

**Core self-test library compliant with IEC 60730, IEC 60335 UL 60730,
UL 1998 documentation**

Analog Input/Output Test

Document revision history

Date	Author	Version	Notes
11/2015	Jozef Sedlak	0.1	Initial version
11/2015	Jozef Sedlak	1.0	Version for certification
10/2016	Jozef Sedlak	1.1	NXP, Test for MKE1xF devices added
11/2018	Jozef Sedlak	3.0	Release with new compilers supported.

1	ANALOG INPUT/OUTPUT TEST ARCHITECTURE.....	4
2	ANALOG INPUT/OUTPUT TEST IN COMPLIANCE WITH IEC/UL STANDARDS	5
3	ANALOG INPUT/OUTPUT TEST IMPLEMENTATION	6
3.1	IEC60730B_CM4_CM7_AIO_INPUTINIT_CYCLIC()	10
3.3	IEC60730B_CM4_CM7_AIO_INPUTTRIGGER()	10
3.4	IEC60730B_CM4_CM7_AIO_INPUTSET()	11
3.5	IEC60730B_CM4_CM7_AIO_INPUTSET_K3s()	12
3.6	IEC60730B_CM4_CM7_AIO_INPUTSET_CYCLIC()	13
3.7	IEC60730B_CM4_CM7_AIO_INPUTCHECK()	14
3.8	IEC60730B_CM4_CM7_AIO_INPUTCHECK_KE()	15
3.9	IEC60730B_CM4_CM7_AIO_INPUTCHECK_K3s()	16
3.10	IEC60730B_CM4_CM7_AIO_INPUTCHECK_CYCLIC()	17

1 Analog Input/Output Test Architecture

The Analog IO test procedure performs the plausibility check of the digital IO interface of the processor. The Analog IO test can be performed once after the microcontroller reset and also during the runtime.

The identification of a safety error is ensured by the specific FAIL return in the case of an Analog IO error. The application developer must compare the return value of the test function with the expected value. If this is equal to the FAIL return, then the jump into safety error handling function must occur. The safety error handling function may be specific according to the application and is not a part of our library. The main purpose of this function is to put the application into a safety state.

The principle of the Analog IO test is based on sequence execution, where a certain analog level is connected to a defined analog input. The test function checks whether the converted value is in tolerance. The test covers the check of the analog input interface and checks three reference values: reference high, reference low, and bandgap voltage. See the device specification document to setup the correct values. Test functions covers these ADC types: 16-bit SAR (MKV3x, MKE1xF), 12-bit LPADC (MK32W0x), 12-bit cyclic (MKV4x) .

The block diagram for the Analog IO test is shown in the following figure:

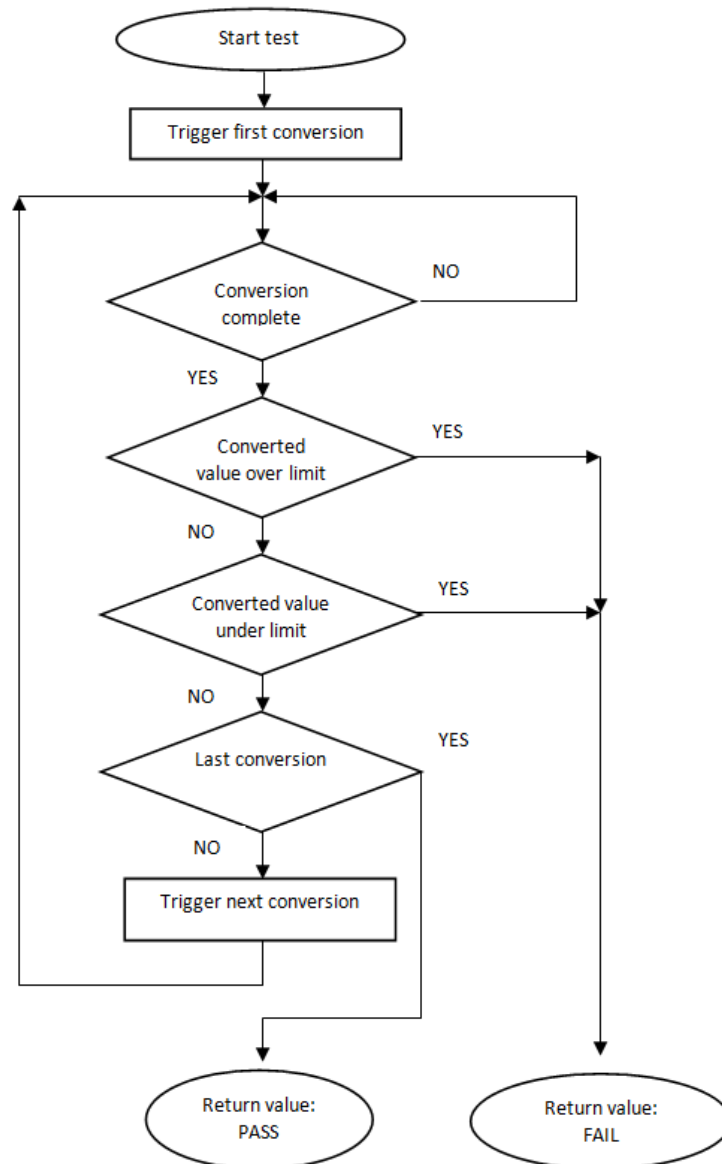


Figure 1. Block diagram for Analog input test

2 Analog Input/Output Test in compliance with IEC/UL standards

The performed overload test fulfils safety requirements according to IEC 60730-1, IEC 60335, UL 60730, and UL 1998 standards as described in the following table:

Table 1. Analog Input/Output Test in Compliance with the Standards

Test	Component	Fault/Error	SW / HW Class	Acceptable Measures
Input/Output periphery	7. Input/Output periphery (7.2 – Analog I/O)	Abnormal operation	B/R.1	Plausibility check

3 Analog Input/Output Test Implementation

Test functions for the Analog IO test are placed in *IEC60730_B_CM4_CM7_aio.c* and are written as c functions. The header file with the function prototypes is *IEC60730_B_CM4_CM7_aio.h*.

IEC60730_B_CM4_CM7.h is the common header file for safety library.

The following functions are called to test the Analog input:

- *IEC60730B_CM4_CM7_AIO_InputInit ()*
- *IEC60730B_CM4_CM7_AIO_InputInit_Cyclic()*

- *IEC60730B_CM4_CM7_AIO_InputTrigger()*

- *IEC60730B_CM4_CM7_AIO_InputSet ()*
- *IEC60730B_CM4_CM7_AIO_InputSet_k3s()*
- *IEC60730B_CM4_CM7_AIO_InputSet_Cyclic()*

- *IEC60730B_CM4_CM7_AIO_InputCheck()*
- *IEC60730B_CM4_CM7_AIO_InputCheck_ke()*
- *IEC60730B_CM4_CM7_AIO_InputCheck_k3s()*
- *IEC60730B_CM4_CM7_AIO_InputCheck_Cyclic()*

The principle of the analog input test is based on a conversion of analog inputs with known voltage values and the test checks if the converted values fit into the limits defined by user. Normally it should be about 10 percent around the desired reference values. The test is triggered by the function *IEC60730B_CM4_CM7_AIO_InputTrigger()*. The test is divided into 3 parts: the initialization, the test execution, and the end of the test. ADC modules implemented on supported devices has some differences. Therefore if needed, dedicated functions are introduced (see the list above).

Below is an example of the function calling.

1. Configuration of parameters

```
#define TESTED_ADC ADC0
#define ADC_RESOLUTION 12
#define ADC_MAX ((1<<(ADC_RESOLUTION))-1)
#define ADC_REFERENCE 3.06
#define ADC_BANDGAP_LEVEL 1.0
#define ADC_BANDGAP_LEVEL_RAW (((ADC_BANDGAP_LEVEL)*(ADC_MAX))/(ADC_REFERENCE))
#define ADC_DEVIATION_PERCENT 10
#define ADC_MIN_LIMIT(val) (((val) * (100 - ADC_DEVIATION_PERCENT)) / 100)
#define ADC_MAX_LIMIT(val) (((val) * (100 + ADC_DEVIATION_PERCENT)) / 100)
#define IEC60730B_CFG_AIO_CHANNELS_CNT 3
#define IEC60730B_CFG_AIO_CHANNELS_LIMITS_INIT \
{\
    {ADC_MIN_LIMIT(0), ADC_MAX_LIMIT(10)}, \
    {ADC_MIN_LIMIT(ADC_MAX), ADC_MAX_LIMIT(ADC_MAX)}, \
    {ADC_MIN_LIMIT(ADC_BANDGAP_LEVEL_RAW), ADC_MAX_LIMIT(ADC_BANDGAP_LEVEL_RAW)}\
}
#define IEC60730B_CFG_AIO_CHANNELS_INIT {30, 29, 27}
#define IEC60730B_CFG_AIO_SAMPLES_INIT {2}
#define ADC_COMMAND_BUFFER 2
```

2. Variables definition

```
IEC60730B_CM4_CM7_AIO_TEST_STR aio_Str;
const IEC60730B_CM4_CM7_AIO_LIMITS_STR \
IEC60730B_ADC_Limits[IEC60730B_CFG_AIO_CHANNELS_CNT] = \
IEC60730B_CFG_AIO_CHANNELS_LIMITS_INIT;

unsigned char IEC60730B_ADC_inputs[IEC60730B_CFG_AIO_CHANNELS_CNT] = \
IEC60730B_CFG_AIO_CHANNELS_INIT;

unsigned char IEC60730B_ADC_samples[IEC60730B_CFG_AIO_CHANNELS_CNT] = \
IEC60730B_CFG_AIO_SAMPLES_INIT;
```

3. Initialization of the test

```
IEC60730B_CM4_CM7_AIO_InputInit(&aio_Str, \
(IEC60730B_CM4_CM7_AIO_LIMITS_STR*)IEC60730B_ADC_Limits, (unsigned \
char*)IEC60730B_ADC_inputs, IEC60730B_CFG_AIO_CHANNELS_CNT, ADC_COMMAND_BUFFER);

IEC60730B_CM4_CM7_AIO_InputTrigger(&aio_Str);
```

4. The test

```
for(uint8_t i=0;i<4;i++)
{
    psSafetyCommon->IEC60730B_aio_test_result = IEC60730B_CM4_CM7_AIO_InputCheck(&aio_Str, \
    (unsigned long *)TESTED_ADC);

    switch(psSafetyCommon->IEC60730B_aio_test_result)
    {
    case IEC60730B_ST_AIO_START:
        IEC60730B_CM4_CM7_AIO_InputSet(&aio_Str, (uint32_t*)TESTED_ADC);
        break;
    case IEC60730B_ST_AIO_FAIL:
        psSafetyCommon->ui32SafetyErrors |= AIO_TEST_ERROR;
        SafetyErrorHandling(psSafetyCommon);
        break;
    case IEC60730B_ST_AIO_INIT:
        IEC60730B_CM4_CM7_AIO_InputTrigger(&aio_Str);
        break;
    case IEC60730B_ST_AIO_PASS:
        IEC60730B_CM4_CM7_AIO_InputTrigger(&aio_Str);
        break;
    default:
        __asm("NOP");
        break;
    }
}
```


3.1 IEC60730B_CM4_CM7_AIO_InputInit()

This function initializes one instance of the Analog input test.

Function prototype:

```
void IEC60730B_CM4_CM7_AIO_InputInit(IEC60730B_AIO_TEST_STR* pObj,  
IEC60730B_AIO_LIMITS_STR* pLimits, unsigned char* pInputs, unsigned char cntMax , unsigned char  
cmdBuffer);
```

Function inputs:

pObj -pointer to the analog test instance
pLimits -pointer to the array of limits used in the test
pInputs -pointer to the array of input numbers used in the test
cntMax -size of the input and the limits arrays.
cmdBuffer - specify the command buffer for conversion. Has effect only in case of MK32W0x devices.
 - for MK32W0x - must be between 1 to 15
 - for other devices various 8-bit number

Function output:

Void. The function does not return a value.

Function performance:

The function duration is approximately 36 cycles (0.45 μ s)¹

Function size is 26 bytes.²

3.1 IEC60730B_CM4_CM7_AIO_InputInit_Cyclic()

This function initializes one instance of the Analog input test. It is dedicated for KV4x devices.

Function prototype:

```
void IEC60730B_CM4_CM7_AIO_InputInit_Cyclic(IEC60730B_AIO_TEST_STR* pObj,  
IEC60730B_AIO_LIMITS_STR* pLimits, unsigned char* pInputs, unsigned char* pSamples, unsigned char  
cntMax);
```

Function inputs:

pObj - pointer to the analog test instance
pLimits - pointer to the array of limits used in the test
pInputs - pointer to the array of input numbers used in the test
pSamples - pointer to the array of sample number used in the test
cntMax - size of the input and the limits arrays.

Function output:

Void. The function does not return a value.

Function performance:

The function duration is approximately 35 cycles (0.44 μ s)³

Function size is 22 bytes.²

3.3 IEC60730B_CM4_CM7_AIO_InputTrigger()

This function sets up the analog input test to start the execution of the test.

Function prototype:

```
void IEC60730B_CM4_CM7_AIO_InputTrigger(IEC60730B_AIO_TEST_STR* pObj);
```

Function inputs:

pObj –pointer to the analog test instance.

Function output:

Void. The function does not return a value.

Function performance:

The function duration is approximately 18 cycles (0.23 μ s)¹

The function size is 24 bytes.²

3.4 IEC60730B_CM4_CM7_AIO_InputSet()

This function executes the first part of the AIO test sequence. This part sets up the ADC input channel. When the ADC converter is configured for SW trigger, this function also triggers the conversion. This function can be called when the ADC module is idle and ready for the next conversion.

Function prototype:

```
IEC60730B_RESULT IEC60730B_CM4_CM7_AIO_InputSet(IEC60730B_AIO_TEST_STR* pObj, ADC_Type* pAdc);
```

Function inputs:

pObj - pointer to the analog test instance.

pAdc - pointer to the base address of ADC module.

Function output:

The function returns a value of uint32_t data type. It can have the following values:

- IEC60730B_ST_AIO_FAIL (0x00000701)
- IEC60730B_ST_AIO_PASS (0x00000000)
- IEC60730B_ST_AIO_PROGRESS (0x00000702)
- IEC60730B_ST_AIO_INIT (0x00000704)

Function performance:

The function duration is approximately 57 cycles. (0.71 μ s)¹

The function size is 48 bytes.²

3.5 IEC60730B_CM4_CM7_AIO_InputSet_k3s()

Dedicated for MK32W0x devices, this function executes the first part of the AIO test sequence. This part sets up the ADC input channel. When the ADC converter is configured for SW trigger, this function also triggers the conversion. This function can be called when the ADC module is idle and ready for the next conversion.

Function prototype:

```
IEC60730B_RESULT IEC60730B_CM4_CM7_AIO_InputSet_k3s(IEC60730B_AIO_TEST_STR* pObj,  
ADC_Type* pAdc);
```

Function inputs:

pObj - pointer to the analog test instance.

pAdc - pointer to the analog converter.

Function output:

The function returns a value of unsigned long data type. It can have the following values:

- IEC60730B_ST_AIO_FAIL (0x00000701)
- IEC60730B_ST_AIO_PASS (0x00000000)
- IEC60730B_ST_AIO_PROGRESS (0x00000702)
- IEC60730B_ST_AIO_INIT (0x00000704)

Function performance:

The function duration is approximately 60 cycles. (1.25 μ s)⁴

The function size is 84 bytes.²

3.6 IEC60730B_CM4_CM7_AIO_InputSet_Cyclic()

Dedicated for MKV4x devices, this function executes the first part of the AIO test sequence. This part sets up the ADC input channel. When the ADC converter is configured for SW trigger, this function also triggers the conversion. This function can be called when the ADC module is idle and ready for the next conversion.

Function prototype:

```
IEC60730B_RESULT IEC60730B_CM4_CM7_AIO_InputSet_Cyclic(IEC60730B_AIO_TEST_STR* pObj,  
ADC_Type* pAdc);
```

Function inputs:

pObj - pointer to the analog test instance.

pAdc - pointer to the analog converter.

Function output:

The function returns a value of unsigned long data type. It can have the following values:

- IEC60730B_ST_AIO_FAIL (0x00000701)
- IEC60730B_ST_AIO_PASS (0x00000000)
- IEC60730B_ST_AIO_PROGRESS (0x00000702)
- IEC60730B_ST_AIO_INIT (0x00000704)

Function performance:

The function duration is approximately 97 cycles. (1.21 μ s)³

The function size is 114 bytes.²

3.7 IEC60730B_CM4_CM7_AIO_InputCheck()

This function executes the second part of the AIO test sequence. This part reads the converted analog value and checks if the value fits into the predefined limits. The test is finished, when this function reports IEC60730B_AIO_PASS or IEC60730B_AIO_FAILED.

Function prototype:

```
IEC60730B_RESULT IEC60730B_CM4_CM7_AIO_InputCheck(IEC60730B_AIO_TEST_STR* pObj,  
ADC_Type* pAdc);
```

Function inputs:

pObj - pointer to the analog test instance.

pAdc- pointer to the analog converter.

Function output:

The function returns a value of uint32_t data type. It can have the following values:

- IEC60730B_ST_AIO_FAIL (0x00000701)
- IEC60730B_ST_AIO_PASS (0x00000000)
- IEC60730B_ST_AIO_PROGRESS (0x00000702)
- IEC60730B_ST_AIO_START (0x00000703)
- IEC60730B_ST_AIO_INIT (0x00000704)

Function performance:

The function duration is up to 50 cycles. (0.63 μ s)¹

The function size is 112 bytes.²

3.8 IEC60730B_CM4_CM7_AIO_InputCheck_ke()

Dedicated for MKE1xF devices. This function executes the second part of the AIO test sequence. This part reads the converted analog value and checks if the value fits into the predefined limits. The test is finished, when this function reports IEC60730B_AIO_PASS or IEC60730B_AIO_FAILED.

Function prototype:

```
IEC60730B_RESULT IEC60730B_CM4_CM7_AIO_InputCheck_ke(IEC60730B_AIO_TEST_STR* pObj,  
ADC_Type* pAdc);
```

Function inputs:

pObj - pointer to the analog test instance.

pAdc - pointer to the analog converter.

Function output:

The function returns a value of unsigned long data type. It can have the following values:

- IEC60730B_ST_AIO_FAIL (0x00000701)
- IEC60730B_ST_AIO_PASS (0x00000000)
- IEC60730B_ST_AIO_PROGRESS (0x00000702)
- IEC60730B_ST_AIO_START (0x00000703)
- IEC60730B_ST_AIO_INIT (0x00000704)

Function performance:

The function duration is up to 63 cycles. (0.63 μ s)⁵

The function size is 112 bytes.²

3.9 IEC60730B_CM4_CM7_AIO_InputCheck_k3s()

Dedicated for MK32W0x devices. This function executes the second part of the AIO test sequence. This part reads the converted analog value and checks if the value fits into the predefined limits. The test is finished, when this function reports IEC60730B_AIO_PASS or IEC60730B_AIO_FAILED.

Function prototype:

IEC60730B_RESULT IEC60730B_CM4_CM7_AIO_InputCheck_k3s(IEC60730B_AIO_TEST_STR pObj,
unsigned long* pAdc, unsigned long resFifo);*

Function inputs:

pObj - pointer to the analog test instance.
pAdc - pointer to the base address of ADC module.
resFifo - 32-bit value of the RESFIFO register

Function output:

The function returns a value of unsigned long data type. It can have the following values:

- IEC60730B_ST_AIO_FAIL (0x00000701)
- IEC60730B_ST_AIO_PASS (0x00000000)
- IEC60730B_ST_AIO_PROGRESS (0x00000702)
- IEC60730B_ST_AIO_START (0x00000703)
- IEC60730B_ST_AIO_INIT (0x00000704)

Function performance:

The function duration is up to 80 cycles. (1.67 μ s)⁴
The function size is 138 bytes.²

3.10 IEC60730B_CM4_CM7_AIO_InputCheck_Cyclic()

Dedicated for MKV4x devices. This function executes the second part of the AIO test sequence. This part reads the converted analog value and checks if the value fits into the predefined limits. The test is finished, when this function reports IEC60730B_AIO_PASS or IEC60730B_AIO_FAILED.

Function prototype:

IEC60730B_RESULT IEC60730B_CM4_CM7_AIO_InputCheck_Cyclic(IEC60730B_AIO_TEST_STR pObj,
unsigned long* pAdc);*

Function inputs:

pObj - pointer to the analog test instance.
pAdc - pointer to the base address of ADC module.

Function output:

The function returns a value of unsigned long data type. It can have the following values:

- IEC60730B_ST_AIO_FAIL (0x00000701)
- IEC60730B_ST_AIO_PASS (0x00000000)
- IEC60730B_ST_AIO_PROGRESS (0x00000702)
- IEC60730B_ST_AIO_START (0x00000703)
- IEC60730B_ST_AIO_INIT (0x00000704)

Function performance:

The function duration is up to 60 cycles. (0.75 μ s)³
The function size is 136 bytes.²

-
- 1- Execution time measured with the use of MKV31/ 80MHz CPU clock/ 20 MHz Flash clock
 - 2- Function compiled by IAR v8.22.2
 - 3- Execution time measured with the use of MKV46/ 80MHz CPU clock/ 20 MHz Flash clock
 - 4- Execution time measured with the use of MK32W/ 48MHz CPU clock/ 24 MHz Flash clock
 - 5- Execution time measured with the use of MKE18F/ 100MHz CPU clock/ 25 MHz Flash clock

4 Analog I/O Test Module test concept

Content moved to TestReport_IEC60730B_AIO_rev3_0 document

5 Analog I/O Test Validation

Content moved to TestReport_IEC60730B_AIO_rev3_0 document

Table V. Validation

Date	Validated by	Validated Revision of Document	Validated Version of Source Code	Validation Result
11/2015	Jaroslav Lepka	1.0	1.0	P
11/2016	Pavel Sustek	1.1	1.1	P
11/2018	Jaroslav Lepka	3.0	3.0	P

Validation result options:

P – Passed

F – Failed

N/A – Not applicable

How to Reach Us:

Home Page:

nxp.com

Web Support:

nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all **liability, including without limitation consequential or incidental damages.** "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating **parameters, including "typicals," must be validated for each customer application by customer's technical experts.** NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address:
nxp.com/SalesTermsandConditions.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, Freescale, and the Freescale logo are trademarks of NXP B.V. All other product or service names are the property of their respective owners.

ARM, the ARM logo, and Cortex are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved.

© 2016 NXP B.V.

