Application Note

Application Design of ADS131M08 High-Precision 24-Bit Differential Synchronous ADC with MSPM0

Leon Yan *East China AA2 MCU FAE*

ABSTRACT

This document was translated from a simplified Chinese source. (ZHCAGB0)

The ADS131M08 is an eight-channel, simultaneous-sampling, 24-bit, Δ-Σ analog-to-digital converter (ADC) with wide dynamic range, low power consumption, and energy measurement-specific features. It is therefore ideal for energy metering, power metering, and circuit breaker applications. The ADC input can be directly connected to a resistor divider network or a power transformer to measure voltage, or to a current transformer or Rogowski coil to measure current. Individual ADC channels can be configured independently based on the sensor input. A low-noise, programmable gain amplifier (PGA) provides gain from 1 to 128 to amplify low-level signals. Additionally, the device integrates channel-to-channel phase calibration, offset, and gain calibration registers to eliminate signal chain errors.

The device integrates a low-drift, 1.2V reference for controlled cost and reduced PCB footprint. An optional cyclic redundancy check (CRC) in the data input, data output, and register map ensures communication integrity.

Applications include commercial and residential e-meters, circuit breakers, protection relays, power quality monitors, battery test equipment, battery management systems, and more.

This application note provides a systematic overview of the hardware interface, MSPM0's SPI configuration, ADS131M08-specific registers and initialization process, and multi-channel data reads for an engineered implementation of the ADS131M08 and MSPM0 microcontroller (MCU). This document is intended to provide foundational technical guidance to embedded system engineers on developing high-precision, multi-channel simultaneous measurement systems.

Table of Contents

List of Figures

1 Introduction to ADS131M08

The ADS131M08, a 24-bit, 8-channel simultaneous sampling product in TI's high-precision ADC family, offers 24-bit high-precision resolution with the core features of a built-in programmable gain amplifier (PGA). It is designed for applications requiring precise phase alignment across multiple channels and dynamic range.

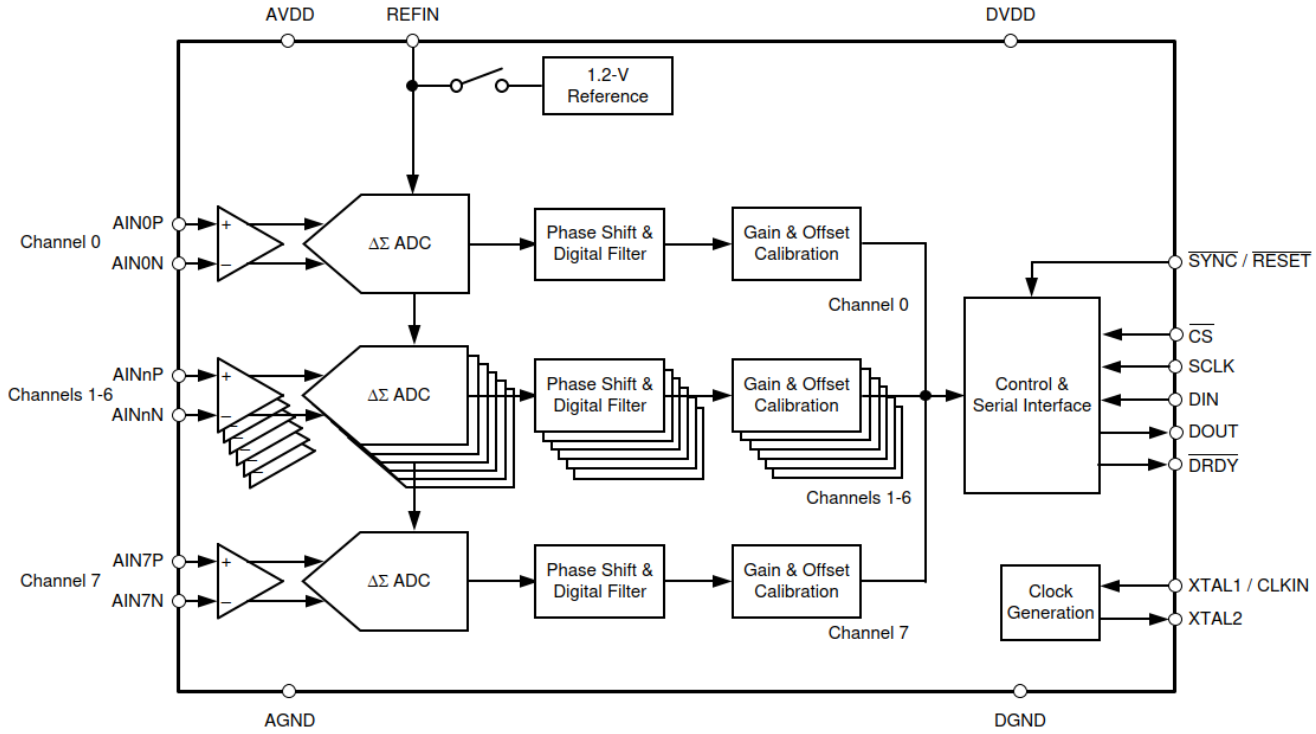

Figure 1-1. ADS131M08 block diagram

1.1 Major Applications of ADS131M08

The ADS131M08 delivers wide dynamic range, low power consumption, and energy measurement-specific features that make it ideal for many industrial applications.

Three-phase four-wire/three-wire energy metering: Simultaneous sampling of three-phase (A/B/C) voltages and currents (total of six channels) combined with zero-order/neutral current sensing (the 7th and 8th channels) is the ideal solution for building IEC Class 0.5S/0.2S three-phase e-meters. The simultaneous sampling feature ensures power calculation without phase error.

Simultaneous multi-sensor acquisition: In structural health monitoring and industrial control, it is often necessary to collect signals from multiple strain gauges, thermocouples, or pressure sensors at the same time. The simultaneous sampling capability of the ADS131M08 accurately captures the state of all sensors at the same physical instant and is critical for analyzing instantaneous correlation between variables.

Motor control and power analysis: It can be used to monitor phase voltages and phase currents of three-phase motors, enabling accurate power, efficiency, and power factor calculations. The eight channels can be expanded to simultaneously monitor other critical parameters like DC bus voltage.

Acoustic array and vibration analysis: Simultaneous acquisition of multiple microphones or accelerometers via the ADS131M08, combined with subsequent signal processing algorithms, can be used for source positioning, modal analysis, and so forth.

The ADS131M08 offers significant system-level advantages over the M02/4/6 models in terms of channel integration and data consistency, eliminating the complexity of clock synchronization and data stitching using multi-chip ADCs.

2 Introduction to MSPM0 Microcontroller and Its SPI Peripheral

The latest generation of MSPM0 microcontroller is based on the Cortex M0+ core, with the main frequency ranging from 24MHz to 80MHz. It offers rich configurations and multiple enhanced features. A 12-bit ADC with up to a 4MHz sampling rate and a 12-bit 1MHz DAC on the internal analog interface deliver high-performance support for data acquisition and analog output. Multiple internal integrated operational amplifiers (OPAs) save external devices and reduce the PCB footprint. The internal multi-channel DMA provides a high-speed path for data transmission between multiple peripherals and memories, reducing the utilization of the MCU core. The newly introduced Event System brings more flexible Event Post and Subscribe mechanisms, offering greater selectivity and flexibility to the interconnection between various modules and peripherals.

2.1 MSPM0 SPI Interface

- Supports MCLK/2 bit rates, up to 32Mbps (in both controller and peripheral modes)
- Can be configured as a controller or a peripheral
- Configurable chip selection for the controller and peripheral modes
- Programmable clock prescaler and bit rate
- Programmable data frame size from 4 bits to 16 bits (controller mode) and 7 bits to 16 bits (peripheral mode)
- Supports the PACKEN feature that allows two 16-bit FIFO entries to be packed into a 32-bit value to improve CPU performance
- Sends and receives FIFOs (four entries with 16 bits each), and supports DMA data transmission
- Supports TI mode, Motorola mode, and National Microwire format

2.2 MSPM0's SPI Operation Mode

The MSPM0's SPI interface supports a rate of up to 32Mbps, as well as TI mode, Motorola mode, and National Microwire format. In systems communicating with the ADS131M08, it must operate in Mode 1 (CPOL = 0, CPHA = 1), where the idle clock is low, data changes on the falling edge (the second edge) of the clock, and is sampled on the rising edge (the first edge).

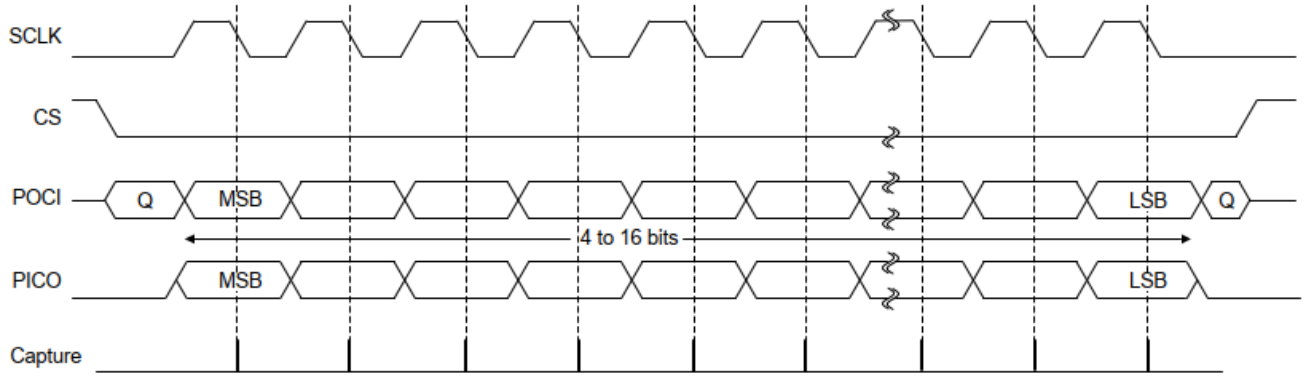

Figure 2-1. MSPM0 SPI Operation Mode SPO = 0, SPH = 1

3 Introduction to ADS131M08 Pin Functions

Before diving into register configuration and hardware design, it is important to fully understand the core pin functions of the ADS131M08. As a high-precision, multi-channel simultaneous sampling ADC, its pinout and design directly affect the performance, power consumption, and immunity of the system. Below is a functional breakdown and design guide for each key pin group.

3.1 ADS131M08 Pin Functions

The ADS131M08 is designed with separate analog and digital power rails to achieve the ideal signal-to-noise ratio (SNR) and reduce digital noise interference on analog channels.

AVDD/AGND (analog supply/ground): Supplies the internal analog front end, PGA, modulator, and reference voltage buffer. The typical voltage is 3.3V. Consider using low-noise power supplies such as an LDO in the design. Also, place high-quality decoupling capacitors close to the chip. The AVSS must be well connected to the analog ground plane.

DVDD/DGND (digital supply/ground): Provides power to the digital core, SPI interface, and clock circuitry. The typical voltage is 1.8V to 3.6V, which matches the I/O voltage of the MSPM0. Decoupling is also required in the design and can be shared with the system's digital power supply. The DGND is connected to the digital ground plane and can be designed with a single-point connection to the analog ground (AGND) to avoid ground-loop noise. Note that if the single-point grounding layout is not done properly, it can introduce additional noise.

Analog input pins (AINxP/AINxN): The ADS131M08 supports up to eight fully differential input channels (AIN0P/N to AIN7P/N). The fully differential input structure effectively reduces the common-mode noise (such as 50/60Hz line frequency interference), providing superior dynamic range and immunity compared to the single-ended input structure.

Note that the input range is determined by the PGA gain (1 to 128) and the reference voltage. Ensure that the sensor output signal falls within this range to avoid clipping.

For the input impedance, the ADS131M08's input impedance is 330kΩ. Each pair of differential inputs can be connected in series with small resistors (such as 10-100Ω) and in parallel with TVS/ESD protection devices to limit the transient current and improve ESD performance. Route the differential pair traces as equally long, equally spaced, and tightly coupled as possible, and keep them away from high-speed digital signal lines (such as SCLK).

Clock pin (CLKIN): The ADS131M08 requires an external main clock to drive its modulator and digital filter. Typical clock frequencies are 2.048MHz, 4.096MHz, or 8.192MHz (corresponding to different output data rates (ODR)). The clock duty cycle must be close to 50%, and the jitter should be as low as possible because clock jitter translates directly to the ADC noise. It can also be used in conjunction with MSPM0 CLKOUT to provide a stable, synchronized clock source to the ADS131M08, ensuring that the ADC is synchronized with the clock of the MCU. Therefore, simultaneous sampling is possible in a multi-device cascade.

SPI interface pins (SCLK, MOSI, MISO, and CS): SPI is the digital interface to configure the ADS131M08 registers and read conversion data. The SPI interface voltage is determined by the DVDD. Ensure its compatibility with the MSPM0 master I/O level. A standard four-wire SPI connection is adopted, and SPI mode 1 (CPOL = 0, CPHA = 1) is supported. Note that the CS signal must remain low throughout the SPI communication.

Reference voltage pin (REFIN): The ADC accuracy is directly dependent on the stability of the reference voltage. The ADS131M08 integrates a 1.2V reference internally. Users can also enable the external input path by providing an external input reference source through the REFIN pin and configuring the EXTREF_EN register bit.

DRDY (Data Ready): DRDY is a critical status indication pin in the ADS131M08 that provides the essential synchronization and control capabilities for data transmission. The DRDYn pin changes from high to low when a new complete data frame (containing the conversion results of all enabled channels) is ready to be read. Therefore, the MCU is informed and will read the ADC results after it queries through an interrupt or polling mode.

SYNC/RESET Pin: SYNC/RESET is a multi-function pin that triggers the ADS131M08 reset if it remains low for a period longer than tw(RSL) (2048 CLKIN cycles: 250uS). During normal operation, simultaneous sampling is triggered when the SYNC/RESET produces a falling-edge pulse with a width less than tw(SYL) (2047 CLKIN cycles: 249uS) but greater than one CLKIN cycle (122nS).

3.2 ADS131M08 SPI Interface

Achieving robust and efficient communication with the ADS131M08 requires analysis and understanding of the SPI protocol details. This chapter provides an in-depth analysis of several key points to avoid misuse in engineering applications.

SPI mode: The ADS131M08 supports SPI mode 1 (CPOL = 0, CPHA = 1), where the idle clock is low, and data changes on the falling edge (the second edge) of the clock and is sampled on the rising edge (the first edge). This is the mode that best matches the MSPM0 SPI peripheral.

Clock rate limitation: The maximum SCLK frequency is 25MHz. Over-speed communication will cause data misalignment. It is recommended to verify at a low speed during the debug phase before changing to a high frequency.

Data Format: All SPI communication follows the MSB-first principle. The alignment of the command word to the LSB of the data frame is handled at the software level.

Chip Select (CS): CS must remain low throughout the communication, including the complete transmission of commands, responses, and data frames. Pulling CS high too early will interrupt communication and cause a chip state machine error. Note that SPI hardware CS signals generally cannot be used because it is hard to guarantee that CS remains low throughout the communication. Therefore, a GPIO pin is required to control the CS signal.

4 ADS131M08 Commands and Data Frame Structure Details

The ADS131M08 contains eight sets of operation commands, divided into two major parts: basic control commands and register access commands.

COMMAND	DESCRIPTION	COMMAND WORD	RESPONSE
NULL	No operation	0000 0000 0000 0000	STATUS register
RESET	Reset the device	0000 0000 0001 0001	1111 1111 0010 1000
STANDBY	Place the device into standby mode	0000 0000 0010 0010	0000 0000 0010 0010
WAKEUP	Wake the device from standby mode to conversion mode	0000 0000 0011 0011	0000 0000 0011 0011
LOCK	Lock the interface such that only the NULL, UNLOCK, and RREG commands are valid	0000 0101 0101 0101	0000 0101 0101 0101
UNLOCK	Unlock the interface after the interface is locked	0000 0110 0101 0101	0000 0110 0101 0101
RREG	Read *nnn nnnn* plus 1 registers beginning at address *a aaaa a*	101*a aaaa annn nnnn*	*dddd dddd dddd dddd* or 111*a aaa annn nnnn* (1)
WREG	Write *nnn nnnn* plus 1 registers beginning at address *a aaaa a*	011*a aaaa annn nnnn*	010*a aaaa ammm mmmm* (2)

(1) When *nnn nnnn* is 0, the response is the requested register data *dddd dddd dddd dddd*. When *nnn nnnn* is greater than 0, the response begins with 111*a aaaa annn nnnn*, followed by the register data.

(2) In this case *mmm mmmm* represents the number of registers that are actually written minus one. This value may be less than *nnn nnnn* in some cases.

Figure 4-1. ADS131M08 Command Format

4.1 Basic Control Commands:

The basic control commands are the first six in the table, which are binary command words with fixed lengths.

- NULL (00000000000000): Null operation, typically used to maintain communication or as a placeholder
- RESET (00000000000001): Software reset that restores the chip to its initial state
- STANDBY (000000000100010): Places the device into low-power standby mode

- WAKEUP (0000000000110011): Wakes the device from standby mode to normal operating mode
- LOCK (00000101010101): Locks the interface to prevent accidental writes
- UNLOCK (00000110010101): Unlocks the interface to resume normal operation

4.2 Register Access Commands:

The register access commands are the last two and most important commands in the table.

- RREG command format: Read command ID (101)

  Address field (a aaaa a): 7-bit register start address

  Number field (nnn nnnn): 7-bit number. Actual number of registers read = nnn nnnn + 1
- WREG command format: Write command ID (011)

  Address field (a aaaa a): 7-bit register start address

  Number field (nnn nnnn): 7-bit number. Actual number of registers written = nnn nnnn + 1

4.3 Key Features of Communication Protocol:

1. Bulk register access: Both RREG and WREG support bulk operations of reading from and writing to multiple registers at consecutive addresses at once, greatly improving data transmission efficiency, especially when multiple parameters are configured.
2. Interface security locking mechanism: When the LOCK command locks an interface, only NULL, UNLOCK, and RREG commands can be executed, which effectively prevents configuration errors caused by accidental writes. The UNLOCK command is used to restore full access.
3. Write operation feedback: The WREG command response contains confirmation of the actual number of writes, which provides a mechanism to verify the success and integrity of the write operations and enhances the reliability of communication.
4. Operation mode management: The STANDBY/WAKEUP command switches the operation mode and supports the low-power standby mode for battery-powered applications.

5 ADS131M08 Registers

The register configuration of the ADS131M08 is at the heart of achieving accurate multi-channel measurement. Below is a detailed description of the key registers, covering status monitoring, configuration, interrupt, and data read.

ADDRESS	REGISTER	RESET VALUE	BIT 15 / BIT 7	BIT 14 / BIT 6	BIT 13 / BIT 5	BIT 12 / BIT 4	BIT 11 / BIT 3	BIT 10 / BIT 2	BIT 9 / BIT 1	BIT 8 / BIT 0
DEVICE SETTINGS AND INDICATORS (Read-Only Registers)										
00h	ID	28xxh	RESERVED				CHANCNT[3:0]			
			RESERVED							
01h	STATUS	0500h	LOCK	F_RESYNC	REG_MAP	CRC_ERR	CRC_TYPE	RESET	WLENGTH[1:0]	
			DRDY7	DRDY6	DRDY5	DRDY4	DRDY3	DRDY2	DRDY1	DRDY0
GLOBAL SETTINGS ACROSS CHANNELS										
02h	MODE	0510h	RESERVED		REGCRC_EN	RX_CRC_EN	CRC_TYPE	RESET	WLENGTH[1:0]	
			RESERVED			TIMEOUT	DRDY_SEL[1:0]		DRDY_HiZ	DRDY_FMT
03h	CLOCK	FF0Eh	CH7_EN	CH6_EN	CH5_EN	CH4_EN	CH3_EN	CH2_EN	CH1_EN	CH0_EN
			RESERVED			OSR[2:0]			PWR[1:0]	
04h	GAIN1	0000h	RESERVED	PGAGAIN3[2:0]			RESERVED	PGAGAIN2[2:0]		
			RESERVED	PGAGAIN1[2:0]			RESERVED	PGAGAIN0[2:0]		
05h	GAIN2	0000h	RESERVED	PGAGAIN7[2:0]			RESERVED	PGAGAIN6[2:0]		
			RESERVED	PGAGAIN5[2:0]			RESERVED	PGAGAIN4[2:0]		
06h	CFG	0600h	RESERVED			GC_DLY[3:0]				GC_EN
			CD_ALLCH	CD_NUM[2:0]			CD_LEN[2:0]			CD_EN
07h	THRSHLD_MSB	0000h	CD_TH_MSB[15:8]							
			CD_TH_MSB[7:0]							
08h	THRSHLD_LSB	0000h	RESERVED				DCBLOCK[3:0]			

Figure 5-1. ADS131M08 Register 1

ID and STATUS registers: While debugging SPI communication, the user can indirectly determine whether the SPI communication is normal by reading these two registers and also check the current status of the chip.

MODE Register (address 0x02): Configures the global operation mode. This includes several key points: the software RESET bit, communication frame length setting, CRC setting, and DRDY setting.

CLOCK register (address 0x03): Configures the oversampling ratio (OSR) and modulator power mode.

GAIN1 and GAIN2 registers (addresses 0x04 and 0x06): These are key configurations for the ADS131M08. The GAIN1 register independently controls the PGA gain for channels 0 to 3. The GAIN2 register independently controls the PGA gain for channels 4 to 7.

CFG register (address 0x05): The global configuration register that configures the global chopping parameters and current sensing capability parameters.

THRSHLD_MSB and THRSHLD_LSB registers: Configure the current sensing mode parameters.

CHANNEL-SPECIFIC SETTINGS						
09h	CH0_CFG	0000h	PHASE0[9:2]			
			PHASE0[1:0]	RESERVED	DCBLK0_DIS0	MUX0[1:0]
0Ah	CH0_OCAL_MSB	0000h	OCAL0_MSB[15:8]			
			OCAL0_MSB[7:0]			
0Bh	CH0_OCAL_LSB	0000h	OCAL0_LSB[7:0]			
			RESERVED			
0Ch	CH0_GCAL_MSB	8000h	GCAL0_MSB[15:8]			
			GCAL0_MSB[7:0]			
0Dh	CH0_GCAL_LSB	0000h	GCAL0_LSB[7:0]			
			RESERVED			

Figure 5-2. ADS131M08 Register 2

CHx_CFG register: Configures independent parameters of individual channels, including ADC input configuration, DC blocking, and phase compensation for each channel.

CH0_OCAx_xxx and CH0_GCALx_xxx registers: Contain Offset and Gain calibration values for each channel.

6 Typical ADS131M08 Configuration Process and Key Points

6.1 ADS131M08 Initialization

- For reliability after a power-on reset, a second reset can be done through the SYNC/RESET pin, followed by reading the ID and STATUS registers to confirm the SPI communication and ADS131M08 status are normal.
- Configure the MODE register to program the data word length, CRC, and DRDY mode.
- Configure the CLOCK register to select the OSR and modulator power mode to balance accuracy with speed.
- Program the appropriate PGA gain for each channel through the GAIN1/GAIN2 registers.
- Configure the CFG register to set the global chopping parameters and the current sensing capability parameters.
- Enable and configure the desired channels one by one through the CHx_CFG registers. It is recommended to turn off unused channels.
- Load the calibrated Offset and Gain calibration values for each channel.
- After the critical configuration is complete, a lock sequence can be written to the LOCK register to prevent unintended changes to the runtime configuration.

At this point, the ADS131M08 is already functioning properly, and the DRDY pin generates low-level pulses corresponding to the OSR, indicating that the ADC sampling results are updated periodically.

6.2 Key Points of ADS131M08 Initialization

- Following the power-on reset, the ADS131M08 must be initialized in the following sequence to ensure that all channels are working as expected.
- Clock and interface preparation: First, make sure that the MSPM0's CLKOUT or an external crystal has output a stable clock to the ADS131M08's CLKIN pin, and complete the master mode configuration of the MSPM0 SPI peripheral (Mode 1, with an appropriate rate such as 2MHz).
- Configure the MODE register to a 24-bit word length. Other word lengths complicate communication. If the MCU model supports 32-bit transmission, also consider configuring it to a 32-bit word length. Enable CRC and configure DRDY behavior.
- To configure the GAIN1 and GAIN2 registers, set the PGA gains for channels 0-3 and 4-7, respectively, depending on the signal amplitude of each channel. This is a critical step to match different sensor ranges.
- The GAIN1/GAIN2 configurations must be closely aligned to the hardware front-end design to prevent signals from overranging or from not fully utilizing the ADC dynamic range.
- When configuring the CLOCK register, set the OSR and modulator power modes based on the target output data rate and accuracy requirements. A higher sampling rate means a lower ENOB along with higher ADC sampling and conversion noise.
- Note that if chopping is enabled in the CFG register, the final ADC sampling rate will be lower than the configured OSR, which is normal and requires an actual oscilloscope measurement to determine the final sampling rate.

7 MSPM0 SPI Configuration and API Function Implementation

Clock configuration: Configure the SPI peripheral clock (such as dividing down from the system clock) to a rate not exceeding the ADC limit. A lower rate (such as 2MHz) is recommended for initial debugging.

SPI mode: Configure Clock Polarity to Low and Phase to Data captured on second clock edge to meet the configuration requirements of Mode 1. Also, the Frame Size of the SPI is configured to 8Bit.

CS pin: The ADS131M08 requires the CS to remain low throughout the communication. For example, reading the conversion results of the eight-channel ADC takes a 30-byte read process, during which the CS must stay low. This is impossible for the MCU hardware CS signal, so software manipulation of GPIO is needed to implement the CS function. In the MSPM0's SPI configuration, a three-wire SPI interface must be selected so that the CS is controlled by software.

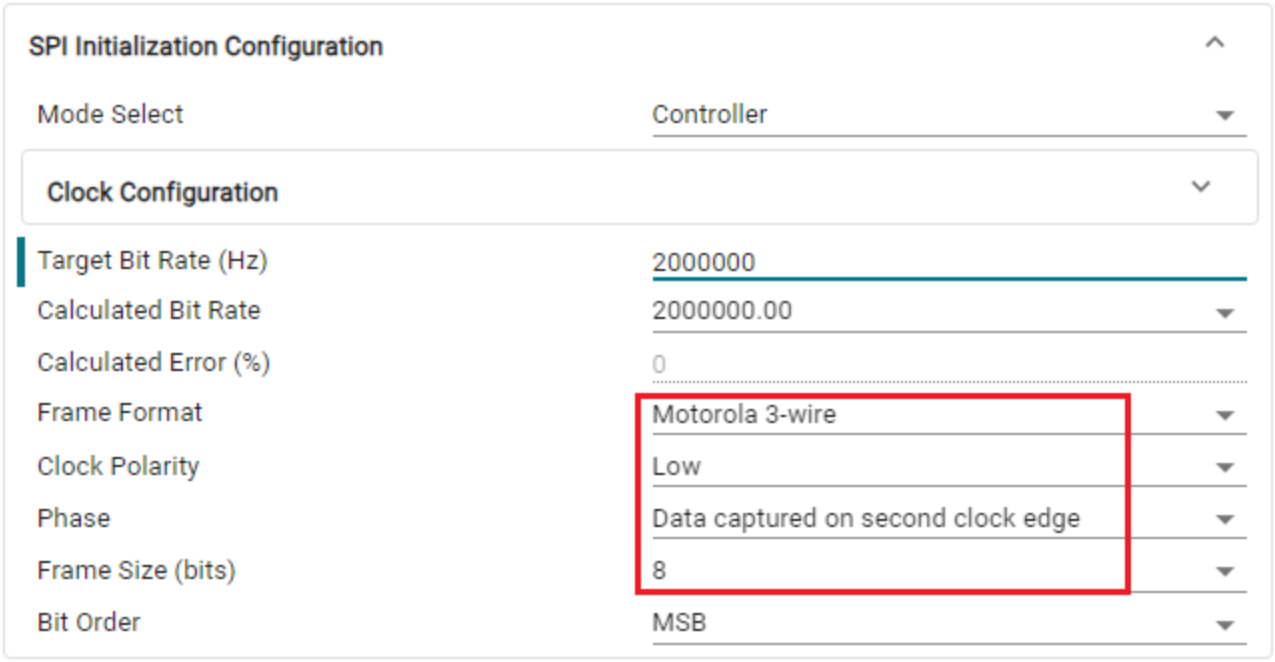

Figure 7-1. MSPM0's SPI Configuration

7.1 MSPM0 SPI Sending and Receiving Functions

Below describes how to implement the lowest level of SPI data sending and receiving. The most basic blocking mode is currently used, and users can consider the interrupt mode later.

```
uint8_t spiSendReceiveByte(const uint8_t dataTx)
{
    /* --- INSERT YOUR CODE HERE ---
    * This function should send and receive single bytes over the SPI.
    * NOTE: This function does not control the /CS pin to allow for
    * more programming flexibility.
    */
  while (DL_SPI_isBusy(SPI_0_INST));
  DL_SPI_transmitData8(SPI_0_INST, dataTx);
  while (DL_SPI_isBusy(SPI_0_INST));
  return(DL_SPI_receiveDataBlocking8(SPI_0_INST));
}
```

7.2 ADS131M08 Command Sending Function

The ADS131M08 command frame sending is described in this section. It proceeds in a different way from register reads and writes and must be handled separately.

```
uint16_t sendSpecialCommand(uint16_t opcode)
{
    /* tAssert if this function is used to send any of the following opcodes */
    tAssert(OPCODE_RREG != opcode); /* Use "readSingleRegister()" */
    tAssert(OPCODE_WREG != opcode); /* Use "writeSingleRegister()" */
    tAssert(OPCODE_LOCK != opcode); /* Use "lockRegisters()" */
    tAssert(OPCODE_UNLOCK != opcode); /* Use "unlockRegisters()" */
    tAssert(OPCODE_RESET != opcode); /* Use "resetDevice()" */

     // Build TX and RX byte array
#ifdef ENABLE_CRC_IN
     uint8_t dataTx[8] = { 0 }; // 2 words, up to 4 bytes each = 8 bytes maximum
     uint8_t dataRx[8] = { 0 };
#else
     uint8_t dataTx[4] = { 0 }; // 1 word, up to 4 bytes long = 4 bytes maximum
     uint8_t dataRx[4] = { 0 };
#endif
     uint8_t numberOfBytes = buildSPIarray(&opcode, 1, dataTx);

      /* Set the nCS pin LOW */
    setCS(LOW);
     // Send the opcode (and crc word, if enabled)
    int i;
    for (i = 0; i < numberOfBytes; i++)
       {
         dataRx[i] = spiSendReceiveByte(dataTx[i]);
       }
     /* Set the nCS pin HIGH */
    setCS(HIGH);
     // Combine response bytes and return as a 16-bit word
     uint16_t adcResponse = combineBytes(dataRx[0], dataRx[1]);
   return adcResponse;
}
```

7.3 ADS131M08 Register Write Function

The ADS131M08 register write operation introduced in this section requires combining the frame format according to its standard format.

```
voidwriteSingleRegister(uint8_t address, uint16_t data)
{
      /* Check that the register address is in range */
    tAssert(address < NUM_REGISTERS);

      // (OPTIONAL) Enforce certain register field values when
      // writing to the MODE register to fix the operation mode
    if (MODE_ADDRESS == address)
      {
       data = enforce_selected_device_modes(data);
      }
     // Build TX and RX byte array
#ifdef ENABLE_CRC_IN
      uint8_t dataTx[12] = { 0 }; // 3 words, up to 4 bytes each = 12 bytes maximum
      uint8_t dataRx[12] = { 0 };
#else
      uint8_t dataTx[8] = { 0 }; // 2 words, up to 4 bytes long = 8 bytes maximum
      uint8_t dataRx[8] = { 0 };
#endif
      uint16_t opcodes[2];
      opcodes[0] = OPCODE_WREG | (((uint16_t) address) << 7);
      opcodes[1] = data;
      uint8_t numberOfBytes = buildSPIarray(&opcodes[0], 2, dataTx);
      // Send command
   spiSendReceiveArrays(dataTx, dataRx, numberOfBytes);
      // Update internal array
    registerMap[address] = data;

      // (RECOMMENDED) Read back register to confirm register write was successful
```

```
    readSingleRegister(address);
     // NOTE: Enabling the CRC words in the SPI command will NOT prevent an invalid W
}
```

7.4 ADS131M08 Register Read Function

The ADS131M08 register read operation introduced in this section requires combining the frame format according to its standard format.

```
uint16_t readSingleRegister(uint8_t address)
{
      /* Check that the register address is in range */
    tAssert(address < NUM_REGISTERS);

      // Build TX and RX byte array
#ifdef ENABLE_CRC_IN
      uint8_t dataTx[8] = { 0 }; // 2 words, up to 4 bytes each = 8 bytes maximum
      uint8_t dataRx[8] = { 0 };
#else
      uint8_t dataTx[4] = { 0 }; // 1 word, up to 4 bytes long = 4 bytes maximum
      uint8_t dataRx[4] = { 0 };
#endif
      uint16_t opcode = OPCODE_RREG | (((uint16_t) address) << 7);
      uint8_t numberOfBytes = buildSPIarray(&opcode, 1, dataTx);

      // [FRAME 1] Send RREG command
    spiSendReceiveArrays(dataTx, dataRx, numberOfBytes);

      // [FRAME 2] Send NULL command to retrieve the register data
    registerMap[address] = sendSpecialCommand(OPCODE_NULL);

    returnregisterMap[address];
}
```

7.5 ADS131M08's ADC Data Read Function

When the MCU side detects the falling edge of DRDY, it indicates that the currently configured channel has completed sampling and conversion, and the ADS131M08 results can be read, where up to eight channels can be read directly and continuously.

```
boolreadData(adc_channel_data *DataStruct)
{
     int i;
        uint8_t crcTx[4] = { 0 };
        uint8_t dataRx[4] = { 0 };
        uint8_t bytesPerWord = getWordByteLength();

#ifdef ENABLE_CRC_IN
       // Build CRC word (only if "RX_CRC_EN" register bit is enabled)
       uint16_t crcWordIn = calculateCRC(&DataTx[0], bytesPerWord * 2, 0xFFFF);
       crcTx[0] = upperByte(crcWordIn);
       crcTx[1] = lowerByte(crcWordIn);
#endif

       /* Set the nCS pin LOW */
     setCS(LOW);
#if 1
       uint8_t dataRxTemp[32];
       uint8_t dataTxTemp[32] = {0};

    setCS(LOW);
    spiSendReceiveByteMulti(dataTxTemp, dataRxTemp, 30); //3 + 8*3 + 3 = 30
    setCS(HIGH);
       DataStruct->response = combineBytes(dataRxTemp[0], dataRxTemp[1]);
       DataStruct->channel0 = signExtend(&dataRxTemp[3]);
       DataStruct->channel1 = signExtend(&dataRxTemp[6]);
       DataStruct->channel2 = signExtend(&dataRxTemp[9]);
       DataStruct->channel3 = signExtend(&dataRxTemp[12]);
       DataStruct->channel4 = signExtend(&dataRxTemp[15]);
       DataStruct->channel5 = signExtend(&dataRxTemp[18]);
       DataStruct->channel6 = signExtend(&dataRxTemp[21]);
       DataStruct->channel7 = signExtend(&dataRxTemp[24]);
```

```
    return 1;
#endif
}
```

8 Conclusion

The combination of the ADS131M08 and the MSPM0 microcontroller provides a cost-effective solution for systems requiring high-precision, multi-channel simultaneous data acquisition. Its core advantages include:

Integration and synchronization: A single chip enables true simultaneous sampling across eight channels, simplifying the system architecture and fundamentally addressing the clock synchronization and data alignment challenges of multi-chip ADC implementations.

Master-slave collaboration: It is optional to use the CLKOUT pin of the MSPM0 as the clock source of the ADS131M08, satisfying the synchronization needs of the MCU with the ADC in certain applications.

Flexible accuracy configuration: Independent GAIN1/GAIN2 registers allow for differentiated gain settings on the first and last four channels, better suited for applications with mixed signal amplitudes such as three-phase metering.

This document guides users to thoroughly understand the ADS131M08's register settings, SPI interface communication, and hardware pin design considerations, laying a solid foundation for users to apply the combined MSPM0 and ADS131M08 solution in their products.

9 References

1. ADS131M08 Eight-Channel, Simultaneously-Sampling, 24-Bit, Δ-Σ ADC Data Sheet (Rev. B)
2. MSPM0G350x Mixed-Signal Microcontroller with CAN-FD Interface Data Sheet (Rev. C)
3. How Arm Cortex-M0+ MCUs Optimize General-Purpose Processing, Sensing and Control
4. MSPM0 MCU Development Guide (Rev. G)
5. How SysConfig Drives Embedded System Development

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.

Last updated 10/2025