June, 2010

Industrial Safety starts with IEC/UL 60730 Standards

FTF-ENT-F0714

Dugald Campbell - Large Appliance Systems Solutions Eng.

Donnie Garcia - Industrial and Multi-Market Systems Engineer

Contents

- ►Introduce 60730 and how it applies to MCUs
- ►Classification Class B Class C
- ►Components for Class B
  - Component Table Matrix
- ►Popular Measures for Class B
  - Component Table
  - CPU Stuck at
  - Program Flow
    - Time Slot Monitoring
    - Interrupts
    - Watchdog Timeout Test
  - Flash Memory
  - RAM memory
  - Communication
- ►Components for Class C
  - Component Table Matrix
- ►Popular Measures for Class C
  - Component Table Highlight CPU, RAM
  - CPU Instruction Test
  - RAM walking 1s Test
- ►Freescale's Safety Offerings
  - Software for Class B
  - Software for Class C
  - Device hardware feature
- ►Summary

What is IEC 60730?

- IEC 60730 – Automatic electrical controls for household and similar use – Part 1: General requirements

- Applies to Automatic Electrical Controls to perform safely within the household
- Discusses mechanical, electrical, electronic, environmental, endurance, EMC, abnormal operation of AC appliances
- Specifically for MCUs, Annex H: Requirements for Electronic Controls details new test and diagnostic methods to ensure the operation of embedded control hardware and software for appliances are safe

IEC 60335-1 (IEC 60730-1)

- **IEC 60335-1** Household and similar electrical appliances – Safety-Part 1 General Requirements
  - Compliance safety requirements for Large Appliance Manufacturers

↓

- IEC 60335-1 **Annex R** – Software Evaluation
  - Software shall be evaluated in accordance with the following clauses of **Annex H** of **IEC 60730-1**, as modified below:

↓

- **IEC 60730-1 Annex H** – Requirements for electronic controls
  - This chapter centers around **Table H.11.12.7**

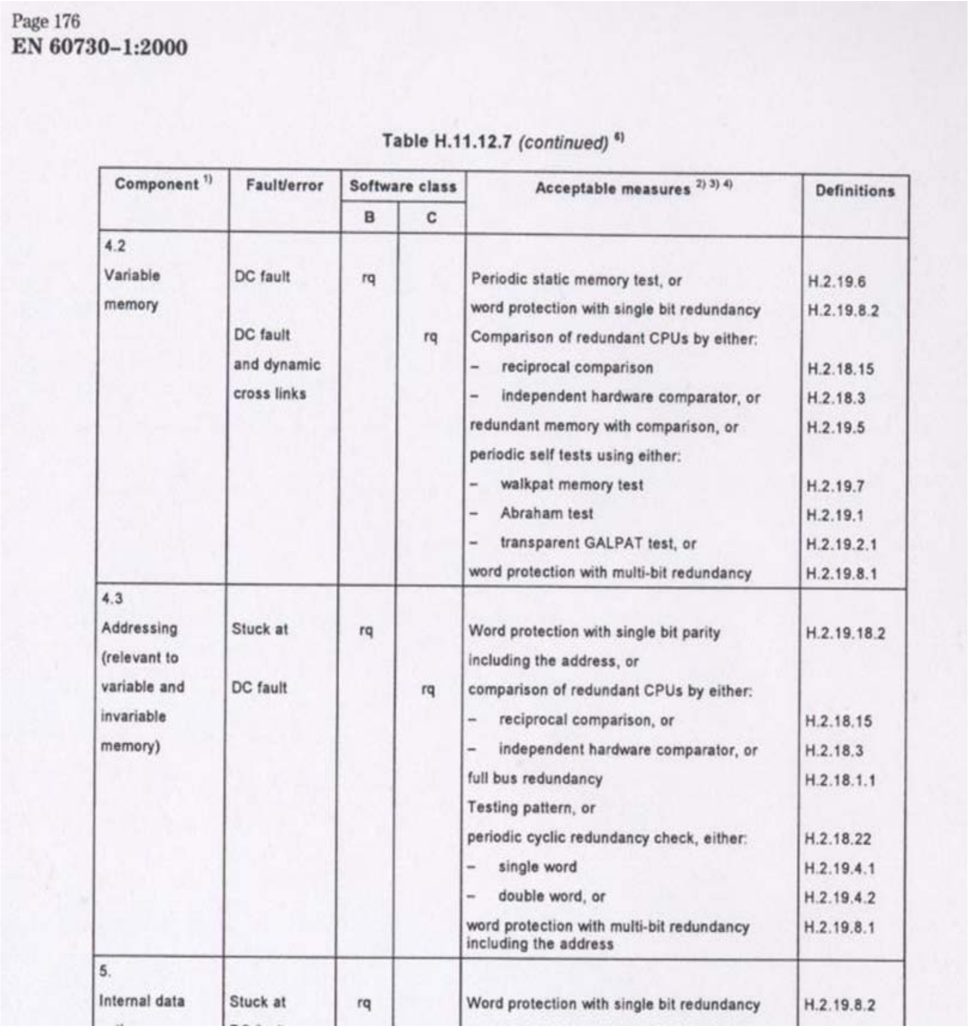
↓

- IEC 60730-1 **Annex H Table H.11.12.7**
  - Discusses the various embedded "components" that have to be tested to comply for class B and class C electronic controls
  - Provides optional "measures" that are required to ensure reliable and safe operation of the embedded "component"

Table H.11.12.7 Annex H IEC 60730-1

- Discusses the various embedded “components” that have to be tested to comply for class B and class C electronic controls

- Provides optional “measures” that are required to ensure reliable and safe operation of the embedded “component”



Page 176
EN 60730–1:2000

Table H.11.12.7 *(continued)* [6]

Component [1]	Fault/error	Software class		Acceptable measures [2] [3] [4]	Definitions
		B	C		
4.2					
Variable	DC fault	rq		Periodic static memory test, or	H.2.19.6
memory				word protection with single bit redundancy	H.2.19.8.2
	DC fault		rq	Comparison of redundant CPUs by either:	
	and dynamic			– reciprocal comparison	H.2.18.15
	cross links			– independent hardware comparator, or	H.2.18.3
				redundant memory with comparison, or	H.2.19.5
				periodic self tests using either:	
				– walkpat memory test	H.2.19.7
				– Abraham test	H.2.19.1
				– transparent GALPAT test, or	H.2.19.2.1
				word protection with multi-bit redundancy	H.2.19.8.1
4.3					
Addressing	Stuck at	rq		Word protection with single bit parity	H.2.19.18.2
(relevant to				including the address, or	
variable and	DC fault		rq	comparison of redundant CPUs by either:	
invariable				– reciprocal comparison, or	H.2.18.15
memory)				– independent hardware comparator, or	H.2.18.3
				full bus redundancy	H.2.18.1.1
				Testing pattern, or	
				periodic cyclic redundancy check, either:	H.2.18.22
				– single word	H.2.19.4.1
				– double word, or	H.2.19.4.2
				word protection with multi-bit redundancy including the address	H.2.19.8.1
5.					
Internal data	Stuck at	rq		Word protection with single bit redundancy	H.2.19.8.2

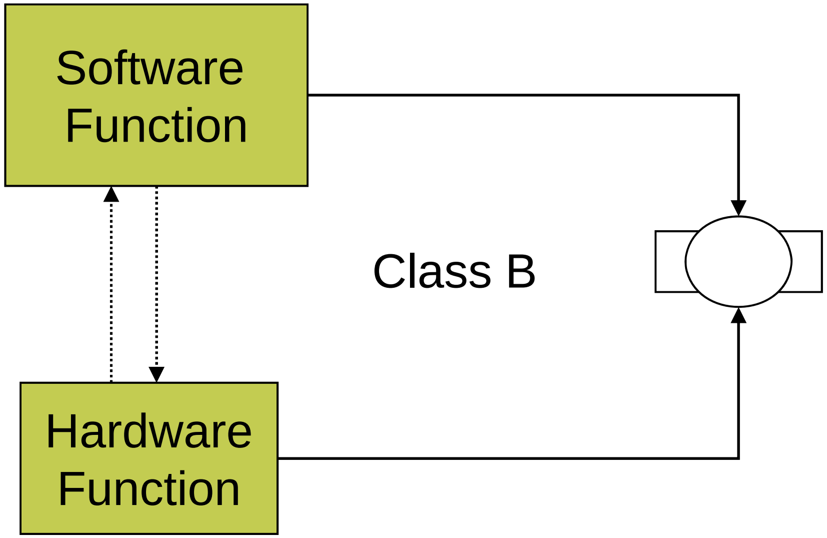
IEC 60730 Classification of Appliances

- Class A are products with no feature/function that can harm a human being
- Class B
  - IEC 60730-1: Control functions intended to prevent unsafe operation of the controlled equipment. Examples are: thermal cut-offs and door locks for laundry equipment
  - IEC 60335-1: Software that includes code intended to prevent hazards if a fault, other than a software fault occurs in the appliance
- Class C
  - IEC 60730-1: Control functions which are intended to prevent special hazards (e.g., explosion of the controlled equipment)
    - Examples are automatic burner controls and thermal cut-outs for closed water heater systems (unvented)
  - IEC 60335-1: Software that includes code intended to prevent hazards without the use of other protective devices

Example Hazard: Overheating of Motor

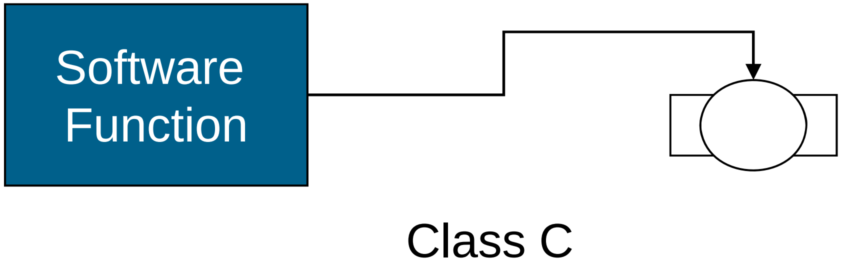
Hardware PTC monitors temperature
Software also monitors motor current
One function fails the other ensures safe operation

Class B – a fault occurring in a safety critical s/w routine will not result in a hazard due to another s/w routine or redundant h/w intervening.



Software only monitors motor current
If function fails then hazard will occur
Need more thorough diagnostics to ensure the software function is reliably working

Class C – a fault occurring in a safety critical software routine will result in a hazard.

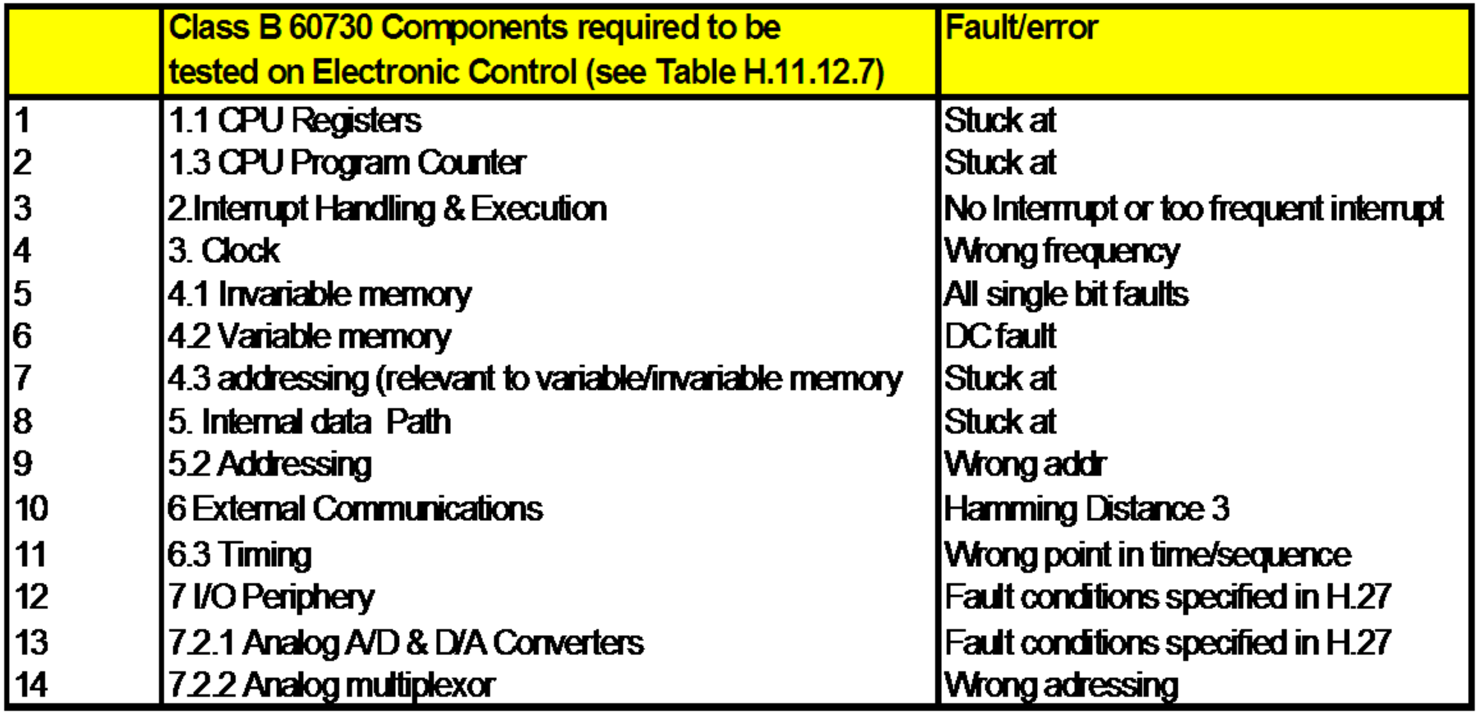


Industrial Safety Starts with Silicon:

Class B

60730 Class B Components

Appliance Manufacturers are required to implement “measures” to ensure that the above components are reliably working



	Class B 60730 Components required to be tested on Electronic Control (see Table H.11.12.7)	Fault/error
1	1.1 CPU Registers	Stuck at
2	1.3 CPU Program Counter	Stuck at
3	2.Interrupt Handling & Execution	No Interrupt or too frequent interrupt
4	3. Clock	Wrong frequency
5	4.1 Invariable memory	All single bit faults
6	4.2 Variable memory	DC fault
7	4.3 addressing (relevant to variable/invariable memory	Stuck at
8	5. Internal data Path	Stuck at
9	5.2 Addressing	Wrong addr
10	6 External Communications	Hamming Distance 3
11	6.3 Timing	Wrong point in time/sequence
12	7 I/O Periphery	Fault conditions specified in H.27
13	7.2.1 Analog A/D & D/A Converters	Fault conditions specified in H.27
14	7.2.2 Analog multiplexor	Wrong adressing

Class B Test Matrix

IEC 60730 CLASS B	Components	Registers Stuck at:	Program Counter stuck at	Interrupt handling and execution	clock	Invaraible Memory	Varaible memory	addressing Stuck at	Internal data path Stuck at	Addressing Wrong address	Hamming Distance 3	Timing	Wrong sequence	Input/Output Periphery
Acceptable measures	**Defininitions**													
Comparison of redundant CPUs be either														
reciprocal comparison,	**H.2.18.15**							X				X		
independent hardware comparator,	**H.2.18.3**							X				X		
full bus redundancy.	**H.2.18.1.1**							X						
Word protection with single bit redundancy	**H.2.19.8.2**	X	X			X	X	X	X	X				
Word protection with multi-bit redundancy including address	**H.2.19.8.1**								X		X			
Frequency monitoring	**H.2.18.10.1**				X									
Time-slot and logical monitoring,	**H.2.18.10.3**											X		
Independent time-slot monitoring or	**H.2.18.10.4**		X	X	X							X	X	
Logical monitoring of the program sequence.	**H.2.18.10.2**		X										X	
Transfer redundancy	**H.2.18.2.2**										X			
Protocol test	**H.2.18.14**													
Scheduled transmission.	**H.2.18.18**											X	X	
Periodic self-test	**H.2.16.6**													
Static memory test	**H.2.19.6**	X	X				X							
Periodic modified checksum;	**H.2.19.3.1**					X								
Multiple checksum,	**H.2.19.3.2**					X								
Periodic CRC-single word,	**H.2.19.4.1**							X	X		X			
Periodic CRC double word	**H.2.19.4.2**							X	X					
testing pattern	**H.2.18.22**							X	X	X				
Functional test	**H.2.16.5**	X	X	X										
Plausibility check	**H.2.18.13**													X

Dual MCU/CPU

ECC type

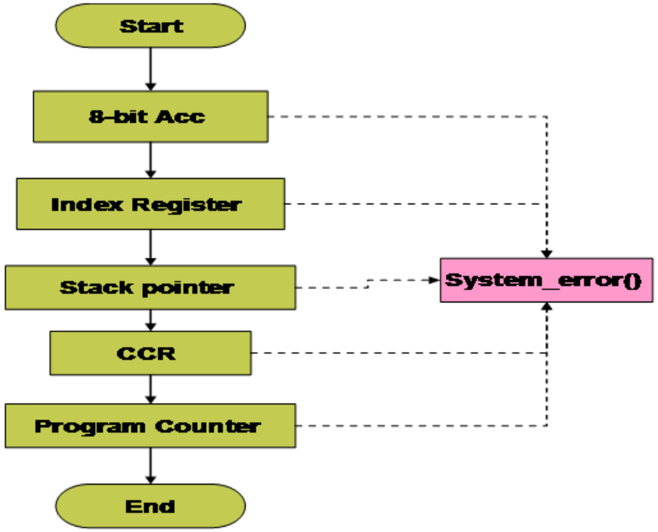
Indep. WDOG

S/W Design

Periodic
Self checks

Pre-application code

CPU Registers “Stuck At”

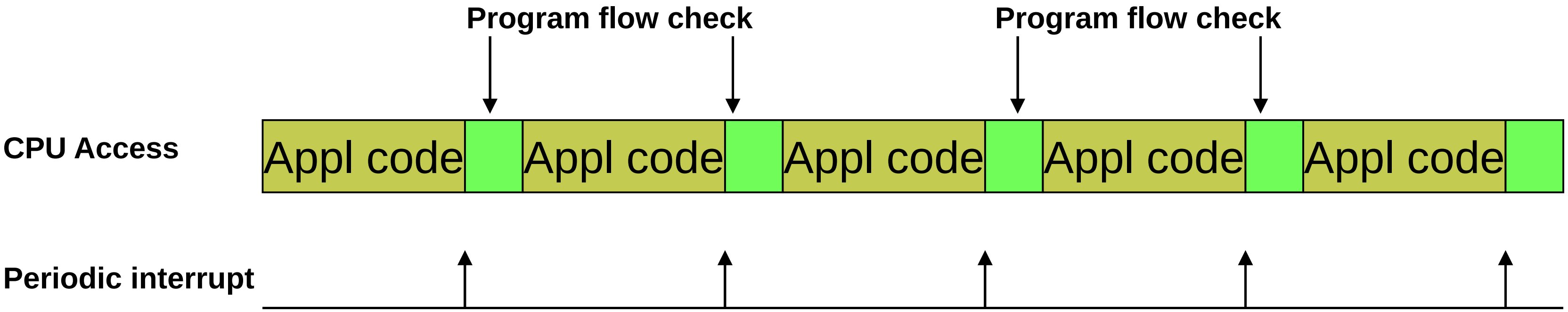


Using #0x55 and #0xAA data
Check each CPU register for “stuck at”

- Functional test H.2.16.5 - A single channel structure in which test data is introduced to the functional unit prior to its operation
- Periodic self-test H.2.16.6 - A single channel structure in which components of the control are periodically tested during operation using either:
  - Static memory test H.2.19.6 - a fault/error control technique which is intended to detect only static errors
  - Word protection with single bit redundancy H.2.19.8.2 - a fault/error control technique in which a single bit is added to each word in the memory area under test and saved, creating either even or odd parity. As each word is read, a parity check is conducted.

CPU Program Counter, Interrupt Handling, Clock, External Communications and Timing

- The measure: Time-slot monitoring or H.2.18.10.4 – 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. An example is a watchdog timer.
- Covers checking and verifying of the following components:
  - CPU Program Counter, Interrupt Handling, Clock, External Communications &Timing

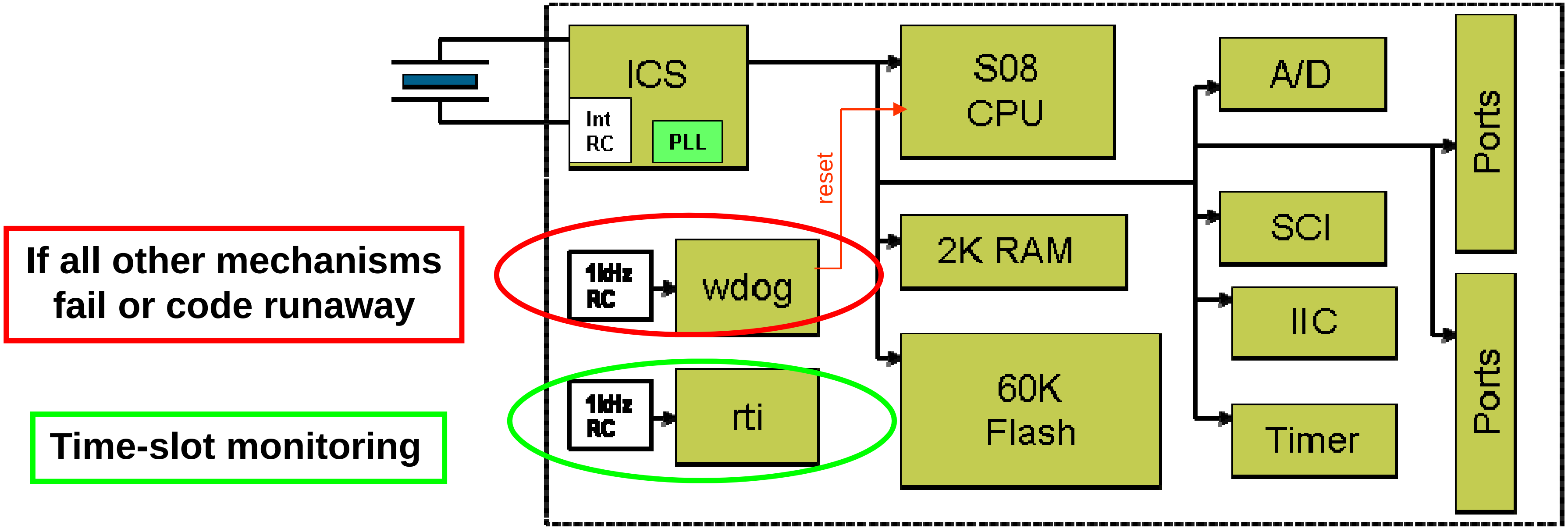


🟩 Time-slot monitoring; a periodic check on program code flow

A Periodic Interrupt e.g., timer overflow interrupts the application periodically and within the ISR some checks are made.

Time Slot Monitoring

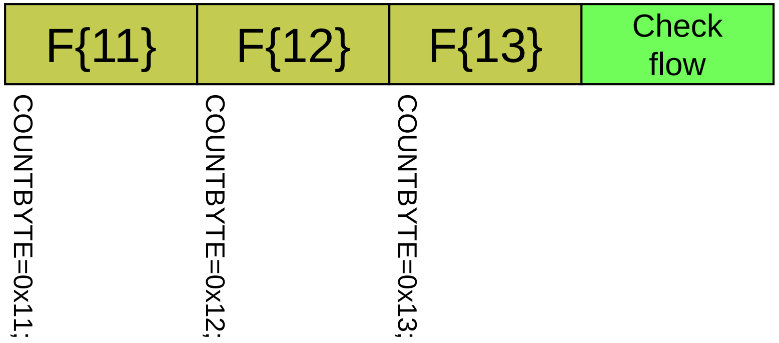
- ▶ Watchdogs should and must be deployed as the backup if all other safety mechanisms fail and/or there is code runaway
- ▶ Not really designed for periodic interrupts to execute time slot monitoring
- ▶ A better feature is an “independent clock” timer module e.g., S08AC60 RTI



Block diagram of Freescale MC9S08AC60 microcontroller

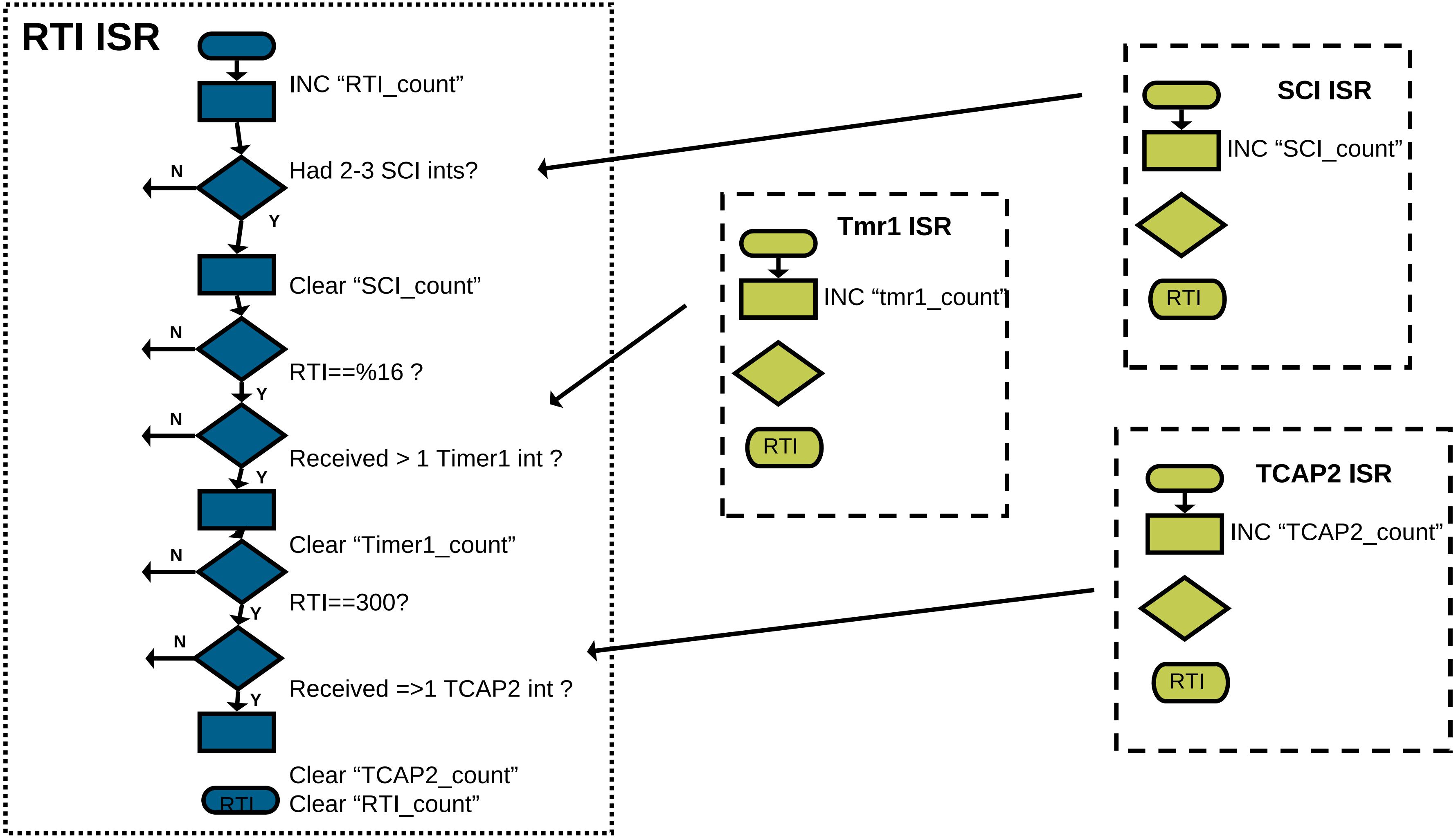
Token Passing – Program Flow

- A simple form of token passing is that you deploy a variable in RAM called COUNTBYTE and for each significant function you increment this COUNTBYTE by 1
- On the knowledge of how long the program takes to execute these various functions then the COUNTBYTE can be read within the ISR, and compared to previous captured values
- Caution: within each software function it is not recommended that you increment the COUNTBYTE by a certain value, but actually set the COUNTBYTE to a fixed value
- On real time embedded systems interrupts can occur at any random time and therefore are more difficult to monitor along with the program flow as described above. Therefore only the frequency of interrupts can be monitored then checked within the same periodic ISR routine.



```
….
If (COUNTBYTE < (previousCOUNTBYTE+2)) Error;
If (COUNTBYTE > (previousCOUNTBYTE+6)) Error;
/* prrogram flow OK */
previousCOUNTBYTE = COUNTBYTE;
…..
```

Token Passing on interrupts



RTI ISR

INC “RTI_count”

Had 2-3 SCI ints?

Clear “SCI_count”

RTI==%16 ?

Received > 1 Timer1 int ?

Clear “Timer1_count”

RTI==300?

Received =>1 TCAP2 int ?

Clear “TCAP2_count”

Clear “RTI_count”

RTI

Tmr1 ISR

INC “tmr1_count”

RTI

SCI ISR

INC “SCI_count”

RTI

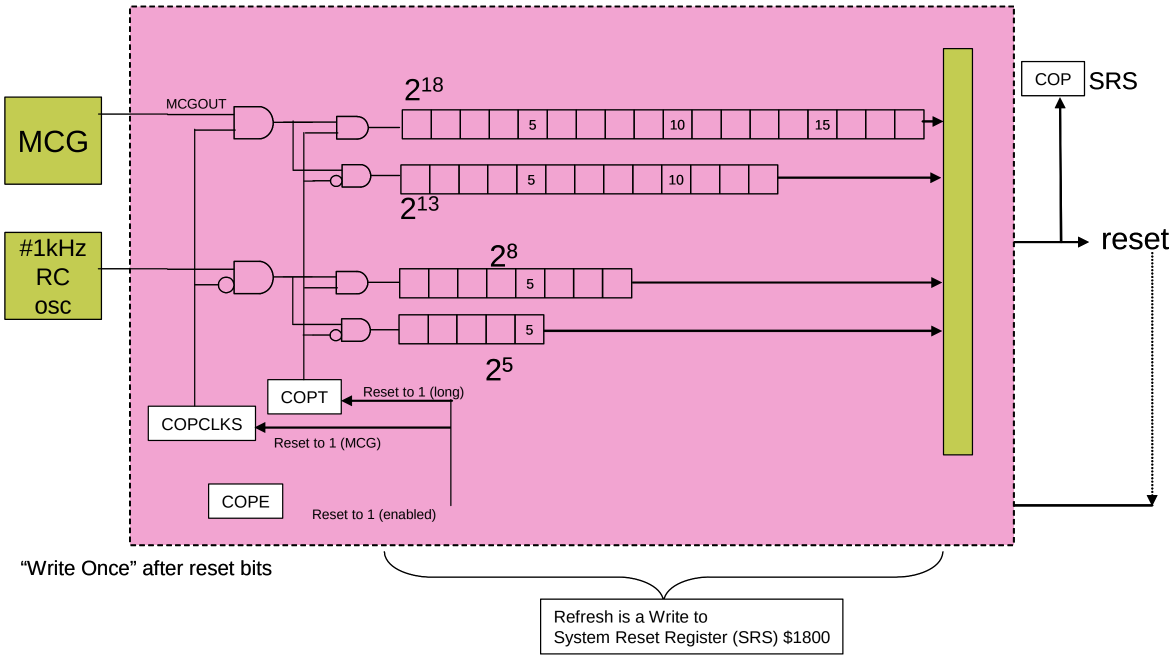
TCAP2 ISR

INC “TCAP2_count”

RTI

Independent Clocked Watchdog

- S08AC60 Watchdog using 1Khz RC oscillator is independent of CPU clock source
- Providing reliable protection against Clock faults (too fast/slow, **stuck clock**) and code runaway
- Watchdog must provide an asynchronous reset to all peripherals and input/output ports
- A timeout test should be initiated after power on reset, prior to running application code



“Write Once” after reset bits

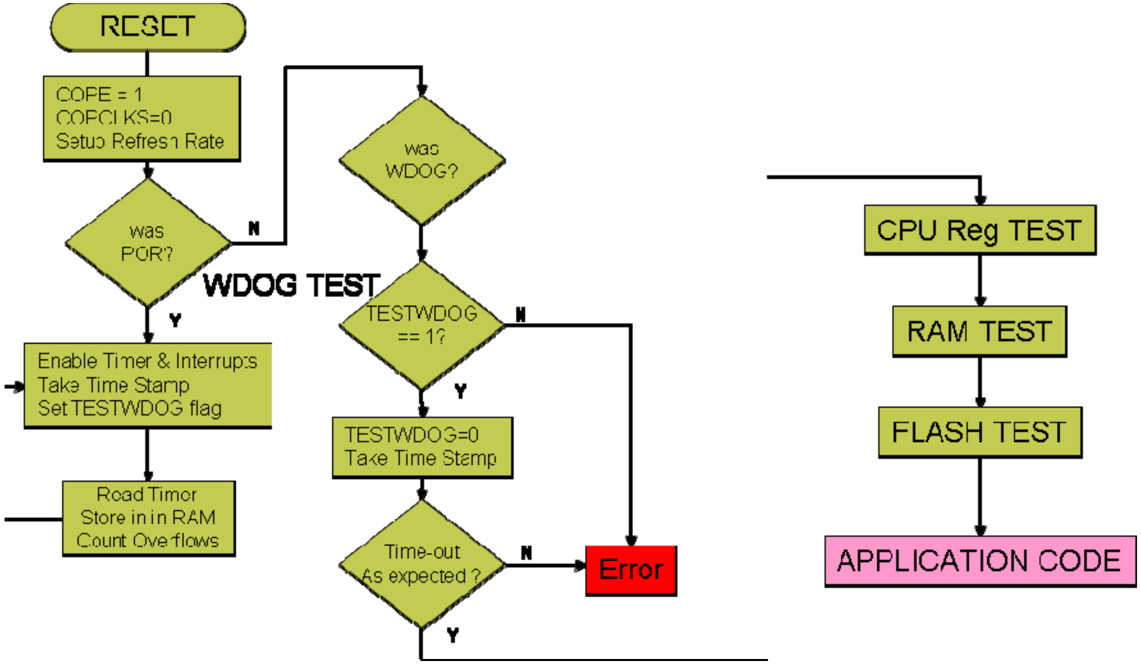
Refresh is a Write to
System Reset Register (SRS) $1800

Watchdog Timeout and Reset Test

Although not specified in 60730, for integrated independent clocked watchdogs on The same silicon of MCU, then it is an expectation to test that the watchdog is working correctly and that it:

- ► Times out as expected, and
- ► Resets the MCU into a known safe state

This watchdog test is executed prior to other periodic tests and application code



Invariable Memory (Flash) – All Single Bits Faults

- Periodic modified checksum; H.2.19.3.1 - a fault/error control technique in which a single word representing the contents of all words in memory is generated and saved. During self test, a checksum is formed from the same algorithm and compared with the saved checksum. This technique recognizes all the odd errors and some of the even errors.

Manu.s need prove
Their modified chksums can
Catch all single bit faults

OR

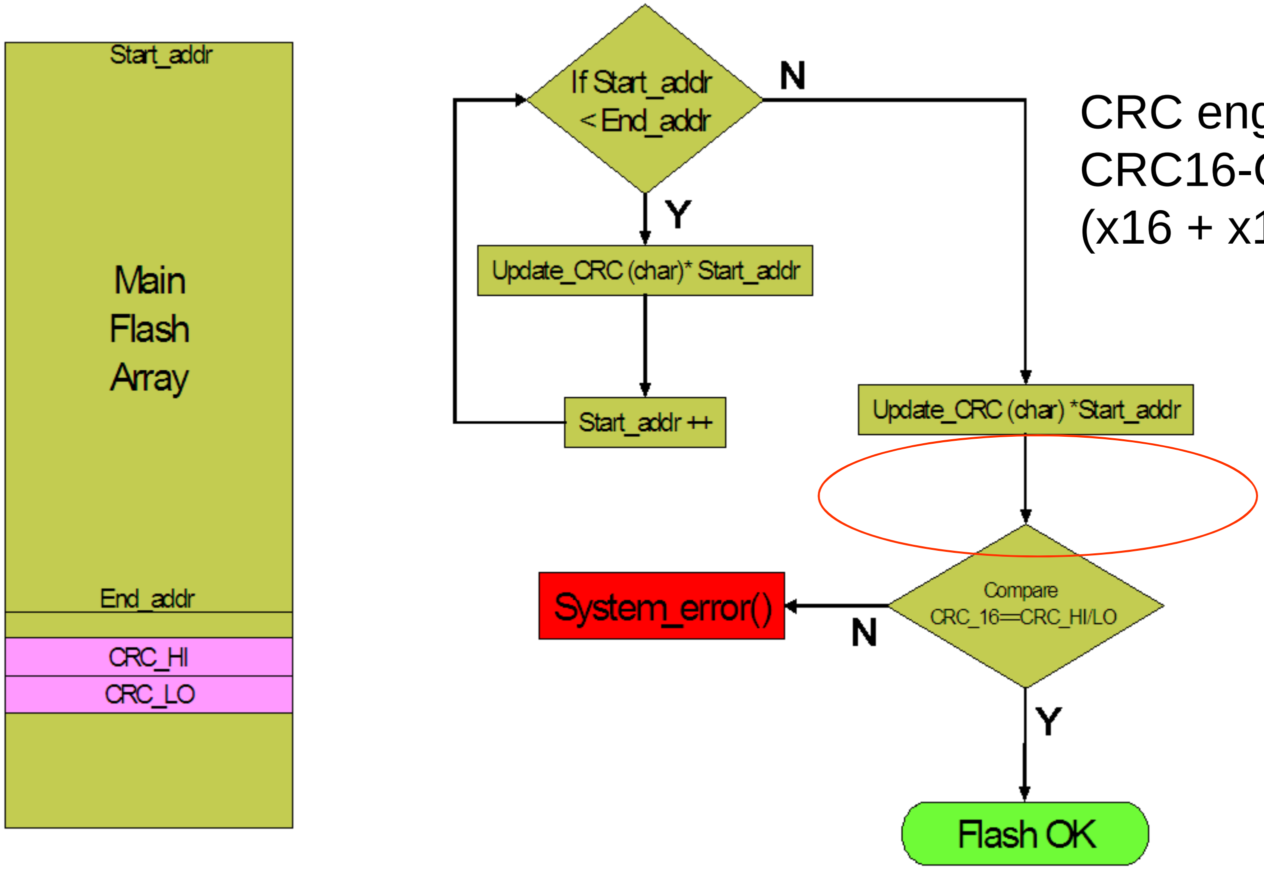
- Multiple checksum, H.2.19.3.2 - a fault/error control technique in which separate words representing the contents of the memory areas to be tested are generated and tested. During self test, a checksum is formed from the same algorithm and compared with the saved checksum for that area. This technique recognizes all odd errors and some of the even errors.

OR

- Word protection with single bit redundancy H.2.19.8.2

A CRC (16-bit) signature of the invariable memory is the preferred method of ensuring there are no single faults.

Flash CRC Test



CRC engine complying to CRC16-CCITT specification. (x16 + x12 + x5 + 1 polynomial)

>64k h/w CRC recommended

Note:
It is recommended that one CRC 16-bit signature is reliable For detecting single bit faults flash blocks < 32Kbytes. Large Flash arrays will require multiple CRC signatures.

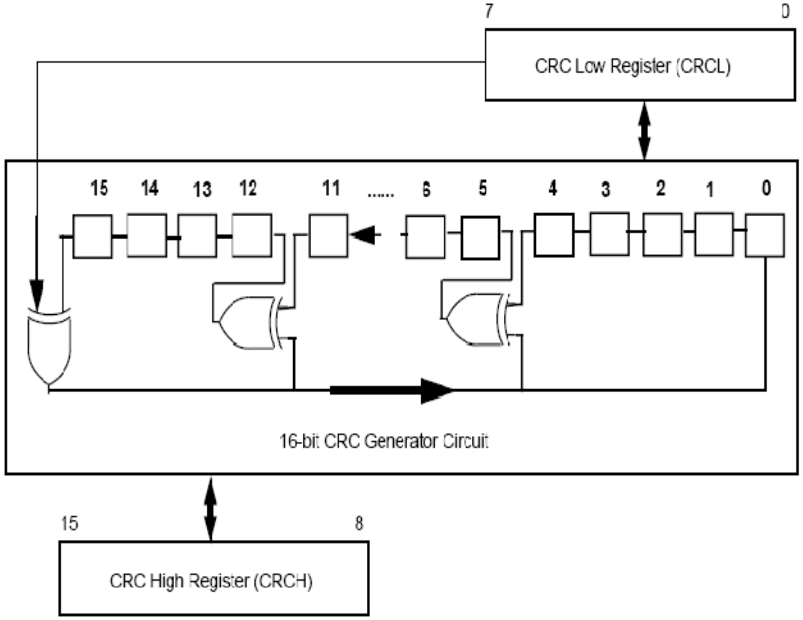
Flash CRC Test – Hardware Implementation

CRC engine complying to CRC16-CCITT specification. (x16 + x12 + x5 + 1 polynomial).

One byte shifted through CRC in 1 CPU cycle

Deployed on HCS08ACxx and MCF51ACxx, devices

Can be used for Flash, RAM and communication transfers



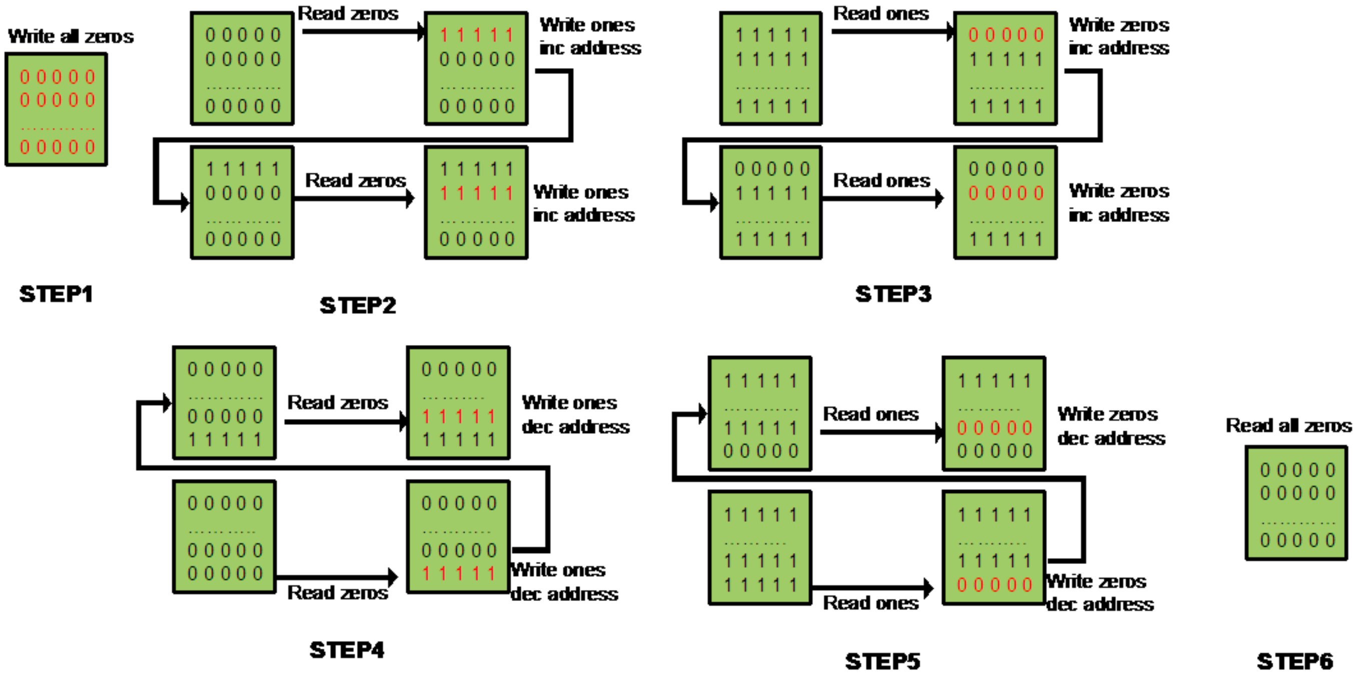
Seed by writing to CRCH, then CRCL
Update via CRCL only
A read of both CRCH and CRCL provides the current CRC signature

Variable Memory – DC Fault

- **Periodic static memory test H.2.19.6** - a fault/error control technique which is intended to detect only static errors

or

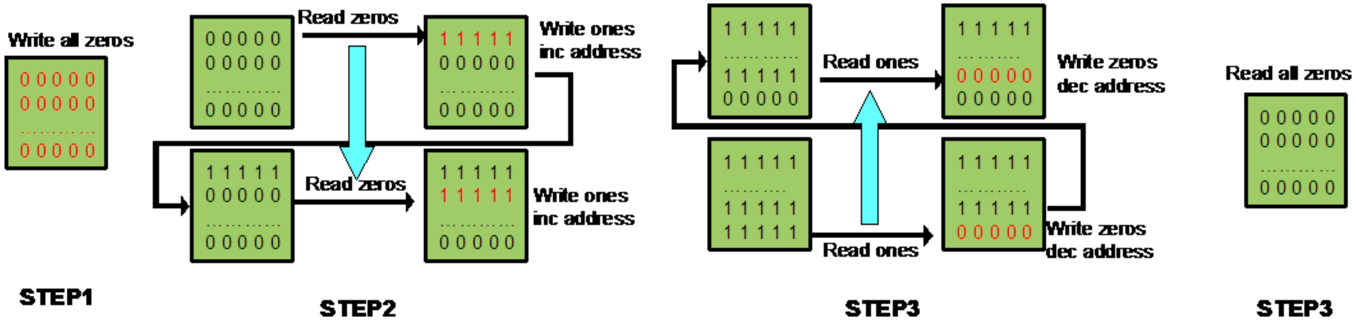
- **Word protection with single bit redundancy H.2.19.8.2** - (hardware error code correction)



March C (van der Goor, 1991)

March X Pattern

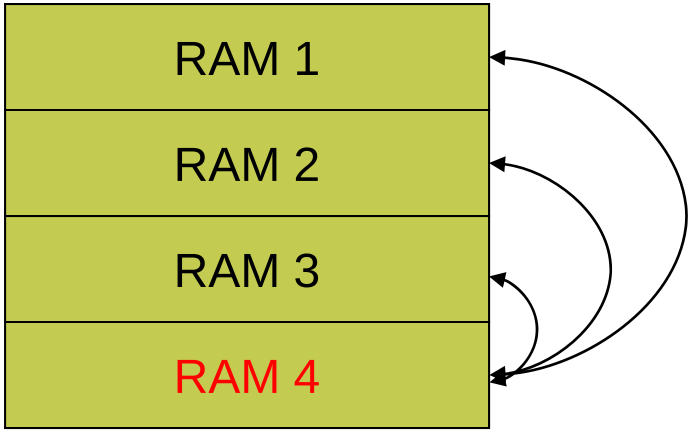
March X pattern is a subset of the March C pattern
Which detects the majority of failure mechanisms of the March C
But with a faster execution time



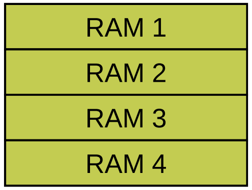
Both March C and X tests are destructive in nature in that they overwrite any data existing in the RAM and cannot be deployed mid application…unless we can make this a transparent test

Transparent March

- Split RAM into four segments
- 4th segment is “shadow” RAM used to temporarily store other segment variables until March test completed
- At a convenient time complete the following:
  - RAM 1 copy to RAM 4
  - Verify copy is successful
  - Deploy MARCH test on RAM 1
  - Copy RAM 4 to RAM 1
  - Verify copy is successful
  - Deploy normal application code



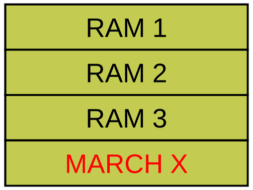
Making a “Destructive” into “Transparent”



RAM 1
RAM 2
RAM 3
RAM 4

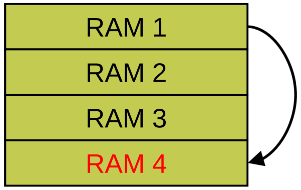
Segment RAM

Redundant RAM segment



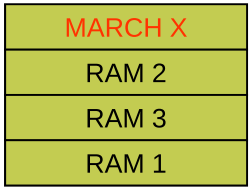
RAM 1
RAM 2
RAM 3
MARCH X

March X on RAM4



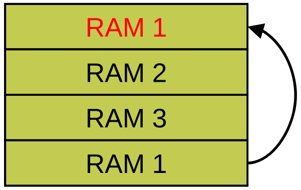
RAM 1
RAM 2
RAM 3
RAM 4

Copy RAM1 to RAM4.
Verify data copied



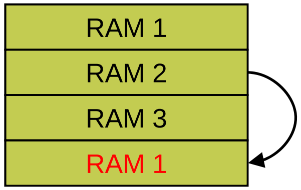
MARCH X
RAM 2
RAM 3
RAM 1

March X on RAM1



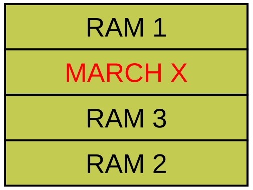
RAM 1
RAM 2
RAM 3
RAM 1

Copy RAM4 to RAM1.
Verify data copied



RAM 1
RAM 2
RAM 3
RAM 1

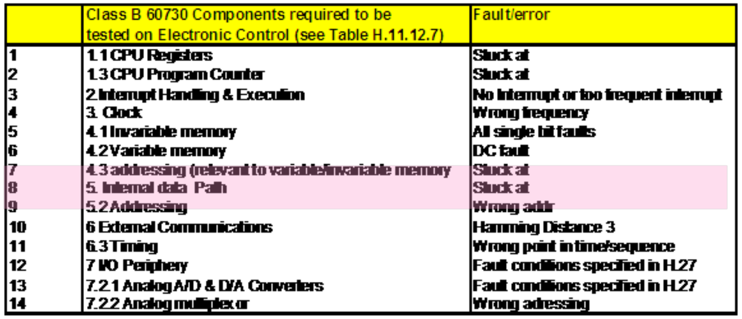
Copy RAM2 to RAM4.
Verify data copied



RAM 1
MARCH X
RAM 3
RAM 2

March X on RAM2

Class B Memory Address and Data Path



	Class B 60730 Components required to be tested on Electronic Control (see Table H.11.12.7)	Fault/error
1	1.1 CPU Registers	Stuck at
2	1.3 CPU Program Counter	Stuck at
3	2 Interrupt Handling & Execution	No Interrupt or too frequent interrupt
4	3. Clock	Wrong frequency
5	4.1 Invariable memory	All single bit faults
6	4.2 Variable memory	DC fault
7	4.3 addressing (relevant to variable/invariable memory	Stuck at
8	5. Internal data Path	Stuck at
9	5.2 Addressing	Wrong addr
10	6 External Communications	Hamming Distance 3
11	6.3 Timing	Wrong point in time/sequence
12	7 I/O Periphery	Fault conditions specified in H.27
13	7.2.1 Analog A/D & D/A Converters	Fault conditions specified in H.27
14	7.2.2 Analog multiplexor	Wrong adressing

- **4.3 Addressing**

(relevant to variable and invariable memory) stuck at

- **5. Internal data path stuck at**
- **5.2 Addressing - Wrong address**

These components intended for external memory microprocessor based designs. These components are tested by other measures on single chip microcontrollers.

External Communications Hamming Distance 3

- Word protection with multi-bit redundancy including address H.2.19.8.1.

Or

- CRC-single word, H.2.19.4.1 - a fault/error control technique in which a single word is generated to represent the contents of memory. During self test the same algorithm is used to generate another signature word which is compared with the saved word. The technique recognizes all one-bit, and a high percentage of multi-bit, errors.

Or

- **Transfer redundancy** H.2.18.2.2 – a form of code safety in which data is transferred at least twice in succession and then compared. This technique will recognize intermittent errors.

Or

- Protocol test H.2.18.14 - a fault/error control technique in which data is transferred to and from computer components to detect errors in the internal communications protocol.

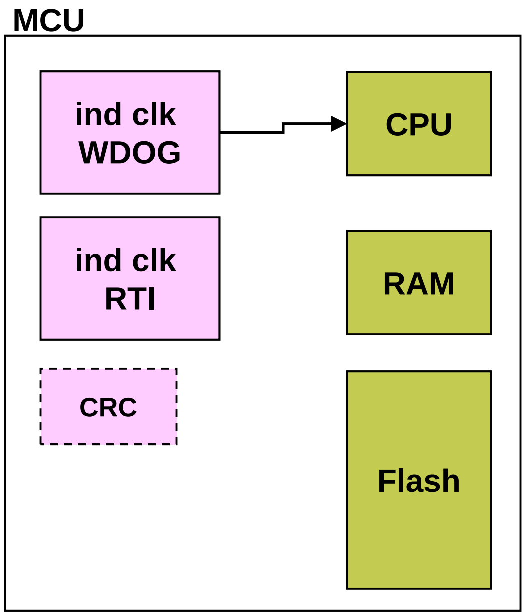
Plausibility Check

- ►7. I/O Periphery - Fault conditions specified in H.27
- ►7.2.1 A/D & D/A converters - Fault conditions specified in H.27
- ►7.2.2 Analog Multiplexer – Wrong addressing

Plausibility check H.2.18.13 - a fault/error control technique in which program execution, inputs or outputs are checked for inadmissible program sequence, timing or data. Examples are the introduction of an additional interrupt after the completion of a certain number of cycles or checks for division by zero.

I/O Periphery, For digital outputs, checks can be made to verify no short circuits or open circuits between adjacent signals and power supply. Manufacturers will utilize redundant input pins on MCUs to check on key signal pins that a short or open-circuit would lead to a hazard.
For analogue signals A/D and D/A checks on the boundary limits of the absolute value that should be made.
i.e., A input A/D pin should only see a small range of values with the full voltage conversion range, any value outside would be ignored in software.
Analogue multiplexers Today most manufacturers will need to have the capability to provide a known d.c. value to all input A/P pins. This allows test software to check if the multiplexer is working. Future analogue multiplexers should provide additional redundant channels on each pin so that a comparison between two channels can be made to verify that the multiplexer is working as expected.

Class B Generic MCU Requirements Summary



Software

- CPU Register “SA faults” Test
- March C and MARCH X (transparent) RAM Test
- Modified Checksum or CRC Flash Test
- Independent WDOG Test
- Plausibility Tests for key digital and analog I/O signals

- Time Slot monitoring of program flow
- and interrupt behavior
  - Token passing
  - Independent RTI

Hardware

- Independent clocked WDOG
- Independent Real Time interrupt

Nice to have:

- **CRC Engine for 64K+ memory devices**
- **Loss of Clock/Lock Reset**

Class C

60730 Class C – Components to be Tested

	Class C 60730 Components required to be tested on Electronic Control (see Table H.11.12.7)	Fault/error
1	1.1 CPU Registers	DC fault
2	1.3 CPU Program Counter	Stuck at
3	1.2 CPU Instruction Decoding & Execution	Wrong decoding or execution
4	2.Interrupt Handling & Execution	No Interrupt or too frequent interrupt
5	3. Clock	Wrong frequency
6	4.1 Invariable memory	99.6% coverage of all info errors
7	4.2 Variable memory	DC fault & dynamic cross links
8	4.3 addressing (relevant to variable/invariable memory	Stuck at
9	5. Internal data Path	Stuck at
10	5.2 Addressing	Wrong addr
11	6 External Communications	Hamming Distance 4
12	6.3 Timing	Wrong point in time/sequence
13	7 I/O Periphery	Fault conditions specified in H.27
14	7.2.1 Analog A/D & D/A Converters	Fault conditions specified in H.27
15	7.2.2 Analog multiplexer	Wrong addressing

CRC as
Done in Class B

Class C Test Matrix

Components

Optional Measures

Acceptable measures	Defininitions	1.1 Registers:DC fault	1.2Wrong decoding & execution	1.3 Program Counter Stuck at	1.4 Addressing: DC Fault	1.5 Data paths instr. Decodeing: DC fault & execution	2. Interrupt handling &execution	3.Clock	4.1 Invariable memory:99.6% of all infor errors	Variable memory: DC fault dynamic cross links	4.3 addressing oboth variable & invariabl:dc fault	5.Internal Data path: DC fault	5.2 Wrong address	6. External Comms: hamming dist 4	6.2 Addressing	6.4 Timing	7.I/O Periphery	7.2 Analog
Comparison of redundant CPUs by either			1		1													
-reciprocal comparison	H.2.18.15	~~X~~	~~X~~	~~X~~	~~X~~	~~X~~	~~X~~	~~X~~	~~X~~	~~X~~	~~X~~	~~X~~	~~X~~	~~X~~	~~X~~	~~X~~	~~X~~	~~X~~
-independent hardware comparator,	H.2.18.3	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
input comparison	H.2.18.8																X	X
multiple parallel outputs	H.2.18.11																X	X
output verification	H.2.18.12																X	X
testing pattern	H.2.18.22																X	X
code safety	H.2.18.2																X	
Internal error detection,	H.2.18.9	X	X			X												
redundant memory with comparison,	H.2.19.5	X							X	X								
Periodic self-test using either																		
- walkpat memory test	H.2.19.7	X								X								
- Abraham test	H.2.19.1	X								X								
- transparent GALPAT test	H.2.19.2.1	X								X								
word protection with multi-bit redundancy	H.2.19.8.1	X							X	X	X	X	X		X			
including the address, or data redundancy,	H.2.18.2.1											X	X	X				
static memory test and word protection	H.2.19.6	X																
with single bit redundancy	H.2.20.8.2	X																
Periodic self-test using equivalence class test	H.2.18.5		X															
Periodic self-test and monitoring using either	H.2.16.7			X	X	X												
-independent time-slot and logical monitoring	H.2.18.10.3			X			X									X		
- internal error detection	H.2.18.9			X														
the address lines	H.2.18.22				X	X					X	X	X					
full bit bus parity including the address	H.2.18.1.1				X						X				X			
Periodic self-test using a testing pattern of:multibit parity	H.2.18.1.2					X												
Frequency monitoring	H.2.18.10.1							X										
time-slot monitoring	H.2.18.10.4							X								X		
crc -single word	H.2.19.4.1								X		X				X			
crc -double word	H.2.19.4.2								X		X			X	X			
protocol test	H.2.18.14											X			X			
transfer redundancy	H.2.18.2.2														X			
scheduled transmission	H.2.18.18															X		
Logical monitoring	H.2.18.10.2															X		

Dual MCU/CPU/channel

ECC type

Periodic Self checks

S/W Design

Indep. WDOG

S/W Design

1.2 Instruction Decoding and Execution

Acceptable measure are:

1.2 Instruction decoding and execution	Wrong decoding and execution	rq	Comparison of redundant CPUs by either		
			-reciprocal comparison		H.2.18.15
			-independent hardware comparator,	or	H.2.18.3
			Internal error detection,	or	H.2.18.9
			Periodic self-test using equivelance class test	or	H.2.18.5

IEC 60730 Class C Requirement to test Instruction Decoding and Execution

Acceptable measure to test is:

Periodic self-test using equivalence class test

H.2.18.5 Equivalence Class Test

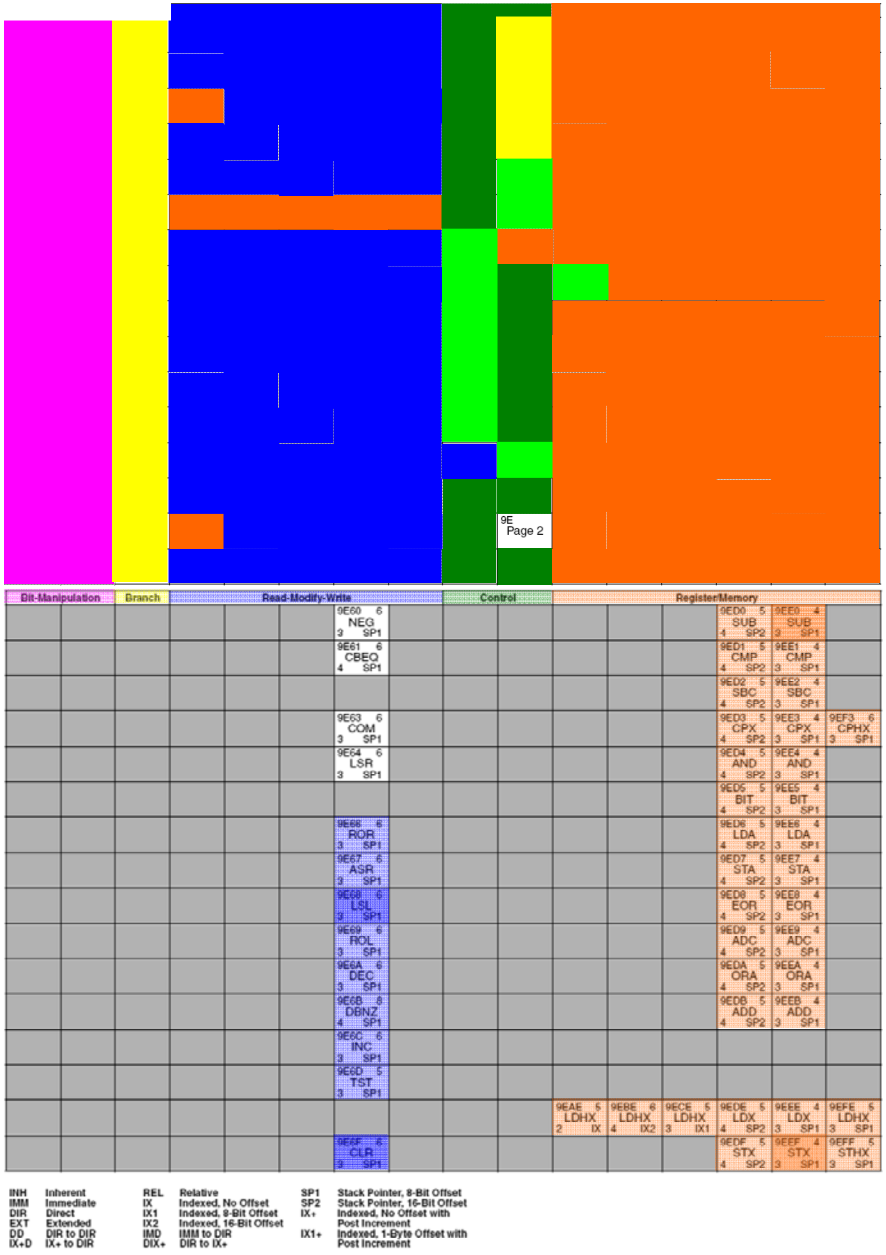
- H.2.18.5 equivalence class test

- 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.

- 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 can be formed from the following:
  - Data from a valid range
  - Data from invalid range
  - Data from the bounds
  - Extreme values and their combinations

- The tests within a group are run with different addressing modes, so that the entire group executes all addressing modes

S08 CPU Instruction Grouping



Bit-Manipulation		Branch	Read-Modify-Write					Control		Register/Memory					
						9E60 6 NEG 3 SP1							9ED0 5 SUB 4 SP2	9EE0 4 SUB 3 SP1	
						9E61 6 CBEQ 4 SP1							9ED1 5 CMP 4 SP2	9EE1 4 CMP 3 SP1	
													9ED2 5 SBC 4 SP2	9EE2 4 SBC 3 SP1	
						9E63 6 COM 3 SP1							9ED3 5 CPX 4 SP2	9EE3 4 CPX 3 SP1	9EF3 6 CPHX 3 SP1
						9E64 6 LSR 3 SP1							9ED4 5 AND 4 SP2	9EE4 4 AND 3 SP1	
													9ED5 5 BIT 4 SP2	9EE5 4 BIT 3 SP1	
						9E66 6 ROR 3 SP1							9ED6 5 LDA 4 SP2	9EE6 4 LDA 3 SP1	
						9E67 6 ASR 3 SP1							9ED7 5 STA 4 SP2	9EE7 4 STA 3 SP1	
						9E68 6 LSL 3 SP1							9ED8 5 EOR 4 SP2	9EE8 4 EOR 3 SP1	
						9E69 6 ROL 3 SP1							9ED9 5 ADC 4 SP2	9EE9 4 ADC 3 SP1	
						9E6A 6 DEC 3 SP1							9EDA 5 ORA 4 SP2	9EEA 4 ORA 3 SP1	
						9E6B 8 DBNZ 4 SP1							9EDB 5 ADD 4 SP2	9EEB 4 ADD 3 SP1	
						9E6C 6 INC 3 SP1									
						9E6D 5 TST 3 SP1									
										9EAE 5 LDHX 2 IX	9EBE 6 LDHX 4 IX2	9ECE 5 LDHX 3 IX1	9EDE 5 LDX 4 SP2	9EEE 4 LDX 3 SP1	9EFE 5 LDHX 3 SP1
						9E6F 6 CLR 3 SP1							9EDF 5 STX 4 SP2	9EEF 4 STX 3 SP1	9EFF 5 STHX 3 SP1

INH	Inherent	REL	Relative	SP1	Stack Pointer, 8-Bit Offset
IMM	Immediate	IX	Indexed, No Offset	SP2	Stack Pointer, 16-Bit Offset
DIR	Direct	IX1	Indexed, 8-Bit Offset	IX+	Indexed, No Offset with Post Increment
EXT	Extended	IX2	Indexed, 16-Bit Offset		
DD	DIR to DIR	IMD	IMM to DIR	IX1+	Indexed, 1-Byte Offset with Post Increment
IX+D	IX+ to DIR	DIX+	DIR to IX+		

Note: All Sheet 2 Opcodes are Preceded by the Page 2 Prebyte (9E)

Prebyte (9E) and Opcode in Hexadecimal — 9E60 6 — HCS08 Cycles
NEG — Instruction Mnemonic
Number of Bytes — 3 SP1 — Addressing Mode

- The S08 instructions were analyzed and placed into the 6 different groups (as shown in instruction map diagrams below:

- Register/Memory Tests
- Control
- Read Modify Write
- Branch
- Bit Manipulation
- Stack Pointer

S08 CPU Instruction Test

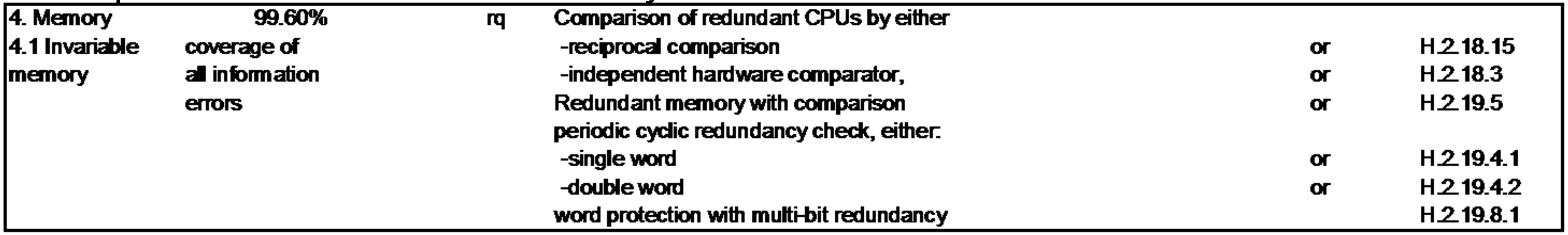
- Memory Footprint: 2148 bytes (this can be reduced if instructions are not utilized in application code)
- Execution Time: 3666 CPU BUS cycles (183.3 us at 20MHz)
- Reviewed, tested and certified by Tuev-Sued GmbH

TÜV SÜD

- Instructions not tested: (as they require hardware considerations)
- STOP WAIT BGND BIH BIL RSP SWI

4.1 Invariable Memory – 99% Coverage for all Information Errors

Acceptable measures for class C systems are:

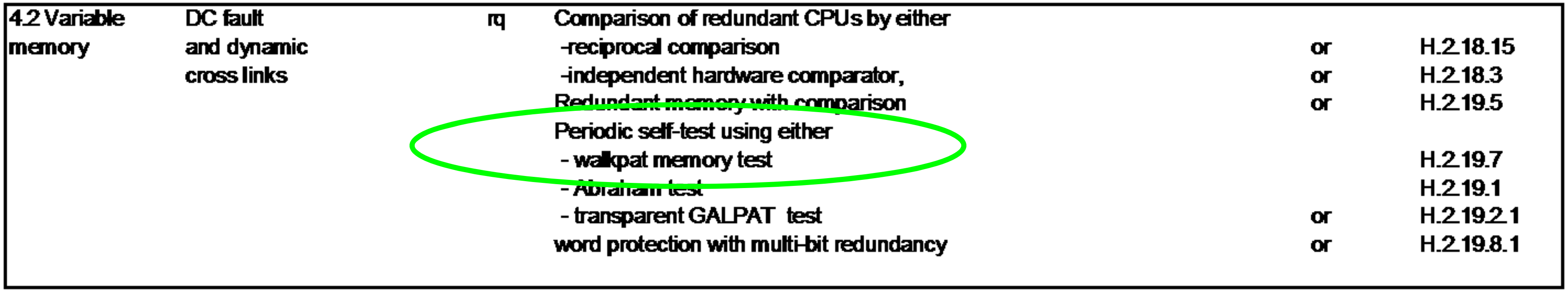


4. Memory	99.60%	rq	Comparison of redundant CPUs by either		
4.1 Invariable memory	coverage of all information errors		-reciprocal comparison	or	H.2.18.15
			-independent hardware comparator,	or	H.2.18.3
			Redundant memory with comparison	or	H.2.19.5
			periodic cyclic redundancy check, either:		
			-single word	or	H.2.19.4.1
			-double word	or	H.2.19.4.2
			word protection with multi-bit redundancy		H.2.19.8.1

- ► Dual CPU/MCU implementation
- ► Redundant Memory with comparison – two areas of flash that can be regularly
- ►checked with each other, or executed from and result compared
- ► Periodic cyclic redundancy check – 16-bit or 32-bit CRC
- ► Word protection with multi-bit redundancy – ECC hardware

4.2 Variable Memory

Acceptable measures for class C systems are:



4.2 Variable memory	DC fault and dynamic cross links	rq	Comparison of redundant CPUs by either		
			-reciprocal comparison	or	H.2.18.15
			-independent hardware comparator,	or	H.2.18.3
			Redundant memory with comparison	or	H.2.19.5
			Periodic self-test using either		
			- walkpat memory test		H.2.19.7
			- Abraham test		H.2.19.1
			- transparent GALPAT test	or	H.2.19.2.1
			word protection with multi-bit redundancy	or	H.2.19.8.1

- ▶IEC 60730 Class C Requirement to test
- ▶Variable memory (RAM) for DC faults

- ▶Acceptable measure to test is:

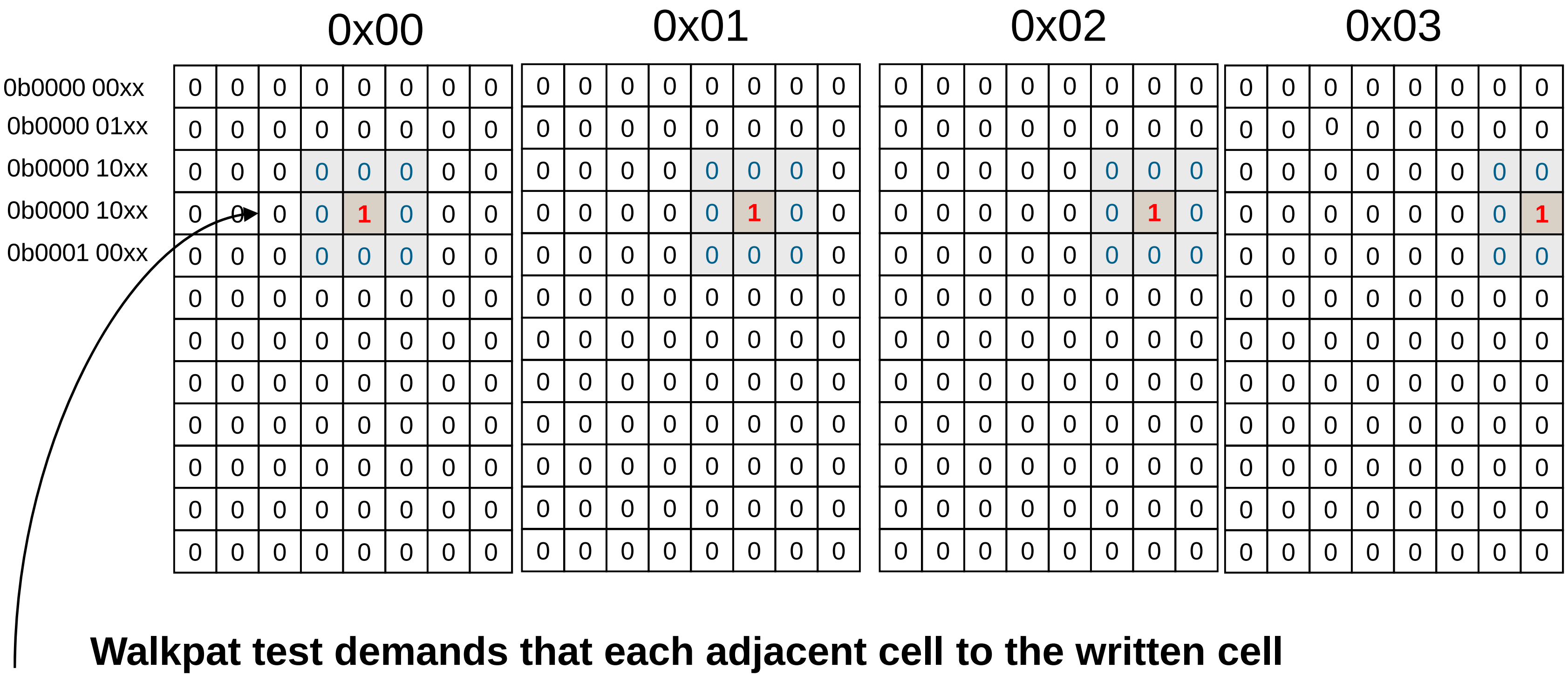
- ▶Periodic self-test using “walkpat memory test”

H.2.19.7 Equivalence Class Test

- H.2.19.7 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 operation. A bit inversion is performed on the first cell and the remaining memory areas 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 memory under test and preceding as above.
- This technique recognizes all static bit errors as well as errors in interfaces between memory cells

A walking 1s pattern followed by a walking 0s pattern

Walk Pat RAM Test

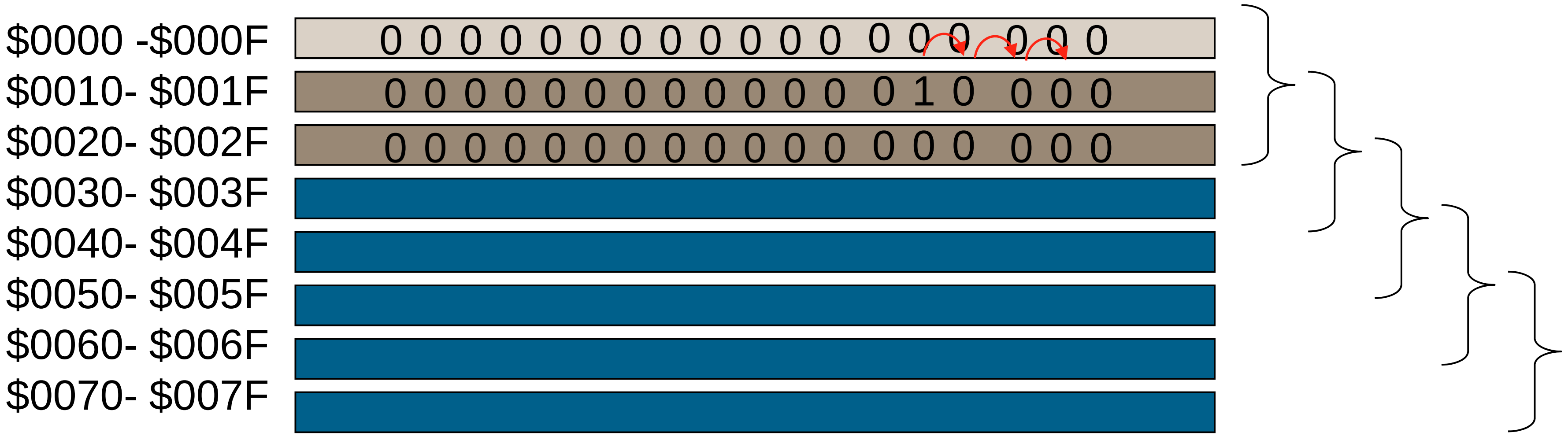


	0x00								0x01								0x02								0x03							
0b0000 00xx	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0b0000 01xx	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0b0000 10xx	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0b0000 10xx	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
0b0001 00xx	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Walkpat test demands that each adjacent cell to the written cell is checked to have the opposite state

Two things are required to ensure speedy execution times in application
1) RAM split into sizeable segments
2) Need to understand the RAM topology to ensure that the walking 1s pattern is testing the adjacent cells as intended

Walking 1s



$0000 -$000F
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

$0010- $001F
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0

$0020- $002F
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

$0030- $003F

$0040- $004F

$0050- $005F

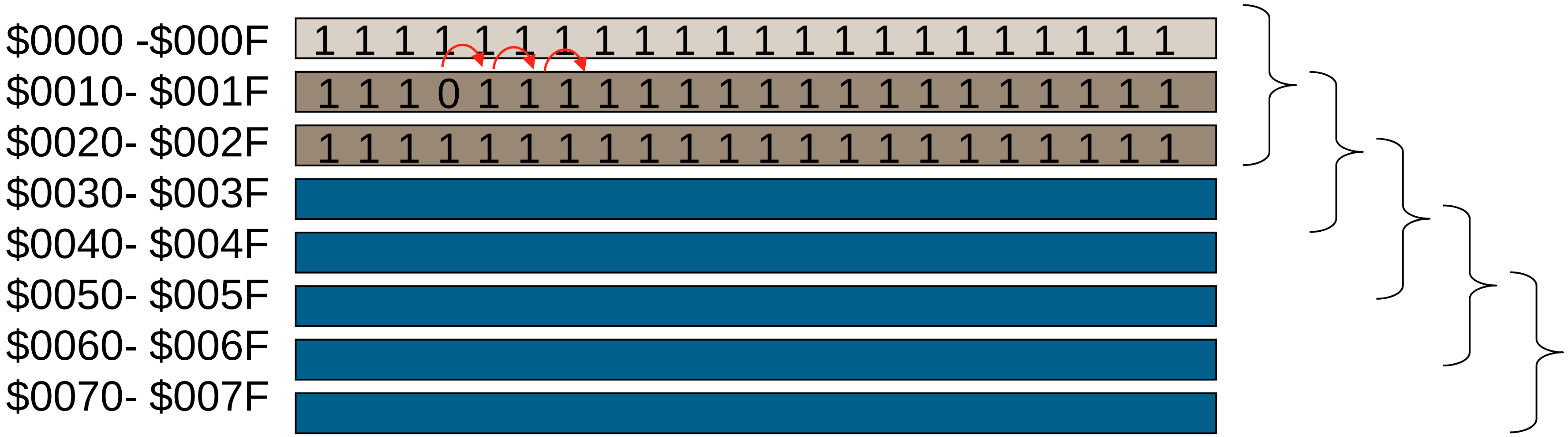
$0060- $006F

$0070- $007F

0 0 0
0 1 0
0 0 0

When cell set to 1
The 8 adjacent cells to the
Test cell are verified to be 0

Walking 0s



$0000 -$000F 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

$0010- $001F 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

$0020- $002F 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

$0030- $003F

$0040- $004F

$0050- $005F

$0060- $006F

$0070- $007F

```
1 1 1
1 0 1
1 1 1
```

When cell set to 0
The 8 adjacent cells to the
Test cell are verified to be 1

Walking 1s RAM Test

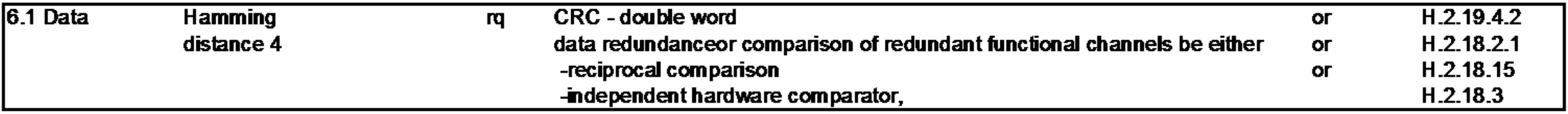
Memory footprint: Walking 1s only: 1245 bytes
Walking 1s and 0s: 2174 bytes

Execution time for 16 byte row:

Walking 1s	12544 CPU cycles	(627uS@20Mhz)
Walking 1s+0s	27016 CPU cycles	(1.35ms@20Mhz)

Execution time for 2048 bytes (16 bytes at a time)
Walking 1s+0s 2.765 seconds at 20 Mhz

6.1 External Communication Data – Hamming Distance 4



6.1 Data	Hamming	rq	CRC - double word	or	H.2.19.4.2
	distance 4		data redundanceor comparison of redundant functional channels be either	or	H.2.18.2.1
			-reciprocal comparison	or	H.2.18.15
			-independent hardware comparator,		H.2.18.3

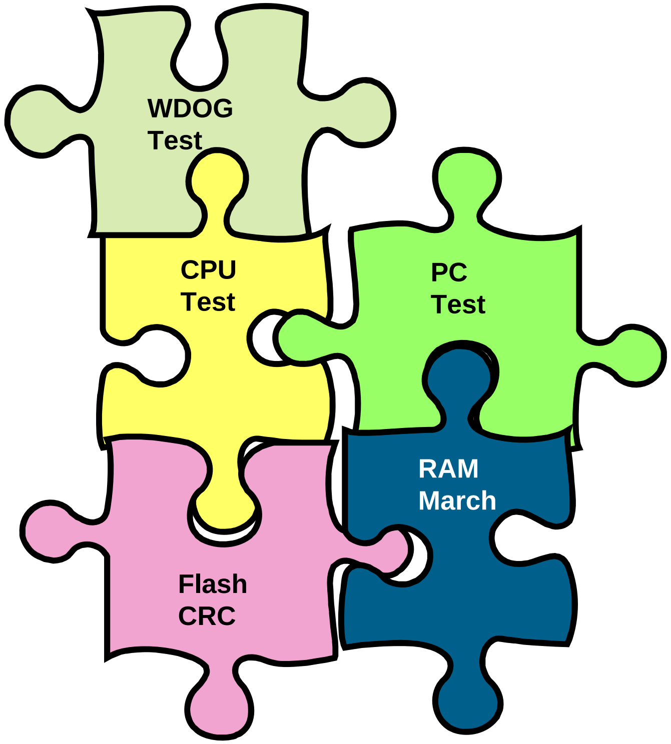
- ► CRC double word – 32-bit CRC of data transmitted/received

- ► Data redundancy with comparison – send data twice and comparison

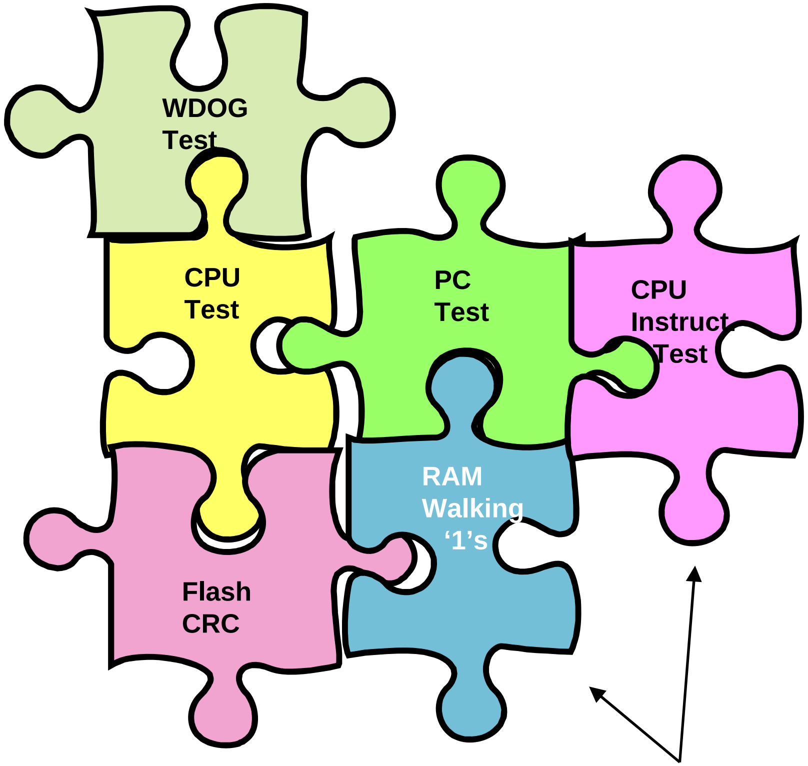
- ► Comparison of redundant function channels – use two interface mediums and compare receptions with each other

Freescale Offerings

Freescale will Provide Pieces of the 60730 Jigsaw

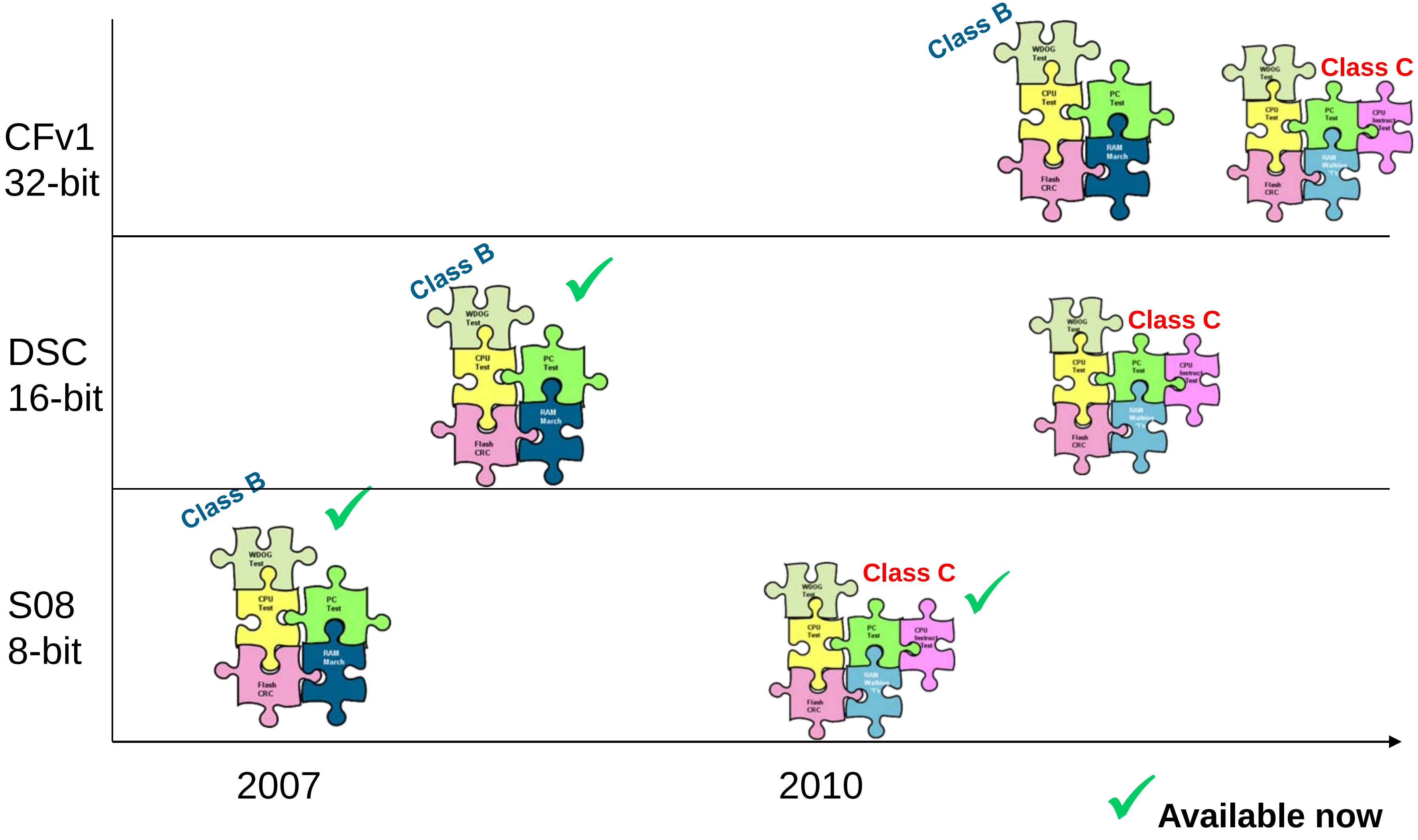


Class B Routines



Class C Routines

Freescale 60730 Software Roadmap



CFv1
32-bit

Class B

Class C

DSC
16-bit

Class B ✓

Class C

S08
8-bit

Class B ✓

Class C ✓

2007

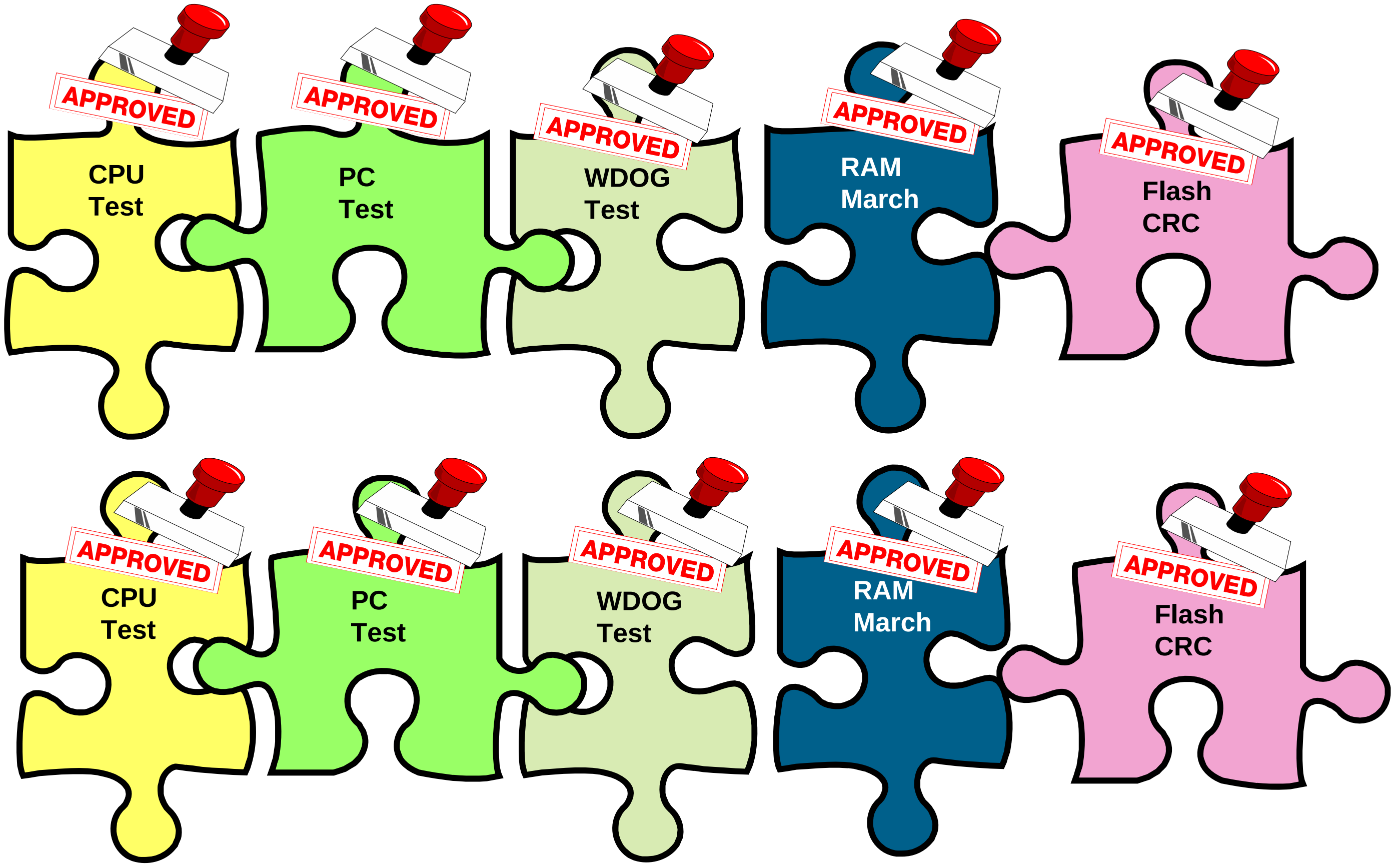
2010

✓ Available now

V.D.E. Approved IEC60730 Class B Safety Software Routines from Freescale

MC9S08ACxx

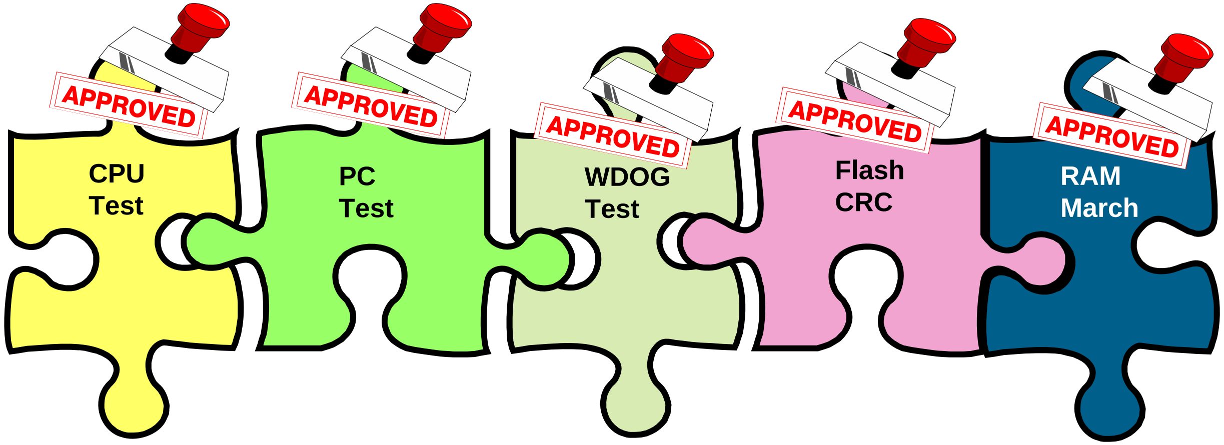
DSC56F80xxxx



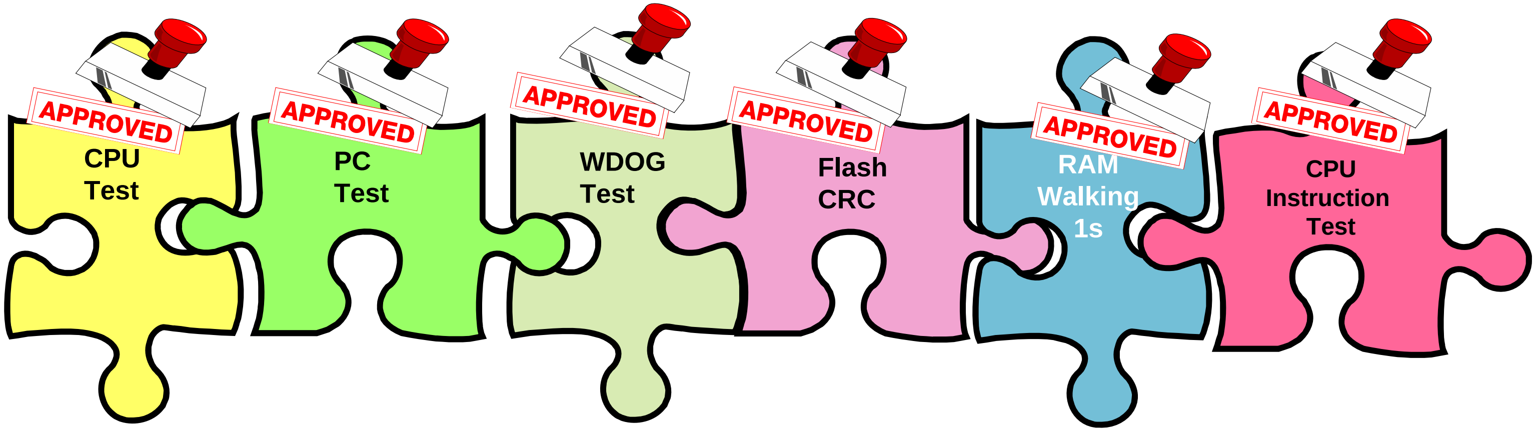
“All pieces have been certified by VDE to help accelerate manufacturer development of Automatic Controls”

Approved IEC 60730 Safety Software Routines S08ACxx

CLASS B



CLASS C



For S08 we have both class B and class C certified routines

Generic MCU Requirements for IEC/UL 60730

Class B

Hardware

Independent Clocked WDOG
Independent real time interrupt

Software

CPU Register “SA faults” Test
March C and MARCH X (transparent) RAM Test
Modified Checksum or CRC Flash Test.
Independent WDOG Test
Plausibility Tests

Time Slot monitoring of program flow and interrupt behavior

Class C

Hardware

Independent Clocked WDOG
Independent real time interrupt
2nd CPU or CPU Instruction Test
CRC engine

Software

CPU Register “walkpat” Test
CPU Instruction Set Test
GALPAT/walking 1’s RAM Test
CRC Flash Test.
Independent WDOG Test
Plausibility Tests

Time Slot monitoring of program flow and interrupt behavior

IEC/UL 60730 Summary

- **To help manufacturers gain 60730 compliance easier, MCUs are expected to have:**

  For Class B

  - **An independent clocked watchdog**
  - **An independent clocked periodic interrupt**
  - **CRC engine (in hardware for >64Kbyte devices)**
  - **Software**
    - **Watchdog Timeout Test**
    - **CPU Register Test**
    - **RAM March Test**
    - **Flash CRC Signature Test**

  For Class C (in addition to Class B)

  - **Redundant CPU – with comparison – for complex safety systems**
  - **CPU Instruction Test (software or hardware)**
  - **ECC on RAM or Walking 1s0s Software Test Routine**
  - **Freescale provides software routines to test RAM, Flash, CPU Instruction decode, Watchdog Timeout and Reset**

freescale™
semiconductor