|  |  |
| --- | --- |
| Document Name: | **Excalibur Launcher - SDD** |
|  | **S**oftware **D**esign **D**escription |
| Document File Path in StarTeam: | C:\Users\shia\_l.CENTAURI\Documents\Visual Studio 2015\Projects\ExcaliburLauncher |
| Revision: | B |

Revision History Control

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rev. | Description (changes, additions) | Author | Approved by | Issue Date |
| A | Initial version | Shia Leigh |  | 21/2/2017 |
| B | 2nd Version | Shia Leigh |  | 18/12/2017 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Table of Contents**

[1. Introduction 3](#_Toc501296855)

[1.1 Scope 3](#_Toc501296856)

[1.2 Acronyms and Definitions 3](#_Toc501296857)

[1.3 References 3](#_Toc501296858)

[2. Component Dynamic Model 4](#_Toc501296859)

[2.1 Display Excalibur Devices Found in the Registry 4](#_Toc501296860)

[2.2 Verify if Devices found in Registry Exist in PC 7](#_Toc501296861)

[2.3 Device Configuration Screen (Adding and Updating Devices) 9](#_Toc501296862)

[2.3.1 Device Configuration Screen (unique Identifier / Socket Number) 9](#_Toc501296863)

[2.3.2 Device Configuration Screen (Ethernet Connection) 10](#_Toc501296864)

[2.3.3 Device Configuration Screen (USB Connection) 11](#_Toc501296865)

[2.4 Save Configuration to Registry 12](#_Toc501296866)

[4. Key Characteristics 14](#_Toc501296867)

[4.1 Compatibility with ExcConfig 14](#_Toc501296868)

[5. Risks 15](#_Toc501296869)

[5.1 Access to Excalibur software tools written in C and C++ 15](#_Toc501296870)

[6. T.B.D Follow-up 16](#_Toc501296871)

[Appendix A: Tasks List 17](#_Toc501296872)

[Use Case: 3.2 Verify if Devices found in Registry Exist in PC 17](#_Toc501296873)

[Use Case: 3.4 Save Configuration to Registry 17](#_Toc501296874)

# Introduction

## Scope

See SRS

## Acronyms and Definitions

|  |  |
| --- | --- |
| EL | Excalibur Launcher |
|  |  |

## References

|  |  |  |
| --- | --- | --- |
|  | Document Title | Revision |
|  | Excalibur Launcher - SRS | **B** |
|  |  |  |

# Component Dynamic Model

## Display Excalibur Devices Found in the Registry

* **Processing**

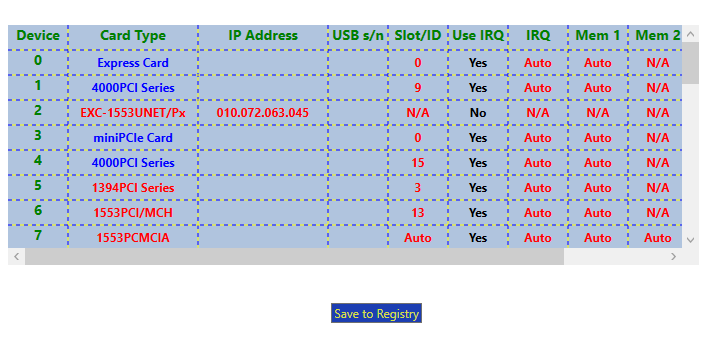
Read the registry for defined Excalibur devices and display devices found



* **Description**

Loop through the section of the registry where Excalibur devices are stored. Read registry data and copy into internal structure. Display devices found in the registry in a grid.

* **GUI Interfaces**

****

* **SW Interfaces**

1. Read from Registry
   * string ReadSubKeyValue(string subKey, string key)
2. Parameters for Read from and write to registry
   * SubKey - "SYSTEM\\CurrentControlSet\\Services\\Excx64\\Devices\\ExcaliburDevice\\" + device number
   * Key
     + "DeviceType"
     + "ExcConfigDeviceType"
     + "Serial\_Number"
     + "ID"
     + "IRQ"
     + "Mem"
     + "MemExtra"
     + "IO"
     + If UNET or Ethernet connection
       - "IsRemmote"
       - "ConnectType"
       - "DebugTheseModules"
       - If NET
         1. "IPAddress"
         2. "MACAddress"
         3. "UDPPort"
       - If USB
         1. "ProductID"
         2. "VendorID"

* **Data Items**

public struct sRegisteryDevices

{

public string strDeviceType;

public ushort DeviceType;

public string ExcConfigDeviceType;

public string ConnectType;

public int DebugTheseModules;

public int isRemote;

public string ProductID;

public string VendorID;

public string MACAdress;

public int UDPPort;

public ushort dev;

public string IPAdd;

public string USBSN;

public int SlotID;

public int IRQ;

public int Mem;

public int MemExtra;

public string IO;

public sModules[] mods;

` };

public struct sModules

{

public string ModuleName;

public ushort ModuleType;

public ushort mod;

};

public sRegisteryDevices[] regDevices = new sRegisteryDevices[mNumDevices];

public sRegisteryDevices[] RegDevs

{

get{ return regDevices; }

}

List of Board Types

EXC4000\_BRDTYPE\_PCI = 0x4000

EXC4000\_BRDTYPE\_CPCI = 0x4001

EXC4000\_BRDTYPE\_MCH\_PCI = 0x4002

EXC4000\_BRDTYPE\_MCH\_CPCI = 0x4003

EXC4000\_BRDTYPE\_MCH\_PMC = 0x4004

EXC4000\_BRDTYPE\_429\_PMC = 0x4005

EXC2000\_BRDTYPE\_PCI = 0x4006;

EXC4000\_BRDTYPE\_P104P = 0x4007

EXC4000\_BRDTYPE\_PCIHC = 0x4008

EXC4000\_BRDTYPE\_708\_PMC = 0x4009

EXC4000\_BRDTYPE\_1553PX\_PMC = 0x400A

EXC4000\_BRDTYPE\_DISCR\_PCI = 0x400D

EXC4000\_BRDTYPE\_PCIE = 0xE400

7EXC4000\_BRDTYPE\_PCIE64 = 0xE464

EXC2000\_BRDTYPE\_PCIE = 0xE406

EXCARD\_BRDTYPE\_1553PX = 0xE401

EXCARD\_BRDTYPE\_429RTX = 0xE402

MINIPCIE\_BRDTYPE\_429RTX = 0xE404

EXC4500\_BRDTYPE\_PCIE\_VPX = 0xE450

EXC\_BRDTYPE\_664PCIE = 0xE664

EXC\_BRDTYPE\_1394PCI = 0x1394

EXC\_BRDTYPE\_1394PCIE = 0xEF00

EXC\_BRDTYPE\_UNET = 0x5502

EXC\_BRDTYPE\_RNET = 0x5505

EXC-4000PCI Module Types

EXC4000\_MODTYPE\_SERIAL = 2

EXC4000\_MODTYPE\_MCH = 3

EXC4000\_MODTYPE\_RTX = 4

EXC4000\_MODTYPE\_PX = 5

EXC4000\_MODTYPE\_MMSI = 6

E7XC4000\_MODTYPE\_708 = 7

EXC4000\_MODTYPE\_MA = 8;

EXC4000\_MODTYPE\_CAN = 0xc

EXC4000\_MODTYPE\_DIO = 0xd;

EXC4000\_MODTYPE\_H009 = 0x9;

EXC4000\_MODTYPE\_NONE = 0x1F;

EXC4000\_MODTYPE\_CAN825 = 0x28

EXC4000\_MODTYPE\_SERIAL\_PLUS = 0x12

EXC4000\_MODTYPE\_AFDX\_TX = 0x1C

EXC4000\_MODTYPE\_AFDX\_RX = 0x1A

EXC4000\_MODTYPE\_ETHERNET = 0x1B

## Verify if Devices found in Registry Exist in PC

* **Processing**

Loop through the internal array of devices and check if the device is installed in the PC.

* **GUI Interfaces**

Using the same display described in the precious use case, if a device is found then display in blue if not found display in red.

* **SW Interfaces**
* Load DLL’s

Needed to use functions to find installed cards

* 1. LoadLibrary("exc4000ms.dll")
  2. LoadLibrary("Exc1394Ms.dll”)
* Call lpGet4000UniqueID(i, &wID)
  1. If no card found returns negative number(see error\_devio.h)
* Check for Module Type (Not part on initial development
  1. Ret = lpGetV4000ModuleType(device, i, &wModType);
* **Data Items**

n/a

## Device Configuration Screen (Adding and Updating Devices)

* **Processing**

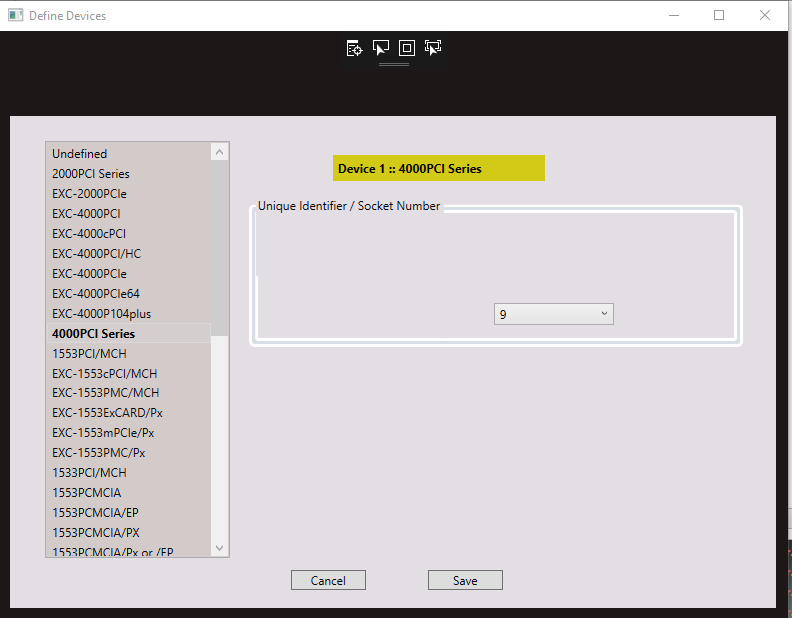
Setup configuration for selected card by selecting the card type and the appropriate setting for the card. The *EL* will display the parameters for each type of card as shown in GUI Interfaces

For example, in the case of 4000PCI cards, (3.3.1) users will define a identifier or socket number which must match the dip switch settings on the card itself. Other cards like the UNET1553Px have the option to connect either by defining an IP Address for ethernet (3.3.2) or a serial number in the case of USB, (3.3.3).

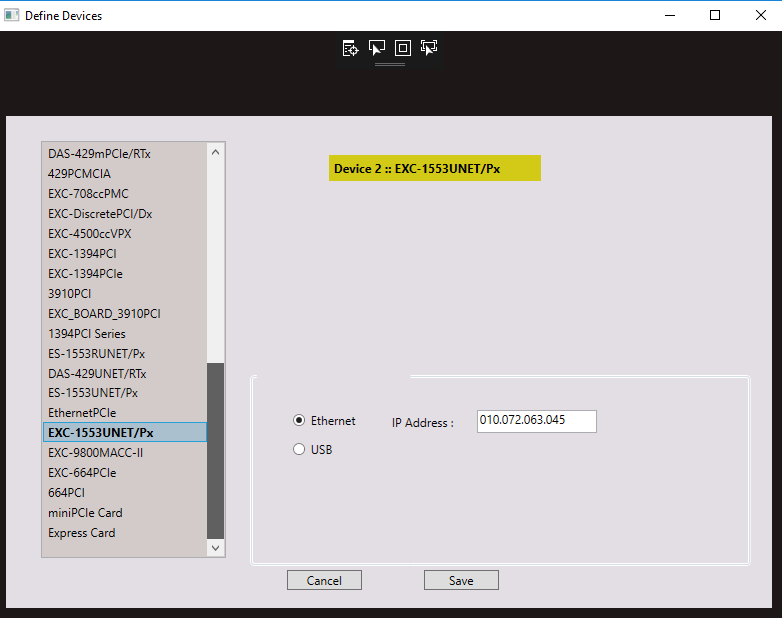
.

* **GUI Interfaces**

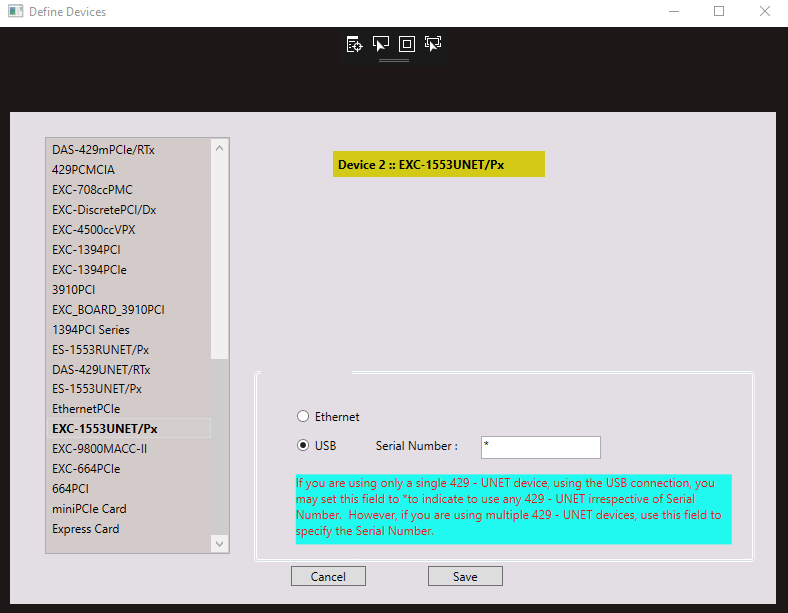
### Device Configuration Screen (unique Identifier / Socket Number)



### Device Configuration Screen (Ethernet Connection)



### Device Configuration Screen (USB Connection)



* **SW Interfaces**

Configuration details will be saved to internal structure regDevices, (see use case 3.1)

* **Data Items**

, (see use case 3.1)

## 2.3.4 Save Configuration to Registry

* **Processing**

Save new device configuration to registry

* **Description**

When the user selects save in the ***Device Configuration Screen,*** regDevices will be updated with the updated information.

The updated details will be redisplayed in the main screen

If the user selects ***Save to Registry, EL*** will loop from regDevices and save the entries to the registry.

* **GUI Interfaces**

See use case 3.1

* **SW Interfaces**

Write to the Registry

* 1. Void WriteSubKeyValue(string subKey, string key, object Value)
* **Data Items**

See 3.1

* **Unit Testing**
  + **Software**
    - Excalibur Launcher
    - ExcConfig
  + **Process**
    - Run ExcConfig add various devices and save to registry
    - Run ***EL*** verifydevices saved in ExcConfig are displayed with correct parameters
    - Reverse process by saving same devices from ***EL*** and then running ExcConfig. Check ExcConfig displays devices and configurations correctly
    - Delete all the Excalibur device information form the registry, (with great care DO NOT delete anything else).
    - Run ***EL*** and setup different devices and configurations and save to registry
    - Again, run ExcConfig and check compatibility**.**
    - Save configuration from the ExcConfig and reopen ***EL*** and check compatibility.

# 

# Key Characteristics

## Compatibility with ExcConfig

***EL*** is capable of recognizing a greater number of cards then **ExcConfig.** While these will work if only ***EL*** is utilized many cards will not be found if **ExcConfig** is run**.**

Therefore, at this stage different cards such is the 4000 series will be saved to the registry using the general configuration used by **ExcConfig.**

At a later stage we may update to use a wider range of cards

# Risks

## Access to Excalibur software tools written in C and C++

**Description:**

Excalibur software tools are written in C+ and C, the **EL** is written in C#. We need a software development tool that connects programs written in C and C++.

**Probability:1`**

**Severity:** 2

**Mitigation:**

SWIG has none been included with C# code and seems to be working using the software functions uses for far

# T.B.D Follow-up

|  |  |  |
| --- | --- | --- |
| **TBD** | **Responsibility** | **Due Date** |
| Launcher was developed to be compatible with ExcConfig. But is capable of detecting a greater range of cards. but for now, disabled to keep compatible with ExcConfig | Shia | SW Manager to decide when appropriate |
|  |  |  |
|  |  |  |

Appendix A: Tasks List

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case: 3.1** Display Excalibur Devices Found in the Registry | | | |
| ***Task***  ***#*** | ***Task***  ***Description*** | ***Duration***  ***in Days*** | ***Comment*** |
| 1 | function to Read from Registry | 1 day |  |
| 2 | Design GUI for opening screen | 5 days |  |
| 3 | Develop GUI for opening screen | 3 days |  |
| 4 | Develop Opening Screen | 5 days |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case: 3.2 Verify if Devices found in Registry Exist in PC | | | |
| ***Task***  ***#*** | ***Task***  ***Description*** | ***Duration***  ***in Days*** | ***Comment*** |
| 1 | Functions to Read devices in PC | 1 day |  |
| 2 | Programming | 5 days |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case: 3.3** Device Configuration Screen (Adding and Updating Devices | | | |
| ***Task***  ***#*** | ***Task***  ***Description*** | ***Duration***  ***in Days*** | ***Comment*** |
| 1 | Design GUI for configuration screen | 5 days |  |
| 2 | Develop GUI for configuration screen | 3 days |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case: 3.4 Save Configuration to Registry | | | |
| ***Task***  ***#*** | ***Task***  ***Description*** | ***Duration***  ***in Days*** | ***Comment*** |
| 1 | 1 day |  |  |
| 2 | Programming | 3 days |  |