

# **MG82G5E32**

## **IEC-60730 Sample Code**

### **User Guide**

## Contents

1	IEC-60730 specification overview.....	3
2	IEC-60730 test item description.....	4
2.1	CPU Register Test.....	4
2.2	CPU PC Test.....	4
2.3	Interrupt Test.....	4
2.4	Clock Test.....	4
2.5	Invariable Memory Test(ROM).....	5
2.6	Variable Memory Test(RAM).....	5
2.7	IO Test.....	6
2.8	AD Test.....	6
3	Matter needing attention.....	6
4	Revision History.....	7

## **1 IEC-60730 specification overview**

Regulatory agencies in the United States and Europe have put forward home appliance safety designs that meet regulatory requirements. The International Electrotechnical Commission has introduced the IEC-60730 safety development standard for household appliances. In Europe, these requirements are defined by IEC-60730. In October 2007, all home appliances sold on the European market must comply with the IEC-60730 standard. Its purpose is to avoid failure or at least to ensure that there is no risk of harm to the user in any failure of the equipment. When microcontrollers develop these devices, semiconductor suppliers must consider the impact of these standards on home appliance manufacturers. Appendix H of IEC-60730 describes three software Class A~C classifications. When using a microcontroller (MCU). At the beginning of development, designers must consider that MCUs with hardware and software functions need to meet the requirements of the IEC-60730 standard. This sample program is based on Class B.

## **2 IEC-60730 test item description**

The main test items of IEC-60730 Class B are as follows:

### **2.1 CPU Register Test**

This test item is for the CPU core related register to test whether it can be written and read normally, and this kind of register is more core, so use Assembly to perform the test, if the test result is wrong, an endless loop will be executed.

Test register name: R0 ~ R7, A, B, DPH, DPL, SP.

Function name: Test\_Reg

### **2.2 CPU PC Test**

This test item is to test the Program Counter. The user predefines the program address of 15 subroutines, calls these 15 subroutines and compares whether the PC is the same as the pre-defined program address. This test item is destructive, it should be called and executed when the system is reset. If the test result is wrong, an endless loop will be executed.

Function name: Test\_PC

### **2.3 Interrupt Test**

This test item is executed by two interrupts with fixed interrupts at different times. The main test is to test whether the number of interrupts generated by interrupt A is within the expected range. These two interrupts use the same clock source. Take the sample program provided by Megawin to illustrate, Timer0 uses the IHRCO clock source, set to generate an interrupt every 1ms, Timer1 also uses the IHRCO clock source, set to generate an interrupt every 100us, in the Timer0 interrupt service routine to compare the number of interrupts of Timer1 is within the pre-defined range, if it exceeds the range, an error must be returned.

Function name: unsigned char IntTest(void)

Return: return "TEST\_NORMAL" if the comparison is correct, and return "TEST\_FUNC\_ERROR" if it is wrong.

### **2.4 Clock Test**

This test item is executed by two interrupts at different times, fixed interrupts and from different clock sources. The main test is to test whether the number of interrupts generated by interrupt C and interrupt A is within the expected range. Take the sample program provided by Megawin as an example. Timer0 uses IHRCO clock source and configure to generate an interrupt every 1ms. RTC uses ILRCO clock source and configure to generate an interrupt about every 0.5s, in the RTC interrupt service routine to compare the number of interrupts of Timer0 is within the predefined range, if it exceeds Range, an error must be returned.

Function name: unsigned char ClockTest (void)

Return: return "TEST\_NORMAL" if the comparison is correct, and return "TEST\_FUNC\_ERROR" if it is wrong.

## 2.5 Invariable Memory Test(ROM)

This test item is to check whether the ROM has a single bit fault, and use CRC16 to perform the test. This test item will record the CRC code generated by the MCU hardware CRC Engine in the 0x79FC ~ 0x79FF of the IAP (if the address is blank 0xFF, it will be record here), every boot will use CRC Engine to check the flash and generate CRC code, and then compare with the CRC code recorded by IAP to see if it is correct. If the CRC code compares incorrectly, an endless loop will be executed.

Function name: unsigned int MakeCRCCode(unsigned char Addr\_H, unsigned char page)

Parameter Addr\_H: Check the starting address of flash.

Parameter page: the number of pages to be checked, 1 page = 512bytes.

Return: CRC code generated by CRC Engine.

Function name: unsigned char CRCTest(unsigned char Addr\_H, unsigned int crc\_original)

Parameter Addr\_H: Check the starting address of flash.

Parameters crc\_original: CRC code to be compared with IAP.

Return: return "TEST\_NORMAL" if the comparison is correct, and return "TEST\_FUNC\_ERROR" if it is wrong.

When the CPU runs at 12MHz, using the CRC Engine to check the 1K ROM Size, it takes 88us.  
When the CPU runs at 12MHz, using the CRC Engine to check the 2K ROM Size, it takes 188us.  
When the CPU runs at 12MHz, using the CRC Engine to check the 4K ROM Size, it takes 344us.  
When the CPU runs at 12MHz, using the CRC Engine to check the 8K ROM Size, it takes 697us.  
When the CPU runs at 12MHz, using the CRC Engine to check the 16K ROM Size, it takes 1.38ms.

When the CPU runs at 16MHz, using the CRC Engine to check the 1K ROM Size, it takes 66us.  
When the CPU runs at 16MHz, using the CRC Engine to check the 2K ROM Size, it takes 130us.  
When the CPU runs at 16MHz, using the CRC Engine to check the 4K ROM Size, it takes 268us.  
When the CPU runs at 16MHz, using the CRC Engine to check the 8K ROM Size, it takes 522us.  
When the CPU runs at 16MHz, using the CRC Engine to check the 16K ROM Size, it takes 1.037ms.

When the CPU runs at 24MHz, using the CRC Engine to check the 1K ROM Size, it takes 44us.  
When the CPU runs at 24MHz, using the CRC Engine to check the 2K ROM Size, it takes 87us.  
When the CPU runs at 24MHz, using the CRC Engine to check the 4K ROM Size, it takes 172us.  
When the CPU runs at 24MHz, using the CRC Engine to check the 8K ROM Size, it takes 350us.  
When the CPU runs at 24MHz, using the CRC Engine to check the 16K ROM Size, it takes 689us.

When the CPU runs at 32MHz, using the CRC Engine to check the 1K ROM Size, it takes 33us.  
When the CPU runs at 32MHz, using the CRC Engine to check the 2K ROM Size, it takes 65us.  
When the CPU runs at 32MHz, using the CRC Engine to check the 4K ROM Size, it takes 129us.  
When the CPU runs at 32MHz, using the CRC Engine to check the 8K ROM Size, it takes 262us.  
When the CPU runs at 32MHz, using the CRC Engine to check the 16K ROM Size, it takes 517us.

## 2.6 Variable Memory Test(RAM)

This test item is to check whether the RAM can be written and read normally. Each boot will perform a write and read test on all the RAM on the IC. After the test, it will be cleared to 0x00. If the test result is an error, an infinite loop will be executed.

Function name: Test\_RAM

## 2.7 IO Test

This test item is to test the IO input & output function. When testing the output function, it may not be possible to output the "level to be tested by the user" due to the user's circuit relationship.

Function name: unsigned char Pn\_Input\_Test(unsigned char value)

Parameter value: fill in the expected value of Pn

Return: return "TEST\_NORMAL" if the comparison is correct, and return "TEST\_FUNC\_ERROR" if it is wrong.

Function name: unsigned char Pn\_Output\_Test(unsigned char value)

Parameter value: Fill in the expected output value of Pn to see if the output is correct.

Return: If the output is correct, it will return "TEST\_NORMAL", if it is wrong, it will return "TEST\_FUNC\_ERROR".

## 2.8 AD Test

This test item is to test the ADC function. Test whether the AD value converted by the selected AD input channel is within the expected range. If the result is not within the range, the program will execute an endless loop.

Function name: unsigned char AD\_Test(unsigned char AIN\_num, unsigned int value\_low, unsigned int value\_upper)

Parameter AIN\_num: AD channel

Parameter value\_low: the minimum value of the expected range

Parameter value\_upper: the maximum value of the expected range

Return: If the obtained AD value is within the range, it will return "TEST\_NORMAL", if it is wrong, it will return "TEST\_FUNC\_ERROR".

## 3 Matter needing attention

This program belongs to the Demo Code, and the user needs to adjust the test time (interval & timing) and IO pin according to the actual application.

## 4 Revision History

Revision	Description	Date
V1.00	(1) Initial release.	2021/06/08