



SightLine  
APPLICATIONS

## EAN-ARM-Application-Development

PN: EAN-ARM-Application-Development

11/2/2018

**Contact:**

Web: [sightlineapplications.com](http://sightlineapplications.com)

Sales: [sales@sightlineapplications.com](mailto:sales@sightlineapplications.com)

Support: [support@sightlineapplications.com](mailto:support@sightlineapplications.com)

Phone: +1 (541) 716-5137

**Export Controls**

Exports of SightLine products are governed by the US Department of Commerce, Export Administration Regulations (EAR); classification is ECCN 4A994. The [export summary sheet](#) located on the support/documentation page of our website outlines customers responsibilities and applicable rules. SightLine Applications takes export controls seriously and works to stay compliant with all export rules.

**Copyright and Use Agreement**

© Copyright 2018, SightLine Applications, Inc. All Rights reserved. The SightLine Applications name and logo and all related product and service names, design marks and slogans are the trademarks, and service marks of SightLine Applications, Inc.

Before loading, downloading, installing, upgrading or using any Licensed Product of SightLine Applications, Inc., users must read and agree to the license terms and conditions outlined in the [End User License Agreement](#).

All data, specifications, and information contained in this publication are based on information that we believe is reliable at the time of printing. SightLine Applications, Inc. reserves the right to make changes without prior notice.

**Alerts**

The following notifications are used throughout the document to help identify important safety and setup information to the user:

**⚠ CAUTION:** Alerts to a potential hazard that may result in personal injury, or an unsafe practice that causes damage to the equipment if not avoided.

**❗ IMPORTANT:** Identifies crucial information that is important to setup and configuration procedures.

**📄** *Used to emphasize points or reminds the user of something. Supplementary information that aids in the use or understanding of the equipment or subject that is not critical to system use.*



## Contents

1	Overview .....	1
1.1	Developing On-Board Applications .....	1
1.1.1	Lua .....	1
1.1.2	C/C++ .....	1
1.2	Sample Applications .....	2
1.3	Associated Documents .....	2
1.4	SightLine Software Requirements .....	3
2	Prerequisites .....	3
3	Conventions .....	3
4	Preparation .....	3
4.1	VMware .....	3
4.2	Setup R: Drive .....	4
4.3	CodeSourcery .....	6
4.4	Code Composer Studio .....	6
4.5	Windows Environment Variables .....	6
4.6	Sample Application Installation .....	7
5	Debugging and Deployment .....	7
5.1	Debug Mode .....	8
5.2	Access the U-Boot Console .....	9
5.2.1	1500-OEM: Booting from NFS .....	9
5.2.2	1500-OEM: Access NAND File System after NFS Boot .....	9
5.2.3	3000-OEM: Booting from NFS .....	10
5.2.4	3000-OEM: Access NAND Filesystem after NFS Boot .....	10
5.2.5	1500-OEM: Booting from NAND (Deploy Mode) .....	10
5.2.6	1500-OEM: Access Files on NFS after NAND Boot .....	11
5.2.7	3000-OEM: Booting from NAND (Deploy Mode) .....	11
5.2.8	3000-OEM: Access Files on NFS after NAND Boot .....	11
5.2.9	1500-OEM and 3000-OEM: Access Files on NAND through SCP Protocol .....	11
6	Start Development .....	12
6.1	Building the Application .....	12



- 6.2 Deploying the Application ..... 14
  - 6.2.1 1500-OEM Startup Script ..... 14
  - 6.2.2 3000-OEM Startup Script ..... 14
  - 6.2.3 Configuring U-Boot to Boot from NAND..... 14
- 7 Troubleshooting..... 15
  - 7.1 General Troubleshooting ..... 15
  - 7.2 NFS Boot Troubleshooting ..... 15
    - 7.2.1 Wifi Adapter Conflicts ..... 15
    - 7.2.2 IP Address Conflicts or Mismatch ..... 16
    - 7.2.3 Windows 10 Update Issues..... 16
    - 7.2.4 Reinstall VMware Application..... 16
  - 7.3 Shift+S Interrupt Issues ..... 16
  - 7.4 Questions and Additional Support..... 17
- Appendix A - Using VMware ..... 18
- Appendix B - Using GDB Debugging in CCS..... 19
- Appendix C - Updating the VMware Environment (2.23.1 or later)..... 21

**List of Figures**

- Figure 1: Booting from NFS..... 8
- Figure 2: Booting from NAND ..... 8
- Figure 3: Disable Bridging ..... 15
- Figure 4: Shift+S in Tera Term..... 17

**List of Tables**

- Table 1: Lua and C/C++ Comparison Table ..... 2
- Table 2: Software Prerequisites..... 3
- Table 3: System Variables ..... 7



## 1 Overview

This document describes the development environment setup and build, deployment, and debug processes for applications compiled to run on the ARM processor of SightLine video processing boards.

The sample programs for ARM-side development:

- can be built under Windows using the CodeSourcery cross-compiler,
- use TI CodeComposer Studio 5.x for development, debugging, and deployment,
- can communicate with the VideoTrack application running on the SLA-hardware using the SightLine Command and Control protocol over IP sockets

### 1.1 Developing On-Board Applications

SightLine provides two primary ways for customers to develop their own on-board applications: C/C++ and Lua. Each technology has benefits and costs for solving a problem. It is impossible to prescribe the right technology for every scenario. This section helps provide general guidelines to assist in understanding the tradeoffs.

#### 1.1.1 Lua

Lua is recommended for light-weight applications that need to perform simple data processing and interaction with the onboard video processing VideoTrack application. Applications such as dynamic on-screen displays based on telemetry data, or simple command and control from serial ports are good uses for Lua. Lua scripts are executed in-line with our video processing and cannot be synchronized with the processing of video frames. Issues such as increased latency and other performance impacts can arise from Lua scripts that can be very complex.

See the [EAN-Script Development](#) for more information for developing and using Lua scripts.

#### 1.1.2 C/C++

If an application requires complex data handling, frequent real-time access to IO, or should be run in parallel with VideoTrack, SightLine recommends creating C/C++ applications that can be run on the ARM processor.

When reviewing options, contact [Support](#) to discuss your application.



**Table 1: Lua and C/C++ Comparison Table**

Benefits	Drawbacks
<p><b>Lua</b></p> <ul style="list-style-type: none"> <li>• Simple to deploy</li> <li>• Frame synchronized execution</li> <li>• Can leverage numerous examples from SightLine or the internet</li> </ul>	<p><b>Lua</b></p> <ul style="list-style-type: none"> <li>• Not as widely used as C/C++</li> <li>• Access to IO is complex and difficult</li> <li>• Real-time debugging is not available</li> <li>• Networking is not yet supported</li> </ul>
<p><b>C/C++</b></p> <ul style="list-style-type: none"> <li>• Wide acceptance within the embedded programming industry</li> <li>• Can be easy to test on a PC before deploying on target hardware.</li> <li>• Real-time debugging</li> <li>• Complete access to IO, file system, etc.</li> <li>• Can leverage numerous examples from SightLine or the internet</li> <li>• Can run in parallel to existing applications</li> <li>• Portable to numerous platforms</li> </ul>	<p><b>C/C++</b></p> <ul style="list-style-type: none"> <li>• Deploying application to launch at run time can be error prone (file location, system permission, etc.)</li> <li>• Existing setup procedure is complex<sup>1</sup> (VMWare, CCStudio, mapped drives, NFS booting, ...)</li> </ul>

## 1.2 Sample Applications

This document was written for the SLA-Gimbal sample application. However, the same procedures are applicable to other sample applications described below. The sample applications send data to VideoTrack on port 14003 reserved for user programs. The sample applications receive responses and telemetry on port 16002. Multiple applications can simultaneously transmit packets to port 14003, but each application must use a unique receiving port.

**SLA-Gimbal** Provides an example of controlling a gimbal to follow a track. Receives TrackPosition messages from the VideoTrack ARM application over a local UDP socket. Track position information is used to update a PID controller, which then sends serial control commands to an external gimbal microcontroller to steer in the direction of the track.

**SLA-GPIO** Toggles SD card video recording on/off using a GPIO input, displays recording status via an LED attached to another GPIO output, and initiates a snapshot to the SD card using a second GPIO input.

**SLA-Landing** Receives landing aid telemetry from VideoTrack and sends commands to an autopilot.

**Overlay DLL** Provides access to the image data prior to encoding for rendering custom overlays.

## 1.3 Associated Documents

[EAN-GPIO-and-I2C](#): Describes how to create an application compiled for the 1500-OEM ARM processor that reads the GPIO state and sends commands to the VideoTrack1500 application.

[EAN-Script Development](#): Describes everything needed to develop and run custom scripts in Lua on the 1500-OEM and 3000-OEM hardware.

<sup>1</sup> These tools and procedures are complex but used industry wide with TI embedded systems.



## 1.4 SightLine Software Requirements

**ⓘ IMPORTANT:** The Panel Plus software version should match the firmware version running on the board.

## 2 Prerequisites

The software listed in [Table 1](#) is based on working in a development environment using a PC running Windows 7 or above.

**Table 2: Software Prerequisites**

Software	Purpose
VMware Player (see the <a href="#">VMware</a> section)	Hosts Linux sources Hosts shared folder for PC development Hosts TFTP and NFS for debugging applications on SLA-hardware
Code Composer Studio from Texas Instruments (see the <a href="#">Code Composer Studio</a> section)	Used to compile applications for deployment on SLA-hardware
CodeSourcery from Mentor Graphics (see the <a href="#">CodeSourcery</a> section)	Cross-compiler
<a href="#">PuTTY</a> or <a href="#">TeraTerm</a>	Used to debug output and issuing commands on SLA-hardware

## 3 Conventions

SLA-hardware	SightLine embedded video processor (1500-OEM, 3000-OEM, etc.)
OMAP Logic #	Typed U-Boot command to serial terminal on 1500-OEM
SLA3000#	Typed U-Boot command to serial terminal on 3000-OEM
DM-37x#	1500 Linux console (usually through serial port, or use SSH terminal)
root@sla3000:~#	3000 Linux console (usually through serial port, or use SSH terminal)
host>	Ubuntu Linux virtual machine console
PC>	Windows command prompt (cmd.exe)
<< some text >>	Indicates that user supplied information is required in place of << some text >>

## 4 Preparation

### 4.1 VMware

Download [VMware Player](#). For Windows 10, use VMware Player version 6.

Download the UbuntuSLA Virtual Machine from the [Software Downloads](#) page. A [zip utility](#) is needed to extract the archive. See [Using VMware](#) in the Appendix to start the VMware session.



## 4.2 Setup R: Drive

Once the virtual machine has booted, map the *R:* drive to the `/home/slroot` directory on the Ubuntu VM. The mapped drive contains the Linux kernel source/header files to include in the application and bootable network file system where the application will be located.

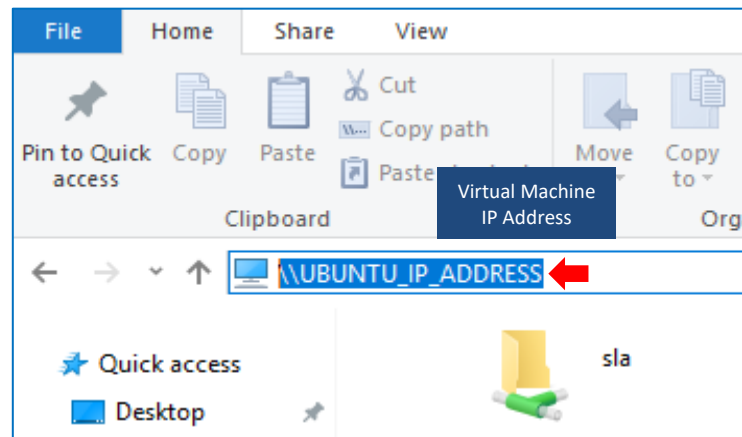
1. Use the `ifconfig` command from the Linux terminal to determine the VM IP address.

```

Terminal
File Edit View Terminal Help

slroot @ ~
$ ifconfig
eth6    Link encap:Ethernet  HWaddr 00:0c:29:d2:d2:5e
        inet addr:192.168.1.103  Bcast:192.168.1.255  Mask:255.255.255.0
  
```

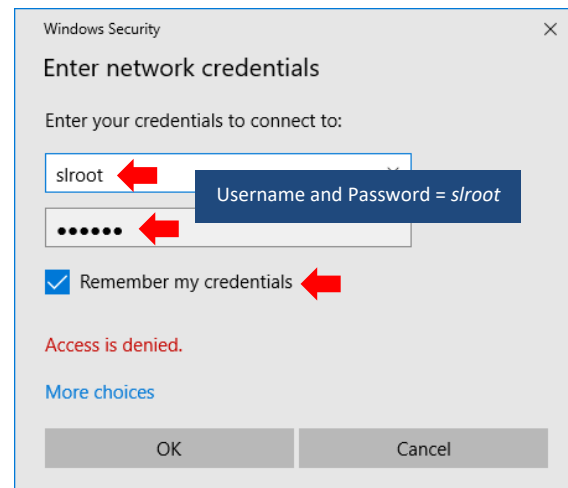
2. Map the *sla* share to the *R:* drive:
  - a. Open Windows File Explorer.
  - b. Enter the VM IP address in address bar (the *sla* share should appear).
  - c. Right click on the *sla* share and select *Map network drive...*
  - d. Select *R:* from the *Drive* drop-down menu.
  - e. Enter the path: `\\IP_ADDRESS\sla` (Example: `\\192.168.1.177\sla`)
  - f. Check *Reconnect at logon*.
3. Select *Finish*.



*Samba version 1 support is required to access shares on the Ubuntu VM. Some Windows 10 systems ship without SMBv1 support enabled. If the *sla* share does not appear in the list or Windows displays a message that `\\IP_ADDRESS` cannot be accessed (error 0x80040005), see the [Enable SMBv1 Support](#) section in the Appendix for instructions to verify that the Samba version 1 client is installed.*


4. Enter *slroot* for both the username and password.
5. Check *Remember my credentials*.

*It may be necessary to double click on the mapped drive to re-establish the connection following a restart, cold boot, or after resuming from sleep.*

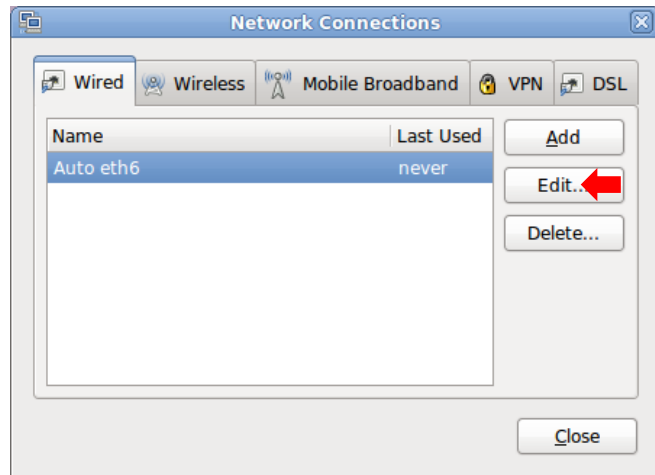







 *Configuring the Ubuntu VM with a static IP address will prevent DHCP address changes from breaking the mapped drive connection.*

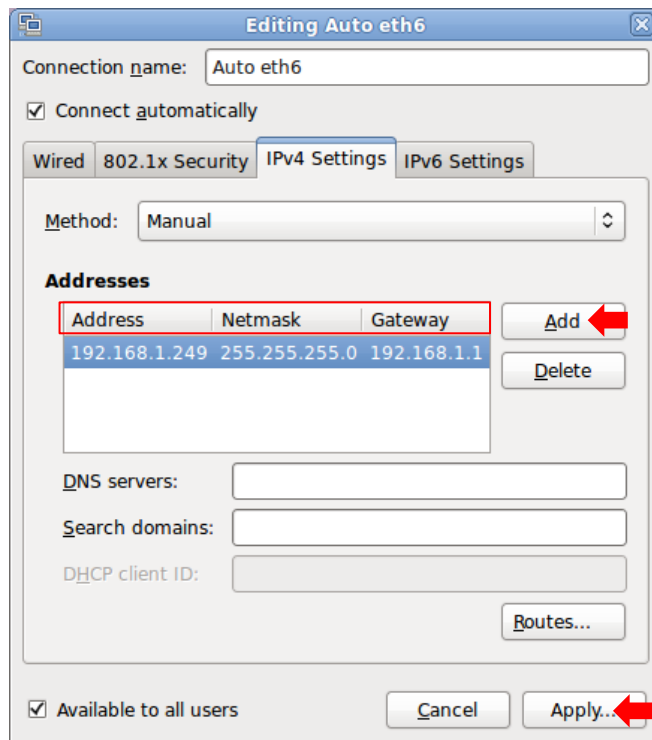
1. In the VMware Player window select *System » Preferences » Network Connections*.
2. Select the appropriate adapter (in the example this is *eth6*). Click *Edit*.



3. Select the *IPv4 Settings* tab and enter the *Address, Netmask* and *Gateway* information. Click *Add* when complete.

 *A DNS server can also be entered (8.8.8.8 is Google public DNS A).*

4. Click *Apply*.
5. Enter the password *slroot* and click *Authenticate*. Restart the virtual machine. Open a terminal window and enter *ifconfig* to validate the IP address.





### 4.3 CodeSourcery

CodeSourcery from Mentor Graphics is the standard for embedded Linux cross-compiler/linker tool sets (*toolchain*). CodeSourcery for Windows is required to compile applications for the ARM processor on SightLine boards under Windows.

Currently the sample applications/libraries are built with one of the following versions:

- [arm-2009q1-203-arm-none-linux-gnueabi](#) (most common)
- [2010.09-50 version](#)

When installing CodeSourcery, select the *Minimal* option. This installs all components other than the IDE. All other installation settings should be left at default.

### 4.4 Code Composer Studio

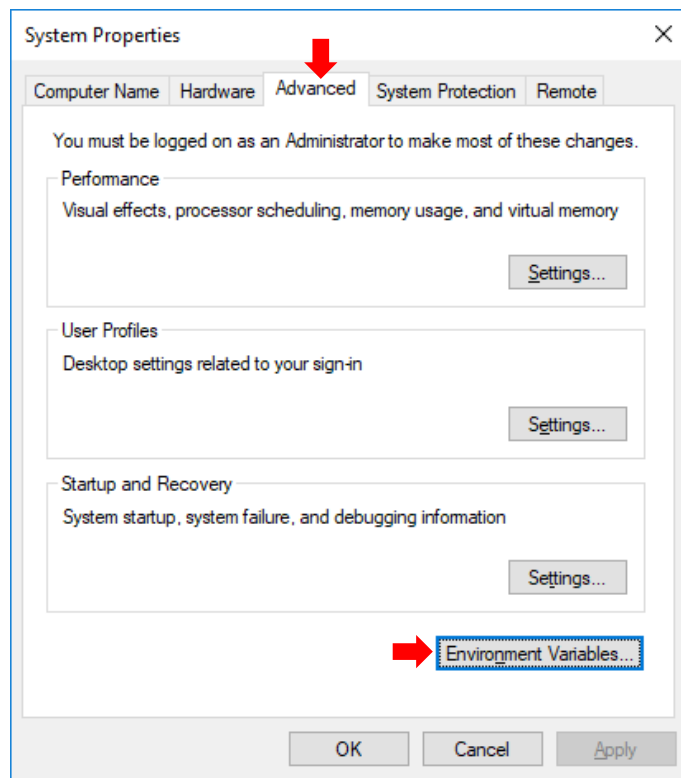
Use [Code Composer Studio v5.5](#) (CCS) as the IDE to develop and debug the application. Install CCS with default options. When prompted for *Processor Support*, check *Select All*. When CCS is launched for the first time, it will prompt the user to select a license. Select *FREE LICENSE* at the prompt.

 For Windows 7/8, use [Code Composer Studio v5.4](#)

### 4.5 Windows Environment Variables

The System Environment Variables must be defined on the host PC.

1. In the *Start* menu search field, enter *Edit the system environment variables* to open the *Advanced* tab of *System Properties* dialog window.
2. Click *Environment Variables*.
3. In the *System variables* dialog window click *New*.
4. Add the following variable names and values shown in [Table 2](#).
5. Click *OK* after each entry and repeat.





Variable names are case sensitive.

**Table 3: System Variables**

Variable Name:	Variable Value:
SL_1500FS	R:\sla1500fs (1500 only)
SL_1500ArtiFacts	R:\sla1500fs\root (1500 only)
SL_3000FS	R:\sla3000fs (3000 only)
SL_3000ArtiFacts	R:\sla3000fs\home\root (3000 only)
SL_TOOL_CHAIN	C:\Program Files (x86)\CodeSourcery\Sourcery G++ Lite
SL_TOOL_PREFIX	arm-none-linux-gnueabi-
SL_XDC	C:\ti\xdctools_3_25_00_48
SL_SDK	C:\SL\SDK

6. Copy cs-rm.exe in *C:\Program Files (x86)\CodeSourcery\Sourcery G++ Lite\bin* to *rm.exe*
  - a. Run cmd.exe as administrator.
  - b. Enter: *cd C:\Program Files (x86)\CodeSourcery\Sourcery G++ Lite\bin*
  - c. Enter: *copy cs-rm.exe rm.exe*

```

Administrator: Command Prompt
Microsoft Windows [Version 10.0.16299.192]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd C:\Program Files (x86)\CodeSourcery\Sourcery G++ Lite\bin

C:\Program Files (x86)\CodeSourcery\Sourcery G++ Lite\bin>copy cs-rm.exe rm.exe
1 file(s) copied.

C:\Program Files (x86)\CodeSourcery\Sourcery G++ Lite\bin>_

```

## 4.6 Sample Application Installation

Download the ARM code examples package from the [software downloads](#) page on the SightLine website. Launch the installer and follow the installation prompts.

## 5 Debugging and Deployment

This section discusses software installation on the target SLA-hardware (deployment) and troubleshooting problems with custom applications (debugging).

**IMPORTANT:** The server IP address (*serverip*, Ubuntu VM IP address) must be accessible from the SLA-hardware IP address (*ipaddr*). The SLA-hardware will boot from the server IP address. Connect the SLA-hardware and PC Ethernet interfaces to a network switch. Disable all other network adapters on the PC (wireless, VPN, etc.). Interfaces connected to other networks may interfere with the following steps.

**3000-OEM only:** If NFS boot does not work (T T T T T ... is displayed). Power cycle the SLA-hardware. Shift-S into U-Boot, and then enter *ping serverip* at the command prompt to verify the network connection.



## 5.1 Debug Mode

This section refers to hardware specific files and folders located on the Ubuntu Linux virtual machine running in VMware Player. In the following statements, replace **\*\*** with the first two digits of the numeric portion of SLA-hardware model number, e.g., *15* for the 1500-OEM or *30* for the 3000-OEM.

While debugging, it is easier to use the TFTP server and the Network File System (NFS) hosted by the Ubuntu VM.

The SLA-hardware boots using the kernel on the Linux host: `/tftpboot/ulmage.sla**00`

The SLA-hardware mounts the root filesystem (/) from the NFS directory on the Linux host (see the diagram below).

From the Linux host running on the target SLA-hardware, the root directory (/) of the filesystem is the same as `/tftpboot/sla**00fs/` on the Ubuntu VM.

Example: when a 1500-OEM boots from the TFTP server on the Ubuntu VM and mounts `/tftpboot/sla**00fs` as the root file system, *DM-37x# ls '/'* is equivalent to *host> ls ~/sla1500fs*.

From the Ubuntu VM: `/tftpboot/sla**00fs` and `~/sla**00fs` point to `/home/slroot/sla**00fs`.

Use NFS to access the same files from the SLA-hardware, Ubuntu VM, and Windows. Under Windows, `R:\sla**00fs` points to the NFS directory that the SLA-hardware mounts as the root filesystem.

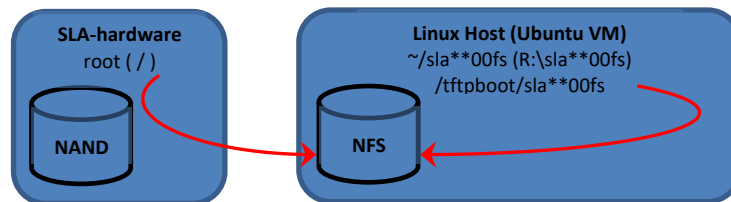


Figure 1: Booting from NFS

When deploying, the SLA-hardware should boot from NAND and use the NAND as the root file system.

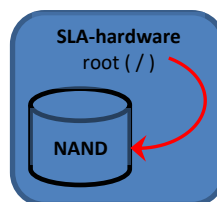


Figure 2: Booting from NAND

Switching between NFS boot and NAND boot requires changing U-Boot environment variables.



## 5.2 Access the U-Boot Console

U-Boot is a boot loader for embedded Linux systems that is used on the SLA-hardware. A serial connection is required to access U-Boot. The default baud rate is 115200.

1. Connect serial port 0 on the SLA-hardware to the PC.
2. Open a terminal emulator and connect to the appropriate COM port.
3. With focus in the terminal window, hold Shift+S and apply power to the SLA-hardware. This key combination interrupts the boot process and enters the U-Boot console.

### 5.2.1 1500-OEM: Booting from NFS

Change the U-Boot environment variables to boot from NFS. From the 1500-OEM U-Boot console, enter the following commands to change the boot defaults. *After completing these steps, the 1500-OEM will boot from NFS by default.*

1. *OMAP Logic # setenv serverip <<server ip address>>* IP address of the Ubuntu virtual machine
2. *OMAP Logic # setenv ipaddr <<ip address>>* Static IP address of the SLA-hardware (optional)
3. *OMAP Logic # setenv bootcmd run nfsboot* Set the boot mode {nfsboot, nandboot}
4. *OMAP Logic # setenv silent* Disable silent mode to allow interaction with Linux through serial 0
5. *OMAP Logic # save* Save the parameters to NAND flash
6. *OMAP Logic # boot* Continue loading the Linux Kernel

### 5.2.2 1500-OEM: Access NAND File System after NFS Boot

In some instances, it may be necessary to access the NAND filesystem after booting from NFS.

1. *DM-37x# mkdir /mnt/flash* Create a mount point (this directory may already exist)
2. *DM-37x# mount -t yaffs2 /dev/mtdblock5 /mnt/flash* Mount the NAND filesystem partition so that */mnt/flash* points to the root filesystem ('/') on NAND
3. *DM-37x# cp /root/YourApp /mnt/flash/root/* Copy an application from NFS to NAND



### 5.2.3 3000-OEM: Booting from NFS

Change the U-boot environment variables to boot from NFS. From the 3000-OEM U-Boot console, enter the following commands to change the boot defaults. *After completing these steps, the 3000-OEM will boot from NFS by default.*

1. `SLA3000 # setenv serverip <<server ip address>>` IP address of the Ubuntu virtual machine
2. `SLA3000 # setenv ipaddr <<ip address>>` Static IP address of the 3000-OEM (optional)
3. `SLA3000 # setenv bootcmd run nfsboot` Set the boot mode {nfsboot, nandboot}
4. `SLA3000 # setenv silent` Disable silent mode to allow interaction with Linux through serial 0
5. `SLA3000 # save` Save the parameters to NAND flash
6. `SLA3000 # boot` Continue loading the Linux kernel

### 5.2.4 3000-OEM: Access NAND Filesystem after NFS Boot

In some instances, it may be necessary to access the NAND filesystem after booting from NFS.

1. `root@sla3000:~# mount -w -o remount /` Remount the root NFS filesystem as R/W. The 3000-OEM filesystem is read-only by default. Replace `-w` with `-r` to remount as read-only.
2. `root@sla3000:~# mkdir /mnt/flash` Create a mount point (this directory may already exist)
3. `root@sla3000:~# ubiattach /dev/ubi_ctrl -m 4 -O 2048` -Create UBI device and attach underlying MTD device describing the raw flash memory
4. `root@sla3000:~# mount -t ubifs ubi0:rootfs /mnt/flash` Mount the NAND filesystem partition so that `/mnt/flash` points to the root filesystem ('/') on NAND
5. `root@sla3000:~# cp /home/root/YourApp /mnt/flash/home/root/` Copy an application from NFS to NAND

### 5.2.5 1500-OEM: Booting from NAND (Deploy Mode)

Change the U-Boot environment variables to boot from NAND (Deploy Mode).

1. `OMAP Logic # setenv bootcmd run nandboot` Set the boot mode {nfsboot, nandboot}
2. `OMAP Logic # setenv ipaddr` Optional: remove the IP address variable as this will be determined at runtime by VideoTrack
3. `OMAP Logic # setenv silent 1` Enable silent mode (not required, but recommend for final deployment)
4. `OMAP Logic # save` Save the parameters to NAND flash
5. `OMAP Logic # boot` Continue loading the Linux kernel



### 5.2.6 1500-OEM: Access Files on NFS after NAND Boot

Similar mapping of remote drives can be performed from the embedded Linux environment. To access files on the NFS share when booted from NAND:

1. `DM-37x# mount -o port=2049,nolock,proto=tcp -t nfs <<Linux host ip address>>:/tftpboot/sla1500fs /mnt/nfs`  
`/mnt/nfs` now points to the `~/sla1500fs` directory on the Ubuntu VM.
2. To copy YourApp from NFS to the `/root` directory on NAND:  
`DM-37x# cp /mnt/nfs/root/<<YourApp>> /root`

### 5.2.7 3000-OEM: Booting from NAND (Deploy Mode)

Change the U-Boot environment variables to boot from NAND (Deploy Mode).

1. `SLA3000 # setenv bootcmd run nandboot` Set the boot mode {nfsboot, nandboot}
2. `SLA3000 # setenv ipaddr` Optional: remove the IP address variable as this will be determined at runtime by VideoTrack
3. `SLA3000 # setenv silent 1` Enable silent mode (not required, but recommend for final deployment)
4. `SLA3000 # save` Save the parameters to NAND flash
5. `SLA3000 # boot` Continue loading the Linux kernel

### 5.2.8 3000-OEM: Access Files on NFS after NAND Boot

Similar mapping of remote drives can be performed from the embedded Linux environment. To access files on the NFS share when booted from NAND:

1. `SLA3000 # mount -w -o remount /`
2. `SLA3000 # mount -o port=2049,nolock,proto=tcp -t nfs <<Linux host ip address>>:/tftpboot/sla3000fs /mnt/nfs`  
`/mnt/nfs` now points to the `~/sla3000fs` directory on the Ubuntu VM. To copy YourApp from NFS to the `/root` directory on NAND:  
`SLA3000 # cp /mnt/nfs/root/home/<<YourApp>> /home/root`

### 5.2.9 1500-OEM and 3000-OEM: Access Files on NAND through SCP Protocol

The NAND filesystem can be accessed from Windows or Linux systems using the SCP protocol. Free GUI SCP clients such as [WinSCP](#) are available for Windows. The `scp` command is included in many common Linux distributions.

See [EAN-Using-WinSCP](#) for additional information.

 For the 3000-OEM, remount the filesystem as R/W before connecting (`mount -w -o remount /`).



## 6 Start Development


1. Start VMware Player and open the UbuntuSLA VMware image.

 *If the Update Manager window pops up, close the window.*

2. Apply power to the SLA-hardware.
  - a. Review the [Debugging and Deployment](#) section. Ensure that U-Boot is configured to boot from NFS by default.
  - b. Login to the embedded Linux environment:
    - 1500-OEM: both username and password are *root*.
    - 3000-OEM: username is *root* (no password).
  - c. The system may attempt to launch VideoTrack at startup. The first time that VideoTrack is launched, it will fail to start because the license file is not present. Skip to the next step.
  - d. If VideoTrack starts successfully, click *Enter* to stop the process.
3. The license file is required to run VideoTrack from NFS. Copy the license file from NAND to NFS.

1500-OEM:

- a. `DM-37x# mkdir /mnt/flash`
- b. `DM-37x# mount -t yaffs2 /dev/mtdblock5 /mnt/flash`
- c. `DM-37x# cp /mnt/flash/root/*.license /root/`

 *Reboot the 1500-OEM after copying the license file.*

3000-OEM:

- a. `root@sla3000:~# mount -w -o remount /`
- b. `root@sla3000:~# ubiattach /dev/ubi_ctrl -m 4 -O 2048`
- c. `root@sla3000:~# mount -t ubifs ubi0:rootfs /mnt/flash`
- d. `root@sla3000:~# cp /mnt/flash/home/root/*.license ~/`

 *Reboot the 3000-OEM after copying the license file.*

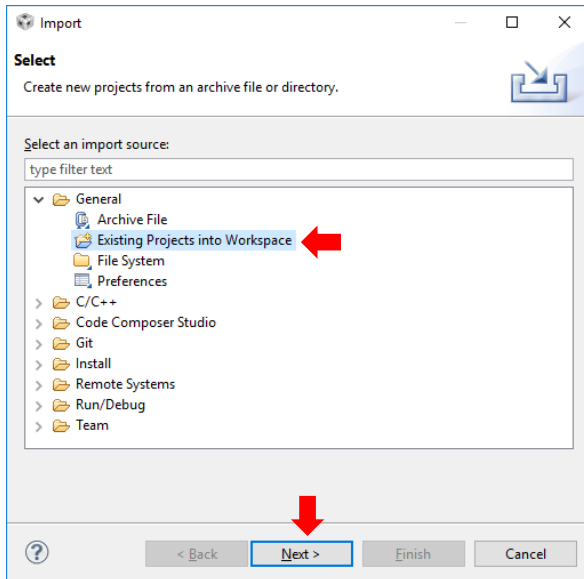
### 6.1 Building the Application

1. Import the project:
  - a. Open the Code Composer Studio application.
  - b. Select Workspace `C:\SightLine Applications\SLA-Examples-Arm ***\workspace\`.
  - c. *The application could be any of the sample applications or a custom application.*
  - d. Select *View » Project Explorer*.
  - e. Select *File » Import*.





- f. Expand *General*, select *Existing Projects into Workspace*, and click *Next*.



- g. In the *Import Projects* dialog, click *Browse...* and navigate to *(C:\SightLine Applications\SLA-Examples-Arm \*\*\*\workspace\)*. Select the desired project directory and click *Finish*.

A predefined CCS 5.x preferences file (*ccs\_eclipse\_preference.epf*) is located in the workspace folder. Import this file via the *Import Preferences* dialog. Expand *General* and select *Preferences*.

When importing preferences, ensure that *Import all* is selected, and no other options are checked.

## 2. Build the project:

- a. Right click on a project, then select *Build Project*.
- b. Test the application. From the embedded Linux console:
  - `DM-37x# ./YourApp_Debug`
  - For debugging, see the [Using GDB Debugging in CCS](#) in the Appendix.

Windows often fails to automatically re-establish mapped drive connections after rebooting or resuming from sleep. If the build fails because *R:* is not accessible, navigate to the root directory of the mapped drive via file explorer to ensure that the connection is active.

With an executable project that depends on another static library project, editing a file in the static library project may not cause the executable to be rebuilt. It is safer to manually build each project (the file may need to be edited in the executable project to correctly rebuild the executable).



## 6.2 Deploying the Application

Copy the custom application from NFS to NAND. See the [Debugging and Deployment](#) section for detailed instructions

### 6.2.1 1500-OEM Startup Script

When the 1500-OEM starts, it executes `/etc/rc.d/rc.local` at the end of initialization. If the application needs to start automatically edit the script file accordingly.

The `rc.local` file exists both on NFS and NAND. Edit the script on the NAND for deployment. It may be necessary to insert a 10-20 second `sleep` before executing the custom application. This ensures that VideoTrack is running when the application is launched.

Near the end of this script there is an example of delaying then starting `rtspMain`. This can be copied and modified to start the application with an appropriate delay. Building a release version of the application for deployment would be beneficial.

### 6.2.2 3000-OEM Startup Script

When the 3000-OEM starts, it executes `/home/root/sla3000_init.sh` at the end of initialization. If the application needs to start automatically edit the script file accordingly.

The `sla3000_init.sh` file exists both on NFS and NAND. Edit the script on the NAND for deployment. It may be necessary to insert a 10-20 second `sleep` before executing the custom application. This allows time for VideoTrack to start before the application is launched.

The SLA3000 file system must be changed to writeable before editing the script. Near the end of this script there is an example of delaying then starting `rtspMain`. This can be copied and modified to start the application with an appropriate delay. Building a release version of the application for deployment would be beneficial.

```
mount -w -o remount /          # To make the filesystem writable.
```

### 6.2.3 Configuring U-Boot to Boot from NAND

See the Deploy Mode section for the [1500-OEM](#) or [3000-OEM](#) on how to boot from NAND.



## 7 Troubleshooting

### 7.1 General Troubleshooting

Applications may not function as expected when deployed to the SLA-hardware. The VideoTrack application can operate differently when booted from NFS or NAND.

Check the param\*.txt file in the /root (1500-OEM) or /home/root (3000-OEM) directory. This file stores VideoTrack configuration settings.

Verify that the same param\*.txt (parameter) file is present on NFS and NAND. Deleting or renaming this file may help. This resets the system to factory default parameters.

- ✓ Ensure that the NAND and NFS filesystems contain matching board firmware versions.
- ✓ Disable silent mode before running applications (setenv silent from the U-Boot prompt).
- ✓ For long command line commands, create a shell (.sh) script that executes them for ease of use, e.g., starting GDB server on target during debugging, mounting NFS/NAND filesystems, etc.
- ✓ Verify that a valid license file has been copied from the SLA-hardware to the NFS filesystem.
- ✓ Make sure all file permissions are set properly (chown, chmod a+x, etc.).

### 7.2 NFS Boot Troubleshooting

This section describes how to troubleshoot problems if the hardware NFS boot is failing. Use the following methods in the order in that they are presented (most common to least common).

#### 7.2.1 Wifi Adapter Conflicts

If using a WiFi adapter it may be conflicting with the wired adapter. Try disabling the WiFi adapter. If this corrects the problem, try disabling bridging to the WiFi adapter (**Figure 3**). You should then be able to reenale the WiFi adapter and not have issues with NFS Boot/VMware.

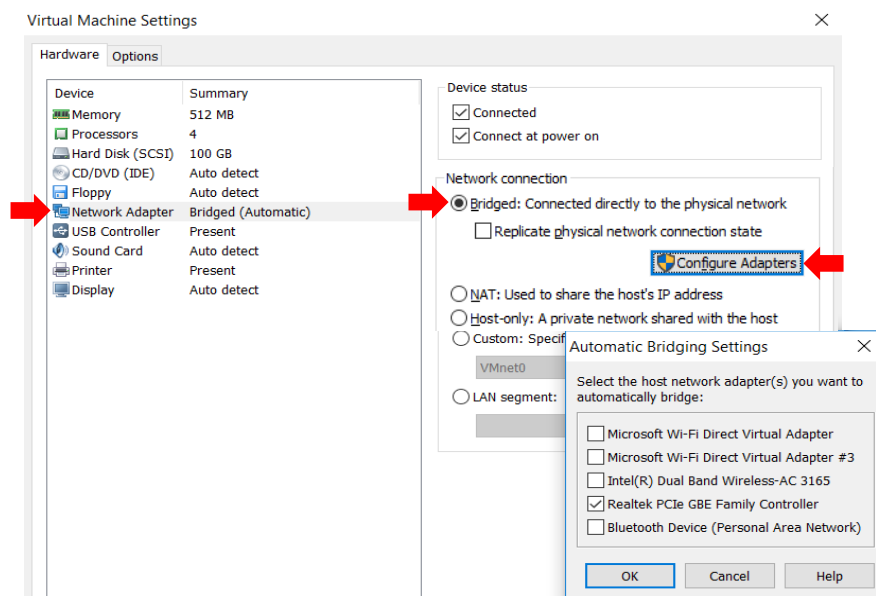


Figure 3: Disable Bridging



### 7.2.2 IP Address Conflicts or Mismatch

1. Open a terminal window on the VMware machine and type *ifconfig*. This shows the IP address of the VMware server (*serverip*). This can be a DHCP address or a static IP address (if setting a static IP using the instructions in [Setup R: Drive](#)).
2. Check the IP address that the system is trying to boot from. Break into the boot using shift-S, and then type: `printenv serverip`

*Make sure this matches the IP address of the VMware (serverip).*

3. Check the SLA-hardware IP address. Type: `printenv ipaddr`

*This ipaddr must be on the same subnet as the serverip for the NFS boot to work. If there is a mismatch it will prevent the NFS boot. As an example of a mismatch, the TFTP IP server address is 192.168.1.46, and the SightLine IP address is 192.168.3.99.*

4. Open a cmd window on the PC and *ping* the *serverip* address.
5. For the 3000-OEM, use shift-S to break into the boot sequence, and then *ping* the *serverip* address from the uboot prompt. This feature is not available on the 1500-OEM

### 7.2.3 Windows 10 Update Issues

If there has been Windows 10 update, it can remove the VMware Bridge protocol and prevent the VMWare image from coming up with an IPV4 address after the update. Use the following steps to restore functionality.

1. Go to *Control Panel » Network and Internet » Network Sharing Center » Change adapter settings*.
2. Click on *VMware Network Adapter* and choose *Properties*.
3. Check *VMware Bridge Protocol*.
4. Reboot the PC.

### 7.2.4 Reinstall VMware Application

Windows update has been found to remove or disable drivers. Try reinstalling the VMWare application following a Windows update.

### 7.3 Shift+S Interrupt Issues

Shift+S interrupt issues most often occur with USB to RS-232 adapters. PCI controllers can also cause problems. Prolific PL2303-based USB to RS-232 adapters have performed well during tests. While Shift+S issues are infrequent with these adapters, they can still happen. This may be due to an inherent problem in Windows.

Try unplugging the USB to RS-232 adapters from the USB port. Wait ~10 seconds for Windows to recognize that the adapter is unplugged. Plug the adapter back in. Restart the terminal emulator. If this does not work restart the PC. In most cases this should fix the problem.



### Shift+S notes:

- The default baud rate for the boot loader console is 115200. This is defined by the baud rate U-Boot environment variable.
- When silent mode is enabled, the VideoTrack application changes the baud rate for *Serial Port 0* during the initialization process (default 57600).
- VideoTrack does not change the baud rate when silent mode is disabled. The Linux serial console uses the U-Boot baud rate.
- During SightLine testing, there have not been any issues with adapters using the default Windows device settings (normally 9600/8-N-1) and configuring the port in the terminal emulator.
- The terminal window is shown in [Figure 4](#) Immediately after entering the U-Boot console. The 3000-OEM uboot prompt is: SLA3000#

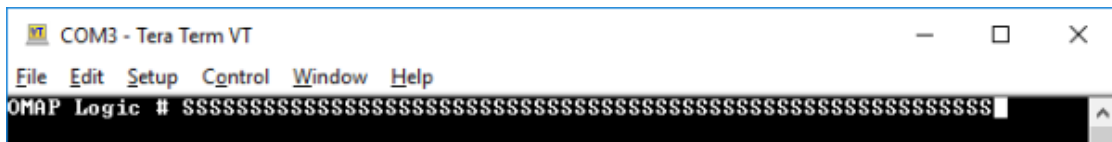


Figure 4: Shift+S in Tera Term

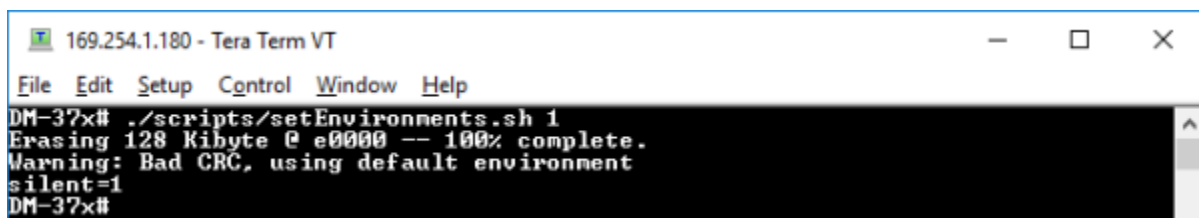
- The OMAP Logic # prompt should appear in the terminal window within ~5 seconds of applying power to the board. You can release the key combination as soon as the prompt appears.

### Reset U-boot Environment variables:

It is also possible that one or more of the related U-Boot environment variables have been changed. If you are early in the ARM development process, reset the U-Boot environment variables to default using the following steps:

1. Use Tera Term to establish an SSH session to the target.
2. Enter *root* as the username and password.
3. From the DM-37x# prompt, type:

```
./scripts/setEnvironments.sh 1 (3000 prompt will be root@sla3000:~#)
```



4. Power cycle the board and test the Shift+S interrupt.

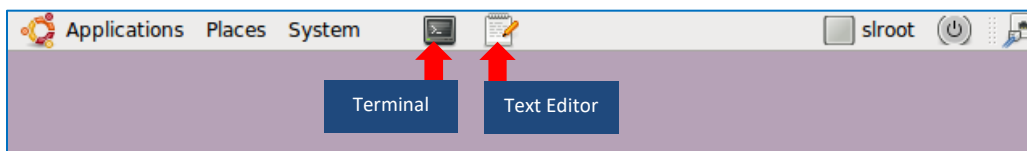
## 7.4 Questions and Additional Support

For questions and additional support, please contact [Technical Support](#). Additional support documentation and Engineering Application Notes (EANs) can be found on the Support pages of the SightLine Applications [website](#).

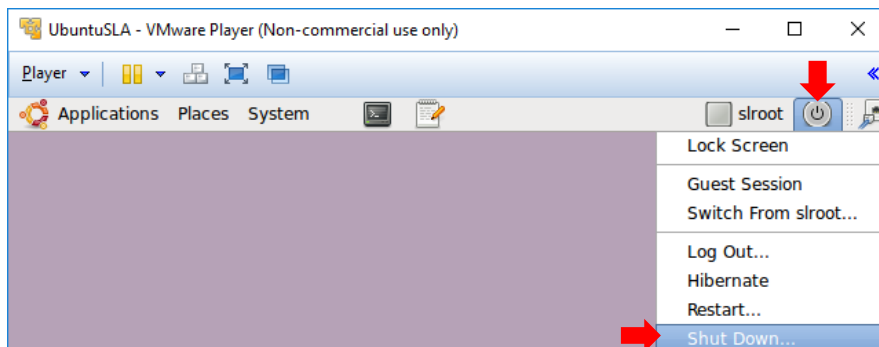


## Appendix A - Using VMware

1. From the *Player* drop-down menu select *File » Open*.
2. Navigate to the UbuntuSLA\*\*\* directory extracted from the downloaded archive, select UbuntuSLA.vmx, and click *Take Ownership* if prompted.
3. Select the UbuntuSLA virtual machine and click the green triangle button to start the VM.
4. Select *I copied it*. VMware will update the virtual machine configuration and continue booting.
5. At the login window, click on the UbuntuSLA user.
6. Enter password: *slroot* (username is also *slroot*). The virtual machine is now ready to use. Shortcuts to a terminal application and text editor (*gedit*) are available in the menu bar.

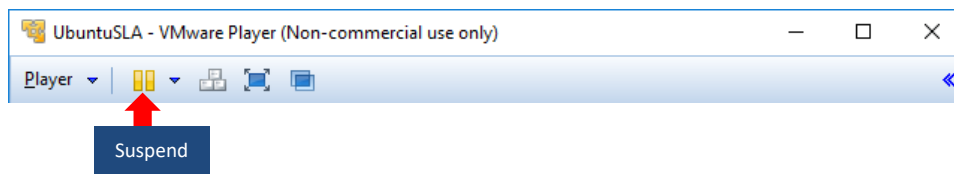


7. To shut down the Virtual Machine, select *Shut Down* from the power button drop-down menu.



### Disable Sleep and Hibernate (optional):

Disabling sleep and hibernate prevents accidentally suspending the system from the Ubuntu GUI. VMware Player provides a suspend function that can be used to save the system state.

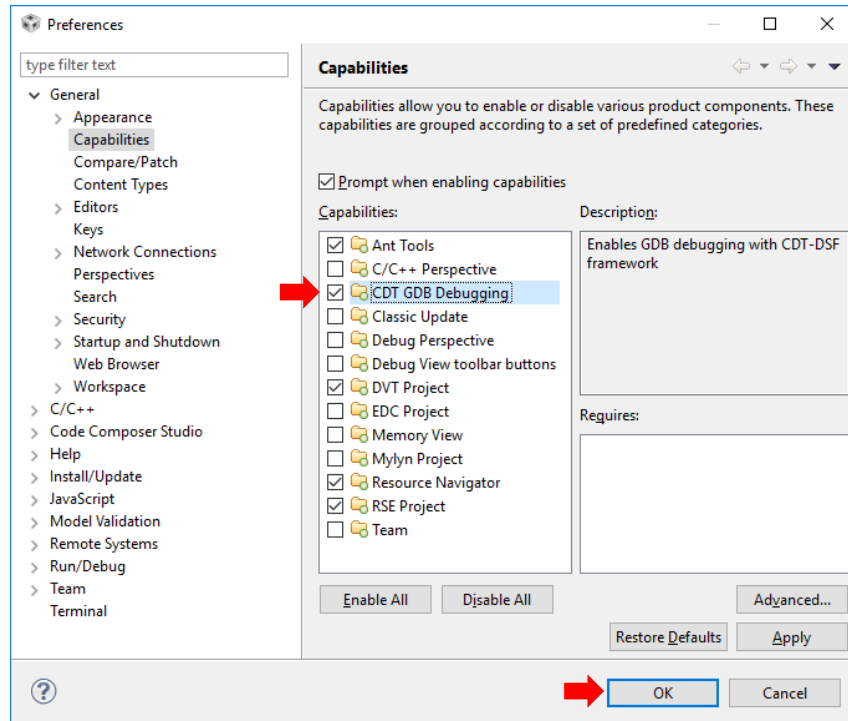


1. `host> sudo gedit /usr/share/polkit-1/actions/org.freedesktop.upower.policy`
2. Enter password: *slroot*
3. Change both instances of `<allow_active>yes</allow_active>` to `<allow_active>no</allow_active>`
4. Save changes and restart the virtual machine.

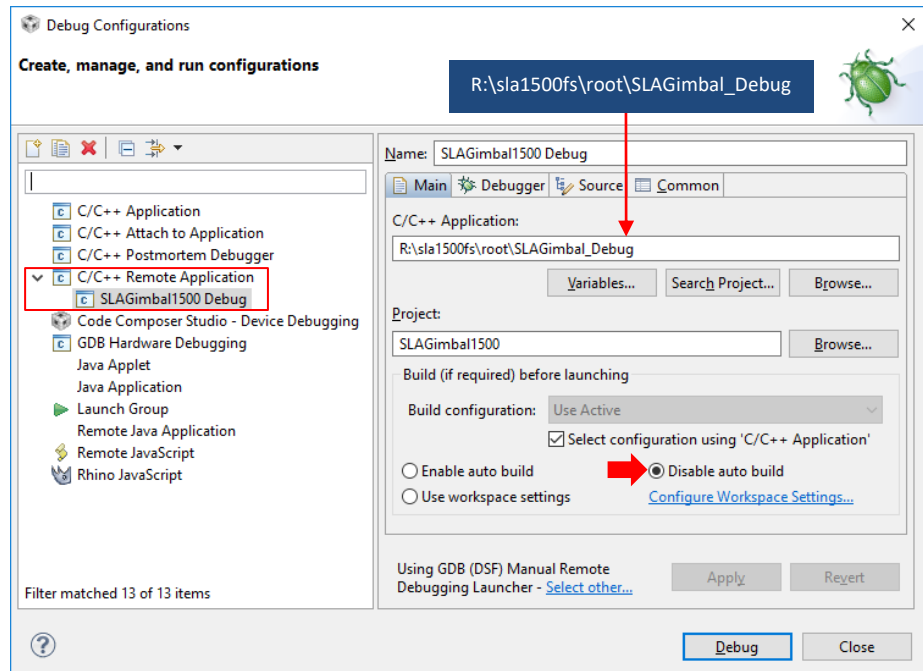


## Appendix B - Using GDB Debugging in CCS

1. From the *Window* drop-down menu in Code Composer Studio select *Preferences*.
2. Expand the *General* menu in the sidebar tree. Check *CDT GDB debugging* in the *Capabilities* list. Click *OK* to proceed.

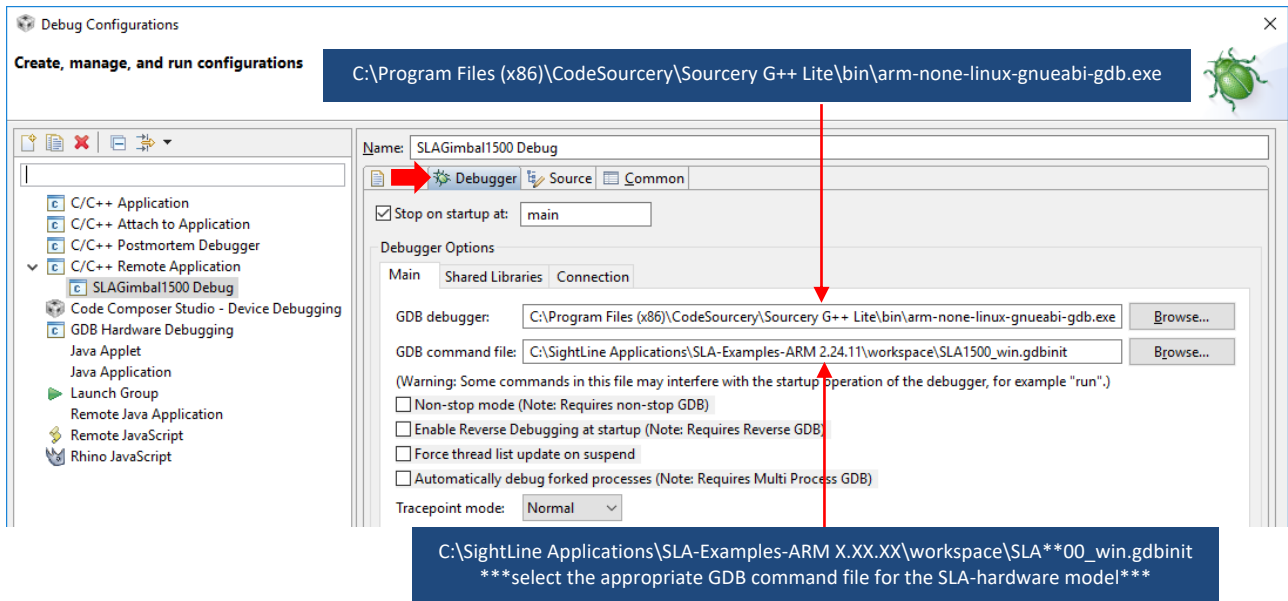


3. SLAGimbal1500 is used in this an example. Right click on *SLAGimbal1500* and select *Debug As » Debug Configurations*. Right click on *C/C++ Remote Application*, select *New*, and configure as shown.



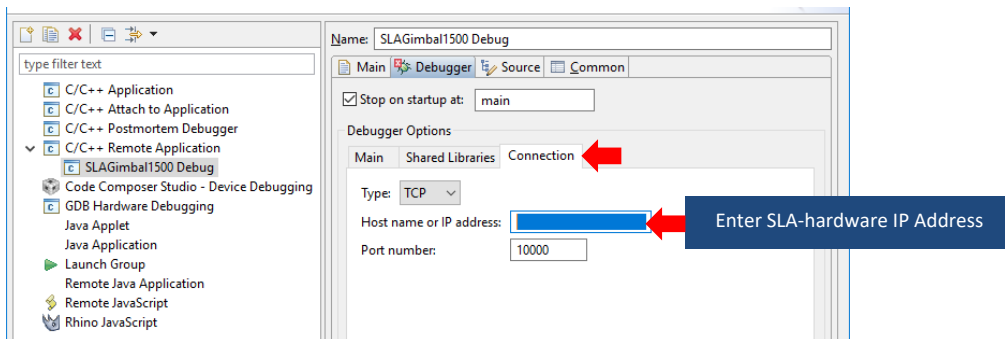


4. Select the *Debugger* tab and copy the configuration as shown.



5. Under *Debugger Options* select the *Connection* tab and enter the IP address of the SLA-hardware

To determine the IP address, enter the *ifconfig* command at the embedded Linux console.



6. Start the GDB server on the SLA-hardware:

- a. `DM-37x# gdbserver :10000 SLAGimbal_Debug`
- b. Click *Debug* in the *Debug Configurations* dialog to start the debugging process.

When the execution is stopped at a breakpoint, CCS may not find the correct thread in the Debug window (usually at the top left corner). If this occurs, selecting the correct thread in the Debug window (which is in Suspended mode) will enable the Resume and Step functions.





## Appendix C - Updating the VMware Environment (2.23.1 or later)

It is important to keep the development environment up to date as SightLine releases new firmware. This section explains how to update the firmware on the Ubuntu virtual machine.

Prerequisites:

- UbuntuSLA VMware image for 2.23.1 (or later) release
- SLA-1500 / SLA-3000 Upgrade Utility installed

### UbuntuSLA Image Version

From a terminal or SSH session on the Ubuntu VM, enter: `ls ~/Downloads/sla1500`

If the directory exists, the image version is 2.23.1 or later.

### Copy Firmware from Program Files

This section refers to hardware-specific files and folders. In the following statements, replace **\*\*** with the first two digits of the numeric portion of SLA-hardware model number, e.g. *15* for the 1500-OEM or *30* for the 3000-OEM.

1. Install the firmware Upgrade Utility on the host PC – the default installation directory is:  
C:\Program Files (x86)\SightLine Applications\
2. Copy the *firmware* directory (under *SLA-\*\*00 Upgrade Utility x.xx.xx*) to: R:/Downloads/sla\*\*00/

### sla\_updateDevEnv.sh

1. `host> cd ~/Downloads/sla**00/firmware`
2. Run the *sla\_updateDevEnv.sh* script. There are two methods for updating the environment:
  - a. Incremental update – some files overwritten, most remain unchanged:

```
host> ./sla_updateDevEnv.sh update
```

- b. Reset the environment to the factory default state:


```
host> ./sla_updateDevEnv.sh factory
```

This method completely deletes the existing filesystem including user-created files. Backing up files is recommended even when using the various options to preserve the license file, root filesystem, etc.

The *factory* option is recommended when downgrading to an earlier release (≥2.23.1).

```
 host> ./sla_updateDevEnv.sh (# displays help).
```

3. Follow the instructions and select the desired options.
4. A confirmation message will be displayed when the process is complete.

 *The script may also display a **ToDo:** message.*



## ToDo Message

If a *ToDo* message is displayed, additional steps are required to complete the development environment update. The instructions may be different from release to release. It is not necessary to perform the indicated steps more than once for a given firmware version.

Example *ToDo* message:

```
ToDo: *** run the following commands on the 3000-OEM ***  
On the serial Terminal, hold Shift-S to break into u-boot, then power cycle  
3000.  
At u-boot prompt (SLA3000#) set NFS server by: set serverip xx.xx.xx.xx  
Start NFS boot: run nfsboot  
Logon to 3000 (username=root, no password).  
mount -w -o remount /           # To make the filesystem writable.  
/etc/init.d/38xx-demo          # To initialize graphics libraries.
```

## Enable SMBv1 Support

1. Use the WinKey(**⊞**)+R to open a *Run* prompt.
2. Enter *appwiz.cpl* to open the Programs and Features control panel app.
3. Select *Turn Windows features on or off* from the sidebar.
4. Expand *SMB 1.0/CIFS File Sharing Support*.
5. Check *SMB 1.0/CIFS Client* and *SMB 1.0/CIFS Automatic Removal*.

