



Embedded Test Automation Using TestFrame

1 Introduction

Test automation is as important for embedded software systems as for PC-based software. Benefits over manual testing include faster execution, better repeatability and the all-important human nature element: if it's easy to do, it's more likely to be done.

However, test automation often involves expensive tools and long learning curves. The approach outlined here has been used on real projects and is based on TestFrame, originally developed by Hans Buwalda (<http://www.happytester.com>) using the *action words* principle, using widely-available spreadsheet software.

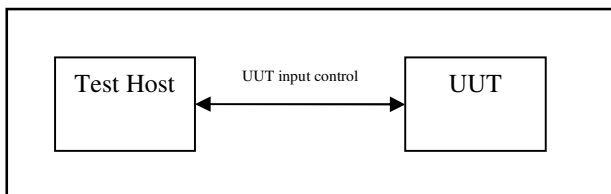
1.1 Main Features of this TestFrame Implementation

- Test scripts are written in spreadsheet format.
- The building blocks of the tests are formed from a vocabulary of action words.
- The actions words hide the details of test execution from the test engineer, allowing full concentration on testing rather than communication protocols or proprietary script syntax.
- Action words are interpreted by a *navigation script*, which translates them into communication port commands for controlling the embedded Unit Under Test (UUT).
- Test development and navigation script development can be done independently by appropriately-skilled engineers.
- Manual tests, automatic tests or a mixture of both can be scripted using TestFrame.
- Results are collected and archived automatically.

2 Test Automation Scenarios

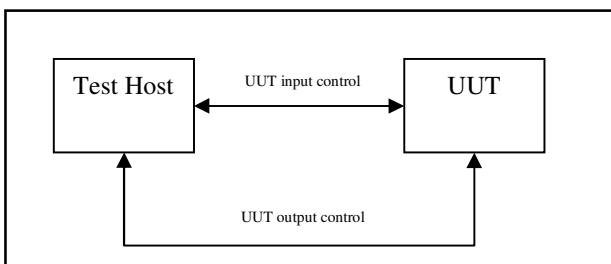
2.1 Scenario 1: Command Only

The Test Host uses a serial port to exercise an embedded UUT and compare responses with those defined by action words.



2.2 Scenario 2: Command and Log

The Test Host uses a serial port to exercise the UUT and compare responses, and uses another serial port to simulate and/or log a UUT controlled-device protocol.

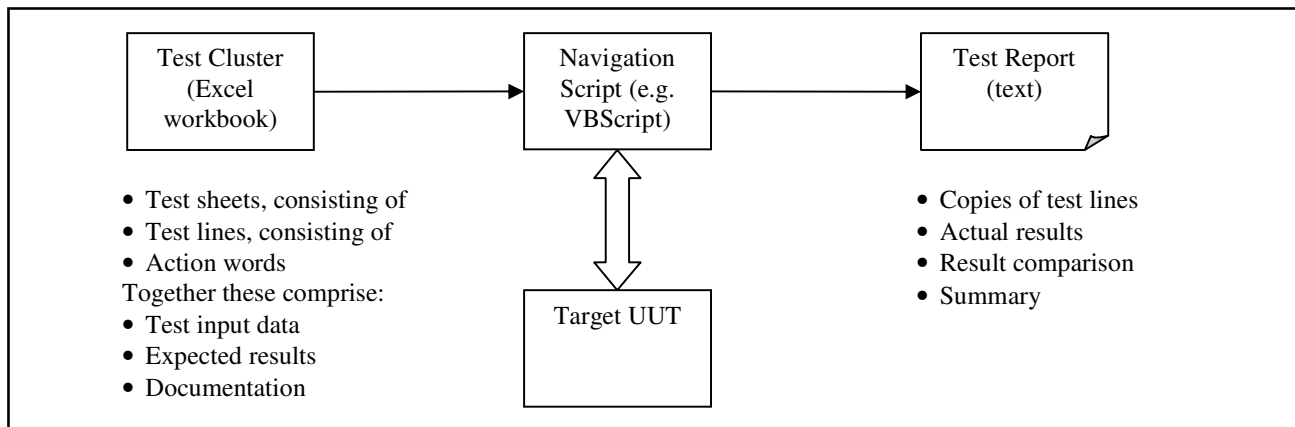




Embedded Test Automation Using TestFrame

3 Elements of TestFrame

Test conditions are packaged in Test Cluster workbooks and kept separate from the mechanics of a Navigation Script that communicates with the UUT. The power and convenience of spreadsheets can be used to help generate test data.

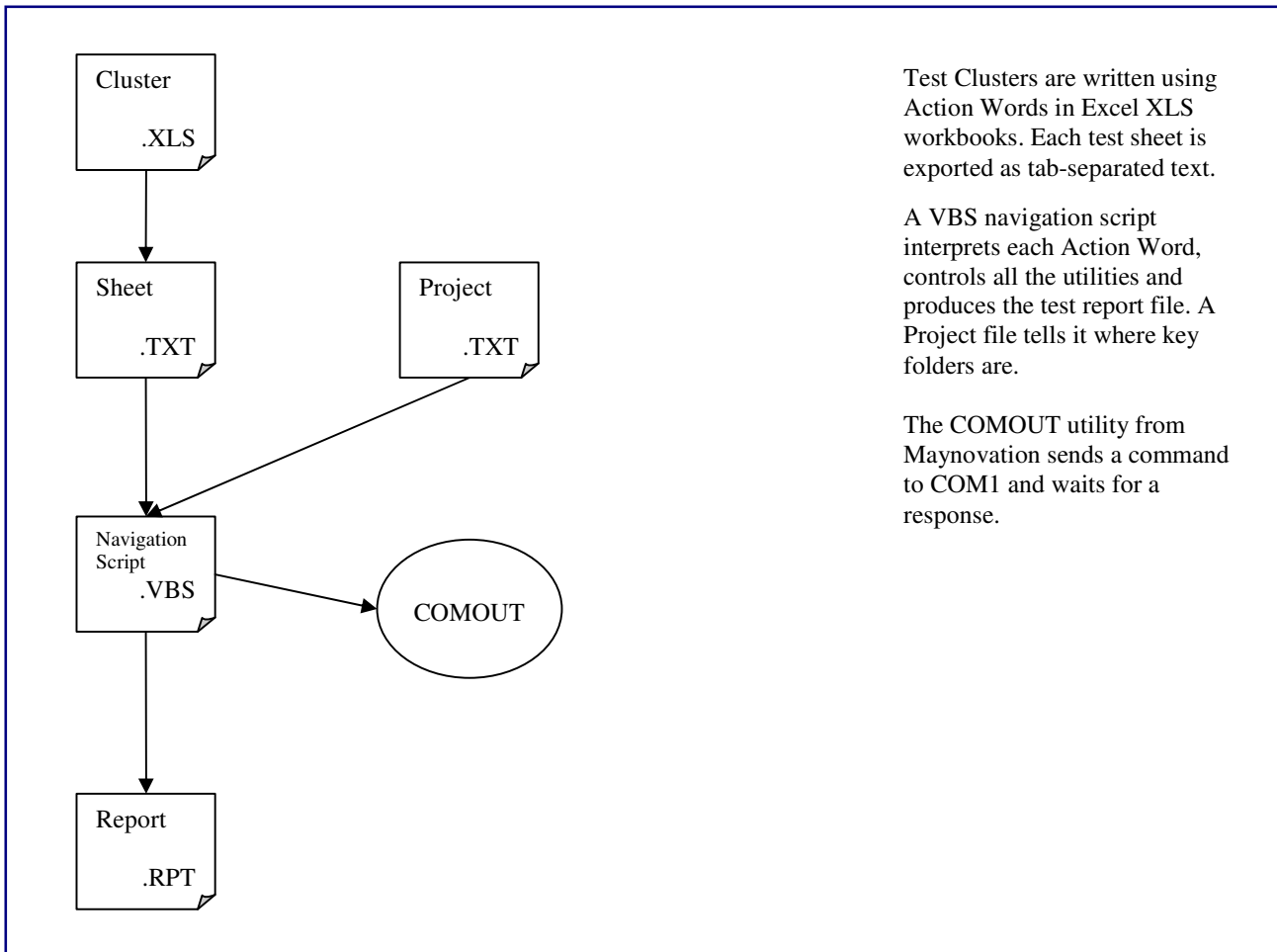




Embedded Test Automation Using TestFrame

4 Test System Files and Utilities

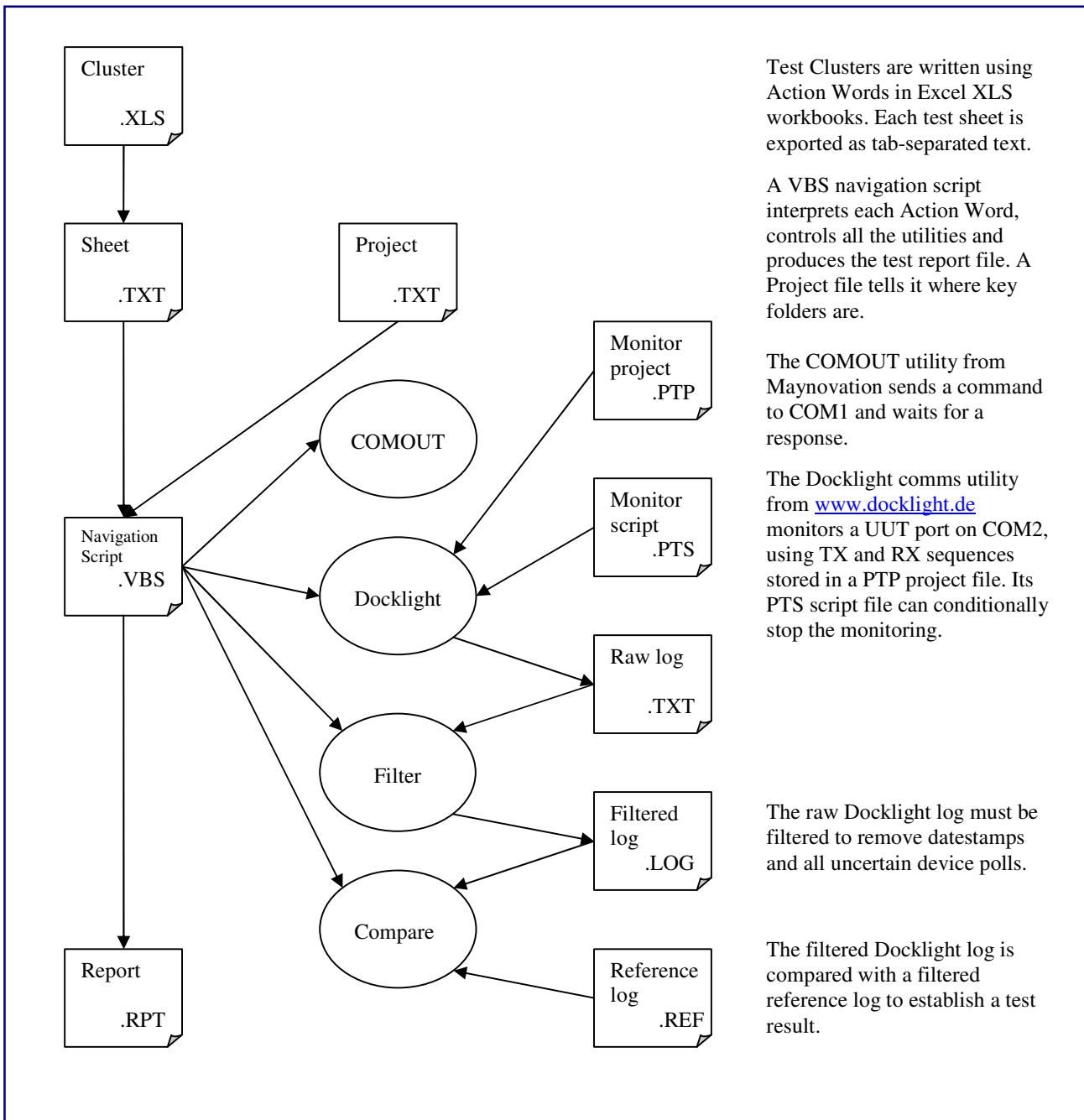
4.1 Scenario 1: Command Only





Embedded Test Automation Using TestFrame

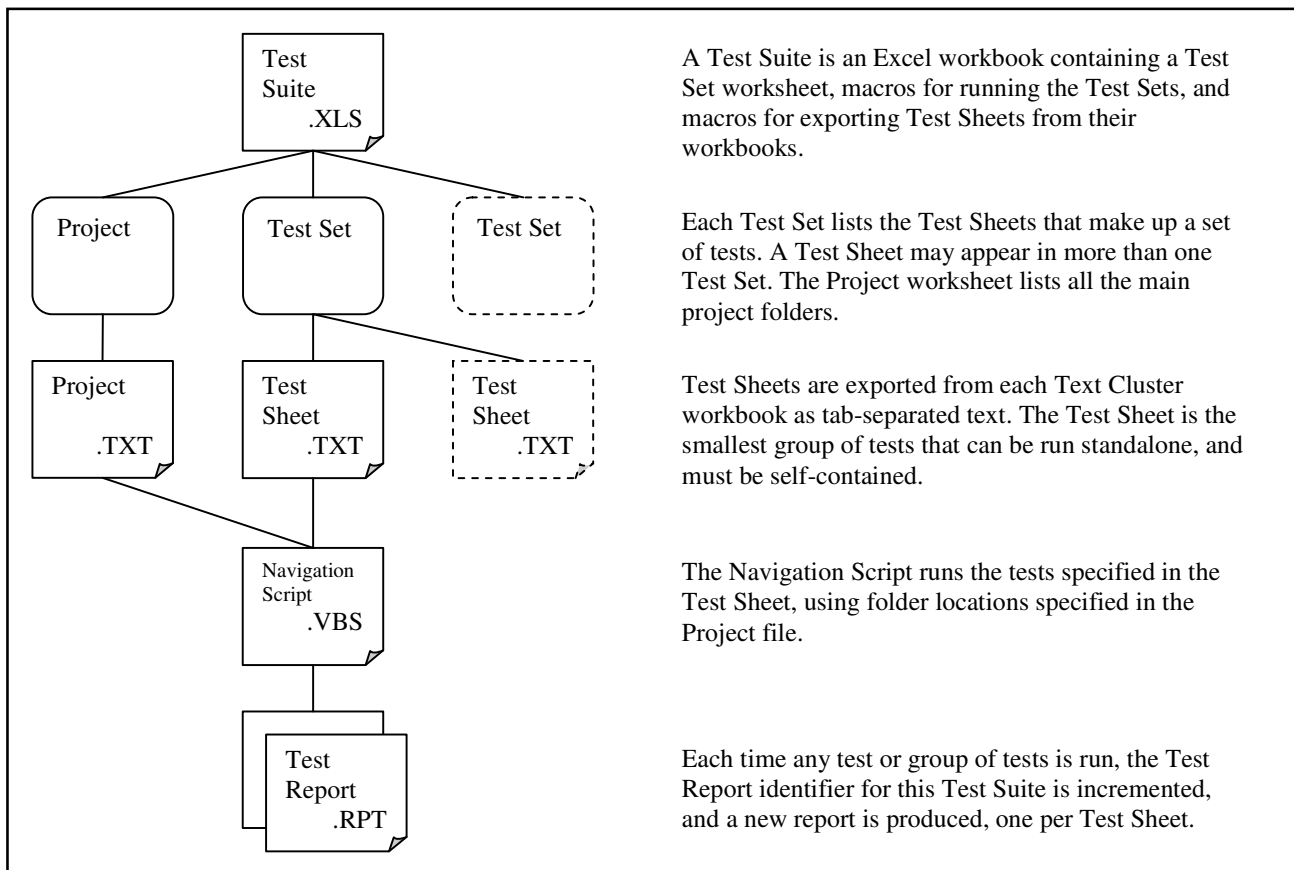
4.2 Scenario 2: Command and Log





Embedded Test Automation Using TestFrame

5 Test Organisation



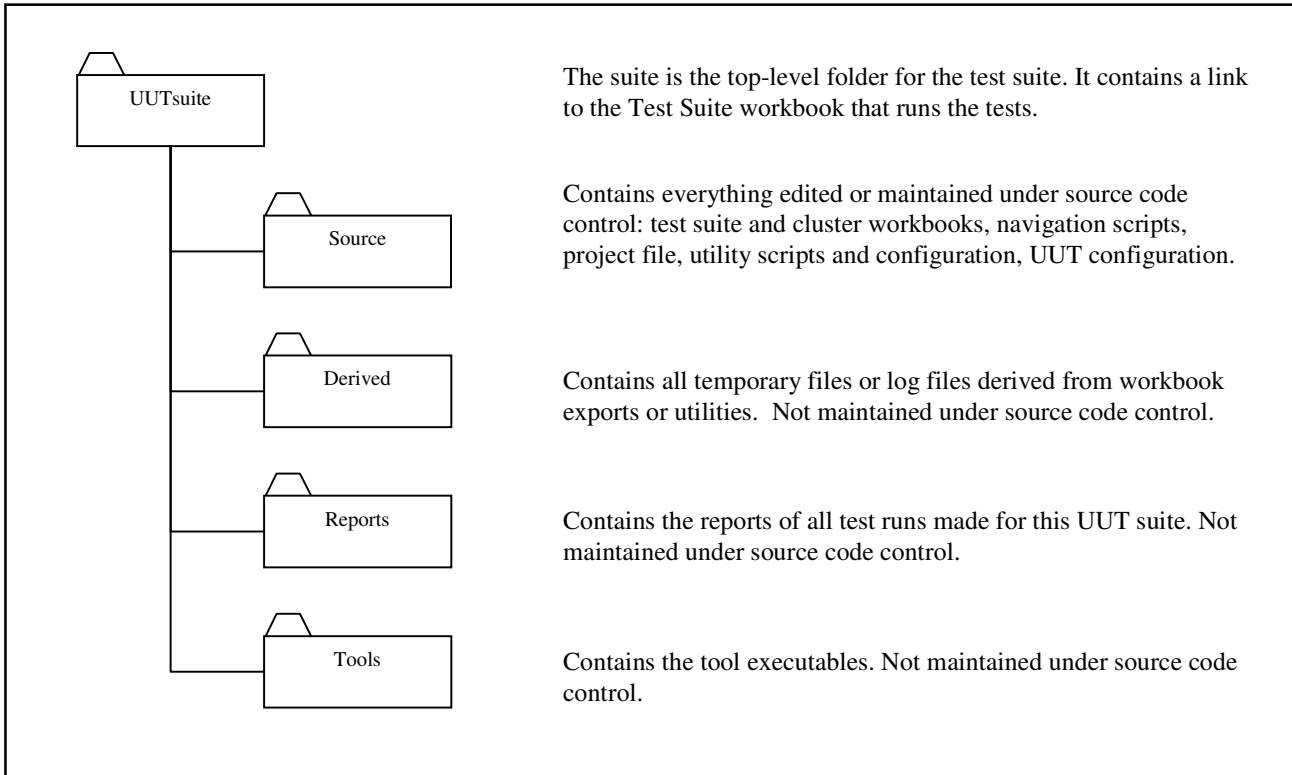
6 File Names

File Type	File Name Example	File Name Components
Test Suite	Tsu-XYZ.xls	Sorting prefix - UUT identifier (for example, the XYZ product)
Test Cluster	Tcl-XYZ-Breadth.xls	Sorting prefix - UUT identifier - cluster identifier
Test Sheet	Tsh-XYZ-Breadth-Limits.txt	Sorting prefix - UUT identifier - cluster identifier - sheet identifier, derived from Cluster XLS tab name
Test Report	Tre-XYZ-Breadth-Limits-0001.rpt	Sorting prefix - UUT identifier - cluster identifier - sheet identifier - UUT test run number
Project	Nav-XYZ-Project.txt	Sorting prefix - UUT identifier - project identifier
Navigation Script	Nav-XYZ.vbs	Sorting prefix - UUT identifier
Utility Config	Uco-XYZ-ProtocolA.cfg	Sorting prefix - UUT identifier - protocol identifier
Raw Utility Log	Ulo-XYZ-Breadth-Limits.txt	Sorting prefix - UUT identifier - cluster identifier - sheet identifier
Filtered Utility Log	Ulo-XYZ-Breadth-Limits.log	Sorting prefix - UUT identifier - cluster identifier - sheet identifier
UUT Config	Uut-XYZ-Breadth.v3	Sorting prefix - UUT identifier - cluster or sheet identifier



Embedded Test Automation Using TestFrame

7 Test File Locations



8 Test Process

Phase	Activity (some are <i>optional</i>)	Output
Test System Design (done once for UUT)	Write Navigation Script	VBS file
	Write Test Suite spreadsheet	XLS file
Test Design (done often)	Write Test Cluster spreadsheets	XLS files
	<i>Write UUT Configurations</i>	CFG files
	<i>Write Docklight project and script for each UUT protocol to be monitored</i>	PTP and PTS files
Reference Run	Run each Test Sheet for first time and review Test Report. Save reference file(s)	RPT and REF files
Regression Run	Run Test Suite in full, comparing against reference files	RPT files
Retest Run	Run selected Test Sheets only	RPT files



Embedded Test Automation Using TestFrame

9 Examples

9.1 Test Cluster (Part)

The following extract shows a mixture of action words that prompt the test operator and issue remote control commands to the UUT. The UUT response is checked against the expected values.

Note the use of Excel colour and formatting to aid readability.

	A	B	C	D	E	F	G
1	test sheet	CONFIGURATON EXPANSION LIMITS					
2	identifier	CFGE					
3	version	1.0					
4	author	Brian Mayhew					
5							
6	test case	TC-CFGE-1	SFS-215	Factory Default Configuration For			
7		<i>instruction</i>					
8	prompt	Power down both UUTs, and connect the expansion cable.					
9	prompt	Fit four Quad cards.					
10	prompt	Connect the alarm panel to the Slave UUT.					
11	prompt	Switch on alarm panel switches 1 and 4, and switch off the rest.					
12	prompt	Power up both UUTs.					
13	prompt	Using VisiPC, select a new Integrated configuration using File-New, and transf					
14		<i>port</i>	<i>preset</i>	<i>speed</i>	<i>response</i>		
15	recall preset	2	100	100	ER		
16	recall preset	2	99	100	OK		
17		<i>port</i>	<i>seq</i>	<i>response</i>			
18	recall video seq	2	9	ER			
19	recall video seq	2	8	OK			
20		<i>port</i>	<i>view</i>	<i>response</i>			
21	select view	2	9	ER			
22	select view	2	8	OK			
23		<i>port</i>	<i>zone</i>	<i>response</i>			
24	select zone	2	9	ER			
25	select zone	2	8	OK			
26		<i>port</i>	<i>user</i>	<i>response</i>			
27	log on new user	2	65	ER			
28	log on new user	2	64	OK			



Embedded Test Automation Using TestFrame

9.2 Navigation Script (Part)

This VBScript fragment interprets the action words “recall preset” and “recall video seq”.

```
'-----  
'Args(2)port, (3)preset, (4)speed, optional (5)response  
Case "recall preset":  
  If ArgTotal < 4 Then  
    Result = fnLogError()  
  Else  
    If Args(4) > 0 Then  
      sCommand = "@16," & Args(3) & Args(4)  
    Else  
      sCommand = "@8," & Args(3)  
    End If  
    sExpecting = fnResponse(5)  
    Result = fnSerialCommandCompare(Args(2), sCommand, sExpecting)  
  End If  
'-----  
'Args(2)port, (3)seq, optional (4)response  
Case "recall video seq":  
  If ArgTotal < 3 Then  
    Result = fnLogError()  
  Else  
    sCommand = "@14," & Args(3)  
    sExpecting = fnResponse(4)  
    Result = fnSerialCommandCompare(Args(2), sCommand, sExpecting)  
  End If
```