BARIX IPAM-400 OEM
SW SDK Documentation

Boot and Update Strategy ARM Platform v6.1

Document Revision Table

Date	Version	Who	Change
04th April 2017	v.1	JR	First draft
12th April 2017	v.2	AB	First version
7th June 2017	v.3	AB	Proposal for change
15th June 2017	v.4	AB	Updated document title Added Firmware Upgrade image format Added Firmware Upgrade image validation process Changed Micro-SD layout
27th July 2017	v.5	AB	Added u-boot-env update target Implementation changes
25th September 2017	V.6	AB	Added install scripts
21st April 2018	V.6.1	ASI	Converted Confluence document in MS Word format

TABLE OF CONTENTS

1 Introduction

This Document describes the boot strategy, the rescue image and the firmware upgrade procedure of the Barix ARM platform.

2 System Overview

The Barix ARM platform (IPAM400) uses an Allwinner H3 CPU chip.

Min. 256MB DDR3 RAM provide ample space for code and data.

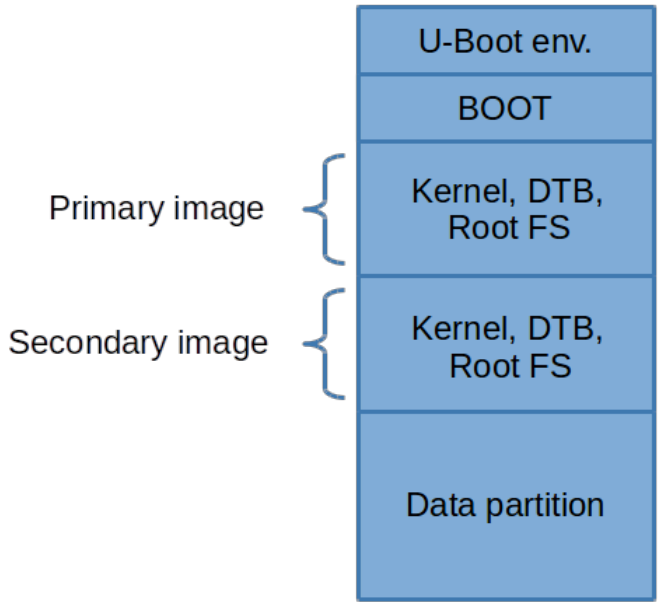
This document assumes that the Linux image including system and data partitions are stored on the Micro-SD, and a small system maintenance image (named Rescue image) is kept in a SPI flash to help recover devices in the field.

3 Micro-SD

The Micro-SD will be formatted so that it contains partitions for two Linux images plus a large user data space.

One image will be defined the primary image, the other is the secondary image and is used in case a newly downloaded image does not work. The function of the two images is swapped after a successful firmware upgrade is executed.

The Micro-SD layout is shown in the following picture:



The Linux Kernel and the kernel device tree (DTB) are embedded in the Root FS. This reduces the total number of needed partitions and simplifies the handling of the updates.

4 Boot loader

The initial boot stage function is performed by the U-Boot. This is composed by a U-Boot SPL that can be directly loaded by the processor FEL and a complete U-Boot that is executed afterwards.

U-Boot including SPL is loaded in the factory into the SPI flash.

U-Boot will be configured/modified so that it implements the defined boot strategy.

U-Boot makes uses of an environment area stored on the Micro-SD (called U-Boot environment) to store a set of nonvolatile variables that can also be accessed by the Linux images.

5 Boot strategy

U-Boot is only installed in SPI flash and NOT on the Micro-SD. This is crucial for the following procedure to work as otherwise, a Micro-SD based U-Boot will start.

The H3 will scan the Micro-SD, (non existing) NAND flash first for a valid U-Boot SPL and not find these. As a next step, it will find the U-Boot SPL in the SPI flash and start it.

U-Boot will then take over.

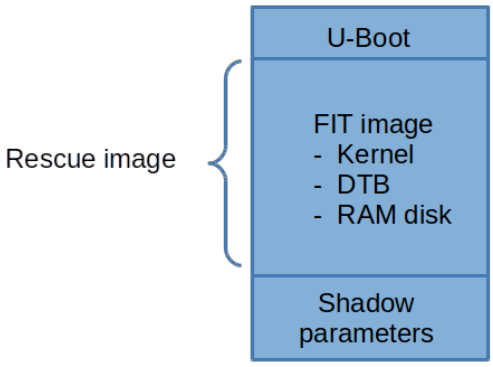
U-Boot will be a customized version normally doing the following:

- if the "reset/factory defaults" button is pressed at boot-up, directly start the rescue image. Otherwise:
- check on Micro-SD if kernel and kernel device tree of the primary image can be loaded. If so, load them and boot using the primary root file-system. Otherwise:
- check on Micro-SD if kernel and kernel device tree of the secondary image can be loaded. If so, load them and boot using the secondary root file-system. Otherwise:
- start the Rescue image in SPI flash.

A special handling is implemented by the U-Boot in case a new image has been written by the firmware upgrade procedure to the secondary partition (upgrade_available flag set to 1 in U-Boot environment) . In this case if booting or activation of the new image fails for a defined number of times (the upgrade_available flag remains set to 1 for a number of restarts exceeding a boot count limit) the U-Boot boots the original image ensuring that the device does not get bricked. The partition booting properly is configured as the primary partition.

6 SPI flash

The following picture shows the SPI flash partitioning.



The Rescue image is a FIT image containing the Linux Kernel, the Kernel DTB, and the RAM disk with the rescue root fs.

The shadow parameters partition contains a set of parameters that can be used by the Rescue image and the Firmware Upgrade Client. See Network parameters, certificates etc for additional info.

Such partition can use a FAT32 fs containing a set of text files with the current settings.

7 Rescue Image

A rescue image will be installed into the SPI flash together with a set of shadow parameters used by the rescue procedure including the network parameters (see Network parameters, certificates etc for additional info).

Such rescue image performs the following operations:

- if, the "reset/factory defaults" button is pressed at image activation time and remains pressed for a certain time, reset the shadow network parameters to default
- establish a connection with an update server from a list of possible servers stored in the shadow parameters area (HTTP and HTTPS protocols shall be supported)
- submit to the update server a set of information about the device including: MAC address, Hardware ID, ….
- download from the update server a full firmware image for the device and validate it
- format the Micro-SD
- install the image on the Micro-SD
- reboot the device

8 Firmware Upgrade Client

A firmware upgrade client/procedure needs to by integrated into the Linux images. Such client is responsible of installing Linux image updates. It can also download and activate the new images.

The firmware upgrade client can perform the following operations:

- establish a connection with an update server from a list of possible servers stored in the shadow parameters area (HTTP and HTTPS protocols shall be supported)
- submit to the update server a set of information about the device including: MAC address, Hardware ID, current image version, ….
- if an update is available, download from the update server a firmware upgrade image for the device and validate it
- install the new downloaded image in the secondary Linux image partitions
- inform U-Boot that a new image has been flashed (set upgrade_available flag to 1)
- reboot the device

Additional notes

- firmware upgrade client can validate the proper startup of a new image by marking it as active (set upgrade_available flag to 0, if found to 1) or this can be delegated to an external application
- firmware upgrade client can operate on a image that has been downloaded separately

9 Firmware Upgrade image format

A Firmware Upgrade image is a single file containing all the information and the data needed to upgrade a running Linux system or to fully restore/setup a system.

All the artifacts needed for the upgrade are packed into a tar archive together with a couple of additional files named header and header.sig containing respectively the meta information and the digital signature of the header file.

Here is the content of a sample Firmware Upgrade image file:

```
software-update.qiba (tar format)
   |
   +---header
   |
   +---header.sig
   |
   +---rootfs.tar.gz
   |
   +---data.tar.gz
   |
   +---....
```

The header file is encoded in JSON and it describes all the components present in the image.

The header.sig file contains the signature of the digest computed on the header file.

Here is a header file sample showing the general format used:

```
{
   "version": "1",
   "machines":  [ "barix-ipam400", "orange-pi-zero" ],
   "description": "update-core-image-qiba-barix-ipam400-20170727111923",
   "images": [
     {
       "target": "rootfs",
       "version": "core-image-qiba-barix-ipam400-20170727111923",
       "filename": "core-image-qiba-barix-ipam400.tar.gz",
       "sha256":
"ac802fd79e821ad0b562379e5e101ad67080986c3749a3fe8bb74230cb90ef3f"
     },
     {
       "target": "datafs",
       "filename": "data.tar.gz",
       "sha256":
"faaaa3096b01c196d20903e21ec88757834e68f09de4c2edd721ad8b83a9628e"
     }
   ]
 }
```

The JSON fields contained in the header file have the following meaning:

version: Version of the current qiba image file format and header file.

Machines: List of the machines that are compatible with this update.

Description: Short description of the update content.

Images: List of the images contained in this update.

For each image we have the following fields:

Target: Specifies the target partition to be updated. The target also implies the format of the associated update file. An update is meant to fully replace the content of a partition. Incremental or partial upgrades are not supported.

The following targets are currently supported:

target	file format	destination
rootfs	gzip compressed tar archive containing Linux kernel, kernel DTB and full rootfs.	SD card root partition
datafs	gzip compressed tar archive containing default storage data.	SD card storage partition
scripts	gzip compressed tar archive containing pre-install and post-install scripts.	Install scripts

When updating the SD card rootfs partition the dual root file-system strategy described in this document is used.

The following additional targets are also supported. They are meant to be used during production programming.

Target	destination	file format
u-boot	U-Boot image file	SPI flash U-Boot partition
u-boot-env	U-Boot environment image file	SD card U-Boot environment area
rescue-image	FIT image containing kernel, kernel DTB and ram-disk of the rescue image	SPI flash rescue image partition
shadow-parameters	OEM specific shadow parameters.	SPI flash shadow parameters partition See Network parameters, certificates etc for additional info.

filename

Name of the associated update file.

version

Version of the update file. This applies to the rootfs target only.

sha256

SHA256 message digest of the associated file.

10 Firmware Upgrade image validation

After a successful download and decompression of a Firmware Upgrade image the following process is used on the device to validate the upgrade:

1. the signature contained in the header.sig file is verified by using the public key stored in the shadow parameters area
2. the SHA256 checksums provided in header file are compared with the corresponding checksums computed locally on each individual image file

If any of the above checks fails the upgrade gets rejected.

Additional notes

The header.sig file in the Firmware Upgrade image contains the signature of the digest computed on the header file using the SHA256 algorithm.

On the host where the Upgrade image is created this can be computed with OpenSSL using the private key:

```
openssl dgst -sha256 -sign privatekey.pem header > header.sig
```

and it can be verified on the device using the associated public key:

```
openssl dgst -sha256 -verify publickey.pem -signature header.sig header
```

11 Network parameters, certificates etc

The actual network parameters as configured in the last running valid image are shadowed into a special partition of the SPI flash, which is not normally accessible to the user.

This area can also be used to store application specific keys, the URL for the update server etc.

The recovery image and the firmware upgrade client will use these parameters to contact an update server and download a valid image.

In OEM products, an "update URL" will be set in the shadow area, so the rescue image will know where to access the device image. For standard Barix devices, Barix will run a high availability server (for example in the domain barix-update.com) from which each device can download its actual firmware (indicated by an application key and/or MAC address)

An OEM device accessing the Barix server will be reverted to an OEM update server (OEM device MAC address ranges need to be registered on the Barix update server for that purpose).

12 User Control of Boot Cycle, factory loading

The devices using the ARM platform will provide at least one user accessible "reset/factory defaults" button.

While the functionality of the button, once the standard image is started, is out of the scope of this document, the functionalities of the button are described here.

With the button pressed while entering U-Boot, the system will start the rescue image immediately to allow a recovery from a fatal micro-SD formatting problem and the like.

By pressing the button for a longer time (timing indicated to user by the LEDs), the user can also reset the network parameters used for the recovery image, which are normally "shadowed" from/by the standard image, to DCHP default. This allows to rescue the device even if network settings have been completely screwed.

Furthermore, a non-user accessible jumper on the PCB (Case opening needed to access it) will allow the "factory fresh" loading of the device from a USB stick (CPU ROM functionality).

13 Disclaimer

The procedure described in this document is not implemented yet and subject to improvements and changes.

14 Legal Information

All information is subject to change without notice.

All mentioned trademarks belong to their respective owners and are used for reference only.

Barix, Exstreamer, Instreamer, SonicIP and IPzator are trademarks of Barix AG, Switzerland and are registered in certain countries. For information about our devices and the latest version of this manual please visit ***www.barix.com***.

Barix AG
Ringstrasse 15a
8600 Dübendorf

SWITZERLAND

Phone: +41 43 433 22 11
Fax: +41 44 274 28 49

Internet

web: ***www.barix.com***

email: ***sales@barix.com***

support: ***support@barix.com***