



Anshu Madwesh, Divyansh Mittal, Anil Swargam, Anshu Choudhary, and Krunal Bhargav.

## ABSTRACT

This application note describes how we can integrate ADC present on MSMP0 into AM62x with the help 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. AM62x does not come with an on-board ADC and hence this paper aims to demonstrate how we can integrate MSMP0 Microcontroller's ADC into AM62x. 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 AM62x SoC via SPI. This paper will further delve into overall data flow, hardware and software setup, steps to execute application code and the expected results.

## Table of Contents

<b>1 Introduction</b> .....	2
1.1 SPI Transaction Dataflow.....	2
1.2 AM62x Processor.....	2
1.3 MSPM0L130x Microcontroller.....	3
<b>2 Hardware Setup</b> .....	5
2.1 A53 Core Hardware Setup.....	5
2.2 M4F Core Hardware Setup.....	6
<b>3 Software Setup</b> .....	7
3.1 Cloning the Beyond SDK GitHub Repository.....	7
3.2 SK-AM62x Software Setup.....	7
3.3 LP-MSPM0L130x Software Setup.....	8
<b>4 Steps for Execution</b> .....	10
4.1 Run Project on LP-MSPM0L130x.....	10
4.2 Run Project on SK-AM62x.....	10
<b>5 Results</b> .....	12
5.1 Single Byte Single Channel.....	13
5.2 Single Byte Multi Channel.....	15
5.3 Multi Byte Single Channel.....	16
5.4 Multi Byte Multi Channel.....	18
<b>6 Summary</b> .....	19
<b>7 References</b> .....	20

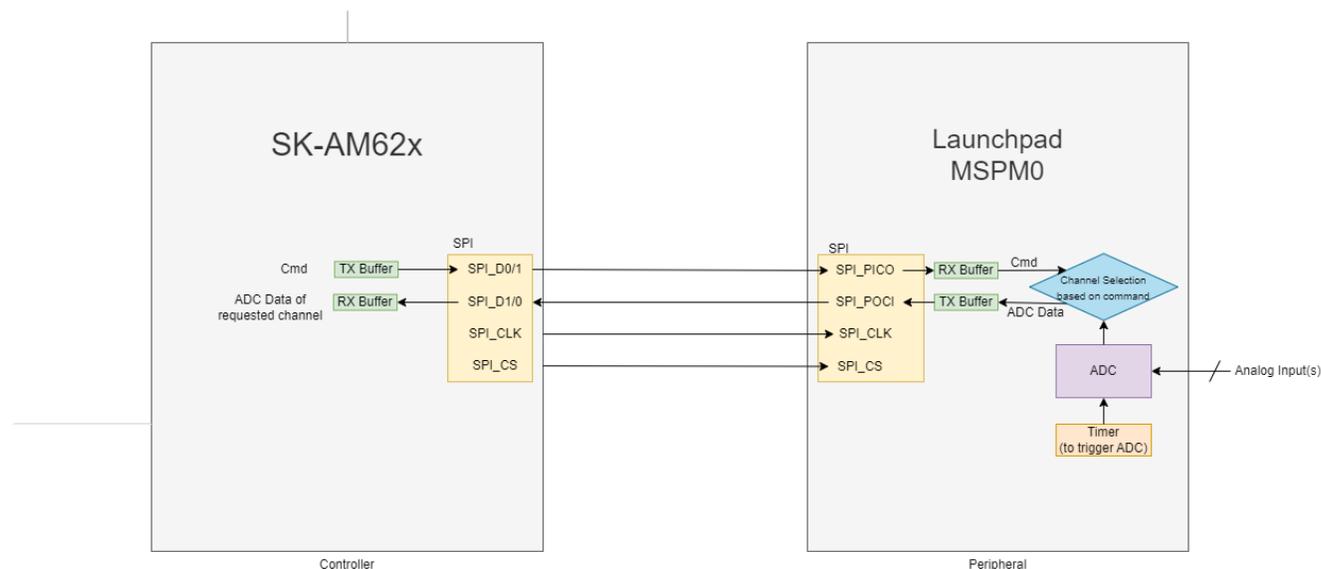
## Trademarks

All trademarks are the property of their respective owners.

## 1 Introduction

### 1.1 SPI Transaction Dataflow

We configure the ADC present on MSPM0L130x Microcontroller and establish an SPI interface with AM62x microprocessor Starter-Kit. Here, AM62x has been 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.<sup>1</sup>



**Figure 1-1. Overall Dataflow Between Controller (SK-AM62x) and Peripheral (LP-MSPM0L130x)**

#### 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 SK-AM62x and 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.

### 1.2 AM62x Processor

#### 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.

<sup>1</sup> 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.

For more details, see [AM62x Sitara Processors Data Sheet](#).

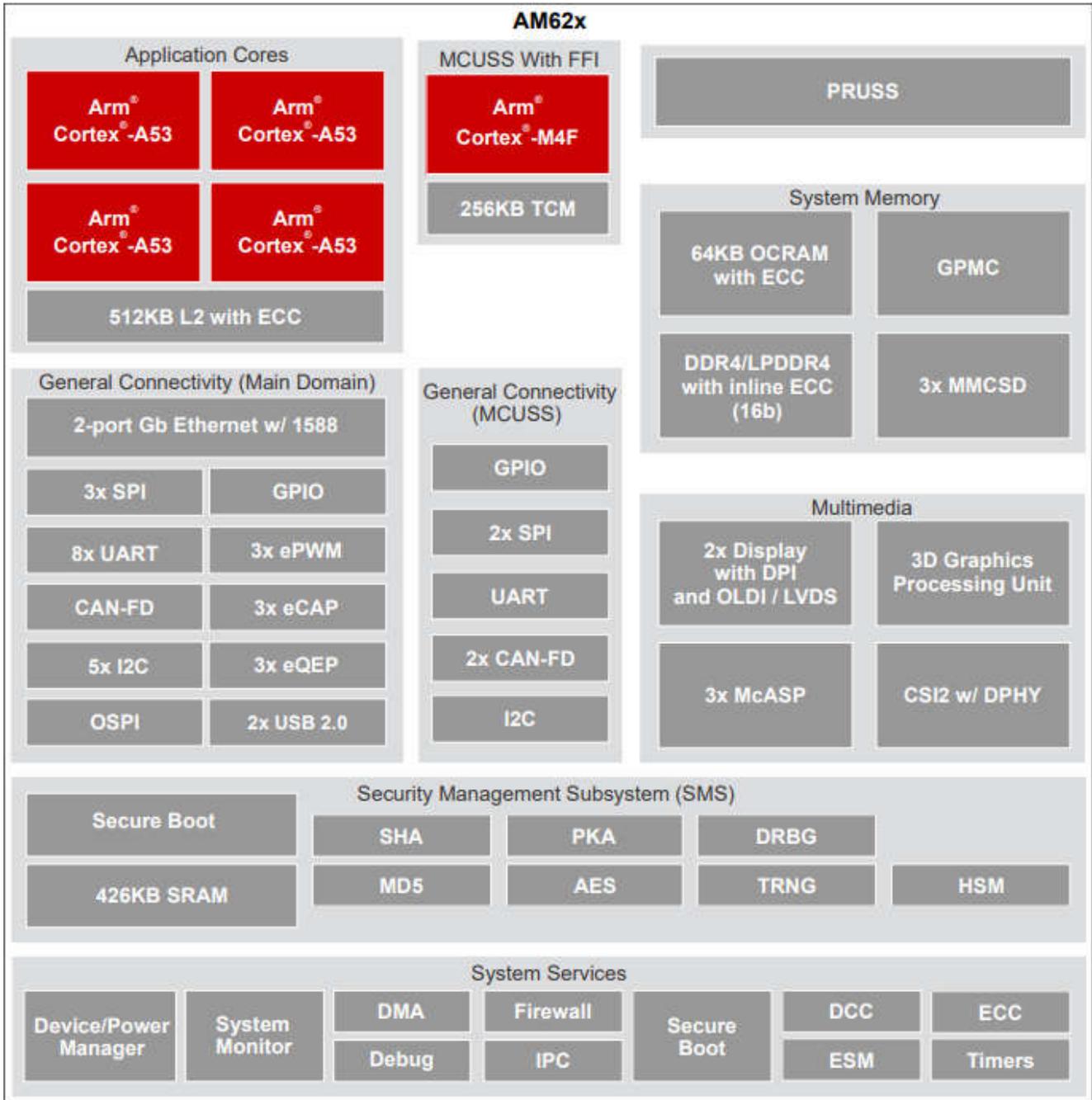


Figure 1-2. AM62x Simplified Block Diagram

### 1.3 MSPM0L130x Microcontroller

The [MSPM0L130x Microcontroller](#), shown in Figure 1-3, is an easy-to-use evaluation module (EVM).

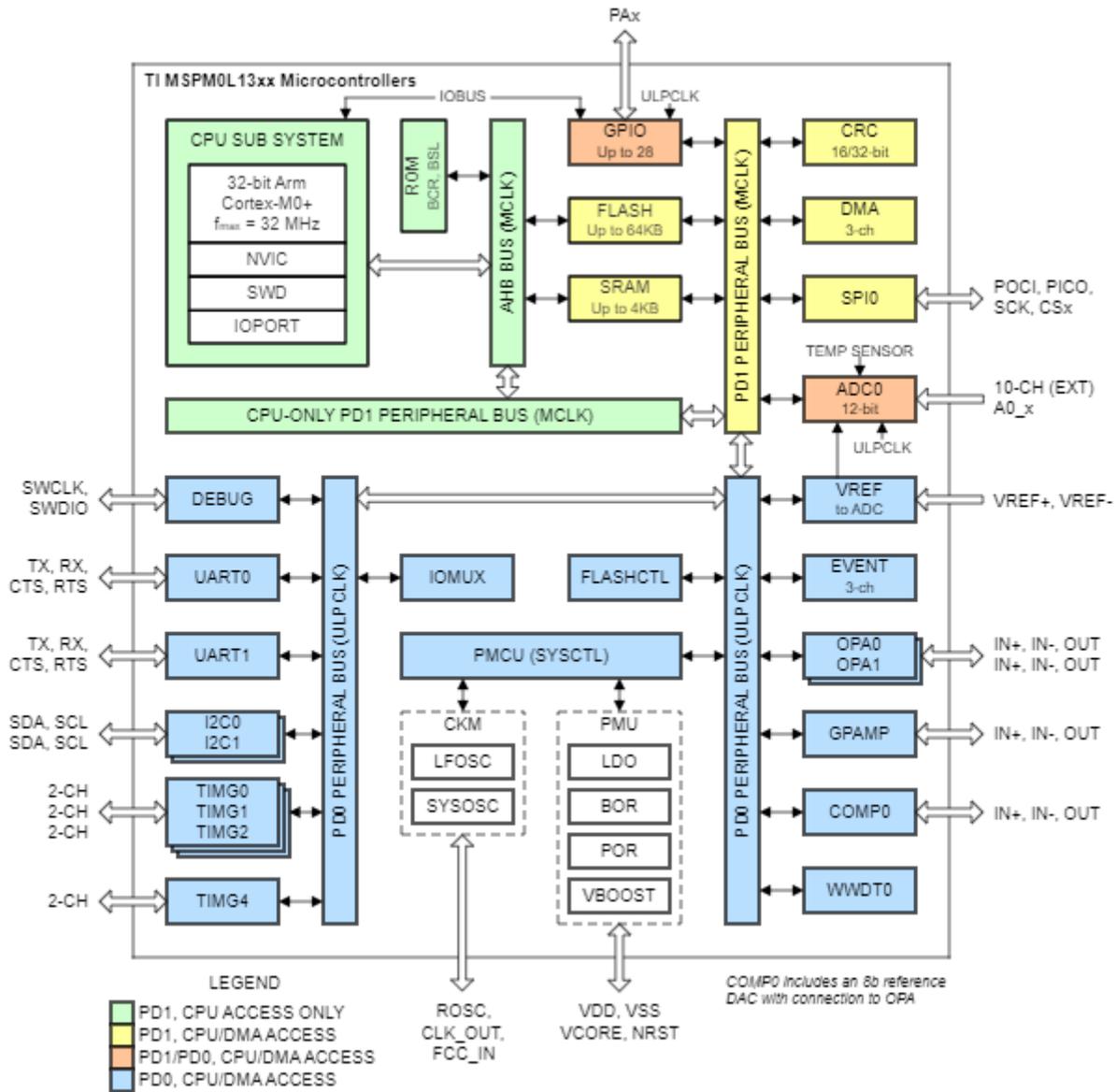


Figure 1-3. MSPM0L130x Simplified Block Diagram

The main compute and interface subsystems from a machine vision context in AM62x 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.

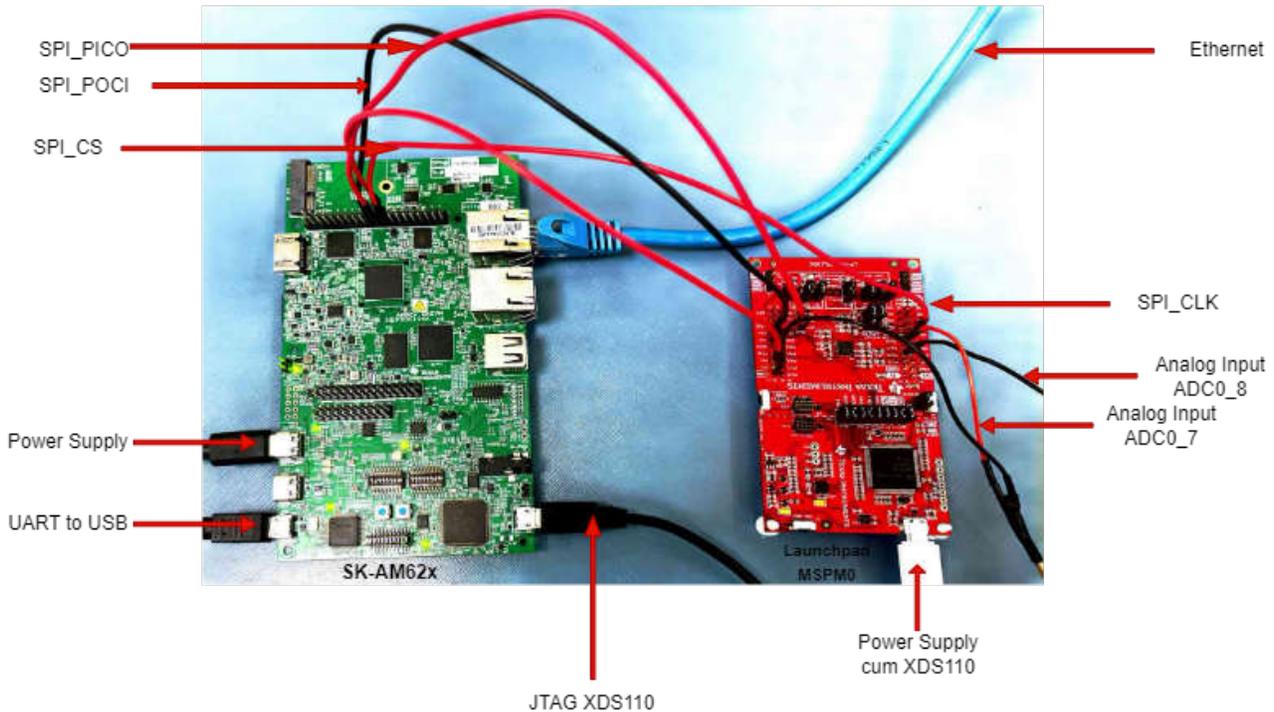
For more details, see [MSPM0L130x Microcontroller Data Sheet](#).

## 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 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.

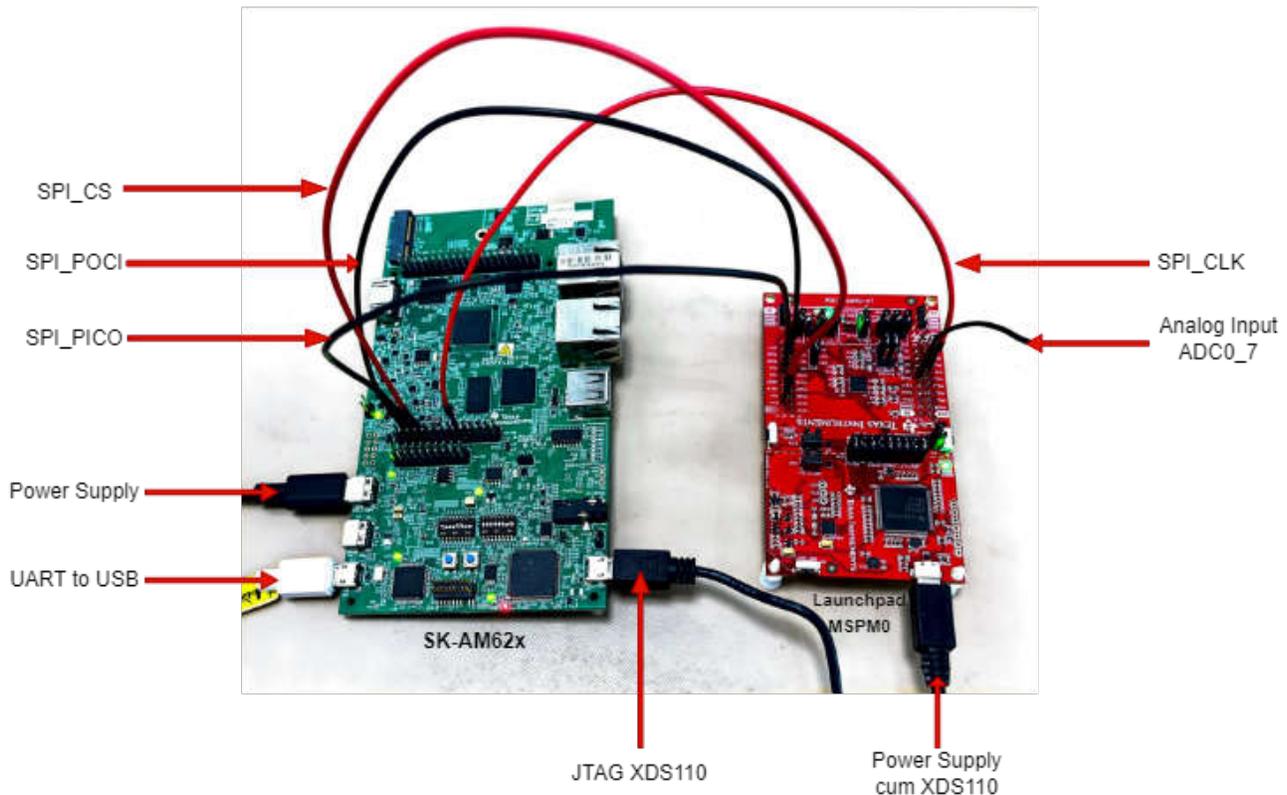


**Figure 2-1. Cable Connections between SK-AM62x (A53 Core) and LP-MSPM0L1306 for SPI communication.**

- For SK-AM62x:
  - Connect type-C power supply with 5V adapter.
  - 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 inter-board 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.
  - 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.

## 2.2 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 with 5V adapter.
  - 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 inter-board 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.

## 3 Software Setup

Here are the steps for setting up the AM62x depending on what core will be used 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/am62x/MSPM0-ADC-RTC-Attach/MSPM0-ADC-Attach-SPI*
- Select one of the folders depending on the application (*x\_Byte\_x\_Channel\_SPI*)
- In 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-installation-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
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 any method given on [Compiling Example Hello World Program](#). Using either method, you should have an executable file loaded into the SD card in the end. For more details, see the steps below.

```
HOST$ cd <Beyond-SDK-installation-path>/Beyond-SDK/am62x/MSPM0-ADC-RTC-Attach/MSPM0-ADC-Attach-SPI/
<x_Byte_x_Channel_SPI>/Controller/AM62x-A53_Core_MAIN_Domain/
HOST$ cp <target-filename>.c <psdk-installation-path>/linux-devkit/
HOST$ cd <psdk-installation-path>/linux-devkit/
HOST$ source environment-setup
HOST$ ${CC} <target-filename>.c -o <output-filename>
HOST$ sudo cp <output-filename> /media/<USER>/root/home/root
HOST$ exit
```

- Insert the SD card back into SK-AM62x and reboot the device.

### 3.2.2 M4F Core

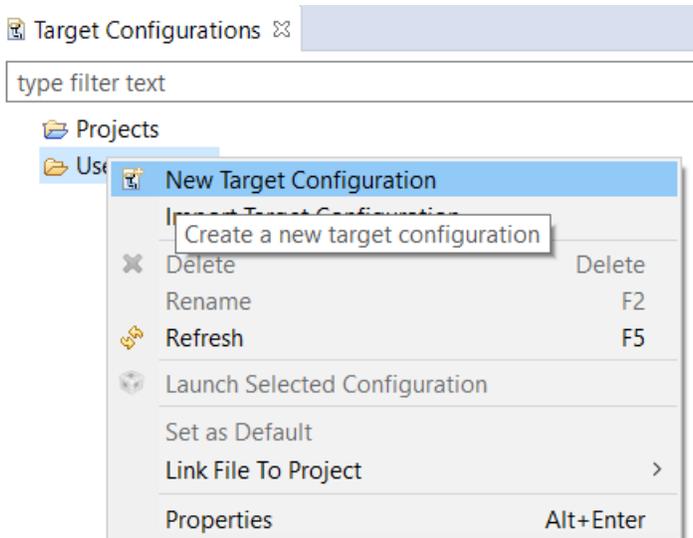
- Perform the setup of CCS ([Code Composer Studio](#)) for AM62x documented here: [Getting Started Steps for AM62x](#). Make sure to select “MSPM0 32-bit Arm Cortex-M0+ General Purpose MCUs” in the “Select Components” window during CCS installation.
- Import the CCS projects into the Project Explorer (File > Import > Code Composer Studio > CCS Projects). Find the CCS projects in this path: *<Beyond-SDK-installation-path>/Beyond-SDK/am62x/MSPM0-ADC-RTC-Attach/MSPM0-ADC-Attach-SPI/<x\_Byte\_x\_Channel\_SPI>/Controller/AM62x-M4F\_Core\_MCU\_Domain/*
- For more general CCS support, see the [CCS User’s Guide](#).

### 3.3 LP-MSPM0L130x Software Setup

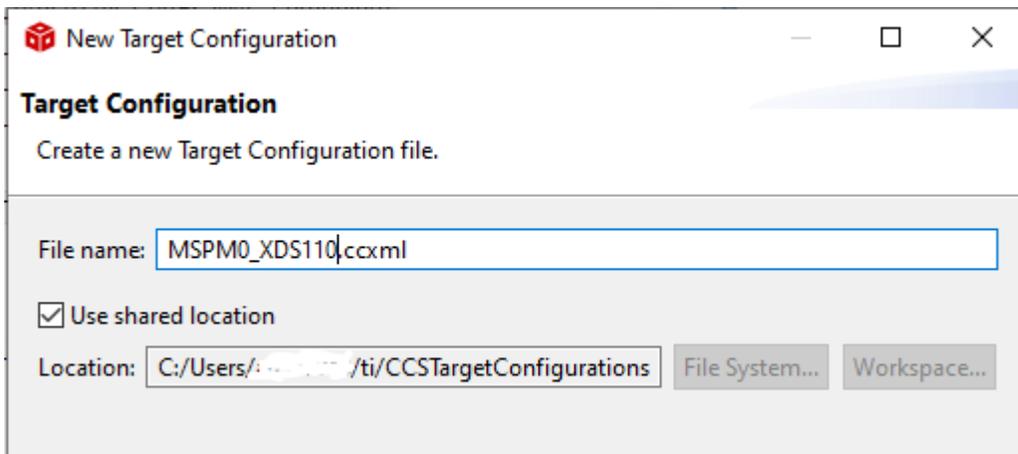
CCS ([Code Composer Studio](#)) can be used for development on the MSPM0 Launchpad. Review the [CCS User’s Guide](#) for general CCS related help. To enable development on MSPM0 devices, select “MSPM0 32-bit Arm Cortex-M0+ General Purpose MCUs” in the “Select Components” window during CCS installation.

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

- Create a new target configuration

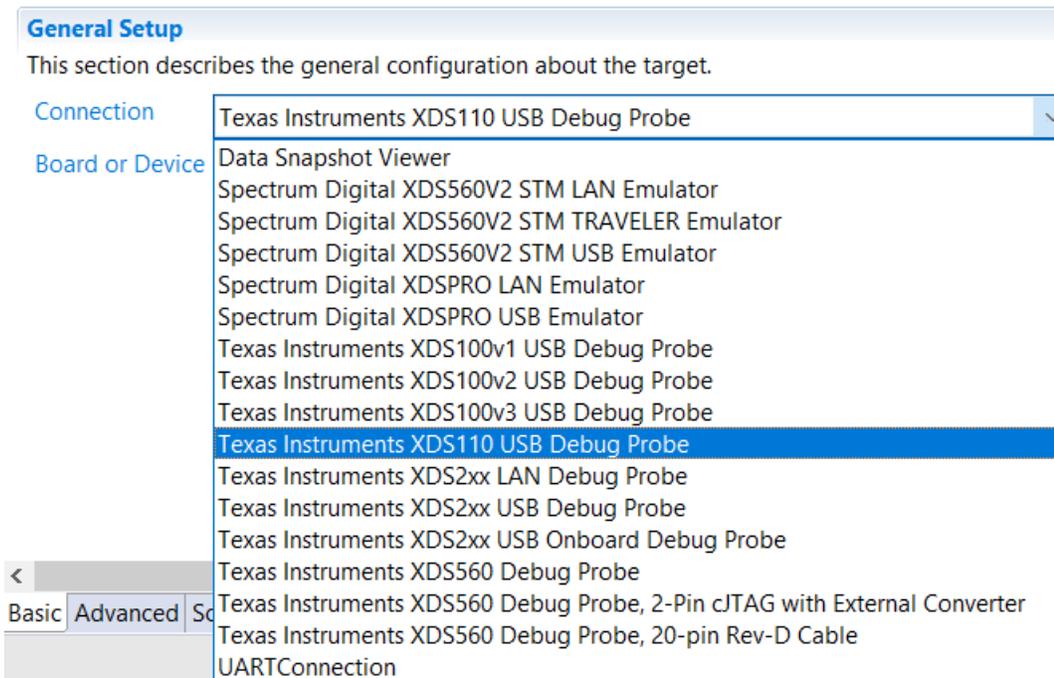


- Give a nice name to the new target configuration, typically {soc name}\_{JTAG type}

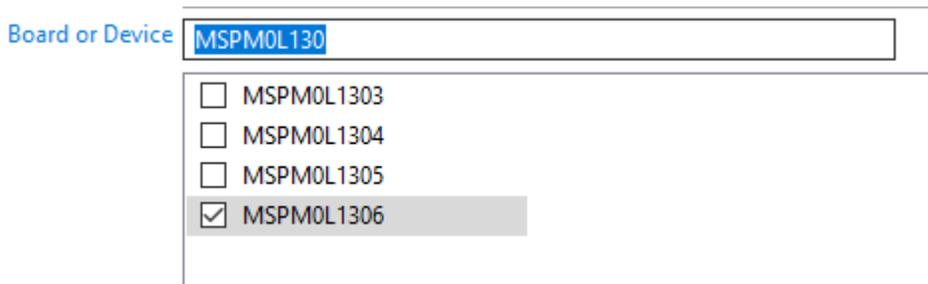


- Select connection as XDS110 USB Debug Probe

## Basic



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



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

## 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 prebuilt MSPM0 binaries can be 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/am62x/MSPM0-ADC-RTC-Attach/MSPM0-ADC-Attach-SPI/<x_Byte_x_Channel_SPI>/Peripheral/MSPM0/`
  - b. The binary can be found in `<Beyond-SDK-installation-path>/.../MSPM0/Debug/<file-name>.out`
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. Right-click on `MSPM0_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/CORTEX\_M0P"
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 MSPM0 project, and click "OK".
10. This will write the flash with the binary.
11. Select Run -> Resume

For additional support with CCS, review the [Code Composer Studio User's Guide](#).

### 4.2 Run Project on SK-AM62x

This section contains the execution steps depending on what core is being used as the controller.

#### 4.2.1 A53 Core

On the serial monitor obtained through [AM62x Starter Kit EVM 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
```

#### 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/am62x/MSPM0-ADC-RTC-Attach/MSPM0-ADC-Attach-SPI/<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.

## 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 = 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**



## 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

### 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**

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

```
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 = 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**

## 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)

CommandID = 0, Data = 2501	CommandID = 1, Data = 4094
CommandID = 0, Data = 3421	CommandID = 1, Data = 4095
CommandID = 0, Data = 3993	CommandID = 1, Data = 4092
CommandID = 0, Data = 4079	CommandID = 1, Data = 4088
CommandID = 0, Data = 3657	CommandID = 1, Data = 4095
CommandID = 0, Data = 2822	CommandID = 1, Data = 4091
CommandID = 0, Data = 1798	CommandID = 1, Data = 4095
CommandID = 0, Data = 840	CommandID = 1, Data = 4095
CommandID = 0, Data = 191	CommandID = 1, Data = 4095
CommandID = 0, Data = 16	CommandID = 1, Data = 4089
CommandID = 0, Data = 359	CommandID = 1, Data = 4088
CommandID = 0, Data = 1128	CommandID = 1, Data = 4095
CommandID = 0, Data = 2132	CommandID = 1, Data = 4095
CommandID = 0, Data = 3123	CommandID = 1, Data = 4090
CommandID = 0, Data = 3840	CommandID = 1, Data = 4086
CommandID = 0, Data = 4095	CommandID = 1, Data = 4093
CommandID = 0, Data = 3859	CommandID = 1, Data = 4086
CommandID = 0, Data = 3154	CommandID = 1, Data = 4086
CommandID = 0, Data = 2172	CommandID = 1, Data = 4095
CommandID = 0, Data = 1160	CommandID = 1, Data = 4088
CommandID = 0, Data = 370	CommandID = 1, Data = 4089
CommandID = 0, Data = 20	CommandID = 1, Data = 4090
CommandID = 0, Data = 172	CommandID = 1, Data = 4093
CommandID = 0, Data = 807	CommandID = 1, Data = 4092
CommandID = 0, Data = 1767	CommandID = 1, Data = 4090
CommandID = 0, Data = 2789	CommandID = 1, Data = 4094
CommandID = 0, Data = 3632	CommandID = 1, Data = 4089
CommandID = 0, Data = 4074	CommandID = 1, Data = 4095
CommandID = 0, Data = 4002	CommandID = 1, Data = 4092
CommandID = 0, Data = 3443	CommandID = 1, Data = 4090
CommandID = 0, Data = 2535	CommandID = 1, Data = 4084
CommandID = 0, Data = 1509	CommandID = 1, Data = 4090
CommandID = 0, Data = 618	CommandID = 1, Data = 4095
CommandID = 0, Data = 88	CommandID = 1, Data = 4089
CommandID = 0, Data = 63	CommandID = 1, Data = 4095
CommandID = 0, Data = 549	CommandID = 1, Data = 4087
CommandID = 0, Data = 1410	CommandID = 1, Data = 4095
CommandID = 0, Data = 2440	CommandID = 1, Data = 4089
CommandID = 0, Data = 3376	CommandID = 1, Data = 4087
CommandID = 0, Data = 3974	CommandID = 1, Data = 4092
CommandID = 0, Data = 4089	CommandID = 1, Data = 4092

Figure 5-6. Multi Byte Multi Channel Results

## 6 Summary

The AM62x 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 ADC present onboard on MSM0 into AM62x. The application showed low latency and SPI operation speeds as high as 16 MHz, which the maximum that can be attained through MSPM0L130x.

## 7 References

1. Texas Instruments, “AM625,” [Online]. Available: <https://www.ti.com/product/AM625>
2. Texas Instruments, “MSPM0L1306,” [Online]. Available: <https://www.ti.com/product/MSPM0L1306>
3. Texas Instruments, “AM625 Sitara Processor Data Sheet,” [Online]. Available: <https://www.ti.com/document-viewer/am625/datasheet>
4. Texas Instruments, “AM625 Sitara Processor Data Sheet,” [Online]. Available: <https://www.ti.com/document-viewer/mspm0l1306/datasheet>
5. Texas Instruments, “SK-AM62 User’s Guide,” [Online]. Available: <https://www.ti.com/document-viewer/lit/html/spruj40>
6. Texas Instruments, “LP-MSPM0L1306 User’s Guide,” [Online]. Available: <https://www.ti.com/document-viewer/lit/html/slau869>
7. Texas Instruments, “SK-AM62 Quick Start Guide,” [Online]. Available: [https://dev.ti.com/tirex/content/tirex-product-tree/am62x-devtools/docs/am62x\\_skevm\\_quick\\_start\\_guide.html](https://dev.ti.com/tirex/content/tirex-product-tree/am62x-devtools/docs/am62x_skevm_quick_start_guide.html)
8. Texas Instruments, “Kernel: Foundational Components: Processors SDK Linux AM62x,” [Online]. Available: [https://software-dl.ti.com/processor-sdk-linux/esd/AM62X/09\\_00\\_00\\_03/exports/docs/linux/Foundational\\_Components\\_Kernel\\_Users\\_Guide.html](https://software-dl.ti.com/processor-sdk-linux/esd/AM62X/09_00_00_03/exports/docs/linux/Foundational_Components_Kernel_Users_Guide.html)
9. Texas Instruments, “SPI Kernel Driver: Foundational Components: Processors SDK Linux AM62x,” [Online]. Available: [https://software-dl.ti.com/processor-sdk-linux/esd/AM62X/09\\_00\\_00\\_03/exports/docs/linux/Foundational\\_Components/Kernel/Kernel\\_Drivers/SPI.html](https://software-dl.ti.com/processor-sdk-linux/esd/AM62X/09_00_00_03/exports/docs/linux/Foundational_Components/Kernel/Kernel_Drivers/SPI.html)
10. Texas Instruments, “Compiling Hello World Program: Linux Academy,” [Online]. Available: [https://dev.ti.com/tirex/explore/node?node=A\\_\\_Ael48yqnWh8ZfS-MiLdhcA\\_\\_linux\\_academy\\_am62x\\_\\_XaWts8R\\_\\_LATEST&search=am62x](https://dev.ti.com/tirex/explore/node?node=A__Ael48yqnWh8ZfS-MiLdhcA__linux_academy_am62x__XaWts8R__LATEST&search=am62x)
11. Texas Instruments, “Getting Started: AM62xMCU SDK,” [Online]. Available: [https://software-dl.ti.com/mcu-plus-sdk/esd/AM62X/latest/exports/docs/api\\_guide\\_am62x/GETTING\\_STARTED.html](https://software-dl.ti.com/mcu-plus-sdk/esd/AM62X/latest/exports/docs/api_guide_am62x/GETTING_STARTED.html)

## IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), 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 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](#) 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.

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

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265  
Copyright © 2023, Texas Instruments Incorporated