Application Report
SPRACI4A–September 2018–Revised December 2018

AWR1642/AWR1843 Application Startup Sequence

ABSTRACT

This document describes the startup sequence for the AWR1642/AWR1843 application.

Contents

List of Figures

Trademarks

ARM, Cortex are registered trademarks of Arm Limited.
All other trademarks are the property of their respective owners.

1 Introduction

The AWR1642/AWR1843 device can be generally divided into three subsystems:

- Master subsystem (MSS): ARM® Cortex®-R4F and associated peripherals, hosts the user application.
- DSP subsystem: TI C674x and associated peripherals, hosts the user application.
- Radar/millimeter wave block: Programmed using predefined message transactions specified by TI (reference driver provided by TI).

User application components (R4F and DSP) are expected to be stored in the serial data flash (SDF) and interfaced to the AWR1642/AWR1843 device over the quad serial peripheral interface (QSPI).

The bootloader of the AWR1642/AWR1843 device relocates the image stored in the SDF to the R4F and DSP memory subsystems. Toward the end of this process, the bootloader passes the R4F application of the control user. The user image is responsible for unhalting (starting execution) of the DSP core.

Boot up of the R4F application requires a system initialization sequence for a healthy application execution. This document explains the sequence in which a developer writes an MSS (R4F) application on the AWR1642/AWR1843 device.

2 Startup Sequence

The MSS application must have the following startup sequence at bootup time.

1. Clear all the ESM group errors and register interrupt handlers for any future ESM errors. If using TI RTOS, then ESM errors are cleared before entering main. For any other RTOS, check the RTOS implementation to clear ESM errors.[*ESM_init*]
2. Unhalt BSS (a register write) and wait on the completion of the APLL closed loop calibration (poll on another status register). Wait for RadarSS power up event. [*SOC_unhaltBSS, SOC_waitAPLLCalibration*]

   The clock after this point is stable at 200 MHz for MSS and BSS, and 600 MHz for DSP.
3. Initialize Memory Protection Unit (MPU) settings. [*SOC_mpu_config*]
4. Set clock source for QSPI, CAN, and CAN-FD peripheral if the application uses any of these interfaces. [*SOC_setPeripheralClock*]

Copyright © 2018, Texas Instruments Incorporated

5. Unhalt DSS: refer to the mmWave-SDK implementation. [*SOC_unhaltDSS*]
6. If the SOC is a secure device, disable the firewall for JTAG and LOGGER (UART), which is required by the application. For code implementation, refer to the mmWave-SDK. [*SOC_isSecureDevice*]

   This step is required only during the development phase, and not in the production version of the application.
7. Start the BIOS and further initialize required peripherals. [*BIOS_start*]

NOTE: mmWave-SDK functions have been mapped to each of the above steps for reference purposes.

The following is a reference code snippet for the above steps which the developer must implement in the MSS application. Refer to the mmWave-SDK for detailed information regarding these APIs.

```
/*! \brief
 * Registers Read/Write MACROS
 */
#define REG_WRITE32(w_addr, w_data)      (*((uint32 *)((w_addr)))) = ((uint32)((w_data)))

#define REG_READ32(w_addr)                    (*((uint32 *)((w_addr))))


void main(void)
{
    /* Clear ESM interrupts */
    ESM_init ();

    /* unhalt BSS */
    SOC_unhaltBSS();

    /* wait for BSS power up */
    waitAPLLCalibration();

    /* initialize MPU (Memory Protection Unit) settings, refer mmWave-SDK for implementation */
    SOC_mpu_config();

    /* Configure the peripheral module clock source and divisor (if required). Refer mmWave-
SDK */
    SOC_setPeripheralClock();

    /* un-halt DSS */
    SOC_unhaltDSS();

    /* Check if the SOC is a secure device. Please refer the mmWave-SDK for implementation */
    if (SOC_isSecureDevice(socHandle, &errCode))
    {
        /* Disable firewall for JTAG and LOGGER (UART) which is needed by the demo
            Please refer the mmWave-SDK for implementation */
        SOC_controlSecureFirewall(…..);
    }

    /* Create a Task */
    Task_create(TaskFunction, &taskParams, NULL);

    /* Start BIOS */
    BIOS_start();
}
void ESM_init ()
{
    /* [ESMSR1] ESM Group 1: 0-31 errors, write-clear */
    REG_WRITE32 (0xFFFFF518, REG_READ32 (0xFFFFF518));

    /* [ESMSR2] ESM Group 2: 0-31 errors, write-clear */
```


Copyright © 2018, Texas Instruments Incorporated

```
    REG_WRITE32 (0xFFFFF51C, REG_READ32 (0xFFFFF51C));

    /* [ESMSR3] ESM Group 3: 0-31 errors, write-clear */
    REG_WRITE32 (0xFFFFF520, REG_READ32 (0xFFFFF520));

    /* [ESMSR4] ESM Group 1: 32-63 errors, write-clear */
    REG_WRITE32 (0xFFFFF558, REG_READ32 (0xFFFFF558));

    /* [ESMSSR2] ESM Group 2 Shadow register: 0-31 errors, write-clear */
    REG_WRITE32 (0xFFFFF53C, REG_READ32 (0xFFFFF53C));
}

void SOC_unhaltBSS()
{
    /* Clear MSS_TOPRCM->BSSCTL[31:16] bits to un-halt BSS */
    REG_WRITE32 (0xFFFFE108, (REG_READ32 (0xFFFFE108) & 0x0000FFFF));
}

int waitAPLLCalibration()
{
    /* Read MSS_TOPRCM->SPARE0[19:16], to confirm that BSS power up is done */
    while((REG_READ32 (0xFFFFE1EC) & 0x000F0000) == 0xF0000);
}

SOC_unhaltDSS()
{
    uint32 powerStatus, unhaltStatus;
    /* Get current DSS status:  DSS_REG->GEMPWRSMCFG4[17] */
    unhaltStatus = ((REG_READ32 (0x500006CC) >> 17) & 0x1);
    /* check if DSS_STC is triggered by Bootloader */
    if(unhaltStatus)
    {
        /* check if DSS_STC-> STCGSTAT is triggered by Bootloader */
        if(REG_READ32 (0x50040014) & 0x1)
        {
            /* clear 3rd bit before un-halting DSS: DSS_REG-> STCPBISTSMCFG1[3] */
            REG_WRITE32 (0x5000074C, (REG_READ32 (0x5000074C) & 0xFFFFFFF7));
        }
        /* clear 17th bit - to un-halt DSS: DSS_REG-> GEMPWRSMCFG4[17] */
        REG_WRITE32 (0x500006CC, (REG_READ32 (0x500006CC) & 0xFFFDFFFF));

        while(1)
        {
            /* Get the power mode status: Have we transitioned?:
               DSS_REG-> GEMPWRSMCFG3[19:18] */
            powerStatus = (REG_READ32 (0x500006C8) & 0xC0000);
           if(powerStatus == 0xC0000)
            {
                  /* YES: Transitioning has been done. */
                  break;
             }
        }
        /* clear 18th bit to enable monitoring event outside from DSS
           DSS_REG-> GEMPWRSMCFG4[18]) */
        REG_WRITE32 (0x500006CC, (REG_READ32 (0x500006CC) & 0xFFFBFFFF));
    }
}
```

Figure 1 explains the control flow of the bootloader and MSS application.

Copyright © 2018, Texas Instruments Incorporated

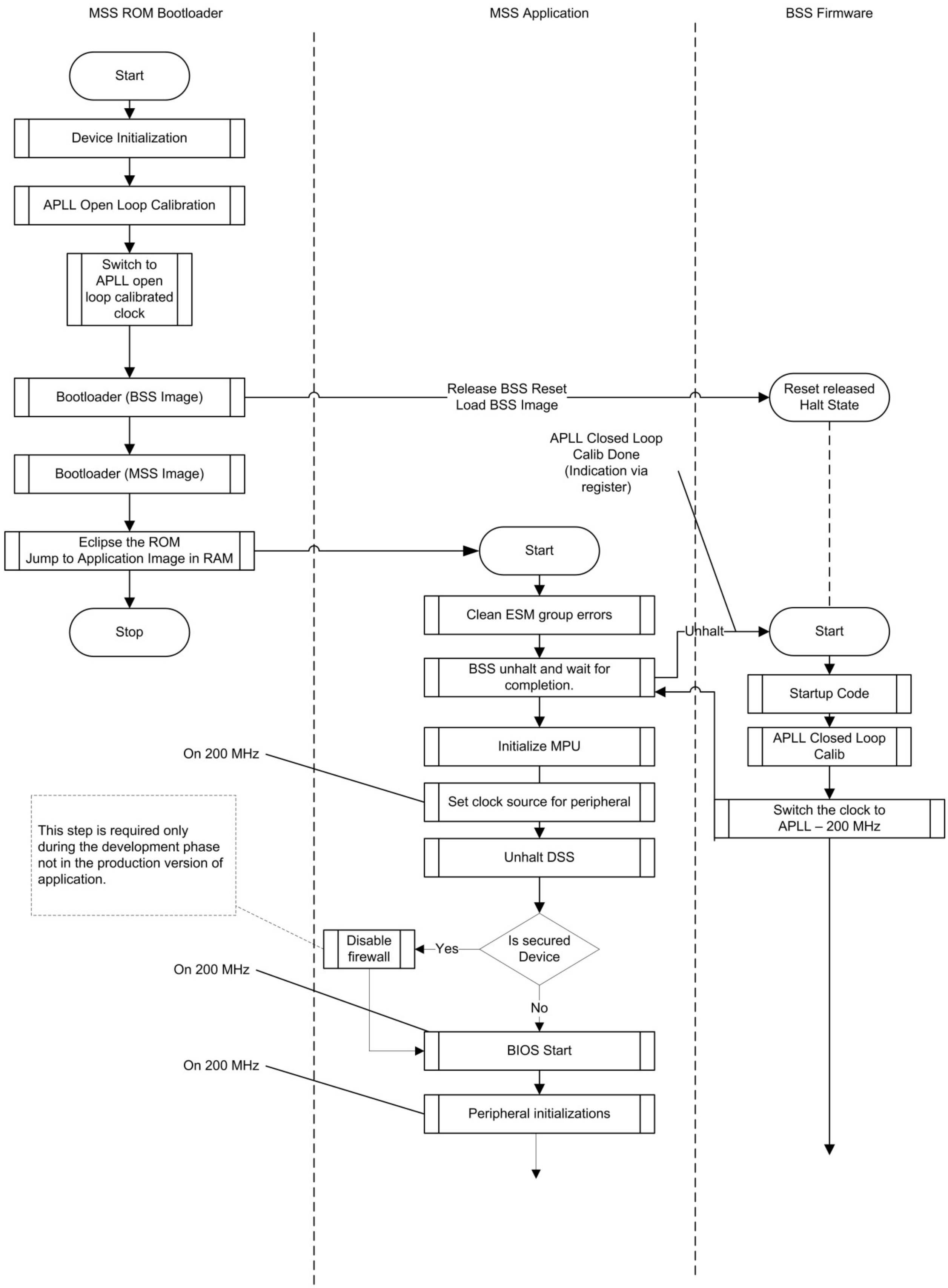

Figure 1. Bootloader Control Flow

Copyright © 2018, Texas Instruments Incorporated

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from Original (September 2018) to A Revision **Page**

Copyright © 2018, Texas Instruments Incorporated

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES "AS IS" AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI's products are provided subject to TI's Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI's provision of these resources does not expand or otherwise alter TI's applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2018, Texas Instruments Incorporated