Parker, Bryan

From: Sullivan, Glenn <glenn.sullivan@cardinalhealth.com>
Sent: Wednesday, May 27, 2015 5:08 PM
To: Parker, Bryan; Pelke, Patricia
Cc: Claunch, Scott
Subject: RAI for Indy Pharmacy
Attachments: NRC RAI 4.zip; ATT00001.txt

Bryan, I have attached the following information in a zip file because one file was too large:

I have highlighted all the areas in the calibration document from Lab Impex and have confirmed from them that all the sources used are NIST traceable and no gases are used to calibrate this system.

I have attached the Validation and Verification document and have highlighted the areas of interest and have indicated the pages for ease of review.

Please let me know if you need anything else. Also, if you would send me an email indicating that you have received these documents. I will also be sending a hard copy via Fed Ex for your files.

I appreciate that you have taken the time to review this, but it would be of an utmost importance if we could have the licensed issued prior to your departure tomorrow afternoon.

Thanks

g

Glenn P. Sullivan, MBA
Manager, Regulatory Management, NPS
7200 Cardinal Place West, Dublin, OH 43017
614.757.9586 dir | 614.652.4838 fax

Cardinal Health
Nuclear Pharmacy Services
7000 Cardinal Place
Dublin, OH 43017
614.757.5000 tel

CardinalHealth

cardinalhealth.com

May 27, 2015
Bryan A. Parker
U.S. Nuclear Regulatory Commission
Region III – Division of Nuclear Materials Safety
2443 Warrenville Rd, Suite 210
Lisle, IL 60532-4352

Re: Additional Information for New Radioactive Materials License, Mail Control Number 584197, Cardinal Health – Radium-223 (Ra-223) Dispensing Facility, Indianapolis, IN (CN 584197)

Dear Mr. Parker:

Cardinal Health 414, LLC (Nuclear Pharmacy Services, hereafter Cardinal Health) submits additional information for our radioactive materials application for the above referenced control number in response to the request from your Department via phone call dated May 27, 2015. The Agency request is in italics followed by the Cardinal Health response.

1. *Please abbreviate the calibration procedure as detailed by Lab Impex and confirm that all sources used for calibration are NIST traceable.*

   Attachment B is in a short form indicating the calibration of the Lab Impex System using NIST traceable sources as indicated by Lab Impex. Those areas highlighted are the areas in which it is calibrated for radiation detection on an annual basis not to exceed 12 months. Those areas in gray, even in highlighted areas, are only done during initial calibration.

2. *Please indicate how computer algorithm is validated and verified via an independent means.*

   Several areas of the Verification and Validation procedure indicate areas in which Independence must be utilized. Independence is required for several roles and/or tasks when reviewing the software lifecycle.

   Independence is noted in the following areas:

   - Section 4.2.1.2 Validation Test(s) Production section (Page 15 of 29). This section is the initial design phase of the software validation and includes independence.

- Section 4.5.1 Verification and section 4.5.2 Validation (page 20 of 29): The test engineer should be independent to the software engineer who coded the software release. The definition of an Independent Engineer is listed on page 24 of 29.

- Section 5.0 Independence contains a table defining the independence required for specific tasks and reviews in the software engineering lifecycle located on page 28 of 29

A copy of the System Procedure for Software Design has been attached. Areas of interest have been highlighted.

If you have any additional questions please do not hesitate to contact me at 614-757-9586.

Sincerely,

Glenn Sullivan
Manager, Health Physics
Quality and Regulatory
Nuclear Pharmacy Services

Enclosure:

Attachment A – Short Form Calibration
Attachment B – System Procedure for Software Design

cc: Gregory Even, RSO NRC Xofigo license
Scott Claunch, CRSO
Rick Hasselkus
Katherine Benson
Willie Regits
Stacy Sternberg

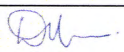
LIS DOCUMENT CONTROL RECORD

TITLE: SmartCAM 47 QPDI

0006/068

ISSUE: 1.3

Amendment No.	AMENDMENT RECORD AND DETAILS
0	Original based on 0006-53 1.7
1	Amended DP Sensor set up. Remove Auto Analogue Test. Place Auto Digi output test in production test.
2	Amended after CR dry run
3	Changed Upper KEV to 5700. Added Tick Box to 5.1 to move result to 5.2 due to missing 5.1 box.
4	Updated to show optional steps not required for annual recalibration. Optional text is shown in a lighter gray text.
5	
6	
7	
8	
9	

Amendment		**0**	**1**	**2**	**3**	**4**
PREPARED	BY	PM	SB	CR	SB	
	DATE	11/06/2012	03/09/2012	30/10/2012	16/04/13	
APPROVED	BY					
	DATE			17/01/13	16/04/13	
Amendment		**5**	**6**	**7**	**8**	**9**
PREPARED	BY					
	DATE					
APPROVED	BY					
	DATE					

SmartCAM 47

1 EQUIPMENT REQUIRED

1.1 Calibrated digital flow meter, record serial number in space provided.
1.2 Calibrated DVM, record serial number.
1.3 Calibrated barometer, record serial number in space provided.
1.4 Calibrated radioactive sources – ^{241}Am, ^{36}Cl, ^{14}C, ^{137}Cs.All Sources are NIST Traceable
1.5 Vacuum Pump.
1.6 Clean filter card.
1.7 PC with 9205 and VNC software.
1.8 Result sheet.

2 INITIAL CHECKS BEFORE SWITCH ON

2.1 Check the SmartCAM 47 has been tested to an approved production test and passed all tests specified.
2.2 Check build quality is in perfect condition.
2.3 Check crimps on connectors on I/O PCB
2.4 Record all serial numbers in the spaces provided on the result sheet.
2.5 Check that all software versions are at the latest issue, referring to LIS document 1000-CV03.
2.6 Record all software/firmware versions in the space provided.
2.7 Record the job number in the space provided.
2.8 Confirm that the resistance between the controller and head is less than 1Ω.
2.9 Check that the display is central and that all characters on the screen are visible.
2.10 Confirm touch screen is calibrated.

3 BAROMETRIC SET UP

3.1 Switch on unit and leave on with door closed for 1 hour.
3.2 Ensure that the security mode has been entered from the SmartCAM main screen
3.3 Using a calibrated barometer, record the pressure measured in mBar.
3.4 Select pressure from RHS menu and record the SmartCAM pressure reading. Check that it is within 10% of the calibrated barometer reading. Record Reading.
3.5 If the pressure is out of tolerance, recalibrates as per 0002-095-1-2.

4 FMU SET UP

4.1 Connect the pump and DFM to the SmartCAM 47 head. Ensure a new filter paper is fitted. Run pump for 15 minutes.
4.2 Adjust air flow to 1.06 cfm and confirm SmartCam flow display shows 1.06 cfm ±10%
4.3 Adjust air flow to 1.59 cfm and confirm SmartCam flow display shows 1.59 cfm ±10%
4.4 Adjust air flow to 1.31 cfm and confirm SmartCam flow display shows 1.31 cfm ±10%
4.5 If the flows are out of tolerance calibrate flow as per 0002-095-1-2.

5 FILTER DIFFERENTIAL TEST

5.1 In GPIO Options… → DP… →Filter DP Tab Confirm 34.47 in Full Scale DP.
5.2 Set the flow to 1.31 cfm. Check that the DP reading is 140mBar +/- 25%. Record.

6 ALPHA ENERGY CALIBRATION

6.1 In 'GPIO options' 'Barometric' tab. Press the button labelled 'Record energy cal environment'. Click yes to overwrite existing settings. Click OK.
6.2 Ensure 'Shift' reads '00000' before commencing. If it is not then repeat section 3.
6.3 If the unit is in peak fit mode, check that the five alpha channels are set from left to right as ^{238}U, ^{241}Am, ^{218}Po, ^{214}Po & ^{212}Po. If the unit is in ROI mode, check that the five alpha channels are set from left to right as Lo-ROI, Hi-ROI, ^{218}Po, ^{214}Po & ^{212}Po.
6.4 Insert the ^{241}Am source card in the card slot on the SmartCAM 47 head.
6.5 Check that the 'Confirm Filter Change' message appears. Click 'Yes'

6.6 From the 'Calibration' menu at the top of the display, select 'Alpha Energy...' and set the four parameters and five red bars as shown below in figure 6.1.

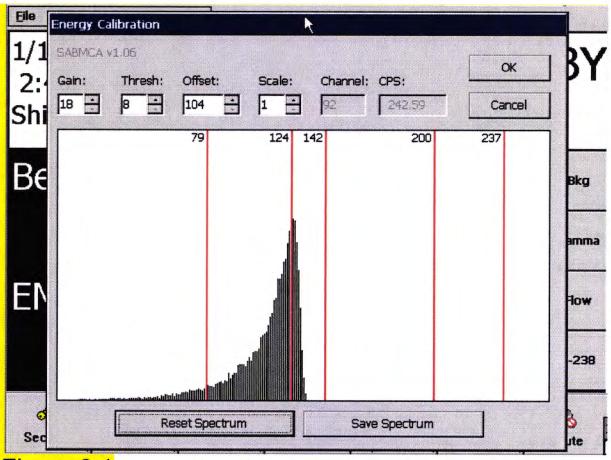


Figure 6.1

6.7 Check that after a short period of time, the spectrum takes the shape as shown in figure 6.1. Adjust RV1 on the IO board until the peak is in line with the '124' line, as demonstrated by figure 6.1.

6.8 Click the 'OK' at the top right of the window.

7 THRESHOLD & HV SET UP

7.1 Connect the SmartCAM 47 unit to the head via the interconnecting cable. Apply power to the unit.

7.2 Refer to Figure 3.1 to aid location of the components in the following tests on the IO board:

7.3 Check that the voltage between TP40 (orange ▨) (+) and TP52 (-) is 325mV ±3mV. Adjust RV6 (orange) if necessary.

7.4 Check that the voltage between TP41 (blue ▨) (+) and TP52 (-) is 325mV ±3mV. Adjust RV3 (blue) if necessary.

7.5 Check that the voltage between TP39 (green ▨) (+) and TP52 (-) is 300mV ±3mV. Adjust RV5 (green) if necessary.

7.6 Check that the voltage between TP42 (black ▨) (+) and TP52 (-) is 300mV ±3mV. Adjust RV4 (black) if necessary.

7.7 Check that on the IO board the voltage between the cathode of D2 and TP38 (0V) is 69V ±1V. Adjust RV7 if necessary.

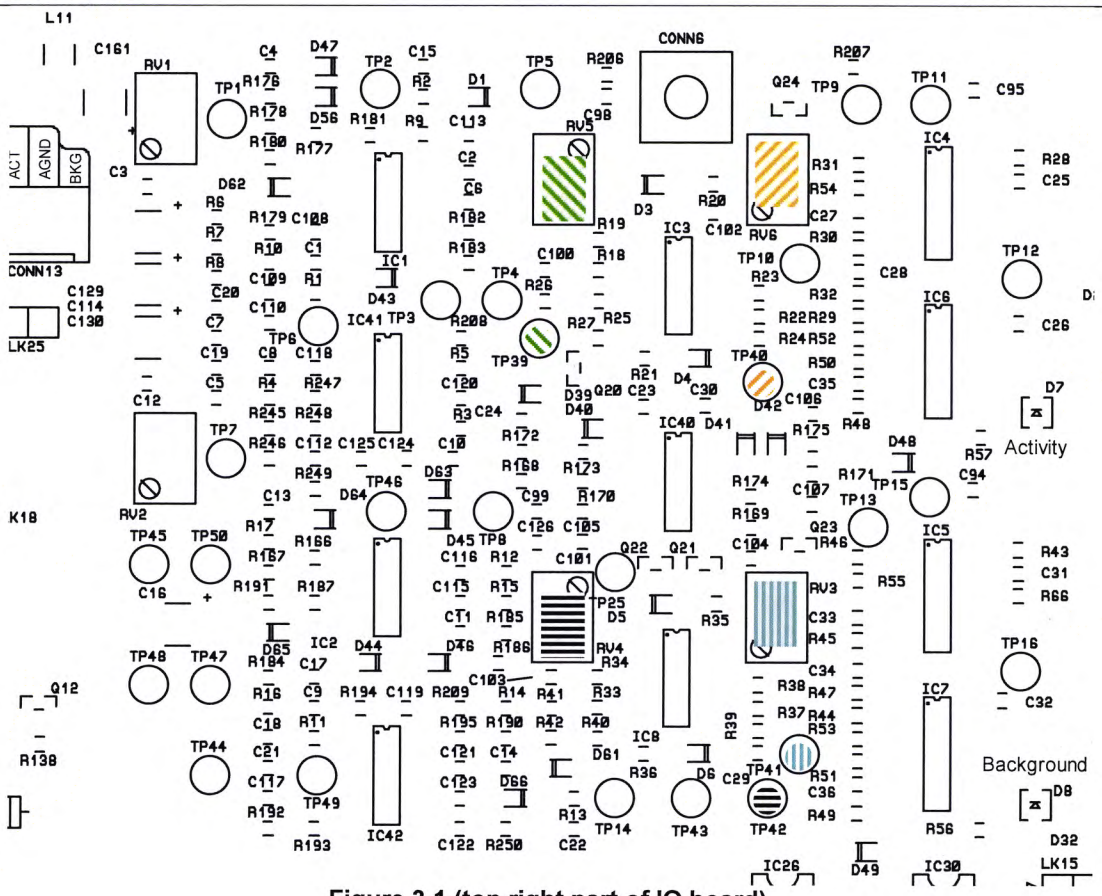


Figure 3.1 (top right part of IO board)

8 BACKGROUND CHANNEL CALIBRATION

8.1 Attach a ^{137}Cs source to the side of the SmartCAM 47 head, in the position shown in figure 7.1:

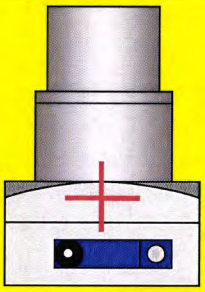


Figure 7.1

8.2 Set the display within the SmartCAM 47 application to show the raw channel data from the Beta and Bkg channels.

8.3 Within 'Channel Options' from the Monitor Menu, select the beta tab and select 'None' for Background Channel.

8.4 Adjust RV2 on the IO board until the channels show the same count rate, within 10% of the lower of the two. Note: the time constants for both channels may need changing.

8.5 Remove the ^{137}Cs source.

8.6 Check that the thresholds are still within the limits set in section 6.

8.7 Set the Background Channel back to 'Bkg' within the Beta tab of Channel Options.

9 ROI VERSION SETTINGS

9.1 If the Smart CAM is intended to be a ROI version check the 'yes' box, if not check to 'no' box and skip the rest of this section.
9.2 From the 'Monitor' menu select 'Instrument options' and go to the 'Peak fit' tab. Check the 'Use Regions of Interest' box.
9.3 Check and record that the thresholds are set as follows: Upper (keV): 5700, Middle (keV): 3500, Lower (keV): 2500.
9.4 From the 'Monitor' menu select 'Channel options' and set the channel names of the alpha isotopes to read (from left to right) Lo-ROI Chronic, Hi-ROI Chronic, Po-218 Chronic, Po-214 Chronic and Po-212 Chronic.

10 ALPHA EFFICIENCY CALIBRATION

10.1 Select a ^{241}Am source, record serial number and the emission rate (2π).
10.2 From the 'Calibration' menu at the top of the display, select 'Alpha Efficiency'
10.3 Insert the source surface emission rate and click the 'Start Calibration' button.
10.4 Click 'Accept' and check that the calculated efficiency is between 20-25%. Record the 4π efficiency.

11 BETA EFFICIENCY CALIBRATION

11.1 Select a ^{14}C source, record serial number and the emission rate (2π).
11.2 Insert the ^{14}C source card in the card slot on the SmartCAM 47 head.
11.3 From the 'Calibration' menu at the top of the display, select 'Beta Efficiency'
11.4 Insert the source surface emission rate and click the 'Start Calibration' button.
11.5 Click 'Accept' and check that the calculated efficiency is between 4-8%. Record the 4π efficiency.
11.6 Select a ^{36}Cl source, record serial number and the emission rate (2π).
11.7 Insert the ^{36}Cl source card in the card slot on the SmartCAM 47 head.
11.8 From the 'Calibration' menu at the top of the display, select 'Beta Efficiency'
11.9 Insert the source surface emission rate and click the 'Start Calibration' button.
11.10 Click 'Accept' and check that the calculated efficiency is between 18-25%. Record the 4π efficiency.0

12 BACKGROUND ADUSTMENT

12.1 Set the display within the SmartCAM 47 application to show the raw channel data from the Beta and Bkg channels, and set the time constants for both of these channels to 5 minutes.
12.2 Perform an instrument reset and record the count rate after 5 minutes for both beta and bkg channels and check that it does not exceed 0.5cps in either channel.

13 RADON TESTS

13.1 Connect the pump and DFM to the SmartCAM 47 head. Adjust the airflow on the head to give 37LPM on the DFM.
13.2 From the 'Calibration' menu at the top of the display, select 'Alpha Energy…'
13.3 Allow at least 8 hours to elapse. Check that peaks can be observed on the 'Energy Calibration' spectrum at the '142' and '200' lines.
13.4 Click 'Cancel' on the energy calibration window.

14 ADDITIONAL TESTS

14.1 Plug the SmartCAM 47 into an active network, via the Ethernet connector on the rear or bottom of the unit.
14.2 Determine the IP address of the SmartCAM 47 and confirm that it can be accessed using VNC software.
14.3 Using 'FileZilla' or similar FTP software, check that the root of the flash disc can be successfully accessed.
14.4 Check Works Order (if available) for any customer communications protocols required and test accordingly.
14.5 Remove the mains power from the unit. Check that the unit remains on, showing the 'Power Fail' message with the sounder active.
14.6 Select 'Exit' from the 'File' menu. Save a copy of the following parameter configuration files: \CFDisk\default.cfg and \CFDisk\SmartCAM.ini
14.7 Select 'Shutdown' from the 'File' menu.
14.8 When prompted '… safe to switch off', remove and replace the battery connection from CONN37 to power down the unit.

14.9 Remove all temporary stickers, wrappings, debris, loose wires, fastenings and paperwork from the unit to ensure that it is ready for shipping.

14.10 Remove any excess thread sealant from the flow control valve on the head, as well as any other lubricants or sealants on any surface.

14.11 Check that there are no surface scratches or abrasions on any surface.

14.12 Clean the external surfaces and the enclosure of the CAM from dust, particulate, weld splatter, corrosion, flux, oils and greases and ink.

14.13 Fill out a 'TESTED' label and affix to the unit in an appropriate place.

14.14 Fit a clean filter paper.

RESULT SHEET

Job Number	
SmartCAM 47 Unit Serial Number	
IO Board Serial Number	
Processing Board Serial Number	
SmartCAM 47 Head Serial Number	

IO PIC Firmware Version	

Touchscreen PIC Firmware Version	
LISMCA Firmware	
SmartCAM 47 Application Version	

Equipment

DFM	
DVM	
Barometer	

2.1 Pass ☐
2.2 Pass ☐
2.3 Pass ☐
2.4 Pass ☐
2.5 Pass ☐
2.6 Pass ☐
2.7 Pass ☐
2.8 Pass ☐
2.9 Pass ☐
2.10 Pass ☐

3.3
3.4

4.1 Pass ☐
4.2 cfm
4.3 cfm
4.4 cfm

5.1 Pass ☐
5.2

Test Passed:

6.1 Pass ☐
6.2 Pass ☐
6.3 Pass ☐
6.4 Pass ☐
6.5 Pass ☐
6.6 Pass ☐
6.7 Pass ☐
6.8 Pass ☐

7.3 Pass ☐
7.4 Pass ☐
7.5 Pass ☐
7.6 Pass ☐
7.7 Pass ☐

8.1 Pass ☐
8.2 Pass ☐
8.3 Pass ☐
8.4 Pass ☐

8.5 Pass ☐
8.6 Pass ☐
8.7 Pass ☐

9.1 Yes ☐ No ☐
9.2 Checked ☐
9.3 Upper
Middle
Lower
9.4 Pass ☐

10.1 LIS No.
Rate CPS
10.4 Pass ☐
%

11.1 LIS No.
Rate CPS
11.5 Pass ☐
%
11.6 LIS No.
Rate CPS
11.10 Pass ☐
%

12.1 Pass ☐

12.2 Beta
Bkg

13.1 Pass ☐
13.2 Pass ☐
13.3 Pass ☐
13.4 Pass ☐

14.1 Pass ☐
14.2 Pass ☐
14.3 Pass ☐
14.4 Pass ☐
14.5 Pass ☐
14.6 Pass ☐
14.7 Pass ☐
14.8 Pass ☐
14.9 Pass ☐
14.10 Pass ☐
14.11 Pass ☐
14.12 Pass ☐
14.13 Pass ☐
14.14 Pass ☐

Quality Assured:

X _____

Test Engineer

Date:

X _____

Quality Engineer

Date:

LIS DOCUMENT CONTROL RECORD

TITLE: SmartCAM 47 QPDI

0006/068

ISSUE: 1.3

Amendment No.	AMENDMENT RECORD AND DETAILS
0	Original based on 0006-53 1.7
1	Amended DP Sensor set up. Remove Auto Analogue Test. Place Auto Digi output test in production test.
2	Amended after CR dry run
3	Changed Upper KEV to 5700. Added Tick Box to 5.1 to move result to 5.2 due to missing 5.1 box.
4	Updated to show optional steps not required for annual recalibration. Optional text is shown in a lighter gray text.
5	
6	
7	
8	
9	

Amendment		**0**	**1**	**2**	**3**	**4**
PREPARED	BY	PM	SB	CR	SB	
	DATE	11/06/2012	03/09/2012	30/10/2012	16/04/13	
APPROVED	BY					
	DATE			17/01/13	16/04/13	

Amendment		**5**	**6**	**7**	**8**	**9**
PREPARED	BY					
	DATE					
APPROVED	BY					
	DATE					

SmartCAM 47

1 EQUIPMENT REQUIRED

1.1 Calibrated digital flow meter, record serial number in space provided.
1.2 Calibrated DVM, record serial number.
1.3 Calibrated barometer, record serial number in space provided.
1.4 Calibrated radioactive sources – ^{241}Am, ^{36}Cl, ^{14}C, ^{137}Cs.All Sources are NIST Traceable
1.5 Vacuum Pump.
1.6 Clean filter card.
1.7 PC with 9205 and VNC software.
1.8 Result sheet.

2 INITIAL CHECKS BEFORE SWITCH ON

2.1 Check the SmartCAM 47 has been tested to an approved production test and passed all tests specified.
2.2 Check build quality is in perfect condition.
2.3 Check crimps on connectors on I/O PCB
2.4 Record all serial numbers in the spaces provided on the result sheet.
2.5 Check that all software versions are at the latest issue, referring to LIS document 1000-CV03.
2.6 Record all software/firmware versions in the space provided.
2.7 Record the job number in the space provided.
2.8 Confirm that the resistance between the controller and head is less than 1Ω.
2.9 Check that the display is central and that all characters on the screen are visible.
2.10 Confirm touch screen is calibrated.

3 BAROMETRIC SET UP

3.1 Switch on unit and leave on with door closed for 1 hour.
3.2 Ensure that the security mode has been entered from the SmartCAM main screen
3.3 Using a calibrated barometer, record the pressure measured in mBar.
3.4 Select pressure from RHS menu and record the SmartCAM pressure reading. Check that it is within 10% of the calibrated barometer reading. Record Reading.
3.5 If the pressure is out of tolerance, recalibrates as per 0002-095-1-2.

4 FMU SET UP

4.1 Connect the pump and DFM to the SmartCAM 47 head. Ensure a new filter paper is fitted. Run pump for 15 minutes.
4.2 Adjust air flow to 1.06 cfm and confirm SmartCam flow display shows 1.06 cfm ±10%
4.3 Adjust air flow to 1.59 cfm and confirm SmartCam flow display shows 1.59 cfm ±10%
4.4 Adjust air flow to 1.31 cfm and confirm SmartCam flow display shows 1.31 cfm ±10%
4.5 If the flows are out of tolerance calibrate flow as per 0002-095-1-2.

5 FILTER DIFFERENTIAL TEST

5.1 In GPIO Options… → DP… →Filter DP Tab Confirm 34.47 in Full Scale DP.
5.2 Set the flow to 1.31 cfm. Check that the DP reading is 140mBar +/- 25%. Record.

6 ALPHA ENERGY CALIBRATION

6.1 In 'GPIO options' 'Barometric' tab. Press the button labelled 'Record energy cal environment'. Click yes to overwrite existing settings. Click OK.
6.2 Ensure 'Shift' reads '00000' before commencing. If it is not then repeat section 3.
6.3 If the unit is in peak fit mode, check that the five alpha channels are set from left to right as ^{238}U, ^{241}Am, ^{218}Po, ^{214}Po & ^{212}Po. If the unit is in ROI mode, check that the five alpha channels are set from left to right as Lo-ROI, Hi-ROI, ^{218}Po, ^{214}Po & ^{212}Po.
6.4 Insert the ^{241}Am source card in the card slot on the SmartCAM 47 head.
6.5 Check that the 'Confirm Filter Change' message appears. Click 'Yes'

6.6 From the 'Calibration' menu at the top of the display, select 'Alpha Energy...' and set the four parameters and five red bars as shown below in figure 6.1.

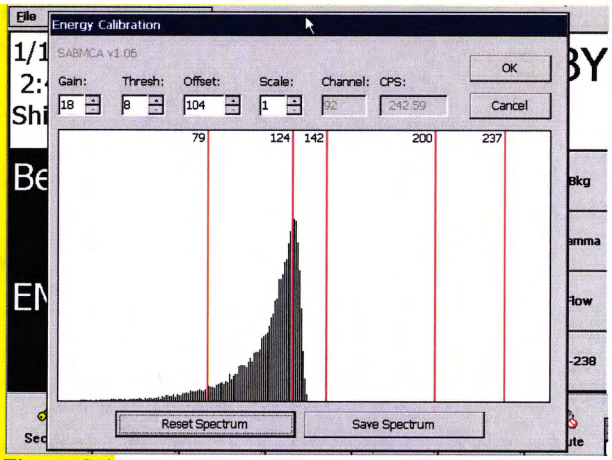


Figure 6.1

6.7 Check that after a short period of time, the spectrum takes the shape as shown in figure 6.1. Adjust RV1 on the IO board until the peak is in line with the '124' line, as demonstrated by figure 6.1.

6.8 Click the 'OK' at the top right of the window.

7 THRESHOLD & HV SET UP

7.1 Connect the SmartCAM 47 unit to the head via the interconnecting cable. Apply power to the unit.

7.2 Refer to Figure 3.1 to aid location of the components in the following tests on the IO board:

7.3 Check that the voltage between TP40 (orange) (+) and TP52 (-) is 325mV ±3mV. Adjust RV6 (orange) if necessary.

7.4 Check that the voltage between TP41 (blue) (+) and TP52 (-) is 325mV ±3mV. Adjust RV3 (blue) if necessary.

7.5 Check that the voltage between TP39 (green) (+) and TP52 (-) is 300mV ±3mV. Adjust RV5 (green) if necessary.

7.6 Check that the voltage between TP42 (black) (+) and TP52 (-) is 300mV ±3mV. Adjust RV4 (black) if necessary.

7.7 Check that on the IO board the voltage between the cathode of D2 and TP38 (0V) is 69V ±1V. Adjust RV7 if necessary.

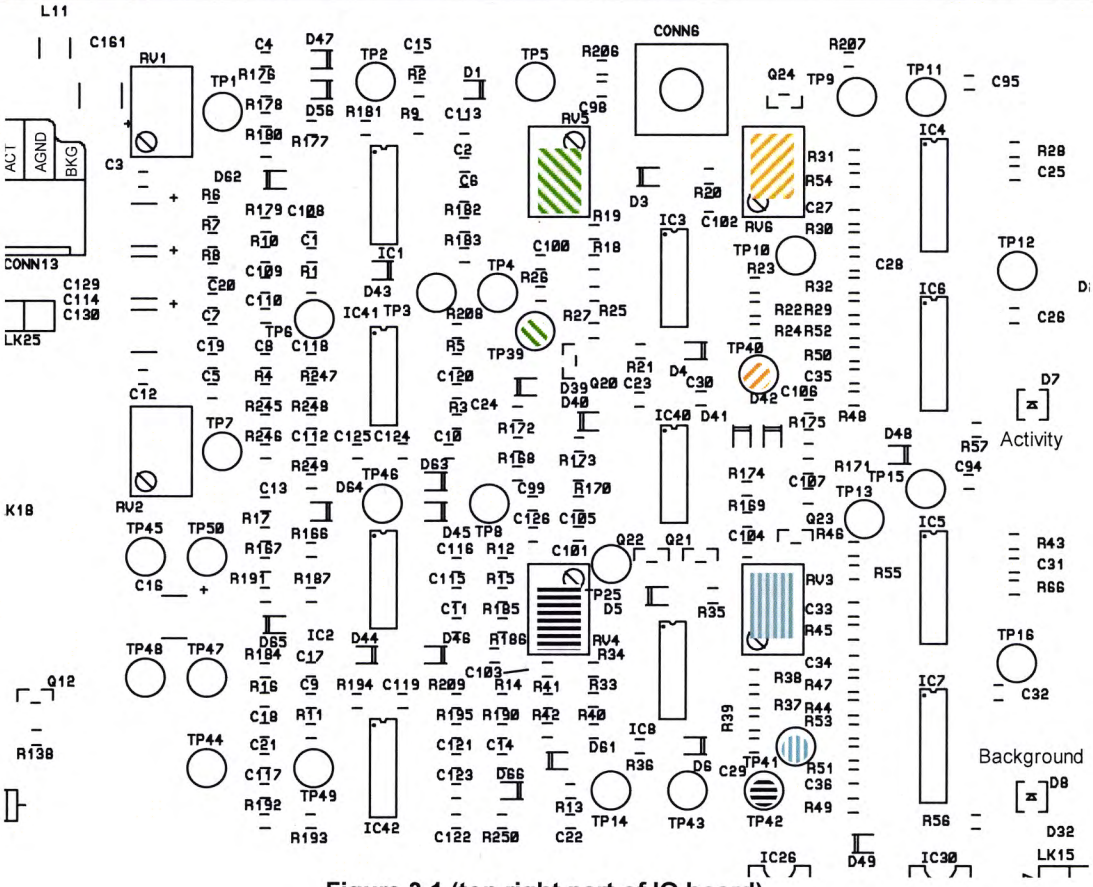


Figure 3.1 (top right part of IO board)

8 BACKGROUND CHANNEL CALIBRATION

8.1 Attach a ^{137}Cs source to the side of the SmartCAM 47 head, in the position shown in figure 7.1:

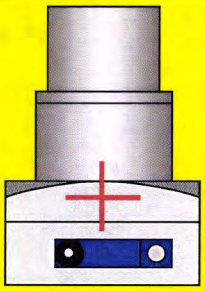


Figure 7.1

8.2 Set the display within the SmartCAM 47 application to show the raw channel data from the Beta and Bkg channels.

8.3 Within 'Channel Options' from the Monitor Menu, select the beta tab and select 'None' for Background Channel.

8.4 Adjust RV2 on the IO board until the channels show the same count rate, within 10% of the lower of the two. Note: the time constants for both channels may need changing.

8.5 Remove the ^{137}Cs source.

8.6 Check that the thresholds are still within the limits set in section 6.

8.7 Set the Background Channel back to 'Bkg' within the Beta tab of Channel Options.

9 ROI VERSION SETTINGS

9.1 If the Smart CAM is intended to be a ROI version check the 'yes' box, if not check to 'no' box and skip the rest of this section.
9.2 From the 'Monitor' menu select 'Instrument options' and go to the 'Peak fit' tab. Check the 'Use Regions of Interest' box.
9.3 Check and record that the thresholds are set as follows: Upper (keV): 5700, Middle (keV): 3500, Lower (keV): 2500.
9.4 From the 'Monitor' menu select 'Channel options' and set the channel names of the alpha isotopes to read (from left to right) Lo-ROI Chronic, Hi-ROI Chronic, Po-218 Chronic, Po-214 Chronic and Po-212 Chronic.

10 ALPHA EFFICIENCY CALIBRATION

10.1 Select a ^{241}Am source, record serial number and the emission rate (2π).
10.2 From the 'Calibration' menu at the top of the display, select 'Alpha Efficiency'
10.3 Insert the source surface emission rate and click the 'Start Calibration' button.
10.4 Click 'Accept' and check that the calculated efficiency is between 20-25%. Record the 4π efficiency.

11 BETA EFFICIENCY CALIBRATION

11.1 Select a ^{14}C source, record serial number and the emission rate (2π).
11.2 Insert the ^{14}C source card in the card slot on the SmartCAM 47 head.
11.3 From the 'Calibration' menu at the top of the display, select 'Beta Efficiency'
11.4 Insert the source surface emission rate and click the 'Start Calibration' button.
11.5 Click 'Accept' and check that the calculated efficiency is between 4-8%. Record the 4π efficiency.
11.6 Select a ^{36}Cl source, record serial number and the emission rate (2π).
11.7 Insert the ^{36}Cl source card in the card slot on the SmartCAM 47 head.
11.8 From the 'Calibration' menu at the top of the display, select 'Beta Efficiency'
11.9 Insert the source surface emission rate and click the 'Start Calibration' button.
11.10 Click 'Accept' and check that the calculated efficiency is between 18-25%. Record the 4π efficiency.0

12 BACKGROUND ADUSTMENT

12.1 Set the display within the SmartCAM 47 application to show the raw channel data from the Beta and Bkg channels, and set the time constants for both of these channels to 5 minutes.
12.2 Perform an instrument reset and record the count rate after 5 minutes for both beta and bkg channels and check that it does not exceed 0.5cps in either channel.

13 RADON TESTS

13.1 Connect the pump and DFM to the SmartCAM 47 head. Adjust the airflow on the head to give 37LPM on the DFM.
13.2 From the 'Calibration' menu at the top of the display, select 'Alpha Energy…'
13.3 Allow at least 8 hours to elapse. Check that peaks can be observed on the 'Energy Calibration' spectrum at the '142' and '200' lines.
13.4 Click 'Cancel' on the energy calibration window.

14 ADDITIONAL TESTS

14.1 Plug the SmartCAM 47 into an active network, via the Ethernet connector on the rear or bottom of the unit.
14.2 Determine the IP address of the SmartCAM 47 and confirm that it can be accessed using VNC software.
14.3 Using 'FileZilla' or similar FTP software, check that the root of the flash disc can be successfully accessed.
14.4 Check Works Order (if available) for any customer communications protocols required and test accordingly.
14.5 Remove the mains power from the unit. Check that the unit remains on, showing the 'Power Fail' message with the sounder active.
14.6 Select 'Exit' from the 'File' menu. Save a copy of the following parameter configuration files: \CFDisk\default.cfg and \CFDisk\SmartCAM.ini
14.7 Select 'Shutdown' from the 'File' menu.
14.8 When prompted '… safe to switch off', remove and replace the battery connection from CONN37 to power down the unit.

14.9	Remove all temporary stickers, wrappings, debris, loose wires, fastenings and paperwork from the unit to ensure that it is ready for shipping.
14.10	Remove any excess thread sealant from the flow control valve on the head, as well as any other lubricants or sealants on any surface.
14.11	Check that there are no surface scratches or abrasions on any surface.
14.12	Clean the external surfaces and the enclosure of the CAM from dust, particulate, weld splatter, corrosion, flux, oils and greases and ink.
14.13	Fill out a 'TESTED' label and affix to the unit in an appropriate place.
14.14	Fit a clean filter paper.

RESULT SHEET

Job Number	
SmartCAM 47 Unit Serial Number	
IO Board Serial Number	
Processing Board Serial Number	
SmartCAM 47 Head Serial Number	

IO PIC Firmware Version	

Touchscreen PIC Firmware Version	
LISMCA Firmware	
SmartCAM 47 Application Version	

Equipment

DFM	
DVM	
Barometer	

2.1 Pass ☐
2.2 Pass ☐
2.3 Pass ☐
2.4 Pass ☐
2.5 Pass ☐
2.6 Pass ☐
2.7 Pass ☐
2.8 Pass ☐
2.9 Pass ☐
2.10 Pass ☐

3.3
3.4

4.1 Pass ☐
4.2 cfm
4.3 cfm
4.4 cfm

5.1 Pass ☐
5.2

Test Passed:

6.1 Pass ☐
6.2 Pass ☐
6.3 Pass ☐
6.4 Pass ☐
6.5 Pass ☐
6.6 Pass ☐
6.7 Pass ☐
6.8 Pass ☐

7.3 Pass ☐
7.4 Pass ☐
7.5 Pass ☐
7.6 Pass ☐
7.7 Pass ☐

8.1 Pass ☐
8.2 Pass ☐
8.3 Pass ☐
8.4 Pass ☐

8.5 Pass ☐
8.6 Pass ☐
8.7 Pass ☐

9.1 Yes ☐ No ☐
9.2 Checked ☐
9.3 Upper
Middle
Lower
9.4 Pass ☐

10.1 LIS No.
Rate CPS
10.4 Pass ☐
%

11.1 LIS No.
Rate CPS
11.5 Pass ☐
%
11.6 LIS No.
Rate CPS
11.10 Pass ☐
%

12.1 Pass ☐

Quality Assured:

12.2 Beta
Bkg

13.1 Pass ☐
13.2 Pass ☐
13.3 Pass ☐
13.4 Pass ☐

14.1 Pass ☐
14.2 Pass ☐
14.3 Pass ☐
14.4 Pass ☐
14.5 Pass ☐
14.6 Pass ☐
14.7 Pass ☐
14.8 Pass ☐
14.9 Pass ☐
14.10 Pass ☐
14.11 Pass ☐
14.12 Pass ☐
14.13 Pass ☐
14.14 Pass ☐

X _____

Test Engineer

Date:

X _____

Quality Engineer

Date:

Uncontrolled Copy When Printed

SYSTEM PROCEDURE

FOR

SOFTWARE DESIGN

SP 14.0

Head Office Tel: **+44 (0)1202 684848**
Northern Office +44 (0) 19467 28128
Head Office Fax: **+44 (0)1202 683571**
E mail: info@labimpex.com
www.labimpex.com

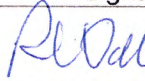
Lab Impex Systems Ltd
Impex House, 21 Harwell Rd
Nuffield Industrial Estate
Poole, Dorset BH17 0GE

Certificate No. Q 09160
BS EN ISO 9001:2008

CONTENTS

FIGURES

TABLES

	Name	Position	Signature	Date
Prepared By	R Walter	Software Design Supervisor		26.08.2011
Approved By	D Morgan	Quality Manager		26.08.2011

Document History

Issue	Date	Author	Summary of Changes
1 - 3	28.08.09	R J Phillipson	First Issue and archive record
4	26.08.2011	R Walter	Inclusion of NQA-1, ISO/IEC 12207:2008, and IEC61508 (specifically part 3) requirements.

1.0 Purpose

The purpose of this procedure is to establish and define in detail the software development and maintenance lifecycle for all software developed at Lab Impex Systems Ltd. This Software Development Procedure is designed to allow LIS Products to comply with the applicable requirements of:

- BS EN ISO 9001:2008 (Ref. 1)
- ISO/IEC 12207:2008 (Ref. 2)
- IEC61508:2010 (Ref. 3)*
- NQA-1:2000, NQA-1:2008 (Ref. 4)

*Note: This procedure is applicable for software development and maintenance of safety systems with a target safety integrity level not greater than SIL2. This procedure describes a software development lifecycle to meet the applicable requirements defined in part 3 of IEC61508:2010.

2.0 Scope

This procedure defines the following software lifecycle phases for all Computer Software Configuration Items:

- Requirements Analysis
  - Definition
  - Identification
  - Interfaces
  - Traceability
  - Planning
  - Impact Analysis
  - Review
- Software Design
  - Initial
  - Detailed
- Design Review
- Implementation
  - Coding
  - Code Review
- Integration
- Testing
  - Verification
  - Validation
  - Test Review
- Installation
- Acceptance
- Post Acceptance Activities
- Configuration Management

This procedure applies to personnel involved in Software Development.

2.1 Acronyms & Abbreviations

BS	British Standard
CCB	Change Control Board
CMP	Configuration Management Plan
CRC	Cyclic Redundancy Check
CSCI	Computer Software Configuration Item
CSC	Computer Software Component
CI	Configured Item
ECR	Engineering Change Request
IEC	International Electrotechnical Commission
ISO	International Standards Organisation
LIS	Laboratory Impex Systems Ltd
NQA	Nuclear Quality Assurance
SAF	Software Acceptance Form
SCR	Software Change Request
SCM	Software Configuration Management
SDD	System Definition Document
SIL	Safety Integrity Level
SP	System Procedure
SQEP	Suitably Qualified and Experienced Person
SRT	Service Request Ticket
SRF	Software Review Form
SSS	Software System Specification
STF	Software Test Form
SRTM	Software Requirements Traceability Matrix
SW	Software
UML	Unified Modelling Language
URS	User Requirements Specification
WI	Work Instruction

3.0 Overview

Application software may be designed and produced as part of product development. The software development process is applicable to a range of products and customer requirements that specify different conformance and qualification. The software development lifecycle is based on a standard V-Model with options for different processes dependant on product/project/customer specific non-functional requirements.

A simplified overview of the Software Development V-Model is provided in Figure 1.

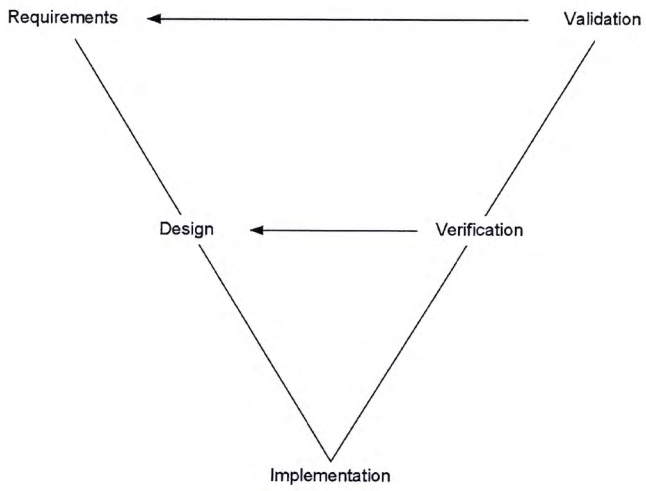


Figure 1: Overview Software Development V-Model

The Software Development V-Model ensures that all software designs are verified and developed software is validated against the input requirements. The V-Model is the basic framework for software development with optional tasks and processes that can be utilised to achieve the required due process and/or rigour.

3.1 Compliance Matrix

To achieve the requirements from the industry standards listed in Section 1.0, the following table (Table 1) defines the processes required. The table provides details of each standard with the additional process tasks that must be undertaken to achieve compliance.

This table should be used during Requirements and Impacts stages of the lifecycle to ensure the appropriate tasks are carried out to meet the requirements.

SP-14 Task	Task Description	Mandatory (all software development)	Standard	
			IEC61508 (up to SIL 2)	NQA-1
Software Requirements Analysis				
4.1.1	Definition of Software Requirements	✓	✓	✓
4.1.2	Identification of Functional Safety Requirements		✓	✓
4.1.3	Definition of Software Interfaces		✓	✓
4.1.4	Requirements Traceability	✓	✓	✓
4.1.5	Implementation Planning		✓	✓
4.1.6	Version Control of Software Requirements		✓	✓
4.1.7	Requirements change management		✓	✓
4.1.8	Impact Analysis	✓	✓	✓
4.1.9	Requirements review			✓
Software Design				
4.2.1	Initial Design Phase			
4.2.2	Initial Design Review			
4.2.3	Detailed Design Phase	✓	✓	✓
4.2.4	Detailed Design Review	✓	✓	✓
Implementation				
4.3.1	Code Documentation		✓	✓
4.3.2	Code Review	✓	✓	✓
Software Integration and Unit Test				
4.4.1	Integration	✓	✓	✓
4.4.3	Unit Testing	✓	✓	✓
4.4.4	Regression Strategy			
Software Testing				
4.5.1	Verification Testing	✓	✓	✓
4.5.2	Validation Testing	✓	✓	✓
4.5.3	Test Results Review	✓	✓	✓
Configuration Management				
4.10.1	SCM Implementation	✓	✓	✓
Functional Safety Assessment				
4.12	Functional Safety Assessment		✓	

Table 1: Compliance Matrix

Note:
This procedure defines the software lifecycle elements of compliance with IEC61508 (Ref. 3) it does not define project planning, management or system design elements.

4.0 Software Development Lifecycle

The Software Development Lifecycle is made up of the following activities:

- Requirements Analysis
- Software Design
- Implementation
- Integration
- Testing
- Installation
- Operation
- Maintenance
- Configuration Management
- Audit
- Functional Safety Assessment

Some of the activities defined within this procedure must only be carried out by personnel specifically qualified for the task. The qualification of personnel is defined in SP6.0 (Ref. 21).

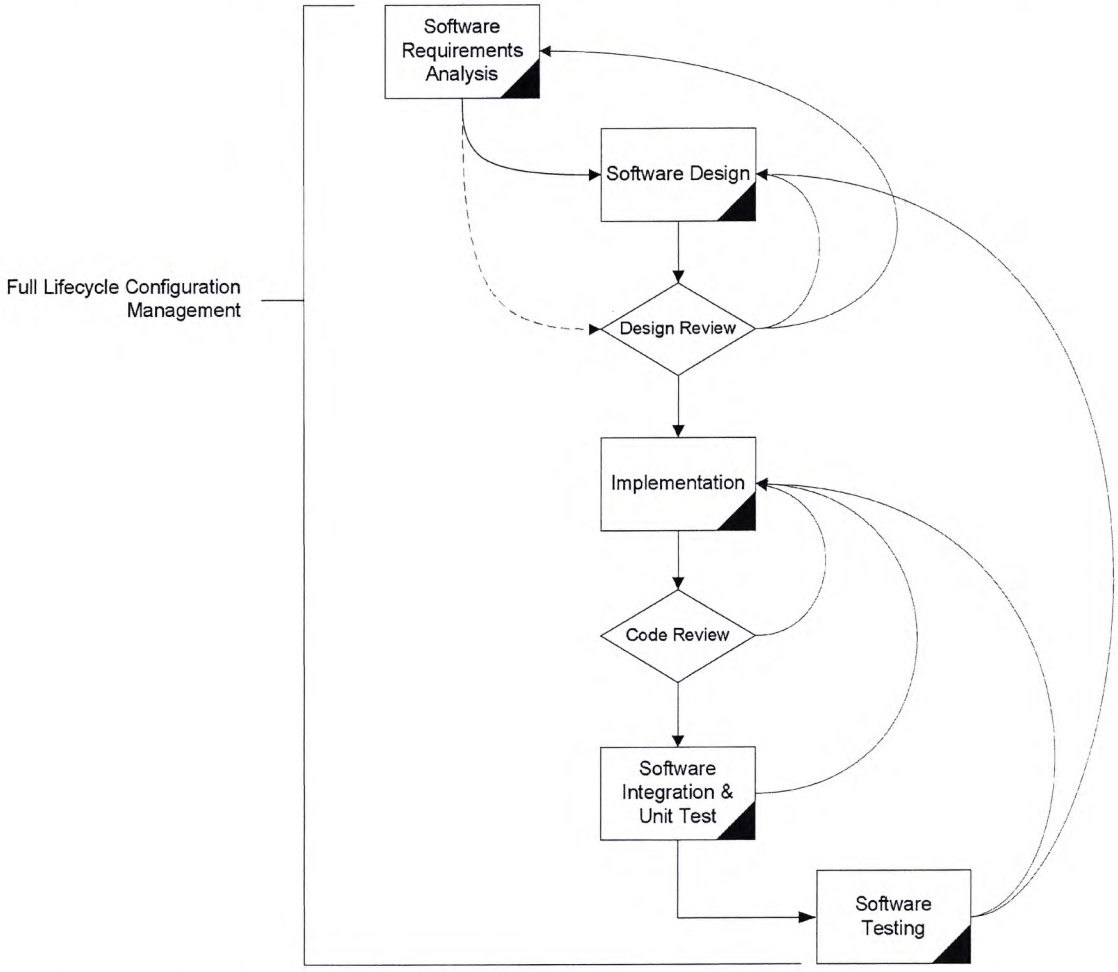


Figure 2: Simplified Software Development Lifecycle

The Software Development Lifecycle is made up of four distinct processes – Requirements, Design, Implementation and Test (see Figure 2); the lifecycle is based on the requirements of ISO/IEC

12207:2008 (Ref. 2). Figure 2 shows the flow of requirements into design, and the rework process from reviews and testing.

4.1 Software Requirements Analysis

The Software Requirements Analysis process defines the software requirements for the system. This process identifies the specific functional software requirements and allows for derivation of inferred requirements from associated system engineering tasks. In addition any non-functional requirements that are software specific are captured and used in the design methodology to tailor the lifecycle.

The software requirements must be documented as part of specification; they may also be captured in the Software Requirements Traceability Matrix (Ref. 11) where qualification method and traceability are defined. The Software Requirements Traceability Matrix can be amended throughout the development lifecycle to provide traceability of all requirements, for example design documentation, testing, etc.

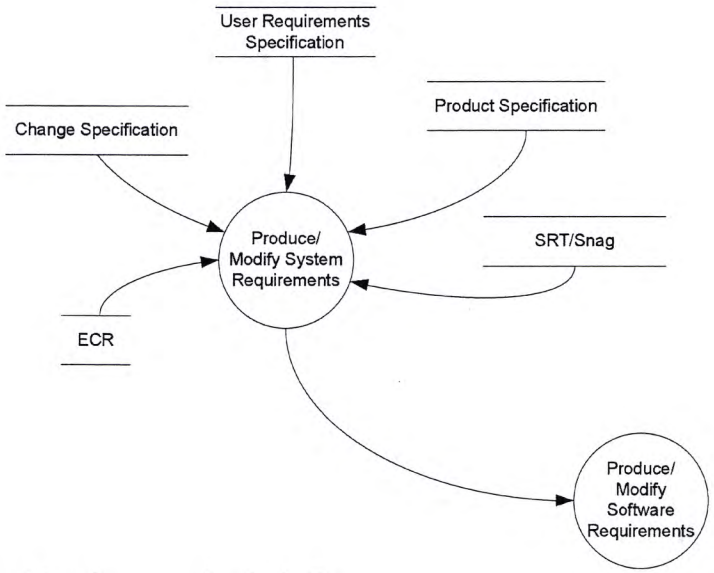


Figure 3: Overview Requirements Analysis Process

The software requirements are captured and derived from system design activities. The system design processes should capture the system requirements from the following inputs:

- Product Specification – new product development specification;
- Change Specification – customer or sales input for specific functionality;
- Engineering Change Request (ECR, Ref. 26) – from Service Request Ticket, System Design Impact, Modification, etc.;
- User Requirement Specification – either internal or external formal requirements specification;
- Service Request Ticket (SRT, Ref. 25) or Software Change Request (Snag/Fault Report) – depending on severity this may have a direct effect on the software requirements and so may not require a formal ECR;
- Prototyping or testing;
- Technical sales input.

The following tasks are addressed by software requirements analysis:

- Software requirements are defined;
- Safety requirements affecting software are identified;
- Software interfaces are defined;
- Software requirements are correct and testable;
- Consistency and traceability are established between the software requirements and system requirements;
- The implementation of the software requirements is planned;
- The software requirements are approved and updates are managed;
- Software requirement change is evaluated for cost, schedule and technical impact;
- The software requirements are baselined and communicated to all affected parties.

4.1.1 Definition of Software Requirements

The software requirements must be defined in a simple unique format for example using the specification or in a Software Requirements Traceability Matrix (Ref. 11). As a minimum the requirements must be documented in a Software Requirements Specification. The Software Requirements should be documented to provide for the following information:

- Requirement Number (ID) - a unique number that identifies the requirement. This number provides a simple level of traceability back to the system identifier; for example, Requirement Number *XYZ.SW.1234* would be the software requirement *1234* of the *XYZ* System.
- Requirement Type – Functional (software or hardware functionality) or Non-Functional (properties, e.g. compliance or conformance requirement), etc.
- Requirement Description

4.1.2 Identification of Functional Safety Requirements

Software requirements that are related to or contribute towards a safety function must be clearly identified in the Software Requirements Traceability Matrix. The identification includes assignment of a target SIL level to each requirement where possible. Functional Safety requirements are a product of Systems and Safety Engineering and are subject to Functional Safety Assessment (see Section 4.12).

4.1.3 Definition of Software Interfaces

For complex software the requirements for software interfaces between major systems must be documented as part of the software requirements. Interface requirements should include details of:

- Interfacing systems
- Interface types
- Constraints

For example:
Interfacing Systems: *Requirement Number XYZ.SW.1234, The datalogger software shall trend data from controllers X and Y.*
Interface type: *Requirement Number XYZ.SW.1234.1, The datalogger software shall interrogate serial (RS232) data from ports 1 & 2 and check for CRC validation.*
Constraints: *Requirement Number XYZ.SW.1234.2, The datalogger software shall provide trending of upto 256 datapoints at a maximum sample time of 100ms.*

4.1.4 Requirements Traceability

All software requirements must have the following traceability information recorded:

- Requirement Rationale – why is this a requirement?
- Source – customer, design, modification (ECR, etc.)
- All assumptions must be recorded and worded as such as part of the rationale

For example:
Interfacing Systems: *Requirement Number XYZ.SW.1234, The datalogger software shall trend data from controllers X and Y. Rationale: Customer Specification for accountancy and recording purposes; Source: XYZ System URS, Document ID nnnn1234, Revision/Version n.*

4.1.5 Implementation Planning

The Software Requirements Analysis should be used to provide initial implementation planning for example, complexity, loading, tools and development system selection, test harness, time/effort, etc. Implementation information should be communicated to Project Management and/or Systems Engineering to allow for integration of the software in the overall system implementation.

Implementation planning should be recorded on the Software Estimate Form (Ref. 9).

The use of any software tool should be controlled using the Software Acceptance Form (Ref. 17).

4.1.6 Version Control of Software Requirements

The Software Requirements must be version controlled as changes and amendments may be made during other phases of the software development lifecycle. As a minimum at the following software development lifecycle phases the Software Requirements must be controlled and where possible versioned:

- On completion and approval of the Software Requirements Analysis
- Post Implementation and Integration before Validation commences

If required (due to change) the software requirements can also be version controlled at:

- After Software Design has been reviewed and approved
- Post Acceptance activities

For example:
Baseline Software Requirements Matrix XYZ.SW at Version 1.0 at the completion of the Requirements Analysis Phase.
Baseline Software Requirements Matrix XYZ.SW at Version 2.0 on approval of software design.

4.1.7 Requirements Change Management

Impacted changes to the software must initiate an update or amendment to the Software Requirements. Initiation of a change must be accompanied by an Impact Analysis to ascertain

which process steps must be repeated, and if there is any impact on the overall System Design. Change Management as part of Configuration Management is described in Section 4.10.

4.1.8 Impact Analysis

An impact analysis/assessment must be undertaken to develop software architecture and system impacts. In addition any conformance requirements must be considered and appropriate tasks prescribed to form an appropriate development lifecycle (as defined in Section 3.1). Factors to consider include:

- Functional safety
- Customer qualification requirements (e.g. NQA-1, etc.)
- Complexity (low/medium/high)
- Operating System impacts
- Sizing and timing impacts
- Feasibility
- Resource loading
- Other constraints
- Software systems and components
- Software reuse
- Modification requirements (impact on existing design)
- Operational requirements
- Maintenance requirements

This impact analysis should be documented in the Software Report Form (Ref. 14) and retained as part of the development record.

4.1.9 Requirements Review

On completion of the Software Requirements Analysis the Software Requirements (either as part of a specification or matrix), Impact Analysis and any supporting information/data may be formally reviewed. The Requirements review should consider the following points:

- Does the prescribed lifecycle meet the requirements?
- Are the software requirements correct and understandable?
- Are they testable?
- Are the Requirement Numbers unique?
- Is the requirement rationale appropriate and correct?
- Does the requirement source reflect the design inputs, e.g. URS, System Design, ECR, modification, etc.
- Are all constraints documented?
- Have the safety requirements been clearly identified?
- Have any updates (changes/amendments) been managed?
- Are the Software Requirements under version control?

See Section 5.0 for qualification and independence during reviews.

4.2 Software Design

The Software Design can be optionally separated into two phases depending on complexity and prescribed activities from the Impact Analysis (see Section 4.1.8):

- Initial – design of software architecture, validation test specification and integration design.
- Detailed – design of the software to permit coding and testing.

For low/medium complexity design a single design phase can be carried out which covers the processes defined in both Initial and Detailed. In this case a Design Review must be held at the end of design activities before commencement of Implementation.

4.2.1 Initial Design Phase

This phase of the design provides the structure for the detailed design of the software that meets the software requirements. The Initial Design Phase process is:

- Software Architecture Design
- Validation Test Production
- Initial Design Review

4.2.1.1 Software Architecture Design

The Software Architecture Design phase uses the requirements to define the software architecture for the system. The software architecture design must include the following:

- Design of a top down software structure that provides logical grouping of related software items.
- Design the naming convention for the software application at appropriate levels, e.g. the main software unit, at group/sub-unit level, and at component level (component level is equivalent to a single software unit or configured item, e.g. a source file).
- Interface naming convention and structure must be defined
- All inputs and outputs to the software system should be uniquely defined using associated naming convention to the interface
- Database architecture must be similarly defined.
- Design of structures and naming convention must comply with Software Configuration Management Generic Implementation Plan (4.10)

The Software Architecture must be documented in the Software System Specification (Template Ref. 10) the following information should be included:

- Top down software/interface/database architecture in the form of textual, model or other appropriate method
- Description of the naming convention and how it applies to the required sub-structures
- Traceability to the requirements of the software item via the Software Requirements Matrix – Design (either using the matrix or by providing requirement details in the SSS)
- The rationale of how the software architecture meets the requirements including how each component level interacts
- Any safety functionality required by the software should be provisioned for in the Software Architecture. If required the safety functions must be partitioned and separated from non-safety functions. Where possible architecture should provide the facilities required during detailed design to ensure:
    - Graceful degradation
    - Watchdog or sequence monitoring
    - Visibility of logic functions
- Any coding standard, guidelines or practise should be considered and referenced in the SSS.
- Integration issues relating to architecture should be considered as part of the design documented.

4.2.1.2 Validation Test(s) Production

During the initial design phase the software validation tests must be defined and documented. The tests should be documented in the Software Test Form (Ref. 15) defining the following:

- Test Case Number – A unique number for each test case.
- Requirement Number (or identifier) – the associated requirement that the test is to validate for.
- Test Steps
- Test Approval(s) – name of test approver(s), role, signature and date (where customer requirements this may involve a customer representative)
- Author of the Validation Tests
- Test Engineer carrying out the tests
- Test equipment – harnesses, simulators, measurement equipment and any other test related equipment
- Significance of test – requirements validation and/or safety requirements validation
- Boxes for test step signoff (name, sign, date)
- Independence required for signoff

4.2.2 Initial Design Review

Once the initial design phase is complete a review of the requirements against the initial design may be required. The initial design review verifies that the proposed design meets and is traceable to the requirements. The Initial Design Review should be documented in the Software Review Form (Ref. 12) and retained as part of the development record. Attendees at the design review should be appropriately aware of the requirements to allow for approval of the initial design. Initial design reviews may be attended or require endorsement/approval by the customer.

See Section 5.0 for qualification and independence during reviews.

4.2.3 Detailed Design Phase

The Detailed Design Phase provides for the design of software that can be implemented and verified against the software requirements using the approved (see Section 4.2.2) software architecture. The detailed design must be documented sufficiently to permit coding of the software and unit/verification testing.

4.2.3.1 Software Detailed Design

The software detailed design must describe the function of each configured item and how each item interacts with other configured items. The design must be documented in detail the SSS and must comply with the initial design. The documentation must describe the functionality in a level that ensures unambiguous software coding. Consideration of the software integration and build must form part of the design.

The design and functionality of each configured item must be traceable directly to the requirements. The traceability must as a minimum be recorded in the Software Requirements Matrix, however it is recommended that the requirement number is stated in the Software System Specification. Traceability must describe which software requirements have been allocated to the functionality within each configured item.

The software detail design should be documented using methods that ensure accuracy and prevent ambiguity at the coding stage. For this purpose the detailed design must be documented using one of the following:

- Pseudocode (requires a Pseudocode dictionary to be produced)
- Basic model – flow charts, block diagrams, context, data flow, etc. (low complexity design and/or modification)
- Modelling system/language, e.g. UML, etc.
- Mathematical representation (specification required)
- Other specification languages (specification required)*

*These methods could include Formal Methods but this procedure is only for software development for systems with a maximum target integrity level of SIL2.

All interface and database designs will be similarly documented in the SSS allowing for coding without the need for additional documentation.

The design of all software, database and interfaces must consider operation and maintenance (including future software maintenance/modification).

For software providing safety functionality (or for high integrity functionality) the detailed design must include the following:

- Software, interface and database design to ensure independence of safety functions
- The SIL of the software system is the SIL of the highest safety function
- Failure of a non-safety function must not affect the safety function
- Fail safe on detection of failures
- Interface communications error checking
- System user levels to prevent the operator/user from modifying the software configuration
- Fault tolerance e.g. watchdogs, status monitoring, surveillance

4.2.3.2 Verification Test(s) Production

Tests must be produced to verify the design of all software configured items. These tests may include the following:

- Module verification
- Software sub-system verification
- Software system verification
- Architecture verification

Tests must be specified using the Software Test Form (Ref. 15) defining the information described in Section 4.2.1.2. The exception to this is the specification of Module verification as unit tests where by only the following information needs to be provided in the STF:

- Test Steps
- Test Approval(s) – name of test approver(s), role, signature and date
- Author of the Module Test(s)
- Test Engineer carrying out the tests
- Test equipment – harnesses, simulators, measurement equipment and any other test related equipment
- Boxes for test step signoff (name, sign, date) – use as tick sheet

4.2.4 Detailed Design Review

Once the Detailed Design Phase is completed a Detailed Design Review must be held reviewing all elements of the design (especially when the Initial Design Review has not been held). The aim of the design review is to check that design meets the requirements and that the tests proposed verify the design (in other words the design has followed the V-Model).

The following criteria should be considered as part of the Detailed Design review:

- Traceability from software requirements to the design and test of the software Configured Item.
- The design must be consistent with the architecture design.
- The design must be consistent within the design hierarchy.
- The design methods, documentation and standards are appropriate for the design.
- The testing must be feasible and provide the required level of verification.

The Detailed Design Review should be documented in the Software Review Form (Ref. 12) and retained as part of the development record. Actions, problems and issues with the design will be clearly identified in the SRF, and shall be tracked to resolution.

If a design review is in the format of a meeting, attendees should be appropriately aware of the requirements to allow for approval of the initial design. Design review meetings may be attended or require endorsement/approval by the customer. The attendees will agree the outcome and actions of the design review meeting . Post design review meeting the SRF should be updated to reflect the actions, outcome, comments, problems and issues and should be issued to all attendees (and any other identified stakeholders).

All Design Review actions (that affect the implementation, integration or testing of the software) must be resolved before progressing to implementation. The resolution/closure of all actions must be recorded on the SRF.

4.3 Implementation

Once the Detailed Design Review is complete and all required actions are closed implementation can begin and the software source code can be produced. The software units must be coded using the required/specified coding practices or guidelines (where these do not exist industry/manufacturer guidelines may be acceptable); these should specified in the Architecture Design phase (see section 4.2.1.1)

Any changes made during implementation must be reflected in the SSS and STF with the appropriately level of design review and endorsement.

The software source code must be reviewed before it is approved for integration, however multiple integration steps with an appropriate review may be used.

4.3.1 Code Documentation

Code documentation should be used where detailed change tracking is not used. Change tracking as part of version control and configuration management is preferable as documentary impacts on the code can be kept to a minimum. If the code must be documented the following should be undertaken:

Code should contain a plain English, readable header identifying:

- The unit, module or configured item name (from architecture/hierarchy)
- Brief module description
- Requirement number(s)
- Software developer(s)
- Revision
- Interfaces
- Version of SSS & SRTM
- Modification history (name, date, modification(s)) including SCR

Each function within the code can have an English, readable description:

  - Function name
  - Brief description
  - Modification/changes (including SCR)

- Endline comments should be avoided
- The code comments should not contain abbreviations
- Build scripts, makefiles and any other script, batch or automation file must be similarly commented and must be added to the configuration.
- Any interface specific files must follow the same level of commenting
- Database files should where possible have similar information recorded in notes or text areas for table construction; any database configuration or script files should also be commented.

- All safety functions should be clearly identified in the header and immediately preceding the coded function.

Commenting should ensure that the code is maintainable and that future modification of the can be completed with minimal risk to functionality while avoiding confusion.

4.3.2 Code review

The software code review is a review or walk through of the software source code prior to integration. The code review may be undertaken by an independent engineer or by a software tool depending on requirements and feasibility. As a minimum the code should be reviewed for function and against the Code Documentation requirements defined in Section 4.3.1.

The Code Review process is defined in WI-14_004, Source Code Review Work Instruction (Ref. 5).

If coding standards have been used in code development these should be used to review the code. The code review should be recorded on SRF; all actions/issues must be resolved before code release.

The code review should be documented using the Software Review Form (Ref. 12) and retained as formal records.

4.4 Software Integration and Unit Test

On completion of implementation the software source configured items must be combined to produce integrated functional software, as defined in the software design. In addition to integration the software units must be verified using the unit tests; the regression strategy must also be defined for modification testing.

4.4.1 Integration

The software source code must be integrated to produce functioning software. This process is made up of the following:

- Production of build scripts
- Compilation of software units
- Production of data files
- Linking build files
- Production of functioning software e.g. executables

Any architecture or software design that requires modification, as part of integration must be updated, reviewed and approved before formal software testing.

4.4.2 Release Process

Once the software has been fully integrated and executables and files for the target have been prepared the software version must be formally released. The release process generates builds for testing, acceptance and production.

The release process is defined in Work Instruction WI-14_006 (Ref. 7).

4.4.3 Unit Testing

The software source code must be unit tested as part or fully integrated functional software. Unit tests must verify the operation of each software function and stress each unit beyond the minimum and maximum expected conditions.

Unit testing must consider each module in isolation, using low level interfaces to ensure that the software functions as per the design. Unit tests must be documented using a STF, and should be retained as part of the design record.

4.4.4 Regression Strategy

During integration a regression testing strategy may be produced that allows for software retest as part of a future releases. Consideration should be given to numbers of tests required to meet reliability or functional safety requirements. The Regression Strategy should be documented and retained as a design record and should be referenced from the appropriate Verification and/or Validation Test Specification.

4.5 Software Testing

Software Testing must be carried out to verify the software function complies with the design (Verification) and that the software is validated as meeting the requirements (Validation). The verification and validation testing follows the process defined in the V-Model described in Section 3.0.

4.5.1 Verification Testing

The verification testing must be run on the release version of the software using the Test Specifications defined and approved during the design phase (see Section 4.2.3.2). Each test step/case must be signed and dated by the test engineer. The test engineer should be independent to the software engineer who coded the software release.

Repeat verification testing for minor releases may be subject to a Regression Strategy as defined in section 4.4.4.

4.5.2 Validation Testing

The validation testing must be run on the release version of the software using the Test Specifications defined and approved during the design phase (see Section 4.2.1.2). Each test step/case must be signed and dated by the test engineer. The test engineer should be independent to the software engineer who coded the software release.

Validation testing may be carried out as part of acceptance testing of the overall system. Modifications to existing system's software may be difficult to validate offline and may require use of the customer's facilities. Wherever the software validation testing takes place it must still validate the software requirements and/or any other key functionality required e.g. safety or environmental functionality.

Repeat validation testing for minor releases may be subject to a Regression Strategy as defined in section 4.4.4.

4.5.3 Test Results Review

Once the verification and validation testing have been completed, for a specific release, a review of the testing should be undertaken. The Test Results Review should review the following:

- Test Results
- Snags
- Performance/condition of test harness/equipment
- Test engineer performance
- Repeat/additional testing

For each release the testing results and minutes of the review should be recorded using the Software Review Form (Ref. 12) and retained as formal records.

4.6 Installation

Installation on the target system should follow the system design and be controlled to ensure that the system operation is not affected due to incorrect installation of the software. Installation must be considered as part of Software Design and Integration although consideration of legacy issues may be required under modification conditions. Typically the following is considered:

- Installation instruction
  - Accuracy
  - Technology changes
  - External system requirements (e.g. version control tool changes, networking, USB, etc.)
- Remote installation (installation on a customer's site)
- Prerequisites
- Versioning
- Labelling
- Backup media
- Customer installation requirements

4.7 Acceptance

Acceptance covers those activities undertaken to prove the software meets the customer requirements. These activities maybe specific system acceptance events such as:

- Factory Acceptance Test
- Customer Acceptance Test
- Site Acceptance Test

The customer usually provides acceptance testing requirements as part of the system requirements. Software acceptance testing is a subset of the overall system test and may involve specific Verification and Validation Testing to prove the requirements for each release. It is usual to carry out formal validation testing (as defined in Sections 4.2.1.2 and 4.5.2) plus any specific functional/proof tests the customer requires.

4.7.1 Test Readiness Review

Before commencement of the acceptance activities a Test Readiness Review may be held to review all previous verification and validation activities including:

- Design reviews
- Code reviews
- Unit tests
- Verification tests
- Validation tests
- Any outstanding issues such as functionality, snags, etc.

The test readiness review must agree that the system is ready to test and that all previous phases of the design, production and testing have been completed as specified in this procedure.

4.7.2 Formal Acceptance Test Specification

The software acceptance testing will form part of the overall acceptance testing, therefore the agreed acceptance tests will form part of the overall acceptance test specification.

4.8 Operation

Operation of the software is subject to normal operation and maintenance of the system and is therefore documented as part of the System Operation and Maintenance Manual. Production of the System Operation and Maintenance Manual is subject to agreement with the customer and is not mandated by this procedure.

4.9 Maintenance

Maintenance of the software is subject to normal operation and maintenance of the system and is therefore documented as part of the System Operation and Maintenance Manual. Production of the System Operation and Maintenance Manual is subject to agreement with the customer and is not mandated by this procedure.

4.9.1 Modification

Where maintenance requires that a software modification is required this will be subject to the process described in this procedure. Any software modification must be carried out under full SCM (as defined in section 4.10) and follows the software development lifecycle including establishing and agreeing requirements and impacts as described in section 4.1.

4.9.2 Other Maintenance Activities

During design, production and delivery and in cases where the customer requires post delivery the following maintenance activities will be undertaken. These tasks are outside the normal operation and maintenance and can be considered as supporting software maintenance.

4.9.2.1 Media Management

As part of normal maintenance procedures the establishment of system media and media management must be undertaken. Media Management must be planned and offered to the customer as part of software release. The Media will be as required by installation on the target system (as defined in 4.6). Release copies of media will be retained and will be appropriately stored

(as per manufacturer's guidelines). Media management will cover media regeneration in line with manufacturers stated failure and safe storage rates/durations. System specific media requirements will be recorded in the Software Specification; media management will be recorded in the System Operation and Maintenance Manual.

4.9.2.2 Anti-Virus

Ant-Virus and Malware protection must be applied throughout the software engineering lifecycle. On a system/project basis a vulnerability assessment of the development, testing and delivery platforms must be made and recorded as a formal record. The vulnerability assessment will cover access configuration and protection for the following:

- Target systems
- Design and development system
- Development/production system
- Other host systems, simulators, test systems and/or data recording/acquisition
- Networking and protection from external access (Company Network, Internet, etc.)

The primary objective of vulnerability assessment should be to reduce (or preferably remove) the risk of virus and/or malware infection.

In addition to the above assessment, all media received from or being supplied to a third party must be virus scanned on an agreed/appropriate separate computer system.

Anti-virus software and malware tools must be appropriate to the hardware and media platform/configuration. Where a hardware platform is identified as not susceptible to virus attack this should be justified and recorded in the System Specification and the Operation and Maintenance Manual.

4.9.2.3 Disaster recovery

Disaster recovery should form part of the system and software design considerations, with the impact of recovery of hardware and software being recorded in the appropriate specification. During integration, test and acceptance a full backup of the installed target should be maintained so that software recovery is accurate to the configuration baseline.

Post delivery, and if the customer requires, disaster recovery planning for the target system can be undertaken. This planning requires consideration of the following:

- Target system configuration
- Target system environment
- Local procedures and controls
- Media
- Recovery instructions
- Human resource requirements
- Tools
- Access
- Customer recovery requirements (availability, recovery time)

The disaster recovery plan should record the above information and develop a solution to suit the system and customer requirements. This plan should be agreed with the customer and retained as a formal record.

4.9.2.4 Software Obsolescence

Software will be termed obsolete when it no longer operates on a supported operating system. Similarly an operating system is considered obsolete when it is no longer operates on available hardware. Future obsolescence, hardware and software upgrade paths and support of the software application and operating system will be considered as part of the software design.

Post delivery Software Obsolescence forms part of system asset care and is subject to agreement with the customer.

4.10 Software Configuration Management

Software Configuration Management (SCM) will be used throughout the software development lifecycle and will be preserved as long as the company provides software maintenance of the system(s). SCM will be achieved using the following processes:

- SCM Implementation, comprising generic SCM and specific product specific SCM
- Identification of SCM Configurations
- Configuration Controls
- Status Control
- Release Management

SCM will provide identification, definition, and baseline control of all Software Configurations and their constituent Configured Items to ensure accurate, consistent, controlled release of software to a target system.

4.10.1 SCM Implementation

The SCM Implementation complies fully with the Company Configuration Management Plan (Ref. 22). This plan contains the company generic Configuration Management methodology including:

- Overview of Configuration Management
- Organisation and Responsibility for Configuration Management
- Generic naming convention
- Detailed Change Control Process
- Planning of Change Control Board (CCB)
- Implementation

The following sections define the detailed Generic SCM process that must be followed for all systems (where possible/practical within the constraints of legacy software systems).

4.10.1.1 Responsibility for SCM

The following table (Table 2) defines the responsibilities for SCM as part of the Company Configuration Management Process.

Role	Responsibility
Engineering Manager	Responsible for Company Configuration Management, and therefore the provision and adherence to Software Configuration Management. Responsible for production of Company Generic CMP. Responsible for Change Control Board
Quality Manager	Assurance activities to verify that SCM is implemented and that it follows the Generic CMP and the SCM activities defined in this document. Assurance of product specific SCM throughout development and product lifecycle.
Software Team Leader	Implementation and local control of SCM Approval of identification, status, change and baselines Responsible for release Responsible for maintaining baseline control Responsible for product specific CMPs Responsible for the use and control of SCM Tools
Independent Engineer	Verify identification, status, change, as part of baseline and release controls Verify product specific CMPs Carry out Verification and Validation Testing.
Software Engineer	Design and maintenance activities to ensure SCM is maintained

Role	Responsibility
	throughout SW Lifecycle and continues into maintenance Production and maintenance of product specific CMPs

Table 2: Responsibilities for SCM

4.10.1.2 SCM Tools

SCM can be controlled using an appropriate tool (or tools) that provide the following:

- Version Control down to Configured Item Level
- Status Control as required by the software development lifecycle (development, test, accepted, production, etc.)
- Lock System appropriate for the development process (branch control, root-tip, etc.)
- Baseline control
- Change Management with appropriate work-flow (change status)
- Work-flow hold points and integration
- Auditing
- Access control
- Appropriate Team Work
- Defect tracking (via change)
- Reporting

The proposed tool(s) must have been approved for use via the following process:

- Parallel trials using manual or alternate approved tool
- Trial must be whole lifecycle
- Trial must include:
    - Architecture, naming convention and structure
    - Status control
    - Baseline control
    - Hold points
    - Work-flow
    - Team Work
    - Defect reports back to source
- The trial must be documented including tool-generated reporting and must be audited by Engineering Manager and Quality Manager

4.10.1.3 SCM Structure

The SCM structure should comply with the following:

- Each System has individual configurations where possible independent of other configurations
- Shared components will be managed and identified (significance of shared components must be documented in the product specific CMP)
- Each system's sub-components will be identified by type e.g. *XYZ.SW, XYZ.IF, XYZ.DB*
- Where possible sub-components will be grouped into common groups
- Plain text will be used for all description information (in addition to abbreviated naming convention)
- Structure must not affect accurate baseline and release

4.10.1.4 Naming convention

Each system will be uniquely identified. All subsystems and components will be identified as being part of a system, e.g. *XYZ.GAMMA, XYZ.MOTOR-CONTROL*, etc.

Naming convention must provide for all software, interface, database, data files, and documentation configured items.

Product specific SCM identification is dependant on Software Architecture Design (See Section 4.2.1.1) and must be documented in a Product Specific Configuration Management Plan.

4.10.1.5 Version Control

Version Control will be applied to each configured item under SCM. The Version Control process will provide the following:

- System and sub-system versioning of configured items by capturing the revision of Source Files, Database, Data Files and Documents
- Baseline Version (see section 4.10.1.8)
- Version and revision status incorporating status (see section 4.10.1.10)
- Version description information including:
    - Plain text description of Version Description
    - Any modification or snags incorporated in the version (including formal modification and snag numbers/identifiers)

Version control must be applied to all software configured items and must be utilised at appropriate stages of the software lifecycle. If required by the customer plans, documents and reports relating to the release and delivery of each version must also be included.

4.10.1.6 Change Control Process

The Change Control Process provides a process by which changes are managed, planned and approved prior to implementation. Change Control provides the following:

- Unique identification of each change
- Traceability from change to defect or requirement
- Change planning
- Impact Analysis and configured item assignment
- Provides approval hold point(s)
- Identifies reviewer, approval and developer
- Provides reporting for CCB, baseline statement and audit
- Change control can cover software, interface, database, data files, and documentation.

Change Control is described in detail in the Change Control Work Instruction (Ref. 23).

4.10.1.7 Change Control Board

A Change Control Board (CCB) must be convened to review version and baseline changes prior to inclusion in SCM. The CCB will review the following:

- Snags and modifications for inclusion in this Baseline
- Impacts on Configured Items
- Which elements of the software lifecycle must be utilised on the snag(s) and/or modification(s)
- Documentation changes
- Baseline impacts
- Release

The CCB will be recorded on the Software Review Form (Ref. 12).

4.10.1.8 Baseline Control

Baseline Control must be applied to each configuration under SCM. Baseline Control will be applied at the following development lifecycle stages:

- Prototype (if required)
- Preliminary Design (if required)
- Detailed Design
- Pre-testing (implementation)
- Pre-acceptance
- Post-acceptance
- As required by CCB approved change

Each baseline must be documented (Baseline Statement) using the Software Report Form (Ref. 14) identifying the following:

- Each module, configured item and sub-system
- Structure
- Overall Baseline Version, status, number and rationale
- Configured item version or revision, status, approval
- Tools usage and approval
- Status of each configured item (if any status other than Production – see section 4.10.1.5)
- Version of underpinning documentation, Requirements (URS, SRTM), Design, Test, etc.
- Baseline approval
- Summary of previous baseline (version, number, rationale, description, etc.)

4.10.1.9 Release

Software Configured Items must only be released on approval/endorsement of the CCB and when all appropriate stages of the Software Development Lifecycle have been met. Release must be tightly controlled and assurance must be sought that all configured items are in Test or Production status (or the customer has given permission for any formal deviation). A full audit trail of the requirements, specification, software development, integration and unit test, SCM (including version control) must be available for each release.

4.10.1.10 Status

The following statuses will be applied to each configured item under SCM:

- Development – The software configured item is in production as a result of a new design, snag or modification.
- Test – The software configured item has completed production and has been unit tested. This status should be applied when the configured item has been released formal testing.
- Production – The software configured item has been tested and is available for release to the customer or product (the software is defect free).

4.11 Audit

This adherence to this procedure must be verified by audit and inspection of the design records and documentation. All aspects of the software lifecycle should be considered for audit on a periodic basis under the normal company audit controls. The audit process is described in the System Procedure for Auditing SP_1.0 (Ref. 20)

4.12 Functional Safety Assessment

A Functional Safety Assessment is specific to systems that must conform with the requirements of IEC61508 (Ref 3). The audit can be carried out at any point during the software development lifecycle. The audit is undertaken

The Functional Safety Assessment must be carried out by an independent competent person as defined by the competency/skills register (Ref. 24). For software development up to and including SIL 2 target systems an independent qualified person within the organisation shall be designated at System Design time. Qualification of a Functional safety assessor is a person who has attended an independent functional safety course and who has carried out at least one witnessed Functional Safety Assessment.

The Functional Safety Assessment is specified in the Functional Safety Assessment Instruction (Ref. 6). Form LIS-14_004 (Ref. 12) is used to record the Audit. Functional Safety Assessment Reports are formal records and must be retained.

5.0 Independence

The following table (Table 3) defines the independence required for specific tasks and reviews in the software engineering lifecycle.

Requirement	Tasks	Independence	Qualification/Role
NQA-1	Requirements Reviewer Design Reviewer	Must be independent (i.e. not involved in specifying) of all design approaches, design considerations, and design inputs.	Senior Engineer, Team Leader or Supervisor in related discipline
IEC61508 SIL 1	Code Reviewer	Independent of the software design	Team Leader or higher
IEC61508 SIL 2	Test results Reviewer	Independent of the system and software design	Departmental head or higher (1 level above Team Leader/Supervisor)

Table 3: Independence

For Functional Safety Assessment reviewer independence see section 4.12.

6.0 References

External Standards

Reference	Title	Version/Date
1.	BS EN ISO 9001:2008, Quality Management System Requirements	2008
2.	ISO/IEC 12207, Systems and software engineering – Software life cycle processes	2008
3.	IEC61508:2010	Ed. 2, 2010
4.	ASME NQA-1-2000, ASME NQA-1-2008	2000, 2008

LIS QMS

Reference	Title	Version/Date
	SP-14 WIs	
5.	WI-14_004, Source Code Review Work Instruction	Iss 2, 26.08.2011
6.	WI-14_005, Functional Safety Assessment	Iss 1, 26.08.2011
7.	WI-14_006, Software Release Procedure	Iss 1, 26.08.2011
8.	WI-14_007, Software Acceptance Procedure	Iss 1, 26.08.2011
	SP-14 Forms & Checklists	
9.	LIS-14_001, Software Estimate Form	Issue 1
10.	LIS-14_002, Software System Specification Template	Issue 1
11.	LIS-14_003, Software Requirements Traceability Matrix	Issue 1
12.	LIS-14_004, Software Review Form	Issue 1
13.	LIS-14_005, Software Code Review Form	Issue 1
14.	LIS-14_006, Software Report Form	Issue 1
15.	LIS-14_007, Software Test Form	Issue 1
16.	LIS-14_008, Software Release Sheet	Issue 1
17.	LIS-14_009, Software Acceptance Form	Issue 1
18.	LIS-14_101, Software Specification Checklist	Issue 1
19.	LIS-14_102, Code Review Checklist	Issue 1
	Related LIS QMS Procedures, WIs & Forms	
20.	SP1.0, System Procedure for Auditing	Iss. 6, 29.03.2011
21.	SP6.0, System Procedure for Resource Management	Iss 4, 30.03.2011
22.	Configuration Management Plan	Draft
23.	WI-10_006, Change Control Work Instruction	Iss 7, 12.10.2005
24.	LIS217, LIS Organization Chart	Iss 12 02.12.2010
25.	LIS029, SRT	Iss 12 08.02.2011
26.	LIS045, ECR	Iss 16