ANALOG DEVICES

AN-1293
APPLICATION NOTE

One Technology Way • P.O. Box 9106 • Norwood, MA 02062-9106, U.S.A. • Tel: 781.329.4700 • Fax: 781.461.3113 • www.analog.com

# A Quick Guide to the ADuCM350 Sequencer

## INTRODUCTION

The ADuCM350 is an ultralow power, integrated, mixed-signal metering solution that includes a microcontroller subsystem for processing, control, and connectivity. The processor subsystem is based on a low power ARM® Cortex™-M3 processor, a collection of digital peripherals, an embedded SRAM and flash memory, and an analog subsystem which provides clocking, reset, and power management capabilities.

This application note details the ADuCM350 analog front end (AFE) sequencer. The role of the ADuCM350 AFE sequencer is to allow offloading of the low level AFE operations from the Cortex-M3 and to provide cycle accurate control over the analog and digital signal processor (DSP) blocks. The ADuCM350 AFE sequencer handles timing critical operations without being subject to system load.

## TABLE OF CONTENTS

Introduction ..... 1
Revision History ..... 2
Basics of the ADuCM350 Sequencer ..... 3
Write Command ..... 3
Timer Command ..... 3
Safety Features ..... 4
Sequence Abort ..... 5
Example Sequences ..... 6
Safety Word ..... 6
Example Write Command ..... 6
Example Wait Command ..... 6
Appendix A: Online CRC-8 Tool ..... 6
Example 1. CRC In Sequence ..... 6
Example 2. Temperature Sensor Measurement ..... 7

## REVISION HISTORY

**5/14—Revision 0: Initial Version**

## BASICS OF THE ADuCM350 SEQUENCER

The AFE sequencer is clocked by the analog clock source (ACLK), with a frequency of 16 MHz. The ADuCM350 uses clock gating internally to minimize power consumption when idle.

The AFE sequencer reads commands first in, first out (FIFO), and depending on the command, either waits for a given period or writes a value to a memory mapped register (MMR). The execution of the command is sequential, with no branching. The AFE sequencer cannot read MMR values or signals from the analog and DSP blocks. If a particular sequence needs conditional execution, the execution must be split between the AFE sequencer and the Cortex-M3, with the decision block executed on the Cortex-M3. In this case, the AFE sequencer is programmed with commands up to the conditional point, and upon completion of those commands, a Cortex-M3 interrupt service routine determines the subsequent commands to be executed and proceeds to load them into the command FIFO.

There are two types of commands that can be executed by the AFE sequencer: write (MSB = 1) and timer (MSB = 0).

### *Write Command*

The ADDRESS field is 6 bits wide, allowing access to 64 MMRs. All MMR accesses are only 32 bits; byte and half word accesses are not allowed. Write only is implied for all accesses. The write command uses a direct mapping between the MMR address and the ADDRESS field. The ADDRESS field corresponds to Bits[7:2] of the 32-bit MMR address.

For example, when the Cortex-M3 wants to write to Register AFE_WG_CFG, it uses the 0x40080014 address. To write to the same register using the AFE sequencer, the ADDRESS field needs to be 0b000101 (Bits[30:25] of the address used by the Cortex-M3). See Figure 1.

The DATA field is 25 bits wide and allows writing to the MMR bits, Bits[24:0]. To keep the width of the command FIFO in line with the advanced high performance (AHB) bus, writing to the full 32 bits of the MMRs via the sequencer is not possible. However, because the MMR bits, Bits[31:25], are not used by any of the MMRs, all assigned MMR bits can be written by the AFE sequencer.

### *Timer Command*

There are two timer commands; wait command and timeout command.

#### Wait Command

The wait command (see Figure 2) introduces wait states in the AFE sequencer execution. When the programmed counter reaches 0, the execution resumes by reading the next command from the FIFO.

#### Timeout Command

The timeout command (see Figure 3) starts a counter that works independently of the AFE sequencer flow. The current value of the counter can be read by the Cortex-M3 at any time during the AFE_SEQ_TIMEOUT register. There are two interrupt bits associated with this command; SEQ_TIMEOUT_FINISHED and SEQ_TIMEOUT_ERR which are asserted at the end of the timeout period. SEQ_TIMEOUT_ERR is asserted if, at the end of the timeout period, the AFE sequencer has not reached the end of execution (END_OF_SEQ equals 0).

When the AFE sequencer execution stops because of an AFE or sequencer write command (writing a 0 to SEQ_EN), the timeout counter does not reset. However, if the Cortex-M3 writes a 0 to SEQ_EN, it is reset, which applies to situations in which the Cortex-M3 needs to abort the sequence.

The TIME unit for both timer commands is 1 ACLK period. For a clock frequency of 16 MHz, the timer resolution is 62.5 ns, and the maximum timeout is ~67.1 seconds, which is true even if the SEQ_WRITE_TIMER bits are nonzero.

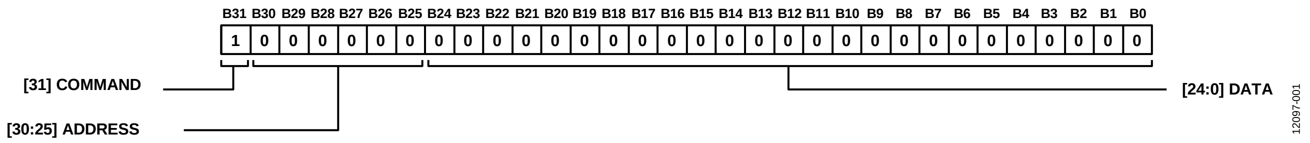

B31 B30 B29 B28 B27 B26 B25 B24 B23 B22 B21 B20 B19 B18 B17 B16 B15 B14 B13 B12 B11 B10 B9 B8 B7 B6 B5 B4 B3 B2 B1 B0
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[31] COMMAND
[30:25] ADDRESS
[24:0] DATA
12097-001

*Figure 1. Write Command*

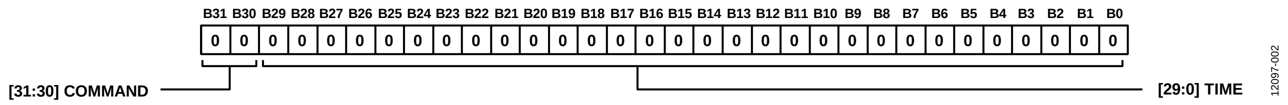

B31 B30 B29 B28 B27 B26 B25 B24 B23 B22 B21 B20 B19 B18 B17 B16 B15 B14 B13 B12 B11 B10 B9 B8 B7 B6 B5 B4 B3 B2 B1 B0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[31:30] COMMAND
[29:0] TIME
12097-002

*Figure 2. Wait Command*

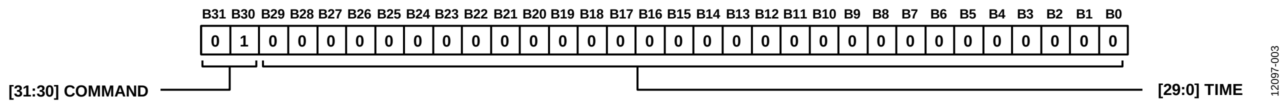

B31 B30 B29 B28 B27 B26 B25 B24 B23 B22 B21 B20 B19 B18 B17 B16 B15 B14 B13 B12 B11 B10 B9 B8 B7 B6 B5 B4 B3 B2 B1 B0
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[31:30] COMMAND
[29:0] TIME
12097-003

*Figure 3. Timeout Command*

### *Safety Features*

Although the MMRs, with the exception of the calibration registers, are located in the address space that can be written by the AFE sequencer, only a subset of them can be written by the sequencer. In addition to the read only registers, sequencer writes to the following registers have no effect:

- AFE_CMD_FIFO_WRITE
- AFE_ANALOG_CAPTURE_INT
- AFE_CMD_FIFO_INT
- AFE_DATA_FIFO_INT
- AFE_SEQ_COUNT
- AFE_SEQ_CRC

If the AFE sequencer attempts to write to any of the previous registers, the write is silently ignored. If such a write is attempted due to an error (such as the wrong command read from the memory), the error is detected when the cyclic redundancy check (CRC) and command count values are checked.

The rate at which AFE sequencer commands are executed is controlled through the SEQ_WRITE_TIMER bits in the AFE_SEQ_CFG register. When a write command is executed by the sequencer, it performs the MMR write and waits SEQ_WRITE_TIMER clock cycles before taking the next command from the FIFO. The effect is the same as when a write command is followed by a wait command. The main purpose is to reduce the code size when generating arbitrary waveforms. The SEQ_WRITE_TIMER bits do not have any effect following a wait or a timeout command.

In addition to a single write command followed by a wait command, multiple write commands can be executed in succession, followed by a wait command. Therefore, any configuration can be set up rapidly by the AFE sequencer, regardless of the number of register writes followed by a precisely executed delay.

#### Sequence Count

The number of commands executed by the sequencer can be read from the AFE_SEQ_COUNT register. Each time a command is read from the FIFO and executed, the counter is incremented by 1. Performing a write to the AFE_SEQ_COUNT register resets the counter.

#### Sequence CRC

The sequencer calculates the CRC of all the commands it executes. The algorithm used is CRC-8, using the polynomial, $x^8 + x^2 + x + 1$.

The CRC-8 algorithm performs on 32-bit input data (sequencer instructions). Each 32-bit input is processed in a one clock cycle, and the result is available immediately for reading by the Cortex-M3.

The CRC value can be read from the AFE_SEQ_CRC register. This register is reset by the same mechanism as the command count, by writing to the AFE_SEQ_COUNT register. The AFE_SEQ_CRC resets to a seed value of 0x01. Attempting to write to the AFE_SEQ_CRC register results in the same error response as writing to any other read only register.

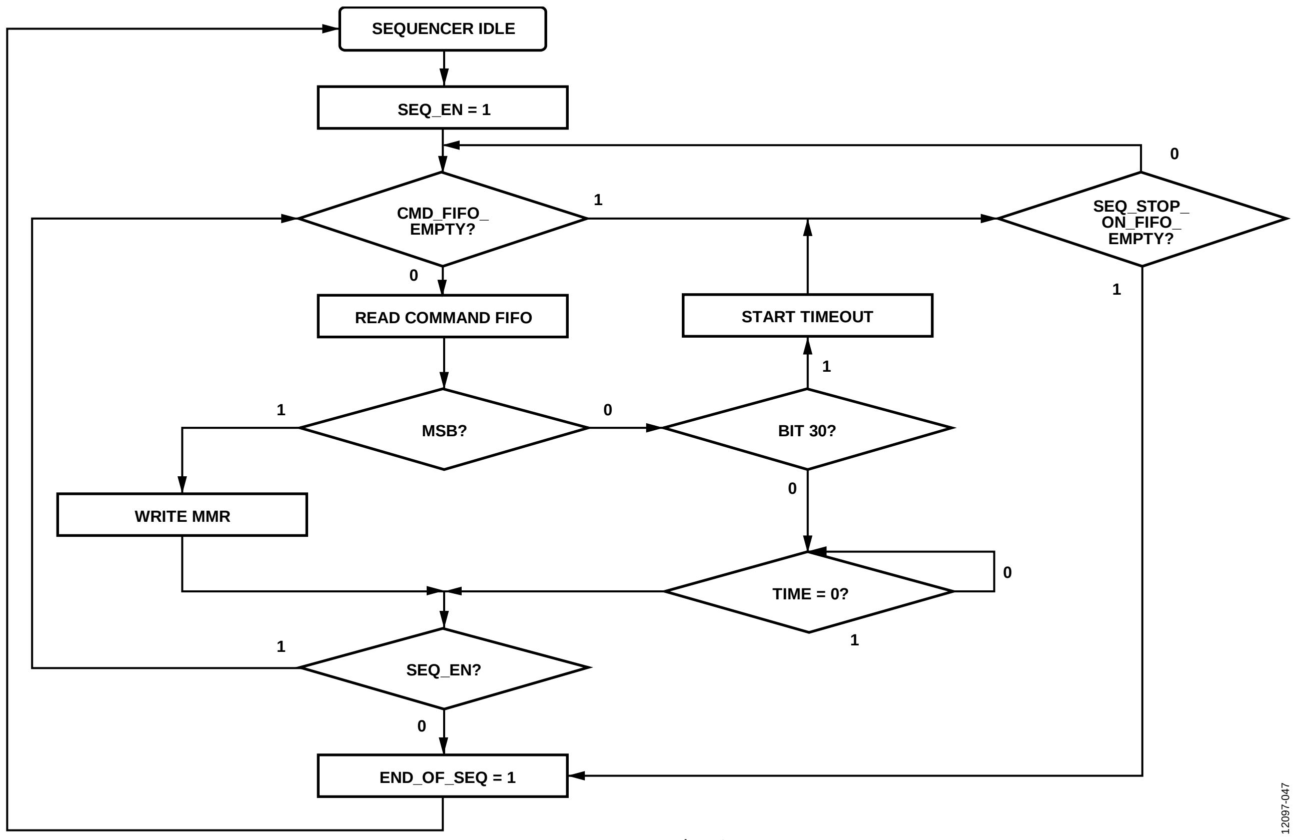


*Figure 4. Execution Flow Diagram*

**Table 1. Abort Sequence Example**

| Action | Register | Bit(s) | Value | Comments |
|---|---|---|---|---|
| Disable Sequencer | AFE_SEQ_CFG | SEQ_EN | 0 | Resets both internal timers (for wait and timeout commands) |
| Disable Command FIFO | AFE_FIFO_CFG | CMD_FIFO_EN | 0 | Resets read and write pointers |
| | | CMD_FIFO_DMA_REQ_EN | 0 | Disables command FIFO direct memory access (DMA) requests |
| Disable DSP Blocks | AFE_CFG | WAVEGEN_EN | 0 | Disables waveform generator |
| | | DFT_EN | 0 | Disables discrete Fourier transform (DFT) engine |
| | | SUPPLY_LPF_EN | 0 | Disables supply rejection filter |
| Stop Temperature Measurement | AFE_CFG | TEMP_CONV_EN | 0 | |
| Stop Analog-to-Digital Conversions | AFE_CFG | ADC_CONV_EN | 0 | |
| Power Down Analog Blocks | AFE_CFG | As needed | | |

Typically, the AFE sequencer is idle. Writing a 1 to the SEQ_EN bit in the AFE_SEQ_CFG register starts the AFE sequencer. Only the Cortex-M3 can write SEQ_EN equals 1 to start the sequencer. The execution flow diagram is shown in Figure 4.

The last command in any sequence writes a 0 to SEQ_EN, which forces the end of the execution and triggers the END_OF_SEQ interrupt.

Attempting to read from the FIFO command when empty, SEQ_STOP_ON_FIFO_EMPTY equals 1, turns the sequencer off and is a valid way to end the sequence. It can be useful to turn the SEQ_STOP_ON_FIFO_EMPTY flag off for sequences with a minimum timing specification and to turn it on for sequences with strict timing.

### *Sequence Abort*

To abort a sequence, the Cortex-M3 needs to write a number of actions to the AFE to disable the sequencer and stop the different blocks that can be used at this point. Table 1 details the possible actions. The first two actions shown (disable the sequencer and disable command FIFO) must always be done. The other actions may or may not be required, depending on the AFE mode of operation at the time the sequence is aborted. The sequencer execution can be paused through the SEQ_HALT bit in the AFE_SEQ_CFG register. When the bit equals 1, the sequencer execution stops, which applies to every AFE function, including FIFO operations, internal timers, waveform generation, and data capture. Cortex-M3 reads from the MMRs are allowed, and this mode is used for debug purposes during software development.

## EXAMPLE SEQUENCES

The following sequence, which powers up the excitation channel, explains the hardware and software features of the sequencer:

```
const uint32_t seq_afe_tempsensmeas[] = {
    0x00080081,   /* safety word: Bits[31:16]
= command count, Bits[7:0] = CRC*/
    0x800210B0,   /* AFE_CFG: TEMP_SENS_EN = 1
*/
    0xA0000103,   /* AFE_ADC_CFG: MUX_SEL =
00011, GAIN_OFFS_SEL = 01 (TS)*/
    0x00000640,   /* Wait 100 µs */
    0x800331B0,   /* AFE_CFG: ADC_CONV_EN = 1,
SUPPLY_LPF_EN = 1, TEMP_CONV_EN = 1 */
    0x00090880,   /* Wait 37 ms */
    0x800200B0,   /* AFE_CFG: ADC_CONV_EN = 0,
SUPPLY_LPF_EN = 0, TEMP_CONV_EN = 0,
TEMP_SENS_EN = 0 */
    0xA0000300,   /* AFE_ADC_CFG: MUX_SEL =
00000 */
   0x82000002,   /* AFE_SEQ_CFG: SEQ_EN = 0 */
};
```

### *Safety Word*

When using the AFE sequencer application programming interfaces (APIs) in the ADuCM350 software development kit, the first value in the sequence array is the safety word.

The safety word consists of the following:

- The sequence count [31:16]
- A CRC-8 [7:0]

In the previous example, the safety word is 00080081.

#### Sequence Count

The number of commands in a sequence (excluding the safety word) is 8 or 0x0008, located in Bits[31:16].

#### CRC-8 Calculation

Use the CRC-8 tool to generate the CRC for the previous hex commands.

Hexstring = 800210B0A0000103000006408003311B000090880800200B0A000030082000002

CRC-8 → 0x81, located in Bits[7:0].

See Appendix A on how to download the CRC-8 tool.

### *Example Write Command*

Take the following line:

```
0xA0000103,   /* AFE_ADC_CFG:   MUX_SEL =
00011, GAIN_OFFS_SEL = 01 (TS) */
```

When the COMMAND field MSB equals 0, it equals a write. The ADDRESS field corresponds to Bits[7:2] of the 32-bit MMR address. Therefore, for the AFE_ADC_CFG register, the hex is 0x40080040, which is located at Bits[7:2] and equals 0100 00.

The DATA field (AFE_ADC_CFG) then results in the GAIN_OFFS_SEL bits, Bits[9:8] equaling 01 and the MUX_SEL bits, Bits[4:0] equaling [0 0011] (TEMP). Therefore, in binary code, it is 1010 0000 0000 0000 0000 0001 000 0011, or in hex code, it is 0xA0000103.

### *Example Wait Command*

Take the following line

```
0x00000640,   /* Wait 100 µs */
```

ACLK is based on 16 MHz, and therefore, a period of 62.5 ns is required.

100 µs/62.5 ns = 1600 (decimal)

When converted to hex code, it equals 0x640.

Note that the fetch and decode cycles are built in to the wait command.

## APPENDIX A: ONLINE CRC-8 TOOL

For this example, it is assumed that users have the CRC-8 tool called Python installed on their PCs.

Run the CRC-8 tool from the command line. Run the command from the directory where the archive was unzipped (for example, pycrc-0.8.1) and has the following format:

python pycrc.py --width 8 --poly 0x7 --reflect-in False --reflect-out False --xor-in 0x01 --xor-out 0x00 --check-hexstring=<sequence>

where <sequence> is the actual sequence as a single hex number.

### *Example 1. CRC In Sequence*

The following is an example of a CRC-8 calculation in the AFE sequencer.

```
const uint32_t seq_afe_excitechanpowerup[] =
{
    0x0003009C,   /* safety word:
Bits[31:16] = command count, Bits[7:0] =
CRC*/
    0x80020EF0,   /* AFE_CFG: DAC_EN = 1,
TIA_EN = 1, INAMP_EN = 1, BUF_EN = 1 */
    0x00000640,   /* Wait 100 µs */
    0x82000002,   /* AFE_SEQ_CFG: SEQ_EN = 0
*/
};
```

The command line is the following:

python pycrc.py --width 8 --poly 0x7 --reflect-in False --reflect-out False --xor-in 0x01 --xor-out 0x00 --check-hexstring = 80020EF00000064082000002

→ 0x9C

Note that the sequence is written as a 24-digit hex number, all of which is obtained from the concatenation of the sequence hex codes, except for the safety word.

### Example 2. Temperature Sensor Measurement

The following is an example of a temperature sensor measurement using the AFE sequencer.

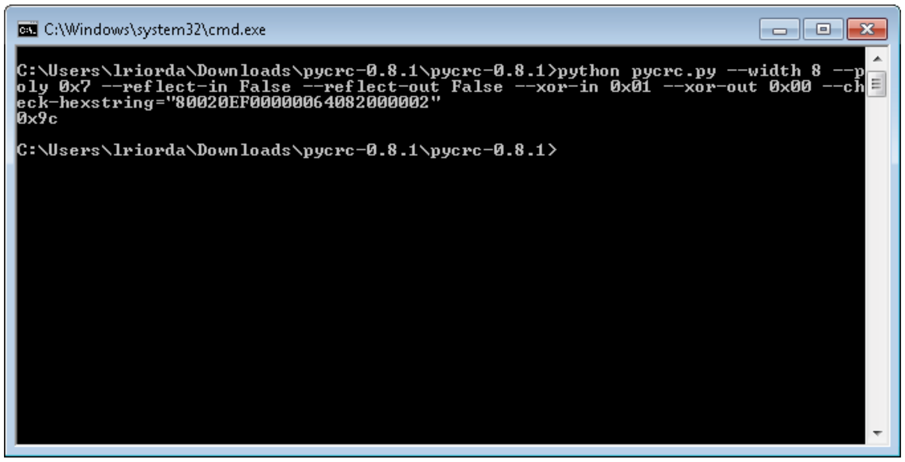
```
const uint32_t seq_afe_tempsensmeas[] = {
    0x00080081,   /* safety word:
Bits[31:16] = command count, Bits[7:0] =
CRC*/
    0x800210B0,   /* AFE_CFG: TEMP_SENS_EN =
1 */
    0xA0000103,   /* AFE_ADC_CFG: MUX_SEL =
00011, GAIN_OFFS_SEL = 01 (TS) */
    0x00000640,   /* Wait 100 µs */
    0x800331B0,   /* AFE_CFG: ADC_CONV_EN = 1,
SUPPLY_LPF_EN = 1, TEMP_CONV_EN = 1 */
    0x00090880,   /* Wait 37 ms */
    0x800200B0,   /* AFE_CFG: ADC_CONV_EN = 0,
SUPPLY_LPF_EN = 0, TEMP_CONV_EN = 0,
TEMP_SENS_EN = 0 */
    0xA0000300,   /* AFE_ADC_CFG: MUX_SEL =
00000 */
   0x82000002,   /* AFE_SEQ_CFG: SEQ_EN = 0 */
};
```

The command line is the following:

python pycrc.py --width 8 --poly 0x7 --reflect-in False --reflect-out False --xor-in 0x01 --xor-out 0x00 --check-hexstring=800210B0A0000103000006408000331B000090880800200B0A000030082000002

→0x81

The pycrc tool also allows the user to write the C code that implements the CRC calculation.



```
C:\Users\lriorda\Downloads\pycrc-0.8.1\pycrc-0.8.1>python pycrc.py --width 8 --p
oly 0x7 --reflect-in False --reflect-out False --xor-in 0x01 --xor-out 0x00 --ch
eck-hexstring="80020EF00000064082000002"
0x9c

C:\Users\lriorda\Downloads\pycrc-0.8.1\pycrc-0.8.1>
```

12097-005

*Figure 5. CRC-8 PYCRC Tool*

## NOTES