Holtek MCU UL / IEC 60730 Certification Measures

D/N: AN0584EN

Introduction

The International Electrotechnical Commission (IEC) has produced the safety standard IEC 60730 for household appliance development.

The IEC 60730-1 Standard (Automatic Electrical Controls for Household and Similar Use – Part 1: General Requirements) defines the test and diagnostic methods that ensure the safe operation of the controlled equipment used in household appliances. According to this standard, designers of household appliances must ensure that no injuries or damage to surrounding properties occur during product normal use or in the case of careless or incorrect operation by the user. Annex H is the key part of the standard which classifies the software into three categories: Class A, B and C. Household appliance manufacturers must design their products in accordance with the rules of these three classes.

According to the standard and the actual situation of different ICs, this document will offer recommendations for measures to assist users' products to pass the IEC 60730 certification. It will also explain the corresponding clauses in the IEC 60730 and test contents to help users understand the requirements of each test item more clearly, so that they can develop targeted self-test programs to accelerate the process to obtain certification.

Functional Description

The IEC 60730 standard defines three classes of household appliances, the differences between them are as follows.

Class A: This class is for situations where appliance safety does not rely on software or where the appliances will not cause injuries, an example of which could be LED lighting products. As no software certification is required, this class will not be explained in this document.

Class B: Control functions to prevent injuries due to unsafe operation of household appliances, such as electronically controlled washing machine door locks, motor thermal shutdown mechanisms etc.

Class C: Control functions to prevent special hazards such as hazardous explosions which may occur in electronic ignition gas stoves.

Class B Test Summary Table

Item	Component/Function	Fault/Error	Certification Measures	Definition
1.1	CPU/Registers	Stuck-at fault	Periodic self-test using static memory test – uses data 0x55, 0xAA to check each CPU register	H.2.19.6
1.3 2 3 6.3	CPU/Program counter Interrupt handling and execution Clock External communication/Timing	Stuck-at fault No interrupt or too frequent interrupt Wrong frequency Wrong point in time/wrong sequence	Independent time-slot monitoring and logical monitoring	H.2.18.10.3
4.1 4.3 5.1 5.2	Invariable memory Addressing to invariable memory Internal data path to invariable memory Addressing to invariable memory	All single bit faults Stuck-at fault Stuck-at fault Wrong address	Periodic modified checksum or Periodic CRC-16 check	H.2.19.3.1 H.2.19.4.2
4.2 4.3 5.1 5.2	Variable memory Addressing to variable memory Internal data path to variable memory Addressing to variable memory	DC fault Stuck-at fault Stuck-at fault Wrong address	Periodic self-test using static memory test – March C- algorithm or March X algorithm	H.2.19.6
6.1 6.2	External communication/Data External communication/Addressing	Hamming distance ≥ 3 Wrong address	Transfer redundancy including data, address – positive code and inverse code	H.2.18.2.2
7.1 7.2.1 7.2.2 5.1 5.2	Digital I/O Analog I/O (A/D and D/A) Analog multiplexer Internal data path to I/O component Addressing to I/O component	Fault conditions specified in H.27 Fault conditions specified in H.27 Wrong addressing Stuck-at fault Wrong address	Plausibility check	H.2.18.13
	Watchdog (independent clock source)	Too fast/too slow/clock stuck	Reset CPU after time-out – test once after power-on	

Class C Test Summary Table

Item	Component/Function	Fault/Error	Certification Measures	Definition
1.1	CPU/Registers	DC fault	Periodic self-test using walkpat memory test	H.2.19.7
1.2	CPU/Instruction decoding and execution	Wrong decoding and execution	Periodic self-test using equivalence class test	H.2.18.5
1.3 2 3	CPU/Program counter Interrupt handling and execution Clock	DC fault No interrupt or too frequent interrupt Wrong frequency	Independent time-slot monitoring and logical monitoring	H.2.18.10.3
1.4 1.5	CPU/Addressing Data paths instruction decoding	DC fault DC fault and wrong execution	Covered by 1.2, 4.3, 5.1, 5.2	
4.1 4.3 5.1 5.2	Invariable memory Addressing to invariable memory Internal data path to invariable memory Addressing to invariable memory	99.6% coverage of all information errors DC fault DC fault Wrong addressing and multiple addressing	Periodic CRC-16 check	H.2.19.4.2
4.2 4.3 5.1 5.2	Variable memory Addressing to variable memory Internal data path to variable memory Addressing to variable memory	DC fault and cross links DC fault DC fault Wrong addressing and multiple addressing	Periodic self-test using walkpat memory test	H.2.19.7
6.1 6.2	External communication/Data External communication/Addressing	Hamming distance ≥ 4 Wrong and multiple addressing	CRC-16 check including data, address	H.2.19.4.2
6.3	External communication/Timing	Wrong point in time/wrong sequence	Time-slot monitoring	H.2.18.10.4
7.1 7.2.1 7.2.2 5.1 5.2	Digital I/O Analog I/O (A/D and D/A) Analog multiplexer Internal data path to I/O component Addressing to I/O component	Fault conditions specified in H.27 Fault conditions specified in H.27 Wrong addressing DC fault Wrong addressing and multiple addressing	Testing pattern for input/output	H.2.18.22
	Watchdog (independent clock source)	Too fast/too slow/clock stuck	Reset CPU after time-out – test once after power-on	

The two common fault types involved in the IEC 60730 regulations are described below.

Stuck-at Fault: Due to impurities, CMOS gate oxide breakage, electrostatic damage, etc., the memory cells or signal lines are open-circuited or short-circuited – Stuck Open / Stuck at 1/ Stuck at 0.

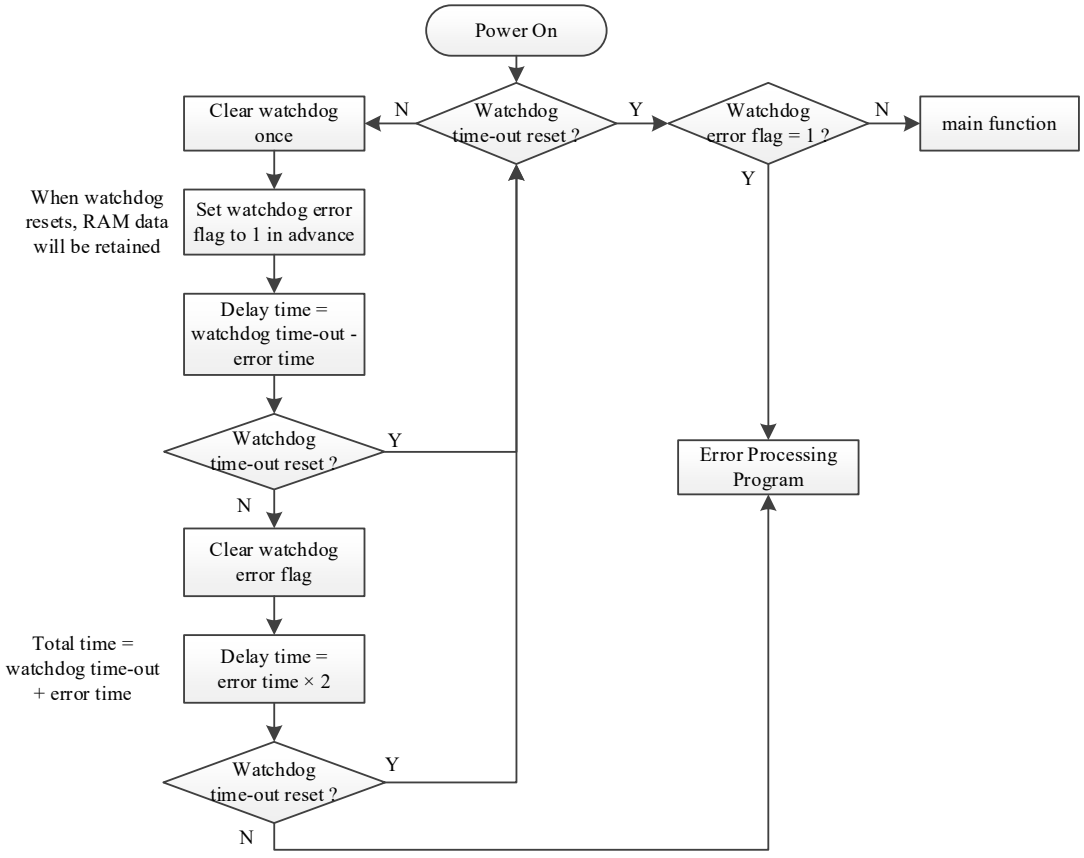
DC Fault: Multiple fault models between the memory cells or signal lines including stuck-at fault, bridging fault, etc.

Class B Certification Measures

Watchdog

Test Definition: Although not specified in the IEC 60730, regarding the watchdog integrated in the same wafer as the MCU, it is usually required to test whether it is operating normally to avoid the situation that the reset time is too short or too long or the watchdog is stuck. It is also required to use an independent clock source different from the system clock, such as a 32kHz LIRC clock or a 32768Hz crystal clock, to ensure that the MCU's input/output ports can be asynchronously and correctly reset to a known and safe state if necessary.

Certification Measure: Execute this test once after MCU power-on and before running other programs. Considering the system clock error as well as the frequency shift under different voltage and temperature conditions, the acceptable error range of the watchdog time-out should be appropriately expanded. Refer to the following flowchart.



CPU Registers

Test Definition: Static Memory Test – a fault/error control technique which is intended to detect only static errors.

Certification Measure: Starting from the ACC register, fill all the CPU registers with 0x55 and 0xAA respectively (some certification bodies may require additional data of 0x00 and 0xFF), except for some special purpose registers that may cause CPU abnormities. Then read the register contents and make comparisons to test whether the registers are working normally or not.

CPU Program Counter, Interrupt Handling and Execution, Clock, External Communication Timing

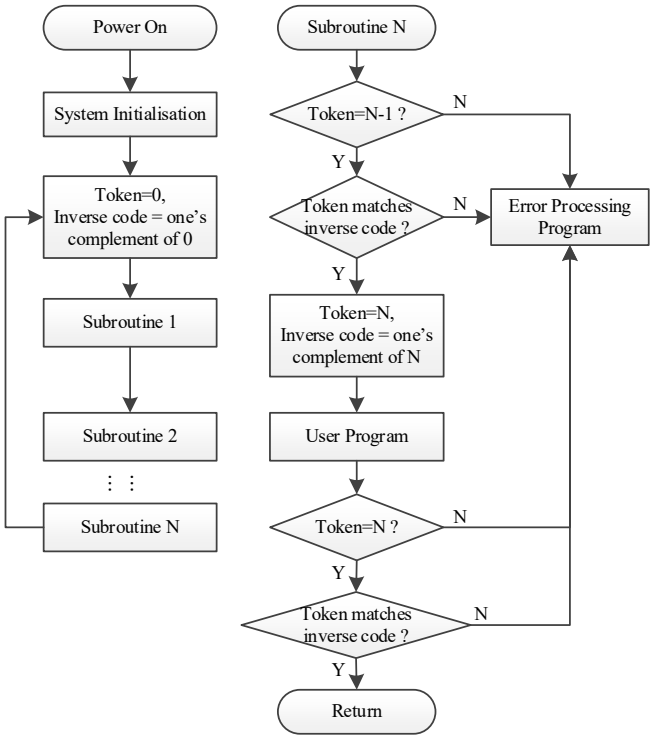
Test Definition: Independent Time-slot and Logical Monitoring – a fault/error control technique in which timing devices with an independent time base are periodically triggered in order to monitor the program function and sequence.

Certification Measure: Token passing method.

CPU Program Counter

A unique token variable can be defined and each subroutine has its own unique token number. Then pass the token through the subroutines and make comparisons to confirm that the subroutines are executed in order. Considering that the memory itself may be faulty, it is necessary to set an inverse code of the token to verify the correctness of the token itself.

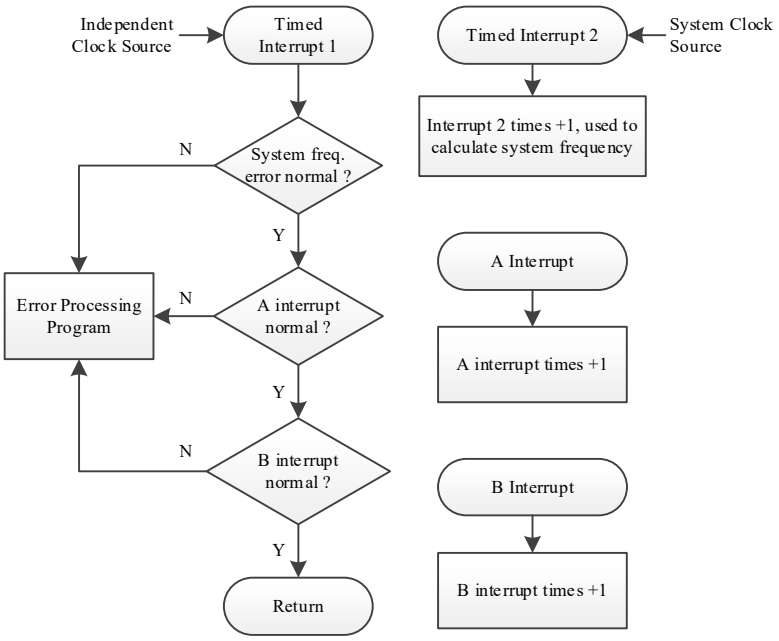
Furthermore, first determine the actual run time of the subroutine, then periodically query the token within a fixed time interruption, and compare the token with its previous value to check whether the program remains in the subroutine for too long.



Interrupt and System Clock

Due to the randomness of interrupt occurrences, it is more difficult to implement monitoring. A unique token should be defined for each interrupt, which is used to record the enter times to the present interrupt. Then check the number of occurrences of other interrupts within a periodic interrupt of an independent clock, which is to determine whether there are frequent interrupts or whether no interrupt has occurred.

Two timed interrupts are required in order to monitor the system clock. One uses the system clock as a time base and the other uses an independent clock. The two timed interrupts cross-validate the number of interrupts to detect whether the system clock frequency exceeds the specification.



Invariable Memory

Test Definition: Modified Checksum – a fault/error control technique in which a single word representing the contents of all words in the memory is generated and saved. During the self-test, a checksum is formed from the same algorithm and compared with the saved checksum. This technique recognises all the odd errors and some of the even errors.

CRC-double Word – a fault/error control technique in which at least two words are generated to represent the memory contents. During the self-test, the same algorithm is used to generate the same number of signature words which are compared with the saved words. The CRC-16 algorithm of invariable memory ensures that there is no single bit fault.

Certification Measure: Modified Checksum is recommended for the HT8 series while the CRC-16 algorithm is recommended for the HT32 series. The checksum/CRC signature is pre-stored in the EEPROM or Flash memory.

For the HT8 series of MCUs, the Flash data is composed of a high byte and a low byte and the checksum generated by the compiler also includes two bytes. Refer to the following formula:

CheckSum16 = CheckSum16 + Flash High Byte + Flash Low Byte

The initial value of CheckSum16 is 0x0000. The calculation result, if correct, will be equal to the "Program Checksum" generated by the compiler.

For the HT32 series of MCUs, CRC16-CCITT – $x^{16}+x^{12}+x^5+1$ is recommended. A CRC signature for a 32KB Flash capacity is recommended, which can provide a more reliable detection of single bit fault. If the Flash capacity is larger than 32KB, multiple CRC signatures are allowed. The hardware CRC function is suggested if the Flash capacity is larger than 64KB as the software calculation incurs a greater cost in time.

The following are three methods for CRC software calculation.

1. Calculation in bits: Execute a modulo two division using the first bit of the data and the polynomial to get a remainder. Then shift the remainder to the right by one bit and add the second bit (an addition without carry), then use the result to execute a modulo two division with the polynomial. Follow this way until all the bits have been utilised for the calculation. This method occupies a small space but results in a large calculation amount, which is not recommended for use.
2. Calculation in bytes: Work out 256 16-bit CRC codes corresponding to 0x00~0xFF in advance, which are stored in the Flash memory. Then the program can directly call these CRC codes to execute calculation in bytes. This method occupies a larger space but requires a smaller calculation amount, which is recommended for use.
3. Calculation in half-bytes: Only 16 CRC codes corresponding to 0x0~0xF are required to be stored, the occupied space of which is only 1/16 of the byte calculation method and the calculation amount of which is approximately doubled. This is a compromise between the calculation amount and occupied space.

Variable Memory

Test Definition: Marching Memory Test – a static memory test in which data is written to the memory area under test as in a normal operation. Every cell is then inspected in ascending order and a bit inversion performed on the contents. The inspection and bit inversion are then repeated in descending order.

Certification Measure: Transparent March C- or March X algorithm and dynamic redundancy check of global variables.

The variable memory can be roughly simplified into several functional modules, which are memory matrix, row address decoder, column address decoder, multiplexer, and read/write driving circuit.

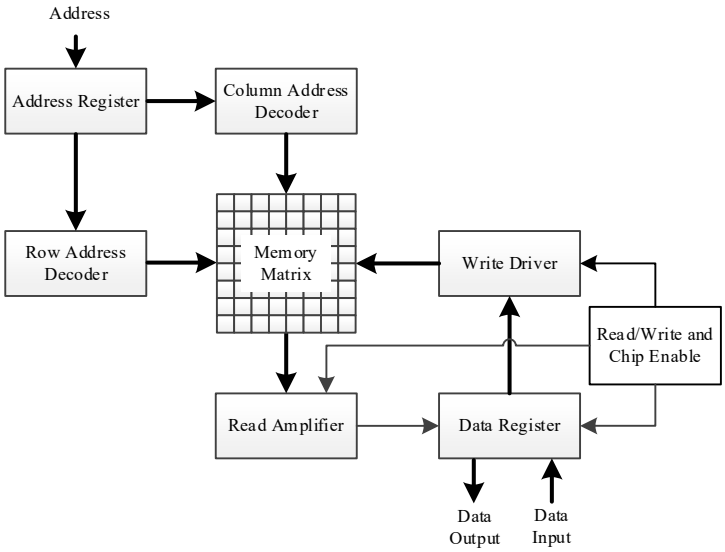


Figure 1. Variable Memory Functional Modules

The memory has a complicated structure and there are also complicated fault models corresponding to it, as shown in the following table.

Fault Model	Fault Description
SAF (Stuck-At Fault)	The memory cells or signal lines remain stuck at a certain logic value – constant 0 or constant 1. It is a common fault in memory manufacturing at present.
TF (Transition Fault)	The memory cell cannot be changed from 0 to 1 or from 1 to 0.
CF (Coupling Fault)	Changing the data of cell i causes data transition of cell j. The coupling problem of two adjacent cells can be divided into the following three situations: ● CFin – inversion: reading from or writing to memory cell i causes data inversion of cell j ● CFid – idempotent: reading from or writing to memory cell i causes the data of cell j to be fixed at 0 or 1 ● CFst – state: writing a constant logic value of 0 or 1 to memory cell i causes a read/write error on cell j
BF (Bridging Fault)	Fault of adjacent memory units due to bridging.
RF (Retention Fault)	The memory cell cannot maintain its initial logic value after a period of time (T).
AF (Address decoder Fault)*	There are four situations: one address cannot be used to access any memory cell; one address can access multiple memory cells; one memory cell cannot be accessed by any address; one memory cell can be accessed by multiple addresses.
Read/Write Circuit Fault*	Generally appears as a stuck-at fault or a bridging fault.

* Note: Address decoder faults and read/write circuit faults, which can be modeled and mapped as memory cell faults, will not be detected separately.

With regard to memory faults, there are multiple test algorithms, among which the March algorithm is a commonly used one. Its basic principle is to use a finite state machine to read from and write to all bits one by one. The instructions of this algorithm are relatively simple, only including reading and writing 0/1 as well as address changes. Through continuous reading and writing operations of the memory, almost all memory faults can be detected.

In order to improve the test efficiency, different test steps can be used to derive many variants such as MATS, March C+, March C, March C-, March X, etc., on the March algorithm basis.

The March C- and March X algorithms are suggested according to the program complexity and fault coverage, as shown below.

Algorithm	Fault Coverage	Algorithm Steps
March C-	SAF, AF, TF, CF	(w0); ↑(r0,w1); ↑(r1,w0); ↓(r0,w1); ↓(r1,w0); (r0)
March X	SAF, AF, TF, CFin	↑(w0); ↑(r0,w1); ↓(r1,w0); ↓(r0)

In the above algorithm steps, the meaning of each symbol is as follows.

Symbol	Meaning
↑	Address ascending order – from address 0 to address (n-1)
↓	Address descending order – from address (n-1) to address 0
No arrow	Optional address in ascending or descending order
( )	Single test step – perform read/write operations on a single cell in the order shown in the bracket
w0, w1	Write 0 or 1 to a single cell
r0, r1	Read from a single cell and verify whether the value is 0 or 1

For example, ↑(r0,w1) means starting from address 0, read and verify whether its data is “0”, then write “1” to address 0. Increase to address 1 and perform read and write operations to the address. Repeat this procedure until all addresses have been tested.

Data test using traditional March algorithm is performed in bits.

Since the variable RAM is arranged in bytes, it is required to expand the test data in order to improve the test efficiency and fault coverage. The expanded data is called data background. Now W1 represents to write a forward data background and W0 represents to write an inverse data background.

8-bit Data Background

Forward Data Background	Inverse Data Background
00000000	11111111
01010101	10101010
00110011	11001100
00001111	11110000

32-bit Data Background

Forward Data Background	Inverse Data Background
00000000000000000000000000000000	11111111111111111111111111111111
01010101010101010101010101010101	10101010101010101010101010101010
00110011001100110011001100110011	11001100110011001100110011001100
00001111000011110000111100001111	11110000111100001111000011110000
00000000111111110000000011111111	11111111000000001111111100000000
00000000000000001111111111111111	11111111111111110000000000000000

Either the March C- or March X test will overwrite the original RAM data. The IEC 60730 standard is required to perform periodic tests, which means that only one test after power-on is insufficient. Therefore, a proper measure called a transparent test should be used in order to not overwrite the original RAM data.

The transparent test needs to divide the RAM into several areas. For example, the RAM is divided into three areas, RAM1, RAM2 and RAM3, among which RAM3 will be a test backup area which is only used to back up the data of other RAM areas during the test. After performing a March test in RAM3, back up the data of RAM1 to RAM3, perform a March test and restore the data. Then execute the same test sequence for RAM2. Disable the interrupt functions before starting the test to avoid the interrupted data becoming abnormal.

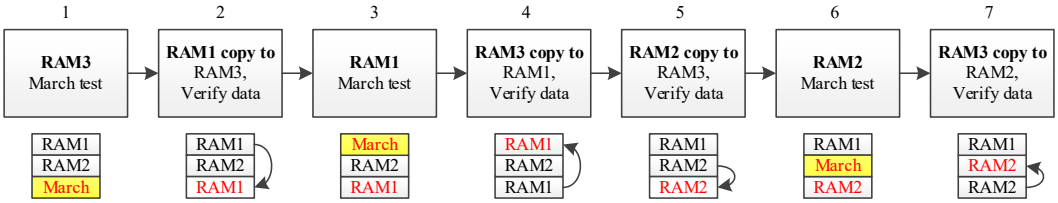


Figure 2. Transparent March Test

As the March algorithm test needs a longer time, the whole test can be divided into several fragments and only one RAM area is tested each time. This can avoid situations where the MCU resources are occupied for a long period of time.

The test mentioned above can only detect memory static faults. If any dynamic fault occurs on the memory, i.e., the data stored in the memory is changed without resulting in physical damage to the memory. This could be due to external radiation interference which causes certain memory data to change value, here a dynamic redundancy check method should be used to ensure the data stability.

A dynamic redundancy check of global variables means to store global variables, especially those safety-relevant variables, in an inverse code format into a physical area reserved for redundant storage.

Users need to divide the entire memory into at least three areas:

Area1: Temporary variable area or for compiler use

Area2: Global variable storage area

Area3: Global variable redundancy area

When storing data, first store the data into Area2, then store the inverse code of the data into the corresponding location in Area3. When reading data, read out the data of the corresponding location in Area2 and Area3 simultaneously, then check whether they are the inverse code of each other. If yes, continue the operation, otherwise, enter the error processing program.

External Communication

Test Definition: Transfer Redundancy – a form of code safety in which data is transferred at least twice in succession and then compared. This technique can recognise intermittent errors.

Hamming Distance – a statistical measure representing the capability of code to detect and correct errors. The Hamming distance of two code words is equal to the number of positions different in the two code words, for example, the Hamming distance of “1011101” and “1001001” is 2.

Certification Measure: The transmitter sends an address and data followed by a corresponding inverse code. The receiver compares the consistency of the two to check whether the data transfer is correct after receiving the data.

Additionally, a checksum or CRC signature can be added at the end of a group of data. In this way the transfer correctness can be checked only using one group of data.

I/O

Test Definition: Plausibility Check – a fault/error control technique in which inputs or outputs are checked for inadmissible data.

Testing Pattern – a fault/error control technique used for periodic testing of the input units, output units and interfaces of the controller. A test pattern is introduced to the unit and the results are compared to the expected values. Mutually independent means for introducing the test pattern and evaluating the results are used. The test patterns are constructed so as not to influence correct control operation.

Certification Measure: Use plausibility check to detect the specified fault status.

For digital I/Os, select an appropriate I/O to output 0 or 1 and then check whether the I/O status is abnormal and whether there is a short circuit or an open circuit between the I/O and the power supply. For critical signal pins that may cause danger, redundant input pins can be used to check whether the signal status is normal.

For analog I/Os, input a constant voltage and perform an A/D conversion, then check whether the converted value is within an acceptable range.

Class C Certification Measures

Watchdog, PC Pointer, Interrupt, Clock, Invariable Memory, External Communication, I/O

Use the same or similar test measures as Class B.

Instruction Decoding and Execution

Test Definition: Equivalence Class Test – a systematic test intended to determine whether the instruction decoding and execution are performed correctly. The test data is derived from the CPU instruction specification.

Certification Measure: Instructions are classified as follows:

- Move instructions
- Operation instructions
- Bit and Rotate instructions
- Conditional processing instructions
- Other instructions

Similar instructions are grouped and the input data set is subdivided into specific data intervals (equivalence classes). Each instruction within a group processes at least one set of test data so that the entire group processes the entire test data set. The test data can be formed from the following:

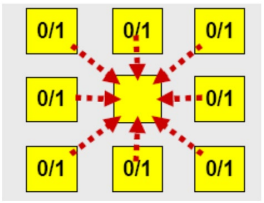
- Data from a valid range
- Data from an invalid range
- Data from the boundaries
- Extreme values and their combinations

Variable Memory

Test Definition: Walkpat Memory Test – a fault/error control technique in which a standard data pattern is written to the memory area under test as in normal operations. A bit inversion is performed on the first cell and the remaining memory area is inspected. Then the first cell is again inverted and the memory inspected. This process is repeated for all memory cells under test. A second test is conducted by performing a bit inversion of all cells in the memory under test and proceeding as above. This technique recognises all static bit errors as well as errors in interfaces between memory cells.

Certification Measure: Transparent walkpat test using logic "1" and "0" and dynamic redundancy check of global variables.

The situations where memory cells are in an incorrect state due to the different operations of the adjacent cells in a nine-rectangle-grid, is called Neighborhood Pattern Sensitive Faults (NPSF). The main cause of NPSF is the mutual interference between high-density storage cells. The NPSF detection actually includes detection of stuck-at fault, coupling fault and other types of memory faults.



Figre 3

For the walkpat test using logic "1", first clear the entire RAM to 0, then set the highest bit of the first byte to "1" and check whether the adjacent bits have experienced any changes. Next, right shift the "1" by one bit and check if any adjacent bits have changed state. Repeat the shift operation seven times until the lowest bit has been checked. Then move on to the highest bit of the next byte. Repeat this procedure until the entire RAM has been checked.

During this test, the logic "1" is shifted through the entire RAM step by step from left to right and from top to bottom.

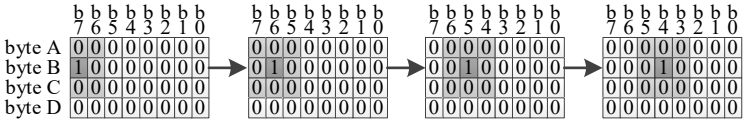


Figure 4

Similar to the above test, the walkpat test using logic "0" is performed by first setting the entire RAM to "1" and then shifting "0" step by step from left to right then from top to bottom.

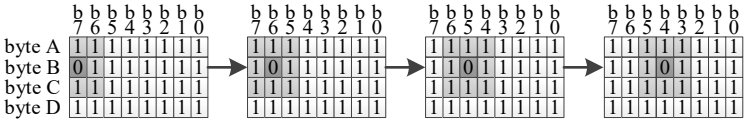


Figure 5

In the same way as the March algorithm test, the walkpat test will also overwrite the original RAM data. Therefore, it is also necessary to divide the RAM, and then follow the three steps of data backup, walkpat test and data recovery to execute the transparent test and dynamic redundancy check of global variables. For more details, refer to the Invariable Memory section of Class B.

Conclusion

This document has provided some recommended measures based on the IEC 60730 standard and the actual situation of Holtek MCUs, which will assist users' products to pass the IEC 60730 certification. It has also given a detailed introduction and explanation of some clauses in the IEC 60730 and the corresponding test measures. Users can clearly understand the various test items without the need to read the IEC 60730 text and look for certification measures, and develop targeted self-test programs to speed up the certification process.

Reference Material

"IEC 60730-1: Automatic Electrical Controls for Household and Similar Use", fourth edition, 2009

HT8 and HT32 series of MCUs' datasheet and user manual.

Versions and Modification Information

Date	Author	Version
2021.05.18	陈康超(chad)	V1.00

Disclaimer

All information, trademarks, logos, graphics, videos, audio clips, links and other items appearing on this website ('Information') are for reference only and is subject to change at any time without prior notice and at the discretion of Holtek Semiconductor Inc. and its related companies (hereinafter 'Holtek', 'the company', 'us', 'we' or 'our'). Whilst Holtek endeavors to ensure the accuracy of the Information on this website, no express or implied warranty is given by Holtek to the accuracy of the Information. Holtek shall bear no responsibility for any incorrectness or leakage.

Holtek shall not be liable for any damages (including but not limited to computer virus, system problems or data loss) whatsoever arising in using or in connection with the use of this website by any party. There may be links in this area, which allow you to visit the websites of other companies. These websites are not controlled by Holtek. Holtek will bear no responsibility and no guarantee to whatsoever Information displayed at such sites. Hyperlinks to other websites are at your own risk.

Limitation of Liability

In no event shall Holtek Limited be liable to any other party for any loss or damage whatsoever or howsoever caused directly or indirectly in connection with your access to or use of this website, the content thereon or any goods, materials or services.

Governing Law

This disclaimer is subjected to the laws of the Republic of China and under the jurisdiction of the Court of the Republic of China.

Update of Disclaimer

Holtek reserves the right to update the Disclaimer at any time with or without prior notice, all changes are effective immediately upon posting to the website.