Generating Test cases through Markov Models for undertaking Comprehensive Testing of Embedded Systems

D. Bala Krishna Kamesh[#1], Dr. A. K. Vasishtha[#2], Dr. JKR Sastry[#3] and Dr. V Chandra Prakash[#4]
[1]Scholar, ShriVenkateshwara University, Gajraula, [2] Professor, ShriVenkateshwara University, Gajraula,
[3,4]KL University, Vaddeswaram, Guntur district,

Abstract- Cleanroom Software Engineering (CRSE) methodology has incorporated in it, the Markov models based on which the number of test cases that should be used for testing can be determined. Model based statistics are developed based on Markov model and the same are used for determining the number of test cases that should be used for testing each path contained in a Usage model.A very few methods have been presented in literature which define the actual process to be used for generating the test cases especially considering the testing of embedded systems. Every embedded system should be tested comprehensively and testingmust include both hardware and software states that exist in every path of the Markov model. Testing embedded systems comprehensively requiresusage of many methods which includeScaffolding, Logical analysis, etc. and undertaking testing at different locations. In this paper, a formal method has been presented which generates test cases using Markov model for achieving comprehensivetesting of an embedded system.

Keywords- Markov model, usage model, Comprehensive Testing, Embedded Systems, Clean Room Software Engineering

I. Introduction

Development and testing of embedded software is difficult as the software consists of a large number of concurrently executing and interacting tasks. Each task in embedded software is executed at different timeintervals under different conditions and with different timing requirements. Furthermore,the time available to develop and test an embedded system is usually quite limited due to the relatively short lifetime of the products.

Cost-effective testing of embedded software is of critical concern in maintaining competitive edge. Testing an embedded system manually is quite time consuming and a costly proposition. Tool based testing of an embedded system is to be considered and put into use to reduce the cost of testing and also complete the testing of the system rather earlier.

Testing and debugging an embedded system is difficult and time consuming for simple reason that the embedded system has neither storage device nor adequate user interface. The users are extremely intolerable of buggy embedded systems. Embedded systems deal with external environment by way of sensing the physical parameters and also must provideoutputs that control the external environment.

In embedded systems, the issue of testing must consider both hardware and software. The mal-functioning of hardware can be detected through software failures. The TARGETembedded system does not support hardware and software platforms needed for testing the software, hardware and both. The testing of an embedded system cannot be done just using the TARGET due lack of resources such as user interface, storage etc. which are required for undertaking actual testing. The software is developed on HOSTmachine and the major portion of testing is carried on the HOST before the software is moved to the TARGET machinefor continuing further testing.

Comprehensive testing of an embedded system requires pre-identification of various test methods, the location of carrying testing and the kind of testing that can be conducted using specific method. Comprehensive testing includes testing Hardware,

The entire embedded system application code can be divided primarily into components namely Hardware independent code and Hardware dependent code. Hardware independent code is a set of tasks that perform mundane housekeeping and data processing, whereas the hardware dependent code is either interrupt service routines or the drivers that control the operations of the device. It is necessary to identify different types of test cases that test both Hardware independent code and Hardware dependent code. The testing techniques and the testing locations where testing must be carried should address the requirements of testing both the hardware independent and hardware dependent code.

ES software development process is undertaken on a HOSTcomputer, since the TARGET machine will not have sufficient resources to undertake the software development and testing. On completing the

development and undertaking the testing to certain extent, the code is then copied on to the ROM, EPROM or Flash memory of the TARGET and then the hardware is again tested along with the HOST. The Embedded system is connected to the production system and the testing is carried again. Testing the embedded system after connecting it to the production system is absolutely necessary as any amount of testing carried by simulating the events initiated by the production system will not clearly depict the working of the production system. The real production system may initiate many unforeseen and uncommon events which may not have been considered at the time of developing the system.

If an analysis is carried on the entire code of an ES application, it will be revealed that 80% of code is hardware independent code and the rest is hardware dependent code. It is easy, faster, and cost effective to complete testing of the hardware independent code on the HOST machine itself.

The testing of the hardware independent code can be carried at the HOSTby using any of the techniques that include Scaffolding, Assert Macros and Instruction set simulators.A section of the code is completely dependent on hardware and cannot be tested without the TARGETmachine. Again hardware dependent code cannot be tested by way of only using the TARGET due to lack of resources required to carry out testing. Therefore,testing of hardware dependent code is to be carried along with the HOSTmachine.

Testing the hardware requires integration of hardware testing gadgets with the TARGETmachine &HOST. The gadgets must be fed with inputs from the HOSTfor undertaking particular type of Hardware testing. The Hardware testing gadgetsare connected to the TARGET through probes and the TARGET is connected to the HOST using RS232C or RJ45 physical interface. Test cases are fed to a Logic Analyzer through HOST and the Logic analyzer after conducting the test shall make available the Test results to the HOST.

Some of the testing required by an embedded system cannot be conducted by using software alone. Hardware equipment like Logic Analyzers is required to carry testing of complex mechanisms that need specific timing of the signals. The timing of various signals must be measured and presented in a graphical form. Sometimes, the time duration during which the signals are valid must also be measured. For proper processing, the timing of the signals and the sequence in which signals are processed must be tested. The triggering of the signals whether edge triggered or level triggered and the voltage levels of the signals must be measured and stored in order to prepare and display timing of the signals. The occurrence of the events can be tested by way of sensing the change in the level or edge of a signal.

A microcontroller which is a part of an embedded system at times is replaced by an In-Circuit Emulator in the TARGETmachine. The Emulation Software is stored in a separate memory different from application memory and it maintains the execution status of the application in its memory. The execution of the application is controlled by the emulation software. The emulation software also will have a communication software component to communicate with the HOST. HOST can feeda series of test cases to In-Circuit Emulator and the test execution results are sent back to the HOST.

Various types of testing are to be conducted to ensure that anembedded system is built properly. The testing types include unit testing, Environment testing, Integration testing, END-TO-END testing, Regression testing, etc.

Testing requirements can be projected right in the beginning of the development of the system. The testing scenarios can be projected from the end user's point of view. Test scenarios involve conducting any testing type.

An Embedded system will undergo changes either due to change in hardware, software or both. When changes are made, the same are to be tested quite rapidly. The change made at one location may affect some other locations of hardware, or software or both. The changes must be traced to Hardware or Software or both and all affected components are to be tested thoroughly. It is necessary to select all the test cases that are related to changes made to the affected components.

The type of testing to be done requires a decision on the location at which the testing is to be carried and also the kind of test method that should be used for undertaking testing. The test cases are generated to carry testing at a location using a test method conforming to the type of testing required.

To carry testing of an embedded system comprehensively, test cases are to be generated that cater for testing hardware and software that are associated with every path of the usage model. In literature, very few methods have been proposed[7], [9] which help generating the test cases. The methods

do not address the issue of testing both the hardware and the software of the embedded systems comprehensively. This paper presents a method that helps testing the embedded system comprehensively. The method is developed considering that it fits into CRSE methodology and the method uses Markov model effectively for the generation of test cases.

II. RELATED WORK

Testing modules of embedded systems can be carried by isolating the modules at run time and carry the testing[4]. This method has, however, failed to test the occurrence of events in a particular sequence.

Testing of embedded software is limited by the design of the hardware.When the hardware evolves, agile methods work well provided multiple test strategies are used [6]. This has powerful implications for improving the quality of high reliability systems, which commonly have embedded software at their heart.

END-TO-END Integration testing [10] of an embedded system requires specifying test scenarios as thin threads; each thread representing a single function. Interaction scenarios can be considered [5] for conducting integration testing since the integration testing must consider sequence of external input events and internal interactions.

Regression testing [10] has been a popular quality testing technique. Itis based on either code or software design. Functional regression testing can be carried based on test scenarios. A Web based tool has also been developed to undertake regression testing.

Testing of embedded system can be undertaken by simulating the hardware on the HOST and integrating the ES software with the simulators[4]. This approach, however, will not be able to deal with all kinds of test scenarios related to hardware. The complete behavior of hardware, specially unforeseen behavior, cannot be simulated on a HOST machine.

Verification patterns[11] can be used for undertaking the testing of the embedded systems, the key concept of this being recognizing the scenarios into patterns and applying the testing approach whenever similar patterns are recognized in any embedded application. But, the key to this approach is the ability to identify all test scenarios that occur across all types of embedded applications.

Even though several authors have suggested several approaches to conducting several types of testing, no attempt has been made so far to identify all possible test scenarios considering both hardware and software. Many tools are also available in the market for testing embedded applications to carry fragments of testing and even the fragments of the testing are not done in a unified manner. The tools failed to address the comprehensive testing requirements considering both hardware and Software.

III. TEST CASE GENERATION FRAMEWORK

The generation of test cases for testing the embedded systems comprehensively requires construction of a usage model based on stimulus-response sequences [2] and development of a Markov model by way of assigning probability of occurrence of arcs emanating from each node of the usage model[3].

Test cases are to be generated for each of the paths contained in the usage model considering both hardware and software states that exist on the path. The paths which are generated considering the Markov model that has been built for TMCNRS (Monitoring and controlling the temperatures within the Nuclear Reactor Systems)are shown in the Table I.

Different types of test methods and the locations where testing is to be carried are shown in the Table II. For comprehensive testing of embedded systems, all the test case types that are required to be tested must be recognized quite ahead and a repository of the same must be maintained [8]. The repository can be updated from time to time. The test case types that must be used are shown in the Table III.

Some test gadgets such as logic analyzer have to be used for undertaking the testing of the hardware and the embedded software. The gadgets are to be fed with a command and command line arguments that represent a test case. Testing the functioning of the hardware is normally undertaken by using a command language and defining the command line arguments. The test gadgets provide the output when the command and the command line arguments are fed. Logic Analyzer (CDLOGIC) and a simulator (SIMCS51) have been used for testing the hardware and software related to TMCNRS.

The command language to be used for testing the hardware considering CDLOGIC and SIMSC51 is shown in the Table IV.

Both HW states and SW states exist in each and every path of the usage model. The SW states are program slices that operate on a single data and many

control dependencies[1]. Hardware states encapsulate internal processing. All the states, both hardware and software that exist on the paths of the Markov model can be identified by using Table I.

Tracing the source code is carried to identify the input variables and output variables that are associated with SW states (Program slices). Table V shows the program slices and their related Input/output variables.

The type of testing that should be conducted considering hardware or a software state can be determined by mapping the test case types to the respective states. Table VI shows the test types to be used for conducting testing in relation to hardware devices and program slices that are contained in different paths of the Markov model.

Test cases can be generated using the following algorithm.

1. Read all the paths in the usage model into anarray
2. For each of the paths in the usage model
   - 2.1 Estimatethe number of test cases that should be generated for each of the paths contained in the Markov Model
   - 2.2 Count the number of hardware devices and software slices that are contained in each of the paths form Table I
   - 2.3 List the Hardware states and software slices that are included in each of the paths from the repository as shown Table I
   - 2.4 Apportion total test cases for each of the paths evenly among the HW and Software states
   - 2.5 For each of the hardware devicesthat exist in a path, select related test case types from Table II and generate as many test cases required as estimated for each of the paths.
   - 2.6 For each of the program slices, trace the associated input variables and expected output variables as shown in the Table V
   - 2.7 For each of the software slicesthat existin a path, select related test case types from Table VI and generate as many test cases required by selecting proper tests values in respect of the input variables that are associated with the program slices as shown in the Table V.

The test cases generated for testing some of the paths contained in the usage model have been shown in the Table VI In all,653 Test cases have been generated and only a few of the test cases generated have been shown in the Table VI.

IV. CONCLUSIONS

Comprehensive testing of the embedded systems requires testing of HW, SW and both. Test case types that should be used for undertaking the comprehensive testing can be pre-identified and mapped to the HW devices and SW slices that act as states in the usage model. The actual test cases can be generated based on the total number of test cases that must be generated for each of the paths in the usage model. The test cases can be sequenced as per the sequence of occurrence of HW and SW states in each of the paths.

REFERENCES

[1] D.Bala Krishna Kamesh, Prof. A.K. Vasishtha, Dr. JKR. Sastry and Dr. V. Chandra Prakash,"Estimating the Failure rates of Embedded Software through Program Slices", *International Journal of Systems and Technology*, Vol. 6, Issue 1,pp.49-58,2013.

[2] D. Bala Krishna Kamesh, Dr. A. K. Vasishtha, Dr. JKR Sastry and Dr. V. Chandra Prakash, "Developing Usage Models through Stimulus Response Sequences", *International Journal of Advances in Science and Technology*, Vol. 7, No. 4, 2013.

[3] D. Bala Krishna Kamesh, Dr. A. K. Vasishtha, Dr. JKR Sastry and Dr. V. Chandra Prakash, "Assessing the Probability of Failure and Distribution of Test Cases among Paths in the Usage Model", (Communicated) 2013.

[4] Jacobson, Booch G and Rumbaugh J, "The Unified Software Development Process", Addison *Wesley*, Reading, MA, 1999.

[5] Lee N.H and Cha S.D, "Generating Test Sequences from a set of MSCs", *Computer Networks*, 2003.

[6] Nancy Van Schooenderwoert, "Taming the embedded Tiger-Agile Test Technique for embedded Software", *IEEE Proceedings of the Agile Development Conference ADC*, 2004.

[7] Prowell S.J.,"JUMBL: A tool for Model Based Statistical Testing", *proceedings of the 36th Hawaii International Conference on System Science*, 2002.

[8] Sastry J.K.R, Rajasekhara Rao K, and SasiBhanu J,"Comprehensive requirements specification of a Cost effective testing Tool", *Proceedings of CSI National Conference on Software Engineering, NCSOFT-2007*,pp.73-85, 2007.

[9] Sergiy A. Vilkomir, Thomas Swain and Jesse H. Poore, "Combinatorial test case selection with Markovian usage models", *Fifth International Conference on Information Technology: New Generations*,2008.

[10] Tsai W.T, Bai X, Paul R and Yu L, " Scenario-Based Function Regression Testing", *Proc. of IEEE COMPSAC*,pp.496-501,2001.

[11] Tsai W.T, Yu L, Paul R and Saim A, "Scenario-Based Object-Oriented Test Frameworks for Testing Distributed Systems",*IEICE Transactions*,2003.

TABLE I
TEST PATHS DERIVED FROM THE MARKOV MODEL RELATED TO TMCNRS

Test Path Number	Test Path Description	State on the path	State serial Number	HW / SW State
Path-1	Write Initial message to LCD	Reset	1	HW
		Microprocessor	13	HW
		Display Initial Message	14	SW
		LCD	50	HW
Path-2	Write enter password message on LCD	Reset	1	HW
		Microprocessor	13	HW
		Display password enter message	15	SW
		LCD	51	HW
Path-3	Read Key1 and write to LCD	Reset	1	HW
		Key1	2	HW
		Keyboard	7	HW
		A/D Converter	12	HW
		Microprocessor	13	HW
		Read key	16	SW
		Write key	17	SW
		LCD	51	HW
Path-4	Read Key2 and write to LCD	Reset	1	HW
		Key1	2	HW
		Key2	3	HW
		Keyboard	7	HW
		A/D Converter	12	HW
		Microprocessor	13	HW
		Read key	16	SW
		Write key	17	SW
		LCD	51	HW
Path-5	Read Key3 and write to LCD	Reset	1	HW
		Key1	2	HW
		Key2	3	HW
		Key3	4	HW
		Keyboard	7	HW
		A/D Converter	12	HW
		Microprocessor	13	HW
		Read key	16	SW
		Write key	17	SW
		LCD	51	HW
Path-6	Read Key4 and write to LCD	Reset	1	HW
		Key1	2	HW
		Key2	3	HW
		Key3	4	HW
		Key4	5	HW
		Keyboard	7	HW
		A/D Converter	12	HW
		Microprocessor	13	HW
		Read key	16	SW
		Write key	17	SW
		LCD	51	HW
Path-7	Read Key5 and write to LCD	Reset	1	HW
		Key1	2	HW
		Key2	3	HW
		Key3	4	HW
		Key4	5	HW
		Key5	6	HW
		Keyboard	7	HW
		A/D Converter	12	HW
		Microprocessor	13	SW

Test Path Number	Test Path Description	State on the path	State serial Number	HW / SW State
Path-7	Read Key5 and write to LCD	Read key	16	SW
		Write key	17	SW
		LCD	51	HW
Path-8	Compare password and write password mismatch on to LCD	Reset	1	HW
		Key1	2	HW
		Key2	3	HW
		Key3	4	HW
		Key4	5	HW
		Key5	6	HW
		Keyboard	7	HW
		A/D Converter	12	HW
		Microprocessor	13	HW
		Compare Password	18	SW
		Write Password mismatch	19	SW
		LCD	51	HW
Path-9	Read Ref1 Temperature and write to LCD	Reset	1	HW
		Key1	2	HW
		Key2	3	HW
		Key3	4	HW
		Key4	5	HW
		Key5	6	HW
		Keyboard	7	HW
		A/D Converter	12	HW
		Microprocessor	13	HW
		Compare Password	18	SW
		Request HOST for Ref1	20	SW
		RS232C	21	HW
		HOST	22	HW
		Read HOST for Ref1	23	SW
		WriteRef1 to LCD	24	SW
		LCD	51	HW
Path-10	Read Ref2 Temperature and write to LCD	Reset	1	HW
		Key1	2	HW
		Key2	3	HW
		Key3	4	HW
		Key4	5	HW
		Key5	6	HW
		Keyboard	7	HW
		A/D Converter	12	HW
		Microprocessor	13	HW
		Compare Password	18	SW
		Request HOST for Ref1	20	SW
		RS232C	21	HW
		HOST	22	HW
		Read HOST for Resf1	23	SW
		WriteRef1 to LCD	24	SW
		Request HOST for Ref2	25	SW
		RS232C	26	HW
		HOST	27	HW
		Read Ref2 from HOST	28	SW
		WriteRef2 to LCD	29	HW
		LCD	51	HW

Test Path Number	Test Path Description	State on the path	State Serial Number	HW / SW State
Path-11	Read Temp1 and write to LCD	Sen1	8	HW
		OP1	10	HW
		A/D Converter	12	HW
		Microprocessor	13	HW
		Temp1Read	30	SW
		WriteTemp1toLCD	31	SW
		LCD	51	HW
Path-12	Read Temp1 and send to HOST	Sen1	8	HW
		OP1	10	HW
		A/D Converter	12	HW
		Microprocessor	13	HW
		Read Temp1	30	SW
		Send Temp1 to HOST	32	SW
		RS232C	52	HW
		HOST	53	HW
Path -13	Compare Temp1 with Ref1 Temperature and set Pump1 ON	Sen1	8	HW
		OP1	10	HW
		A/D Converter	12	HW
		Microprocessor	13	HW
		Read Temp1	30	SW
		Compare Temp1 with Ref1	33	SW
		Pump1ON	38	SW
		Relay1	40	HW
		Pump1	41	HW
Path-14	Compare Temp1 with Ref1 Temperature and set Pump1OFF	Sen1	8	HW
		OP1	10	HW
		A/D Converter	12	HW
		Microprocessor	13	HW
Path-14	Compare Temp1 with Ref1 Temperature and set Pump1 ON	Read Temp1	30	SW
		Compare Temp1 with Ref1	33	SW
		Pump1OFF	39	SW
		Relay1	40	HW
		Pump1	41	HW
Path-15	Read Temp2 and write to LCD	Sen2	9	HW
		OP2	11	HW
		A/D Converter	12	HW
		Microprocessor	13	HW
		Temp2 Read	34	SW
		Write Temp2 to LCD	35	SW
		LCD	51	HW
Path-16	Read Temp2 and send to HOST	Sen2	9	HW
		OP2	11	HW
		A/D Converter	12	HW
		Microprocessor	13	HW
		Temp2 Read	34	SW
		Send Temp2 to HOST	36	SW
		RS232C	52	HW
		HOST	53	HW
Path-17	Compare Temp2 with Ref2 Temperature and set Pump2 ON	Sen2	9	HW
		OP2	11	HW
		A/D Converter	12	HW
		Microprocessor	13	HW
		Temp2 Read	34	SW
		Compare Temp2 with Ref2	37	SW
		Pump2 On	42	SW
		Relay2	44	HW
		Pump2	45	HW

Test Path Number	Test Path Description	State on the path	State Serial Number	HW / SW State
Path-18	Compare Temp2 with Ref2	Sen2	9	HW

	Temperature and set Pump2 off	OP2	11	HW
		A/D Converter	12	HW
		Microprocessor	13	HW
		Temp2 Read	34	SW
		Compare Temp2 with Ref2	37	SW
		Pump2 Off	43	SW
		Relay2	44	HW
		Pump2	45	HW
Path-19	Compare Temp1 and Temp2 and write Temp mismatch on to LCD	Sen2	9	HW
		OP2	11	HW
		A/D Converter	12	HW
Path-19	Compare Temp1 and Temp2 and write Temp mismatch on to LCD	Microprocessor	13	HW
		Temp2 Read	34	SW
		Compare Temp1 with Temp2	46	SW
		Write Temp1 and Temp2 Mismatch	47	SW
		LCD	51	HW
Path-20	Compare Temp1 and Temp2 and set buzzer On	Sen2	9	HW
		OP2	11	HW
		A/D Converter	12	HW
		Microprocessor	13	HW
		Temp2 Read	34	SW
		Compare Temp1 with Temp2	46	SW
		Buzzer ON	48	SW
		Buzzer	50	HW
Path-21	Compare Temp1 and Temp2 and set buzzer Off	Sen2	9	HW
		OP2	11	HW
		A/D Converter	12	HW
		Microprocessor	13	HW
		Temp2 Read	34	SW
		Compare Temp2 with Ref2	37	SW
		Compare Temp1 and Temp2	46	SW
		Buzzer Off	49	SW
		Buzzer	50	HW

TABLE II
TESTING METHODS AND LOCATIONS OF TESTING

Method ID	**Description**	**Location**
M1	Scaffolding	HOST
M2	Simulation	HOST
M3	Assert Macro	HOST
M4	Third Party Tool	HOST
M5	Logic Analysis	TARGET
M6	In-Circuit Emulator	HOST+TARGET
M7	Monitors	HOST+TARGET

TABLE III
TEST CASE TYPES

Test Case Type Number	Test Case Description	Test Method-ID
1.	Testing the hardware independent code simulating Input/output functions that deal with the Hardware devices	M1
2.	Testing for processing input fed by a single device through directly calling Interrupt service routines	M1
3.	Testing passage of time through calling the interrupt routine of the timer devices	M1
4.	Testing for the interaction between hardware devices	M1
5.	Testing for interaction of Task control code with hardware devices	M1
6.	Testing for interaction of multiple tasks through control tasks to provide processing through the same device	M1
7.	Testing interaction among several devices	M1
8.	Testing the chain of tasks by way of simulating the occurrence of a sequence of events	M1
9.	Testing the chain of tasks by simulating the simultaneous occurrence of a set of events	M1
10.	Testing the tasks by simulating the occurrence of uncommon events	M1
11.	Testing the interaction between the tasks with control task that deals with the devices by way of bypassing the interrupts raised by the Hardware	M1
12.	Testing the process that several tasks initiates requests to be processed by a device and the device raises an interrupt that processing is complete	M1
13.	Testing for Time delay	M1
14.	Testing the communication between multiple instances of hardware independent code	M1
15.	Testing for the proper operation of resources such as Queues, Mail Boxes and Pipes	M1
16.	Testing for the overflow condition i.e., trying to write to the Queue or Mail box or Pipe when it is already full	M1
17.	Testing for underflow condition i.e., trying to read a Queue, or Mail Box or Pipe when it is empty	M1
18.	Testing response time for each of the event based processing	M2
19.	Response time for processing the events that occur simultaneously	M2
20.	Testing significance of the bytes (Endean)	M2
21.	Testing processing 16 bit and 32 bit variables in an expression	M2
22.	Type casting	M2
23.	Testing for byte addressing	M2
24.	Testing for parity	M2
25.	Testing the built-in peripheral devices	M2
26.	Testing Reading from ROM	M2
27.	Testing Reading from RAM and writing into RAM	M2
28.	Testing Peripheral interface testing through DMA	M2
29.	Testing throughput	M2
30.	Testing proper execution of startup code	M2
Test Case Type Number	**Test Case Description**	**Test Method Method-ID**

31.	Testing for proper identification of tasks with RTOS	M2
32.	To test whether memory pools are created properly	M2
33.	To test whether the handles required related to Queues, Mail Boxes, Pipes, Events and Timing have been created within the startup code	M2
34.	To test the occurrence of shared data problem	M2
35.	Testing for the occurrence of interrupt priority inversion	M2
36.	Testing for the occurrence of dead locks	M2
37.	Testing for usage of proper devices addressed specially MAC addresses.	M3
38.	Testing for proper data bits related to the devices	M3
39.	Testing for proper setting of values for data bits	M3
40.	Testing for range of values to be contained in a variable	M3
41.	Testing for proper function calling sequences	M3
42.	Testing memory leaks due to the call to OS	M4
43.	Testing memory leaks due to the calls to third party tools	M4
44.	Testing for writing to the end of an array	M4
45.	Testing for usage of freed pointers	M4
46.	Testing for proper memory allocation without any overlapping as the system runs	M4
47.	Testing for writing to a Queue when it is full	M4
48.	Testing for reading from a Queue when it is empty	M4
49.	Testing for proper functional interaction	M4
50.	Testing for Task interactions	M4
51.	Testing for Stack allocation for every call	M4
52.	Testing for race condition	M4
53.	Testing for priority inversion	M4
54.	Testing for deadly embrace	M4
55.	Testing for passing of parameters due to function calls	M4
56.	Testing for change of value of a variable or set of variables and also recording of peak values of the variables	M4
57.	Testing for change of values of the variables by triggering certain events	M4
58.	Testing for timing of the signals	M5
59.	Testing for occurrence of signals in a proper sequence	M5
60.	Testing for validity of the signals	M5
61.	Testing for proper fetching of addresses related to instructions	M5
62.	Testing for writing of proper values to the memory addresses	M5
63.	Testing for the state of a signal when a clock input is either edge triggered or level triggered in terms of address in the address lines, data in the data lines, Read/Write on the control lines and Read enable/ Write enable signals	M5
64.	Testing for the timing of the execution of first instruction of an interrupt routine after asserting an interrupt signal by peripheral devices.	M5
Test Case Type Number	**Test Case Description**	**Test Method Method-ID**
65.	Testing for the timing of accessing of a device through filters	M5
66.	To test the sequence of occurrence of a signal	M5

67.	Testing the in-built peripheral devices	M6
68.	Testing for the response time of a function	M6
69.	Testing for weak code which is not used by the application	M6
70.	Testing for changes in data at specified memory locations	M6
71.	Testing for inter-task communication through Mail boxes, Queues and Pipes including the overflow and underflow conditions	M6
72.	Testing power saving requirements	M6
73.	To switch off the power to microprocessor and to test switching on the power to Microprocessor when any of the events occurs	M6
74.	To test the powering on and off of the peripherals through software	M6
75.	Testing for proper execution of the code after turning on and off the power in power saving mode	M6
76.	To test proper processing the Key inputs	M6
77.	To test for proper display of output on LCD	M6
78.	Testing for scanning of multiple JTAG device connections	M7
79.	Testing for multiple debugging connections to JTAG devices	M7
80.	Testing for hardware system bring up	M7
81.	Testing for faulty devices early in the development life cycle	M7
82.	Test for value at memory address 1 >memory address 2	M1
83.	Test for value at memory address 1 <memory address 2	M1
84.	Test for value at memory address 1 = memory address 2	M1

TABLE IV
COMMAND LANGUAGE INTERFACE

Command	Command Description
1ADR	Testing for byte addressing
1BYT	Testing for significance of bytes
1COV	Processing for 16 bit and 32 bit variables in expression
1CST	Testing for casting checking
1PAR	Testing for parity checking
1RAA	Reading from ROM and writing to RAM
1RES	Testing for response time computation
1RRA	Testing for reading from RAM
1THR	Testing for throughput
4TCK	Testing for change of variable due to occurrence of an event
4TIA	Testing for task interactions
4TMA	Testing for memory overlaps
4TMC	Testing for memory leaks due to third party calls
4TML	Testing for native memory leaks
4TMS	Testing for memory leaks due to RTOS
4TPK	Testing for peak value of a variable
4TPP	Testing for proper parameter passing to the function call
4TSA	Testing for proper stack allocation
5IEDT1	Testing for timing of signal
5IEDT2	Testing for validity of signals
5IEDT3	Testing for validity of signals in proper sequence
5OEDT	Testing for timing of accessing of a device
6CHD	Testing for change in data
6CHD1	Testing for Hardware bring up
6CHD2	Testing for power consumption
7FLT	Testing of faulty devices

TABLE V
INPUT VARIABLES ASSOCIATED WITH PROGRAM SLICES

Serial Number of the Program Slice	Name of the Program Slice	Input Variable-1		Input Variable-2		Input Variable-3		Input Variable-4		Output Variable-1, 2		Output Variable-3,4	
		Type	Name of the Variable	Type	Name of the Variable	Type	Name of the Variable	Type	Name of the Variable	Type	Name of the Variable	Type	Name of the Variable
1.	Write Initial Messages to LCD	CHAR	array1	CHAR	array2	CHAR	array3			CHAR	array1	CHAR	array3
										CHAR	array2		
2.	Write Password Message to LCD	CHAR	array4							CHAR	array4		
3.	Read Key ()	CHAR	key							CHAR	key		
4.	Write Key ()	CHAR	tatkal							CHAR	"*"		
5.	Compare Password ()	CHAR	tatkal	CHAR	Array5					INT	passwdsta		
6.	Write Password Mismatch to LCD	CHAR	tatkal	CHAR	passkey					INT	array5		
7.	Request HOST for Ref1	CHAR	Array6	CHAR	Array7	CHAR	array8			CHAR	Array6	CHAR	array8
										CHAR	Array7		
8.	Read Ref1 – Read from HOST	INT	Digit0	INT	Digit1					INT	Digit0	INT	Digit1
9.	Request REF2 for HOST	CHAR	Array9							CHAR	Array9		
10.	Read RefF2– Read from HOST	INT	Ascii0	INT	Ascii1					INT	Ascii0	INT	Ascii1
11.	Write ref1 to LCD	CHAR	Array12	CHAR	Array13					CHAR	Array12	CHAR	Array13
12.	Write ref2 to LCD	CHAR	Array12	CHAR	Array13					CHAR	Array12	CHAR	Array13
13.	ReadTemp1	INT	Value1	INT	Value2	INT	Temp1			INT	Temp1		
14.	Write Temp1 to LCD	INT	Temp1	CHAR	Array10					INT	Temp1		
15.	Send Temp1 to HOST	INT	Temp1							INT	Temp1		
16.	CompareTemp1with Ref1 ()	INT	Temp1	INT	Ref1	INT	TT1			INT	compsta		
17.	ProcessPump1On	INT	Temp1	INT	rref1	INT	PUMP1-STA	PUMP1		BOOL	PUMP1		

Serial Number of the Program Slice	Name of the Program Slice	Input Variable-1		Input Variable-2		Input Variable-3		Input Variable-4		Output Variable-1, 2		Output Variable-3,4	
		Type	Name of the Variable	Type	Name of the Variable	Type	Name of the Variable	Type	Name of the Variable	Type	Name of the Variable	Type	Name of the Variable
18.	ProcessPump1Off	INT	Temp1	INT	rref1	INT	Ref1	PUMP1		BOOL	PUMP1		
19.	ReadTemp2	INT	Value2	INT	Value3	INT	Temp1			INT	Temp2		
20.	Write temp2 to LCD	INT	Temp2	CHAR	Array11					INT	Temp2		
21.	Send Temp2 to HOST	INT	Temp2							INT	Temp2		
22.	CompareTemp2 With Ref2	INT	Temp2	INT	rref2	INT	TT2			INT	compsta		
23.	Write Temp Mismatch to LCD	INT	Temp2	CHAR	Array11					INT	Temp1		
24.	ProcessPump2On	INT	Temp2	INT	rref2	INT	Ref2	PUMP2		BOOL	PUMP2		
25.	ProcessPump2Off	INT	Temp2	INT	rref2	INT	Ref2	PUMP2		BOOL	PUMP2		
26.	Process Temp1 and Temp2	INT	Temp1	INT	Temp2					INT	Temp1	INT	Temp2
27.	Process Buzzer On	BOOL	Buzzer							Bool	Buzzer		
28.	Process Buzzer Off	BOOL	Buzzer							Bool	Buzzer		

TABLE VI
TEST CASE TYPE MAPPING AND GENERATED TEST CASES

Test Path Number	Test Path Description	State on the path	State serial	HW / SW State	Test Type Serial Number	Test Type Description	Input Variable / Command	Input Value	Output Variable	Expected output value
Path -13	Compare Temp1 with Ref1 Temperature and set Pump1 ON	**Sen1**	**8**	**HW**	**81**	Testing for Faulty devices early in the development life cycle	COM-7FLT-SENS1		SENS2-STA	**0/1**
					58	Testing for Timing of the signals	COM-5IEDT-SENS1		SENS2-TIME	**0-50**
					59	Testing for occurrence of signals in a proper sequence	COM-5IEDT3-SENS1-SENS2		SENS1-SENS2-SEQ	**12/21**
		OP1	**10**	**HW**	**81**	Testing for Faulty devices early in the development life cycle	COM-7FLT-OP1		OP2-STA	**0/1**
					60	Testing for validity of the signal	COM-5IEDT2-OP1		OP2-SIGVAL	**0/1**
					59	Testing occurrence of signals in a proper sequence	COM-5IEDT3-OP1-OP2		OP1-OP2-SEQ	**12/21**
		A/D Converter	**12**	**HW**	**81**	Testing for Faulty devices early in the development life cycle	COM-7FLT-ATOD		ATOD-STA	**0/1**
					60	Testing for validity of the signal	COM-5IEDT2-ATOS		ATOD-SIGVAL	**0/1**
		Microprocessor	**13**	**HW**	**81**	Testing for Faulty devices early in the development life cycle	COM-7FLT-MP		MP-STA	**0/1**
					70	Testing for changes in data at specified at memory Locations	COM-6CHD	#1001	#1001	**0-255**
						Testing for changes in data at specified at memory Locations	COM-6CHD	#1002	#1002	**0-255**
		Read Temp1	**30**	**SW**	13	Testing for Time delay	**COM-5IEDT-ATOD-CH1**	**-**	**TEMP1-TIME**	**100**
					40	Test for a range of values contained in a variable	**TEMP1**	**0-255**	**TEMP1-LCD**	**0-255**
					29	Testing Throughput	**COM-1THR-ATOD-CH1**	**-**	**TEMP1-THRU**	**10/Sec**
					57	Testing for change of variable when Temp1 is sensed	**COM-SIEDT-ATOD-CH1**	**LOW-EDGE**	**TEMP1-LCD**	**0-255**
					21	Testing processing 16 bit and 32 Bits for proper conversion	**COM-1COV-TEMP1**	**TEMP 2**	**TEMP1-WORD**	**16**
					68	Testing for response time	**COM-1RES-TEMP2**		**TEMP1-RES-LCD**	**75-100**
		Compare Temp1 with Ref1	**33**	**SW**	82	Test for value at Memory address 1 > Memory address 2	**TEMP1**	35	PUMP2-ON	**1**
							REF1	**32**		
					83	Test for value at Memory address 1 < Memory address 2	**TEMP1**	30	PUMP2-OFF	**0**
							REF1	**32**		
					84	Test for value at Memory address 1 = Memory address 2	**TEMP1**	**35**	**PUMP2-OFF**	**0**
							REF1	**35**		

Test Path Number	Test Path Description	State on the path	State serial	HW / SW State	Test Type Serial Number	Test Type Description	Input Variable / Command	Input Value	Output Variable	Expected output value
Path -13	Compare Temp1 with Ref1 Temperature and set Pump1 ON	**Pump1ON**	**38**	**SW**	1	Testing the hardware independent code simulating Input/output functions that deal with the Hardware devices	**PUMP1-ON**	**1**	**PUMP1-ON**	**TRUE**
							PUMP1-OFF	**0**	**PUMP1-OFF**	**FALSE**
					40	Testing range of values contained in a variable	**PUMP1-ON**	**0-1**	**PUMP1-VOL**	**5/12**
					68	Testing for response time	**COM-1RES-PUMP1**	**-**	**PUMP1-RES**	**100**
		Relay1	**40**	**HW**	**81**	Testing for Faulty devices early in the development life cycle	COM- 7FLT-RELAY1	-	PUMP2-STA	**0/1**
		Pump1	**41**	**HW**	**81**	Testing for Faulty devices early in the development life cycle	COM-7FLT-PUMP1	0-1	REALY-STA	**0/1**
					65	Testing for the timing of accessing of a device through filters	COM-5OEDT-PUMP1	-	PUMP2-TIME	**100**
Path -14	Compare Temp1 with Ref1 Temperature and set Pump1 off	**Sen1**	**8**	**HW**	**81**	Testing for Faulty devices early in the development life cycle	COM-7FLT-SENS1		SENS2-STA	**0/1**
					58	Testing for Timing of the signals	COM-5IEDT-SENS1		SENS2-TIME	**0-50**
					59	Testing for occurrence of signals in a proper sequence	COM-5IEDT3-SENS1-SENS2		SENS1-SENS2-SEQ	**12/21**
		OP1	**10**	**HW**	**81**	Testing for Faulty devices early in the development life cycle	COM-7FLT-OP1		OP2-STA	**0/1**
					60	Testing for validity of the signal	COM-5IEDT2-OP1		OP2-SIGVAL	**0/1**
					59	Testing occurrence of signals in a proper sequence	COM-5IEDT3-OP1-OP2		OP1-OP2-SEQ	**12/21**
		A/D Converter	**12**	**HW**	**81**	Testing for Faulty devices early in the development life cycle	COM-7FLT-ATOD		ATOD-STA	**0/1**
					60	Testing for validity of the signal	COM-5IEDT2-ATOS		ATOD-SIGVAL	**0/1**
		Microprocessor	**13**	**HW**	**81**	Testing for Faulty devices early in the development life cycle	COM-7FLT-MP		MP-STA	**0/1**
					70	Testing for changes in data at specified at memory Locations	COM-6CHD	#1001	#1001	**0-255**
						Testing for changes in data at specified at memory Locations	COM-6CHD	#1002	#1002	**0-255**
		Read Temp1	**30**	**SW**	13	Testing for Time delay	**COM-5IEDT-ATOD-CH1**	**-**	**TEMP1-TIME**	**100**
					40	Test for a range of values contained in a variable	**TEMP1**	**0-255**	**TEMP1-LCD**	**0-255**
					29	Testing Throughput	**COM-1THR-ATOD-CH1**	**-**	**TEMP1-THRU**	**10/Sec**
					57	Testing for change of variable when Temp1 is sensed	**COM-SIEDT-ATOD-CH1**	**LOW-EDGE**	**TEMP1-LCD**	**0-255**

Test Path Number	Test Path Description	State on the path	State serial	HW / SW State	Test Type Serial Number	Test Type Description	Input Variable / Command	Input Value	Output Variable	Expected output value
Path -14	Compare Temp1 with Ref1 Temperature and set Pump1 off	**Read Temp1**			21	Testing processing 16 bit and 32 Bits for proper conversion	**COM-1COV-TEMP1**	**TEMP 2**	**TEMP1-WORD**	**16**
					68	Testing for response time	**COM-1RES-TEMP2**		**TEMP1-RES-LCD**	**75-100**
		Compare Temp1 with Ref1	**33**	**SW**	82	Test for value at Memory address 1 > Memory address 2	**TEMP1**	35	PUMP2-ON	**1**
							REF1	**32**		
					83	Test for value at Memory address 1 < Memory address 2	**TEMP1**	30	PUMP2-OFF	**0**
							REF1	**32**		
					84	Test for value at Memory address 1 = Memory address 2	**TEMP1**	**35**	**PUMP2-OFF**	**0**
							REF1	**35**		
Path -14	Compare Temp1 with Ref1 Temperature and set Pump1 off	**Pump1OFF**	**38**	**SW**	1	Testing the hardware independent code simulating Input/output functions that deal with the Hardware devices	**PUMP1-ON**	**1**	**PUMP1-ON**	**TRUE**
							PUMP1-OFF	**0**	**PUMP1-OFF**	**FALSE**
					40	Testing range of values contained in a variable	**PUMP1-ON**	**0-1**	**PUMP1-VOL**	**5/12**
					68	Testing for response time	**COM-1RES-PUMP1**	**-**	**PUMP1-RES**	**100**
		Relay1	**40**	**HW**	**81**	Testing for Faulty devices early in the development life cycle	COM- 7FLT-RELAY1	-	PUMP2-STA	**0/1**
		Pump1	**41**	**HW**	**81**	Testing for Faulty devices early in the development life cycle	COM-7FLT-PUMP1	0-1	REALY-STA	**0/1**
					65	Testing for the timing of accessing of a device through filters	COM-5OEDT-PUMP1	-	PUMP2-TIME	**100**
Path-17	Compare Temp2 with Ref2 Temperature and set Pump2 on	**Sen2**	**9**	**HW**	**81**	Testing for Faulty devices early in the development life cycle	COM-7FLT-SENS2		SENS2-STA	**0/1**
					58	Testing for Timing of the signals	COM-5IEDT-SENS2		SENS2-TIME	**0-50**
					59	Testing for occurrence of signals in a proper sequence	COM-5IEDT3-SENS1-SENS2		SENS1-SENS2-SEQ	**12/21**
		OP2	**11**	**HW**	**81**	Testing for Faulty devices early in the development life cycle	COM-7FLT-OP2		OP2-STA	**0/1**
					60	Testing for validity of the signal	COM-5IEDT2-OP2		OP2-SIGVAL	**0/1**
					59	Testing occurrence of signals in a proper sequence	COM-5IEDT3-OP1-OP2		OP1-OP2-SEQ	**12/21**
		A/D Converter	**12**	**HW**	**81**	Testing for Faulty devices early in the development life cycle	COM-7FLT-ATOD		ATOD-STA	**0/1**
					60	Testing for validity of the signal	COM-5IEDT2-ATOS		ATOD-SIGVAL	**0/1**
		Microprocessor	**13**	**HW**	**81**	Testing for Faulty devices early in the development life cycle	COM-7FLT-MP		MP-STA	**0/1**
					70	Testing for changes in data at specified at memory Locations	COM-6CHD	#1001	#1001	**0-255**
							COM-6CHD	#1002	#1002	**0-255**

Test Path Number	Test Path Description	State on the path	State serial	HW / SW State	Test Type Serial Number	Test Type Description	Input Variable / Command	Input Value	Output Variable	Expected output value
Path-17	Compare Temp2 with Ref2 Temperature and set Pump2 on	**Temp2 Read**	**34**	**SW**	13	Testing for Time delay	**COM-5IEDT-ATOD-CH2**	**-**	**TEMP2-TIME**	**100**
					40	Test for a range of values contained in a variable	**TEMP2**	**0-255**	**TEMP2-LCD**	**0-255**
					29	Testing Throughput	**COM-1THR-ATOD-CH2**	**-**	**TEMP2-THRU**	**10/Sec**
					57	Testing for change of variable when Temp1 is sensed	**COM-SIEDT-ATOD-CH1**	**LOW-EDGE**	**TEMP1-LCD**	**0-255**
Path-17	Compare Temp2 with Ref2 Temperature and set Pump2 on	**Temp2 Read**	**34**	**SW**	21	Testing processing 16 bit and 32 Bits for proper conversion	**COM-1COV-TEMP2**	**TEMP 2**	**TEMP2-WORD**	**16**
					68	Testing for response time	**COM-1RES-TEMP2**		**TEMP2-RES-LCD**	**75-100**
		Compare Temp 2 with Ref2	**37**	**SW**	82	Test for value at Memory address 1 > Memory address 2	TEMP2	35	PUMP2-ON	**1**
							REF2	**33**		
					83	Test for value at Memory address 1 < Memory address 2	TEMP2	33	PUMP2-OFF	**0**
							REF2	**35**		
					84	Test for value at Memory address 1 = Memory address 2	TEMP2	33	**PUMP2-OFF**	**0**
							REF2	**33**		
		Pump2 On	**42**	**SW**	1	Testing the hardware independent code simulating Input/output functions that deal with the Hardware devices	**PUMP2-ON**	**1**	**PUMP2-ON**	**TRUE**
							PUMP2-OFF	**0**	**PUMP2-OFF**	**FALSE**
					40	Testing range of values contained in a variable	**PUMP2-ON**	**0-1**	**PUMP2-VOL**	**5/12**
					68	Testing for response time	**COM-1RES-PUMP2**	**-**	**PUMP2-RES**	**100**
		Relay2	**44**	**HW**	**81**	Testing for Faulty devices early in the development life cycle	COM- 7FLT-RELAY2	-	PUMP2-STA	**0/1**
		Pump2	**45**	**HW**	**81**	Testing for Faulty devices early in the development life cycle	COM-7FLT-PUMP2	0-1	REALY-STA	**0/1**
					65	Testing for the timing of accessing of a device through filters	COM-5OEDT-PUMP2	-	PUMP2-TIME	**100**
Path-18	Compare Temp2 with Ref2 Temperature and set Pump2 off	**Sen2**	**9**	**HW**	**81**	Testing for Faulty devices early in the development life cycle	COM-7FLT-SENS2		SENS2-STA	**0/1**
					58	Testing for Timing of the signals	COM-5IEDT-SENS2		SENS2-TIME	**0-50**
					59	Testing for occurrence of signals in a proper sequence	COM-5IEDT3-SENS1-SENS2		SENS1-SENS2-SEQ	**12/21**
		OP2	**11**	**HW**	**81**	Testing for Faulty devices early in the development life cycle	COM-7FLT-OP2		OP2-STA	**0/1**
					60	Testing for validity of the signal	COM-5IEDT2-OP2		OP2-SIGVAL	**0/1**

Test Path Number	Test Path Description	State on the path	State serial	HW / SW State	Test Type Serial Number	Test Type Description	Input Variable / Command	Input Value	Output Variable	Expected output value
Path-18	Compare Temp2 with Ref2 Temperature and set Pump2 off				**59**	Testing occurrence of signals in a proper sequence	COM-5IEDT3-OP1-OP2		OP1-OP2-SEQ	**12/21**
		A/D Converter	**12**	**HW**	**81**	Testing for Faulty devices early in the development life cycle	COM-7FLT-ATOD		ATOD-STA	**0/1**
					60	Testing for validity of the signal	COM-5IEDT2-ATOS		ATOD-SIGVAL	**0/1**
		Microprocessor	**13**	**HW**	**81**	Testing for Faulty devices early in the development life cycle	COM-7FLT-MP		MP-STA	**0/1**
					70	Testing for changes in data at specified at memory Locations	COM-6CHD	#1001	#1001	**0-255**
							COM-6CHD	#1002	#1002	**0-255**
Path-18	Compare Temp2 with Ref2 Temperature and set Pump2 off	**Temp2 Read**	**34**	**SW**	13	Testing for Time delay	**COM-5IEDT-ATOD-CH2**	**-**	**TEMP2-TIME**	**100**
					40	Test for a range of values contained in a variable	**TEMP2**	**0-255**	**TEMP2-LCD**	**0-255**
					29	Testing Throughput	**COM-1THR-ATOD-CH2**	**-**	**TEMP2-THRU**	**10/Sec**
					57	Testing for change of variable when Temp1 is sensed	**COM-SIEDT-ATOD-CH1**	**LOW-EDGE**	**TEMP1-LCD**	**0-255**
					21	Testing processing 16 bit and 32 Bits for proper conversion	**COM-1COV-TEMP2**	**TEMP 2**	**TEMP2-WORD**	**16**
					68	Testing for response time	**COM-1RES-TEMP2**		**TEMP2-RES-LCD**	**75-100**
		Compare Temp 2 with Ref2	**37**	**SW**	82	Test for value at Memory address 1 > Memory address 2	TEMP2	35	PUMP2-ON	**1**
							REF2	**33**		
					83	Test for value at Memory address 1 < Memory address 2	TEMP2	33	PUMP2-OFF	**0**
							REF2	**35**		
					84	Test for value at Memory address 1 = Memory address 2	TEMP2	33	**PUMP2-OFF**	**0**
							REF2	**33**		
		Pump2 ON	**42**	**SW**	1	Testing the hardware independent code simulating Input/output functions that deal with the Hardware devices	**PUMP2-ON**	**1**	**PUMP2-ON**	**1**
							PUMP2-OFF	**0**	**PUMP2-OFF**	**0**
					40	Testing range of values contained in a variable	**PUMP2-ON**	**0-1**	**PUMP2-VOL**	**5/12**
					68	Testing for response time	**COM-1RES-PUMP2**	**-**	**PUMP2-RES**	**100**
		Relay2	**44**	**HW**	**81**	Testing for Faulty devices early in the development life cycle	COM- 7FLT-RELAY2	-	PUMP2-STA	**0/1**
		Pump2	**45**	**HW**	**81**	Testing for Faulty devices early in the development life cycle	COM-7FLT-PUMP2	0-1	REALY-STA	**0/1**
					65	Testing for the timing of accessing of a device through filters	COM-5OEDT-PUMP2	-	PUMP2-TIME	**100**

Test Path Number	Test Path Description	State on the path	State serial	HW / SW State	Test Type Serial Number	Test Type Description	Input Variable / Command	Input Value	Output Variable	Expected output value
Path-20	Compare Temp1 and Temp2 and set buzzer On	**Sen2**	**9**	**HW**	**81**	Testing for Faulty devices early in the development life cycle	COM-7FLT-SENS2		SENS2-STA	**0/1**
					58	Testing for Timing of the signals	COM-5IEDT-SENS2		SENS2-TIME	**0-50**
					59	Testing for occurrence of signals in a proper sequence	COM-5IEDT3-SENS1-SENS2		SENS1-SENS2-SEQ	**12/21**
		OP2	**11**	**HW**	**81**	Testing for Faulty devices early in the development life cycle	COM-7FLT-OP2		OP2-STA	**0/1**
					60	Testing for validity of the signal	COM-5IEDT2-OP2		OP2-SIGVAL	**0/1**
					59	Testing occurrence of signals in a proper sequence	COM-5IEDT3-OP1-OP2		OP1-OP2-SEQ	**12/21**
		A/D Converter	**12**	**HW**	**81**	Testing for Faulty devices early in the development life cycle	COM-7FLT-ATOD		ATOD-STA	**0/1**
					60	Testing for validity of the signal	COM-5IEDT2-ATOS		ATOD-SIGVAL	**0/1**
		Microprocessor	**13**	**HW**	**81**	Testing for Faulty devices early in the development life cycle	COM-7FLT-MP		MP-STA	**0/1**
					70	Testing for changes in data at specified at memory Locations	COM-6CHD	#1001	#1001	**0-255**
					70	Testing for changes in data at specified at memory Locations	COM-6CHD	#1002	#1002	**0-255**
		Temp-2 Read	**34**	**SW**	13	Testing for Time delay	**COM-5IEDT-ATOD-CH2**	**-**	**TEMP2-TIME**	**100**
					40	Test for a range of values contained in a variable	**TEMP2**	**0-255**	**TEMP2-LCD**	**0-255**
					29	Testing Throughput	**COM-1THR-ATOD-CH2**	**-**	**TEMP2-THRU**	**10/Sec**
					57	Testing for change of variable when Temp1 is sensed	**COM-SIEDT-ATOD-CH1**	**LOW-EDGE**	**TEMP1-LCD**	**0-255**
					21	Testing processing 16 bit and 32 Bits for proper conversion	**COM-1COV-TEMP2**	**TEMP 2**	**TEMP2-WORD**	**16**
					68	Testing for response time	**COM-1RES-TEMP2**		**TEMP2-RES-LCD**	**75-100**

Test Path Number	Test Path Description	State on the path	State serial	HW / SW State	Test Type Serial Number	Test Type Description	Input Variable / Command	Input Value	Output Variable	Expected output value
Path-20	Compare Temp1 and Temp2 and set buzzer On	**Compare Temp1 with Temp 2**	**46**	**SW**	13	Testing for Time delay	**COM-5IEDT-ATOD-CH2**	**-**	**TEMP2-TIME**	**100**
					34	Testing for shared data problem	**TEMP1**	**23**	**SHARED-TEMP1-TEMP2-STA**	**0/1**
							TEMP2	**23**		
					82	Test for ABS (Memory address 1 - Memory address 2) > 2	**TEMP1**	**33**	**BUZZER-ON**	**1**
							TEMP2	**36**		
					83	Test for ABS (Memory address 1 - Memory address 2) < 2	**TEMP1**	**33**	**BUZZER-OFF**	**0**
							TEMP2	**34**		
					84	Test for value at Memory address 1 = Memory address 2	**TEMP1**	**33**	**BUZZER-OFF**	**0**
							TEMP2	**33**		
		Buzzer ON	**48**	**SW**	1	Testing the hardware independent code simulating Input/output functions that deal with the Hardware devices	**TEMP1**	**33**	**BUZZER-ON**	**0/1**
							TEMP2	**36**		
					40	Testing range of values contained in a variable	**TEMP1**	**180**	**0-255**	**180**
							TEMP2	**200**	**0-255**	**200**
					68	Testing for response time	**COM-1RES-TEMP2-TEMP1**			**3-5**
		Buzzer	**50**	**HW**	**81**	Testing for Faulty devices early in the development life cycle	COM-7FLT-BUZZER		BUZZER -STA	**1**
Path-21	Compare Temp1 and Temp2 and set buzzer Off	**Sen2**	**9**	**HW**	**81**	Testing for Faulty devices early in the development life cycle	COM-7FLT-SENS2		SENS2-STA	**0/1**
					58	Testing for Timing of the signals	COM-5IEDT-SENS2		SENS2-TIME	**0-50**
					59	Testing for occurrence of signals in a proper sequence	COM-5IEDT3-SENS1-SENS2		SENS1-SENS2-SEQ	**12/21**
		OP2	**11**	**HW**	**81**	Testing for Faulty devices early in the development life cycle	COM-7FLT-OP2		OP2-STA	**0/1**
					60	Testing for validity of the signal	COM-5IEDT2-OP2		OP2-SIGVAL	**0/1**
					59	Testing occurrence of signals in a proper sequence	COM-5IEDT3-OP1-OP2		OP1-OP2-SEQ	**12/21**
		A/D Converter	**12**	**HW**	**81**	Testing for Faulty devices early in the development life cycle	COM-7FLT-ATOD		ATOD-STA	**0/1**
					60	Testing for validity of the signal	COM-5IEDT2-ATOS		ATOD-SIGVAL	**0/1**
		Micro Processor	**13**	**HW**	**81**	Testing for Faulty devices early in the development life cycle	COM-7FLT-MP		MP-STA	**0/1**
					70	Testing for changes in data at specified at memory Locations	COM-6CHD	#1001	#1001	**0-255**
					70	Testing for changes in data at specified at memory Locations	COM-6CHD	#1002	#1002	**0-255**

Test Path Number	Test Path Description	State on the path	State serial	HW / SW State	Test Type Serial Number	Test Type Description	Input Variable / Command	Input Value	Output Variable	Expected output value
Path-21	Compare Temp1 and Temp2 and set buzzer Off	**Temp2 Read**	**34**	**SW**	13	Testing for Time delay	**COM-5IEDT-ATOD-CH2**	**-**	**TEMP2-TIME**	**100**
					40	Test for a range of values contained in a variable	**TEMP2**	**0-255**	**TEMP2-LCD**	**0-255**
					29	Testing Throughput	**COM-1THR-ATOD-CH2**	**-**	**TEMP2-THRU**	**10/Sec**
					57	Testing for change of variable when Temp1 is sensed	**COM-SIEDT-ATOD-CH1**	**LOW-EDGE**	**TEMP1-LCD**	**0-255**
					21	Testing processing 16 bit and 32 Bits for proper conversion	**COM-1COV-TEMP2**	**TEMP2**	**TEMP2-WORD**	**16**
					68	Testing for response time	**COM-1RES-TEMP2**		**TEMP2-RES-LCD**	**75-100**
		Compare Temp1 with Temp 2	**46**	**SW**	13	Testing for Time delay	**COM-5IEDT-ATOD-CH2**	**-**	**TEMP2-TIME**	**100**
					34	Testing for shared data problem	**TEMP1**	**23**	**SHARED-TEMP1-TEMP2-STA**	**0/1**
							TEMP2	**23**		
					82	Test for ABS (Memory address 1 - Memory address 2) > 2	**TEMP1**	**33**	**BUZZER-ON**	**1**
							TEMP2	**36**		
					83	Test for ABS (Memory address 1 - Memory address 2) < 2	**TEMP1**	**33**	**BUZZER-OFF**	**0**
							TEMP2	**34**		
					84	Test for value at Memory address 1 = Memory address 2	**TEMP1**	**33**	**BUZZER-OFF**	**0**
							TEMP2	**33**		
		Buzzer OFF	**48**	**SW**	1	Testing the hardware independent code simulating Input/output functions that deal with the Hardware devices	**TEMP1**	**33**	**BUZZER-ON**	**0/1**
							TEMP2	**36**		
					40	Testing range of values contained in a variable	**TEMP1**	**180**	**0-255**	**180**
							TEMP2	**200**	**0-255**	**200**
					68	Testing for response time	**COM-1RES-TEMP2-TEMP1**			**3-5**
		Buzzer	**50**	**HW**	**81**	Testing for Faulty devices early in the development life cycle	**COM-7FLT-BUZZER**		BUZZER -STA	**0**