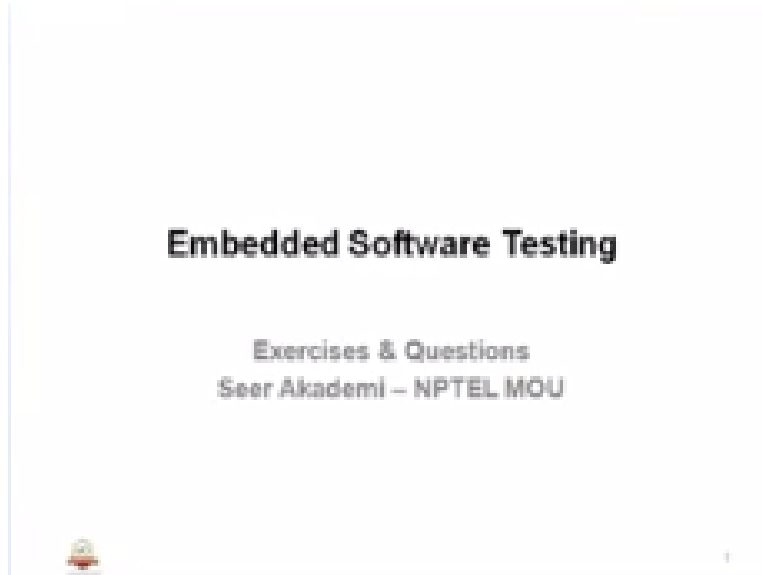


Embedded Software Testing

Welcome you to the next session embedded software testing.

(Refer Slide Time: 00:11)



Is the exercises and questions, and practical sessions of serious of testing. Here we see mode practical aspects of embedded software testing incomes of photo exercises walkthrough videos at tools demonstrations and followed by questions and give answers and I impose some questions for you can have a look understand and you can write back to me. And having understood the email software testing I units.

(Refer Slide Time: 00:51)

Embedded Software Testing

Unit 1 to 5: Overview

Lecture 1

Geer Akademi – NPTEL MOU

we know that a different complexity and different types of embedded systems can be tested in different approaches like a black box, white box in terms of test approach and in terms of test levels we know that edit testing integration system and accepting system no part of the testing. so we studied about all the different units like fundamentals of testing then testing methods testing tools and offline testing static and all of these coatings for code reviews, then we studied about integrating the software and testing the integration aspects.

Finally we studied about the test management in terms of how the entire process of testing can be vanished, the defects or will be artifacts on that test. So, related to all these minutes all these five units, so you will have at least one or two examples demonstration, lab sessions, questions and answers pertaining to these units. And accordingly we will progress on and their software testing the modules.

(Refer Slide Time: 02:26)

Exercise 1

- Identify Test cases and scenarios from a set of requirements with grouping, test procedures. Example to consider for this is the SRS for EUI (attached below).



- Fill the Test review checklist filling for high-level testing. (with the below sample)



The first one being the effect size it is on test case design the scenarios and testing the embedded software. so we know that the four days of the testing is a part of the life cycle and the life cycle contains requirements, design, code and test. And in order to test we definitely need to have a requirement document. Here is an example of SRS, so the first exercise is identifying test cases and the scenarios from a set of requirements with grouping, test procedures. Example to consider for this SRS is EUI. So there is an example waited for all these members of the testing it is called EUI. EUI is nothing but the emitter instrument an embedded unit instrument.

It is a generic make a quilter of instrument having certain features and functional goods and it interfaces externally, we certain messages I just general unit. So we will try to understand the exercise and try to lay out answers for this exercise. So our aim is to create test cases and scenarios from the set of requirements. We will try to study the requirements. So what is the requirement?

(Refer Slide Time: 04:26)

Requirements Specification for Embedded Unit Instrument

SRS_1 The Embedded Unit Instrument (EUI) software operates in the operational modes

- INIT
- OPERATIONAL
- MAINTENANCE
- DOWNLOAD
- FAIL

So this document is about requirement specification for embedded with the instrument. So this particular unit, having about says 15 requirements and there are a group of requirements they are categorized into various features or functions so the first quiz group is about modes of operation.

(Refer Slide Time: 04:52)

Figure 1 Modes of operation

Modes of Operation

SRS_2 The EUI shall set/leave to Fail mode in the following cases (CR logic)

- When Mode is in Maintenance mode, if Received Message is equal to 'BICRUP'
- When Mode is in Maintenance mode, if Received Message is equal to 'SICRUP'
- When Mode is in Init mode, if Status is in '_Error state'
- When Mode is in Operational mode, if Maintenance mode status is in not_Enabled
- When Mode is in Operational mode, if Received Message is not equal to 'CURP'
- When Mode is in Maintenance mode, if Transmit Message is not equal to 'CURP'
- When Mode is in Init mode, if Max_Enables alive less than 1 sec
- When mode is in Operational mode, if Enable state alive less than 1 sec

Next set of requirements is on performance requirements.

(Refer Slide Time: 05:00)

	<p>Transmit Message to 'CRUP'</p> <ul style="list-style-type: none"> When mode is in Maintenance mode, if Received Message is equal to '00CRUP' When mode is in Maintenance mode, if Transmit Message is equal to 'CRUP' When mode is in Idle mode, if Status is in <i>rs_Error</i> state When mode is in Idle mode, if <i>rs_Error</i> state alive at least 1 sec
SRS_8	<p>When Mode is in Maintenance mode, if Received Message is equal to '00CRUP', the EU shall set Mode to Download mode, wait for 1 second to enter into perform Download management</p> <ul style="list-style-type: none"> When mode is in Maintenance mode, if Received Message is equal to '00CRUP' When mode is in Maintenance mode, if wait 1 second
	<p>Performance requirements:</p>
SRS_7	<p>The entire process of booting up till Operational mode shall take not more than 1 sec.</p>

Then next set of requirements you have of communication the next set is about to watch dog out.
(Refer Slide Time: 05:07)

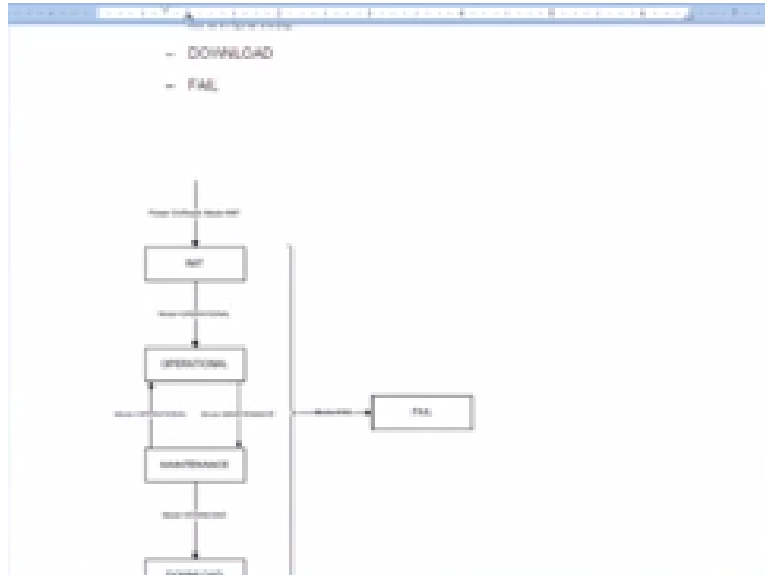
SRS_8	The processor workload for the worst-case shall not exceed 75%.
SRS_9	Flash memory shall have a reserve of at least 50%.
SRS_10	RAM memory shall have a reserve of at least 50%.
SRS_11	EEPROM memory shall have a reserve of at least 50%.
	Communication requirements:
SRS_12	The EU shall communicate through CAN messages with outside interface.
SRS_13	The EU shall communicate its alive status at least a second.
	Watchdog requirements:
SRS_14	<p>The EU shall configure the internal watchdog to generate a reset to the micro-controller with threshold duration of 500ms</p> <ul style="list-style-type: none"> When micro-controller is reset, if watchdog generate pulse duration of greater than 500ms
SRS_15	<p>The EU shall pulse the internal watchdog at least once every 500ms or less</p> <ul style="list-style-type: none"> When micro-controller operations are running continuously, if watchdog generate pulse at least once every 500ms or less

So likewise we have about one two three four set of requirements this is general law requirement whatever in the beginning. so we will try to do the testing of this, so as I said in one of the session to have a embedded testing done appropriately to write the test cases approach and do the design and draw, create these scenarios and do the execution the first and foremost thing that we need to have is the system. System is what? The embedded software system that system we need to have a clear understanding.

So how we can have an understanding based on the system understanding document our system spec. Here in this case we directly have the software requirements and in general I try to tell

what is some is about this system is in an embedded unique instrument so it works in different modes.

(Refer Slide Time: 06:20)



And it performs to some requirement like it is supposed to have certain messages for certain actions for certain messages.

(Refer Slide Time: 06:26)

connected)

Figure 1 Modes of operation

Modes of Operation:

SRS_2 The EUI shall setMode to Fail mode in the following cases (OR logic)

- When Mode is in Maintenance mode, if Received Message is equal to "11CRUF"
- When Mode is in Maintenance mode, if Received Message is equal to "02CRUF"
- When Mode is in Init mode, if Status is in „Error“ state
- When Mode is in Operational mode, if Maintenance mode status is in not „Enable“
- When Mode is in Operational mode, if Received Message is not equal to „CURF“
- When Mode is in Maintenance mode, if Transmit Message is not equal to „CURF“
- When Mode is in Init mode, if (SIS_Empress) alive less than 1 sec
- When mode is in Operational mode, if Enable state alive less than 1 sec
- When mode is in Download mode, if wait more than 1 sec

SRS_4 When Mode is in Standby (Status is in the Standby) the EUI shall switch to:

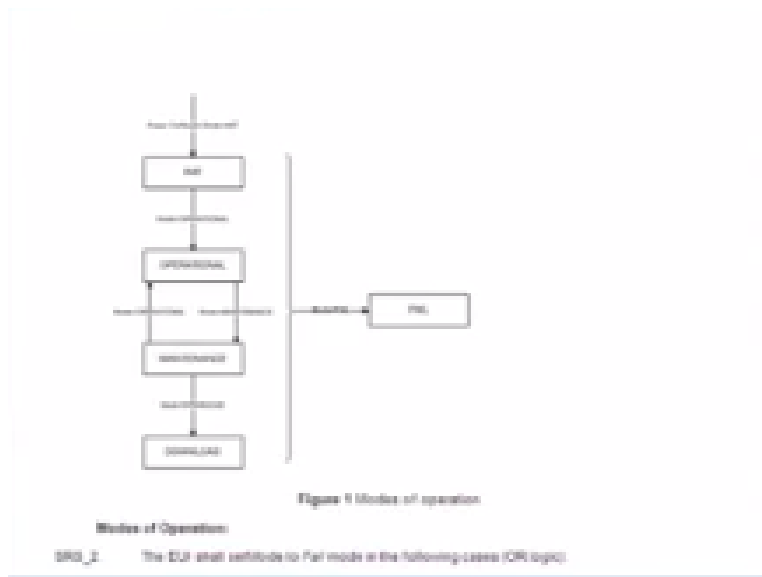
And it is supposed to work at certain speed and it should have certain limitation in the memory and how it can interface with the ex mode is explained in the communication requirement.

(Refer Slide Time: 06:43)

SRS_8	The processor workload for the worst-case shall not exceed 75%.
SRS_9	Flash-memory shall have a reserve of at least 50%.
SRS_10	RAM-memory shall have a reserve of at least 50%.
SRS_11	EEPROM-memory shall have a reserve of at least 50%.
Communication requirements:	
SRS_12	The EU shall communicate through CAN messages with outside interface.
SRS_13	The EU shall communicate its alive status at least a second.
Watchdog requirements:	
SRS_14	The EU shall configure the internal watchdog to generate a reset to the micro-controller with threshold duration of 500ms. <ul style="list-style-type: none"> When micro-controller is reset, if watchdog generate pulse duration of greater than 500ms.
SRS_15	The EU shall pulse the internal watchdog at least once every 500ms or less. <ul style="list-style-type: none"> When micro-controller operations are running continuously, if watchdog generate pulse at least once every 500ms or less.

And there is a additional unit we're it called washed off. So these are the top level elements of the embedded software of the system and how it works or repeated in the image softer world or in the embedded system is that it works in five modes those in it, operational, maintenance, download and fail. So you can see this with diagram.

(Refer Slide Time: 07:11)



This embedded system will be working in this one of the modes of operation when it is powered up until it is all down or until it is used for different set of operations. So we have a to start with the init mode then it goes to your operational mode then it can go to maintenance mode or it comes back to operational mode. And now maintenance mode it can further go to download. These are basic operations of basic modes of operation that this is embedded unit to the

instrument will work come on. And you can see the arrow specifically created, so that accordingly the following happen and from operational to init is no way to go back.

Similarly from download to maintenance there is no way to go back. So there should be a requirement how we can go back to the init mode. so those are all called state level. so that sort of a complexity or that level be good let's try to see if we can retail which of session or let us try to keep it simplistic saying that my embed system is having this many operations and this much requirements. For this only I will try to write test case and understand how I can test it that is the aim of this exercise.

This is basically or less, nothing is that at any point of time if during this one two three four modes of operation, the unit can go to a mode called fail mode. That means in the fail mode, during initialization it could fail during operational requirements or operational activity or during maintenance and download operations it can go to a fail mode. So from the fail mode it will not go to any mode with be stuck or it will never be up.

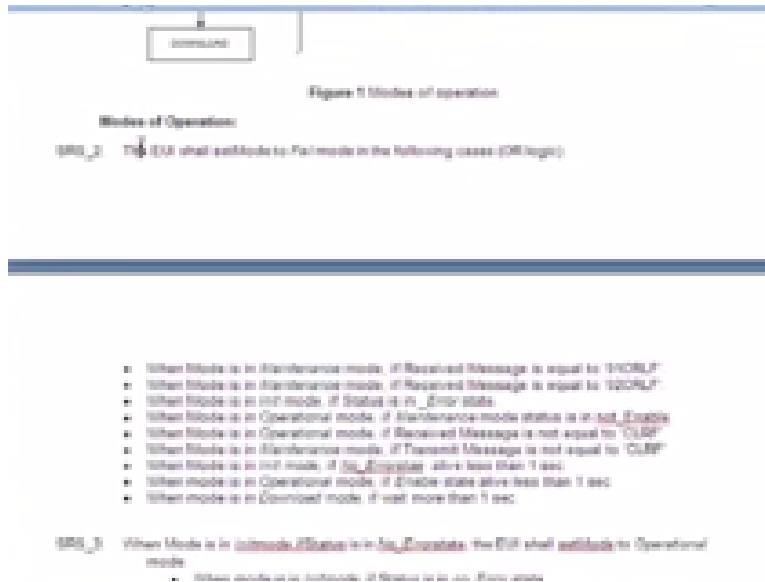
So in order to restart probably user has to restart the system, so that from fail mode it can recover from the init operational. So basically any embedded systems you take will have to slow mechanisms two levels of flow mechanism. One is initialization other one is operation. Once it is initialized it is good to operate or it is good to run. So in operational mode what are the specific requirements of the embedded system specific resource requirements of the business of the queue or the client based requirements it is supposed to work, if you one so those two basically are the prime elements.

But that is not sufficient in the emergency because it is to take care of many other things like failures for and we should be maintained a level we should be downloadable always. So accordingly there are different operations or sub operations it will be up. So let us try to keep it simple saying that this embedding instrument works in five modes of operation, and those are an in it operation maintenance and download. And at any point of time it can move very small. Have you said that so we have got a understanding on what this instrument does. So the first SRS or the reverse requirement so basically requirements are identical is the tag called SRS software requirement specification. So first one is talking about in general five modes of operations.

The next requirement will try to understand it is basically under the category modes of operation, so what are the modes that this instrument can go? And where and all it can go to switch and backward? So you can see for each of that we have a separate requirement like weather is mode what will happen operation mode what will happen and in the initialization mode what will happen and where it can go to failure mode.

Well it's very important to understand a requirement first.

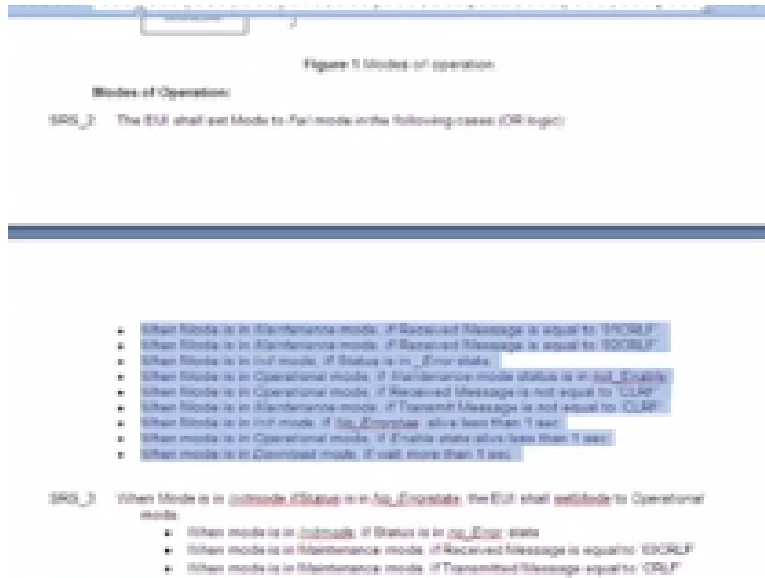
(Refer Slide Time: 11:58)



so let us try to understand for an extent what it is of course you can go through this offline and try it on this time and can elaborate mode you are understanding when we are done with the test cases probably it will be a generic one or the basic ones, which is enough to launch further who are mode test cases and we have studied in the one of the classes over partition equivalence and analysis type of test cases all this. so that can be elaborated further based on our expertise and when we dig mode into no requirements but generically we try to draw test cases or these set of requirements to start with.

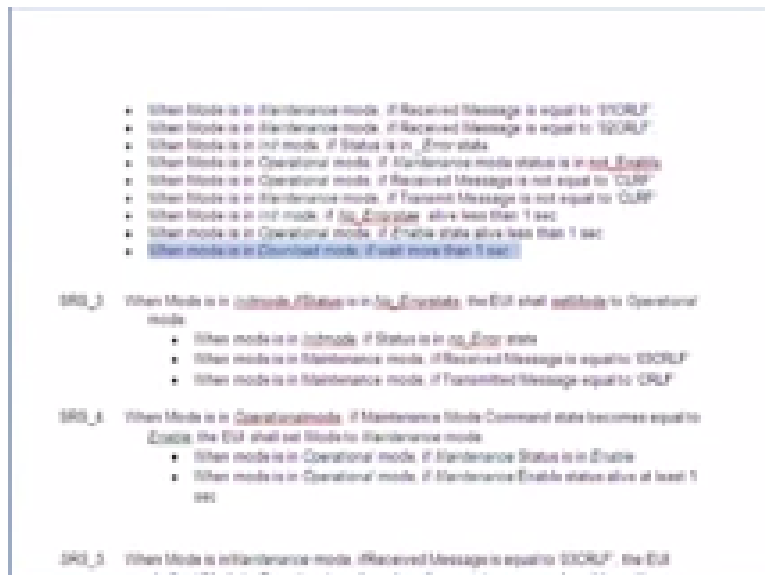
So SRS to sales about the EUI set mode to failure mode in the following cases or logic. So what it means is the init will set the border so it different modes of operations are there right. So it will send them mode to failure mode under the following conditions.

(Refer Slide Time: 13:14)



So this condition can occur in any order or at any priority. That is why it is clearly told as R logic R logic means.

(Refer Slide Time: 13:22)



This alone can happen or this send this both can happen or multiple public's can happen or 1st 3rd 4th 5th anything can happen.

(Refer Slide Time: 13:31)

- When Mode is in Maintenance mode, if Received Message is equal to 'EORLP'
- When Mode is in Maintenance mode, if Received Message is equal to 'EORLP'
- When Mode is in Init mode, if Status is in 'Error State'
- When Mode is in Operational mode, if Maintenance mode status is in 'not_Enabled'
- When Mode is in Operational mode, if Received Message is not equal to 'CLR'
- When Mode is in Maintenance mode, if Transmitted Message is not equal to 'CLR'
- When Mode is in Init mode, if (No_Corrupt) alive less than 1 sec
- When Mode is in Operational mode, if Enable state alive less than 1 sec
- When Mode is in Download mode, if wait more than 1 sec

SRS_2 When Mode is in (Initmode/Status) is in (No_Errorstate), the ECU shall set Mode to Operational mode

- When mode is in (Initmode) if Status is in (No_Error) state
- When mode is in Maintenance mode, if Received Message is equal to 'EORLP'
- When mode is in Maintenance mode, if Transmitted Message equal to 'CLR'

SRS_3 When Mode is in (Operationalmode), if Maintenance Mode Command state becomes equal to (Error), the ECU shall set Mode to Maintenance mode

- When mode is in Operational mode, if Maintenance Status is in (Error)
- When mode is in Operational mode, if Maintenance Enable state alive of least 1 sec

SRS_4 When Mode is in Maintenance mode, if Received Message is equal to 'EORLP', the ECU shall set Mode to (Failure mode)

Any of this happens it will enter into failure mode. So we need to draw a test case for this. What are they of some global requirements for this, so these are all about one two three four five six seven eight nine types of cases are there.

(Refer Slide Time: 13:54)

Figure 3 Modes of operation

Modes of Operation:

SRS_1 The ECU shall set Mode to Fail mode under following cases (OR logic)

- When Mode is in Maintenance mode, if Received Message is equal to 'EORLP'
- When Mode is in Maintenance mode, if Received Message is equal to 'EORLP'
- When Mode is in Init mode, if Status is in 'Error state'
- When Mode is in Operational mode, if Maintenance mode status is in 'not_Enabled'
- When Mode is in Operational mode, if Received Message is not equal to 'CLR'
- When Mode is in Maintenance mode, if Transmitted Message is not equal to 'CLR'
- When Mode is in Init mode, if (No_Corrupt) alive less than 1 sec
- When Mode is in Operational mode, if Enable state alive less than 1 sec
- When Mode is in Download mode, if wait more than 1 sec

SRS_2 When Mode is in (Initmode/Status) is in (No_Errorstate), the ECU shall set Mode to Operational mode

- When mode is in (Initmode) if Status is in (No_Error) state
- When mode is in Maintenance mode, if Received Message is equal to 'EORLP'

Each is will have its own requirement line. When they embed instrument is in maintenance mode it will go to failure mode. When embedded instrument is in init mode it will go to fail one. When it is in operational mode it can go to failure mode. When it is download mode of course it will go to failure mode. So we don't have this diagram that it can go to failure mode init operational maintenance and download, so four levels of entering into a failure mode is there. is That is what

has been explained here and further it could be different events vertical event like in maintenance we have one two three events are there.

(Refer Slide Time: 14:46)

- When Mode is in Maintenance mode, if Received Message is equal to '01CRLF'
- When Mode is in Maintenance mode, if Received Message is equal to '02CRLF'
- When Mode is in Init mode, if Status is in „Error state
- When Mode is in Operational mode, if Maintenance mode status is in not_Enabled
- When Mode is in Operational mode, if Received Message is not equal to 'CRLF'
- When Mode is in Maintenance mode, if Transmitted Message is not equal to 'CRLF'
- When Mode is in Init mode, if No_Error alive less than 1 sec
- When mode is in Operational mode, if Enable state alive less than 1 sec
- When mode is in Download mode, if wait more than 1 sec

SRS_3 When Mode is in Initmode, if Status is in No_Error state, the EUI shall setMode to Operational mode

- When mode is in Initmode, if Status is in no_Error state
- When mode is in Maintenance mode, if Received Message is equal to '02CRLF'
- When mode is in Maintenance mode, if Transmitted Message equal to 'CRLF'

SRS_4 When Mode is in Operationalmode, if Maintenance Mode Command state becomes equal to Enable, the EUI shall set Mode to Maintenance mode

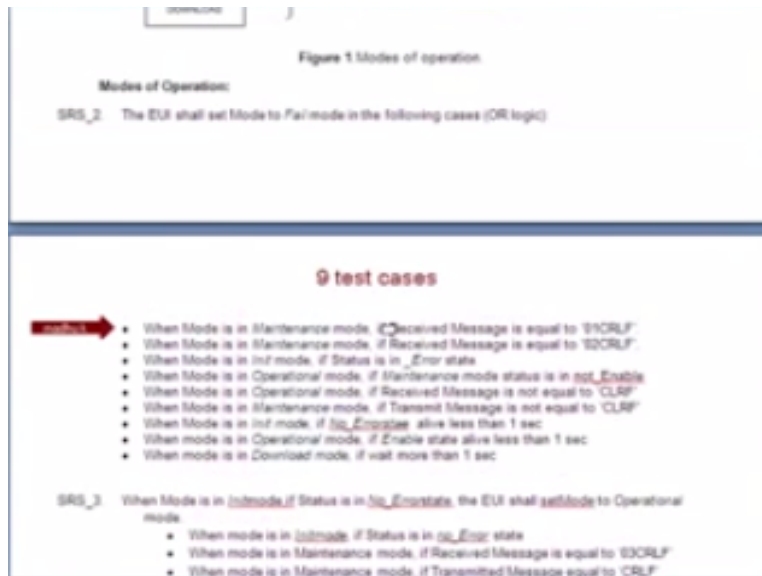
- When mode is in Operational mode, if Maintenance Status is in Enable
- When mode is in Operational mode, if Maintenance Enable status alive at least 1 sec

SRS_5 When Mode is in maintenance mode, if Received Message is equal to '02CRLF', the EUI shall set Mode to Operational mode and confirm maintenance mode exit by setting TransmitMessage to 'CRLF'

Which will be, responsible for entering to failure mode. Similarly we have to pull your events init mode. That can make it the other mode. Similarly we have operational mode as well. The next requirement is about when the body is init mode if the is status is. Let me actually explain any one mode requirement when mode is in maintenance, if there is a message received equal to 01 CPR LM. Don't bother about what is we are you.

Let us try to understand the first one.

(Refer Slide Time: 15:54)



When mode is in maintenance mode if received a message is equal to 01 CR LF. When mode is in maintenance mode if receive 02 CR LF. So when it is in maintenance mode there are two events that can happen 01 CR LF. 02 CR LF. forget about what is CR LF that is a message so when they embed any the instrument is use a message having the content 01 CR LF, it should go to failure mode. Understood similarly we have another case in maintenance mode when the embedded unit instrument in the maintenance mode if the transmitted message is not equal to CRLF.

That means, here we have received the message and if the transmitted message from the embedding instrument has not becomes CR LF. This CRLF, it will go to failure mode. Similarly in the initialization if there is another occurs or visit error condition, it will also go to failure mode. So like this we need to identify test cases, so definitely we have nine test cases right because we have nine sub requirements 1 2 3 4 5 6 7 8 9.s

So 9 different test cases we have each test case we are going to provide a input as what. What is maintenance mode? Then we have to transmit a 01 CRLF externally so that the unit receives your own CRLF and we expect a output as failure mode. so this three elements are required for testing things, like ways we have the next one we should make sure that the embedding to instrument is in maintenance mode and we receive a message as 01 CRLF and we can expect the output as a failure mode.

Similarly we have for everything the test cases are better, and interestingly there is a R logic. R logic means what? Multiples of these events can occur. so we should be able to test with multiple events that is a you get 1, you get 2, you get 3 and in test case we can have all you want 1 2 3 4 and in the next test case you can have all the 1,2,3,4, 9. so combinations of these events is also very important which is nothing but what we extra get about is the MCGST or they complete coverage of this requirement.

So that we have covered this R logic so that R of all this have been tested R of one or two also assisted it does not mean that we should test for everything every or may not be required it is just tell in a R line, that is what do we do with the partition equivalence we will try to test with first R of first two maybe d in between one R conditionally we can test and in the last also we have we can test it. And the last case having all for all this the other important thing that we need to understand is whether it is viable to test it.

Why I am telling is we know that this instrument can go in the mode and that mode it will be there available. so is it possible for us to a minute all the mode one instance it cannot be true, so in that case we need to understand whether we will be able to do are the inputs that can be triggered or not. If it is possible to trigger then we should do this R logic since it is not possible we will not be able to do.

So it's very important to understand the OB system logic and understanding of the entire system. so that it is easy to write so 9k test cases definitely we can do it and R logic definitely we can do for sudden cases and R logic for the entire case may not be possible because we may not be able to emulate, there are how many conditions maintenance is worn in with these one operational mode is one download modes. So 4 modes switched into emulate required time is it possible.

So we should limit our test cases according to the need of the system and those are test case is valid should try to understand whether is it possible to test it that's what is this requirement is about. The next one SRS, when the mode is initialization mode, now we are coming to individual modes basically in terms of its power up and other things. So when word is in init mode instead of this no error state the EUI shall set about to operational mode.

We know that you go to failure mode in the initial, becomes failures or there is an error in the initialization it will go to what failure. so the opposite of this so you may be having a negative performance in this test case only so it is also important I will tell you not to just have the positive values of 01 CRLF. We should be able to scramble lick this one as CR something like CR LF 10. so different type of message we should be able to power up like using that other which is other than what is specified here so that is a negative to test case or other than the entire R operations something like that.

So anyway we may not be able to have an answer what will happen if this is in not in error state if the initialization is successful. so that is what is this requirement talks about. When the mode is in in it mode if the status that means the operations of the insulation having no errors, the instrument shall set the mode to operational mode that means it has moved to operational. Because insulation is successful if it is not moved it would have gone to failure mode. That is what the requirement is about.

So once you have understood this is Unicode is out. So trigger inputs with a failure case trigger input between second suitable two cases are done. And in either case we are missing. let's set understand this one when mode is in init mode, if no error state is alive less than one second it

means the initialization is stuck or mode than a second but there is of failure means error is not there still it will go to a failure that means. What understands? First case is initializing is happening and some failures are called the initialization and that will lead to failure mode and during installation, initialization is complete it is successful it will go to operation.

Now the third case we have ancient policy with no error this is successful but it is not able to go to operational due to some issues or some other thing you can call it as insulation failure but it is not a insulation failure because error set for that is no error. So still it is a failure it will again go to failure why because it is not able to go to operational in certain time, what is the time? One second.

So this is another case the typical embed system requirement this is formally one of the example embedding instrument that I have picked up from my experience and genetically I try to put it, so nothing in this stool for any application or anything we can draw it to a different application for whatever it is one of the example.

(Refer Slide Time: 25:09)



further understanding about this init mode if the status is no error then you go to operational mode conditionally what are those conditions when the mode is in init mode if strategy is no error state and when mode is maintenance mode if received message is equal to 03 CR LF then work what is in maintenance mode if transition with transmitted message is equal to CRLF. When mode is in operational mode the next requirement is maintenance mode command state becomes equal to enable EUI mode to maintenance mode. When mode is in operational mode it maintenance status alive at least one set is all similar requirement but specific to different modes.

So these are the requirements we need to understand and try to understand the entire system this is about operational requirement is the entire operations how different modes of operations are

there any other things what we have missed probably the download mode of operations is not put here.

(Refer Slide Time: 26:26)

SRS_3 When Mode is in *Init mode*, if Status is in *no_Error* state, the EUI shall set Mode to Operational mode.

- When mode is in *Init mode*, if Status is in *no_Error* state
- When mode is in Maintenance mode, if Received Message is equal to '03CRLF'
- When mode is in Maintenance mode, if Transmitted Message equal to 'CRLF'

SRS_4 When Mode is in *Operational mode*, if Maintenance Mode Command state becomes equal to *Enable*, the EUI shall set Mode to Maintenance mode.

- When mode is in Operational mode, if Maintenance Status is in *Enable*
- When mode is in Operational mode, if Maintenance Enable status alive at least 1 sec

SRS_5 When Mode is in Maintenance mode, if Received Message is equal to '03CRLF', the EUI shall set Mode to Operational mode and confirm maintenance mode exit by setting Transmit Message to 'CRLF'.

- When mode is in Maintenance mode, if Received Message is equal to '03CRLF'
- When mode is in Maintenance mode, if Transmit Message is equal to 'CRLF'
- When mode is in *Init mode*, if Status is in *no_Error* state
- When mode is in *Init mode*, if *no_Error* state alive at least 1 sec

SRS_6 When Mode is in Maintenance mode, if Received Message is equal to '04CRLF', the EUI shall set Mode to Download mode, wait for 1 second to enter into perform Download management.

- When mode is in Maintenance mode, if Received Message is equal to '04CRLF'
- When mode is in Maintenance mode, if wait 1 second

Performance requirements:

SRS_7 The entire process of booting up till Operational mode shall take not more than 1 sec.

So under what conditions it will go to download. It is really so that download itself will go from maintenance maybe because it can go firmly the comment there is a requirement here understand this requirement.

(Refer Slide Time: 25:46)

- When mode is in Maintenance mode, if Received Message is equal to '03CRLF'
- When mode is in Maintenance mode, if Transmitted Message equal to 'CRLF'

SRS_4 When Mode is in *Operational mode*, if Maintenance Mode Command state becomes equal to *Enable*, the EUI shall set Mode to Maintenance mode.

- When mode is in Operational mode, if Maintenance Status is in *Enable*
- When mode is in Operational mode, if Maintenance Enable status alive at least 1 sec

SRS_5 When Mode is in Maintenance mode, if Received Message is equal to '03CRLF', the EUI shall set Mode to Operational mode and confirm maintenance mode exit by setting Transmit Message to 'CRLF'.

- When mode is in Maintenance mode, if Received Message is equal to '03CRLF'
- When mode is in Maintenance mode, if Transmit Message is equal to 'CRLF'
- When mode is in *Init mode*, if Status is in *no_Error* state
- When mode is in *Init mode*, if *no_Error* state alive at least 1 sec

SRS_6 When Mode is in Maintenance mode, if Received Message is equal to '04CRLF', the EUI shall set Mode to Download mode, wait for 1 second to enter into perform Download management.

- When mode is in Maintenance mode, if Received Message is equal to '04CRLF'
- When mode is in Maintenance mode, if wait 1 second

Performance requirements:

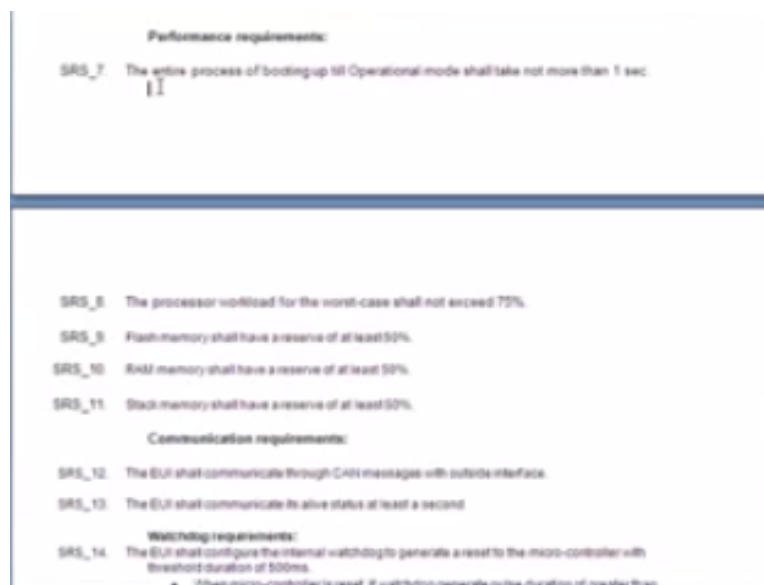
SRS_7 The entire process of booting up till Operational mode shall take not more than 1 sec.

you try to draw few requirements some cases it's here for you when mode is in maintenance mode definitely if received message is equal to 04 CRLF you need to shall be mode work at this

it has accepted this message and it is good to go for download operation. Then condition of the requirement actually, I will then wait for one second to enter into perform download mode that means once it enters into download mode you should wait for one second to enter into download mode of activities. And thus the conditions are when mode is in maintenance mode if received messages are going to CRLF and what it is means more intuitive consider gate. So these two are sub level requirements for this enter a permit.

So this is how the requirement should be understood try to draw test cases. The next set of requirement performance requirement the entire process of booting up till operational mode shall we take not mode once again.

(Refer Slide Time: 28:01)



so you know what is boot, so when the embedded system is powered on it will be booted this it will be powered up all the components all the peripherals are all the elements that are part of the image systems powered up. And with time it requires to make it up and running till then whatever is going to happen it is nothing but booting the process is called booting process. so here the requirement the entire process of booting up till it becomes operational should not take more than one second.

Actually speaking there are words actually which you should be careful shell, should will may be set up. so each one has its own meaning basically I will tend to probably explain one of the mode for it is one of the requirements understanding actually this one point of the embedded software testing, but it is better to have an understanding all these words shell means compulsorily the requirements will be tested, other requirement should be implemented at that level you try to understand.

Come back to performance requirements so these are all performance related of the embedded systems there are about three four or five requirements are there, the first one the seventh one is the entire process of booting up till operation both shall take not more than one second, that means once you power on the system once it becomes operational the duration rate which this is becoming operational should happen by less than one second this is a meaning.

So this is bit challenging to understand first second thing is you may come up with lot of questions before you try to start asking how I can test, it's very important so we need to understand what is booting up operation? What is the operational mode? When I can say that it is operational mode? And what are the hooks? Or what is the strategy that I can apply to test this requirement?

So in general embedded systems will have boot up identification, operational identification again these are derived from the requirements. So your first question this should be first and foremost questions should be only these things like how I can identify whether it is booting? How I can identify whether it is operational? If these two are answered your basic component is clear. Or testing for I basic component is clear the strategy for working out.

So probably this requirement at this level it is very generous so probably we can draw sub level requirements I am just trying to put it actually because I not put the sub level or is also called as derived requirement no you want to say. so that can have something like the boot up notification you or the Boot up process starts, when there is R cycle at in number 2d. Similarly there could be operational requirements one of the operational requirement you should be able to identify saying that it is operational.

So probably will take constant mode, here communication requirements we have so this will happen during operational requirement. So what it says is the UHL communicate through canvasses the outside processes, that mean one of the interface requirements should be able to identify our operational requirement moths. So it is clear right now with these two it should on the test.

Again next question is how we can test it so there are certain things like there is a one second that means we need to measure the time it is very clear. So time how you can measure? Time to be measured, how we can measure with the help of measuring instrument. Such as our telescope to measure duration how many hooks you need to, because, one at the start one at the required end basically constant, so one at the booting up identification time other one is at the operational one. So we have a pin number 20 the first from being pinned on deck thing pin 20 we should 1 hook up the second one should be interface hook up.

The interface requirement as we thought yeah can message, can message basically we defined if a they can port. We say hardware interface there you can put the oscilloscope or we can very well see the message analyzers are there, such as canalize can scope. we set up any of these

instrument can be used so there is a how we can measure now it is definitely pin 20 scope starts showing the wave basically and we record the time it is called a start time.

Similarly here we record the message receive time in the can, because both are running at the same system we know that be I'm log or tide is called time log or timestamp of this. So, net duration of this timestamp equal to message received time- start time.

(Refer Slide Time: 35:59)

EST. BOOTUP IDENTIFICATION, OPERATIONAL IDENTIFICATION -- REQUIREMENTS --

DERIVED REQUIREMENTS:
BOOTUP PROCESS STARTS WHEN THERE IS POWER CYCLE AT PIN 20
INTERFACE REQ. -- IDENTIFY OPERATIONAL REQUIREMENTS.

TIME TO MEASURED, -> MEAS. INSTRUMENT SUCH AS OSCILLOSCOPE.

2. START - REQUIRED END.

1. PING - START TIME
2. INTERFACE (CAN MSG) -- CAN PORT -- MESSAGE ANALYZERS (ANALYZERS, CANSCOPE, ...) - MESSAGE RECEIVE TIME

TRACLOG/TIMESTAMP
NET DURATION OF - MESSAGE RECEIVE TIME - START TIME < 1 SEC.

SRS_8. The processor workload for the worst-case shall not exceed 75%.

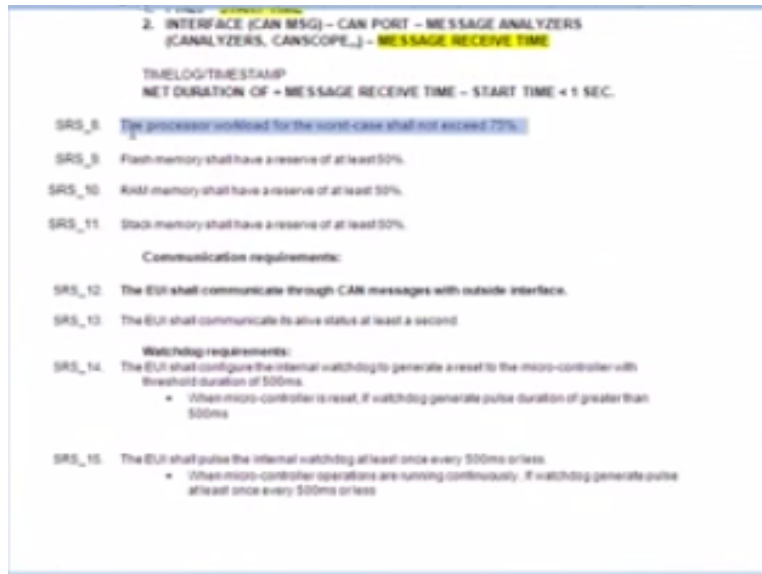
SRS_9. Flash memory shall have a reserve of at least 50%.

SRS_10. RAM memory shall have a reserve of at least 50%.

SRS_11. Stack memory shall have a reserve of at least 50%.

So this is very clear this should not be or this should be less than one second. so our requirement is tested, so like this wish you able to test this requirements this is this will become 0 strategy the strategy got it. So we are in divide it test case, to measure this and make sure that at any point of time when you boot up should not exceed the one second limit, otherwise the requirement has played. So we should be having a question in terms of how we can have identified e test hooks for testing this and what are the measuring diagrams that become hot problems performs requirement.

(Refer Slide Time: 36:53)



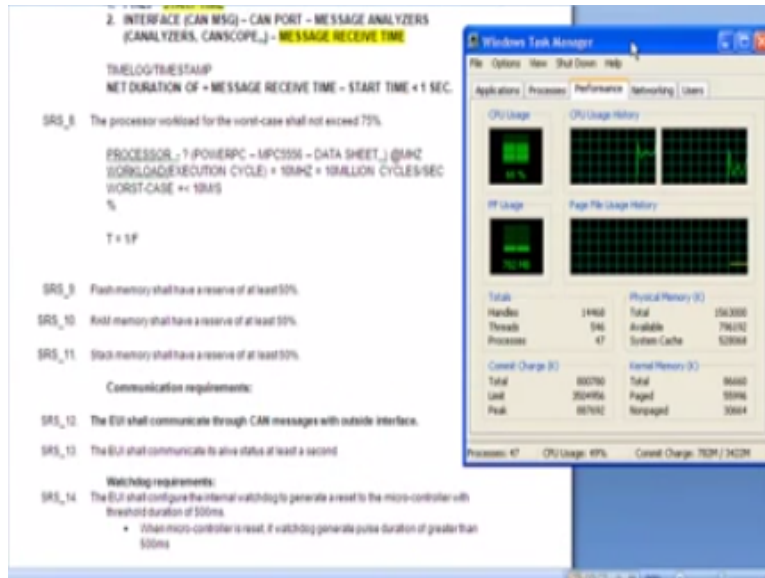
Similarly we have the next one card the processor workload for the worst ph shall not exceed 20ph. so here there are different jargons embedded system jargons one is processor other one is workload it is also called as execution sica. And I know this watch dog case another one is percentage. so we need to first understand what processor is used it could be any micro control if the microcontroller suppose any power pc or pre scale let us say PowerPC example MPC equal five six if it is used, so we need to have a data sheet.

So we know that at what rate it is running that means at certain cycles of speed the processor is capable of executing. once we know that this is the execution suppose if it is the warm herds it is executing the instructions of the execution cycle at one second, because we know that IM equals 1 by F. Now F is a frequency he is a time so one odds it means once again till early 1 negative 1 into 10 to the power 6 like this we are going to have the processor details first then workload.

That mean how many instructions it can execute so if it is 10 Menards the equals 10 million cycles per second it can execute, then worst case. so worst case we have defined that means at max not mode than this that means 10 million per second, that is the worst case or you can say also equal worst case it can go to 10 million per second and generally it will not go that much but if the load, load means the activities are the interactions of operations happening even in microcontroller or the processor is so high that the entire processor is completely loaded completely occupied you can see in windows and all that there is something called a taskbar. Task manager you can see the Lord here networking basically it can say a processor I would say the formula let's go.

So here is on performance requirement for this processor.

(Refer Slide Time: 39:57)



Whatever I have I hope you are able to see this. here you can see the performance the CPU usage see this is a good bar right complete bar, so currently as I speak so there is a audio recorded with a typing world so many things are happening in the windows, and at every instance there is a percentage that is getting changed. you can see 47 percent 54 percent 40 percent every instance it is showing accordingly it is graphing this side, so this they are captured using the instruction cycle so how many instructions are getting used in the CPU is the Intel base CPU so forty-three percent.

Similarly for this embedded instrument meaning we cannot afford to have a mode than seventy percent at least we should have a program or the entire embed systems developed in such a way that will not be loading the application more than 75 percents that's all understanding purpose. And there are other things like page file usage, cache, so many be things part of let us not look into details we come across memory and hostage probably you will come to know what it is. So that is what the basic concept of workload of the processor and percentages 75 percents, this is a formula we use for calculation.

So this for understanding, the question is how you can measure? like we have a task manager remembers for my own decisions we have any task managers we may not have so the requirement should be much clear and mode detail such get to work load how it can be tested that is what is called as testability. Testability of this requirement what is it? I would say no none that means it can be tested at this bring the requirement just like this so we may have to dig further into other requirements to understand what can be done oh and what can be used which requirement I can correlate for that this just ability can be applied.

So that is also important because there are multiple team working on different , performance requirements some one percent is working communication one percent, song while you are one percent sorry to other interaction such that each one is very clear of what they are doing as well

as how others are doing their testing so it's very important to understand the system from that perspective. Usually in general what they are doing is? They use the Map file probably I will go through the map file.

(Refer Slide Time: 43:16)

1. PING - START TIME
2. INTERFACE (CAN MSG) - CAN PORT - MESSAGE ANALYZERS (CANALYZERS, CANSCOPE...) - MESSAGE RECEIVE TIME

TIMEOUT/TIMESTAMP
NET DURATION OF = MESSAGE RECEIVE TIME - START TIME + 1 SEC.

SRS_8 The processor utilization for the worst-case shall not exceed 75%

PROCESSOR : 7 POWERPC - MPC030 - DATA SHEET : @100MHz
WORLDWAD/EXECUTION CYCLE : = 100MHz = 100000000 CYCLES/SEC
WORST-CASE = 100MS
% = 75%

TRUE

REPEATABILITY : 3

MAP FILE = EXECUTION PROCESSOR = DIVIDED INTO VARIOUS TASKS
TASKS + TIME = COLLECTION OF ALL THESE TASKS = TOTAL TIME

SRS_9 Flash memory shall have a reserve of at least 50%.

SRS_10 RAM memory shall have a reserve of at least 50%.

SRS_11 Stack memory shall have a reserve of at least 50%.

Communication requirements.

SRS_12 The ECU shall communicate through CAN messages with outside interface.

SRS_13 The ECU shall communicate its alive status at least a second.

The entire execution, of the processor is divided into various tasks and we know that each tasks where the time is going to take so collection of all this asks equal to the total time. and where we can get the tasks you can get in the map file how we can generate uniform based on the linker linking on the target so connection of all this total time will we will become the total time of the all the task, so again total time that is available based on this formula we are going to arrive at so this is 0 we can write the test case for this particular one.

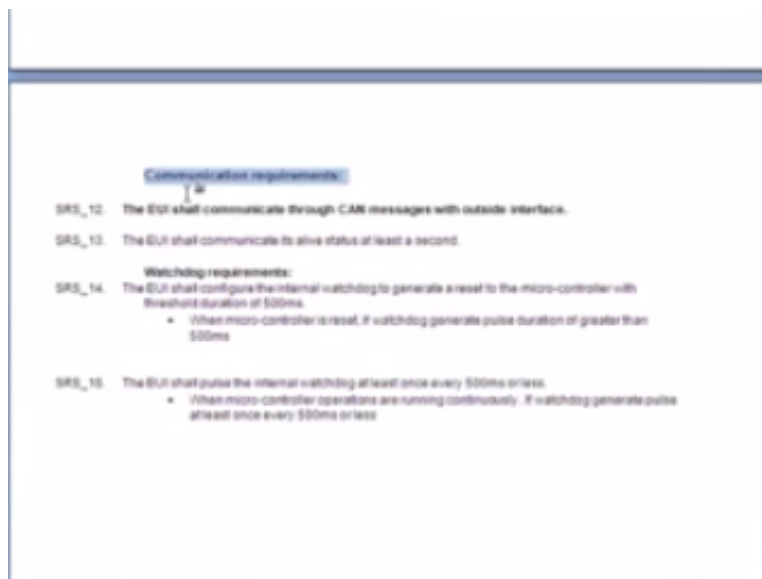
The next one is the memory similar to this we can use the map file for testing the memory like we know that there is a 1 MB of flash memory available and how much memory has been occupied by the embedded instrument, you should not exceed fifty percent that means 500 what is a half of or the 50 percent half of the equal to find it KB. so this is all we are going to measure the space where we can see the size it is based on the map file other typical it is a typical embedded system I am trying to talk, there could be different ways you can see are the HL L file linker file or the definition of the linker it's a suitable size physical size.

So that it is already telling you how much is going to occupy the memory, based on that we know that, what is my proto size instrument? of the embedded system. And how much is being occupied. so if it is more than 500 kb against this requirement has failed, and a bit less than if it is more than 500 kb requirement has failed. Else requirement is passing. This are we are going to write the test case and do an analysis and arrive at the conclusion.

So basically we saw something like the method that we use using analysis these methods are called as analysis especially for performance, requirements are done with analysis. Analysis of memory, analysis of map file, linker and the memory, and the data sheets and all the stuff we part of with them. so other part of a memory is RAM but also should have the more than 50 percent, so this also can be done with the help of the contents that is defined in the linker and the corresponding map file, so that will tell how much memory this embedded in it as what produced.

Another part is the stack you know what is stack so a stack memory should have a reserve will be possible. The next type of requirement is about communication requirement. You know what is communication? It is communication between two elements of the embedded software, it could be subsystems or it could be in system muscles external void.

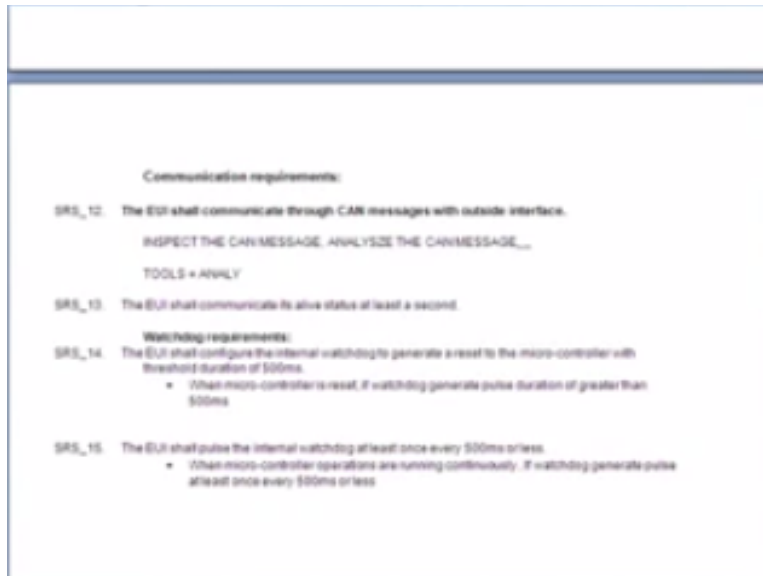
(Refer Slide Time: 47:27)



so in this case what is that we have let's try to understand, the communication a celestial requirement is about the UA shall communicate through can message with outside interface of this, this unit will interact with outside world which can message. So this unit test it this is a very simple requirement oh you can test it. you can inspect the can message whether it is coming out ugly, or analyzed be coming out of the instrument that will prove that this requirement is satisfied that means embedded into instrument is transmitting the message or communicating with external void or solving the requirement.

So we will prove that there could be some tools required to analyze the can messages.

(Refer Slide Time: 48:28)



what are the can messages that is all probably for the next set of requirements I will try to understand like whether it is a 02 CRLF. we know that there are different modes of operations it will transmit alive message you can see here so this is a alive message which is we transmit and those alive message can be observed in the can at regular interval so there is a requirement saying that every one second, the messages are analyzed. so using a tool something like analyzer, we can see that there is a message coming out of the embedded in it instrument so this is what we are going to prove so we got this case is written.

(Refer Slide Time: 49:37)



and inputs we know output we can expect and we can do the testing, the next requirement you a shall communicate it' is a live status at least once exactly only it is depth below so together we

can test this requirement to the above one or while testing this, this testing also can be taken care. So this gives us switch off sorry let's start this which are simpler one switch on the EU I wait for one second expect the can message or can what is that called alive status. Every second next you switch on switch of the EUI equal to no expect no can a live message so these other case that's all pretty simple. so two test cases we can draw out of this one this is about communication requirement but this is a simpler one I have put there could be a more complicated ones where you need to combine as i said in the official one different requirements oh that is called as a grouping grouping of the requirements here I have grouped both the requirements I have all right 12 and 13.

Looping of requirements or I would say a grouping of tests given to address. The multiple requirements so best then the good thing that we can adopt is plainly you write the test cases first you take one by one write a test case for this registers for giving some other requirement then you group all of them under communication requirements that will be called a perfect test strategy for communication to come.

So that the strategy will have communication requirements record test cases that is all we can now take care of that, the next one is about washed up requirements you know washed up by now and it's part of the immune systems there is a independent unit it could be an internal watch dog or external watch dog, its basic job is to make sure that the processor and the software that is working inside the software is alive and is working all the time and if it is not working this watch dog will do a reset of that processor that is all it does.

So to do that it will have requirement in terms in terms of at what rate it should reset at what rate it should monitor etc so that is why it is called as the watch dog it basically watches. so if the embed unit instrument is not align it will try to reset and if it is alive it will not bark here,so that is why the name came as a dog actually. so to satisfy the dog you need to make sure that you are alive and it alive should be monitored by the watchdog unit. let's not going deep into wash them because I hope this explanation is enough for now okay let us try to understand no requirement so that we can write the test cases is it.

The UHL configure a move on to generate a reset to the microcontroller with threshold duration of five new medicines. So as I said so watch doggies should be configured the configuration when it is done during the installation so the configuration of the watchdog should be done such that it is configured at 500 seconds the watchdog is configured for 500 seconds. So when the microcontroller is reset if watchdog generate pulse duration of greater than piracy.

So the watchdog unit internally we will rate for 500 second, and it will do a reset after we 500 in second will generate a pulse basic. the embedded instrument shall calls the internal watchdog at least once every form of my second or less, you know why because watch dog unit will know that the regular controller threshold still it is making it happy that that next 500 seconds it can

execute its operations and in between it is going to pulse watchdog so that there is no result occurred, and it is clear that the embedding trained unit instrument is working fine.

So when micro controller operation performing continuously we have watchdog generate pulse at least once every 500 milliseconds or less, that means if micro controller operations are running a long time, more than five milliseconds it is going to reset it and the major controllers responsibility to make sure that it should pulse the watch it off so that there is no reset occurring. so that is what the meaning of the watchdog is the internal watch dog what I did to put it.


So that is what the understanding of the recall so it should be very clear about this one, if it is a complex system definitely we will all generic, idea then what are the models that we are working or the embed software testing that models at least and surrounding models should be clear of how it is going to be operated what are the requirements and what are the references that i need to know for this for example in this case, we need to have a reference of the embedded processor which processor I am using so which pin need to go through and now i can use it.

what are the instruments that I need it's also very important to arrive at the strategy so it is very important to understand requirements first then based on that we can go through the, we can write the test cases.

(Refer Slide Time: 56:50)


Exercise 1

- Identify Test cases and scenarios from a set of requirements with grouping, test procedures. Example to consider for this is the SRS for EUI (attached below).



SRS EUI

- Fill the Test review checklist filling for high-level testing. (with the below sample)



Rules checklist

In the next session we will try to identify the test case and try to understand the test cases and for these requirements we will write the scenarios a try to the group now this case is for the functionality are the feature set of the requirements. so next file one is about understand the requirements edge' test cases and scenarios from the set of recover with grouping an example to consider the SRS for the embedded in instrument, and extension of the this exercise is once we are done with the requirement understanding at test case this writing we should be able to write

the test case review checklist review technique is called for high level testing, because we are doing the requirement is, because this checklist is very important translate to open it these are rules for verification of distances.

(Refer Slide Time: 57:54)

RULES FOR VERIFICATION OF TEST CASES			
Rule	Verification objective description	Yes/No/NA	Comment
1	01. Are templates followed during development of Test Case document?		
2	02. Are the valid equivalence classes analysis applied to all inputs of high level requirements?		
3	03. Are limits values used for external signals to create specific cases?		
4	04. For inputs structured in a table, are bounds and one intermediate value of the table used to create cases?		
5	05. Is MCCOC functional analysis used for the logic equations?		
6	06. Are particular test cases created for time-related functions and state transitions?		
7	07. Is one robustness test cases created for each external inputs?		
8	08. Is identification format of test cases unique?		
9	09. Is each high level requirement covered by at least one test case, and by the correct one?		
10	10. Is each test case created covered by one scenario, and by the correct one?		
11	11. Are values of inputs and outputs defined for each test case created?		
12	12. Is each scenario qualified as 'normal_range', 'robustness' or 'review' type?		
13	13. Is the COTS test coverage analysis available?		
Guidelines			
14	01. Is the description of the scenarios clear and accurate?		
15	02. Are the expected results defined for each test scenario?		

so what we do basically here is this check list will make sure that our verification is strong and correct for the UN requirements, so that is what we try to do it and if there is no then we to provide a comment if there is yes it is fine, you are taken care of this checklist, and if it is not applicable we need to pop up so this is also an exercise we need to fill up this column as well as this one so we will try to go through this in the next session of this exercise. After that we will try to understand.

(Refer Slide Time: 58:36)

Exercise 2

- Identify and establish traceability matrix for the developed Test scenarios from the set of EUI requirements.
 - Identify upward traceability (TCs to Req.)
 - Identify downward traceability (Req. to TCs)

The accessibility as the exercise and we have more exercises.

(Refer Slide Time: 58:41)

Exercise 3

- Coverage testing tools and their details:
 - For a given set of requirements after identifying the test case/scenarios:
 - Establish a coverage using coverage tools
 - Establish structural coverage analysis
 - Generate various metrics and report for the entire set
- LDRAcover and Structural Coverage Analysis (HD).mp4*

Like exercise 3, 4 coverage tool using the LDRA will try to understand the coverage tool.

(Refer Slide Time: 58:53)

Exercise 3 – unit testing

- Stub function creation and unit testing using LDRA unit testing tool

Automate Unit Test Generation & Management with LDRAunit (HD).mp4

in LDRA extension unit testing we try to understand.

(Refer Slide Time: 58:59)

Exercise 4

- Target based testing demonstration:
 - Create project as per the example demonstrated in below link.
 - Learn how to debug the code
 - Learn how to set breakpoint and monitor the variables
 - Learn how to watch memory
 - Go through the linker generated map file for analysis

CodeWarrior Intro Create a Project and Debug Watch in HD (HD).mp4

Then the next class is about next exercise is about code warrior that is another ID, the next one is about few lab sessions from the YouTube Google we try to understand from various embedded projects.

(Refer Slide Time: 59:16)

Exercise 5

- Test Case & Defect Management with Testlink / Bugzilla:
 - Using the below link, establish the Testlink requirements, test cases, execution and results and the traceability
 - <http://demo.testlink.org/latest/login.php?note=logout>
 - Generate the Testlink report
 - Go through the Bugzilla possibly with an example.

How they are using it the last exercise set is being on the test case management and defect management using testlink bugzilla.

(Refer Slide Time: 59:26)

Exercise 6

- Walkthrough of Understand for C/C++, a static analysis tool:
 - Example embedded C code
 - Configure the project into Understand for C/C++ tool
 - Select the metrics for the analysis
 - Generate the project report
 - Analyze the report

And the fore most of the last one is the static analysis using understanding CC purpose.

(Refer Slide Time: 59:36)

Exercise 6

1. Out of 4 levels of testing below, which one static analysis tool:
 - Example embedded C code
 - Configure the project into Understand for C/C++ tool
 - Select the metrics for the analysis
 - Generate the project report

That tool will try to follow it, so with that we will end today's session and we will continue in the exercise 1 and 2 in the next session.