Linux BSP Integration

ABSTRACT:
The document describes the integration of Linux BSP

KEYWORDS:
Vision SDK, Linux BSP, Integration

APPROVED:		
AUTHOR	**SIGN-OFF SIGNATURE #1**	**SIGN-OFF SIGNATURE #2**
Cuong Nguyen Huy		

Revision History

VERSION	DATE	AUTHOR	CHANGE DESCRIPTION
1.0	14-Dec-18	Cuong Nguyen Huy	Initial version

Table of Contents

1 Introduction

In this document, the Linux BSP integration is described.

The first part of this document covers u-boot, Linux changes, patches, description.

The second part then describes how to build the Linux BSP.

1.1 Purpose

The purpose of this document is to present the Linux BSP and help users to bring up examples quickly.

1.2 Audience Description

This document is intended to S32V234 Vision SDK users.

1.3 References

Id	*Title*	*Location*
[1]	*Auto Linux BSP Quick Start*	*https://bitbucket.sw.nxp.com/projects/ALB/repos/alb-quality/browse/BSP-release-specific* *(delivered with each BSP release)*
[2]	*Auto Linux BSP Release Notes*	*https://bitbucket.sw.nxp.com/projects/ALB/repos/alb-quality/browse/BSP-release-specific* *(delivered with each BSP release)*
[3]	*Auto Linux BSP User Manual*	*https://bitbucket.sw.nxp.com/projects/ALB/repos/alb-quality/browse/BSP-release-specific* *(delivered with each BSP release)*
[4]	*Auto Linux BSP Software Architectural Design Specification*	*https://bitbucket.sw.nxp.com/projects/ALB/repos/alb-quality/browse/BSP-release-specific* *(delivered with each BSP release)*

Table 1 - References Table

1.4 Definitions, Acronyms, and Abbreviations

Term/Acronym	*Description*
ACF	*APEX Core Framework*
APEX2	*A parallel image processing accelerator HW block, designed by CGV, part of S32V234 SoC.*
APEX COMPILER	*A set of tools (Synopsys IP Programmer for APEX2 or NXP APU compiler) that allow compilation of code for APEX2 subsystem.*
ARM	*Family of RISC architectures*
BSP	*Board Support Package*
CGV	*Abbreviation for Cognivue company.*
DCU	*Display Controller Unit*
FDMA	*Fast DMA HW block.*
GHS	*Greenhills*
ISP	*Image Signal Processor subsystem of the S32V234 SoC.*
MIPI CSI	*MIPI Alliance standard for Camera Serial Interface.*
OpenCL	*Open Computing Language*
OpenCV	*Open Source Computer Vision*
RTOS	*Real Time Operating System*
SDK	*System Development Kit*
Sequencer	*M0+ core and its firmware controlling the ISP subsystem.*
SRAM	*Static RAM.4MB memory area used for fast data buffering during ISP processing.*

Table 2 - Acronyms Table

1.5 Document Location

s32v234_sdk/docs/vsdk/VisionSDK_BSP_Integration.pdf

2 General Description

This section contains a general description of the Auto Linux BSP components and how they relate to each other.

The Auto Linux BSP respects the general layout of a BSP, containing a bootloader (U-Boot) the Linux kernel, a RFS, which can contain various libraries and middleware, and sample applications. The purpose of ALB is to enable other projects, including client projects, to build on top of the ALB and add supplemental components such as drivers or applications.

For all open source components included in the BSP, if community versions of the components exist, they are reused in ALB, after applying the necessary modifications to enable execution on NXP hardware. The modifications include: ALB specific configuration of components, driver modifications, device tree definitions for NXP AMP specific hardware, new driver implementations, implementation of new features, writing new sample applications etc.

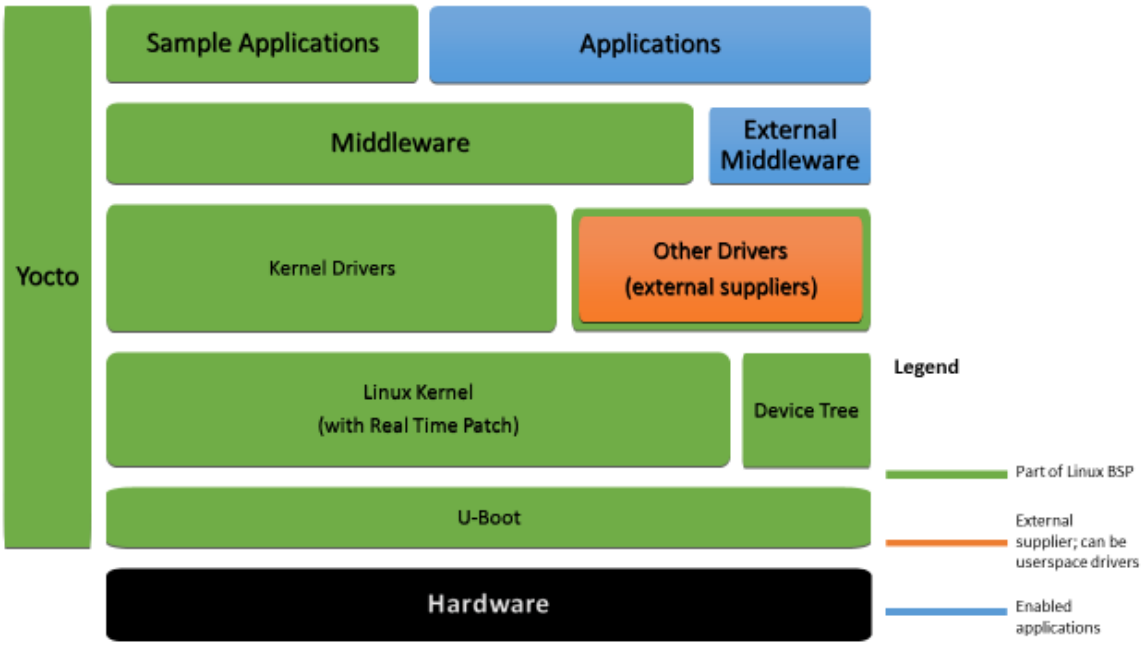

Figure 1 - Linux BSP Overview

2.1 U-Boot

U-Boot is a boot loader responsible for the initialization of specific hardware components to a basic level, enough to allow the loading of the Linux kernel image, the device tree blob and, if used, an initial RAM file system (initramfs or initrd) from a boot media, such as an SD card, eMMC, QSPI flash, network etc.

Once the necessary images are loaded in RAM, U-Boot will hand over control to the Linux kernel, and after the Linux kernel is started, U-Boot has finalized its job and the memory it has used will be reused by the Linux kernel, according to the memory management policies Linux has in place.

2.2 Linux Kernel

The Linux kernel is responsible for memory and resource management, scheduling, driver loading and unloading, interrupts handling and providing APIs (syscalls, ioctls, pseudo or virtual filesystems) for communication between the kernel or kernel drivers and user space applications.

The Linux kernel from the ALB is a version mostly comprised of a specific community version, the base version, and ALB specific additions to allow the kernel to enable NXP hardware to work.

The Linux kernel source also provides a reference implementation of the drivers, showcasing the way the driver for a specific peripheral should work, in case an implementation on another operating system is desired.

2.2.1.1Real Time Patch

The Real Time Patch is a set of modifications on top of the Linux Kernel which aims to provide soft real-time capabilities to the Linux kernel. The net effect should be a more predictable kernel with lower latency. This does not mean ”a faster kernel”, but only a more predictable kernel.

2.3 Device Tree

The Device Tree provides a representation of the hardware, allowing the creation of a generic Linux image which can be reused on different hardware without modification. This is possible because the Linux kernel can initialize only the drivers that are present in the hardware configuration, without hard-coding hardware related information in the drivers or the kernel. Also, the availability of the hardware description allows the operating system to reuse a specific driver for all instances of a peripheral, each instance having different configuration information based on the information from the device tree.

The Device Tree is mandatory for all ARM and ARM64 kernels, and can be reused in conjunction with other operating systems.

2.4 Kernel Drivers

Kernel Drivers provide a hardware abstraction layer by implementing Linux kernel APIs and provides mechanisms for user space applications to interact with the hardware (usually indirectly, through libraries). The interaction paths are driver and hardware dependent, so they can differ from driver to driver. Exact specifics on the communication, capabilities and limitations shall be provided in other documents such as the Release Notes document, ALB User Manual or documentation included in the source code.

These drivers are part of the same source as the Linux kernel, but, in binary form (.ko modules) can be provided separately from the kernel image, although they are built from the same source tree.

2.5 Other Drivers

It is possible that for some of the peripherals/modules, the drivers are not part of the Linux kernel source, and have a separate source tree and maintainer or maintainer team. They are considered part of the BSP, but some limitations might apply (e.g. the driver is split in a kernel and a user space part, but the userpace part is provided in binary form only).

2.6 Middleware

Middleware is comprised of various libraries, scripts, and other code that allows the building and/or execution of custom applications on top of the Linux BSP. Typical examples of middleware include libraries, stacks and toolkits such as QT, X11, OpenCL, OpenVG, OpenGL etc.

The exact content of this part of the BSP can supplemented, as per the needs of the customers, users and developers or the sample applications which are provided as part of the ALB.

2.7 Sample Applications

These are applications which serve the purpose of showing spcific capabilities, usage modes of hardware or software components, and to give an idea to the BSP users on how to implement a specific feature into a larger scope application.

2.8 Yocto

The Yocto component is responsible for providing a way of building all BSP components and tools necessary to create the binary images to be ran on the target hardware.

It provides developers and clients the flexibility to supplement the provided images with other tools, binaries and utilities that should be part of the final RFS on the target.

2.9 Hardware, Enabled Applications and Enabled Middleware

The BSP relies on specific hardware and enables various applications and middleware that can be provided by third parties such as clients or Vision SDK project. The hardware and these enabled components are not part of the BSP.

2.10 Release Content Description

SoC	Component and Version	Boards supported & tested
S32V234 maskset: 0N71N, 1N71N, 1N81U	Linux v4.14.34 U-boot v2016.01 Yocto 2.4	S32V234 EVB (SCH-28899 Rev D, Rev F1) S32V234 EVB (SCH-29288 REV B1) S32V234 PCIe (SCH-28677 Rev D2) S32V234 SBC (R3V1A1) Bluebox 2.0 (rev X2)
S32V34x		NXP_S32V344_ECU_M-2017.06-R6.0.0_P0_FixedVDK
S32G275		NXP_S32G275_ECU_N-2017.12-R6.0.0_P0_FixedVDK
S32R45x		NXP_S32R45_ECU_NoBBE32_M-2017.06-R6.0.0_P0_FixedVDK
LS2084A	Linux v4.9.35 U-boot v2017.07 Yocto 2.4	Bluebox 2.0 (rev X2)

Table 3 - Release Content for BSP 18.0

3 Linux BSP Features

This chapter describes the features of the Linux BSP:

- S32V234 SoC support addition to U-boot 2014.04 and Linux 3.18.9 Kernel
- S32V234 SoC support addition to U-boot 2016.01 and Linux 4.1.26 Kernel

3.1 U-Boot Bootloader

- Added support for gate/ungate in clocking framework
- Added support for S32V234EVB board REV.A
- Update DDR3 timings for S32V234EVB REV.A
- Enabled switch connection for the EVB and PCIe boards
- Defined ft_board_setup() for all s32v234 platforms
- Added CONFIG_CSE3 option in menuconfig
- Configure SRAM as non-cacheable from u-boot
- Support for init SRAM and start M4 in uboot
- Update the S32V234 memory map according to the new usage since the u-boot has been moved to DDR0.
- Enable by default the Image instead of uImage
- Introduce the possibility to configure the DDR type, BCM switch and Serial Baudrate for S32V234 boards
- Move the clock framework to the SOC level
- Support for EMMC boot
- Support for booti
- Support for all available DDR memory
- Core timer support as timer
- Linflex driver (only UART support)
- Clock tree support
- Pinmuxing support
- SDHC support
- PCIe RC support
- Booting from SDHC support
- ENET driver activation and testing on s32v234evb in RGMII mode

- TFTP booting
- Added QSPI boot support (only for U-boot)
- Add initial support for PCIe board
- Linflex UART works in FIFO mode
- Add Fixed PHY to allow the usage of BCM
- SMP support
- Activate BCM ENET support for S32V234 TMDP board
- Activate ENET support for S32V234 PCIe board using BCM 89610 PHY
- Add XRDC config. Allows ISP HW access the SRAM
- Introduce the QOS settings for DCU on S32V234 TMDP
- Enable PCie_EP
- Secure Boot support
- Functional Reset support
- Added XRDC config for pcie, tmdp & fvb boards
- Init RNG on secure boot flow
- Enable DDR Handshake in order to avoid DDR restoring issue
- Correct DDR size in configuration files
- Merge all new u-boot versions until v2016.01
- Tested on v2015.10 and v2016.01
- QSPI/hyperflash read and write support
    - Software protection against accidental erases and writes into hyperflash
- Document the memory map
- Introduce irq support for PCIe link reset recovery
- Simplified guarding mechanism for NXP ARM64 bit cores by defining CORE_64BIT_PERIPHERALS_32BIT6 macro.
- Adapted ESDHC driver for S32V344 boards.
- Added PCIE driver support for S32V344 boards.
- Added initial support for S32V344.
- Preliminary support for Mini Bluebox in U-boot.
- Enabled DCU/FB driver in U-boot.
- Enabled the clock for DEC200 by default.
- Increased image size to support OS images with size bigger than 8MB.

- Validated the kernel booting with Kernel Image, dtb and ramdisk stored in Hyperflash.
- Support CUT2.0 sample.
- Support EVB SCH 29288 rev B1.
- Initial support for SPI NOR flash was added. The support was tested using Spansion flash S70FL01GS.
- Validated PCIe feature on BBMini for cut2.0 using the external oscilator.
- Initial support for s32v344
- Initial support for s32g275
- KRAM shall be unlocked on CSE enabled samples
- U-boot shall have support for DSPI
- Display CPU revision on startup
- CONFIG_FSL_CSE3 is activated by default
- U-boot shall have support for DSPI
- SJA1105 ethernet switches shall be programmed over SPI
- U-boot shall enable all i2c buses
- U-boot refactoring for common chassis
- U-boot shall start all 4 cores on s32gen1 platforms
- UART console shall be supported in u-boot on s32gen1 platforms
- U-Boot shall add XRDC support for S32V344
- U-Boot should have a config file designed for booting from NOR Flash
- MAC address should be uniquely assigned per unit
- Modify default env vars for s32v234 to support nfs booting
- Enable the Ethernet for BBMini
- U-boot shall support S32gen1 clocking
- Rename S32V244 to S32V344
- Support for VDK R5.0.0
- Linux BSP should use GCC6.x compiler
- MAC address should be set based on ethaddr u-boot environment, and warn if using default MAC
- Add support for the BlueBox 2.0 platform on S32V234 and LS2084A
- U-boot shall provide a driver for the PCF85263 RTC
- U-boot shall contain a default config for Microsys SBC-S32V234 board
- [s32gen1] u-boot BSP shall support software reset

- [s32gen1] VDK bootup/speed up performance improvements
- [s32gen1] Activate DSPI_CLK
- [s32gen1][Blind support] QSPI support
- Add platform dependent to code to allow SRAM initialization on s32-gen1
- Remove duplicated code for initsram command to a shared file for S32V234 and S32-gen1
- Speed up of SRAM initialisation in S32V234 low level init
- SRAM initalization uses harded-coded values unsuitable for S32-Gen1
- [s32gen1] The SRAM controller shall be supported in U-boot on S32gen1 platfroms
- [s32gen1] U-Boot shall provide basic pinmuxing support for S32gen1 platforms
- [s32gen1] U-boot shall support FlexNOC on S32gen1 platforms
- [s32gen1] U-Boot shall support SDHC
- [s32gen1] MSCM updates according to latest specs
- [s32gen1] Need to provide reset cause
- [s32gen1] Aurora PLL has been removed
- Add deregister capability to GIC module
- [s32gen1] Investigate secondary core bootup
- Cleanup s32 platform makefile
- Add disabled CONFIG from defconfigs to Kconfig for sim targets
- [s32gen1] Merge and synchronize next-gen specific ’configs/s32*_defconfig’ across the 2 u-boot versions
- Add initial support for u-boot v2018.07
- U-Boot shall provide basic pinmuxing support for S32gen1 platforms
- Export fprintf and gic_register_handle to be used from standalone applications
- [s32gen1] CORE_CLK and DDR_CLK support

3.2 Linux Kernel

- FB driver support for GPU Tiled format data
- Enabled PCI support in S32V234 defconfig
- Added support for linflex IPG clock
- Introduced support for gating on S32V
- Enabled by default GPIO module for S32V234 platforms
- Added and enabled s32v234pciebcm dts file
- Added UART1 entry and pinctrl definitions in S32V234 dtb
- Introduce Support for Real Time Patch: Set default preemptible to Low-Latency
- Add GPIO support for S32V234 platform
- UART support for FIFO with DMA
- Framebuffer and DCU driver:
    - VSYNC support
    - 1080p double-buffering
    - Support for Multiple color and grayscale
    - Support for CLUTs
    - Support for FB setup according to boot parameters
    - Fixed initialization of videomode names
- Remove VIUlite driver
- Enable CAN for PCIE and TMDP boards
- PIT driver improvements:
    - PIT can now be used on any core.
    - PIT can now be used in a system with more than two cores.
- STM driver support
- Enable BSD_PROCESS_ACCT
- Enable FANOTIFY for notification & interception of FS events
- Enable IP multicasting
- eDMA support
- Kernel enable loopback device, core dump, tun/tap module, taskstat
- Support for PTP
- Update the PIT driver in order to offer the possibility to be used as system timer

- Support for EMMC read/write
- Support for Enhanced Mode in ENET driver
- Support for all available DDR memory
- Interrupt support for VIUlite
- Device Tree support for H.246 Encoder and JPEG Decoder
- ARM-Cortex A53 GIC support
- Core timer support as kernel OS tick timer
- Basic SoC device tree support
- FSL Linflex support (only UART support)
- PCIe RC support
- Clock framework integrated
- I2C driver activated
- SIUL2 driver for pinmuxing
- Watchdog driver
- Update the ENET node from DTS file
- Add support for Vivante GPU driver
- Add initial version of security CSE3 driver
- ENET driver
    - Root over NFS
- Add initial support for PCIe board
- Added fec node in dts for BCM configuration
- Enable Fixed PHY to allow the usage of BCM
- Added mipi-csi2 device tree nodes
- Added viulite0 and viulite1 device tree nodes
- Added PCIe EP and RC mode
- Enable System IPC V and Message POSIX Queue features
- Activate ENET support for S32V234 TMDP board
- DTS cpu nodes for SMP (enable-method and cpu-release-addr fields)
- PCIe MSI support
- Introduce the FlexCan support for S32V234 EVB board
- Activate ENET support for S32V234 PCIe board using BCM 89610 PHY
- Configure kernel to boot with systemd init system

- SMP support
- PMU support
- Activate VFAT fs support
- Introduce preliminary support for VIULite driver
- Add PCIe Internal DMA Single block mode
- Add PCIe internal DMA linked list mode
- Implement the reset driver for S32V
- Port all S32V234 patches from 3.18 to 4.1 kernel
- Enable CMA
- Enable BINFMT_SCRIPT option
- Add Crypto API support and improved the CSE driver
- Port PCIe driver from 3.18 to 4.1 kernel
- Enable VIULITE driver for VIULITE0
- Add FDMA and VSEQ DTS nodes
- Enable CONFIG_PM option for GPU driver
- Added Inter Core Communication support over PCIe (BlueBox)
- FB/DCU: Enable underrun interrupts and count these events. By default undrun counter is not active. Only writing text "on" into undrun_mode file will activate counter.
- FB/DCU: Change DCU interrupts to be active high level-sensitive instead of low-to-high edge triggered.
- Updated SPI3 node with a generic example for slave node.
- Preliminary support for Mini Bluebox: add the dts for MiniBluebox, port configuration for SJA1105 switches, support for NXP TJA1100/TJA1102 PHY, TJA1145 CAN receiver.
- Introduce the clock gating definitions for Vision IPs.
- FB/DCU: Added Linux clock gating support to DCU driver.
- Inter Core Communication improvements.
- Validated Inter Core communication on the Mini Bluebox, EVB board.
- Support CUT2.0 sample.
- Support EVB SCH 29288 rev B1.
- Add initial support for ADC.
- Add initial support for MTD (Memory Technology Device).
- Enable USB 2.0 & 3.0.
- Allow the users to find at runtime which configuration options are enabled.

- Initial support for s32v344
- Initial support for s32g275
- Add support for LXC containers
- Enable options to support INITRD/INITRAMFS by default in the kernel configuration
- s32v234-pinfunc.h: Remove the obsolete pinctrl file from dts directory
- Linflex UART support on next generation platforms
- PIT shall be enabled on next generation platforms
- STM shall be enabled on next generation platforms
- Basic support for edma3 shall be added
- Support for 32b frames in spi driver
- Linux shall support S32gen1 clocking
- Support for VDK R5.0.0
- Rename S32V244 to S32V344
- Linux BSP should use GCC6.x compiler
- Update edma memory map according with VDK5.0.0
- SPI shall be supported on S32gen1 platforms
- Add support for the BlueBox 2.0 platform on S32V234 and LS2084A
- Linux BSP shall provide a device tree for the Microsys SBC-S32V234 board
- SPI shall be supported on S32gen1 platforms
- Linux shall have minimal kgdb support on supported platforms
- Merge with the upstream 4.14-rt version
- [s32gen1] Linux BSP shall support software reset
- I2C Shall run at 400Kbps
- UART driver shall set the DMA timeout according to baudrate
- Protect console_write with lock
- [s32gen1] VDK bootup/speed up performance improvements
- [s32gen1][Blind support] Linux BSP shall support SIUL2 – GPIO
- [s32gen1] The Linux BSP kernel shall contain an I2C driver (blind support)
- [s32gen1] Linflex UART driver shall work with eDMA3
- [sja1105x] The nxp.ko module needs be inserted only if it not present
- [sja1105x] Integrate patch that resolves a cross-talk problem with the port connecting to the S32R274

- [s32gen1] Linux BSP shall provide clock gating support via MC_ME
- [s32gen1] Linux BSP shall support FlexCAN
- [s32gen1] Linux BSP shall support uSDHC
- [s32gen1] Linux BSP shall support SWT
- [s32gen1] Linux BSP shall support Ethernet
- [s32gen1] Linux BSP MC_ME shall have clock support
- [s32gen1] Reset functionality must also set the functional register
- [s32gen1] Clock updates according to latest specs
- [s32gen1] Include VIRTIO in simulator
- [s32gen1] Correct I2C clock
- Enable CONFIG_SYNC_FILE in s32v234_defconfig
- Save ARM64 defconfig over s32v234_defconfig
- Some DFS_DVPORTn[MFN] values are outdated
- [s32gen1] Aurora PLL has been removed
- Correct Linflex clock
- Store Extended SPI Mode capability as a dts property
- Unify s32v234 and s32gen1 defconfig
- Fix trivial conflict in spi-fsl-dspi.txt
- [s32gen1][linux] CORE_CLK and DDR_CLK support
- Implement driver that allows complete access to hardware registers from userspace
- [s32gen1] FCCU shall be supported on common chassis
- [S32gen1] Linux BSP's kernel shall provide pinmuxing support for S32gen1 platforms
- Support CANFD for S32V234
- [s32v234] Linux BSP should support also 800MHz and 2 cores S32V2xx derivatives
- [s32gen1] Linux BSP shall have a functional I2C driver

3.3 Yocto

- Added sample application for networking
- Added sample application for GPIO driver
- Added sample application for PCIe
- Added support for Inter Core Communication basic configuration tool
- Added support for Inter Core Communication library
- Support for Secure boot based on FIT images
- Enabled acl quota at packages in full-image
- Added sample application to illustrate the benefits of multicore processing
- Support generate single sdcard image.
- Activate nfs-common package
- Support fb.modes setting file for all image
- Initial Linux BSP build support via YOCTO
    - fsl-image-s32v2xx image for EVB, PCIE and TMDP boards
- Reduce final archive size with cleanup scripts and public repos
- Updates for Linux 4.1
- Added Inter Core Communication support over PCIe (BlueBox)
- Added pciutils package to the Yocto image.
- Included GPU driver v5.0.11.p8.1 in the Yocto archive.
- Added support for S32V234 BBMINI board.
- Included the Framebuffer Viewer tool in the S32V234 full image.
- Include an MPI demo that runs a puzzle-solving application in parralel on multiple S32V234 boards
- Add recipe for the mpich MPI implementation
- Upgrade gdb version to 7.11.1
- Add lrzsz tool for zmodem/xmodem/ymodem file transfer
- Include the glmark2 benchmark tool
- Add glibc-utils which included locale tool
- Upgrade gcc linaro version to 4.9.3 2015.05
- Add support for EVB SCH 29288 rev B1. The s32v234evb machine will now generate images for EVB SCH 29288 board. For the old board, EVB SCH 28899, use the s32v234evb28899 machine

- Add initial support for SJA1105P switch enables ethernet for EVB SCH 29288 rev B1 board
- Upgrade GPU with Vivante 6.2.2
- Include virtual ethernet demo application
- Initial support for s32v344
- Initial support for s32g275
- Linux BSP shall have an image for S32R45x (Racerunner Multi)
- Remove http://git.freescale.com/source/ mirror URL from default configuration
- Document how to use CAF mirrors instead of git.freescale.com
- The U-Boot Yocto build should also build the flash version
- Correct release archive to facilitate the release
- Create meta-alb repository
- Make meta-alb use the sja1105 repo
- fsl-s32v-yocto-bsp manifest should clone meta-alb in msta-fsl-s32v directory
- Corrections necessary in meta-alb for bsp15.0
- Publish alb-demos, alb-fb-apps and icc-linux on CAF
  https://source.codeaurora.org/external/autobsps32/
- Create PREMIRRORS to map sw-stash links to bitbucket
- Upgrade the Yocto to Rocko base
- The linux BSP shall use meta-freescale layer
- Rename S32V244 to S32V344
- Yocto sources shall be available from public sources
- Update to GCC 6.3.x compiler
- Remove glmark2 patch/recipe from meta-alb and yocto archive
- Publish meta-alb to CAF
- Integrate galcore ALB repo into meta-alb yocto
- Remove SDCARD image for simulator targets
- Yocto build shall generate a flash bootable image
- Add support for the BlueBox 2.0 platform on S32V234 and LS2084A
- Yocto shall support at least one image for Microsys SBC-S32V234 machine/board
- Yocto add basic support for generating Ubuntu 16.04 image for s32v234evb board
- Port fsl-image-bluebox* extended images from meta-bluebox
- Update ls2084abbmini support from meta-bluebox

- Port all remaining machines from meta-bluebox
- Port the ls2 desktop image from meta-bluebox
- ls1043ardb: add pci-vdev into meta-alb
- Create a recipe for the 4.14 kernel
- Remove COMPATIBLE_MACHINE
- Prepend the BBPATH in order to allow class overrides
- [ubuntu] improvements and fixes
- Add permanent entry for “meta-adas” in the LAYER_LIST
- Build the LS side with the DN selected Toolchain
- Update auto_yocto_bsp README GPU support documentation
- Align binutils with the tools team version
- Align isl with the tools team version
- Align cloog with the tools team version
- Align ppl with the tools team version
- Align gettext with the tools team version
- Align NXP GCC with the tools team distribution
- Improve glmark2 bbappend to avoid overwrite SRC_URI
- Create new meta-adas archive QCOMM compliant
- [s32v234bbmini] Port Virtual Ethernet patch for kernel 4.9
- Included GPU driver v6.2.2 in the Yocto archive
- Enable ipc-shm for s32r s32v234 s32v3
- Ubuntu: add basic proxy support
- [Toolchain][LS2] Enable loop optimization support to LS2 toolchain
- [s32v] Replace the rootfs format cpio.gz from s32v rootfs images
- Remove RPROVIDES for ipc-shm kernel module
- [sja1105x] Rework init-nfs-boot script
- [pcie-demos] Add kernel support for pcie demos by distro feature
- [s32][u-boot-env] extract the u-boot environment used by yocto from the u-boot itself
- [yocto]: upgrade default DHCP client
- [bluebox] fix sdcard generation for bluebox images
- meta-adas: add support for using an external APU toolchain for APEX graphs
- TMU: enable TMU kernel settings by default

- Maintainer Info wrong
- meta-adas: add a check for minimum host gcc required by VSDK build system
- [ubuntu] add support for Ubuntu root file system for SBC board
- [yocto] Add a yocto recipe for the v2018.07 u-boot version
- ubuntu: create a separate sstate-cache folder for ubuntu machines
- host-prepare.sh: add requirements needed for "rocko" branch
- [yocto] Improve the default value of "THREADS" and "JOBS" in nxp-setup-alb.sh
- The s32gen1 images in recompiled binaries do not work with default dtb
- Update sample location in ipc-shm recipe
- [Microsys] Add support for SBC to pcie demos

4 Build the Linux BSP

All the steps described below have been run and validated on Ubuntu-16.04 LTS (native or through a virtual machine). It is then recommended to install Ubuntu-16.04 LTS before going through the following sections.

4.1 Setting up and building U-boot bootloader

4.1.1 Downloading the U-boot bootloader source code

There are two ways of obtaining the source for this component, each described below. Choose the one which is appropriate for your situation.

1. Cloning the GIT repository:

In a Linux terminal window, type in the following commands

```
git clone https://source.codeaurora.org/external/autobsps32/u-boot
cd u-boot
git checkout -b <branch_name> v2016.01_bsp19.0
ls
```

The contents of the u-boot source code should appear here.

2. If you already have a clone of the repository, you can run the following commands in the root directory of the existing repository:

```
git fetch origin v2016.01_bsp19.0
git checkout -b <branch_name> v2016.01_bsp19.0
ls
```

The contents of the u-boot source code should appear here.

4.1.2 Building the U-boot bootloader

In the same Linux terminal window as above, type in the following commands:

```
make ARCH=aarch64 CROSS_COMPILE=/path/to/your/toolchain/dir/bin/aarch64-linux-gnu- <board>_defconfig
make CROSS_COMPILE=/path/to/your/toolchain/dir/bin/aarch64-linux-gnu
```

This command should generate the u-boot image with IVT header and Program data (u-boot.s32) that can be written onto the SD.

Note:

If targeting the s32v234evb, s32v234tmdp, s32v234pcie, s32v344, s32g275, s32r45x, s32v234sbc board replace <board> with the board name in the first command above.

4.2 Setting up and building the Linux kernel

4.2.1Downloading the Linux kernel source code

There are two ways of obtaining the source for this component, each described below. Choose the one which is appropriate for your situation.

1. Cloning the GIT repository:

In a Linux terminal window, type in the following commands

```
git clone https://source.codeaurora.org/external/autobsps32/linux
cd linux
git checkout -b <branch_name> v4.14.34_bsp19.0
ls
```

The contents of the linux kernel source code should appear here.

2. If you already have a clone of the repository, you can run the following commands in the root directory of the existing repository:

```
git fetch origin v4.14.34_bsp19.0
git checkout -b <branch_name> v4.14.34_bsp19.0
ls
```

The contents of the Linux kernel source code should appear here.

4.2.2Building the Linux kernel

In the same linux terminal window as above, type in the following commands:

```
make ARCH=arm64 CROSS_COMPILE=/path/to/your/toolchain/dir/bin/aarch64-linux-gnu- <soc_name>_defconfig
make ARCH=arm64 CROSS_COMPILE=/path/to/your/toolchain/dir/bin/aarch64-linux-gnu
```

If targeting the s32v234evb, s32v234tmdp, s32v234pcie, s32v234sbc board replace <soc_name>_defconfig with s32v234_defconfig, and for s32v344, s32g275, s32r45x replace this variable with s32gen1_defconfig in the first command above.

This command should generate the kernel binary (Image) in arch/arm64/boot and the board device tree blobs (e.g. s32v234-evb.dtb, s32v234-pcie.dtb, s32v234-tmdp.dtb, s32v344-simulator.dtb, s32g275-simulator.dtb, s32r45x-simulator.dtb...) in arch/arm64/boot/dts/freescale/.

Note:

- If you want to use uImage instead of Image follow the command below:

```
<path>/u-boot/tools/mkimage -A arm64 -O linux -T kernel -C none -a 0x80080000 -e 0x80080000 -n "Linux" -d arch/arm64/boot/Image arch/arm64/boot/uImage
```

- If you want to add GPU drivers (see chapter ‘Setting Up the Graphics Drivers’) you should set CMA (Contiguous Memory Allocator) size to 32MB using

```
make ARCH=arm64 CROSS_COMPILE=/path/to/your/toolchain/dir/bin/aarch64-linux-gnu- \
s32v234_defconfig menuconfig
```

and from Device Drivers -> Generic Driver Options -> [*] DMA Contiguous Memory Allocator -> set (32) Size in Mega Bytes instead of 16 (default)

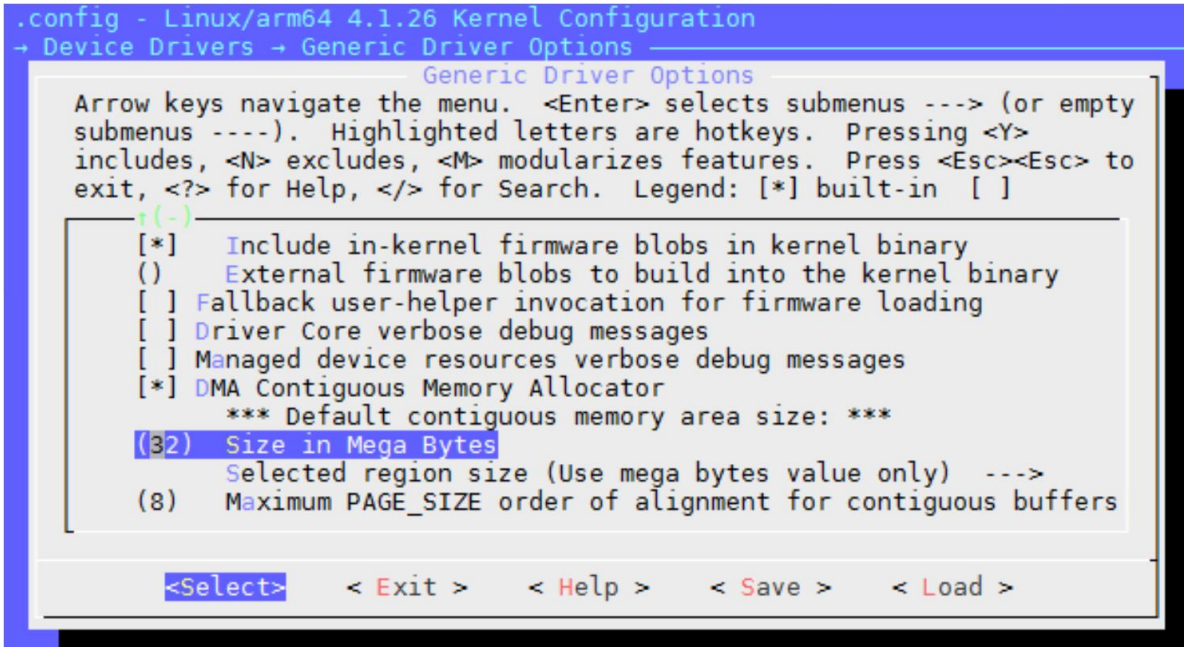

Figure 2 - Save to .config and Exit

- Console I/O is configured differently for different S32X boards: make sure “console=ttyLF0” is used for S32V234 EVB, respectively “console=ttyLF1” for S32V234 PCIe in the CONFIG_CMDLINE kernel config option.
- Linux Kernel considers that its RAM starts from Kernel’s ENTRY_POINT (0x80080000) ignoring the DDR below.
- uImage’s ENTRY_POINT and dtb’s load address need to be in the same 512MB space address.