Application Note

MSPM0 – ADC Attach on AM62x and AM62L using SPI

Divyansh Mittal, Anshu Madwesh, Yashraj Motwani, Anil Swargam, Anshu Choudhary, Krunal Bhargav, Alisa Thomas

ABSTRACT

This application note describes how to integrate ADC present on MSMP0 into 2 processors acting as Controllers - AM62x and AM62L, with the help of the Serial Peripheral Interface (SPI) to support high speed ADC data transmission. AM62x is a heterogeneous processor equipped with up to four Arm Cortex A53 processors and one Arm Cortex M4F Core. AM62L is a lite version of the AM62x featuring two Arm Cortex A53 Cores without the Arm Cortex M4 MCU core. The AM62x does not come with an on-board ADC whereas, the AM62L has a ~10 ENOB ADC. This paper aims to demonstrate the process of integrating the MSMP0 Microcontroller's ADC into AM62x and AM62L. This addition can enable the ADC on AM62x and provide higher resolution options for AM62L. The MSPM0 Microcontroller is equipped with one multi-channel ADC, through which we can monitor several analog signals and transmit any/all digital signals and transmit to SoC via SPI. This document will further delve into overall data flow, hardware and software setup, steps to execute application code and the expected results.

Table of Contents

Trademarks

All trademarks are the property of their respective owners.

Copyright © 2026 Texas Instruments Incorporated

1 Introduction

1.1 SPI Transaction Dataflow

We configure the ADC present on MSPM0L130x Microcontroller and establish an SPI interface with AM62x SK/AM62L microprocessor . Here, the Processors - AM62x/AM62L, are configured as the Controller and MSPM0L130x as the Peripheral. To obtain ADC data from any channel, the controller can initiate a SPI transaction with corresponding command in TX buffer. The peripheral on receiving command from controller starts transmitting the ADC data loaded into its TX buffer based on the channel requested. Controller receives the expected number of bytes from peripheral and then ends the transaction. The peripheral continuously keeps on reading and updating ADC data values. The frequency of these updates depends on the timer used to trigger the ADC. [1]

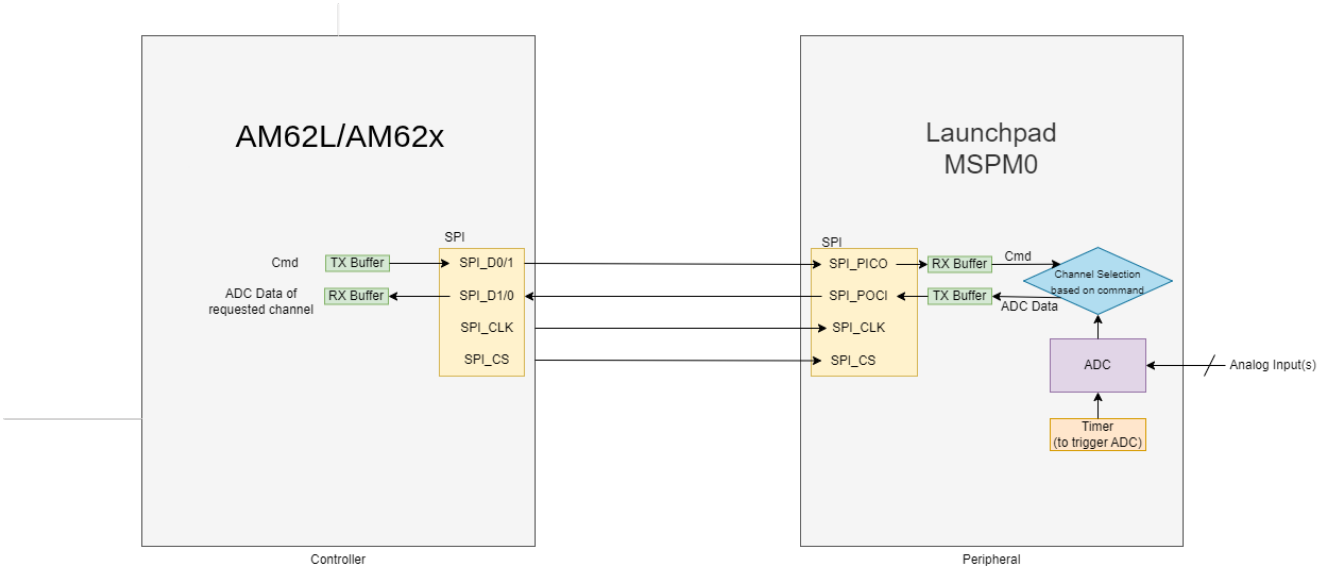


Figure 1-1. Overall Dataflow Between Controller (SK-AM62x or AM62L EVM) and Peripheral (LPMSPM0L130x)

Pipelining in case Full Duplex SPI when Multi-Channel Mode is used:

In full duplex SPI mode, data is simultaneously transmitted and received over the same set of clock cycles. Hence, in the case of multi-channel ADC usage, when command is sent by controller, it simultaneously receives ADC data corresponding to its last command.

The steps involved in running this application are:

1. Hardware setup involving connections between the Controller - SK-AM62x/AM62L EVM and the Peripheral, LP-MSPM0L130x.
2. Software setup that includes one-time pre-execution steps.
3. Execution of applications on both boards to enable SPI transactions.
4. Result analysis.
5. System performance analysis and power consumption estimation.

[1] Note: The use of “Master” and “Slave”, along with “MOSI/MISO” terminology is being considered obsolete. These terms will be replaced with “Controller” and “Peripheral”, and “PICO/POCI” respectively.

Copyright © 2026 Texas Instruments Incorporated

1.2 AM62x and AM62L Processors

AM62x Processor

The AM62x Sitara Microprocessor, shown in Figure 1-2, is a heterogeneous processor designed for a wide variety of embedded applications. SPI can be enabled through MAIN domain on A53 Core. Figure 1-2 shows a simplified block diagram for AM62x.

For more details, see AM62x Sitara Processors Data Sheet.

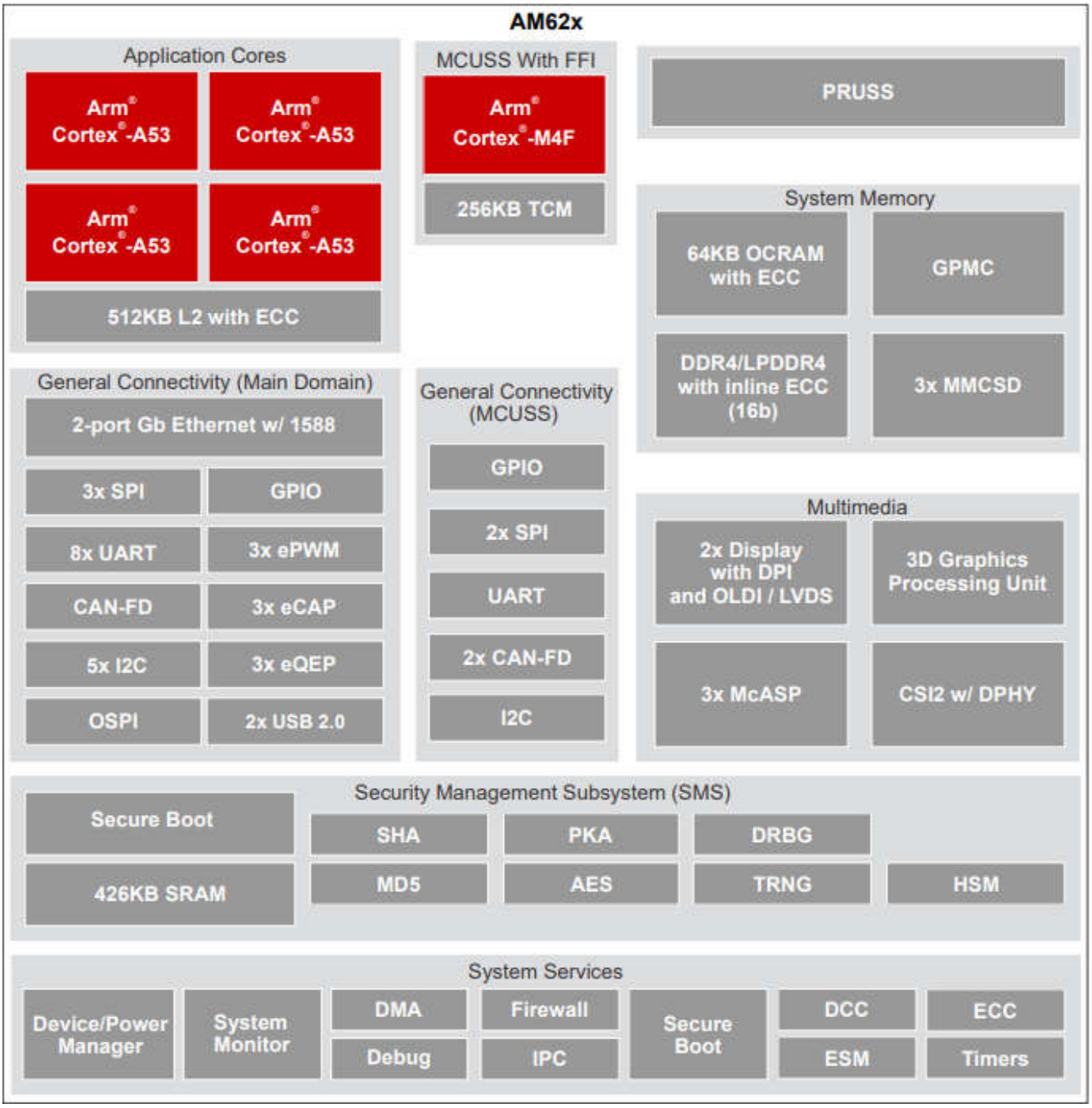


Figure 1-2. AM62x Simplified Block Diagram

Copyright © 2026 Texas Instruments Incorporated

AM62L Processor

The AM62L Sitara Microprocessor, shown in Figure 1-3, is a low-cost & performance optimized processor of the AM6x family. SPI can be enabled through MAIN domain on A53 Core. Figure 1-3 shows a simplified block diagram for AM62L.

For more details, see AM62L Sitara Processors Data Sheet.

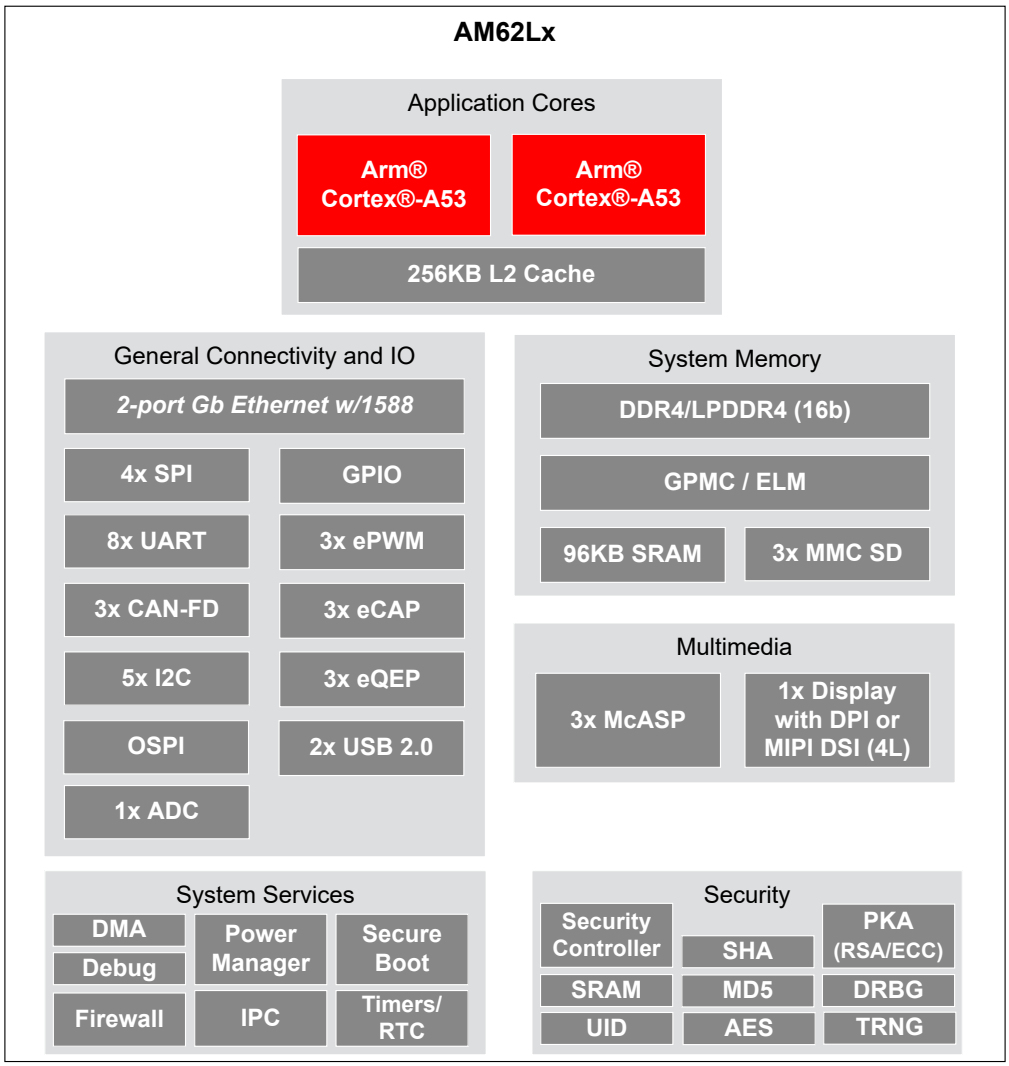


Figure 1-3. AM62L Simplified Block Diagram

Copyright © 2026 Texas Instruments Incorporated

1.3 MSPM0L130x Microcontroller

The MSPM0L130x Microcontroller, shown in Figure 1-4, is an easy-to-use evaluation module (EVM).

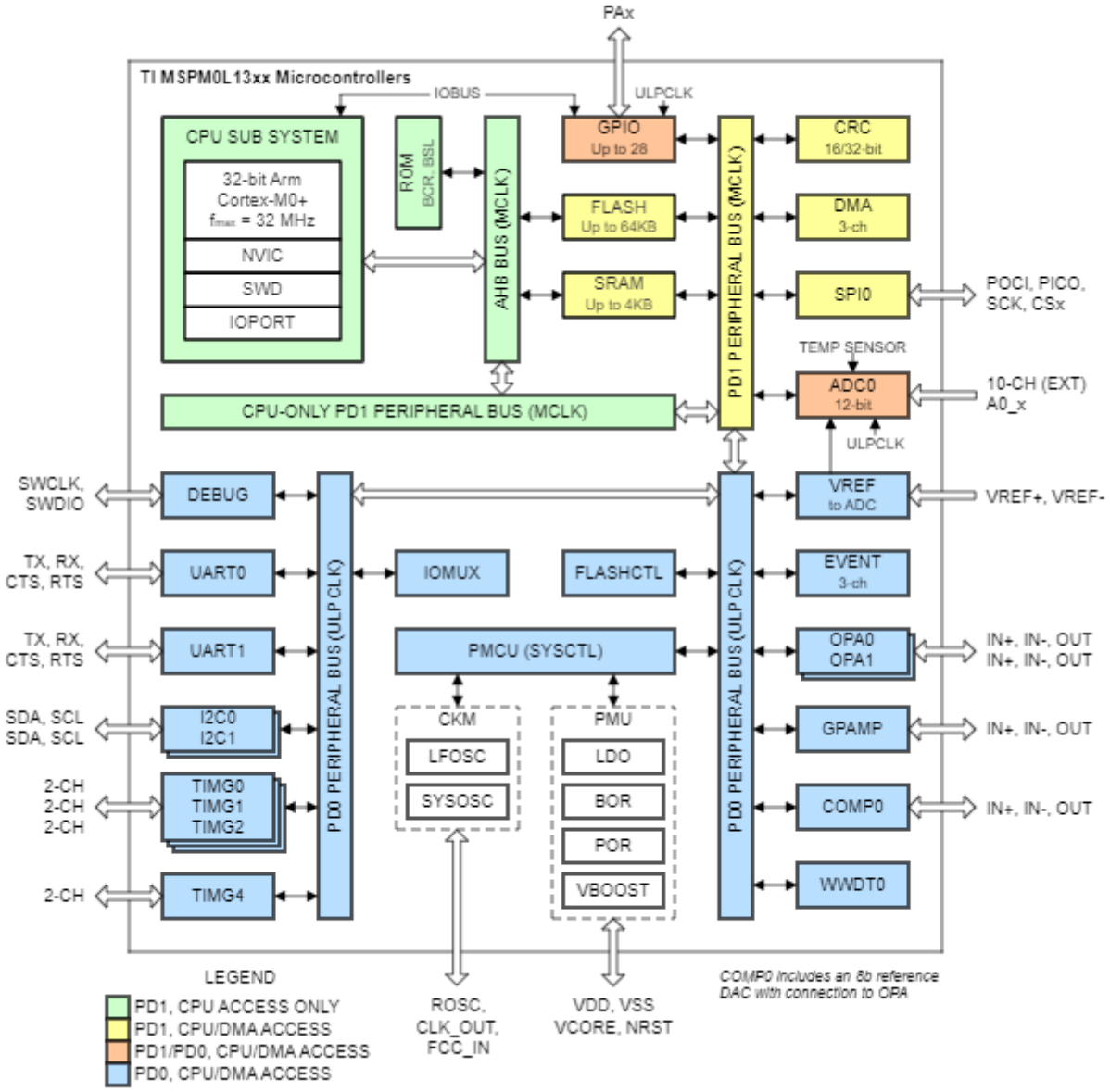


Figure 1-4. MSPM0L130x Simplified Block Diagram

The main compute and interface subsystems on the M0L device are as follows:

- Arm Cortex-M0+ core: This platform can operate at up to 32-MHz frequency. It is cost-optimized MCU offering high-performance analog peripheral integration.
- The onboard ADC supports fast 12-, 10-, and 8-bit analog-to-digital conversions. It implements a 12-bit SAR core, sample and conversion mode control, and up to 4 independent conversion-and-control buffers and offers 1.68-Msps conversion rate at a resolution of 12 bits
- Has SPI module that can operate at speeds as high as 16 Mbits/s.

For more details, see MSPM0L130x Microcontroller Data Sheet.

Copyright © 2026 Texas Instruments Incorporated

2 Hardware Setup

Following cable connections must be performed for the application codes to be run. Note that the pins chosen for these connections are for a particular SPI channel. If any modifications in SPI channel or pin-muxing are made, corresponding pins needs to be checked through datasheet and then used.

2.1 AM62x

The AM62x consists of 2 domains (MAIN - 4xA53 and MCU-1xM4F). The hardware setup is shown for both the cores.

2.1.1 A53 Core Hardware Setup

For using A53 core, the peripheral pins for SPI on SK-AM62x are present in the User Expansion Header. Figure 2-1 shows the hardware setup.

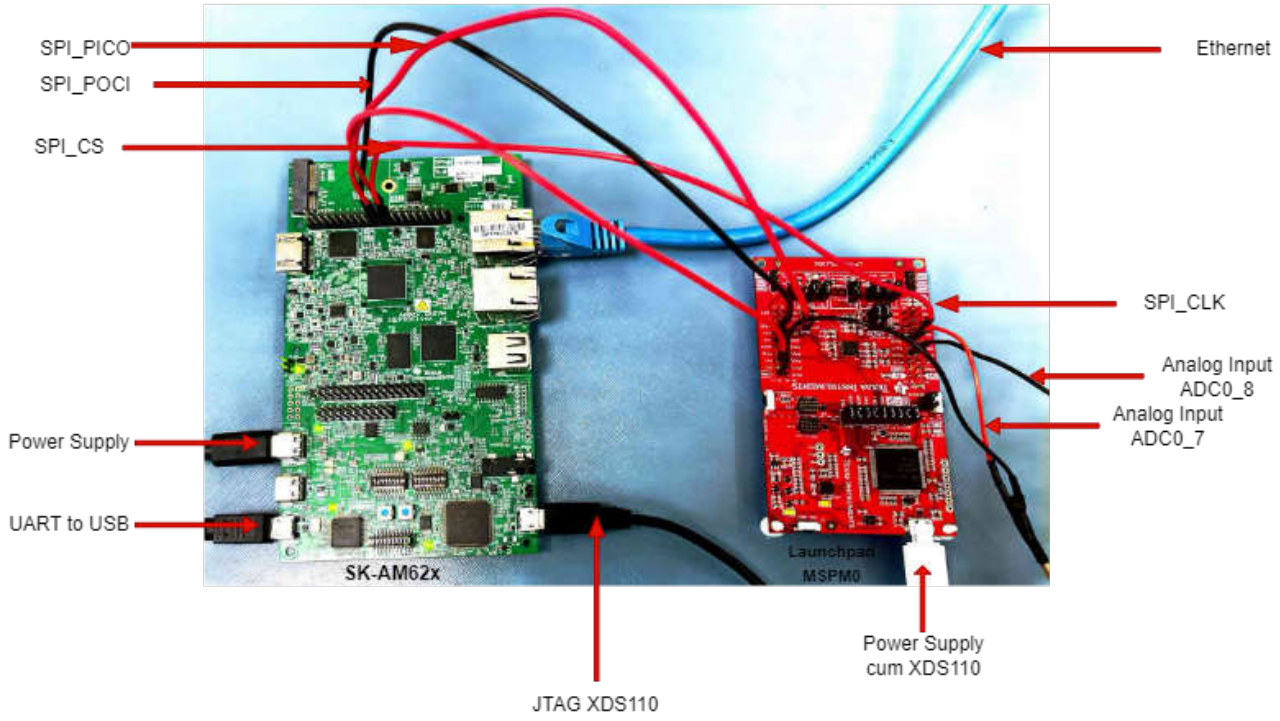


Figure 2-1. Cable Connections between SK-AM62x (A53 Core) and LP-MSPM0L1306 for SPI communication.

- For SK-AM62x:
  - Connect type-C power supply.
  - Connect UART-to-USB and USB for JTAG XDS110 to your computer.
- For LP-MSPM0:
  - Connect power supply cum XDS110 to your computer.
  - Connect analog signal input to J3_PA18 (ADC0_7) in Launchpad MSPM0.
  - Connect analog signal input to J3_PA16 (ADC0_8) in Launchpad MSPM0 if needed.
- For SK-AM62x to LP-MSPM0 connections
  - Connect pin-19 (B13: SPI0_D0) in SK-AM62x User Expansion Connector to J2_PA4 (SPI_POCI) in Launchpad MSPM0.
  - Connect pin-21 (B14: SPI0_D1) in SK-AM62x User Expansion Connector to J2_PA5 (SPI_PICO) in Launchpad MSPM0.

Copyright © 2026 Texas Instruments Incorporated

- Connect pin-23 (A14: SPI0_CLK) in SK-AM62x User Expansion Connector to J1_PA6 (SPI_CLK) in Launchpad MSPM0.
- Connect pin-24 (A13: SPI0_CS0) in SK-AM62x User Expansion Connector to J2_PA3 (SPI_CS(PWM)) in Launchpad MSPM0.

2.1.2 M4F Core Hardware Setup

For using M4F core, the peripheral pins for SPI on SK-AM62x are present in the MCU Header. Figure 2-2 shows the hardware setup.

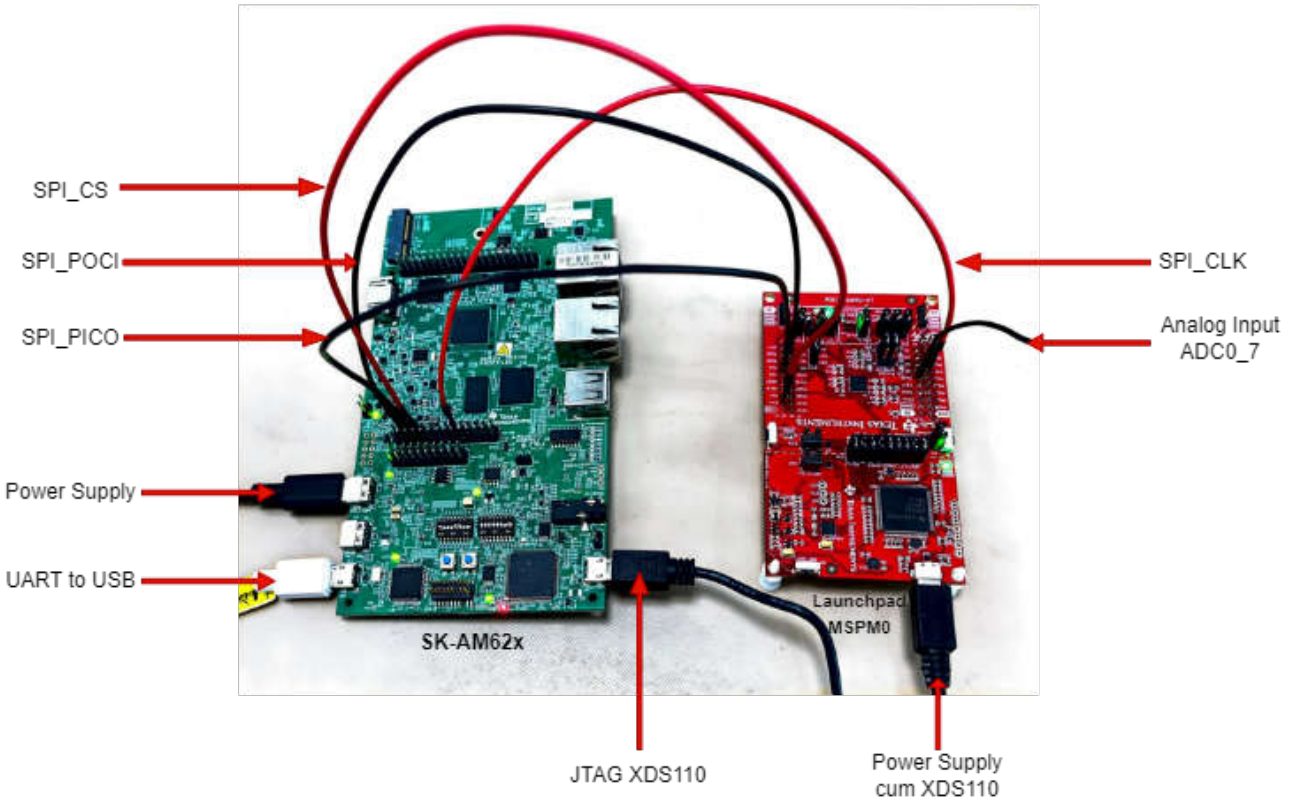


Figure 2-2. Cable Connections between SK-AM62x (M4F Core) and LP-MSPM0L1306 for SPI communication.

- For SK-AM62x:
  - Connect type-C power supply.
  - Connect UART-to-USB and USB for JTAG XDS110 to your computer.
- For LP-MSPM0:
  - Connect power supply cum XDS110 to your computer.
  - Connect analog signal input to J3_PA18 (ADC0_7) in LP-MSPM0.
  - Connect analog signal input to J3_PA16 (ADC0_8) in Launchpad MSPM0 if needed.
- For SK-AM62x to LP-MSPM0 connections:
  - Connect pin-4 (C9: MCU_SPI0_D1) in SK-AM62x MCU-header to J2_PA4 (SPI_POCI) in LP-MSPM0.
  - Connect pin-6 (D9: MCU_SPI0_D0) in SK-AM62x MCU-header to J2_PA5 (SPI_PICO) in LP-MSPM0.
  - Connect pin-8 (B8: MCU_SPI0_CS1) in SK-AM62x MCU-header to J2_PA3 (SPI_CS(PWM)) in LP-MSPM0.
  - Connect pin-18 (A7: MCU_SPI0_CLK) in SK-AM62x MCU-header to J1_PA6 (SPI_CLK) in LP-MSPM0.

Copyright © 2026 Texas Instruments Incorporated

2.2 AM62L

AM62L has only 1 domain - MAIN with 2xA53 cores and the hardware setup for SPI data transfer through ADC was implemented on it. Hardware setup is as below.

2.2.1 A53 Core Hardware Setup

For using A53 core, the peripheral pins for SPI on AM62L are present in the User Expansion Header. Figure 2-3 shows the hardware setup.

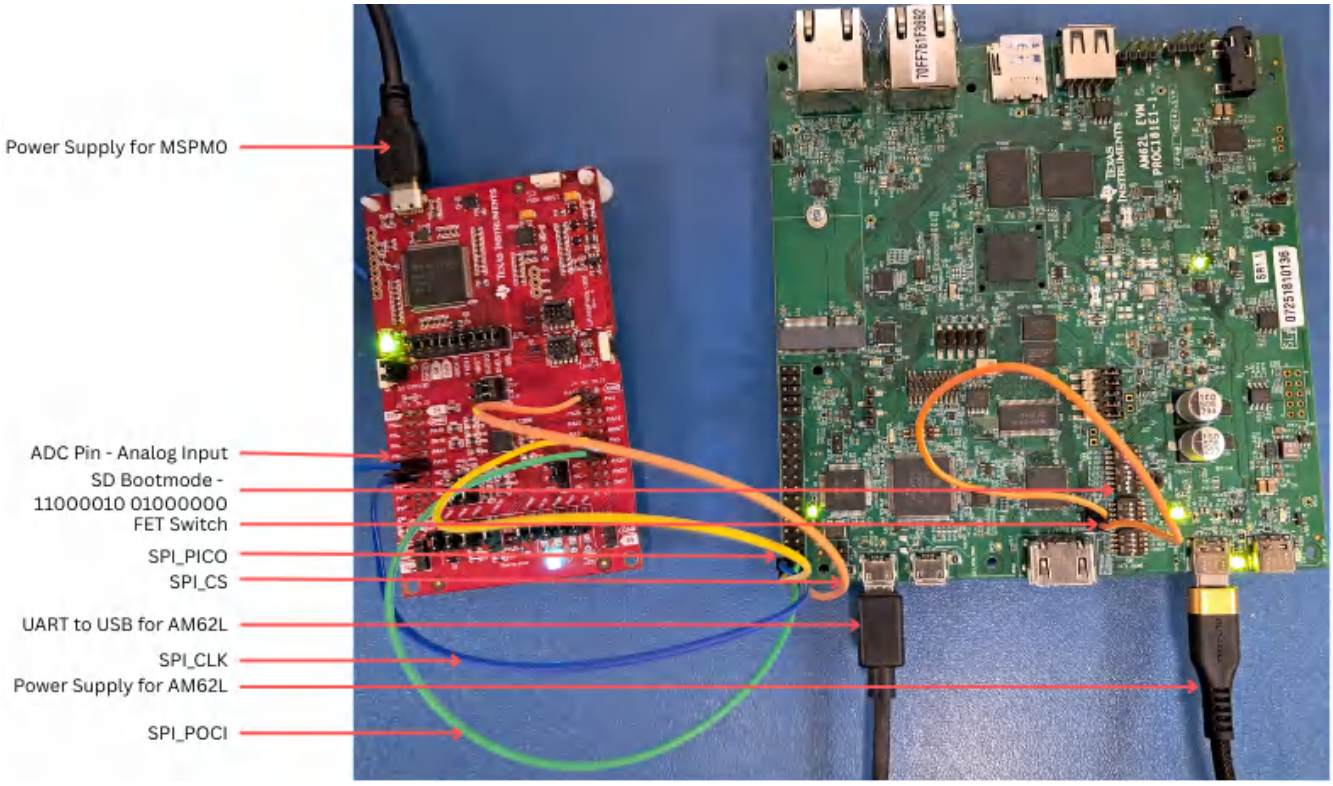


Figure 2-3. Cable Connections between AM62L (A53 Core) and LP-MSPM0L1306 for SPI communication.

- For AM62L:
  - Connect type-C power supply
  - Connect UART-to-USB and USB for JTAG XDS110 to your computer.
- For LP-MSPM0:
  - Connect power supply cum XDS110 to your computer.
  - Connect analog signal input to J3_PA18 (ADC0_7) in LP-MSPM0.
  - Connect analog signal input to J3_PA16 (ADC0_8) in Launchpad MSPM0 if needed.
- For AM62L EVM to MSPM0L connections
  - Connect pin-28 (SPI1_D0_EXP) in AM62L EVM User Expansion Connector to J2_PA4 (SPI_POCI) in Launchpad MSPM0Lx.
  - Connect pin-29 (SPI1_D1_EXP) in AM62L EVM User Expansion Connector to J2_PA5 (SPI_PICO) in Launchpad MSPM0Lx.
  - Connect pin-27 (SPI1_CLK_EXP) in AM62L EVM User Expansion Connector to J1_PA6 (SPI_CLK) in Launchpad MSPM0Lx.
  - Connect pin-30 (SPI1_CS0_EXP) in AM62L EVM User Expansion Connector to J2_PA3 (SPI_CS(PWM)) in Launchpad MSPM0Lx.
- The TMDS62LEVM FET switch J29 controls the SOC output to the HDMI or the IO Expander, with the HDMI being the default. Hence, it needs to be toggled. This configuration ensures that upon boot, Pin 1 of the I2C expander at address 0x23 is automatically set to a High output state.

Copyright © 2026 Texas Instruments Incorporated

3 Software Setup

Here are the steps for setting up the AM62x or AM62L, and the MSPM0L1306.

3.1 Cloning the Beyond SDK GitHub Repository

- Beyond SDK is a GitHub repository that contains some of the files needed for this experiment
- Find the files in *Beyond-SDK/collaterals/appnotes/slaaej0-MSPM0-ADC-Attach*
- Choose the Controller - AM62x or AM62L
- Select one of the folders depending on the application (x_Byte_x_Channel_SPI)
- For the AM62x Controller, the A53 Core folder contains the C file and the M4 Core contains a CCS project
- In Peripheral, there is a CCS project for the MSPM0
- Here is how to clone the GitHub repository:

```
HOST$ mkdir <Beyond-SDK-installation-path>
HOST$ cd <Beyond-SDK-installation-path>
HOST$ git clone https://github.com/TexasInstruments/Beyond-SDK.git
```

3.2 SK-AM62x Software Setup

3.2.1 A53 Core

- Follow the setups provided on AM62x Starter Kit EVM Quick Start Guide.
- Processor SDK Linux for AM62x Version 9.0 was used in the experiment below.
- Setup the SPI driver in Linux by modifying the kernel device tree using the following steps:

1. Find the k3-am625-sk.dts device tree file on the path *<psdk-installation-path>/board-support/ti-linux-kernel/arch/arm64/boot/dts/ti*
2. Modify the file as follows:

Inside &main_pmx0{...} add:

```
main_spi0_pins_default: main-spi0-pins-default {
        pinctrl-single,pins = <
             AM62X_IOPAD(0x01bc, PIN_OUTPUT, 0) /* (A14) SPI0_CLK */
             AM62X_IOPAD(0x01c0, PIN_INPUT, 0) /* (B13) SPI0_D0 */
             AM62X_IOPAD(0x01c4, PIN_OUTPUT, 0) /* (B14) SPI0_D1 */
             AM62X_IOPAD(0x01b4, PIN_OUTPUT, 0) /* (A13) SPI0_CS0 */
        >;
};
```

At the end of the file, add:

```
&main_spi0 {
     status = "okay";
     pinctrl-names = "default";
     pinctrl-0 = <&main_spi0_pins_default>;
     spidev@0 {
         spi-max-frequency = <16000000>;
         reg = <0>;
         compatible = "rohm,dh2228fv";
     };
};
```

- Recompile the kernel using the steps given on User Guide - Processor SDK AM62x. While following steps on this page, customize the kernel by using *menuconfig* as per the Kernel Configuration section provided on SPI Kernel Driver. For more details, see the steps below.

```
HOST$ cd <psdk-installaion-path>/board-support/ti-linux-kernel/
HOST$ make defconfig ti_arm64_prone.config
HOST$ make ARCH=arm64 menuconfig
      Device Drivers --->
          [*] SPI support
              <*> User mode SPI device driver support
#Save these changes to the .config file
```


Copyright © 2026 Texas Instruments Incorporated

```
HOST$ make Image dtbs modules
HOST$ sudo cp ./arch/arm64/boot/Image /media/<USER>/root/boot/
HOST$ sudo cp ./arch/arm64/boot/dts/ti/k3-am625-sk.dtb /media/root/boot/dtb/ti
HOST$ sudo -E env “PATH=$PATH” INSTALL_MOD_PATH=/media/<USER>/root make modules_install
HOST$ sync; sync
```

- Copy the target C file to the SDK path and compile the C project file using the method given on Compiling Example Hello World Program.
- Insert the SD card back into SK-AM62x and reboot the device.

3.2.2 M4F Core

- Perform the setup of CCS (Code Composer Studio) for AM62x documented here: Getting Started Steps for AM62x. Make sure to select “MSPM0 32-bit Arm Cortex-M0+ General Purpose MCUs” in the “Select Components” window during CCS installation.
- Import the CCS projects into the Project Explorer (File > Import > Code Composer Studio > CCS Projects). Find the CCS projects in this path: *<Beyond-SDK-installation-path>/Beyond-SDK/collaterals/appnotes/slaaej0-MSPM0-ADC-Attach/am62x/<x_Byte_x_Channel_SPI>/Controller/AM62x-M4F_Core_MCU_Domain/*
- For more general CCS support, see the CCS User’s Guide.

3.3 AM62L Software Setup

3.3.1 A53 Core

- Follow the steps provided on AM62L Getting Started Guide.
- Processor SDK Linux for AM62L Version 11.0.15 was used in the experiment below.
- Setup the SPI driver in Linux by applying the following patch in ti-linux-kernel repo within AM62L SDK.
- Note: The HDMI is disabled in the patch provided below. This is because, on the TMDS62LEVM, the HDMI is muxed with the Expansion header and only one of them can be used at a time.

```
diff --git a/arch/arm64/boot/dts/ti/k3-am62l3-evm.dts b/arch/arm64/boot/dts/ti/k3-am62l3-evm.dts
index 2dd056ce0538..6cf837274c6e 100644
--- a/arch/arm64/boot/dts/ti/k3-am62l3-evm.dts
+++ b/arch/arm64/boot/dts/ti/k3-am62l3-evm.dts
@@ -36,6 +36,7 @@ memory@80000000 {

     hdmi0: connector-hdmi {
         compatible = "hdmi-connector";
+        status = "disabled";
         label = "hdmi";
         type = "a";

@@ -389,6 +390,15 @@ AM62PX_IOPAD(0x0188, PIN_INPUT, 0) /* (A9) MCASP0_AXR1 */
         >;
     };

+    main_spi1_pins_default: main-spi1-pins-default {
+        pinctrl-single,pins = <
+            AM62LX_IOPAD(0x008c, PIN_OUTPUT, 4) /* (H22) SPI1_CLK */
+            AM62LX_IOPAD(0x0080, PIN_INPUT, 4) /* (K22) SPI1_D0 */
+            AM62LX_IOPAD(0x0084, PIN_OUTPUT, 4) /* (J23) SPI1_D1 */
+            AM62LX_IOPAD(0x0088, PIN_OUTPUT, 4) /* (K23) SPI1_CS0 */
+        >;
+    };
+
     pmic_irq_pins_default: pmic-irq-default-pins {
         pinctrl-single,pins = <
             AM62LX_IOPAD(0x01e8, PIN_INPUT, 0) /* (C8) EXTINTn */
@@ -410,6 +420,17 @@ &main_uart0 {
     bootph-all;
 };

+&main_spi1 {
+    status = "okay";
+    pinctrl-names = "default";
+    pinctrl-0 = <&main_spi1_pins_default>;
+    spidev@0 {
+        spi-max-frequency = <24000000>;
```


Copyright © 2026 Texas Instruments Incorporated

```
+        reg = <0>;
+        compatible = "rohm,dh2228fv";
+    };
+};
+
 &main_uart1 {
     pinctrl-names = "default";
     pinctrl-0 = <&main_uart1_pins_default>;
@@ -488,11 +509,14 @@ exp2: gpio@23 {

     sii9022: bridge-hdmi@3b {
         compatible = "sil,sii9022";
+        status="disabled";
         reg = <0x3b>;
         interrupt-parent = <&exp1>;
         interrupts = <16 IRQ_TYPE_EDGE_FALLING>;
         #sound-dai-cells = <0>;
         sil,i2s-data-lanes = < 0 >;
+        pinctrl-names = "default";
+        pinctrl-0 = <&main_dpi_pins_default>;
         bootph-all;

         ports {
@@ -774,8 +798,6 @@ partition@7fe0000 {

 &dss {
     status = "okay";
-    pinctrl-names = "default";
-    pinctrl-0 = <&main_dpi_pins_default>;
     bootph-all;
 };

diff --git a/arch/arm64/configs/defconfig b/arch/arm64/configs/defconfig
index 94dfc265a61c..157d0d62fb53 100644
--- a/arch/arm64/configs/defconfig
+++ b/arch/arm64/configs/defconfig
@@ -1828,3 +1828,5 @@ CONFIG_CORESIGHT_STM=m
 CONFIG_CORESIGHT_CPU_DEBUG=m
 CONFIG_CORESIGHT_CTI=m
 CONFIG_MEMTEST=y
+CONFIG_UHID=y
+CONFIG_SPI_SPIDEV=y
\ No newline at end of file
```

- Rebuild Linux and install Image, dtbs and modules on SD/eMMC.
- Use and compile the C file linked here in Linux Userspace
- For the Multi Byte Example, there are 2 methods of receiving the data at the controller's end. One method is to send the data in 2 clocks - 8 bit each from the MSPM0. The other option is to send the entire data in a single clock burst of double duration. Both the methods have been tested and can be implemented from the C files linked here.
- **Note:** In case of difficulty while running the SPI script in Linux Userspace for higher SPI frequencies, connect the Ground pins of the MSPM0 and AM62L.

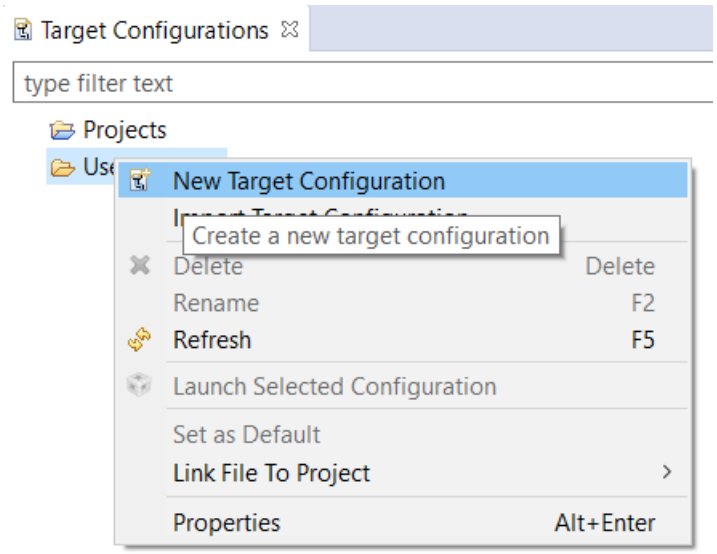
3.4 LP-MSPM0L130x Software Setup

CCS (Code Composer Studio) can be used for development on the MSPM0 Launchpad. Review the CCS User's Guide for general CCS related help. To enable development on MSPM0 devices, select "MSPM0 32-bit Arm Cortex-M0+ General Purpose MCUs" in the "Select Components" window during CCS installation.

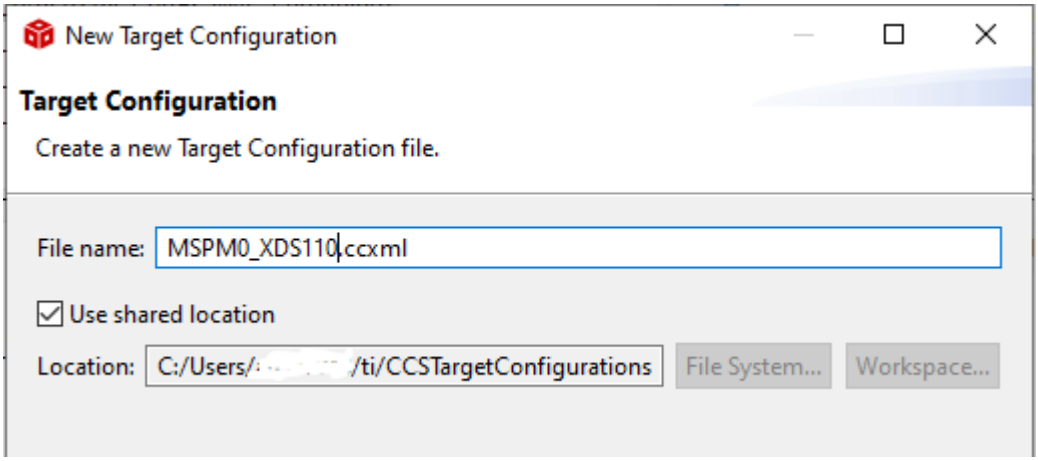
Follow the following steps for adding new target configuration for MSPM0:

- Create a new target configuration

Copyright © 2026 Texas Instruments Incorporated

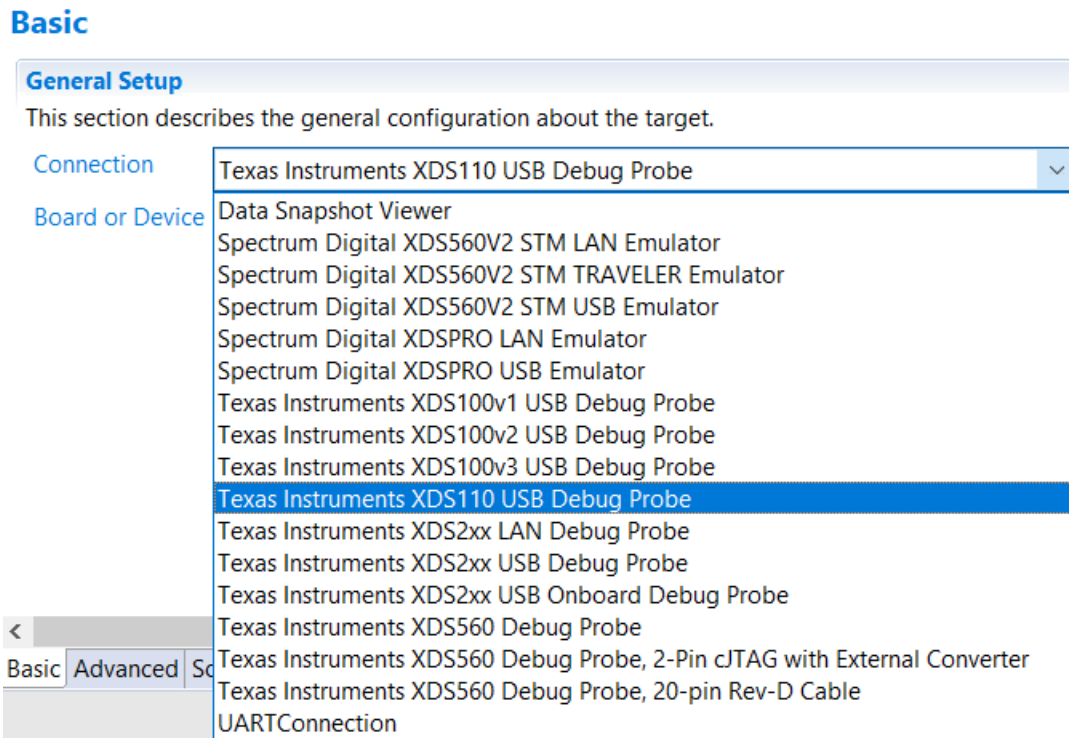


- Give a nice name to the new target configuration, typically {soc name}_{JTAG type}

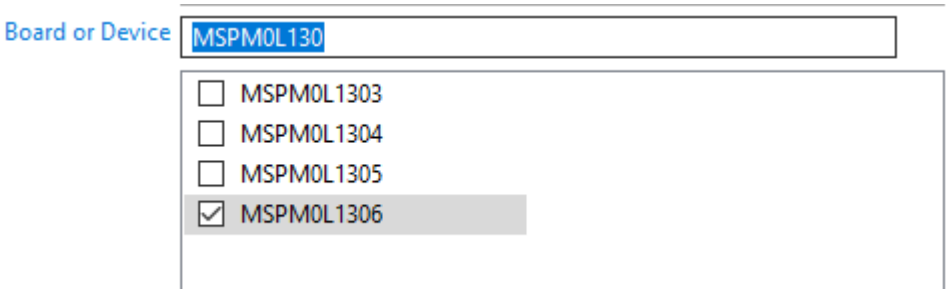


- Select connection as XDS110 USB Debug Probe

Copyright © 2026 Texas Instruments Incorporated



- In "Board or Device" type "MSPM0L130" and select "MSPM0L1306"



- Click "Save" to save the newly created target configuration.

Copyright © 2026 Texas Instruments Incorporated

4 Steps for Execution

This section will discuss the steps to execute the project on both the controller and the peripheral.

4.1 Run Project on LP-MSPM0L130x

After creating the target configurations, the MSPM0 binaries can be built and written to the on-chip flash:

1. Import the CCS project in the Workspace. This is the path to the project:
   a. *<Beyond-SDK-installation-path>/Beyond-SDK/collaterals/appnotes/slaaej0-MSPM0-ADC-Attach/<controller>/<x_Byte_x_Channel_SPI>/Peripheral/MSPM0/*
2. Build the imported CCS project.
   a. If there is a **Duplicate Name of GPIO Pin** error, then do the following:
      i. Open the System Configuration file indicated by .syscfg file extension
      ii. Open ADC12 Configuration
      iii. Go to the Pin Configuration Section and Open ADC12 Channel 7 Pin
      iv. Change the name of the pin from “ti_driverlib_gpio_GPIOPinGeneric0” to “ti_driverlib_gpio_GPIOPinGeneric6”
      v. Save the change and Build the project again
3. The built binary can be found in *<Beyond-SDK-installation-path>/…/MSPM0/Debug/<file-name>.out*
4. Right-click on *MSPM0_XDS110.ccxml* in Target Configurations window
5. Select “Launch Selected Configuration”
6. In the Debug window, click on the "Texas Instruments XDS110 USB Debug Probe_0/CORTEX_M0P"
7. Select Run -> Connect Target
8. Select Run -> Reset -> Subsystem Reset
9. Select Run -> Load -> Load Program
10. Browse to the binary for the MSPM0 project, and click “OK”.
11. This will write the flash with the binary.
12. Select Run -> Resume

Note that on the newer MSPM0 SDK versions, this may give build errors. To resolve this, import a fresh spi_peripheral_echo_interrupts_LP_MSPM0L1306_nortos_ticlang project using Resource Explorer in CCS and then:

1. Copy the spi_peripheral_echo_interrupts.c from Beyond SDK to the fresh project.
2. Copy the sysconfig setting as in Beyond SDK for ADC12, SPI, TIMER, and EVENT in the fresh project.

For additional support with CCS, review the Code Composer Studio User’s Guide.

4.2 Run Project on SK-AM62x/AM62L EVM

This section contains the execution steps depending on what core is being used as the controller. Note that the same A53 Core steps can be followed for the AM62L EVM.

4.2.1 A53 Core

On the serial monitor obtained through AM62x Starter Kit EVM Quick Start Guide and AM62L Quick Start Guide, go to the location of the executable and run it using:

```
root@am62xx-evm:~#./<executable_name> -D <spidriver_name_from_/dev_folder> -s <speed> -v
```

Example:

```
root@am62xx-evm:~#./spidev_adc_multibyte_multichannel -D /dev/spidev1.0 -s 16000000 -v
```


Copyright © 2026 Texas Instruments Incorporated

4.2.2 M4F Core

Write prebuilt SK-AM62x binaries to the on-chip flash:

1. Import the CCS project into a different Workspace than the MSPM0 Workspace
   a. *<Beyond-SDK-installation-path>/Beyond-SDK/collaterals/appnotes/slaaej0-MSPM0-ADC-Attach/am62x/<x_Byte_x_Channel_SPI>/Controller/AM62x-M4F_Core_MCU_Domain/Debug/<file-name>.out*
   b. This allows for two different debugging sessions for each device
2. Build the imported CCS project.
3. Right-click on AM62x_XDS110.ccxml in Target Configurations window
4. Select “Launch Selected Configuration”
5. In the Debug window, click on the "Texas Instruments XDS110 USB Debug Probe_0/ BLAZAR_Cortex_M4F_1"
6. Select Run -> Connect Target
7. Select Run -> Reset -> Subsystem Reset
8. Select Run -> Load -> Load Program
9. Browse to the prebuilt binary for the AM62x project and click “OK”.
10. This will write the flash with the binary.
11. Select Run -> Resume

5 Results

In the example application codes used:

- *Single Byte* refers to 8-bit ADC data transmission.
- *Multi Byte* refers to SPI transmission capable of transmitting as many bytes as configured. In the examples, 2-byte data has been configured. Out of 16 bits transferred, only lower 12-bits contain ADC data since maximum resolution of ADC on MSPM0 is 12 bits.
- *Single Channel* refers that only 1 analog signal is monitored by ADC. Controller has to send a dummy command to initiate transaction, but command value need not be checked at peripheral.
- *Multi Channel* refers that ADC is converting multiple analog signals sequentially. Controller has to send a valid command to initiate transaction and receive corresponding channel data.
- AM62x and AM62L controllers give similar outputs.

Copyright © 2026 Texas Instruments Incorporated

5.1 Single Byte Single Channel

Note that to obtain the following results, we have considered the following analog inputs for 8-bit ADC.

Command 0x00: ADC Channel 7: Sinusoidal Signal (3.3Vpp, 1.65V DC offset, @2Hz)

```
Data =   6
Data =  12
Data =  20
Data =  30
Data =  41
Data =  54
Data =  68
Data =  83
Data =  99
Data = 116
Data = 131
Data = 149
Data = 165
Data = 181
Data = 196
Data = 210
Data = 222
Data = 233
Data = 241
Data = 248
Data = 253
Data = 255
Data = 255
Data = 255
Data = 252
Data = 246
Data = 239
Data = 230
Data = 219
Data = 206
Data = 192
Data = 178
Data = 162
Data = 145
Data = 129
Data = 112
Data =  96
Data =  80
Data =  66
Data =  52
Data =  39
Data =  28
Data =  18
Data =  11
Data =   5
Data =   1
Data =   0
Data =   1
Data =   3
Data =   8
Data =  15
```

Figure 5-1. Single Byte Single Channel Sinusoidal Wave Results

Copyright © 2026 Texas Instruments Incorporated

Command 0x00: ADC Channel 7: Square Wave Signal (3.3Vpp, 1.65V DC offset, @2Hz, 50%duty)

```
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
```

Figure 5-2. Single Byte Single Channel Square Wave Results

Copyright © 2026 Texas Instruments Incorporated

5.2 Single Byte Multi Channel

Note that to obtain the following results, we have considered the following analog inputs for 8-bit ADC:

Command 0x00: ADC Channel 7: Sinusoidal Signal (3.3Vpp, 1.65V DC offset, @2Hz)

Command 0x01: ADC Channel 8: DC Signal (3.3V)

```
CommandID =  0, Data = 102          CommandID =  1, Data = 255
CommandID =  0, Data =  72          CommandID =  1, Data = 255
CommandID =  0, Data =  44          CommandID =  1, Data = 255
CommandID =  0, Data =  23          CommandID =  1, Data = 255
CommandID =  0, Data =   8          CommandID =  1, Data = 255
CommandID =  0, Data =   0          CommandID =  1, Data = 255
CommandID =  0, Data =   2          CommandID =  1, Data = 255
CommandID =  0, Data =  11          CommandID =  1, Data = 255
CommandID =  0, Data =  29          CommandID =  1, Data = 255
CommandID =  0, Data =  52          CommandID =  1, Data = 255
CommandID =  0, Data =  81          CommandID =  1, Data = 255
CommandID =  0, Data = 112          CommandID =  1, Data = 255
CommandID =  0, Data = 146          CommandID =  1, Data = 255
CommandID =  0, Data = 177          CommandID =  1, Data = 255
CommandID =  0, Data = 206          CommandID =  1, Data = 255
CommandID =  0, Data = 229          CommandID =  1, Data = 255
CommandID =  0, Data = 246          CommandID =  1, Data = 255
CommandID =  0, Data = 255          CommandID =  1, Data = 255
CommandID =  0, Data = 255          CommandID =  1, Data = 255
CommandID =  0, Data = 248          CommandID =  1, Data = 255
CommandID =  0, Data = 233          CommandID =  1, Data = 255
CommandID =  0, Data = 210          CommandID =  1, Data = 255
CommandID =  0, Data = 183          CommandID =  1, Data = 255
CommandID =  0, Data = 151          CommandID =  1, Data = 255
CommandID =  0, Data = 119          CommandID =  1, Data = 255
CommandID =  0, Data =  87          CommandID =  1, Data = 255
CommandID =  0, Data =  57          CommandID =  1, Data = 255
CommandID =  0, Data =  33          CommandID =  1, Data = 255
CommandID =  0, Data =  14          CommandID =  1, Data = 255
CommandID =  0, Data =   3          CommandID =  1, Data = 255
CommandID =  0, Data =   0          CommandID =  1, Data = 255
CommandID =  0, Data =   6          CommandID =  1, Data = 255
CommandID =  0, Data =  19          CommandID =  1, Data = 255
CommandID =  0, Data =  40          CommandID =  1, Data = 255
CommandID =  0, Data =  66          CommandID =  1, Data = 255
CommandID =  0, Data =  97          CommandID =  1, Data = 255
CommandID =  0, Data = 129          CommandID =  1, Data = 255
CommandID =  0, Data = 162          CommandID =  1, Data = 255
CommandID =  0, Data = 192          CommandID =  1, Data = 255
CommandID =  0, Data = 218          CommandID =  1, Data = 255
CommandID =  0, Data = 238          CommandID =  1, Data = 255
CommandID =  0, Data = 251          CommandID =  1, Data = 255
CommandID =  0, Data = 255          CommandID =  1, Data = 255
CommandID =  0, Data = 253          CommandID =  1, Data = 255
CommandID =  0, Data = 241          CommandID =  1, Data = 255
CommandID =  0, Data = 222          CommandID =  1, Data = 255
CommandID =  0, Data = 197          CommandID =  1, Data = 255
```

Figure 5-3. Single Byte Multi Channel Results

Copyright © 2026 Texas Instruments Incorporated

5.3 Multi Byte Single Channel

Note that to obtain the following results, we have considered the following analog inputs for 12-bit ADC.

Command 0x00: ADC Channel 7: Sinusoidal Signal (3.3Vpp, 1.65V DC offset, @2Hz)

```
Data = 3879
Data = 4061
Data = 4095
Data = 4021
Data = 3810
Data = 3487
Data = 3070
Data = 2587
Data = 2062
Data = 1540
Data = 1065
Data =  641
Data =  307
Data =   97
Data =    6
Data =   52
Data =  232
Data =  527
Data =  915
Data = 1381
Data = 1896
Data = 2422
Data = 2915
Data = 3356
Data = 3719
Data = 3970
Data = 4095
Data = 4086
Data = 3953
Data = 3694
Data = 3326
Data = 2879
Data = 2377
Data = 1857
Data = 1338
Data =  884
Data =  497
Data =  209
Data =   43
Data =   13
Data =  111
Data =  346
Data =  683
Data = 1113
Data = 1603
Data = 2120
Data = 2639
Data = 3116
Data = 3527
```

Figure 5-4. Multi Byte Single Channel Sinusoidal Wave Results

Copyright © 2026 Texas Instruments Incorporated

Command 0x00: ADC Channel 7: Square Wave Signal (3.3Vpp, 1.65V DC offset, @2Hz, 50%duty)

```
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data =    8
Data =    9
Data =    8
Data =    7
Data =    6
Data =    6
Data =    7
Data =    8
Data =    7
Data =    3
Data =    8
Data =    6
Data =    0
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data =   13
Data =    9
Data =   13
Data =    9
Data =   12
Data =    7
Data =    8
Data =    7
Data =    7
Data =    0
Data =    4
Data =   11
Data = 4095
```

Figure 5-5. Multi Byte Single Channel Square Wave Results

Copyright © 2026 Texas Instruments Incorporated

5.4 Multi Byte Multi Channel

Note that to obtain the following results, we have considered the following analog inputs for 12-bit ADC:

Command 0x00: ADC Channel 7: Sinusoidal Signal (3.3Vpp, 1.65V DC offset, @2Hz)

Command 0x01: ADC Channel 8: DC Signal (3.3V)

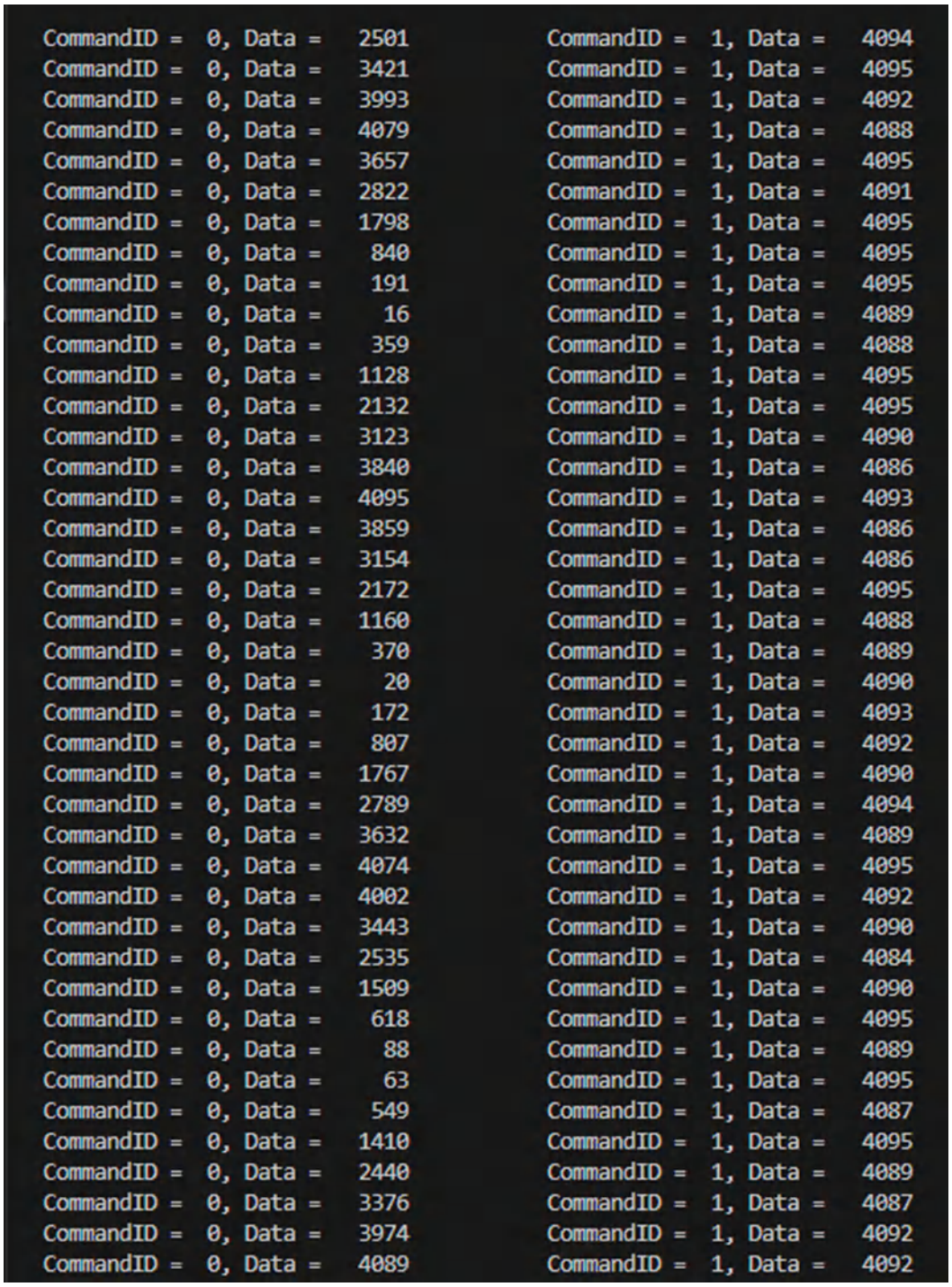


Figure 5-6. Multi Byte Multi Channel Results

Copyright © 2026 Texas Instruments Incorporated

6 Summary

The AM62x and AM62L is an ideal option for a wide range of embedded applications. Most of the embedded applications require to collect real world analog signals from sensors. This document presents steps followed to integrate a 8-bit and a 12-bit ADC using the MSM0 to the AM62x and AM62L processors. The integration addresses a critical limitation where the AM62x processor lacks any onboard ADC capability, while the AM62L processor has a ~10 ENOB ADC that may require higher 12-bit resolution for certain applications.

The solution architecture positions the AM62x/AM62L processors as the SPI controller (using either A53 or M4F cores) and the MSPM0L130x microcontroller as the peripheral device. Communication occurs via full-duplex SPI at speeds up to 50 MHz for the controller but limited to 16Mbits/s for the MSPM0L Peripheral. The system supports multiple operating modes including single-byte or multi-byte transfers combined with single-channel or multi-channel configurations, allowing monitoring of multiple analog inputs simultaneously. ADC sampling is timer-triggered with continuous data updates to maintain real-time performance.

The hardware setup requires either an SK-AM62x starter kit or AM62L EVM connected to an LP-MSPM0L1306 LaunchPad via SPI connections through expansion headers, along with analog signal sources for testing. Software implementation involves Linux kernel modifications for SPI driver support, device tree configuration for proper pin-muxing, CCS projects for both the MSPM0 peripheral and AM62x M4F core, and compiled executables for A53 core Linux userspace execution. The solution was successfully validated using sinusoidal and square wave test inputs (3.3Vpp at 2Hz frequency) for both the Controllers - SK-AM62x and AM62L, demonstrating accurate ADC data capture across all configuration modes with reliable performance at the maximum supported SPI speed.

Copyright © 2026 Texas Instruments Incorporated

7 References

1. Texas Instruments, *AM625*, product page.
2. Texas Instruments, *AM62L*
3. Texas Instruments, *MSPM0L1306*, product page.
4. Texas Instruments, *AM625 Sitara Processor*, datasheet.
5. Texas Instruments, *AM62Lx Sitara Processors*, datasheet.
6. Texas Instruments, *MSPM0L130x Mixed-Signal Microcontrollers*, datasheet.
7. Texas Instruments, *SK-AM62 User's Guide*, user's guide.
8. Texas Instruments, *AM62L User's Guide* user's guide.
9. Texas Instruments,*LP-MSPM0L1306 User's Guide*, user's guide.
10. Texas Instruments, *SK-AM62x Quick Start Guide*, quick start guide.
11. Texas Instruments, *AM62L Quick Start Guide*, quick start guide.
12. Texas Instruments, *MSPM0 Quick Start Guide*, quick start guide.
13. Texas Instruments, *Kernel: Foundational Components: Processors SDK Linux AM62x*, user's guide.
14. Texas Instruments, *Kernel Foundational Components: Processors SDK Linux AM62L*, user's guide.
15. Texas Instruments, *Processor SDK Linux for AM62x*, webpage.
16. Texas Instruments, *Linux SDK for AM62L*, webpage.
17. Texas Instruments, *Compiling "Hello World" Program*, webpage.
18. Texas Instruments, *SK-AM62x Overview Getting Started Guide*, user's guide.
19. Texas Instruments, *AM62L Overview Getting Started Guide*, user's guide.
20. Texas Instruments, *MSPM0L1306 LaunchPad Development Kit*, user's guide.

8 Revision History

Changes from Revision * (November 2023) to Revision A (April 2026)	Page
• Updated the abstract after adding AM62L details	1
• Added AM62L Processor figure	3
• Added a basic introduction for AM62x and AM62L hardware setup, and information on the Hardware setup for AM62L with MSPM0 for SPI communication	6
• Updated the path for files and steps for cloning the github repository, updated link for compiling Hello World Example in AM62x A53 core section, added details for AM62L software setup and provided the patch for enabling SPI on AM62L and disabling the HDMI	9
• Updated path for project and steps for building the binary for running project on MSPM0, added link for AM62L quick start guide for A53 core and updated the CCS project path from Github for M4F core	14
• Updated with details based on AM62L addition	22
• Added required AM62L links and updated broken AM62x links	23

Copyright © 2026 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, regulatory 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 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, TI’s General Quality Guidelines, 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. Unless TI explicitly designates a product as custom or customer-specified, TI products are standard, catalog, general purpose devices.

TI objects to and rejects any additional or different terms you may propose.

Copyright © 2026, Texas Instruments Incorporated

Last updated 10/2025