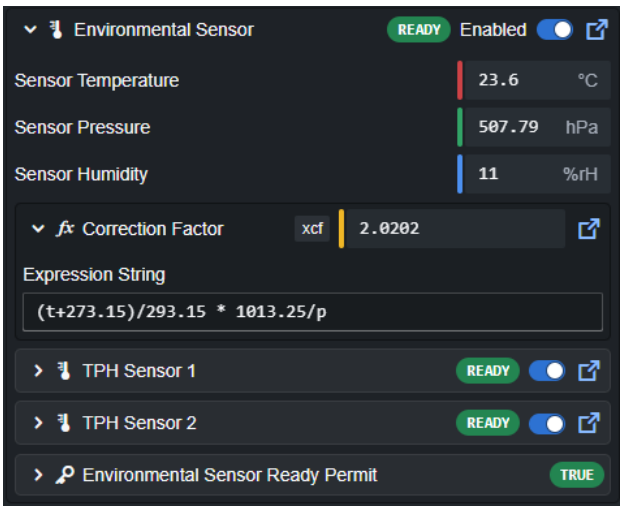
Different IGX hardware models support different combinations of sensors, so this document provides a general overview of how sensor data is handled rather than device specific details. Regardless of the hardware platform, IGX follows the same philosophy: gather environmental data at regular intervals, validate the readings, provide a consistent data model, and offer configurable tools for correction, monitoring, and interlocking.

Environmental sensing is especially important in gas filled detectors such as ion chambers. Gas density depends on temperature and pressure, so variations in the surrounding environment can influence detector response even when everything else is stable. Humidity control also matters for long term reliability since excess moisture can degrade components over time. By exposing sensor data and associated correction tools, IGX provides a structured way for users to maintain consistent measurements, follow local practices, and adapt the system to the standards used in their region.

10.1 Ion Chamber Internal Sensors

Accurate measurement of temperature, pressure, and humidity inside an ion chamber is essential. Variations in pressure and temperature change the gas volume, which directly affects the gain of the ion chamber. Humidity control also plays a key role in long-term detector reliability since excess moisture can degrade components and eventually lead to failures.

For this reason, Pyramid typically includes redundant TPH sensors in ion chamber designs and provides software support to read them on IGX based devices.



24 : Typical IGX Env. Sensor GUI

10.1.1 Automated Gas Correction Factor Calculation

IGX can calculate a gas correction factor using sensor data and the ideal gas law. The default equation is:

```
(t + 273.15) / 293.15 * 1013.25 / p
```

Where `t` is the temperature in Celsius and `p` is the pressure in hectopascals. Temperature is converted to Kelvin before applying the standard ideal gas law correction.

Users are free to modify the equation or add additional terms to follow the standard practices of their institution or country.

IGX periodically reads both sensors and checks that the values fall within reasonable limits. It will either select one sensor or average the two, depending on whether a sensor appears to be

malfunctioning. As new readings arrive, the correction factor is updated in real time. The result is typically close to 1 and is expected to be multiplied into the relevant calculations.

By default, IGX only computes the correction factor, and it can be configured to trigger an interlock if sensor data moves outside user defined limits.

It is up to the user to decide how to apply the correction factor, if they choose to use it at all.

Some users may choose to apply the correction factor only once during commissioning and then monitor it periodically to ensure it has not drifted too far. Others may apply the correction factor directly to the target dose, while some may prefer to apply it to the current scalars on the current inputs in real time. The specific application requirements will determine which strategy is the most appropriate.

10.1.2 Gas Correction Considerations

Even with a perfectly hermetically sealed ion chamber, the surrounding environment can still influence the gas volume. Thin chamber windows, which are common in many proton applications, are sensitive to external air pressure. Shifts due to weather patterns, altitude, or building air handling systems can place positive or negative forces on the windows and change the internal pressure. Environmental temperature changes will also alter the internal gas temperature over time. In these situations, the gas correction factor must be calculated using the internal sensors.

If the chamber is vented to air, the internal pressure and temperature will match the external environment and must be compensated using either internal or external sensors.

11 IGX PWM Module

IGX provides built-in support for outputting a Pulse Width Modulated (PWM) signal from certain physical outputs. A PWM signal is a repeating waveform that switches on and off at a configurable rate, where the proportion of time spent on versus off (the duty cycle) can be adjusted. PWM is commonly used for controlling LED brightness, driving motor or valve speed, generating reference or gate signals for external electronics, and modulating an output signal's intensity in closed-loop control applications.

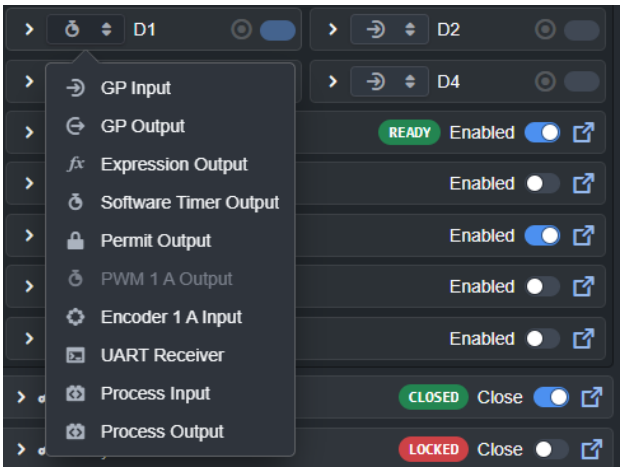
PWM output is provided by a PWM module. A device may have more than one PWM module available, each capable of driving two independent output channels, referred to as Channel A and Channel B. Both channels share the same frequency but have independently configurable duty cycle, enable, and polarity.

11.1 Enabling a PWM Output

Any physical output that supports PWM will have a **Mode** setting. To use the output as a PWM signal, set its Mode to the corresponding PWM option (for example, "PWM A Output" or "PWM B Output"). Once selected, the output will be driven by its associated PWM module and will continue to output the configured PWM waveform until the Mode is changed to something else.

An output can only be assigned one function at a time. Some outputs that support PWM may also support other special functions, such as capture or encoder input. Selecting PWM mode on an output will disable whichever other function was previously active on it.

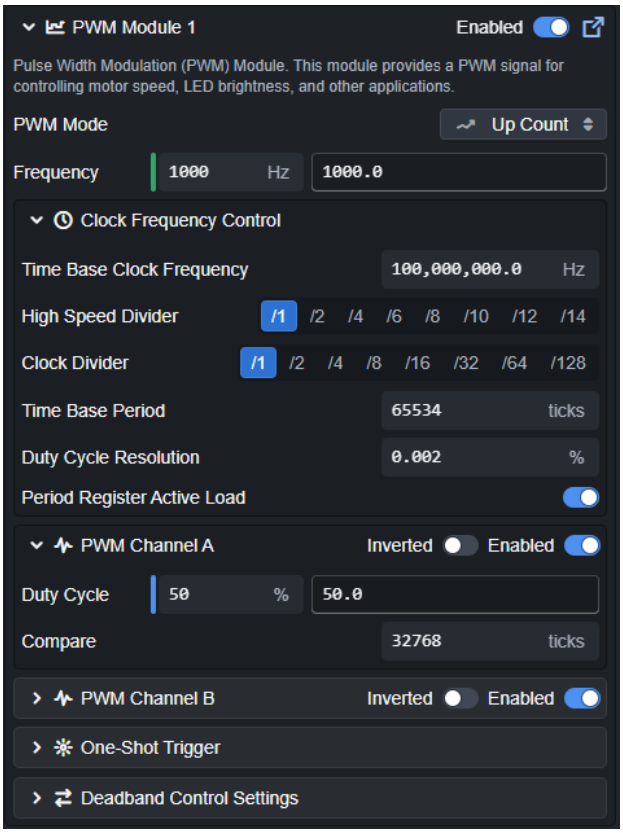
The PWM module driving the output must also be enabled. Each PWM module has its own configuration section in the GUI where its output is enabled and configured.



25 : D1 Configured to PWM 1 A Output

11.2 PWM Module Configuration

- **Module Enabled**: Master enable for the module. Must be turned on for either channel to produce an output.
- **PWM Frequency**: The switching frequency of the waveform, in Hz. Shared by both Channel A and Channel B. Can be set to a fixed value or a live expression.
- **Channel A Enabled / Channel B Enabled**: Enables output on the individual channel.
- **Channel A Duty Cycle / Channel B Duty Cycle**: The percentage of time that the channel's output is high, from 0–100%. Can be set to a fixed value or a live expression.
- **Channel A Inverted / Channel B Inverted**: Inverts the output polarity for the channel.



26 : PWM Configuration GUI

- **PWM Mode**: Selects how the module's internal counter behaves: **Up Count** and **Down Count** produce a continuously repeating waveform, **Freeze** holds the output in its current state, and **One-Shot** outputs a single pulse each time it is triggered.
- **Trigger Q-Pulse**: A button that manually fires a single output pulse when PWM Mode is set to One-Shot.
- **Clock Divider** and **High Speed Divider**: Two dividers that together set the module's internal timing resolution. See below for how they differ and how to use them.
- **Period Register Active Load**: Controls when a change to PWM Frequency, Clock Divider, or High Speed Divider takes effect. When enabled, the change applies immediately, which may produce one irregular cycle during the transition. When disabled, the change is held and applied cleanly at the start of the next cycle.

11.2.1 Clock Divider vs. High Speed Divider

Every PWM module counts against a fixed internal base clock. Clock Divider and High Speed Divider each scale that base clock down, and their effects multiply together to produce the module's actual internal counting rate (shown in the read-only **Time Base Clock Frequency** field below). They exist as two separate controls because each one offers a different set of steps:

- **Clock Divider** offers large, doubling steps: /1, /2, /4, /8, /16, /32, /64, /128. Use it to quickly get into the general range of counting rate your target frequency needs, especially for very low frequencies.
- **High Speed Divider** offers smaller, closer-spaced steps: /1, /2, /4, /6, /8, /10, /12, /14. Use it to fine-tune the combined divide ratio to values that Clock Divider's big doubling steps can't reach on their own (such as an overall divide of /6, /10, /12, or /14).

In practice, set Clock Divider first to reach the right general range for your target frequency, then adjust High Speed Divider to dial in the exact value.

11.2.2 How Frequency and Duty Cycle Resolution Interact

Internally, a PWM module counts up (or down) to a set point and repeats, and the requested frequency and duty cycle are translated into that count. This means the achievable duty-cycle resolution (how finely you can adjust the on/off ratio) depends on how many counts fit into a single cycle at the configured frequency. Higher frequencies leave fewer counts per cycle, and therefore coarser duty-cycle steps; lower frequencies leave more counts per cycle, and therefore finer duty-cycle steps. Raising Clock Divider or High Speed Divider lowers the counting rate, which allows lower frequencies and finer duty-cycle resolution; lowering them raises the counting rate, which allows higher frequencies.

This same limit also caps how low a frequency can be reached at all: since the counter can only count up to a fixed maximum before it must repeat, a frequency below `Time Base Clock Frequency / 65534` cannot be represented and will be clipped to that floor. If a target frequency is unexpectedly low, raise Clock Divider or High Speed Divider to bring the floor down with it.

The following read-only fields let you confirm exactly what the module resolved your requested settings to:

- **Time Base Clock Frequency**: The effective internal counting rate after applying Clock Divider and High Speed Divider.
- **Time Base Period**: The number of internal counts in one full cycle at the configured frequency.
- **Duty Cycle Resolution**: The smallest duty-cycle step achievable at the current frequency, expressed as a percentage.
- **Channel A Compare / Channel B Compare**: The internal count value at which the channel's output toggles, corresponding to its configured duty cycle.

If **Duty Cycle Resolution** is coarser than your application requires, try reducing Clock Divider or High Speed Divider so more counts are available at your target frequency.

11.3 Dead-Band (Complementary Outputs)

The dead-band feature can synthesize a delayed, complementary output for switching applications where two outputs (for example, the high side and low side of the same circuit) must never be active at the same instant. It works as two independent delay modules: a rising-edge module that outputs onto Channel A, and a falling-edge module that outputs onto Channel B.

- **Deadband Rising Edge Enabled**: When enabled, the rising-edge delay module drives Channel A's output.
- **Deadband Rising Edge Delay**: How long the rising-edge module delays its output, in ticks (0–1023).
- **Deadband Rising Edge Input B**: By default the rising-edge module delays Channel A's own waveform. Enable this to have it delay Channel B's waveform instead.
- **Deadband Rising Edge Inverted**: Inverts the polarity of the rising-edge module's output.
- **Deadband Falling Edge Enabled**: When enabled, the falling-edge delay module drives Channel B's output.
- **Deadband Falling Edge Delay**: How long the falling-edge module delays its output, in ticks (0–1023).
- **Deadband Falling Edge Input B**: By default the falling-edge module delays Channel A's waveform. Enable this to have it delay Channel B's waveform instead.
- **Deadband Falling Edge Inverted**: Inverts the polarity of the falling-edge module's output.

11.4 Driving PWM From Live Expressions

PWM Frequency, **Channel A Duty Cycle**, and **Channel B Duty Cycle** all accept either a fixed value or a live expression. This allows a PWM output to automatically track another live measurement or calculated value in real time. For example, it can drive an LED brightness output proportional to a live sensor reading, or scale a fan or pump duty cycle off a live temperature reading. See the IGX Expression Language Guide (see page 54) for more information on writing expressions.

11.5 Example Configurations

The following examples show how the fields above might be combined for a few typical applications.

11.5.1 Monitoring a Current Reading as a PWM Signal (Duty Cycle Modulation)

- **Module Enabled**: On
- **PWM Frequency**: 1000 Hz, fixed
- **Channel A Enabled**: On, **Channel A Duty Cycle**: expression `(i1 / 200) * 100`
- **Clock Divider**: /2, **High Speed Divider**: /1

This uses a live expression to turn a current input reading (here, `i1`) into a duty-cycle-encoded output. If `i1` is a current reading in nA, this expression scales a 0–200 nA signal onto a 0–100% duty cycle, so a downstream system reading the PWM output can recover the current value from the duty cycle alone. Adjust the divisor in the expression (200 in this example) to match the full-scale range of the signal you are monitoring. Clock Divider is set to /2 here because a 1000 Hz cycle does not fit in the module's counter at the default divider setting, as described above.

11.5.2 Encoding a Live Reading as PWM Frequency (Frequency Modulation)

- **Module Enabled**: On
- **PWM Frequency**: expression `100 + (i1 * 50)`
- **Channel A Enabled**: On, **Channel A Duty Cycle**: 50%, fixed
- **Clock Divider**: /16, **High Speed Divider**: /1

Instead of encoding the measurement in the duty cycle, this example encodes it in the frequency itself: an idle current input (`i1` = 0) produces a steady 100 Hz output, and the frequency rises by 50 Hz for every 1 nA of current. The duty cycle is left fixed at 50% since only the frequency carries the signal here. This style of output is useful for equipment that reads a frequency directly, such as a frequency counter, tachometer input, or older rate meter, rather than a duty cycle or analog voltage.

Because the expression's lowest expected value (the 100 Hz idle frequency) is well below what the default dividers can reach, Clock Divider is lowered to /16 so that the module's counter comfortably covers the entire expected frequency range, including its low end. When designing a frequency-modulated output like this, always check that the lowest frequency your expression can produce stays above `Time Base Clock Frequency / 65534` at your chosen divider settings, or it will be silently clipped to that floor instead of tracking your expression.

11.5.3 LED Brightness Control

- **Module Enabled**: On
- **PWM Frequency**: 2000 Hz, fixed (or an expression tied to a brightness setpoint)
- **Channel A Enabled**: On, **Channel A Duty Cycle**: 0–100%, following the desired brightness level
- **Clock Divider** / **High Speed Divider**: /1 and /1 (defaults)

At the default dividers, 2000 Hz resolves to a Time Base Period of about 50,000 ticks, giving a Duty Cycle Resolution around 0.002%, far finer than the eye can perceive, while staying well above the flicker threshold.

11.5.4 Motor or Fan Speed Control

- **Module Enabled**: On
- **PWM Frequency**: 20,000 Hz, fixed (chosen to stay above the audible range for most drivers)
- **Channel A Enabled**: On, **Channel A Duty Cycle**: an expression tracking a live speed or temperature setpoint
- **Clock Divider** / **High Speed Divider**: /1 and /1 (defaults)

At the default dividers, 20,000 Hz still resolves to a Duty Cycle Resolution around 0.02%, more than fine enough for smooth speed control.

11.5.5 Slow-Cycling Valve or Heater Control

- **Module Enabled**: On
- **PWM Frequency**: 1 Hz, fixed (one full on/off cycle per second)
- **Channel A Enabled**: On, **Channel A Duty Cycle**: the desired percentage of time the load is powered
- **Clock Divider**: /128
- **High Speed Divider**: /12

A 1 Hz cycle is far too slow for the default dividers. This combination brings the Time Base Clock Frequency down to about 65.1 kHz, giving a Time Base Period of about 65,100 ticks, just under the module's maximum, so the full 1 Hz cycle fits without clipping. Without lowering the dividers this way, the module cannot represent a cycle this slow.

11.5.6 Complementary Switching Pair (Half-Bridge Gate Drive)

- **Channel A Enabled** and **Channel B Enabled**: On
- **Channel A Duty Cycle** and **Channel B Duty Cycle**: 50%
- **Channel B Inverted**: On, so Channel B runs opposite to Channel A
- **Deadband Rising Edge Enabled** and **Deadband Falling Edge Enabled**: On
- **Deadband Rising Edge Delay** and **Deadband Falling Edge Delay**: 100 ticks each

At the default Time Base Clock Frequency of 100 MHz, 100 ticks is about 1 microsecond. This inserts a small gap around every transition so Channel A and Channel B are never both active at once, protecting a driven circuit (such as a half-bridge) from a damaging shoot-through condition.

11.5.7 Manually Firing a Single Test Pulse

- **PWM Mode**: One-Shot
- **Channel A Duty Cycle**: sets the width of the single pulse, as a percentage of the configured period
- **Trigger Q-Pulse**: press to fire one pulse on demand

This is useful for testing wiring or checking a downstream device's response to a pulse without leaving the output running continuously.

12 IGX Dose Controller

The IGX Dose Controller is a process controller designed to precisely deliver a prescribed dose of charge by continuously measuring current inputs from a connected device.

At its core, the Dose Controller operates by:

1. Activating or not activating a current source via a digital on/off signal.
2. Integrating the incoming current over time to accumulate a total charge.
3. Deactivating the current source (or signaling another system to do so) once the prescribed charge has been reached.

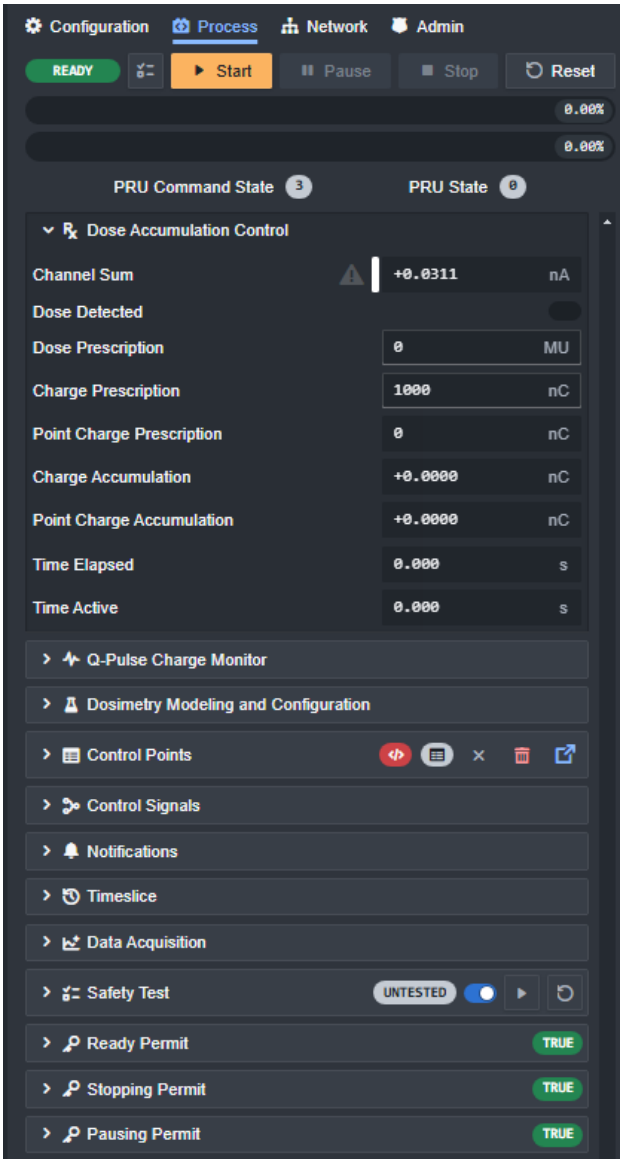
This automated control ensures accurate and repeatable dose delivery, making it ideal for applications that require precise charge integration and process control.

The Dose Control web GUI provides access to all features of the dose delivery process, allowing for manual control, real-time monitoring, and configuration of dose prescriptions.

Through this interface, you can:

- Start, pause, resume, stop, and reset the dose controller.
- Set target prescriptions in terms of dose or charge and monitor delivery in real time.
- View system status and ensure all required conditions are met before starting a session.

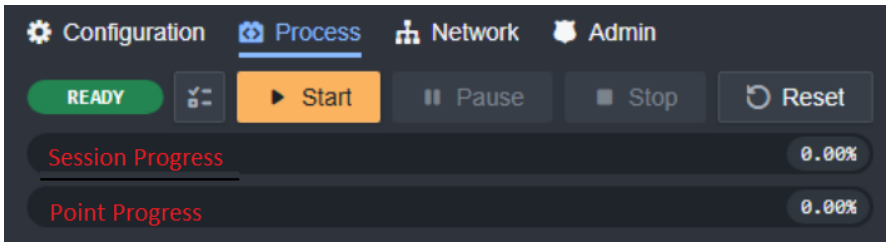
The following sections provide a detailed breakdown of each stage of the Dose Control process, including configuration, monitoring, and safety features.



12.1 Dose Controller Features

12.1.1 Dose Session Management

The dose control panel provides manual control and real-time monitoring of the Dose Controller. It allows users to transition the controller through different states, provided that all required permits are granted. If an action is unavailable at a given time, the corresponding button will appear grayed out, indicating that the system does not currently allow that operation.



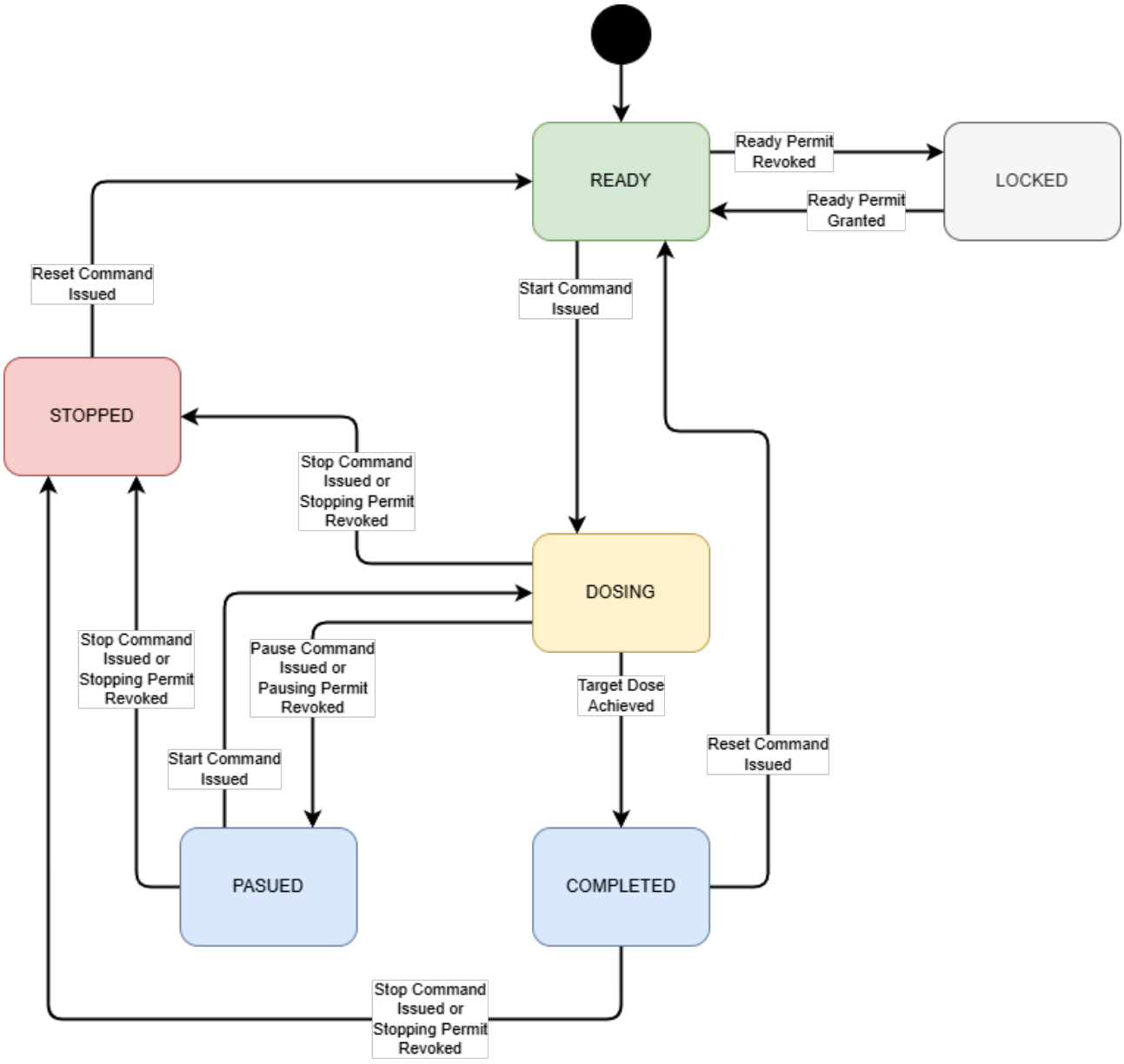
27 : High-level Dose Session Controls

High-Level Dose Session Controls

The following IO control and monitor the Dose Controller session:

- **State**: READY Displays the current state of the Dose Controller, located in the upper right corner. The possible states include:
  - **READY (GREEN)** – The session is prepared to start.
  - **LOCKED (GRAY)** – The session cannot start due to the ready permit being revoked.
  - **DOSING (ORANGE)** – The session is actively delivering dose, though the beam may or may not be on.
  - **PAUSED (BLUE)** – The session is temporarily halted but can be resumed.
  - **STOPPED (RED)** – The session is terminated and cannot be resumed.
  - **COMPLETED (BLUE)** - The session has hit the target and no longer dosing.
- **Safety Test**: Runs a series of *optional* safety checks before starting a session. The controller can be configured to prevent a start command unless the test passes.
- **Start Button**: Start Begins or resumes a dose session.
- **Pause Button**: Pause Temporarily pauses the session, allowing it to be resumed later.
- **Stop Button**: Stop Permanently stops (terminates) the session, preventing it from being resumed later. The session must be reset in order to continue to the next state.
- **Reset Button**: Reset Clears out all accumulated dose values and resets the session, the controller returns the READY state if the ready permit is granted, otherwise it will be in the LOCKED state.
- **Session Progress**: Displays the percentage completion for the entire session, including all control points across the session. Only relevant if you have more than one control point.
- **Point Progress**: Displays the percentage completion for the current active control point.

The state machine diagram (shown below) illustrates how the controller transitions between different states.



The controller cannot transition from READY to DOSING unless the ready, pausing, and stopping permits are all granted.

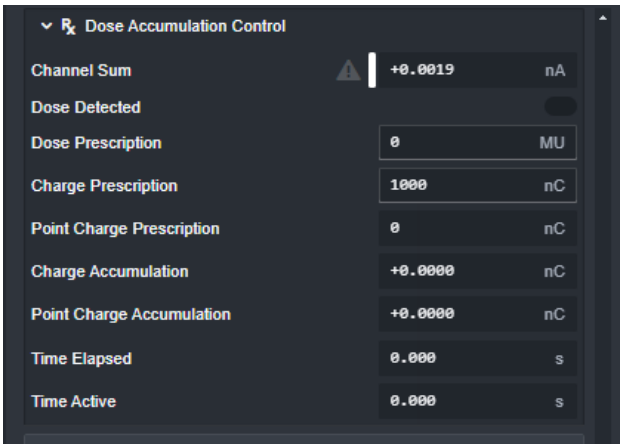
The controller state is evaluated at a minimum of 500 times and normally 1,000 times per second, ensuring that:

- If a stopping or pausing permit is revoked, the controller will automatically move into the STOPPED or PAUSED state.
- The pause and stop functions are always available while in the DOSING state, allowing manual intervention at any time.
- The stop function is available in the COMPLETED state in case the system must terminate due to an unexpected error condition.

12.1.2 Dose Accumulation Control

The Dose Accumulation Control system continuously monitors and integrates incoming current to ensure accurate dose delivery.

The following Input/Output (IO) parameters provide real-time updates on the dose accumulation process:

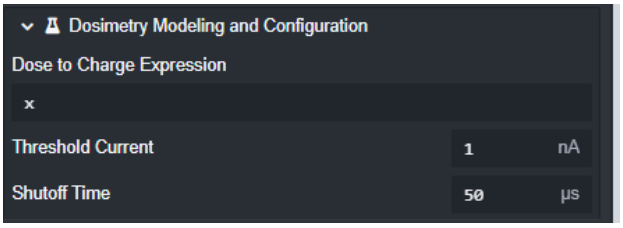


28 : Dose Accumulation Control GUI

- **Channel Sum**: Displays the live current input that is actively happening.
- **Dose Detected:** This LED will be true if the current input is greater than the threshold current limit.
- **Dose Prescription**: Defines the target dose in Monitor Units (MU). Changing this value will automatically update the Charge Prescription. If Dose Prescription is not used, set it to zero.
- **Charge Prescription**: Specifies the total target charge in nanocoulombs (nC) that the controller must accumulate. This field is read-only unless Dose Prescription is set to zero.
- **Point Charge Prescription**: Represents the target charge for the active control point within the control point table.
- **Charge Accumulation**: Displays the total accumulated charge in nanocoulombs (nC) for the entire session across all control points.
- **Point Charge Accumulation**: Displays the accumulated charge in nanocoulombs (nC) for the currently active control point.
- **Time Elapsed**: Tracks the total time since the dose process started, including time spent in DOSING or PAUSED states.
- **Time Active**: Tracks the time spent actively delivering dose at the current control point. This only increments while in the DOSING state.

12.2 Dosimetry Modeling and Configuration

The Dosimetry Modeling and Configuration section of the IGX GUI allows you to modify key dosimetry parameters to fit the specific requirements of your application. These settings ensure precise dose delivery and enable the dose controller to function correctly by modeling how charge and dose are related, defining beam detection criteria, and accounting for system response time.



29 : Modeling and Configuration GUI

- **Dose to Charge Expression:** A mathematical expression that converts between dose (MU) and charge (Coulombs). This expression is only needed if dose-based prescriptions are used; if working solely with charge-based prescriptions, it can be left unconfigured.
- **Threshold Current:** Defines the minimum current level required for the system to recognize an active beam. This helps distinguish real beam delivery from background noise, preventing false triggers in the dose controller.
- **Shutoff Time:** A configurable delay that accounts for how quickly the system can turn off the beam. This allows the dose controller to anticipate when to stop beam delivery to prevent

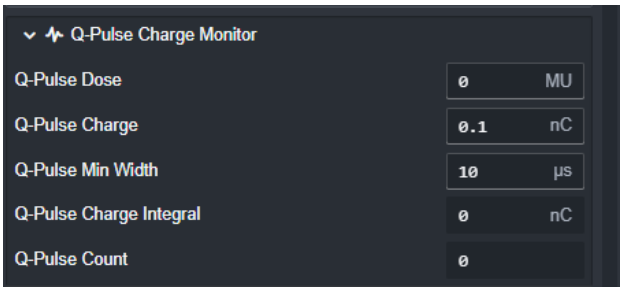
overshooting the prescribed dose. A setting of zero disables this behavior, meaning the system will not compensate for turn-off delays.

12.2.1 Practical Considerations

- **Precise dose control**: Proper configuration of these parameters ensures accurate dose delivery, reducing the risk of over- or under-dosing.
- **System-specific tuning**: Each radiation system has different characteristics, so these parameters should be adjusted based on beam response times, background noise levels, and dose calibration requirements.
- **Ensuring accurate modeling**: The Dose to Charge Expression should be validated using known calibration data to ensure correct dose calculations.

12.3 Q-Pulse Charge Monitor

The Pulse Charge Monitor provides a digital pulse output that represents a defined amount of measured charge. This output pulse is normally high and momentarily drops when triggered. Each pulse corresponds to a specific dose or charge value, allowing external systems to track accumulated charge without directly reading the analog current.



30 : Q-Pulse Charge Monitor GUI

This pulse signal can be used for an independent dose termination system or integrated into other systems requiring a simple digital interface for monitoring charge delivery.

- **Q-Pulse Dose:** The Monitor Units (MU) assigned to each pulse.
- **Q-Pulse Charge:** The charge value per pulse in nanocoulombs (nC).
- **Q-Pulse Min Width:** The minimum pulse width in microseconds (µs) to ensure proper recognition by external systems.
- **Q-Pulse Charge Integral:** The total integrated charge from all pulses since the session started and entered the DOSING state.
- **Q-Pulse Count:** The total number of pulses generated since the session started and entered the DOSING state.

12.4 Control Signals

The dose controller uses a series of special software inputs and outputs called “Control Signals”. These signals can be re-wired to a variety of the device’s physical inputs and outputs such as TTL or fiber optics.

31 : Control Signals GUI

- **in_gate**: A versatile signal that pauses delivery when it goes low and resumes when it goes high. If not assigned, it's ignored by the controller. It's useful for implementing respiratory gating or other fast signals, with a response time between 10 and 20 microseconds. It can also serve as a simple pause button or an input from a PLC.
- **in_safety**: This signal feeds into the "Safety and Depositing Agreement Interlock" and "Safety State Interlock" but has no default behavior. It can terminate or pause delivery based on an external input and is best for latching-type errors. For example, you could use a digital signal from the accelerator to indicate beam extraction status. If the device detects a safety disagreement (e.g., beam extraction while *out_dose_enable* is low), the controller will halt the delivery.
- **out_point_started:**
- **out_point_done:**
- **out_dose_enable**: This signal turns the beam on and off under nominal conditions, allowing the controller to accurately hit the prescribed dose by accounting for beam-off latency based on measured input current.
- **out_q_pulse**: This monitor pulse output triggers each time a specified aliquot of charge is measured. While the controller may fall behind temporarily, it will catch up by the end of delivery to reflect the correct total. To avoid bunching up of pulses when the beam current is high, configure a larger Q-Pulse charge.

12.5 Controller Notifications

This section lets you reconfigure when the dose controller should post notifications.

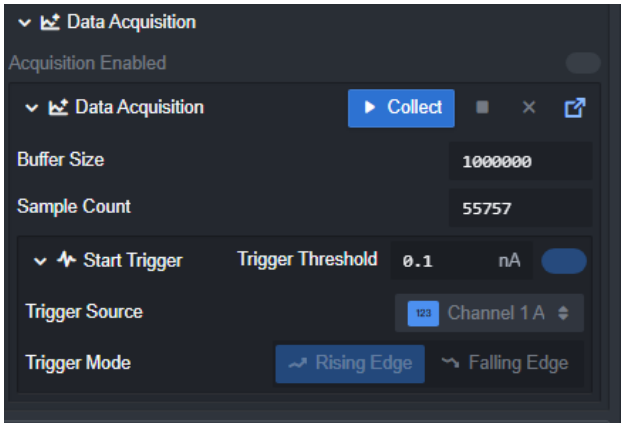
- **Notification Mode:** Sets the conditions when the controller should post a notification.
- **Reset Notifications:** Setts the conditions when the controller should automatically clear any notifications, if ever.

32 : Notification Configuration GUI

12.6 Automatic Data Acquisition

This section lets you start and stop a data acquisition session when the dose controller starts or stops. This is helpful for capturing high speed data during a dose session.

The parameters of displayed are an exact duplicate of the device's typical data acquisition parameters.



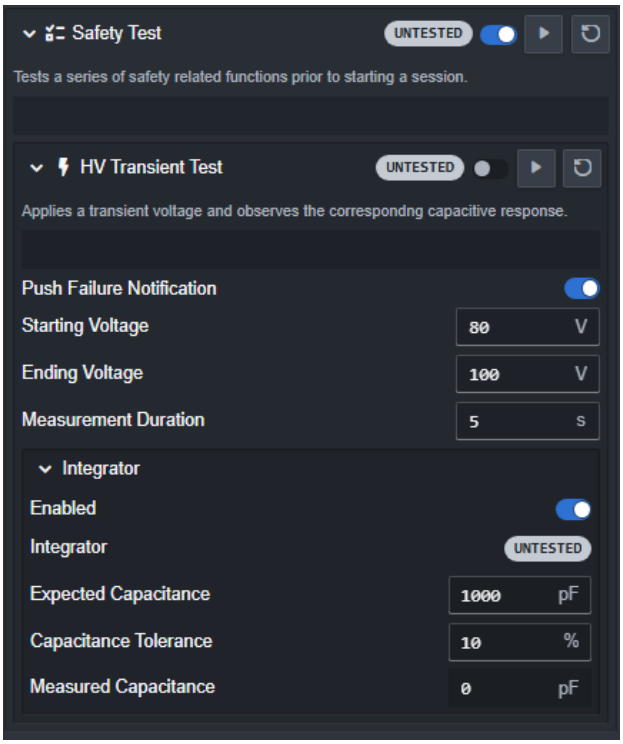
33 : Dose Control Data Acquisition GUI

12.7 Safety Test

The safety test is an optional automated diagnostic test designed to verify that the dosimetry system is safe and fully functional before initiating treatment. This test ensures that critical system components are working as expected and that there are no faults that could affect dose delivery.

The test can be run at any time when the controller is not actively running. It can be performed before every session for maximum safety or conducted periodically as needed.

The safety test is constructed out of a series of optional sub-tests.



34 : Safety Test GUI

Each test has the following common IO:

- **State:** The current state of a test case. Either untested, pass, or fail.
- **Enable:** Turning on the test will enable it to run when the start button is pressed.
- **Start:** Starts the test and executes it if enabled. If disabled, the test always passes.
- **Reset:** Forces the test state back to untested.
- **Push Failure Notification:** When enabled, the test will post a notification if the test fails.

12.7.1 HV Transient Test

The HV Transient Test is available on all devices equipped with a high-voltage (HV) power supply. It is designed to verify the integrity and calibration of the system by applying a controlled voltage transient and measuring the resulting capacitance.

This test ensures that:

- HV and signal cables are properly connected.
- The electrometer is correctly calibrated and operational.
- The HV power supply is functioning correctly.
- There are no short circuits or open circuits in the detector.

By conducting this test, multiple critical aspects of the system can be verified simultaneously with minimal effort.

Test Methodology

The test applies a fixed voltage transient to the HV power supply. This results in a charge accumulation through capacitive coupling on the input signals. The total charge is then integrated, and the capacitance is calculated using the formula: C = Q / V

where:

- C is the capacitance (in farads),
- Q is the charge (in coulombs),

- V is the applied voltage (in volts).

The test passes if the measured capacitance falls within the configured tolerance range of the expected capacitance.

Test IO Parameters

- **Starting Voltage:** The voltage to output at the start of the test.
- **Ending Voltage:** The voltage to ramp up to during the test.
- **Measurement Duration:** The duration of the charge integrating period during the test.
- **Integrator:**
  - **Expected Capacitance:** The ideal capacitance that should be considered a perfectly passing test.
  - **Capacitance Tolerance:** The amount of allowable tolerance from the expected capacitance.
  - **Measured Capacitance:** The measured capacitance after the transient test is completed.

Test Procedure

1. Manually ensure all cables are connected and all test parameters are configured correctly.
2. At the start of the test, the device HV output will be set to the starting voltage and remember the user's last HV command for later.
3. The device will wait until the HV output and any transient current settles.
4. Then the device will start integrating charge and sets the HV output to the ending voltage.
5. The device waits the configured amount of time, then calculates the capacitance and reports the result.
6. The device turns sets the HV output to whatever the value was configured to before the test started.

Interpretation of Results

- **Pass**: The measured capacitance is within the expected tolerance range.
- **Fail**: The measured capacitance is outside the expected range, indicating potential issues such as disconnected cables, calibration errors, or faults in the detector.

Troubleshooting

If the test fails, consider the following checks:

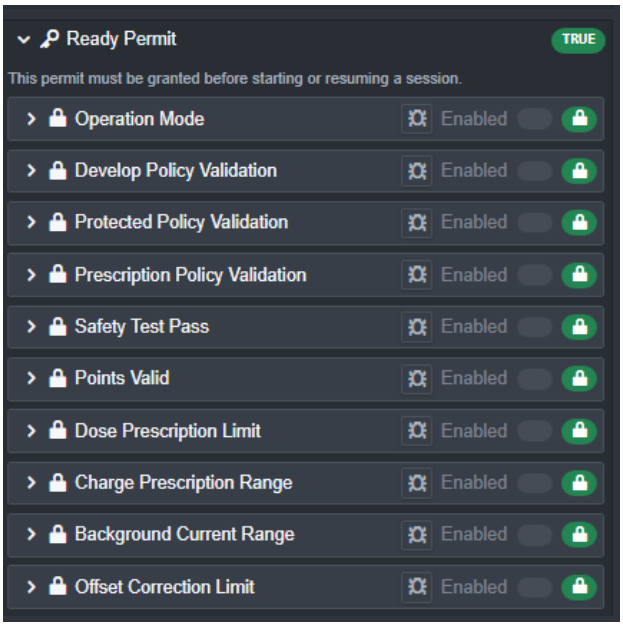
- Inspect HV and signal cable connections.
- Recalibrate the electrometer.
- Verify that the power supply is functioning properly.
- Check for potential short circuits or open circuits in the detector.

12.7.2 Ready Permit

The Ready Permit is an interlock mechanism designed to prevent the initiation or resumption of a dosing session until all prerequisite conditions have been met. These interlocks ensure that the system is in a safe and ready state before treatment begins.

If the Ready Permit is revoked during an active session, it does **not** interrupt or stop the session. It only prevents a new session from starting.

This allows the interlock conditions to be highly stringent, enforcing strict checks on parameters that may fluctuate significantly during a session without causing unnecessary interruptions.



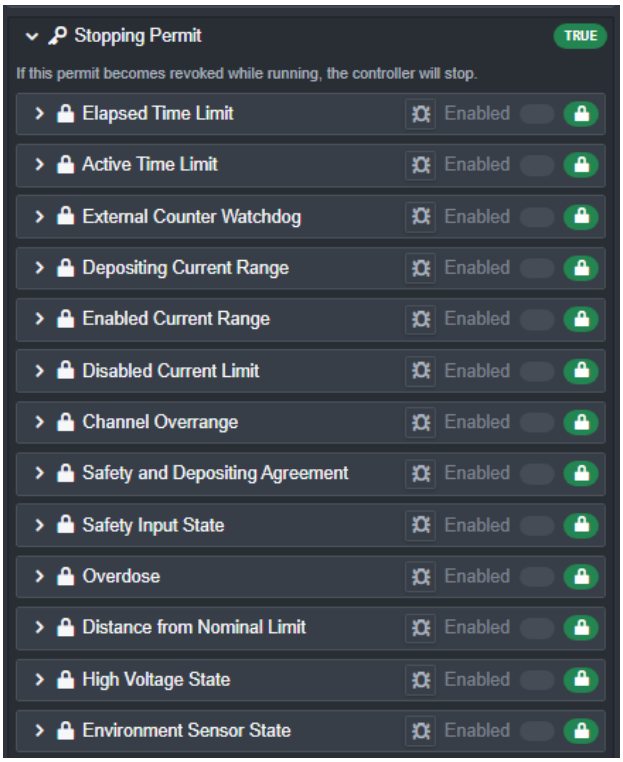
35 : Ready Permit Interlocks

12.7.3 Stopping Permit

The Stopping Permit is a critical safety interlock that ensures a session is immediately terminated if any of the associated interlocks are faulted.

- A session cannot start unless the Stopping Permit is granted.
- If the Stopping Permit is revoked at any time during a session, including when the session is paused, the session is immediately terminated.

Interlocks within this permit should be configured to detect **severe fault conditions** that require an immediate stop, and a complete system reset. These conditions should represent serious safety or operational failures that make continuing the session unsafe or impossible.



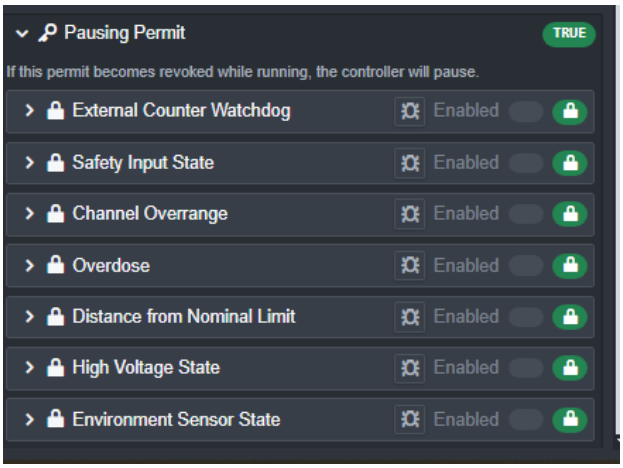
36 : Stopping Permit Interlocks

12.7.4 Pausing Permit

The Pausing Permit is designed to detect issues that warrant temporarily pausing a session, allowing a human operator to assess the situation and determine whether the session can safely continue.

- If the Pausing Permit is revoked, the session is paused, but not terminated.
- These interlocks should be configured for conditions that cause a delay in delivery, but do not necessarily indicate a complete system failure.

These conditions are often referred to as **recoverable faults**. Many interlocks in this permit may be similar or identical to those in the Stopping Permit, but with less stringent limits. This allows the system to differentiate between critical failures requiring an immediate stop and issues that might be resolved through operator intervention.



37 : Pausing Permit Interlocks

12.8 IGX Simple Dose Controller Example

12.8.1 Simple Dose Control Example

The Dose Controller application is highly flexible and feature rich. However, setting up dose control for the first time, especially with interlocks, can be challenging since a consistent set of conditions must be met before operation. This section provides a simplified example of dose control that remains a valid mode of operation for irradiation with a single control point, such as dose delivery from a proton beamline using lateral scattering and energy modulation.

12.8.2 Example Configuration

In this example, the device will be set up with the following behavior:

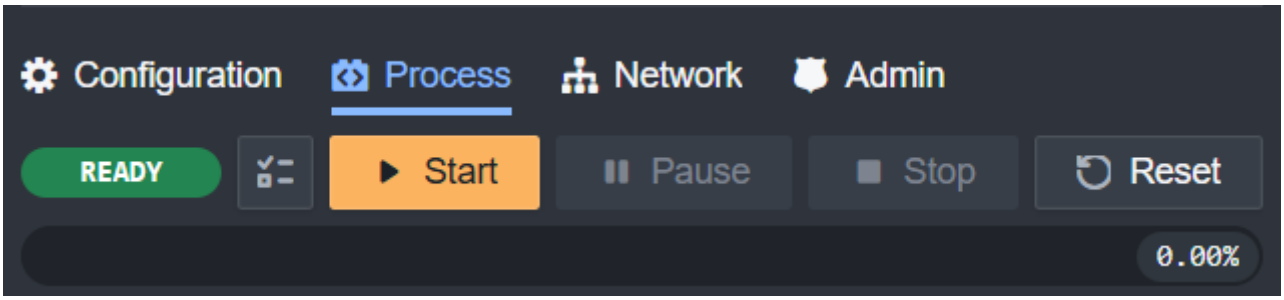
- **Dose Enable:** The dose enable output will be controlled by fiber optic transmitter FT2.
- **Q-Pulse:** Pulses indicating a defined quantum of measured charge will go to fiber optic transmitter FT3 and digital output D2 on the expansion port.
- **Automatic Dosing Stop Conditions:** The device will automatically stop dosing if any of the following conditions occur:
  - The sum of the measured currents exceeds the configured limits.
  - The total elapsed time exceeds the predefined limit.

This example keeps the configuration simple, but in a real-world setup, additional conditions can be implemented to:

- Inhibit the start of dosing with the ready permit.
- Respond to excess dark current
- React to other system-specific interlocks

These advanced configurations are beyond the scope of this example but are available for more complex control scenarios.

Important Note: The control fields will remain disabled unless the Dose Controller is in the ready state. If needed, press the reset button to return the system to the ready state before proceeding.



38 : Dose Controller State and Control Buttons

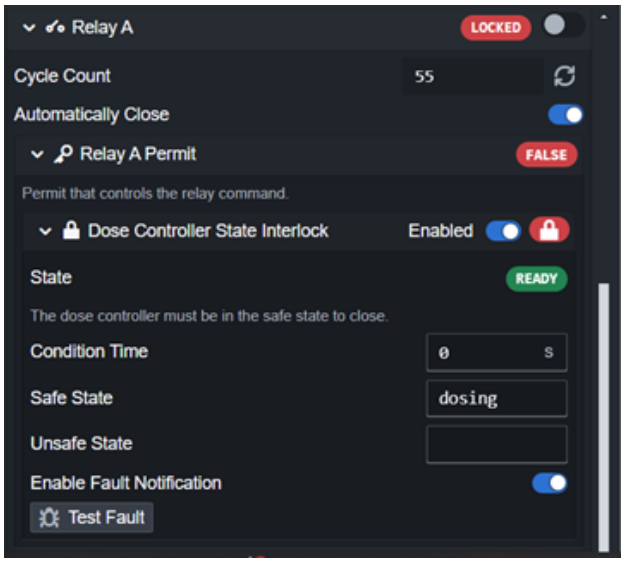
Device IO Configuration

To integrate the device's physical, IO with the Dose Controller, we must first map the hardware IO to the appropriate software signals. For digital and fiber optic IO, the configuration requires:

1. Specifying the IO type, in this case, it is always set as a process output, meaning that the Dose Controller has control over this IO.
2. Assigning the process signal, this determines which internal software process signal the IO corresponds to within the controller.

The IO selections in this example are not mandatory, you can customize the configuration based on your specific application needs.

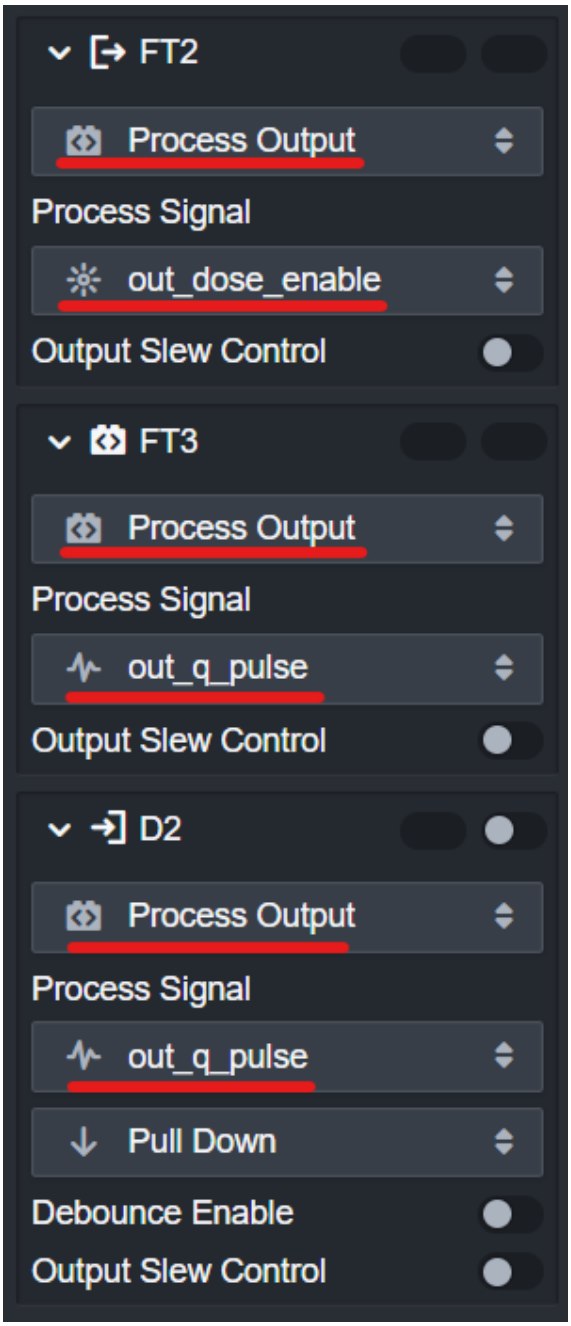
Navigate to the Configuration tab and set up the relay and digital outputs as follows:



39 : Relay A Configuration

This relay configuration will automatically close the relay if the permit is granted (all interlocks are ok) and open the relay if the controller is in any state other than dosing.

You might connect this relay to a safety input which would turn off the radiation source, through the secondary path.



40 : Digital IO Configuration

Interlock Configuration

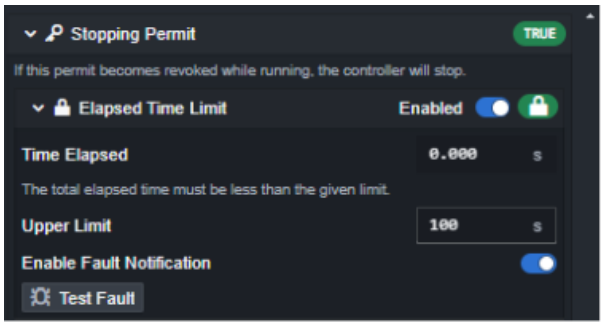
To ensure safe operation, we need to configure interlocks that will automatically stop the Dose Controller if certain conditions are violated.

These interlocks define additional stopping conditions beyond simply reaching the dose target. If any of the specified conditions are not met, the system will open the relay, effectively stopping dosing.

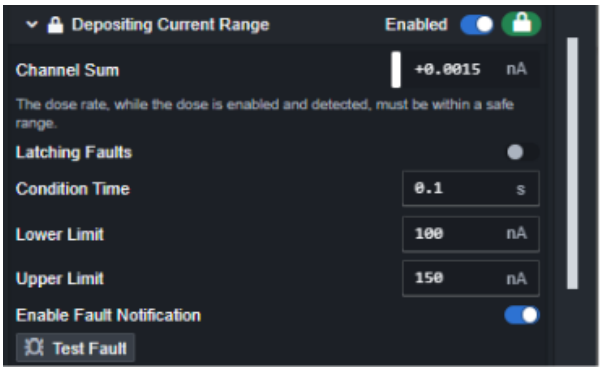
For this example, configure the interlocks as follows:

- **Total time limit:** Must not exceed 100 seconds.
- **Current sum limit:** The total current from all four channels must stay within 100 nA to 150 nA.

These interlocks ensure that dosing is terminated if the process runs too long or if the measured current falls outside the acceptable range. More complex interlock conditions can be added based on specific application requirements.



41 : Elapsed Time Limit Interlock



42 : Depositing Current Range Interlock

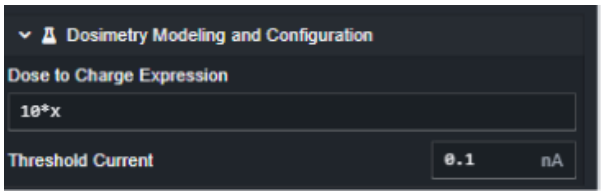
Dose Model and Q-Pulse Configuration

To properly control dose delivery, we need to configure dose scaling and the total dose target. The dose controller allows dose to be measured in:

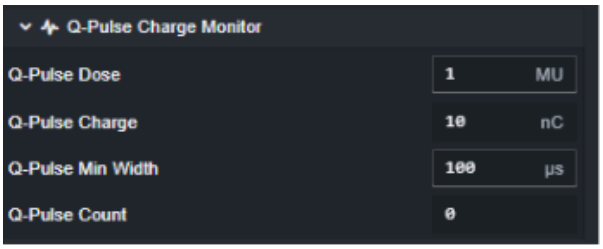
- Nanocoulombs (nC): Direct measurement of charge.
- Monitor Units (MU): A user-defined unit where one MU corresponds to a specific amount of charge.

For this example, we will use the following setup:

- **Dose to Charge Model:** Define the relationship between charge and MU such as 1 MU is equal to 10 nC. This expression is frequently just a linear relationship but can also be modified to take into account variables such as ion chamber pressure and temperature or accelerator energy.
- **Pulse output configuration:** Configure the controller to generate one Q-Pulse per 1 MU.



43 : Dose Model Configuration



44 : Q-Pulse Configuration

Setting Up a Current Source

To ensure proper dose delivery, connect a suitable test current source, making sure that:

- It is gated by Relay A, allowing the system to control when current is applied.
- It is gated by Fiber Optic Transmitter FT2, ensuring synchronization with the dose measurement process.

This example includes redundant shutoff mechanisms, a common practice in medical systems to enhance safety and reliability. However, if you are only testing the software, using just one of these gating methods is sufficient.

- **For best accuracy:** Use fiber optic transmitter (FT2), it provides sharper on/off timing and maximizes dose accuracy.
- **For easier wiring:** Use Relay A, it may be simpler to set up for testing purposes.

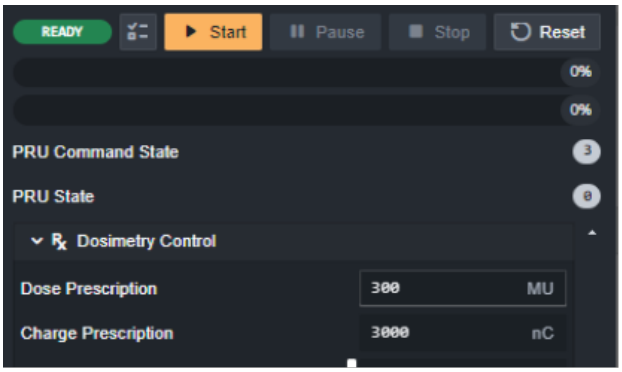
Choosing the appropriate gating method depends on your testing needs and accuracy requirements.

12.8.3 Running the Example Setup

With the setup complete, the final step is to set the target dose and start the dose controller.

Setting the Dose Prescription (Target)

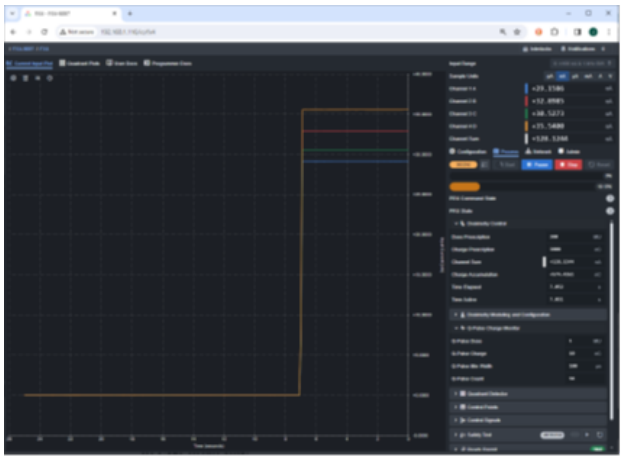
- Define the dose prescription in Monitor Units (MU) based on your application's requirements.
- The target dose is automatically converted to total charge to be delivered before the system automatically stops.



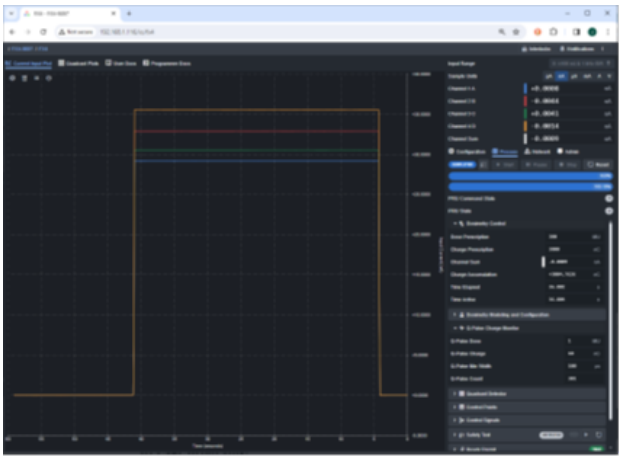
45 : Dose Prescription Under Dosimetry Control

Starting Dose Delivery

1. Press the Start button to begin dose delivery.
2. The device will deliver the configured dose based on the set MU value.
3. Once the target dose is reached, the system will automatically turn off the current, stopping the session.



46 : Dose Controller Starting



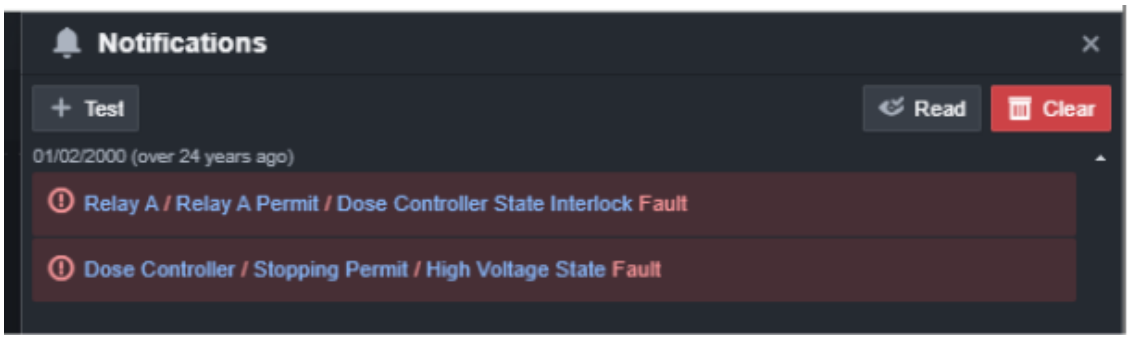
47 : Dose Controller Completed

Resetting for the Next Prescription

- After delivery is complete, press the reset button.
- The controller will return to the Ready state, allowing it to accept a new prescription for the next session.

Tracking Faults

If any of the conditions that allow dosing are violated, then the device will open the relay to stop the current and the notifications list will identify the reason:



48 : Fault Notifications

12.9 IGX Dose Controller for PBS

12.9.1 Pencil Beam Scanning Dosimetry Background

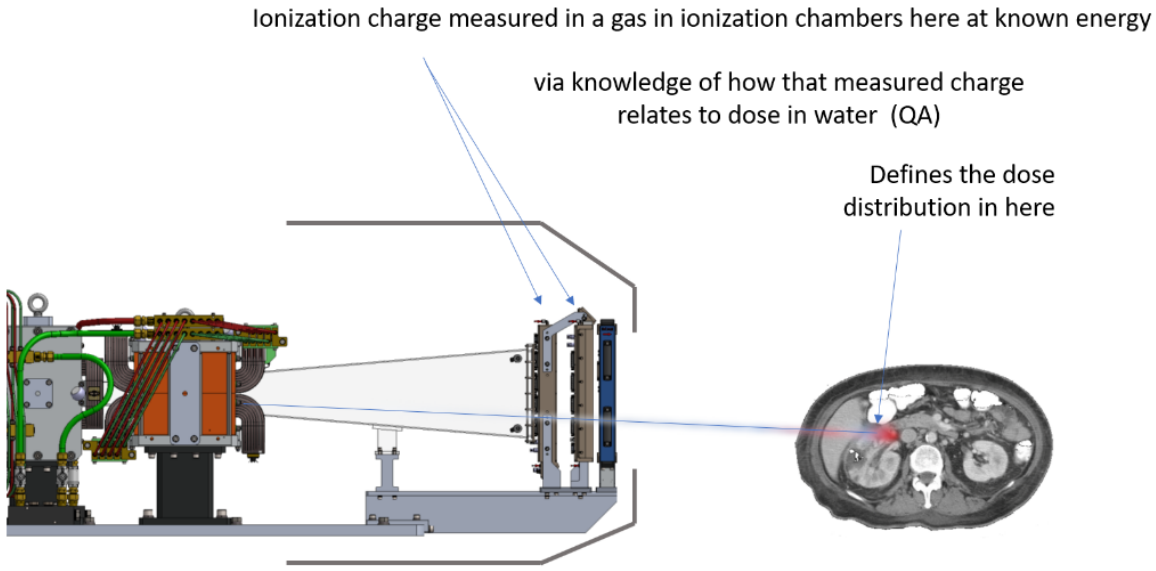
Since it is not possible to directly measure the dose delivered to a radiotherapy patient during treatment, alternative measurements are used as indicators of dose.

- The most common measurement is charge collected by an ionization chamber that the beam passes through before reaching the patient.
- Prior to treatment, the beam is calibrated using a water phantom containing traceable standard detectors, which serve as a patient surrogate.
- Treatment planning software accounts for differences between the water phantom and the actual patient.

Dose is expressed in Monitor Units (MU), which factor in:

- The dose delivered to the patient.
- The ionization chamber gain.
- The beam delivery configuration.

During quality assurance (QA) checks, the correct dose distribution in the water phantom is verified. This ensures that the measured charge from the ionization chamber can be reliably correlated with MU, allowing the device to receive and control dose delivery in MU.



49 : Pyramid's Proton Therapy Scanning Nozzle

Pyramid Nozzle Background

The Pyramid Nozzle System utilizes the Dose Controller to implement a step-and-shoot dosimetry system, ensuring precise and controlled dose delivery.

Step-and-Shoot Dosimetry Process:

1. A map of control points is loaded onto the device, each defining a target dose for a specific spot.
2. The software sequentially delivers the dose for each spot.
3. Dose delivery is gated by an external process that moves the beam. The dose controller is informed of beam positioning being complete using the `in_sync` process signal.
4. Safety conditions are enforced to verify accurate dose delivery.

The Dose Controller provides all the necessary features to fully automate and control this process, ensuring safe and effective dose administration.

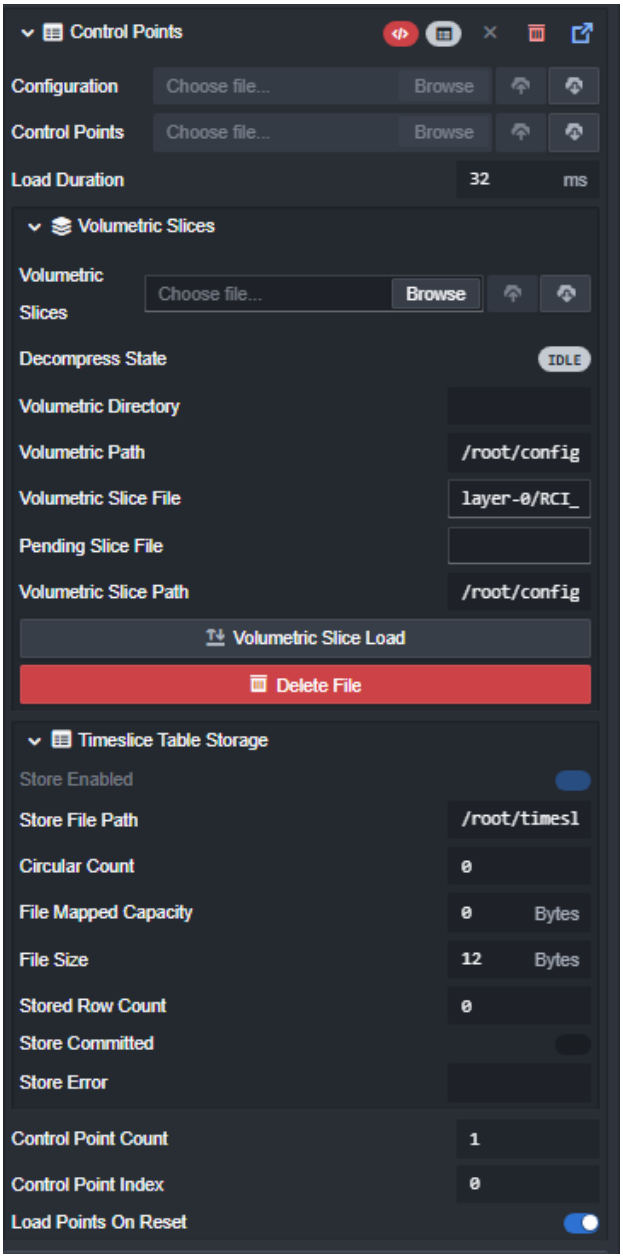
12.9.2 Control Point Table

The Control Point Table is a structured set of rows and columns, where:

- Each row represents an individual control point.
- Each column corresponds to a specific parameter associated with that control point.

This system allows the Dose Controller to receive a predefined sequence of target parameters, such as dose prescription (target dose), and efficiently execute the entire series of control points in rapid succession.

This feature is particularly useful for complex dose delivery plans where multiple target points need to be processed with minimal delay.



50 : Control Point Configuration GUI

- **Configuration:** Upload or download an XML file that defines how columns in the control point table are structured and stored.
- **Control Points:** Upload or download a CSV file containing control point data. If the input CSV does not match the XML configuration, the device will set the table upload state to error (red).
- **Volumetric:** Allows uploading a `.tar.gz` file containing multiple CSV files, which can be programmatically switched between using external software. This enables **preloading volumetric data** for fast layer switching during treatment.
  - **Volumetric Slices:** Upload or download a `.tar.gz` file containing the CSV files for volumetric dose delivery.
  - **Volumetric Path:** The file path where the decompressed directory of volumetric slice files is stored.
  - **Volumetric Slice File:** The file path of the active layer file. External software should set this string to the desired layer or use the Pending Slice File IO.
  - **Pending Slice File:** A path to the **next file** to be loaded after the current one completes.
    - If set to a valid path, the controller will immediately load this file upon reset.
    - The pending slice path will then be cleared automatically.
  - **Volumetric Slice Path:** A **readback** of the full path of the currently loaded slice file.
  - **Volumetric Slice Load:** Pressing this button (manually or programmatically) will **force** the Volumetric Slice File to load.
  - **Delete File:** Deletes all CSV and `.tar.gz` files from the device.
- **Timeslice Table Storage:** Contains readback values related to the timeslice table (described below).
- **Control Point Count:** Displays the number of currently loaded control points.
- **Control Point Index:** Indicates the most recent control point being actively processed. This can be used to **track the progress** of a dose session.
- **Load Points on Reset:** When set to true, the controller will automatically reload the last uploaded control point table upon reset. Useful for repeatedly delivering the same dose map without re-uploading.

12.9.3 Timeslice Table

The Timeslice Table records periodic time slices during a session, providing a detailed snapshot of system behavior at regular intervals. This feature is primarily used for debugging and analysis, helping developers better understand system performance and diagnose issues.

This data can be used to analyze session behavior and identify potential improvements or issues.

51 : Timeslice Debug IO

Important Notes

- The Timeslice Table is not configurable at this time.
- The GUI provides developer debugging I/O, which assists Pyramid's development team in future enhancements.

While not directly user-configurable, the timeslice table serves as an important diagnostic tool for system performance evaluation and future software development.

13 IGX User-Managed System Care

This section of the manual is designed to assist users who wish to service their IGX devices independently. While user-managed system care can be a rewarding and cost-effective way to maintain your device, it is essential to follow the guidelines provided in this manual to ensure the safety and longevity of your IGX device. Should you require assistance with any of these advanced service mode activities, Pyramid's expert technicians are always available to help.

Before attempting any user-managed system care tasks, make sure to review the safety precautions outlined in the manual, and ensure that you have the necessary tools, equipment, and knowledge to perform the tasks safely and effectively. Additionally, it is essential to work in a clean, well-lit, and organized environment to avoid potential damage to your device.

Here are some general guidelines to follow when performing user-managed system care on your IGX device:

1. **Regular Inspection**: Regularly inspect your IGX device for any signs of wear, damage, or loose connections. This can help you identify potential issues before they become more significant problems. Be sure to consult the device's user manual for specific inspection intervals and procedures.
2. **Firmware Updates**: Keep your IGX device up-to-date with the latest firmware releases from Pyramid. Regular updates can enhance the performance, stability, and security of your device. Refer to the manual for instructions on how to check and install firmware updates. Some medically controlled systems will be restricted to certain versions of IGX that have more formal testing. Please consult your Pyramid representativity in this case.
3. **Cleaning**: Keep your IGX device clean and free of dust, debris, and other contaminants. Regular cleaning can improve the device's performance, prevent overheating, and reduce the risk of damage. Be sure to follow the recommended cleaning procedures outlined in the user manual.
4. **Component Replacement**: Over time, some components in your IGX device may need to be replaced due to wear or damage. Always use genuine Pyramid replacement parts to ensure optimal performance and compatibility. Consult the user manual for guidance on identifying and replacing worn or damaged components.
5. **Troubleshooting**: If you encounter any issues with your IGX device, consult the troubleshooting section of the user manual for possible solutions. If the issue persists, don't hesitate to contact Pyramid's support team for assistance.

Remember, while user-managed system care can be an effective way to maintain your IGX device, Pyramid's team of skilled technicians is always available to provide support and assistance when needed. If you are unsure about any aspect of servicing your device, it is best to consult with Pyramid's experts to ensure the safety and longevity of your investment.

13.1 IGX SD Card Flashing Guide

13.1.1 Introduction

This guide will walk you through the step-by-step procedure for flashing an SD card. Flashing refers to writing the necessary firmware and data to the SD card so that it can be used with Pyramid devices. This procedure is useful both when you're creating an SD card for the first time and when you're updating an existing card.

Our SD cards contain the essential firmware and data that Pyramid devices rely on for their proper operation.

You may need to create a new SD card if you're setting up devices for manufacturing, or if you're trying to recover a card that is damaged beyond simple repair.

13.1.2 Get the Micro SD Card

Any micro-SD card with a capacity of 32GB should be suitable for use. For optimal performance, we recommend using a micro-SDHC UHS-I Class 10 U1 card, specifically one made by Samsung. Please note that as SD card technology evolves rapidly, the specific card model we recommend may change over time. Manufacturers may discontinue or stop supporting older card models, so it's important to check for the most current recommendations if you're purchasing a new card.

13.1.3 Get the Image File

The first thing you'll need is a file called an "image." The term "image" comes from the fact that it is a snapshot of a device's state at a particular moment in time.

An image file is essentially a copy of the contents from one SD card, stored as a single file. This file contains everything needed for the device to operate correctly.

While the file extension for many images is commonly '.iso', Pyramid image files are typically compressed using gzip compression. As a result, Pyramid image files have the extension **.iso.gz**.

Important: If your image file is compressed (i.e., it ends in .iso.gz), **do not decompress it** before flashing the card. The flashing process can handle the compression automatically.

Additionally, **do not attempt to flash an image from a network drive**. In our experience, we've encountered issues when trying to flash images directly from network locations. Always ensure that the image file is stored locally on your computer or device before starting the flashing process

13.1.4 Get the Required Flashing Software

There are several programs available for flashing SD cards, but we cannot guarantee that all of them will work correctly. For instance, we have encountered issues with Win32 Disk Imager, where it sometimes fails to flash the card properly and doesn't report the failure correctly.

To ensure consistency and standardization, everyone at Pyramid uses the same tool called **Etcher**.

Download Etcher Here[16]

Etcher is a reliable tool that works on **Windows**, **Linux**, and **Mac** operating systems. It offers several key benefits:

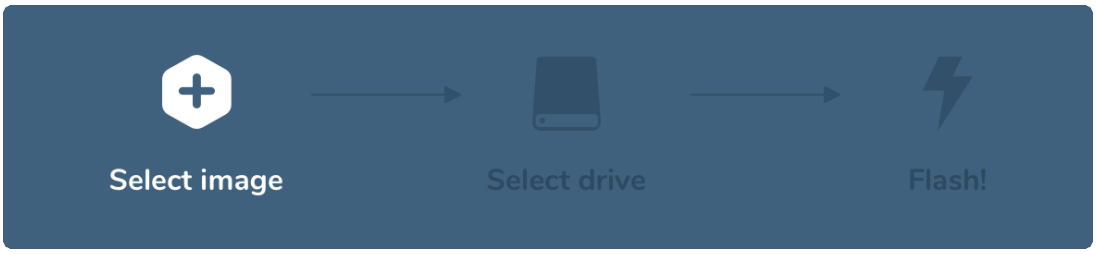
- **Speed**: It flashes SD cards quickly.
- **Convenience**: It automatically handles compressed images, so you don't need to configure anything.
- **Validation**: After flashing, Etcher validates the card to confirm that the process was successful, and that the data was written correctly.
- **User-friendly Interface**: Etcher features a simple, clean interface, making it easy for anyone to use, even those who are not familiar with flashing tools.

13.1.5 Flash the Card

Start by launching Etcher on your computer. Once it's open, select the image file that you previously downloaded (the .iso.gz file). Navigate to the file's location on your computer's file system and choose it.

Next, insert your SD card into the computer. Depending on the type of ports your computer has, you might need a USB adapter or a full-size SD card adapter to connect the card properly.

16. https://www.balena.io/etcher/



Etcher should automatically detect the SD card once inserted. If multiple drives are connected to your computer, you may need to manually select the correct one to ensure you're flashing the right card.

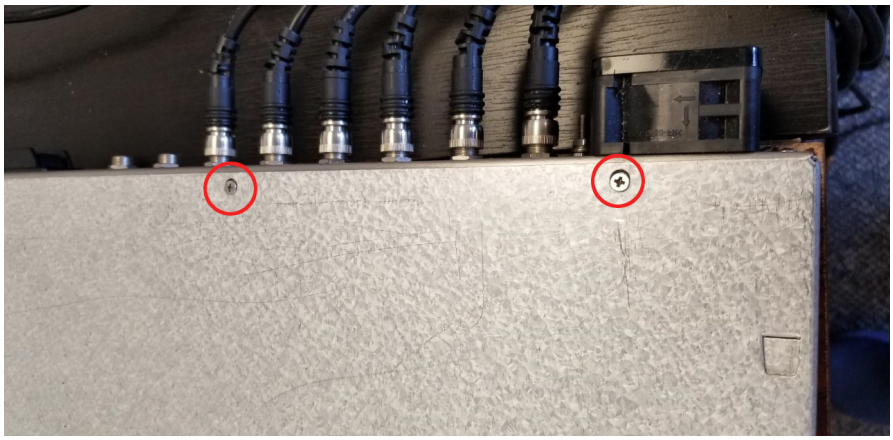
Once the SD card is correctly selected, click the **"Flash"** button to begin the process. The flashing will take several minutes, so feel free to take a break and grab a coffee while you wait.

When the flashing process is complete, Etcher will notify you. If everything worked correctly, your SD card is now ready for use in your Pyramid device.

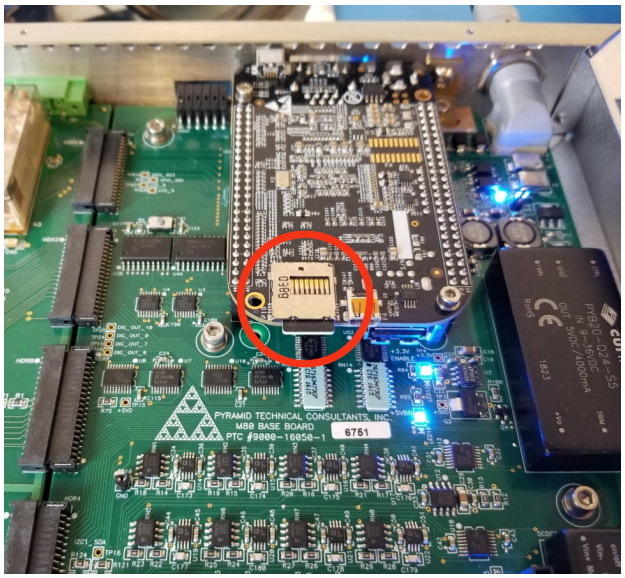
If Etcher reports an error, don't worry. Simply try flashing the card again, or if the issue persists, try using a different SD card. Occasionally, a faulty card may cause the process to fail.

13.1.6 Card Installation or Removal

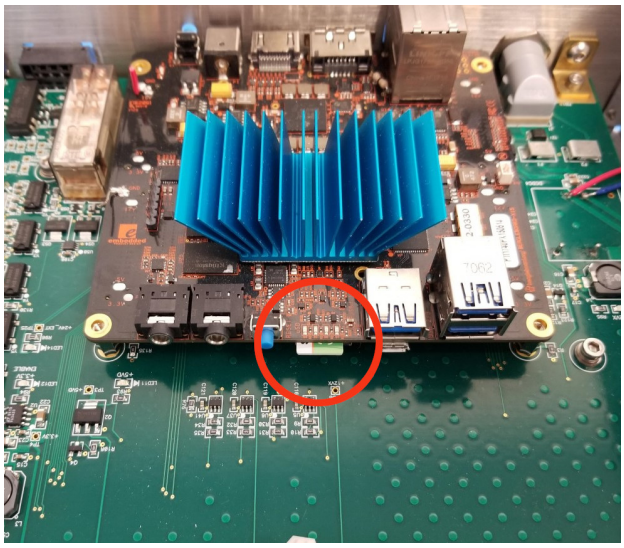
Before you can install or remove an SD card from a Pyramid device, you'll need to access the SD card reader and control board by removing the top panel of the device. For our **1U** or **2U rack-mounted devices**, this can be done by unscrewing the four small Phillips-head screws located at the rear top of the device. Once the screws are removed, gently slide the cover off by pushing it towards the rear of the device.



The location of the SD card reader can vary depending on the specific device, but it will generally resemble the examples shown below.



52 : SD card on a BBB



53 : SD card on an X15

To remove the SD card, gently press it inwards, as if you were pressing a button. This action will cause the card to pop back out, making it easy to remove.

To install a new SD card, simply reverse the process: gently press the card into the slot, and it will click into place, securely locking it.

13.1.7 Reboot the Device

Once the new SD card is installed, you can power on the device as usual. If the device has an Ethernet port, you should see the LEDs on the port light up and flicker once an Ethernet cable is connected. This indicates that the device is communicating over the network and is ready for use.

13.2 IGX SD Card Imaging Guide

13.2.1 Summary

This document provides instructions for creating and compressing SD card disk images on Linux and Windows 10 systems.

For Linux, the procedure uses the `dd` and `gzip` command-line tools to create and compress the disk image. Users must identify the SD card device using the `lsblk` command and then execute the `dd` command to create the disk image with the `.iso.gz` extension.

For Windows 10, the process requires the use of Win32 Disk Imager to create the disk image and 7-Zip to compress it. Users must install and run Win32 Disk Imager, select the appropriate SD card device, and create the disk image with the `.iso` extension. Afterward, 7-Zip is used to compress the disk image into a `.iso.gz` file.

Both procedures require sufficient free space on the computer to store the disk image and its compressed version.

13.2.2 Linux Procedure

Creating a disk image of an SD card on Linux involves the use of two command-line tools: `dd` and `gzip`. These tools are typically pre-installed on most Linux distributions. The `dd` command is used for copying and converting data, while `gzip` is a data compression tool. In this procedure, you will create an image of the SD card and compress it as you go.

1. **Identify the SD card device**: To find the correct device file for your SD card, run the following command:

   ```
   lsblk
   ```

   This command lists all the block devices on your system. Look for the SD card in the list, typically named `mmcblk0` or `mmcblk0p1`. Note that the actual name may vary depending on your system.
2. **Create and compress the disk image**: Run the following command to create a disk image of the SD card and compress it using `gzip`:

   ```
   # Standard Command
   sudo dd if=/dev/mmcblk0 | gzip > filename.iso.gz
   # Command With Progress
   sudo dd status=progress if=/dev/mmcblk0 | gzip > filename.iso.gz
   ```

   Replace `/dev/mmcblk0` with the appropriate device file for your SD card and replace `filename` with your desired name for the disk image. This command may take around 20 minutes to complete, depending on the size of the SD card. Ensure that the destination for the output file has enough free space to accommodate the compressed image.

For more information on the `dd` command, refer to the official GNU Coreutils documentation[17]. For additional details on `gzip`, consult the gzip manual page[18].

13.2.3 Windows 10 Procedure

To create and compress an SD card disk image that results in a `.iso.gz` file on Windows 10, follow these steps:

1. **Download and install Win32 Disk Imager**: Download the latest version of Win32 Disk Imager from the official website[19] and install it on your computer.
2. **Insert the SD card**: Insert the SD card into your computer's SD card slot or an external card reader.
3. **Launch Win32 Disk Imager**: Run Win32 Disk Imager and select the appropriate drive letter for your SD card in the "Device" dropdown menu.
4. **Create the disk image**: Click on the folder icon next to the "Image File" field and choose a destination for the output file. Enter a file name with the `.iso` extension, for example, `filename.iso`. Click the "Read" button to start creating the disk image. This process may take some time, depending on the size of the SD card.
5. **Download and install 7-Zip**: Download and install the latest version of 7-Zip[20] if you don't already have it on your computer.
6. **Compress the disk image**: After the disk image has been created, you can compress it using 7-Zip. Right-click the `.iso` file, choose "7-Zip" from the context menu, and select "Add to archive...". In the "Archive format" dropdown, choose "gzip", and click "OK" to start compressing the image. Once the compression is complete, the file will have the `.iso.gz` extension, for example, `filename.iso.gz`.

Ensure that you have enough free space on your computer to store the disk image and its compressed version.

For more information about Win32 Disk Imager, visit the official documentation[21]. For further details on using 7-Zip, consult the 7-Zip help documentation[22].

17. https://www.gnu.org/software/coreutils/manual/html_node/dd-invocation.html
18. https://www.gnu.org/software/gzip/manual/gzip.html
19. https://sourceforge.net/projects/win32diskimager/
20. https://www.7-zip.org/
21. https://sourceforge.net/p/win32diskimager/wiki/Home/
22. https://www.7-zip.org/support.html

13.3 BeagleBone Black Replacement Guide

13.3.1 Purpose

This document provides step-by-step instructions for replacing the beagle bone found in some Pyramid devices. The beagle bone is a processing unit that Pyramid uses in its designs. If it becomes damaged, it is easy to replace as it is a standard component. This guide ensures that the replacement process is completed safely and correctly.

Replacing a beagle bone in a Pyramid device is a straightforward process, but it is essential to handle the components with caution and care. By following these instructions and taking the necessary precautions, you can successfully replace the beagle bone and ensure the continued operation of your device.

If you require further assistance or are unsure about any of the steps, it is recommended to contact Pyramid support or refer to their documentation. Contact us at support@ptcusa.com[23].

Procuring a BeagleBone Black

To procure a BeagleBone Black, users have several options. They can either purchase it directly from Pyramid or from BeagleBoard.org[24]. BeagleBone Blacks that are purchased from BeagleBoard.org[25] will not come with any SD card. If you are simply going to swap the SD card from your original unit, then this will not be a problem, but if you want a complete replacement, then we recommend buying from Pyramid.

Purchase from Pyramid

1. Email sales@ptcusa.com[26], and request a quote for a BeagleBone Black, pre-loaded with an SD card containing the software of your choice.
2. We will reach out to you to verify the software choice and complete the sales order.

Purchase from BeagleBoard.org

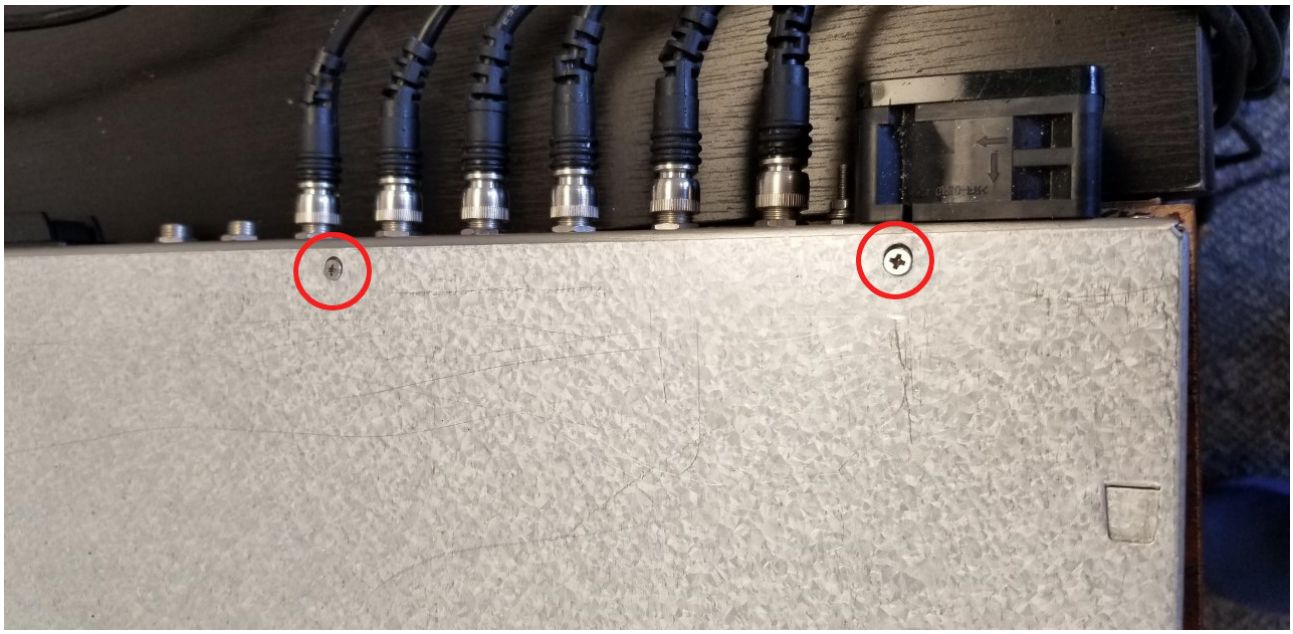
1. Visit the official website: https://beagleboard.org/black
   On the website, you'll find detailed information about the BeagleBone Black, its features, and specifications.
2. Click the "Buy Now" button to see a list of authorized distributors.
3. Choose your preferred distributor from the list and follow their purchase process.

Replacement Procedure

⚡ It is important to handle electronic equipment with caution. Be sure to use appropriate Electrostatic Discharge (ESD) protection, such as an ESD wrist strap, to avoid damaging the components.

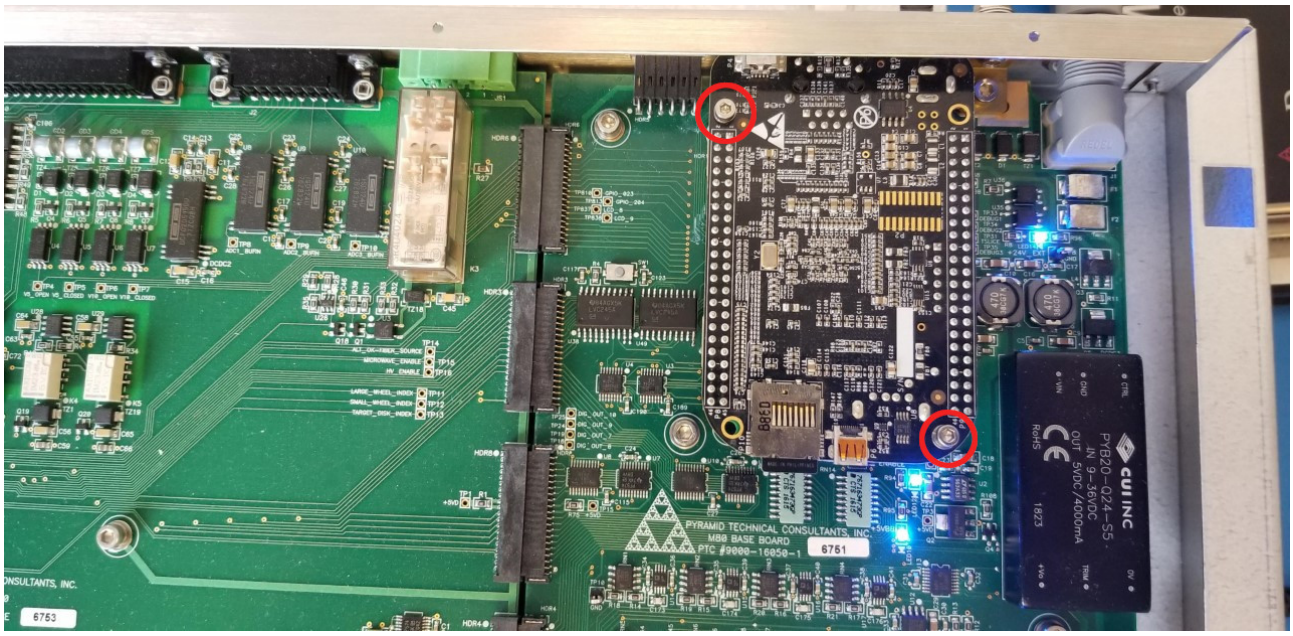
For our 1U or 2U rack mountable devices, the top panel can be removed by unscrewing the four small Phillips-head screws located on the top of the rear end of the device. Then slide the cover off by pushing back towards the rear end of the device.

23. mailto:support@ptcusa.com
24. http://BeagleBoard.org
25. http://BeagleBoard.org
26. mailto:sales@ptcusa.com



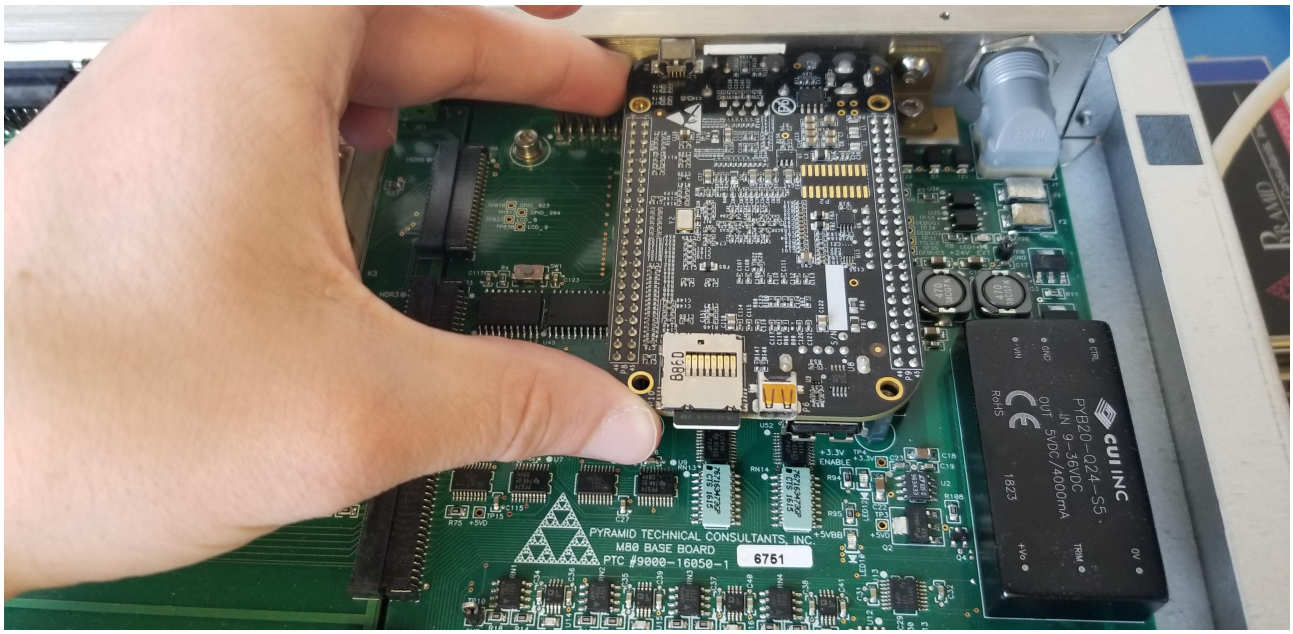
54 : Top Panel Screw Locations

Before the beagle bone can be removed, there will be two or four M3 hex screws that have to be unscrewed. These screws hold the beagle bone in place.



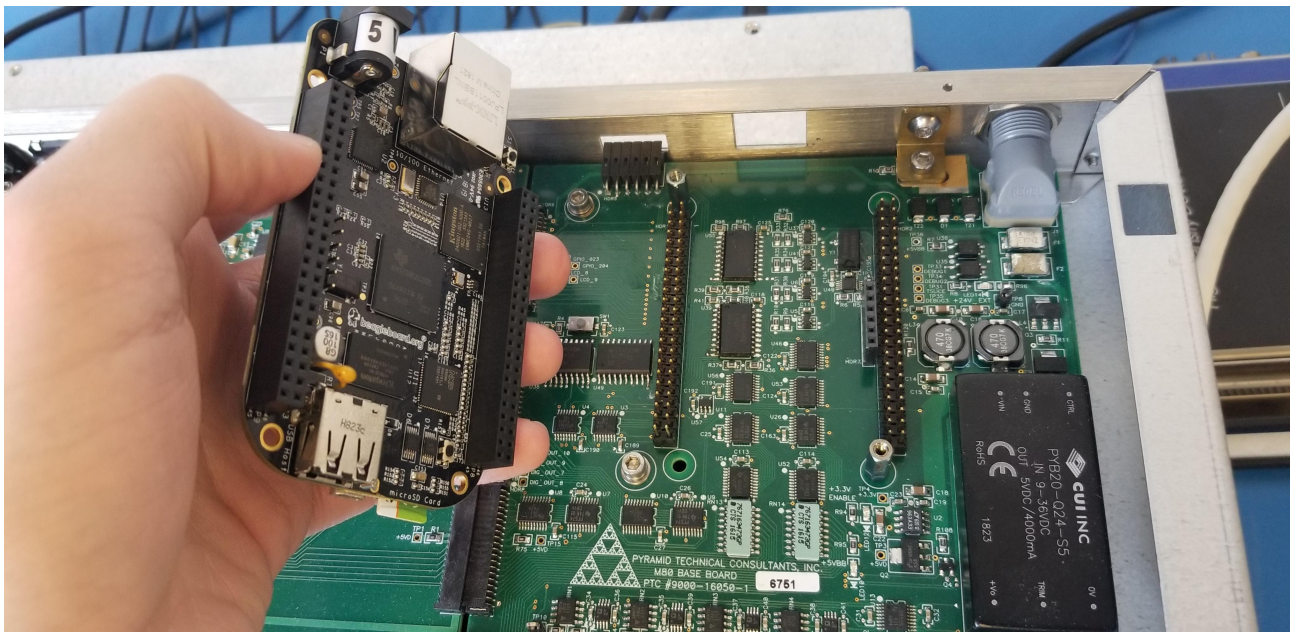
55 : Mounting Screw Locations

After unscrewing the mounting screws, grab the board by the corners, and pull it directly up. The fit may be tight, so you might have to wiggle the board slightly to remove it.



56 : Where to Grab the Device

Once the old board has been removed, make sure to inspect the pins on the main board. They should all be straight, and nothing should be missing. If there is any damage to the pins or sockets, they should be fixed before installing the new beagle bone.



57 : Inspecting the Connector Pins

Take the new beagle bone and put it into position. Press down directly on the corners, making sure that each pin is seated completely into the corresponding socket.

Screw back in the two or four M3 screws to hold the beagle bone in place. Do not over-tighten these screws as it could cause damage to the board.

Slide the top panel back on and re-screw the four small screws into place. Be careful not to over-tighten these screws, as they can strip very easily.