AN12942

RT600 ADC Temperature Sensor Usage

Rev. 0 — August 2020 Application Note

Contents

1 Introduction

This application note provides an overview of how to use RT600 SDK software with EVK hardware to validate the RT600 built-in temperature sensor using the ADC controller. The SDK lpadc sample application is used as an example setup. Channel 7 is used to provide a sample voltage signal input to the ADC. While running the project, typing any key in the debug console triggers the conversion. ADC watermark interrupt is asserted when the number of data words stored in the ADC results FIFO is greater than the watermark value. In ADC ISR, the watermark flag is cleared when the conversion result value is read. Also, the result is printed when the execution returns to the main function.

2 Temperature sensor overview

The temperature sensor transducer uses an intrinsic pn-junction diode reference and outputs a Complement To Absolute Temperature (CTAT) voltage. The temperature sensor is linear with a slight curvature. The output voltage is measured over different ranges of temperatures and fit with linear-least-square lines. After power-up, the temperature sensor output must be allowed to settle to its stable value before it is used as an accurate ADC input.

2.1 Characteristics

VDDA_BIAS = 3.3 V, All other supplies = 1.8 V

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
DT_{sen}	sensor temperature accuracy	T_{amb} = −20 °C to +70 °C	[1]	-		2.77	°C
E_L	linearity error	T_{amb} = −20 °C to +70 °C	[1]	-	-	2.79	°C

[1] Absolute temperature accuracy. Based on characterization. Not tested in production.

Figure 1. Temperature sensor static and dynamic characteristics

2.2 Connection between ADC channel input and analog output

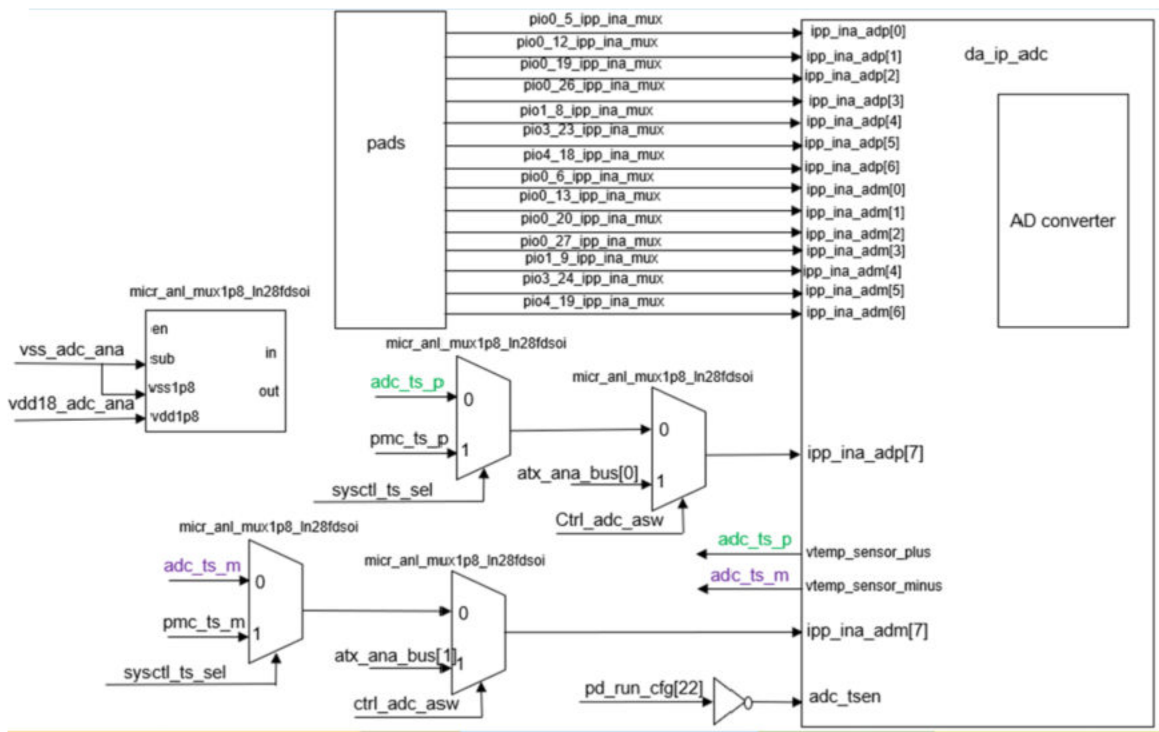

Figure 2. ADC channel input and analog output

2.3 Registers needed for ADC temperature sensor

2.3.1 Temperature sensor control (SYSCTL0_TEMPSENSORCTL)

This register enables the on-chip temperature sensor to be measured by the ADC.

Table 213. Temp sensor control (SYSCTL0_TEMPSENSORCTL: offset = 0xE0C)

Bit	Symbol	Value	Description	Reset value
0	TSSRC		Temperature Sensor Source:	0x0
		0	ADC Built-in Temperature Sensor.	
		1	Reserved.	
31:1	-	-	Reserved	-

NOTE

This register is not available as of now in RT600 SDK and will be available from SDK 2.9.0 onwards. By default, the on-chip temperature sensor is selected, therefore no changes are required for SDK 2.7.0 or lower versions.

2.3.2 Run configuration register 1 clear (SYSCTL0_PDRUNCFG1_CLR)

Writing a 1 to a bit position in this register clears the corresponding position in PDRUNCFG1. This is a write-only register.

Table 200. Run configuration register 0 clear (SYSCTL0_PDRUNCFG0_CLR: offset = 0x630) *...continued*

Bit	Symbol	Value	Description	Reset value
20	AUDPLLANA_PD		Audio PLL analog functions	-
		0	No effect	
		1	Clears the PDRUNCFG0 Bit	
21	ADC_PD		ADC analog functions	-
		0	No effect	
		1	Clears the PDRUNCFG0 Bit	
22	ADC_LP		ADC low power mode	-
		0	No effect	
		1	Clears the PDRUNCFG0 Bit	
23	ADCTEMPSNS_PD		ADC temperature sensor	-
		0	No effect	
		1	Clears the PDRUNCFG0 Bit	

3 Demo application

3.1 Environment

3.1.1 Hardware environment

- Board
  - — MIMXRT685EVK
- Debugger
  - — Integrated CMSIS-DAP debugger on the board
- Miscellaneous
  - — 1 Micro USB cable
  - — PC
- Board Setup
  - — Set VREF_L to GND, VREF_H to 1.8 V by connecting jumpers JP9, JP10, and J25 as shown in the following figure.

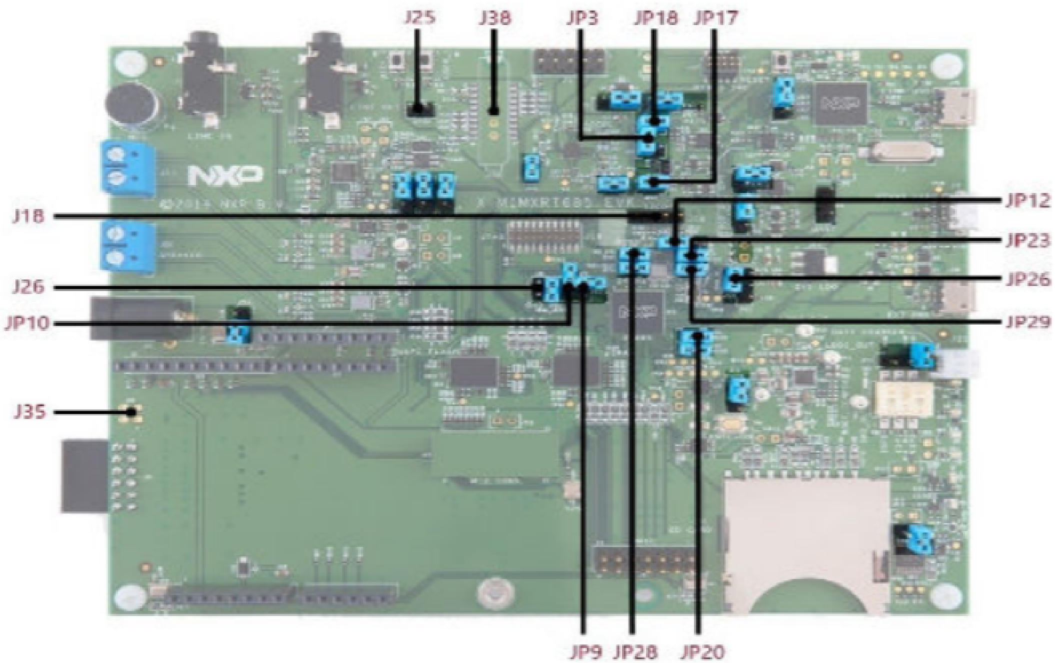

Figure 3. Board jumper setting

3.1.2 Software environment

- Tool chain
  - IAR embedded workbench 8.50.1 or MCUXpresso IDE v11.1.1 or Keil 5.29
- Software package
  - SDK_2.7.0_EVK-MIMXRT685

3.2 Steps and result

1. Follow the Getting Started with MCUXpresso SDK for MIMXRT600 (available inside SDK➜docs) to go through the steps for opening lpadc_interrupt project (SDK\boards\evkmimxrt685\driver_examples\lpadc\interrupt).
2. Open the file lpadc_interrupt.c (lpadc_interrupt\source) and modify as follows:

   The ADC temperature sensor is internally connected to channel 7 as shown in Figure 2. Set the channel to 7 by modifying the macro `DEMO_LPADC_USER_CHANNEL`.

```
#define DEMO_LPADC_USER_CHANNEL 7U /* temperature sensor */
```

3. Open clock_config.c (lpadc_interrupt\source) and in the function `BOARD_BootClockRUN()`:

   Program other system registers, group 0 (SYSTCTL0) to enable the ADC temperature sensor on the ADC analog switch. Then enable it by clearing SYSCTL0_PDRUNCFG0.

```
//SYSCTL0->TEMPSENSORCTL = 0; /*enable the ADC built-in temperature sensor on ADC analog switch
*/
```

NOTE

The TEMPSENSORCTL register is not available as of now in RT600 SDK and will be available from SDK 2.9.0 onwards. By default, the on-chip temperature sensor is selected and no changes required for SDK 2.7.0 or lower versions. Therefore, the above code is commented out as of now.

```
SYSCTL0->PDRUNCFG0_CLR = SYSCTL0_PDRUNCFG0_ADCTEMPSNS_PD_MASK; //enable
ADCTEMPSNS_PD
```

4. Open fsl_lpadc.c (lpadc_interrupt\drivers), the STS bits of CMDH1 register are set to 7. So, the sample time is 131 (3 + 2STS) ADCK cycles since a sample time of over 35 is required. A long sample time allows higher impedance inputs to be accurately sampled. This is done by modifying the `smapleTimeMode` which is member of conversion commands configuration structure inside `LPADC_GetDefaultConvCommandConfig` function.

```
config->sampleTimeMode = kLPADC_SampleTimeADCK131;
```

5. Follow the Getting Started with MCUXpresso SDK for MIMXRT600 (available inside SDK➜docs) to go through the steps for building and running lpadc_interrupt demo.
6. When running the demo, type any key in the debug console which triggers the conversion.
7. Results:

   During the test, the heat gun is used to heat the RT600 chip. The following interrupt count intervals show when the heater is OFF or ON:

   ADC interrupt count 1 – 34 (heater is OFF)

   ADC interrupt count 35 – 80 (heater is ON)

   ADC interrupt count 81 – 100 (heater is OFF)

Actual LPADC interrupt example output:

LPADC interrupt example

ADC Full Range: 4096

Full channel scale (Factor of 1).

Press any key to get user channels ADC value.

```
ADC value: 1757

ADC interrupt count: 1

ADC value: 1756

ADC interrupt count: 2

ADC value: 1754

ADC interrupt count: 3

ADC value: 1756

ADC interrupt count: 4

ADC value: 1755

ADC interrupt count: 5

ADC value: 1757

ADC interrupt count: 6

ADC value: 1756
```

```
ADC interrupt count: 7

ADC value: 1759

ADC interrupt count: 8

ADC value: 1757

ADC interrupt count: 9

ADC value: 1758

ADC interrupt count: 10

ADC value: 1754

ADC interrupt count: 11

ADC value: 1757

ADC interrupt count: 12

ADC value: 1756

ADC interrupt count: 13

ADC value: 1755

ADC interrupt count: 14

ADC value: 1756

ADC interrupt count: 15

ADC value: 1758

ADC interrupt count: 16

ADC value: 1755

ADC interrupt count: 17

ADC value: 1757

ADC interrupt count: 18

ADC value: 1758

ADC interrupt count: 19

ADC value: 1755

ADC interrupt count: 20

ADC value: 1758

ADC interrupt count: 21

ADC value: 1754

ADC interrupt count: 22

ADC value: 1756

ADC interrupt count: 23

ADC value: 1755

ADC interrupt count: 24

ADC value: 1758

ADC interrupt count: 25

ADC value: 1758
```

```
ADC interrupt count: 26

ADC value: 1754

ADC interrupt count: 27

ADC value: 1754

ADC interrupt count: 28

ADC value: 1759

ADC interrupt count: 29

ADC value: 1755

ADC interrupt count: 30

ADC value: 1756

ADC interrupt count: 31

ADC value: 1755

ADC interrupt count: 32

ADC value: 1755

ADC interrupt count: 33

ADC value: 1756

ADC interrupt count: 34

ADC value: 1737

ADC interrupt count: 35

ADC value: 1736

ADC interrupt count: 36

ADC value: 1732

ADC interrupt count: 37

ADC value: 1733

ADC interrupt count: 38

ADC value: 1731

ADC interrupt count: 39

ADC value: 1731

ADC interrupt count: 40

ADC value: 1728

ADC interrupt count: 41

ADC value: 1726

ADC interrupt count: 42

ADC value: 1722

ADC interrupt count: 43

ADC value: 1725

ADC interrupt count: 44

ADC value: 1720
```

```
ADC interrupt count: 45

ADC value: 1722

ADC interrupt count: 46

ADC value: 1722

ADC interrupt count: 47

ADC value: 1721

ADC interrupt count: 48

ADC value: 1716

ADC interrupt count: 49

ADC value: 1717

ADC interrupt count: 50

ADC value: 1718

ADC interrupt count: 51

ADC value: 1721

ADC interrupt count: 52

ADC value: 1717

ADC interrupt count: 53

ADC value: 1719

ADC interrupt count: 54

ADC value: 1716

ADC interrupt count: 55

ADC value: 1717

ADC interrupt count: 56

ADC value: 1717

ADC interrupt count: 57

ADC value: 1716

ADC interrupt count: 58

ADC value: 1719

ADC interrupt count: 59

ADC value: 1717

ADC interrupt count: 60

ADC value: 1716

ADC interrupt count: 61

ADC value: 1712

ADC interrupt count: 62

ADC value: 1715

ADC interrupt count: 63

ADC value: 1713
```

```
ADC interrupt count: 64

ADC value: 1713

ADC interrupt count: 65

ADC value: 1711

ADC interrupt count: 66

ADC value: 1716

ADC interrupt count: 67

ADC value: 1714

ADC interrupt count: 68

ADC value: 1710

ADC interrupt count: 69

ADC value: 1714

ADC interrupt count: 70

ADC value: 1712

ADC interrupt count: 71

ADC value: 1711

ADC interrupt count: 72

ADC value: 1713

ADC interrupt count: 73

ADC value: 1711

ADC interrupt count: 74

ADC value: 1710

ADC interrupt count: 75

ADC value: 1712

ADC interrupt count: 76

ADC value: 1710

ADC interrupt count: 77

ADC value: 1712

ADC interrupt count: 78

ADC value: 1712

ADC interrupt count: 79

ADC value: 1711

ADC interrupt count: 80

ADC value: 1749

ADC interrupt count: 81

ADC value: 1749

ADC interrupt count: 82

ADC value: 1744
```

```
ADC interrupt count: 83

ADC value: 1749

ADC interrupt count: 84

ADC value: 1748

ADC interrupt count: 85

ADC value: 1748

ADC interrupt count: 86

ADC value: 1746

ADC interrupt count: 87

ADC value: 1746

ADC interrupt count: 88

ADC value: 1745

ADC interrupt count: 89

ADC value: 1750

ADC interrupt count: 90

ADC value: 1748

ADC interrupt count: 91

ADC value: 1745

ADC interrupt count: 92

ADC value: 1746

ADC interrupt count: 93

ADC value: 1746

ADC interrupt count: 94

ADC value: 1745

ADC interrupt count: 95

ADC value: 1747

ADC interrupt count: 96

ADC value: 1746

ADC interrupt count: 97

ADC value: 1745

ADC interrupt count: 98

ADC value: 1749

ADC interrupt count: 99

ADC value: 1750

ADC interrupt count: 100
```

The ADC output decreases with rise in temperature, as it measures the VBE of the diode plotted over PVTs versus temperature, see the following figure.

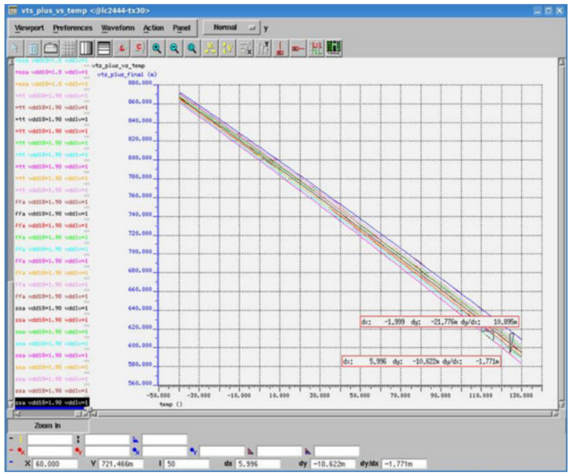

Figure 4. VTS_Plus versus temperature plot

4 Conclusion

The example shows how the SDK software with EVK hardware can be used to validate RT600 built-in temperature sensor using the ADC controller. It also shows, how the built-in temperature sensor can be used to indicate temperature value based on ADC output value.

5 References

1. RT600 User Manual
2. RT600 Data Sheet
3. MCUXpresso SDK Release Notes for EVK-MIMXRT685 (available inside SDK)
4. Getting Started with MCUXpresso SDK for EVK-MIMXRT685 (available inside SDK)

6 Revision history

Table 1. Revision history

Revision number	Date	Substantive changes
0	08/2020	Initial release

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: August 2020
Document identifier: AN12942