DEVELOPMENT GUIDE

Basler dart BCON for LVDS Development Kit

Document Number: AW001435
Version: 03 Language: 000 (English)
Release Date: 02 August 2018

BASLER
the power of sight

Contacting Basler Support Worldwide

Europe, Middle East, Africa

Basler AG
An der Strusbek 60–62
22926 Ahrensburg
Germany

Tel. +49 4102 463 515
Fax +49 4102 463 599

support.europe@baslerweb.com

The Americas

Basler, Inc.
855 Springdale Drive, Suite 203
Exton, PA 19341
USA

Tel. +1 610 280 0171
Fax +1 610 280 7608

support.usa@baslerweb.com

Asia-Pacific

Basler Asia Pte. Ltd.
35 Marsiling Industrial Estate Road 3
#05–06
Singapore 739257

Tel. +65 6367 1355
Fax +65 6367 1255

support.asia@baslerweb.com

www.baslerweb.com

Table of Contents

1 About this Document

This document provides technical information about the Basler dart BCON for LVDS Development Kit. The document describes the hardware and software components and tells you how to use the dart BCON for LVDS Development Kit Board Support Package (BSP). The BSP allows you to develop your own embedded vision solutions by customizing the software provided on the microSD card.

1.1 Target Audience

The Basler dart BCON for LVDS Development Kit is aimed at experienced hardware and software engineers proficient in electronics, software development, and embedded system design.

This document is written for a target audience that has intermediate to advanced technical skills in the areas mentioned above.

The document assumes that users have experience in the following areas:

- System on Chip (SoC) or System on Module (SoM) architectures
- Serial communication
- FPGA programming
- Development of frame grabbing procedures
- Bus architectures such as I²C
- Embedded Linux operating systems

1.2 Typographic Conventions

Font	Element or Symbol	Example
Arial, bold	Program interface such as menu commands, windows, or button names; file names; path names	Press the **OK** button.
`Consolas`	Command line output	`Command (m for help):`
`Consolas, orange`	Command line input	`$ umount /dev/sdb1`
`Consolas, blue`	Emphasis (importance) in command line output	`/dev/sdb1 on /media/$USER/DB05-047F type vfat`

2 Hardware Overview

2.1 Delivered Parts

The Basler dart BCON for LVDS Development Kit contains the following components:

- Basler dart daA2500-14lc camera, S-mount
- Evetar lens M12B0816W ƒ/1.6, f8 mm, 1/2"
- Processing board (Avnet MicroZed 7010 SoM)
- Carrier board (Basler MicroZed BCON for LVDS carrier card)
- microSD card (preloaded with the development kit image, see Section 3.1)
- Flexible flat cable, 0.2 m
- USB 2.0 cable (type A plug to Micro B plug), 1.8 m
- GigE cable Cat 5e, 2.0 m
- Power supply, 15 W, 5 VDC @ 3 A
- International power cable plugs
- Mounting kit (including camera mounting plate, spacer bolts, screws, and nuts)
- Quick Install Guide

For hardware specifications, see the *Quick Install Guide*.

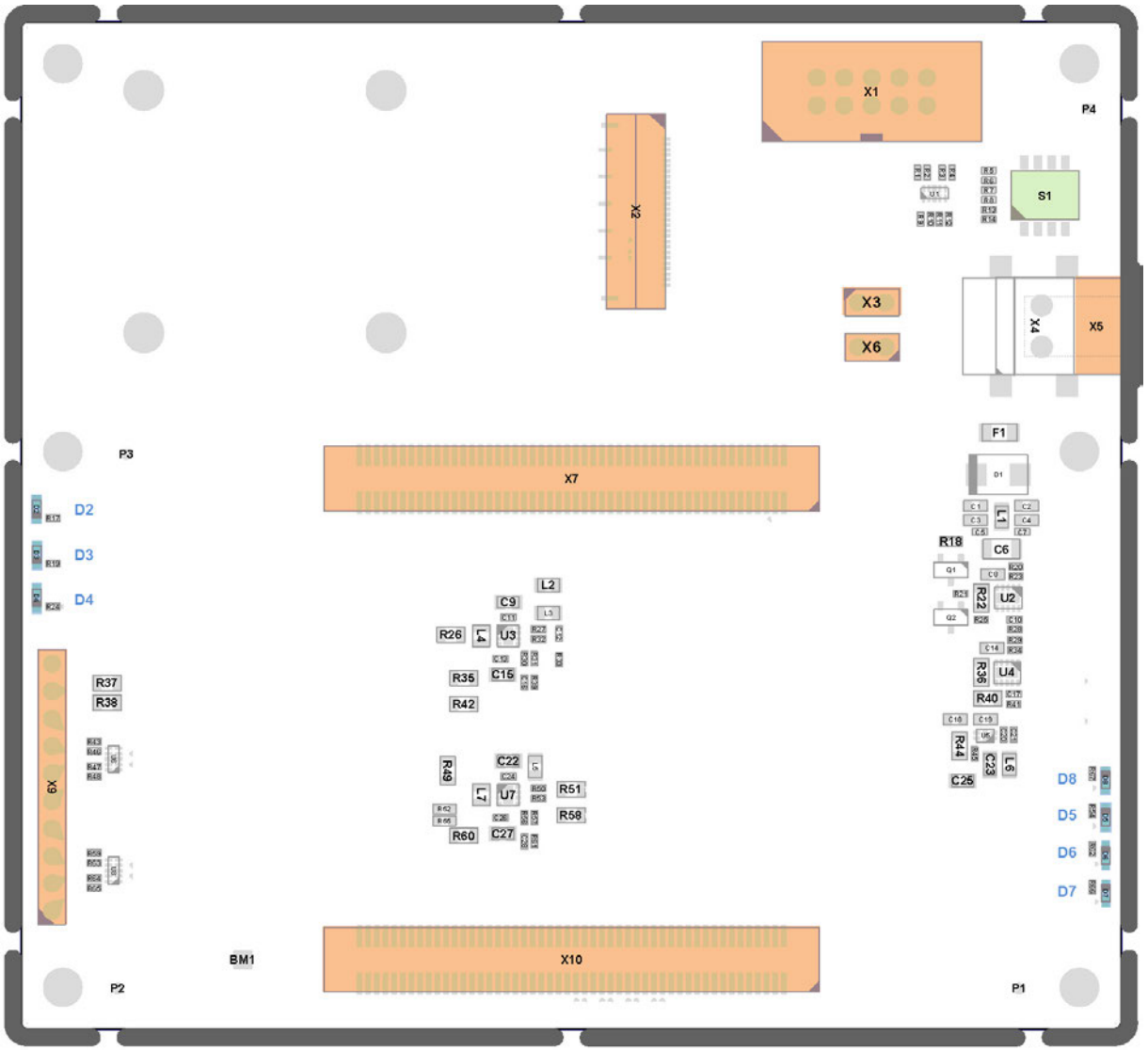
2.2 Installation

For information about installing the Basler dart BCON for LVDS Development Kit, see the *Quick Install Guide* delivered with the development kit. The document is also available on the Basler website: www.baslerweb.com/en/sales-support/downloads/document-downloads/dart-bcon-for-lvds-development-kit-quick-install-guide.

2.3 Avnet MicroZed 7010 SoM Processing Board

For detailed information about the Avnet MicroZed board, see the *MicroZed Hardware User Guide*. You can download the document from zedboard.org/support/documentation/1519.

2.4 Basler MicroZed BCON for LVDS Carrier Card



2.4.1 Connectors

Connector	Pin	Designation	Notes
X1	1	GPIO[0]	
	2	5V	
	3	GPIO[1]	
	4	5V	
	5	GPIO[2]	
	6	3.3V	
	7	Ext_Trigger_In	
	8	GND	
	9	GND	
	10	GND	

Connector	Pin	Designation	Notes
X9	1	GPIO[3]	
	2	GPIO[4]	
	3	GPIO[5]	
	4	GPIO[6]	
	5	GPIO[7]	
	6	GPIO[8]	
	7	GPIO[9]	
	8	GPIO[10]	
	9	3.3V	Changeable to 5 VDC by moving R38 to R37
	10	GND	
X3	1	I2C_SCL	BCON for LVDS I^2C
	2	I2C_SDA	BCON for LVDS I^2C
X6	1	I2C_ID	
	2	GND	
X5		-	5 VDC power supply (GND outside)
X2		-	FFC connector to BCON for LVDS camera
X7		to MicroZed Board JX2	
X10		to MicroZed Board JX1	

2.4.2 LEDs

LED	Designation	Notes
D2	USER_LED[2]	User LED 2, green
D3	USER_LED[1]	User LED 1, green
D4	USER_LED[0]	User LED 0, green
D8	PG_5V	Power good 5 VDC carrier board supply, green
D5	PG_5.0V_CAM	Power good 5 VDC camera supply, green
D6	PG_2.5V[a]	Power good 2.5 VDC supply, green
D7	PG_3.3V[b]	Power good 3.3 VDC supply, green

[a] Falsely designated as “PG_3.3V” on revision 2 of the carrier board.
[b] Falsely designated as “PG_2.5V” on revision 2 of the carrier board.

2.4.3 DIP Switch S1

Switch S1	State	Description
SW[0]_high	On (default)	Set user switch 0 to high
	Off	Set user switch 0 to low
SW[1]_high	On (default)	Set user switch 1 to high
	Off	Set user switch 1 to low
Trig_Out	On (default)	Use trigger generator controlled by software
	Off	Disable trigger generator
I2C_ID	On (default)	Use I²C ID configurable by software (0 or 1)
	Off	Use I²C ID of 1

3 Software Overview

3.1 Basler dart BCON for LVDS Development Kit Image

The dart BCON for LVDS Development Kit Image provides a preinstalled Linux 32-bit operating system that consists of the following main components:

- A root file system taken from the Ubuntu 16.04 LTS distribution. The root file system can be maintained using the standard Ubuntu package management tools (e.g. apt, dpkg, or synaptic).
- Integrated into the root file system:
  - An installation of the Basler pylon Camera Software Suite, including sample code
  - User space tools and libraries providing access to the BCON for LVDS camera and associated hardware (see Section 3.1.1)
- A custom-built Linux kernel with associated hardware driver modules
- A boot loader (UBOOT) to load and start the Linux kernel

You can download the image file **basler_dart_bcon_for_lvds_devkit_2016_3-X.Y.Z.img.gz** from www.baslerweb.com/en/sales-support/downloads/software-downloads/dart-bcon-for-lvds-development-kit-image.

Alternatively, you can create your own image file using the Board Support Package (BSP).
For more information about the BSP, see Section 3.2 and Chapter 6.

The image file can then be loaded onto the microSD card delivered with the development kit. For more information, see Section 5.

3.1.1 Tools and Libraries

The following tools and libraries are integrated into the root file system of the development kit image. The source code is also available as part of the Board Support Package.

BCON Control Tool

The BCON Control Tool **bconctl** allows you to turn the camera on and off, toggle the I²C address of the camera, trigger the camera, or turn the LEDs on the board on or off.

The prebuilt application is located in **/usr/bin**. You can use the command line help to find out more.

The source code is located in **/opt/bcon/sources/bconctl**.

The tool uses the BCON Control Library (see below).

BCON Control Library

The BCON Control Library API allows you to use the functionality provided by the BCON Control Tool in your own application.

The prebuilt library is located in **/usr/lib**.

The source code of the library is located in **/opt/bcon/sources/libbconctl**. The source code features a Doxygen style documentation.

The API of the BCON Control Library is described in the **basler/bconctl.h** header file, which is located in **/usr/include/**.

BCON Grab Sample

The BCON Grab Sample is a sample application written specifically for the Basler dart BCON for LVDS Development Kit. It makes use of the onboard camera trigger generator.

The prebuilt application is located in **/usr/bin**.

The source code is in **/opt/bcon/sources/bcongrab**. This sample uses pylon and the BCON Control Library.

3.1.2 pylon Samples

The pylon samples allow you to build and evaluate sample code using the pylon API. For more information, see the installed *C Programmer's Guide and Reference Documentation* and *C++ Programmer's Guide and Reference Documentation*.

The pylon sample code and the Programmer's Guides are located in the home directory of the default user “ubuntu”.

The following samples are applicable for BCON for LVDS cameras:

- Samples/C++/Grab
- Samples/C++/Grab_UsingBufferFactory
- Samples/C++/ParametrizeCamera_AutoFunctions
- Samples/C++/ParametrizeCamera_GenericParameterAccess
- Samples/C++/ParametrizeCamera_LoadAndSave
- Samples/C++/ParametrizeCamera_NativeParameterAccess
- Samples/C++/ParametrizeCamera_UserSets
- Samples/C++/Utility_Image
- Samples/C++/Utility_ImageFormatConverter
- Samples/C++/Utility_ImageLoadAndSave
- Samples/C/GenApiParam
- Samples/C/OverlappedGrab
- Samples/C/ParametrizeCamera
- Samples/C/SimpleGrab

To build the pylon samples, use the included makefiles. Example:

```
$ cd pylon-*-armhf/Samples/C++/Grab && make
```

3.2 Basler dart BCON for LVDS Development Kit Board Support Package

The dart BCON for LVDS Development Kit Board Support Package (BSP) allows you to customize the components of the microSD card image or check the associated source files.

The package contains an Ubuntu root file system, a custom-built Linux kernel plus matching device drivers, a boot loader, a Vivado FPGA project, and all source files required to modify and rebuild the image file.

You can download the BSP from www.baslerweb.com/en/sales-support/downloads/software-downloads/basler-dart-bcon-for-lvds-development-kit-board-support-package. For more information, see Chapter 6.

4 Connecting to the Board via Remote Access

4.1 Login Credentials

To log in to the preinstalled Ubuntu Linux 32-bit operating system, use the following credentials:

User: ubuntu
Password: ubuntu

4.2 Text Console via SSH

The preinstalled system features an SSH server that you can connect to using a suitable client program, e.g. PuTTY if you are on a Windows machine. This will give you a text-mode command shell.

4.3 Windows Remote Desktop Connection

For easy use without any software installation, you can connect to the processing board via Windows Remote Desktop (RDP) connection.

The development kit supports running an X11rdp session.

For more information, see the *Quick Install Guide* delivered with the Basler dart BCON for LVDS Development Kit. The document is also available on the Basler website: www.baslerweb.com/en/sales-support/downloads/document-downloads/dart-bcon-for-lvds-development-kit-quick-install-guide.

4.4 Remote X-Server for Windows

For optimum performance, you can download and install an X-server for Windows, e.g. VcXsrv, available at sourceforge.net/projects/vcxsrv.

The development kit supports running an XDMCP session.

5 Flashing the microSD Card

5.1 Windows

To flash the image file to the microSD card:

1. Download the image file from www.baslerweb.com/en/sales-support/downloads/software-downloads/dart-bcon-for-lvds-development-kit-image or create your own image file using the Board Support Package (see Section 6.5).
2. Extract the image file, e.g. by using 7-Zip, available at www.7-zip.org.
3. Download and install the Win32 Disk Imager tool, available at sourceforge.net/projects/win32diskimager.
4. Open Win32 Disk Imager, verify the drive letter matches that of the microSD card to write, and select the uncompressed image file.
5. Click the **Write** button.

On Linux, you can also copy individual components to the microSD card. This gives you greater flexibility. For more information, see Section 5.2.3.

5.2 Linux

5.2.1 Locating the microSD Card Device

To locate the microSD card device:

1. Make sure the microSD card isn't inserted.
2. Use the following command to check the existing device nodes:
   ```
   $ ls /dev/sd*
   ```
3. Take note of all device nodes named **/dev/sd***.
4. Insert the microSD card.
5. Repeat step 2 to check for new device nodes.
   A new device node should be displayed, e.g. **/dev/sdb**. If the microSD card already has partitions on it, these will also be represented by device nodes, e.g. **/dev/sdb1** and **/dev/sdb2**. If the partitions contain file systems, the file systems may have been mounted automatically, depending on your development system's configuration.

The following example shows how to determine the device node for a card containing a single partition:

```
$ ls /dev/sd*
/dev/sda /dev/sda1 /dev/sda2
```

[microSD card inserted]

```
$ ls /dev/sd*
/dev/sda /dev/sda1 /dev/sda2 /dev/sdb /dev/sdb1
```

In this example, the microSD card is represented by the **/dev/sdb** device node, with one partition **/dev/sdb1** present.

Using this information, you can now perform the following tasks:

- Flash the full image file to the microSD card (see Section 5.2.2).
- Copy individual components to the microSD card (see Section 5.2.3).

5.2.2 Flashing the Image File

To flash the image file to the microSD card:

1. Download the image file from www.baslerweb.com/en/sales-support/downloads/software-downloads/dart-bcon-for-lvds-development-kit-image or create your own image file using the Board Support Package (see Section 6.5).
2. Unmount all file systems that are present on the microSD card. Assuming there are two partitions carrying file systems, unmount them as follows:

```
$ umount /dev/sdb1
$ umount /dev/sdb2
```

3. Flash the image file to the microSD card:

```
$ zcat basler_dart_bcon_for_lvds_devkit_2016_3-X.Y.Z.img.gz | sudo dd of=/dev/sdb
```

5.2.3 Copying Individual Components

As an alternative to flashing the microSD card as described in Section 5.2.2, you may start with a blank card, stepwise assembling the contents from individual components. This gives you greater flexibility. For example, you can fully utilize cards larger than 4 GB, or you can even use smaller cards. Also, copying selected components is significantly faster than copying the full image file.

You will need the following components, which are in **images/linux** after you have performed a full software build as described in Section 6.2:

- The bootloader (**boot.bin**)
- The kernel (**image.ub**)
- The root file system (**rootfs.cpio.gz**)

5.2.3.1 Step 1: Partitioning the microSD Card

You have to perform this step only once.

In the example below, a 4 GB microSD card with a single FAT32 partition is being used and will be re-partitioned for the development kit.

To partition the microSD card:

1. Unmount any file systems present on that card that may have been mounted automatically. Assuming the microSD card contains a file system in the **/dev/sdb1** partition, execute the following command:

```
$ umount /dev/sdb1
```

2. Start **fdisk**:

```
$ sudo fdisk /dev/sdb
```

3. Create the BOOT partition (FAT32, 128 MB) by following the procedure given below:

```
Command (m for help): d
Selected partition 1
Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): <ENTER>
Partition number (1-4, default 1): <ENTER>
Using default value 1
First sector (2048-7744511, default 2048): <ENTER>
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-7744511, default 7744511): +128M

Command (m for help): t
Selected partition 1
Hex code (type L to list codes): c
Changed system type of partition 1 to c (W95 FAT32 (LBA))
```

4. Stay in **fdisk** and create the ROOTFS partition (Linux) in the remaining unpartitioned space by following the procedure below:

```
Command (m for help): n
Partition type:
   p   primary (1 primary, 0 extended, 3 free)
   e   extended
Select (default p): <ENTER>
Partition number (2-4, default 2): <ENTER>
Using default value 2
First sector (264192-7744511, default 264192): <ENTER>
Using default value 264192
Last sector, +sectors or +size{K,M,G} (264192-7744511, default 7744511): <ENTER>
Using default value 7744511

Command (m for help): p
Disk /dev/sdb: 3965 MB, 3965190144 bytes
49 heads, 48 sectors/track, 3292 cylinders, total 7744512 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

```
Disk identifier: 0x00000000

Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048      264191      131072    c  W95 FAT32 (LBA)
/dev/sdb2          264192     7744511     3740160   83  Linux
Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

5.2.3.2 Step 2: Creating File Systems

You have to perform this step only once.

To create file systems on the partitions:

1. Execute the following commands:

```
$ sudo mkfs.vfat -n BOOT /dev/sdb1
$ sudo mkfs.ext4 -L ROOTFS /dev/sdb2
```

5.2.3.3 Step 3: Copying the Components

After partitioning the microSD card and creating file systems, you can copy individual components as desired.

The following examples assume that you extracted the BSP to the user's home directory (see Section 6.2). Also, it's assumed that your Linux system is configured to automatically mount file systems under **/media/$USER** upon insertion. If this is not the case, mount the file system manually and modify the "umount" command accordingly.

To copy the bootloader and kernel:

1. Execute the following commands:

```
$ sudo cp -t /media/$USER/BOOT ~/ basler_
dart_bcon_for_lvds_devkit_2016_3/images/linux/images/{boot.bin,image.ub}
$ umount /media/$USER/BOOT/
```

To copy the root file system (rootfs):

1. Execute the following commands:

```
$ sudo rm -rf /media/$USER/ROOTFS/*
```

```
$ cd /media/$USER/ROOTFS
$ zcat ~/basler_dart_bcon_for_lvds_devkit_2016_3/images/linux/images/rootfs.cpio.gz |
sudo cpio -idv --no-absolute-filenames
$ cd; umount /media/$USER/ROOTFS/
```

6 Using the Board Support Package (BSP)

The dart BCON for LVDS Development Kit Board Support Package (BSP) contains everything needed to re-create the software components installed on the microSD card delivered with the development kit.

You must install the required tools separately on a development system (a PC running a reasonably recent Linux 64-bit distribution). Because you must download the requisite tools, the development system needs internet access.

6.1 Installing PetaLinux

To rebuild the software components, such as the boot loader, Linux kernel, and root file system, you must install PetaLinux, available from the Xilinx website.

To install PetaLinux:

1. Set up an account and download the required version 2016.3 here: www.xilinx.com/member/forms/download/xef.html?filename=petalinux-v2016.3-final-installer.run&akdm=1
2. Call the **petalinux-v2016.3-final-installer.run** download file with a single argument, giving the path to the desired installation directory. For seamless execution of the scripts provided as part of the BSP, Basler recommends the following call:
   ```
   $ sudo petalinux-v2016.3-final-installer.run /opt/petalinux-v2016.3-final
   ```

6.2 Rebuilding the Development Kit Software

After you have installed PetaLinux, everything is set up to rebuild the development kit software. This includes the Linux kernel, the boot loader, and the BCON user space tools. The FPGA configuration remains unchanged unless you rebuild the FPGA project (see Section 6.3).

To rebuild the development kit software:

1. Download the BSP from www.baslerweb.com/en/sales-support/downloads/software-downloads/basler-dart-bcon-for-lvds-development-kit-board-support-package.
2. Source the PetaLinux build environment:
   ```
   $ . /opt/petalinux-v2016.3-final/settings.sh
   ```
3. Create a new PetaLinux project:
   ```
   $ petalinux-create -t project -s <bsp file>
   ```
   where <bsp file> is the path of the BSP file.
   A new directory is created, named after the BSP file with the version number removed, e.g. **basler_dart_bcon_for_lvds_devkit_2016_3**.
4. Change your current directory to the newly created directory:
   ```
   $ cd basler_dart_bcon_for_lvds_devkit_2016_3
   ```
5. Rebuild the Linux kernel and the boot loader by executing the following command:
   ```
   $ make all
   ```
   Two new directories named **build** and **images** are created. The **images** directory contains

the newly produced software artifacts. The **build** directory is only used during the build process.

6. Rebuild the root file system archive:
   ```
   $ make build-rootfs
   $ make rootfsimage
   ```

6.3 Rebuilding the FPGA Project

To rebuild the FPGA project, you must install the Xilinx Vivado development suite, available from the Xilinx website.

After setting up an account, you can obtain the required version 2016.3 here: www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/archive.html

Follow the instructions given on the download page. The software should be installed in the **/opt/Xilinx** directory.

To modify the FPGA project:

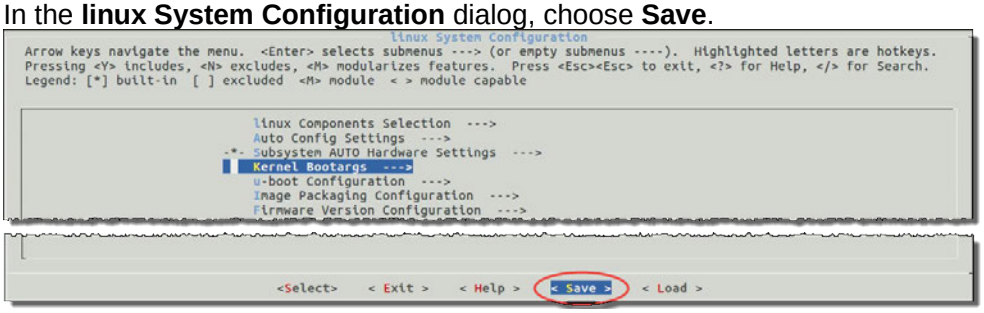
1. If you are modifying the FPGA project for the first time:
   a. Source the PetaLinux build environment:
      ```
      $ . /opt/petalinux-v2016.3-final/settings.sh
      ```
   b. Create a new PetaLinux project:
      ```
      $ petalinux-create -t project -s <bsp file> -n <project name>
      ```
      where <bsp file> is the path of the BSP file and <project name> is the name of your custom project.
2. If you already sourced PetaLinux, launch a separate shell and perform the following steps in that shell to avoid conflicts.
3. Source the Vivado environment.
   ```
   $ . /opt/Xilinx/Vivado/2016.3/settings64.sh
   ```
4. Start the Vivado GUI:
   ```
   $ vivado
   ```
5. In the Vivado GUI, load the Vivado project found in **<project name>/hardware/MZ7010_BCON_CARRIER/bcon_for_lvds_petalinux.xpr** where <project name> is the name of your PetaLinux project.

To rebuild the FPGA project:

1. If you already sourced Vivado, launch a separate shell and perform the following steps in that shell to avoid conflicts.
2. If you have not already done so, source the PetaLinux build environment:
   ```
   $ . /opt/petalinux-v2016.3-final/settings.sh
   ```
3. Change your current directory to the PetaLinux project directory:
   ```
   $ cd <project name>
   ```
   where <project name> is the name of your PetaLinux project.
4. Rebuild the FPGA project:
   ```
   $ make build-fpga
   ```
5. Invoke the “use-fpga” make target:
   ```
   $ make use-fpga
   ```

(If you don't invoke "use-fpga", the build process of the development kit software will continue to use the pre-existing FPGA configuration.)
After a while, the **linux System Configuration** dialog opens.

6. In the **linux System Configuration** dialog, choose **Save**.



7. When prompted for a configuration file name, choose **Ok** and **Exit**.
8. Back in the **linux System Configuration** dialog, choose **Exit**.
9. Rebuild the development kit software:

```
$ make all
```

6.4 Build Artifacts

After building the software, you can find the following build results in the **images/linux** directory:

- **boot.bin:** Boot loader binary
- **image.ub:** Linux kernel image
- **rootfs.cpio.gz:** Root file system archive

The other files present in this directory are by-products of the build process. They are not required to create a bootable microSD card.

6.5 Creating a microSD Card Image File

To create a ready-to-use microSD card image:

1. Invoke the "diskimage" make target.

   ```
   $ sudo make diskimage
   ```

   The resulting image is named **disk.img** and placed in the current directory.

   The image is created from the contents of the **images/linux** directory (see Section 6.4).

Revision History

Document Number	Date	Changes
AW001435**01**000	19 May 2017	Initial release version of this document.
AW001435**02**000	02 June 2017	Revised the procedures in Section 6.3.
AW001435**03**000	02 August 2018	Changed name of the development kit to *Basler dart BCON for LVDS Development Kit*. Updated download links accordingly.