AN13819

How to Reach ADC Maximum Conversion Speed on LPC55S16

Rev. 0 — 3 March 2023 Application note

Document information

Information	Content
Keywords	ADC LPC55S16
Abstract	This application note describes how to test maximum ADC conversion speed on LPC55S16 and provides example code to implement tests.

1 Introduction

1.1 Introduction

This application note describes how to test maximum Analog-to-Digital Converter (ADC) conversion speed on LPC55S16 and provides example code to implement tests.

The 16-bits ADC is a successive-approximation ADC designed for operation within an integrated microcontroller system-on-chip. It is available on all LPC55S1x/LPC551x devices.

The main features of ADC module are:

- Linear successive approximation algorithm:
  - Differential operation with 16-bit or 13-bit resolution.
  - Single-ended operation with 16-bit or 12-bit resolution.
  - Support for two simultaneous single-ended conversions or one pair of differential conversion.
- Channel support for up to 10 analog input channels for conversion of external pins:
  - Select external pin inputs paired for conversion as differential channel input.
  - Measurement of on-chip analog sources, temperature sensor, or band gap.
- Configurable speed options to accommodate operation in low-power modes of SoC:
  - Trigger detect with up to 16 trigger sources with priority level configuration. Software or hardware trigger option for each.
  - 15 command buffers allow independent option selection and channel sequence scanning.
  - Automatic compare for less-than, greater-than, within range, or out-of-range with store on true and repeat until true options.
- Interrupt, DMA, or polled operation.

1.2 Test result

The data sheet describes:

- In the 12-bit mode, the maximum conversion speed can reach 2.0 Msps.
- In the 16-bit mode, the maximum conversion speed can reach 2.3 Msps.

12.2 16-bit ADC characteristics [11]

Table 37. 16-bit ADC static characteristics

T_{amb} = −40 °C to +105 °C; V_{DDA} = 1.8 V to 3.6 V; ADC calibrated at T = 25 °C.

Symbol	Parameter	Conditions	Min[2]	Typ[2]	Max[2]	Unit
V_{IA}	analog input voltage		0	-	V_{DDA}	V
CADIN	input capacitance		-	-	5	pF
$f_{clk(ADC)}$	ADC input clock frequency	Low Power Mode (PWRSEL=00)	4		24	MHz
		High Speed Mode (PWRSEL=11)	4		48	MHz
f_s [11]	sampling frequency	12-bit, MODE=0. ADCLK =48MHz, AVG=1,STS=3,PWRSEL=3	-	-	2.3	Msamples/s
		16-bit, MODE=1. ADCLK =48MHz, AVG=1,STS=3,PWRSEL=3	-	-	2.0	Msamples/s
E_D	differential linearity error	16-bit differential mode, CTYPE = 2 [1][2][3][4][5]	-0.99	-	2.6	LSB
		16-bit single ended mode, CTYPE = 1 [1][2][3][4][5]	-1	-	9.5	LSB
$E_{L(adj)}$	integral	16-bit differential mode, CTYPE = 2 [1][2][3][4][6]	-16	-	16	LSB

Figure 1. ADC maximum conversion speed in data sheet

This application note provides example code to verify the conversion speed. The results are as below:

- In the 12-bit mode, the maximum conversion speed can reach 2.083 Msps.
- In the 16-bit mode, the maximum conversion speed can reach 2.326 Msps.

As shown in Figure 1, the tests are aligned with data sheet.

2 Implementation

To reach the maximum conversion speed, use DMA to receive and transfer ADC conversion result. So, the example code is based on SDK example:

\SDK_2_12_0_LPCXpresso55S16\boards\lpcxpresso55s16\driver_examples\lpadc\dma

Make sure you have already been familiar with that example above and related hardware.

1. To reach the maximum conversion speed, configure ADC in the fastest mode, which includes:
   a. ADC input clock: ADCCLK = 48 MHz (maximum allowable ADC clock on LPC55S16).
   b. No hardware average: ADC->CMDH[x].AVGS = 0.
   c. ADC Sample Time Select ((STC set to 3ADC clock): CMDH[x].STS = 0.
   d. ADC maximum power: ADC -> CFG. PWRSEL = 3
   e. ADC set to continue conversion mode, which means that after the conversion command execution complete, the next conversion command automatically starts.

   Corresponding SDK code:

```
/* Configure ADC. */
LPADC_GetDefaultConfig(&lpadcConfigStruct);
lpadcConfigStruct.enableAnalogPreliminary = true;
lpadcConfigStruct.conversionAverageMode = kLPADC_ConversionAverage1;
lpadcConfigStruct.powerLevelMode=kLPADC_PowerLevelAlt4;
lpadcConfigStruct.referenceVoltageSource = DEMO_LPADC_VREF_SOURCE;
lpadcConfigStruct.FIFO0Watermark = 2;
```

```
/* Set conversion CMD configuration. */
LPADC_GetDefaultConvCommandConfig(&g_LpadcCommandConfigStruct);
g_LpadcCommandConfigStruct.channelNumber = DEMO_LPADC_USER_CHANNEL;
g_LpadcCommandConfigStruct.sampleTimeMode = kLPADC_SampleTimeADCK3;
g_LpadcCommandConfigStruct.loopCount = 1;
g_LpadcCommandConfigStruct.hardwareAverageMode = kLPADC_HardwareAverageCount1;
g_LpadcCommandConfigStruct.conversionResolutionMode = kLPADC_ConversionResolutionHigh;
//  g_LpadcCommandConfigStruct.conversionResolutionMode = kLPADC_ConversionResolutionStandard;

g_LpadcCommandConfigStruct.chainedNextCommandNumber = DEMO_LPADC_USER_CMDID;
```

2. Configure the DMA module:
   The DMA module is working in the Ping-Pong mode, which means that two DMA descriptors are created and used: Descriptor A and Descriptor B.
   When Descriptor A transfer exhausts, DMA automatically starts Descriptor B transfer. When Descriptor B transfer exhausts, DMA automatically starts Descriptor A transfer.
   Note:
   a. *Set the DMA destination increment to **kDMA_AddressInterleave1xWidth**, which means that after each transfer, the destination address moves to the next buffer position.*
   b. *To enable continuing DMA transfer, set `clrTrig` to be true in `g_XferConfig` structure. Otherwise, the DMA stops transfer after one cycle.*

   Corresponding SDK code:

```
SDK_ALIGN(uint32_t s_dma_table[DMA_DESCRIPTOR_NUM * sizeof(dma_descriptor_t)],
    FSL_FEATURE_DMA_LINK_DESCRIPTOR_ALIGN_SIZE);

const uint32_t g_XferConfig =
    DMA_CHANNEL_XFER(true,                          /* Reload link descriptor after current exhaust, */
                     true,                          /* Clear trigger status. */
                     true,                          /* Enable interruptA. */
                     false,                         /* Not enable interruptB. */
                     sizeof(uint32_t),             /* Dma transfer width. */
                     kDMA_AddressInterleave0xWidth, /* Dma source address no interleave  */
                     kDMA_AddressInterleave1xWidth, /* Dma destination address no interleave  */
                     sizeof(uint32_t)*ADC_DMA_SIZE              /* Dma transfer byte. */
    );
```

```
static void DMA_Configuration(void)
{
    dma_channel_config_t dmaChannelConfigStruct;

#if defined(DEMO_DMA_HARDWARE_TRIGGER) && DEMO_DMA_HARDWARE_TRIGGER
    /* Configure INPUTMUX. */
    INPUTMUX_Init(DEMO_INPUTMUX_BASE);
    INPUTMUX_AttachSignal(DEMO_INPUTMUX_BASE, DEMO_DMA_ADC_CHANNEL, DEMO_DMA_ADC_CONNECTION);
#endif /* DEMO_DMA_HARDWARE_TRIGGER */

    /* Configure DMA. */
    DMA_Init(DEMO_DMA_BASE);
    DMA_EnableChannel(DEMO_DMA_BASE, DEMO_DMA_ADC_CHANNEL);
    DMA_CreateHandle(&g_DmaHandleStruct, DEMO_DMA_BASE, DEMO_DMA_ADC_CHANNEL);
    DMA_SetCallback(&g_DmaHandleStruct, DEMO_DMA_Callback, NULL);

    /* Prepare and submit the transfer. */
    DMA_PrepareChannelTransfer(&dmaChannelConfigStruct,             /* DMA channel transfer configuration structure. */
                               (void *)DEMO_LPADC_RESFIFO_REG_ADDR,  /* DMA transfer source address. */
                               (void *)adc_result,              /* DMA transfer destination address. */
                               g_XferConfig,                        /* Xfer configuration */
                               kDMA_PeripheralToMemory,              /* DMA transfer type. */
                               NULL,                                /* DMA channel trigger configurations. */
                               (dma_descriptor_t *)&(s_dma_table[0]) /* Address of next descriptor. */
    );
    DMA_SubmitChannelTransfer(&g_DmaHandleStruct, &dmaChannelConfigStruct);

    /* Set two DMA descripters to use ping-pong mode.  */
    DMA_SetupDescriptor((dma_descriptor_t *)&(s_dma_table[0]), g_XferConfig, (void *)DEMO_LPADC_RESFIFO_REG_ADDR, (void
*)adc_result, (dma_descriptor_t *)&(s_dma_table[4]));
    DMA_SetupDescriptor((dma_descriptor_t *)&(s_dma_table[4]), g_XferConfig, (void *)DEMO_LPADC_RESFIFO_REG_ADDR, (void
*)adc_result, (dma_descriptor_t *)&(s_dma_table[0]));
}
```

3. The example code also enables SysTick timer. It is used as precise timer counter to calculate ADC conversion speed. Besides, basic statistics are also calculated (mean and standard deviation) for user to evaluate ADC performance.

2.1 Running the demo

1. Modify the *\driver_examples\lpadc\dma* example code based on previous description. Compile and download to code. Open UART terminal with 115200-N-8-N-1. Press the RESET button to run the code. Then the welcome log displays as below:
   LPADC DMA Example
   ADC CLK: 48000000
   CORE CLK: 150000000
   ADC Full Range: 65536
   ADCResolution: 16bit
2. Connect to function generator or any other ADC input source (Figure 2 shows the `P0_23` position).

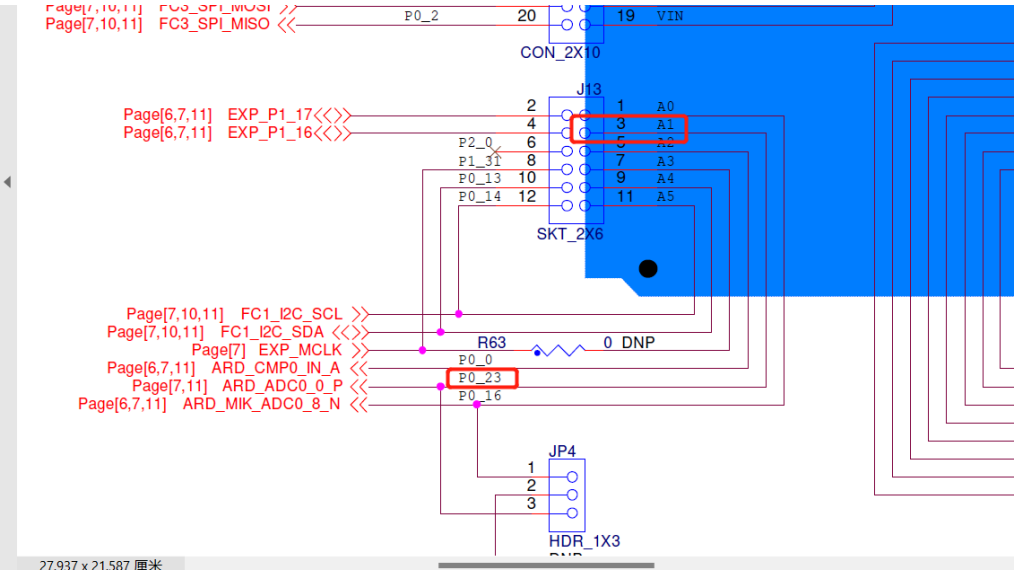

Figure 2. P0_23(ADC0_0, J13-3) used in this example in LPC55S16_EVK

3. Press any key to start ADC conversion test. The program automatically calculates conversion time and speed. The result is displayed on the UART terminal:
   TIME: 48us(2.083MS/s)
   AVG: 10984.940000
   STD: 65.317379

3 Conclusion

This application note provides example code to test ADC maximum conversion speed. The result shows that in both 12-bit and 16-bit mode, the ADC maximum conversion speed can reach specifications in data sheet.

4 Note About the Source Code in the Document

Example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2023 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN

ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

5 Revision history

Rev.	Date	Description
0	3 March 2023	Initial release

6 Legal information

6.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

6.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

6.3 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Contents

1 Introduction 2
1.1 Introduction 2
1.2 Test result 2
2 Implementation 3
2.1 Running the demo 4
3 Conclusion 5
4 Note About the Source Code in the Document 5
5 Revision history 6
6 Legal information 7

Date of release: 3 March 2023
Document identifier: AN13819