FTF 2016
TECHNOLOGY FORUM

QorIQ LS1043A PORTING NEW U-BOOT ETHERNET DRIVER AND AUTOMATIC IMAGES UPDATE PROCESS

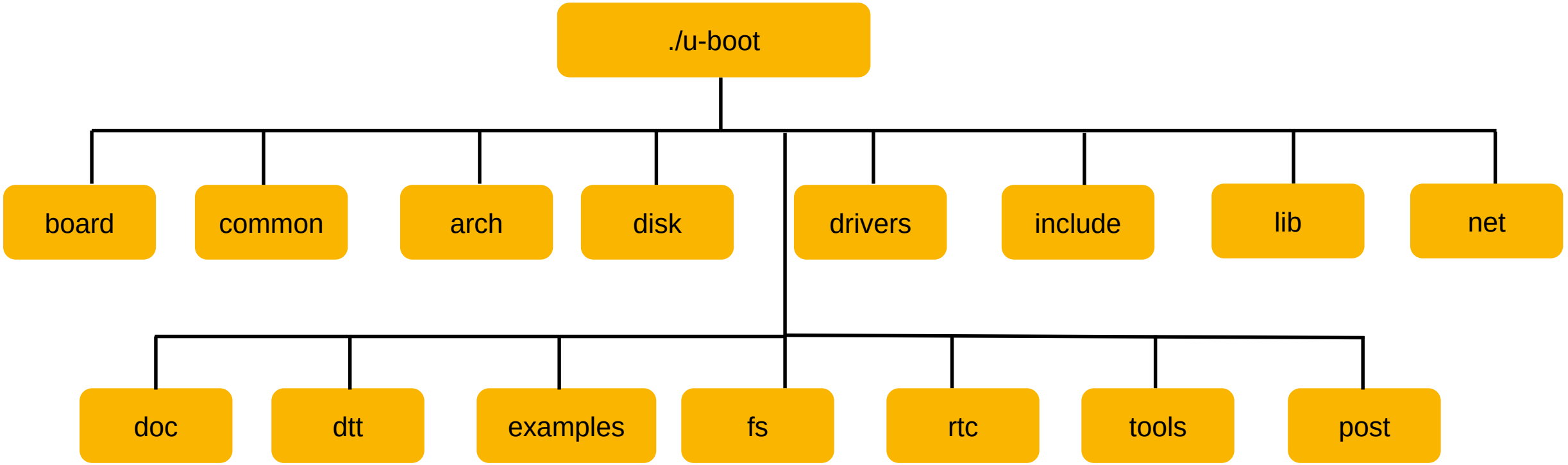
WILSON LO
APPLICATION ENGINEERING
FTF-DES-N1851
MAY 2016

AGENDA

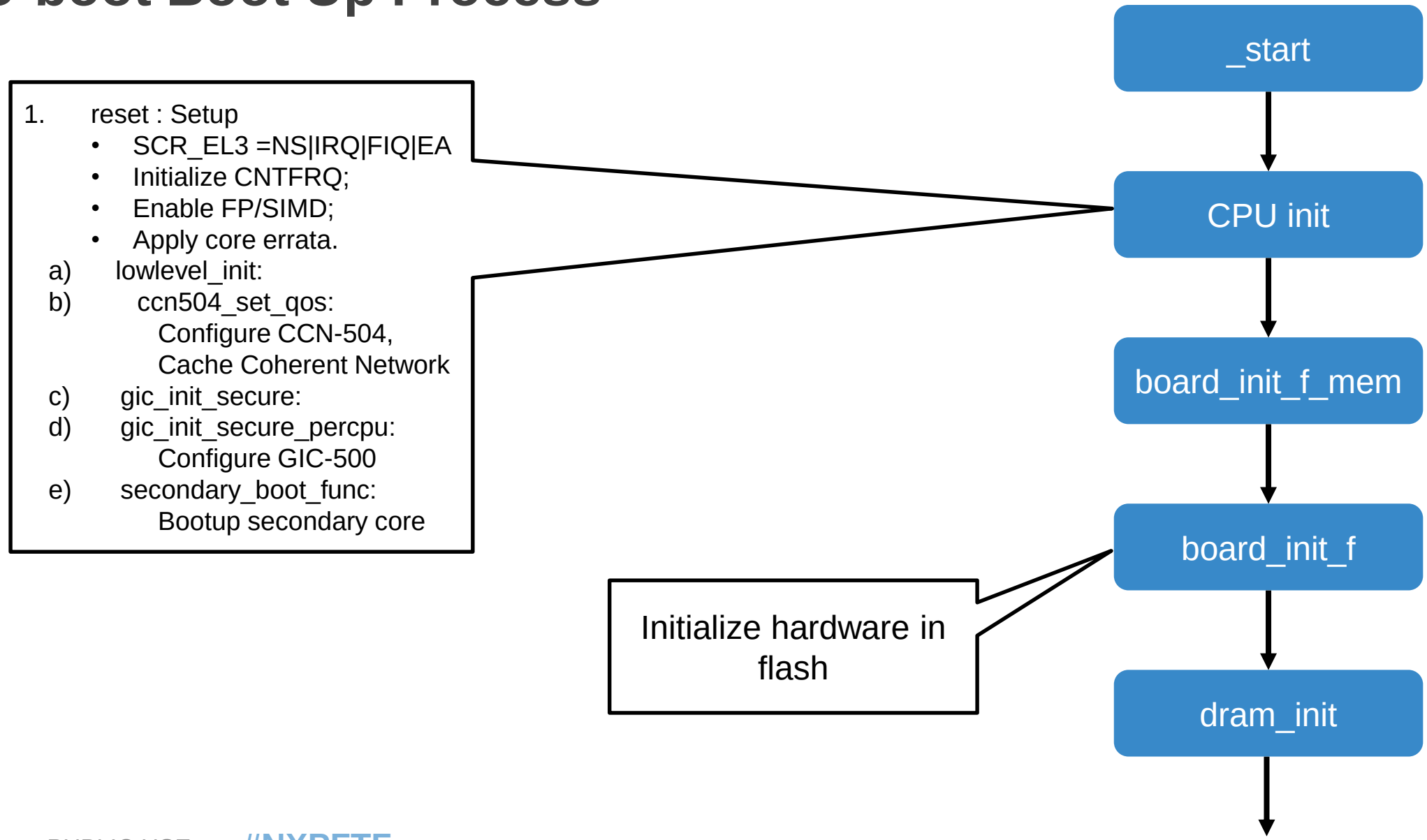
- Initialization of Ethernet Interface
- Ethernet Driver Structure
- Add a New Driver
- Verification of New Driver
- Image Automatic Update

INITIALIZATION OF ETHERNET INTERFACE

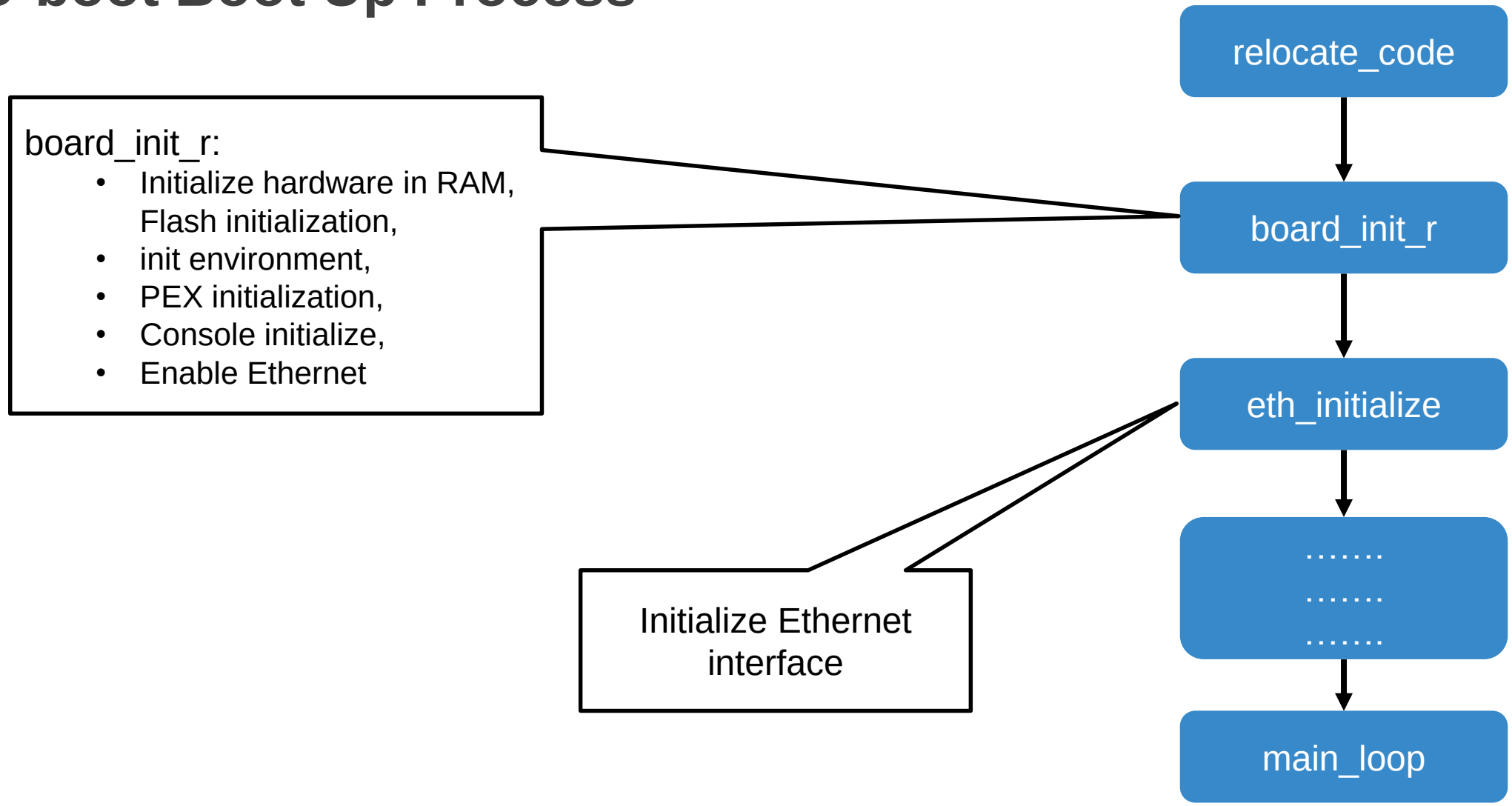
U-boot Source Code Structure



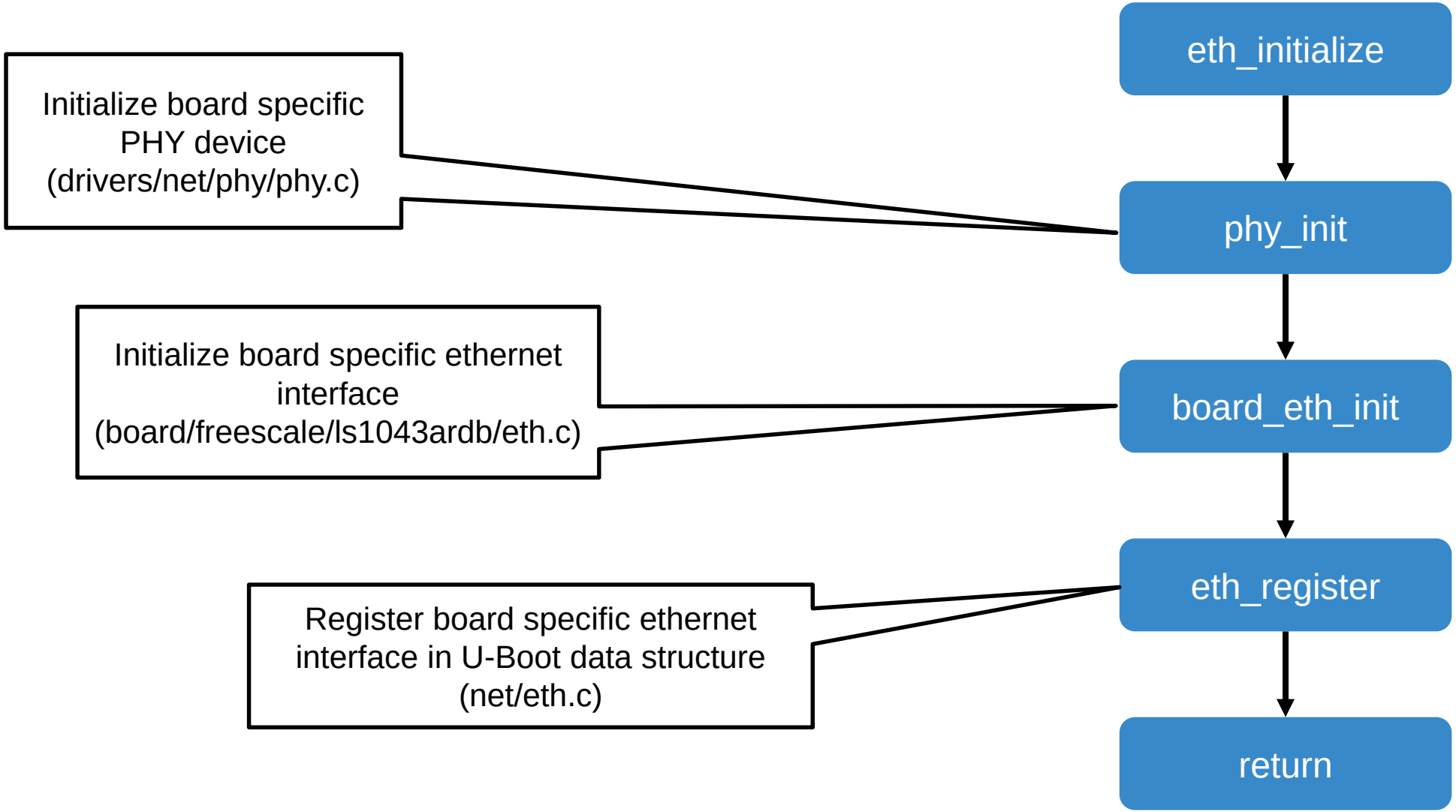
U-boot Boot Up Process



U-boot Boot Up Process



Uboot eth PHY Initialization



ETHERNET DRIVER STRUCTURE

Procedure of Adding New Ethernet PHY Drivers

If the Ethernet PHY used is not already supported within U-Boot, the PHY driver code can be updated to add support for the new device.

For example, if the user wishes to add a new Vitesse PHY, they should follow the example below shown for the VSC8514 PHY.

- In file drivers/net/phy/vitesse.c,
  - Creates a new structure in the format below.
  - The startup and shutdown functions may be reuse of already existing routines but if the device has different programming requirements these can be changed to call new functions.

- In drivers/net/phy/vitesse.c,
  - Ensure the new PHY is registered by adding a phy_register() call, for example:

    phy_register(&VSC8514_driver);

- For other PHY vendors users should refer to the drivers/net/phy options where other files are available such as broadcom.c

Uboot PHY Structure

- PHY device initialization entry (file drivers/net/phy/vitesse.c)

```
int phy_vitesse_init(void)
{
        phy_register(&VSC8641_driver);
        phy_register(&VSC8601_driver);
        phy_register(&VSC8234_driver);
        phy_register(&VSC8244_driver);
        phy_register(&VSC8211_driver);
        phy_register(&VSC8221_driver);
        phy_register(&VSC8574_driver);
        phy_register(&VSC8514_driver);
        phy_register(&VSC8662_driver);
        phy_register(&VSC8664_driver);
        phy_register(&cis8201_driver);
        phy_register(&cis8204_driver);
        return 0;
}
```

Uboot PHY Structure

- PHY driver data structure (file drivers/net/phy/vitesse.c)

```
static struct phy_driver VSC8211_driver = {
        .name = "Vitesse VSC8211",
        .uid = 0xfc4b0,
        .mask = 0xffff0,
        .features = PHY_GBIT_FEATURES,
        .config = &vsctesse_config,
        .startup = &vitesse_startup,
        .shutdown = &genphy_shutdown,
};
```

PHY Device Initialization Routine

- PHY device initialization code (file drivers/net/phy/vitesse.c)

```
static int vitesse_config(struct phy_device *phydev)
{
	/* Override PHY config settings */
	phy_write(phydev, MDIO_DEVAD_NONE,
MIIM_CIS82xx_AUX_CONSTAT,
			MIIM_CIS82xx_AUXCONSTAT_INIT);

	 /* Set up the interface mode */
	phy_write(phydev, MDIO_DEVAD_NONE,
MIIM_CIS82xx_EXT_CON1,
			MIIM_CIS8201_EXTCON1_INIT);

	genphy_config_aneg(phydev);
	return 0;
}
```

ADD A NEW DRIVER

Enable the PHY on U-Boot

- Define the MACRO definition in board_specific_header file, e.g. include/configs/ls1043ardb.h
  - #define CONFIG_PHY_VITESSE
  - #define RGMII_PHY1_ADDR 0x1
  - #define RGMII_PHY2_ADDR 0x2

```
int phy_init(void)
{
#ifdef CONFIG_PHY_AQUANTIA
        phy_aquantia_init();
#endif
#ifdef CONFIG_PHY_ATHEROS
        phy_atheros_init();
#endif
…
…
…
#ifdef CONFIG_PHY_VITESSE
        phy_vitesse_init();
#endif
        return 0;
}
```

Uboot PHY Structure

- PHY device initialization entry (file drivers/net/phy/vitesse.c)

```
int phy_vitesse_init(void)
{
        phy_register(&VSC8641_driver);
        phy_register(&VSC8601_driver);
        phy_register(&VSC8234_driver);
        phy_register(&VSC8244_driver);
        phy_register(&VSC8211_driver);
        phy_register(&VSC8221_driver);
        phy_register(&VSC8574_driver);
        phy_register(&VSC8514_driver);
        phy_register(&VSC8662_driver);
        phy_register(&VSC8664_driver);
        phy_register(&cis8201_driver);
        phy_register(&cis8204_driver);
        return 0;
}
```

Uboot PHY Structure

- PHY driver data structure (file drivers/net/phy/vitesse.c)

```
static struct phy_driver VSC8514_driver = {
        .name = "Vitesse VSC8514",
        .uid = 0x70670,
        .mask = 0xffff0,
        .features = PHY_GBIT_FEATURES,
        .config = &vsc8514_config,
        .startup = &vitesse_startup,
        .shutdown = &genphy_shutdown,
};
```

PHY Device (VSC8541) Initialization Routine

```
int vsc8514_config(struct phy_device *phydev)
{
/* configure register to access 19G */
phy_write(phydev, MDIO_DEVAD_NONE,
PHY_EXT_PAGE_ACCESS,
            PHY_EXT_PAGE_ACCESS_GENERAL);

val = phy_read(phydev, MDIO_DEVAD_NONE,
MIIM_VSC8514_GENERAL19);
if (phydev->interface == PHY_INTERFACE_MODE_QSGMII) {
   /* set bit 15:14 to '01' for QSGMII mode */
   val = (val & 0x3fff) | (1 << 14);
   phy_write(phydev, MDIO_DEVAD_NONE,
         MIIM_VSC8514_GENERAL19, val);
    /* Enable 4 ports MAC QSGMII */
   phy_write(phydev, MDIO_DEVAD_NONE,
     MIIM_VSC8514_GENERAL18, MIIM_VSC8514_18G_QSGMII);}

val = phy_read(phydev, MDIO_DEVAD_NONE,
MIIM_VSC8514_GENERAL18);
/* When bit 15 is cleared for command completed */
while ((val & MIIM_VSC8514_18G_CMDSTAT)&& timeout--)
        val = phy_read(phydev,          MDIO_DEVAD_NONE,
                              MIIM_VSC8514_GENERAL18);
```

```
…
…
phy_write(phydev, MDIO_DEVAD_NONE,
                PHY_EXT_PAGE_ACCESS, 0);

/* configure register to access 23 */
val = phy_read(phydev, MDIO_DEVAD_NONE, MIIM_VSC8514_GENERAL23);
/* set bits 10:8 to '000' */
val = (val & 0xf8ff);
phy_write(phydev, MDIO_DEVAD_NONE,    MIIM_VSC8514_GENERAL23, val);

 /* Enable Serdes Auto-negotiation */
phy_write(phydev, MDIO_DEVAD_NONE,                    PHY_EXT_PAGE_ACCESS,
          PHY_EXT_PAGE_ACCESS_EXTENDED3);
…
phy_write(phydev, MDIO_DEVAD_NONE,    MIIM_VSC8514_MAC_SERDES_CON, val);
…
       genphy_config_aneg(phydev);
       return 0;
}
```

VERIFICATION OF NEW DRIVER

Verification of New Driver

- Define the environment valuable in
  - U-Boot
  - Board IP address (ipaddr)
  - Active Ethernet interface (ethact)
  - Interface MAC address (ethaddr)

```
=> print
baudrate=115200
bootargs=console=ttyS0,115200 root=/dev/ram0
earlycon=uart8250,0x21c0500,115200
bootdelay=3
console=ttyAMA0,38400n8
eth1addr=00:e0:0c:00:77:01
eth2addr=00:e0:0c:00:77:02
eth3addr=00:e0:0c:00:77:03
eth4addr=00:e0:0c:00:77:04
eth5addr=00:e0:0c:00:77:05
eth6addr=00:e0:0c:00:77:06
ethact=FM1@DTSEC1
ethaddr=00:e0:0c:00:77:00
ethprime=e1000#0
ipaddr=192.168.1.2
serverip=192.168.1.1
stderr=serial
stdin=serial
stdout=serial
```

Verification of New Driver

```
U-Boot 2015.01QorIQ-SDK-V1.7+g8a1f578 (Jun 09 2015 -
02:38:26)

Clock Configuration:
      CPU0(A53):1500 MHz  CPU1(A53):1500 MHz
CPU2(A53):1500 MHz
      CPU3(A53):1500 MHz
      Bus:      400  MHz  DDR:      1600 MHz
Reset Configuration Word (RCW):
      00000000: 0810000f  0c000000 00000000 00000000
      00000010: 14550002 80004002 e0025000 61002000
      00000020: 00000000 00000000 00000000 00038800
      00000030: 00000000 00001100 00000096 00000001
Board: LS1043ARDB, boot from vBank 0
CPLD:  V1.4
PCBA:  V2.0
SERDES Reference Clocks:
SD1_CLK1 = 156.25MHZ, SD1_CLK2 = 100.00MHZ
I2C:   ready
```

```
DRAM:  2 GiB (DDR4, 32-bit, CL=12, ECC off)
Waking secondary cores to start from fff29000
All (4) cores are up.
Using SERDES1 Protocol: 5205 (0x1455)
Flash: 128 MiB
NAND:  512 MiB
MMC:   FSL_SDHC: 0
EEPROM: Invalid ID (5a 5a 5a 5a)
PCIe1: disabled
PCIe2: Root Complex no link, regs @ 0x3500000
PCIe3: Root Complex no link, regs @ 0x3600000
In:    serial
Out:   serial
Err:   serial
Net:   fman_port_enet_if:68: port(FM1_DTSEC3) is OK
fman_port_enet_if:74: port(FM1_DTSEC4) is OK
Fman1: Uploading microcode version 106.4.15
FM1@DTSEC1, FM1@DTSEC2, FM1@DTSEC3, FM1@DTSEC4,
FM1@DTSEC5, FM1@DTSEC6, FM1@TGEC1
Hit any key to stop autoboot:  0
```

Verification of New Driver

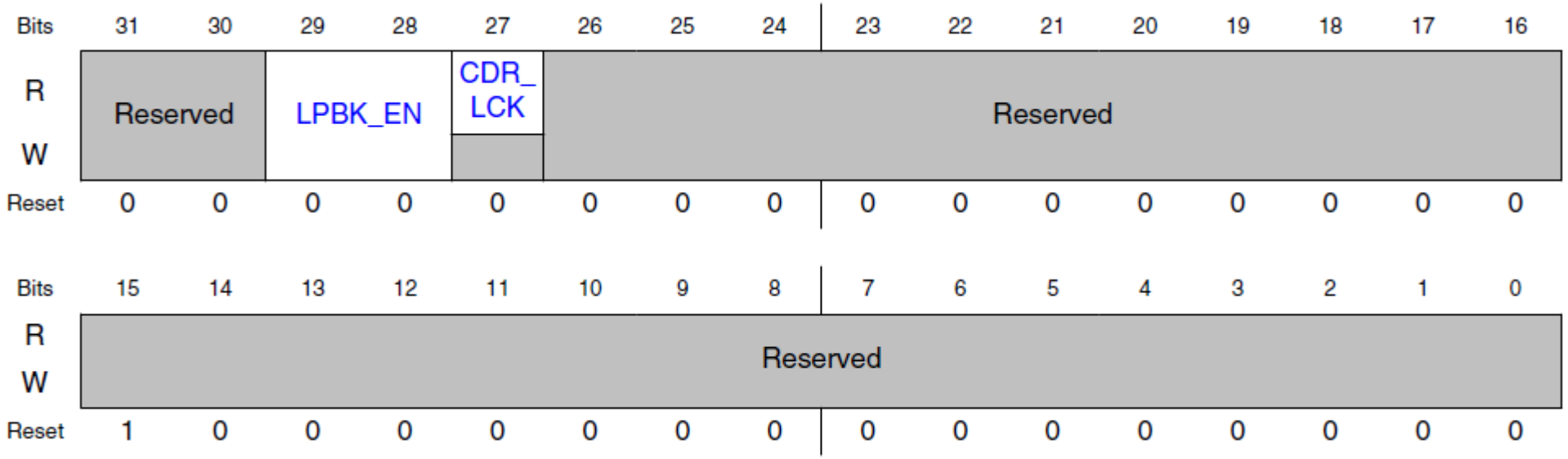
- Access PHYs through MDIO command

```
=> mdio list
FSL_MDIO0:
1 - RealTek RTL8211F <--> FM1@DTSEC3
2 - RealTek RTL8211F <--> FM1@DTSEC4
4 - Vitesse VSC8514 <--> FM1@DTSEC1
5 - Vitesse VSC8514 <--> FM1@DTSEC2
6 - Vitesse VSC8514 <--> FM1@DTSEC5
7 - Vitesse VSC8514 <--> FM1@DTSEC6
FM_TGEC_MDIO:
1 - Aquantia AQR105 <--> FM1@TGEC1
```

Verification of New Driver

- PING – Digitial loopback
  - Test Control/Status Register 3 - Lane n (LN*n*TCSR3)
  - Execute “ping $ipaddr” command

```
FM1@DTSEC5, FM1@DTSEC6, FM1@TGEC1
Hit any key to stop autoboot:  0
⇒ mw 1ea083c 10000000
⇒ md 1ea083c 1
1ea083c : 18000000
⇒ ping $ipaddr
192.168.1.2 is alive
```



Verification of New Driver

- PING – PHY loopback
  - Set the PHY in Loopback mode (PHY device dependent)
  - Execute “ping $ipaddr” command

```
FM1@DTSEC5, FM1@DTSEC6, FM1@TGEC1
Hit any key to stop autoboot:  0

⇒ mdio write FM1@DTSEC5 0 6100
⇒ ping $ipaddr
192.168.1.2 is alive
```

Verification of New Driver

- PING command

```
FM1@DTSEC5, FM1@DTSEC6, FM1@TGEC1
Hit any key to stop autoboot:  0

⇒ serverip=192.168.1.1
⇒ ping $serverip
192.168.1.1 is alive
```

Verification of New Driver

- Test the interface with heavy traffic
  - tftp <local memory address> <file name>

```
FM1@DTSEC5, FM1@DTSEC6, FM1@TGEC1
Hit any key to stop autoboot:  0

⇒ serverip=192.168.1.1
⇒ Setenv ethact FM1@DTSEC5
⇒ tftp a0000000 kernel-ls1043ardb.itb
Using FM1@DTSEC5 device
TFTP from server 192.168.1.1; our IP address is 192.168.1.2
Filename 'kernel-ls1043ardb.itb'.
Load address: 0xa0000000
Loading:
#################################################################
#################################################################
#########################
21.7 MiB/s
done
Bytes transferred = 2316647 (235967 hex)
⇒
```

IMAGE AUTOMATIC UPDATE

Description

- A command to upgrade/re-program all SDK images or selected image on FSL boards (QDS & RDB) automatically from FSL server through network or USB or SDHC interfaces
- The existing on board images will be back up on the alternative NOR flash bank before re-programming the flash

Command Usage

NAME

update – upgrade/re-program all SDK images or selected image on FSL boards automatically. In default mode, the command will start the image download through the network interface. If this connection fails, the update process will then start on USB interface. If USB interface is not available, it will then try to connect via the SDHC interface.

SYNOPSIS

update

update [–if ethernet/usb/sdhc] [-f u-boot/linux/FMan uCode]

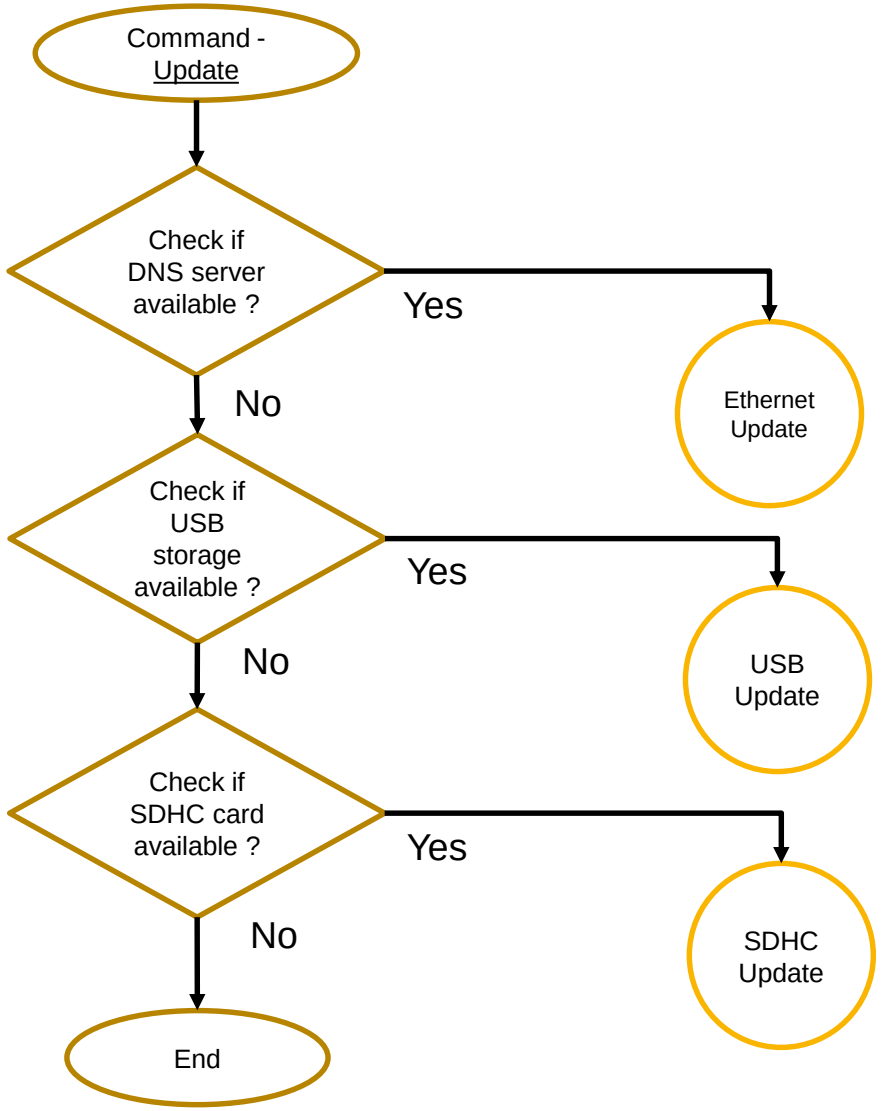
OPTIONS

<default> : In default mode, the command will start the image download through the network interface. If this connection fails, the update process will then start on USB interface. If USB interface is not available, it will then try to connect via the SDHC interface.

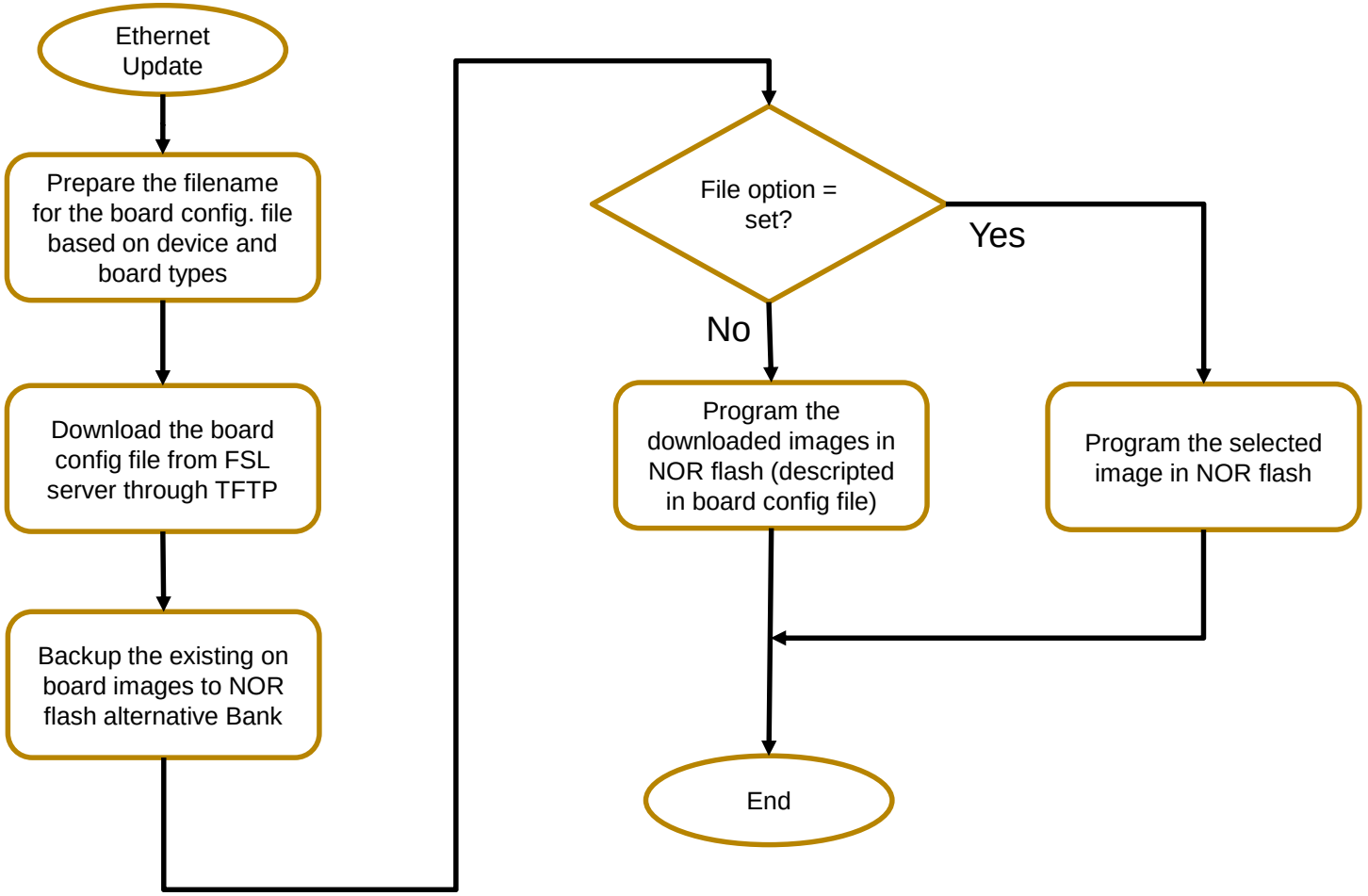
-if <interface name> : ethernet (default) or usb or sdhc

-f <image> : u-boot, linux, dtb or ramdisk image. Without this option, all images will be programmed on NOR flash memory

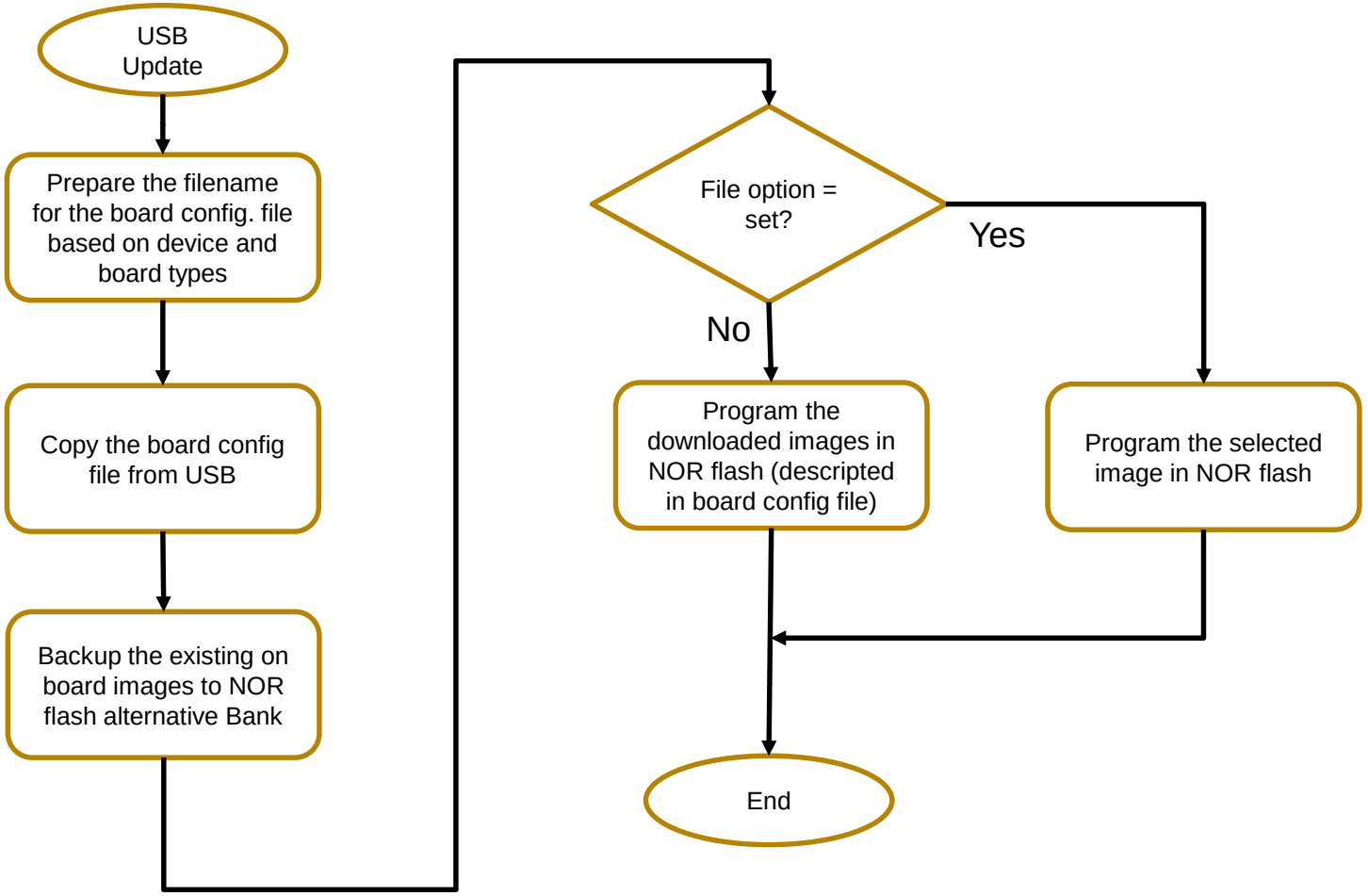
Design Concept – Main Routine



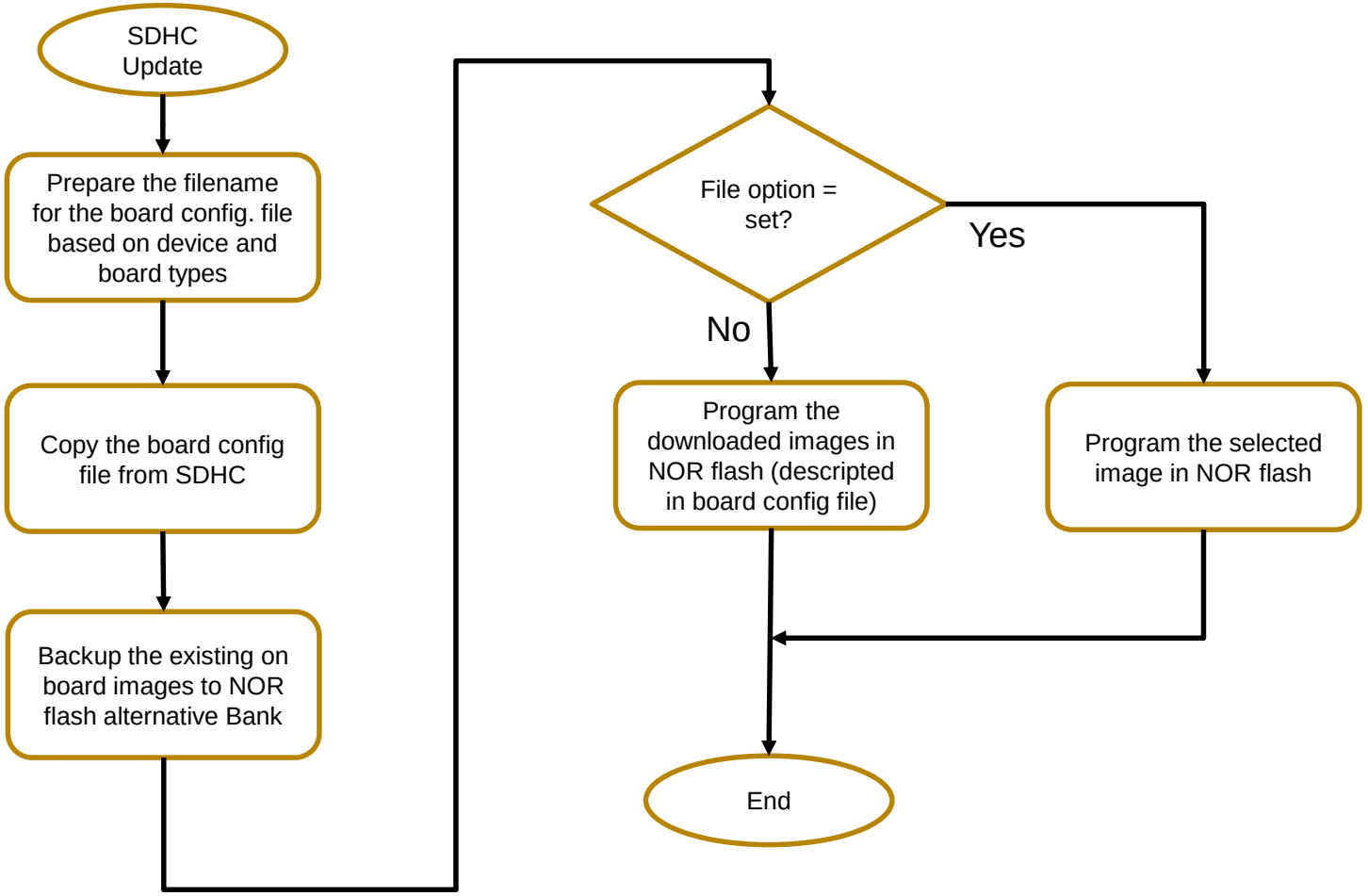
Design Concept – Command Through Ethernet



Design Concept – Command Through USB



Design Concept – Command Through SDHC



Board Configuration File Naming Rule

A. Update all image
   - <Device name><Board type>_<Device_SVR>.its
   - e.g. LS1043A_82480020.its

B. Update individual image
   - <Device name><Board type>_<Device_SVR>_<number>.its
   - number
     - 1= U-Boot image,
     - 2= Linux image,
     - 3=uCode
   - e.g. LS1043A_82480020_1.its

System Recovery SD Card Images

- Description
  - In case the on board U-Boot image is corrupted. Users can boot up the board with U-Boot and recover all images, including RCW, U-boot, Linux kernel, Device Tree File, Ramdisk and also FMan uCode.

- Procedure
  - Download the SD card image for your board from Freescale website
  - Change the on board dip switch to boot from SDCARD (refer to the board specific Quick Start Guide document)
  - Insert the SD Card on your board
  - Power on your board. You will see the U-Boot boot up.
  - Run the “update –if SDHC” command to recover all images

Command Console Output

```
=> help update
update - Automatic recover/upgrade images
Usage:
update update [-if <ethernet/usb/sdhc>] [-f <uboot/linux/mc/dpc/dpl>]

⇒    Update
Please server for upgrade/recover :
1) External server
2) Customer local TFTP server


Entry a location of your server : 2
Entry your server IP :10.81.55.3
Un-Protected 1 sector
Erasing Flash...
. done
Erased 1 sectors
Writing to Flash... 9....8....7....6....5....4....3....2....1....9....8....7....6....5....4....3....2....1....done
Protected 1 sectors
PING to the server IP
Using e1000#0 device
host 10.81.55.3 is alive
File to download = LS1043ARDB_recover.itb through Ethernet
Auto-update from TFTP: trying update file 'LS1043ARDB_recover.itb'
Using e1000#0 device
TFTP from server 10.81.55.3; our IP address is 10.81.52.199
Filename 'LS1043ARDB_recover.itb'.
Load address: 0x80100000
```

```
Loading:
#################################################################################
#########################################################
3.7 MiB/s
done
Bytes transferred = 2316647 (235967 hex)
Processing update 'update@1' :sha1+

***** U-Boot images *****
Erasing 0x60100000 - 0x60103ffff
..... done
Erased 4 sectors
Copying to flash...9....8....7....6....5....4....3....2....1....done
Processing update 'update@2' :sha1+
Copying to flash...9....8....7....6....5....4....3....2....1....done


***** RamDisk image *****
Erasing 0x680800000 - 0x6808fffff
. done
Erased 18 sectors
Copying to flash...9....8....7....6....5....4....3....2....1....done
Processing update 'update@3' :sha1+
Copying to flash...9....8....7....6....5....4....3....2....1....done

***** Linux Image *****

Erasing 0x680700000 - 0x68071ffff
. done
Erased 4 sectors
Copying to flash...9....8....7....6....5....4....3....2....1....done
=>
```

SOFTWARE PRODUCTS AND S

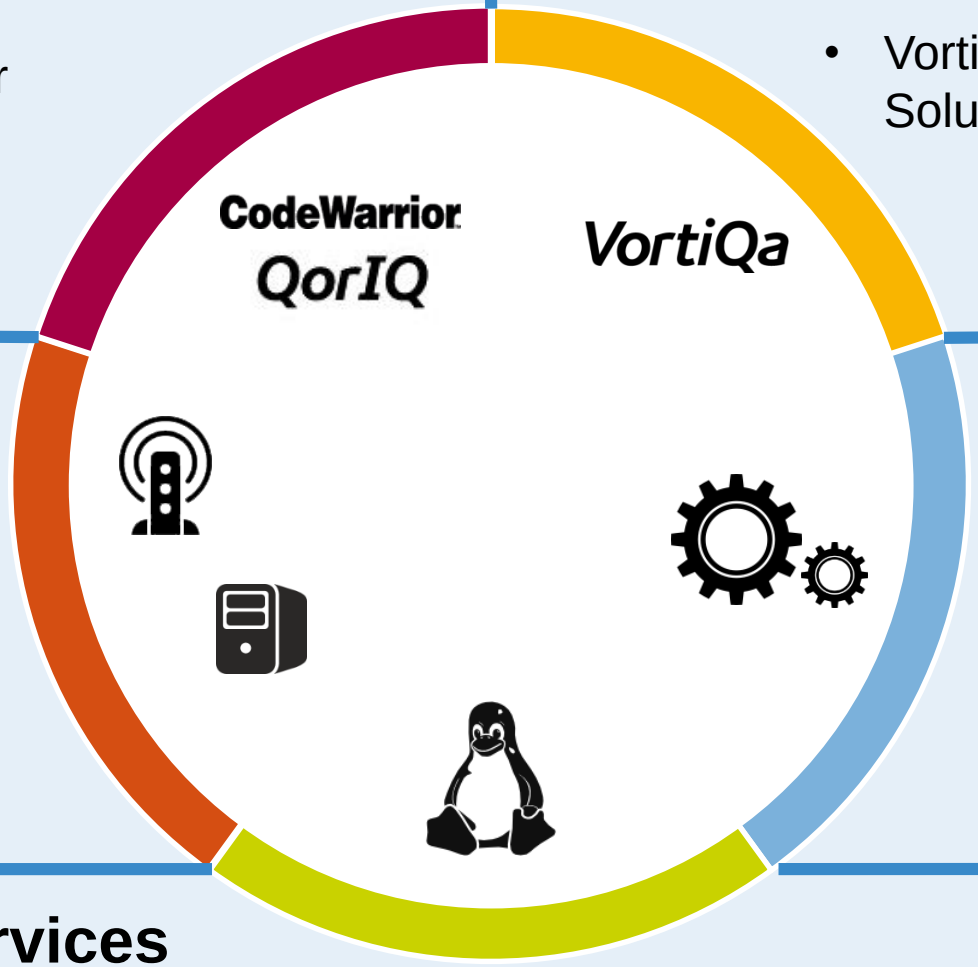
Visit us in the Tech Lab – #247

Development Tools

- CodeWarrior

Runtime Products

- VortiQa Software Solutions



Solutions Reference

- IoT Gateway
- OpenWRT+

Integration Services

- Security Consulting
- Hardened Linux

Linux® Services

- Commercial Support
- Performance Tuning

Accelerate **Customer Time-to-Market**

Deliver **Commercial Software, Support, Services and Solutions**

Simplify **Software Engagement with NXP**

Create **Success!**

NXP

Info online at www.nxp.com/networking-services

SECURE CONNECTIONS
FOR A SMARTER WORLD

ATTRIBUTION STATEMENT