2023

MSPM0 – ADC Attach on AM62x using SPI

DIVYANSH MITTAL

Table of Contents

1 Abstract 2

2 Introduction 2

2.1 SPI Transaction Dataflow 2

2.2 AM62x Processor 4

2.3 MSPM0L130x Microcontroller 5

3 Hardware Setup 6

3.1 A53 Core 6

3.2 M4F Core 7

4 Software Setup 8

4.1 SK-AM62x 8

4.1.1 A53 Core 8

4.1.2 M4F Core 8

4.2 LP-MSPM0L130x 9

5 Steps for Execution 10

5.1 Step1: Run Project on LP-MSPM0L130x 10

5.2 Step2: Run Project on SK-AM62x 10

5.2.1 A53 Core 10

5.2.2 M4F Core 10

6 Expected Results 11

6.1 Single Byte Single Channel 12

6.2 Single Byte Multi Channel 13

6.3 Multi Byte Single Channel 14

6.4 Multi Byte Multi Channel 15

7 Summary 16

8 References 16

MSPM0 – ADC Attach on AM62x using SPI

1 Abstract

This paper 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.

2 Introduction

2.1 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.

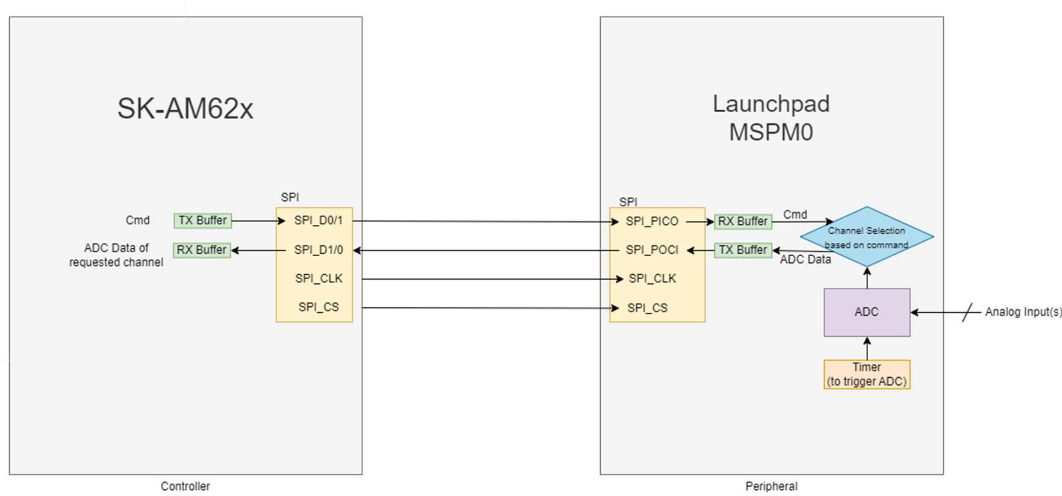


Figure 1 shows the overall dataflow between Controller and Peripheral.

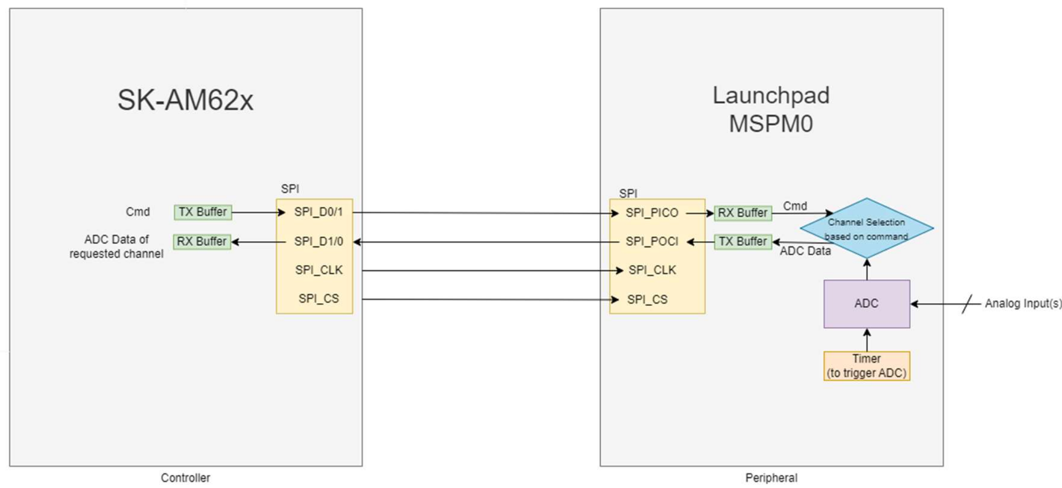


Figure 1: Overall Dataflow Between Controller (SK-AM62x) and Peripheral (LP-MSPM0L130x).

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.

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.

2.2 AM62x Processor

The AM62x Sitara Microprocessor, shown in Figure 2, is a heterogeneous processor designed for a wide variety of embedded applications. SPI can be enabled through MCU domain on M4F Core or MAIN domain on A53 Core. Figure 2 shows a simplified block diagram for AM62x.

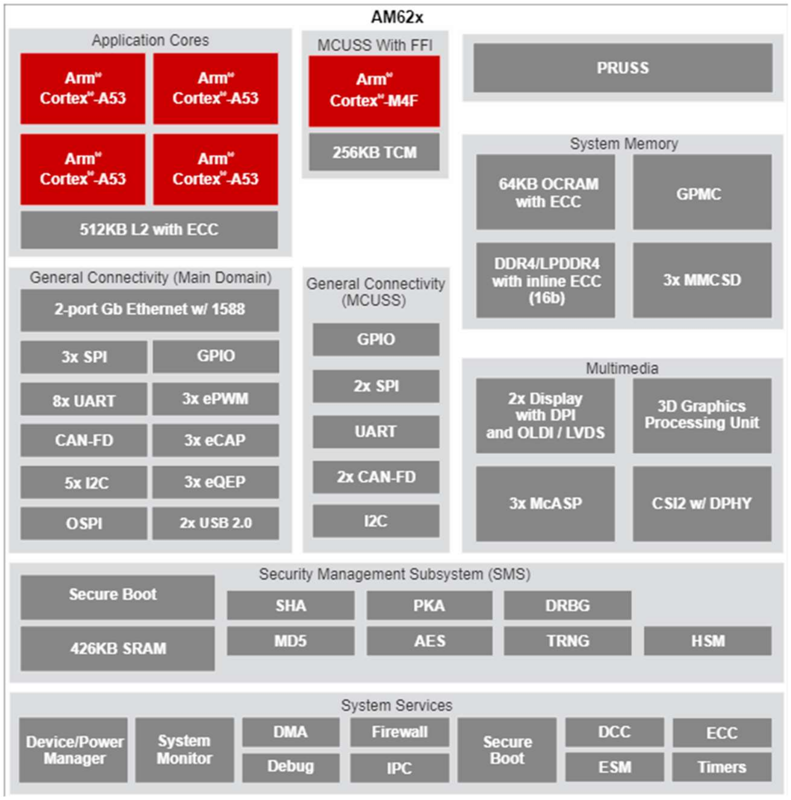


Figure 2. AM62x Simplified Block Diagram

For more details, see AM62x Sitara Processors Data Sheet.

2.3 MSPM0L130x Microcontroller

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

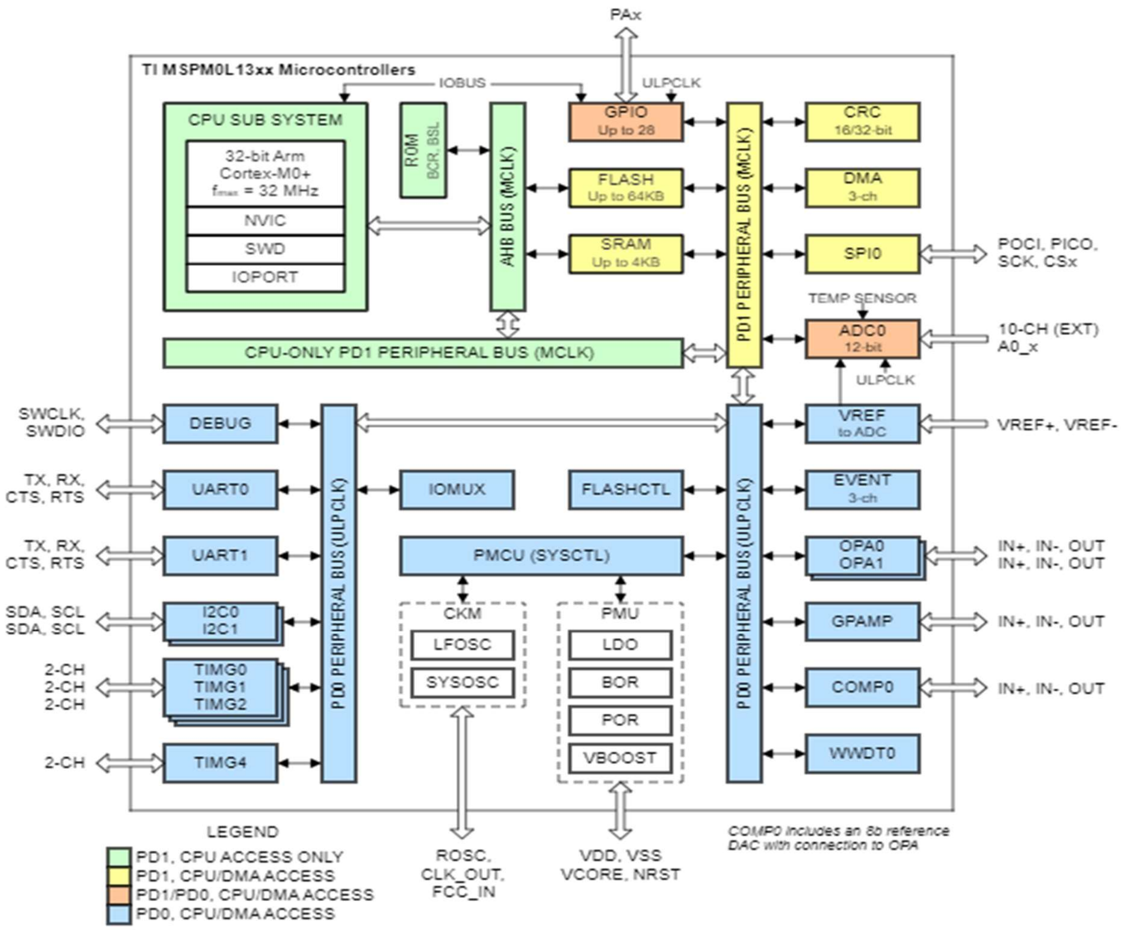


Figure 3. MSPM0L130x Simplified Block Diagram

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.

3 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.

3.1 A53 Core

For using A53 core, the peripheral pins for SPI on SK-AM62x are present in the User Expansion Header. Figure 5 shows the hardware setup.

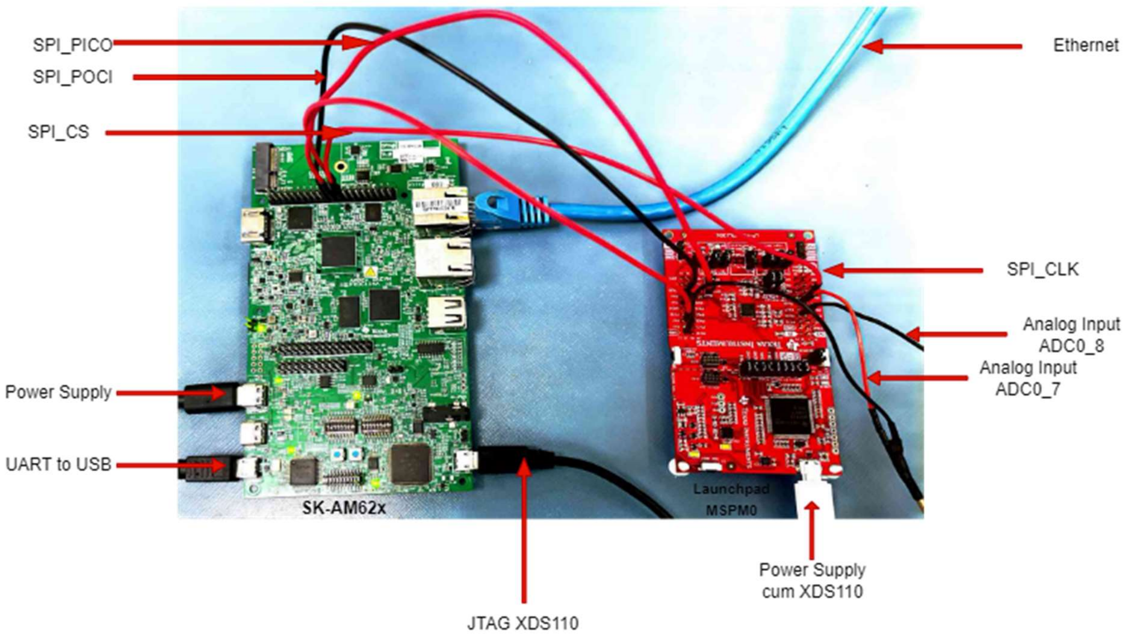


Figure 4: Cable Connections between SK-AM62x (A53 Core) and LP-MSPM0L1306 for SPI communication.

- 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 LP-MSPM0.
- 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.

3.2 M4F Core

For using M4F core, the peripheral pins for SPI on SK-AM62x are present in the MCU Header. Figure 5 shows the hardware setup.

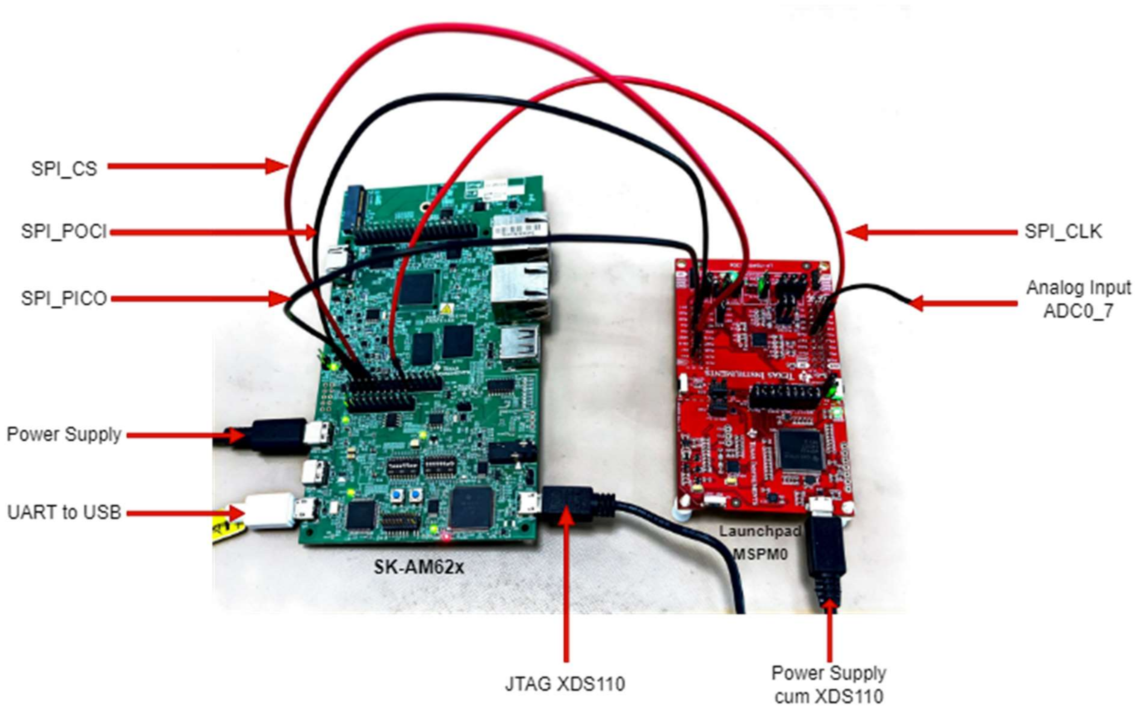


Figure 5: Cable Connections between SK-AM62x (M4F Core) and LP-MSPM0L1306 for SPI communication.

- 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.
- 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.

4 Software Setup

4.1 SK-AM62x

4.1.1 A53 Core

- Follow the setups provided on [AM62x Starter Kit EVM Quick Start Guide].
- Setup the SPI driver in Linux by modifying the kernel device tree using the following steps:
  - Find the k3-am625-sk.dts device tree file on the path `<psdk-installation-path>/board-support/linux-5.10.168+gitAUTOINC+2c23e6c538-g2c23e6c538/arch/arm64/boot/dts/ti`
  - 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 configurations provided on [SPI Kernel Driver].
  - Copy the generated k3-am625-sk.dtb on the root partition SD card at the location /boot by replacing the existing k3-am625-sk.dtb file.
  - Compile the downloaded 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.
- Insert the SD card back into SK-AM62x and reboot it.

4.1.2 M4F Core

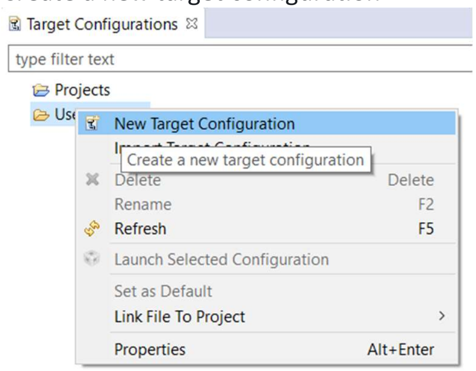
M4F Core MCU Domain:

- Perform the setup of Code Composer Studio (CCS) for AM62x documented here: [Getting Started Steps for AM62x].
- Move the downloaded project folder into your CCS workspace and import the CCS projects into the Project Explorer.

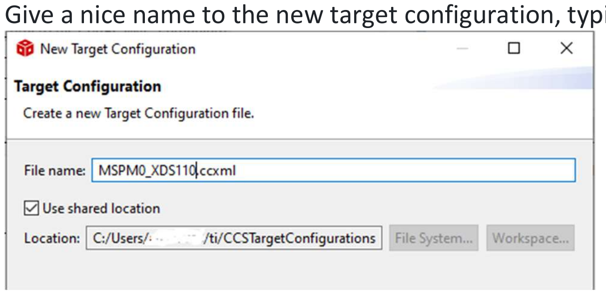
4.2 LP-MSPM0L130x

The same version of CCS, as used for AM62x M4F core, can be used for development on the MSPM0 Launchpad. To enable development on MSPM0 devices, select “MSPM0 32-bit Arm Cortex-M0+ General Purpose MCUs” in the “Select Components” window during CCS installation.

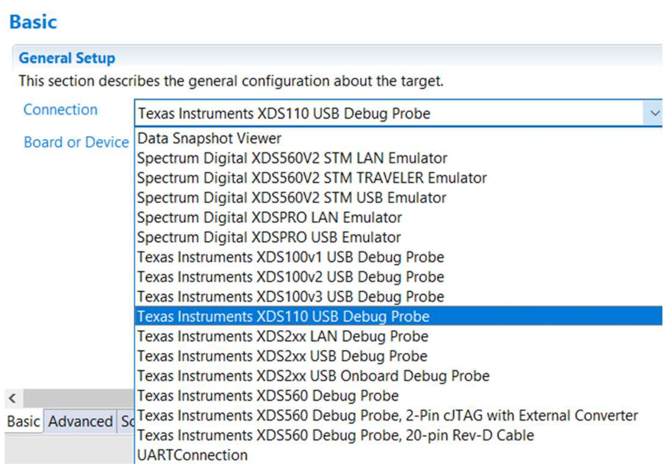
- Follow the following steps for adding new target configuration for MSPM0:
  a. Create a new target configuration



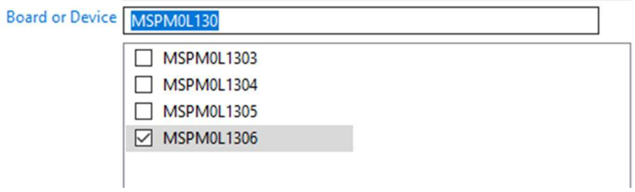
  b. Give a nice name to the new target configuration, typically {soc name}_{JTAG type}



  c. Select connection as XDS110 USB Debug Probe



d. In "Board or Device" type "MSPM0L130" and select "MSPM0L1306"



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

5 Steps for Execution

5.1 Step1: Run Project on LP-MSPM0L130x

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

1. Build the imported CCS project.
2. Right-click on MSPM0_XDS110.ccxml in Target Configurations window
3. Select "Launch Selected Configuration"
4. In the Debug window, click on the "Texas Instruments XDS110 USB Debug Probe_0/CORTEX_M0P"
5. Select Run -> Connect Target
6. Select Run -> Reset -> Subsystem Reset
7. Select Run -> Load -> Load Program
8. Browse to the pre-built binary for the MSPM0 project, and click "OK".
9. This will write the flash with the binary.
10. Select Run -> Resume

5.2 Step2: Run Project on SK-AM62x

5.2.1 A53 Core

On the serial monitor obtained through [AM62x Starter Kit EVM Quick Start Guide](), goto the location of the executable and run it using:

```
./<executable_name> -D <spidriver_name_from_/dev_folder> -s <speed> -v
```

Example:

```
./spidev_adc_multibyte_multichannel -D /dev/spidev3.0 -s 16000000 -v
```

5.2.2 M4F Core

Write pre-built SK-AM62x binaries to the on-chip flash:

1. Build the imported CCS project.
2. Right-click on AM62x_XDS110.ccxml in Target Configurations window
3. Select "Launch Selected Configuration"
4. In the Debug window, click on the "Texas Instruments XDS110 USB Debug Probe_0/BLAZAR_Cortex_M4F_1"
5. Select Run -> Connect Target

6. Select Run -> Reset -> Subsystem Reset
7. Select Run -> Load -> Load Program
8. Browse to the pre-built binary for the AM62x project and click “OK”.
9. This will write the flash with the binary.
10. Select Run -> Resume

6 Expected 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.

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

Sinusoidal Wave

```
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
```

Square Wave

```
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 =   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
```

6.2 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) and
- 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
```

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

Sinusoidal Wave

```
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
```

Square Wave

```
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 =   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
```

6.4 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) and
- 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
```

7 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.

8 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
8. Texas Instruments, “Kernel: Foundational Components: Processors SDK Linux AM62x,” [Online]. Available: https://software-dl.ti.com/processor-sdk-linux/esd/AM62X/08_06_00_42/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/08_06_00_42/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__AHboa0Lb4CnJkHPouAdFaA__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