



# Cyclone **FX** Programmers

## User Manual



## Purchase Agreement

P&E Microcomputer Systems, Inc. reserves the right to make changes without further notice to any products herein to improve reliability, function, or design. P&E Microcomputer Systems, Inc. does not assume any liability arising out of the application or use of any product or circuit described herein.

This software and accompanying documentation are protected by United States Copyright law and also by International Treaty provisions. Any use of this software in violation of copyright law or the terms of this agreement will be prosecuted.

All the software described in this document is copyrighted by P&E Microcomputer Systems, Inc. Copyright notices have been included in the software.

P&E Microcomputer Systems authorizes you to make archival copies of the software and documentation for the sole purpose of back-up and protecting your investment from loss. Under no circumstances may you copy this software or documentation for the purpose of distribution to others. Under no conditions may you remove the copyright notices from this software or documentation.

This software may be used by one person on as many computers as that person uses, provided that the software is never used on two computers at the same time. P&E expects that group programming projects making use of this software will purchase a copy of the software and documentation for each user in the group. Contact P&E for volume discounts and site licensing agreements.

P&E Microcomputer Systems does not assume any liability for the use of this software beyond the original purchase price of the software. In no event will P&E Microcomputer Systems be liable for additional damages, including any lost profits, lost savings or other incidental or consequential damages arising out of the use or inability to use these programs, even if P&E Microcomputer Systems has been advised of the possibility of such damage.

By using this software, you accept the terms of this agreement.

©2015-2023 P&E Microcomputer Systems, Inc.

ARM and Cortex are registered trademarks of ARM Ltd. or its subsidiaries.

NXP, ColdFire, and Kinetis are registered trademarks of NXP Semiconductors.

Texas Instruments and TI are registered trademarks of Texas Instruments Incorporated.

STMicroelectronics is a registered trademark of STMicroelectronics, Inc.

macOS is a registered trademark of Apple, Inc.

All other product or service names are the property of their respective owners.

P&E Microcomputer Systems, Inc.  
98 Galen St.  
Watertown, MA 02472  
617-923-0053  
<http://www.pemicro.com>

Manual version: 3.2c

August 2023

# 1 INTRODUCTION

PEmicro's **Cyclone FX** production programmers are powerful, fast, and feature rich in-circuit programming solutions. PEmicro offers two models which have the same feature set and only vary by the devices supported.

Part# CYCLONE-FX-ARM supports a wide variety of ARM Cortex devices.

Part# CYCLONE-FX-UNIV supports those ARM Cortex devices as well as the following NXP device families: Kinetis, LPC, S32, MPC55xx-57xx), MPC5xx/8xx, DSC, S12Z, RS08, S08, HC08, HC(S)12(X), ColdFire. It also supports Infineon's TriCore™ (DAP only - AUDO™ TC1xx and AURIX™ TC2xx/TC3xx) and STMicroelectronics' SPC5 & STM8 (with STM8 adapter). A separate Renesas adapter additionally allows support for Renesas' R8C, H8, H8S/Tiny, M16C, M16C80, M32C, RL78, RX, RX63T, and RH850 devices.

**Cyclone FX** programmers are designed to withstand the demands of a production environment. They are Stand-Alone Programmers (SAP) that can be operated manually or used to host automated programming. In manual SAP mode the Cyclone is operated using the touchscreen LCD Menu and/or the Start button. Host-controlled SAP mode, for automated programming, is accomplished using the Cyclone Control Suite. **Cyclone FX** programmers also include ProCryption Security features. This allows users to create RSA/AES encrypted programming images that use their own uniquely generated ImageKey. These encrypted images may only be used to program when on Cyclones that are also pre-configured with the same ImageKey. This keeps the user securely in control of both their IP and the programming process. See **CHAPTER 12 - SAP IMAGE ENCRYPTION**.

The example above shows the rear label for Cyclone LC Universal Rev. B, date code 0423. As the date code 0423 is later than the 0620 date code, this Cyclone would not include the SDHC hardware, even if the case had the corresponding slot.

## 1.1 Supported Devices

An up-to-date list of devices supported by PEmicro's Cyclone programmers, including vendors/families, is available at PEmicro's website on the Cyclone product page, under the tab "Choosing a Cyclone Model": [pemicro.com/cyclone](http://pemicro.com/cyclone).

For ARM devices specifically: if PEmicro supports an ARM Cortex-M device then it is supported by all models of Cyclone. A list of ARM device manufacturers where users can locate their specific supported device part number is available at: <https://www.pemicro.com/partners/index.cfm>

## 1.2 PEcloud: Secure Management of Remote Programming

PEcloud is a cloud-based programming job management tool that enables customers to monitor usage of and exercise additional control over their production programming images - in real-time, anywhere in the world. PEcloud users create programming jobs and manage them via this easy-to-use service. Jobs being programmed in remote manufacturing locations can be paused, deleted, updated, and their programming logs can be inspected. There currently is no cost to use PEcloud.

**Note:** Cyclone LC programmers are provided with ProCryption Security features (part# LIC-CYC-PROCRYPTION) when used with PEcloud, due to its stringent security requirements.

To learn more about the PEcloud tool, please refer to **CHAPTER 10 - PEcloud - SECURE MANAGEMENT OF REMOTE PROGRAMMING**.

## 1.3 Distinguishing Windows-Only And macOS/Linux-Specific Content

Many control and automation features offered by the Cyclone Control Suite are available for macOS and Linux platforms, in addition to Windows. See **CHAPTER 8 - CYCLONE PROGRAMMER AUTOMATED CONTROL**. However, some elements such as Image Creation remain Windows-only.

The matrix below indicates at a high level which Cyclone features are available for each platform:


	Windows	macOS	Linux
Cyclone Control GUI	x	x	x
Cyclone Control Console	x	x	x
Cyclone Control SDK	x	x	x
Image Creation Utility	x		
SAP Image Compiler	x		
Initiate Programming with Barcode Scanner	x		
Automatic Serial Number Mechanism	x		

**Table A-1. Cyclone Feature Support**

In this user manual, information that pertains to the Windows platform only, or is particular to macOS/Linux, is indicated by the symbols below. Content can be considered to apply to all three platforms unless otherwise specified.

Windows-only: 

macOS: 

Linux: 

### 1.3.1 Minimum Required Versions

The following are the minimum versions required for macOS/Linux support.

macOS: 10.13 High Sierra

Linux: 18.04 LTS

## 2 QUICK START GUIDE FOR SAP OPERATION

This guide will allow the user to set up and program a simple Stand-Alone Programming (SAP) image with the Cyclone by completing the following steps.

- Installing the Cyclone software
- Setting up the Cyclone hardware
- Creating a stand-alone programming image
- Deploying the image to a Cyclone
- Launching Cyclone programming

This guide is intended as a supplement to the Cyclone's User Manual, which contains in-depth information about the topics covered here and much more.

**Note:** There have been significant changes in the image creation and deployment process, including a revised Cyclone Image Creation Utility, and the addition of a new Build & Deploy Utility to the previous workflow. Users who are familiar with previous versions of the Image Creation Utility should be sure to note these changes.

### 2.1 Installing The Cyclone Software

First, the Cyclone software should be installed on the user's PC. It can be downloaded from the Support & Downloads tab on the pemicro.com Cyclone product page, or directly from [https://www.pemicro.com/downloads/download\\_file.cfm?download\\_id=481](https://www.pemicro.com/downloads/download_file.cfm?download_id=481).



For macOS/Linux, download the .TGZ file from the Support & Downloads tab, or directly from [https://www.pemicro.com/downloads/download\\_file.cfm?download\\_id=577](https://www.pemicro.com/downloads/download_file.cfm?download_id=577).

Once the software is downloaded, the user should install it on their PC. If Cyclone software is already installed on the PC, it is recommended that the old installation be removed before the user installs the latest software.

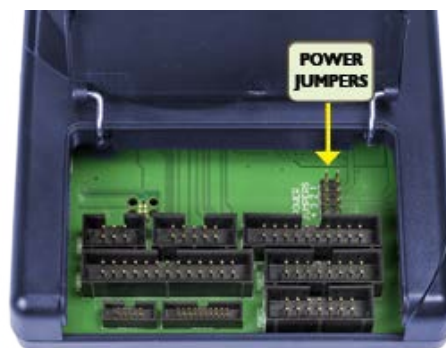
### 2.2 Setting Up The Cyclone Hardware

#### Step 1. Configure Cyclone power settings

The Cyclone has several different power configurations. The label on the bottom of the Cyclone indicates the appropriate Jumper settings for each. The user should install the Jumpers as indicated for their desired power configuration.

The Jumpers are located underneath the Cyclone's access panel. They are labeled "Power Jumpers." and numbered from 1-4. The Cyclone-LC-ARM is shown in the example below; the jumper location will be similar for all Cyclone models.

Power Management JPR# 4 3 2 1	
Target Power IN -> Target Power OUT	8 . . .
Internal Power -> Target Power OUT	. . 8 8
Internal Power -> Target Power Pin	. . 8 8 8
Target Power IN -> Target Power Pin	8 8 8 .
Target Powered Independently	. . . .



If power is provided via the Cyclone, the user may need to configure the programming image accordingly. Image creation and configuration is discussed in **Section 2.3 - Creating A Stand-Alone Programming Image**.

For more information on the various power configurations, the user should refer to their Cyclone's User Manual. There is also a blog post that covers this topic at: [http://www.pemicro.com/blog/index.cfm?post\\_id=121](http://www.pemicro.com/blog/index.cfm?post_id=121)

## Step 2. Connect Cyclone to a PC (for programming image setup)

The Cyclone programmer should be connected to the PC via USB, Serial, or Ethernet. Cables for each of these options are included with the Cyclone.

**Note:** An Ethernet connection requires IP setup on the Cyclone unit; please refer to the Cyclone's User Manual for more information.

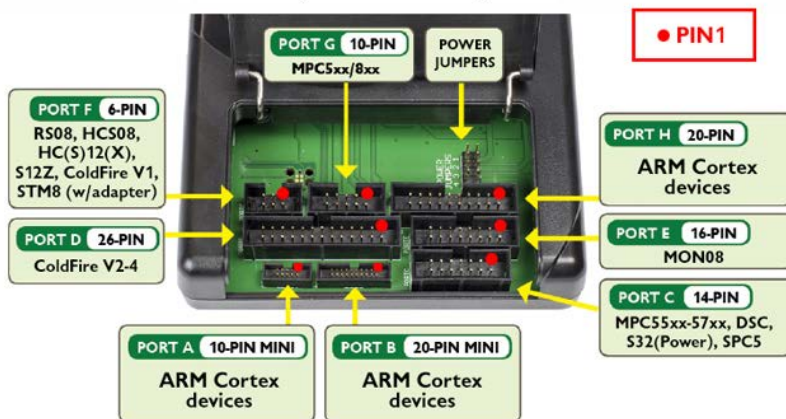


## Step 3. Connect Cyclone to target

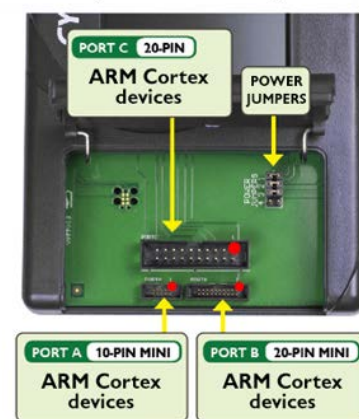
A ribbon cable should be connected from the appropriate Cyclone header (located under the Cyclone's access panel) to the header for your target device. Ribbon cables are provided with the Cyclone.



## Cyclone Universal & Cyclone FX Universal (FX Model Shown)



## Cyclone ARM & Cyclone FX ARM (FX Model Shown)



### Step 4. Plug in power to the Cyclone

The provided power supply should be plugged into the System Power jack of the Cyclone programmer. Other power connections should be made according to the power configuration selected in Step 1.



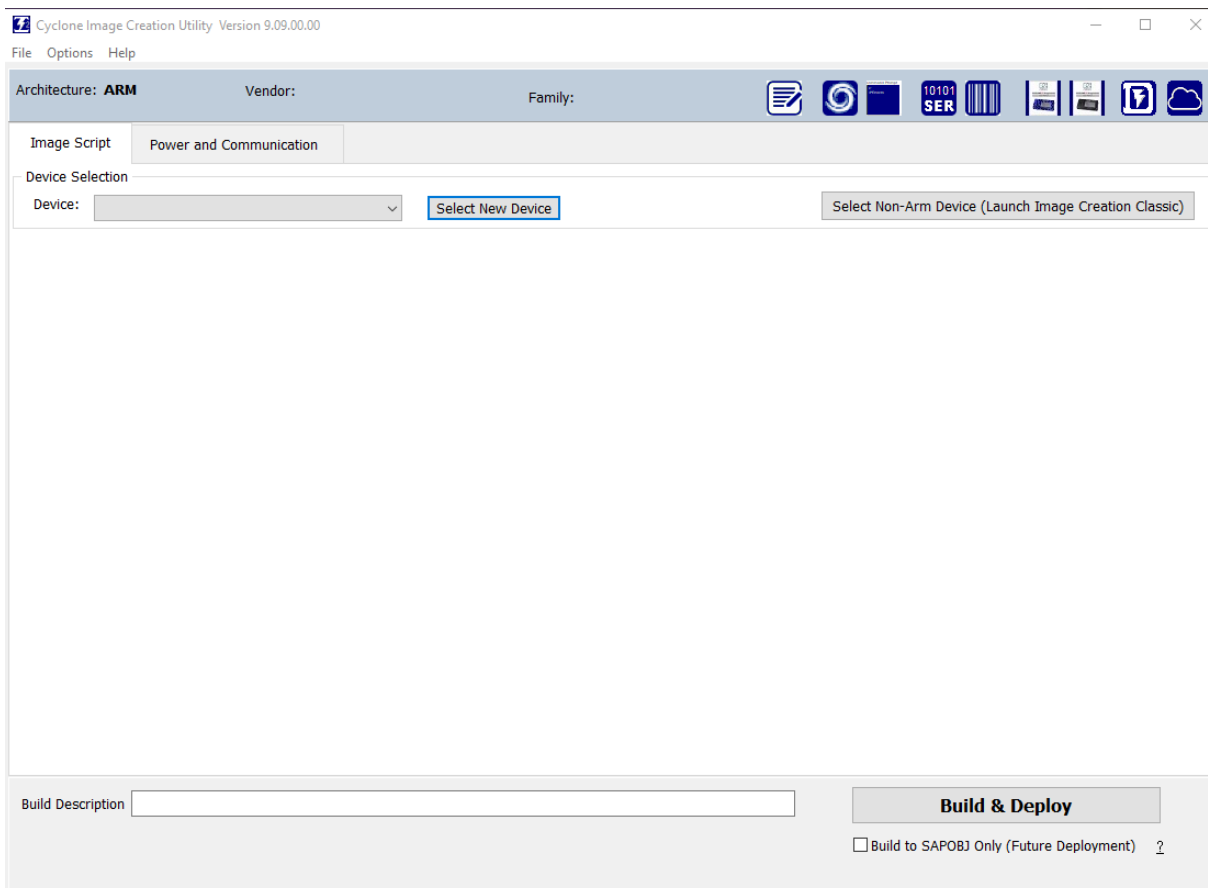
On power-up the user may need to agree to a firmware update on the Cyclone unit.

## 2.3 Creating A Stand-Alone Programming Image

A stand-alone programming (SAP) image is the result of pre-processing the programming algorithms, data to be programmed, programming options, and scripted programming commands. These are combined into a single encrypted file. This SAP image can then be loaded onto the Cyclone and used to program, without need for the Cyclone to be connected to a PC.

The Cyclone Image Creation Utility, shown below, allows the user to configure and save SAP images. A simple programming image can be created in 6 steps:

- Step 1. Run Cyclone Image Creation Utility**
- Step 2. Select Device**
- Step 3. Set Up Programming Sequence**
- Step 4. Add Basic Programming Commands**
- Step 5. Configure Additional Settings**
- Step 6. Enter The Build Description**



The following instructions walk the user through each of these steps:

### Step 1. Run Cyclone Image Creation Utility (ARM devices)

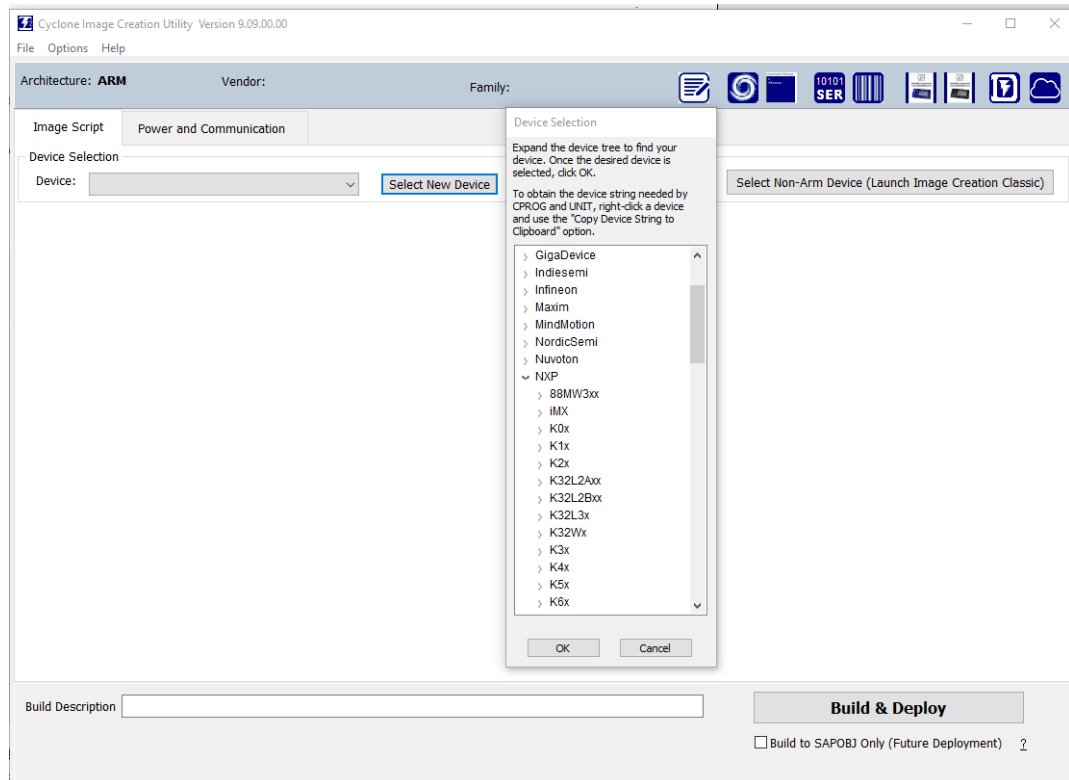
*CreateImage.exe* is in the “ImageCreation” folder, in the location where the Cyclone software was installed. For an in-depth description of the Cyclone Image Creation Utility, see **Section 6.1 - Cyclone Image Creation Utility**.

**Note:** For non-ARM devices the user should instead run *CreateImageClassic.exe*. This utility is not documented separately as it is extremely similar to *CreateImage.exe*, just organized differently. Please follow the same concepts and procedures as below.

### Step 2. Select Device

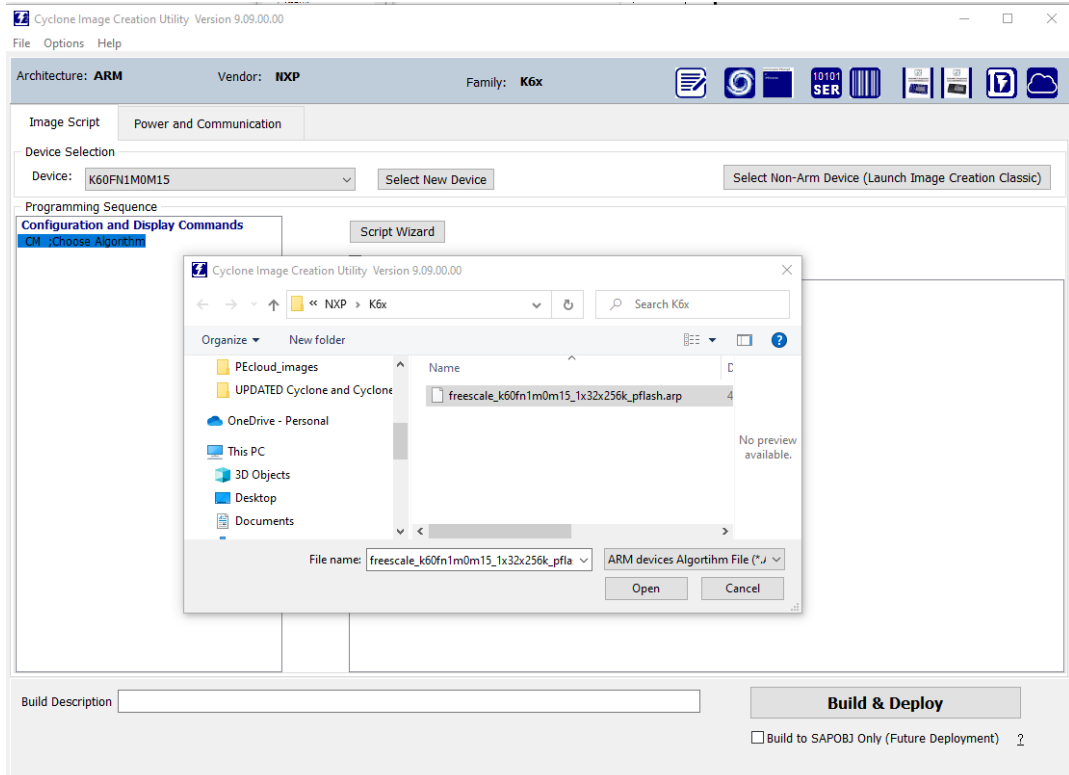
Select New Device is used to choose the target device, and then the specific device or architecture.





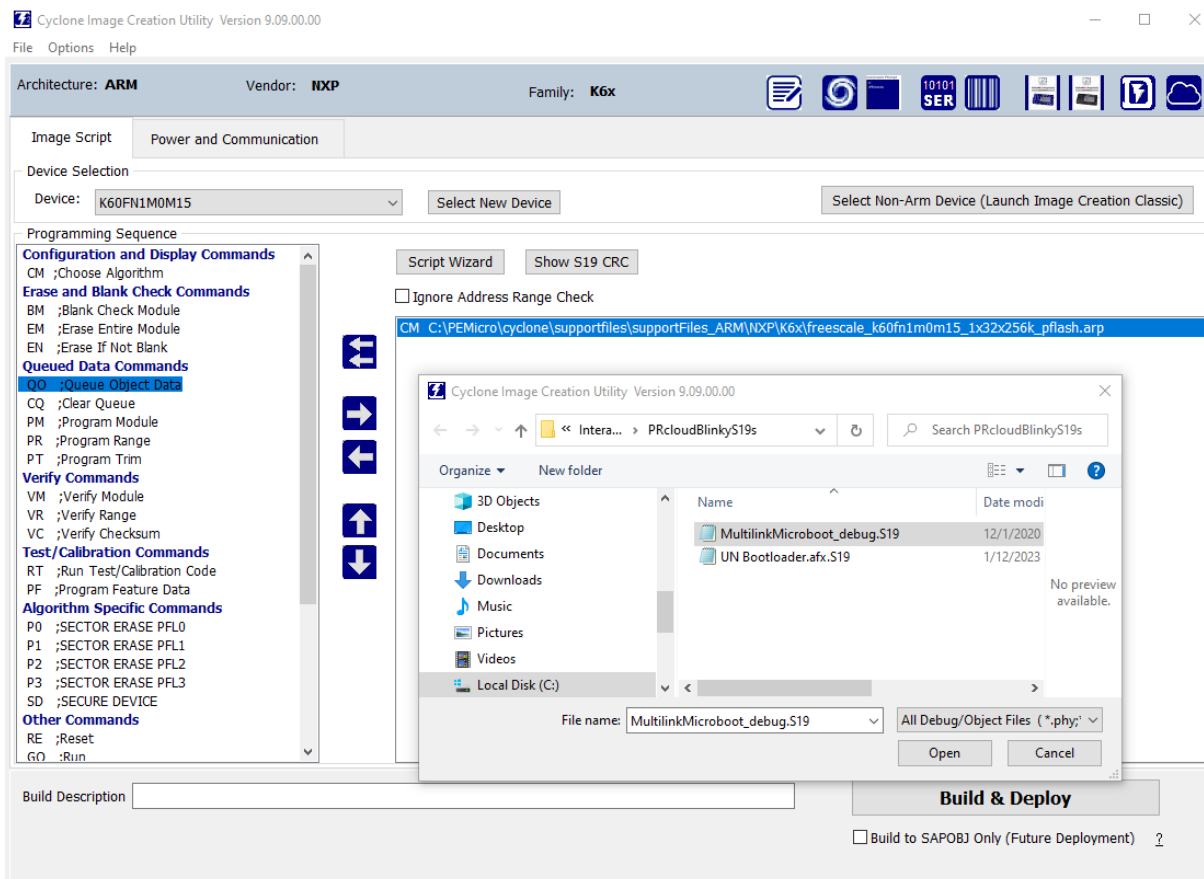
### Step 3. Programming Sequence Setup

The user should double-click on CM in the Programming Sequence window to choose the appropriate Algorithm for the target device. They can navigate to the algorithm using the dialog provided.



Based on the algorithm that was selected, additional commands will be made available in the box of programming commands on the left.

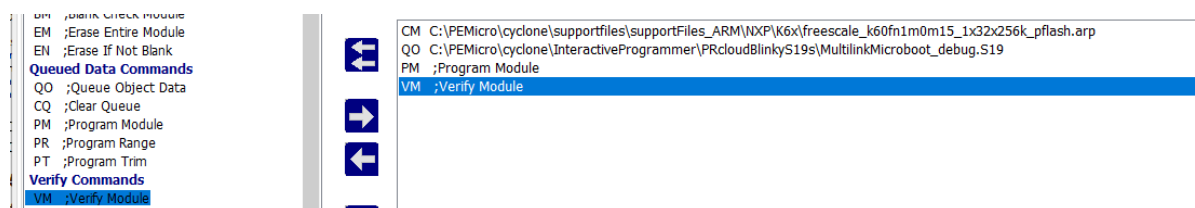
The user should then double-click on the QO command to add an Object File to the Queue.



#### Step 4. Adding Basic Programming Commands

The user should then add other basic programming commands, using the list of commands on the left side of the Programming Sequence area. The arrow and buttons allow the user to add, remove, and re-sequence the commands, in the box on the right. As an example, some basic commands might be

- Erase (if needed)
- Program
- Verify (if needed)



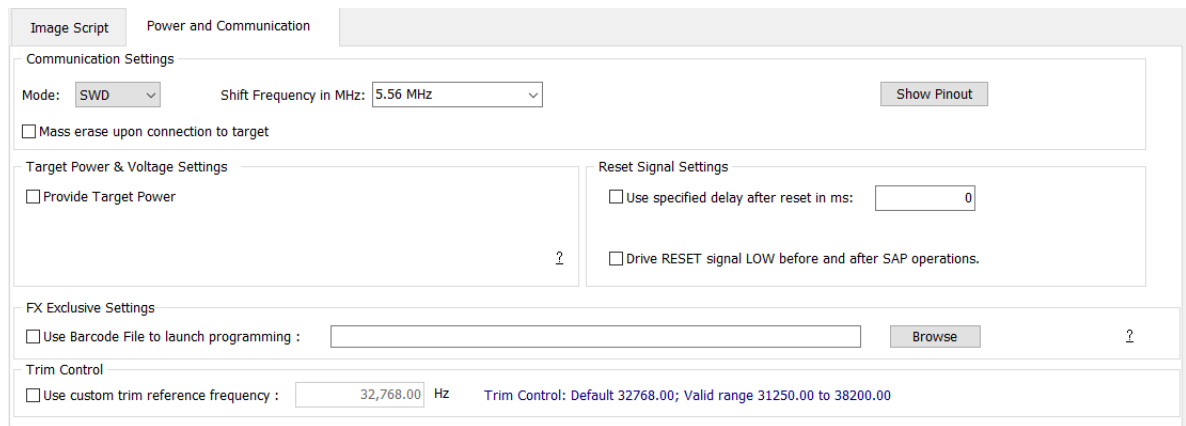
**Note:** Launch Script Wizard can also be used to quickly complete Steps 3 and 4 (not available for NXP i.MX devices).

#### Step 5. Power & Communication Settings

Click into the Power and Communications tab (to the right of the Image Script tab). Here the user should then specify any other settings that the SAP image should contain, if applicable, in order to program correctly, such as:

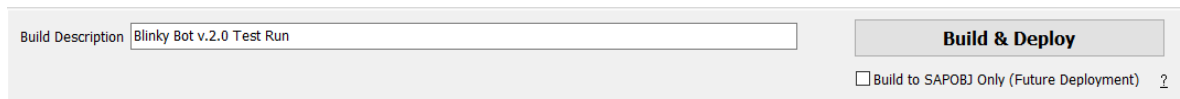
- Communication SWD vs JTAG
- Shift frequency
- Target Power and Voltage Settings
- Reset Signal Settings

- Trim Control



## Step 6. Enter The Build Description

The user must enter a description of this Build in the Build Description field before proceeding.



## 2.4 Deploying The Image To A Cyclone

Now that the user has completed the basic configuration of the image they can move on to the Deploy phase, which allows additional configuration and actual deployment of the Image.

### a. Launch the Image and Job Deployment Form

To complete basic configuration and then build and deploy the programming image, click the “Build & Deploy” button in the lower right corner of the Image Creation Utility. This launches the Image and Job Deployment Form.

### b. Select Build/Deploy Method

For the purposes of this Quick Start Guide, in the Generate section of the Deploy tab, select “Deploy Programming Image.” The option in the other sections here are beyond the scope of this guide and are explained in later chapters of user manual.

### c. Add Image Description

The user will find the Image Description box at the bottom of the display. It is populated by the Build Description text that was entered previously. The user may alter it if they wish to give the Image a different name.

### d. Deploy image to Cyclone

For this quick start guide we will save the programming image to the Cyclone’s internal memory. Make sure that the “Deploy Image to Cyclone” checkbox is checked.



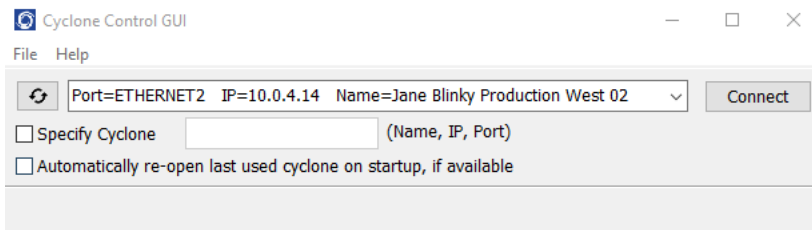
### e. Deploy - Launch Cyclone Control GUI

Click the Deploy button in the lower right corner. This will launch the Cyclone Control GUI which will be used to deploy the SAP image.

### f. Connect to a Cyclone

The top section of the Cyclone Control GUI displays a drop-down box which should

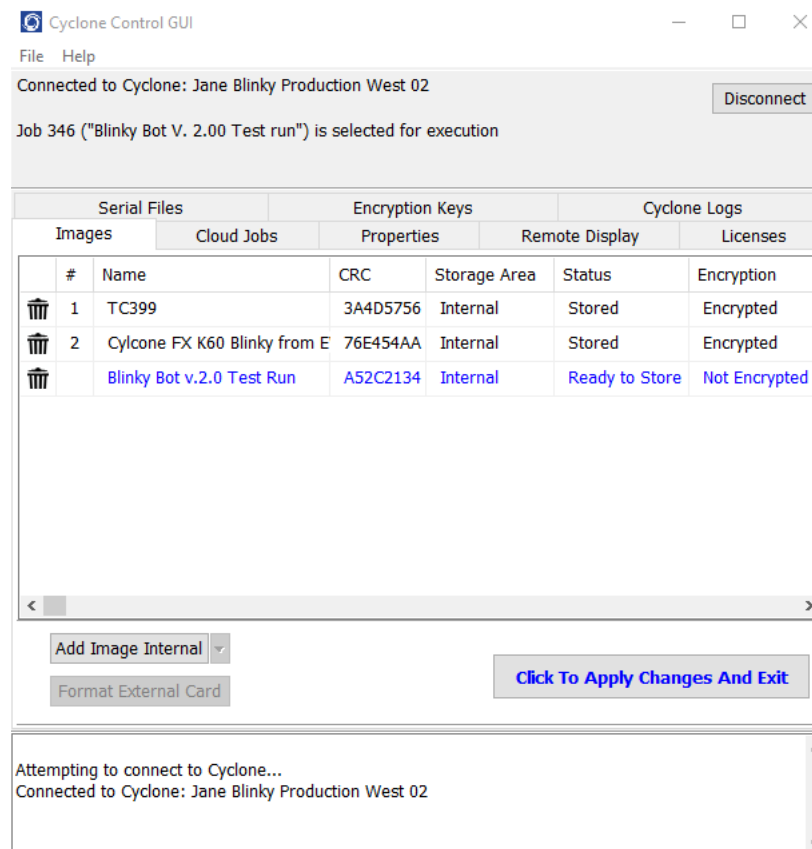
contain any Cyclones that are detected.



Select your Cyclone from the list and click Connect. If your Cyclone is not displayed in the drop-down list, please make sure that you have correctly followed the instructions above to connect your Cyclone, and make sure that the Cyclone is powered on. Use the button on the left to refresh the display.

#### g. Build & Deploy

Once connected to the Cyclone, the Images tab of the Cyclone Control GUI displays any images currently stored in the Cyclone's memory. The image that the user is saving will be displayed at the bottom of the list and highlighted in blue text, with a Status that reads "Ready to Store".



Click the button that says "Click To Apply Changes and Exit." Congrats! Barring any errors, the Cyclone Control Gui will close and the image will be successfully stored on the Cyclone, ready to be programmed.

### 2.4.1 Advanced Features

Depending on their model/licenses, some Cyclone programmers can take advantage of several advanced features that are beyond the scope of this Getting Started guide, such as RSA/AES encryption of programming images and programming restrictions on images (see **Section 6.3.1.1 - ProCryption Security Features**), the PEcloud remote programming management system (see **CHAPTER 10 - PEcloud - SECURE MANAGEMENT OF REMOTE PROGRAMMING**), and use of a barcode scanner to launch programming (see **CHAPTER 13 - USING A BARCODE SCANNER TO**

## SELECT AN IMAGE & INITIATE PROGRAMMING).

### 2.5 Launching Cyclone Programming

The newly loaded programming image should have become the image selected by the Cyclone by default. If this is not the case, use the Cyclone's menu (MENU -> Select Local Image or Cloud Job -> Local Programming Images) to select your image from those saved in memory.

There are three ways to launch programming.

1. Cyclone Start Button Press - The user simply presses the Start button located on top of the Cyclone programmer.



2. Cyclone Control Console (command-line utility) - The user writes a script that specifies parameters and initiates programming using the command line. More information is available in the Cyclone's User manual or at: [http://www.pemicro.com/blog/index.cfm?post\\_id=142](http://www.pemicro.com/blog/index.cfm?post_id=142)
3. SDK - The SDK is a software library that is used in conjunction with the user's own code. The user writes a customer application that uses this library of functions to launch programming. More information is available in the Cyclone's User Manual, or at: [http://www.pemicro.com/blog/index.cfm?post\\_id=139](http://www.pemicro.com/blog/index.cfm?post_id=139)

The "Success" or "Error" LED will illuminate to let the user know the result of programming.

**Note:** If programming is unsuccessful when using this quick start procedure, the user may instead wish to use the included PROG software for their target device. The PROG software allows the user to manually walk through the programming procedure step by step, which may help determine which part of setup or programming function is causing difficulty.



### 3 CYCLONE FX HARDWARE

The following is an overview of the features and interfaces of **Cyclone FX** programmers. Many of these interfaces are labeled on the underside of the plastic case.



Figure 3-1: Cyclone FX Top View

#### 3.1 Touchscreen LCD

The LCD Touchscreen displays information about the Cyclone's configuration and the programming process, and also allows the user to navigate the Cyclone's menus. The location of the Touchscreen LCD is shown in **Figure 3-1**.

#### 3.2 LED Indicators

The LED indicators for Error or Success will illuminate depending on the results of the programming process and provide a clear visual indication of the results. The location of the LED Indicators is shown in **Figure 3-1**.

#### 3.3 Start Button

The Start Button can be used to begin the programming process manually, provided that the Cyclone is properly configured. The location of the Start Button is shown in **Figure 3-1**.

#### 3.4 Access Panel

The Access Panel can easily be opened to allow the user to connect/disconnect ribbon cables from the headers, or to configure the Cyclone's Power Jumpers to select one of the available Power Management setups. The location of the Access Panel is shown in **Figure 3-1**; a layout of the headers and jumpers beneath the Access Panel is shown in **Figure 3-11**.





Figure 3-2: Cyclone **FX** Right Side View

### 3.5 Cyclone System Power

The **Cyclone FX** programmer requires a regulated 6V DC Center Positive power supply with 2.5/5.5mm female plug. Cyclones derive power from the Power Jack located on the right end of the unit. The location of Cyclone System Power is shown in **Figure 3-2**.

### 3.6 RS232 Communication (Serial Port)

The **Cyclone FX** provides a DB9 Female connector to communicate with a host computer through the RS232 communication (115200 Baud, 8 Data bits, No parity, 1 Stop bit). The location of the Serial Port is shown in **Figure 3-2**.

### 3.7 Ethernet Communication

The **Cyclone FX** provides a standard RJ45 socket to communicate with a host computer through the Ethernet Port (10/100 BaseT). The location of the Ethernet Port is shown in **Figure 3-2**.

### 3.8 USB Communications

The **Cyclone FX** provides a USB connector for Universal Serial Bus communications between the Cyclone and the host computer. The **Cyclone FX** is a USB 2.0 **Full-Speed** compliant device. The location of the USB Port is shown in **Figure 3-2**.

### 3.9 Electromechanical Relays

Inside the **Cyclone FX** programmer, two electromechanical relays are used to cycle target power. The specifications of the relays are as following:

<b>Maximum switched power:</b>	30W or 125 VA
<b>Maximum switched current:</b>	1A
<b>Maximum switched voltage:</b>	150VDC or 300VAC
<b>UL Rating:</b>	1A at 30 VDC 1A at 125 VAC

PEmicro only recommends switching DC voltages up to 24 Volts.



Figure 3-3: Cyclone **FX** Front Side View

### 3.10 Power Connectors

The **Cyclone FX** programmers provide a Target Power Supply Input Jack and a Target Power Supply Output Jack with 2.5/5.5 mm Pin Diameter. The power jacks are connected or disconnected by two electromechanical relays. When connected, the Center Pin of the Target Power Supply Input Jack is connected to the Center Pin of the Target Power Supply Output Jack. When disconnected, both terminals of the Target Power Supply Output Jack are connected to GND via a 1W, 100 Ohm resistor. The location of Target Power In is shown in **Figure 3-3**, and the location of Target Power Out is shown in **Figure 3-3**.

### 3.11 Reset Button

The Reset Button performs a hard reset of the Cyclone system. The location of the Reset Button is shown in **Figure 3-3**.



Figure 3-4: Cyclone **FX** Rear Side View

### 3.12 SDHC Port (FX Only)

**Note:** The SDHC port is only active on **Cyclone FX** programmers (and some legacy Cyclone LC models, with license).

The SDHC port allows the user to store programming images that are, individually or collectively, larger than the Cyclone's internal memory. It also makes it quicker and more convenient to swap programming images. PEmicro offers certified SDHC cards on our website at [pemicro.com](http://pemicro.com). The location of the SDHC Port is shown in **Figure 3-4**.

Programming images are managed on the SD card in exactly the same way as they are in the Cyclone's internal memory. Please see **Section 6.6 - Managing Multiple SAP Images** for more information about using the Manage Images utility.

To view detailed information about the status of the SDHC card/port, tap the icon bar at the top of the touchscreen menu. This status can provide you with relevant information if you are encountering any difficulty while trying to use an SDHC card.

SDHC cards with a memory capacity up to 4GB are supported. Cards with a larger capacity may

work but have not been tested by PE micro.

### 3.13 Programming Control Port

The Programming Control Port is a 10-pin interface which allows external hardware the ability to launch programming and read success, failure, and busy/idle status from the **Cyclone FX** through standard I/O signals. This allows simple automated control of the Cyclone without using one of the communications interfaces (Serial, USB, Ethernet).

The port can be configured to work with I/O signals from 1.6V-5.5V. The signal port provides a 5V power output which can be used to power external electronics.



Figure 3-5: Port and Pin 1 Location (Shown on Cyclone FX)

#### 3.13.1 Pin-Out And Definitions

Below is the pin-out of the programming control port and the definitions of the signals:

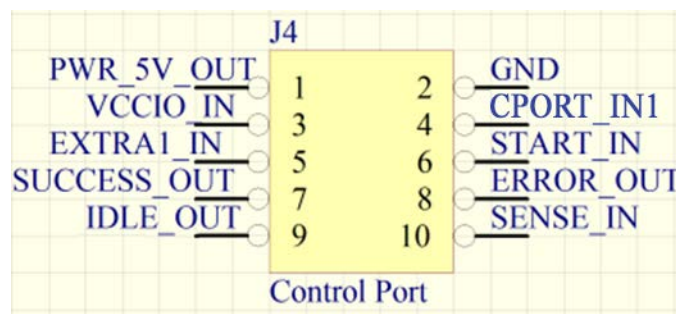


Figure 3-6: Programming Control Port Pin-Out

Pin#	Signal Name	Description
1	PWR_5V_OUT	The Cyclone can provide a constant 5V, 100mA output on this pin which may be used to power external circuitry.
2	GND	Should be connected to the GND of any controlling circuitry.
3	VCCIO_IN	This voltage is used to drive the output pins of the programming control port and is used as a reference for the input pins. This voltage can be from 1.6v-5.5v. This pin may be directly connected to the PWR_5V_OUT output pin of the Programming Control Port, if 5v operation of the port is desired. This signal has an internal 100K Ohm pullup resistor and a 470 Ohm series resistor.
4	CPORT_IN1	Falling edge sensitive pin which will clear the status (Success or Error) LED of the Cyclone (if ~SENSE_IN is low and triggering is enabled with the "controlporttriggerenable" property in the Cyclone). This signal has an internal 100K Ohm pullup resistor and a 470 Ohm series resistor.
5	EXTRA_IN21	Currently unused.
6	~START_IN	Falling edge sensitive pin which will launch programming of the currently selected image on the Cyclone (if ~SENSE_IN is low and triggering is enabled with the "controlporttriggerenable" property in the Cyclone). This signal has an internal 100K Ohm pullup resistor and a 470 Ohm series resistor.

Pin#	Signal Name	Description
7	~SUCCESS_OUT	A low value indicates the most recent programming operation was successful. A high value means that there is no success to report (programming is on-going, an error occurred, no programming launch has occurred, too much time has passed since last programming operation, etc). This signal has an internal 100K Ohm pullup resistor and a 470 Ohm series resistor.
8	~ERROR_OUT	A low value indicates the most recent programming operation failed. A high value means that there is no error to report (programming is on-going, programming was successful, no programming launch has occurred, too much time has passed since last programming operation, etc). This signal has an internal 100K Ohm pullup resistor and a 470 Ohm series resistor.
9	~IDLE_OUT	A low indicates the Cyclone is IDLE and can accept a programming launch operation. A high indicates the Cyclone is currently running a programming operation. This signal has an internal 100K Ohm pullup resistor and a 470 Ohm series resistor.
10	~SENSE_IN	An input which must be driven low to enable the programming control port inputs. This signal has an internal 100K Ohm pullup resistor and a 470 Ohm series resistor.

### 3.13.2 Powering the Programming Control Port

To allow the Cyclone to drive the port outputs to show the current programming state, and to allow the Cyclone to properly interpret input signals on the port, the Programming Control Port I/O voltage must be set. This is done by connecting an external voltage of 1.6v-5.5v to the VCCIO\_IN pin of the port. This VCCIO\_IN pin may be directly connected to the PWR\_5V\_OUT output pin of the Signal Control Port if 5v operation of the port is desired. If connecting to external circuitry which runs at a different voltage, such a 3v, simply connect the desired voltage to the VCCIO\_IN pin.

Once powered, the outputs (~SUCCESS\_OUT, ~ERROR\_OUT, and ~IDLE\_OUT) become actively driven.

### 3.13.3 Enabling Triggering on the Programming Control Port

The Programming Control Port allows programming to be launched from the Programming Control Port. For this to happen, the triggering must previously have been enabled in the settings of the Cyclone (see below) and the port must be detected as active.

The Programming Control Port is considered active if there is an appropriate voltage applied to the VCCIO\_IN pin and the ~SENSE\_IN signal is driven low.

To enable trigger in the settings of the Cyclone, the “controlporttriggerenable” property in the Cyclone must have been enabled at some point (this setting is stored in the Cyclone once set). This can be done via the Cyclone Control SDK, Cyclone Control Console, or Cyclone Control GUI.

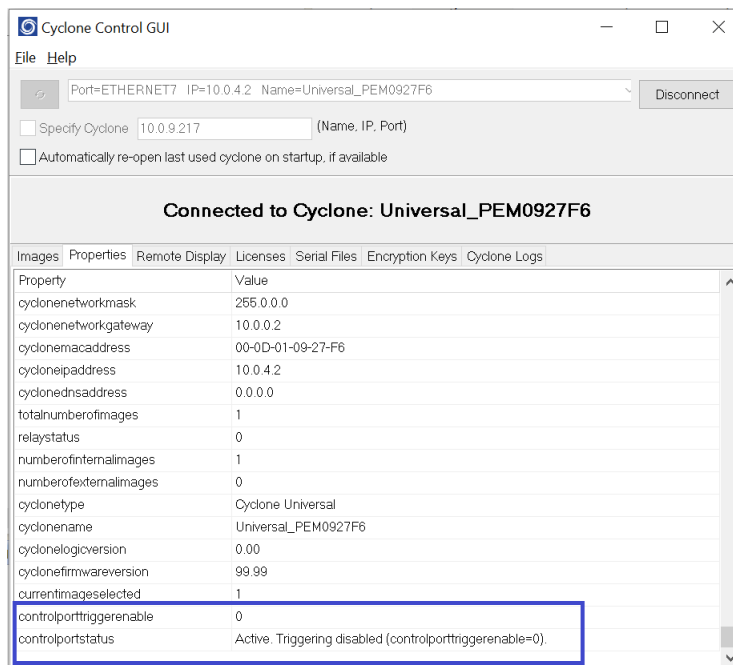
For the Cyclone Control Console, an example command which reads this property for a Cyclone named “Universal\_PEM0927F6” is:

```
cycloneControlConsole -cyclone=Universal_PEM0927F6 -getproperty=hardwareproperties,0,controlporttriggerenable
```

For the Cyclone Control Console, an example command which sets this property for a Cyclone named “Universal\_PEM0927F6” is:

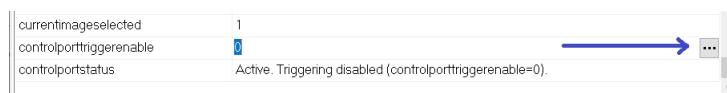
```
cycloneControlConsole -cyclone=Universal_PEM0927F6 -setproperty=hardwareproperties,0,controlporttriggerenable,1
```

In the Cyclone Control GUI, this property may be accessed by opening the Cyclone and choosing the “Properties” tab:



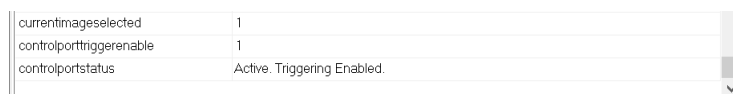
**Figure 3-7: Properties Tab**

To set the property, highlight it and click the ... button on that line. Enter the value 1 to enable and 0 to disable triggering



**Figure 3-8: Setting The Property**

The triggering should then show as active (note that ~SENSE\_IN will also need to be driven low for triggering to work):



**Figure 3-9: Active Triggering**

### 3.13.4 Timing on the Programming Control Port

In its default idle state, the Cyclone will drive the ~IDLE\_OUT signal low. When a falling edge is detected on the ~START\_IN signal, the ~SUCCESS\_OUT and ~ERROR\_OUT signals will be driven high and then the ~IDLE\_OUT signal will be driven high (BUSY). Once programming is complete, the ~SUCCESS\_OUT and ~ERROR\_OUT signals will be driven to reflect the result of programming and then the ~IDLE\_OUT signal will be driven back low.

A safe mechanism to launch programming on the Cyclone if controlled by edge sensitive circuitry:

1. Wait for ~IDLE\_OUT to be low
2. Drive the ~START\_IN signal low then back high (this edge is detected in HW on the Cyclone and won't be missed)
3. Wait for a negative edge on ~IDLE\_OUT (i.e. wait for the Cyclone to go to BUSY then IDLE)
4. Read the ~SUCCESS\_OUT and ~ERROR\_OUT signals to determine result

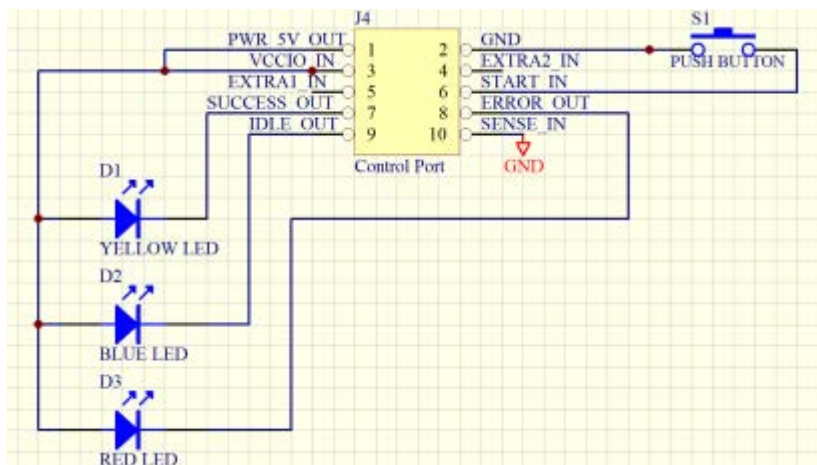
A safe mechanism to launch programming on the Cyclone without using edge sensitive circuitry (polled method):



1. Wait for  $\sim$ IDLE\_OUT to be low
2. Drive the  $\sim$ START\_IN signal low then back high (this edge is detected in HW on the Cyclone and won't be missed)
3. Wait 100mS for the Cyclone to recognize the  $\sim$ START\_IN falling edge on this port and to set the  $\sim$ IDLE\_OUT signal
4. Wait for  $\sim$ IDLE\_OUT to go back low
5. Read the  $\sim$ SUCCESS\_OUT and  $\sim$ ERROR\_OUT signals to determine result

### 3.13.5 Push Button / LED Example Circuit

This following example circuit has a push button that triggers programming. The three LEDs show the Success, Failure, and Idle/Busy lines.



**Figure 3-10: Example Circuit**

After setting up the above connections, pushing the button will trigger programming and the Cyclone state can be seen on the LEDs.

**Note:** No separate license is required to use this port.

An example of how to use Raspberry Pi to launch programming via the Programming Control Port is shown at:

[http://www.pemicro.com/blog/index.cfm?post\\_id=222](http://www.pemicro.com/blog/index.cfm?post_id=222)

### 3.14 USB Expansion Port

The location of the USB Expansion Port is shown in **Figure 3-3**. The USB Expansion Port supports use of a bar code scanner, which can provide the user with helpful features during the programming process. For detailed information on how to use the barcode scanner with a Cyclone FX, please read **Section 13 - USING A BARCODE SCANNER TO SELECT AN IMAGE & INITIATE PROGRAMMING**.

### 3.15 Optional Oscillator (MON08 Only)

**Cyclone FX** programmers with MON08 support (PEmicro Part# CYCLONE-FX-UNIV only) provide a software configurable 9.8304MHz or 4.9152 MHz oscillator clock signal to Pin 13 of the MON08 Connector. The user may use this clock signal to overdrive the target RC or crystal circuitry. If this signal is not used, just leave Pin 13 of the target MON08 header unconnected.

Please note that if the target already uses an oscillator as its clock, the Cyclone will NOT be able to overdrive it. The clock should have sufficient drive to be used with a target system even if the target system has an RC circuit or crystal connected.



### 3.16 Cyclone Time / Real Time Clock

**Cyclone FX** programmers are equipped with a Real Time Clock (RTC) module designed to keep accurate timing even when the Cyclone is turned off.

The Date & Time are displayed on the home screen. Date/Time settings can be configured by navigating to the following menu using the touchscreen display:

**Main Menu / Configure Cyclone Settings / Configure Time Settings**

For more information on the available configuration options, see **Section 5.2.4.6 - Configure Time Settings (Cyclone Time / Real Time Clock)**.

### 3.17 Power Jumper Settings

The Power Jumpers must be set differently for various power management options that the **Cyclone FX** offers. If the target is being powered independently of the **Cyclone FX**, all pins in the Power Jumpers header must instead be left unpopulated. To reveal the Power Jumpers header, lift the access panel on the left end of the **Cyclone FX**. The location is indicated as Power Jumpers in **Figure 3-11**. Please see **CHAPTER 4 - TARGET POWER MANAGEMENT** for the correct jumper settings for the Cyclone's power management options. A quick guide to these settings is also located on the underside label of the **Cyclone FX**.

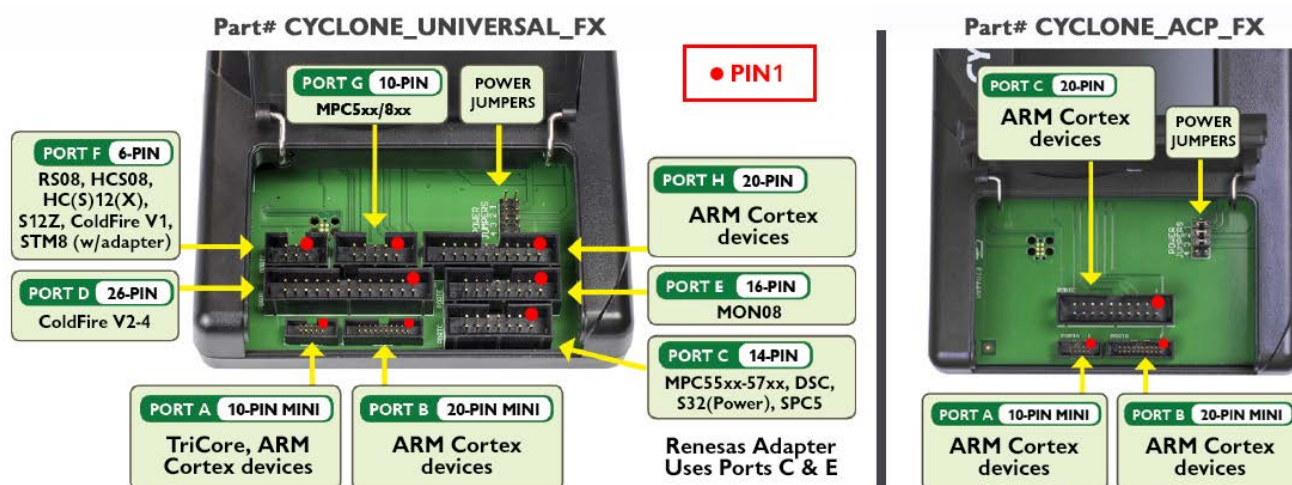
### 3.18 Debug Connectors

When purchasing a **Cyclone FX** programmer, the user is able to choose between two part numbers, each corresponding to a different level of device support. See the sticker on the underside of the Cyclone to determine the PEmicro part# for your specific **Cyclone FX** programmer.

**PEmicro Part# CYCLONE-FX-ARM** supports ARM Cortex devices only, so this programmer provides one shrouded, un-keyed, 0.100-inch pitch dual row 0.025-inch square header, and two shrouded, keyed 0.050-inch pitch dual row mini headers.

**PEmicro Part# CYCLONE-FX-UNIV** supports ARM Cortex devices and additionally supports target connections to many 8-/16-/32-bit NXP architectures, so this programmer provides six shrouded, un-keyed, 0.100-inch pitch dual row 0.025-inch square headers, and two shrouded, keyed 0.050-inch pitch dual row mini headers.

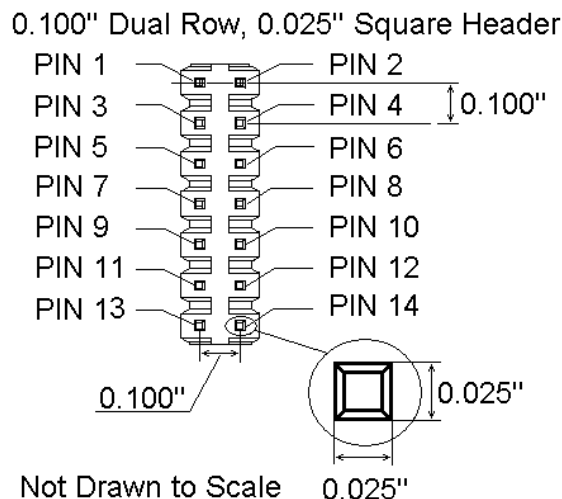
To reveal the headers and connect/disconnect ribbon cables, lift the access panel on the left end of the Cyclone. Each header is designated for one or more specific target architectures, as indicated in **Figure 3-11**.



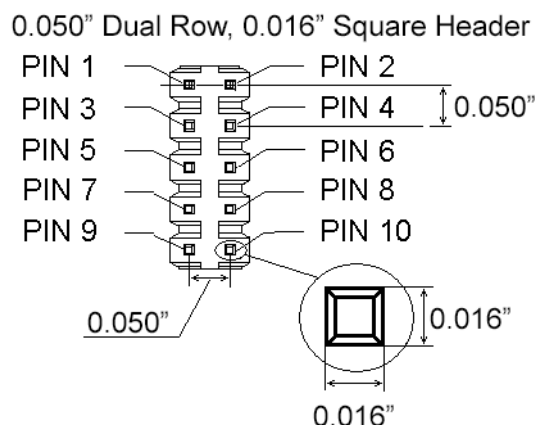
**Figure 3-11: Target Headers & Power Jumpers (CYCLONE-FX-UNIV vs. CYCLONE-FX-ARM)**

Mechanical drawings are shown below whose dimensions are representative of the pin size and spacing of these headers.

**Note:** The number of pins depicted in the mechanical drawings may differ from the Cyclone headers; the drawings are provided simply to demonstrate pin size and spacing.



**Figure 3-12: 20-Pin Un-Keyed Header Dimensions**



**Figure 3-13: Mini 10-Pin and Mini 20-Pin Keyed Header Dimensions**

### 3.19 Target Headers For Part# CYCLONE-FX-ARM

PEmicro Part# CYCLONE-FX-ARM features 3 ports labeled A-C.

#### 3.19.1 PORT A: 10-Pin Keyed Mini Connector (Kinetis, S32 (ARM), other PEmicro-Supported ARM devices)

##### 3.19.1.1 JTAG Pin Assignments

The Cyclone provides a keyed 10-pin 0.050-inch pitch double row connector for ARM targets. The location of the this header is indicated as PORT A in **Figure 3-11**. The 10-pin keyed mini connector pin definitions for JTAG mode are as follows:

##### 10-Pin Keyed Mini Connector JTAG Mode Pin Assignments

PIN 1 - TVCC	TMS - PIN 2
PIN 3 - GND	TCK - PIN 4
PIN 5 - GND	TDO - PIN 6
PIN 7 - NC	TDI - PIN 8
PIN 9 - JTAG_MOD/NC*	RESET - PIN 10

**Note:** \* PIN9: Users of NXP i.MX processors are recommended to connect the JTAG\_MOD signal to this

pin to allow programming when secure JTAG is enabled. For all other processors, this pin should be left as NC.

**Cyclone FX** programmers also support SWD Mode. This replaces the JTAG connection with a clock and single bi-directional data pin.

### 3.19.1.2 SWD Mode Pin Assignments

#### 10-Pin Keyed Mini Connector SWD Mode Pin Assignments

PIN 1 - TVCC	TMS/SWDIO - PIN 2
PIN 3 - GND	TCK/SWCLK - PIN 4
PIN 5 - GND	NC* - PIN 6
PIN 7 - NC	NC* - PIN 8
PIN 9 - JTAG_MOD/NC**	RESET - PIN 10

**Note:** \*The pin is reserved for internal use within the PEmicro interface.

**Note:** \*\* PIN9: Users of NXP i.MX processors are recommended to connect the JTAG\_MOD signal to this pin to allow programming when secure JTAG is enabled. For all other processors, this pin should be left as NC.

SWD Mode is selected from the “Communication Mode” drop-down box in the Cyclone Image Creation Utility:

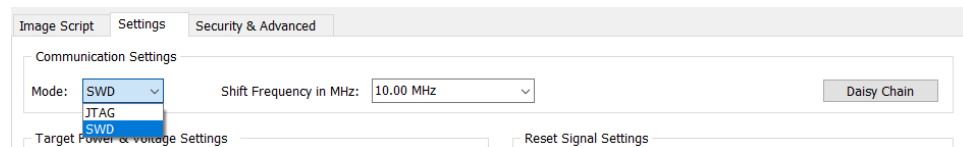


Figure 3-14: Communications Mode Selection

#### 3.19.1.2.1 High-Performance Communications

If high-performance options are available for the selected device they will appear in the “Shift Frequency in MHz” drop-down. **Cyclone FX** programmers are capable of high-performance communications when using certain ARM Cortex targets in SWD mode.

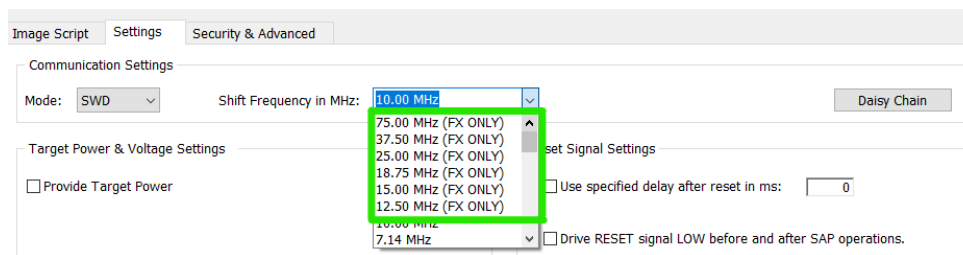


Figure 3-15: High-Performance Options

## 3.19.2 PORT B: 20-Pin Keyed Mini Connector (Kinetis, S32 (ARM), other PEmicro-Supported ARM devices)

### 3.19.2.1 JTAG Mode Pin Assignments

The Cyclone provides a keyed 20-pin 0.050-inch pitch double row connector for ARM targets. The location of the this header is indicated as PORT B in **Figure 3-11**. The 20-pin keyed mini connector pin definitions for JTAG mode are as follows:

#### 20-Pin Keyed Mini Connector JTAG Mode Pin Assignments

PIN 1 - TVCC	TMS - PIN 2
PIN 3 - GND	TCK - PIN 4
PIN 5 - GND	TDO - PIN 6
PIN 7 - NC	TDI - PIN 8

PIN 9 - JTAG_MOD/NC**	<u>RESET</u>	- PIN 10
PIN 11 - NC	NC*	- PIN 12
PIN 13 - NC	NC*	- PIN 14
PIN 15 - GND	NC*	- PIN 16
PIN 17 - GND	NC*	- PIN 18
PIN 19 - GND	NC*	- PIN 20

**Note:** \*The pin is reserved for internal use within the PEmicro interface.

**Note:** \*\* PIN9: Users of NXP i.MX processors are recommended to connect the JTAG\_MOD signal to this pin to allow programming when secure JTAG is enabled. For all other processors, this pin should be left as NC.

### 3.19.2.2 SWD Mode Pin Assignments

**Cyclone FX** programmers also support SWD Mode. This replaces the JTAG connection with a clock and single bi-directional data pin.

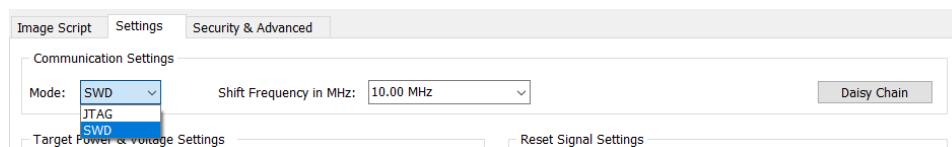
#### 20-Pin Keyed Mini Connector SWD Mode Pin Assignments

PIN 1 - TVCC	TMS/SWDIO	- PIN 2
PIN 3 - GND	TCK/SWCLK	- PIN 4
PIN 5 - GND	NC*	- PIN 6
PIN 7 - NC	NC*	- PIN 8
PIN 9 - JTAG_MOD/NC**	<u>RESET</u>	- PIN 10
PIN 11 - NC	NC*	- PIN 12
PIN 13 - NC	NC*	- PIN 14
PIN 15 - GND	NC*	- PIN 16
PIN 17 - GND	NC*	- PIN 18
PIN 19 - GND	NC*	- PIN 20

**Note:** \*The pin is reserved for internal use within the PEmicro interface.

**Note:** \*\* PIN9: Users of NXP i.MX processors are recommended to connect the JTAG\_MOD signal to this pin to allow programming when secure JTAG is enabled. For all other processors, this pin should be left as NC.

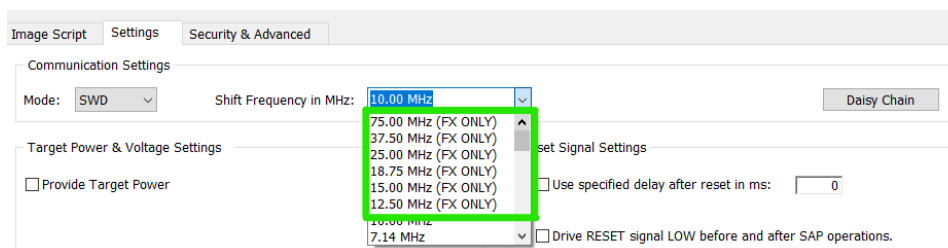
SWD Mode is selected from the “Communication Mode” drop-down box in the Cyclone Image Creation Utility:



**Figure 3-16: Communications Mode Selection**

#### 3.19.2.2.1 High-Performance Communications

If high-performance options are available for the selected device they will appear in the “Shift Frequency in M Hz” drop-down. **Cyclone FX** programmers are capable of high-performance communications when using certain ARM Cortex targets in SWD mode.



**Figure 3-17: High-Performance Options**

### 3.19.3 PORT C: 20-Pin Debug Connector (Kinetis, S32 (ARM), other PEmicro-Supported ARM devices)

#### 3.19.3.1 JTAG Mode Pin Assignments

The Cyclone provides a 20-pin 0.100-inch pitch double row connector for ARM targets. The location of this header is indicated as **PORT C** under Part# CYCLONE-FX-ARM in **Figure 3-11**. The 20-pin standard connector pin definitions for JTAG mode are as follows:

##### 20-Pin Standard Connector JTAG Mode Pin Assignments

PIN 1 - <u>TVCC</u>	NC* - PIN 2
PIN 3 - <u>TRST</u> or NC	GND - PIN 4
PIN 5 - <u>TDI</u>	GND - PIN 6
PIN 7 - <u>TMS</u>	GND - PIN 8
PIN 9 - <u>TCK</u>	GND - PIN 10
PIN 11 - NC*	GND - PIN 12
PIN 13 - <u>TDO</u>	GND - PIN 14
PIN 15 - <u>RESET</u>	GND - PIN 16
PIN 17 - NC*	GND - PIN 18
PIN 19 - NC*	GND - PIN 20

**Note:** \*The pin is reserved for internal use within the PEmicro interface.

#### 3.19.3.2 SWD Mode Pin Assignments

**Cyclone FX** programmers also support SWD Mode. This replaces the JTAG connection with a clock and single bi-directional data pin.

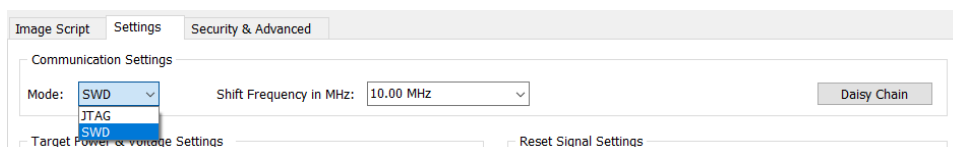
##### 20-Pin Standard Connector SWD Mode Pin Assignments

PIN 1 - <u>TVCC</u>	NC* - PIN 2
PIN 3 - <u>TRST</u> or NC*	GND - PIN 4
PIN 5 - NC*	GND - PIN 6
PIN 7 - <u>TMS/SWDIO</u>	GND - PIN 8
PIN 9 - <u>TCK/SWCLK</u>	GND - PIN 10
PIN 11 - NC*	GND - PIN 12
PIN 13 - NC*	GND - PIN 14
PIN 15 - <u>RESET</u>	GND - PIN 16
PIN 17 - NC*	GND - PIN 18
PIN 19 - NC*	GND - PIN 20

**Note:** \*The pin is reserved for internal use within the PEmicro interface.

SWD Mode is selected from the "Communication Settings" drop-down box in the Settings tab of the Cyclone Image Creation Utility:

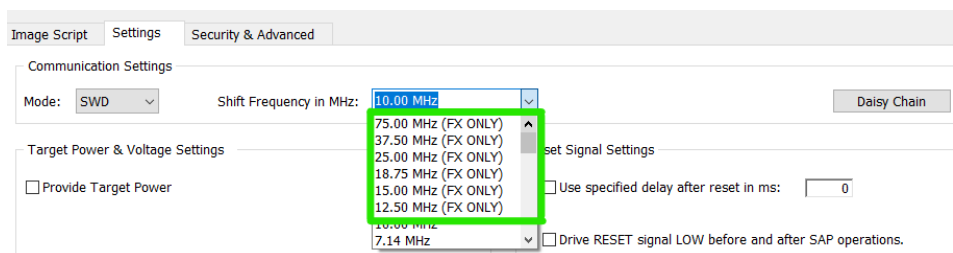




**Figure 3-18: Communications Mode Selection**

### 3.19.3.2.1 High-Performance Communications

If high-performance options are available for the selected device they will appear in the “Shift Frequency in MHz” drop-down. **Cyclone FX** programmers are capable of high-performance communications when using certain ARM Cortex targets in SWD mode.



**Figure 3-19: High-Performance Options**

## 3.20 Target Headers For Part# CYCLONE-FX-UNIV

PEmicro Part# CYCLONE-FX-UNIV features 6 ports labeled A-H.

### 3.20.1 PORT A: 10-Pin Keyed Mini Connector (NXP Kinetis & S32 (ARM), other PE micro-Supported ARM devices, Infineon TriCore - DAP only)

#### 3.20.1.1 JTAG Mode Pin Assignments

The Cyclone provides a keyed 10-pin 0.050-inch pitch double row connector for ARM targets. The location of the this header is indicated as PORT A in **Figure 3-11**. The 10-pin keyed mini connector pin definitions for JTAG mode are as follows:

#### 10-Pin Keyed Mini Connector JTAG Mode Pin Assignments

PIN 1 - TVCC	TMS - PIN 2
PIN 3 - GND	TCK - PIN 4
PIN 5 - GND	TDO - PIN 6
PIN 7 - NC	TDI - PIN 8
PIN 9 - JTAG_MOD/NC*	RESET - PIN 10

**Note:** \* PIN9: Users of NXP i.MX processors are recommended to connect the JTAG\_MOD signal to this pin to allow programming when secure JTAG is enabled. For all other processors, this pin should be left as NC.

#### 3.20.1.2 SWD Mode Pin Assignments

**Cyclone FX** programmers also support SWD Mode. This replaces the JTAG connection with a clock and single bi-directional data pin.

#### 10-Pin Keyed Mini Connector SWD Mode Pin Assignments

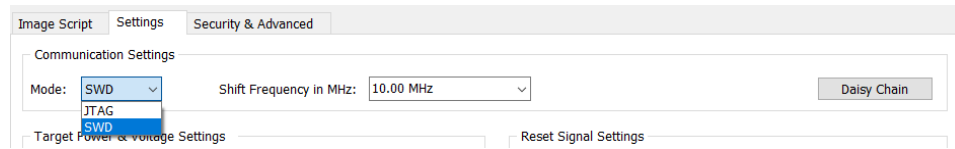
PIN 1 - TVCC	TMS/SWDIO - PIN 2
PIN 3 - GND	TCK/SWCLK - PIN 4
PIN 5 - GND	NC* - PIN 6
PIN 7 - NC	NC* - PIN 8
PIN 9* - JTAG_MOD/NC**	RESET - PIN 10

**Note:** \*The pin is reserved for internal use within the PE micro interface.



**Note:** \*\* PIN9: Users of NXP i.MX processors are recommended to connect the JTAG\_MOD signal to this pin to allow programming when secure JTAG is enabled. For all other processors, this pin should be left as NC.

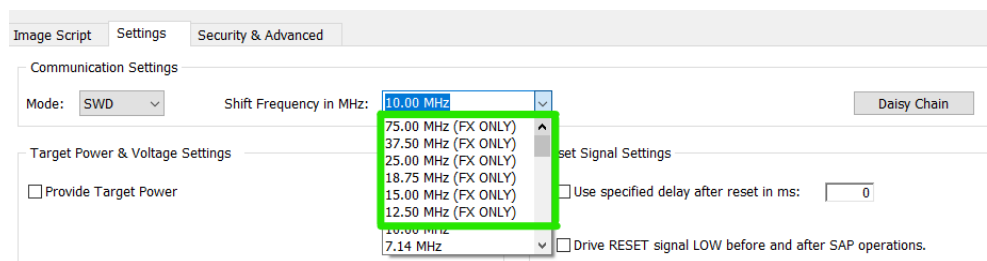
SWD Mode is selected from the “Communication Mode” drop-down box in the Cyclone Image Creation Utility:



**Figure 3-20: Communications Mode Selection**

### 3.20.1.2.1 High-Performance Communications

If high-performance options are available for the selected device they will appear in the “Shift Frequency in MHz” drop-down. **Cyclone FX** programmers are capable of high-performance communications when using certain ARM Cortex targets in SWD mode.



**Figure 3-21: High-Performance Options**

### 3.20.1.3 DAP Connector Pin Assignments (Target Board Has Dedicated DAP Connector Port)

Users whose target board does not have a dedicated DAP Connector should refer instead to Port H in **Section 3.20.8.3 - DAP Connector Pin Assignments (Target Board Does Not Have Dedicated DAP Connector Port)**.

The keyed 10-pin 0.050-inch pitch double row connector of the CYCLONE-FX-UNIV model supports Infineon TriCore targets (AUDO™ TC1xx and AURIX™ TC2xx/TC3xx). The location of the this header is indicated as PORT A in **Figure 3-11**. The 10-pin keyed mini connector pin definitions for DAP connectors are as follows:

#### 10-Pin Keyed Mini Connector DAP Pin Assignments

PIN 1 - TVCC	DAP1 - PIN 2
PIN 3 - GND	DAP0 - PIN 4
PIN 5 - GND	NC* - PIN 6
PIN 7 - NC	NC* - PIN 8
PIN 9 - NC*	RESET - PIN 10

**Note:** \*The pin is reserved for internal use within the PEmicro interface.

## 3.20.2 PORT B: 20-Pin Keyed Mini Connector (Kinetis, S32 (ARM), other PEmicro-Supported ARM devices)

### 3.20.2.1 JTAG Mode Pin Assignments

The Cyclone provides a keyed 20-pin 0.050-inch pitch double row connector for ARM targets. The location of the this header is indicated as PORT B in **Figure 3-11**. The 20-pin keyed mini connector pin definitions for JTAG mode are as follows:

#### 20-Pin Keyed Mini Connector JTAG Mode Pin Assignments

PIN 1 - TVCC	TMS - PIN 2
--------------	-------------

PIN 3 - <b>GND</b>	<b>TCK</b> - PIN 4
PIN 5 - <b>GND</b>	<b>TDO</b> - PIN 6
PIN 7 - <b>NC</b>	<b>TDI</b> - PIN 8
PIN 9 - <b>JTAG_MOD/NC**</b>	<b>RESET</b> - PIN 10
PIN 11 - <b>NC</b>	<b>NC*</b> - PIN 12
PIN 13 - <b>NC</b>	<b>NC*</b> - PIN 14
PIN 15 - <b>GND</b>	<b>NC*</b> - PIN 16
PIN 17 - <b>GND</b>	<b>NC*</b> - PIN 18
PIN 19 - <b>GND</b>	<b>NC*</b> - PIN 20

**Note:** \*The pin is reserved for internal use within the PEmicro interface.

\*\* PIN9: Users of NXP i.MX processors are recommended to connect the JTAG\_MOD signal to this pin to allow programming when secure JTAG is enabled. For all other processors, this pin should be left as NC.

### 3.20.2.2 SWD Mode Pin Assignments

**Cyclone FX** programmers also support SWD Mode. This replaces the JTAG connection with a clock and single bi-directional data pin.

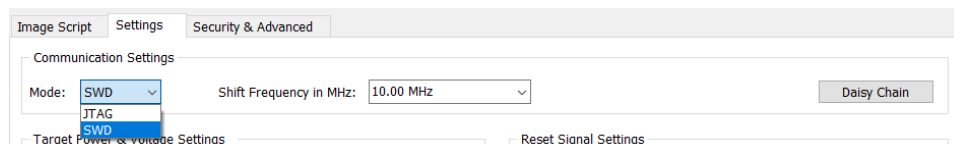
#### 20-Pin Keyed Mini Connector SWD Mode Pin Assignments

PIN 1 - <b>TVCC</b>	<b>TMS/SWDIO</b> - PIN 2
PIN 3 - <b>GND</b>	<b>TCK/SWCLK</b> - PIN 4
PIN 5 - <b>GND</b>	<b>NC*</b> - PIN 6
PIN 7 - <b>NC</b>	<b>NC*</b> - PIN 8
PIN 9 - <b>JTAG_MOD/NC**</b>	<b>RESET</b> - PIN 10
PIN 11 - <b>NC</b>	<b>NC*</b> - PIN 12
PIN 13 - <b>NC</b>	<b>NC*</b> - PIN 14
PIN 15 - <b>GND</b>	<b>NC*</b> - PIN 16
PIN 17 - <b>GND</b>	<b>NC*</b> - PIN 18
PIN 19 - <b>GND</b>	<b>NC*</b> - PIN 20

**Note:** \*The pin is reserved for internal use within the PEmicro interface.

\*\* PIN9: Users of NXP i.MX processors are recommended to connect the JTAG\_MOD signal to this pin to allow programming when secure JTAG is enabled. For all other processors, this pin should be left as NC.

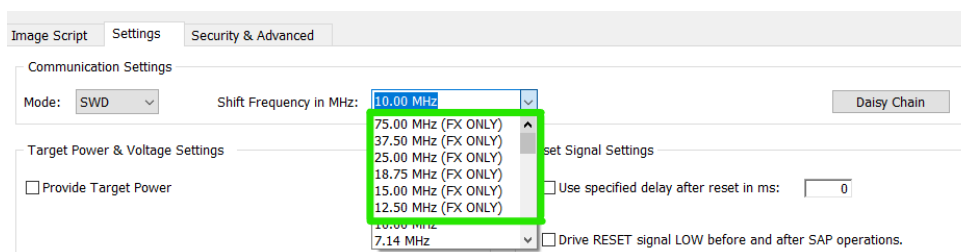
SWD Mode is selected from the “Communication Mode” drop-down box in the Cyclone Image Creation Utility:



**Figure 3-22: Communications Mode Selection**

#### 3.20.2.2.1 High-Performance Communications

If high-performance options are available for the selected device they will appear in the “Shift Frequency in MHz” drop-down. **Cyclone FX** programmers are capable of high-performance communications when using certain ARM Cortex targets in SWD mode.



**Figure 3-23: High-Performance Options**

### 3.20.3 PORT C: 14-Pin Debug Connector (MPC55xx-57xx, SPC5, DSC, S32 (Power))

The Cyclone provides a standard 14-pin 0.100-inch pitch dual row 0.025-inch square header for MPC55xx-57xx, DSC (MC56F8xxx), S32R, or STMicroelectronics' SPC5 targets. The location of the this header is indicated as PORT C in **Figure 3-11**.

#### MPC55xx-57xx, SPC5, or S32 (Power) Pinout

TDI	1	2	GND
TDO	3	4	GND
TCK	5	6	GND
NC	7	8	NC
RESET	9	10	TMS
VDDE7	11	12	GND
RDY	13	14	JCOMP

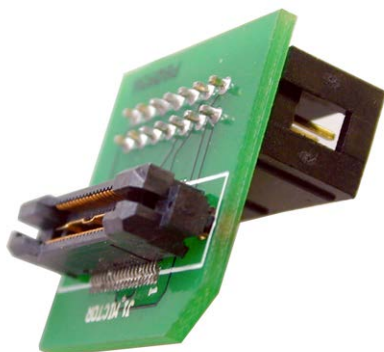
#### DSC Pinout

TDI	1	2	GND
TDO	3	4	GND
TCK	5	6	GND
NC	7	8	NC/KEY
RESET	9	10	TMS
TVCC	11	12	GND
NC*	13	14	TRST

\*The pin is reserved for internal use within the PEmicro interface.

#### 3.20.3.1 BERG14-to-MICTOR38 Optional Connector

PEmicro offers a 14-pin BERG to 38-pin MICTOR adapter, sold separately, that may be used on Port C of the **Cyclone FX**. The PEmicro part number is BERG14-TO-MICTOR38.



**Figure 3-24: BERG14-TO-MICTOR38 Adapter (Sold Separately)**

**PORT D: 26-Pin Debug Connector (ColdFire V2/3/4)**

The Cyclone provides a standard 26-pin 0.100-inch pitch dual row 0.025-inch square header for ColdFire MCF52xx/53xx/54xx family of microprocessors. This port connects to the target hardware using either the ColdFire extension cable for synchronous ColdFire targets such as MCF5272 & MCF5206E (PEmicro part# CABLE-CF-ADAPTER, sold separately), or a standard 26-pin ribbon cable for asynchronous ColdFire targets (included). Please refer to each processor's user manual to identify whether it is a synchronous or asynchronous interface. The location of the this header is indicated as PORT D in **Figure 3-11**.

**ColdFire V2/3/4 Pinout**

N/C	1	2	BKPT
GND	3	4	DSCLK
GND	5	6	NC*
RESET	7	8	DSI
TVCC	9	10	DSO
GND	11	12	PST3
PST2	13	14	PST1
PST0	15	16	DDATA3
DDATA2	17	18	DDATA1
DDATA0	19	20	GND
N/C	21	22	N/C
GND	23	24	CLK
TVCC	25	26	TA

\*The pin is reserved for internal use within the PEmicro interface.

The ColdFire adapter for Synchronous targets and ribbon cable for Asynchronous targets is pictured below:



**Figure 3-25: ColdFire Adapter (part# CABLE\_CF\_ADAPTER (Rev. B), for synchronous ColdFire targets, sold separately)**



**Figure 3-26: ColdFire Ribbon Cable (for asynchronous ColdFire targets, included with Cyclone)**

### 3.20.5 PORT E: 16-Pin Debug Connector (MON08)

The Cyclone provides a 16-pin 0.100-inch pitch double row connector for MON08 targets. The location of this header is indicated as PORT E in **Figure 3-11**. The MON08 header adopts the standard pin-out from MON08 debugging with some modifications. The general pin-out is as follows:

#### MON08 Signals

PIN 1 - NC*	GND	- PIN 2
PIN 3 - NC**	RST	- PIN 4
PIN 5 - NC*	IRQ	- PIN 6
PIN 7 - NC*	MON4	- PIN 8
PIN 9 - NC*	MON5	- PIN10
PIN11 - NC*	MON6	- PIN12
PIN13 - OSC	MON7	- PIN14
PIN15 - Vout	MON8	- PIN16

\*The pin is reserved for internal use within the PEmicro interface.

\*\*The pin is reserved for internal use within the PEmicro interface only when using an MR8 target.

**Note:** The pins labeled as MON04 - MON08 vary from device to device. To view the exact pinout for your specific MON08 part (e.g., 68HC908AB) the user may consult the NXP user manual for their device, or view the pinout in either the Cyclone Image Creation Utility or the PROG08SZ interface by selecting their specific device in the GUI.

### 3.20.6 PORT F: 6-Pin Debug Connector (RS08, HCS08, HC(S)12(X), S12Z, ColdFire +/V1, STM8 w/ adapter)

The Cyclone provides a standard 6-pin 0.100-inch pitch dual row 0.025-inch square header for ColdFire V1, S12Z, 68(S)12(X), 68HCS08, RS08, and STMicroelectronics' STM8 targets. The location of this header is indicated as PORT F in **Figure 3-11**. The header uses the NXP standard pin configuration, listed here for reference:

#### ColdFire V1, 68(S)12(X), 68HCS08, and RS08 Signals

PIN 1 - BKGD	GND - PIN 2
PIN 3 - NC	RESET - PIN 4
PIN 5 - NC	TVCC - PIN 6



### S12Z Signals

**Note:**\* indicates optional signal

PIN 1 - <b>BKGD</b>	<b>GND</b> - PIN 2
PIN 3 - <b>PDO*</b>	<b>RESET</b> - PIN 4
PIN 5 - <b>PDOCLK*</b>	<b>TVCC</b> - PIN 6

### 6-Pin STM8 Signals

PIN 1 - <b>SWIM**</b>	<b>GND</b> - PIN 2
PIN 3 - <b>NC*</b>	<b>RESET</b> - PIN 4
PIN 5 - <b>NC*</b>	<b>TVCC</b> - PIN 6

\*The pin is reserved for internal use within the PEmicro interface.

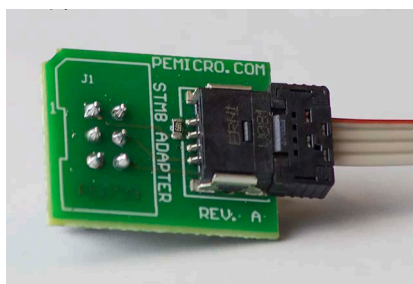
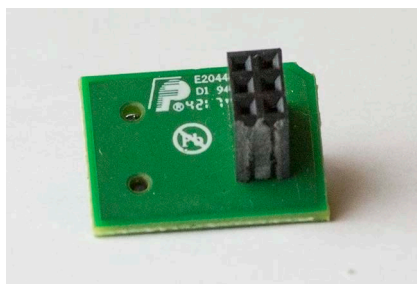
\*\* All the signals are direct connect except the SWIM line which requires a 680 Ohm pull-up

PEmicro also offers a separate STM8 adapter (part# CU-CUFX-STM8-ADPT) that can be plugged into the 6-pin header of the Cyclone (see **Figure 3-27**). The adapter offers 4 pins signals from an ERNI connector.

### 4-Pin STM8 Signals

(Requires STM8 Adapter, sold separately)

PIN 1 - <b>TVCC</b>	<b>SWIM</b> - PIN 2
PIN 3 - <b>GND</b>	<b>RESET</b> - PIN 4



**Figure 3-27: STM8 Adapter: 1) Bottom, 2) Top, 3) Connected To 6-Pin Header of Cyclone\_Universal\_FX (Adapter Sold Separately)**

#### **3.20.7 PORT G: 10-Pin Debug Connector (Power MPC5xx/8xx)**

The Cyclone Universal provides a standard 10-pin 0.100-inch pitch dual row 0.025-inch square header for Power MPC5xx/8xx BDM targets. The location of the this header is indicated as PORT G in **Figure 3-11**.

#### **Power MPC5xx/8xx BDM Pinout**

VFLS0*	1	2	$\overline{\text{SRESET}}$
GND	3	4	DSCK
GND	5	6	VFLS1*
$\overline{\text{HRESET}}$	7	8	DSDI
TVCC	9	10	DSDO

\*The pin is reserved for internal use within the PEmicro interface, no connection needed.



### 3.20.8 PORT H: 20-Pin Debug Connector (Kinetis, S32 (ARM), other PEmicro-Supported ARM devices, Infineon TriCore - DAP only)

#### 3.20.8.1 JTAG Mode Pin Assignments

The Cyclone provides a 20-pin 0.100-inch pitch double row connector for ARM targets. The location of this header is indicated as **PORT H** under Part# CYCLONE-FX-UNIV in **Figure 3-11**. The 20-pin standard connector pin definitions for JTAG mode are as follows:

##### 20-Pin Standard Connector JTAG Mode Pin Assignments

PIN 1 - <u>TVCC</u>	<b>NC</b> - PIN 2*
PIN 3 - <u>TRST</u> or <b>NC*</b>	<b>GND</b> - PIN 4
PIN 5 - <b>TDI</b>	<b>GND</b> - PIN 6
PIN 7 - <b>TMS</b>	<b>GND</b> - PIN 8
PIN 9 - <b>TCK</b>	<b>GND</b> - PIN 10
PIN 11 - <b>NC*</b>	<b>GND</b> - PIN 12
PIN 13 - <u><b>TDO</b></u>	<b>GND</b> - PIN 14
PIN 15 - <u><b>RESET</b></u>	<b>GND</b> - PIN 16
PIN 17 - <b>NC*</b>	<b>GND</b> - PIN 18
PIN 19 - <b>NC*</b>	<b>GND</b> - PIN 20

\*The pin is reserved for internal use within the PEmicro interface.

#### 3.20.8.2 SWD Mode Pin Assignments

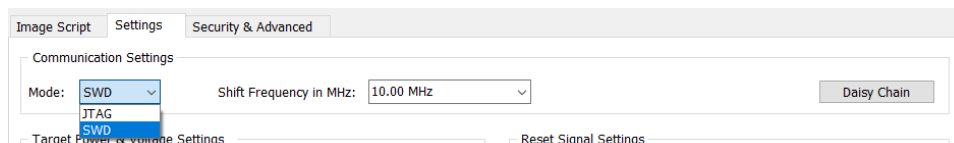
**Cyclone FX** programmers also support SWD Mode. This replaces the JTAG connection with a clock and single bi-directional data pin.

##### 20-Pin Standard Connector SWD Mode Pin Assignments

PIN 1 - <u>TVCC</u>	<b>NC*</b> - PIN 2
PIN 3 - <u>TRST</u> or <b>NC*</b>	<b>GND</b> - PIN 4
PIN 5 - <b>NC*</b>	<b>GND</b> - PIN 6
PIN 7 - <b>TMS/SWDIO</b>	<b>GND</b> - PIN 8
PIN 9 - <b>TCK/SWCLK</b>	<b>GND</b> - PIN 10
PIN 11 - <b>NC*</b>	<b>GND</b> - PIN 12
PIN 13 - <b>NC*</b>	<b>GND</b> - PIN 14
PIN 15 - <u><b>RESET</b></u>	<b>GND</b> - PIN 16
PIN 17 - <b>NC*</b>	<b>GND</b> - PIN 18
PIN 19 - <b>NC*</b>	<b>GND</b> - PIN 20

\*The pin is reserved for internal use within the PEmicro interface.

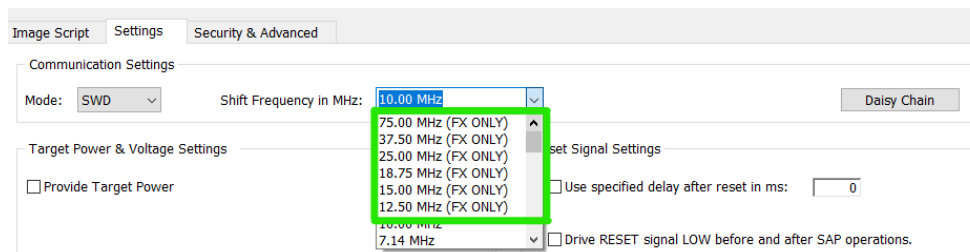
SWD Mode is selected from the “Communication Mode” drop-down box in the Cyclone Image Creation Utility:



**Figure 3-28: Communications Mode Selection**

##### 3.20.8.2.1 High-Performance Communications

If high-performance options are available for the selected device they will appear in the “Shift Frequency in MHz” drop-down. **Cyclone FX** programmers are capable of high-performance communications when using certain ARM Cortex targets in SWD mode.



**Figure 3-29: High-Performance Options**

### 3.20.8.3 DAP Connector Pin Assignments (Target Board Does Not Have Dedicated DAP Connector Port)

Users whose target board has a dedicated DAP Connector should instead refer to Port A in **Section 3.20.1.3 - DAP Connector Pin Assignments (Target Board Has Dedicated DAP Connector Port)**.

If the user's target board does not have a dedicated DAP connector port, the user may choose to wire pins to the 20-pin 0.100-inch pitch double row connector of the CYCLONE-FX-UNIV model (can be used with Infineon TriCore targets AUDO™ TC1xx and AURIX™ TC2xx/TC3xx). The location of the this header is indicated as **PORT H** in **Figure 3-11**. The 20-pin standard connector pin definitions for DAP connectors are as follows:

#### 20-Pin Standard Connector JTAG Mode Pin Assignments

PIN 1 - <b>TVCC</b>	<b>NC</b> - PIN 2*
PIN 3 - <b>TRST</b> or <b>NC*</b>	<b>GND</b> - PIN 4
PIN 5 - <b>NC*</b>	<b>GND</b> - PIN 6
PIN 7 - <b>DAP1</b>	<b>GND</b> - PIN 8
PIN 9 - <b>DAP0</b>	<b>GND</b> - PIN 10
PIN 11 - <b>NC*</b>	<b>GND</b> - PIN 12
PIN 13 - <b>NC*</b>	<b>GND</b> - PIN 14
PIN 15 - <b>RESET</b>	<b>GND</b> - PIN 16
PIN 17 - <b>NC*</b>	<b>GND</b> - PIN 18
PIN 19 - <b>NC*</b>	<b>GND</b> - PIN 20

\*The pin is reserved for internal use within the PEmicro interface.

**Note:** **Pin3 - TRST** needs to be high at power on reset release for enabling DAP, so short circuiting TRST to VDD may be advised.

### 3.20.9 PORT C & PORT E (Renesas Adapter)

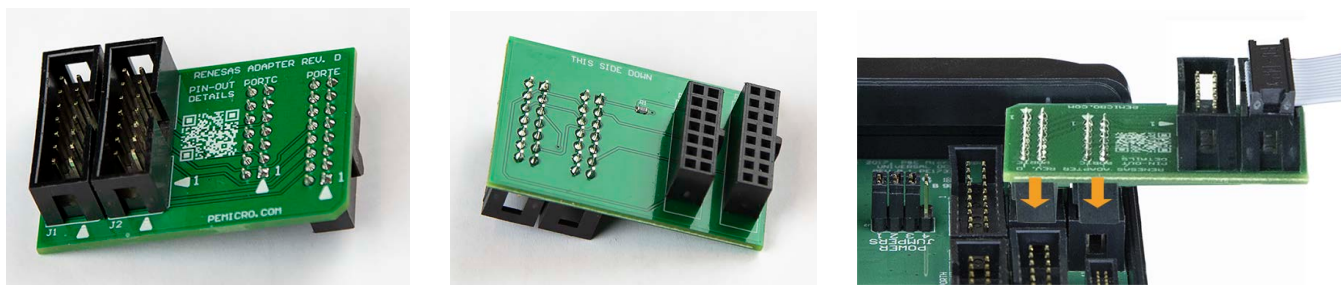
PEmicro also offers a separate Renesas adapter (part# CUFx-RENEsas-ADPT) that can be plugged into the 14-pin and 16-pin headers of the Cyclone (PORT C & PORT E - see **Figure 3-27**). The adapter connects to the target via a 14-pin ribbon cable.

**Note:** The user should take care to properly align the 2 adapter headers over the Cyclone's PORT C and PORT E pins before pressing the adapter down to install.

Rev. D of the adapter features 2 headers on the top side, labeled J1 and J2. The labels are visible in the first Figure below. **The J2 header should be used for RL78 devices.** Otherwise the J1 header should be used. Revs. A-C feature only one header on the top.

#### **Renesas Pin Signals (Requires Renesas Adapter, sold separately)**

Signal definition depends on the specific Renesas architecture selected. Please view the appropriate Renesas device datasheet.

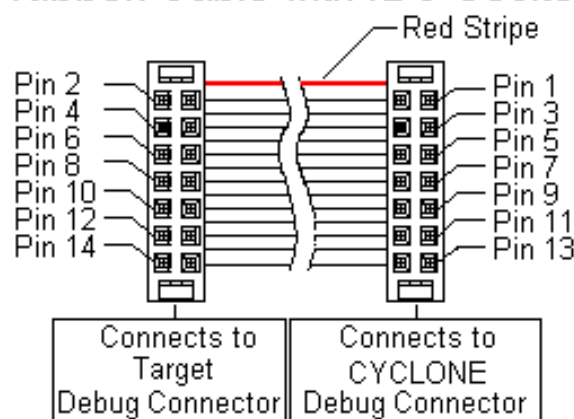


**Figure 3-30: Renesas Adapter Rev. D: 1) Top, 2) Bottom, 3) Connected To 14-Pin & 16-Pin Headers of Cyclone\_Universal\_FX (Adapter Sold Separately)**

### 3.21 Ribbon Cable

**Cyclone FX** programmers communicate with the target through ribbon cables. The ribbon cables for standard debug connectors have a 0.100-inch centerline dual row socket IDC assembly (not keyed). The ribbon cables for 10- and 20-pin mini debug connectors have a 0.050-inch centerline dual row socket IDC assembly (keyed). The ribbon cables are designed such that the Cyclone's Debug Connector has the same pinout as the Target Header, i.e., Pin 1 of the Cyclone's Debug Connector is connected to Pin 1 of the Target Header. As an example, **Figure 3-31** sketches the connection mechanism (looking down into the sockets) for a 14-pin ribbon cable. Ribbon cables for other supported architectures use a similar scheme, but may have more or fewer pins.

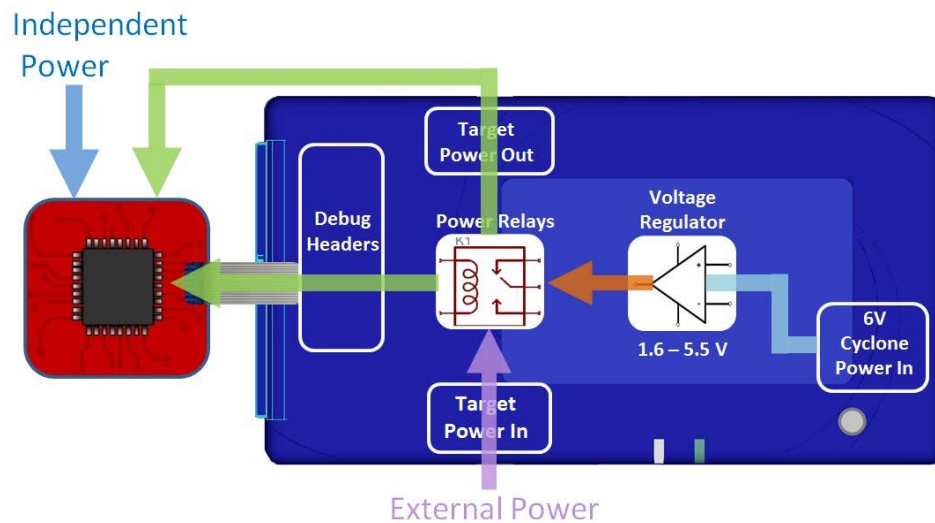
#### Ribbon Cable with IDC Socket



**Figure 3-31: Ribbon Cable Example Diagram, When Looking Into IDC Socket**

## TARGET POWER MANAGEMENT

Different target devices may require different power schemes which depend on the design of the target board, target voltages, and even the device architecture. PEmicro has designed the **Cyclone FX** to be capable of powering a target before, during, and after programming. Power can be sourced at many voltage levels from the Cyclone itself, or sourced by an external power supply and switched by the Cyclone.



**Figure 4-1: Five different paths to power a target**

The versatility of the Cyclone power scheme gives the user the utmost flexibility, and includes the following features:






- Provides power through a power jack or through the debug connector
- Provides internally generated voltage from 1.6v-5.5v at up to 500mA
- Switches an external power supply voltage, up to 24V at 1amp
- Selectively powers the target before, during, and after programming
- Powers down the target connections between programming operations
- Uses power switching to aid entry into debug mode for certain targets
- Provides target voltage and current measurement capabilities

If target power is required, each target board may vary where the power is sourced from, externally or internally, and how it is channeled to the target: through the debug header or to a separate connector to the board. Power that is passed through and managed by the Cyclone goes through power relays so it can be power cycled. This is extremely useful because it also allows the power to be off during setup and automatically powered on by the Cyclone for programming. For some devices, the process of entering debug mode requires that the device be powered down and powered back up. Power can also be left in a desired power state, either on or off.

### 4.1 Cyclone Configuration

There are two different places Power Management is configured and they should be **matched**: first, by the **jumpers** on the **Cyclone FX**, and second, in the **setup** of the programming image. The Cyclone jumpers are the most important because they are the physical connection to the target. The Cyclone has an easy access panel that reveals debug header connections for a variety of different architectures, and a 2x4 jumper block for configuring power management of the target. The specific location of the jumpers is indicated by the label POWER JUMPERS in **Figure 4-3**. This set of 4 jumpers can be used to set 5 different power management schemes for the target.

**Note:** If these jumpers are not set correctly, the Cyclone will not function as intended.

1	Target is powered independently	
2	Power provided externally (center +) and managed by Cyclone, power out to debug ribbon cable.	
3	Power provided externally (center +) and managed by Cyclone, power out to 2.5 mm output jack (center +)	
4	Power provided by Cyclone, power out to debug ribbon cable	
5	Power provided by Cyclone, power out to 2.5 mm output jack (center +)	

**Figure 4-2: Cyclone Power Schemes & Corresponding Jumper Settings**

The bottom edge of the **Cyclone FX** has a Power In jack for externally provided power, and the top edge of the Cyclone has Power Out jack, for when power schemes including these are used (see **Figure 4-3**). One of the provided ribbon cables is connected to the appropriate debug header based on the specific target architecture.

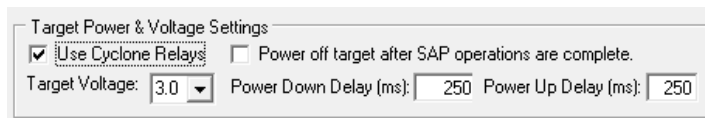


**Figure 4-3: Cyclone Hardware Features: Power Jumpers and Target Headers**

The power settings that are set by the jumpers are a physical connection and take precedence.



After the basic hardware setup, target power and voltage settings are also set in the creation of a SAP (stand-alone programming) image. At a minimum the SAP image contains all the commands to Erase, Program, and Verify a programming image. More sophisticated power selections in the SAP image can control the relays, target voltage, delays, and power down after SAP operations, as shown in the selection dialog.



**Figure 4-4: Target Power & Voltage Settings**

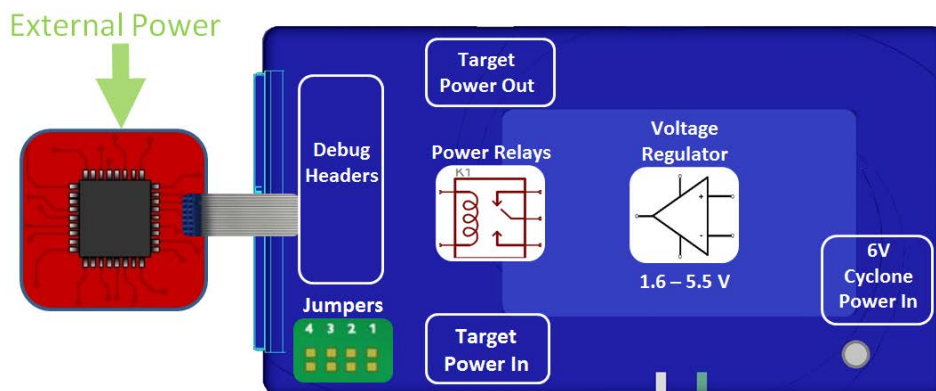
Target voltages (with appropriate jumper settings) in the range of 1.6 to 5.5 volts may be provided. There is also the option to select the internal Cyclone relays to power cycle the Cyclone during programming, and set the length of delays during power up and down. This is extremely useful to make sure the power is off when hooking up the target. Power cycling is especially important for architectures that require it to enter debug mode. The SAP image settings may even be used to turn off the target power once programming is completed, to ensure that the microcontroller is left in a halted state and not running.

## 4.2 Cyclone Setup

Below is a tutorial that demonstrates how to set up the **Cyclone FX** in each of the 5 power configurations. A very common configuration is the independently powered target. In this power scenario, the Cyclone will detect and use the power on the target for the appropriate debug communication voltages.

### 4.2.1 Independently Powered Target

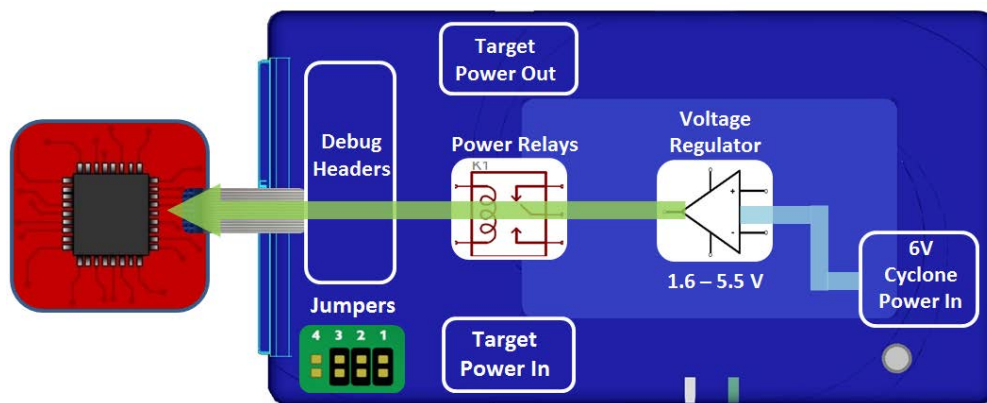
In the simplest and most common scenario, no jumpers are set, so the target is powered independently from the Cyclone. No power is passed through the debug header, just the standard debug signals. The Cyclone automatically detects the target power and sets the debug signals to match.



**Figure 4-5: Independently Powered Target**

### 4.2.2 Power provided by the Cyclone to the debug cable

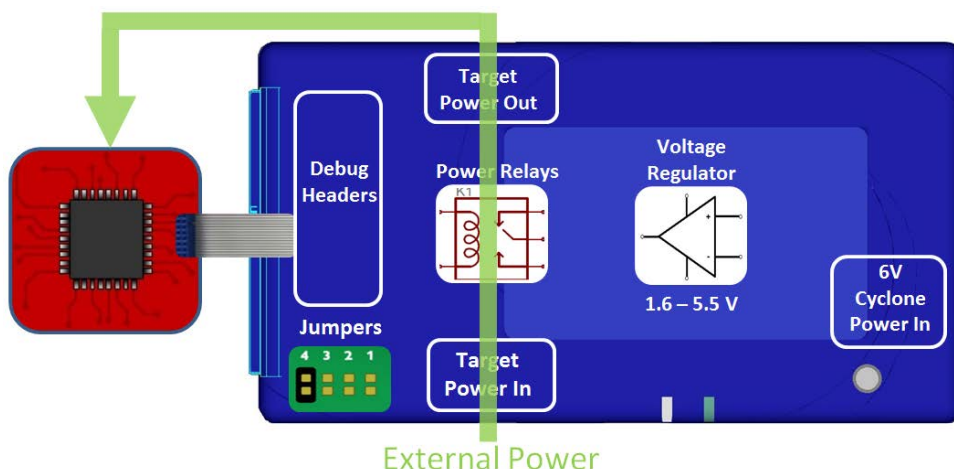
It is also possible for the Cyclone to generate power through an internal regulator in the range of 1.6 to 5.5 Volts. In the jumper configuration below, the Cyclone generates the power through a voltage regulator, and passes it through the power relays and out through the debug ribbon cable, which is set up during the SAP image creation. There is only one connection to the target processor which will handle both the communication and the power. In this scenario, external power must not be connected to the Power In jack since it is already being provided.



**Figure 4-6: Power Provided by the Cyclone to the Debug Cable**

#### 4.2.3 External Power passed through the Cyclone and out 2.5 mm barrel port

It is also possible to provide external power, passed through the Cyclone power relays, and back out to be available to power the target board externally. This is useful when the user wants to control the power to the target and the target board has an external power connector. Setting a single jumper will connect the barrel port input connector on the bottom edge of the Cyclone, through the relays, to a matched 2.5 mm barrel port output connector on the top edge of the Cyclone, so that the power can be routed into and back out of the Cyclone.



**Figure 4-7: External Power Passed Through the Cyclone and Out 2.5 mm Barrel Port**

#### 4.2.4 External Power passed through the Cyclone to the debug cable

In a slightly different scenario, the user may wish to provide power to the target through the debug cable. On the bottom edge of the Cyclone is a 2.5 mm Power In port barrel which will pass power through target relays which lets the Cyclone take control of the power cycling during programming. This simple setup requires only an input to the Cyclone and a single ribbon cable connection to the target board that handles both communication and power. The external power provided must be between 1.6 to 5.5 volts.

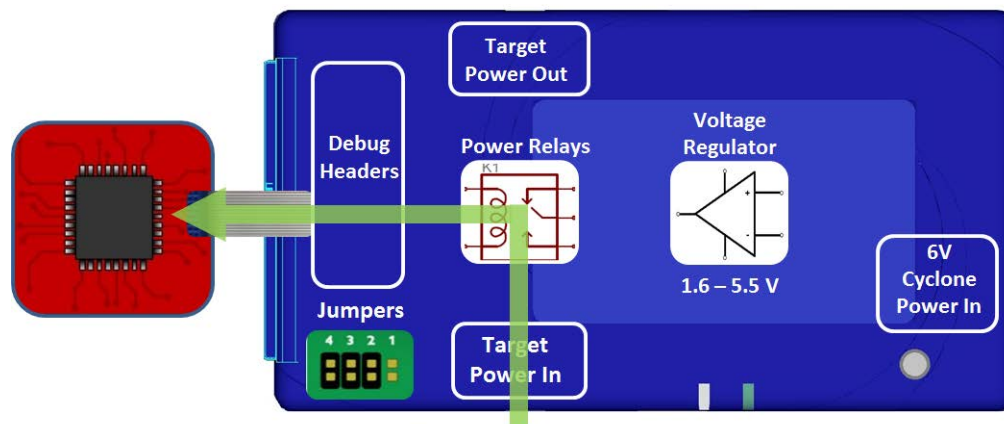


Figure 4-8: External Power Passed Through the Cyclone to the Debug Cable

#### 4.2.5 Power provided by the Cyclone and out 2.5 mm barrel port

In a slightly different scenario, the user may wish to have the Cyclone provide power, but power the target via an external connector on the target. The voltage supplied to the target is determined by the settings in the SAP image. When generating the SAP image the Cyclone relays must be selected as well as the correct voltage level for the target.

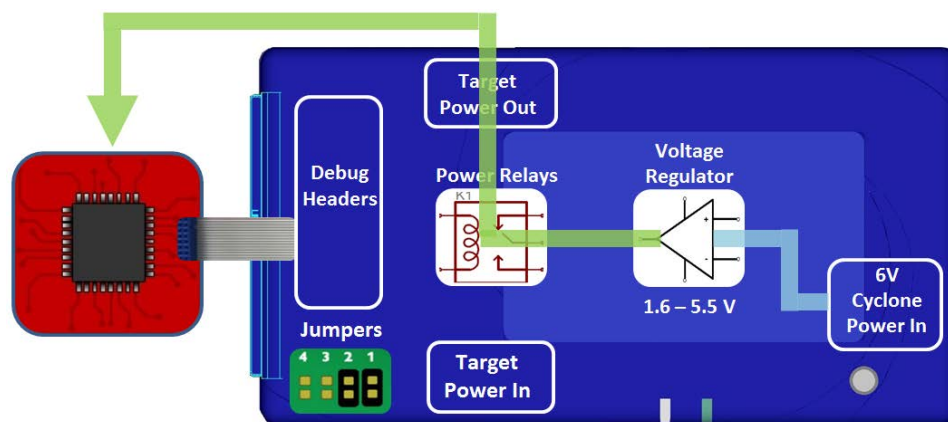


Figure 4-9: Power Provided by the Cyclone and Out 2.5 mm Barrel Port

### 4.3 Setup Reminders

The most important step when providing power out to a target is to check the Cyclone's **jumper settings** to make sure they match the intended power setup. The jumpers control the power settings which determine how power is supplied, regardless of the SAP image settings. If the jumpers are set for power to be provided through the Cyclone, and the target is externally powered, this is a conflict and may cause damage to the board.

In the case where power is being supplied through the Cyclone and the target is not being powered on, the user should **first check the jumper settings** to make sure they match the intended power setup. Second, the user should check to make sure the SAP image has the 'Use Cyclone Relays' box checked with the appropriate voltage level selected.

## 5 TOUCHSCREEN LCD MENU

This chapter describes the Cyclone's touchscreen LCD menu. **Figure 5-1** shows an overview of the menu structure.

**Note:** This menu will change as features are added to the **Cyclone FX**, so if your menu does not match what is displayed here, please check PEmicro's website, [www.pemicro.com](http://www.pemicro.com), for a user manual containing the latest LCD Menu operations information.



















### 5.1 Home Screen

The home screen appears when the **Cyclone FX** is powered on, or when the  Home button is tapped.

#### 5.1.1 Icons

A row of icons in the upper right corner indicates the status of various attributes of the Cyclone.

**Note:** The user may tap on the row of icons to view the meaning of each of the currently displayed icons.

<b>Cyclone Unit Status:</b> Ok / Bad		
<b>Programming Status:</b> Ready / Busy		
<b>USB-To-PC Enumerated:</b> Yes / No		
<b>Real-Time clock Enabled &amp; Working:</b> Yes / No		
<b>Cyclone Power Relays:</b> Closed / Open		
<b>Target Device Is Powered*:</b> Yes / No		
<b>SDHC Memory Card:</b> None / Valid / Unformattd / Reset Cyclone**		
<b>PEcloud:</b> Provisioned for PEcloud and Online		
<b>Barcode Scanner:</b> Detected / Not Detected		

\* Target Device Is Powered - "Yes" indicates that the **Cyclone FX** detects power on the Vcc pin of the target device programming header.

\*\* SDHC Memory Card (Cyclone FX Only, and legacy Cyclone LC with license) - "Reset Cyclone" indicates that the Cyclone needs to be reset before the SDHC card will register as Valid. The user can push the Reset button which is located on the front side of the Cyclone, below the LED indicators.

#### 5.1.2 Configurable Display Area

The main area of the home screen can be configured to optionally display the following information, by using the Cyclone IP Configuration Utility (see **Section 5.2.4.4.4 - Configure Home Screen**):

1. Firmware version of the Cyclone (always shown).
2. IP address assigned to the Cyclone.
3. Name assigned to the Cyclone.
4. Number of programming images in the Cyclone's memory.
5. Name of the selected programming image.
6. First serial number associated with the selected image
7. Current status.
8. Results of the last operation performed.
9. Time and date.



10. Status Window and Main Menu button (always shown).
11. Programming count & limit
12. Target voltage and/or current

### 5.1.3 Status Window

The status window appears in the lower left corner of the home screen and displays the results of programming operations.

### 5.1.4 Error Information Icon

When the Cyclone experiences an error during programming operations, the Info icon will appear to the left of the Menu button (or AUX button, if configured).



Press the Info icon to view a detailed description of the error.

### 5.1.5 AUX Button (Appears If Configured)

The Cyclone allows the user to add an Auxiliary (AUX) button to the home screen which will perform a specific function when pressed. The specific function is chosen by the user when the AUX button is configured. The AUX button will appear on the home screen to the left of the “Menu” button, in the lower right corner of the home screen.



**Figure 5-1: AUX Button On Home Screen (configured to perform CRC32 function)**

For information on how to configure the AUX button, see **Section 5.2.5 - Status**.

## 5.2 Main Menu

The Main Menu is accessible by pressing the “Menu” button when the Home Screen is displayed. The Main Menu contains the following selections.

### 5.2.1 Select Local Image or Cloud Job

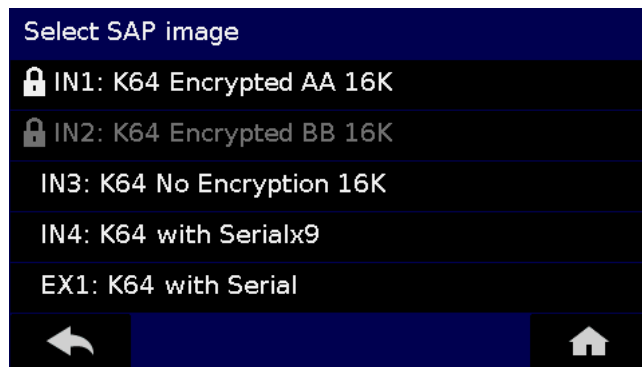
The Cyclone can store two type of files for programming: local Images and PEcloud Jobs. To set a specific file as the selected file for programming you will first need to choose between these two types, as the latter will access PEcloud to present the most current list of available Jobs.

#### 5.2.1.1 Local Programming Images

Displays a list of the available local programming images so that the user may select one for programming. Images in the Cyclone’s internal memory are preceded by the designation “**IN**” and numbered sequentially, i.e., **IN1: first image name**, **IN2: second image name**. If the SDHC port of a Cyclone contains a memory card, any programming images that reside on the SD card will be preceded by the designation “**EX**” in similar fashion (SDHC Port is Cyclone FX only, and legacy Cyclone LC with license).

Encrypted (eSAP) images will display a lock icon to the left of the image description. If the ImageKey needed to decrypt an eSAP image is missing from the Cyclone, the image will appear grayed out. and cannot be selected for programming until the appropriate ImageKey is loaded onto the Cyclone (see image #2 in **Figure 5-2**).





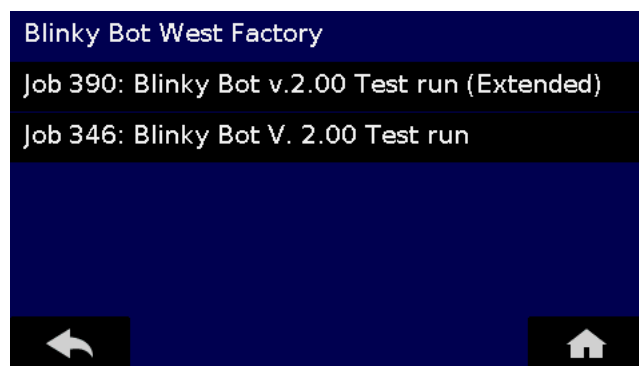
**Figure 5-2: Lock Icon Denotes Encrypted Images**

You may tap the appropriate image to select it. The image name shown is the Image Description specified on the Image and Job Deployment Manager form of the Cyclone Configuration Utility when saving the image to the Cyclone/SD card.

### 5.2.1.2 Cloud Programming Jobs

Cloud programming jobs are stored on PEcloud and assigned to specific Virtual Factories. When selected, this menu will prompt the user to connect to PEcloud, if not currently connected. It will also prompt the user to provision the Cyclone for PEcloud if it is not provisioned.

Once connected to PEcloud it will update the LCD display to show the current listing of jobs available to this specific Cyclone. If the Cyclone is a member of a Cyclone Group in PEcloud then that Cyclone Group is associated with a Virtual Factory (VF). The VF name is displayed at the top of the LCD screen. This screen displays any available Jobs assigned to that VF (some Jobs may be paused or finished).



**Figure 5-3: Listing of PECloud Jobs Available to Cyclone**

Jobs are numbered and named. Selecting a job will make it the default item for programming, and the Virtual Factory and Job description will be displayed on the home screen. For more about programming Jobs and PEcloud, please refer to **CHAPTER 10 - PEcloud - SECURE MANAGEMENT OF REMOTE PROGRAMMING**.

### 5.2.2 View Job Details

This option displays a list of PEcloud Jobs available to the Cyclone. Clicking on a specific job will display data related to the job, such as the Job ID, Job Name, ESAP UID, ESAP Name, and PE Key UID and Key Name. Use the arrows to scroll up and down. This information is not often used; contact PEmicro if you need more information about this page.

### 5.2.3 Current Image Options

This menu presents options that allow the user to select or configure programming Images on the Cyclone **FX**.

**Note:** These apply to Images only and not to any PEcloud Jobs that might be loaded onto the Cyclone, with the exception of **Section 5.2.3.1.1 - Launch Programming**, which will launch programming if a Job is currently selected.

### 5.2.3.1 Execute Image Function

Execute Specific SAP Function presents four Stand-Alone Programming functions that you may execute by tapping the function that you wish to execute:

#### 5.2.3.1.1 Launch Programming

This allows the user to execute the programming function. The **Cyclone FX** will program the target device, if able, using the currently selected programming image. This is functionally equivalent to pressing the Start button.

#### 5.2.3.1.2 Verify Data In Target

Performs a verify function on the data that has been programmed into the target device.

#### 5.2.3.1.3 Toggle Power

Toggles the target power and makes sure all ports are driven to debug mode level.

#### 5.2.3.1.4 Power Cycle Device To Run User Code

Toggles the target power and maintains tri-state mode for all signals.

#### 5.2.3.1.5 Validate Image CRC32

Allows the user to perform a CRC32 validation on the currently selected programming image.

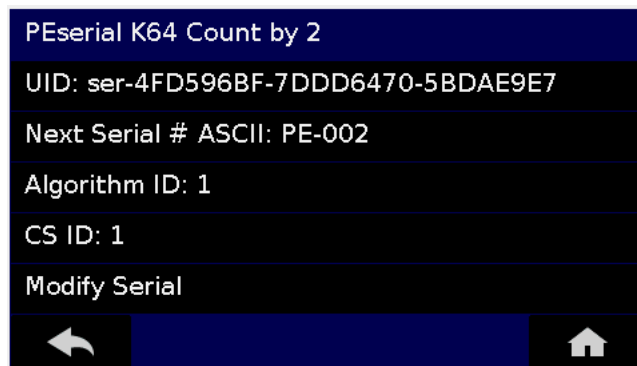
### 5.2.3.2 Set Image Validation

Allows the user to choose between two validation settings: 1) validate the image each time the Start button is pressed, or 2) do not validate the image.

### 5.2.3.3 Serial Numbers

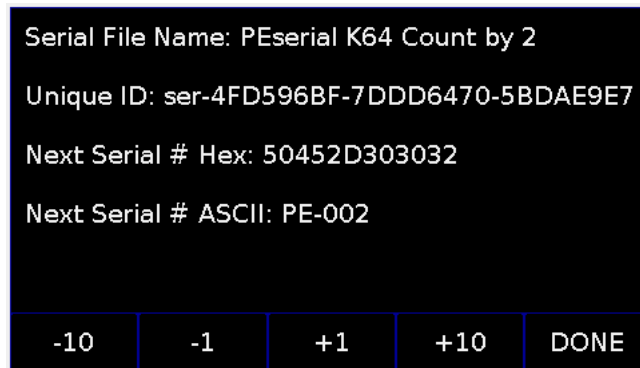
Displays the serial number information associated with the currently selected programming image. If there is none, it will display, "This Image Contains No Serial Numbers." There may be one or more Serial Files associated with the image. The user can click on a specific Serial File name to see the following information about that Serial File:

- UID - Unique identifier of the serial number
- Next Serial # ASCII - Next serial number to be programmed (shown in ASCII format)
- Algorithm ID - Displays the algorithm ID of the serial file
- CS ID - Displays the ID of the CS (Choose Serial) command in the SAP file.



**Figure 5-4: Serial File Selection**

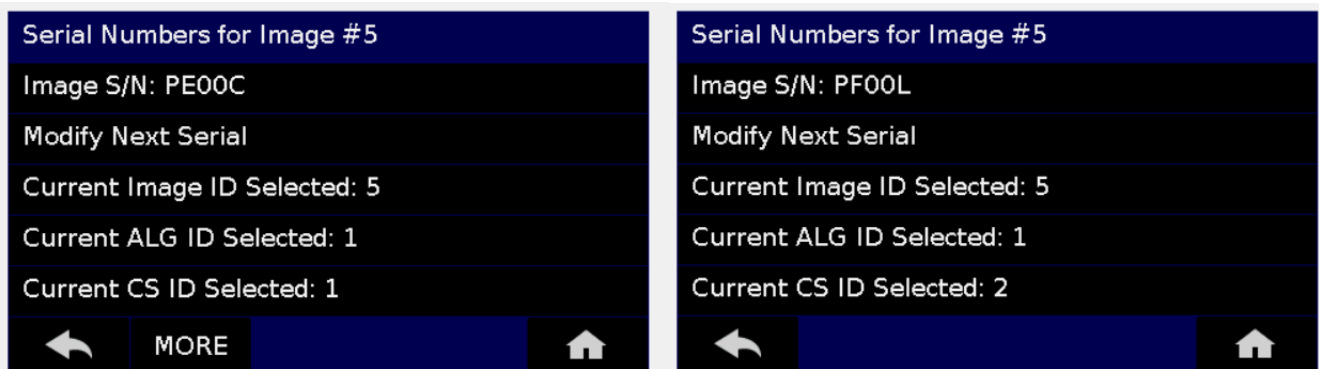
The user can also click on “Modify Serial Number” to edit that serial number. It is possible to Decrease or Increase the Next Serial by -10, -1, +1, +10. This is often done to address issues in the production process, such as during initial setup.



**Figure 5-5: Increase or Decrease Serial Number**

The adjustment buttons will display “Increase Not Allowed” and “Decrease Not Allowed” if the image/algorithm/CS file that the user has selected does not allow for this operation.

**Note:** Older serial files are specific to one programming image, unlike PEmicro’s current serial files. If viewing Serial Number info for an image that references one or more older serial files, the Cyclone will not show links to the serial files but will instead display serial number info for the first serial file. The user can then Modify Next Serial to change the next serial number, or click MORE at the bottom to proceed to the next serial file (if any).



**Figure 5-6: Serial Number View With Older Serial Files (CS IDs #1 & #2)**

#### 5.2.3.4 Show Image Restriction Stats (Requires FX or ProCryption Security Activation)

Displays current statistics, if any, for Image Programmed Count & Maximum Allowed, Errors Logged & Maximum Allowed, and Date Range Allowed. These limits can be set in the Security Features section of the Cyclone Image Creation Utility (see **Section 6.1.10.2 - Image Restrictions**).

**Note:** When *Current Image Stats* is displayed as a home screen item, only Image Programmed Count & Maximum Allowed are displayed on the home screen.

#### 5.2.3.5 Show Run Test Logs (Cyclone FX Only)

The user may program and run a series of custom test applications in the target processor before final programming is allowed to occur. This screen displays information logged during this process. See **CHAPTER 15 - RUN TEST - CUSTOM TEST APPLICATIONS IN THE PROGRAMMING SEQUENCE**.

## 5.2.4 Configure Cyclone

Presents options that allow the user to choose to configure the **Cyclone FX** name, network settings, PEcloud settings, time/date settings, and LCD touchscreen display settings, or to set the display to dynamic.

### 5.2.4.1 Edit Cyclone Name

Allows the user to edit the name of the Cyclone using the on-screen keyboard. Click “Done” to save the new Cyclone name or “Cancel” to exit without saving a new Cyclone name. This name will be displayed on the **Cyclone FX** home screen if the Cyclone is configured to do so.

### 5.2.4.2 Cloud Settings

Provides options for Cyclone provisioning and PEcloud logout.

#### 5.2.4.2.1 Provision/De-Provision Cyclone

Will display an option to either provision or de-provision the Cyclone, depending on its current state. To provision a Cyclone, type in the 8-digit code associated with that Cyclone in PEcloud and enter. If the user believes there is an already created but unused provision code they would like to use, they can check Cyclone Groups in the PEcloud interface for unused provision codes shown in red, as depicted below.

cyclone group actions
+ provision new cyclones

cyclone	model	actions
unprovisioned Cyclone PIN: 4859#### (2616) 8-digit Code: <b>4859####</b>	Undefined Rev. 0 UID: -1	actions

Please Enter the Registration Code  
4859####|

0	1	2	3	4	5	6	7	8	9
q	w	e	r	t	y	u	i	o	p
a	s	d	f	g	h	j	k	l	@
.	z	x	c	v	b	n	m	SHIFT	
DONE		CANCEL		SPACE			DEL		

Figure 5-7: a) Provisioning Code From PEcloud Cyclone Group, b) Cyclone Entry Screen

See **Section 10.5.2.2 - Provision New Cyclones** for more information about generating provision codes using a Cyclone Group action.

When de-provisioning a Cyclone, if the action is resolved properly the screen will report a Success, otherwise it will indicate an error with a recommended solution.

#### 5.2.4.2.2 Cloud Log Out

Will disconnect the Cyclone from communications with PEcloud.

### 5.2.4.3 Configure Network Settings

Presents options that allows the user to view or edit various IP settings, toggle the IP settings between static and dynamic, and re-name the **Cyclone FX**.

#### 5.2.4.3.1 Show Current IP Settings

This menu allows you to view the **Cyclone FX** IP address, Mask, and Gateway, MAC address, and DNS Server address. You may also tap these entries to edit, as long as the Cyclone is set to Static IP mode.

### **Dynamic vs. Static**

There are two schemes for assigning IP addresses. One is the Static IP addressing mode. This involves the user manually setting the IP address for every device on the network. In this case, it falls to the user to ensure the IPs assigned do not conflict and are within the boundaries of the network. The other is the Dynamic Host Configuration Protocol (DHCP). This involves setting up a separate server to manage the IP addresses. The server is given a list of valid IP addresses for the network. Using a predetermined set of rules, each new device that wishes to connect to the network is given an IP address by the server. This takes the task of managing the validity and uniqueness of IP addresses out of the user's hands and relegates it to the server. **Cyclone FX** programmers are capable of using either Static IP addressing or DHCP.

#### **5.2.4.3.2 Edit Static IP Settings**

This menu allows you to edit the Cyclone's IP address, Mask, and Gateway, and view the Cyclone's MAC address. If you are unable to edit these values, you may wish to check to be certain that the Cyclone is not set to Dynamic IP mode.

##### **IP**

Edit IP Numbers allows the user to set an IP number for the Cyclone. The current IP number is displayed on the second line. Tap a number to edit and use the touchscreen keyboard to set the new number. When you are finished, hit Done. If you change your mind and decide not to save, hit Cancel to leave the IP number as is and return to the Main Menu.

##### **Mask**

Edit IP Mask allows the user to set an IP Mask for the Cyclone. The current IP Mask is displayed on the second line. Use the Up/Down buttons to scroll through the characters. To select a character, hit the Select button. When you are finished, scroll through the characters until you reach the -> (right-arrow) character. Selecting this character will complete the process. The default IP mask is 255.255.255.0.

##### **Gateway**

Edit IP Gateway allows the user to set the IP Gateway for the Cyclone. The current IP Gateway is displayed on the second line. Use the Up/Down buttons to scroll through the characters. To select a character, hit the Select button. When you are finished, scroll through the characters until you reach the -> (right-arrow) character. Selecting this character will complete the process.

##### **MAC Address**

Displays the current MAC address for the Cyclone.

##### **DNS Server**

Displays the current DNS Server address for the Cyclone.

#### **5.2.4.3.3 Enable/Disable Dynamic IP**

Allows the user to toggle the Cyclone configuration between utilizing a Static IP address or a Dynamic IP address. The user must reset the **Cyclone FX** after changing from Static to Dynamic or vice-versa. The reset button on the front side of the unit may be used.

#### **5.2.4.4 Configure Screen**

This menu presents options that allow the user to adjust or customize the Cyclone's LCD touchscreen display in various ways.

##### **5.2.4.4.1 Change Screen Brightness**

Allows the user to adjust the brightness of the LCD touchscreen. The "Increase" and "Decrease" buttons will raise or lower the brightness level, respectively, in increments of 10%. Brightness can be adjusted from between 100% - 10%. Press "Done" to exit.

##### **5.2.4.4.2 Calibrate Screen**

Allows the user to click on specified points on the LCD touchscreen in order to calibrate the



accuracy of the touch function. Follow the on-screen instructions.

#### 5.2.4.4.3 Set Progress Details

This configures the display to present more detailed information during the progress of programming, including the specific programming steps that are performed and specific information about the programming and verifying procedure. The user may select “Show Details, Keep Last Progress On,” “Show Progress Details,” or “Hide Progress Details.”

#### 5.2.4.4.4 Configure Home Screen

This menu allows you to choose what information to display on Lines 2-8 of the home screen. Available elements to display consist of information such as: the current IP address, the Cyclone name, the number of images, etc. In this way the user can customize the display to provide the information that they find most useful. There is a separate button for each of Lines 2-8. Tapping on the button for a specific Line brings up a list of elements that you can choose to display on that Line of the home screen. If the list of elements is greater than one page, tap the More button to view the rest of the available elements. Tap the element that you want to display on that line and then tap Done to save your selection.

**Note:** The **Cyclone FX** allows one additional option for display: the target device’s voltage and/or current.

#### 5.2.4.5 Configure Storage

This menu selection allows the user to format the Cyclone’s internal memory. This menu will also allow a Cyclone FX user to format an SD card located in the Cyclone FX’s memory expansion slot. Select “Format Internal Storage” or “Format External SD Card”. (SDHC Port is Cyclone FX only, and legacy Cyclone LC with license). The user will be prompted to ensure that they wish to format the corresponding memory. Tap “Yes” to format, or “Cancel” to go back to the previous menu option without formatting the memory.

#### 5.2.4.6 Configure Time Settings (Cyclone Time / Real Time Clock)

The **Cyclone FX** is equipped with a Real Time Clock (RTC) module designed to keep accurate timing even when the Cyclone is turned off. The Date & Time are displayed on the home screen.

This menu presents options that allow the user to configure the Cyclone’s various date/time/ timezone settings, including formatting options.

##### 5.2.4.6.1 Modify Date / Time

1. **Update Time from Internet** - Connects to an SNTP server, fetches the current time, and saves it to the Cyclone. When executed it displays a message that this can freeze the Cyclone for up to 3 minutes – This is due to an invalid ARP response due to a bad gateway configuration. Proper configuration will ensure the problem is resolved. If the network connection is not configured/connected this displays a message that the time failed to update. If it is successful no message is displayed.
2. **Set Time Zone Hours** - Allows you to set the timezone offset, in hours +/-, from GMT time
3. **Set Time Zone Minutes** - Allows you to set the timezone offset, in minutes +/-, from GMT time

##### 5.2.4.6.2 Set Time-Date Display

Allows you set the Cyclone’s Time-Date Display to one of the following configurations:

1. Display Date Only
2. Display Time Only

### 3. Display Date and Time

#### 5.2.4.6.4 Set Date Formatting

Allows you to select how the date is displayed. The options are:

1. YYYY-MM-DD
2. MM-DD-YYYY
3. DD-MM-YYYY
4. MM/DD/YYYY

#### 5.2.4.6.5 Set Time Formatting

Allows you to select how the time is displayed. The options are:

1. HH:MM (24-hour)
2. HH:MM (AM/PM)
3. HH:MM:SS (24-hour)
4. HH:MM:SS (AM/PM)

#### 5.2.4.7 Configure AUX Button

Allows the user to configure an auxiliary (AUX) button which (if configured) will be labeled appropriately and displayed to the left of the Menu button on the Cyclone's touchscreen LCD. When pressed, the AUX button will perform the task for which it has been configured. The options that may be assigned to the AUX button are:

1. No Operation - No operation is assigned to the AUX button and it will not be displayed on the LCD screen.
2. Perform Verify Only - Verifies the data on the target device against the data in the programming image.
3. Toggle Power - Toggles the Cyclones power relays off/on.
4. Validate Image CRC32 - Validates the CRC32 of the data on the target device against that of the data in the programming image.
5. Power Cycle Device To Run User Code - Toggles the target power and maintains tri-state mode for all signals.
6. Launch Image Programming - Launches the selected programming image. Replaces the hardware Start Button.

#### 5.2.4.8 Enable/Disable Start Button

This menu option gives the user the option to disable the physical Start Button. Programming can then be initiated via the Cyclone Control Suite, or by a digital start button on the Cyclone screen if you have the AUX button set to the "Launch Image Programming" option.

The physical Start Button can always be re-enabled via this same menu toggle.

#### 5.2.4.9 Configure USB Host Device (FX Only)

This menu selection allows the user to either Enable or Disable a barcode scanner connected to the CYCLONE FX's USB Extension Port.

Select **Enable** to provide power to and turn on a connected barcode scanner. The Enable setting is persistent. Once Enabled, the scanner will be powered and turned on whenever the Cyclone is powered. To disable the barcode scanner select **Disable**.

### 5.2.5 Status

This menu contains a selection that allow the user to view status information regarding various aspects of the Cyclone. This menu will likely be expanded with future updates.

#### **5.2.5.1 Shared Serial Numbers**

Displays and Shared Serial Number files accessible to the Cyclone. The user can select one to edit.

#### **5.2.5.2 Show Current IP Settings**

Allows the user to view the Cyclone's Current IP Mode, IP Address, Mask, Gateway, and MAC Address.

#### **5.2.5.3 Show Logs**

Allows the user to view the Barcode Scanner Log and the Last DLM State Log, if these exist.

## 6 CREATING AND DEPLOYING PROGRAMMING IMAGES/JOB

The Cyclone Image Creation Utility is used to create and deploy Programming Images and PEcloud Jobs. For the distinction between these, the user may wish to refer to **Section 6.2 - Stand-Alone Programming Images vs Cloud-Connected Programming Jobs**.

**Note:** The Image Creation Utility is **Windows only**. Users that wish to manage/automate programming via macOS/Linux will need a Windows platform to create programming Images and Jobs.

The tools available in the Image Creation Utility (ICU) help the user to construct one of these self-contained files based on data sets to be programmed and programming instructions. They are then saved or deployed appropriately. During the deployment phase, users with access to ProCryption Security will have the opportunity to add encryption and to restrict some of the programming functions of the Image or Job. One important feature of encryption is that only Cyclones provisioned with the user's custom key will be able to load, decrypt, and an Image or Job that was encrypted using that key.

In-depth information can be found in **Section 6.5.3.1 - Encryption (ProCryption Security)**. A thorough technical description of what is involved in the encryption process itself is available in **CHAPTER 12 - SAP IMAGE ENCRYPTION**.

**Note:** ProCryption Security is provided automatically for any Cyclones making use of the PEcloud service, due to its robust security standards.

### 6.1 Cyclone Image Creation Utility

This section describes in detail how use the Cyclone Image Creation Utility, shown in **Figure 6-1**, to configure and create an Image, Job, or SAPOBJ, and how to deploy an Image or Job so that it can be used for stand-alone programming by a **Cyclone FX**. The **Cyclone FX** does not require a target to be connected during the configuration process, however if a configured Image is to be deployed directly to the Cyclone then it must be powered on and one of the communications interfaces must be connected.

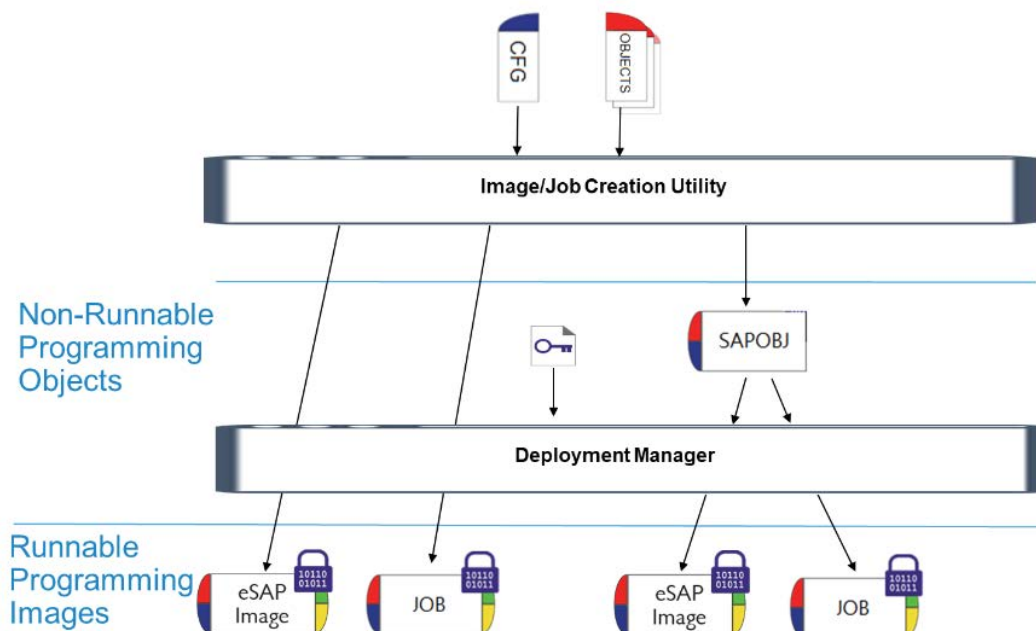
**Note:** The Cyclone Image Creation Utility is for ARM devices only; Image Creation Classic should be launched instead for non-ARM devices.

### 6.2 Stand-Alone Programming Images vs Cloud-Connected Programming Jobs

The Cyclone programmer can execute one of two types of programming objects:

1. **Stand-Alone Programming Images:** Programming Images are completely self-contained archives which hold all data, scripts, and algorithms necessary to program a specific hardware configuration with a specific set of data. There are no external dependencies outside of the image. The Cyclone only needs to be powered and connected to the target to program (though it can be automatically controlled through its Serial/USB/Ethernet as desired).
2. **Cloud-Connected Programming Jobs:** Programming Jobs are very similar to Programming Images except they require a connection to a user's Virtual Factory in the PEcloud to operate. The Cyclone will request from the PEcloud permission to program each target and will report the result to the PEcloud. This gives the user the ability later to start, stop, update, create, and monitor their programming jobs in use on their Cyclones regardless of location. Programming Jobs require the Cyclone to have a network connection to a network which can connect to the cloud.

Both Programming Images and Jobs are based upon a set of programming steps and data files specified by the customer. Programming Images and Jobs can either be built directly from all of the source data files or from a pre-compiled SAPOBJ programming data archive. Below is an image that shows when images and jobs are created directly in the ICU (left), and when they are deployed later by using the SAPOBJ as input for the Deployment Utility (right).



**Figure 6-8: Paths For Creating & Deploying Images & Jobs**

When would the user wish to use an intermediate SAPOBJ archive? This is most commonly done when the user wishes to generate count restricted programming jobs/images and will be generating many such restricted images from the exact same data set. For instance, the user in December generates a programming Job of 500 units for Widget X Version 1.23 and then comes back in February and April to generate jobs of 1000 and 900 units based on the same exact programming data. Having built a SAPOBJ programming data archive, the user wouldn't need any of the original data files or need to re-specify the programming steps to go through (this information is all stored in the SAPOBJ file). The user would just build the Jobs directly from the SAPOBJ archive. More information can be found at:

"The Difference Between SAP Images and Job Images"

[https://www.pemicro.com/blog/index.cfm?post\\_id=261](https://www.pemicro.com/blog/index.cfm?post_id=261)

"The Magic of SAPOBJ"

[https://www.pemicro.com/blog/index.cfm?post\\_id=262](https://www.pemicro.com/blog/index.cfm?post_id=262)

Next we will discuss each area of the ICU and what it is used for.

## 6.2.1 Image Script Tab

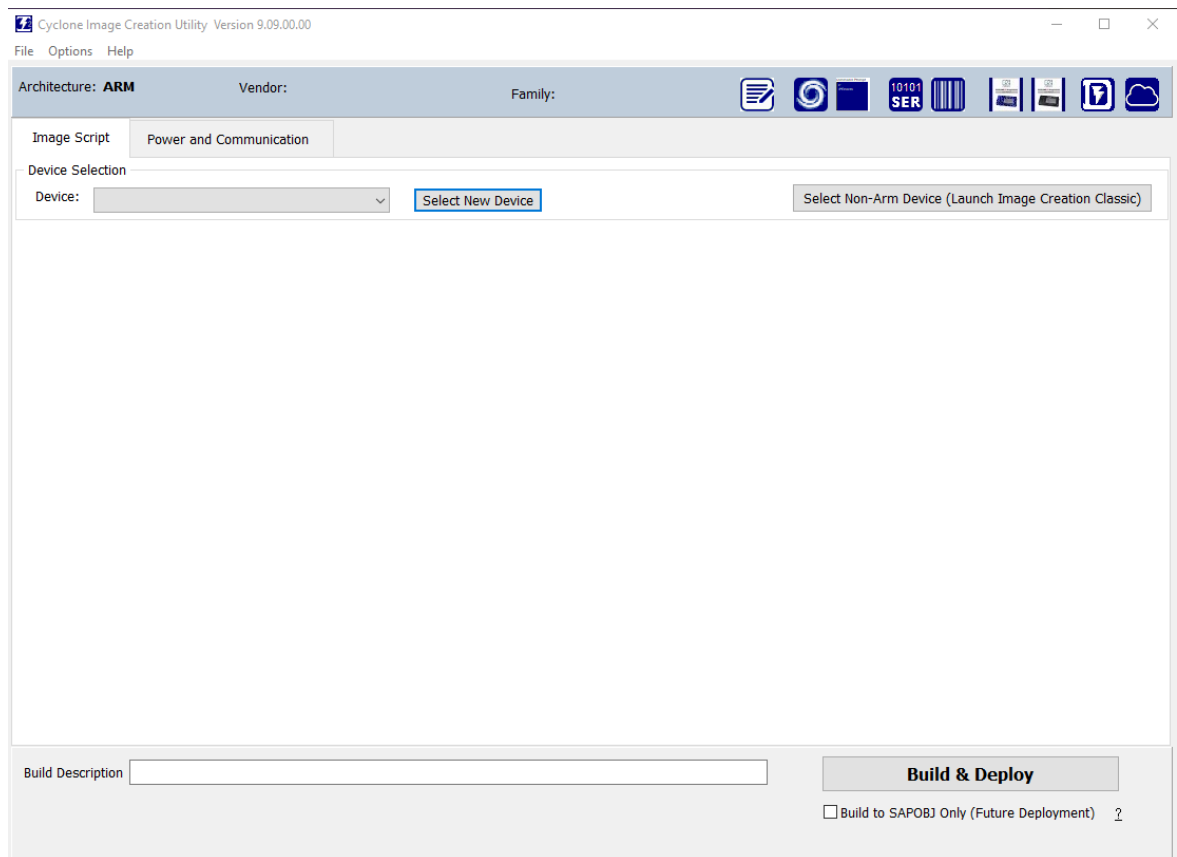
The Image Script tab is where the user will select their device and assemble the programming script, including references to the appropriate programming algorithm, s-record, binaries, serialization data, etc.

### 6.2.1.1 Device Selection - Non-ARM Devices

For non-ARM devices, the user should instead run the Cyclone Image Creation Utility **Classic Version** (CreateImageClassic.exe). If already running the Cyclone Image Creation Utility, the user may click the "Select Non-ARM Device" button to launch the Classic Version.

**Note:** The Cyclone Image Creation Utility and the Classic Version are extremely similar in terms of functionality, they are merely organized differently. The Classic Version is not documented here; it is intended that the user should be able to operate the Classic Version using the concepts and descriptions included here for the Cyclone Image Creation Utility.

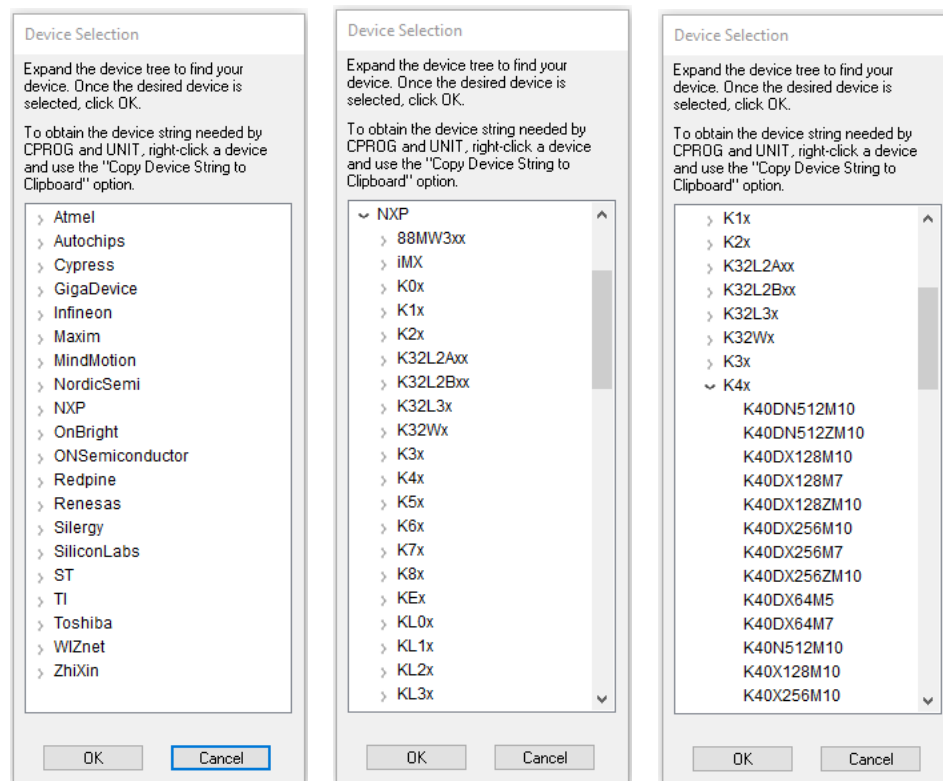




**Figure 6-1: Cyclone Image Creation Utility**

#### 6.2.1.2 Device Selection - ARM Devices

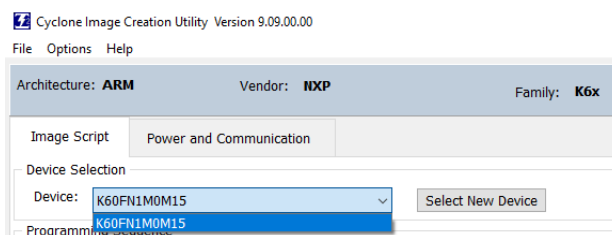
The “Select New Device” button launches the Device Selection window, shown below. This dialog directs the user how to navigate the device tree to select their ARM device.



**Figure 6-2: Device Selection Window - Drill Down To Specific Target Device**

#### 6.2.1.2.1 Device Box

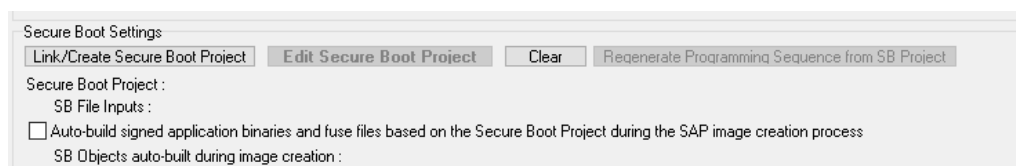
The Device box can be used to choose from any devices that have recently had SAP images saved to Cyclone or disk.



**Figure 6-3: Device Drop-Down - Select From Recently Saved Devices**

#### 6.2.1.3 Secure Boot Settings - NXP i.MX RT and LPC55Sxx Devices

If the selected device is an NXP i.MX RT or LPC55Sxx device with secure boot capability, a Secure Boot Settings area will appear:

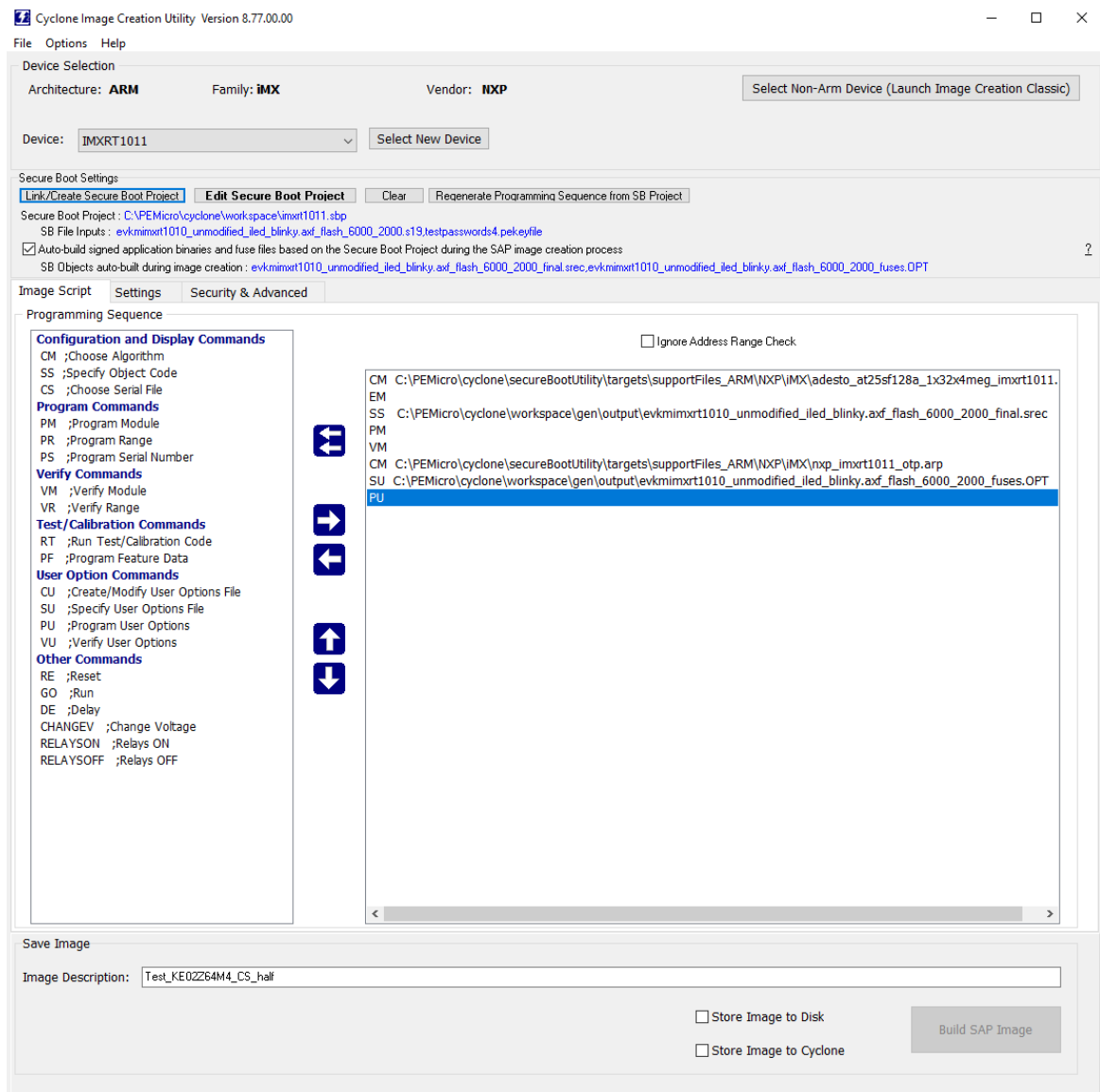


**Figure 6-4: Secure Boot Settings (Older Version of ICU Shown)**

The area allows the user to coordinate with PEmicro's Secure Boot Helper Utility, which is a separate utility that helps the user to more easily manage secure boot settings and configurations. Settings that the user has configured in the Secure Boot Utility can be loaded into the Image Creation Utility as a resulting programming sequence.

In the Secure Boot Settings area, buttons are displayed which allow the user to do the following:

- “Link/Create Secure Boot Project” prompts the user to either choose an existing Secure Boot Project (.SBP) file or to create a new one. The Secure Boot Helper Utility will then launch and load the project.
- “Edit Secure Boot Project” prompts the user to select an .SBP project file. It will then launch the Secure Boot Helper Utility and open that project for editing.
- “Clear” removes the link to any loaded .SBP project. The user will be asked if they wish to clear the programming sequence as well.
- “Regenerate Programming Sequence From SB Settings” allows the user to load re-load the programming sequence from the .SBP project if they have made changes to the project which they wish to be reflected in the programming sequence. Note that any changes made to the programming sequence in the Image Creation Utility itself will be lost when the new sequence is loaded. This typically happens only if the device or binary has been changed in the .SBP.



**Figure 6-5: Project Linked and Programming Sequence Loaded (Older Version of ICU Shown)**

When a project is linked, inputs to the Secure Boot Project such as an S19 file and PEKeyfile will be displayed next to SB File Inputs. The object to be built during image creation is also displayed below.

The user can check the checkbox for “Auto-build signed application and fuse files based on the Secure Boot Project during the SAP image creation process” if they wish that to occur during image creation.

#### Additional Assistance

The Secure Boot Helper Utility User Manual is available at PEmicro’s website:

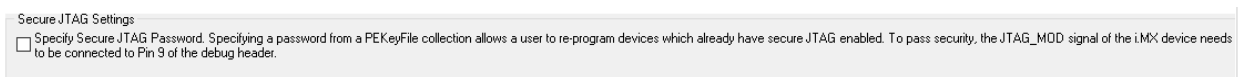
[http://www.pemicro.com/downloads/download\\_file.cfm?download\\_id=582](http://www.pemicro.com/downloads/download_file.cfm?download_id=582)

The website also features a blog article that walks the user through the process of setting up a production programming image with the Secure Boot Utility and the Cyclone Image Creation Utility:

[http://www.pemicro.com/blog/index.cfm?post\\_id=204](http://www.pemicro.com/blog/index.cfm?post_id=204)

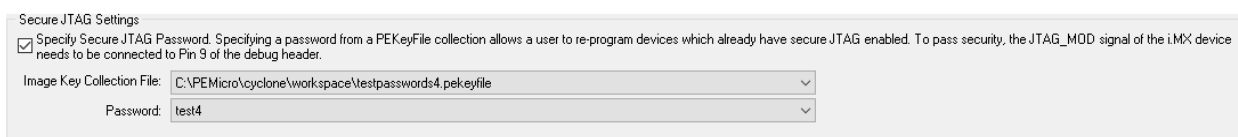
#### 6.2.1.4 Secure JTAG Settings - NXP i.MX RT Only

This setting is displayed on the Security & Advanced Tab. Compatible NXP i.MX RT devices allow the user to enable Secure JTAG. This is a security feature which allows regulation of JTAG access to the device, protected by password. Users can program a device that has Secure Boot enabled by providing the correct password. PEmicro’s Secure Boot Helper Utility helps the user to manage this feature by allowing the user to create named password objects which are stored in a PE Keyfile Collection.



**Figure 6-6: Secure JTAG Settings**

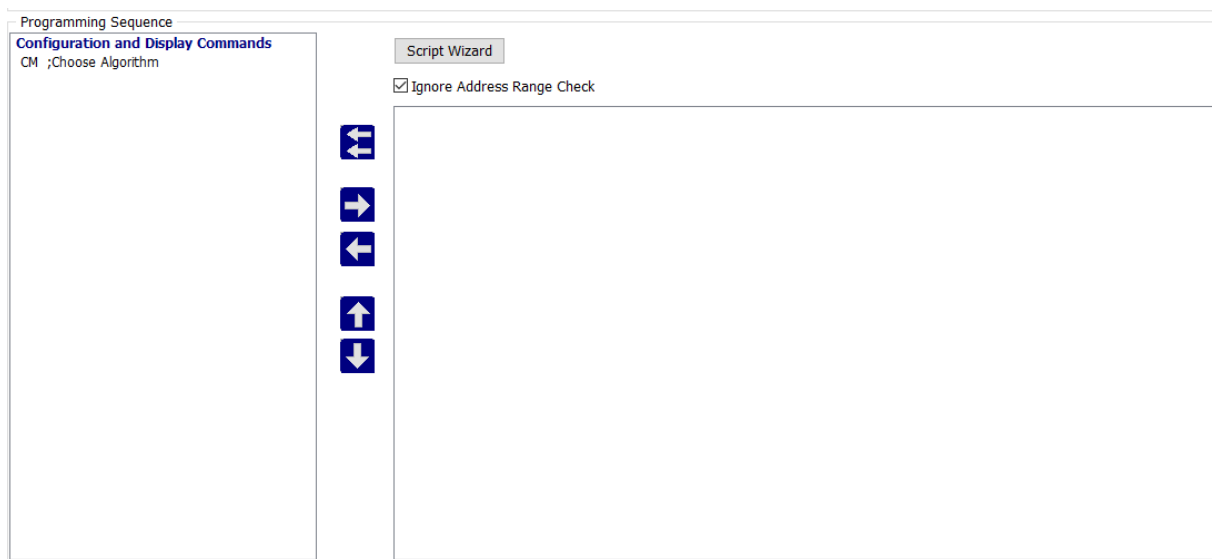
If the user checks the box provided, the Image Creation Utility prompts the user to navigate to a PE Keyfile Collection. On the line below they will then choose the appropriate password object from that collection in order to program a device with Secure JTAG enabled.



**Figure 6-7: Image Key Collection File and Password Object**

#### 6.2.1.5 Programming Sequence

This is the two-panel interface directly below the Device Selection area. This is where the user creates the sequence of commands to be carried out during programming. The left panel provides a list of available programming functions. The right panel displays the ordering of the functions.

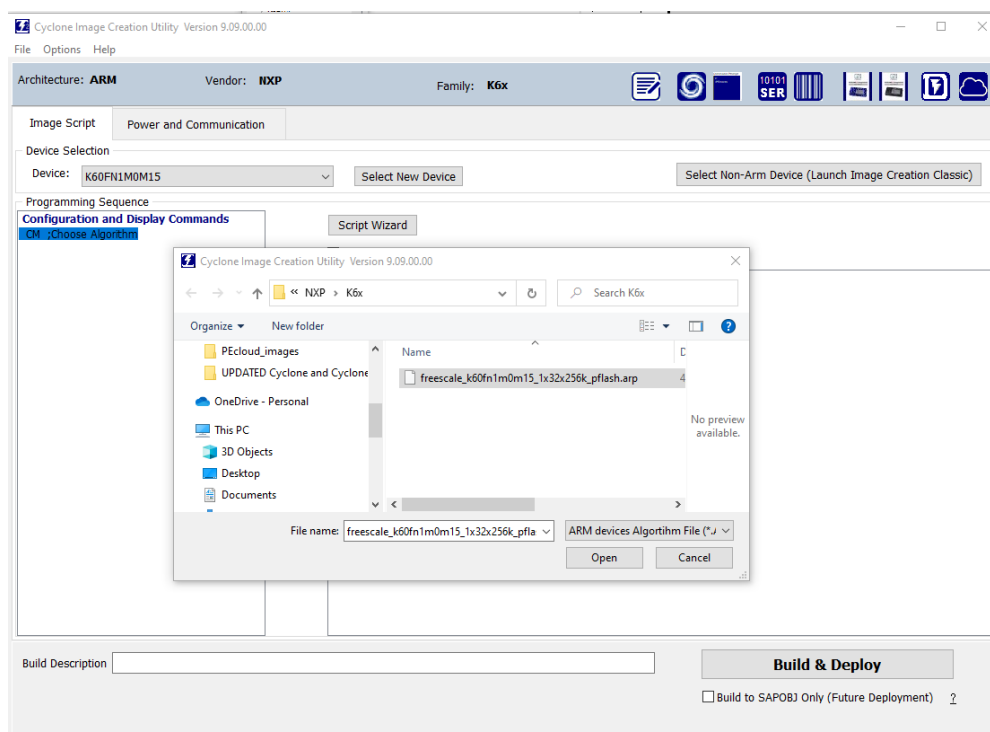


**Figure 6-8: Specify Programming Sequence**

There are two ways to create the programming sequence.

#### 6.2.1.5.1 Manual Selection

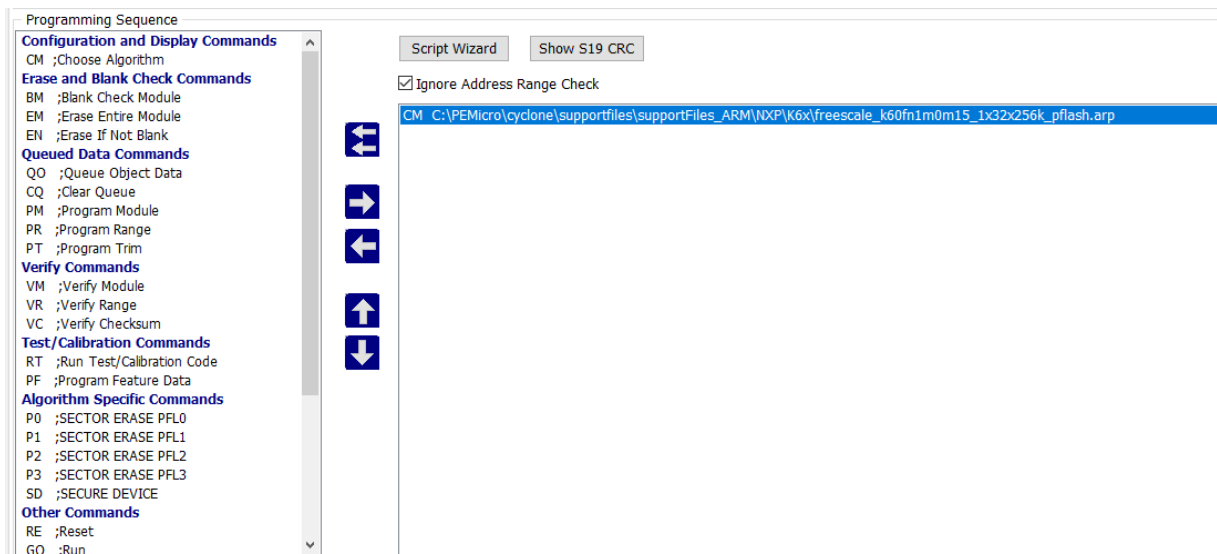
The user can individually specify the algorithm and the object code and then add commands. To specify the programming algorithm for the target, double-click on the Choose Algorithm (CM) function in the left panel. Or, it can be highlighted and added to the right panel using the arrow (->). This opens the “Specify Programming Algorithm to Use” dialog.



**Figure 6-9: Specify Programming Algorithm To Use**

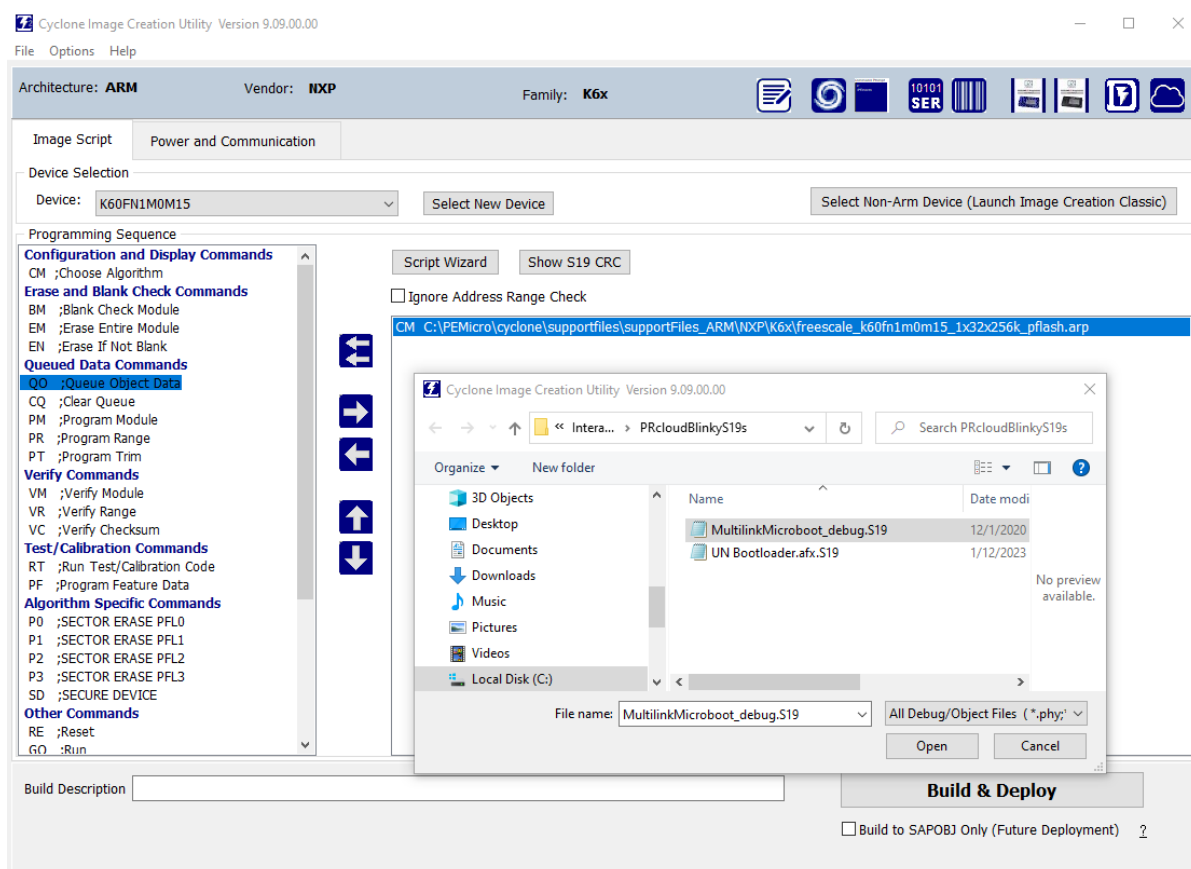
The user should select the programming algorithm to be used. Once the algorithm is selected, the full list of programming functions becomes available in the left panel.





**Figure 6-10: Programming Functions Available**

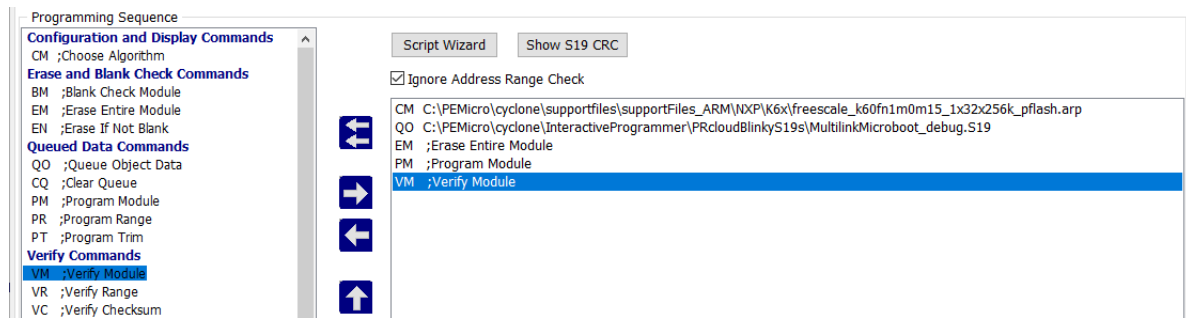
Similarly, to specify the S-Record to be programmed into the target, the user may double-click on Queue Object Data (QO) in the left panel or highlight it and add it using the arrow (->). This opens a dialog which allows you to select the appropriate S-Record.



**Figure 6-11: Specify Object File To Load**

Next, the user would add additional programming functions to complete the programming script by selecting programming operation commands from the Programming Sequence area. See **Section 6.2.1.6 - Programming Operations** for a description of these commands. The commands can be added by double-clicking them, or by selecting them and using the arrow (->). Commands can also

be removed or resequenced; see **Section 6.2.1.5.3 - Function Buttons**.

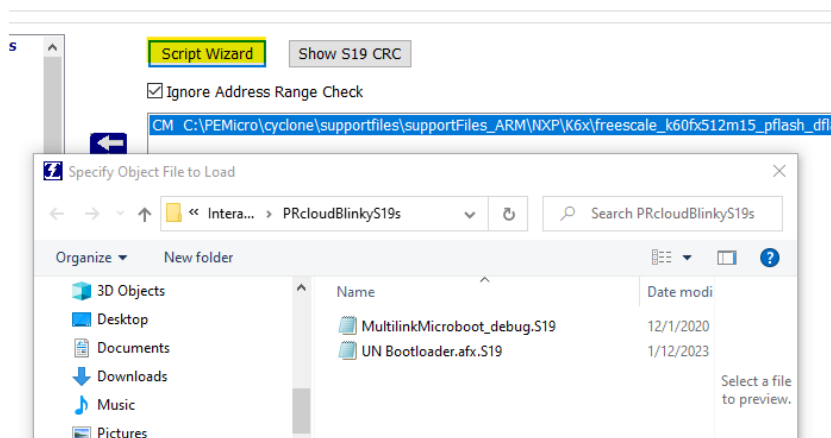


**Figure 6-12: Add Programming Functions**

#### 6.2.1.5.2 Script Wizard

Another method that can be used to create a programming sequence is the Launch Script Wizard button.

**Note:** When NXP's i.MX RT devices with Secure Boot are selected, the Script Wizard Button will not appear.



**Figure 6-13: Script Wizard Button**

**Note:** Launch Script Wizard removes any commands that are already in the programming sequence window and begins a new sequence.

The Launch Script Wizard button will automatically prompt the user for a programming algorithm, followed by an object file, and then adds commands (EN - Erase if not blank, PM - Program module, VC - Verify Checksum) to create a default programming script. The user can then use the programming commands on the left and function buttons to modify the programming sequence as needed.

#### 6.2.1.5.3 Function Buttons

The Clear Script button will remove all programming commands from the right panel. If one of these is the CM command, then it will also remove commands associated with the selected algorithm from the left panel.

The Add Selected Command button will add the selected programming command to the end of the programming sequence. Double-clicking the command has the same effect.

The Remove Selected Command button will be used to remove a selected command from the right panel. The user can also simply hit the Delete button on their keyboard when the command is selected in the programming sequence.

The Move Up and Move Down buttons will allow the user to move the selected programming command up or down within the sequence.

#### 6.2.1.5.4 Programming Sequence Complete

Once the programming sequence is complete, the programming image can be saved as a SAPOBJ archive file and/or the user can continue to the Build & Deploy phase to deploy an Image directly to a Cyclone or disk, or a Job to PEcloud online.

#### 6.2.1.6 Programming Operations

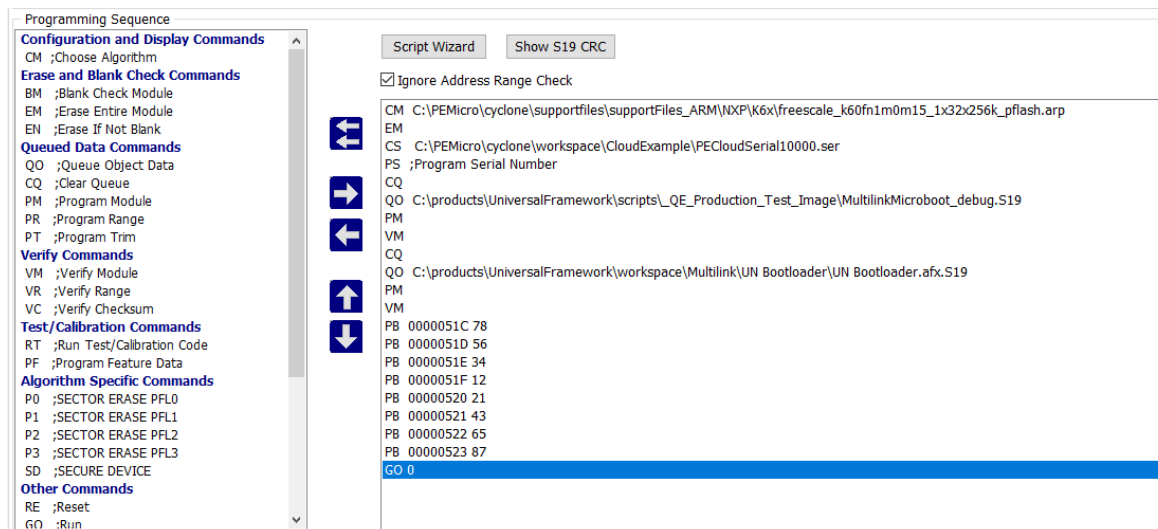


Figure 6-14: Programming Operations Dialog Section

In the Programming Sequence field, the user may specify the algorithm, object file, and operations to be carried out. Here are some common operations.

##### 6.2.1.6.1 Queue Object Data

Presents a list of available programming files. Each programming file contains information on how to program a particular module. Usually, the name of the file indicates what kind of module it relates to.

##### 6.2.1.6.2 Specify Object File

Asks for the name (and/or path) to a file of object files to be used in programming or verifying a module. If the file is not found, an error message is given. The currently-selected file is shown in the S19 file selected window. The programmer accepts S1, S2, and S3 records. All other file records are treated as comments. If you do not specify a file-name extension .S19 is used by default. The programmer also supports ELF/Dwarf 2.0, 3.0, and 4.0 object files.

Your .S19 file may contain data for both EEPROM and flash. If you know that your S19 file contains the correct data, "Ignore S19 Range" may be checked. This will cause any out of range errors to be ignored.

##### 6.2.1.6.3 Erase Address Range.

Note: **For ARM and Infineon Tricore devices.** This command uses sector erase to erase all sectors between the start and end address parameters.

##### 6.2.1.6.4 Erase If Not Blank

This command performs a blank check of the module and erases it if it is not blank.

Note: **For ARM and Infineon Tricore devices** this command appears as **Erase All Non-blank Sectors**. This command uses sector erase to erase all non-blank sectors of flash memory. If the sector is already blank, the erase is skipped for that sector. This can offer better performance compared to Erase Entire Module for very large flash memories.

#### 6.2.1.6.5 Erase Module

If “Erase Module” is specified, the Cyclone will erase the EEPROM/flash on the target device after entering the Monitor Mode or BDM mode.

Note: **For ARM and Infineon Tricore devices** this command appears as **Erase Entire Module**. This command uses mass or chip erase if available to erase the entire flash memory. If the flash memory is not erasable, an error message is returned. This can be slow for very large flash memories.

#### 6.2.1.6.6 Erase Object File Ranges

Note: **For ARM and Infineon Tricore devices**. This command uses sector erase to erase all sectors that are found in the user's selected object file.

#### 6.2.1.6.7 Blank Check Module

If “Blank Check Module” is checked, the Cyclone will check to see if the flash/EEPROM on the target device is erased.

#### 6.2.1.6.8 Start Code Running (GO)

If supported, this will appear in the Choose Programming Function window. For ARM devices, after the “GO” command is selected, the user will be prompted to select between a hard or soft reset. A hardware reset will run the user's application code (which is programmed into flash memory) only after a physical reset of the MCU (power cycle of MCU, power cycle of PE micro hardware, reset of PE micro hardware, etc.). A software reset will allow PE micro tools to perform a reset of the MCU without the need of any physical interaction, and the application code will be able to run directly after the programming sequence.

#### 6.2.1.6.9 Program Bytes

Prompts for a starting address, which must be in the module. You are then asked to enter in hexadecimal a byte to be programmed into the current location. Clicking the OK button will automatically advance to the next data byte location.

#### 6.2.1.6.10 Program Words

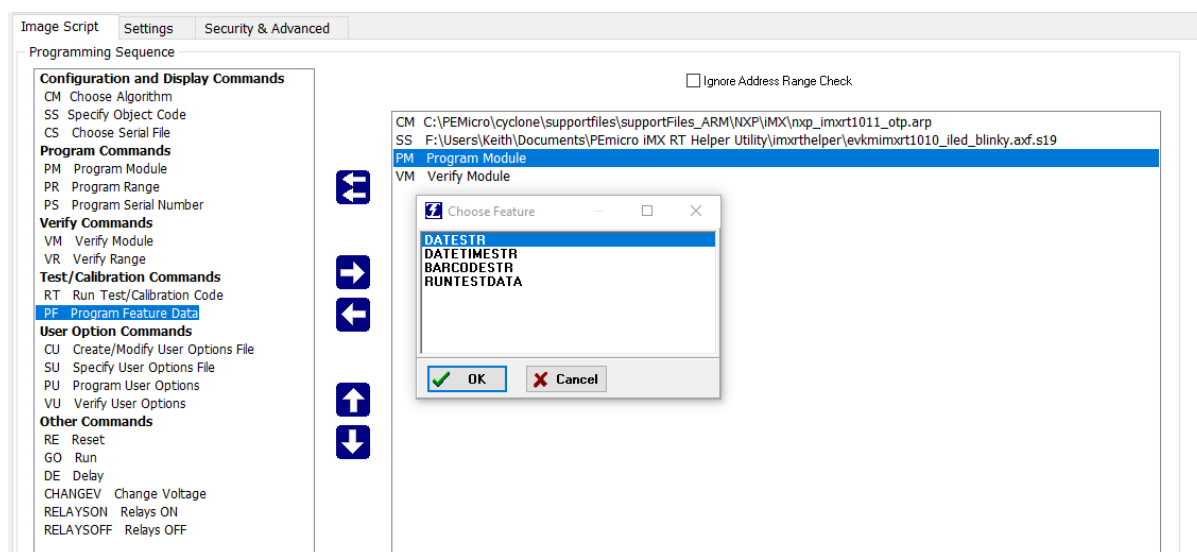
Prompts for a starting address, which must be in the module. You are then asked to enter, in hexadecimal, a word to be programmed into the current location. Clicking the OK button will automatically advance to the next data word location.

#### 6.2.1.6.11 Program Module

This command will program the selected S-record file into EEPROM/flash. For this command to work, you must have previously selected an S-record file.

#### 6.2.1.6.12 Program Feature Data

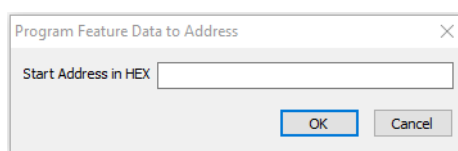
The Program Feature Data option on the Cyclone Image Creation Utility gives the user more options to program dynamic data on the target device. To use Program Feature Data select the “PF” command when creating a programming image. A window will show you the options for feature data to program.



**Figure 6-15: Using PF Command (Dynamic Data) (Older ICU Shown)**

The options can be 1) a string of the current date (YYYY-MM-DD), 2) a string of the current date and time, 24-hour clock (YYYY-MM-DD HH:MM:SS), 3) Run Test data (see **CHAPTER 15 - RUN TEST - CUSTOM TEST APPLICATIONS IN THE PROGRAMMING SEQUENCE**).

To 4) program the barcode into the flash of the target device, BARCODESTR should be selected. The next window contains the hex address of where the dynamic data will be stored:



**Figure 6-16: Program Feature Address Dialog (Hex)**

#### 6.2.1.6.13 Verify Module

This command will verify that the selected S-record file was programmed into the EEPROM/flash. For this command to work, you must have previously selected an S-record file.

#### 6.2.1.6.14 Verify Checksum

This command verifies the module content via a CRC calculation. This command is typically much faster than performing a full Verify Module command.

#### 6.2.1.6.15 Choose Serial File

This command becomes available once a programming algorithm is selected. It specifies the serial file that holds the serial numbers to be programmed to the target.

#### 6.2.1.6.16 Program Serial Number

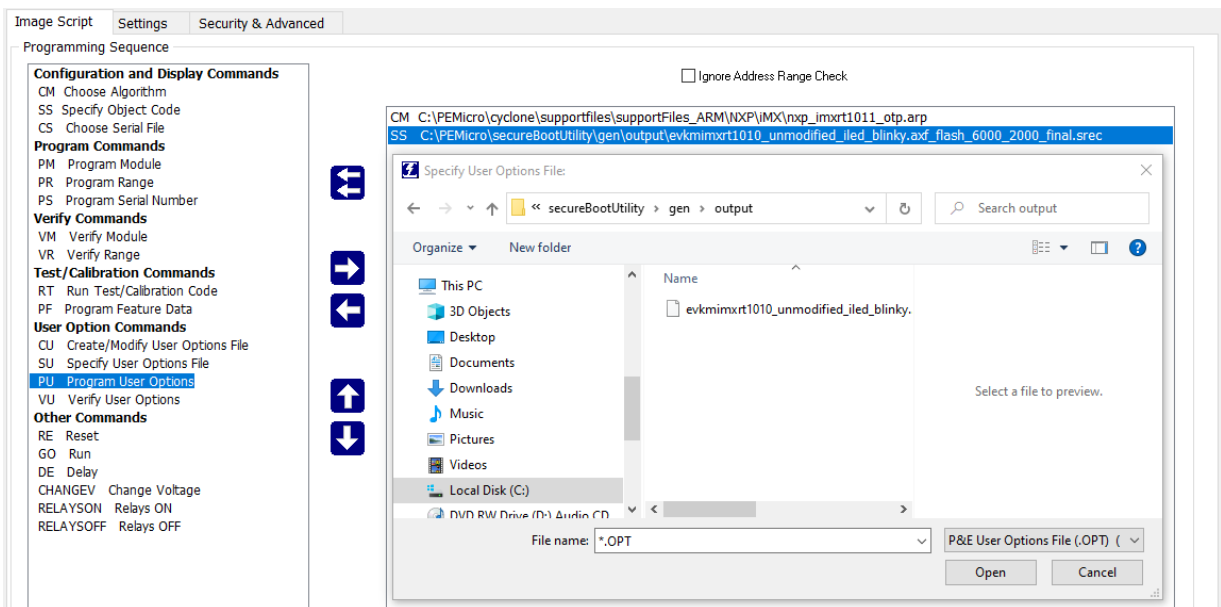
This command becomes available once a programming algorithm is selected. It will instruct the Cyclone to program the serial number to the target once executed. As with other commands, the serial number will not be programmed until the SAP operations are carried out.

When using a barcode scanner as part of the programming process, a Barcode Test file must be included with the programming script of the SAP image. The "Use Barcode File" selection is enabled, and the exact file specified, in the FX Exclusive Settings section of this window. Please refer to **Section 6.5.7.1 - Use Barcode File**. For more information on using a barcode scanner with the Cyclone FX please see **CHAPTER 13 - USING A BARCODE SCANNER TO SELECT AN IMAGE & INITIATE PROGRAMMING**.



### 6.2.1.6.17 User Options Commands

Some ARM devices have areas of flash memory dedicated to programming user configuration data. As some writes to such areas can be sensitive or permanent, it is important that the developer is able to write these options correctly the first time and avoid mis-programming adjacent options that they wish to leave untouched. [http://www.pemicro.com/blog/index.cfm?post\\_id=177](http://www.pemicro.com/blog/index.cfm?post_id=177)

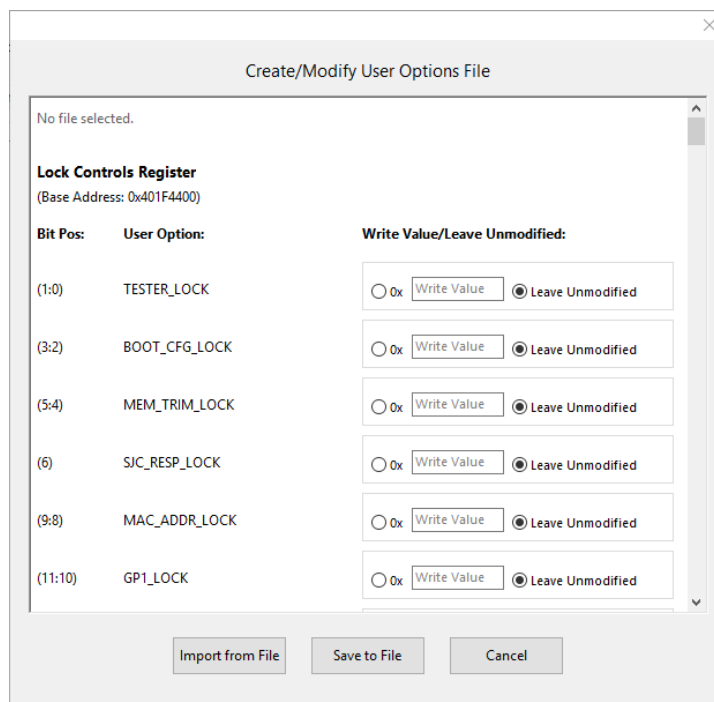


**Figure 6-17: Program User Options (PU Command) - Select User Options File (Older ICU Shown)**

#### Create/Modify User Options File

When the Create/Modify User Options File (CU) command is selected a new window will open, in which the names of existing user area(s) and included options are displayed. For each user option, the developer will have the ability to either "Write [a new] Value" or "Leave [the option] Unmodified." New values must be written in hexadecimal format.

The user may also import and modify an existing User Options file by clicking the Import from File button at the bottom of the dialog.



**Figure 6-18: Create User Options File Window**

Once all values have been written in their appropriate fields, the developer can save the user options (.OPT) file by clicking Save to File. Once a user options (.OPT) file has been created/saved, it can be used to program the device's user options by using the Specify User Options File (SU) and Program User Options (PU) commands.

### **Specify User Options File**

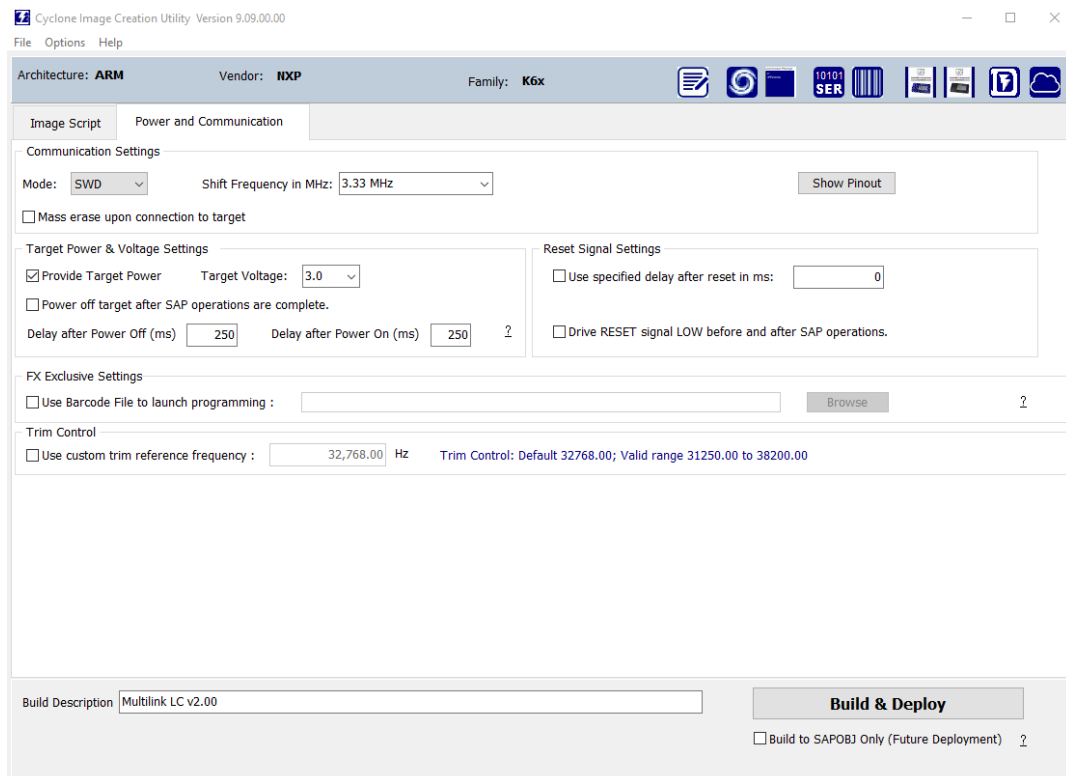
With the Specify User Options File (SU) command, the developer can select an existing user options (.OPT) file.

### **Program User Options**

Once a file has been specified, the Program User Options (PU) command can be used to write the values specified by the file. For most devices, new option values won't take effect until the device is reset.

## **6.2.2 Power And Communication Tab**

Additional settings related to target power, communications, signal settings, trim, and others are available on this tab.



**Figure 6-19: Power And Communications Tab**

#### 6.2.2.1 Communication Settings

These are located in the Settings tab. **Cyclone FX** programmers support multiple communication modes and communication rates. A user needs to select proper communication mode and rate from the drop down list after programming operations are specified. The debug connector pin definitions are listed for reference.

#### 6.2.2.2 Target Power and Voltage Settings

These are located in the Settings tab. A user may elect to use Cyclone to supply power to the target. In this case, the Target Voltage specifies the target MCU I/O voltage level.

The user needs to take into account the power discharge time for the Power Down delay. The reset driver delays, power stabilization time, and the target clock stabilization time should be considered for the Power Up delay.

A checkbox is available for a user to instruct the Cyclone to turn off target power after SAP operations. If unchecked, the target power will remain on.

The user has the option to provide Reset Delay if certain reset monitoring devices are used. The Cyclone will delay for the specified time after allowing the target out of reset.

#### 6.2.2.3 Reset Signal Settings

Allows the user to specify delay after reset, and choose to drive the RESET signal low before and after programming operations.

#### 6.2.2.4 FX Exclusive Settings

**Cyclone FX** users can launch programming via a bar code scanner. This area is where they must specify if they would like to do so for this programming image. A corresponding Barcode File must also be selected. See **CHAPTER 13 - USING A BARCODE SCANNER TO SELECT AN IMAGE & INITIATE PROGRAMMING** for more details.

#### 6.2.2.5 Trim Control

Allows the user to specify a custom trim reference frequency.

#### 6.2.2.6 Build Description

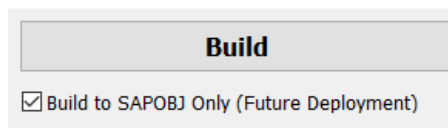
The Cyclone Image Creation Utility allows the user to name their build in the Build Description box. The build description will be used to populate the Image/Job Description box if the user chooses Build & Deploy.



**Figure 6-20: Build Description Box**

### 6.3 Create a SAPOBJ

Instead of Deploying to an Image or Job, the user has an option to check the “Build to SAPOBJ Only (future Deployment)” checkbox before clicking Build. This will create a SAPOBJ from the build rather than immediately launching the Image and Job Deployment Manager form.



**Figure 6-21: Create SAPOBJ**

Nothing prevents the user from simply continuing on to deployment after saving a SAPOBJ for archival purposes.

### 6.4 Build And Deploy an Image or Job

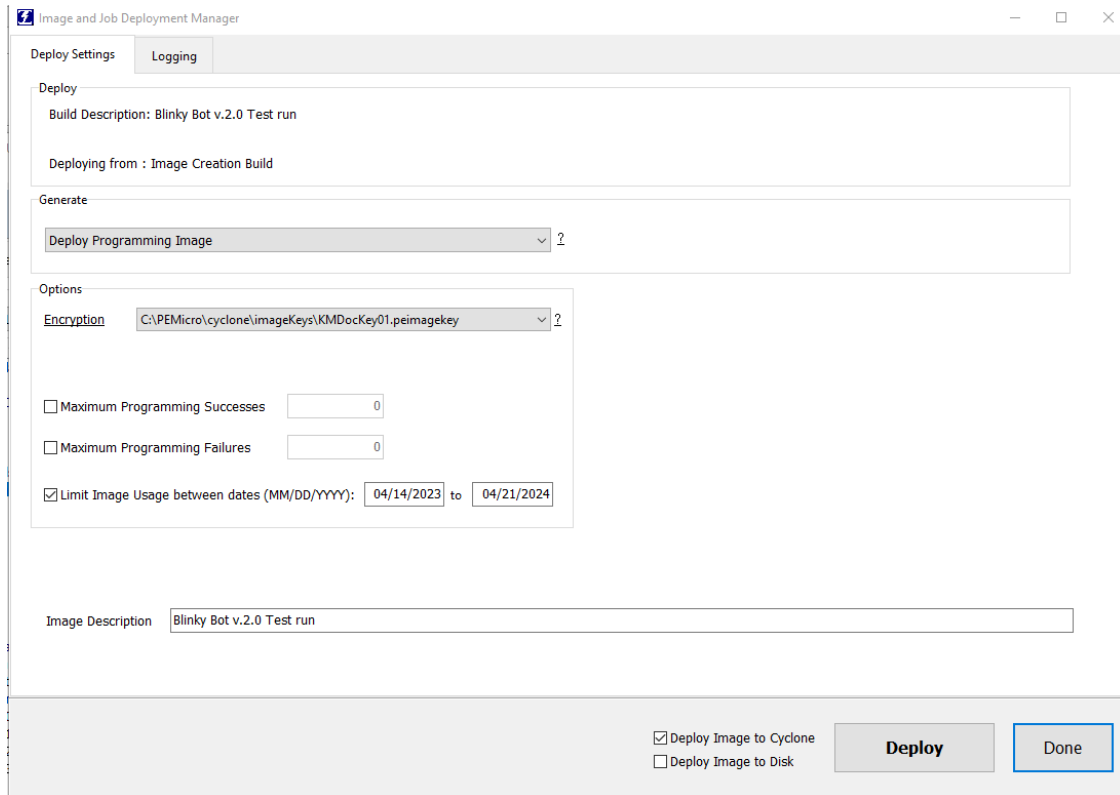
This launches the Image and Job Deployment Manager form where the user may configure additional settings, some optional and some (specifically for PEcloud Jobs) required, before deployment.



**Figure 6-22: Build & Deploy an Image or Job**

### 6.5 Image and Job Deployment Manager Form

Here is the Image and Job Deployment form.

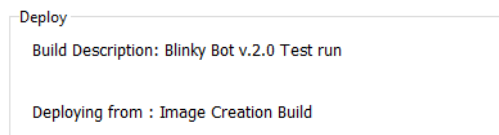


**Figure 6-23: Image and Job Deployment Manager Form (Deploy Image Selected)**

### 6.5.1

#### Deploy Area

The Deploy area displays the name of the Build that is being deployed with this form, as well as the origin of the build.

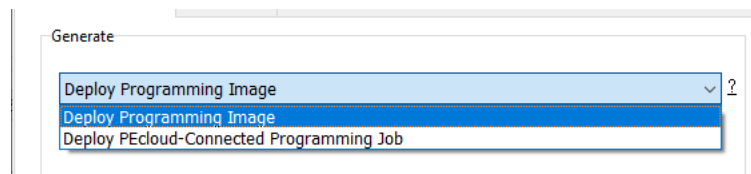


**Figure 6-24: Deploy Area**

### 6.5.2

#### Deploy Settings Tab

The Generate drop-down allows the user to choose whether to deploy a Programming Image or a PEcloud-connected Job.



**Figure 6-25: “Generate” Drop-Down**

When the user makes this selection they may have additional configurations available. The programming restrictions differ slightly in format, and Cloud deployment will require a few additional settings as compared to Image deployment. But they are similar enough that they can be covered together.



### 6.5.3 Options Area

The settings in the Options area are optional for Image deployment, however cloud Jobs require encryption.

**Note:** ProCryption Security is provided automatically for any Cyclones making use of the PEcloud service, due to its robust security standards.

#### 6.5.3.1 Encryption (ProCryption Security)

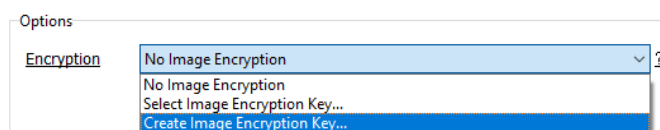
Users can create unique ImageKeys which can be used to encrypt their programming images. SAP images encrypted in this way can only be loaded onto a Cyclone that has been provisioned with the identical ImageKey.

**Note:** Both the Cyclone and the ImageKey are needed to decrypt the image. If the ImageKey is later removed from the Cyclone, the encrypted SAP image cannot be decrypted for programming. Likewise, the encrypted image cannot be read on a PC.

This section will detail how to use the Cyclone Image Creation Utility to encrypt a SAP image. For a more in-depth description of Cyclone SAP image encryption, please see **CHAPTER 12 - SAP IMAGE ENCRYPTION**. For detailed information about how to use and manage encrypted programming images during the production process, the user should refer specifically to **Section 12.4 - Managing Encryption For Production Programming**.

##### 6.5.3.1.1 Creating an ImageKey

If an ImageKey has not been created or a new ImageKey is required, the user should select the "Image Encryption" combo-box in the Cyclone Image Creation Utility and choose the "Create Image Encryption Key..." option.



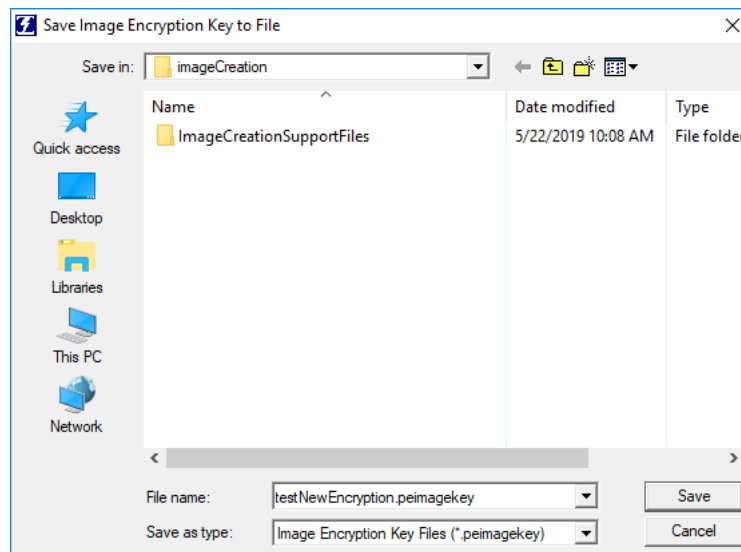
**Figure 6-26: Create Image Encryption Key - Drop Box**

This will pop up a box asking for a descriptive ImageKey Name (this name will be used for display in many dialogs):



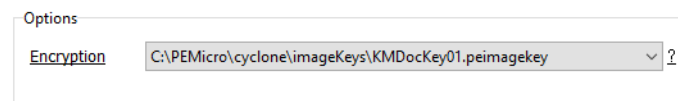
**Figure 6-27: Create ImageKey**

After entering the name, the user should click Generate Encryption Key. This will bring up a dialog which allows the user to choose the save location:



**Figure 6-28: Save Image Encryption Key To File Dialog**

The user should navigate to the desired location and then click "Save". The ImageKey will be generated and automatically selected in the Cyclone Image Creation Utility, such that generating an image will use this ImageKey for encrypting.



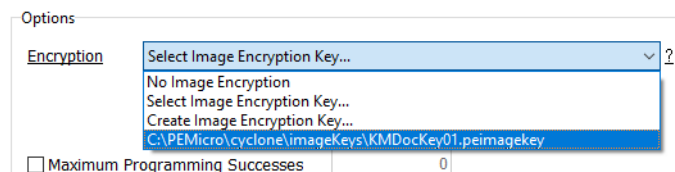
**Figure 6-29: ImageKey Selected After Creation**

**Note:** Every ImageKey created is unique and may not be recreated. This means that once generated, the user should keep the ImageKey in a secure place. Users may also wish to keep track of which SAP images have been encrypted with each ImageKey, as the current software does not track this information.

By default, the ImageKey will stay selected in the Image Creation Utility. If a different ImageKey is required for encryption, or the user does not wish to encrypt their SAP image, the corresponding change may easily be selected using the drop-down box.

#### 6.5.3.1.2 Encrypting An Image/Job

To create an encrypted programming image, the user sets up their parameters in the Cyclone Image Creation Utility as usual, and then simply selects the desired ImageKey in the "Image Encryption" combo-box.



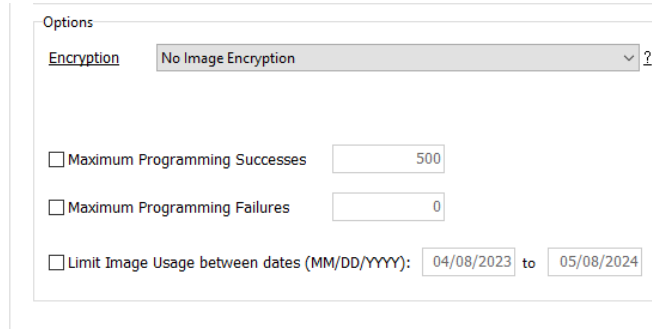
**Figure 6-30: ImageKey Selection**

The image will automatically be generated as an encrypted image, encrypted with the selected ImageKey. An encrypted stand-alone programming image is called an eSAP (Encrypted Stand Alone Programming) file. This eSAP file may be downloaded to any Cyclone which has been provisioned with the same ImageKey (i.e. the ImageKey has already been added to the Cyclone). This is discussed in **Section 12.4.1 - Provisioning a Cyclone with an ImageKey**.

### 6.5.3.1.3 Programming Restrictions

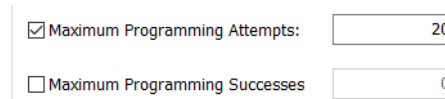
There are any number of reasons why the user may want to place restrictions on the use of specific programming images/jobs on a Cyclone programmer: from added ease when managing production to a desire to protect intellectual property.

These settings are located beneath the Encryption box in the Options section of the Deploy Settings tab. **Cyclone FX** programmers, and **Cyclone LC** programmers with ProCryption Security activated, can take advantage of these powerful security features.



**Figure 6-31: Encryption and Image Restrictions (Image Deployment Selected)**

Eagle-eyed users may note that part of the programming restrictions for Image deployment differs just slightly vs. those for Job deployment. Max Successes/Max Failures vs. Max Attempts/Max Successes.



**Figure 6-32: Image Restrictions (Cloud Deployment Selected)**

Users can fill in these settings to limit programming of the Image/Job based on the attributes listed, including date range.

Even if restricted programming Images/Jobs are deleted from Cyclone's internal memory or an SD card, the Cyclone platform has a persistent memory that continues to tie security restrictions to that programming Image/Job. Thus, if it is removed and re-added to a Cyclone, the associated counts are maintained and would continue counting from where it left off. Also, if the SD Card is moved from Cyclone to Cyclone, the count is maintained in both Cyclones as well as the SD Card.

Every time an image/job is generated by the Cyclone Image Creation utility, it is encoded with a unique image ID number. All counts are stored relative to this unique ID number. So, when an image/job is regenerated in the Cyclone Image Creation utility, it will have its own counts which will not conflict with the previously generated image, even if they are otherwise exactly the same. In this way, the user can regenerate to allow a new batch of targets to be programmed.

**Note:** The user may set more than one type of restriction on programming. The ability to program the image will be restricted by whichever triggers first. E.g., if the user creates settings to allow 100 programs, and also sets an allowed date range restriction, the ability to program the image will be restricted as soon as the first of these conditions is triggered.

Currently the user may set the following restrictions:

### 6.5.3.1.4 Limit Image Usage Between Dates

When "Limit Usage Between Dates" is checked and the start and end dates are specified with valid dates (format: DD/MM/YYYY), the Cyclone operator will only be allowed to program the corresponding programming image when the date is on or between the dates specified. The Cyclone has an onboard battery and clock which keeps a clock running even when power to the Cyclone is removed. This clock date is the one used for comparison to the UTD Date specified in the image. The ability to limit programming to a date is useful for making sure that an image will stop working after a period of time. This could be for security purposes, or to make sure that a new

and updated image will need to be uploaded to the Cyclone after a period of time (for instance, to not allow a firmware more than a year old to be programmed onto a target).

**Note:** For any date restrictions applied to a Job, the time for the Start date is considered to be 00:00:00 UTC of the Start date, and the time for the End date is considered to be 23:59:59 UTC of the End date.

#### 6.5.3.1.5 Maximum Number of Successes Allowed

When “Maximum Number of Successes Allowed” is checked and a number is specified in the corresponding box (minimum = 1), the Cyclone operator will only be able to execute a number of successful programming operations of this programming image less than or equal to the number specified. The current programming count can be displayed on the main screen of the Cyclone or it can be seen on the image’s statistics page (see **Section 6.5.3.1.8 - Image Restriction Statistics**).

#### 6.5.3.1.6 Maximum Number of Failures Allowed (Images Only)

When “Maximum Number of Failures Allowed” is checked and a number is specified in the corresponding box (minimum = 1), the Cyclone operator will only be able to execute programming operations on the current image until the maximum number of errors specified has been reached. This restriction exists largely to prevent an operator from intentionally generating an error as part of the programming process in an attempt to circumvent the count restrictions. A recommended limit on this number would be on the order of 5% of the allowed programming counts.

#### 6.5.3.1.7 Maximum Programming Attempts (PEcloud Job Only)

When “Maximum Programming Attempts” is checked and a number is specified in the corresponding box (minimum = 1), the Cyclone operator will only be able to execute a number of either successful or failed programming operations of this programming image less than or equal to the number specified.

#### 6.5.3.1.8 Image Restriction Statistics

Statistics related to any specified restrictions for the currently selected programming Image may be viewed by navigating in the touchscreen menu to *Current Image Operations - Show Current Image Stats*. For more information on viewing programming image stats, see **Section 5.2.3.4 - Show Image Restriction Stats (Requires FX or ProCryption Security Activation)**.

In addition, the statistics for Number of Programs & Maximum Allowed can be set to display on the home screen by navigating in the touchscreen menu to *Configure Cyclone Settings -> Configure Screen -> Configure Home Screen*. For more information on how to configure the Cyclone’s home screen, see **Section 5.2.4.4.4 - Configure Home Screen**.

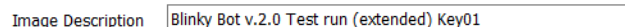
#### 6.5.3.1.9 Image Description

The user must name their image; it can be useful to summarize the purpose of their current configuration for future reference by adding text to the Image Description box. The description will be either programmed into the Cyclone, deployed in a Job to the cloud, or saved into an encrypted file.

By default this box will be populated by the text from the Build Description field on the initial screen of the Image Creation Utility.

The image description will appear on the Cyclone’s touchscreen LCD for image identification. This field will not affect the Cyclone’s operations with the target.

**Tip:** If your image will be encrypted, it can be helpful to indicate this as part of the Image Description.

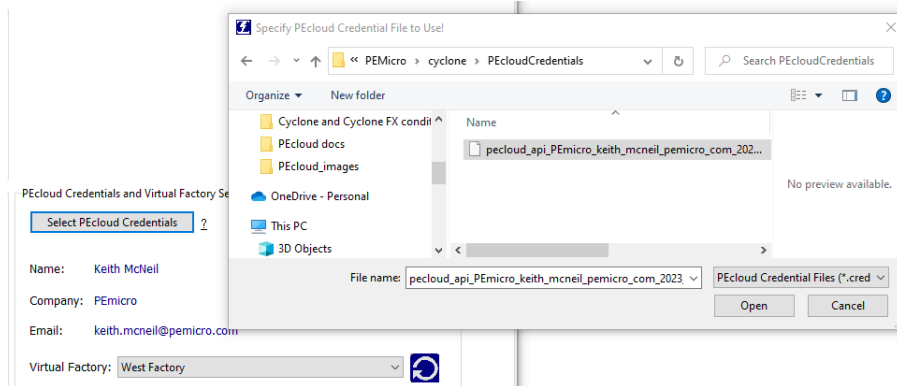


**Figure 6-33: Image Description Box**

#### 6.5.4 PECloud Credentials And Virtual Factory Selection

If the user has selected to Deploy Cloud-Connected Programming Job they will have additional configurations to make before deployment.

1. The user must select their PEcloud credentials if they have not already been selected. These credentials are downloaded from the users PEcloud account and should be saved in a safe place.

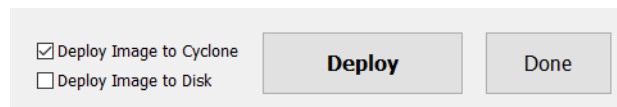


**Figure 6-34: Select PEcloud Credentials**

2. Select the Virtual Factory in the user's PEcloud account to which the Job will be deployed. The button to the right allows the user to manually refresh the list of VFs if changes have been made recently.

#### 6.5.5 Deploy

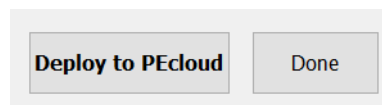
If the user is deploying an Image, they will see the following options at the bottom of the display:



**Figure 6-35: Deploy To Cyclone or Disk**

The user should check the appropriate box to deploy to a Cyclone or to a disk drive before clicking Deploy.

Users deploying a cloud Job will see the following option:



**Figure 6-36: Deploy To Cloud**

Click the "Deploy To Cloud" button to load your Job to PEcloud online. It will appear in the Virtual Factory to which it was deployed.

#### 6.5.6 Renesas ID Code Protection

Renesas RA and Synergy devices offer several means of security, one of which is ID Code protection. After the MCU starts up in boot mode, ID authentication is performed when a host such as a PC is connected, in order to prevent unauthorized access. PEmicro tools allow users to take advantage of ID Code protection within the OCD/Serial Programmer ID Setting Register (OSIS) of supported Renesas ARM processors.

The idea behind ID Code protection is for the user to write four 32-bit words into the OSIS register with ID code bits [127] and [126] indicating the level of security issued. This can be accomplished using algorithm-specific programming commands in the Cyclone Image Creation Utility. PEmicro



also provides a python script that allows users to unlock their device through a successful ID Code match.

For details about the procedures involving algorithm-specific commands and the python script, please visit: [https://www.pemicro.com/blog/index.cfm?post\\_id=228](https://www.pemicro.com/blog/index.cfm?post_id=228)

### 6.5.7 FX Exclusive Features

This area contains a hardware feature that is exclusive to the **Cyclone FX**, and as such cannot be licensed by **Cyclone LC** programmers.

#### 6.5.7.1 Use Barcode File

PEmicro's CYCLONE **FX** programmers can be configured to use a bar code scanner (connected to the extension port) as part of the programming process. During setup the user will create a Barcode Test file using the provided Barcode Test Generator utility.

Here, the user should then check the Use Barcode File checkbox if they wish to include a Barcode Test file in the programming script of their SAP image.

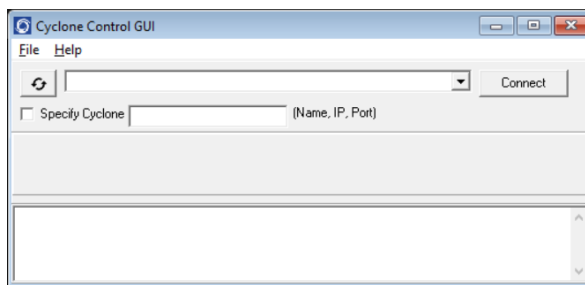


**Figure 6-37: Use Barcode File**

The user can then Browse to the file they wish to select. This is required when using the bar code scanner as part of the programming process. For more information on how to incorporate a barcode scanner into the Cyclone's programming process, please see **CHAPTER 13 - USING A BARCODE SCANNER TO SELECT AN IMAGE & INITIATE PROGRAMMING**.

### 6.5.8 Store Image To Cyclone

"Store Image to Cyclone" allows the current configuration to be programmed into the Cyclone. The Cyclone will then be ready for operations. After you click "Store Image To Cyclone," the Cyclone Control GUI will pop up so that you can choose the Cyclone onto which you wish to save the SAP image.



**Figure 6-38: Cyclone Control GUI: Choose Cyclone for Stored Image**

The Cyclone Control GUI drop-down list allows the user to select from all the Cyclones available. In the case of a Cyclone present on a different network (i.e., not displayed automatically in the drop-down list), the user may specify its IP address by using the Specify Cyclone checkbox and typing the identifier of the Cyclone.

Click on "Connect" to access the specified Cyclone. A click on the "Apply Changes" button will then store the image on the selected Cyclone.

### 6.5.9 Store Image To Disk

"Store Image To Disk" allows the current configuration to be saved onto the hard drive. The image can then be transferred to the Cyclone's internal flash (or an installed SD card) via the Manage Images Utility.

### 6.5.10 Save Cyclone Configuration

“Save Cyclone Configuration,” in the file menu, allows the user to save the configuration into a file, which may be used for future reference, e.g., comparing the Cyclone contents with the file to see if they are the same.

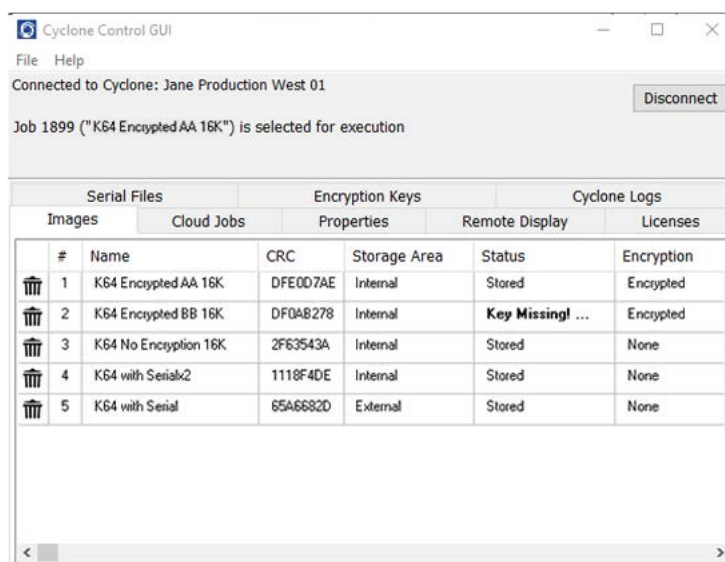
### 6.5.11 Load Cyclone Configuration

“Load Cyclone Configuration” in the file menu allows the user to load a configuration that has previously been saved in order to create a new image.

## 6.6 Managing Multiple SAP Images

The Cyclone Control GUI, shown below in **Figure 6-39**, allows the Cyclone to store and manage multiple images in the Cyclone’s internal memory and, for the **Cyclone FX** only (or legacy Cyclone LC with license), on any compatible memory card that is loaded into the SDHC port.

Any programming images that have been created and saved to the disk using the Cyclone Image Creation Utility may be loaded collectively onto the Cyclone, with one exception. Encrypted images may only be loaded if their ImageKey resides on the Cyclone.



**Figure 6-39: Manage Images Utility - Image #2 Missing Key**

Upon opening a selected Cyclone in the Cyclone Control GUI, the user is provided in the first tab with a list of the images currently on the unit’s internal memory which are marked with a **Storage Area** label of "Internal". A list of images on any installed SDHC card will also be displayed with a **Storage Area** label of "External"

You can add images with the "Add Image Internal" button under the images panel. These images will appear with a "Status" label of "Ready to Store". These images are not yet in the Cyclone, the "Apply Changes" button will have to be clicked for the changes to take effect.

If the Cyclone’s SDHC port contains a formatted memory card, the user can also add images to the external memory by using the drop-down next to the "Add Image Internal" button and selecting "External". Alternatively, once an image has been added to the proposed changes and it is in the "Ready to Store" state, the user may right click on the image and click the "Switch storage to External" option.

In **Figure 6-39**, note that the **Encryption** area displays “Encrypted” or “None” to indicate whether or not each programming image is has been encrypted. The **Status** area will display “Key Missing!” if the ImageKey for an encrypted image has been removed from the Cyclone.

For information on how to encrypt SAP Images and the significance of the ImageKey, please refer to **Section 6.5.3.1 - Encryption (ProCryption Security)**. For an overview of Cyclone SAP Image encryption and the process of using encrypted images in the production process, please see

### 6.6.1 Delete Images From Internal/External Memory

Any images that are already stored on the Cyclone or installed SD card can be deleted by just clicking on the trash can on the left of the image or by selecting the image and clicking the Delete key on the keyboard, the image status will be changed to "Ready to Erase." The image will be removed after "Apply Changes" is clicked.

### 6.6.2 Add/Remove Images From The Commit Changes Panels

Once the images that you wish to load appear in the images tab, you must press "Apply Changes" to update the Cyclone accordingly. No actual updates will occur to the Cyclone's internal/external memory or installed SD card until the user selects "Apply Changes."

**Note:** Any SAP images that are already stored on older Cyclone models such as the Cyclone PRO, MAX, Renesas, STMicro, or Cyclone LC ARM - Rev. A/B (or on a CompactFlash card in one of those units, if applicable) can not be removed individually and can only be erased by removing all images.

## 6.7 Launching Additional Tools From the ICU

The ICU features a set of buttons in the upper right corner that allows the user to quickly launch, from left to right:

- Blog
- Control GUI
- Control Console
- Serialize Tool
- Bar Code Utility
- User Manuals (Cyclone FX and Cyclone LC)
- Image & Job Deployment Manager
- PEcloud Website.



**Figure 6-40: Buttons For Launching Additional Tools**

## 7 CYCLONE PROGRAMMER MANUAL CONTROL

The **Cyclone FX** must be configured before it can serve as a Stand-Alone Programmer. The user may manually control the Cyclone via the LCD touchscreen menu and/or the Start button, or via PC software. See **CHAPTER 8 - CYCLONE PROGRAMMER AUTOMATED CONTROL** for information on how to use the Cyclone Control Suite to configure, control, and automate Cyclone operations. The target power management schemes remain the same for each control method.

### 7.1 Operation Via Start Button

There is a Start button on the top of the Cyclone which is used for stand-alone programming. It is specified as follows.

<u>Button</u>	<u>Function</u>
START	Start executing the tasks pre-configured into the Cyclone of the currently selected programming image.

#### 7.1.1 LED Indicators

The Cyclone has two (2) LEDs to indicate the current operation stage.

<u>LED</u>	<u>FUNCTION</u>
Error	The Cyclone failed to execute the functions as instructed.
Success	The Cyclone executed the functions successfully.

#### 7.1.2 Procedure via Start Button / LEDs

The following steps must be followed in order for the Cyclone to operate properly after it has been configured:

1. Turn off the target power supply if the "POWER IN" Jack is adopted.
2. Turn off the Cyclone system power.
3. Set the correct Power Management jumper settings.
4. Connect the target power supply to the "POWER IN" Jack, if applicable.
5. Connect the "POWER OUT" Jack to the target board power, if applicable.
6. Connect the ribbon cable to the target board debug connector.
7. Turn on the Cyclone system power.
8. Turn on the target power supply, if applicable.
9. Press the "START" button on the Cyclone.

When the "Success" LED lights up, you have successfully programmed your target.

#### 7.1.3 Example

After the user programs the contents and procedures into the Cyclone's on-board flash, the Cyclone may be used as a Stand-Alone Programmer. Suppose the user wants to perform the following instructions for a target device:

- 1) Erase Module
- 2) Program Module
- 3) Verify Module.

If the Cyclone is providing power to the target board, the "Target Power" icon will illuminate on the LCD display.


The Cyclone will then perform the operations. If they are performed successfully, the "Success" LED will be illuminated. One stand-alone programming cycle will have just been completed.

### 7.2 Operation Via LCD Touchscreen Menu

Once the **Cyclone FX** is configured for stand-alone programming it may be operated by making

selections from the touchscreen LCD menu. This section describes the menu functions that allow the user to easily execute stand-alone programming functions using the touchscreen LCD.



















## 7.3 Home Screen

The home screen appears when the Cyclone is powered on, or when the  Home button is tapped.

### 7.3.1 Icons

A row of icons in the upper right corner indicates the status of various attributes of the Cyclone.

**Note:** The user may tap on the row of icons to view the meaning of each of the currently displayed icons.

<b>Cyclone Unit Status:</b> Ok / Bad		
<b>Programming Status:</b> Ready / Busy		
<b>USB-To-PC Enumerated:</b> Yes / No		
<b>Real-Time clock Enabled &amp; Working:</b> Yes / No		
<b>Cyclone Power Relays:</b> Closed / Open		
<b>Target Device Is Powered*:</b> Yes / No		
<b>SDHC Memory Card:</b> None / Valid / Unformattd / Reset Cyclone**		
<b>PEcloud:</b> Provisioned for PEcloud and Online		
<b>Barcode Scanner:</b> Detected / Not Detected		

\* Target Device Is Powered - "Yes" indicates that the **Cyclone FX** detects power on the Vcc pin of the target device programming header.

\*\* SDHC Memory Card (Cyclone FX only, and legacy Cyclone LC with license) - "Reset Cyclone" indicates that the Cyclone FX needs to be reset before the SDHC card will register as Valid. The user can push the Reset button which is located on the front side of the Cyclone, below the LED indicators.

### 7.3.2 Configurable Display Area

The main area of the home screen can be configured to optionally display the following information, by using the Cyclone IP Configuration Utility (see **Section 5.2.4.4.4 - Configure Home Screen**):

1. Firmware version of the Cyclone (always shown).
2. IP address assigned to the Cyclone.
3. Name assigned to the Cyclone.
4. Number of programming images in the Cyclone's memory.
5. Name of the selected programming image.
6. First serial number associated with the selected image
7. Current status.
8. Results of the last operation performed.
9. Time and date.
10. Status Window and Main Menu button (always shown).
11. Programming count & limit
12. Target voltage and/or current



## 7.4 Status Window

The status window appears in the lower left corner of the home screen and displays the results of programming operations.

### 7.4.1 Error Information Icon

When the Cyclone experiences an error during programming operations, the Info icon will appear to the left of the Menu button (or AUX button, if configured).



Press the Info icon to view a detailed description of the error.

### 7.4.2 AUX Button (Appears If Configured)

The Cyclone allows the user to add an Auxiliary (AUX) button to the home screen which will perform a specific function when pressed. The specific function is chosen by the user when the AUX button is configured. The AUX button will appear on the home screen to the left of the “Menu” button, in the lower right corner of the home screen.

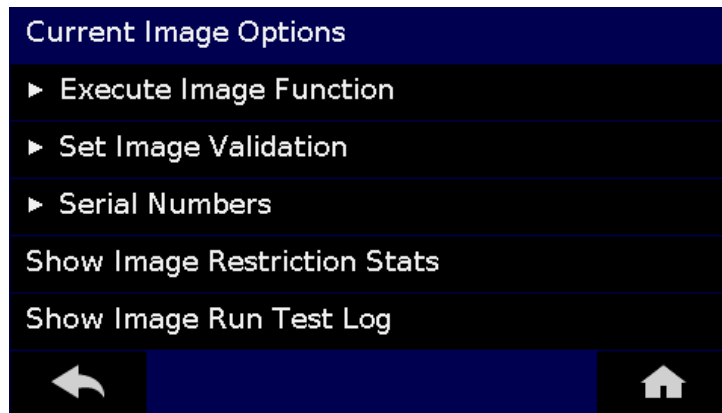


**Figure 7-1: AUX Button On Home Screen (configured for perform CRC32 function)**

For information on how to configure the AUX button, see **Section 5.2.5 - Status**.

### 7.4.3 Main Menu

The Main Menu is accessible by pressing the “Menu” button when the Home Screen is displayed. The Main Menu screen contains five selections. From these, select “Current Image Options.”



**Figure 7-2: Touchscreen LCD Menu - Current Image Options**

The menu selections in “Current Image Options” will allow the user to execute programming operations, verify data, toggle power, validate the programming image, and modify the upcoming serial number if necessary.

**Note:** These apply to Images only and not to any PEcloud Jobs that might be loaded onto the Cyclone, with the exception of **Section 7.4.3.1.1 - Launch Programming**, which will launch programming if a Job is currently selected.

#### 7.4.3.1 Execute Image Function

Execute Specific SAP Function presents four Stand-Alone Programming functions that you may execute by tapping the function that you wish to execute:

#### 7.4.3.1.1 Launch Programming

This allows the user to execute the programming function. The Cyclone will program the target device, if able, using the currently selected programming image. This is functionally equivalent to pressing the Start button.

#### 7.4.3.1.2 Verify Data In Target

Performs a verify function on the data that has been programmed into the target device.

#### 7.4.3.1.3 Toggle Power

Toggles the target power and makes sure all ports are driven to debug mode level.

#### 7.4.3.1.4 Power Cycle Device To Run User Code

Toggles the target power and maintains tri-state mode for all signals.

#### 7.4.3.1.5 Validate Image CRC32

Allows the user to perform a CRC32 validation on the currently selected programming image.

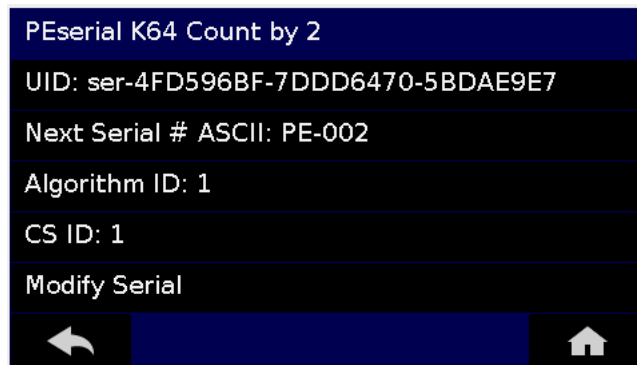
### 7.4.3.2 Set Image Validation

Allows the user to choose between two validation settings: 1) validate the image each time the Start button is pressed, or 2) do not validate the image.

### 7.4.3.3 Serial Numbers

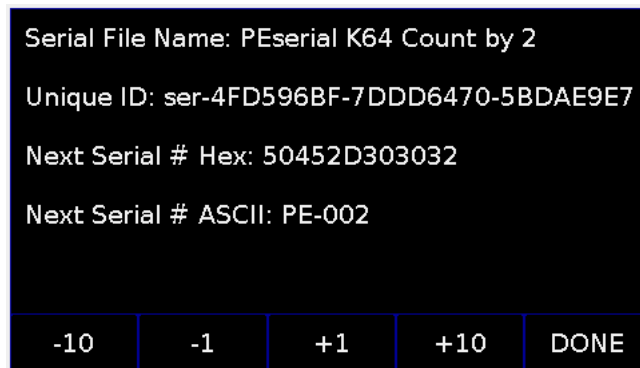
Displays the serial number information associated with the currently selected programming image. If there is none, it will display, "This Image Contains No Serial Numbers." There may be one or more Serial Files associated with the image. The user can click on a specific Serial File name to see the following information about that Serial File:

- UID - Unique identifier of the serial number
- Next Serial # ASCII - Next serial number to be programmed (shown in ASCII format)
- Algorithm ID - Displays the algorithm ID of the serial file
- CS ID - Displays the ID of the CS (Choose Serial) command in the SAP file.



**Figure 7-3: Serial File Selection**

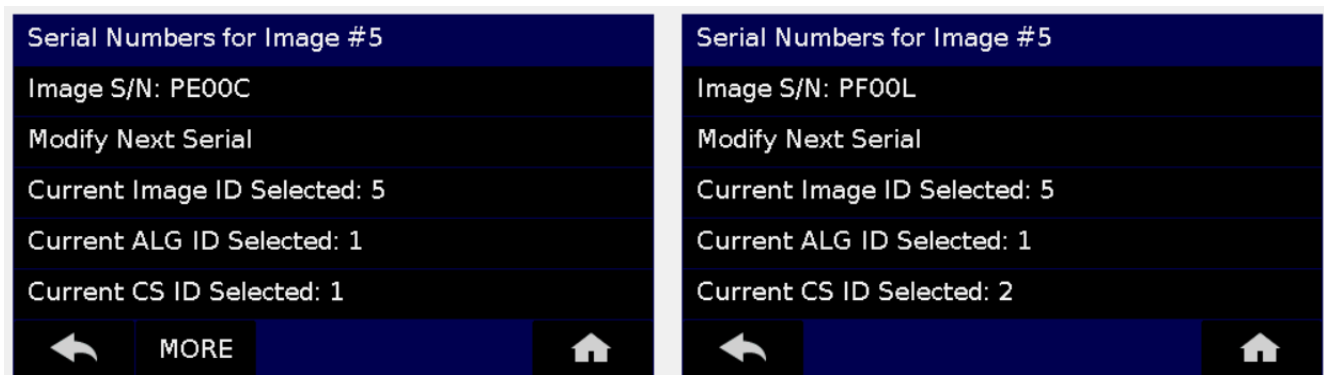
The user can also click on "Modify Serial Number" to edit that serial number. It is possible to Decrease or Increase the Next Serial by -10, -1, +1, +10. This is often done to address issues in the production process, such as during initial setup.



**Figure 7-4: Increase or Decrease Serial Number**

The adjustment buttons will display “Increase Not Allowed” and “Decrease Not Allowed” if the image/algorithm/CS file that the user has selected does not allow for this operation.

**Note:** Unlike PEmicro’s current serial files, earlier “legacy” serial files are specific to one programming image. When viewing Serial Number info for an image that references one or more legacy serial files, the Cyclone will not show links to the serial files but will instead display serial number info for the first serial file. The user can then Modify Next Serial to change the next serial number, or click MORE at the bottom to proceed to the next serial file (if any).



**Figure 7-5: Serial Number View With Legacy Serial Files (CS IDs #1 & #2)**

#### 7.4.3.4 Show Image Restriction Stats (Requires FX or ProCryption Security Activation)

Displays current statistics, if any, for Image Programmed Count & Maximum Allowed, Errors Logged & Maximum Allowed, and Date Range Allowed. These limits can be set in the Security Features section of the Cyclone Image Creation Utility (see **Section 6.1.10.2 - Image Restrictions**).

**Note:** When *Current Image Stats* is displayed as a home screen item, only Image Programmed Count & Maximum Allowed are displayed on the home screen.

## 7.5 PROG Interactive Programming Software

Cyclones include PROG programming software executables for each supported device family. These are located in the path PEmicro -> Cyclone -> InteractiveProgrammer from the install directory. The Cyclone must be used as the hardware interface as PROG software included with the Cyclone has its license permanently installed on the Cyclone.

Users may find that the PROG software is helpful for identifying and correcting issues that are encountered via the other Cyclone programming methods, as the workflow is streamlined and those transitioning from PEmicro’s Multilink probes may find the interface more familiar.

Detailed operating instructions for PROG software are beyond the scope of this manual. A separate user manual for the specific PROG being used can be found on the PROG product page

of [pemicro.com](http://pemicro.com).

For scripted programming, CPROG executables are also included in the same folder.

## 8 CYCLONE PROGRAMMER AUTOMATED CONTROL

Users who wish to control, configure, and automate one or more Cyclone units have several options available via the Cyclone Control Suite. All of these options are available to Windows users, and many are also available to macOS/Linux users.

### 8.1 Cyclone Control Suite - Overview

The Cyclone Control Suite is a new generation of automated control software developed to support PC-based control of **Cyclone LC** and **Cyclone FX** stand-alone programmers.

Ways to control a Cyclone include programming launch, results recovery, management of images resident on a Cyclone, addition of unique programming data for each target, as well as recovery of descriptive errors.

The Cyclone Control Suite is included with all touchscreen Cyclones. Much of the feature set works for all touchscreen Cyclones, however some advanced features require a Cyclone-resident Advanced Automation License which comes built into **Cyclone FX** programmers and is available as an upgrade for **Cyclone LC**.

#### 8.1.1 macOS/Linux Support Notes

The following information applies to operation of the Cyclone with macOS and Linux.

##### 8.1.1.1 Minimum Required Versions

The following are the minimum versions required for macOS/Linux support.

macOS: 10.13 High Sierra

Linux: 18.04 LTS

##### 8.1.1.2 Usage Differences

##### 8.1.1.3 macOS/Linux users should note some subtle differences between using the Cyclone Control Suite with macOS/Linux vs, Windows. **Cyclone Control Suite Notes**

macOS/Linux users should note that almost all Cyclone Control Suite features are standard for both Cyclone LC and Cyclone FX. The ability to run tests before programming remains an advanced control feature exclusive to the Cyclone FX model.

**Users should also note:**

1. Cyclones cannot be controlled via Serial Port (RS232) using the Cyclone Control SDK or Cyclone Control Console with macOS/Linux.
2. While control and automation of existing programming images is available for Windows/macOS/Linux platforms via the Cyclone Control Suite, the utility that is used to create and configure programming images (see **Section 6.1 - Cyclone Image Creation Utility**) is currently Windows only. Users should plan their projects accordingly.

##### 8.1.1.4 Distinguishing Windows-Only And macOS/Linux-Specific Content

Information that pertains to the Windows platform only, or is particular to macOS/Linux, is indicated by the symbols below. Content can be considered to apply to all three platforms unless otherwise specified.

Windows-only: 

macOS: 



Linux: 

#### 8.1.2 Components

The Cyclone Control Suite consists of three major components:

1. **Cyclone Control SDK** – This is a Software Development Kit with a comprehensive API allowing multiple Cyclones to be managed simultaneously from a user developed custom



application that loads the provided Cyclone Control library file. The library file can be loaded from many programming languages that are able to load a such a file (C/C++, Delphi, C#, Java, Python, etc) as well as environments like LabVIEW. Examples and interface code are provided in C/C++ (MSVC  and GCC), Delphi/FPC , and more. See **Section 8.2 - Cyclone Control SDK**.








2. **Cyclone Control Console** – This is a powerful command-line application can be launched from a script, a command-line, or another application and allows control of one or more Cyclones simultaneously. The command-line application displays comprehensive status messages and also returns an error code which can be recovered from the calling application. See **Section 8.2.12 - PEcloud Functions**.
3. **Cyclone Control GUI** – This is an interactive GUI based application which provides an easy way to control Cyclones and manage images resident in the Cyclones. Given its graphical nature, it is very easy to explore Cyclone Control Suite capabilities to intuitively control or interact with a Cyclone. See **Section 8.4 - Cyclone Control GUI**.

The Console and GUI were both built using the SDK and are good examples of the types of applications that the SDK can be used to build.

Additional sample applications come as part of the installation. They contain defined build scripts that can be used to build the sample application without any modifications.

### 8.1.3 Standard Features

The three control applications all provide the following Standard features for all Cyclones:

- Cyclone control via multiple connection types:
  -  USB, Serial, or Ethernet
  -   : USB, Ethernet
- Select and Launch Images by Name or Enumeration
- Recover programming result and descriptive error information
- Use automatically counting local (Cyclone stored) serial numbers
- Read/write Cyclone properties
- Read Image and target Properties and Status
-  Remote Display Access and the ability to “touch” the screen
- Add/Remove/Update multiple images in the Cyclone (Console, SDK)
- Simultaneously (Gang) Control multiple Cyclones
  -  USB, Serial, or Ethernet
  -   : USB, Ethernet
- Program (and Read) Dynamic Data in addition to fixed image data
- Overlay Programming for ECC Flash

#### 8.1.3.1 Overlay Programming For ECC Flash

When a device has ECC-enabled flash, one typically must program the data to a sector of flash, and the ECC bits associated with that sector, at the same time. When programming serial numbers, MAC addresses, lot dates, calibration values, etc, the existing startDynamicData function cannot be used because it programs the dynamic data after the primary application data is programmed. This invalidates the ECC bits and corrupts the flash. Previously the only workaround to this issue was to modify the S-Record file manually with the dynamic data prior to programming. Because every single device has unique data, this workaround quickly becomes infeasible for mass production.

Control SDK and Control Console functions have been added that allow dynamic data to be inserted programmatically, on the fly, into the S19 of the SAP image that is being programmed.

### 8.1.4 PEmicro Compatible Hardware

The following lists the PEmicro hardware that is compatible with the Cyclone Control Suite. To ensure proper operations, PEmicro recommends upgrading all Cyclone units to the latest firmware.

- Cyclone FX Universal
- Cyclone FX ARM
- Cyclone LC Universal
- Cyclone LC ARM
- Cyclone PRO (Standard features only)
- Cyclone MAX (Standard features only)
- Cyclone for ARM (Standard features only)
- Cyclone for Renesas (Standard features only)
- Cyclone for STMicro (Standard features only)

## 8.2 Cyclone Control SDK






The Cyclone Control SDK allows a Windows/macOS/Linux user to interact with the Cyclone via a library file (such as Windows' `.dll`, macOS' `.dylib`, or Linux' `.so` files).

### 8.2.1 Introduction

The Cyclone Control SDK is one of the three components that comprise the Cyclone Control Suite. Its library allows the user to create an application on the PC that can directly control one or more PEmicro Cyclone units. These interface routines are designed to be compiled into visual and non visual applications running on Windows, macOS, or Linux operating systems. macOS/Linux SDK files are contained in a separate `.tar` file (see **Section 8.2.5.4 - Compiling**).

The actual interface routines are located in the "cyclonecontrolsdk" library file. This library file is callable from almost any 32-bit / 64-bit Windows, macOS, or Linux development environment. Since the way the library file is called varies depending on the compiler used, PEmicro provides the library interface code and sample applications for each of the following compilers.

**Note:** PEmicro's blog offers articles with detailed setup instructions for some of the options below; see the accompanying links where applicable:

- Borland Delphi 2.0+ (Pascal) 
- GCC
- Microsoft Visual C 
  - Setup info: [http://www.pemicro.com/blog/index.cfm?post\\_id=139](http://www.pemicro.com/blog/index.cfm?post_id=139)
- Microsoft Visual C# 
  - Setup info: [http://www.pemicro.com/blog/index.cfm?post\\_id=161](http://www.pemicro.com/blog/index.cfm?post_id=161)
- NI LabVIEW 2018 
  - Setup info: [http://www.pemicro.com/blog/index.cfm?post\\_id=157](http://www.pemicro.com/blog/index.cfm?post_id=157)
- Python 
  - Setup info: [http://www.pemicro.com/blog/index.cfm?post\\_id=201](http://www.pemicro.com/blog/index.cfm?post_id=201)

### 8.2.2 Backwards Compatibility With Classic Cyclone Control API

The "cyclonecontrolsdk" library file is backwards compatible with many of the classic Cyclone Control API calls. In each header file, there is a constant variable or define which is declared as the file name of the library file. The value of this variable should be changed to the new filename "cyclonecontrolsdk.dll" instead of the old "cyclone\_control.dll". After this modification, rebuild the project and it should continue working with the new library file.

### 8.2.3 Windows 32-Bit and 64-Bit Support

The SDK supports both 32-bit and 64-bit applications. The user will note a deploy folder for all Windows deliverables at: `INSTALLDIR\cycloneControl\controlsdk\deploy\`.

Within this deploy folder, the 32-bit DLL is inside the win32 sub-folder, and the 64-bit DLL is inside the win64 sub-folder.

### 8.2.4 Getting Started with the Cyclone Control Library File - Windows Users

This section outlines the steps you need to take to begin developing your own custom application and offers tips and suggestions to get the Cyclone Control Library File working with your PEmicro hardware smoothly.

**Note:** **Section 8.2.1 - Introduction** includes links to detailed examples for certain compilers which detail how to set up a programming image and use the SDK with some advanced options.

#### 8.2.4.1 Example Programs

Located in the installation directory of the package, you will find two example programs that you can use as a reference for your own application. The examples are located in the following directories:

```
INSTALLDIR\cycloneControl\controlsdk\examples\pascal\
INSTALLDIR\cycloneControl\controlsdk\examples\c\
INSTALLDIR\cycloneControl\controlsdk\examples\labview2018
INSTALLDIR\cycloneControl\controlsdk\examples\python
INSTALLDIR\cycloneControl\controlsdk\examples\msvc
INSTALLDIR\cycloneControl\controlsdk\examples\msvcsharp
```

These example programs are a valuable reference to use when starting your own custom application.

#### 8.2.4.2 Starting Your Own Project

To gain access to the functions available in the library file the following files need to be added to the new project workspace:

##### Delphi 2.0+ Projects

```
INSTALLDIR\cycloneControl\controlsdk\examples\pascal\cyclone_control_api.pas
```

All other source files which will call functions from the library file should include the above file using the Delphi “uses” command.

##### MSVC 5.0+ Projects

```
INSTALLDIR\cycloneControl\controlsdk\examples\c\cyclone_control_api.h
INSTALLDIR\cycloneControl\controlsdk\examples\c\cyclone_control_api.c
```

All other source files which will call functions from the library file should include the above header file with the C/C++ #include directive.

##### MSVC# 2017 Projects

```
INSTALLDIR\cycloneControl\controlsdk\examples\msvcsharp\visual_sap_control\cyclone_control
```

\_api.cs

## NI LabVIEW 2018 Projects

INSTALLDIR\cyclone\cycloneControl\controlsdk\examples\labview2018\user.lib\cyclonecontrolsdk\cyclonecontrolsdk.lvlib

INSTALLDIR\cyclone\cycloneControl\controlsdk\examples\labview2018\user.lib\cyclonecontrolsdk\Vis\\*.vi

The pre-built VIs that we provide include modifications for error handling using error clusters. Please visit our blog post “Automated Flash Programming with LabVIEW” for more information on how to develop your own LabVIEW project.

## Python Projects

INSTALLDIR\cycloneControl\controlsdk\examples\python\cycloneControlSDK.py

### 8.2.5 Getting Started with the Cyclone Control Library File - macOS/Linux Users

This section outlines the steps needed to begin developing a custom application and offers tips and suggestions to get the Cyclone Control SDK library working smoothly with PEmicro hardware.

#### 8.2.5.1 SDK Contents

macOS/Linux users will require the CycloneControlSDK.tar file, an archived (zipped) file which includes macOS and Linux releases.

The release is structured along the following directories:

- Linux: C and C++ example files, the “cyclonecontrolconsole” command line utility, CycloneControlGUI, and the “libcyclonecontrolsdk.so” library
- macOS: C and C++ example files, the “cyclonecontrolconsole” command line utility, CycloneControlGUI, and the “libcyclonecontrolsdk.dylib” library
- Firmware: The latest firmware releases
- User manuals: The user manual for the Cyclone programmer.

#### 8.2.5.2 Example Programs

Located in the macOS and Linux directories of the package, you will find two example programs that you can use as a reference for your own application. The examples are located in the following directories:

INSTALLDIR/macOS/c: C example files for macOS

INSTALLDIR/macOS/cpp: C++ example files for macOS

INSTALLDIR/linux/c: C example files for Linux

INSTALLDIR/linux/cpp: C++ example files for Linux

These example programs are a valuable reference to use when starting your own custom application.

#### 8.2.5.3 Starting Your Own Project

The Cyclone Control Console (cyclonecontrolconsole) comes ready to use. To use it, you can simply invoke a command-line command such as:

```
./cyclonecontrolconsole -cyclone=usb1 -listimages
```

```
./cyclonecontrolconsole -cyclone=10.1.2.3 -launchimage=1
```

The first command will query a Cyclone unit connected via a USB cable and list any programming

images that are on board.

The second command will remotely launch the first programming image that is on a Cyclone with IP number 10.1.2.3.

For a full description please see **Section 8.2.12 - PEcloud Functions**.

#### 8.2.5.4 Compiling M L

In order to compile (or use the Makefile), you will need to have the appropriate compiler (such as gcc or g++) available. These compilers are often native to Linux. Mac users will need to have Xcode and its command-line utilities installed.

##### 8.2.5.4.1 How To Build M L

The examples show how you can write a custom application using the SDK library. To build the example files, change directory into the appropriate folder (such as "cd example\cpp") and invoke the Makefile by calling "make" from a command line.

Please note that the Makefile only builds the example file, it does not copy the files necessary to immediately use the compiled program. To do that, please perform a "make install", this will copy the appropriate firmware files as well as the libcyclonecontrolsdk library, into a "deploy" folder, thereby making the custom program readily usable.

Once you have performed "make" and "make install", you can *cd* into the deploy directory and invoke the program as follows:

```
./apiExample -cyclone=usb1 -listimages
```

This will show the programming images that are on a Cyclone connected through a USB cable.

##### 8.2.5.4.2 Running An Application On macOS M

By default, MacOS does not allow applications from unidentified developers to run on the system. This would cause a message stating that the application "cannot be opened because the developer cannot be verified." There are two ways to resolve this issue:

1. You can simply navigate to the file on an explorer, right click on the executable, and select the "Open with" option to open the application with Terminal.app. You will then be able to open the application.
2. You can completely disable this feature through the System Preferences menu, although it is not a recommended option.

## 8.2.6 Initialization

### Loading the DLL (C/C++ Projects only)

Before calling any routines from the DLL, the DLL must be loaded into memory. To do this, the following function has been provided in the included header files. Refer to Chapter 4 of this manual for a detailed description of this function.

```
loadLibrary( );
```

For Delphi (Pascal) and C# users, this process is transparent for the user and no action is required.

### Enumerate all ports

After loading the DLL, a call to the following function is required in order to properly initialize all devices. This function should only be called once, typically at the beginning of the application.



```
enumerateAllPorts( );
```

### Connect to the PEmicro hardware interface

The next step is to establish communications with the PEmicro Cyclone unit. This is accomplished with the following function call:

```
connectToCyclone( );
```

Refer to Chapter 4 of this manual for a detailed description of this function. This call returns the handle to the Cyclone unit which is used in all other routines in the DLL to identify the Cyclone. Note that the special case of a return value of 0 indicates an error contacting the Cyclone. This function will be called once for each Cyclone.

## 8.2.7 Finalization


Before closing the application, it is recommended that the session with the PEmicro hardware be terminated and the DLL unloaded from memory.

These calls should always be made before the application closes:

```
disconnectFromAllCyclones( );  
unloadLibrary();
```

Note that the “unloadLibrary” call is only required for C/C++ applications. For the Delphi and C# example projects, the DLL is automatically unloaded when the application closes.

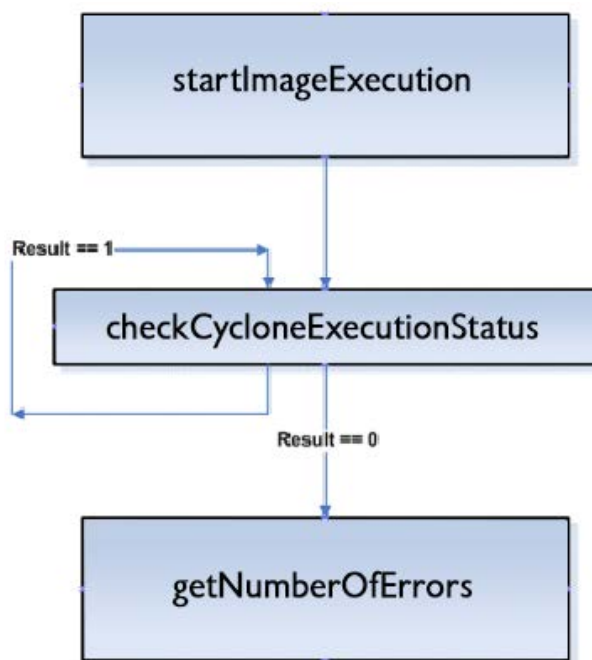
## 8.2.8 Initial Cyclone Setup

The Cyclone Image Creation Utility  software, which is included with each Cyclone, is used to create the standalone images that will be stored in the non-volatile memory of the Cyclone.

**Note:** Mac/Linux users will need a Windows platform for the image creation and configuration process.

These programming images contain the FLASH / EEPROM programming algorithms, the actual binary data to be programmed, the sequence of programming operations, and user specified Cyclone settings.

Prior to using the Cyclone Control Suite, these standalone images need to be created. Please refer to the user’s manual of your Cyclone unit for more information on standalone images and image creation.



**Figure 8-1: Typical programming procedure flow chart**

**Figure 8-1** describes the most common sequence of calls to the DLL after successfully connecting to the Cyclone unit.

- Initiate programming operations. “startImageExecution” carries out the programming operations defined in the stand-alone image stored on the Cyclone unit.
- Wait for programming completion. Note that no error checking is provided by the “checkCycloneExecutionStatus” call. A result of 0 will be returned even if an error has occurred or if communication with the Cyclone is lost.
- Retrieve the error code from the Cyclone unit to determine if the programming was successful.

## 8.2.10

### External Memory Storage Support

Some Cyclones support external memory storage. The Cyclone Control SDK and Cyclone Control Console both support images residing on external memory cards. The parameter “selectedMediaType” is used to select between Cyclone internal Flash and external memory.

Image numbers will go in ascending order starting with image number 1. Internal images will be counted first and external image numbers will start after the last internal image number. Image number 1 will refer to the first image in the internal memory if there are any images in the Cyclone internal memory, if there are no internal images, image 1 will refer to the first image in the external memory.

Modifying images residing in external memory will only be available on Cyclones with the proper license. Certain Cyclones may require a supplementary license to support external memory.

## 8.2.11

### Application Programming Interface (API)

This chapter describes the API of the “CycloneControlSDK.dll” in detail. A C/C++ function prototype is given here. Header files are provided for the following languages:

Pascal

C/C++

### 8.2.11.1 Constants

Name	32-bit Value
CyclonePortType_USB	5
CyclonePortType_Ethernet	6
CyclonePortType_Serial	7
CycloneInformation_IP_Address	1
CycloneInformation_Name	2
CycloneInformation_Generic_Port_Number	3
CycloneInformation_Cyclone_Type_String	4
MEDIA_INTERNAL	1
MEDIA_EXTERNAL	2

### 8.2.11.2 DLL Loading / Unloading Calls

#### 8.2.11.2.1 loadLibrary

*bool loadLibrary(char \*filepath);*

This function loads the CycloneControlSDK.dll into memory and gives the user access to all of the functions available in the library. **This routine must be called before any of the other routines can be called.**

@returnvalue	True if the load was successful, false otherwise.
--------------	---

#### 8.2.11.2.2 unloadLibrary

*void unloadLibrary(void);*

This function unloads the DLL loaded with loadLibrary( ). This call should be made before the application starts to unload itself.

#### 8.2.11.2.3 enumerateAllPorts

*void enumerateAllPorts(void);*

This function performs all necessary initialization in order to successfully communicate with a Cyclone. The function is called once before the first call to "connectToCyclone".

#### 8.2.11.2.4 disconnectFromAllCyclones

*void disconnectFromAllCyclones(void);*

This function closes all open Cyclones (if any) and frees all dynamic memory used by the DLL. The function is called before the user application is closed.

#### 8.2.11.2.5 version

```
char *version(void);
```

This call returns a pointer to a null-terminated string that contains the version number of the DLL.

@returnvalue	A pointer to a null-terminated string containing the version number of the DLL.
--------------	---

#### 8.2.11.2.6 queryNumberOfAutodetectedCyclones

```
uint32_t *queryNumberOfAutodetectedCyclones(void);
```

This function returns the number of Cyclones connected locally on USB and on your local network.

@returnvalue	The number of Cyclones.
--------------	-------------------------

#### 8.2.11.2.7 queryInformationOfAutodetectedCyclone

```
char *queryInformationOfAutodetectedCyclone(int32_t autodetectIndex, int32_t informationType);
```

@parameter autodetectIndex	Specifies the index of the detected Cyclone by the function queryNumberOfAutodetectedCyclones(). The valid range for this parameter is from 1 to the number of Cyclones detected.
@parameter informationType	Specifies the property of the Cyclone to return. The possible values are: <ul style="list-style-type: none"> <li>CycloneInformation_IP_Address</li> <li>CycloneInformation_Name</li> <li>CycloneInformation_Generic_port_number</li> </ul>
@returnvalue	A pointer to a null-terminated string containing the property value of the specified Cyclone.

### 8.2.11.3 Cyclone Connecting / Disconnecting Calls

#### 8.2.11.3.1 connectToCyclone

```
uint32_t connectToCyclone(char *nameIpOrPortIdentifier) ;
```

This function opens a session with a Cyclone and tests the connection. The handle returned by this function is passed as a parameter to other functions provided by the DLL. If you connect to a Cyclone that already has a handle, the same handle is returned. If there is a failure contacting the Cyclone, the function returns a 0.

Note that the DLL does not support multiple Cyclones connected via the serial port. If you require more than one Cyclone to use a serial port connection, PEmicro recommends using the RS232

<p>@parameter nameIPOrPortIdentifier</p>	<p>A pointer to a null-terminated string which uniquely identifies the Cyclone connected to the host PC.</p> <p>If identifying by IP address, the string should be in the format of xxx.xxx.xxx.xxx, where xxx = 0...255.</p> <p>If identifying by name, the string should contain the name of the Cyclone.</p> <p>If identifying by port and the Cyclone is connected by USB, the string should be USB# where # is 1...8. If the Cyclone is connected by Ethernet, the string should be in the format of xxx.xxx.xxx.xxx, where xxx = 0...255.</p> <p>If the Cyclone is connected by Serial, the string should be COM1.</p>
<p>@returnvalue</p>	<p>The handle to the opened Cyclone unit. A return value of 0 indicates a failure to connect to the specified Cyclone unit.</p>

#### 8.2.11.3.2 connectToMultipleCyclones

```
bool connectToMultipleCyclones(char *nameIpOrPortIdentifierArray,
multipleCycloneHandleArrayPtrType cycloneHandleArrayPointer, int32_t
*numberOfCycloneOpensAttempted);
```

This function returns a array of handles to opened Cyclones from a null-terminated String of comma delimited identifiers.

<p>@parameter nameIpOrPortIdentifierArray</p>	<p>A null terminated string containing one or more Cyclone identifiers (name, IP address, or port number) delimited by commas.</p> <p>Example: USB1,209.1.10.2,Orion,COM1</p> <p>If identifying by IP address, the string should be in the format of xxx.xxx.xxx.xxx, where xxx = 0...255.</p> <p>If identifying by port and the Cyclone is connected by USB, the string should be USB# where # is 1...8.</p> <p>If the Cyclone is connected by Serial, the string should be COM1.</p>
---	--



@parameter multipleCycloneHandleArrayPtrType	A pointer to an array of Cyclone handles. Each element of the array corresponds to the position of the identifier in the previous parameter. If the function connected to the Cyclone, the value of the array element would correspond to its handle otherwise it will be 0.
@parameter numberOfCycloneOpensAttempted	This value will be modified with the number of Cyclones that the function attempted to open. It is also the size of the multipleCycloneHandleArrayPtrType structure.
@returnvalue	True if every Cyclone was identified and has a valid handle.  False if there were any errors identifying or connecting to any of the Cyclones.

### 8.2.11.3.3 setLocalMachineIpNumber

```
void setLocalMachineIpNumber(char* ipNumber);
```

If a PC has multiple network interface cards, this function sets the IP address of the network card to use communicate with the Cyclones. This is called prior to calling any other functions.

@parameter ipNumber	A pointer to a null-terminated character string in the format xxx.xxx.xxx.xxx, where xxx = 0...255, representing the IP address of the network card.
---------------------	--

## 8.2.11.4 Controlling Cyclone Programming

### 8.2.11.4.1 checkCycloneExecutionStatus

```
uint32_t checkCycloneExecutionStatus(uint32_t cycloneHandle);
```

Checks to see if the Cyclone has completed a programming operation started with the "startImageExecution" function.

After this function returns with completed value, "getLastErrorCode" should be called to determine the programming result. A result of 0 will be returned even if a programming error has occurred or if communication with the Cyclone is lost.

@parameter cycloneHandle	The handle of the Cyclone to perform a status check on.
@returnvalue	1 = Currently programming 0 = Completed (with or without error)

### 8.2.11.4.2 dynamicReadBytes

```
bool dynamicReadBytes(uint32_t cycloneHandle, uint32_t targetAddress, uint16_t dataLength, char *buffer);
```

This function reads a specified number of bytes from a specified memory address of the target

processor. This call is only valid after first having made a call to the “startImageExecution” function in the sequence of commands.

**Note:** This function is only supported by the Cyclone FX or the Advanced Control Suite Upgrade License.

@parameter cycloneHandle	The handle of the Cyclone that will perform the dynamic read.
@parameter targetAddress	The first memory address of the target processor where the data will be read.
@parameter dataLength	The number of total bytes to read from the target processor
@parameter buffer	A pointer to the array where the data read will be stored. This array must have been allocated prior to calling this function.
@returnvalue	True if the data was successfully read False otherwise

#### 8.2.11.4.3 getNumberOfErrors

*uint32\_t getNumberOfErrors(uint32\_t cycloneHandle);*

This function returns a count of all the errors recorded in the DLL and in the Cyclone.

@parameter cycloneHandle	The handle of the Cyclone to get a count of the errors.
@returnvalue	The number of errors in the DLL and in the Cyclone.

#### 8.2.11.4.4 getErrorCode

*int32\_t getErrorCode(uint32\_t cycloneHandle, uint32\_t errorNum);*

This function returns the error code of the specified error number recorded in the DLL or in the Cyclone. It should be called when the function getNumberOfErrors() is greater than or equal to 1.

@parameter cycloneHandle	The handle of the Cyclone to retrieve the error code.
@parameter errorNum	This specifies the error number. The valid range for this parameter is from 1 to the total number of errors.
@returnvalue	The error code of the DLL or Cyclone

#### 8.2.11.4.5 getLastErrorAddr

*uint32\_t getLastErrorAddr(uint32\_t cycloneHandle);*

If the “getErrorCode” function returns a non-zero value (indicating an error has occurred), this routine can be used to query the address where the error occurred.

@parameter cycloneHandle	The handle of the Cyclone from which to request the error address.
@returnvalue	The memory address where the last programming error occurred.

#### 8.2.11.4.6 **getDescriptionOfErrorCode**

```
char *getDescriptionOfErrorCode(uint32_t cycloneHandle, int32_t errorCode);
```

This function returns a description of the error code.

@parameter cycloneHandle	The handle of the Cyclone to retrieve the error code description.
@parameter errorCode	The error code to check.
@returnvalue	A pointer to a null-terminated character string that contains the error code description.

#### 8.2.11.4.7 **resetCyclone**

```
bool resetCyclone(uint32_t cycloneHandle, uint32_t resetDelayInMs);
```

This function performs a hard reset of the Cyclone. It is the same as pressing the reset button. This is considered a legacy call and does not need to be called by the application.

@parameter cycloneHandle	The handle of the Cyclone that will be reset.
@parameter resetDelayInMs	The reset delay, specified in milliseconds. The delay should be at least 5500 ms.
@returnvalue	True if reset was successful False otherwise

#### 8.2.11.4.8 **clearOverlayProgramData**

```
bool clearOverlayProgramData(uint32_t cycloneHandle);
```

This function clears all dynamic overlay data for the specified Cyclone handle. It is recommended to call this function before **specifyOverlayProgramData**. See **Section 8.2.11.4.9 - specifyOverlayProgramData**.

@parameter cycloneHandle	The handle of the Cyclone to clear dynamic overlay data.
@returnvalue	True if the dynamic overlay data was cleared successfully. False otherwise.

#### 8.2.11.4.9 **specifyOverlayProgramData**

```
bool specifyOverlayProgramData(uint32_t cycloneHandle, uint32_t targetAddress,  
uint32_t dataLength, char* buffer);
```

This function passes the dynamic data that the user wants to program to a specified Cyclone handle. Prior to programming, the dynamic data will be overlaid onto the data in the user's s-record. Must be called prior to **startImageExecution**. See **Section 8.2.11.4.1 - startImageExecution**. See also **Section 8.2.11.4.8 - clearOverlayProgramData**.

**Note:** This function is only supported by the Cyclone FX or the Advanced Control Suite Upgrade License.

@parameter cycloneHandle	The handle of the Cyclone to begin dynamic overlay data programming.
@parameter targetAddress	The first memory address of the target processor where the dynamic overlay data should be written.
@parameter dataLength	The total number of bytes written.
@parameter buffer	A pointer to the array which holds the data to be written.
@returnvalue	True if the dynamic overlay data was added successfully. False otherwise.

## 8.2.11.5 Configuration / Image Maintenance Calls

### 8.2.11.5.1 getImageDescription

```
char *getImageDescription(uint32_t cycloneHandle, uint32_t imageId) ;
```

This function returns the description of a particular image stored on the Cyclone (internal Flash or external memory card). This description is specified by the user when the image is created.

@parameter cycloneHandle	The handle of the Cyclone to get an image description.
@parameter imageId	Used to select which image stored on the Cyclone to read the description from. The valid range of this parameter is from 1 to the total number of images in the Cyclone with the count starting from internal memory and then external memory.  If a Cyclone only stores one image, this parameter should be set to 1.
@returnvalue	A pointer to a null-terminated character string which contains the image description

### 8.2.11.5.2 selectCloudJobForExecution

```
bool selectCloudJobForExecution(uint32_t cycloneHandle, uint32_t jobId);
```

This function attempts to select a cloud Job as the default Job to be programmed.

@parameter cycloneHandle	The handle of the Cyclone to initiate the connection.
@parameter jobId	The ID number of the Job to select.
@returnvalue	Returns true if specified Job was found and selected. Otherwise returns false.

### 8.2.11.5.3 selectLocalImageForExecution

```
bool selectLocalImageForExecution(uint32_t cycloneHandle, uint32_t imageId);
```

This function attempts to select a SAP Image as the default Image to be programmed.

@parameter cycloneHandle	The handle of the Cyclone to initiate the connection.
@parameter imageID	The ID number of the SAP Image to select.
@returnvalue	Returns true if specified Image was found and selected. Otherwise returns false.

#### 8.2.11.5.4 startExecutionOfSelectedImageOrJob

*bool startExecutionOfSelectedImageOrJob(uint32\_t cycloneHandle);*

This function attempts to execute programming of the selected SAP Image or Cloud Job.

@parameter cycloneHandle	The handle of the Cyclone to initiate the connection.
@returnvalue	Returns true if programming execution of the Image or Job was initiated. Otherwise returns false.

#### 8.2.11.5.5 formatCycloneMemorySpace

*bool formatCycloneMemorySpace(uint32\_t cycloneHandle, uint32\_t selectedMediaType);*

This function erases all images stored on the selected media type.

@parameter cycloneHandle	The handle of the Cyclone that will have its images erased
@parameter selectedMediaType	This parameter selects between Cyclone internal Flash (selectedMediaType = 1) or external memory (selectedMediaType = 2).
@returnvalue	True if the erasure was successful False otherwise

#### 8.2.11.5.6 eraseCycloneImage

*bool eraseCycloneImage(uint32\_t cycloneHandle, uint32\_t imageId);*

This function erases the specified image that is stored on the Cyclone. This function is not supported by legacy Cyclones.

@parameter cycloneHandle	The handle of the Cyclone that will have its image erased
@parameter imageId	Selects the image on the Cyclone to use. The valid range of this parameter is from 1 to the total number of images in the Cyclone with the count starting from internal memory and then external memory.  If a Cyclone only stores one image, this parameter is 1.
@returnvalue	True if the erasure was successful False otherwise



### 8.2.11.5.7 addCycloneImage

```
uint32_t addCycloneImage(uint32_t cycloneHandle, uint32_t selectedMediaType, bool
replaceImageOfSameDescription, char *aFile);
```

This function adds a specified stand-alone programming image into the selected media type. The image files have a .SAP file extension and are created with the Cyclone Image Creation Utility. If the Cyclone's storage limits are reached, this routine will return an error.

@parameter cycloneHandle	The handle of the Cyclone that will accept the new image
@parameter selectedMediaType	This parameter selects between Cyclone internal Flash (selectedMediaType = 1) or external memory (selectedMediaType =2).
@parameter replaceImageOfSameDescription	Set to True if you want the image to overwrite any existing images with the same description Set to False if you do not want the image to overwrite any existing images with the same description. An error will occur.
@parameter aFile	A pointer to a null-terminated character string which contains the full path to the .SAP file to be added.
@returnvalue	The image number of the image that was just added. This number is used as the "imageld" parameter for some function calls.  A return value of "0" indicates an error has occurred during the process.

### 8.2.11.5.8 countCycloneImages

```
uint32_t countCycloneImages(uint32_t cycloneHandle);
```

This function returns the number of stand-alone programming images currently stored in the internal Flash and the external memory card of the Cyclone.

@parameter cycloneHandle	The handle of the Cyclone to query for the image count
@returnvalue	The total number of images stored in internal and external memory.

### 8.2.11.5.9 getPropertyValue

```
char *getPropertyValue(uint32_t cycloneHandle, uint32_t resourceOrImageId, char
*categoryName, char *propertyName);
```

This function reads a property value of the Cyclone or a stored SAP image. Examples of properties are Cyclone Name, Cyclone IP Address, Image Name or Image media type. There are different categories with different properties. Refer to the header file for a list of valid category and property

names. The function `getPropertyList` will return a list of valid properties for each category.

@parameter cycloneHandle	The handle of the Cyclone from which to read the property.
@parameter resourceOrImageId	The id for image properties is the image id on the Cyclone. The id for Cyclone or Network properties is 0.
@parameter categoryName	A pointer to a null-terminated character string that contains the category of the property that will be read.
@parameter propertyName	A pointer to a null-terminated character string that contains the name of the property that will be read.
@returnvalue	A pointer to a null-terminated character string that contains the value of the property.

#### 8.2.11.5.10 setPropertyValue

```
bool setPropertyValue(uint32_t cycloneHandle, uint32_t resourceOrImageId, char *
categoryName, char *propertyName, char *newValue);
```

This function changes the property of the Cyclone to the value specified. Only certain properties can be changed using this call. This function will return false if the property was not changed. Refer to the header file for a list of valid category and property names. The function `getPropertyList` will return a list of valid properties for each category.

@parameter cycloneHandle	The handle of the Cyclone from which to read the property.
@parameter resourceOrImageId	The id for image properties is the image id on the Cyclone. The id for Cyclone or Network properties is 0.
@parameter categoryName	A pointer to a null-terminated character string that contains the category of the property that will be modified.
@parameter propertyName	A pointer to a null-terminated character string that contains the name of the property that will be modified.
@parameter newValue	A pointer to a null-terminated character string that contains the new value of the property.
@returnvalue	True if the data was successfully set False otherwise

#### 8.2.11.5.11 getPropertyList

```
char *getPropertyList(uint32_t cycloneHandle, uint32_t resourceOrImageId, char
*categoryName);
```

This function returns a list of valid category and property names that can be used with the “`getPropertyValue`” and “`setPropertyValue`” functions. Refer to the header file for a list of valid category and property names.

@parameter cycloneHandle	The handle of the Cyclone that will return the property list.
@parameter resourceOrImageId	The id for image properties is the image id on the Cyclone. The id for Cyclone or Network properties is 0.

@parameter categoryName	A pointer to a null-terminated character string that contains the category of the property list.
@returnvalue	A pointer to a null-terminated character string that contains a list of property names.

## 8.2.11.6 Features Calls

### 8.2.11.6.1 getFirmwareVersion

```
char *getFirmwareVersion(uint32_t cycloneHandle);
```

This function reads the firmware version of the selected Cyclone.

@parameter cycloneHandle	The handle of the Cyclone of which to read the firmware version.
@returnvalue	Returns a pointer to a null-terminated character string containing the firmware version.

### 8.2.11.6.2 cycloneSpecialFeatures

```
bool cycloneSpecialFeatures(uint32_t featureNum, bool setFeature, uint32_t paramValue1, uint32_t paramValue2, uint32_t paramValue3, void *paramReference1, void *paramReference2);
```

This function is used for executing special features described by the featureNum parameter. Refer to the parameter specifications below for details.

@parameter featureNum	CYCLONE_GET_IMAGE_DESCRIPTION_FROM_FILE
@parameter setFeature	Indicates whether the image file has been decoded. If previous operations has already decoded the file, then set it to true. Otherwise set it to false.
@parameter paramValue1	Ignored, set it to 0.
@parameter paramValue2	Ignored, set it to 0.
@parameter paramValue3	Ignored, set it to 0.
@parameter paramReference1	A pointer to a pointer to a null-terminated character string that contains the image description of the SAP file.
@parameter paramReference2	A pointer to a null-terminated character string which contains the full path to the specified SAP file.
@returnvalue	True if the image description was read False otherwise

@parameter featureNum	CYCLONE_GET_IMAGE_CRC32_FROM_FILE
-----------------------	-----------------------------------

@parameter setFeature	Indicates whether the image file has been decoded. If previous operations has already decoded the file, then set it to true. Otherwise set it to false.
@parameter paramValue1	Ignored, set it to 0.
@parameter paramValue2	Ignored, set it to 0.
@parameter paramValue3	Ignored, set it to 0.
@parameter paramReference1	A pointer to an unsigned 32-bit value containing the CRC32 of the SAP file.
@parameter paramReference2	A pointer to a null-terminated character string which contains the full path to the specified SAP file.
@returnvalue	True if the CRC32 was read False otherwise

@parameter featureNum	CYCLONE_GET_IMAGE_SETTINGS_FROM_FILE
@parameter setFeature	Indicates whether the image file has been decoded. If previous operations has already decoded the file, then set it to true. Otherwise set it to false.
@parameter paramValue1	The type of setting to extract {configuration script=1, image unique id etc.=2, serial numbers=3, additional settings=4, crc_exclusive settings=5}.
@parameter paramValue2	Ignored, set it to 0.
@parameter paramValue3	Ignored, set it to 0.
@parameter paramReference1	A pointer to a pointer to a null-terminated character string that contains the settings of the SAP file.
@parameter paramReference2	A pointer to a null-terminated character string which contains the full path to the specified SAP file.
@returnvalue	True if the image settings was read False otherwise

@parameter featureNum	CYCLONE_GET_IMAGE_COMMAND_LINE_PARAMS_FROM_FILE
@parameter setFeature	Indicates whether the image file has been decoded. If previous operations has already decoded the file, then set it to true. Otherwise set it to false.
@parameter paramValue1	Ignored, set it to 0.
@parameter paramValue2	Ignored, set it to 0.

@parameter paramValue3	Ignored, set it to 0.
@parameter paramReference1	A pointer to a pointer to a null-terminated character string that contains the command line parameters of the SAP file.
@parameter paramReference2	A pointer to a null-terminated character string which contains the full path to the specified SAP file.
@returnvalue	True if the command line parameters was read False otherwise

@parameter featureNum	CYCLONE_GET_IMAGE_SCRIPT_FILE_FROM_FILE
@parameter setFeature	Indicates whether the image file has been decoded. If previous operations has already decoded the file, then set it to true. Otherwise set it to false.
@parameter paramValue1	Ignored, set it to 0.
@parameter paramValue2	Ignored, set it to 0.
@parameter paramValue3	Ignored, set it to 0.
@parameter paramReference1	A pointer to a null-terminated character string that contains the name of the script file.
@parameter paramReference2	A pointer to a null-terminated character string which contains the full path to the specified SAP file.
@returnvalue	True if the script file was read False otherwise

@parameter featureNum	CYCLONE_TOGGLE_POWER_NO_DEBUG
@parameter setFeature	Ignored, set it to false.
@parameter paramValue1	Ignored, set it to 0.
@parameter paramValue2	Ignored, set it to 0.
@parameter paramValue3	Ignored, set it to 0.
@parameter paramReference1	Ignored, set it to null.
@parameter paramReference2	Ignored, set it to null.
@returnvalue	True if the power was toggled. False otherwise



@parameter featureNum	CYCLONE_SET_ACTIVE_SECURITY_CODE
@parameter setFeature	Ignored, set it to true.
@parameter paramValue1	The security code type (e.g. MON08, Renesas, PPC Nexus 64 bit, PPC Nexus 256 bit ignored, set it to 0).
@parameter paramValue2	The length of the security code bytes.
@parameter paramValue3	Ignored, set it to 0.
@parameter paramReference1	A pointer to an array containing the security code bytes.
@parameter paramReference2	A pointer to a null-terminated character string containing the security type string (such as 'MON08', 'RENASAS', 'PPCNEXUS').
@returnvalue	True if the security code was set False otherwise

## 8.2.12 PEcloud Functions

These functions relate to the PEcloud service.

### 8.2.12.1 initiateConnectionToCloud

```
bool initiateConnectionToCloud(uint32_t cycloneHandle);
```

This function attempts to establish an online connection to PEcloud.

@parameter cycloneHandle	The handle of the Cyclone to initiate the connection.
@returnvalue	Returns a boolean value. True means a connected to PEcloud has been established; false means this connection was not established.

### 8.2.12.2 countCloudVirtualFactories

```
uint32_t countCloudVirtualFactories(uint32_t cycloneHandle);
```

This function counts the Virtual Factories in the connected PEcloud account.

@parameter cycloneHandle	The handle of the Cyclone connected to PEcloud.
@returnvalue	Returns the number of Virtual Factories in the connected PEcloud account

### 8.2.12.3 getCloudVirtualFactoryDescription

```
char *getCloudVirtualFactoryDescription(uint32_t cycloneHandle, uint32_t factoryIndex );
```

This function retrieves the description for a specific Virtual Factory in the connected PEcloud

account.

@parameter cycloneHandle	The handle of the Cyclone connected to PEcloud.
@parameter factoryIndex	The index value of the Virtual Factory to reference.
@returnvalue	Returns the description of the Virtual Factory referenced by the factory index..

#### 8.2.12.4 countCloudJobs

*uint32\_t countCloudJobs(uint32\_t cycloneHandle, uint32\_t factoryIndex);*

This function counts the number of Cloud Jobs assigned to the Cyclone in PEcloud. Each Cyclone can only belong to one Cyclone Group, and each Cyclone Group can only belong to one Virtual Factory.

@parameter cycloneHandle	The handle of the Cyclone connected to PEcloud.
@parameter factoryIndex	The index value of the Virtual Factory to reference.
@returnvalue	Returns the number of Cloud Jobs assigned to the Cyclone.

#### 8.2.12.5 getCloudJobId

*uint32\_t getCloudJobId(uint32\_t cycloneHandle, uint32\_t factoryIndex, uint32\_t jobIndex);*

This function returns the ID number of the Job in the associated Virtual Factory based on its index number.

@parameter cycloneHandle	The handle of the Cyclone connected to PEcloud.
@parameter factoryIndex	The index value of the Virtual Factory to reference.
@parameter jobIndex	The index value of the Job.
@returnvalue	Returns a boolean value. True means a connected to PEcloud has been established; false means this connection was not established.

#### 8.2.12.6 getCloudJobDescription

*char \*getCloudJobDescription(uint32\_t cycloneHandle, uint32\_t jobId);*

This function returns the description of the specified Job.

@parameter cycloneHandle	The handle of the Cyclone to initiate the connection.
@parameter jobId	The Job ID number.
@returnvalue	Returns the description of the specified Job

#### 8.2.12.7 Additional PECloud Commands

For selectCloudJobForExecution and startExecutionOfSelectedImageOrJob please refer to **Section 8.2.11.5 - Configuration / Image Maintenance Calls.**

## 8.3 Cyclone Control Console


The Cyclone Control Console, the next component of the Cyclone Control Suite, is an application that controls Cyclone operations through the use of simple console commands. This application is very easy for Windows/macOS/Linux users to set up and use and offers functionality very similar to using the Cyclone Control library file directly.

To find the Cyclone Control Console application, navigate to the following directory:

[INSTALLDIR]\Cyclone Control\

The Cyclone Control software performs all operations specified in simple commands. A separate batch file would typically be used to launch the utility with the correct parameters.

### 8.3.1 Startup

- Connect all Cyclone units to the PC via RS232 , USB, or Ethernet. Any combination of different connections is allowed. The exception is that only one RS232 serial port connection can be used; no more than one Cyclone can be connected via the RS232 serial port.
- Connect all Cyclones to their target systems. This is done using a ribbon cable that connects from the Cyclone to a debug header on the target board.
- Power up the PC, all Cyclone units, and all target systems that require external power.
- Run the CycloneControlConsole software from the command prompt.

### 8.3.2 Command-Line Parameters

The command-line utility supports the following commands:

#### -listcyclones

Shows a list of auto-detected Cyclones

#### -cyclone=[cyclone identifier]

Opens the Cyclone or Cyclones by name or identifier. The Cyclone argument can be a single identifier or a comma delimited list. Available identifiers are cyclone name, port number, or IP address.

#### -listimages

List the images present on all open Cyclones. If there are no open Cyclones the command will list the images on all detected Cyclones.

#### -launchimage=[image name or number]

Launch a specific image on the open Cyclones. The image can be identified by name or image number.

#### -specifyOverlayData=[cyclone number],[address],[data]

Performs programming of dynamic data with the selected Cyclone unit. [cyclone number] is the index of the connected Cyclone in the order in which it was entered. [address] is the starting memory address. [data] are the bytes of data to be programmed. All values should be in hexadecimal format. No more than 255 bytes may be programmed this way.

The data specified will replace the data contained in the SAP image, whether those locations are blank or not. If the original locations are not blank, and if the VC or VV commands exist in the SAP image, the VC or VV commands will fail since the original data were replaced with these overlay data. But the VM command verifies against the overlay data.

The "-launchimage" command must be issued together with this command. It does not matter whether -specifyoverlaydata or -specifyoverlaystring is issued before or after the -launchimage command, since each command has its priority assigned to make sure they are executed accordingly.

### **-specifyOverlayString=[cyclone number],[address],[string]**

Performs programming of dynamic data with the selected Cyclone unit. [cyclone number] is the index of the connected Cyclone in the order in which it was entered. [address] is the starting memory address. [string] is the data in string format. No more than 255 bytes may be programmed this way.

The data specified will replace the data contained in the SAP image, whether the locations are blank or not. If the original locations are not blank, and if the VC or VV commands exist in the SAP image, the VC or VV commands will fail since the original data were replaced with these overlay data. But the VM command verifies against the overlay data.

The "-launchimage" command must be issued together with this command. It does not matter whether -specifyoverlaydata or -specifyoverlaystring is issued before or after the -launchimage command, since each command has its priority assigned to make sure they are executed accordingly.

### **-showproperties=[category],[image id if applicable]**

List all available properties in a [category] in the open Cyclones or a stored SAP image by using the [image id if applicable] argument. Refer to the header file for a list of valid category and property names. Using an empty string as the [category] value will return the available categories.

### **-addimageinternal=[filename]**

Add a specific programming image to the internal memory of the open Cyclones. [filename] is the path and filename of the image to be programmed into the Cyclones.

### **-addimageexternal=[filename]**

ONLY SUPPORTED BY CYCLONE FX.

Add a specific programming image to the external memory of the open Cyclones. [filename] is the path and filename of the image to be programmed into the Cyclones.

### **-eraseallinternalimages**

Perform a format of the internal memory connected to the open Cyclones. This will cause all internal images to be erased.

### **-eraseallexternalimages**

Perform a format of the external memory connected to the open Cyclones. This will cause all external images to be erased.

### **-eraseimage=[image name or number]**

Erase an individual image from the open Cyclones. The image can be identified by image name or by image number.

### **-listserialfiles**

List Serial Files in Cyclone

### **-addserialfile=[file path]**

Add Serial File to Cyclone

### **-deleteserialfile=[object ID or display index]**

Delete Serial File from Cyclone. Can be deleted by ID or by display index.

### **-firmwareupdate=[firmware update mode]**

Set the firmware update mode to “auto”, “dontupdate”, “forceupdate”. The default mode is “auto.”

### **-help**

Display a list of available commands.

**Note:** The following command-line parameters require a **Cyclone FX**, or a **Cyclone LC** with the ProCryption Security Activation License.

### **-listencryptionkeys**

List Encryption Key Files that reside on Cyclone

### **-addencryptionkey=[file path]**

Securely add ImageKey file to Cyclone (encrypted with Cyclone-specific RSA public key). The file path is the path to the ImageKey.

### **-deleteencryptionkey=[object ID or display index]**

Delete ImageKey File from Cyclone. Can be deleted by ID, or by display index number (-listencryptionkeys returns a numbered list of ImageKeys; the display index corresponds to these numbers).

**Note:** The following command-line parameters are used with the **PEcloud** service.

### **-cloudcredentials=[file path]**

Specifies which pecloud credentials file (located at [file path]) to use for the current command (required to be able to access things like the vf info and check that the user has access to it)

### **-cloudgetvfinfo=[showcomplete or blank]**

Lists all jobs and job information (job name, job id, status, attempts count, successes count, and failures count) in each virtual factory the user specified in cloudcredentials has access to. If showcomplete is specified it will show all jobs including completed jobs, if left blank it will just list the running jobs.

### **-cloudgetjobinfo=[job id or job name]**

Lists job information specified by the job id or job name (job name, job id, status, attempts count, successes count, and failures count).

### **-cloudsetserialcount=[Serial UID],[Count]**

Sets the serial count for the user specified in cloudcredentials based on the Serial UID and count inputted

### **-cloudgetjoblog=[job id or job name],[Number of Entries]**



Lists the [Number of Entries] first logs stored in the job specified by the job id or job name. Prints out the programming count number, result, and dynamic data associated with that programming attempt.

**-debugprovisioncyclone=[VF id],[Cyclone Group id]**

Provisions a cyclone to the virtual factory and group specified by the VF id and cyclone group id if the user in -cloudcredentials has access to them.

**-clouddeletevfjobs=[VF id]**

Deletes all jobs (including completed ones) from the virtual factory specified by [VF id]

**-launch**

Launches the currently selected Programming Image or Cloud Job. Will login to PEcloud if not logged in.

**-listjobs**

Shows list of cloud Jobs available to a Cyclone. Will login to PEcloud if not logged in.

**-provisioncyclone=[provisioning pin]**

Provisions the cyclone specified in -cyclone to the cloud given the 8-digit pin on the PECloud website

**-deprovisioncyclone**

Deprovisions the cyclone(s) specified in -cyclone from the cloud

**-selectjob=[job id or job name]**

Selects a cloud Job for execution/launch. Will login to PEcloud if not logged in.

**-selectimage=[image name or #]**

Selects a programming Image for execution/launch.

### 8.3.3 Examples

This section provides some basic examples that demonstrate how to use the Cyclone Control Console to connect to the Cyclone, manage programming images, launch programming, etc.

**Note:** A more expansive list of examples is available on our website at:

**[http://www.pemicro.com/blog/index.cfm?post\\_id=142](http://www.pemicro.com/blog/index.cfm?post_id=142)**

The commands should be separated by a space. Every command starts with a “-” character, arguments follow the “=” character.

#### 8.3.3.1 Typical Usage

```
CycloneControlConsole.exe -cyclone=10.0.1.1 -launchimage=1
```

This example connects to a single Cyclone identified by its IP address 10.0.1.1 and executes its

first image. This is the most common usage of the Cyclone Control Utility.

```
CycloneControlConsole.exe -cyclone=USB1 -launchimage=1
```

This example connects to a single Cyclone enumerated on USB1 and executes its first image.

#### 8.3.3.2 Controlling Multiple Cyclones

```
CycloneControlConsole.exe -cyclone=USB1,10.0.1.2,10.0.1.3 -  
launchimage=2
```

This example connects to three separate Cyclone units. Two units are connected via USB (10.0.1.1 and 10.0.1.2) and the third is connected via Ethernet (10.0.1.3). The three Cyclone units are configured to execute image #2.

#### 8.3.3.3 Executing more than 1 image on the same Cyclone

```
CycloneControlConsole.exe -cyclone=CycloneFX_1,CycloneFX_2  
-launchimage=1 -launchimage=2
```

Two Cyclone units are connected via Ethernet and both Cyclone units execute their first image. Afterwards, the both Cyclones will execute their second image.

This example is useful when the processor's code is split into two separate images. For example, one image could contain the bootloader while the second image contains the main application code.

#### 8.3.3.4 Image Management – Modifying External Images

```
CycloneControlConsole.exe -cyclone=USB1 -eraseallexternalimages  
-addexternalimage=c:\images\externalImage1.SAP
```

In this example, a Cyclone unit is connected via USB and it has an image stored on its external memory card. The external image is erased and a new one is added. This type of command should only be used when an image needs to be updated.

#### 8.3.3.5 Image Management – Modifying Images on Two Cyclones in Parallel

```
CycloneControlConsole.exe -cyclone=USB1,USB2 -eraseallinternalimages  
-addinternalimage=c:\images\Image1.SAP
```

In this example, two Cyclone units connected via USB have their images erased and a new image is added to both Cyclone units. This type of command should only be executed when images need to be updated.

### 8.3.3.6 Overlay of Dynamic Data

```
CycloneControlConsole.exe -cyclone=USB1 -  
specifyOverlayData=1,1080,80,69,77,73,67,82,79 -launchimage=1
```

This example demonstrates the overlaying of dynamic data for programming. The Cyclone is connected via the USB port. Any stale data is cleared and a new set of dynamic data is passed to the Cyclone. In this case, 7 bytes of data is overload onto your s-record starting at address 0x1080. Then image #1 is executed.

## 8.4 Cyclone Control GUI

As part of the Cyclone Control Suite, PEmicro includes the Cyclone Control GUI, a graphical application that gives the users access to all the functions of the latest Cyclone Control.

**Note:** This new application replaces the previous Image Manager Utility and Cyclone Config IP Utility. The Cyclone Control GUI allows users to add and remove images from the internal Cyclone memory as well as from the external memory cards. The utility also allows the user to view and edit Cyclone properties, to view the Cyclone LCD remotely, to view and add Cyclone Licenses, and to view and manage Serial Files and Encryption Keys.

The utility is composed of three main parts: a connection dialog, the control tabs, and a status window.

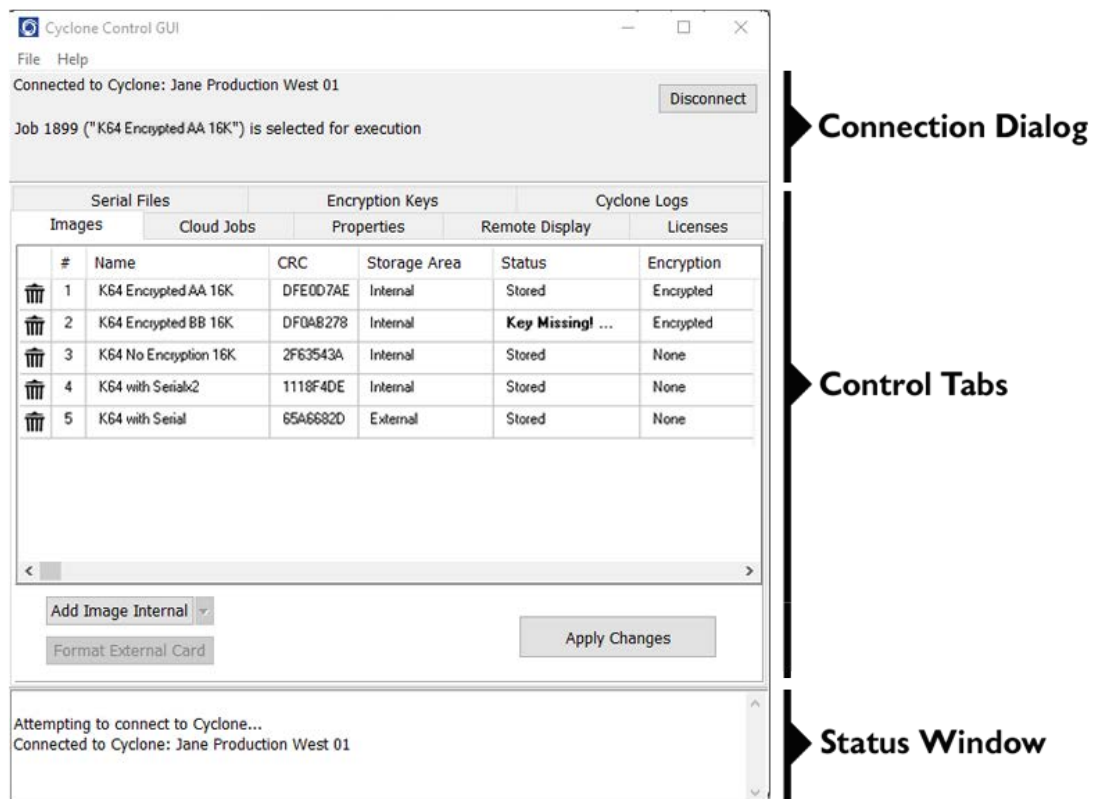


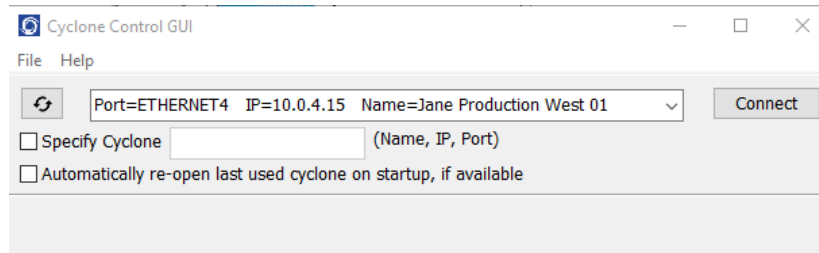
Figure 8-2: Areas of the Cyclone Control GUI

### 8.4.1 The Connection Dialog

Allows the user to specify a Cyclone to connect with, as well as specify the connection options. The utility will always automatically upgrade the firmware of a Cyclone if the firmware in the Cyclone is outdated. However, firmware update can also be forced by using the checkbox in File-

>Force Firmware Update. This option will update the firmware on the Cyclone with the latest firmware in the same folder as the Cyclone Control GUI.

At launch, the Cyclone Control GUI will show all the Cyclones detected on the network and those attached by USB connections in a drop-down list. Cyclones can also be connected to by using the “Specify Cyclone” checkbox and specifying a Cyclone by identifier. This identifier can be the Cyclone name, its IP address or its port number.



**Figure 8-3: Select Cyclone From Drop Down**

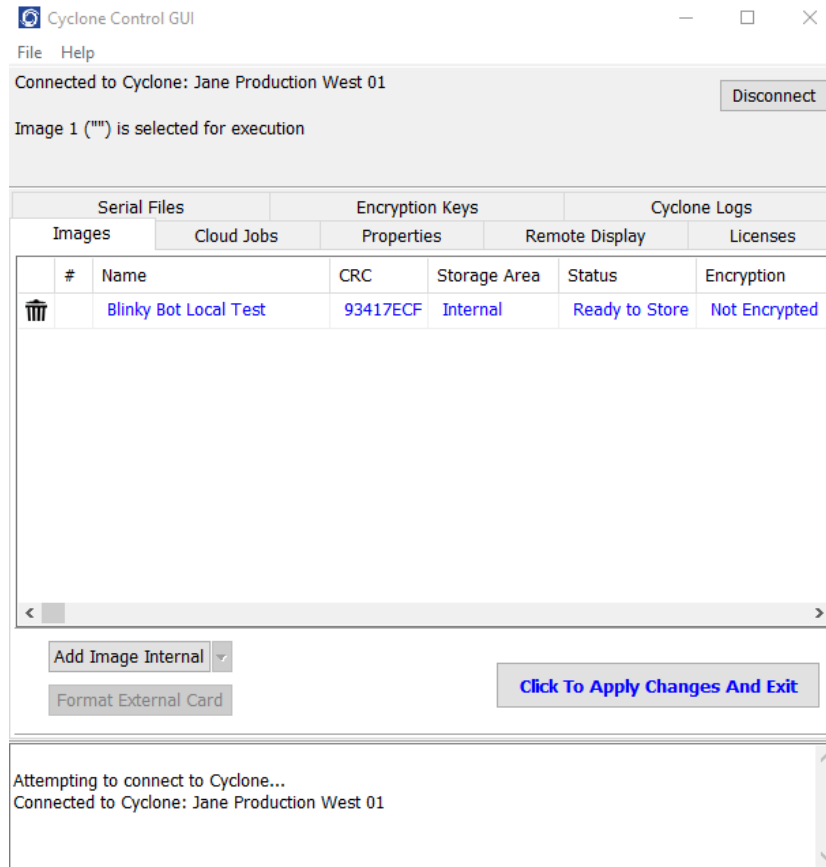
Once the Cyclone is selected, clicking on the “Connect” button will bring a series of tabs that will allow full access to the Cyclone.

## 8.4.2 The Control Tabs

The control tabs allow access to the Cyclone. Through these tabs you can view, add, and erase Cyclone images, you can view and modify properties, you can modify licenses, see the Cyclone screen remotely and view/manage Serial Files and Encryption Keys.

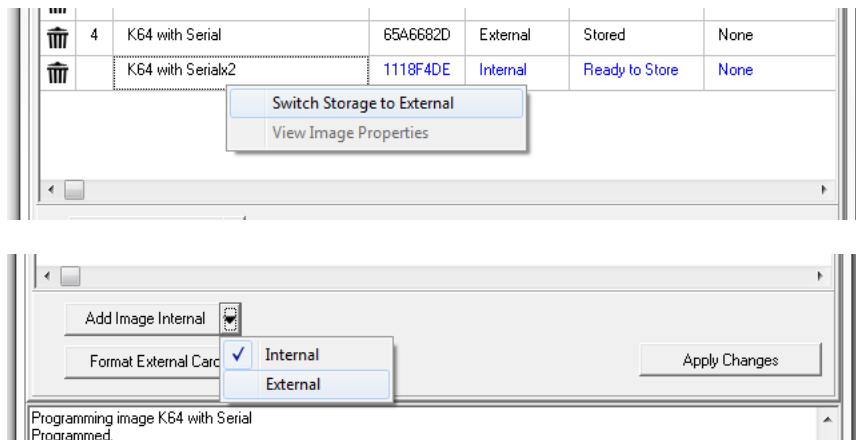
### 8.4.2.1 Images Tab

The Images tab allows the user to view and modify the Cyclone images both in internal Cyclone memory and on external memory cards. To add a new image to the Cyclone, click on the “Add Image Internal” button and selected the image you want to add. For the changes to take effect, the “Apply Changes” button needs to be clicked, this will place the image in the Cyclone memory.



**Figure 8-4: Images Tab - Image Ready to Store**

**CYCLONE FX** programmers (and legacy Cyclone LC with license) can also store images in external memory (an SDHC card). The storage area for the Add Image button is toggled between Internal and External by clicking on the drop down arrow on the right of the “Add Image” button. Users can also right-click on an uncommitted image to toggle the storage between Internal and External.



**Figure 8-5: Changing Storage Area - Right Click or Drop-Down Selections**

**Note:** An image displayed in blue is not yet committed, so disconnecting from the Cyclone before the “Apply Changes” button is clicked will discard any changes not committed.

Erasing an image can be done by clicking on the trashcan icon at the left of the image, it can also be done by selecting the image and click the DELETE key on the keyboard. The “Apply Changes” buttons must be clicked for any changes to the Cyclone to take place.



To format the external memory card click on the “Format External Card” button. This will erase all image information stored in the external card.

#### 8.4.2.1.1 Categories

The following information is displayed for each image in the Images Tab:

**Delete Image** - Clicking the trash icon will designate an image for deletion. Apply Changes must be clicked to execute this action.

**Number** - Displays the Cyclone’s numbering for each image.

**Name** - Displays the Image Name.

**CRC** - Displays the CRC value for each image.

**Storage Area** - “IN” designates that the image is stored in the Cyclone’s internal memory; “EX” designates that it is stored in external memory (SDHC Card).

**Status** - This displays whether the image is “Stored” on the Cyclone, “Ready to store”, or “Ready to erase”. A stored image that is encrypted but missing its ImageKey will always show “Key Missing!” to indicate that the user must provision the Cyclone with this key in order to program that encrypted image.

**Encryption** - Displays “Encrypted” for encrypted images, and “None” for images that have not been encrypted.

For more about encrypted programming images, please see **Section 6.3.1.1 - ProCryption Security Features**.

#### 8.4.2.1.2 View Image Properties

From this tab a user can also access the image properties of images in the Cyclone. Just right click on the image then click on “View Image Properties” and a window with public image properties will pop up.

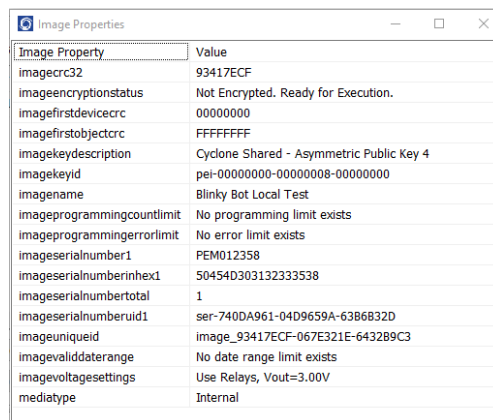


Image Property	Value
imagecrc32	93417ECF
imageencryptionstatus	Not Encrypted. Ready for Execution.
imagefirstdevicecrc	00000000
imagefirstobjectcrc	FFFFFFFF
imagekeydescription	Cyclone Shared - Asymmetric Public Key 4
imagekeyid	pei-00000000-00000008-00000000
imagename	Blinky Bot Local Test
imageprogrammingcountlimit	No programming limit exists
imageprogrammingerrorlimit	No error limit exists
imageserialnumber1	PEM012358
imageserialnumberhex1	504540303132333538
imageserialnumbertotal	1
imageserialnumberuid1	ser-740DA961-04D9659A-63B6B32D
imageuniqueid	image_93417ECF-067E321E-6432B9C3
imagevaliddaterange	No date range limit exists
imagevoltage settings	Use Relays, Vout=3.00V
mediatype	Internal

**Figure 8-6: Image Properties Window**

Properties like the image name, the voltage settings, image CRC, encryption info, and all current serial numbers can be viewed from this window. Use this feature to make sure your image settings are correct or check that the serial numbers change accordingly.

## 8.4.2.2 Properties Tab

Serial Files		Encryption Keys		Cyclone Logs	
Images	Cloud Jobs	Properties	Remote Display	Licenses	
Property		Value			
currentimageselect		1			
cyclonefirmwareversion		11.10			
cyclonehardwareversion		Rev. C			
cyclonelogicversion		3.4			
cyclonename		<a href="#">Jane Production West 01</a>			
cyclonetype		Cyclone Universal FX			
numberofexternalimages		0			
numberofinternalimages		1			
relaystatus		1			
selectedjobimagedescription		Blinky Bot Local Test			
selectedjobimagenumber		1			
selectedjobimagetype		IMAGE			
totalnumberofimages		1			
rh850tagcode		<a href="#">0</a>			
controlportstatus		Active. Triggering disabled (controlporttriggerenable=0).			
controlporttriggerenable		<a href="#">0</a>			
hardware_button_enabled		<a href="#">1</a>			
lcd_screen_enabled		<a href="#">1</a>			
touch_screen_enabled		<a href="#">1</a>			
connectedtocloud		true			
cyclonednsaddress		<a href="#">10.0.0.14</a>			
cycloneipaddress		<a href="#">10.0.4.15</a>			
cyclonemacaddress		00-0D-01-87-D7-E2			
cyclonenetworkgateway		<a href="#">10.0.0.2</a>			
cyclonenetworkmask		<a href="#">255.0.0.0</a>			
advancedautomationlicense		1			
externalmemorystatus		externalmemorynotdetected			
isencryptionsupported		1			
isexternalmemorysupported		1			
isindividualimageasesupported		1			
isremoterviewsupported		1			

**Figure 8-7: Properties Tab**

The properties tab shows all the network, Cyclone and image properties. It also shows properties for the supported features of the Cyclone. From this tab the Cyclone firmware and logic versions, the cyclone type, and the number of images are available. Also from this tab some of the properties can be modified.

### 8.4.2.2.1 Modifying A Cyclone Property

Some properties in the Cyclone are modifiable. The properties that can be modified will be displayed in blue.

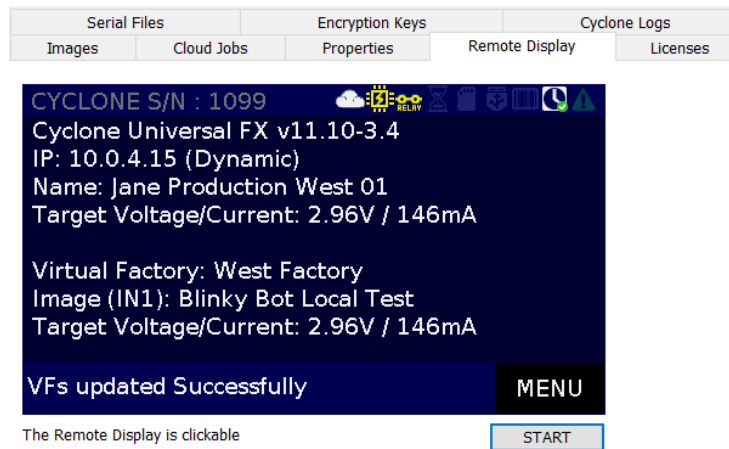
cyclonelogicversion	3.4
cyclonename	<a href="#">Jane Production West 01</a>
cyclonetype	Cyclone Universal FX

**Figure 8-8: Modifying Cyclone Properties**

Double-clicking on these properties will change the blue text to black and allow the user to edit the field. Edit the value and then click “Enter” to save the new value, the window will refresh and the value shown will be the updated value of the property.

### 8.4.2.3 The Remote Display Tab

This tab will show the current display of the Cyclone. The utility checks every second for changes in the display and updates the image to the current display. This image is also clickable so clicks on the virtual screen are also registered by the Cyclone.

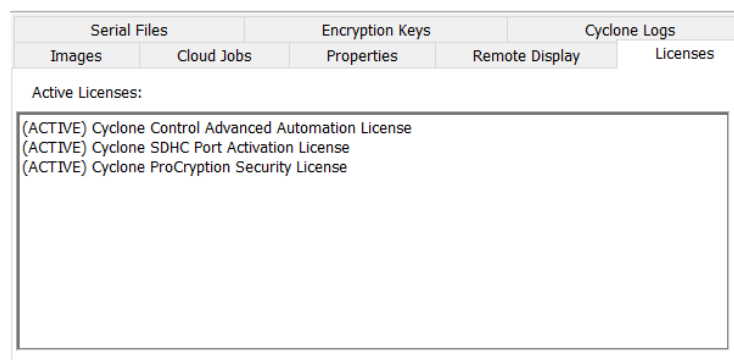


**Figure 8-9: Remote Display & Control of Cyclone Screen**

#### 8.4.2.4 Licenses Tab

New licenses can be added to the Cyclone using the Licenses tab. Clicking on “Add New License” will prompt the PEmicro License Activation Form. This form takes in the Installation Code for a product and can automatically validate and register the installation.

This tab also shows the current licenses active in the Cyclone.



**Figure 8-10: Licenses Tab**

#### 8.4.2.5 Serial Files Tab

The Serial Files Tab displays any Serial Files that are on the Cyclone. Serial Files can be “deleted” with the “Delete Serial File” button, but in this case it simply means that they will not be displayed. An image file that uses a “deleted” Serial File will simply cause the file to re-appear and resume with its next serial number.

**Note:** Older images may reference Serial Files that are image-specific and cannot be shared among images (like current Serial Files). These will not appear in the Serial Files tab. The Cyclone menu can be used to make changes to the serialization of these images.

Images		Cloud Jobs	Properties	Remote Display	Licenses
Serial Files		Encryption Keys		Cyclone Logs	
#	Type	Name	ID		
1	serialfile	PE Test Serial	ser-740DA961-04D9659A-6386832D		
2	serialfile	Widget Serial	ser-69EA91AA-6AF1C005-64113F00		
3	serialfile	Widget Serial	ser-923B1C1B-0334C828-641141A4		
4	serialfile	Widget Serial	ser-97EBC775-6E0660B6-64114203		

<

Delete Serial File

**Figure 8-11: Serial Files Tab**

#### 8.4.2.6 Encryption Keys Tab

**CYCLONE FX** programmers and **CYCLONE LC** programmers with the ProCryption Security Activation License are able to encrypt programming images with a user-generated ImageKey. In order to load an encrypted image onto a Cyclone, or to program with an encrypted image once it is on the Cyclone, the ImageKey used to encrypt that image must be present on the Cyclone.

The Encryption Keys tab displays any ImageKeys that reside on the Cyclone. The Name and unique ID number for each ImageKey is shown.

Images		Cloud Jobs		Properties		Remote Display		Licenses	
Serial Files			Encryption Keys			Cyclone Logs			
#	Type	Name				ID			
1	peimagekey	Cloud Demo Key				pei-C141A39C-119F404F-63B6A775			
2	peimagekey	KMDockKey01				pei-30825DD1-77169A83-641C5995			

Add Encryption Key
Delete Encryption Key

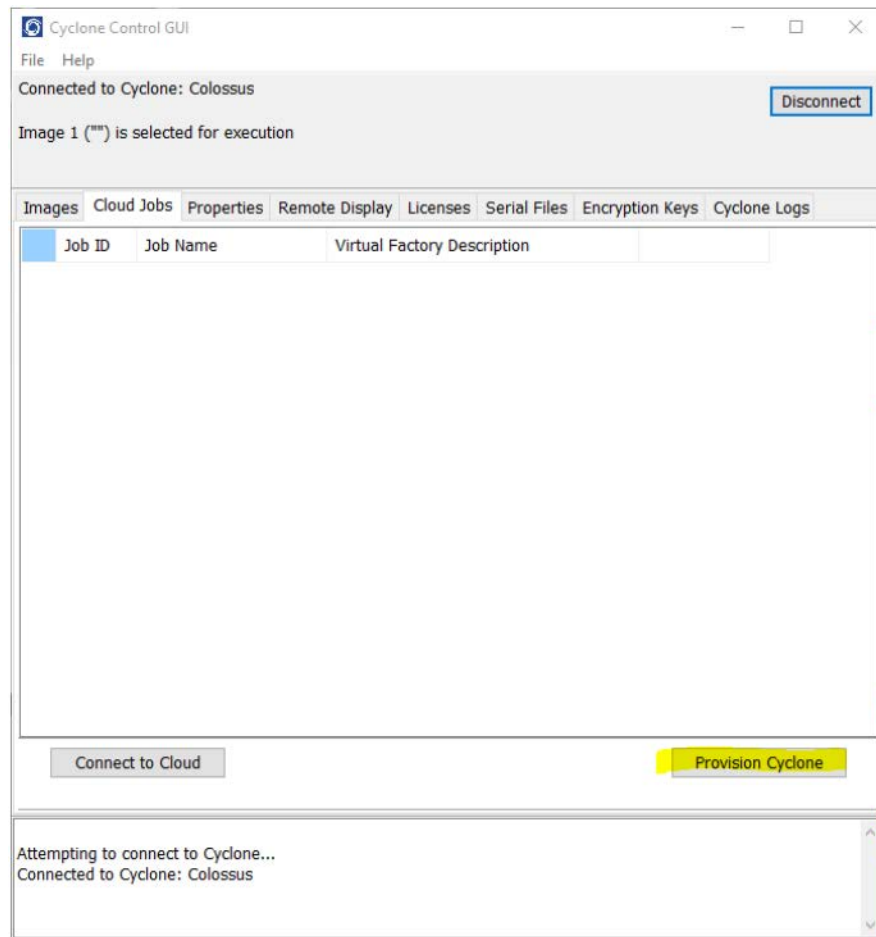
### Figure 8-12: Encryption Keys Tab

The Add Encryption Key button allows the user to browse to an ImageKey that they wish to add to the Cyclone. Encrypted (eSAP) programming images cannot be loaded onto the Cyclone unless the ImageKey that was used to encrypt them is present on the Cyclone. The transfer of an ImageKey to a Cyclone using the Cyclone Control Suite is always secured by encrypting the ImageKey with an RSA public key specific to the Cyclone that it is connected to.

The Delete Encryption Key button will delete the selected ImageKey from the Cyclone. The user should bear in mind that if the ImageKey is missing for an eSAP image, that image will be locked and cannot be programmed.

### 8.4.2.7 Cloud Jobs Tab

If the Cyclone is not provisioned for PEcloud, the button in the lower right will display “Provision Cyclone.”

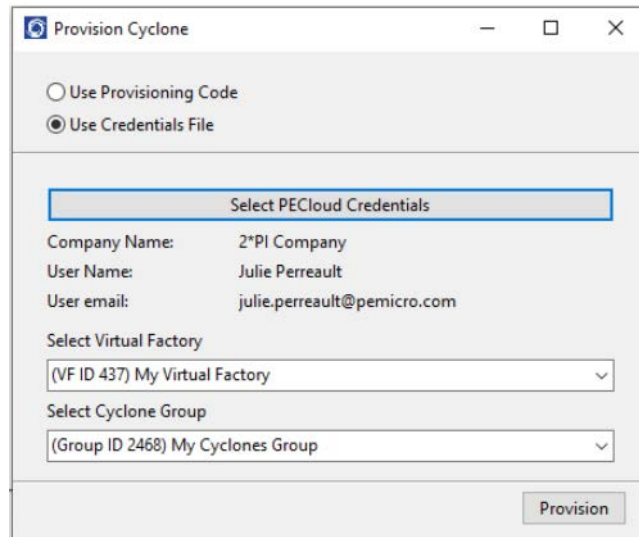


**Figure 8-13: Provision Cyclone Button**

Clicking on this button will bring up a dialog which allows the user to provision the Cyclone in one of two ways.

1. If the user has a provision code generated by the PEcloud Cyclone Group, they may select Use Provisioning Code and type in their code.
2. If the user has downloaded their PEcloud API credentials, they may provision the Cyclone using that file by selecting Use Credentials File. Once their credentials are selected the user must choose a Virtual Factory and Cyclone User Group from those provided. The newly provisioned Cyclone will join that VF/group.





The Provision Cyclone dialog box contains the following fields and controls:

- Radio buttons: ☐ Use Provisioning Code, ☒ Use Credentials File
- Section: Select PECloud Credentials
  - Company Name: 2\*PI Company
  - User Name: Julie Perreault
  - User email: julie.perreault@pemicro.com
- Section: Select Virtual Factory
  - Dropdown menu: (VF ID 437) My Virtual Factory
- Section: Select Cyclone Group
  - Dropdown menu: (Group ID 2468) My Cyclones Group
- Provision button

**Figure 8-14: Provision Cyclone Dialog**

If the Cyclone is already provisioned for a PEcloud account the user will see a listing of Jobs that are available to the Cyclone based on which Cyclone Group it is assigned to, and which Virtual Factory that Cyclone Group is assigned to.

Serial Files

Encryption Keys

Cyclone Logs

Images

Cloud Jobs

Properties

Remote Display

Licenses

Job ID	Job Name	Virtual Factory Description
1904	Blinky Bot v.2.0 First Production Run	West Factory
1903	Blinky Bot v.2.0 Test Run	West Factory

<

Refresh Jobs

Deprovision Cyclone

**Figure 8-15: Cyclone Jobs Tab**

The Cyclone may be de-provisioned from the PEcloud account by clicking the “De-provision Cyclone” button. This removes all access to the PEcloud account and any ability to program cloud Jobs, regardless of whether the Job is already loaded into memory. The Cyclone must always first obtain permission from the cloud before programming a cloud Job.

These Jobs may contain various limitations including number of successful programs, date range, and more. These are incorporated into the Job when it is deployed as part of the security settings. Any limitations are not visible to the Cyclone operator. During programming, if an error message is displayed which suggests that an unexpected limitation has been encountered, operators are encouraged to consult the PEcloud admin/manager with any questions as it pertains to their specific usage scenario.

#### 8.4.2.8 Cyclone Logs Tab

The Cyclone Logs tab can display logging of particular functions performed by the Cyclone. Contact PEMicro for more information.



**Figure 8-16: Status & Error Window**

This section of the Cyclone Control GUI displays the status of the utility as well as any errors during the connection or any of the actions performed by the utility. It'll show detailed errors on images that failed to be properly added or actions that require additional licensing.

This new status and error window increases the visibility of the user into the tasks the utility is performing as well as the issues that may arise.

## 8.5

### CPROG Scripted Programming Software

Cyclones include CPROG scripted programming software executables for each supported device family. These are located in the path PEmicro -> Cyclone -> InteractiveProgrammer from the install directory. The Cyclone must be used as the hardware interface as CPROG software included with the Cyclone has its license permanently installed on the Cyclone.

Some users who wish to perform scripted programming, especially those transitioning from PEmicro's Multilink probes, may find the CPROG software more familiar to work with than the Cyclone Control Console, which can also be used for this purpose.

Detailed operating instructions for CPROG software are beyond the scope of this manual. A separate user manual for the specific CPROG being used can be found on the PROG product page of pemicro.com.

For interactive manual programming, PROG executables are also included in the same folder.

## 9 SAP IMAGE COMPILER (SCRIPTED PROGRAMMING & IMAGE CREATION)

PEmicro's Cyclone SAP Image Compiler, or CSAP, is an essential component in the Stand-Alone Programming (SAP) image creation process. It is designed to work in tandem with the Cyclone Image Creation Utility, by running in the background, but it can also be called directly by the user. The CSAP image compiler typically takes either a .CFG file that was generated by the Image Creation Utility, or a SAPOBJ file, and uses it to locate and combine all of the components that will be included in the specific SAP image that is being created. For more about why the user would choose to use one vs. the other, please refer to **Section 9.1 - CFG Files Vs. SAPOBJ Files**.

However the user can also use a command line to submit a CFG file and various Command-Line Parameters directly to the image compiler. This allows users to write scripts that can automate the image creation/re-creation process.

In order for the user to do this, they will need to know what Command-Line Parameters are available and how they are used, and what items can/must be included in a CFG file, including Programming Commands and Configuration Commands. There is also a specific Command-Line Parameter that allows the user to easily substitute values inside a specific CFG file. This can enable a single CFG file to be used to create different SAP images.

### 9.1 CFG Files Vs. SAPOBJ Files

Users often save a configuration file (.CFG) so they can later regenerate a new SAP image from the same configuration and reference files. However, if any of the files have changed or been lost, like the binary file or the serial file, then it is not possible to recreate and image the programs with exactly the same data. The Cyclone Image Creation Utility addresses this by adding the ability to build a SAP object file (.SAPOBJ).

By checking the 'Build to SAPOBJ only' checkbox in the Image Creation Utility, a non-executable SAPOBJ object will be created that organizes everything needed for the programming image, including the programming script, programming settings, algorithm, power and communications settings, binary file - all in one place. Since all of the original information is saved into one file, if the user wants to start another production run and have the exact same programming data as a previous production, they can simply use the SAPOBJ to create an identical one. This eliminates potential issues if any of these original files are moved, deleted, or overwritten. They can load the desired SAPOBJ file, run the new Utility called the Deployment manager, add the desired restrictions and encryption and deploy a SAP image or programming job from it.

Read more at: [https://www.pemicro.com/blog/index.cfm?post\\_id=262](https://www.pemicro.com/blog/index.cfm?post_id=262)

### 9.2 Launching From the Command Line

The image compiler can be launched from the command line to create a SAP image. Below is an example of a command-line that the user might put together, along with descriptions of its various components.

#### 9.2.1 Command-Line Example

Below is an example of using the command line to launch CSAP for ARM Cortex devices. The command line specifies where to locate the .CFG file and other necessary information.

```
>csapacmpz.exe ! -INPUTCFG="C:\MyWorkspace\MyProject\
KL25Z128_script.cfg" -DEPLOYSAPFILE="C:\MyWorkspace\
MyProject\KL25Z128_image.sap" -DEPLOYDESCRIPTION="Test_KL25Z128"
```

Most command-line parameters can be used with any device, but some are specific to certain architectures or device types.

### 9.2.1.1 CSAP Executable

The user must first specify the particular CSAP executable that is compatible with their device. See the area of the example in green, which is specifically for ARM devices:

```
>csapacmpz.exe ! -INPUTCFG="C:\MyWorkspace\MyProject\
KL25Z128_script.cfg" -DEPLOYSAPFILE="C:\MyWorkspace\
MyProject\KL25Z128_image.sap" -DEPLOYDESCRIPTION="Test_KL25Z128"
```

Below is a list of which executable corresponds to which architecture/device type.

#### Target Architecture / Executable Name

**ARM-based devices (all manufacturers) :** CSAPACMPZ.exe  
**MAC71XX, MAC72XX:** CSAPARMZ.exe  
**HC(S)12(X):** CSAPBDM12Z.exe  
**ColdFire V1:** CSAPBDMCFV1Z.exe  
**ColdFire V2, V3, V4:** CSAPBDMCFZ.exe  
**MPC5xx/8xx:** CSAPBDMPPCZ.exe  
**DSC:** CSAPDSCZ.exe  
**HCS08:** CSAPHCS08Z.exe  
**HC08:** CSAPMON08Z.exe  
**MPC55XX-57XX:** CSAPPPCNEXUSZ.exe  
**RS08:** CSAPRS08Z.exe  
**S12Z:** CSAPS12ZZ.exe  
**STM8:** CSAPWIZ01.exe  
**Renesas:** CSAPWIZ00.exe  
**Infineon TriCore:** CSAPTRICORE.exe

### 9.2.2 Filename and Additional Command-Line Parameters

After specifying the executable, shown the user may also include one or more other command-line parameters. The area of the example in violet shows an example of these other parameters in the command line.

```
>csapacmpz.exe ! -INPUTCFG="C:\MyWorkspace\MyProject\
KL25Z128_script.cfg" -DEPLOYSAPFILE="C:\MyWorkspace\
MyProject\KL25Z128_image.sap" -DEPLOYDESCRIPTION="Test_KL25Z128"
```

### 9.2.3 List of Valid Command-Line Parameters

**Note:** If there are white spaces inside a parameter, the parameter should be specified with double quotes ". The command-line parameter identifier is case insensitive.

Here is the listing of valid parameters:

- [?] Use the '?' character option to cause the utility to wait and display the result of configuration in the image compiler window. If the user does not use a batch file to test error level, this provides a method to display the configuration result. This option should be the FIRST command-line option.

[hideapp]	This will cause the CSAP executable programmer to NOT display a visual presence while running, with the exception of appearing on the taskbar. (32-bit applications only)
[/logfile logfilename]	This option opens a logfile of the name "logfilename" which will cause any information which is written to the status window to also be written to this file. The "logfilename" should be a full path name such as c:\mydir\mysub-dir\mylog.log.
[paramn=s]	A command- line parameter which can modify the executing script by replacing special tags (/PARAMn). Can be used to replace any part of the script including programming commands, filenames, and parameters. Valid values of n are 0..9. s is a string which will replace any occurrence of /PARAMn in the script file. See <b>Section 9.3.4 - Using Command Line Parameters Inside a .CFG File</b> for more information and examples.

### Deprecated Parameters

These parameters have been supplanted by replacements, however they will still function.

[filename]	A configuration file containing configuration commands and comments, default = PROG.CFG. Mandatory. See -inputcfg.
[/imagefile imagefilename]	Used when the SAP file should be saved to disk instead of stored on the Cyclone. This specifies the path and filename for the SAP file. A user may later update a Cyclone with this image file. See -deploysapfile.

Brackets indicate that a parameter is optional, but note that a parameter must be chosen from the available options for both Input and Output.

### Input Paramaters

One of the following two parameters is **required** to select the input source:

[-inputcfg="script filename with full path"]	Specify a script file to be used as input
[-inputsapobj="SAPOBJ filename with full path"]	Specify a SAPOBJ file as input to create an image or job

### Output Parameters

One of the following three parameters is **required** to select the output type:

[-deploytocloud]	Deploy a Job directly to cloud
[-outputsapobj="SAPOBJ filename with full path"]	Output a SAPOBJ to the specified path
[-deploysapfile="image filename with full path"]	Generate a SAP image

### Name / Description Parameter

Set a description/name for the SAP image or Job:

[-deploydescription="description for image or job"]	Sets text as the description for a SAP image of cloud Job. This becomes the Name for a cloud Job.
---	---

## ImageKey File Parameter

Specifies a PE ImageKey file for encryption:

`[-peimagekeyfile="peimagekey file name with full path"]` Specifies a PE ImageKey file

## Programming Restriction Parameters

The following parameters can be used to add restrictions to programming:

`[-restrictdatefrom=mm/dd/yyyy]` Add restriction start date to an image or cloud job

`[-restrictdateto=mm/dd/yyyy]` Add restriction end date to an image or cloud job

`[-restrictmaxprograms=n]` Restrict maximum number of programs to positive integer *n*. Applies to SAP images or cloud Jobs.

`[-restrictmaxerrors=n]` Restrict maximum number of programming errors to positive integer *n*. Applies to SAP images only.

`[-restrictmaxattempts=n]` Restrict maximum number of programming attempts to positive integer *n*. Applies to cloud Jobs only.

## Cloud Function Parameters

The following parameters can be used with the cloud:

`[-job_vf_id=virtual factory id number]` Associates the cloud job with the specified Virtual Factory

`[-forcenewjob]` Force a Job to be deployed to the cloud. It will replace/overwrite a Job on the cloud.

`[-cloudcredentials="file name for cloud credentials with full path"]` Specifies path to cloud credentials

## Usage Examples:

To use a script file to generate a SAPOBJ file:

```
-inputcfg="C:\PEMicro\cyclone\imageCreation\ImageCreationSupportFiles\cyclone2.cfg"
-outputsapobj="test obj.sapobj"
```

To use an existing SAPOBJ file to generate an image:

```
-inputsapobj=testobj.sapobj -DEPLOYDESCRIPTION="This is a test" -deploysapfile=C:\PEMicro\cyclone\imageCreation\ImageCreationSupportFiles\test1.esap -peimagekeyfile=C:\PEMicro\cyclone\workspace\huajun_test3.peimagekey
```

To use a script file to generate a SAP image:



```
-inputcfg="C:\PEMicro\cyclone\imageCreation\ImageCreationSupportFiles\cyclone2.cfg" -
DEPLOYDESCRIPTION="This is a test" -deploysapfile=C:\PEMicro\cyclone\imageCre-
ation\ImageCreationSupportFiles\test1.esap -peimagekeyfile=C:\PEMicro\cyclone\work-
space\huajun_test3.peimagekey
```

Use an existing SAPOBJ file to generate a cloud Job directly:

```
/LOGFILE "C:\PEMicro\cyclone\imageCreation\ImageCreationSupportFiles\cyclone2.deploylog" -
INPUTSAPOBJ="C:\PEMicro\cyclone\imageCreation\ImageCreationSupportFiles\cyclone2.sap-
obj" -DEPLOYOPTIONS="C:\PEMicro\cyclone\imageCreation\ImageCreationSupport-
Files\cyclone2.deployoptions" -RESTRICTMAXPROGRAMS=1000 -
RESTRICTMAXATTEMPTS=1000 -RESTRICTDATEFROM=05/30/2023 -RESTRICTDA-
TETO=05/30/2024 -PEIMAGEKEYFILE="C:\PEMicro\cyclone\workspace\factory1.peimagekey" -
JOB_VF_ID=30 -DEPLOYDESCRIPTION="Test_KE02Z64M4_EN_PM_VC_1000" -DEPLOY-
TOCLOUD -CLOUDCREDENTIALS="C:\PEMicro\cyclone\workspace\pecloud_api_PEmi-
cro_Kevin_P_pemicro_com_2023_04_29_15_22_59.credentials"
```

### 9.3 Configuration (.CFG) File Contents

A Configuration (.CFG) file includes programming commands and the location of the binary files and programming algorithm to be used during programming. It may also include configuration commands and may refer to utilities that can augment the programming process, such as serialization, or setup information for use of a bar code scanner during programming.

**Note:** CSAP only supports serial files in JSON format.

Because the CFG file is essential to the process of creating a SAP image, the command-line used to call CSAP must always use the [-inputcfg=""] parameter to specify a .CFG file. This file will instruct the image compiler which components will be used to create the eventual SAPOBJ and where to find them, among other things. On the other hand, when creating a SAP image or cloud Job a .CFG file is not necessarily required as input.

#### 9.3.1 Sample .CFG File

A .CFG file is a pure ASCII file that includes one command per line. It will always include two main types of commands: Configuration Commands and Programming Commands. It can also include a certain type of command-line parameter that can serve as a placeholder for some of the script contents.

Below is a sample .CFG for NXP's Kinetis KL25Z128 device. Lines in the file that begin with semicolons are comment lines.

The first several lines are comments that describe some of the attributes of the programming setup. The next several lines, which begin with a colon, are Configuration Commands that are read before programming. The final several lines are the Programming Commands that will be executed during the programming process.

```
; Automatically generated configuration file
; Silicon Manufacturer is NXP
; Silicon Architecture is ARM Based (Kinetis, LPC, etc.)
;
:ALLOWOUTOFRANGE 1
:DEVICE NXP_K7x_K70FN1M0M15
:USESVD 1
:DEBUGFREQUENCY 5560
:SAPGUIVERSION 352E3737
```

```
:PROVIDEPOWER
:POWERVOLTAGE 3.0
:POWERDOWNDELAY 250
:POWERUPDELAY 250
:KEEPPOWERON 0
:CUSTOMTRIMREF 31250.00
:NEWIMAGE
:DESCRIBEIMAGE Test_K70
CM
C:\PEMicro\cyclone\supportfiles\supportFiles_ARM\NXP\K7x\freescall
_k70fn1m0m15_1x32x256k_pflash.arp
CQ
QO C:\test\nxp\armcortex\mk_x_32_pflash_dflash_m5_05A0_1FFF.s19
EN ;Erase if not Blank
PM ;Program Module
VC ;Verify Checksum
```

### 9.3.2 Configuration Commands

Configuration Commands are commands that will be executed at startup, before the programming process. You can see configuration commands used inside a sample .CFG file above in **Section 9.3.1 - Sample .CFG File**. They listed are in the middle of the file. They always begin with a colon. A listing of valid Configuration Commands and their formats is included below.

#### 9.3.2.1 Target Power Related Configuration Commands

##### 9.3.2.1.1 :PROVIDEPOWER n

*Processors: All (EXCEPT MON08)*

Determines whether the Cyclone should provide power to the target. (This is the same as legacy option :USEPRORELAYS n).

**Note:** Not all hardware interfaces support this command. Valid values of n are:

- 0 : Cyclone does NOT provide power to target. (default)
- 1 : Enable Cyclone to provide power to target.

##### 9.3.2.1.2 :POWERVOLTAGE n.n

*Processors: All*

Use this command if the Cyclone is providing/switching power to the target, otherwise omit this command. Specifies the target voltage as a real number. Acceptable range is from 1.6V - 5.0V.

:POWERVOLTAGE 3.3      Specifies target voltage as 3.3V

##### 9.3.2.1.3 :KEEPPOWERON n

*Processors: All*

Determines whether power provided to the target should be turned off when the application terminates. NOTE: Not all hardware interfaces support this command. Valid values of n are:

- 0 : Turn power off upon exit (default)
- 1 : Keep power on upon exit

##### 9.3.2.1.4 :POWERDOWNDELAY n

*Processors: All*

Amount of time to delay when the power to the target is turned off for the target's power supply to drop to below 0.1v. n is the time in milliseconds.

#### 9.3.2.1.5 :POWERUPDELAY n

*Processors: All*

Amount of time to delay when the power to the target is turned on OR the target is reset, and before the software attempts to talk to the target. This time can be a combination of power on time and reset time (especially if a reset driver is used). n is the time in milliseconds.

### 9.3.2.2 Trim Related (If supported) Configuration Commands

#### 9.3.2.2.1 :CUSTOMTRIMREF nnnnnnnn.nn

*Processors: All*

Desired internal reference clock frequency for the "PT; Program Trim" command. This frequency overrides the default internal reference clock frequency. Valid values for "nnnnnnnn.nn" depend on the particular device being programmed. Please refer to the electrical specifications of your device for valid internal reference frequency clock range.

Where:

**nnnnnnnn.nn** : Frequency in Hertz with two decimal places

### 9.3.2.3 Information Processing Configuration Commands

#### 9.3.2.3.1 :CLEARSTATUS n

*Processors: All*

Specifies the amount of time after programming, in milliseconds, before the Success indicator (LED) will be turned off.

#### 9.3.2.3.2 :DESCRIBEIMAGE *string*

*Processors: All*

***string*** This is a string that describes the SAP image. Will overwrite the command-line parameter "imagecontent" if both are present in the .CFG file.

Example:

:DESCRIBEIMAGE KL25Z128 TEST IMAGE

#### 9.3.2.3.3 :ALLOWOUTOFRANGE n

*Processors: All*

Sets whether programming will continue when data is out of range.

:ALLOWOUTOFRANGE 1      Allows programming to continue when some data is out of range (addresses not in module). Out of range data is ignored.

:ALLOWOUTOFRANGE 0      Requires all programming data to be in range of the module. Out of range data causes an error on image creation.

### 9.3.2.4 Image Launch Settings Configuration Commands

#### 9.3.2.4.1 :BARFILE *barfile*

*Processors: All*

Specifies that a .BAR file will be used during programming (programming initiated by bar code scanner).

***barfile***      Indicates the path and filename of the .bar file.

Example:

:BARFILE C:\PEMicro\cyclone\imageCreation\barcodeTest.bar

### 9.3.2.5 Connection Related Configuration Commands (ARM)

#### 9.3.2.5.1 :DEVICE *string*

*Processors: ARM, DSC, MAC7xxx, TriCore*

For ARM and TriCore devices, this is a mandatory parameter that specifies the device. It is required because the debug protocol changes between manufacturers and sometimes also between families or devices.

Where:

***string***: represents the device and follows the “VENDOR\_FAMILY\_DEVICE” format.

For ARM and TriCore devices, the easiest way to obtain the device ***string*** is from the Device Selection dialog in the Cyclone Image Configuration Utility software. In the PEMICRO Connection Manager, click “Select New Device” to open the Device Selection dialog. Expand the device tree to find your device, then right-click and select “Copy Device String to Clipboard.”

#### 9.3.2.5.2 :DEBUGFREQUENCY *n*

*Processors: ARM, ColdFire, PPC, MAC7xxx, TriCore*

Specifies the communications frequency with the target device.

*n* Frequency in Kilohertz.

#### 9.3.2.5.3 :USESVD *n*

*Processors: ARM*

Allows the user to specify SWD (single wire debug) mode instead of JTAG mode.

**0** Specifies that JTAG mode will be used during communications (default).

**1** Specifies that SWD (single wire debug) mode will be used during communications.

#### 9.3.2.5.4 :JTAGTAPNUM *n*

*Processors: ARM*

The JTAG Tap Number is the index of the target device in the daisy chain. The first device connected to TDI is index 0, the next is index 1, and so on. The index of the last device connected to the TDO of the debugger is the total number of devices in the chain minus 1.

**Note:** This command is mandatory for JTAG daisy chain configurations. It is also important to specify JTAG communications using the :USESVD 0 command/value.

*n* Specifies the index number of a device in the daisy chain.

For more information on how to program or debug with a daisy chain setup, read:

[http://www.pemicro.com/blog/index.cfm?post\\_id=136](http://www.pemicro.com/blog/index.cfm?post_id=136)

#### 9.3.2.5.5 :JTAGPREIR *n*

*Processors: ARM*

Every JTAG device has an IR register that is a certain width in bits. In the daisy chain the number of Pre-IR bits is the sum of all of the IR registers between your device and the TDO pin.

**Note:** This command is mandatory for JTAG daisy chain configurations. It is also important to specify JTAG communications using the :USESVD 0 command/value.

*n* Specifies the total length of IR registers following the target device. The first device in the daisy chain is index 0.

For more information on how to program or debug with a daisy chain setup, read:

[http://www.pemicro.com/blog/index.cfm?post\\_id=136](http://www.pemicro.com/blog/index.cfm?post_id=136)

#### 9.3.2.5.6 :RSTLOWPOSTSAP

*Processors: All*

Drives the RESET signal LOW before and after SAP operations.

### 9.3.2.6 Connection Related (S08, S12, ColdFire V1, RS08, S12Z) Configuration Commands

#### 9.3.2.6.1 :DRIVEBKGDLOW n

*Processors: S08, S12, CFV1, S12Z* **Note: Not RS08.**

By default, the BKGD signal **is driven low** before and after programming operations are complete.

- 0 Do NOT drive BGND low before and after programming operations.
- non-zero or missing Drive BGND signal low before and after programming operations.

Example:

:DRIVEBKGDLOW 0 Does NOT drive the BKGD signal LOW after operations are complete.

#### 9.3.2.6.2 :RSTLOWPOSTSAP

*Processors: All*

Drives the RESET signal LOW before and after SAP operations.

### 9.3.2.7 Connection Related Configuration Commands (ColdFire V2, V3, V4)

#### 9.3.2.7.1 :USEPST SIGNALS n

*Processors: ColdFire*

Specifies whether to use PST signals during debug.

- 0 Do not use PST signals.
- 1 Use PST signals.

#### 9.3.2.7.2 :RSTLOWPOSTSAP

*Processors: All*

Drives the RESET signal LOW before and after SAP operations.

### 9.3.2.8 Connection Related (MPC5xxx, SPC5xxx) Processors

#### 9.3.2.8.1 :DEBUGFREQUENCY n

*Processors: ARM, ColdFire, PPC, MAC7xxx, DSC, PPCNexus*

Specifies the communications frequency with the target device.

n Frequency in Kilohertz.

#### 9.3.2.8.2 :UNCENSOR n

*Processors: PPCNexus, TriCore*

This parameter should be used if 64-bit and 256-bit censorship passwords are needed to bypass security.

**Note:** The ASCII version of the password must have each long word separated by a dash.

**n** Password represented as a hexadecimal value, with no symbols or spaces

#### 9.3.2.8.3 :RSTLOWPOSTSAP

*Processors: All*

Drives the RESET signal LOW before and after SAP operations.



### 9.3.2.9 Connection Related - DSC Processors

#### 9.3.2.9.1 :DEVICE string

*Processors: ARM, DSC, MAC7xxx, TriCore*

Describes the device being programmed.

**string** Describes the device being programmed.

#### 9.3.2.9.2 :RSTLOWPOSTSAP

*Processors: DSC, STM8*

Drives the RESET signal LOW before and after SAP operations.

### 9.3.2.10 Connection Related (Infineon TriCore TC1xx, TC2xx, TC3xx) Processors

#### 9.3.2.10.1 :DEVICE string

*Processors: ARM, DSC, MAC7xxx, TriCore*

For ARM and TriCore devices, this is a mandatory parameter that specifies the device. It is required because the debug protocol changes between manufacturers and sometimes also between families or devices.

Where:

**string:** represents the device and follows the "VENDOR\_FAMILY\_DEVICE" format.

For ARM and TriCore devices, the easiest way to obtain the device **string** is from the Device Selection dialog in the Cyclone Image Configuration Utility software. In the PEMICRO Connection Manager, click "Select New Device" to open the Device Selection dialog. Expand the device tree to find your device, then right-click and select "Copy Device String to Clipboard."

#### 9.3.2.10.2 :DEBUGFREQUENCY n

*Processors: ARM, ColdFire, PPC, MAC7xxx, DSC, PPCNexus, TriCore*

Specifies the communications frequency with the target device.

n Frequency in Kilohertz.

#### 9.3.2.10.3 :RSTLOWPOSTSAP

*Processors: All*

Drives the RESET signal LOW before and after SAP operations.

#### 9.3.2.10.4 :UNCENSOR n

*Processors: PPCNexus, TriCore*

This parameter should be used if 64-bit and 256-bit censorship passwords are needed to bypass security.

**Note:** The ASCII version of the password must have each long word separated by a dash.

**n** Password represented as a hexadecimal value, with no symbols or spaces

### 9.3.2.11 Connection Related - MON08 Processors

When using MON08 devices, the user must specify :POWERVOLTAGE and :POWERUPDELAY & :POWERDOWNDelay.

#### 9.3.2.11.1 :DEVICECLOCK n

For Class 5, 6, 7, and 8 devices. Controls whether the Cyclone should drive a clock to the target or whether the PE micro interface should tristate its clock output. Valid values of n are:

0 : Clock driven by Cyclone. Must use :OUTPUTCLOCK to specify.



1 : Target self-clocked, Cyclone clock output disabled

#### 9.3.2.11.2 :OUTPUTCLOCK n

Specifies the clock when :DEVICECLOCK is set to 0 (driven by Cyclone). Valid values of n are:

0 : 4.9152 MHz  
1 : 9.8304 MHz

#### 9.3.2.11.3 :CLOCKDIVIDER n

For Class 5, 6, 7, and 8 devices. Often one of the port pins of the target processor controls the ratio of the BUS clock to the External clock. Valid values of n are:

0 : Divide by 2 (usually and if applicable)  
1 : Divide by 4 (usually and if applicable)

#### 9.3.2.11.4 :BAUD n

Sets the baud rate to n. Serial port connection only If :BAUD is specified, include :FORCEPASS followed by :SECURITYCODE.

#### 9.3.2.11.5 :FORCEPASS

Specifies that security should be passed on startup of the software instead of waiting for an EM (Erase Module) command. The :SECURITY code command must also be provided.

#### 9.3.2.11.6 :SECURITYCODE hh hh hh hh hh hh hh hh

Specifies the 8 bytes of security code to use at startup which correspond to the addresses \$FFF6-\$FFFD of the target HC08 device. The parameter for this is a string containing 8 bytes of data in HEX separated by white spaces.

#### 9.3.2.11.7 :DEVICETYPE string

Specifies the target device family. As an example, the device type for a 68HC908KX8 would be KX. The allowed device type values are:

AB,AP,AS,AT,AZ,BD,EY,GP,GR,GR4/8,GT,GZ,JB12,JB16, JB1/8,JG,JK,JL,JR,JW,KX,LB,LD, LJ,LK,LT,LV,MR4/8,MR16/32,QB,QC,QL,QT,QY,RF,RK,SR

### 9.3.2.12 Connection Related - STMicroelectronics' STM8 Processors

#### 9.3.2.12.1 :ARCHTYPE n

*Processors: STM8*

Specifies the STM8 family via the numeral n, where n indicates the following:

161 = STM8S/STM8A  
162 = STM8L101X  
163 = STM8L15X  
164 = STM8L16X  
165 = STM8AF  
166 = STM8AL

#### 9.3.2.12.2 :COMMSMODE 0

*Processors: STM8*

Indicates that communications speed should be controlled automatically.

#### 9.3.2.12.3 :FORCEPASS

*Processors: STM8*

If this command is present, read-out protection will be ignored, i.e. the target will be unsecured and the programming process will continue.

If this command is missing, read-out protection will NOT be ignored. If the device is protected the programming process will not proceed.

#### 9.3.2.12.4 :RSTLOWPOSTSAP

*Processors: All*

Drives the RESET signal LOW before and after SAP operations.

### 9.3.2.13 Connection Related - Renesas Processors

#### 9.3.2.13.1 :ARCHTYPE n

*Processors: Renesas*

Specifies the Renesas family via the numeral n, where n indicates the following:

- 177 = R8C
- 178 = H8 or HS8/Tiny
- 179 = M16C
- 180 = M16C/80
- 181 = M32C
- 182 = RL78
- 183 = RX
- 184 = RX63T
- 185 = RH850

#### 9.3.2.13.2 :FORCEPASS

*Processors: Renesas*

If this command is present, read-out protection will be ignored, i.e. the target will be unsecured and the programming process will continue.

If this command is missing, read-out protection will NOT be ignored. If the device is protected the programming process will not proceed.

#### 9.3.2.13.3 :RSTLOWPOSTSAP

*Processors: All*

Drives the RESET signal LOW before and after SAP operations.

### 9.3.3 Programming Commands

Programming Commands are the commands that will be executed during the programming process. These are the main commands that will manipulate and verify data on your device. A list of programming commands and their formats is included below.

See **Section 9.3.1 - Sample .CFG File** to view programming commands within a CFG file. The example programming commands are at the bottom of the file.

**Note:** The command parameter formats *starting\_addr*, *ending\_addr*, *base\_addr*, and *byte*, *word* are hexadecimal by default.

**BM** Blank check module.

**BR** *starting\_addr ending\_addr* Blank check range.

**CM** Choose module (algorithm) file. Note: Certain modules may require a base address to be specified

**CQ** Clears queue of object files to be programmed (see QO command)

**CU** either

**EB *starting\_addr ending\_addr*** Erase byte range.

**EW *starting\_addr ending\_addr*** Erase word range.

**EM** Erase module.

**EN** Blank check and erase

**GO** Starts device running. Can be used as final command if you want the device to run for testing. Should be immediately preceded by an 'RE' command.

**PB *starting\_addr byte ... byte*** Program bytes.

**PF *feature\_ID starting\_addr*** Program feature data. *feature\_ID* must be: *datestr*, *datetimestr*, *barcodestr*, or *runtestdata*. See **Section 6.2.1.6.12 - Program Feature Data**.

**PW *starting\_addr word ... word*** Program words.

**PM** Program module.

**PT [*address*]** Program trim (Devices with trim only. Address is only necessary for devices that have no dedicated non-volatile trim register)

**RE** Reset chip.

**QO *path*** Specify binary data file (S19/Elf/Hex). *path* indicates file path to the binary. Path is added to a queue of object files to be programmed. See also: CQ command.

**VC** Verify the programmed device using a checksum

**VM *starting\_addr ending\_addr*** Verify module.

**VR *starting\_addr ending\_addr*** Verify range.

**VV *type*** Verify module CRC. Type is CRC8 or CRC16.

**DE *timeinms*** - Delays "*timeinms*" milliseconds

### 9.3.4 Using Command Line Parameters Inside a .CFG File

The user may wish to make their .CFG files more versatile by inserting one or more placeholders into the script, and then specifying the values for those placeholders later, on the command line, when calling the CSAP executable that will reference that script.

The command-line parameter */PARAMn=s* can be used to insert text into a .CFG file. It can replace any part of the script including programming commands, filenames, and parameters. *n* is a numeral from 0..9, which allows you to use multiple versions of this command line parameter in a .CFG file. *s* represents a string which will replace any occurrence of */PARAMn* in the script file.

As an example, the following section of a .CFG script features three instances of */PARAMn=s*: */PARAM1*, */PARAM2*, and */PARAM3*:

RE ;Reset the MCU

CM */PARAM1* ;Choose Flash Module

EM ;Erase the module

BM ;Blank Check the module

CQ ; Clear Queue of object files to be programmed

QO */PARAM2* ;Specify the S19 to use, it is added to the Queue of Object Files

PM ;Program the module with the S19

*/PARAM3* ;Verify the module again

When the user calls the CSAP executable that will reference this CFG file they will need to specify

the values of these parameters on the command line. See the example below, where the first of these parameters is used to specify a programming algorithm (.SRP), the second an .S19 file, and the third a programming command (VM).

```
CSAPACMPZ
/PARAM1=C:\PEMICRO\Freescale_MK40X256_PFlash_DFlash.ARP
"/PARAM2=C:\PEMICRO\EXAMPLE FILES\TEST.S19"
/PARAM3=VM
```

**Note:** Notice that /PARAM2 is enclosed in double quotation marks. This is because the parameter has a space in its value (Example Files). The surrounding double quotation marks are required in this situation, in order to indicate to Windows that it is a single parameter.

The complete example command line would be as below (note that this is one **continuous** line; no line breaks):

```
C:\PROJECT\CSAPACMPZ -INPUTCFG="C:\PROJECT\GENERIC.CFG" /
PARAM1=C:\PEMICRO\Freescale_MK40X256_PFlash_DFlash.ARP /
PARAM2=C:\PEMICRO\EXAMPLE FILES\TEST.S19" /PARAM3=VM
```

### 9.3.5 Sample Batch File

Here is an example of how to call a command-line programmer and test its error code return in a simple batch file. Sample batch files are given for Windows NT/2000/XP/Vista/7/8/10/11.

#### 9.3.5.1 Windows NT/2000/XP/Vista/7/8/10/11:

```
C:\PEMicro\CYCLONE\IMAGECREATION\IMAGECREATIONSUPPORTFILES\
CSAPACMPZ.EXE -INPUTCFG="C:\PROJECT\ENGINE.CFG"
PORT=USB1
if errorlevel 1 goto bad
goto good
:bad
ECHO BAD BAD BAD BAD BAD BAD BAD BAD
:good
ECHO done
```

**Note:** Path names of files that are relative to the CSAP executable can also be used.

## 9.4 CSAP Error Returns

An Error code is returned by the Image Compiler so that a script file or an application launching the Image Compiler can check for it. The error codes used are:

- 0 - Program completed with no errors.
- 1 - Canceled by user.
- 2 - Error reading S record file.
- 3 - Verify error.
- 4 - Verify canceled by user.
- 5 - S record file is not selected.
- 6 - Starting address is not in module.

- 7 - Ending address is not in module or is less than starting address.
- 8 - Unable to open file for uploading.
- 9 - File write error during upload.
- 10 - Upload canceled by user.
- 11 - Error opening algorithm file.
- 12 - Error reading algorithm file.
- 13 - Device did not initialize.
- 14 - Error loading .PCP file.
- 15 - Error enabling module just selected.
- 16 - Specified S record file not found.
- 17 - Insufficient buffer space specified by .PCP to hold a file S-record.
- 18 - Error during programming.
- 19 - Start address does not point into module.
- 20 - Error during last byte programming.
- 21 - Programming address no longer in module.
- 22 - Start address is not on an aligned word boundary.
- 23 - Error during last word programming.
- 24 - Module could not be erased.
- 25 - Module word not erased.
- 26 - Selected algorithm file does not implement byte checking.
- 27 - Module byte not erased.
- 28 - Word erase starting address must be even.
- 29 - Word erase ending address must be even.
- 30 - User parameter is not in the range.
- 31 - Error during algorithm-specified function.
- 32 - Specified port is not available or error opening port.
- 33 - Command is inactive for this .PCP file.
- 34 - Cannot enter background mode. Check connections.
- 35 - Not able to access processor. Try a software reset.
- 36 - Invalid algorithm file.
- 37 - Not able to access processor RAM. Try a software reset.
- 38 - Initialization canceled by user.
- 39 - Error converting hexadecimal command number.
- 40 - Configuration file not specified and file prog.cfg does not exist.
- 41 - Algorithm file does not exist.
- 42 - Error in io\_delay number on command line.
- 43 - Invalid command line parameter.
- 44 - Error specifying decimal delay in milliseconds.
- 47 - Error in script file.
- 49 - Cable not detected
- 50 - S-Record file does not contain valid data.
- 51 - Checksum Verification failure - S-record data does not match MCU memory.
- 52 - Sorting must be enabled to verify flash checksum.
- 53 - S-Records not all in range of module. (see "v" command line parameter)

- 54 - Error detected in settings on command line for port/interface
- 58 - Device requires a power cycle.
- 60 - Error calculating device CRC value
- 61 - Error - Device CRC does not match value given
- 70 - Error - CSAP is already running
- 71 - Error - Must specify both the INTERFACE and PORT on the command line
- 72 – The selected target processor is not supported by the current hardware interface.



## 10 PEcloud - SECURE MANAGEMENT OF REMOTE PROGRAMMING

### 10.1 Introduction

PEcloud is a cloud-based programming management service that enables customers to monitor usage and have added control over their Cyclone production programming, in real-time, anywhere in the world.

Cyclones execute programming of images which contain all of the necessary data and instructions to flash program a target's memory. PEcloud introduces a new type of encrypted programming image called a Job. A Job is a programming image that uses a secure connection from the Cyclone to the PEcloud platform to provide additional control, features, and visibility to the user.

Users will upload programming Jobs to PEcloud. Its online interface then allows the users to manage these Jobs in their own Virtual Factories (VFs). The user organizes each VF space as they see fit in order to connect specific Cyclones, programming Jobs, and users together logistically. This allows for unprecedented visibility of, and control over, programming operations. Jobs being used in manufacturing can be paused, deleted, or updated, and programming logs inspected.

PEmicro uses AWS, which is respected for its uptime and performance, to help enable PEcloud. There is currently no cost for using PEcloud.

### 10.2 Creating a PECloud Account

Users will need to create an online account at [cloud.pemicro.com](http://cloud.pemicro.com). This account is distinct from a standard pemicro.com account; they are currently not connected in any way.



  
v594

Create a new PECloud account

Company Name

If you do not provide one we will use your first and last names as your company name

Admin First Name

Admin Last Name

Admin Email

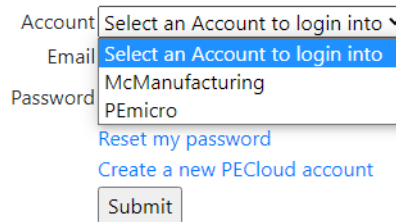
Admin Password

Invitation Code

I agree to the Terms Of Service ☐

**Figure 10-1: PEcloud - Create Account**

Note that later when a user is logging into PEcloud it is possible that they will have multiple roles as a user in various accounts. Each permission is managed separately by their respective admin, and the user will choose the account they wish to access using a drop down box that appears above their login credentials:



**Figure 10-2: User Roles in Multiple Accounts: Login**

Once logged into an account they have created, or one to which they have been invited, the user can always see who the administrators are for the account by choosing “View Account Admins” from their account actions drop-down menu - see **Section 10.7 - PEcloud Account Actions**.

### 10.2.1 Additional Help

By clicking on “Help Center” on the left side of the PEcloud interface the user can quickly access the following resources:

1. Getting Started Guide
2. PEcloud Chapter of the Cyclone User Manual
3. Blog post overview of PEcloud
4. Direct download link for the Cyclone LC/Cyclone FX software installer

The URL for the Help Center is [https://cloud.pemicro.com/help\\_center/index.cfm](https://cloud.pemicro.com/help_center/index.cfm)

For an overview of Cyclone setup for PEcloud, please visit:

[https://www.pemicro.com/blog/index.cfm?pot\\_id=272](https://www.pemicro.com/blog/index.cfm?pot_id=272)

## 10.3 PEcloud Requirements

In order to operate one or more Cyclones successfully with the PEcloud service, the following requirements need to be met.

### 10.3.1 Cyclone Part Number

The following Cyclone part numbers can be used with PEcloud.

- CYCLONE-LC-ARM, any Rev.
- CYCLONE-LC-UNIV, any Rev.
- CYCLONE-FX-ARM, any Rev.
- CYCLONE-FX-UNIV, any Rev.

### 10.3.2 ProCryption Security (Job Encryption)

Currently PEMicro will provide ProCryption Security features to any Cyclone LC models using PEcloud that do not have this license, as encryption is a requirement to meet PEcloud’s security standards.

### 10.3.3 ImageKey Provisioning

In order to securely encrypt programming Images and Jobs, PEMicro’s ProCryption Security enables the user to create a personal PE ImageKey and select it for the encryption process when creating Images or Jobs in the Cyclone Image Creation Utility. A Cyclone will be unable to load or read this file unless it contains a copy of the same PE ImageKey.

**Note:** For security reasons, users will want to have physical possession of the Cyclones when they provision them with their personal ImageKey. So an early step in the setup process is often to use the Cyclone Image creation Utility to create a PE ImageKey and then load it onto Cyclones while they are still under the user’s control, before then sending them to manufacturing locations. Please

refer to **Section 6.5.3.1 - Encryption (ProCryption Security)** for detailed instructions regarding PE ImageKeys.

#### 10.3.4 PECloud Provisioning

Cyclones must also be specifically provisioned for PEcloud with a separate code. This can be done in a few ways.

##### 10.3.4.1 Manual Entry Into Cyclone

The user can generate a code in the PEcloud online interface then enter it manually into the Cyclone. For more information please refer to **Section 5.2.4.2.1 - Provision/De-Provision Cyclone**.

##### 10.3.4.2 Via Cyclone Control GUI

Provisioning can also be done from the Cyclone Control GUI - Cloud Jobs tab. This options allows provisioning via either the provisioning code generated by PEcloud or via the PEcloud API credentials file that the user may download. Please refer to **Section 8.4.2.7 - Cloud Jobs Tab**.

#### 10.3.5 PEcloud API Credentials

The user will need to download and save their cloud API credentials. This is needed so that PEmicro utilities such as the Cyclone Image Creation Utility can securely deploy programming Jobs to the user's PEcloud account. The user will need to select their credentials when preparing a Job for deployment. These credentials should be kept safe. The creation process is very simple, as detailed in **Section 10.7 - PEcloud Account Actions**.

#### 10.3.6 Scripted CSAP Job Deployment

Please see **CHAPTER 9 - SAP IMAGE COMPILER (SCRIPTED PROGRAMMING & IMAGE CREATION)** for descriptions of the command-line parameters and other commands used to deploy cloud Jobs via CSAP.

#### 10.3.7 Naming Conventions

Users should note that Virtual Factory names, Cyclone Group names, User Group names, and Role names should be unique within a PEcloud account and may not be duplicated.

### 10.4 PEcloud Workflow Overview

Once the user's PEcloud Cyclones are properly provisioned and online at the manufacturing location they can be used to perform programming Jobs that are managed with the PEcloud service. The user can create and configure programming Jobs locally and deploy them, via PEcloud, to one or more remote Cyclones. These Jobs can include restrictions which limit the programming of a Job by date range, or number of program attempts, or number of program successes.

The Jobs are deployed by the user to a particular Virtual Factory (VF) in their PEcloud account; this is specified by the user in the deployment process. Virtual Factories are the way that PEcloud organizes jobs. When the user first logs in to their PEcloud account they will see the screen for a pre-made VF called "My Virtual Factory" This factory can be re-named, and have Jobs assigned to it. In order to accomplish those programming Jobs it will also need a Cyclone Group assigned to it.

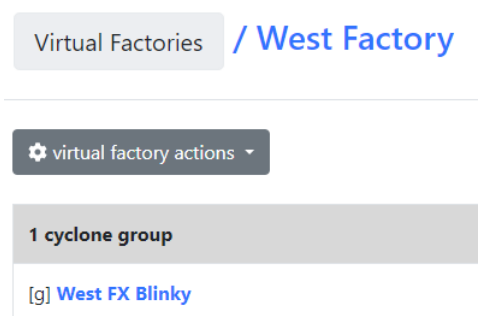
When viewing a specific VF the user is presented with a list of Jobs that have been assigned to that factory; below is an example that shows a VF with two Jobs assigned - one Job that is Running (in progress), and one Job that is Finished (programming tasks complete, or the user has simply marked it as Finished).

2 jobs	status	actions
Job Number: <a href="#">1904</a> Job Name: Blinky Bot v.2.0 First Production Run Creation Date/Time: 2023-04-09 18:22:59.0 User Creator: Keith McNeil Key name: KMDocKey01 Key ID: pei-30825DD1-77169A83-641C5995	<b>Running</b> last job status update: '2023-04-09 18:23:01' UTC last program activity: '2023-04-09 18:23:01' UTC  date restrictions: 2023-Apr-08 to 2024-May-08 UTC  program attempts: 0 / 500 max. program successes: 0 / 500 max. program failures: 0	<input type="button" value="job actions"/>
Job Number: <a href="#">1903</a> Job Name: Blinky Bot v.2.0 Test Run Creation Date/Time: 2023-04-09 17:10:04.0 User Creator: Keith McNeil Key name: KMDocKey01 Key ID: pei-30825DD1-77169A83-641C5995	<b>Completed</b> last job status update: '2023-04-09 17:10:06' UTC last program activity: '2023-04-10 13:09:35' UTC  date restrictions: none  program attempts: 5 / 5 max. program successes: 5 / unlimited program failures: 0	<input type="button" value="job actions"/>

**Figure 10-3: Jobs List (View of a Virtual Factory)**

**Note:** For any date restrictions applied to a Job, the time for the Start date is considered to be 00:00:00 UTC of the Start date, and the time for the End date is considered to be 23:59:59 UTC of the End date.

For any Jobs that the user sends to a Virtual Factory, the programming of those Jobs will be accomplished by the Cyclones in any Cyclone Groups that the user has assigned to that VF. This information is displayed at the top left under the factory actions:



**Figure 10-4: Cyclone Group Assigned to Virtual Factory**

Once a VF has been empowered by a Cyclone Group to accomplish the actual programming, everything should be in place for the user to take advantage of all the tools that PEcloud offers to get instant feedback about their Jobs' progress, pause and resume Jobs if necessary, assign additional users with varying permissions, view logged programming data, and much more.

## 10.5 Preparing and Using PEcloud

In order to get this process started, the user will need to prepare their PEcloud account and organize it in the way that they see fit. The main organizational feature of PEcloud is the Virtual Factory (and that is also the place to access other necessary components, such as Cyclone Groups, since these are specific to a single VF). For an overview of PEcloud setup, please see: [https://www.pemicro.com/blog/index.cfm?post\\_id=272](https://www.pemicro.com/blog/index.cfm?post_id=272)

### 10.5.1 Virtual Factories

A user who first logs in to PEcloud will find that it includes one Virtual Factory by default, called "My Virtual Factory." Virtual Factories are the main organizational areas of PEcloud. A user with three remote manufacturing facilities might wish to create three Virtual Factories, one for each real-world location, but they can be organized in any way that suits the user.

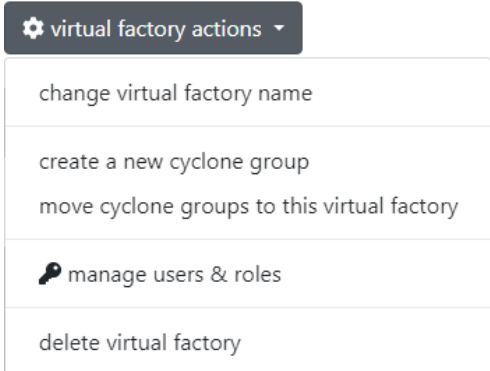
Jobs that are deployed to a VF will be displayed in a list that includes relevant Job data. More

detail on the information that is displayed is featured in **Section 10.5.3 - Jobs Display**.

In order to complete programming jobs that are deployed to a Virtual Factory the user will need to assign one or more Cyclone Groups to that Virtual Factory. The tools to manage a Virtual Factory are located near the top of each VF page.

### 10.5.1.1 Virtual Factory Actions

Clicking on Virtual Factory Actions offers the following actions:



**Figure 10-5: Virtual Factory Actions List**

The user may:

#### 10.5.1.1.1 Change Virtual Factory Name

Allows the user to edit the name of the current VF.

#### 10.5.1.1.2 Create A New Cyclone Group

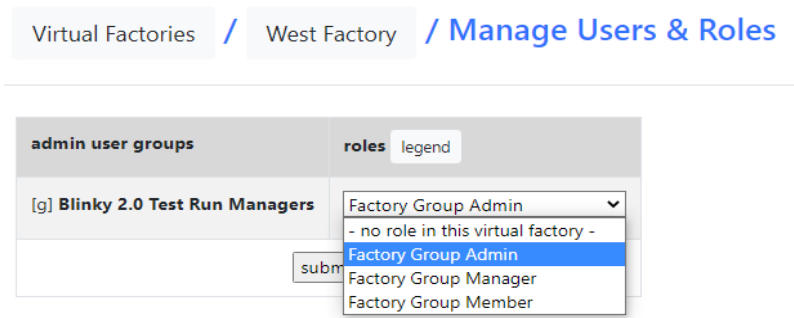
Creates a new, empty Cyclone Group which the user will name. It will be assigned to the current VF by default. The user can then use Cyclone Group actions to provision new Cyclones, or move a Cyclone from another Group to this Group. See **Section 10.5.2.1 - Cyclone Group Actions**.

#### 10.5.1.1.3 Move Cyclone Groups To This Virtual Factory

Move an existing Cyclone Group that is currently assigned to another VF to the current VF.

#### 10.5.1.1.4 Manage Users & Roles

Displays the User Groups that are associated with the current VF. Admin and Manager level users, and any other users granted the appropriate permission, may adjust the roles of these Users Groups with regards to the current VF.



**Figure 10-6: Assign Role**

Clicking on "legend" will display the actions permitted by each role, including any custom-created roles.

**Note:** Unlike User Permissions, which are assigned on a user by user basis, Virtual Factory Roles (and their respective Permissions) are assigned on a **user group by user group** basis. Care should be taken when organizing User Groups to make sure that all of the members of a User Group are eligible for the same VF Permission level. A user who requires a different VF Role than the rest of the group should be removed and placed in a separate user group, and the appropriate role assigned to that group.

Factory Group Admin
Create/Delete/Modify/Move Cyclone Groups
Move activated Cyclones to my Cyclone Group
View Cyclone Groups
Provision/De-Provisioning Cyclones
Add/Remove User Groups & set User Group Roles
Change existing User Group Roles
Create and Remove Jobs
Pause and Continue Jobs
View Jobs
View Virtual Factory
Factory Group Manager
Move activated Cyclones to my Cyclone Group
View Cyclone Groups
Change existing User Group Roles
Create and Remove Jobs
Pause and Continue Jobs
View Jobs
View Virtual Factory
Factory Group Member
View Cyclone Groups
View Jobs
View Virtual Factory

**Figure 10-7: User Roles - Permitted Actions (Legend)**

Note that admins and users with the appropriate permission may create new roles, each with a customized set of permissions, via the Role Manager. Please refer to **Section 10.7 - PEcloud Account Actions**.

#### 10.5.1.1.5 Delete Virtual Factory

Delete the current VF **and any Cyclone Groups currently assigned to that VF**. The user can first move Cyclone Groups to a different VF if they wish to preserve the Cyclone Groups.

### 10.5.2 Cyclone Groups

Cyclone Groups are used to specify the particular Cyclones that will accomplish the programming for a Virtual Factory, so without a VF's Cyclone Group no Jobs will be accomplished. A user who

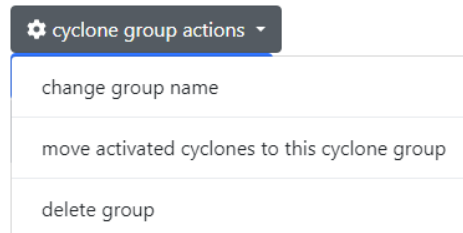


first logs in to PEcloud will find that a Cyclone Group called “My Cyclone Group” has been created by default, inside of the default VF, called “My Virtual Factory.” For a newly created VF the user will need to create a Cyclone Group from within that VF and provision new Cyclones as needed.

**Note:** Each individual Cyclone can be assigned to only one Cyclone Group, and each Cyclone Group can be assigned to only one Virtual Factory. A Cyclone Group is always assigned to a VF, and because of this Cyclone Groups are accessed through their VF screen.

### 10.5.2.1 Cyclone Group Actions

Clicking on Cyclone Group Actions offers the following actions:



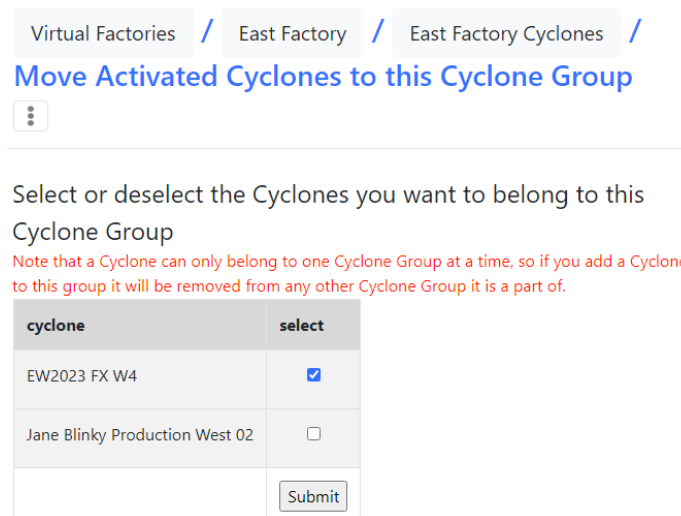
**Figure 10-8: Cyclone Group Actions**

#### 10.5.2.1.1 Change Group Name

Allows the user to edit the name of the current Cyclone Group

#### 10.5.2.1.2 Move Activated Cyclone To This Cyclone Group

Brings up a listing of provisioned Cyclones. Check the box next to any Cyclones to add to the current group. The user should be aware that any Cyclones they choose will be removed from their existing Cyclone Group, if they belong to one. Currently this display does not indicate to which Cyclone Group the Cyclone currently belongs.



**Figure 10-9: Add Cyclone To Current Group**

#### 10.5.2.1.3 Delete Group

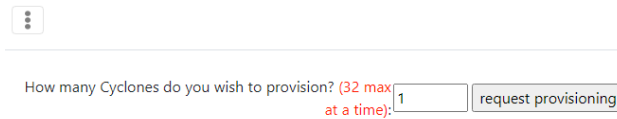
Allows the user to delete the current Cyclone Group. Any Cyclones in the deleted group will be left “floating,” unassigned to a Cyclone Group or VF. These can be re-assigned to a different Cyclone Group by going to that group and using the above action “Move Activated Cyclone To This Cyclone Group.”

### 10.5.2.2 Provision New Cyclones

The Provision New Cyclones button allows the user to create up to 32 new provision codes for the

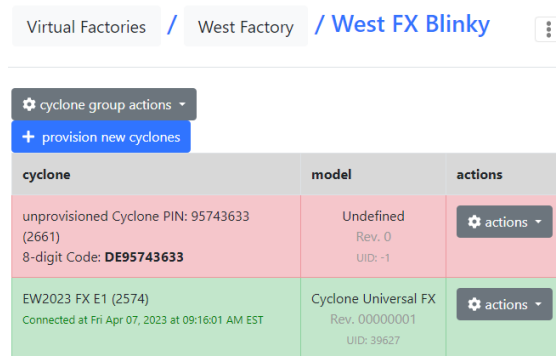
current Cyclone Group. In order to securely link a Cyclone to the user's PEcloud account a provisioning code must be entered into a Cyclone using the Cyclone's menu option Configure Cyclone -> Cloud Settings -> Provision Cyclone. If a Cyclone is already provisioned then the option will switch to De-Provision Cyclone. This task can also be performed using the Remote Display tab of the Cyclone Control GUI.

### Provision New Cyclones



**Figure 10-10: Provision New Cyclones**

PEcloud will display any new provision codes once they are created. These are ready to be entered into a Cyclone you would like to securely link to your account.



cyclone	model	actions
unprovisioned Cyclone PIN: 95743633 (2661) 8-digit Code: <b>DE95743633</b>	Undefined Rev: 0 UID: -1	actions
EW2023 FX E1 (2574) Connected at Fri Apr 07, 2023 at 09:16:01 AM EST	Cyclone Universal FX Rev: 00000001 UID: 39627	actions

**Figure 10-11: New Code(s) Available**

The provision code may be submitted to the Cyclone via the Cyclone Control GUI, from the Cloud tab (this interface also provides the option to provision using cloud API credentials). Alternately, Cyclones may be provisioned directly through their menu screens; please refer to **Section 5.2.4.2.1 - Provision/De-Provision Cyclone** for more detail. Once the user has provisioned a Cyclone, when the PEcloud screen is refreshed that Cyclone will display as green, and the time and date that it was provisioned will be displayed, along with the Cyclone's model and serial number. It is now part of the Cyclone Group from which the code was requested.

## 10.5.3 Jobs Display

Any Jobs that have been deployed to a Virtual Factory will be displayed in a listing that features considerable information about each Job. This includes the following:

- Job ID (links to the Job log), Job name, Job creator
- Encryption Key name & ID
- Programming restrictions, if any
- Job Status, last Status activity, last Programming activity, shows progress towards any programming restrictions

When the Status field for a job is green, that job is Running.

When it is red the Job has been Paused and can be Started again later.

When it is gray the Job has Finished.

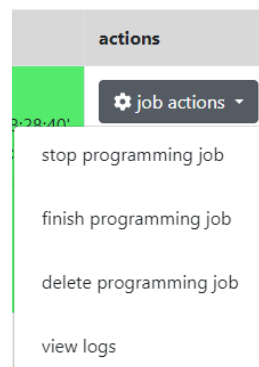
2 Jobs	status	actions
Job Number: <a href="#">1904</a> Job Name: Blinky Bot v.2.0 First Production Run Creation Date/Time: 2023-04-09 18:22:59.0 User Creator: Keith McNeil Key name: KMDocKey01 Key ID: pei-30825DD1-77169A83-641C5995	<b>Running</b> last job status update: '2023-04-09 18:23:01' UTC last program activity: '2023-04-09 18:23:01' UTC  date restrictions: 2023-Apr-08 to 2024-May-08 UTC  program attempts: 0 / 500 max. program successes: 0 / 500 max. program failures: 0	<input type="button" value="⚙️ job actions ▼"/>
Job Number: <a href="#">1903</a> Job Name: Blinky Bot v.2.0 Test Run Creation Date/Time: 2023-04-09 17:10:04.0 User Creator: Keith McNeil Key name: KMDocKey01 Key ID: pei-30825DD1-77169A83-641C5995	<b>Completed</b> last job status update: '2023-04-09 17:10:06' UTC last program activity: '2023-04-10 13:09:35' UTC  date restrictions: none  program attempts: 5 / 5 max. program successes: 5 / unlimited program failures: 0	<input type="button" value="⚙️ job actions ▼"/>

**Figure 10-12: Jobs List - One Job Running, One Job Finished**

**Note:** For any date restrictions applied to a Job, the time for the Start date is considered to be 00:00:00 UTC of the Start date, and the time for the End date is considered to be 23:59:59 UTC of the End date.

### 10.5.3.1 Job Actions

Clicking on Job Actions displays the following options:



**Figure 10-13: Running Job Options**

Depending on the status of the Job, the options are:

#### 10.5.3.1.1 Start/Pause Programming Job

Starts or pauses the programming job. The option displayed is based on the current Status of the Job. Cyclones that have loaded this Job can only program it when it is listed as Running, because Jobs seek permission from PEcloud.

#### 10.5.3.1.2 Finish Programming Job

Marks the Job as "Finished" even if all the of the programming cycles allotted to the Job are not yet complete. Cyclones that have loaded this Job will no longer be able to program it, because Jobs seek permission from PEcloud.

#### 10.5.3.1.3 Delete Programming Job

Removes the programming Job from PEcloud. Cyclones that have loaded this Job will no longer be able to program it, because Jobs seek permission from PEcloud.

#### 10.5.3.1.4 View Logs

Displays the programming log associated with this Job. This will display the time at which

programming occurred, the result, duration, name of the Cyclone that performed the programming, and whether any dynamic data was programmed. Here is an example for a short test run:

Virtual Factories / West Factory / [Job 1903 Logs](#)

Number of lines to get from the end of the logs:  [Get Logs](#)

Programming Count	Request Time	Duration	Result	Cloud Image	Cyclone	Dynamic Data
1	Mon Apr 10, 2023 at 09:06:51 AM EST	6 secs	Success	Blinky v2.00	Jane Blinky Production West 02	Yes
2	Mon Apr 10, 2023 at 09:07:59 AM EST	6 secs	Success	Blinky v2.00	EW2023 FX W4	Yes
3	Mon Apr 10, 2023 at 09:08:35 AM EST	4 secs	Success	Blinky v2.00	Jane Blinky Production West 02	Yes
4	Mon Apr 10, 2023 at 09:08:57 AM EST	5 secs	Success	Blinky v2.00	EW2023 FX W4	Yes
5	Mon Apr 10, 2023 at 09:09:31 AM EST	4 secs	Success	Blinky v2.00	Jane Blinky Production West 02	Yes


**Figure 10-14: Jobs Log**

## 10.5.4 User Groups

A group called “My User Group” is created by default when the PEcloud account is first launched, with the account owner as its sole member.

On the left panel of the PEcloud display, underneath Virtual Factories, is the link for User Groups. The PEcloud account owner (and anyone to whom admin access is later assigned) can include and organize additional users by creating or modifying User Groups and sending invitations. The User Groups display shows which User Groups the user belongs to, and potentially others depending on the user’s permission level.

⚙ user groups actions ▾

user group	my role in this group	# of users	connected virtual factories
 <b>Account Administrators</b>	Account Administrator	2	Users with the Account Administrator Role have access to all Virtual Factories
<b>Just Larry</b>	No Roles assigned	1	Factory East, No Invites No Role assigned
<b>My Users Group</b>	User can Administer/Invite other members	2	Factory West Factory Group Manager

**Figure 10-15: User Groups Display**

The display indicates which role if any the user has been assigned in each user group, how many users are in each group, whether each group is part of a Virtual Factory, and if so what VF Role has been assigned to that group.

Note that the Account Administrators group is a special group with its own set of roles/permissions that differ from those of the other standard User Groups.

### 10.5.4.1 Permissions (Important)

There are two types of permissions to keep in mind when creating User Groups, and the way that they function in PEcloud will to some extent dictate your choices, so please read this section carefully.

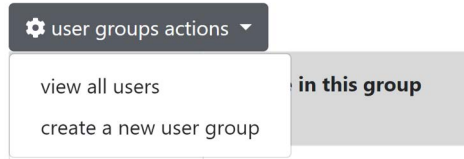
The first type of permission is User Permissions, covered in this section. The other type is Virtual

Factory Permissions, covered in **Section 10.5.1.1 - Virtual Factory Actions**.

**Note:** Unlike User Permissions, which are assigned on a user by user basis, Virtual Factory Permissions are assigned on a **user group by user group** basis. Therefore care should be taken when organizing User Groups to make sure that all of the members of a User Group are eligible for the same VF Permission level, if that is a consideration. A user needing a different VF permission than the rest of the group will need to be removed and added to a separate group.

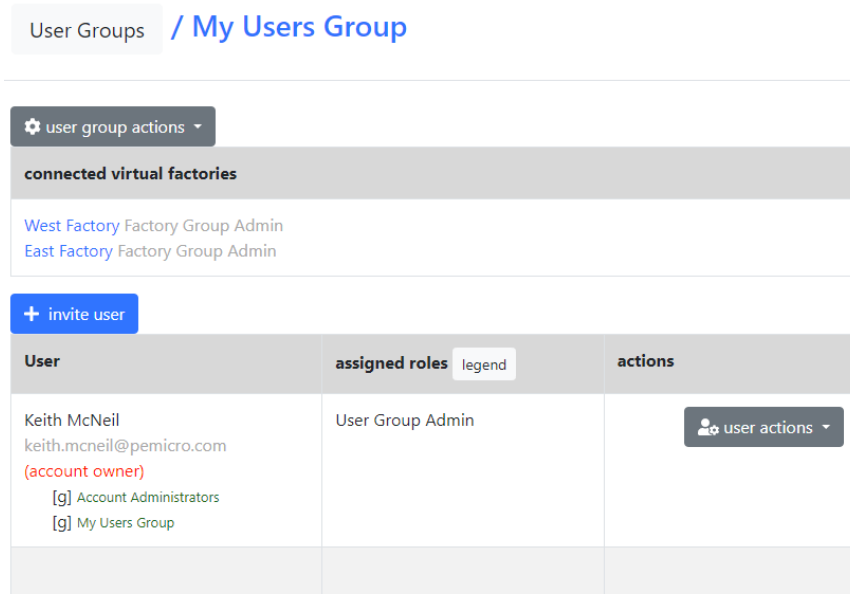
#### 10.5.4.2 User Group Actions

The User Groups Actions drop-down allows the user to view all users for this PEcloud account (in this case, due to having that permission), or to create a new User Group.



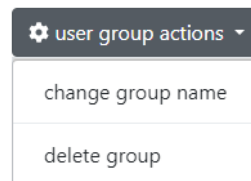
**Figure 10-16: User Groups Actions**

The user can also click on an specific User Group to edit. If they clicked on My Users Group they would see the members along with group- and member-level actions available to them.



**Figure 10-17: User Group View**

The main actions drop-down changes from “User Groups” to “User Group” and includes the following:




**Figure 10-18: User Group Actions**

These actions are self-explanatory.

Individual users of a group have several user actions available in the drop-down menu on the far right of that user that will be active or inactive depending on the role of the active user and whether

other users are present that may be managed.

edit user
change password
remove user from this group
 manage group roles
delete user
view logs

**Figure 10-19: User Actions**

#### 10.5.4.2.1 Edit User

Allows the user to edit this user's information.

#### 10.5.4.2.2 Change Password

Allows the active user to change this user's password to the active user's account.

#### 10.5.4.2.3 Remove User From This Group

This will remove the user from the group.

#### 10.5.4.2.4 Manage Group Roles

Group Roles differ depending on whether the user is managing the Account Administrators user group or a standard user group.

#### **Account Administrators Group - Roles/Permissions**

The following User Roles/Permissions can be assigned:

permission type	role	legend	actions
Account Permissions	Account Administrator		<input type="radio"/>
	Account Manager		<input type="radio"/>
			<input type="button" value="Submit"/>

**Figure 10-20: Account Administrators Group - User Roles**

Clicking on "legend" will show the actions permitted by each of the roles. As the user might expect, the admin role permissions are more global and powerful than the standard role permissions.



Account Administrator
Create/Modify Account Role Definitions
Modify/View/AutomaticMemberOf Any Virtual Factory as an Administrator
Modify/View any Cyclone Group as an Administrator (Including All Cyclones)
Modify/View/AutomaticMemberOf Any User Group as an Administrator (Including All Users)
Assign Group Roles within a Virtual Factory
Create/View New Virtual Factories
Create/View New Cyclone Groups within Virtual Factories
Create/View New User Groups
Account Manager
Create/View New Virtual Factories
Create/View New Cyclone Groups within Virtual Factories
Create/View New User Groups

**Figure 10-21: Account Administrators Group - User Permissions Legend**

Note that custom User Roles with user-selected permissions may be created via the Role Manager. Please refer to **Section 10.7 - PEcloud Account Actions**.

### Standard User Group - Roles/Permissions

The following User Roles/Permissions can be assigned:

permission type	role <span>legend</span>	actions
User Group Permissions	Member of this User Group	<input checked="" type="radio"/>
	User can Administer/Invite other members	<input type="radio"/>
		<input type="button" value="Submit"/>

**Figure 10-22: Standard User Group - User Roles**

Clicking on “legend” will show the actions permitted by each of the roles.

## User Group Permissions Legend



Member of this User Group
View User Groups and its members
User can Administer/Invite other members
View User Groups and its members
Manage Group Roles For Any User
Invite New Users
Ability to add this User Group to a Virtual Factory

**Figure 10-23: Standard User Group - User Permissions Legend**

Note that custom User Roles with user-selected permissions may be created via the Role Manager. Please refer to **Section 10.7 - PEcloud Account Actions**.

### 10.5.4.2.5 Delete User

This will delete the user from the active user's account. The deleted user will no longer see a separate log in for the active user's account (see **Section 10.2 - Creating a PEcloud Account**).

### 10.5.4.2.6 View Logs

User actions are not currently logged.

When viewing a specific User Group there will be an option to invite a new user to the group via email. The invitee will receive an invite link that they can click on to join. They will need to create their own PEcloud login if they do not already have one.

[User Groups](#) / [My Users Group](#) / [Invite User](#)

---

email of new user:

Or pick an existing user to which you have access below:

[u] Keith McNeil  
keith.mcneil@pemicro.com

already invited

use specified or selected user

**Figure 10-24: User Group View - Invite New User**

The inviter will also immediately assign a role for the user that they invite.

## 10.6 Creating and Deploying PEcloud Jobs

Now that the user is familiar with the various components and settings of PEcloud, it is time to return to the process which feeds PEcloud: building Jobs and deploying them to specific Virtual Factories, from where remote Cyclones can fetch and program them according to the organization and constraints designed by the user. Because this user manual already contains sections with very specific information about the utilities used to accomplish these tasks, this section will simply walk the user through an example while referring the user to sections of the manual where the tools/utilities used are covered in greater detail.

Let's walk through an example now of how someone might set up and use PEcloud.

There is also a somewhat more technically-oriented walk-through available to read online:

**“A Walkthrough of PEcloud”**

[https://www.pemicro.com/blog/index.cfm?post\\_id=265](https://www.pemicro.com/blog/index.cfm?post_id=265)

A user might want to produce a run of boards for the latest version of their new product using a facility in another city. But for this production run they would like more control over the programming process, as well as instant feedback on the results. Most importantly, their valuable IP must remain safe.

To get everything up and running using PEcloud, here are the necessary steps:

1. Load encryption ImageKeys onto Cyclone programmers before sending them to the manufacturing facility
2. Join and configure PEcloud
3. Provision Cyclones
4. Build a programming Job
5. Deploy the Job to PEcloud
6. Monitor the results in real-time and make adjustments as needed
7. Optional: Add additional users

**Step 1. Purchase and provision Cyclone programmers**

- a. For this example we will assume the user wants to have 8 Cyclone FX ARM programmers on their production line. They purchase the Cyclones and have them delivered locally so that they can perform the crucial next step.
- b. The user must use Cyclone Image Creation Utility to create a PE ImageKey for encrypting their data, then load a copy of this ImageKey onto all 8 Cyclone programmers. These Cyclones will now be the only Cyclones able to load and execute any programming Jobs that have been encrypted with this same key.

**Note:** It is very important that the user stores their personal ImageKey somewhere secure.

More at section **Section 6.5.3.1.1 - Creating an ImageKey.**

- c. The Cyclones can then be sent to the remote manufacturing facility.

**Step 2. Join and Configure PEcloud**

- a. The user creates a PEcloud account at [cloud.pemicro.com](http://cloud.pemicro.com). More at **Section 10.2 - Creating a PEcloud Account**
- b. The user must download their PEcloud API credentials to their PC where they can be accessed them later when deploying programming Jobs to PEcloud. More at **Section 10.7 - PEcloud Account Actions**
- c. The PEcloud account already includes a Virtual Factory (VF) by default, called “My Virtual Factory.” The user may want to re-name it, for example “Blinky Bot 2.0 Production”. More at **Section 10.5.1.1 - Virtual Factory Actions**
- d. That VF also contains a Cyclone Group by default, called “My Cyclone Group.” The user might also want to rename this, for example “Cyclone FX ARM Programmers”. More at **Section 10.5.2.1 - Cyclone Group Actions**

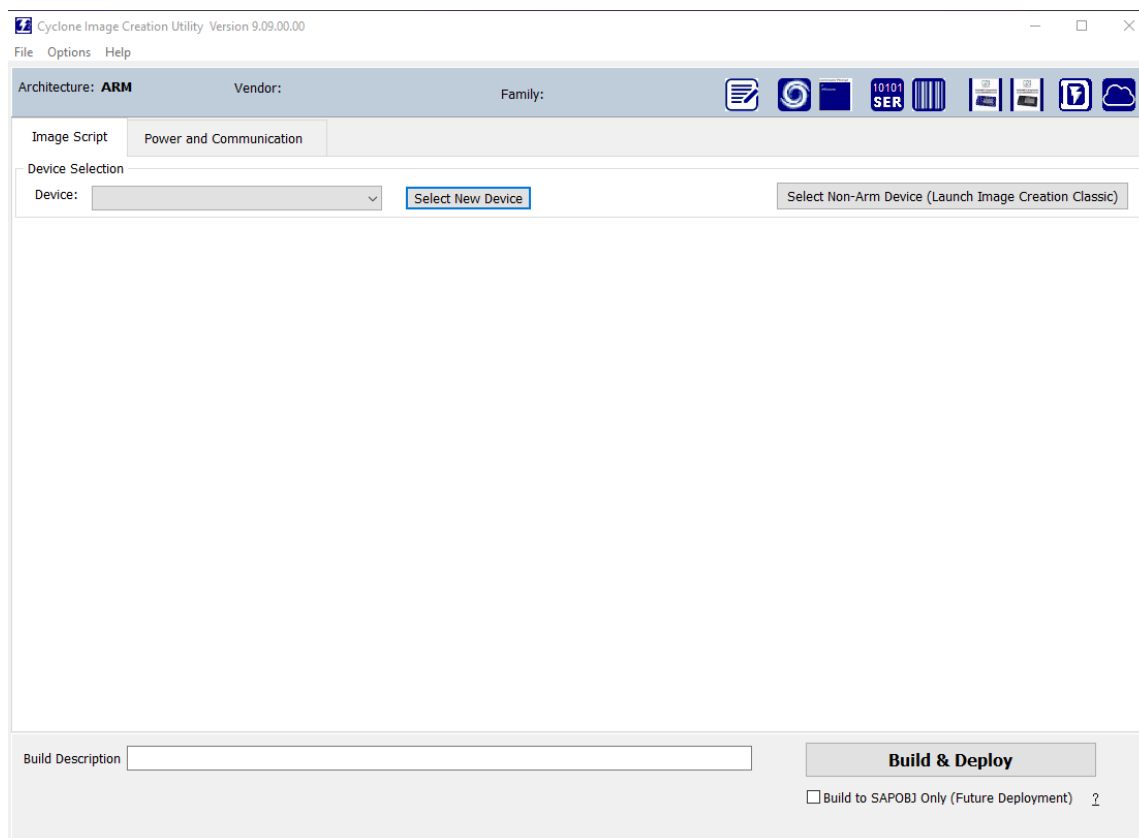
**Step 3. Provision Cyclones**

- a. Once the Cyclones are powered and online at the factory, they can be provisioned with codes from the PEcloud account. To create the provision codes, the user can click on the “Cyclone FX ARM Programmers” group and choose Provision New Cyclones, and in this example request 8 codes, one for each Cyclone. More at **Section 10.5.2.2 - Provision New Cyclones**

- b. To assign the codes the user can open the Cloud Jobs tab of the Cyclone Control GUI to connect to each Cyclone and then use the Provision Cyclone button; this can be completed via codes or API credentials. **Section 8.4.2.7 - Cloud Jobs Tab**
- c. Once the PEcloud page is refreshed it will show that the provisioned Cyclones are all now connected to the PEcloud account, as part of the “Cyclone FX ARM Programmers” group, which will program any Jobs that are deployed to the “Blinky Bot 2.0 Production” virtual factory. More at **Section 10.4 - PEcloud Workflow Overview**

#### Step 4. Creating A Programming Job.

- a. The user decides to build a programming job from scratch because the binaries are new, but they anticipate they may need to recreate this same Job later with slightly different restrictions, so they will also create a SAPOBJ archive.



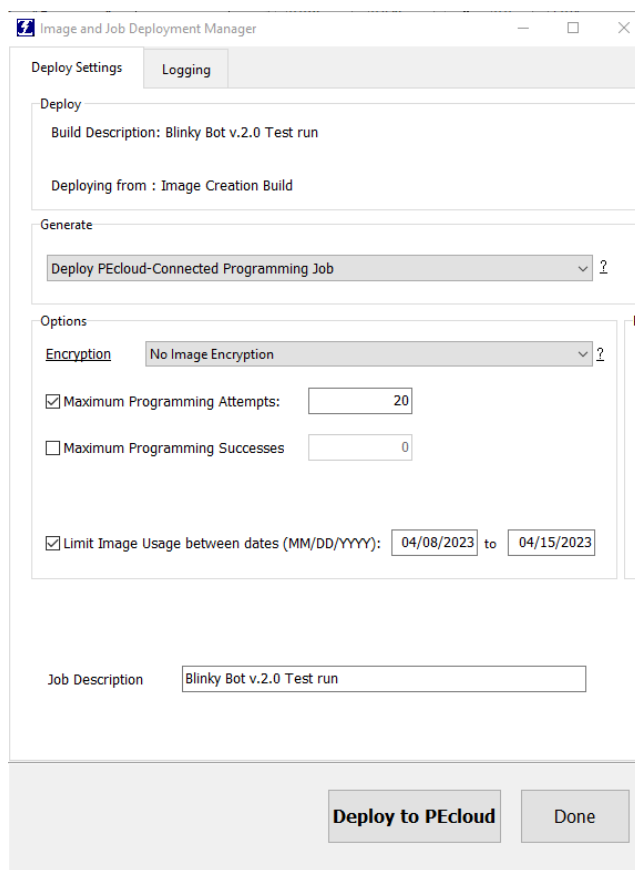
**Figure 10-25: Cyclone Image Creation Utility**

- b. The Cyclone Image Creation Utility is used to specify the programming algorithm for their specific ARM Cortex M device, binaries, and additional settings. The completed programming script might also include shared serialization; if so the eight Cyclones could share a serial number file and program the devices accordingly. **Section 6.2.1 - Image Script Tab.**
- c. Before deploying the Job the user can take the step of saving the configuration as a .SAPOBJ. This is an archival file that allows the user to preserve everything that has already been set up already in the ICU. Later the user can easily load this file and pick up at this same step - ready to complete the deliverable as they see fit. **Section 6.3 - Create a SAPOBJ.**
- d. The user then continues from here to also complete the configuration as a cloud Job, including encryption, and deploy it to their PEcloud account. To do so they hit the Build & Deploy button to launch the utility's Build and Deploy form.

**Note:** For the most detailed information regarding the Image Creation Utility the user should please refer to the entire **Section 6.1 - Cyclone Image Creation Utility.**

## Step 5. Deploying a Programming Job to PECloud

- e. The user wants to use PEcloud to monitor and manage programming remotely, so they choose “PEcloud Connected Programming Job” from Deploy Settings -> Generate.



The screenshot shows the 'Image and Job Deployment Manager' window with the 'Deploy Settings' tab selected. The 'Deploy' section shows 'Build Description: Blinky Bot v.2.0 Test run' and 'Deploying from : Image Creation Build'. The 'Generate' section has a dropdown menu set to 'Deploy PEcloud-Connected Programming Job'. The 'Options' section includes an 'Encryption' dropdown set to 'No Image Encryption', a checked checkbox for 'Maximum Programming Attempts' set to 20, an unchecked checkbox for 'Maximum Programming Successes' set to 0, and a checked checkbox for 'Limit Image Usage between dates (MM/DD/YYYY)' set to '04/08/2023' to '04/15/2023'. The 'Job Description' field at the bottom contains 'Blinky Bot v.2.0 Test run'. At the bottom right, there are two buttons: 'Deploy to PEcloud' and 'Done'.

**Figure 10-26: Build and Deploy Form**

- f. This form also allows the user to make two important additions to their configuration before deploying it as a cloud Job - encryption and programming restrictions. Under Options -> Encryption they browse to the place where they have safely stored their PE ImageKey that they created back in Step 1 - in this example, a thumb drive - which they also loaded onto the Cyclones. This ImageKey will be used to encrypt the Job, and only the Cyclone that were loaded with the same key will be able to de-crypt it. See **Section 6.5.3.1.2 - Encrypting An Image/Job**
- g. The user decides that they would like to start with a test run of 20 units, which need to be done within one week. They sets Maximum Programming Successes to 20, and under Limit Image Usage Between Dates they set the first date to the Cyclones' expected setup date at the remote facility, and the next to one week later. (If later there's an unexpected delay, or they decides to produce more test units, they can easily load the SAPOBJ, click Build & Deploy, and enter a new limit or dates before pushing a replacement image to PEcloud). See **Section 6.5.3.1.3 - Programming Restrictions**
- h. The Job needs a description, so the user edits this field to say “Blinky Bot v.2.0 Test Run (first).” See **Section 6.5.3.1.9 - Image Description**
- i. They select their PEcloud API credentials and then choose the “Blinky Bot 2.0 Production” virtual factory as the place to which to deploy the Job. See **Section 6.5.4 - PECloud Credentials And Virtual Factory Selection**
- j. Finally the user clicks Deploy To PEcloud and the Job is deployed. They logs into their PEcloud account and sees the Job waiting in their virtual factory. See **Section 6.5.5 -**

## Deploy

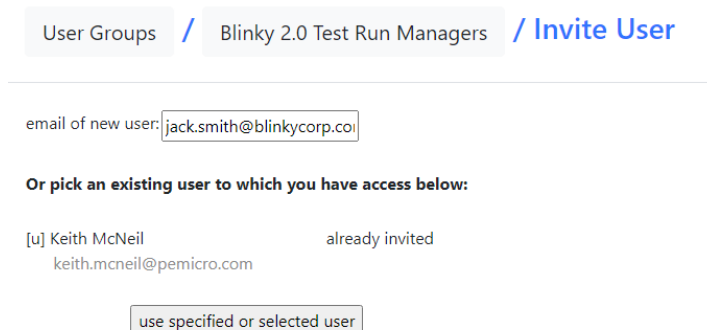
**Note:** For the most detailed information regarding the Image And Job Deployment Form the user should please refer to **Section 6.4 - Build And Deploy an Image or Job**.

### Step 6. Monitor The Results And Make Adjustments As Needed

- The user can now exercise some control over when programming is allowed to occur, and can monitor the results of programming in real time. Their Job actions include: Pause / Start (resume), Finish, or Delete programming Job, and View Logs. See **Section 10.5.3.1 - Job Actions**.
- The user might click “Pause Programming” in order to be absolutely sure that no programming occurs before they have a brief discussion with the remote facility operator. They can then “Start Programming” again when ready. See **Section 10.5.3.1 - Job Actions**.
- To be sure no one has already jumped the gun, they could also click onto the Log for that Programming Job to check if any programming has already been executed. Later during the programming run they can refer back to the log to check the programming speeds and the success rate of the test run.

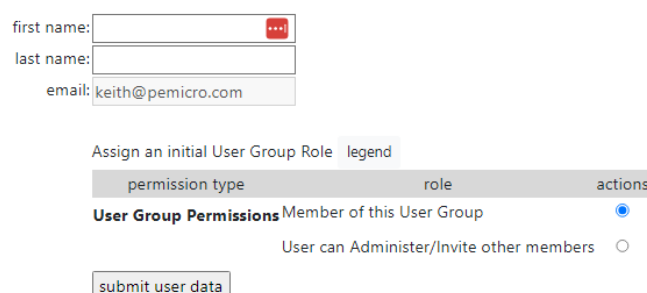
### Step 7. Optional: Add Additional Users

- The PEcloud admin might want to add user to help monitor and manage programming. They click on User Groups -> My Users Group. Under User Group Actions they might, in this example, first hit “change group name” and make it “Blinky 2.0 Test Run Managers.” See **Section 10.5.4 - User Groups**.
- They would then send an invite to the person they have communicated with about helping monitor programming when needed.



**Figure 10-27: Invite To User Group**

- The user determines what adjustments the invited user will be allowed to make to other user roles. In this example they might assign the invited user the Group Member role as they will not need to manage users.



permission type	role	actions
User Group Permissions	Member of this User Group	<input checked="" type="radio"/>
	User can Administer/Invite other members	<input type="radio"/>

**Figure 10-28: Assign User Role For Invitee**



- d. The User Permissions that were just set are the first of two permissions levels. The other is Virtual Factory Permissions. These control the specific actions the user can perform in the VF. In this example, because the user and the invitee are in the same User Group they would be required to have the same level of Virtual Factory Permissions. Here, the user is comfortable with this, but if they ever change their mind they would need to assign the invitee to a different User Group, and then adjust the Virtual Factory Permissions for that User Group. See **Section 10.5.4.1 - Permissions (Important)** and **Section 10.5.1.1 - Virtual Factory Actions**.
- e. When the invited user receives their invitation email, they click on the invite link and create their PEcloud credentials for access to that account. When they log in they sees the invite and can access the “Blinky Bot 2.0 Production” virtual factory where they has the same VF Permissions granted to them as those of the original user.

## 10.7 PEcloud Account Actions

The PEcloud account actions are accessed by clicking on the username at the top right of the PEcloud interface screen. A drop-down box displays the options, which are mostly self-explanatory.

View Account Admins
Change my Company Name
Change my User Name
Change my password
Logout
Roles Manager
Download API Credentials
Provide feedback

**Figure 10-29: Account Options Drop-Down List**

### 10.7.1 View Account Admins

Shows a list of any admin members of the PEcloud account that the user is currently logged into. Admins can help with permissions and other settings if needed.

### 10.7.2 Change My Company Name

Allows the user to edit their company name.

### 10.7.3 Change My User Name

Allows the user to change their user name.

### 10.7.4 Change My Password

Allows the user to change their password

### 10.7.5 Logout

This will log the user out of their PEcloud account and log out any connected Cyclones

### 10.7.6 Roles Manager

PEcloud includes some default Roles with specific sets of permissions assigned. These are not

editable. However the Role Manager enables users with the appropriate permission to create new Roles where the individual permissions for that role may be specified one-by-one.

## Roles Manager

New Role Name: 
  
Virtual Factory Permissions

Permission Type	Role	actions
Account Permissions	Account Administrator	<a href="#">view</a>
	Account Manager	<a href="#">view</a>
Virtual Factory Permissions	Factory Group Admin	<a href="#">view</a>
	Factory Group Manager	<a href="#">view</a>
	Factory Group Member	<a href="#">view</a>
User Group Permissions	Member of this User Group	<a href="#">view</a>
	User can Administer/Invite other members	<a href="#">view</a>

**Figure 10-30: Roles Manager Display**

The current roles are listed at the bottom, and at the top is an interface where the user can create a new role by entering its name into the new Role Name field and then choosing which type of permission category the role will belong to. Note that a new role with zero permissions is created immediately when the user hits “Create New Role.” The user must then add a custom set of permissions on the following Edit page and submit.

### Example:

The Factory Group Manager role is not normally allowed to provision/de-provision Cyclones, but the user could create a custom role called “Factory Manage and Provision” that adds that permission:

Edit Role Name:

Permission Family	Permission	actions
Virtual Factory Management	Create/Delete/Modify/Move Cyclone Groups	<input checked="" type="checkbox"/>
	Move activated Cyclones to my Cyclone Group	<input checked="" type="checkbox"/>
	View Cyclone Groups	<input checked="" type="checkbox"/>
	Provision/De-Provisioning Cyclones	<input checked="" type="checkbox"/>
	Add/Remove User Groups & set User Group Roles	<input type="checkbox"/>
	Change existing User Group Roles	<input checked="" type="checkbox"/>
	Create and Remove Jobs	<input type="checkbox"/>
	Pause and Continue Jobs	<input checked="" type="checkbox"/>
	View Jobs	<input checked="" type="checkbox"/>
	View Virtual Factory	<input checked="" type="checkbox"/>
		<a href="#">Terms Of Service &amp; Privacy Policy</a>
		<input type="button" value="Submit"/>

**Figure 10-31: Create/Edit Custom Role**

Custom roles will appear in the main listing with “edit” and “delete” links that can be used to update custom permissions or remove the role completely.

### 10.7.7 Download API Credentials

PEcloud API credentials allows PEmicro utilities to access the cloud on the user’s behalf when necessary - for example, in the Build and Deployment form when deploying Jobs to PEcloud. In order to allow this, the user must download API Credentials. It’s very important to keep these safe and secured.

## Download API Credentials



[+ Create new API Credentials](#)

API Credentials are used by PEmicro utilities from your PC to access the PEcloud on your behalf. Please note that everytime you create or regenerate your credentials that the previous one is revoked. And also note that these credentials are tied to your user account exclusively, so they will only be able to access the same resources you have access to.

**Figure 10-32: Download Credentials**

Once downloaded these can also be revoked and regenerated by using the buttons provided.

## Download API Credentials

API Credentials are used by PEMicro utilities from your PC to access the PEcloud on your behalf. Please note that e credentials that the previous one is revoked. And also note that these credentials are tied to your user account exc same resources you have access to.

credentials	actions
API Credentials active	<a href="#">Revoke Credentials</a> <a href="#">regenerate Credentials</a>

**Figure 10-33: API Credentials Active**

### 10.7.8 Provide Feedback

PEcloud is a new concept and PEMicro hopes to develop it in collaboration with its users (while in the meantime ironing out any rough patches). We very much welcome feedback on any aspect of the user's experience as well as what they might find useful going forward. We aim to continue improving PEcloud and to provide the features that customers will find the most helpful, in an intuitive interface.

### 10.8 Notifications

Occasionally PEMicro may share noteworthy information using PEcloud's notifications system. The user will see a red indication next to the notification bell near their account name; this indicates how many new notifications are available.



**Figure 10-34: Notification Bell**

Clicking on the bell and then choosing "Manage My Notificatons" will display the user's list of notifications for the user to read and manage.

[All my notifications](#)

You have 1 notification [delete all](#)

notification
<p>Tue Aug 08, 2023 at 09:42:11 PM</p> <p>PEcloud 1.1 on August 12th 2023</p>

**Figure 10-35: Notifications List**

### 10.9 Roles and Permissions Clarifications

PEcloud's User/Admin/Virtual Factory role & permission system, which includes a Role Manager for creating new roles with custom permissions, gives the user detailed control over what their users are allowed to view and what actions they can take. This also means the admin should be aware of some scenarios where roles/permissions may not respond in exactly the way the user anticipates.

#### Double Roles

For example, if a user has for some reason been given both an Admin Role and also a User Role, there may be a situation where an overlap of permissions occurs. In this case, removing a

permission from one role may have no effect without also removing it from the other role. The admin should be careful to check to make sure that they are implementing the setup that they intend. PEmicro recommends limiting a user to either a User Group Role or an Administrator Group Role if possible.

### Overlap

Here are a couple of other examples that may cause unintended confusion:

- a. If the user wants the person they are managing to be able to add a User Group to a Virtual Factory, they will need to make sure that person has the related permission in both their User Role and their Virtual Factory Role.

- b. The Virtual Factory Role permissions:

“Add/Remove User Groups and Set User Group Roles,” and

“Change Existing User Group Roles”

have a somewhat complex relationship. For example, even if the manager has removed the permission to “Change Existing User Group Roles,” this can potentially be circumvented by using the permission “Add/Remove User Groups and Set User Group Roles” to set an existing permission by removing a group and then adding it again. This is because adding a User Group and Choosing a User Role are currently linked. Current plans are to unlink these in a future version of PEcloud, where the user should be able to add a User Group to a VF without choosing a role at the same time.

Please feel free to contact PEmicro with any questions regarding user permissions.

## 11 ETHERNET CONFIGURATION

This section describes the mechanism used by the Cyclone device to transact data over an Ethernet network. It primarily focuses on the User Datagram Protocol (UDP), which is a popular method for sending data over a network when the speed of a data transaction is of more concern than the guarantee of its delivery. The Cyclone takes advantage of the UDP protocol's penchant for speed, and adds an extra layer of logic to guarantee the delivery of UDP packets in order to offer a best-of-both-worlds solution.

### 11.1 Network Architectures

Before delving into the innards of Ethernet message passing, it is prudent to briefly describe the different network architectures in use today, and how they pertain to the operation of the Cyclone. Computers are, of course, connected to one another through intermediary devices in order to form networks. There are several classes of these intermediary devices, but they generally fall into one of the following three groups:

#### Hubs

At the most basic level, computers are connected to one another through a Hub. A Hub is a device with several ports that are used to connect multiple computers together. It is a repeater device – a Hub simply copies the data incoming on one port as data outgoing on the other ports. In this manner, if there are four computers connected through a Hub, and if the first computer is sending data to the second computer, then the third and the fourth computers will also receive an identical copy of that data. Hubs are usually used to set up a small Local Area Network (LAN), which may have on the order of 10 to 20 computers.

#### Switches

The aforementioned type of process, where the data is simply replicated onto every available port, quickly becomes inefficient for larger sized networks. For this reason, a larger sized LAN employs the usage of Switches instead of Hubs. A Switch is essentially a smart Hub, in that it limits the input and output of data to the two transacting computers.

#### Routers

Larger networks, such as Wide Area Networks (WANs), or the Internet for that matter, use progressively more sophisticated devices to transact data. At the core of these devices is the Router, which functions as a switch between networks.

The Cyclone performs irrespective of the connection mechanism, with one very important caveat: it needs to be set up with the appropriate network parameters for the underlying network architecture.

### 11.2 Network Parameters

A typical network becomes operational not after the physical connections have been established, but after network parameters in the form of IP (Internet Protocol) numbers have been assigned to the individual computers. An IP number is a unique string that consists of four numbers ranging between 0 and 255, separated by dots, e.g., 192.168.1.2. Every computer that is on a network needs to have a unique IP number. The computer uses this IP number to identify itself on the network, and also to address the recipient of its data.

Assignment of this IP number is sufficient information to transact data on a simple network connected by a hub. On a more complex network, however, routing information becomes important. The routing information consists of two more IP numbers. The first of these is called the Subnet Mask, and is used to determine whether or not the destination address resides on the same subnet (i.e., doesn't need to be forwarded to another network). The other IP number is the Gateway Address, which is the address of the computer that handles forwarding and receiving of packets to and from other networks.

Before first use, the Cyclone needs to be programmed with a unique IP number, the Subnet Mask IP number, and also the default Gateway's IP number. This can be done via the USB or the Serial port, and is described in greater detail in the "Configuring the Cyclone" section of this manual.



## 11.3 Internet Protocol

Once the network has been established, and the IP numbers have been assigned, data can be transacted over a network with one of several protocols. By far the most prevalent protocol is the Transmission Control Protocol (TCP), which runs on top of the Internet Protocol in what is collectively known as the TCP/IP protocol. The TCP/IP protocol was developed by the Department of Defense to connect different computers from different vendors by a “network of networks,” which has become what is known as the Internet today.

The primary purpose of the TCP/IP protocol was to prevent a complete network outage in the case of a nuclear attack, by automatically rerouting data traffic through the functioning part of the network. As such, the TCP/IP mechanism guaranteed delivery of data packets by introducing a system of acknowledgments and sequence numbers for the data packets. This mechanism, while good for transacting large amounts of data (such as email or file transfers), is unsuitable in the real-time type environment in which the Cyclone operates. Because the Cyclone needs to transact data as quickly as possible to the target, it takes advantage of TCP/IP’s alternative, the UDP/IP protocol.

Unlike TCP/IP, the UDP/IP protocol is a connectionless, single-packet protocol that sends short data packets at the expense of not guaranteeing their delivery. This makes the UDP/IP protocol efficient in real-time applications such as broadcasting video over the Internet, where the occasional loss of a frame of data is not going to hamper the overall viewing experience. Left unmodified, the UDP/IP, with its lack of guarantees for packet delivery, would be unusable in an environment where the delivery of a single byte of data needs to be guaranteed. The Cyclone firmware adds mechanisms to the UDP/IP protocol, without affecting its underlying efficiency, to guarantee delivery of data packets.

## 11.4 Connecting The Cyclone Device

There are two methods for establishing a connection between a Cyclone and a PC with an Ethernet cable. The most basic method is to connect the Cyclone directly to a PC, via a cross-over Ethernet cable. However, the more common method is to place the Cyclone and the PC on the same network through a Hub.

### 11.4.1 Connecting the Cyclone to the PC over a network

The Cyclone was intended for use on a network of multiple computers (and other Cyclones). There are many possible network configurations, and to describe them all is beyond the scope of this document. However, most configurations are a modification of a basic theme, which is that of connecting one or more PCs through a Hub to one or more Cyclones.

In order to connect these devices to the Hub, you will need to use the provided straight-through Ethernet cable. The straight-through cable, which is the “standard” Ethernet cable, is used to connect devices of different types together, such as a PC to a Hub, or a Hub to a Cyclone.

At this point it once again becomes necessary to program the Cyclone with valid IP numbers, the process for which is described in greater detail in the following section. However, it is important for the Cyclone and the PCs to have matching Subnet and Gateway IP numbers, and for each to have a unique IP number on the network. An example of a setting for above is as follows:

	<u>IP Number</u>	<u>Gateway IP</u>	<u>Subnet Mask</u>
<b>PC1</b>	192.168.100.1	192.168.100.3	255.255.255.0
<b>PC2</b>	192.168.100.2	192.168.100.3	255.255.255.0
<b>CYCLONE</b>	192.168.100.4	192.168.100.3	255.255.255.0
<b>Gateway</b>	192.168.100.3	192.168.100.3	255.255.255.0

It is important to briefly touch upon the underlying network architecture, which can be a 10Mb (Megabit), 100Mb, 10/100Mb, half-duplex, or a full-duplex connection. The details of the underlying network architecture are beyond the scope of this document, but it is sufficient to note that most modern network cards, as well as the Cyclone device, have the capability to configure themselves for the underlying network through the Auto-negotiation mechanism. Auto-negotiation is performed as soon as a network cable is connected to the device, and it sets the operating parameters of the

device to match those of the network.

## 11.4.2 Connecting Cyclone-to-PC via an Ethernet cable

In order to connect the Cyclone to a PC directly via an Ethernet cable, you need to use what is known as a cross-over cable. A cross-over cable, which is not provided by PEmicro, is normally used to connect two similar devices such as a PC to a PC, or a Hub to a Hub. It is a cable that has its receive and transmit wires crossed over so that the similar devices can effectively communicate with one another.

With this configuration, it is still important to assign IP numbers to both the PC and the Cyclone device. Although at first glance it may not seem necessary to assign a Gateway address in this configuration, the Cyclone was designed to operate on a network of more than two computers, and therefore it needs to be programmed with a Gateway address.

Assuming the desktop's IP number to be 192.168.100.1, this is an example of the three IP numbers that would need to be programmed into the Cyclone:

	<u>IP Number</u>	<u>Gateway IP</u>	<u>Subnet Mask</u>
<b>PC</b>	192.168.100.1	none	255.255.255.0
<b>CYCLONE</b>	192.168.100.2	192.168.100.1	255.255.255.0

For more information on programming these IP numbers into the Cyclone device, please see the following section.

## 11.5 Cyclone IP Setup Via LCD Menu

When the user is connecting the Cyclone via Ethernet, before the connection is established between the Cyclone and the network the menu's Home Screen will display the Cyclone's IP address as 0.0.0.0.

Once a connection has been established, the menu's Home Screen displays the Cyclone's IP address and connection setting (Static or Dynamic).

The Ethernet cable can either be attached at the start of Cyclone startup or connected after setup is complete. The connection with the network will be established when the cable is connected. If the Ethernet cable is disconnected after setup is complete, the user should be able to simply reconnect the cable to reestablish networking. However, depending on the setup of the DHCP server, if the Ethernet cable is left unplugged for a considerable time the IP address may expire and connection will have to be set up once again. This can be accomplished by restarting the Cyclone.

### 11.5.1 Configure Network Settings

To configure network settings for the Cyclone, navigate to the following Menu location:

***Main Menu / Configure Cyclone Settings / Configure Network Settings***

The following options will be available under Configure Network Settings:

- Show Current IP Settings
- Edit Static IP Settings
- Enable/Disable Dynamic IP
- Edit Cyclone Name

#### 11.5.1.1 Show Current IP Settings

Show Current IP Settings displays the current IP settings, including:

- Current IP Mode
- IP Number
- Mask
- Gateway

- MAC Address

If you are in Static IP mode, these settings (excluding the MAC address) may be changed by tapping on them. In this case a tap will take you to the Edit menus. If you are in Dynamic IP mode, tapping will show a message that the Cyclone settings cannot be changed.

### **Dynamic vs. Static**

There are two schemes for assigning IP addresses. One is the Static IP addressing mode. This involves the user manually setting the IP address for every device on the network. In this case, it falls to the user to ensure the IPs assigned do not conflict and are within the boundaries of the network. The other is the Dynamic Host Configuration Protocol (DHCP). This involves setting up a separate server to manage the IP addresses. The server is given a list of valid IP addresses for the network. Using a predetermined set of rules, each new device that wishes to connect to the network is given an IP address by the server. This takes the task of managing the validity and uniqueness of IP addresses out of the user's hands and relegates it to the server. **Cyclone FX** programmers are capable of using either Static IP addressing or DHCP.

**Note:** The current IP settings may also be viewed/edited by navigating to:

***Main Menu / Status / Show Current IP Settings***

#### **11.5.1.2 Edit Static IP Settings**

This allows editing of IP, Mask, and Gateway in Static IP mode. In the edit dialogs, the user must enter a valid IP address to continue:

##### **Format**

xxx.xxx.xxx.xxx

##### **Where:**

0 <= xxx <= 255

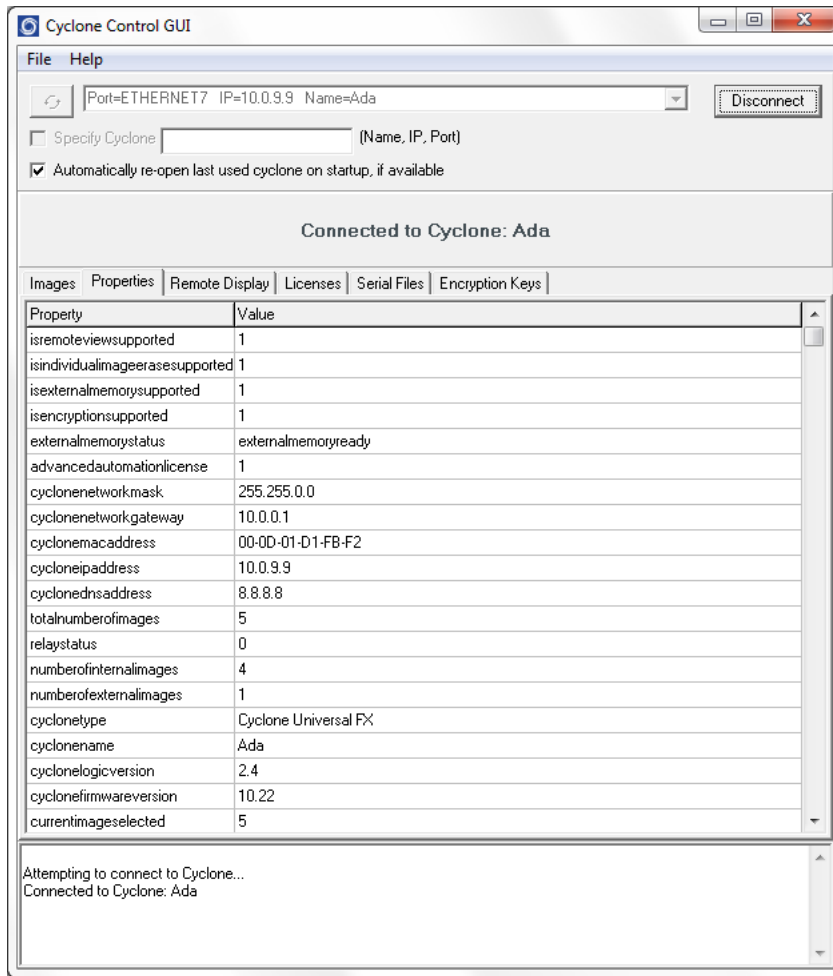
#### **11.5.1.3 Enable/Disable Dynamic IP**

Opens a dialog to toggle the IP settings between Static and Dynamic. Once an option is selected a message is displayed indicating that the Cyclone must be reset for this option to take effect. The reset button on the front side of the Cyclone may be used.

## **11.6 Configuring Cyclone Network Settings using the Cyclone Control GUI**

Before the Cyclone device transacts data on an Ethernet network, it will need to be configured with the relevant network parameters. This configuration can be done in the "Properties" tab of the Cyclone Control GUI.

To access the "Properties" tab, select the Cyclone from the drop-down list in the Cyclone Control GUI and click on "Connect". The "Properties" tab will be accessible once the Cyclone is open.

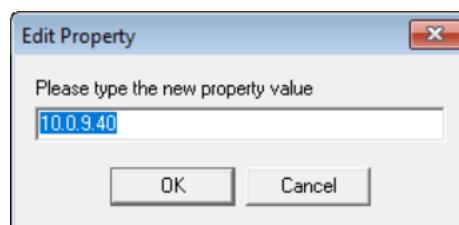


**Figure 11-1: Cyclone Control GUI: Properties Tab Selected**

From this tab, all Cyclone and Network properties can be accessed. Some of this properties are modifiable, including all the properties needed for network configuration. A property that can be edited will show three dots to the right of the property when you select it by clicking on its box. You can click on the three dots or double click on the property value to bring up the edit window.



**Figure 11-2: Dots Displayed At Right When Property Selected**



**Figure 11-3: Edit Property Window**

Once the "OK" button is pressed, the values of the property will be updated in the Cyclone and the value of the property in the Cyclone Control GUI will be refreshed showing the new value.

## 12 SAP IMAGE ENCRYPTION

**CYCLONE FX** programmers and **CYCLONE** programmers with the ProCryption Security Activation License allow users to create RSA/AES encrypted programming images that use their own uniquely generated ImageKey. These encrypted images may only be used to program when on Cyclones that are also pre-configured with the same ImageKey. This keeps the user securely in control of both their IP and the programming process.

### 12.1 Overview

PEmicro uses a combination of industry-standard RSA and AES encryption technologies to encrypt images. When a programming image has been encrypted it requires two different asymmetric keys to be decrypted. The first is a user generated RSA encryption Key that was specified when the programming image was generated. The second is a native key which comes pre-installed in the Cyclone (and does not exist on the PC). This means that an encrypted image may (A) only be loaded onto a Cyclone which holds a copy of a user generated ImageKey, and (B) only be decrypted for programming while on a Cyclone which holds a copy of a user generated ImageKey. The Cyclone Control Suite (GUI, Console, SDK) allows the user to add and delete ImageKeys from Cyclones, much like programming images may be added or deleted. While many users will use only a single ImageKey to encrypt all of their images, Cyclones may have many different keys loaded.

Encrypted images are stored in the Cyclone in their encrypted form. If the ImageKey needed by a programming image is deleted from the Cyclone, the Cyclone loses the ability to load any images encrypted with that ImageKey or program any encrypted images encrypted with that ImageKey that are already loaded. Adding the ImageKey back into a Cyclone gives that Cyclone access to those stored encrypted images which require that ImageKey.

Encrypted images can safely be sent through electronic means to production facilities since they are unusable without a Cyclone which has been pre-loaded with the appropriate ImageKey.

ImageKeys on the PC are themselves partially encrypted so that certain pieces can only be used on a Cyclone. Even with this, they should be handled with care as they can be loaded into any Cyclone.

### 12.2 Encrypting/Decrypting a Programming Image

The Cyclone Image Creation Utility can generate ImageKeys that are used to encrypt SAP images. The steps that are necessary to generate ImageKeys, encrypt SAP images, provision Cyclones to decrypt, and program with encrypted SAP images are detailed in **Section 6.3.1.1 - ProCryption Security Features**.

### 12.3 What is Encrypted in an eSAP File, and How

An encrypted image (eSAP) contains three distinct sections: an informational header, a configuration section, and a stand alone programming (SAP) data section. The ImageKey encrypts each section in different ways to control access to each portion of the eSAP file.

The three eSAP sections are:

#### 1) Informational Header

This section includes the description of the eSAP Image, its unique ID, the ID and name of the ImageKey used to encrypt it, and a checksum of the data. This section is not encrypted.

#### 2) Configuration Section

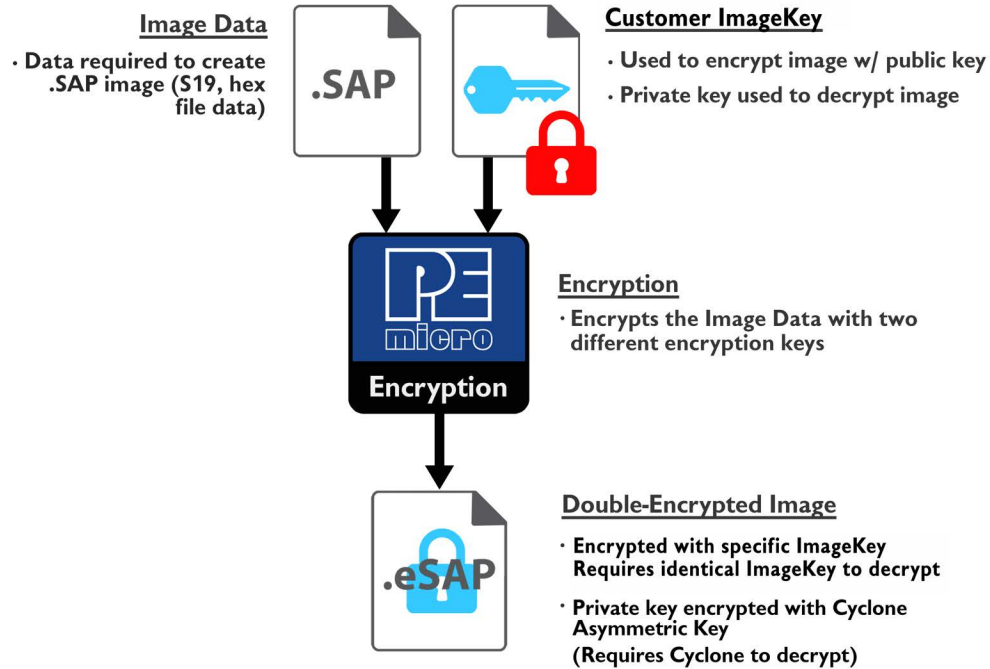
This section contains a copy of the configuration settings used to generate the Image including which algorithm was used, power settings, clock settings, script files, and paths to the binary data files. No programming data from the user's data files is included in this section. This section is encrypted in such a way that if a user has the appropriate ImageKey on the PC, they may import the configuration information from an eSAP file into the Image Creation Utility. This is useful for seeing the settings used to generate an image, and, if the user has all of the data files needed, generate a new programming image file with the same



configuration.

### 3) Stand Alone Programming (SAP) Data

This section contains all of the information a Cyclone needs to program a target as specified in the image creation process. This includes all programming data, algorithms, scripts, settings, etc. This section is encrypted with several keys, including the user generated asymmetric key as well as a native asymmetric key used by the Cyclone. Once encrypted, this section may not be decrypted except by the Cyclone during the programming process.



**Figure 12-4: SAP Encryption**

The end result of the encryption used to proceed the Stand Alone Programming Data is that the section can only be decrypted and used internally on a Cyclone which has a copy of the specified ImageKey provisioned within it. This eSAP section cannot be decrypted on a PC even with the ImageKey

## 12.4 Managing Encryption For Production Programming

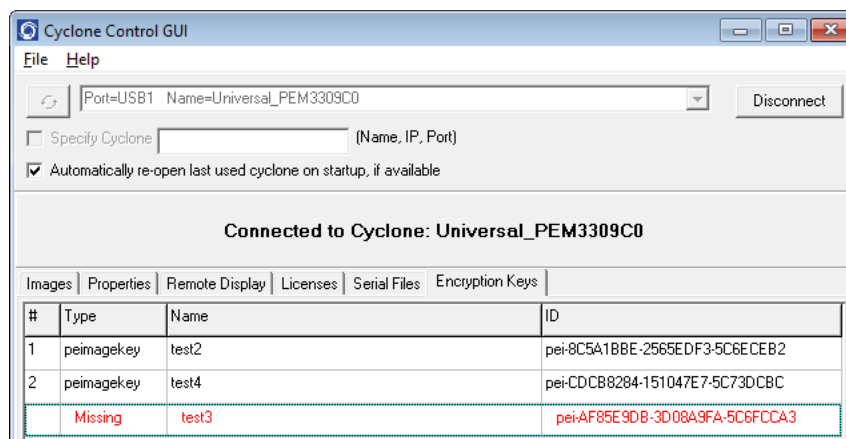
The steps needed to encrypt programming images using the Cyclone Image Creation Utility are detailed in **Section 6.5.3.1 - Encryption (ProCryption Security)**. This section details how to successfully implement the use of these encrypted (eSAP) images into the production programming process.

### 12.4.1 Provisioning a Cyclone with an ImageKey

Cyclones that have been provisioned with an ImageKey are the only Cyclones that are able to load and program eSAP images encrypted with that ImageKey. When the user determines that one or more Cyclone programmer(s) will have access to an encrypted image, they need to load the ImageKey that was used to encrypt that image onto the Cyclone. This can be done with the Cyclone Control Suite GUI, Console, or SDK. The transfer of an ImageKey to a Cyclone using the Cyclone Control Suite is always secured by encrypting the ImageKey with an RSA public key specific to the Cyclone that it is connected to. Instructions on the use of these Control Suite options is explained in **CHAPTER 8 - CYCLONE PROGRAMMER AUTOMATED CONTROL**.

**Figure 12-5** shows the Cyclone Control GUI with the Encrypted Keys tab selected.



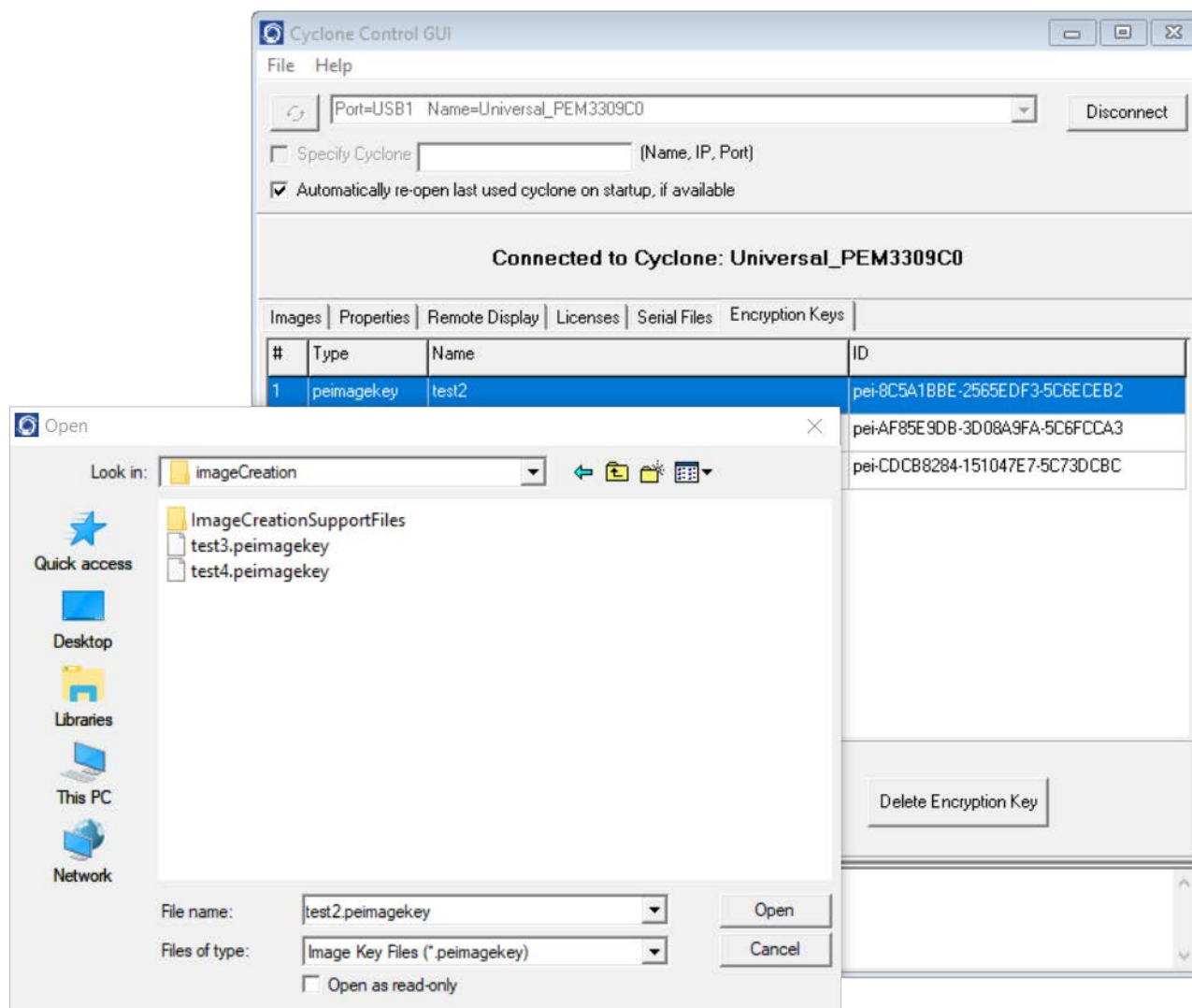


**Figure 12-5: ImageKey Listing for Connected Cyclone**

This tab displays any ImageKeys that are present on the Cyclone to which the user is connected.

**Note:** If there is an encrypted SAP image on the Cyclone whose corresponding ImageKey has been removed, the required ImageKey will be displayed as “Missing,” along with its Name and ID.

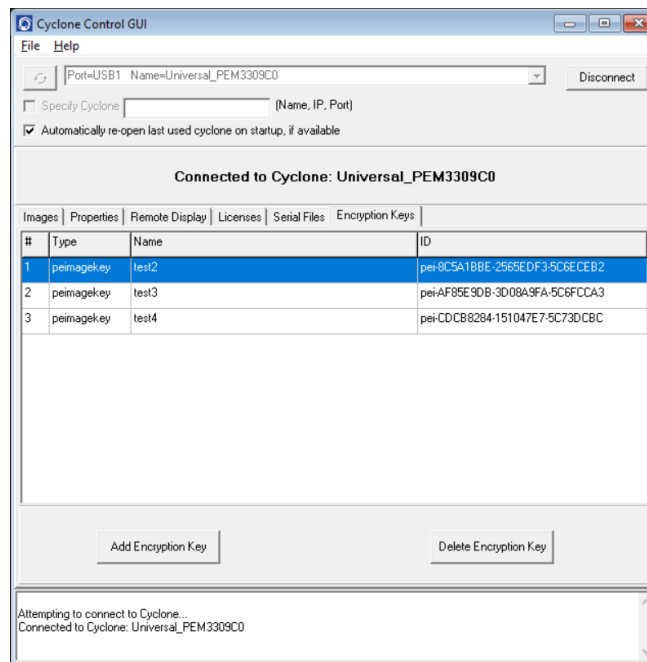
To provision a Cyclone with an ImageKey, the user simply clicks the “Add Encryption Key” button and browses for the ImageKey that they wish to load onto the connected Cyclone.



**Figure 12-6: Browse for ImageKey**

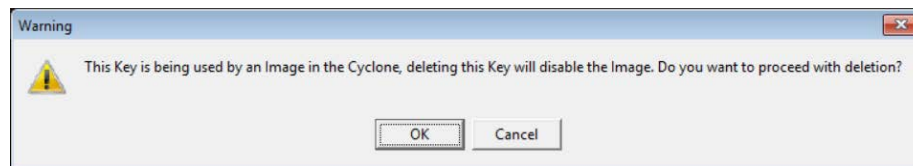
## Removing ImageKeys From A Cyclone

It is also easy to remove an ImageKey from the connected Cyclone. The user selects the ImageKey that they wish to delete and hits the “Delete Encryption Key” button.



**Figure 12-7: ImageKey Selected for Deletion**

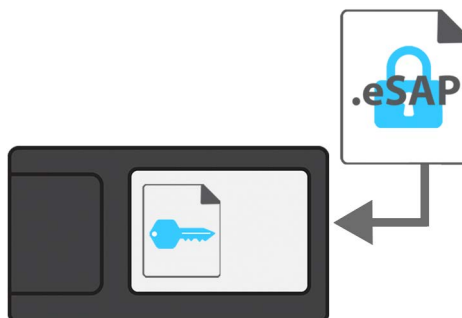
If the ImageKey that was selected for deletion is one that is required to decrypt one or more eSAP images that are on the connected Cyclone, a warning message will be displayed when the Delete button is pressed. The user may then cancel the deletion or confirm that they wish to proceed.



**Figure 12-8: Warning: ImageKey Used By SAP Image on Connected Cyclone**

## Loading and Programming with Encrypted SAP Images

From the user's perspective, loading an eSAP file with the Cyclone Image Creation Utility and programming with that eSAP file looks the same as loading/using a SAP file which is not encrypted (as long as the appropriate ImageKey exists on the Cyclone).



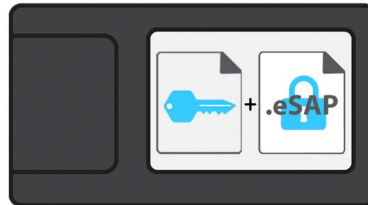
**Figure 12-9: Load Encrypted Image (eSAP) onto Cyclone with Corresponding ImageKey**

When the user attempts to load an encrypted programming image onto a Cyclone, the Cyclone will

refuse to load the image unless the appropriate ImageKey resides on the Cyclone. If the ImageKey is present, the Cyclone will load the eSAP File and automatically decrypt it using both the matching ImageKey and the Cyclone's internal decryption mechanism.

A user may create and use multiple ImageKeys and corresponding eSAP images on a single Cyclone.

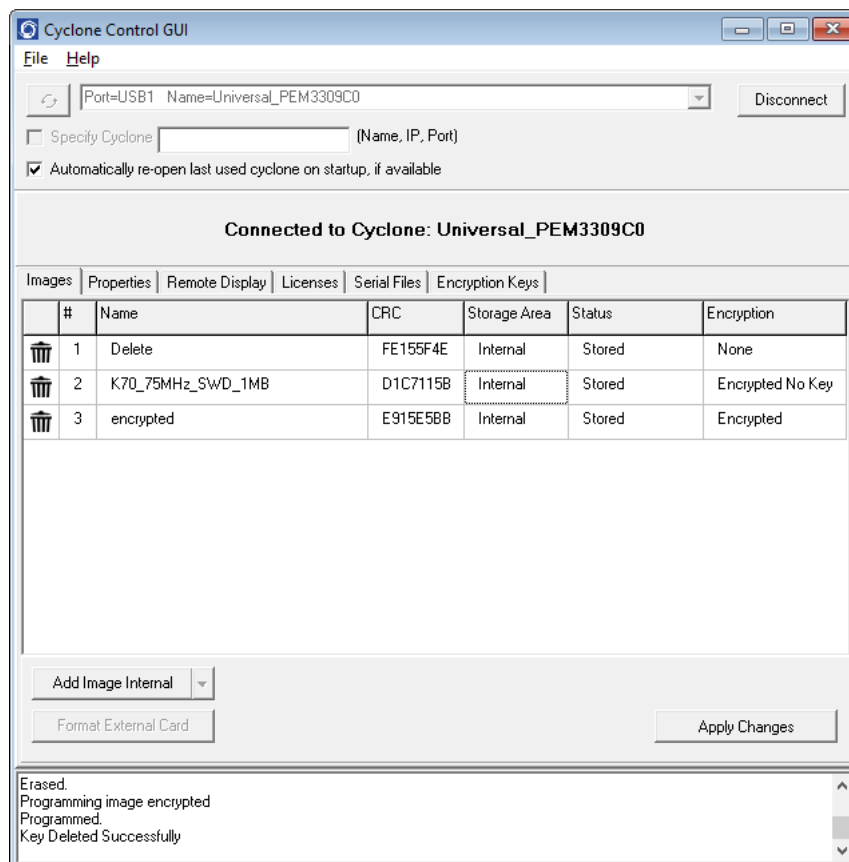
The decryption process is transparent to the user when the Cyclone goes to program the target. The decrypted programming image appears as usual on the Cyclone menu screen. The user can now program devices using the unencrypted SAP Image.



**Figure 12-10: Decryption Occurs Automatically For Programming**

#### 12.4.4 Encryption Status of SAP Images

In the Images tab of the Cyclone Control Suite GUI the user can view the encryption status of SAP images that reside on the connected Cyclone. **Figure 12-11** shows the Cyclone Control GUI with the Images tab selected.



**Figure 12-11: SAP Image Listing for Connected Cyclone**

The rightmost column, labeled Encryption, displays whether each SAP image is unencrypted ("None"), encrypted with the required ImageKey present ("Encrypted"), or encrypted but missing the ImageKey required to decrypt ("Encrypted No Key").

## 12.5 Safer Production That's Easy To Implement

SAP Image encryption with Cyclone programmers is simple to implement and helps keep valuable intellectual property safe. Protected programming images can safely be sent electronically to remote production facilities. This adds much needed convenience and peace of mind to the production process.

## 13 USING A BARCODE SCANNER TO SELECT AN IMAGE & INITIATE PROGRAMMING

### 13.1 Introduction

PEmicro's CYCLONE FX programmers are capable of using a barcode scanner during stand-alone programming. Scanned barcodes can be used to automatically select and program a specific SAP (Stand Alone Programmer) image into a target. This means the programming image does not need to be pre-selected before programming. Also, there is no need to hit the start button. Simply scan the bar code and programming will be initiated.

The Cyclone FX scans the barcode and checks all resident images for a barcode match. If exactly one match is found, the image is selected and used to program the target. If no matches are found, or multiple matches are found, an appropriate error is reported. This reduces manual configuration errors especially when large numbers of programming images and product types may be programmed.

Automatic selection and launch of a specific flash programming image based on a scanned barcode can improve the speed and accuracy of production programming, especially when there is a varied product mix being programmed. Barcode scanning improves accuracy by making the process of selecting a programming image fast, automatic, and less vulnerable to user error.

The barcode itself can optionally be programmed into the target device's memory as part of this manufacturing process. The information that the barcode makes available to the device can make it easier to trace products, track product hardware versions, or provide a way to serialize production. This results in a more efficient manufacturing process.

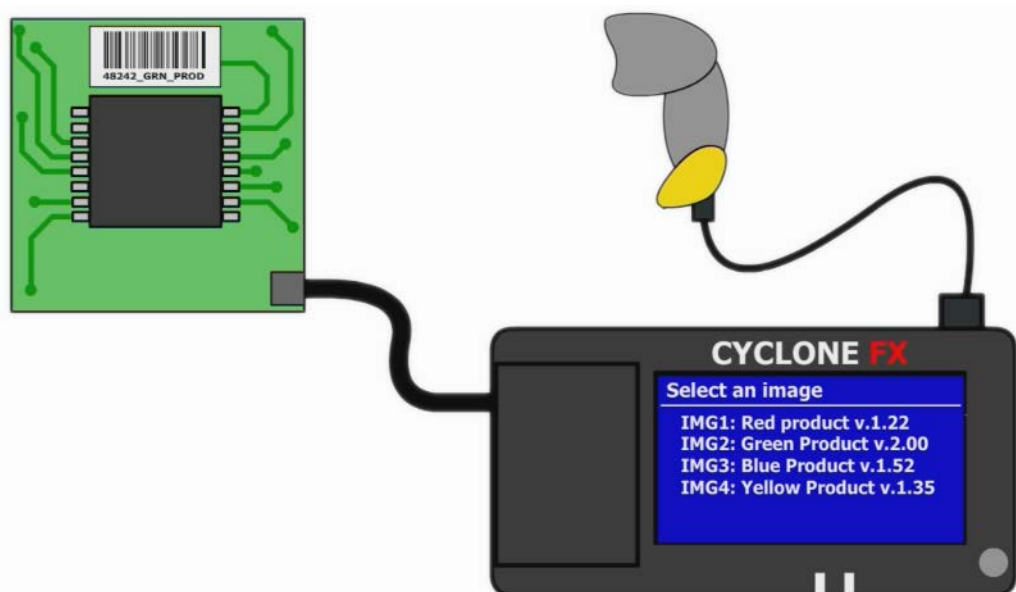
### 13.2 Scanning Procedure

After the CYCLONE FX is configured (see **Section 13.4 - Enabling Barcode Scanner In Cyclone Menu**), the barcode scanner is connected to the Cyclone's host USB port. The scanner is used in Wedge output mode, which emulates a USB keyboard. When the scanner scans a barcode, it transmits the barcode to the Cyclone. The Cyclone then analyzes the barcode and uses it to select a programming image.

To perform this analysis, all programming images on the Cyclone, in both internal and external memory, are reviewed. Each image may contain a set of rules to determine whether a scanned barcode corresponds to that particular image. This set of barcode analysis rules is referred to as an image's "barcode test." If the barcode passes the test on one and only one of the programming images on the Cyclone, that image is automatically selected and programming is initiated. Otherwise, if the scanned barcode does not pass the barcode test on any image, or if it passes the test on more than one image, an error will occur.

Barcode test rules include: barcode length, character type, specific characters, and numerical ranges. Fixed barcodes or barcodes which are unique to an individual product can be properly analyzed and used to select the appropriate programming image.

A logfile is generated to help the user understand how a scanned barcode auto-selects a specific image on the Cyclone (or yields an error). See **Section 13.9 - Troubleshooting** for more information on this logfile.



**Figure 13-1: CYCLONE FX With Barcode Scanner**

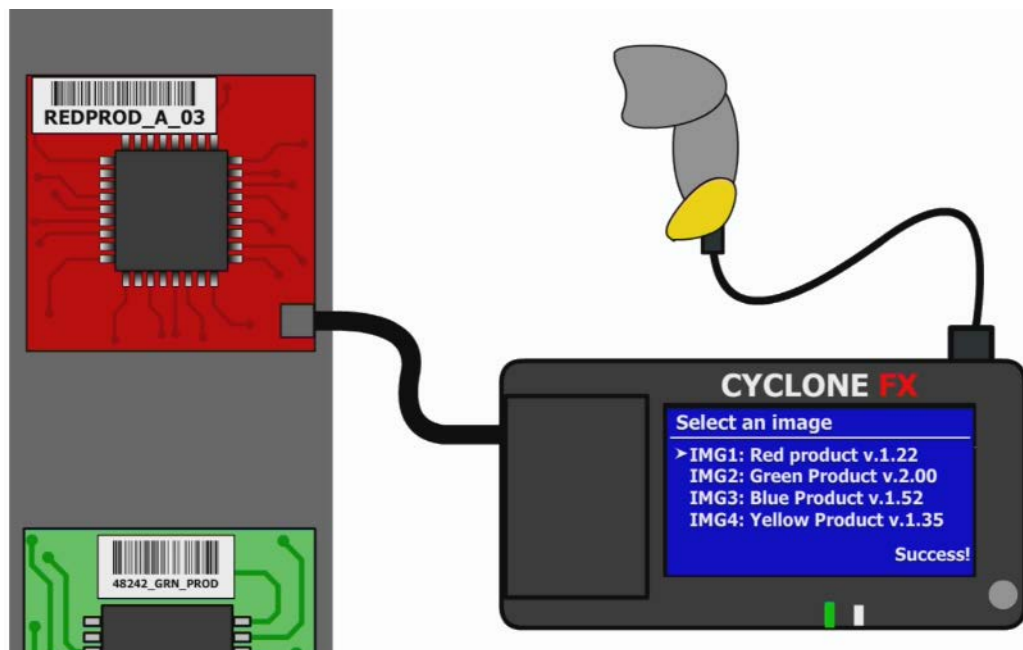
### 13.3 Potential Benefits Of Programming Via Barcode Scan

The use of a barcode scanner as part of the manufacturing process can boost productivity and reduce human error. When the user scans a barcode to select and launch the programming sequence, it frees that user from having to know any details about the product being programmed; point, shoot, and it programs. Programming multiple products with different data can be as simple as connecting each new target board and scanning a barcode that is printed on it.

A manufacturer might, for example, design their products such that every product has a unique barcode printed on it. Part of these barcodes might be a fixed string, and part of them might be changing (for example :WIDGET1-A00143, WIDGET1-A04325, WIDGET1-B03222). In this scenario, the barcode test for the WIDGET1 programming image could check that the barcode starts with the fixed string "WIDGET1-" (product identifier), the next character is an A or B (product version), and that the rest of the barcode is numeric (unique ID). If it passes, the WIDGET1 image would be selected and the target programmed.

This same manufacturer might also set up the programming process for each product to program the barcode itself at a specific point in memory. There are then many ways a device could leverage its barcode number. For example, the running application in the device could check the product version of the barcode to understand whether it is running on hardware revision A or B and make decisions based upon that information. The Unique ID could also be used for this purpose. E.g., if the number is less than 5000, then use high drive strength on the product's port X; otherwise use low drive strength on the product's port X. When connected later to the cloud, it can report its unique identifier back to its home base. Potentially it could use the barcode's Unique ID as part of setting up its MAC address for Ethernet.





**Figure 13-2: Bar Code Scanner On Production Line**

In a completely different programming scenario the manufacturer might not want to place barcodes on the products themselves. E.g., it could be that they do not frequently switch from one product to another on the production line. In this case, they might prefer to create a separate set of directions for each product they manufacture which covers programming, test, and casing of the product. They could then include a fixed barcode as part of the programming instructions. The instructions might state that, before starting a production run of this product the barcode should be scanned by the Cyclone programmer. This removes the manual step of the operator choosing which image to use for each product. The Cyclone selects the image automatically by using the barcode in the instructions (and if there is not a unique match, an error is reported). This can help to reduce user error, particularly if the programming images on the Cyclone carry similar names.

### 13.4 Enabling Barcode Scanner In Cyclone Menu

To use the barcode scanner with a CYCLONE FX, the scanner must first be enabled in the Cyclone menu by navigating to “Configure Cyclone” and then “Configure USB Host Device.” The user should click on “Enable USB Scanner”. The barcode scanner’s instructions should be consulted to understand how to put the scanner into Wedge (keyboard emulation) mode, and to make sure that it terminates the barcode with a CR (carriage return) character. The scanner can then be plugged into the USB host mini connector on the side of the Cyclone. A USB-to-USB-Mini adapter may be necessary if the scanner uses a full size USB connector. When plugged into the Cyclone, the scanner will power up and the Scanner Active icon on the Cyclone LCD screen will illuminate.

Barcode Scanner Active icon: 

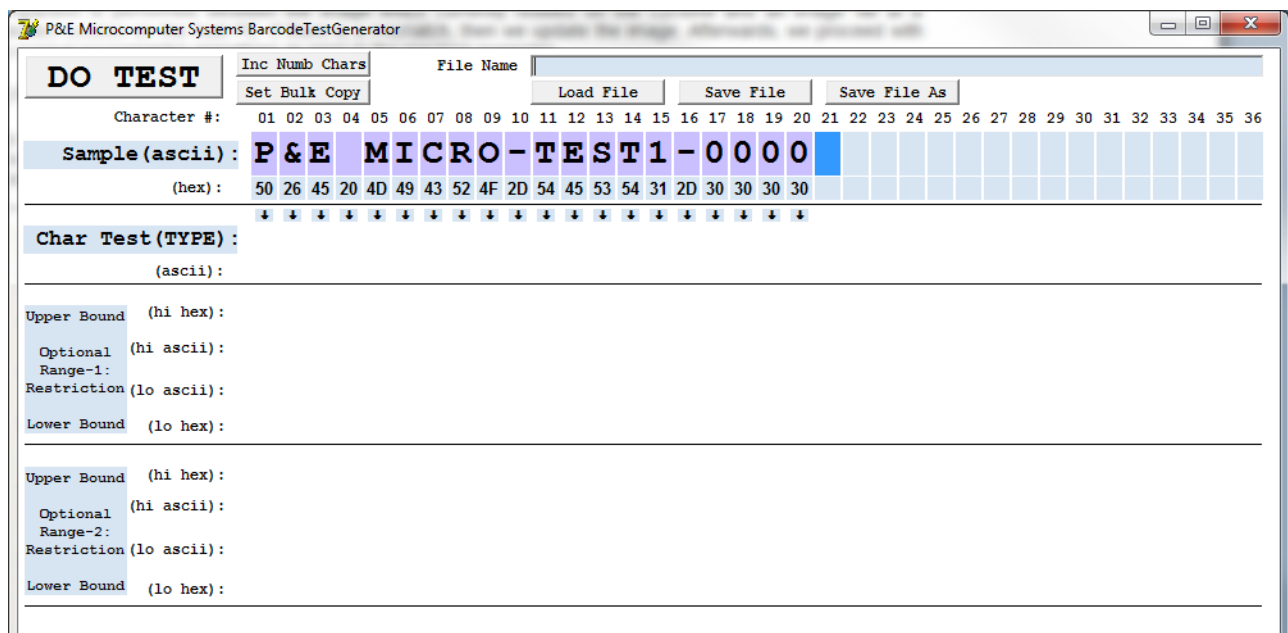
### 13.5 Creating A Barcode Test: Quick Example

This section demonstrates one simple example of how to create a barcode test by using the Barcode Test Generator utility. For in-depth instructions on how to use the Barcode Test Generator, please refer to **Section 13.6 - Creating a Barcode Test: In Depth**.

Whenever a programming image is created the user has the option to add a barcode test to the image. PEmicro provides a Barcode Test Generator utility to enable the user to creating these tests. The following steps demonstrate how to create a sample barcode test.

To begin, the user should launch the Barcode Test Generator utility. The user should then write a prototype of the expected barcode in the “Sample(ascii)” section. In this example, the prototype barcode is “P&E MICRO-TEST1-0000”. See **Figure 13-3**. Please note that this example only

demonstrates one simple approach to creating a barcode; there are many methods available.



**DO TEST**

Inc Numb Chars:  File Name:

Set Bulk Copy:

Character #:	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
Sample (ascii):	P	&	E		M	I	C	R	O	-	T	E	S	T	1	-	0	0	0	0																
(hex):	50	26	45	20	4D	49	43	52	4F	2D	54	45	53	54	31	2D	30	30	30	30																

Char Test (TYPE):

(ascii):

Upper Bound (hi hex):

Optional (hi ascii):

Range-1:

Restriction (lo ascii):

Lower Bound (lo hex):

Upper Bound (hi hex):

Optional (hi ascii):

Range-2:

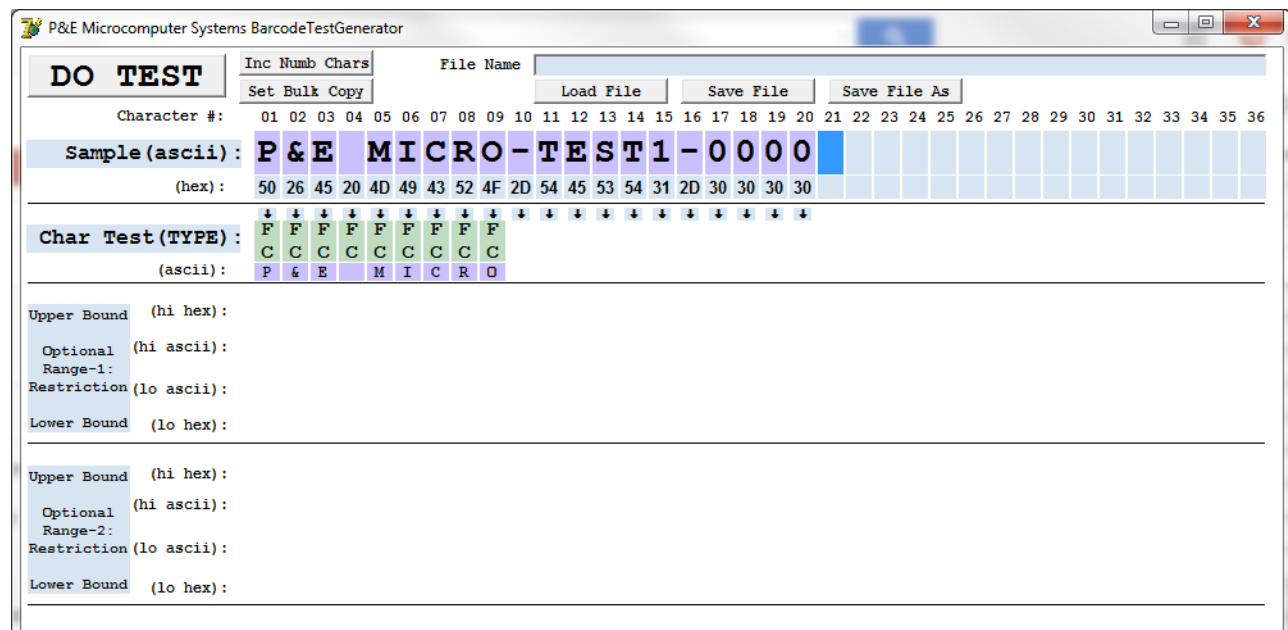
Restriction (lo ascii):

Lower Bound (lo hex):

**Figure 13-3: Create Prototype Barcode**

In this example, the "P&E MICRO" portion of the prototype barcode exists in every bar code the company produces - in this case, header characters that identify the company. Also for this example, a product identifier "TEST1" is in the bar code. Following the identifier, this product also has a 4-digit serial number. This completes the prototype bar code which is representative of this products barcodes.

The next step is to use this prototype barcode to create barcode tests which will be run against scanned barcodes. In this example, the first test should check the header (i.e. "P&E MICRO") at the beginning of the bar code. To enable this, the user should click the small down arrows corresponding to the header section of the barcode. This creates a character test for "P&E MICRO". This string will need to match exactly for the barcode test to pass. The "FC" designation on each character stands for "Fixed Character" with the actual value shown below it. See **Figure 13-4**.



**DO TEST**

Inc Numb Chars:  File Name:

Set Bulk Copy:

Character #:	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
Sample (ascii):	P	&	E		M	I	C	R	O	-	T	E	S	T	1	-	0	0	0	0																
(hex):	50	26	45	20	4D	49	43	52	4F	2D	54	45	53	54	31	2D	30	30	30	30																

Char Test (TYPE):

(ascii):

Upper Bound (hi hex):

Optional (hi ascii):

Range-1:

Restriction (lo ascii):

Lower Bound (lo hex):

Upper Bound (hi hex):

Optional (hi ascii):

Range-2:

Restriction (lo ascii):

Lower Bound (lo hex):

**Figure 13-4: Create Char Test For "P&E MICRO"**

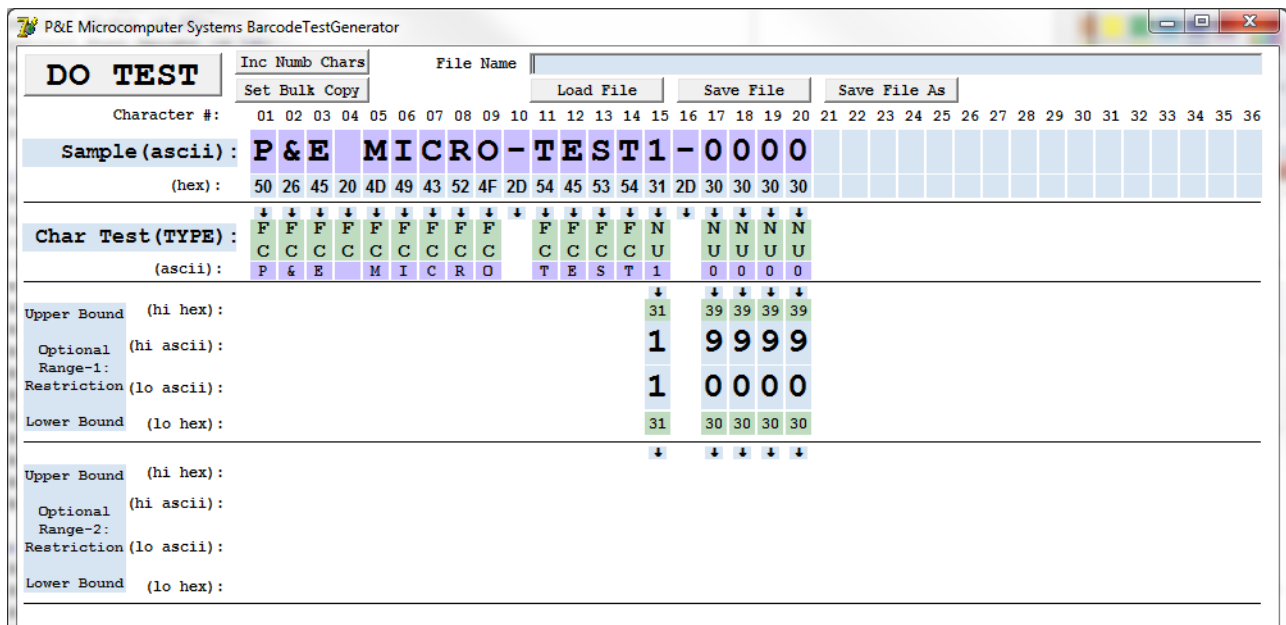
The next test will check whether the scanned barcode matches the product identifier "TEST1" exactly. The user should click the small down arrows for the "TEST1" characters to bring them into the Char test section. See Figure 5.

Sample (ascii):	P	&	E		M	I	C	R	O	-	T	E	S	T	1	-	0	0	0	0
(hex):	50	26	45	20	4D	49	43	52	4F	2D	54	45	53	54	31	2D	30	30	30	30
Char Test (TYPE):	F	F	F	F	F	F	F	F	F		F	F	F	F	N					
(ascii):	P	&	E		M	I	C	R	O		T	E	S	T	1					

Figure 13-5: Create Char Test For "TEST1"

The "1" in "TEST1" is by default treated as a "Numerical Character" for testing reasons ('NC'). This means it is required to have a value of '0'..'9'. To restrict the value of this character, it could either be change to a fixed character test or we can restrict the range of acceptable values. To do the latter, click on the down arrow below the character test and bring the "1" character into the Range-1 test area. Both the lower and upper bounds for this character can then be changed to "1" so that the match must be exactly the number 1. Refer to Figure 6.

The final step is to create a test for the 4-digit serial number part of the prototype. In this example only 501 units of this product will be manufactured which correspond to a specific programming image, each with a unique serial number from 0000 to 0500. The test should make sure that this limit is not exceeded. Therefore, the user should add "0000" to the Char Test by clicking on the small down arrows for the characters. This should then be converted this into a Range test by clicking on the new arrows to add these four characters to the Range-1 test area.

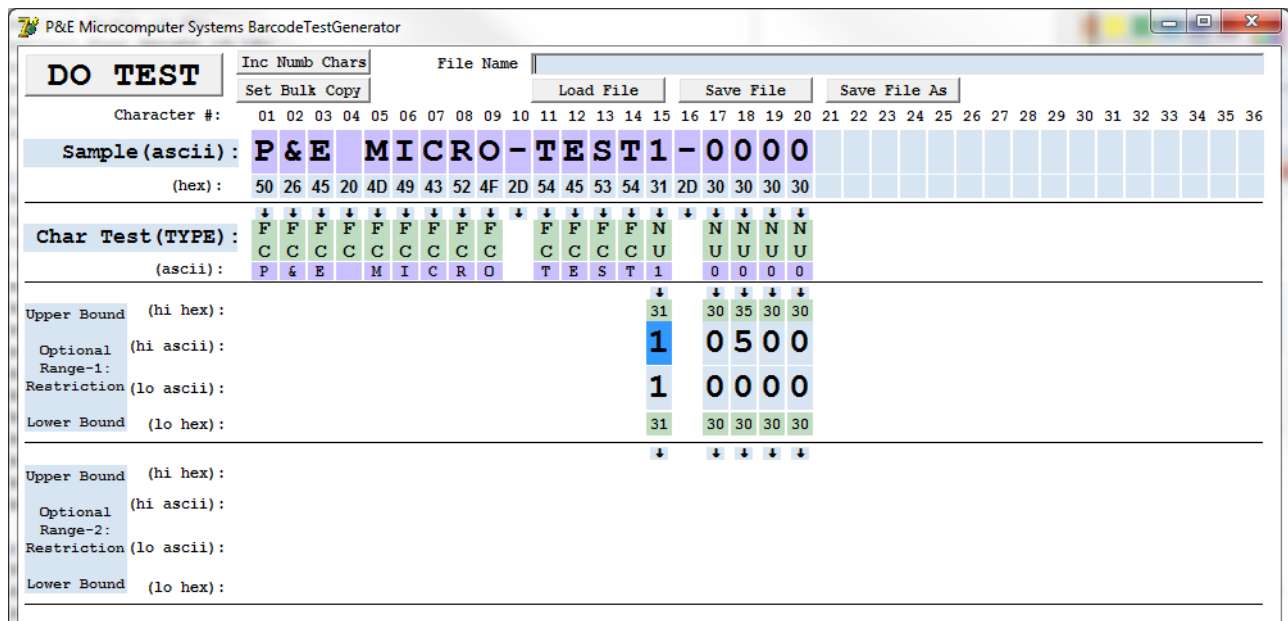


Character #:	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
Sample (ascii):	P	&	E		M	I	C	R	O	-	T	E	S	T	1	-	0	0	0	0																
(hex):	50	26	45	20	4D	49	43	52	4F	2D	54	45	53	54	31	2D	30	30	30	30																
Char Test (TYPE):	F	F	F	F	F	F	F	F	F		F	F	F	F	N		N	N	N	N																
(ascii):	P	&	E		M	I	C	R	O		T	E	S	T	1		0	0	0	0																
Upper Bound (hi hex):															31		39	39	39	39																
Optional Range-1: Restriction (lo ascii):															1		9	9	9	9																
Lower Bound (lo hex):															31		30	30	30	30																

Figure 13-6: Create Range-1 Test

The default for the range test is 0000 to 9999, so after these numerals enter the Range-1 test area, the user should change the upper bound from 9999 to 0500. Any continuous numerical digits are considered to be one numerical value for the purpose of range comparison.

The result in the Barcode Test Generator utility should look like Figure 7:



**DO TEST**

Inc Numb Chars: 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36

File Name: \_\_\_\_\_

Load File Save File Save File As

Character #: 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36

Sample(ascii): P & E MICRO-TEST1-0000

(hex): 50 26 45 20 4D 49 43 52 4F 2D 54 45 53 54 31 2D 30 30 30 30

Char Test(TYPE):

F	F	F	F	F	F	F	F	F	F	F	F	F	F	N	N	N	N	N	N
C	C	C	C	C	C	C	C	C	C	C	C	C	C	U	U	U	U	U	U
P	&	E		M	I	C	R	O		T	E	S	T	1		0	0	0	0

Upper Bound (hi hex): 31 30 35 30 30

Optional Range-1: Restriction (lo ascii): 1 0000

Lower Bound (lo hex): 31 30 30 30 30

Upper Bound (hi hex):

Optional Range-2: Restriction (lo ascii):

Lower Bound (lo hex):

**Figure 13-7: Completed Barcode Test**

This completes setup of the barcode test. The user should now click on “Save File As” on top and name the test “Test1\_barcode.test.bar”. This test is now ready to be added to a programming image using the Cyclone Image Creation utility.

## 13.6 Creating a Barcode Test: In Depth

This section is an in-depth exploration of how to use the Barcode Test Generator utility to create a barcode test. For a quick example of how to create a barcode test, please refer to **Section 13.5 - Creating A Barcode Test: Quick Example**.

The ability to select a programming image and initiate programming via a barcode scanner requires that a user generate a Barcode Test that will be included in the programming script of the SAP image. In order to create the Barcode test, PE micro supplies a Barcode Test Generator utility that facilitates creating a Barcode Test which describes the mechanism for calculating whether an input barcode meets the criterion the user is looking for to be considered a match, which can be an exact match or a match that is in a **Range** selected by the user.

The **Char Test (TYPE)** : and **Range** test fields are first chosen from selected **Sample** (ASCII) characters. It is intended for the user to enter a whole sample barcode and modify the type and to create appropriate tests, **Char Test (TYPE)** : limits the individual characters to types, such as numeric, alphabetic, hexadecimal, etc. Range tests determine if the magnitude of chosen set of Sample input characters viewed as a large number are within a lower and upper bounds. These **Range** tests are used to validate sequence numbers to be within specified bounds. For a test to be valid it must pass any **Char Test (TYPE)** : tests and at least one if any **Range** tests. The **Range** tests are actually done on a set of Sub-Ranges defined as being contiguous set of characters and having all the same **Char Test (TYPE)** : **Figure 13-1** shows a typical main screen for the Barcode Test Generator utility.

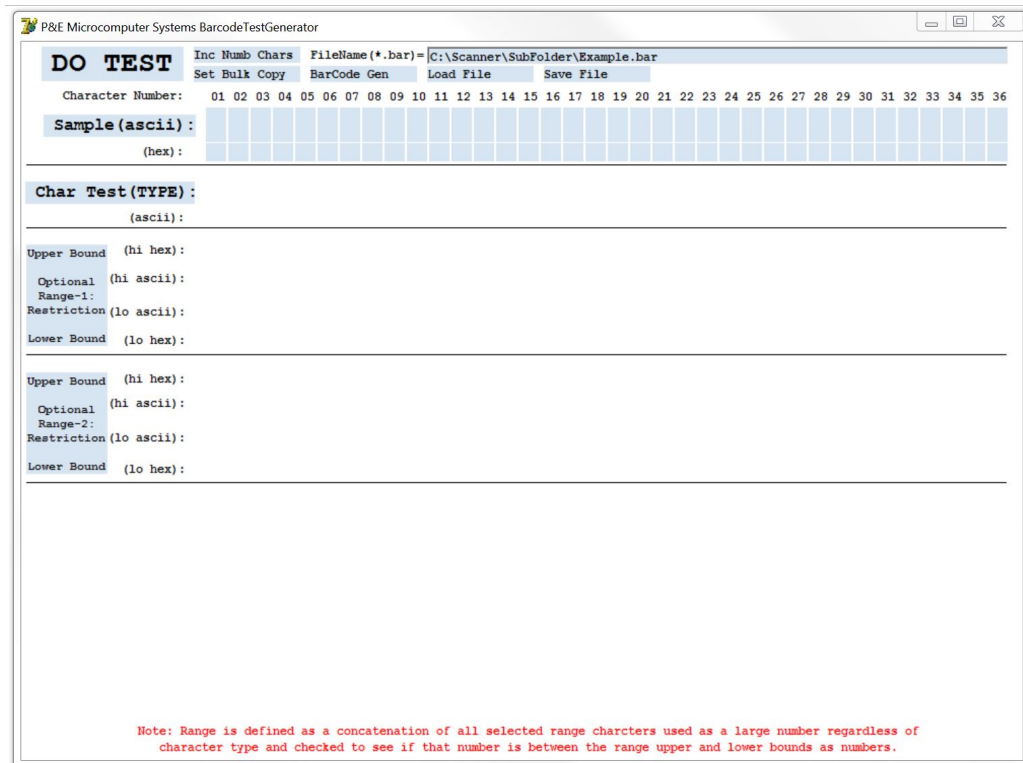


Figure 13-8: Typical Barcode Test Generator Main Screen On Startup

### 13.6.1 Barcode Test Creation and Testing Process

A Barcode Test can be easily created and tested using the following 8 simple steps which are explained in paragraphs following the list below:

1. Create a **Sample (ascii) / (hex)** Barcode
2. Select Characters to be tested by copying them to **Char Test (TYPE) :**.
3. Set desired type of selected characters from step 2.
4. Copy character types from step 3 to **Range-1** or **Range-2** if Range testing.
5. Adjust range character(s) minimums and maximums of character of Step 4.
6. Press the **DO TEST** button to test the Sample against test setup.
7. Make changes to **Sample**, **Char Test (TYPE)**, **Range-1**, or **Range-2** and repeat steps 6 and 7 until an acceptable test has been created.
8. Save the test to a Barcode Test (.bar) file for later use.

The **Sample** character set is used first as a structural frame for creating a Barcode test and then later as an input field for entering multiple barcode samples to verify the Barcode test works as expected.

#### 13.6.1.1 Enter a Sample(ascii)/(hex)

**Sample** characters can be entered either as ascii characters or hexadecimal bytes (2 characters per byte) and in many ways. To start the process, click on either a **Sample (ascii)** box or a **Sample (hex)** box. This makes the box **darker blue** to indicate where the entry cursor is. Once the entry cursor is set characters can be entered into the **Sample** by several methods listed below. When entries are placed in an **Sample (ascii)** box they are also updated in the corresponding **Sample (hex)** box and vice versa:



1. By simply typing in ascii characters or pairs of hexadecimal characters.
2. By copying a string of highlighted characters by Ctrl-C / Ctrl-V method.
3. By scanning in a string of characters starting at the current cursor position.

Character Number:	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
Sample(ascii):	P	&	E		C	y	c	l	o	n	e	-	F	X		S	N	:	0	1	2	3	4		R	e	v	-	B							
(hex):	50	26	45	20	43	79	63	6C	6F	6E	65	2D	46	58	20	53	4E	3A	30	31	32	33	34	20	52	65	76	2D	42							
	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+

**Figure 13-9: Typical Sample Entry Screen Section**

**Figure 13-1** shows a typical entry screen section. The first line gives the character numbers. The second line shows entered characters and un-entered characters in ASCII. The third line shows the entered characters in hexadecimal, and the current cursor position. The last line shows a set of down arrows which when clicked on selects the characters to be tested.

Sample characters are deleted by double clicking on a **Sample (ascii)** or **Sample (hex)** box. For convenience, the entire set of **Sample** characters can be deleted by double clicking on the left screen **Sample (ascii):** button. As characters are entered, the cursor automatically moves to the right stopping at the last Sample entry box. To add additional character boxes, see section 6.2.

When a characters are entered into a **Sample (ascii)** box, the background color of the box is changed to **Light Purple**. This is done to highlight the difference between a space character and un-entered boxes. At the same time, a down arrow is placed under the **Sample (hex)** box to indicate that the Sample character can be transferred to a **Char Test (Type):** field by clicking on the down arrow.

#### 13.6.1.1.4 Sample entry by typing in Characters

If the cursor is in a **Sample (ascii)** box typing in characters will place them in that box. The cursor then automatically moved to the next **Sample (ascii)** box. All ASCII characters including control codes can be entered in **Sample (ascii)** boxes. However, if there are characters stored on the desktop, a Ctrl-V will enter them rather than entering a Ctrl-V character. After two **Sample (hex)** box character entries, the cursor automatically proceeds to the next **Sample (hex)** box. Only the hexadecimal characters, (0-9, A-F, a-f), are allowed in a **Sample (hex)** boxes. Hex values are entered high nibble and then the low nibble.

#### 13.6.1.1.5 Sample entry by copying in a string of highlighted characters

A string of characters can be entered into a series of **Sample (ascii)** boxes starting at the cursor by first highlighting them by a mouse down pass over of the characters, then doing a Ctrl-C and a Ctrl-V. This is a standard Windows System process of placing a string on the desktop and then inserting it somewhere else. However, you cannot do this with characters within the utilities window.

#### 13.6.1.1.6 Sample entry by scanning in a string of characters

Most barcode scanners can place the scanned characters into the Windows keyboard buffer. Hence, they are automatically entered into a sequence of **Sample (ascii)** boxes starting at the cursor. Make sure when setting up a scanner that it is configured to place the scanned characters in the keyboard buffer.

#### 13.6.1.1.7 Deleting an entry

The contents of an entry box may be deleted by double-clicking on the entry box.

#### 13.6.1.2 Copy Characters to be tested to Char Test(TYPE):

When a barcode is scanned during production programming, it is often the case that all the characters do not need to be tested to determine the appropriate standalone programming (SAP) image to be used. Hence, it is necessary to select and copy only those **Sample** characters to be



tested.

Character Number:	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
Sample(ascii):	P	&	E		C	y	c	l	o	n	e	-	F	X		S	N	:	0	1	2	3	4		R	e	v	-	B							
(hex):	50	26	45	20	43	79	63	6C	6F	6E	65	2D	46	58	20	53	4E	3A	30	31	32	33	34	20	52	65	76	2D	42							
Char Test (TYPE):					F	F	F	F	F	F	F	F	F	F		N	N	N	N	N	N	N	N													
(ascii):					C	y	c	l	o	n	e	-	F	X		0	1	2	3	4																

**Figure 13-10: Sample Characters copied to Char Test Section**

Simply clicking on the down arrows between the **Sample** and **Char Test (TYPE)** : Sections of the screen transfers the corresponding character to the test section. The transferred characters are shown as ASCII characters along with an **associated type**. The ASCII characters in the Char Test Section represent transferred value and do not change when the Sample characters are edited. When transferred from the Sample, the associate type is automatically set to either Fixed Character (**FC**) type or if ascii characters (0-9) to NUmeric (**NU**) type. All possible character types are defined in **Section 13.6.2 - Character Types**. Any character types which are represented by a contiguous range of values are indicated by a set of down arrows into the Optional Range Restriction Sections of the display.

**Figure 13-10** shows that only the product name “Cyclone-FX”, the serial number (SN) “01234”, and the revision (Rev-) “B” were selected for character testing. The manufacturer “P&E”, blank spaces, and other characters are ignored during character testing.

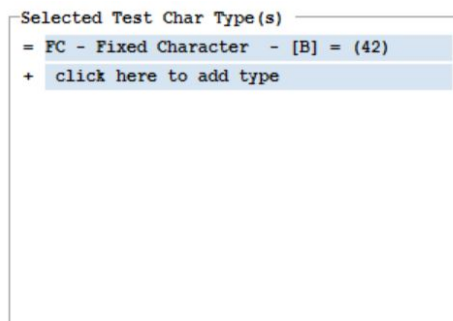
**Note:** Each of the fixed characters in ‘Cyclone-FX’ and ‘B’ require an exact match, while each number in ‘01234’ only has to be in the **NU** range of 0-9 to be a match.

All entries in the Char Test field can be deleted by double clicking the **Char Test (TYPE)** : button. Individual Char Test entries can be deleted by double clicking anywhere on them.

Often setting just the **Char Test (TYPE)** : of tested characters to Numeric (**NU**) or Fixed Character (**FC**) and setting the appropriate ascii value for Fixed Character (**FC**) fields is sufficient to generate a barcode test (although the Barcode Test Generation Utility also allows specification of fairly complex barcodes as well).

### 13.6.1.3 Set desired type of selected characters from step 2.2

Once a Char Test characters have been selected, they can be changed to different types than were transferred from the Sample. An individual associated type is edited by clicking on it to set the cursor position. For example, clicking on the Char Test(TYPE): “B” character type opens the window in Figure 2.3a:

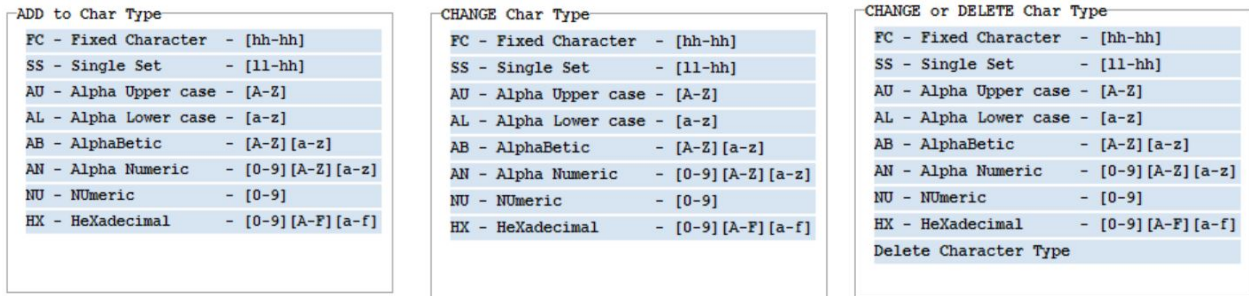


**Figure 13-11: Selected Test Char Type(s) Window**

#### 13.6.1.3.1 Adding, Changing and Deleting **Char Test (TYPE)**:

This window shows all the character types associated with the selected **Char Test** character. Shown types can be changed or deleted by clicking on them or types can be added by or deleted

clicking on “click here to add type.” If clicked, one of the similar windows shown in **Figure 13-12.a**, **Figure 13-12.b**, or **Figure 13-12.c** is opened. Types can also be efficiently copied to other blocks as per **Section 13.6.6.2 - Ctrl-C / Ctrl-V Char Test(TYPE): and Range Copying**.

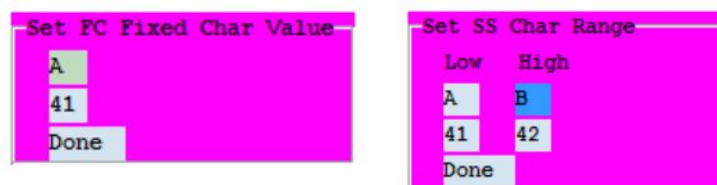


**Figure 13-12: a) On Click Add Figure; b) On Click Type Figure; c) On Click Type (multiple types)**

Clicking on one of the opened window selections causes that option to be added, changed or deleted.

### 13.6.1.3.2 Fixed Character (FC) and Single Set (SS) Sub-Types

Choosing change either **FC** or **SS** type will open respectively the following windows to allow entry of the numeric values of these types.



**Figure 13-13: a) FC Value; b) SS Values**

The **FC** type choice, **Figure 13-13.a**, only requires one value to be entered since a Fixed Character (**FC**) has the same upper and lower values. The Single Set (**SS**) type choice, **Figure 13-13.b**, represents a single contiguous set of any length and hence requires both lower and upper bound values. These values can be entered either as ASCII characters or pairs of hexadecimal digits. The fuchsia color indicates that some action must be taken before proceeding.

### 13.6.1.3.3 Multiple Sub-Types

A given Char Test character can have up to a maximum of 10 sub-types. However, if one selects either numerically adjoining or overlapping sub-type sets, the program automatically compresses the sub-types.

For this example, the “Rev-“ **B** Char Test (TYPE) : was set to allow the characters “A”, “B”, “D”, or “E”. This is shown in **Figure 13-14.a** as two **SS** sub-types. This could have been done by adding three **FC** types for “A”, “D”, and “E” or two **SS** types [“A”-“B”] and [“D”-“E”]. However, the program would compress the **FC** types to **SS** types as shown. As can be seen in **Figure 13-14.b**, the “Rev-“ Char Test (TYPE) : information was changed from an **FC** type to a Multiple (**MT**) type.

180

Char Test (TYPE):	F F F F F F F F F	N N N N N	M
(ascii):	C C C C C C C C C	U U U U U	T
	C y c l o n e - F X	0 1 2 3 4	B
Upper Bound (hi hex):		39 39 39 39 39	42
Optional (hi ascii):		9 9 9 9 9	B
Range-1:			
Restriction (lo ascii):		0 0 0 0 0	B
Lower Bound (lo hex):		30 30 30 30 30	42

Figure 13-16: Transfers from Char Test to Range-1

In **Figure 13-16**, the NUmeric (NU) characters were transferred to the Range-1 test using the down Arrows. Only types with contiguous character sets can be transferred this way. When contiguous types are transferred, the upper and lower bounds are set to the types maximum and minimum values for convenience. However, it is easy to change them later. The MT type was transferred using the bulk copy method of section 6.1 since it is non-contiguous. Because of this, care must be taken about missing value in the Range-1 specified. When non-contiguous sets are transferred, the upper and lower bounds are both set the Char Test (ascii) character value. Such values can be easily changed later.

### 13.6.1.5 Adjust Range Character Upper and Lower Bounds

To change upper and lower bound values, first click on either a bound (ascii) box or a bound (hex) box to set the cursor there to that box. Typing an ascii character or a pair of hexadecimal characters enters them into the appropriate box. Then the cursor is moved to the right one block, circularly over the set of range blocks. Pairs of upper and lower bounds can also be copied to other pairs as describer in Section 6.2. All entries in a range can be deleted by double clicking the Range-1 or the Range-2 buttons. Individual range entries can be deleted by double clicking anywhere on them.

Upper Bound (hi hex):	30 35 34 33 36	42
Optional (hi ascii):	0 5 4 3 6	B
Range-1:		
Restriction (lo ascii):	0 0 0 0 0	A
Lower Bound (lo hex):	30 30 30 30 30	41
Upper Bound (hi hex):	39 30 30 30	45
Optional (hi ascii):	9 0 0 0	E
Range-2:		
Restriction (lo ascii):	0 0 0 0	D
Lower Bound (lo hex):	30 30 30 30	44

Figure 13-17: Adjusted Range-1 and Range-2 Bounds

Notice in **Figure 13-17**, that Range-1 numeric values run between 00000 and 05436. In addition, for Range-1, the lone MT type was adjusted to use only a contiguous range of "A" to "B". Range-1 was adjusted to have the numeric values between 0000 and 9000. Also, its lone MT type was set to a contiguous range of "D"-"E". These choices created only contiguous ranges, i.e. no holes in a range. Ranges which are non-contiguous are allowed if necessary but generate confusing results.

When creating range entries, the Upper Bound should be greater than or equal to the lower bound. If the bounds are not entered that way, an error is generated as shown below in **Figure 13-18** by changing the background color of the appropriate Range button(s) and the bound(s) entry to fuchsia. This can be fixed by swapping the upper and lower bounds.\

Upper Bound (hi hex):	30 35 30 33 36	42
Optional (hi ascii):	0 5 0 3 6	B
Range-1:		
Restriction (lo ascii):	0 0 4 0 0	A
Lower Bound (lo hex):	30 30 34 30 30	41

Figure 13-18: Bounds Entry Error





DO TEST		Inc Numb	Chars	FileName(*.bar)=
Set	Bulk	Copy	BarCode	Gen
Load File	Save File			
Character Number: 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36				
Sample(ascii): P & E Cyclone SN: 09000 Rev-F				
(hex): 50 26 45 20 43 79 63 6C 6F 6E 65 20 20 20 20 53 4E 3A 30 39 30 30 30 20 52 65 76 2D 46				
Char Test(TYPE):				
(ascii):				
Upper Bound (hi hex):				
Optional Range-1: Restriction (lo ascii):				
Lower Bound (lo hex):				
Upper Bound (hi hex):				
Optional Range-2: Restriction (lo ascii):				
Lower Bound (lo hex):				

**Figure 13-21: Failed Test with Multiple Failures**

Clicking on almost anything on the screen clears all error indications.

#### 13.6.1.7 Make changes to Sample(ascii), Char Test(TYPE):, Range-1 , or Range-2 & Repeat Test

This step represents the start of the iterative process used to generate an appropriate test for the product to be programmed. Try different Samples, run DO TEST, make changes to the test, etc., until a satisfactory test is found.

#### 13.6.1.8 Save the test to a Barcode Test (.bar) file for later use.

The tests created can be saved (See **Section 13.6.5 - Saving and Loading Barcode Test (.bar) Files.**) for later inclusion in a Cyclone FX programming SAP image using PEmicro's image creation utility [1]. The saved tests can also be reloaded for correction or as the base for creating other tests.

### 13.6.2 Character Types

Selected characters are tested as part of the **Char Test (TYPE)** : to see if they are included in a set of character called a type. Listed in **Table** below are the type sets allowed in this program. Most of the types have common names and abbreviated mnemonics. Some types are contiguous and others are not. Contiguous means that there is only a single lower and a single upper count for the set with no missing values. A special type called MT (Multiple Types) means that is composed of a number of sub-types.

When forming a Multiple (MT) types, the program will automatically compress either adjacent or overlapping sub-types if possible as they are created. This optimization makes for a more efficient type testing process. In addition, when creating Fixed Characters (FC) or Single Set (SS) types, the program asks for the lower or upper bounds for these sets either as ASCII characters or pairs of hexadecimal digits. FC is the same as an SS with a bounds range of a single character, i.e., the lower and upper bounds have the same value.

**Table M-1. Allowed Character Types**

Type	Type Name	Ascii Range	Hex Range	Contiguous
F C	Fixed Character	selected	[hh-hh]	Y
S S	Single Set	selected	[ll-hh]	Y













**Table M-1. Allowed Character Types**

Type	Type Name	Ascii Range	Hex Range	Contiguous
A U	<b>Al</b> phabetic <b>U</b> pper case	[A-Z]	[41-5A]	Y
A L	<b>Al</b> phabetic <b>L</b> ower case	[a-z]	[61-7A]	Y
A B	<b>Al</b> pha <b>B</b> etic	[A-Z][a-z]	[41-5A] [61-7A]	N
A N	<b>Al</b> pha <b>N</b> umeric	[0-9][A-Z][a-z]	[30-39] [41-5A] [61-7A]	N
N U	<b>NU</b> meric	[0-9]	[30-39]	Y
H X	<b>HeX</b> adecimal	[0-9][A-F][a-f]	[30-39] [41-46] [61-66]	N
M T	<b>Mul</b> Tiple	Multiples of above	Multiples of above	N

### 13.6.2.1 Background Color Codes

**Figure 13-22** shows what the color codes typically represent on the program screen:

		or		Clicked to Select or Double-Clicked to delete
				Indicates Selected ASCII character values
				Indicates Selected hex character values
				Indicates a Failure
				Indicates a Success
				Needs attention to continue or indicates inverted bounds
				Marked by Ctrl-C for Ctrl-V copying
				Indicates the Cursor position

**Figure 13-22: Background Color Codes**

### 13.6.2.2 Font Color Codes

**Figure 13-23** shows what the font colors represent on the program screen:

<b>XXX</b>	Normal character font color
<b>XXX</b>	Font color selected for Bulk Copying, or to indicate an Option Open Option

**Figure 13-23: Font Color Codes**

## 13.6.3 Enabling/Disabling A Barcode Scanner

To enable a barcode scanner that is connected to the Cyclone **FX**'s USB Extension Port, go to:

*Menu -> Configure Cyclone -> Configure USB Host Device -> Enable USB Scanner*

This setting is persistent; the scanner will be powered and turned on when the Cyclone is powered, until the scanner is disabled. To disable the barcode scanner, select:

*Menu -> Configure Cyclone -> Configure USB Host Device -> Disable*

## 13.6.4 Ranges and Range Testing

Testing is done in two steps First, if any **Char Test (TYPE)** : characters are specified a Character Type test is done as indicated in **Section 13.6.4.1 - Character Type Testing**. Then, if either **Range-1** or **Range-2** has entries, a Range test is done as indicate in **Section 13.6.4.2 - Range Testing**.

To pass testing, the **Char Type (TYPE)** : test must pass if it exists. In addition, at least one Range Test must pass if any exist, If not, errors are indicated on the programs main screen.

### 13.6.4.1 Character Type Testing

Each character in **Sample** which has a corresponding **Char Test (TYPE)** : is tested numerically to see if it is within one of the sets for the type specified. Types can have multiple sets of characters. As long as the **Sample** character is in one of the sets, the test is valid for that character. If any individual **Sample** character test fails, the overall test is failed.

### 13.6.4.2 Range Testing

**Ranges** are subdivided into Sub-Ranges which are delineated by contiguous sets of identical **Char Test (TYPE)** : s. A **Range** test consists of testing each Sub-Range. A **Range** test passes if all of its' Sub-Range tests pass. Sub-Range testing checks to see if the selected set of Sub-Range **Sample** characters treated as a large number fits between a lower and an upper bound. The bounds are formed by numerically concatenating the selected Sub-Range bounds boxes, to form a number representing lower and upper bounds for that Sub-Range.

Sub-Range bounds entry boxes are actually tested against their corresponding **Sample** characters on a character by character basis from left to right until all the individual lower and upper bounds for a given **Range** are tested:

On lower bounds, starting at the left, if the corresponding Sample character is:

- |                                 |   |
|---------------------------------|---|
| (a) lower than its lower bound, | lower bound Sub-Range test failed, do next Sub-Range  |
| (b) equal to its lower bound,   | next Sub-Range character to the right must be tested  |
| (c) else,                       | lower bound Sub-Range test passed, do next Sub-Range. |

On upper bounds, starting at the left, if the corresponding Sample character is:

- |                                   |   |
|-----------------------------------|---|
| (a) greater than its upper bound, | upper bound Sub-Range test failed, do next Sub-Range  |
| (b) equal to its upper bound,     | next Sub-Range character to the right must be tested  |
| (c) else,                         | upper bound Sub-Range test passed, do next Sub-Range. |

This process allows the exact character in the range sequence that caused a range test error if any. This also means that the range is tested as a large concatenated number composed of numerical characters. The test is performed on a character by character basis.

## 13.6.5 Saving and Loading Barcode Test (.bar) Files

### 13.6.5.1 5.1 Barcode Test (.bar) File Structure

PEmicro Barcode Test Data is stored in JSON Data Interchange Format files [2]. The information in an empty PEmicro JSON Barcode Test file is shown below where blue characters represent the formatting information and red characters represent what data is inserted in the file by the Barcode Test Generator. All data is in hexadecimal to make the files readable since the data represented can be nonprintable ascii characters. This empty PEmicro JSON Barcode Test file has been reformatted, spacing wise, to make it more readable. There are many sites on the internet which will reformat JSON files [3].

```
{
  "pesettings": {
    "scannertest": {
      "testchars": " ... hex pairs of character test information ... ",
      "range1": " ... hex pairs of range1 lower and upper bounds ... ",
      "range2": " ... hex pairs of range2 lower and upper bounds ... "
    },
    "scannersave": {
      "samplechars": " ... hex pairs of sample data – not used by Cyclone FX ... ",
      "testcharstrings": " ... hex pairs of test data – not used by Cyclone FX ... "
    }
  }
}
```

The unformatted version of an empty PEmicro JSON Barcode Test file looks like:

```
{"pesettings":{"scannertest":{"testchars":"","range1":"","range2":""},"scannersave":{"samplechars":"","testcharstrings":""}}}
```

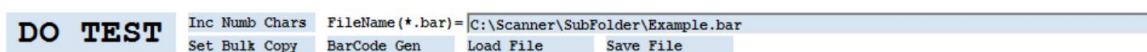
The unformatted version of the PEmicro JSON Barcode Test file for the examples used in this manual looks like:

```
{"pesettings":{"scannertest":{"testchars":"05014343060179790701636308016C6C09016F6F0A016E6E0B0165650C012D2D0D0146460E01585813013039140130391501303916013039170130391D0241424445","range1":"9330301430359530341630339730369D4142","range2":"9430399530309630309730309D4445"},"scannersave":{"samplechars":"0150022603450420054306790763086C096F0A6E0B650C2D0D460E580F201053114E123A13301431153216331734182019521A651B761C2D1D42","testcharstrings":"054346430301434306794643030179790763464303016363086C464303016C6C096F464303016F6F0A6E464303016E6E0B654643030165650C2D464303012D2D0D464643030146460E5846430301585813304E550307303914314E550307303915324E550307303916334E550307303917344E55030730391D424D5406024142024445"}}}
```

### 13.6.5.2 Saving a Barcode Test (.bar) File

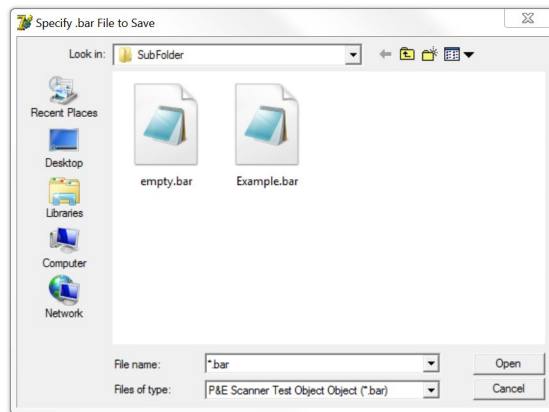
The current information on the program main screen can be saved to a Barcode Test (.bar) file for two reasons. First, so the tests can be used by PEmicro's Cyclone FX, Standalone programmer. Second, so the information can be loaded back into this program for modification, further testing of the tests themselves, or to assist in the creation of new tests.

To save a Barcode Test (.bar) file, click on the **Save File** button in the control area of the main screens.



**Figure 13-24: Program Control Area**

If a filename (editable) exists in the control area, the program tries to save the screen test information to that file. If no filename exists, then the window shown below is opened to search for and/or edit a filename. If Open is clicked, the program attempts to save the test information to that file. On attesting to save a file, the button is recolored respectively on Success or Failure to **Save File** or **Save File**.

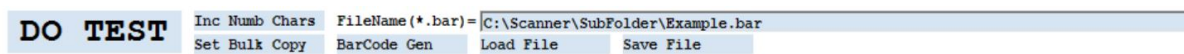


**Figure 13-25: Save PEMicro JSON File Window**

### 13.6.5.3 Loading a Barcode Test (.bar) File

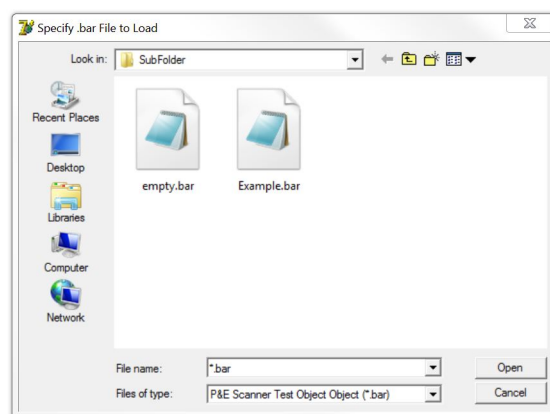
PEmicro JSON Barcode Test (.bar) Information can be loaded into this utility so it can be edited further or to test other barcodes.

To load a Barcode Test (.bar) file, click on the **Load File** button in the control area of the main screen.



**Figure 13-26: Program Control Area**

If a filename (editable) exists in the control area, the program tries to load test information from that file. If no filename exists, then the window shown below is opened to search for and/or edit a filename. If Open is clicked, the program attempts to load the test information from that file. On attempting to load a file, the button is recolored respectively on Success or Failure to Load File or LoadFile.



**Figure 13-27: Load PEMicro JSON File Window**

## 13.6.6 Advanced Program Use

### 13.6.6.1 Bulk Copy Option

Often, it is required to make many transfers from **Sample** characters to **Char Test (TYPE)** : entries or from **Char Test (TYPE)** : entries to **Range-1** or **Range-2**. These transfers are usually done by individually clicking on appropriate down arrows. For large barcodes, this can be very time consuming. With Bulk Copy, Sample characters can be selected / deselected by a mouse entry

into a Sample box. Characters which are selected are indicated by changing the font color to red. Once a set of characters has been selected, they are copied into **Char Test (TYPE) :** entries by clicking on the **Char Test (TYPE) :** button. Bulk Copy turns of the selected character red font highlighting.

In a similar manner, **Char Test (TYPE) :** entries can be highlighted and copied into **Range-1** or **Range-2** entries.

This process takes a little practice to get used to but greatly speeds things up. In addition, this process can be used to copy entries which do not have down arrows assigned to them.

### 13.6.6.1.1 Turning On Bulk Copy

Clicking on the black text **Set Bulk Copy** button in the program control area enables Bulk Copying. When the Bulk Copy option is selected, the text in the **Set Bulk Copy** button is recolored red as a reminder to turn the option off by clicking the button again.

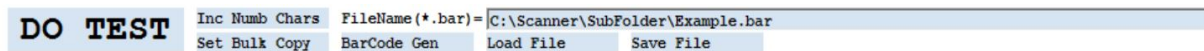


Figure 13-28: Program Control Area

### 13.6.6.1.2 Copy Sample Boxes to Char Test

Figure 13-29 shows bulk copy selected **Sample** characters selected as indicated by the red type fonts. After clicking on the **Char Test (TYPE) :** button the **Sample** characters are transferred as shown in Figure 13-30. Notice that the **Sample** highlighting is turned off after the copy.

Character Number:	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
Sample(ascii):	P	&	E			C	y	c	l	o	n	e	-	F	X		S	N	:	0	1	2	3	4		R	e	v	-	B						
(hex):	50	26	45	20	43	79	63	6C	6F	6E	65	2D	46	58	20	53	4E	3A	30	31	32	33	34	20	52	65	76	2D	42							
Char Test(TYPE):						F	F	F	F	F	F	F	F	F																						
(ascii):						C	C	C	C	C	C	C	C	C																						
						C	y	c	l	o	n	e	-	F	X																					

Figure 13-29: Highlighted Sample entries for Bulk Copy to **Char Test (TYPE):** entries

Character Number:	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
Sample(ascii):	P	&	E			C	y	c	l	o	n	e	-	F	X		S	N	:	0	1	2	3	4		R	e	v	-	B						
(hex):	50	26	45	20	43	79	63	6C	6F	6E	65	2D	46	58	20	53	4E	3A	30	31	32	33	34	20	52	65	76	2D	42							
Char Test(TYPE):	F	F	F		F	F	F	F	F	F	F	F	F	F																						
(ascii):	C	C	C		C	C	C	C	C	C	C	C	C	C																						
	P	&	E		C	y	c	l	o	n	e	-	F	X																						

Figure 13-30: After Sample entries Bulk Copy to **Char Test (TYPE):** entries

### 13.6.6.1.3 Copy Char Test Boxes to a Range

Figure 6.1.3a shows bulk copy **Char Test (TYPE) :** entries selected as indicated by the red type fonts. After clicking on the **Range-1 :** button the **Char Test (TYPE) :** entries are transferred as shown in Figure 6.1.3b. Note that **Range-2** entries did not change. However, **Char Test (TYPE) :** can be bulk copied to **Range-2** in a similar manner. Notice that the **Char Test (TYPE) :** highlighting is turned off after the copy.



[illegible]

**Figure 13-31: Highlighted Char Test(TYPE): entries for Bulk Copy to Range entries**

[illegible]

**Figure 13-32: After Char Test(TYPE): entries Bulk Copy to Range-1 entries**

#### 13.6.6.1.4 Turning Off Bulk Copy

Clicking on the **red** text **Set Bulk Copy** button in the program control area disables Bulk Copying. When the Bulk Copy option is deselected, the text in the **Set Bulk Copy** button is recolored to black.

### 13.6.6.2 Ctrl-C / Ctrl-V Char Test(TYPE): and Range Copying

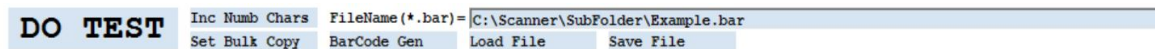
This type of copying is used when it is desired to create multiples of the same **Char** **Test(TYPE) :**, **Range-1** or **Range-2** entries. Once an entry has been created in a field, it can be designated as the copy entry that field. To designate an entry, place the cursor on the entry and type a Ctrl-C. The background color for that entry will change to **yellow** to indicate the designation. Note that only entries with **light green boxes** can be Ctrl-C designated since ASCII entry boxes would simply accept the Ctrl-C character. To designate a different entry, simply redo the designation process on a different entry.

The designated entry in a given field can then be copied into other existing entries in the same field. To accomplish this, place the cursor on an existing entry in the same field. The information in the designated Ctrl-C entry is copied to the cursor location by pressing Ctrl-V. Only cursor entries with **light green boxes** can be Ctrl-V copied to since ASCII entry boxes would simply accept the Ctrl-V character. After copying, the cursor is automatically moved right circularly to the next available entry box.

### 13.6.6.3 Adding Sample Character Spaces

On startup, this utility is configured to allow for up to 36 character spaces. This number was chosen to fit well on most displays. This number can be increased up to 64 characters. To increase the number of display characters, click on the **Inc Numb Chars** button in the control area of the program as shown in Figure 6.3.



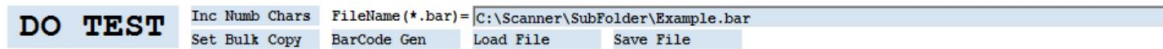


**Figure 13-33: Program Control Area**

Each time the **Inc Numb Chars** button is clicked, an additional character space is added to the display up to a maximum of 64. Also, the width of the program window is increased to accommodate the added character spaces. The number of display characters is only reset to 36 by restarting the utility.

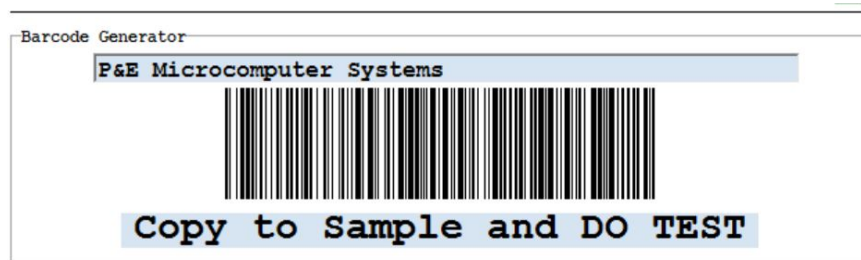
#### 13.6.6.4 On Screen Graphical Barcode Generator

Sometimes it is convenient to see what a particular Barcode looks like. To instantiate an on screen barcode display, click on the BarCode Gen button in the program control area.



**Figure 13-34: Program Control Area**

This opens a new window which has an edit box containing “**P&E Microcomputer Systems**”, the graphical barcode for this text, and a button Copy to **Sample** and **DO TEST**. Clicking into the edit box allows text to be inserted or edited. Any changes to the edit box are automatically displayed in the graphical barcode which can be scanned. An external target board barcode could be scanned and inserted using a Ctrl-V. The graphical display and the text can be observed to make sure the scanner itself is working correctly.



**Figure 13-35: Newly opened Barcode Generator Window**

Text from the edit box can then be transferred to the main utility screen **Sample** characters and tested against the tests being developed by simply clicking on the **Copy to Sample and DO TEST** button. Any prior **Sample** characters are deleted before the transfer and testing take place. When this window option is opened, the text in the **BarCode Gen** button is colored red as a reminder to turn the window off by clicking the button again.

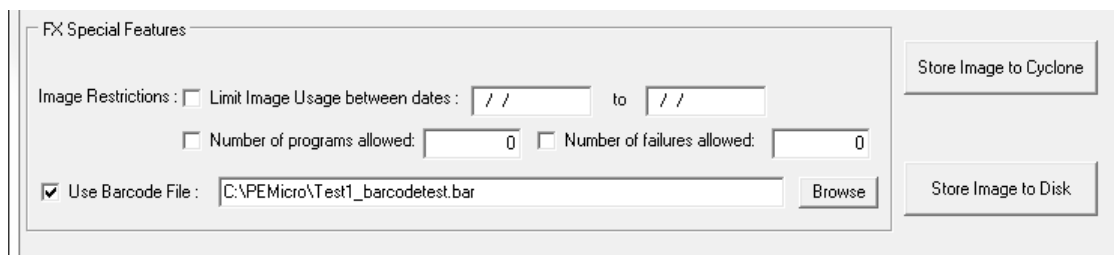
#### 13.6.7 References

- [1] Cyclone Universal and Cyclone Universal FX Manual
- [2] The JSON Data Interchange Format
- [3] JSON Re-formatter

#### 13.7 Adding A Barcode Test Into A Programming Image

Adding the barcode test into an image is done as part of the image creation process. The user should open the Cyclone Image Creation utility and follow the steps to create a programming image. The device should be selected, along with the algorithm, object file, and programming commands. This examples uses the commands EM, PM, VM.

The next step is to add the barcode file. In the “FX Special Features” section at the bottom of the utility, the user should check the “Use Barcode File” checkbox and browse for the appropriate created .bar file. See Figure 8.



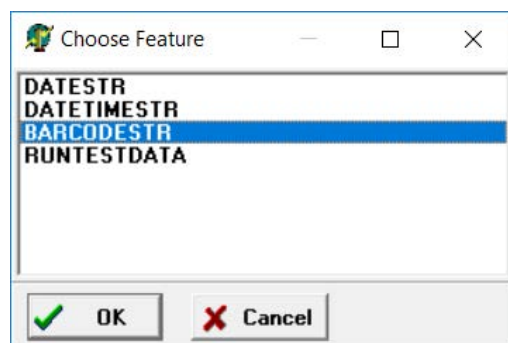
**Figure 13-36: FX Special Features (Add Barcode File To Image)**

After the programming image is stored in the **Cyclone FX**, it is ready to use. When a barcode is scanned the Cyclone will encounter this programming image, which will prompt it to test for the exact string "P&E MICRO-TEST1" and for the decimal numbers 0 to 500 on the last four characters of the barcode. If the barcode passes this test (and no other programming images on the Cyclone also pass) the image is executed.

### 13.8 Programming Barcode Information Into Memory

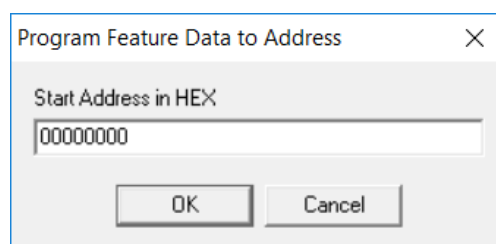
The Cyclone has the ability to program a barcode into the memory of certain ARM devices. This feature will program a null-terminated ASCII representation of the Cyclone's last recorded barcode, you can see the last recorded barcode by adding the barcode information to the homescreen (see **Section 5.2.4.4.4 - Configure Home Screen** for more information).

To program the last recorded barcode, add the "Program Feature Data" command to your programming sequence in the Image Creation Utility and select the "BARCODESTR" feature (**Figure 13-37**).



**Figure 13-37: Program Feature Data**

In the next window enter the starting address in which you want the barcode to be recorded (image 2).



**Figure 13-38: Program Feature Data to Address**

See **Section 6.2.1.6 - Programming Operations** for more information on the Program Feature Data command.

## 13.9 Troubleshooting

An error will sometimes be generated if more than one image corresponds to the barcode, or if no images correspond to the barcode. The Cyclone includes a way to quickly gain insight into the issue. A log file is created every time the barcode scanner operates and it details the scanned barcode as well as the analysis process used to select the appropriate programming image.

To access the barcode log, navigate in the **Cyclone FX's** menu to: Menu->Status->Show Logs->Show Barcode Scanner Log. This Log document contains the details of the last barcode scanner transaction and is overwritten every time the barcode scanner is used.

A sample log file looks like this:

```
---Scanner Test Log Started---
```

```
Scanned barcode: P&E SN: BLUE0138230
```

```
5 total number of images in Cyclone
```

```
Processing image number 1 (Red Image v1.00)
```

```
Image number 1 does not contain a barcode test
```

```
Processing image number 2 (Red Image v1.01)
```

```
Image number 2 does not contain a barcode test
```

```
Processing image number 3 (Widget Rev C)
```

```
Image number 3 does not contain a barcode test
```

```
Processing image number 4 (Widget Rev D)
```

```
Image number 4 does not contain a barcode test
```

```
Processing image number 5 (Green Product v7)
```

```
Image number 5 passes character test
```

```
Image number 5 does not contain a range test
```

```
image number 5 passes Barcode Test
```

```
One Image passes barcode test
```

```
Image number 5 selected to execute
```

```
Success, image selected and program command sent
```

```
---Scanner Test Log Finished---
```

## 14 AUTOMATIC SERIAL NUMBER MECHANISM

When producing a microcontroller- or microprocessor-based product, it is often useful to program a unique serial number into the permanent memory (FLASH) of the product.

PEmicro offers a serial number mechanism to automate this process. Each time you issue a serialization command in the programming software, the current serial number is programmed at a specified address. In addition, the serial number is incremented to the next available serial number and saved for future serialized programming operations.

The Cyclone adopts this automatic serial number mechanism for its stand-alone operations. The user can create a Serial File using PEmicro's Serialize Utility and then include commands in the programming sequence of the programming image to implement that Serial File. If needed, adjustments can later be made to the serial number of a programming image by using the Cyclone menu screen (see **Section 5.2.3.3 - Serial Numbers**).

### 14.1 Understanding Serialization

The automatic serial number mechanism supports serial numbers from 1 to 16 bytes in length. Each byte of a serial number ranges between a lower and an upper bound. This approach allows the individual bytes of the serial number to have distinct properties. Some of the forms these properties can take are:

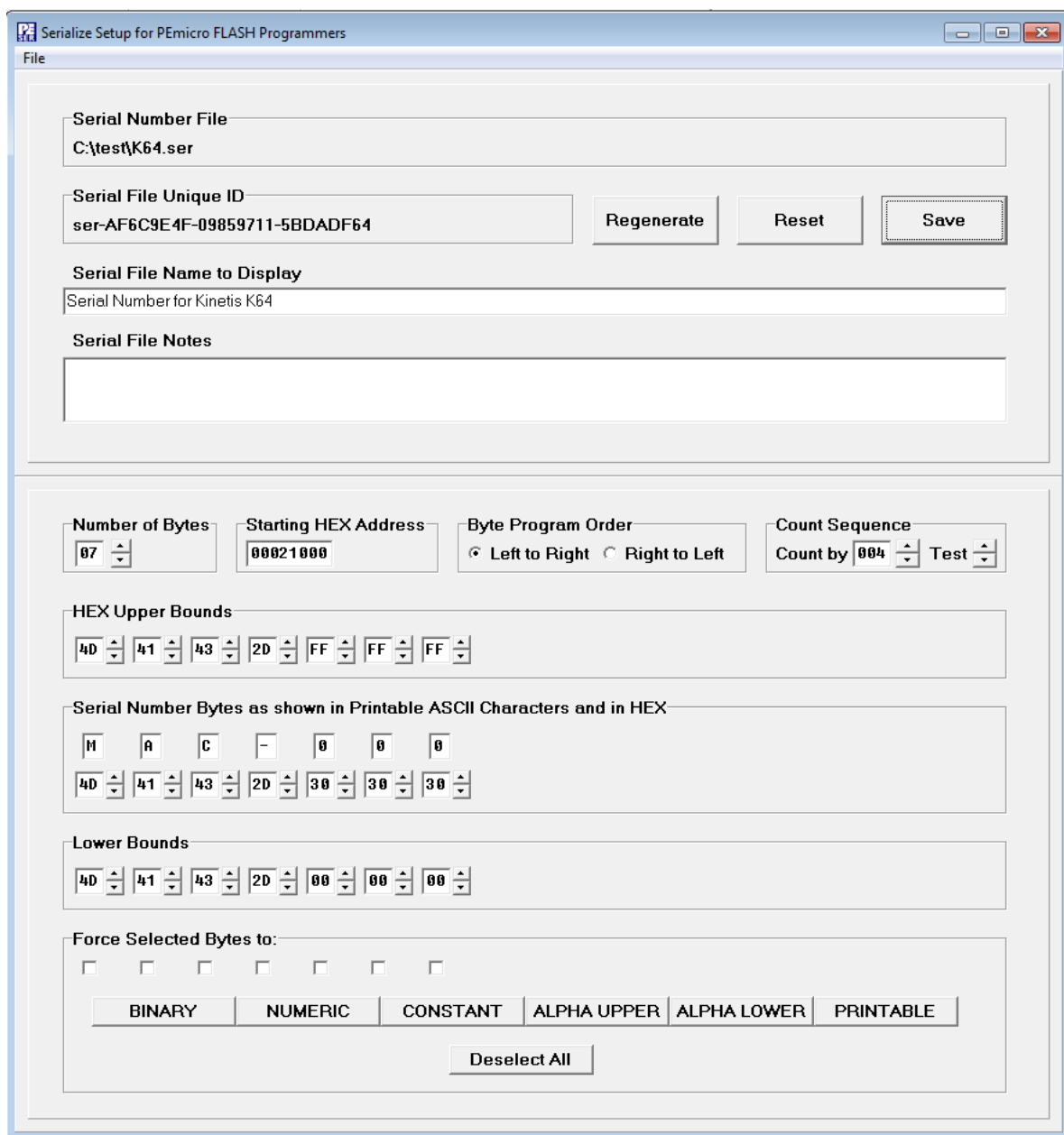
<u>Type</u>	<u>Lower Bound (hex)</u>	<u>Upper Bound (hex)</u>
Constant	Constant	Constant
Binary	00	FF
ASCII Printable	20	7E
ASCII Numeric	30	39
ASCII Upper Case Letter	41	5A
ASCII Lower Case Letter	61	7A
Other	XX	YY

Each serial number and its properties are stored in a separate file. Any file name can be used for the serial number file, however the extension .ser is normally appended because it makes it simpler to locate the file.

A utility called SERIALIZE has been developed to make it easy to create, visualize, edit, and maintain these serial number files.

### 14.2 Serialize Utility

The serialize utility allows the user to create a serial number file with a serial number that is anywhere from 1 to 16 bytes long. Each byte may be bounded as a full binary or as a Number, Constant, Alpha Uppercase, Alpha Lowercase, or ASCII printable character. The serial number file also has a field for a starting hex address that may range from 00000000 to FFFFFFFF. Serial numbers can be programmed in order or in reverse order.



The image shows the 'Serialize Setup for PEmicro FLASH Programmers' window. It has a menu bar with 'File'. The main area is divided into several sections:

- Serial Number File:** A text box containing 'C:\test\K64.ser'.
- Serial File Unique ID:** A text box containing 'ser-AF6C9E4F-09859711-5BDADF64'. To its right are three buttons: 'Regenerate', 'Reset', and 'Save'.
- Serial File Name to Display:** A text box containing 'Serial Number for Kinetis K64'.
- Serial File Notes:** A large empty text area.
- Configuration Section:**
  - Number of Bytes:** A spinner box set to '07'.
  - Starting HEX Address:** A text box containing '00021000'.
  - Byte Program Order:** Two radio buttons: 'Left to Right' (selected) and 'Right to Left'.
  - Count Sequence:** A text box containing 'Count by 004' and a 'Test' button.
- HEX Upper Bounds:** A row of seven spinner boxes containing '4D', '41', '43', '2D', 'FF', 'FF', 'FF'.
- Serial Number Bytes as shown in Printable ASCII Characters and in HEX:**
  - A row of seven character boxes containing 'M', 'A', 'C', '-', '0', '0', '0'.
  - A row of seven spinner boxes containing '4D', '41', '43', '2D', '30', '30', '30'.
- Lower Bounds:** A row of seven spinner boxes containing '4D', '41', '43', '2D', '00', '00', '00'.
- Force Selected Bytes to:**
  - A row of seven checkboxes, all of which are unchecked.
  - A row of seven buttons: 'BINARY', 'NUMERIC', 'CONSTANT', 'ALPHA UPPER', 'ALPHA LOWER', and 'PRINTABLE'.
  - A 'Deselect All' button below the row of buttons.

**Figure 14-1: Serialize Main Screen**

### 14.2.1 Startup And File Options

On startup, the Serialize program will automatically open the last file saved. If no file has been saved, it will default to blank.

- **Reset Button** - Lets you reset to a screen with only a single serial number byte defaulted to binary (00), Hex Address 00000000.
- **Regenerate Button** - Preserves serial number bytes and bounds but assigns a new Serial Number Unique ID
- **Save Button, File→Save** - Brings up a dialog to which is set to save to the current serial file configuration.
- **File→Save As** - Brings up a dialog to select any file
- **File→Load** - Brings up a dialog to select a previously saved file

#### 14.2.1.1 Reset

Instructs the program to start editing a NEW (as yet un-named) serial number file. It will throw away the information for any serial number currently being edited unless that information has been saved (Save Button). The new serial number is initialized with one (1) byte of binary.

#### 14.2.1.2 Save

Pops up a Save dialog with the current serial number being edited into the file name and path as shown in the Serial Number File window. If a file name has not been provided, a dialog will pop up to allow the user select a file name. Save will save any setup information in the file Serialize.ini. This file will initialize the setup information the next time the program is started.

#### 14.2.2 Serial Number File

Displays the path and filename of the open Serial File, if saved.

#### 14.2.3 Serial File Unique ID

Displays the unique ID associated with the Serial File. A unique ID is generated anytime a Serial File is created or modified. Multiple programming images are able to share the same serial numbers by referencing the same ID number (see **Section 14.6 - Shared Serial Numbers**). The user can click the "Regenerate" button to assign a new unique ID to the current Serial File.

#### 14.2.4 Serial File Name To Display

The name that will appear in the JSON viewer, Cyclone menu, and in the Name field of the Cyclone Control GUI. A descriptive name can help to easily differentiate Serial Files.

#### 14.2.5 Serial File Notes

Allows the user to save notes regarding the Serial File. These will appear in the JSON viewer when viewing the properties of the Serial File, see **Section 14.3 - Serial File Properties**.

#### 14.2.6 Number of Bytes in Serial Number

The up and down arrows allow the user to add or delete bytes for the serial number, the maximum in hexadecimal = 0x10 (16 base ten), the minimum = 1.

- Up Arrow Click - Adds new bytes to the Serial Number. Each byte added appears as a new column in the serial number representation. Added bytes are input as Binary Bytes, i.e. the upper bound is FF and the lower bound is 00.
- Down Arrow Click - Deletes bytes from the right end of the Serial Number. Any previously entered byte properties are lost.

#### 14.2.7 Starting HEX Address

Assigns the starting address (in hexadecimal) of the device memory where the serial number will be programmed.

#### 14.2.8 Count Sequence

This window lets you test the serial number by counting up or down through the sequencing of the serial number. The serial number will roll over the upper bounds of the highest serial number back to the lower bounds of the serial number, and vice versa.

- Up Arrow Click - Counts the serial number up.
- Down Arrow Click - Counts the serial number down.

#### 14.2.9 Serial Number Bytes as Hex

There is one display column for each byte in the serial number shown as printable ASCII characters. Non-printable ASCII characters are indicated by a blank gray box.

- Up Arrow Click - Counts the serial number up.



- Down Arrow Click - Counts the serial number down.
- Click into the edit Box to enter any number within range. Values entered will be limited by the upper and lower bounds.

#### 14.2.10 Hex Upper Bounds

There is one display column for each upper bound of the byte in the serial number in hex.

- Up Arrow Click - Increases the upper bound by one with a maximum of FF Hex.
- Down Arrow Click - Decreases the upper bound by one with a minimum of the current serial number byte value.
- Click into the edit Box to enter any number within range (from the Serial number value up to a maximum of Hex FF).

#### 14.2.11 Hex Lower Bounds

There is one display column for each byte of the lower bound of the serial number in hex.

- Up Arrow Click - Increases the lower bound by one with a maximum of the current serial number byte value.
- Down Arrow Click - Decreases the lower bound by one with a minimum of 00 Hex.
- Click into the edit Box to enter any number within range (from the Serial number value to a minimum of 00 Hex).

#### 14.2.12 Binary, Numeric, Constant, Alpha Upper, Alpha Lower, and Printable

Checkboxes at the bottom will select properties to set for each serial number byte. Selections will be highlighted in yellow. The buttons below may be used to set the boundary type for the selected serial byte(s).

#### 14.2.13 Byte Program Order

The user can choose whether the serial number increments from left to right, or from right to left, in the Byte Program Order section.

### 14.3 Serial File Properties

Serial File properties are easy to access, as the generated serial file is stored in JSON format. This makes it simple for the user to save and parse the information.

**Note:** Serial Files in JSON format are compatible with **Cyclone FX** firmware version 10.13 and later.

Users that wish to retrieve the original text serial number format can parse out the JSON file:

```
object->pesettings-> serial_number_definition -> legacyv1encoding
```

#### 14.3.1 Serial File Example

An example file is provided, Example.ser, located in the Image Creation folder of your installation directory. Below is a screen snapshot of the saved serial file as seen from a JSON viewer.

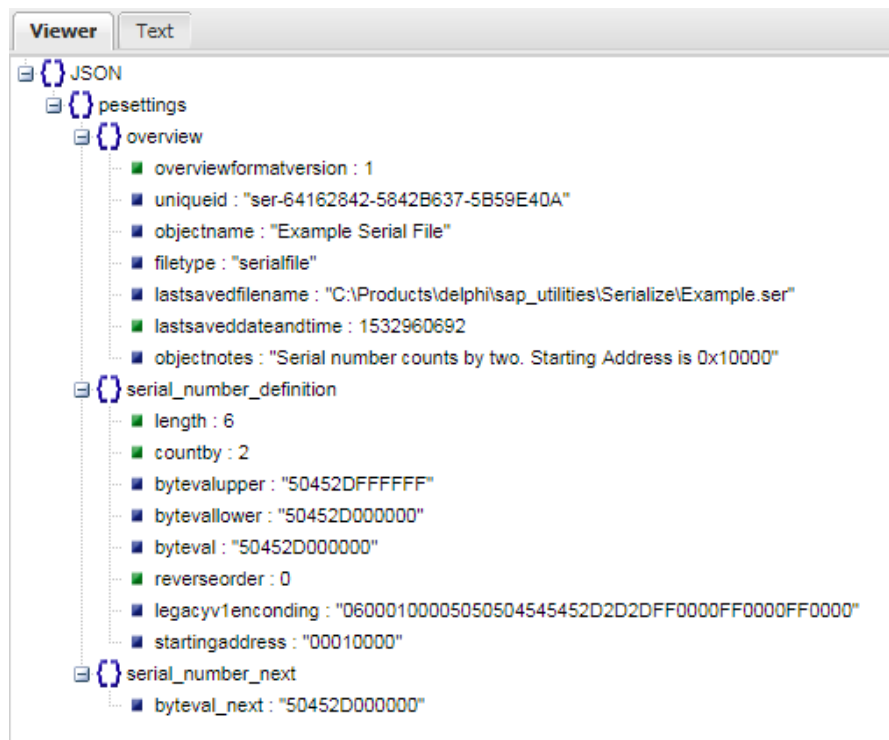


Figure 14-2: Example Serial File In JSON Viewer

## 14.4 Serial Number Handling

The **Cyclone FX** firmware implements the automatic serial number mechanism (see **Section 5.2.3.3 - Serial Numbers**). The same serial number files are used with the Cyclone Image Creation Utility, and the same commands are used to specify the serial number file and initiate serial number programming and incrementation. The serial number data structure is saved in the SAP image. Once a “PS” command is carried out, a serial number is programmed into the target. Only after all operations have been completed successfully does the Cyclone firmware automatically increment the serial number and store it in the Cyclone’s flash for internal images (or external CompactFlash for external SAP images).

The CS and PS commands are not present in the Cyclone Image Creation Utility until a valid programming algorithm is specified.

To complement the Cyclone’s usage in production environments, the Cyclone supports multiple serial number structures for each programming algorithm block. Each SAP image may contain multiple programming algorithms for every memory module it needs to program, and each programming algorithm block may contain multiple serial number structures. The SAP image sequence below illustrates this briefly:

```
CM algorithm_file_1
CQ
QO object_code_1
EM
PM
VC
CS serial_file1.ser
PS
CS serial_file2.ser
```

```

PS
CS serial_file_3.ser
PS
CM algorithm_file_2
CQ
QO object_code_2
EM
PM
VC
CS serial_file4.ser
PS
CS serial_file5.ser
PS

```

#### 14.4.1 Invoking A Serial File Via Command-Line

The command to invoke the serial number file in PEmicro's interactive programming software is "CS Choose Serial File". The command to actually program the serial number to target and automatically increment the serial number afterward is "PS Program Serial Number".

PEmicro's command line software uses the same commands in a command line fashion to invoke the serial number file, initiate its programming, and increment:

```

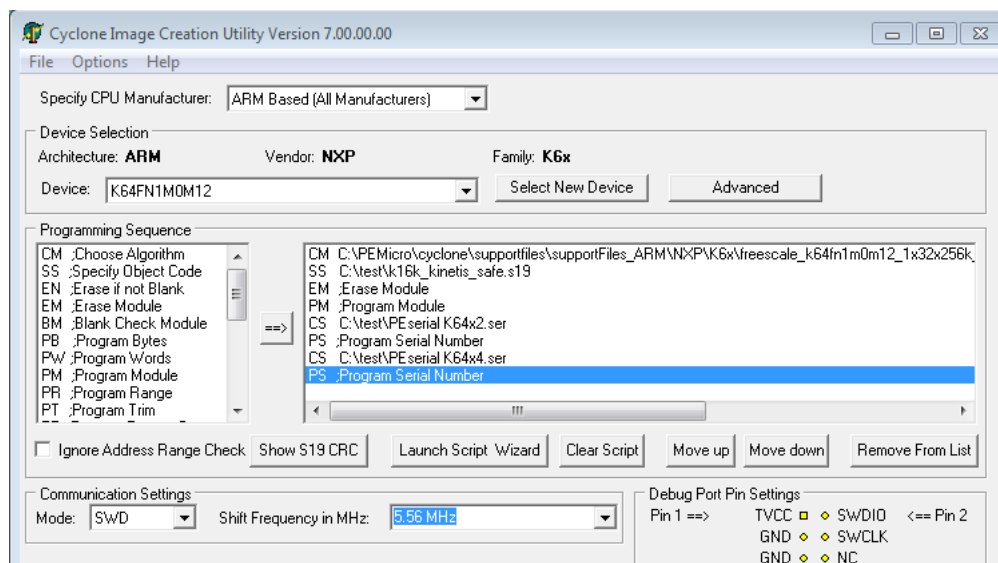
CS serial_number_file.ser
PS

```

#### 14.5 Creating A SAP Image With Multiple Serial Numbers

In the example shown in **Figure 14-3**, a SAP image was created that will select and program two different serial numbers into different locations in memory as part of the programming sequence. The first serial number counts by two with starting address 0x00020000 and the second serial number counts by four with starting address 0x00021000.

As indicated by the programming sequence, first the Algorithm is selected, followed by the S19 file. The part is erased, and then the s19 is programmed into the device. Once this is done, the first serial file is selected and programmed, and then the second serial file. These same unique serial numbers may be used by other SAP images, and each program will increase the serial number by the designated count defined in the serial number file.



**Figure 14-3: Cyclone Image Creation Utility: Serial Files In Programming Sequence**

## 14.6 Shared Serial Numbers

An important feature of the Serialize utility is that it automatically creates a Unique ID anytime a serial file is created or modified. This allows Stand-Alone Programming (SAP) images stored on a Cyclone to share serial numbers that reference the same unique serial number ID. There are several cases where this is very useful. The first is when a user may want to update their firmware for a product to a new version but needs the serial number to have persistence. The user may also have different products that need to be programmed with different firmware, but need to have those products draw from the same serialization sequence.

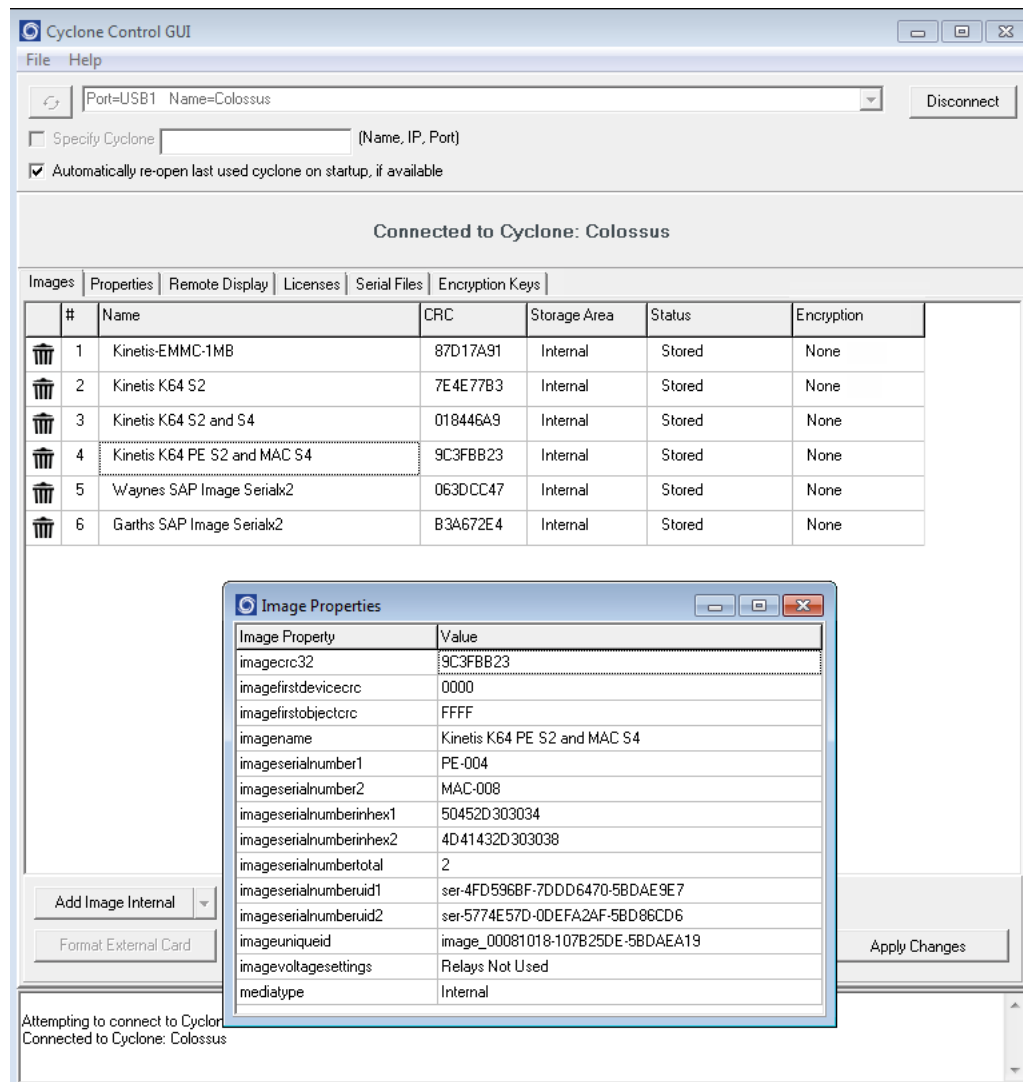
**Note:** Shared serial numbers require **Cyclone FX** firmware version 10.13 and later.

### 14.6.1 Example

This sections describes how a user might configure multiple SAP images on their Cyclone to share the same Serial File.

#### 14.6.1.1 View and Edit SAP image Via Cyclone Control GUI

In **Figure 14-4**, the Cyclone Control GUI has been launched and is looking at the image contents of the Cyclone. In this case it shows that the Cyclone named Colossus contains four different SAP images. To view properties of a particular image properties, the user can simply right click on the desired image and select “View properties” which will pop up an image properties window. In this case it shows Image “Kinetis K64 PE S2 and Mac S4.” The Image Unique ID is visible, as well as the two serial numbers and their respective Unique IDs. It also shows the next serial numbers (in ASCII representation) to be programmed upon launch: PE-000 and MAC-00, respectively.



**Figure 14-4: Cyclone Control GUI: Image Properties Window**

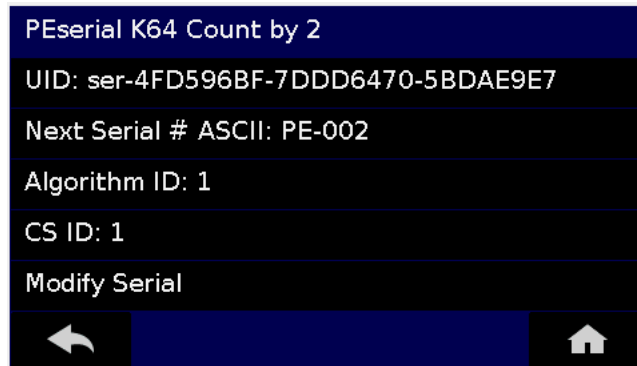
Deleting the image will not delete the associated serial number as other images stored on the Cyclone may be using that same serial number. The serial number is tracked on the Cyclone by the *imageuniqueid* JSON property. If any image is added back onto the Cyclone with that same Serial Unique ID, it will continue to track with that same serial number at the latest count. So if a user has a new version of their firmware, they can create a new SAP image that uses the same serial number as the previous firmware (referenced by same serial unique ID) and the programming will continue the serial number sequence from the current serial number state since that serial file is stored on the Cyclone.

Once a SAP image is loaded with a serial file, and it is selected as the current image on the Cyclone, it is possible to adjust the serial count. This is done by navigating through the following screens on the Cyclone.

Menu → Current Image Options → Serial Number

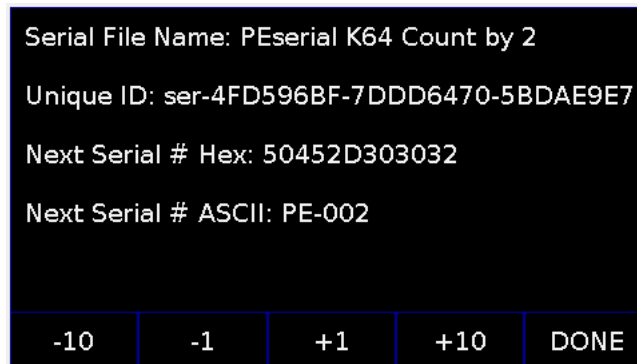
If the image has more than one serial number incorporated into the SAP image, select the Serial number which is to be modified. Otherwise just select 'Modify Serial'.





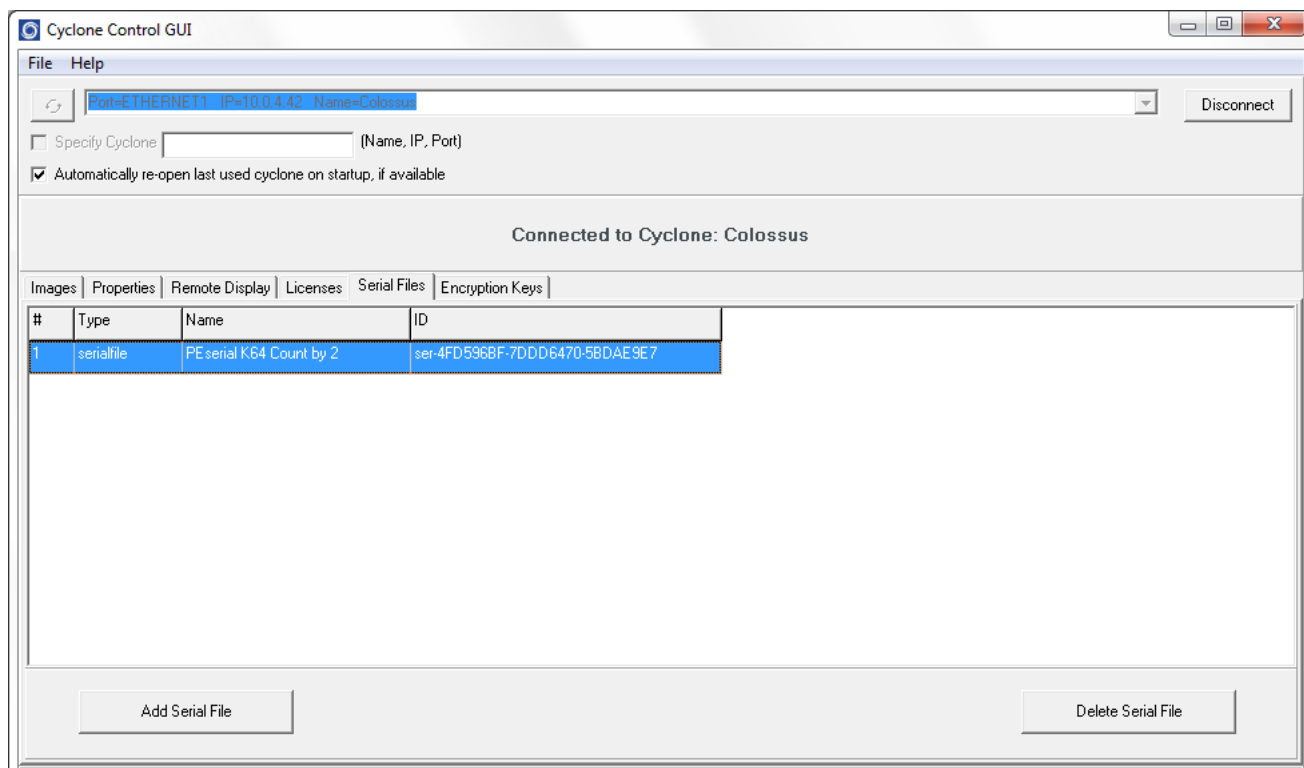
**Figure 14-5: Cyclone Menu Selection**

It is possible to Decrease or Increase the Next Serial by -10, -1, +1, +10. This is often done to address issues in the production process, such as during initial setup.



**Figure 14-6: Increase or Decrease Serial Number**

Note that the Cyclone Control GUI has the ability to show all the Serial Files stored on the Cyclone. This is done by opening a Cyclone, and then navigating to the Serial Files Tab, as shown in **Figure 14-7**. The Serial File tab has a Delete Button at the bottom which will remove the selected serial file from view and reset the tracking of the serial number to its original state. This does not actually delete the serial file. Once any programming is initiated using this 'deleted' serial file, the serial file will reappear back in the list and program using the original serial number.



**Figure 14-7: Cyclone Control GUI: Serial Files Tab**

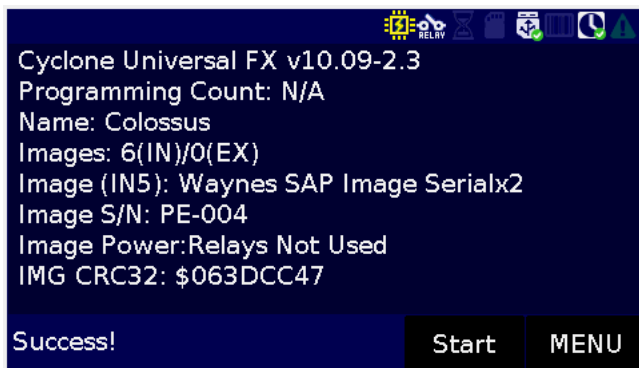
#### 14.6.1.2 Program SAP images With Shared Serialization

The Cyclone screen in **Figure 14-8** shows two SAP images stored on the Cyclone: The first is 'Waynes SAP Image Serialx2' and the second is 'Garths SAP Image Serialx2'. They both use the same serial file named 'PEserial K64 Count by 2' which has a unique ID `ser-4FD596BF-7DDD6470-5BDAE9E7` and the next serial number to be programmed (in ASCII) is PE-002. If we select Waynes SAP image we can see this current PE-002.



**Figure 14-8: View Current Serial Number for "Wayne's SAP Image"**

Once programmed is initiated, PE-002 is programmed into memory and the screen shows the next serial number to be programmed is which is PE-004.



**Figure 14-9: Serial Number Incremented for "Wayne's SAP Image"**

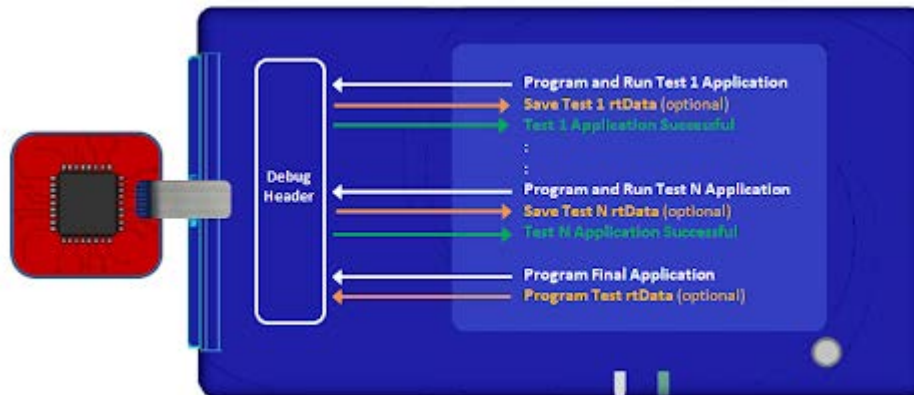
If the Garth's SAP Image is now selected on the Cyclone, it will show the next serial number to be programmed is PE-004 because it tracks the same serial number.



**Figure 14-10: Serial Number Also Incremented for "Garth's SAP Image"**

## 15 RUN TEST - CUSTOM TEST APPLICATIONS IN THE PROGRAMMING SEQUENCE

As part of the production programming process, **Cyclone FX** programmers have the ability to program and run a series of custom test applications in the target processor before final programming is allowed to occur. These custom test applications, written by the end user, indicate to the Cyclone through the debug interface whether the test application was successful or not and also optionally return generated data to the Cyclone for use later in the programming process. If any of the test applications fail, programming does not continue. If all the test applications pass, the final application is programmed into the target and can optionally include any data generated by the test applications. This “Run Test” process is shown here:



Example uses of the Run Test feature during programming:

- Verify that the target hardware works as intended before programming
- Add information from the Cyclone, such as a scanned barcode or current date and time to the final programmed data
- Uniquely fingerprint each target based upon its Unique ID using any algorithm. This fingerprint could be later verified or read by the final application in the field.
- Precisely calibrate an RTC crystal using an External Timer/Counter
- Fetch Data from user input on the Cyclone Screen and add to the final programming data
- Add information from inputs on the target processor to the final programming data, such as:
  - Data could be read from external devices on the PCB such as recording the serial number of external flash memories (to prevent them from being changed for instance)
  - Data could be read from communications interfaces connected to the target processor. For instance, if the target being programmed had a network connection, data could be read from a local PC or an internet server and added to the data to be programmed.

### 15.1 Run Test Availability

The Run Test advanced control/automation feature is a standard feature on **Cyclone FX** programmers. It is not available on **Cyclone LC** models. The user should be sure to install the latest Cyclone installation software and firmware to use the most up-to-date version of the Run Test Feature, as well as the Run Test API files PEtest.c and PEtest.h for including into test applications.

### 15.2 “Run Test” Applications

Run Test applications are written by the user in any development environment which can generate a valid application for the processor being programmed. The test application could be a special version of the final application which conditionally has included test code, or the test application could as easily be a totally different code base. The test application links in the Run Test API code

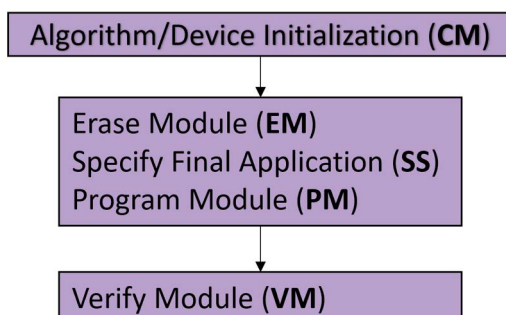
which allows the run test application to communicate to the Cyclone during the test process. While each test application must in the end report back a success/fail result to the Cyclone, it can also query the Cyclone for information and interact with the user through the Cyclone hardware.

As part of each test application that is programmed and run, the test application can optionally return "named" blocks of run test data (rtData) which are saved in the Cyclone until the entire programming process is complete. These rtData blocks may be programmed with any subsequent test application or during final application programming.

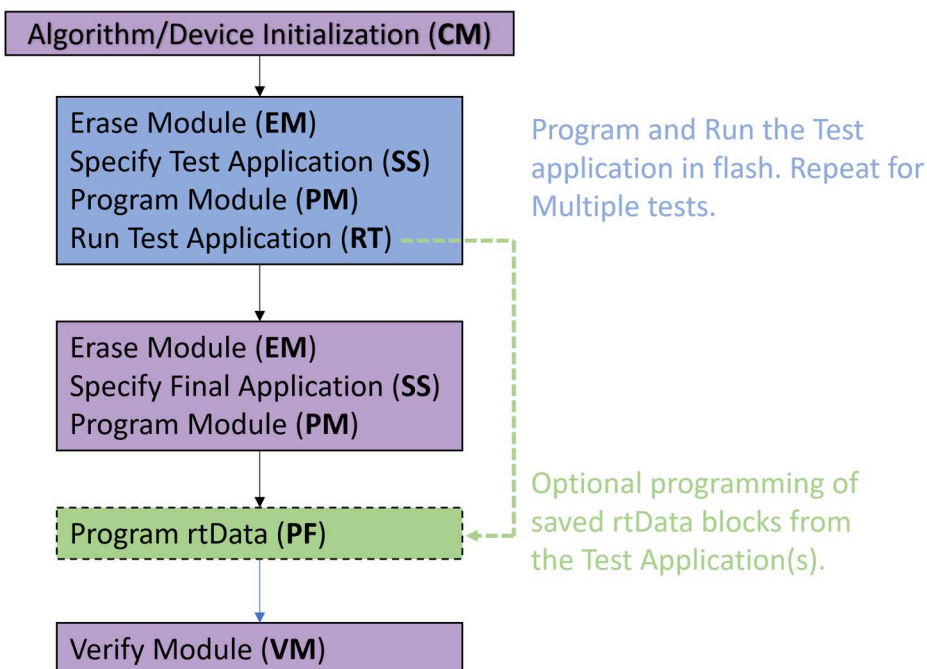
### 15.3 Adding Run Tests to the Cyclone Programming Script

The Image Creation utility, which generates stand-alone programming images, has a script creation wizard which specifies the steps to be executed during target programming.

Here is what a simple, and yet common, programming script without Run Test looks like:



Here is what a simple programming script with Run Test looks like:



The RT command (Run Test) executes the application currently programmed into the flash. The application establishes a connection to the Cyclone through the debug bus, optionally interacts with the Cyclone, optionally saves named blocks of data (rtData) to the Cyclone, and will report success/failure to the Cyclone. A failure immediately terminates the programming script. The PF command (program feature) can be used to program any named rtData blocks back into the device at any time. rtData blocks are discarded once the programming script/image completes.



## 15.4 Run Test API Calls

In order to implement the test applications as part of the programming process, PEmicro provides a Run Test API C and header file. These are linked into the test application and allow the application to communicate to the Cyclone through the debug bus. The API calls let the user display, retrieve, and save a variety of information on the cyclone.

A selection of the Run Test API Calls:

### **void runtest\_initialize\_programmer\_connection(void);**

Initiate the connection to the cyclone programmer across the debug connection. This is expected to be called somewhere near the start of the main() function in the test application. If the Cyclone doesn't receive a connection request from the Run Test application, then it will error after a timeout. Once this routine returns, all other calls may be used until the test application is stopped by the runtest\_stop\_test\_and\_return\_result() call.

### **void runtest\_stop\_test\_and\_return\_result(unsigned long pe\_test\_result);**

Stops the test application from running and sends back the test application result to the Cyclone as pass/fail with error codes. The Cyclone will move on to the next programming step if the result was successful. A zero result denotes success and non-zero means failure.

### **unsigned char runtest\_send\_rtdata\_block\_to\_programmer(void \*dataptr, unsigned long number\_of\_bytes, char \*data\_feature\_description);**

Sends a block of data ("rtData") to the Cyclone, with a specific name to reference it, for later use by the Cyclone. This data can be reprogrammed by the Cyclone into the device using the PF Program Feature command when programming in the final application (or other test applications).

### **void runtest\_get\_date\_time(pe\_date\_struct \*pe\_date\_ptr);**

### **void runtest\_get\_UTC\_date\_time(pe\_date\_struct \*pe\_date\_ptr);**

Allows the test application to retrieve either the local or the UTC date and time from the Cyclone. The Cyclone will fill out the data structure pointed to by the pe\_data\_ptr.

pe_date_ptr	pe_date_struct *	0x20000044
(x)= pe_year	unsigned long	2016 (Decimal)
(x)= pe_month	unsigned long	2 (Decimal)
(x)= pe_day	unsigned long	10 (Decimal)
(x)= pe_hour	unsigned long	16 (Decimal)
(x)= pe_min	unsigned long	49 (Decimal)
(x)= pe_sec	unsigned long	0 (Decimal)
(x)= pe_msec	unsigned long	224 (Decimal)

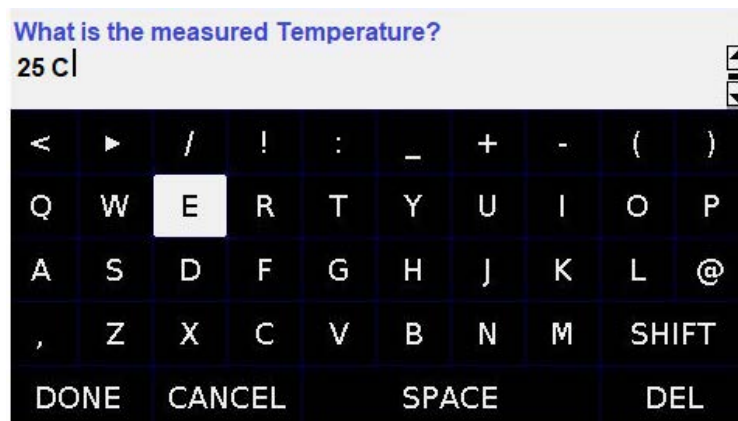
### **unsigned char runtest\_display\_message\_dialog(char\* dialog\_text, char\* button\_one\_text, char\* button\_two\_text, char\* button\_three\_text);**

Allows the test application to pop up a message box with three buttons on the display of the Cyclone or on the PC. All strings are null terminated. If the button text for each button is not blank, then the button will be visible. This is a way to either notify the operator to do something at some point in the test or to ask them a question. Once a button is selected, it will send back the button number from the Cyclone to the running test application in the target. If no button is selected, it will send back a 0 for no data.

### **unsigned char runtest\_display\_input\_query(const char\* message\_string, char \*string\_input\_by\_user, unsigned char max\_string\_length);**

Pops up a message box on the Cyclone screen prompting the user for a response. This default

response on the Cyclone screen for editing is specified by the `string_input_by_user` string. If the user enters a new string, the data in the `string_input_by_user` string will be overwritten. The image below is an example of what is shown on the Cyclone touch screen.



```
unsigned char runtest_write_string_to_log_file(char* string_to_append_to_logfile);
```

Send a text string back to the Cyclone for display/logging.

```
void runtest_request_power_measurement(unsigned long *V_measured, unsigned long *I_measured);
```

Reads the target voltage and current usage as measured from the Cyclone. The voltage is in mV and the current is in uA.

## 15.5 Example Programming Images With Run Test

PEmicro has provided several examples, including application codes and programming scripts, online at: [http://www.pemicro.com/blog/index.cfm?post\\_id=223](http://www.pemicro.com/blog/index.cfm?post_id=223)

## 16 TROUBLESHOOTING

This section answers some common questions that should help the user with various aspects of **Cyclone FX** operation.

### 16.1 My Cyclone Is Non-Responsive, Is There A Way To Re-Activate It?

It is possible that the issue is outdated firmware. In this case, you may wish to use **bootloader mode** to update the Cyclone firmware and then try to re-start the Cyclone.

#### 16.1.1 What Is Bootloader Mode?

Bootloader Mode is a special running mode of the Cyclone Universal and Cyclone Universal FX in which only limited functionality of the Cyclone is allowed. In this mode, the Cyclone will allow communication to a PC via USB, Ethernet or serial ports. In Bootloader Mode the user can update the Cyclone firmware via the cyclone utilities.

The Bootloader screen will display the version of the bootloader, the version of the internal and external application, the name of the Cyclone and it's IP address.

#### 16.1.2 When Is Bootloader Mode Used?

If the Cyclone ever becomes unresponsive, communication to the PC is not possible via USB, Ethernet, or Serial ports and if the cyclone fails to power on.

#### 16.1.3 How Is Bootloader Mode Entered?

You can force the Cyclone into bootloader mode with the following sequence, with the Cyclone powered:

- Press the Reset button
- Press the Start button
- Release the Reset button
- Tap the Cyclone LCD screen 3 times
- Release the Start button

### 16.2 I Received A “SAP Image Needs To Be Updated” Error Using A Next-Gen Cyclone, How Do I Update?

The current PEmicro software that is available for all our Cyclones generates SAP images that are compatible with our newest generation of Cyclones.

However, customers who have generated SAP images using **older versions of our Cyclone software** with Cyclones such as the Cyclone PRO, Cyclone MAX, Cyclone for ARM devices Rev. A/B, etc., will find that these SAP images will not work on the newer **Cyclone LC** and **Cyclone FX** programmers. Simply recreating these images for current generation Cyclones could potentially introduce errors and lose information about commands, settings, and configurations.

Therefore, we created the “SAP\_Convert\_Console.exe” which can be used to convert older generation SAP images into current generation SAP images. Once converted, an image will work not only on **Cyclone LC** and **Cyclone FX** programmers, but it will also remain compatible with the Cyclone for which it was originally created.

#### 16.2.1 How Do I Use SAP\_Convert\_Console.exe?

SAP\_Convert\_Console.exe is a Windows command line utility and the software must be run through the Windows Command Prompt. The utility can be found in the same folder as the Cyclone's software install path.

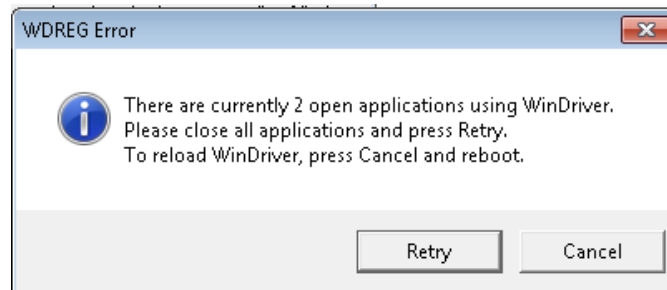
The command line parameter syntax:

```
>SAP_Convert_Console [old_SAP_path] [new_SAP_path]
```

Where:

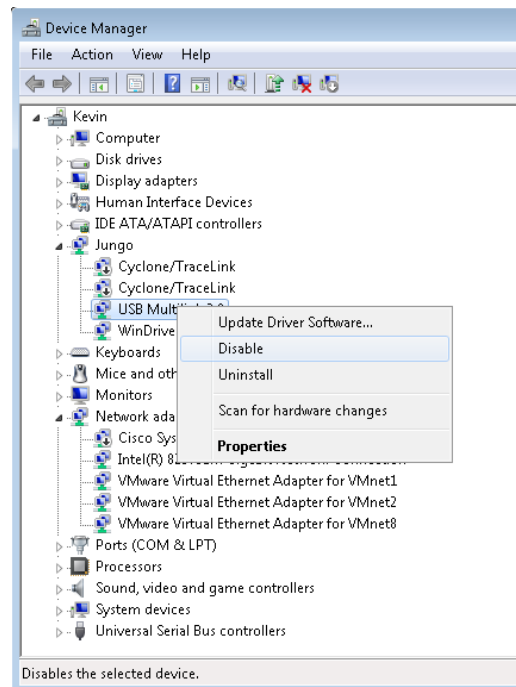
- [old\_SAP\_path]** The relative or full path to the SAP file. Usually has the .SAP file extension.
- [new\_SAP\_path]** Optional parameter where the user can specify a relative or full path to dump the output of the conversion. If path and file name matches the input, then the output file will replace the input file. If this parameter is not specified, the output will be dumped in the same path as the input file renamed with postfix “\_2”. For example if the input is myfile.SAP, then the output will be myfile\_2.SAP and will not replace the original input file.

### 16.3 When Trying To Install The CYCLONE Software, A Popup WDREG Error Occurs Telling Me That There Are Open Devices Using WinDriver.



**Figure 16-1: WDREG Error Message**

The error is that the USB Driver (WinDriver) used by PEmicro devices and software is currently in use and can't be upgraded as such. The simple solution is to shut down any PEmicro software and also unplug any PEmicro Cyclone, Multilink, and OpenSDA devices. If the error pops back up upon Retry, or you are remote from the machine itself, go to the Windows Device Manager and right click on each Multilink, Cyclone, and OpenSDA device and select "Disable".



**Figure 16-2: Disable Interfaces In Windows Device Manager**

Upon clicking the Retry button in the installer, the drivers should now install and the devices will be automatically re-enabled.

## 16.4 What should I do if the target device does not enter debug mode, and I receive the error message “Cannot enter background mode”?

If you receive this message you should check your hardware with a scope, logic analyzer or logic probe. First check for power on, then check to make sure the processor oscillator is running. Finally, look for the startup sequence below by first finding your Cyclone part number and then the Cyclone port for your microprocessor.

### CYCLONE-FX-ARM

---

#### Ports A, B, C - Kinetis, LPC, S32 (ARM) & other ARM Cortex devices

##### ARM JTAG

- RESET is driven low (to processor).
- Activity appears on TCK, TDI and TDO (PC software instructs the processor to enable debug mode).
- RESET is released by the interface and will go high.
- Activity appears on TCK , TDI and TDO (Debug activity).

##### ARM SWD

- RESET is driven low (to processor).
- Activity appears on SWD\_CLK and SWD\_DIO (PC software instructs the processor to enable debug mode).
- RESET is released by the interface and will go high.
- Activity appears on SWD\_CLK and SWD\_DIO (Debug activity).

### CYCLONE-FX-UNIV

---

#### Ports A, B, H - Kinetis, LPC, S32 (ARM) & other ARM Cortex devices

##### ARM JTAG

- RESET is driven low (to processor).
- Activity appears on TCK, TDI and TDO (PC software instructs the processor to enable debug mode).
- RESET is released by the interface and will go high.
- Activity appears on TCK , TDI and TDO (Debug activity).

##### ARM SWD

- RESET is driven low (to processor).
- Activity appears on SWD\_CLK and SWD\_DIO (PC software instructs the processor to enable debug mode).
- RESET is released by the interface and will go high.
- Activity appears on SWD\_CLK and SWD\_DIO (Debug activity).

#### Port G- MPC5xx/8xx

- DSCK is driven high and DSI is driven low (to processor).
- Delay ~1ms.

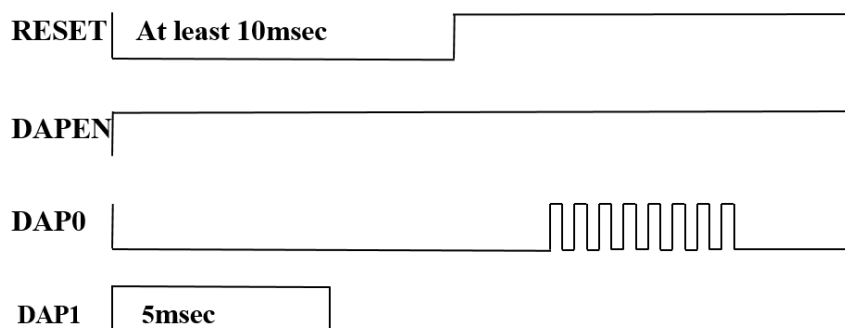


- HRESET or SRESET is driven low (usually this will be HReset).
- Delay ~20ms.
- HRESET is released (tri-state, should be pulled up on target).
- Shifting activity appears on DSCLK, DSI and DSO. (PC software communicating with target to determine if debug mode was successfully entered).

## Port H - Infineon TriCore

### Infineon TriCore - DAP Mode

- The following reset sequence enables DAP Mode:



**Figure 16-3: Reset Sequence**

- DAPEN (Pin 8) signal (if exists) is driven High.
- DAP0 (Pin 4) signal is driven Low.
- RESET (Pin 10) signal is driven Low.
- For some targets, as soon as RESET is driven Low, DAP1 (Pin 2) signal is pulled up High by the target for 5 milliseconds. It is then driven Low.
- After at least 10 milliseconds (target may experience power cycle by PEmicro hardware if so instructed), the RESET signal is tri-stated and pulled up High.
- After the RESET signal stabilizes, DAP0 issues 8 pulses.

## Ports C & E - Renesas, Non-ARM Only (Adapter Required)

### RH850

#### 2-Wire UART Mode

To enter UART mode FLMD0 will need to be High and FLMD1 will need to be Low before Reset is released. Some special devices will need up to 3 pulses on FLMD0 after reset is released and before the low pulses on RxD.

### RL78

Please see Renesas' RL78 documentation for information.

### R8C Tiny

#### Single Wire Mode:

Please see Renesas' R8C Tiny documentation for information.

#### UART Mode:

After selecting UART mode as the communication mode. the bit rate is adjusted.

The bit rate is adjusted to 9600 bps by sending standard time command (00H) 16 times and bit rate 9600 command (B0H) at 9600 bps from the Cyclone (on the MODE pin for Single Wire mode and on the Cyclone's RxD pin for UART mode). When the target receives the bit rate

9600 command normally, bit rate 9600 command (B0H) is returned

### **R8C 3x Lx**

#### UART Mode:

Please see Renesas' R8C 3x Lx documentation for information.

#### Single Wire Mode

After selecting UART mode as the communication mode. the bit rate is adjusted.

The bit rate is adjusted to 9600 bps by sending standard time command (00H) 16 times and bit rate 9600 command (B0H) at 9600 bps from the Cyclone (on the MODE pin for Single Wire mode and on the Cyclone's Rx/D pin for UART mode). When the target receives the bit rate 9600 command normally, bit rate 9600 command (B0H) is returned

### **Port D – Coldfire V2/V3/V4**

- BKPT (Pin-2), DSI (Pin-8), and DSCLK (Pin-4) signals are driven low.
- RESET (Pin-7) is driven low for 20+ milliseconds and released.
- After RESET is released and if the processor has correctly entered background mode, the PST0 (Pin-15), PST1 (Pin-14), PST2 (Pin-13) and PST3 (Pin-12) lines should all be driven high by the processor.
- Activity (changing signals) is seen on the DSI, DSO, and DSCLK signals. The activity on the DSCLK and DSI lines is generated by the PC and the activity on the DSO line is generated by the processor.

### **Port F**

#### **HCS08, S12Z, ColdFire V1, RS08**

- Debug activity is seen on BKGD (Pin-1).

#### **HC(S)12(X)**

- BKGD (Pin-1) and RESET (Pin-4) are pulled low by the interface.
- After 5 milliseconds, RESET (Pin-4) is released and goes high.
- After 10 milliseconds, BKGD (Pin-1) is released and goes high.
- After 20 more milliseconds, debug activity is seen on BKGD (Pin-1).

## 17 ERROR CODES

When the **Cyclone FX** encounters an error during operation it will display an error code with an “i” symbol, which can be click on for more information about that error code. Please contact PEmicro if instructed or if you are unsure of the specific meaning of an error code after clicking “i” and reading the description.

## 18 TECHNICAL INFORMATION

This section contains information about various physical, mechanical, electrical, etc. aspects of the **Cyclone FX** programmers, part # CYCLONE-FX-ARM and part # CYCLONE-FX-UNIV. The information applies to both part numbers unless specified.

### 18.1 Life Expectancy

Start Button: 1 Million Press Rated

### 18.2 Electrical Specifications

Input Voltage: DC 5.9V - 12V, center positive

Maximum Current: Up to 2A (0.7A typical) @ 6V

Up to 2A (0.5A typical) @ 12V

Barrel size: 5.5mm outer diameter, 2.5 mm inner diameter

### 18.3 Mechanical Specifications

Dimensions: 8" L x 4" W x 1.25" H (20.3cm L x 10.2cm W x 3.2cm H)

Weight: 12.7 oz (360 g)

### 18.4 Electromechanical Relays

Max Recommended Switched Voltage: DC 24V

Max Recommended Switched Current: 1A

Output Voltage To Debug Port (when configured for internal power): DC 2V-5V

Max Current To Debug Port (when configured for internal power): 0.5 A

Barrel size: 5.5mm outer diameter, 2.5 mm inner diameter

### 18.5 Internal Memory

CYCLONE FX: 1GB, 200+ programming images

### 18.6 Debug Ports - CYCLONE-FX-ARM

Ports A, B: 0.05" (1.27 mm) Pitch

Ports C,: 0.1" (2.54 mm) Pitch

Characteristic Impedance: 50 ohms (all ports)

### 18.7 Debug Ports - CYCLONE-FX-UNIV

Ports A, B: 0.05" (1.27 mm) Pitch

Ports C, D, E, F, G, H: 0.1" (2.54 mm) Pitch

Characteristic Impedance: 50 ohms (all ports)

### 18.8 International Shipping

HTS Number: 8471500150

ECCN: EAR99

### 18.9 Compliances/Standards

ROHS Compliant, CE Certified, FCC Certified

More information on PEmicro's Cyclone programmers is available at: [pemicro.com/cyclone](http://pemicro.com/cyclone).

# Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

NXP:

U-CYCLONE-FX