

## NRC Publications Archive Archives des publications du CNRC

### Qualisys Track Manager: User Manual Senior, D.

For the publisher's version, please access the DOI link below./ Pour consulter la version de l'éditeur, utilisez le lien DOI ci-dessous.

#### **Publisher's version / Version de l'éditeur:**

<https://doi.org/10.4224/8896115>

*Laboratory Memorandum; no. LM-2004-34, 2004*

#### **NRC Publications Archive Record / Notice des Archives des publications du CNRC :**

<https://nrc-publications.canada.ca/eng/view/object/?id=de61d2e8-121c-49f7-868e-2ec8ac7175ba>

<https://publications-cnrc.canada.ca/fra/voir/objet/?id=de61d2e8-121c-49f7-868e-2ec8ac7175ba>

Access and use of this website and the material on it are subject to the Terms and Conditions set forth at

<https://nrc-publications.canada.ca/eng/copyright>

READ THESE TERMS AND CONDITIONS CAREFULLY BEFORE USING THIS WEBSITE.

L'accès à ce site Web et l'utilisation de son contenu sont assujettis aux conditions présentées dans le site

<https://publications-cnrc.canada.ca/fra/droits>

LISEZ CES CONDITIONS ATTENTIVEMENT AVANT D'UTILISER CE SITE WEB.

**Questions?** Contact the NRC Publications Archive team at [PublicationsArchive-ArchivesPublications@nrc-cnrc.gc.ca](mailto:PublicationsArchive-ArchivesPublications@nrc-cnrc.gc.ca). If you wish to email the authors directly, please see the first page of the publication for their contact information.

**Vous avez des questions?** Nous pouvons vous aider. Pour communiquer directement avec un auteur, consultez la première page de la revue dans laquelle son article a été publié afin de trouver ses coordonnées. Si vous n'arrivez pas à les repérer, communiquez avec nous à [PublicationsArchive-ArchivesPublications@nrc-cnrc.gc.ca](mailto:PublicationsArchive-ArchivesPublications@nrc-cnrc.gc.ca).



National Research  
Council Canada

Conseil national  
de recherches Canada

Institute for  
Ocean Technology

Institut des  
technologies océaniques

---

## Laboratory Memorandum

LM-2004-34

---

# Qualisys Track Manager: User Manual

D. Senior

December 2004



## DOCUMENTATION PAGE

<b>REPORT NUMBER</b>	<b>NRC REPORT NUMBER</b>	<b>DATE</b>	
LM-2004-34		17 Dec. 04	
<b>REPORT SECURITY CLASSIFICATION</b>		<b>DISTRIBUTION</b>	
Unclassified		Unlimited	
<b>TITLE</b>			
<b>Qualisys Track Manager User Manual</b>			
<b>AUTHOR(S)</b>			
Dena Senior			
<b>CORPORATE AUTHOR(S)/PERFORMING AGENCY(S)</b>			
<b>PUBLICATION</b>			
<b>SPONSORING AGENCY(S)</b>			
Institute for Ocean Technology, National Research Council			
<b>IMD PROJECT NUMBER</b>		<b>NRC FILE NUMBER</b>	
421018			
<b>KEY WORDS</b>		<b>PAGES</b>	<b>FIGS.</b>
Linearization, Calibration, QTM		15+	6
		App	<b>TABLES</b>
			App
<b>SUMMARY</b>			
<p>QTM software is a Windows based application, which takes information provided by ProReflex cameras and generates 2D, 3D, and/or 6 degree of freedom (DOF) data. If 3D or 6DOF data is obtained it can be viewed through a 3D viewing window. At least two cameras must be used in order to obtain 3D or 6DOF data. ProReflex cameras emit IR flashes of light, which are reflected by retro-reflective markers back to the cameras. The cameras recognize markers and relay information to QTM where it is processed and output appropriately.</p> <p>This report is intended as a guide through linearization, calibration, and capturing measurement processes. Error analysis concerning linearization and calibration are discussed within the report, while the set-up procedures are included in appendices. Also, the report describes QTMDAC, which is a modified version of QualiDAC. QTMDAC communicates with QTM and generates an analog output.</p>			
<b>ADDRESS</b>			
National Research Council Institute for Ocean Technology Arctic Avenue, P. O. Box 12093 St. John's, NL A1B 3T5 Tel.: (709) 772-5185, Fax: (709) 772-2462			



National Research Council Canada  
Conseil national de recherches  
Canada

Institute for Ocean  
Technology  
Institut des technologies  
océaniques

## **QUALISYS TRACK MANAGER USER MANUAL**

LM-2004-34

Dena Senior

December 2004

## TABLE OF CONTENTS

<b>1.0 INTRODUCTION .....</b>	<b>1</b>
<b>2.0 LINEARIZATION .....</b>	<b>2</b>
<b>2.1 Sensitivity Analysis .....</b>	<b>2</b>
<b>3.0 CALIBRATION .....</b>	<b>5</b>
<b>3.1 Calibration Methods .....</b>	<b>5</b>
<b>3.2 Error Messages .....</b>	<b>6</b>
<b>3.3 Test Results.....</b>	<b>7</b>
<b>4.0 CAPTURING MEASUREMENTS .....</b>	<b>9</b>
<b>4.1 2D and 3D Motion Capture .....</b>	<b>9</b>
<b>4.2 6DOF Motion Capture .....</b>	<b>11</b>
<b>4.2.1 Capturing 6DOF data .....</b>	<b>11</b>
<b>4.2.2 QTMDAC.....</b>	<b>12</b>
<b>5.0 CONCLUSIONS .....</b>	<b>14</b>
<b>5.1 Recommendations.....</b>	<b>14</b>
<b>6.0 REFERENCES .....</b>	<b>15</b>
<b>Appendix A: QTracREADME</b>	
<b>Appendix B: CalibrationREADME</b>	
<b>Appendix C: Wand Calibration Test Results</b>	
<b>Appendix D: QTMDAC Source Code</b>	

## 1.0 INTRODUCTION

An optical tracking system, known as Qualisys, is currently in use at IOT's testing facilities. It consists of MacReflex cameras, and two programs QualiDAC (Windows based software) and Extended Volume Measurement (DOS based software). This system has certain restrictions; therefore, a new software package known as Qualisys Track Manager (QTM) will replace the current EVM software and ProReflex cameras will replace the MacReflex cameras.

QTM software is a Windows based application, which takes information provided by the camera(s) and generates 2D, 3D, and/or 6 degree of freedom (DOF) data. If 3D or 6DOF data is obtained it can be viewed through a 3D viewing window. At least two cameras must be used in order to obtain 3D or 6DOF data.

Both ProReflex and MacReflex cameras emit IR flashes of light, which are reflected by retro-reflective markers back to the cameras. The cameras recognize markers and relay information to QTM where it is processed and output appropriately.

QualiDAC software has been modified in order to accommodate the new QTM software. The ability for QualiDAC to open, close, and communicate over a TCP/IP connection was added. This required use of the QTMClient DLL. The new revised QualiDAC software is referred to as QTMDAC.

## **2.0 LINEARIZATION**

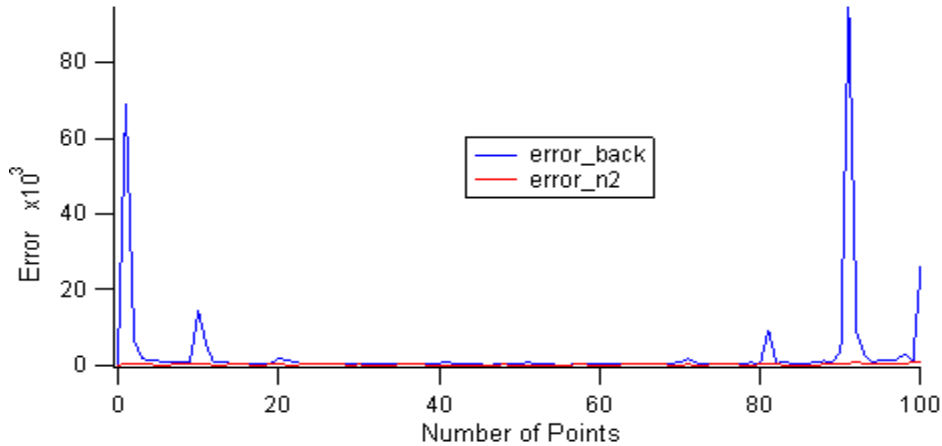
Linearization of each camera lens is a delicate and crucial procedure. This procedure corrects distortion present in each camera lens. If not corrected, distortion in the lens will cause an inaccurate calibration of the camera and furthermore measurements taken by the camera will be incorrect. Therefore, linearization of the lens must be conducted with high precision and accuracy.

In order to linearize camera(s) a program known as QTrac is used along with an 8x10 grid of markers. Once setup to view the grid properly, the camera performs calculations, which reduce lens distortion. The complete linearization procedure is available in Appendix A.

### **2.1 Sensitivity Analysis**

One of the sensitivity tests conducted consisted of moving a camera in different directions off center, then linearizing the camera. The camera was moved either upwards by 10cm, to the right by 10cm, or away from the grid by 10cm.

Moving the camera either to the right or upwards by 10cm caused a marginal amount of discrepancy from the essentially ideal linearization position. This was noticeable around the edges of the lens. However, when the camera was moved away from the grid by 10cm, the error increased drastically toward the edges of the lens. See figure 1.



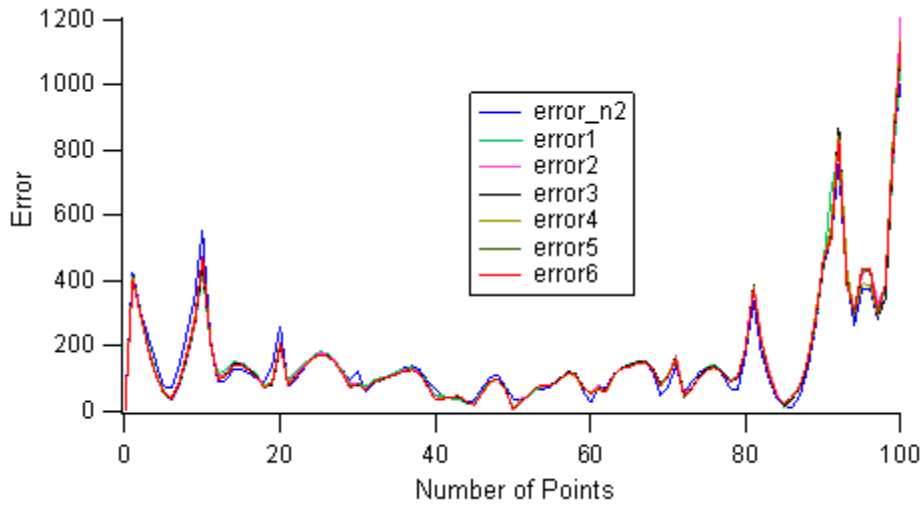
**Figure 1**

**error\_n2:** Linearization error produced from the ideal position for camera 1459.

**error\_back:** Linearization error produced from moving camera 1459 away from the grid by 10cm.

Points 0 to 15 and 80 to 100 contain errors calculated from edges of the lens. Drastic peaks demonstrate the necessity for accurate camera positioning when performing linearization.

The other group of sensitivity tests conducted involved varying aperture and focus settings of the cameras. Changing aperture and focus settings proved to have no effect on the linearization process. See figure 2.



**Figure 2**

**error\_n2:** Linearization error produced from the ideal position for camera 1459.

**error1, error2, error3, error4, error5, and error6:** Linearization errors produced from varying aperture and focus settings of camera 1459.

### **3.0 CALIBRATION**

In order to track and produce 3D data, at least two cameras must be used and they must be calibrated. Calibration when producing 2D data is not necessary.

Calibration of cameras takes place after they have been linearized and setup in the appropriate area. If the cameras are moved to another location or they have been adjusted in anyway a calibration should take place. Generally, before a measurement is captured cameras are calibrated. Locations and positions of the cameras are obtained through this process. Once the coordinate system and positions of the cameras are established, an accurate and confident measurement can be captured.

#### **3.1 Calibration Methods**

There are three methods of calibration:

1. Wand calibration
2. Frame calibration
3. Fixed camera calibration

Each method is reliable and accurate when performed properly. Calibration errors can occur if measurements of the wand length, frames, or locations of markers and cameras are not accurate. Accuracy of each measurement is tremendously important. For a more detailed discussion of the calibration process see Appendix B.

## 3.2 Error Messages

Error messages can occur during the calibration process. These messages include:

- General calibration failure
- Couldn't find the fourth marker of the L-frame
- Couldn't find the three markers in line on the L-frame
- One or more of the markers on the L-frame were unstable

Typically when doing a frame or fixed camera calibration, if a "General calibration failure" occurs, check the calibration settings. If the settings seem to be correct, make sure the marker measurements and camera locations (fixed camera) are exact.

The wand calibration method is the most prone to errors. Generally, to avoid error messages:

1. Use exact wand length and L-frame measurements.
2. Try to move the wand at an appropriate speed. Moving it too quickly will generate "One or more of the markers on the L-frame were unstable"; however, moving it too slowly may cause either one or more of the markers on the L-frame to go undetected by the cameras, or the calibration may be successful with a very low number of points captured. A practice calibration may help in determining how quickly to move the wand.
3. Make sure that complete measurement volume is visible to at least two cameras and that the wand is moved throughout the entire volume.
4. Check to make sure proper calibrations settings are chosen. Particularly, if using advanced reference object settings ensure it is checked. By default, the wand calibration setting uses the standard wand and calibration kit settings, supplied with the system.

### 3.3 Test Results

An extensive amount of testing was done with the wand calibration to build confidence in this method. Tests mainly consisted of varying wand length and L-frame size, then recording the results.

Through using an exact wand length and varying L-frame marker positions it was shown that if one marker is incorrect by 15mm or more error increases by a factor of 10 or greater. As error increases so do standard deviation of wand length, average residual, and the difference between maximum and minimum wand length.

Wand Length = 583.27mm

L-Frame (mm)	Id	X (mm)	Y (mm)	Z (mm)	Points	Avg res (mm)	Error (mm)	Std. Deviation	Max-Min
751.76, 203.75, 550.75	1	2057.22	629.04	4535.79	787	0.26718	-13.47638642	1.65987	16.58508
	2	-1938.71	565.35	4509.56	787	0.26718			
752, 204, 551	1	2067.48	627.2	4531.04	789	0.26839	2.916356741	1.19737	14.20502
	2	-1944.85	563.47	4505.66	789	0.26839			
755, 205, 553	1	2071.75	626.42	4529.09	789	0.27017	9.733580786	1.0788	13.60541
	2	-1947.4	562.66	4504.07	789	0.27017			
760, 205, 555	1	2106.67	619.92	4513.07	793	0.30879	65.53955114	2.34736	11.51135
	2	-1968.31	556.02	4491.22	793	0.3088			
762, 208, 557	1	2095.36	622.12	4518.42	791	0.29008	47.47763135	1.68563	11.59735
	2	-1961.55	558.24	4495.5	791	0.29007			
765, 210, 560	1	2078.66	625.32	4526.11	791	0.27251	20.8607968	1.06619	13.46674
	2	-1951.62	561.44	4501.59	791	0.27251			
770, 215, 565	1	2050.2	630.41	4539.18	775	0.26654	-24.55981401	1.90322	8.30164
	2	-1934.64	566.7	4512.26	775	0.26655			
750, 200, 550	1	2057.22	629.04	4535.79	787	0.26718	-13.47638642	1.65987	16.58508
	2	-1938.71	565.35	4509.56	787	0.26718			
740, 190, 540	1	2115.83	618.33	4509.34	799	0.32999	80.17190789	2.94427	13.23688
	2	-1973.79	554.47	4488.45	799	0.33001			
735, 200, 545	1	1903.69	656.29	4591.44	744	0.56606	-258.3939747	11.84394	48.85986
	2	-1847.21	593.54	4554.57	744	0.56608			
750, 185, 545	1	2167.51	608.5	4484.11	813	0.47007	163.7248075	6.48472	28.2605
	2	-2005.69	544.49	4467.28	813	0.47008			
750, 200, 520	1	2519	525.56	4089.21	475	2.02371	730.9697278	13.62621	53.20776
	2	-2221.51	461.02	4103.12	475	2.02383			

Figure 3



Indicates the actual marker distances

Error and L-frame values are the only two values not given from the calibration files. Error is the difference between calculated distance and measured distance between cameras.

$$\text{Error} = (\text{SQRT} ((X_1 - X_2)^2 + (Y_1 - Y_2)^2 + (Z_1 - Z_2)^2)) - R_d$$

**X<sub>1</sub>, Y<sub>1</sub>, Z<sub>1</sub>**: The coordinates for camera 1 resulting from the calibration

**X<sub>2</sub>, Y<sub>2</sub>, Z<sub>2</sub>**: The coordinates for camera 2 resulting from the calibration

**R<sub>d</sub>**: The actual distance measured between the two cameras.


To acquire the actual marker positions, the cameras were calibrated using a peak frame, which has exact coordinates for each marker. L-frame markers were captured by the cameras; thus, giving their positions. A calibration file using exact wand length and L-frame marker positions was created. To generate all other data, L-frame marker distances were entered into the calibration files. Then, the files were recalibrated and results were recorded.

It was shown that the accuracy of both wand length and L-frame structures marker positions is the governing factor in error results. If both these measurements were incorrect either error vastly increased, or there was a calibration failure since the cameras could not locate markers properly.

## 4.0 CAPTURING MEASUREMENTS

### 4.1 2D and 3D Motion Capture

A single camera can be used to capture 2D data. It does not need calibration; however, it does need to be linearized. When 2 or more cameras are connected both 2D and 3D data is generated. To capture a 2D/3D measurement:

1. Open a new file
2. Select capture icon  or choose the Capture tab then capture. The following screen appears:

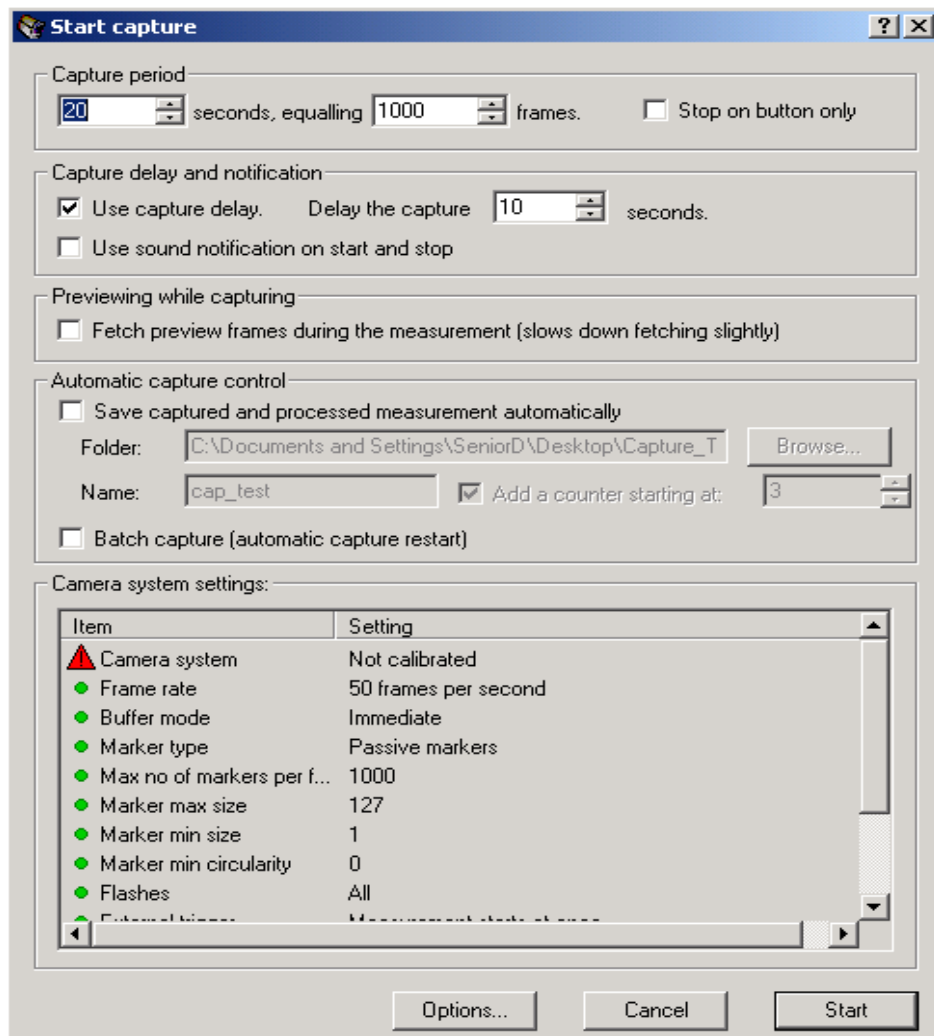


Figure 4

- Next, set the capture options such as capture time and capture delay as required.
- Start the capture. Once the capture has finished 2D data, 3D data, a 2D viewing window and a 3D viewing window are available for display.

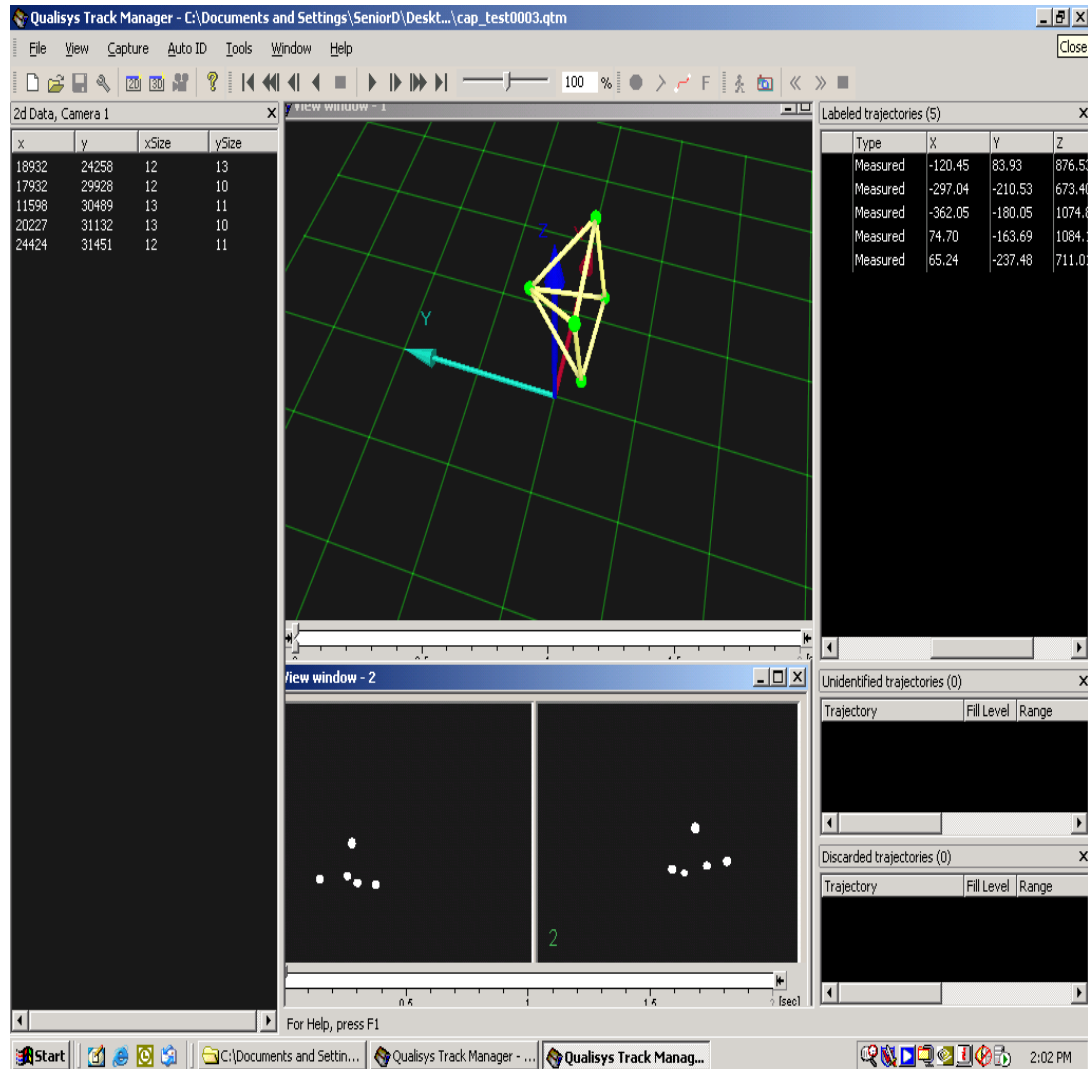


Figure 5

## 4.2 6DOF Motion Capture

### 4.2.1 Capturing 6DOF data

The 6DOF real time output gives x, y, z, roll, pitch, yaw, and residual error of the rigid body being traced. Multiple bodies can be traced at the same time.

To complete a 6DOF capture:

1. Open a new file, place the rigid body in the viewing area of the cameras
2. The body must have at least 4 markers and they cannot be placed in a symmetrical fashion.
  - Go to 3D view window, ensure cameras see body markers
  - Processing→Select Track the Measurement
  - Processing→6DOF tracking→6DOF bodies
  - Choose to acquire the body
3. Under the processing tab deselect “Track the measurement” and select “Tracking 6DOF”
4. Choose Capture tab, and select the 6DOF real time output option.

It is important to de-select the “Keep visualizing while RT tracking” option under Processing→6DOF tracking→RT output. This allows the cameras to capture at the desired frequency. When this option is selected the capture rate lowers significantly because the display graphics are continuously updated, which consumes time.

## 4.2.2 QTMDAC

QTMDAC software converts 6DOF real time data provided by QTM. It provides an analog output as well as displays the key components of the 6DOF data including:

- Number of bodies
- Number of identified and unidentified markers
- Frame count
- Frequency
- Plot of the residual error
- 6DOF data (x, y, z, roll, pitch, yaw)

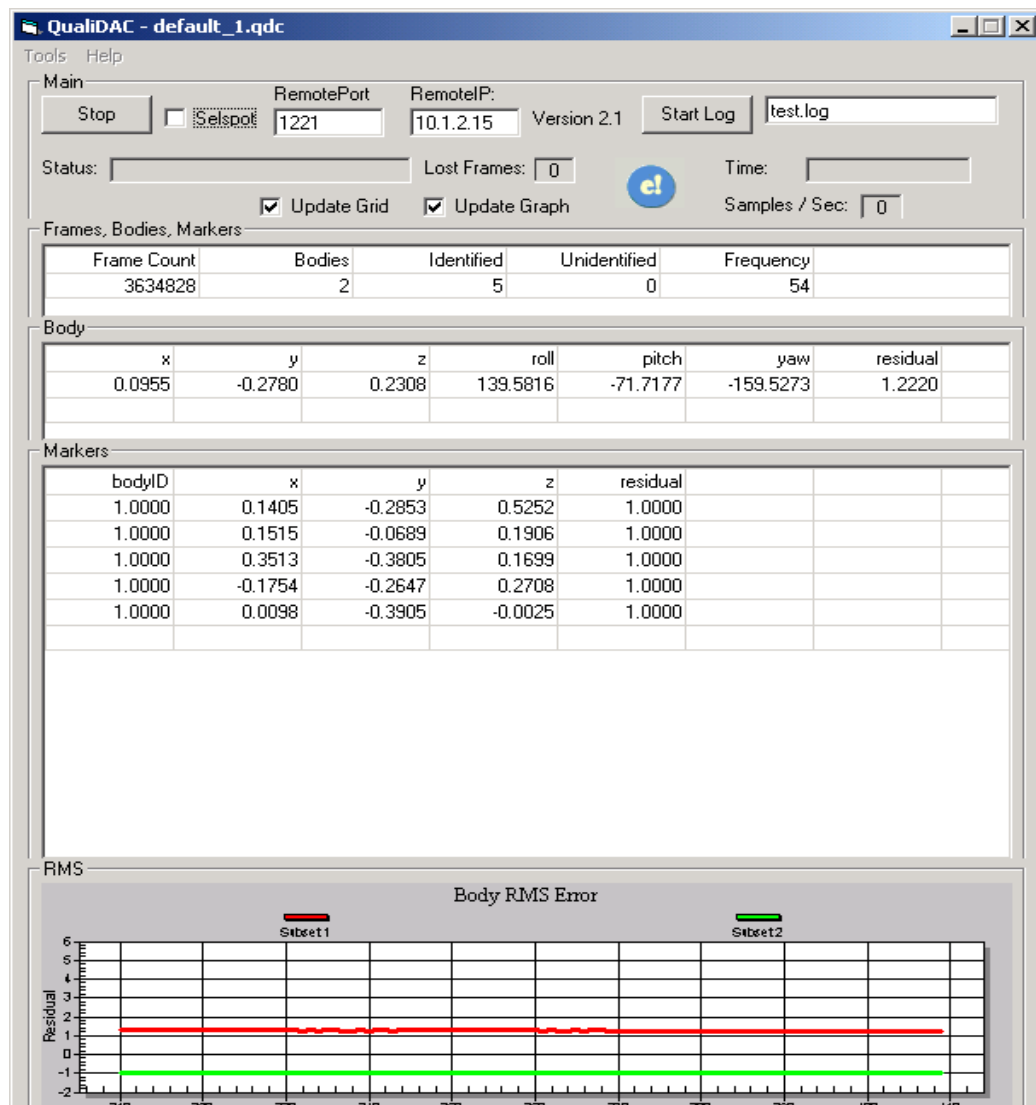


Figure 6

QTMDAC has the capabilities of QualiDAC software with the exception that it can communicate with the QTM software via a TCP/IP (Transmission Control Protocol/Internet Protocol) connection.

A QTMClient header file written in Visual C is used to communicate to QTMClient DLL (Dynamic Link Library) and consequently QTM. In order to allow QTMDAC access to QTM, the header file was converted to VB and became QTMClientVB module. This module is included in QTMDAC and incorporates the QTMClient DLL interface. It contains functions and data types that allow QTMDAC to open, close, or gather data across a connection to QTM.

Functions included in QTMClientVB:

- **QTM\_OpenTCPConnection:** Opens TCP/IP connection between QTM and QTMDAC
- **QTM\_AllocateFrameMemoryUnidentified3D, QTM\_AllocateFrameMemoryIdentified3D, QTM\_AllocateFrameMemory6DOF:** Allocate memory necessary for body and marker information
- **QTM\_ContinuousTransmission:** Determines whether data is sent from QTM to QTMDAC continuously or by frame
- **QTM\_HasNewData:** Checks for new data
- **QTM\_GetFrame:** Get data frame and types of frames
- **QTM\_FreeFrameMemory6DOF, QTM\_FreeFrameMemoryIdentified3D, QTM\_FreeFrameMemoryUnidentified3D:** Releases memory allocated to avoid memory overflow
- **QTM\_CloseConnection:** Closes TCP/IP connection between QTM and QTMDAC

## **5.0 CONCLUSIONS**

The QTM software and ProReflex cameras is an extremely useful motion capture system. It encapsulates the use of multiple cameras (up to 16) and tracking of multiple bodies (up to 6). The rate of capture can reach at least 90Hz with the QTM/QTMDAC system. This is a vast improvement from the previous Qualisys/QualiDAC system that had a maximum of 30Hz capture rate.

### **5.1 Recommendations**

QTM has the ability to output analog data, but it requires the use of force plates and only outputs the force data. Perhaps to reduce the need for 2 pieces of software, the analog output of QTM could be developed to output 6DOF real time data.

Further testing with multiple bodies should be completed if more than 2 bodies are to be traced at the same time. Also, extended calibration method should be tested further. This method incorporates the wand calibration method, but allows cameras that cannot see the L-frame reference structure to be calibrated by the overlap of adjacent cameras field of view, which see the reference frame. It appears to be beneficial in large measurement volumes, but was not tested.

## 6.0 REFERENCES

1. Qualisys Medical AB, Qualisys Track Manager 1.7.187, [www.qualisys.se](http://www.qualisys.se), 2004-10-26
2. Peter King, QualiDAC Software Manual (LM-2003-11), National Research Council, April 2003
3. Geoff Holden, QualiDAC Software Manual Addendum (LM-2003-34), National Research Council, December 2003
4. Visual Basic/VB.NET Message Board, CodeProject, <http://www.thecodeproject.com>, 1999-2004

**Appendix A**  
**QTracREADME**

A working system for QTrac 2.57 with the MacReflex cameras (Firmware 5.0), is as follows:

1. Use Windows 98/2000
2. Install ACB-530 card (4111) with settings as per the "SeaLevel ACB-530 and PC-ACB Installation Instructions" manual (pgs 5-7), with two exceptions:
  - E3 has a jumper across 7 (IRQ 7)
  - E1 has jumpers on T and N.
3. Ensure no other device is using DMA channel 3
  - Go to Start→Programs→Accessories→System Tools→System Information→HW Resources→DMA
  - If a device uses channel 3, change the setting in BIOS
4. Go to Setup→Hardware and change the settings to:
  - Camera/MC→Check VP/Camera
  - Serial→Choose ACB-530 (RS-422), DMA Channel 3, and Port Address is 0x238

**\*\*\*NOTE\*\*\* WHEN USING WIN98 OR 2000 NO DRIVER INSTALLATION (SUCH AS SEAMAC) IS NECESSARY SINCE QTRAC HAS ITS OWN DRIVER FOR THE ACB CARD\*\*\***

QTrac has never been formally supported on Windows NT4 or XP. However, it has been able to run on NT4 by installing a SeaMac driver and ensuring that "Devices" sees the driver. We could not get this to work with SeaMac version 2.4.5. QTrac would complain and give an error message that the driver could not be found. We also tried the ACBII card instead of the ACB-530 but it did not work. Outlined in the manual mentioned above are instructions for installing the ACB-530 and PC-ACB cards with Windows 95 and NT. We followed these instructions but could not get QTrac to recognize the driver.

Windows NT or 98 did not recognize the ACB-530 card as new hardware. It is not shown in the Performance tab under System Properties (Win98). However, QTrac recognizes the MacReflex cameras (when QTrac has the Hardware settings as above).

**CABLE CONNECTIONS (Videoprocessor to ACB-530/PC-ACB):**

HSKi 1<-----19 RTS+  
HSKo 2-----> 9 RXC+  
RXD- 3<----- 2 TX-  
GND 4----- 7 GND  
TXD- 5-----> 3 RX-  
RXD+ 6<-----14 TX+  
Gpo 7----->13 CTS+  
TXD+ 8----->16 RX+  
          +----17 RXC-  
          |  
          +---- 7 GND

**CABLE CONNECTIONS (Videoprocessor to ACB-II):**

HSKi 1<-----22 RTS+  
HSKo 2-----> 8 RXC+  
RXD- 3<-----25 TX-  
GND 4----- 7 GND  
TXD- 5----->13 RX-  
RXD+ 6<-----24 TX+  
Gpo 7----->10 CTS+  
TXD+ 8----->12 RX+  
          +---- 9 RXC-  
          |  
          +---- 7 GND

Linearization of MacReflex cameras and ProReflex cameras using QTrac:

Place the 8x10 linearization grid on a flat surface; attach the centering rod.

**\*\*MAKE SURE THE ROD IS PERPENDICULAR WITH THE GRID TO IMPROVE ACCURACY OF THE LINEARIZATION\*\***

**MacReflex** -- Once a working system is in place as outlined above the cameras must be linearized as follows:

1. Mount the Camera on a tripod, open QTrac→New→Motion Capture; push the play symbol (▶). This displays the markers in the camera's FOV (Field of View).
  - Adjust the camera such that all 80 markers on the grid are contained in the FOV (yellow rectangle).
  - Adjust the camera such that the marker on the center rod is directly in the center of the two vertical markers. **Be as accurate as possible.**
2. If some of the markers are not in the camera's FOV close the Motion Capture window, go to Setup→Hardware, choose Marker Discrimination settings as appropriate, for example Maximum per Frame = 100, Largest = 100, Smallest = 1, and Capture Rate as appropriate (60Hz). Also, make sure the focus and aperture are set properly. Reopen Motion Capture window.
3. Once the camera is positioned properly, cover the two vertical markers and the marker on the rod. **Measure and record the distance from the grid to the focal point of the lens.**
4. Close the Motion Capture window and open New→Linearization, go to Setup→Linearization and enter the "Camera Distance to Grid"
  - Push the play (▶) button
  - Push the capture button
  - Once the linearization is complete save the linearization file and exit.
5. Repeat steps 1 - 4 for remaining MacReflex cameras.

**\*\*NOTE: WHEN USING THE LINEARIZATION WINDOW, TIMEOUTS MAY OCCUR, THIS IS WHY IT IS NECESSARY TO DO THE POSITIONING OF THE CAMERAS IN THE MOTION CAPTURE WINDOW. IF A TIMEOUT OCCURS WHILE LINEARIZING A CAMERA, SIMPLY REOPEN THE LINEARIZATION WINDOW.\*\***

**ProReflex** -- Both QTM and QTrac use the same SeaLevel board setup.

1. Go to S:\Qualisys\QTM Software\Firmware\Old Firmware
  - Open QFI; follow the first two instructions.
  - If the camera has firmware 7.0 or above it must be changed to version 6.\*\*.
2. Mount the camera, open QTrac, go to Setup→Hardware and change the settings to:
  - Camera/MC→Check MCU 240
  - Serial→Choose COM Port (RS-232), COM3
3. Open the Linearization window (New→Linearization).
  - Push the play (▶) button.
  - Adjust the camera such that the rod marker is directly in the center of the two vertical markers.
  - All 80 markers on the grid need to be in the camera's FOV, and they must fill as much of the viewing area (yellow rectangle) as possible.
4. If problems arise with viewing the markers adjust the focus and aperture as described in the "QTM User Manual", and adjust the marker settings under Setup→Hardware.
5. Measure the distance from the camera to the grid and enter it in the "Camera Distance to Grid" setting (Setup→Linearization).
6. Cover the two vertical markers and the rod marker. Capture the linearization and save the file.
7. Repeat steps 1 – 6 for the remaining ProReflex Cameras.

## **Appendix B**

### **CalibrationREADME**

## Calibration


There are three methods of calibration using the QTM (Qualisys Track Manager) Software:

- Wand Calibration
- Frame Calibration
- Fixed Camera Calibration

*Before calibrating the camera system they must be linearized, see “QtracREADME”.*


*For more information about calibration techniques refer to “Qualisys Track Manager User Manual”*

### Wand Calibration:

1. Start QTM, locate the camera system, and load the linearization files for each camera.
2. Place the reference frame in the measurement volume.
3. Go to Tools→Workplace options→Calibration.
  - Select Wand calibration under Calibration type.
  - Choose the appropriate Calibration Kit settings, and “Coordinate system orientation and translation”.
  - Be sure to enter the “Exact wand length (mm)”.
4. If the default calibration and wand kit settings are not suitable:
  - Choose “use advanced definition of reference object instead”.
  - Choose the “Advanced...” tab and enter the appropriate settings.
  - Choose **Apply**, then **Ok**.
5. Open a new file; choose the calibration icon . A Calibration window will appear. Choosing the Calibration Quality setting is dependant on the size of the measurement volume. The time should be set to allow the wand to be moved in as many positions as possible throughout the volume. This allows the cameras to view more points and improves the accuracy of the calibration.
6. Start the calibration, move the wand throughout the measurement volume.

7. When the calibration is completed a message window will appear. This window gives:
  - Locations of the cameras with respect to the calibration coordinate system
  - Number of points used in the calibration for each camera
  - Average residual
  - Standard deviation of wand length
  - Difference in wand length.If the calibration was unsuccessful, an error message will appear in the **Calibration results** window.
8. **Track calibration and view it** provides a view of the entire calibration process. It is important to choose this option before accepting the calibration file to ensure there are no *phantom markers* present.
9. The calibration process is now complete. Choose **Ok**, and begin capturing a measurement.

### **Frame Calibration:**

1. Start QTM, locate the camera system, and load the linearization files for each camera.
2. Place the Frame in the measurement volume.
3. Open a new file; this gives a preview of what the cameras can see. Align the cameras properly.
4. Go to Tools→Workplace options→ Calibration; choose the Frame Calibration option.
5. Take one marker as a reference (0,0,0). Measure all other markers with respect to the reference. **Be as accurate as possible (within 1mm)**. Enter the locations of the markers in the appropriate section. Also, enter the number of markers that can be seen by each camera.
6. Choose **Apply**, then **Ok**. Choose the calibration icon . Start the calibration.

7. When the calibration is completed a message window will appear. This window gives:
  - Locations of the cameras with respect to the calibration coordinate system
  - Number of points used in the calibration for each camera
  - Average residual
  - Standard deviation of wand length
  - Difference in wand length.If the calibration was unsuccessful, an error message will appear in the **Calibration results** window.
8. **Track calibration and view it** provides a view of the entire calibration process. It is important to choose this option before accepting the calibration file to ensure there are no *phantom markers* present.
9. The calibration process is now complete. Choose **Ok**, and begin capturing a measurement.

### **Fixed Camera Calibration:**

1. Start QTM, locate the camera system, and load the linearization files for each camera.
2. Setup the cameras and place at least two reference markers in the measurement volume.
  - Get the exact location of each camera and each reference marker. This is usually accomplished through surveying the measurement volume.
  - Make sure each camera sees at least two markers.
3. Go to Tools→Workplace options→Calibration. Choose “Fixed camera calibration”.
4. Enter the exact camera locations, exact marker locations, and the number of markers seen by each camera in order from left to right.
5. Choose **Apply**, then **Ok**. Open a new file; choose the calibration icon ➤. Start the calibration.

6. When the calibration is completed a message window will appear. This window gives:
  - Locations of the cameras with respect to the calibration coordinate system
  - Number of points used in the calibration for each camera
  - Average residual
  - Standard deviation of wand length
  - Difference in wand length.If the calibration was unsuccessful, an error message will appear in the **Calibration results** window.
7. **Track calibration and view it** provides a view of the entire calibration process. It is important to choose this option before accepting the calibration file to ensure there are no *phantom markers* present.
8. The calibration process is now complete. Choose **Ok**, and begin capturing a measurement.

## **Appendix C**

### **Wand Calibration Test Results**

Tables 1 and 2 contain test results from using one L-Frame reference structure and multiple wand lengths. Dependency on the number of points obtained from calibration was evaluated. Generally, as the number of points gathered increases, error decreases.

L-Frame: 402,300,160

Wand Length (mm)	Id	X (mm)	Y (mm)	Z (mm)	Points	Avg res (mm)	Error (mm)	Std. Deviation	Max-Min
550	1	2657.14	307.23	4317.55	250	1.11461	-715.4473	6.04732	29.32935
	2	-636.17	334.26	4231.2	250	1.11506			
565	1	2712.54	319.11	4426.11	272	1.29275	-637.0327	8.33895	46.47931
	2	-658.83	342.85	4325.07	272	1.29348			
570	1	2728.24	322.83	4467.14	270	1.32083	-621.1835	9.60299	48.40161
	2	-658.79	346.26	4359.64	270	1.32157			
575	1	2744.38	329.03	4495.3	273	1.38674	-595.0032	11.14833	56.54468
	2	-668.73	350.21	4383.79	273	1.38773			
(Original) 580	1	2770.21	329.48	4510.31	280	1.38159	-557.2862	11.36271	48.8429
	2	-680.93	351.84	4408.5	280	1.38195			
585	1	2781.25	327.51	4514.75	290	1.37991	-532.1022	13.2962	48.80109
	2	-695.11	352.09	4414.3	290	1.38043			
590	1	2775.16	324.26	4476.08	328	1.4767	-547.4984	17.64419	53.38318
	2	-686.23	352.09	4392.88	328	1.47783			
595	1	2775.51	319.68	4456.33	330	1.59624	-549.6002	17.88797	50.258
	2	-684	350.27	4384.07	330	1.59778			
600	1	2776.12	318.09	4441.45	277	1.75456	-540.2567	15.51149	49.01251
	2	-692.8	350.69	4373.26	277	1.75771			
605	1	2817.56	305.21	4424.9	253	1.51072	-393.1295	9.0759	37.5694
	2	-798.5	335.13	4354.42	253	1.51171			
610	1	2856.67	302.61	4443.2	271	1.40076	-296.9382	7.53141	40.52362
	2	-855.56	332.82	4370.65	271	1.40121			
620	1	2897.07	305.56	4506.89	267	1.41266	-249.3321	6.9088	40.87671
	2	-862.79	336.74	4435.45	267	1.41343			
625	1	2904.64	308.47	4525.01	271	1.44892	-272.9689	8.20133	50.23566
	2	-831.82	338.27	4466.87	271	1.44969			

Table 1

L-Frame: 402,300,160

Wand Length (mm)	ld	X (mm)	Y (mm)	Z (mm)	Points	Avg res (mm)	Error (mm)	Std. Deviation	Max-Min
550	1	2922.38	311.73	4251.23	46	7.41083	34.2913	15.88385	40.43909
	2	-1120.99	323.09	4165.66	46	7.41068			
565	1	2786.67	273.59	4079.1	822	0.21842	-149.7421	2.21557	8.80963
	2	-1072.29	297.72	3981.96	822	0.21849			
570	1	2811.29	276.02	4115.06	822	0.2202	-115.6517	2.2385	8.88293
	2	-1081.75	300.35	4017.1	822	0.22027			
575	1	2835.9	278.42	4151.15	826	0.22176	-81.5092	2.26209	8.95953
	2	-1091.27	302.97	4052.29	826	0.22184			
(Original) 580	1	2860.5	280.87	4187.1	820	0.22411	-47.5292	2.28535	9.04633
	2	-1100.64	305.62	4087.44	820	0.22418			
585	1	2884.99	283.38	4223.01	811	0.22375	-13.6077	2.31015	9.12927
	2	-1110.06	308.34	4122.49	811	0.22382			
590	1	2909.52	285.84	4259.2	808	0.22604	19.8486	2.36006	9.30847
	2	-1118.98	311.02	4158.04	808	0.22611			
595	1	2934.24	288.25	4295.35	812	0.22792	54.0092	2.38102	9.39728
	2	-1128.41	313.66	4193.37	812	0.22799			
610	1	3007.3	295.78	4403.21	788	0.23405	154.4508	2.49586	8.97662
	2	-1155.76	321.7	4298.76	788	0.23412			
620	1	2868.73	306.49	4213.88	95	9.41029	-55.6734	23.5762	59.44415
	2	-1084.47	314.65	4119.85	95	9.4105			

**Table 2**

Accuracy of tracking an object, once it was calibrated using the wand technique, was tested. An object was moved a measured distance (Measured Dist). It was tracked by QTM and from obtaining x, y, z coordinates of the markers a distance (Calculated Dist) was calculated. Differences between the two distances were recorded (Difference).

Tables 3 and 4 contain data obtained from moving a 5-marker tree structure. Both use a large wand (580mm), but Table 3 uses a larger L-frame than Table 4.

	X	Y	Z		
Tree	-167.408	93.654	431.758	Calculated Dist	343.6501
Large L_Frame	-173.41	85.673	775.263	Measured Dist	355
20041005_112915.qca				Difference	-11.3499

**Table 3**

	X	Y	Z		
Tree	584.76	-386.034	-134.641	Calculated Dist	814.5363
Small L-Frame	499.408	-373.233	675.31	Measured Dist	664
20041004_085101.qca				Difference	150.5363

**Table 4**

Table 5 holds results acquired from moving a 2-marker frame a measured distance.

	X	Y	Z		
2 markers	874.31	-268.8	-485.66	Calculated Dist	278.2362
Cal File: 20041004_085101.qca	883.321	9.278	-483.047	Measured Dist	268
				Difference	10.2362
	1191.99	-282.42	-499.37	Calculated Dist	249.8302
	1198.516	-32.691	-496.546	Measured Dist	235
				Difference	14.8302

**Table 5**

In order to distinguish if results from the wand calibration are accurate, a similar test was conducted using Peak Frame. Peak frame is a frame of markers with exact positions. It is used to calibrate cameras and produces accurate results.

Peak Frame Cal Results

ID	X (mm)	Y (mm)	Z (mm)	Points	Avg Res(mm)	Error (mm)
1	2133.38	753.1	4027.19	10	5.63948	17.3199
2	-1891.85	819.24	4138.79	10	5.51667	

	X	Y	Z		
Peak Frame Cal	-142.461	319.715	344.206	Calculated Dist	475.4734
	-154.391	328.949	-131.028	Measured Dist	479
				Difference	-3.5266

**Table 6**

Since wand calibration results were much higher than Peak Frame results, experimentation to improve the wand calibration technique was needed. First, exact wand length and L-frame marker positions were captured (cameras were calibrated using Peak Frame). Then, wand lengths and L-frame marker positions were calculated.

Large L-Frame:

Long Arm	Short Arm	Long Arm Middle
751.76	550.749	203.753

Large Wand:

X	Y	Z	Calculated Length
-343.537	731.21	-404.462	583.2695
239.288	717.2	-422.407	

Small L-Frame:

Long Arm	Short Arm	Long Arm Middle
401.461	289.348	157.698

Small Wand:

X	Y	Z	Calculated Length
38.26	52.18	14.85	309.1327
347.37	55.79	15.86	

**Table 7**

The following tables have results obtained from using exact wand lengths and L-frame marker positions in different combinations.

Results using 583.2695 mm wand and large frame

ID	X (mm)	Y (mm)	Z (mm)	Points	Avg Res (mm)	Error (mm)
1	2057.22	629.04	4535.79	787	0.26718	-13.4764
2	-1938.71	565.35	4509.56	787	0.26718	

	X	Y	Z		
Moving 5-marker Frame	-53.37	93.28	780.99	Calculated Dist	479.1263
	-51.39	97.89	301.89	Measured Dist	479
				Difference	0.1263

**Table 8**

Results using 309.1327 mm wand and large frame

ID	X (mm)	Y (mm)	Z (mm)	Points	Avg Res (mm)	Error (mm)
1	2063.54	627.98	4504.46	805	0.43909	-8.9057
2	-1937.04	567.6	4482.79	805	0.43909	

	X	Y	Z		
Moving 5-marker Frame	-46.02	93.33	776.23	Calculated Dist	473.4889
	-42.91	97.54	302.77	Measured Dist	479
				Difference	-5.5111

**Table 9**

Results using 583.2695 mm wand and small frame

ID	X (mm)	Y (mm)	Z (mm)	Points	Avg Res (mm)	Error (mm)
1	2862.29	274.4	4213.04	787	0.27637	-5.1243
2	-1140.87	297.23	4098.07	787	0.27636	

	X	Y	Z		
Moving 5-marker Frame	815.21	-156.11	379.29	Calculated Dist	478.2802
	822.26	-143.81	-98.78	Measured Dist	479
				Difference	-0.7198

**Table 10**

Results using 309.1327 mm wand and small frame

ID	X (mm)	Y (mm)	Z (mm)	Points	Avg Res (mm)	Error (mm)
1	2850	270.45	4195.12	711	0.35244	-19.8929
2	-1138.4	295.48	4081.13	711	0.35245	

	X	Y	Z		
Moving 5-marker Frame	811.65	-155.57	377.41	Calculated Dist	476.4213
	818.95	-143.02	-98.79	Measured Dist	479
				Difference	-2.5787

**Table 11**

Camera distance is distance from the reference frame origin to the camera. A camera system, consisting of 2 cameras, was calibrated using different wand and L-frame combinations and using Peak Frame. Each camera's distance was calculated from calibration results. Mean, standard deviation, minimum and maximum values, and average absolute deviation of camera distance was calculated for camera 1 and camera 2. Generation of similar results, camera distances for example, was shown to be difficult using wand calibration. Especially if a number of wand and L-frame combinations are used.

Peak Frame Cal Results

ID	X (mm)	Y (mm)	Z (mm)	Camera Distance
1	2133.38	753.1	4027.19	4619.1697
2	-1891.85	819.24	4138.79	4623.8332

Camera 1	
Mean	4983.9967
Std Deviation	151.6348
Average Absolute Deviation	91.2067
Hi = 5100.75538, Low = 4619.169745	

Cal Results using 583.2695 wand, Large L-Frame

ID	X (mm)	Y (mm)	Z (mm)	Camera Distance
1	2057.22	629.04	4535.79	5020.0833
2	-1938.71	565.35	4509.56	4941.0878

Camera 2	
Mean	4726.5167
Std Deviation	309.9473
Average Absolute Deviation	261.0533
Hi =4941.736361, Low = 4247.220864	

Cal Results using 309.1327 wand, Large L-Frame

ID	X (mm)	Y (mm)	Z (mm)	Camera Distance
1	2063.54	627.98	4504.46	4994.2683
2	-1937.04	567.6	4482.79	4916.2689

Cal Results using 583.2695 wand, Large L-Frame

ID	X (mm)	Y (mm)	Z (mm)	Camera Distance
1	2080.17	642.47	4524.16	5020.7468
2	-1941.99	578.43	4507.2	4941.7364

Cal Results using 583.2695 wand, Large L-Frame

ID	X (mm)	Y (mm)	Z (mm)	Camera Distance
1	2076.13	646.36	4521.36	5017.0503
2	-1934.92	582.45	4504.98	4937.4091

Cal Results using 583.2695 wand, Small L-Frame

ID	X (mm)	Y (mm)	Z (mm)	Camera Distance
1	2862.29	274.4	4213.04	5100.7554
2	-1140.87	297.23	4098.07	4264.2828

Cal Results using 309.1327 wand, Small L-frame

ID	X (mm)	Y (mm)	Z (mm)	Camera Distance
1	2850	270.45	4195.12	5078.8458
2	-1138.4	295.48	4081.13	4247.22089

Cal Results using 583.2695 wand, Large L-Frame

ID	X (mm)	Y (mm)	Z (mm)	Camera Distance
1	2076.81	625.21	4528.46	5021.0534
2	-1949.55	561.55	4504.49	4940.2949

**Appendix D**  
**QTMDAC Source Code**

\*\*\*\*\*Main Form\*\*\*\*\*

-----  
'           QualiDAC  
'           -----  
'

'Description:

' This software is intended to enhance the current Qualisys system  
' capabilities by taking marker and body data directly from the Qualisys  
' system and providing a visual display, analog output, an RS-422 output  
' stream, and text file logging capabilities.  
'

'Author:

' Peter King  
' Work Term V  
' Faculty of Engineering  
' Memorial University of Newfoundland  
' www.peterking.ca  
' peter@engr.mun.ca  
'

'Last Modified:

' 15 Dec 2004  
' Dena Senior  
' Work Term III  
' MUN  
'

'Modifications include:

' Module QTMClientVB added to project  
' The ability to communicate via a TCP/IP connection to QTM software  
' Option to update grid and/or graph  
' Renamed QTMDAC  
'

Option Explicit

Private Const SM\_CXVSCROLL As Long = 2  
Private Const MAX\_COLUMNS As Long = 7  
Private Const MAX\_COLUMNS\_INFO As Long = 5  
Private Const COLUMN\_WIDTH As Long = 75  
Private Const COLUMN\_WIDTH\_INFO As Long = 90  
Private Const HEADER\_COLOR As Long = &HC0FFFF 'Light Yellow  
Private Const ESCKeyCode As Long = 27  
Private Const FRAME\_RATE\_WINDOW\_MODULUS As Long = 301  
Private Const SLEEP\_MIN As Long = 1  
Private Const SLEEP\_SETTLING\_TIME As Long = 1000

Private m\_boolUpdateGrid As Boolean  
Private m\_boolUpdateGraph As Boolean  
Private m\_boolFrameDropped As Boolean  
Private m\_dwPreviousFrame As Long

Private Declare Sub Sleep Lib "Kernel32" (ByVal dwMilliseconds As Long)

Private Declare Function GetSystemMetrics Lib "user32" (ByVal nIndex As Long) As Long  
Private Declare Function timeBeginPeriod Lib "winmm.dll" (ByVal uPeriod As Long) As Long

```

Private Declare Function timeEndPeriod Lib "winmm.dll" (ByVal uPeriod As Long) As Long

Dim f3 As Object          'Type 3 Frame (UnIdentified Markers)
Dim f5 As Object          'Type 5 frame (Body)
Dim isOpen As Boolean     'Open state of PortA
Dim isStart As Boolean    'loop state
Dim str As String         'general purpose output string

'Port A variables
Dim errA As Long          'PortA error state
Dim handleA As Long       'Handle to PortA
Dim RxBufferA As String   'Input buffer
Dim PortInfoA As PORT_INFO 'Structure containing port information
Dim RxCount As Integer    'number of filled Rx Buffers
Dim TxCount As Integer    'number of available Tx buffers

'Qualisys data stream
Dim retbytes As Long      'number of bytes returned from Qualisys
Dim byteBuf() As Byte     'returned data from Qualisys
Dim split As Long        'location where marker data ends and body begins
Dim watchdogcount As Long 'number of times watchdog has reset Qualisys

'TCP/IP variables
Dim moPPUFrameVB As QTMClientVB.QTM_TPPUFrame 'VB version of moPPUframe pointer
Dim moPPUFrame As QTMClientVB.QTM_TPPUFrameC 'VC version of moPPUframe pointer
Dim bStatus As Byte      'status variable
Dim errtcp As Byte       'TCP/IP error state

Dim dwFrameCount As Long
Dim boolGridCleared As Boolean
Dim boolGraphCleared As Boolean

'Variables used to calculate Frequency
Dim startCount As Currency
Dim finishCount As Currency
'-----
'Form_Load
' initializes variables and objects
'-----
Private Sub Form_Load()

    isOpen = False
    fileOpen = False

    m_boolUpdateGrid = chkUpdateGrid.Value
    m_boolUpdateGraph = chkUpdateGraph.Value

    m_boolFrameDropped = False

    m_dwPreviousFrame = -1
    'create frame objects
    Set f3 = New Frame3
    Set f5 = New Frame5

    'get system clock frequency
    QueryPerformanceFrequency curFreq

```

```

'If no configuration file present
If Not frmConfig.initializeConfig(GetSetting("QualiDAC", "Config", "configFile", "default.qdc")) Then
    ' Write config file, then retry initializing the config.
    Set frmConfig.cfgFileStream = frmConfig.cfgFileObj.OpenTextFile("default.qdc", ForWriting, True)
    frmConfig.cfgFileStream.Write defaultConfig
    Set frmConfig.cfgFileStream = Nothing
    If Not frmConfig.initializeConfig("default.qdc") Then
        Unload Me
        Exit Sub
    End If
End If

```

```

ChangePriority (HIGH_PRIORITY_CLASS)

```

```

'For AOB8/16 card
InPortB AUTO    'place in automatic update mode

```

```

' This application uses a freeware grid from vbAccelerator to display data
' This grid enables fast update rates and is intuitive to use
'
' For fastest (flicker free) performance, the Redraw property should be
' set to False while updating the grid and set back to True once the
' grid has been completely updated
'
' The major grid properties used in this application are AddColumn, AddRow, and
' CellDetails
'
' The first 5 AddColumn parameters are key, header, textAlignment,
' iconIndex, and columnWidth
'
' The first 4 CellDetails parameters are row, col, text, textAlignment
'
' vbalGrid6.ocx and SSubTmr6.dll must be registered on the client PC
' to use this component
'

```

```

Dim dwCol As Long

```

```

With vbalGridBody

```

```

    'Turn off Redraw until grid is fully updated (reduces flicker)
    .Redraw = False

```

```

    .DefaultRowHeight = 17

```

```

    'AddColumn - specify the column index as the key
    '     specify an empty string for the header value
    '     leave the text alignment and icon index fields empty
    '     specify 90 for the column width

```

```

    'Set up the display grid to have MAX_COLUMNS

```

```

    For dwCol = 1 To MAX_COLUMNS
        .AddColumn dwCol, "", , , COLUMN_WIDTH
    Next dwCol

```

```

'Turn on Redraw after grid is fully updated (reduces flicker)
.Redraw = True
End With

With vbalGridMrks
'Turn off Redraw until grid is fully updated (reduces flicker)
.Redraw = False

.DefaultRowHeight = 17

'.AddColumn - specify the column index as the key
'    specify an empty string for the header value
'    leave the text alignment and icon index fields empty
'    specify 90 for the column width
'
'
'Set up the display grid to have MAX_COLUMNS
'

For dwCol = 1 To MAX_COLUMNS
    .AddColumn dwCol, "", , , COLUMN_WIDTH
Next dwCol

'Turn on Redraw after grid is fully updated (reduces flicker)
.Redraw = True
End With

With vbalGridInfo
'Turn off Redraw until grid is fully updated (reduces flicker)
.Redraw = False

.DefaultRowHeight = 17

'.AddColumn - specify the column index as the key
'    specify an empty string for the header value
'    leave the text alignment and icon index fields empty
'    specify 90 for the column width
'
'
'Set up the display grid to have MAX_COLUMNS
'

For dwCol = 1 To MAX_COLUMNS_INFO
    .AddColumn dwCol, "", , , COLUMN_WIDTH_INFO
Next dwCol

'Turn on Redraw after grid is fully updated (reduces flicker)
.Redraw = True
End With
End Sub

```

```

'-----
'Startbutton
' initializes main data stream loop
'-----
Private Sub cmdStart_Click()
    mnuTools.Enabled = False

    'start time of data stream
    QueryPerformanceCounter curStartTime

    cmdStart.Visible = False
    cmdStop.Visible = True

    isStart = True

    #If SIMULATION Then
    ' m_dwFrameCount = 0
    ' m_dwMarkerCount = 0
    ' m_dwBodyCount = 0
    '
    ' Display simulated tracking data until the ESC key is pressed or
    ' the form is unloaded
    '
    ' m_boolExiting = False
    ' m_boolEscKeyPressed = False
    '
    ' While ((Not m_boolEscKeyPressed) And (Not m_boolExiting))
    '     Call QTM_GetFrameVBT(moPPUFrameVB, moPPUFrame)
    '
    '     Call DisplayData(moPPUFrameVB)
    '
    '     DoEvents
    '     Sleep SLEEP_MIN
    '     Wend
    '
    ' If (Not m_boolExiting) Then
    '     Me.cmdStart.Enabled = True
    ' End If
    '
    ' Exit Sub
    #End If
    'initialize frame count
    f5.getFrameCount

    commStart      'open comm with Qualisys

    'while stop button not clicked
    While isStart
        getReading  'get data when available
    Wend

    commStop      'close comm with Qualisys

End Sub

```

```

'-----
'commStart
'Open portA and start data stream
'-----
Private Sub commStart()
    droppedFrames = 0
    chkSumCount = 0
    watchdogcount = 0
    cmdLog.Enabled = True
    lblDropped.Caption = "0"
    lblStatus.Caption = vbNullString

'Open portA to communicate with Qualisys
errA = SEAMAC_PortOpen(0, 1, handleA)
If errA = 0 Then
    isOpen = True

'initialize input string to buffer size
errA = SEAMAC_GetPortInfo(handleA, PortInfoA)
RxBufferA = Space$(PortInfoA.wRxFrameSize)
RxBufferB = Space$(PortInfoB.wRxFrameSize)

'Open portB to communicate to remote client
errB = SEAMAC_PortOpen(0, 2, handleB)
errB = SEAMAC_GetPortInfo(handleB, PortInfoB)

'Start Qualisys data stream
str = "#f000A" & vbCr '6D and 3D data
errA = SEAMAC_PutData(handleA, str, Len(str))

str = "#Tffff" & vbCr 'transmit continuous
errA = SEAMAC_PutData(handleA, str, Len(str))

str = "#Gffff" & vbCr 'start data collection
errA = SEAMAC_PutData(handleA, str, Len(str))

tmrWatchDog.Enabled = True
Else

bStatus = QTM_OpenTCPIPConnection("10.1.2.15", "1221")
If (bStatus = 1) Then
    isOpen = True

'Init to NULL otherwise QTM_AllocateFrameMemory...() will return error
ZeroMemory moPPUFrame, LenB(moPPUFrame)

'Set the frame types that is wanted
bStatus = QTM_SetFrameTypes(QTM_FrameTypeUnidentified3D Or _
    QTM_FrameTypeIdentified3D Or _
    QTM_FrameType6DOF Or _
    QTM_FrameTypeRotationMatrix)

' Allocate memory for the largest possible number of markers/bodies
bStatus = QTM_AllocateFrameMemoryUnidentified3D(moPPUFrame, 50)
bStatus = QTM_AllocateFrameMemoryIdentified3D(moPPUFrame, 50)
bStatus = QTM_AllocateFrameMemory6DOF(moPPUFrame, 10)

```

```

        bStatus = QTM_AllocateFrameMemoryRotationMatrix(moPPUFrame, 10)

        ' Start real-time data transfer from QTM
        Call QTM_ContinuousTransmission(True)

    Else
        MsgBox "Cannot open TCP/IP or PortA connection"
        Exit Sub
    End If

'    tmrWatchDogTCP.Enabled = True
End If

'start RMS strip chart
startGraph

ClearGridData
ClearGraphData
boolGridCleared = True
boolGraphCleared = True

picFrameDropped.Enabled = False
picFrameDropped.Visible = False

End Sub
'-----
'getReading
' -Reads data from Qualisys and passes it to
' appropriate objects
'-----
Public Sub getReading()
    Dim returnVal As Long
    Dim data As Byte

    timeBeginPeriod 1
    Sleep SLEEP_SETTLING_TIME

    dwFrameCount = -1
    m_boolFrameDropped = False
    m_dwPreviousFrame = -1

    While (isStart)
        bStatus = QTM_HasNewData()

'        While (bStatus = 0)
'            bStatus = QTM_HasNewData()
'            OutPort DAC6, limit(((ErrRMS - MinRes) / (MaxRes - MinRes)) * 65535)
'            OutPort DAC14, limit(((ErrRMS - MinRes) / (MaxRes - MinRes)) * 65535)
'        Wend

        If (bStatus = 1) Then
            bStatus = QTM_GetFrameVB(moPPUFrameVB, moPPUFrame)
            If (bStatus = 1) Then
                dwFrameCount = dwFrameCount + 1

                If (m_boolUpdateGrid) Then

```

```

        DisplayData moPPUFrameVB
        boolGridCleared = False
    Else
        If (Not boolGridCleared) Then
            ClearGridData
            boolGridCleared = True
        End If
    End If

    If (m_boolUpdateGraph) Then
        With moPPUFrameVB.frame6DOF
            If (.numberOfBodies > 0) Then
                graphData .frameBody(0).residual, .frameBody(.numberOfBodies - 1).residual
            End If
        End With

        boolGraphCleared = False
    Else

        If (Not boolGraphCleared) Then
            ClearGraphData
            boolGraphCleared = True
        End If
    End If

    'If ((dwFrameCount Mod FRAME_RATE_WINDOW_MODULUS) =
    (FRAME_RATE_WINDOW_MODULUS - 1)) Then
        QueryPerformanceCounter finishCount
        vbalGridInfo.CellDetails 2, 5, Format$((dwFrameCount Mod
    FRAME_RATE_WINDOW_MODULUS) / ((finishCount - startCount) / curFreq), "#00"), DT_RIGHT
    'End If

    If ((dwFrameCount Mod FRAME_RATE_WINDOW_MODULUS) = 0) Then
        QueryPerformanceCounter startCount
    End If

    With moPPUFrameVB.frame6DOF
        If (Not m_boolFrameDropped) Then
            If (m_dwPreviousFrame > -1) Then
                If ((.frameCount - m_dwPreviousFrame) < 2) Then
                    m_dwPreviousFrame = .frameCount
                Else
                    picFrameDropped.Visible = True
                    picFrameDropped.Enabled = True
                    m_boolFrameDropped = True
                End If
            Else
                m_dwPreviousFrame = .frameCount
            End If
        End If
    End With

    End If
End With

Else
    OutPort DAC6, limit(((ErrRMS - MinRes) / (MaxRes - MinRes)) * 65535)
    OutPort DAC14, limit(((ErrRMS - MinRes) / (MaxRes - MinRes)) * 65535)

```

```

    End If

Else
    OutPort DAC6, limit(((ErrRMS - MinRes) / (MaxRes - MinRes)) * 65535)
    OutPort DAC14, limit(((ErrRMS - MinRes) / (MaxRes - MinRes)) * 65535)
End If

DoEvents
Sleep SLEEP_MIN
Wend

timeEndPeriod 1

'poll for incoming data
errA = SEAMAC_GetBufferCount(handleA, RxCount, TxCount)

'if data available
If RxCount > 0 And errA = 0 Then
    'clear entire buffer
    While RxCount > 0 And errA = 0
        errA = SEAMAC_GetData(handleA, RxBufferA, PortInfoA.wRxFrameSize, retbytes)
        errA = SEAMAC_GetBufferCount(handleA, RxCount, TxCount)
    Wend
    DoEvents
Else
    DoEvents
    Exit Sub
End If

If errA = 0 Then
    'convert data to array of bytes
    byteBuf = StrConv(RxBufferA, vbFromUnicode)

    'clear the log string
    logString = ""

    '1st part of buffer has marker data
    returnVal = f3.passData(byteBuf)

    'if no error in frame
    If returnVal > -1 Then
        'calculate separation point
        split = returnVal + 1

    'if frame in error
    Else
        'set RMS high on error
        OutPort DAC6, limit(((ErrRMS - MinRes) / (MaxRes - MinRes)) * 65535)
        OutPort DAC14, limit(((ErrRMS - MinRes) / (MaxRes - MinRes)) * 65535)

        droppedFrames = droppedFrames + 1
        lblDropped.Caption = droppedFrames
        Exit Sub
    End If

    '2nd part has body data

```

```

returnVal = f5.passData(MidB(byteBuf, split))

'write to log file
If fileOpen Then
    fileStream.WriteLine logString
End If

'if frame in error
If returnVal < 0 Then
    'set RMS high on error
    OutPort DAC6, limit(((ErrRMS - MinRes) / (MaxRes - MinRes)) * 65535)
    OutPort DAC14, limit(((ErrRMS - MinRes) / (MaxRes - MinRes)) * 65535)

    droppedFrames = droppedFrames + 1
    lblDropped.Caption = droppedFrames
    Exit Sub
End If
End If

End Sub

'-----
'DisplayData
' -Display tracking data
' -The first 4 .CellDetails parameters are row,
'   col, text, textAlignment
'-----

Private Sub DisplayData(ByRef moPPUFrameVB As QTM_TPPUFrame)

    Dim dwRow As Long
    Dim dwIdx As Long

    With vbalGridBody
        'Turn off Redraw until grid is fully updated (reduces flicker)
        .Redraw = False
        .Clear

        'Display 6DOF tracking data
        dwRow = 0

        If (moPPUFrameVB.frame6DOF.numberOfBodies > 0) Then
            dwRow = dwRow + 1

            .AddRow

            .CellDetails dwRow, 1, "x", DT_RIGHT
            .CellDetails dwRow, 2, "y", DT_RIGHT
            .CellDetails dwRow, 3, "z", DT_RIGHT
            .CellDetails dwRow, 4, "roll", DT_RIGHT
            .CellDetails dwRow, 5, "pitch", DT_RIGHT
            .CellDetails dwRow, 6, "yaw", DT_RIGHT
            .CellDetails dwRow, 7, "residual", DT_RIGHT
            ' Call SetRowBackgroundColor(dwRow, HEADER_COLOR)

```

```

dwRow = dwRow + 1

For dwIdx = 0 To (moPPUFrameVB.frame6DOF.numberOfBodies - 1)
  If (moPPUFrameVB.frame6DOF.frameBody(dwIdx).residual > 0) Then
    .AddRow

    .CellDetails (dwRow + dwIdx), 1, Format$(moPPUFrameVB.frame6DOF.frameBody(dwIdx).x,
"#0.0000"), DT_RIGHT
    .CellDetails (dwRow + dwIdx), 2, Format$(moPPUFrameVB.frame6DOF.frameBody(dwIdx).y,
"#0.0000"), DT_RIGHT
    .CellDetails (dwRow + dwIdx), 3, Format$(moPPUFrameVB.frame6DOF.frameBody(dwIdx).Z,
"#0.0000"), DT_RIGHT
    .CellDetails (dwRow + dwIdx), 4,
Format$(moPPUFrameVB.frame6DOF.frameBody(dwIdx).roll, "#0.0000"), DT_RIGHT
    .CellDetails (dwRow + dwIdx), 5,
Format$(moPPUFrameVB.frame6DOF.frameBody(dwIdx).pitch, "#0.0000"), DT_RIGHT
    .CellDetails (dwRow + dwIdx), 6,
Format$(moPPUFrameVB.frame6DOF.frameBody(dwIdx).yaw, "#0.0000"), DT_RIGHT
    .CellDetails (dwRow + dwIdx), 7,
Format$(moPPUFrameVB.frame6DOF.frameBody(dwIdx).residual, "#0.0000"), DT_RIGHT

    If fileOpen Then
      fileStream.Write (" " & (dwIdx + 1) & vbTab & Format$((dwFrameCount Mod
FRAME_RATE_WINDOW_MODULUS) / ((finishCount - startCount) / curFreq), "#00") & " " & vbTab & _
      Format$(moPPUFrameVB.frame6DOF.frameBody(dwIdx).x, "#0.0000") & " " & _
      Format$(moPPUFrameVB.frame6DOF.frameBody(dwIdx).y, "#0.0000") & " " & _
      Format$(moPPUFrameVB.frame6DOF.frameBody(dwIdx).Z, "#0.0000") & " " & _
      Format$(moPPUFrameVB.frame6DOF.frameBody(dwIdx).roll, "#0.0000") & " " & _
      Format$(moPPUFrameVB.frame6DOF.frameBody(dwIdx).pitch, "#0.0000") & " " & _
      Format$(moPPUFrameVB.frame6DOF.frameBody(dwIdx).yaw, "#0.0000") & " " & _
      Format$(moPPUFrameVB.frame6DOF.frameBody(dwIdx).residual, "#0.0000")
    End If
  Else
    .AddRow

    .CellDetails (dwRow + dwIdx), 1, "", DT_RIGHT
    .CellDetails (dwRow + dwIdx), 2, "", DT_RIGHT
    .CellDetails (dwRow + dwIdx), 3, "", DT_RIGHT
    .CellDetails (dwRow + dwIdx), 4, "", DT_RIGHT
    .CellDetails (dwRow + dwIdx), 5, "", DT_RIGHT
    .CellDetails (dwRow + dwIdx), 6, "", DT_RIGHT
    .CellDetails (dwRow + dwIdx), 7, "", DT_RIGHT

    If fileOpen Then
      fileStream.Write (vbTab & vbTab & vbTab & vbTab & vbTab & vbTab & vbTab & vbTab &
vbTab & vbTab & vbTab)
    End If
  End If

Next

If fileOpen Then
  fileStream.Write (vbNewLine)
End If

dwRow = dwRow + dwIdx

```

```

Else
    dwRow = dwRow + 1
End If

'Turn on Redraw after grid is fully updated (reduces flicker)
.Redraw = True
End With

With vbalGridMrks
'Turn off Redraw until grid is fully updated (reduces flicker)
.Redraw = False
.Clear

'Display Identified3D tracking data
dwRow = 0

If (moPPUFrameVB.frameIdentified3D.numberOfWorkers > 0) Then
    dwRow = dwRow + 1

    .AddRow
    '.AddRow

    .CellDetails dwRow, 1, "bodyID", DT_RIGHT
    .CellDetails dwRow, 2, "x", DT_RIGHT
    .CellDetails dwRow, 3, "y", DT_RIGHT
    .CellDetails dwRow, 4, "z", DT_RIGHT
    .CellDetails dwRow, 5, "residual", DT_RIGHT
    ' Call SetRowBackgroundColor(dwRow, HEADER_COLOR)

    dwRow = dwRow + 1

    For dwIdx = 0 To (moPPUFrameVB.frameIdentified3D.numberOfWorkers - 1)
        .AddRow

        .CellDetails (dwRow + dwIdx), 1,
Format$(moPPUFrameVB.frameIdentified3D.frameBody(dwIdx).bodyID, "#0.0000"), DT_RIGHT
        .CellDetails (dwRow + dwIdx), 2,
Format$(moPPUFrameVB.frameIdentified3D.frameBody(dwIdx).x, "#0.0000"), DT_RIGHT
        .CellDetails (dwRow + dwIdx), 3,
Format$(moPPUFrameVB.frameIdentified3D.frameBody(dwIdx).y, "#0.0000"), DT_RIGHT
        .CellDetails (dwRow + dwIdx), 4,
Format$(moPPUFrameVB.frameIdentified3D.frameBody(dwIdx).Z, "#0.0000"), DT_RIGHT
        .CellDetails (dwRow + dwIdx), 5,
Format$(moPPUFrameVB.frameIdentified3D.frameBody(dwIdx).residual, "#0.0000"), DT_RIGHT
        Next

        dwRow = dwRow + dwIdx
    Else
        dwRow = dwRow + 1
    End If

    .AddRow

'Display Unidentified3D tracking data

```

```

'   dwRow = dwRow + 2
'
'   .AddRow
'   .AddRow
'
'   .CellDetails dwRow, 1, "Unidentified3D"
'   'Call SetRowBackgroundColor(dwRow, HEADER_COLOR)
'
'   dwRow = dwRow + 2
'
'   .AddRow
'   .AddRow
'
'   .CellDetails dwRow, 1, "Frame Count"
'   .CellDetails dwRow, 2, "Markers"
'
'   dwRow = dwRow + 1
'
'   .AddRow
'
'   .CellDetails dwRow, 1, moPPUFrameVB.frameUnidentified3D.frameCount
'   .CellDetails dwRow, 2, moPPUFrameVB.frameUnidentified3D.numberOfMarkers
'
'   If (moPPUFrameVB.frameUnidentified3D.numberOfMarkers > 0) Then
'       dwRow = dwRow + 2
'
'       .AddRow
'       .AddRow
'
'       .CellDetails dwRow, 1, "x", DT_RIGHT
'       .CellDetails dwRow, 2, "y", DT_RIGHT
'       .CellDetails dwRow, 3, "z", DT_RIGHT
'       .CellDetails dwRow, 4, "residual", DT_RIGHT
'
'       dwRow = dwRow + 1
'
'       For dwIdx = 0 To (moPPUFrameVB.frameUnidentified3D.numberOfMarkers - 1)
'           .AddRow
'
'           .CellDetails (dwRow + dwIdx), 1,
Format$(moPPUFrameVB.frameUnidentified3D.frameBody(dwIdx).X, "#0.00000"), DT_RIGHT
'           .CellDetails (dwRow + dwIdx), 2,
Format$(moPPUFrameVB.frameUnidentified3D.frameBody(dwIdx).Y, "#0.00000"), DT_RIGHT
'           .CellDetails (dwRow + dwIdx), 3,
Format$(moPPUFrameVB.frameUnidentified3D.frameBody(dwIdx).z, "#0.00000"), DT_RIGHT
'           .CellDetails (dwRow + dwIdx), 4,
Format$(moPPUFrameVB.frameUnidentified3D.frameBody(dwIdx).residual, "#0.00000"), DT_RIGHT
'       Next
'   End If
'
'   'Turn on Redraw after grid is fully updated (reduces flicker)
'   .Redraw = True
End With

With vbalGridInfo
'   'Turn off Redraw until grid is fully updated (reduces flicker)

```

```

.Redraw = False
.Clear

'Display General tracking data
dwRow = 1

.AddRow

.CellDetails dwRow, 1, "Frame Count", DT_RIGHT
.CellDetails dwRow, 2, "Bodies", DT_RIGHT
.CellDetails dwRow, 3, "Identified", DT_RIGHT
.CellDetails dwRow, 4, "Unidentified", DT_RIGHT
.CellDetails dwRow, 5, "Frequency", DT_RIGHT
' Call SetRowBackgroundColor(dwRow, HEADER_COLOR)

dwRow = dwRow + 1

.AddRow

.CellDetails dwRow, 1, moPPUFrameVB.frame6DOF.frameCount, DT_RIGHT
.CellDetails dwRow, 2, moPPUFrameVB.frame6DOF.numberOfBodies, DT_RIGHT
.CellDetails dwRow, 3, moPPUFrameVB.frameIdentified3D.numberOfMarkers, DT_RIGHT
.CellDetails dwRow, 4, moPPUFrameVB.frameUnidentified3D.numberOfMarkers, DT_RIGHT

'Turn on Redraw after grid is fully updated (reduces flicker)
.Redraw = True
End With

End Sub

'Private Sub SetRowBackgroundColor(ByVal dwRow As Long, ByVal dwColor As Long)
' Dim dwCol As Long
'
'
' With vbalGridBody
' For dwCol = 1 To .Columns
' .CellBackColor(dwRow, dwCol) = dwColor
' Next dwCol
' End With
'
' With vbalGridMrks
' For dwCol = 1 To .Columns
' .CellBackColor(, dwCol) = dwColor
' Next dwCol
' End With
'
' With vbalGridInfo
' For dwCol = 1 To .Columns
' .CellBackColor(dwRow, dwCol) = dwColor
' Next dwCol
' End With
'End Sub

```

```

'-----
'Stopbutton
' ends main data stream loop
'-----
Private Sub cmdStop_Click()
' Stop the logging
If fileOpen Then
cmdLog_Click
End If

mnuTools.Enabled = True
Sleep SLEEP_SETTLING_TIME
QTMClientVB.QTM_ContinuousTransmission (False)

cmdStop.Visible = False
cmdStart.Visible = True
isStart = False

End Sub

'-----
'commStop
' Close ports and stop data stream
'-----
Private Sub commStop()
isOpen = False
Set fileStream = Nothing
fileOpen = False
curLastCall = 0
curNow = 0
cmdLog.Enabled = False

QTMClientVB.QTM_ContinuousTransmission (False)
bStatus = QTM_FreeFrameMemoryRotationMatrix(moPPUFrame)
bStatus = QTM_FreeFrameMemory6DOF(moPPUFrame)
bStatus = QTM_FreeFrameMemoryIdentified3D(moPPUFrame)
bStatus = QTM_FreeFrameMemoryUnidentified3D(moPPUFrame)

QTMClientVB.QTM_CloseConnection

' tmrWatchDogTCP.Enabled = False
'Stop Qualisys Data Stream
str = "#B" & vbCr 'reset buffers and stop transmission
errA = SEAMAC_PutData(handleA, str, Len(str))

'Close portA
errA = SEAMAC_PortClose(handleA)
cmdStart.Caption = "Start"

'Close portB
errB = SEAMAC_PortClose(handleB)

tmrWatchDog.Enabled = False
End Sub

```

```

'-----
'Form_Unload
' perform necessary cleanup
'-----
Private Sub Form_Unload(Cancel As Integer)
'clean up objects
Set f3 = Nothing
Set f5 = Nothing
'close ports
If isOpen Then
errA = SEAMAC_PortClose(handleA)
errB = SEAMAC_PortClose(handleB)

bStatus = QTMClientVB.QTM_CloseConnection()
If (bStatus = 1) Then
isOpen = False
isStart = False
End If

End If

End Sub

'-----
'mnuCalibrate_Click
' Displays the calibration dialog
'-----
Private Sub mnuCalibrate_Click()
frmDAC.Visible = True
End Sub

'-----
'mnuConfig
' Initiates configuration file utility
'-----
Private Sub mnuConfig_Click()
frmConfig.Visible = True
End Sub

'-----
'Log button
' Creates text file for logging
'-----
Private Sub cmdLog_Click()
If Not fileOpen Then
Set fileStream = fileObj.CreateTextFile(txtLog.Text, True, False)
'Set fileStream = fileObj.OpenTextFile(txtLog.Text, ForAppending, True)
fileStream.WriteLine("# Log file created by QualiDAC at " & Now & vbNewLine & vbNewLine & _
" ID" & " Frequency" & " x" & vbTab & " y" & vbTab & _
vbTab & "z" & vbTab & " roll" & vbTab & " pitch" & vbTab & "yaw" & vbTab & "
RMS")
fileOpen = True
If selspot Then
Dim fitFileName As String
fitFileName = left(txtLog.Text, InStrRev(txtLog.Text, ".") & "fit"
Set fitFileStream = fileObj.OpenTextFile(fitFileName, ForAppending, True)

```

```

        fitFileOpen = True
    End If
    cmdLog.Caption = "Stop Log"
Else
    fileStream.Close
    Set fileStream = Nothing
    If selspot Then
        fitFileStream.Close
        Set fitFileStream = Nothing
        fitFileOpen = False
    End If
    cmdLog.Caption = "Start Log"
    fileOpen = False
End If
End Sub

Private Sub mnuProcessFile_Click()
    CommonDialog.Filter = "Log File (*.log,*.txt)|*.log;*.txt;"
    CommonDialog.ShowOpen

    Dim F As Integer
    F = FreeFile
    startGraph
    Open CommonDialog.filename For Input As #F
    If Err.Number = 0 Then
        Dim line As String
        Do While Not EOF(F)
            Line Input #F, line
            If InStr(line, "#") <> 1 Then
                logString = ""
                f3.passDataFromFile line
            End If
            DoEvents
        Loop
    End If
End Sub

Private Sub Selspot_Check_Click()
    If Selspot_Check.Value Then
        CommonDialog.Filter = "Body File (*.mdl,*.bod)|*.mdl;*.bod;"
        CommonDialog.ShowOpen
        If Not LoadBodyFile(CommonDialog.filename) Then
            Selspot_Check.Value = False
        Else
            TrackInit
        End If
    End If
End Sub

selspot = Selspot_Check.Value
End Sub

```

```

'-----
'tmrWatchDog_Timer
' checks to see if frame stream has stopped, if so
' resets Qualisys
'-----

Private Sub tmrWatchDog_Timer()
    Dim delayStart As Single

    'if frame count has not increased
    If (f5.getFrameCount = 0) Then

        watchdogcount = watchdogcount + 1
        lblStatus.Caption = "Qualisys Reset: " & CStr(watchdogcount) & " at " & CInt((curNow - curStartTime) /
curFreq) & " sec."

        'Set RMS high durring reset
        OutPort DAC6, limit(((ErrRMS - MinRes) / (MaxRes - MinRes)) * 65535)
        OutPort DAC14, limit(((ErrRMS - MinRes) / (MaxRes - MinRes)) * 65535)

        'reset buffers and stop transmission
        str = "#B" & vbCr
        errA = SEAMAC_PutData(handleA, str, Len(str))

        'Need to delay 200ms after reset
        delayStart = Timer
        While Timer < delayStart + 0.2
            Wend

            str = "#f000A" & vbCr '6D and 3D data
            errA = SEAMAC_PutData(handleA, str, Len(str))

            str = "#Tffff" & vbCr 'transmit continuous
            errA = SEAMAC_PutData(handleA, str, Len(str))

            str = "#Gffff" & vbCr 'start data collection
            errA = SEAMAC_PutData(handleA, str, Len(str))
        End If
    End Sub

Private Sub chkUpdateGrid_Click()
    m_boolUpdateGrid = chkUpdateGrid.Value
End Sub

Private Sub chkUpdateGraph_Click()
    m_boolUpdateGraph = chkUpdateGraph.Value
End Sub

Private Sub picFrameDropped_Click()
    picFrameDropped.Enabled = False
    picFrameDropped.Visible = False

    m_dwPreviousFrame = -1

    m_boolFrameDropped = False
End Sub

```

```
*****Configuration form*****
```

```
Public cfgFileObj As New FileSystemObject  
Public cfgFileStream As Scripting.TextStream  
Dim loadedFilename As String
```

```
'-----  
'Form_Load
```

```
' initialize variables and form objects  
'-----
```

```
Private Sub Form_Load()
```

```
    Main.Enabled = False  
    cmdOpen.Enabled = True  
    cmdNew.Enabled = True  
    cmdSave.Enabled = False  
    cmdLoad.Enabled = False  
    loadFile openedFilename
```

```
End Sub
```

```
'-----  
'Load current config file  
'-----
```

```
Private Sub cmdLoad_Click()
```

```
    Dim success As Boolean  
  
    success = initializeConfig(fileDialog.filename)
```

```
    If success Then  
        Unload Me  
    End If
```

```
End Sub
```

```
'-----  
'create new config file  
'-----
```

```
Private Sub cmdNew_Click()
```

```
    cmdLoad.Enabled = False  
  
    txtConfig.Text = defaultConfig  
  
    cmdOpen.Enabled = True  
    cmdNew.Enabled = True  
    cmdSave.Enabled = True  
    cmdLoad.Enabled = False
```

```
End Sub
```

```
'-----  
'Open existing config file  
'-----
```

```
Private Sub cmdOpen_Click()
```

```
    'generate standard file dialog  
    fileDialog.Filter = "QualiDAC File (*.qdc)*.qdc;"  
    fileDialog.ShowOpen  
    fileDialog.FilterIndex = 1  
    If fileDialog.filename = vbNullString Then Exit Sub
```

```
    loadFile fileDialog.filename  
End Sub
```

```
'-----  
'Loads a file into the configuration window  
'-----
```

```
Private Sub loadFile(filename As String)  
    'open text stream to opened file  
    Set cfgFileStream = cfgFileObj.OpenTextFile(filename, ForReading, False)  
    txtConfig.Text = cfgFileStream.ReadAll  
    Set cfgFileStream = Nothing  
  
    cmdOpen.Enabled = True  
    cmdLoad.Enabled = True  
    cmdNew.Enabled = True  
    cmdSave.Enabled = True  
    Caption = "Configuration - " & Mid(filename, InStrRev(filename, "\") + 1)  
    loadedFilename = filename  
End Sub
```

```
'-----  
'cmdSave  
' saves new or modified configuration file  
'-----
```

```
Private Sub cmdSave_Click()  
    fileDialog.Filter = "QualiDAC File (*.qdc)|*.qdc;"  
    fileDialog.filename = loadedFilename  
    fileDialog.ShowSave  
    fileDialog.FilterIndex = 1  
  
    If fileDialog.filename = vbNullString Then Exit Sub  
  
    Set cfgFileStream = cfgFileObj.OpenTextFile(fileDialog.filename, ForWriting, True)  
    cfgFileStream.Write txtConfig  
    Set cfgFileStream = Nothing  
  
    cmdNew.Enabled = True  
    cmdOpen.Enabled = True  
    cmdSave.Enabled = True  
    cmdLoad.Enabled = True  
End Sub
```

```
'-----  
'initializeConfig  
' Initialize configuration values  
'-----
```

```
Public Function initializeConfig(file As String) As Boolean  
    'error handler  
    On Error GoTo BadConfig  
  
    Dim ret As Boolean  
    ret = True  
  
    'if no file was specified  
    If file = vbNullString Then  
        initializeConfig = False
```

```
Exit Function
End If
```

```
'open configuration file
Set cfgFileStream = cfgFileObj.OpenTextFile(file, ForReading, False)
```

```
'read OutputBody value
If readValue(cfgFileStream.ReadLine) = 0 Then
    outputBody = True
Else
    outputBody = False
End If
```

```
'read BASE address value
BASE = readValue(cfgFileStream.ReadLine)
If BASE < 1 Then
    MsgBox "Invalid BASE Value"
    ret = False
End If
```

```
'set all DAC addresses based on BASE address
AUTO = BASE + 2
```

```
DAC0 = BASE + 0
DAC1 = BASE + 2
DAC2 = BASE + 4
DAC3 = BASE + 6
DAC4 = BASE + 8
DAC5 = BASE + 10
DAC6 = BASE + 12
DAC7 = BASE + 14
DAC8 = BASE + 16
DAC9 = BASE + 18
DAC10 = BASE + 20
DAC11 = BASE + 22
DAC12 = BASE + 24
DAC13 = BASE + 26
DAC14 = BASE + 28
DAC15 = BASE + 30
```

```
'The following sets/verifies the Max/Min values for each of the body signals
' -value is read and assigned to global variable
' -when possible the value is verified
' -if value is erroneous, msgbox is produces and return is set to false
```

```
MaxX = readValue(cfgFileStream.ReadLine)
MinX = readValue(cfgFileStream.ReadLine)
If MaxX <= MinX Then
    MsgBox "Invalid X Value"
    ret = False
End If
```

```
MaxY = readValue(cfgFileStream.ReadLine)
MinY = readValue(cfgFileStream.ReadLine)
If MaxY <= MinY Then
    MsgBox "Invalid Y Value"
```

```

    ret = False
End If

MaxZ = readValue(cfgFileStream.ReadLine)
MinZ = readValue(cfgFileStream.ReadLine)
If MaxZ <= MinZ Then
    MsgBox "Invalid Z Value"
    ret = False
End If

MaxR = readValue(cfgFileStream.ReadLine)
MinR = readValue(cfgFileStream.ReadLine)
If MaxR <= MinR Then
    MsgBox "Invalid Roll Value"
    ret = False
End If

MaxP = readValue(cfgFileStream.ReadLine)
MinP = readValue(cfgFileStream.ReadLine)
If MaxP <= MinP Then
    MsgBox "Invalid Pitch Value"
    ret = False
End If

MaxYw = readValue(cfgFileStream.ReadLine)
MinYw = readValue(cfgFileStream.ReadLine)
If MaxYw <= MinYw Then
    MsgBox "Invalid Yaw Value"
    ret = False
End If

MaxRes = readValue(cfgFileStream.ReadLine)
MinRes = readValue(cfgFileStream.ReadLine)
If MaxRes <= MinRes Then
    MsgBox "Invalid RMS Value"
    ret = False
End If

ErrRMS = readValue(cfgFileStream.ReadLine)

If ErrRMS > MaxRes Then
    ErrRMS = MaxRes
ElseIf ErrRMS < MinRes Then
    ErrRMS = MinRes
End If

'sets/verifies angle hysteresis size
hysteresis = readValue(cfgFileStream.ReadLine)
If hysteresis < 1 Then
    MsgBox "Invalid Hysteresis Value"
    ret = False
End If

'sets/verifies graph vertical scale values
GraphMax = readValue(cfgFileStream.ReadLine)
GraphMin = readValue(cfgFileStream.ReadLine)

```

```

If GraphMax <= GraphMin Then
    MsgBox "Invalid Graph Value"
    ret = False
End If

GraphPoints = readValue(cfgFileStream.ReadLine)
If GraphPoints < 1 Then
    MsgBox "Invalid Graph Graph Value"
    ret = False
End If

'close config file
Set cfgFileStream = Nothing

If ret = True Then
    SaveSetting "QualiDAC", "Config", "configFile", file
    Main.Caption = "QualiDAC - " & Mid(file, InStrRev(file, "\") + 1)
    openedFilename = file
End If

initializeConfig = ret
Exit Function

'Error handler
BadConfig:
    initializeConfig = False
    MsgBox "Bad Configuration File"
End Function
'-----

'readValue
' used with initializeConfig function
' returns the value found at each line of the config file
'-----
Private Function readValue(line As String) As Long
    Dim endPt As Integer
    Dim valStr As String

    endPt = InStr(line, "#")
    valStr = RTrim$(Mid$(line, 1, endPt - 1))
    readValue = CLng(valStr)
End Function

'-----
'Form_Unload
' re-enables main form
'-----
Private Sub Form_Unload(Cancel As Integer)
    Main.Enabled = True
End Sub

```

'\*\*\*\*\*DAC form\*\*\*\*\*'

Private Value As Long

```
'-----  
'cmdChan  
' when selected updates the mas/min display  
'-----
```

Private Sub cmbChan\_Click()

```
lblMax.Caption = "Max: "  
lblMin.Caption = "Min: "  
Select Case cmbChan.ListIndex  
Case 0  
    lblMax.Caption = lblMax.Caption + CStr(MaxX)  
    lblMin.Caption = lblMin.Caption + CStr(MinX)  
Case 1  
    lblMax.Caption = lblMax.Caption + CStr(MaxY)  
    lblMin.Caption = lblMin.Caption + CStr(MinY)  
Case 2  
    lblMax.Caption = lblMax.Caption + CStr(MaxZ)  
    lblMin.Caption = lblMin.Caption + CStr(MinZ)  
Case 3  
    lblMax.Caption = lblMax.Caption + CStr(MaxR)  
    lblMin.Caption = lblMin.Caption + CStr(MinR)  
Case 4  
    lblMax.Caption = lblMax.Caption + CStr(MaxP)  
    lblMin.Caption = lblMin.Caption + CStr(MinP)  
Case 5  
    lblMax.Caption = lblMax.Caption + CStr(MaxYw)  
    lblMin.Caption = lblMin.Caption + CStr(MinYw)  
Case 6  
    lblMax.Caption = lblMax.Caption + CStr(MaxRes)  
    lblMin.Caption = lblMin.Caption + CStr(MinRes)  
Case Else  
End Select  
End Sub
```

```
'-----  
'cmdSend  
' generates the appropriate output for the  
' selected channel  
'-----
```

Private Sub cmdSend\_Click()

```
Dim DACErr As Integer  
Value = CLng(txtValue.Text)  
  
Select Case cmbChan.ListIndex  
Case 0  
    If Value > MaxX Or Value < MinX Then  
        MsgBox "Invalid Value", vbOKOnly  
        Exit Sub  
    End If  
    OutPort DAC0, ((Value - MinX) / (MaxX - MinX)) * 65535  
    OutPort DAC8, ((Value - MinX) / (MaxX - MinX)) * 65535  
Case 1
```

```

    If Value > MaxY Or Value < MinY Then
        MsgBox "Invalid Value", vbOKOnly
        Exit Sub
    End If
    OutPort DAC1, ((Value - MinY) / (MaxY - MinY)) * 65535
    OutPort DAC9, ((Value - MinY) / (MaxY - MinY)) * 65535
Case 2
    If Value > MaxZ Or Value < MinZ Then
        MsgBox "Invalid Value", vbOKOnly
        Exit Sub
    End If
    OutPort DAC2, ((Value - MinZ) / (MaxZ - MinZ)) * 65535
    OutPort DAC10, ((Value - MinZ) / (MaxZ - MinZ)) * 65535
Case 3
    If Value > MaxR Or Value < MinR Then
        MsgBox "Invalid Value", vbOKOnly
        Exit Sub
    End If
    OutPort DAC3, ((Value - MinR) / (MaxR - MinR)) * 65535
    OutPort DAC11, ((Value - MinR) / (MaxR - MinR)) * 65535
Case 4
    If Value > MaxP Or Value < MinP Then
        MsgBox "Invalid Value", vbOKOnly
        Exit Sub
    End If
    OutPort DAC4, ((Value - MinP) / (MaxP - MinP)) * 65535
    OutPort DAC12, ((Value - MinP) / (MaxP - MinP)) * 65535
Case 5
    If Value > MaxYw Or Value < MinYw Then
        MsgBox "Invalid Value", vbOKOnly
        Exit Sub
    End If
    OutPort DAC5, ((Value - MinYw) / (MaxYw - MinYw)) * 65535
    OutPort DAC13, ((Value - MinYw) / (MaxYw - MinYw)) * 65535
Case 6
    If Value > MaxRes Or Value < MinRes Then
        MsgBox "Invalid Value", vbOKOnly
        Exit Sub
    End If
    OutPort DAC7, ((Value - MinRes) / (MaxRes - MinRes)) * 65535
    OutPort DAC15, ((Value - MinRes) / (MaxRes - MinRes)) * 65535
Case Else
End Select

```

End Sub

```

'-----
'FormLoad
' initializes the list box
'-----

```

```

Private Sub Form_Load()
    Main.Enabled = False
    cmbChan.AddItem "X", 0
    cmbChan.AddItem "Y", 1
    cmbChan.AddItem "Z", 2
    cmbChan.AddItem "Roll", 3

```

```
cmbChan.AddItem "Pitch", 4  
cmbChan.AddItem "Yaw", 5  
cmbChan.AddItem "RMS", 6
```

```
End Sub
```

```
'-----  
'FormUnload  
' re-enables the main form  
'-----
```

```
Private Sub Form_Unload(Cancel As Integer)  
    Main.Enabled = True  
End Sub
```

```
'-----  
' Simulation Module  
'
```

```
' Author:  
' Wayne Pearson  
' Software Engineering Group, IOT  
'-----
```

```
Option Explicit
```

```
Public m_dwFrameCount As Long  
Public m_dwMarkerCount As Long  
Public m_dwBodyCount As Long
```

```
Public Function QTM_GetFrameVBT(ByRef moPPUFrameVB As QTM_TPPUFrame, ByRef moPPUFrame  
As QTM_TPPUFrameC) As Byte  
    Dim dwIdx As Long
```

```
    ZeroMemory moPPUFrameVB, LenB(moPPUFrameVB)
```

```
    QTM_GetFrameVBT = 1  
    If (QTM_GetFrameVBT = 1) Then  
        moPPUFrameVB.soh = 0  
        moPPUFrameVB.size = 0  
        moPPUFrameVB.frameType = 0  
        'moPPUFrameVB.frameError =  
        'moPPUFrameVB.frameCalibration =  
        'moPPUFrameVB.frameVerification =  
        moPPUFrameVB.frame6DOF.frameCount = m_dwFrameCount  
        moPPUFrameVB.frame6DOF.numberOfBodies = m_dwBodyCount
```

```
    If (m_dwBodyCount > 0) Then  
        ReDim moPPUFrameVB.frame6DOF.frameBody(m_dwBodyCount - 1)
```

```
        For dwIdx = 0 To (m_dwBodyCount - 1)  
            moPPUFrameVB.frame6DOF.frameBody(dwIdx).x = (1 + (dwIdx * 10))  
            moPPUFrameVB.frame6DOF.frameBody(dwIdx).y = (2 + (dwIdx * 10))  
            moPPUFrameVB.frame6DOF.frameBody(dwIdx).Z = (3 + (dwIdx * 10))  
            moPPUFrameVB.frame6DOF.frameBody(dwIdx).roll = (4 + (dwIdx * 10))  
            moPPUFrameVB.frame6DOF.frameBody(dwIdx).pitch = (5 + (dwIdx * 10))  
            moPPUFrameVB.frame6DOF.frameBody(dwIdx).yaw = (6 + (dwIdx * 10))  
            moPPUFrameVB.frame6DOF.frameBody(dwIdx).residual = (7 + (dwIdx * 10))  
            moPPUFrameVB.frame6DOF.frameBody(dwIdx).events = (8 + (dwIdx * 10))
```

```
        Next  
    End If
```

```
    moPPUFrameVB.frameIdentified3D.frameCount = m_dwFrameCount  
    moPPUFrameVB.frameIdentified3D.numberOfMarkers = m_dwMarkerCount
```

```
    If (m_dwMarkerCount > 0) Then  
        ReDim moPPUFrameVB.frameIdentified3D.frameBody(m_dwMarkerCount - 1)
```

```
        For dwIdx = 0 To (m_dwMarkerCount - 1)  
            moPPUFrameVB.frameIdentified3D.frameBody(dwIdx).bodyID = (1 + (dwIdx * 10) + 100)  
            moPPUFrameVB.frameIdentified3D.frameBody(dwIdx).x = (2 + (dwIdx * 10) + 100)  
            moPPUFrameVB.frameIdentified3D.frameBody(dwIdx).y = (3 + (dwIdx * 10) + 100)
```

```

        moPPUFrameVB.frameIdentified3D.frameBody(dwIdx).Z = (4 + (dwIdx * 10) + 100)
        moPPUFrameVB.frameIdentified3D.frameBody(dwIdx).residual = (5 + (dwIdx * 10) + 100)
    Next
End If

moPPUFrameVB.frameUnidentified3D.frameCount = m_dwFrameCount
moPPUFrameVB.frameUnidentified3D.numberofMarkers = m_dwMarkerCount

If (m_dwMarkerCount > 0) Then
    ReDim moPPUFrameVB.frameUnidentified3D.frameBody(m_dwMarkerCount - 1)

    For dwIdx = 0 To (m_dwMarkerCount - 1)
        moPPUFrameVB.frameUnidentified3D.frameBody(dwIdx).x = (1 + (dwIdx * 10) + 200)
        moPPUFrameVB.frameUnidentified3D.frameBody(dwIdx).y = (2 + (dwIdx * 10) + 200)
        moPPUFrameVB.frameUnidentified3D.frameBody(dwIdx).Z = (3 + (dwIdx * 10) + 200)
        moPPUFrameVB.frameUnidentified3D.frameBody(dwIdx).residual = (4 + (dwIdx * 10) + 200)
    Next
End If

'moPPUFrameVB.frameIdentified3D = GetQTM_TFrameIdentified3D(moPPUFrame)
'moPPUFrameVB.frame6DOF = GetQTM_TFrame6DOF(moPPUFrame)
'moPPUFrameVB.frameRotation = GetQTM_TFrameRotationMatrix(moPPUFrame)
moPPUFrameVB.checksum = 0
moPPUFrameVB.terminator = 0
End If

m_dwFrameCount = m_dwFrameCount + 1
m_dwMarkerCount = 4
m_dwBodyCount = 4
End Function

```

## Option Explicit

```
////////////////////////////////////////////////////////////////  
// 32 BIT PEGRP32B API FUNCTIONS AND DEFINES //  
// For VB /32 bit //  
// Copyright (c) 1996-2003 Gigasoft, Inc. //  
////////////////////////////////////////////////////////////////
```

```
' ** NOTE, this is a partial list of constants.  
' If the constant you need can not be found here,  
' look in PEGRPAPI.H which is the true header file  
' for the DLL.
```

```
Global Const FIRST_DEFAULT_TAB = 0
```

```
Global Const PEDP_ENABLED = 0  
Global Const PEDP_DISABLED = 1  
Global Const PEDP_INSIDE_TOP = 2
```

```
Global Const PETLT_12HR_AM_PM = 0  
Global Const PETLT_12HR_NO_AM_PM = 1  
Global Const PETLT_24HR = 2
```

```
Global Const PEDLT_3_CHAR = 0  
Global Const PEDLT_1_CHAR = 1  
Global Const PEDLT_NO_DAY_PROMPT = 2  
Global Const PEDLT_NO_DAY_NUMBER = 3
```

```
Global Const PEMLT_3_CHAR = 0  
Global Const PEMLT_1_CHAR = 1  
Global Const PEMLT_NO_MONTH_PROMPT = 2
```

```
Global Const PEHS_HORIZONTAL = 0 /* ---- */  
Global Const PEHS_VERTICAL = 1 /* |||| */  
Global Const PEHS_FDIAGONAL = 2 /* \\\\ */  
Global Const PEHS_BDIAGONAL = 3 /* //// */  
Global Const PEHS_CROSS = 4 /* ++++ */  
Global Const PEHS_DIAGCROSS = 5 /* xxxxx */
```

```
Global Const PEGS_NO_GRADIENT = 0  
Global Const PEGS_VERTICAL = 1  
Global Const PEGS_HORIZONTAL = 2
```

```
Global Const PEBS_NO_BMP = 0  
Global Const PEBS_STRETCHBLT = 1  
Global Const PEBS_TILED_BITBLT = 2  
Global Const PEBS_BITBLT_TOP_LEFT = 3  
Global Const PEBS_BITBLT_TOP_CENTER = 4  
Global Const PEBS_BITBLT_TOP_RIGHT = 5  
Global Const PEBS_BITBLT_BOTTOM_LEFT = 6  
Global Const PEBS_BITBLT_BOTTOM_CENTER = 7  
Global Const PEBS_BITBLT_BOTTOM_RIGHT = 8  
Global Const PEBS_BITBLT_CENTER = 9
```

```
Global Const PEQS_NO_STYLE = 0  
Global Const PEQS_LIGHT_INSET = 1
```

Global Const PEQS\_LIGHT\_SHADOW = 2  
Global Const PEQS\_LIGHT\_LINE = 3  
Global Const PEQS\_LIGHT\_NO\_BORDER = 4  
Global Const PEQS\_MEDIUM\_INSET = 5  
Global Const PEQS\_MEDIUM\_SHADOW = 6  
Global Const PEQS\_MEDIUM\_LINE = 7  
Global Const PEQS\_MEDIUM\_NO\_BORDER = 8  
Global Const PEQS\_DARK\_INSET = 9  
Global Const PEQS\_DARK\_SHADOW = 10  
Global Const PEQS\_DARK\_LINE = 11  
Global Const PEQS\_DARK\_NO\_BORDER = 12

// PEP\_dwDESKCOLOR must be set to 1(PEBG\_TRANSPARENT) to enable the below properties //

Global Const PEP\_dwDESKGRADIENTSTART = 2687  
Global Const PEP\_dwDESKGRADIENTEND = 2688  
Global Const PEP\_nDESKGRADIENTSTYLE = 2689  
Global Const PEP\_szDESKBMPFILENAME = 2690  
Global Const PEP\_nDESKBMPSTYLE = 2691

// PEP\_dwGRAPHBACKCOLOR must be set to 1 (PEBG\_TRANSPARENT) to enable the below properties.

Global Const PEP\_dwGRAPHGRADIENTSTART = 2692  
Global Const PEP\_dwGRAPHGRADIENTEND = 2693  
Global Const PEP\_nGRAPHGRADIENTSTYLE = 2694  
Global Const PEP\_szGRAPHBMPFILENAME = 2695  
Global Const PEP\_nGRAPHBMPSTYLE = 2696

// PEP\_dwTABLEBACKCOLOR must be set to 1 (PEBG\_TRANSPARENT) to enable the below properties.

Global Const PEP\_dwTABLEGRADIENTSTART = 2697  
Global Const PEP\_dwTABLEGRADIENTEND = 2698  
Global Const PEP\_nTABLEGRADIENTSTYLE = 2699  
Global Const PEP\_szTABLEBMPFILENAME = 2700  
Global Const PEP\_nTABLEBMPSTYLE = 2701  
Global Const PEBG\_TRANSPARENT = 1  
Global Const PEP\_bPNGISTRANSSPARENT = 2683  
Global Const PEP\_dwPNGTRANSPARENTCOLOR = 2684  
Global Const PEP\_bPNGISINTERLACED = 2685  
Global Const PEP\_nJPGQUALITY = 2686  
Global Const PEP\_bDISABLE3DSHADOW = 3927  
Global Const PEP\_nDROPSHADOWOFFSETX = 2679  
Global Const PEP\_nDROPSHADOWOFFSETY = 2680  
Global Const PEP\_nDROPSHADOWSTEPS = 2681  
Global Const PEP\_nDROPSHADOWWIDTH = 2682  
Global Const PEP\_nHIDEINTERSECTINGTEXT = 2678  
Global Const PEP\_bSTOP = 2677  
Global Const PEP\_nBITMAPGRADIENTMENU = 2702  
Global Const PEP\_bBITMAPGRADIENTMODE = 2703  
Global Const PEP\_nLONGXAXISTICKMENU = 2674  
Global Const PEP\_nLONGYAXISTICKMENU = 2673  
Global Const PEP\_nQUICKSTYLE = 2672  
Global Const PEP\_nQUICKSTYLEMENU = 2671  
Global Const PEP\_nVIEWINGSTYLEMENU = 2640  
Global Const PEP\_nFONTSIZEMENU = 2641  
Global Const PEP\_nDATAPRECISIONMENU = 2642  
Global Const PEP\_nDATASHADOWMENU = 2643  
Global Const PEP\_bSEPARATORMENU = 2654  
Global Const PEP\_nMAXIMIZEMENU = 2655

Global Const PEP\_nCUSTOMIZEDIALOGMENU = 2656  
Global Const PEP\_nEXPORTDIALOGMENU = 2657  
Global Const PEP\_nHELPMENU = 2658

Global Const PEP\_nBORDERTYPEMENU = 2659  
Global Const PEP\_nSHOWLEGENDMENU = 2660  
Global Const PEP\_nLEGENDLOCATIONMENU = 2661  
Global Const PEP\_nSHOWTABLEANNOTATIONSMENU = 2662  
Global Const PEP\_nMULTIAXISSTYLEMENU = 2663  
Global Const PEP\_nFIXEDFONTMENU = 2664

Global Const PEP\_bSHOWALLTABLEANNOTATIONS = 2665  
Global Const PEP\_bSHOWLEGEND = 2666

Global Const PEP\_naCUSTOMMENU = 2667  
Global Const PEP\_naCUSTOMMENUSTATE = 2668  
Global Const PEP\_naCUSTOMMENULOCATION = 2669  
Global Const PEP\_szaCUSTOMMENUTEXT = 2670  
Global Const PEP\_nLASTMENUINDEX = 2675  
Global Const PEP\_nLASTSUBMENUINDEX = 2676

Global Const PEP\_nGRIDLINEMENU = 3164  
Global Const PEP\_nPLOTMETHODMENU = 3165  
Global Const PEP\_nGRIDINFRONTMENU = 3166  
Global Const PEP\_nTREATCOMPARISONSMENU = 3167  
Global Const PEP\_nMARKDATAPOINTSMENU = 3168  
Global Const PEP\_nSHOWANNOTATIONSMENU = 3169  
Global Const PEP\_nUNDOZOOMMENU = 3170

Global Const PEP\_nGRAPHPLUSTABLEMENU = 3430  
Global Const PEP\_nFORCEVERTPOINTSMENU = 3431  
Global Const PEP\_nTABLEWHATMENU = 3432

Global Const PEP\_nINCLUDEDATALABELSMENU = 3696  
Global Const PEP\_fZOOMMINTX = 3697  
Global Const PEP\_fZOOMMAXTX = 3698

Global Const PEP\_nSHOWBOUNDINGBOXMENU = 4058  
Global Const PEP\_nROTATIONMENU = 4059  
Global Const PEP\_nCONTOURMENU = 4060

Global Const PEP\_nPERCENTORVALUEMENU = 3925  
Global Const PEP\_nGROUPPERCENTMENU = 3926

Global Const PEP\_nIMAGEADJUSTLEFT = 2982  
Global Const PEP\_nIMAGEADJUSTRIGHT = 2983  
Global Const PEP\_nIMAGEADJUSTTOP = 2984  
Global Const PEP\_nIMAGEADJUSTBOTTOM = 2985

Global Const PEP\_bMODALDIALOGS = 2978  
Global Const PEP\_bMODELESSONTOP = 2979  
Global Const PEP\_bMODELESSAUTOCLOSE = 2980  
Global Const PEP\_nDIALOGRESULT = 2981

Global Const PEP\_fPOINTPADDING = 3427  
Global Const PEP\_fPOINTPADDINGAREA = 3428

Global Const PEP\_fPOINTPADDINGBAR = 3429

Global Const PEP\_bCUSTOMGRIDNUMBERSY = 3160  
Global Const PEP\_bCUSTOMGRIDNUMBERSRY = 3161  
Global Const PEP\_bCUSTOMGRIDNUMBERSX = 3163  
Global Const PEP\_bCUSTOMGRIDNUMBERSTX = 3695  
Global Const PEP\_structCUSTOMGRIDNUMBERS = 3162  
Global Const PEP\_bCUSTOMGRIDNUMBERSZ = 4055

Global Const PEP\_nTICKSTYLE = 3158  
Global Const PEP\_dwTICKCOLOR = 3159

Global Const PEP\_naPOINTTYPES = 3156  
Global Const PEP\_naSUBSETFORPOINTTYPES = 3157

Global Const PEP\_naSUBSETFORPOINTCOLORS = 3155  
Global Const PEP\_nZOOMSTYLE = 3154

Global Const PEP\_nSHOWPIELABELS = 3921  
Global Const PEP\_bSHOWPIELEGENDD = 3922  
Global Const PEP\_nSLICEHATCHING = 3923  
Global Const PEP\_nSLICESTARTLOCATION = 3924  
Global Const PEP\_nMULTIAXISSEPARATORSIZE = 3153  
Global Const PEP\_nCURSORPROMPTLOCATION = 3152  
Global Const PEP\_szMULTIAXISTITLES = 3150  
Global Const PEP\_nLEGENDSTYLE = 2975  
Global Const PEP\_bNOSMARTTABLEPLACEMENT = 2976  
Global Const PEP\_nSMARTLEGENDTHRESHOLD = 3148  
Global Const PEP\_nMULTIAXISSTYLE = 3149  
Global Const PEP\_bFLOATINGBARS = 3151  
Global Const PEP\_bSIMPLELINELEGENDD = 2973  
Global Const PEP\_bSIMPLEPOINTLEGENDD = 2974  
Global Const PEP\_bDISABLESORTPLOTMETHODS = 3147  
Global Const PEP\_nWORKINGTABLE = 2977  
Global Const PEP\_nTAROWS = 2951  
Global Const PEP\_nTACOLUMNS = 2952  
Global Const PEP\_naTATYPE = 2953  
Global Const PEP\_szaTATEXT = 2954  
Global Const PEP\_dwaTACOLOR = 2955  
Global Const PEP\_naTAHOTSPOT = 2956  
Global Const PEP\_nTAHEADERROWS = 2957  
Global Const PEP\_bTAHEADERCOLUMN = 2958  
Global Const PEP\_naTACOLUMNWIDTH = 2959  
Global Const PEP\_nTAHEADERORIENTATION = 2960  
Global Const PEP\_nTALOCATION = 2961  
Global Const PEP\_nTABORDER = 2962  
Global Const PEP\_dwTABACKCOLOR = 2963  
Global Const PEP\_dwTAFORCOLOR = 2964  
Global Const PEP\_nTATEXTSIZE = 2965  
Global Const PEP\_nTAAXISLOCATION = 2966  
Global Const PEP\_bSHOWTABLEANNOTATION = 2968  
Global Const PEP\_naTAJUSTIFICATION = 2969  
Global Const PEP\_szTAFONT = 2970  
Global Const PEP\_szaTAFONTS = 2971

Global Const PEP\_nDELIMITER = 2950

Global Const PEP\_bDISABLESYMBOLFIX = 2972

Global Const PEP\_nAXISSIZEY = 3143

Global Const PEP\_nAXISLOCATIONY = 3144

Global Const PEP\_nAXISSIZERY = 3145

Global Const PEP\_nAXISLOCATIONRY = 3146

Global Const PEP\_fFONTSIZEMSCNTL = 2945

Global Const PEP\_fFONTSIZEMBCNTL = 2946

Global Const PEP\_fFONTSIZEGNCNTL = 2947

Global Const PEP\_fFONTSIZECPCNTL = 2948

Global Const PEP\_fFONTSIZEALCNTL = 2949

Global Const PEP\_bFIXEDLINETHICKNESS = 3140

Global Const PEP\_bFIXEDSPMWIDTH = 3141

Global Const PEP\_fDASHLINETHICKNESS = 3142

Global Const PEP\_naHORZLINEANNOHOTSPOT = 3138

Global Const PEP\_naVERTLINEANNOHOTSPOT = 3139

Global Const PEP\_nYEARMONTHDAYPROMPT = 3133

Global Const PEP\_nTIMELABELTYPE = 3134

Global Const PEP\_nDAYLABELTYPE = 3135

Global Const PEP\_nMONTHLABELTYPE = 3136

Global Const PEP\_nYEARLABELTYPE = 3137

Global Const PEP\_dwaAPPENDPOINTCOLORS = 3132

Global Const PEP\_bDISABLECLIPPING = 2944

Global Const PEP\_faWORKINGAXESPROPORTIONS = 3131

Global Const PEP\_nBORDERTYPES = 2943

Global Const PEP\_bDATETIMESHOWSECONDS = 3129

Global Const PEP\_structSPRINGDAYLIGHT = 3127

Global Const PEP\_structFALLDAYLIGHT = 3128

Global Const PEP\_structEXTRAAXISX = 3693

Global Const PEP\_structEXTRAAXISTX = 3694

Global Const PEP\_bTRIANGLEANNOTATIONADJ = 3126

Global Const PEP\_fGRIDASPECT = 3124

Global Const PEP\_faGRIDHOTSPOTVALUE = 3123

Global Const PEP\_bVGNAXISLABELLOCATION = 3121

Global Const PEP\_bALLOWGRIDNUMBERHOTSPOTSY = 3122

Global Const PEP\_bALLOWGRIDNUMBERHOTSPOTSX = 3692

Global Const PEP\_fLEFTEDGESPACING = 3117

Global Const PEP\_fRIGHTEDGESPACING = 3118

Global Const PEP\_fAXISNUMBERSPACING = 3119

Global Const PEP\_fPOLARTICKTHRESHOLD = 3804

Global Const PEP\_fPOLARLINETHRESHOLD = 3805

Global Const PEP\_fPOLAR30DEGTHRESHOLD = 3806

Global Const PEP\_bOLDSCALINGLOGIC = 2942

Global Const PEP\_bSHADEDPOLYGONBORDERS = 4056

Global Const PEP\_dwBARBORDERCOLOR = 3116

Global Const PEP\_dwHATCHBACKCOLOR = 2941

Global Const PEP\_naSUBSETHATCH = 2940

Global Const PEP\_naPOINTHATCH = 3114

Global Const PEP\_bYAXISVERTGRIDNUMBERS = 3113

Global Const PEP\_bDAYLIGHTSAVINGS = 3112

Global Const PEP\_bFIXEDFONTS = 2938

Global Const PEP\_hSIZENSCURSOR = 2939

Global Const PEP\_bCONTOURSTYLELEGEND = 3690

Global Const PEP\_szaCONTOURLABELS = 3691

Global Const PEP\_naPLOTINGMETHODS = 3103

Global Const PEP\_nSPEEDBOOST = 3104

Global Const PEP\_nSHOWTICKMARKY = 3106

Global Const PEP\_nSHOWTICKMARKRY = 3107

Global Const PEP\_nSHOWTICKMARKX = 3108

Global Const PEP\_nOHLCCMINWIDTH = 3109

Global Const PEP\_nMULTIAXESSIZING = 3111

Global Const PEP\_nSHOWTICKMARKTX = 3689

Global Const PEP\_bFLOATINGSTACKEDBARS = 3424

Global Const PEP\_nSCROLLINGRANGE = 3425

Global Const PEP\_nSCROLLINGFACTOR = 3426

// FUNCTION / PROPERTY HELPER CODES //

Global Const PECONTROL\_GRAPH = 300

Global Const PECONTROL\_PIE = 302

Global Const PECONTROL\_SGRAPH = 304

Global Const PECONTROL\_PGRAPH = 308

Global Const PECONTROL\_3D = 312

Global Const PESTA\_CENTER = 0

Global Const PESTA\_LEFT = 1

Global Const PESTA\_RIGHT = 2

Global Const PEDO\_DRIVERDEFAULT = 0

Global Const PEDO\_LANDSCAPE = 1

Global Const PEDO\_PORTRAIT = 2

Global Const PEVS\_COLOR = 0

Global Const PEVS\_MONO = 1

Global Const PEVS\_MONOWITHSYMBOLS = 2

Global Const PEFS\_LARGE = 0

Global Const PEFS\_MEDIUM = 1

Global Const PEFS\_SMALL = 2

Global Const PEVB\_NONE = 0  
Global Const PEVB\_TOP = 1  
Global Const PEVB\_BOTTOM = 2  
Global Const PEVB\_TOPANDBOTTOM = 3

Global Const PEAC\_AUTO = 0  
Global Const PEAC\_NORMAL = 1  
Global Const PEAC\_LOG = 2

Global Const PEMC\_HIDE = 0  
Global Const PEMC\_SHOW = 1  
Global Const PEMC\_GRAYED = 2

Global Const PECM\_SHOW = 0  
Global Const PECM\_GRAYED = 1  
Global Const PECM\_HIDE = 2

Global Const PECMS\_UNCHECKED = 0  
Global Const PECMS\_CHECKED = 1

Global Const PECML\_TOP = 0  
Global Const PECML\_ABOVE\_SEPARATOR = 1  
Global Const PECML\_BELOW\_SEPARATOR = 2  
Global Const PECML\_BOTTOM = 3

Global Const PEGPM\_LINE = 0  
Global Const PEGPM\_BAR = 1  
Global Const PEGPM\_STICK = 4  
Global Const PEGPM\_POINT = 2  
Global Const PEGPM\_AREA = 3  
Global Const PEGPM\_AREASTACKED = 4  
Global Const PEGPM\_AREASTACKEDPERCENT = 5  
Global Const PEGPM\_BARSTACKED = 6  
Global Const PEGPM\_BARSTACKEDPERCENT = 7  
Global Const PEGPM\_POINTSPLUSBFL = 8  
Global Const PEGPM\_POINTSPLUSBFLGRAPHED = 9  
Global Const PEGPM\_HISTOGRAM = 10  
Global Const PEGPM\_SPECIFICPLOTMODE = 11  
Global Const PEGPM\_BUBBLE = 12  
Global Const PEGPM\_POINTSPLUSBFC = 13  
Global Const PEGPM\_POINTSPLUSBFCGRAPHED = 14  
Global Const PEGPM\_POINTSPLUSLINE = 15  
Global Const PEGPM\_SPLINE = 16  
Global Const PEGPM\_POINTSPLUSLINE = 17  
Global Const PEGPM\_HORIZONTALBAR = 18  
Global Const PEGPM\_HORZBARSTACKED = 19  
Global Const PEGPM\_HORZBARSTACKEDPERCENT = 20  
Global Const PEGPM\_STEP = 21  
Global Const PEGPM\_RIBBON = 22  
Global Const PEGPM\_CONTOURLINES = 23  
Global Const PEGPM\_CONTOURCOLORS = 24  
Global Const PEGPM\_HIGHLOWBAR = 25  
Global Const PEGPM\_HIGHLOWLINE = 26  
Global Const PEGPM\_HIGHLOWCLOSE = 27  
Global Const PEGPM\_OPENHIGHLOWCLOSE = 28

Global Const PEGPM\_BOXPLOT = 29

Global Const PECPS\_NONE = 0

Global Const PECPS\_XVALUE = 1

Global Const PECPS\_YVALUE = 2

Global Const PECPS\_XYVALUES = 3

Global Const PEAUI\_NONE = 0

Global Const PEAUI\_ALL = 1

Global Const PEAUI\_DISABLEKEYBOARD = 2

Global Const PEAUI\_DISABLEMOUSE = 3

Global Const PEGLC\_BOTH = 0

Global Const PEGLC\_YAXIS = 1

Global Const PEGLC\_XAXIS = 2

Global Const PEGLC\_NONE = 3

Global Const PEAS\_SUMPP = 51

Global Const PEAS\_MINAP = 1

Global Const PEAS\_MINPP = 52

Global Const PEAS\_MAXAP = 2

Global Const PEAS\_MAXPP = 53

Global Const PEAS\_AVGAP = 3

Global Const PEAS\_AVGPP = 54

Global Const PEAS\_P1SDAP = 4

Global Const PEAS\_P1SDPP = 55

Global Const PEAS\_P2SDAP = 5

Global Const PEAS\_P2SDPP = 56

Global Const PEAS\_P3SDAP = 6

Global Const PEAS\_P3SDPP = 57

Global Const PEAS\_M1SDAP = 7

Global Const PEAS\_M1SDPP = 58

Global Const PEAS\_M2SDAP = 8

Global Const PEAS\_M2SDPP = 59

Global Const PEAS\_M3SDAP = 9

Global Const PEAS\_M3SDPP = 60

Global Const PEAS\_PARETO\_ASC = 90

Global Const PEAS\_PARETO\_DEC = 91

Global Const PEPTGI\_FIRSTPOINTS = 0

Global Const PEPTGI\_LASTPOINTS = 1

Global Const PEPTGV\_SEQUENTIAL = 0

Global Const PEPTGV\_RANDOM = 1

Global Const PEGPT\_GRAPH = 0

Global Const PEGPT\_TABLE = 1

Global Const PEGPT\_BOTH = 2

Global Const PETW\_GRAPHED = 0

Global Const PETW\_ALLSUBSETS = 1

Global Const PEDLT\_PERCENTAGE = 0

Global Const PEDLT\_VALUE = 1

Global Const PEMSC\_NONE = 0

Global Const PEMSC\_MIN = 1  
Global Const PEMSC\_MAX = 2  
Global Const PEMSC\_MINMAX = 3

Global Const IDEXPORBTUTTON = 1015  
Global Const IDMAXIMIZEBUTTON = 1016  
Global Const IDORIGINALBUTTON = 1109

Global Const PEHS\_NONE = 0  
Global Const PEHS\_SUBSET = 1  
Global Const PEHS\_POINT = 2  
Global Const PEHS\_GRAPH = 3  
Global Const PEHS\_TABLE = 4  
Global Const PEHS\_DATAPOINT = 5  
Global Const PEHS\_ANNOTATION = 6  
Global Const PEHS\_XAXISANNOTATION = 7  
Global Const PEHS\_YAXISANNOTATION = 8  
Global Const PEHS\_HORZLINEANNOTATION = 9  
Global Const PEHS\_VERTLINEANNOTATION = 10  
Global Const PEHS\_MAINTITLE = 11  
Global Const PEHS\_SUBTITLE = 12  
Global Const PEHS\_MULTISUBTITLE = 13  
Global Const PEHS\_MULTIBOTTOMTITLE = 14  
Global Const PEHS\_YAXISLABEL = 15  
Global Const PEHS\_XAXISLABEL = 16  
Global Const PEHS\_YAXIS = 17  
Global Const PEHS\_XAXIS = 18  
Global Const PEHS\_YAXISGRIDNUMBER = 19  
Global Const PEHS\_RYAXISGRIDNUMBER = 20  
Global Const PEHS\_XAXISGRIDNUMBER = 21  
Global Const PEHS\_TXAXISGRIDNUMBER = 22  
Global Const PEHS\_TABLEANNOTATION = 23  
Global Const PEHS\_TABLEANNOTATION19 = 42

Global Const PESPM\_NONE = 0  
Global Const PESPM\_HIGHLOWBAR = 1  
Global Const PESPM\_HIGHLOWLINE = 2  
Global Const PESPM\_HIGHLOWCLOSE = 3  
Global Const PESPM\_OPENHIGHLOWCLOSE = 4  
Global Const PESPM\_BOXPLOT = 5

Global Const PEZIO\_NORMAL = 0  
Global Const PEZIO\_RECT = 1  
Global Const PEZIO\_LINE = 2

Global Const PETS\_GRIDSTYLE = 0  
Global Const PETS\_THICK = 1  
Global Const PETS\_DOT = 2  
Global Const PETS\_DASH = 3  
Global Const PETS\_1UNIT = 4  
Global Const PETS\_THIN = 5

Global Const PEZS\_FRAMED\_RECT = 0  
Global Const PEZS\_RO2\_NOT = 1

Global Const PECPL\_TOP\_LEFT = 0

Global Const PECPL\_TOP\_RIGHT = 1

Global Const PELS\_2\_LINE = 0

Global Const PELS\_1\_LINE = 1

Global Const PELS\_1\_LINE\_INSIDE\_AXIS = 2

Global Const PELS\_1\_LINE\_TOP\_OF\_AXIS = 3

Global Const PEMAS\_GROUP\_ALL\_AXES = 0

Global Const PEMAS\_SEPARATE\_AXES = 1

Global Const PETAHO\_HORZ = 0

Global Const PETAHO\_90 = 1

Global Const PETAHO\_270 = 2

Global Const PETAL\_TOP\_CENTER = 0

Global Const PETAL\_TOP\_LEFT = 1

Global Const PETAL\_LEFT\_CENTER = 2

Global Const PETAL\_BOTTOM\_LEFT = 3

Global Const PETAL\_BOTTOM\_CENTER = 4

Global Const PETAL\_BOTTOM\_RIGHT = 5

Global Const PETAL\_RIGHT\_CENTER = 6

Global Const PETAL\_TOP\_RIGHT = 7

Global Const PETAL\_INSIDE\_TOP\_CENTER = 8

Global Const PETAL\_INSIDE\_TOP\_LEFT = 9

Global Const PETAL\_INSIDE\_LEFT\_CENTER = 10

Global Const PETAL\_INSIDE\_BOTTOM\_LEFT = 11

Global Const PETAL\_INSIDE\_BOTTOM\_CENTER = 12

Global Const PETAL\_INSIDE\_BOTTOM\_RIGHT = 13

Global Const PETAL\_INSIDE\_RIGHT\_CENTER = 14

Global Const PETAL\_INSIDE\_TOP\_RIGHT = 15

Global Const PETAL\_INSIDE\_AXIS = 100

Global Const PETAL\_INSIDE\_AXIS\_0 = 100

Global Const PETAL\_INSIDE\_AXIS\_1 = 101

Global Const PETAL\_INSIDE\_AXIS\_2 = 102

Global Const PETAL\_INSIDE\_AXIS\_3 = 103

Global Const PETAL\_INSIDE\_AXIS\_4 = 104

Global Const PETAL\_INSIDE\_AXIS\_5 = 105

Global Const PETAL\_OUTSIDE\_AXIS = 200

Global Const PETAL\_OUTSIDE\_AXIS\_0 = 200

Global Const PETAL\_OUTSIDE\_AXIS\_1 = 201

Global Const PETAL\_OUTSIDE\_AXIS\_2 = 202

Global Const PETAL\_OUTSIDE\_AXIS\_3 = 203

Global Const PETAL\_OUTSIDE\_AXIS\_4 = 204

Global Const PETAL\_OUTSIDE\_AXIS\_5 = 205

Global Const PETAL\_INSIDE\_TABLE = 300

Global Const PETAB\_DROP\_SHADOW = 0

Global Const PETAB\_SINGLE\_LINE = 1

Global Const PETAB\_NO\_BORDER = 2

Global Const PETAB\_INSET = 3

Global Const PETAAL\_TOP\_FULL\_WIDTH = 0

Global Const PETAAL\_TOP\_LEFT = 1

Global Const PETAAL\_TOP\_CENTER = 2

Global Const PETAAL\_TOP\_RIGHT = 3

Global Const PETAAL\_BOTTOM\_FULL\_WIDTH = 4

Global Const PETAAL\_BOTTOM\_LEFT = 5  
Global Const PETAAL\_BOTTOM\_CENTER = 6  
Global Const PETAAL\_BOTTOM\_RIGHT = 7  
Global Const PETAAL\_TOP\_TABLE\_SPACED = 8  
Global Const PETAAL\_BOTTOM\_TABLE\_SPACED = 9  
Global Const PETAAL\_NEW\_ROW = 100

Global Const PETAJ\_LEFT = 0  
Global Const PETAJ\_CENTER = 1  
Global Const PETAJ\_RIGHT = 2

Global Const PESTM\_TICKS\_INSIDE = 0  
Global Const PESTM\_TICKS\_OUTSIDE = 1  
Global Const PESTM\_TICKS\_HIDE = 2

Global Const PESPL\_PERCENTPLUSLABEL = 0  
Global Const PESPL\_PERCENT = 1  
Global Const PESPL\_LABEL = 2

Global Const PESH\_MONOCHROME = 0  
Global Const PESH\_BOTH = 1

// HORIZONTAL LINE ANNOTATIONS CAN BE WITH RESPECT TO RIGHT Y AXIS COORDINATES  
// BY ADDING 1000 TO THE FOLLOWING CONSTANTS

Global Const PELT\_THINSOLID = 0  
Global Const PELT\_DASH = 1  
Global Const PELT\_DOT = 2  
Global Const PELT\_DASHDOT = 3  
Global Const PELT\_DASHDOTDOT = 4  
Global Const PELT\_MEDIUMSOLID = 5  
Global Const PELT\_THICKSOLID = 6  
Global Const PELAT\_GRIDTICK = 7  
Global Const PELAT\_GRIDLINE = 8  
Global Const PELT\_MEDIUMTHINSOLID = 9  
Global Const PELT\_MEDIUMTHICKSOLID = 10  
Global Const PELT\_EXTRATHICKSOLID = 11  
Global Const PELT\_EXTRATHINSOLID = 12  
Global Const PELT\_EXTRAEXTRATHINSOLID = 13

Global Const PEPS\_SMALL = 0  
Global Const PEPS\_MEDIUM = 1  
Global Const PEPS\_LARGE = 2  
Global Const PEPS\_MICRO = 3

Global Const PEPT\_PLUS = 0  
Global Const PEPT\_CROSS = 1  
Global Const PEPT\_DOT = 2  
Global Const PEPT\_DOTSOLID = 3  
Global Const PEPT\_SQUARE = 4  
Global Const PEPT\_SQUARESOLID = 5  
Global Const PEPT\_DIAMOND = 6  
Global Const PEPT\_DIAMONDSOLID = 7  
Global Const PEPT\_UPTRIANGLE = 8  
Global Const PEPT\_UPTRIANGLESOLID = 9  
Global Const PEPT\_DOWNTRIANGLE = 10

Global Const PEPT\_DOWNTRIANGLESOLID = 11  
Global Const PEPT\_DASH = 72  
Global Const PEPT\_PIXEL = 73  
Global Const PEPT\_ARROW\_N = 92  
Global Const PEPT\_ARROW\_NE = 93  
Global Const PEPT\_ARROW\_E = 94  
Global Const PEPT\_ARROW\_SE = 95  
Global Const PEPT\_ARROW\_S = 96  
Global Const PEPT\_ARROW\_SW = 97  
Global Const PEPT\_ARROW\_W = 98  
Global Const PEPT\_ARROW\_NW = 99

Global Const PEADL\_NONE = 0  
Global Const PEADL\_DATAVALUES = 1  
Global Const PEADL\_POINTLABELS = 2  
Global Const PEADL\_DATAPOINTLABELS = 3

Global Const PEAZ\_NONE = 0  
Global Const PEAZ\_HORIZONTAL = 1  
Global Const PEAZ\_VERTICAL = 2  
Global Const PEAZ\_HORZANDVERT = 3

Global Const PEBFD\_2ND = 0  
Global Const PEBFD\_3RD = 1  
Global Const PEBFD\_4TH = 2

Global Const PEBS\_SMALL = 0  
Global Const PEBS\_MEDIUM = 1  
Global Const PEBS\_LARGE = 2

Global Const PECG\_COARSE = 0  
Global Const PECG\_MEDIUM = 1  
Global Const PECG\_FINE = 2

Global Const PEAЕ\_NONE = 0  
Global Const PEAЕ\_ALLSUBSETS = 1  
Global Const PEAЕ\_INDSUBSETS = 2

Global Const PECM\_NOCURSOR = 0  
Global Const PECM\_POINT = 1  
Global Const PECM\_DATACROSS = 2  
Global Const PECM\_DATASQUARE = 3  
Global Const PECM\_FLOATINGY = 4  
Global Const PECM\_FLOATINGXY = 5  
Global Const PECM\_FLOATINGXONLY = 6  
Global Const PECM\_FLOATINGYONLY = 7

// GRAPH ANNOTATIONS CAN BE WITH RESPECT TO RIGHT Y AXIS COORDINATES  
// BY ADDING 1000 TO THE FOLLOWING CONSTANTS

Global Const PEGAT\_NOSYMBOL = 0  
Global Const PEGAT\_PLUS = 1  
Global Const PEGAT\_CROSS = 2  
Global Const PEGAT\_DOT = 3  
Global Const PEGAT\_DOTSOLID = 4  
Global Const PEGAT\_SQUARE = 5  
Global Const PEGAT\_SQUARESOLID = 6

Global Const PEGAT\_DIAMOND = 7  
Global Const PEGAT\_DIAMONDSOLID = 8  
Global Const PEGAT\_UPTRIANGLE = 9  
Global Const PEGAT\_UPTRIANGLESOLID = 10  
Global Const PEGAT\_DOWNTRIANGLE = 11  
Global Const PEGAT\_DOWNTRIANGLESOLID = 12  
Global Const PEGAT\_SMALLPLUS = 13  
Global Const PEGAT\_SMALLCROSS = 14  
Global Const PEGAT\_SMALLDOT = 15  
Global Const PEGAT\_SMALLDOTSOLID = 16  
Global Const PEGAT\_SMALLSQUARE = 17  
Global Const PEGAT\_SMALLSQUARESOLID = 18  
Global Const PEGAT\_SMALLDIAMOND = 19  
Global Const PEGAT\_SMALLDIAMONDSOLID = 20  
Global Const PEGAT\_SMALLUPTRIANGLE = 21  
Global Const PEGAT\_SMALLUPTRIANGLESOLID = 22  
Global Const PEGAT\_SMALLDOWNTRIANGLE = 23  
Global Const PEGAT\_SMALLDOWNTRIANGLESOLID = 24  
Global Const PEGAT\_LARGEPLUS = 25  
Global Const PEGAT\_LARGECROSS = 26  
Global Const PEGAT\_LARGEDOT = 27  
Global Const PEGAT\_LARGEDOTSOLID = 28  
Global Const PEGAT\_LARGESQUARE = 29  
Global Const PEGAT\_LARGESQUARESOLID = 30  
Global Const PEGAT\_LARGEDIAMOND = 31  
Global Const PEGAT\_LARGEDIAMONDSOLID = 32  
Global Const PEGAT\_LARGEUPTRIANGLE = 33  
Global Const PEGAT\_LARGEUPTRIANGLESOLID = 34  
Global Const PEGAT\_LARGEDOWNTRIANGLE = 35  
Global Const PEGAT\_LARGEDOWNTRIANGLESOLID = 36

Global Const PEGAT\_POINTER = 37

Global Const PEGAT\_THINSOLIDLINE = 38  
Global Const PEGAT\_DASHLINE = 39  
Global Const PEGAT\_DOTLINE = 40  
Global Const PEGAT\_DASHDOTLINE = 41  
Global Const PEGAT\_DASHDOTDOTLINE = 42  
Global Const PEGAT\_MEDIUMSOLIDLINE = 43  
Global Const PEGAT\_THICKSOLIDLINE = 44  
Global Const PEGAT\_LINECONTINUE = 45

Global Const PEGAT\_TOPLEFT = 46  
Global Const PEGAT\_BOTTOMRIGHT = 47

Global Const PEGAT\_RECT\_THIN = 48  
Global Const PEGAT\_RECT\_DASH = 49  
Global Const PEGAT\_RECT\_DOT = 50  
Global Const PEGAT\_RECT\_DASHDOT = 51  
Global Const PEGAT\_RECT\_DASHDOTDOT = 52  
Global Const PEGAT\_RECT\_MEDIUM = 53  
Global Const PEGAT\_RECT\_THICK = 54  
Global Const PEGAT\_RECT\_FILL = 55

Global Const PEGAT\_ROUNDRECT\_THIN = 56  
Global Const PEGAT\_ROUNDRECT\_DASH = 57

Global Const PEGAT\_ROUNDRECT\_DOT = 58  
Global Const PEGAT\_ROUNDRECT\_DASHDOT = 59  
Global Const PEGAT\_ROUNDRECT\_DASHDOTDOT = 60  
Global Const PEGAT\_ROUNDRECT\_MEDIUM = 61  
Global Const PEGAT\_ROUNDRECT\_THICK = 62  
Global Const PEGAT\_ROUNDRECT\_FILL = 63

Global Const PEGAT\_ELLIPSE\_THIN = 64  
Global Const PEGAT\_ELLIPSE\_DASH = 65  
Global Const PEGAT\_ELLIPSE\_DOT = 66  
Global Const PEGAT\_ELLIPSE\_DASHDOT = 67  
Global Const PEGAT\_ELLIPSE\_DASHDOTDOT = 68  
Global Const PEGAT\_ELLIPSE\_MEDIUM = 69  
Global Const PEGAT\_ELLIPSE\_THICK = 70  
Global Const PEGAT\_ELLIPSE\_FILL = 71

Global Const PEGAT\_DASH = 72  
Global Const PEGAT\_PIXEL = 73

Global Const PEGAT\_STARTPOLY = 74  
Global Const PEGAT\_ADDPOLYPOINT = 75  
Global Const PEGAT\_ENDPOLYGON = 76  
Global Const PEGAT\_ENDPOLYLINE\_THIN = 77  
Global Const PEGAT\_ENDPOLYLINE\_MEDIUM = 78  
Global Const PEGAT\_ENDPOLYLINE\_THICK = 79  
Global Const PEGAT\_ENDPOLYLINE\_DASH = 80  
Global Const PEGAT\_ENDPOLYLINE\_DOT = 81  
Global Const PEGAT\_ENDPOLYLINE\_DASHDOT = 82  
Global Const PEGAT\_ENDPOLYLINE\_DASHDOTDOT = 83

Global Const PEGAT\_STARTTEXT = 84  
Global Const PEGAT\_ADDTEXT = 85  
Global Const PEGAT\_PARAGRAPH = 86

Global Const PEGAT\_MEDIUMTHINSOLID = 87  
Global Const PEGAT\_MEDIUMTHICKSOLID = 88  
Global Const PEGAT\_EXTRATHICKSOLID = 89  
Global Const PEGAT\_EXTRATHINSOLID = 90  
Global Const PEGAT\_EXTRAEXTRATHINSOLID = 91

Global Const PEGAT\_ARROW\_N = 92  
Global Const PEGAT\_ARROW\_NE = 93  
Global Const PEGAT\_ARROW\_E = 94  
Global Const PEGAT\_ARROW\_SE = 95  
Global Const PEGAT\_ARROW\_S = 96  
Global Const PEGAT\_ARROW\_SW = 97  
Global Const PEGAT\_ARROW\_W = 98  
Global Const PEGAT\_ARROW\_NW = 99

Global Const PEDTM\_NONE = 0  
Global Const PEDTM\_VB = 1  
Global Const PEDTM\_DELPHI = 2

Global Const PESC\_POLAR = 0  
Global Const PESC\_SMITH = 1  
Global Const PESC\_ROSE = 2

Global Const PESC\_ADMITTANCE = 3

Global Const PESA\_ALL = 0

Global Const PESA\_AXISLABELS = 1

Global Const PESA\_GRIDNUMBERS = 2

Global Const PESA\_NONE = 3

Global Const PESA\_LABELONLY = 4

Global Const PESA\_EMPTY = 5

Global Const PEMPS\_NONE = 0

Global Const PEMPS\_SMALL = 1

Global Const PEMPS\_MEDIUM = 2

Global Const PEMPS\_LARGE = 3

Global Const PESS\_NONE = 0

Global Const PESS\_FINANCIAL = 1

Global Const PELL\_TOP = 0

Global Const PELL\_BOTTOM = 1

Global Const PELL\_LEFT = 2

Global Const PELL\_RIGHT = 3

Global Const PEHSS\_SMALL = 0

Global Const PEHSS\_MEDIUM = 1

Global Const PEHSS\_LARGE = 2

Global Const PEDS\_NONE = 0

Global Const PEDS\_SHADOWS = 1

Global Const PEDS\_3D = 2

Global Const PEGS\_THIN = 0

Global Const PEGS\_THICK = 1

Global Const PEGS\_DOT = 2

Global Const PEGS\_DASH = 3

Global Const PEGS\_ONEPIXEL = 4

Global Const PEFVP\_AUTO = 0

Global Const PEFVP\_VERT = 1

Global Const PEFVP\_HORZ = 2

Global Const PEFVP\_SLANTED = 3

Global Const PEMAS\_NONE = 0

Global Const PEMAS\_THIN = 1

Global Const PEMAS\_MEDIUM = 2

Global Const PEMAS\_THICK = 3

Global Const PEMAS\_THICKPLUSTICK = 4

Global Const PERI\_INCBY15 = 0

Global Const PERI\_INCBY10 = 1

Global Const PERI\_INCBY5 = 2

Global Const PERI\_INCBY2 = 3

Global Const PERI\_INCBY1 = 4

Global Const PERI\_DECBY1 = 5

Global Const PERI\_DECBY2 = 6

Global Const PERI\_DECBY5 = 7

Global Const PERI\_DECBY10 = 8

Global Const PERI\_DECBY15 = 9

Global Const PERD\_WIREFRAME = 0

Global Const PERD\_PLOTTINGMETHOD = 1

Global Const PERD\_FULLEDETAIL = 2

Global Const PESBB\_WHILEROTATING = 0

Global Const PESBB\_ALWAYS = 1

Global Const PESBB\_NEVER = 2

// PolyModes

Global Const PEPM\_SURFACEPOLYGONS = 1

Global Const PEPM\_3DBAR = 2

Global Const PEPM\_POLYCONDATA = 3

Global Const PEPM\_SCATTER = 4

// Plotting Methods

Global Const PEPLM\_WIREFRAME = 0

Global Const PEPLM\_SURFACE = 1

Global Const PEPLM\_SURFACE\_W\_SHADING = 2

Global Const PEPLM\_SURFACE\_W\_PIXELS = 3

Global Const PEPLM\_SURFACE\_W\_CONTOUR = 4

// Plotting Methods for Scatter Graph

Global Const PEPLM\_POINTS = 0

Global Const PEPLM\_LINES = 1

Global Const PEPLM\_POINTS\_AND\_LINES = 2

Global Const PESC\_NONE = 0

Global Const PESC\_TOPLINES = 1

Global Const PESC\_BOTTOMLINES = 2

Global Const PESC\_TOPCOLORS = 3

Global Const PESC\_BOTTOMCOLORS = 4

Global Const PESS\_WHITESHADING = 0

Global Const PESS\_COLORSHADING = 1

Global Const PEP\_nOBJECTTYPE = 2100

Global Const PEP\_szMAINTITLE = 2105

Global Const PEP\_szSUBTITLE = 2110

Global Const PEP\_nSUBSETS = 2115

Global Const PEP\_nPOINTS = 2120

Global Const PEP\_szaSUBSETLABELS = 2125

Global Const PEP\_szaPOINTLABELS = 2130

Global Const PEP\_faXDATA = 2135

Global Const PEP\_faYDATA = 2140

Global Const PEP\_bMONOWITHSYMBOLS = 2145

Global Const PEP\_nDEFORIENTATION = 2150

Global Const PEP\_bPREPAREIMAGES = 2155

Global Const PEP\_b3DDIALOGS = 2160

Global Const PEP\_bALLOWCUSTOMIZATION = 2165

Global Const PEP\_bALLOWEXPORTING = 2170

Global Const PEP\_bALLOWMAXIMIZATION = 2175  
Global Const PEP\_bALLOWPOPUP = 2180  
Global Const PEP\_nALLOWUSERINTERFACE = 2185  
Global Const PEP\_bALLOWUSERINTERFACE = 2185

Global Const PEP\_dwaSUBSETCOLORS = 2190  
Global Const PEP\_dwaSUBSETSHADES = 2195

Global Const PEP\_nPAGEWIDTH = 2200  
Global Const PEP\_nPAGEHEIGHT = 2205  
Global Const PEP\_rectLOGICALLOC = 2210

Global Const PEP\_bDIRTY = 2215  
Global Const PEP\_bDIALOGSHOWN = 2220

Global Const PEP\_bCUSTOM = 2225

Global Const PEP\_nVIEWINGSTYLE = 2230  
Global Const PEP\_nCVIEWINGSTYLE = 2235

Global Const PEP\_nDATASHADOWS = 2240  
Global Const PEP\_nCDATASHADOWS = 2245  
Global Const PEP\_bDATASHADOWS = 2240  
Global Const PEP\_bCDATASHADOWS = 2245

Global Const PEP\_dwMONODESKCOLOR = 2250  
Global Const PEP\_dwMONOTEXTCOLOR = 2255  
Global Const PEP\_dwMONOSHADOWCOLOR = 2260  
Global Const PEP\_dwMONOGRAPHFORECOLOR = 2265  
Global Const PEP\_dwMONOGRAPHBACKCOLOR = 2270  
Global Const PEP\_dwMONOTABLEFORECOLOR = 2275  
Global Const PEP\_dwMONOTABLEBACKCOLOR = 2280

Global Const PEP\_dwCMONODESKCOLOR = 2285  
Global Const PEP\_dwCMONOTEXTCOLOR = 2290  
Global Const PEP\_dwCMONOSHADOWCOLOR = 2295  
Global Const PEP\_dwCMONOGRAPHFORECOLOR = 2300  
Global Const PEP\_dwCMONOGRAPHBACKCOLOR = 2305  
Global Const PEP\_dwCMONOTABLEFORECOLOR = 2310  
Global Const PEP\_dwCMONOTABLEBACKCOLOR = 2315

Global Const PEP\_dwDESKCOLOR = 2320  
Global Const PEP\_dwTEXTCOLOR = 2325  
Global Const PEP\_dwSHADOWCOLOR = 2330  
Global Const PEP\_dwGRAPHFORECOLOR = 2335  
Global Const PEP\_dwGRAPHBACKCOLOR = 2340  
Global Const PEP\_dwTABLEFORECOLOR = 2345  
Global Const PEP\_dwTABLEBACKCOLOR = 2350

Global Const PEP\_dwCDESKCOLOR = 2355  
Global Const PEP\_dwCTEXTCOLOR = 2360  
Global Const PEP\_dwCSHADOWCOLOR = 2365  
Global Const PEP\_dwCGRAPHFORECOLOR = 2370  
Global Const PEP\_dwCGRAPHBACKCOLOR = 2375  
Global Const PEP\_dwCTABLEFORECOLOR = 2380  
Global Const PEP\_dwCTABLEBACKCOLOR = 2385

Global Const PEP\_dwWDESKCOLOR = 2390  
Global Const PEP\_dwWTEXTCOLOR = 2395  
Global Const PEP\_dwWSHADOWCOLOR = 2400  
Global Const PEP\_dwWGRAPHFORECOLOR = 2405  
Global Const PEP\_dwWGRAPHBACKCOLOR = 2410  
Global Const PEP\_dwWTABLEFORECOLOR = 2415  
Global Const PEP\_dwWTABLEBACKCOLOR = 2420

Global Const PEP\_nDATAPRECISION = 2425  
Global Const PEP\_nCDATAPRECISION = 2430  
Global Const PEP\_nMAXDATAPRECISION = 2431

Global Const PEP\_nFONTSIZE = 2435  
Global Const PEP\_nCFONTSIZE = 2440

Global Const PEP\_szMAINTITLEFONT = 2445  
Global Const PEP\_bMAINTITLEBOLD = 2450  
Global Const PEP\_bMAINTITLEITALIC = 2455  
Global Const PEP\_bMAINTITLEUNDERLINE = 2460  
Global Const PEP\_szCMAINTITLEFONT = 2465  
Global Const PEP\_bCMAINTITLEBOLD = 2470  
Global Const PEP\_bCMAINTITLEITALIC = 2475  
Global Const PEP\_bCMAINTITLEUNDERLINE = 2480

Global Const PEP\_szSUBTITLEFONT = 2485  
Global Const PEP\_bSUBTITLEBOLD = 2490  
Global Const PEP\_bSUBTITLEITALIC = 2495  
Global Const PEP\_bSUBTITLEUNDERLINE = 2500  
Global Const PEP\_szCSUBTITLEFONT = 2505  
Global Const PEP\_bCSUBTITLEBOLD = 2510  
Global Const PEP\_bCSUBTITLEITALIC = 2515  
Global Const PEP\_bCSUBTITLEUNDERLINE = 2520

Global Const PEP\_szLABELFONT = 2525  
Global Const PEP\_bLABELBOLD = 2530  
Global Const PEP\_bLABELITALIC = 2535  
Global Const PEP\_bLABELUNDERLINE = 2540  
Global Const PEP\_szCLABELFONT = 2545  
Global Const PEP\_bCLABELBOLD = 2550  
Global Const PEP\_bCLABELITALIC = 2555  
Global Const PEP\_bCLABELUNDERLINE = 2560

Global Const PEP\_szTABLEFONT = 2565  
Global Const PEP\_szCTABLEFONT = 2570

Global Const PEP\_bCACHEBMP = 2574  
Global Const PEP\_hMEMBITMAP = 2575  
Global Const PEP\_hMEMDC = 2580

Global Const PEP\_bALLOWSUBSETHOTSPOTS = 2600  
Global Const PEP\_bALLOWPOINTHOTSPOTS = 2605  
Global Const PEP\_structHOTSPOTDATA = 2610  
Global Const PEP\_structKEYDOWNDATA = 2612  
Global Const PEP\_bAUTOIMAGERESET = 2615  
Global Const PEP\_bALLOWTITLESIALOG = 2616

Global Const PEP\_nCURSORMODE = 2617  
Global Const PEP\_nCURSORSUBSET = 2618  
Global Const PEP\_nCURSORPOINT = 2619  
Global Const PEP\_nCURSORPROMPTSTYLE = 2620  
Global Const PEP\_bCURSORPROMPTTRACKING = 2621  
Global Const PEP\_bMOUSECURSORCONTROL = 2622  
Global Const PEP\_bALLOWANNOTATIONCONTROL = 2623

Global Const PEP\_naSUBSETSTOLEGEND = 2624  
Global Const PEP\_naLEGENDANNOTATIONTYPE = 2625  
Global Const PEP\_szaLEGENDANNOTATIONTEXT = 2626  
Global Const PEP\_dwaLEGENDANNOTATIONCOLOR = 2627  
Global Const PEP\_nVERTSCROLLPOS = 2628  
Global Const PEP\_bALLOWDEBUGOUTPUT = 2629

Global Const PEP\_szaMULTISUBTITLES = 2630  
Global Const PEP\_szaMULTIBOTTOMTITLES = 2631  
Global Const PEP\_bFOCALRECT = 2632  
Global Const PEP\_fFONTSIZEGLOBALCNTL = 2634  
Global Const PEP\_fFONTSIZETITLECNTL = 2635  
Global Const PEP\_bSUBSETBYPOINT = 2636

Global Const PEP\_ptLASTMOUSEMOVE = 2637  
Global Const PEP\_bALLOWOLEEXPORT = 2638

Global Const PEP\_faZDATA = 2900  
Global Const PEP\_bINVALID = 2905  
Global Const PEP\_bOBJECTINSERVER = 2910  
Global Const PEP\_hwndPARENTALCONTROL = 2915

Global Const PEP\_bPAINTING = 2916

Global Const PEP\_hARROWCURSOR = 2917  
Global Const PEP\_hZOOMCURSOR = 2918  
Global Const PEP\_hHANDCURSOR = 2919  
Global Const PEP\_hNODROPCURSOR = 2920

Global Const PEP\_bNOCUSTOMPARMS = 2921  
Global Const PEP\_bNOHELP = 2922  
Global Const PEP\_szHELPPFILENAME = 2923  
Global Const PEP\_bALLOWTITLEHOTSPOTS = 2924  
Global Const PEP\_bALLOWSUBTITLEHOTSPOTS = 2925  
Global Const PEP\_bALLOWBOTTOMTITLEHOTSPOTS = 2926  
Global Const PEP\_nCHARSET = 2927  
Global Const PEP\_bALLOWJPEGOUTPUT = 2928

Global Const PEP\_bALLOWPAGE1 = 2930  
Global Const PEP\_bALLOWPAGE2 = 2931  
Global Const PEP\_bALLOWSUBSETSPAGE = 2932  
Global Const PEP\_bALLOWPOINTSPAGE = 2933  
Global Const PEP\_bALLOWFONTPAGE = 2934  
Global Const PEP\_bALLOWCOLORPAGE = 2935  
Global Const PEP\_bALLOWSTYLEPAGE = 2936  
Global Const PEP\_bALLOWAXISPAGE = 2937

Global Const PEP\_szXAXISLABEL = 3000  
Global Const PEP\_szYAXISLABEL = 3005  
Global Const PEP\_nVBOUNDARYTYPES = 3010  
Global Const PEP\_fUPPERBOUNDVALUE = 3015  
Global Const PEP\_fLOWERBOUNDVALUE = 3020  
Global Const PEP\_szUPPERBOUNDTEXT = 3025  
Global Const PEP\_szLOWERBOUNDTEXT = 3030

Global Const PEP\_nINITIALSCALEFORRYDATA = 3035  
Global Const PEP\_nSCALEFORRYDATA = 3040  
Global Const PEP\_nYAXISSCALECONTROL = 3045

Global Const PEP\_nMANUALSCALECONTROLRY = 3050  
Global Const PEP\_fMANUALMINRY = 3055  
Global Const PEP\_fMANUALMAXRY = 3060

Global Const PEP\_bNOSCROLLINGSUBSETCONTROL = 3065

Global Const PEP\_nSCROLLINGSUBSETS = 3070  
Global Const PEP\_nCSCROLLINGSUBSETS = 3075

Global Const PEP\_naRANDOMSUBSETSTOGRAPH = 3080  
Global Const PEP\_naCRANDOMSUBSETSTOGRAPH = 3085

Global Const PEP\_nPLOTINGMETHOD = 3090  
Global Const PEP\_nCPLOTINGMETHOD = 3095

Global Const PEP\_nGRIDLINECONTROL = 3100  
Global Const PEP\_nCGRIDLINECONTROL = 3105

Global Const PEP\_bGRIDINFRONT = 3110  
Global Const PEP\_bCGRIDINFRONT = 3115

Global Const PEP\_bTREATCOMPSASNORMAL = 3120  
Global Const PEP\_bCTREATCOMPSASNORMAL = 3125

Global Const PEP\_nCOMPARISONSUBSETS = 3130

Global Const PEP\_bALLOWCOORDPROMPTING = 3200  
Global Const PEP\_bALLOWGRAPHHOTSPOTS = 3205  
Global Const PEP\_bALLOWDATAHOTSPOTS = 3210  
Global Const PEP\_bMARKDATAPOINTS = 3215  
Global Const PEP\_bCMARKDATAPOINTS = 3220

Global Const PEP\_nRYAXISCOMPARISONSUBSETS = 3225  
Global Const PEP\_nRYAXISSCALECONTROL = 3230  
Global Const PEP\_nINITIALSCALEFORRYDATA = 3235  
Global Const PEP\_nMANUALSCALECONTROLRY = 3240  
Global Const PEP\_fMANUALMINRY = 3245  
Global Const PEP\_fMANUALMAXRY = 3250  
Global Const PEP\_szRYAXISLABEL = 3255  
Global Const PEP\_nSCALEFORRYDATA = 3256

Global Const PEP\_bALLOWPLOTCUSTOMIZATION = 3260  
Global Const PEP\_bNEGATIVEFROMXAXIS = 3261  
Global Const PEP\_bMANUALYAXISTICKNLINE = 3262

Global Const PEP\_fMANUALYAXISTICK = 3263  
Global Const PEP\_fMANUALYAXISLINE = 3264  
Global Const PEP\_bMANUALRYAXISTICKNLINE = 3265  
Global Const PEP\_fMANUALRYAXISTICK = 3266  
Global Const PEP\_fMANUALRYAXISLINE = 3267  
Global Const PEP\_fNULLDATAVALUE = 3268  
Global Const PEP\_nPOINTSIZ = 3269  
Global Const PEP\_naSUBSETPOINTTYPES = 3270  
Global Const PEP\_naSUBSETLINETYPES = 3271  
Global Const PEP\_bALLOWBESTFITCURVE = 3272  
Global Const PEP\_nBESTFITDEGREE = 3273  
Global Const PEP\_bALLOWSPLINE = 3274  
Global Const PEP\_nCURVEGRANULARITY = 3275

Global Const PEP\_faAPPENDYDATA = 3276  
Global Const PEP\_szaAPPENDPOINTLABELDATA = 3277

Global Const PEP\_bALLOWLINE = 3279  
Global Const PEP\_bALLOWPOINT = 3280  
Global Const PEP\_bALLOWBESTFITLINE = 3281  
Global Const PEP\_nALLOWZOOMING = 3282  
Global Const PEP\_bZOOMMODE = 3283  
Global Const PEP\_fZOOMMINY = 3284  
Global Const PEP\_fZOOMMAXY = 3285

Global Const PEP\_bFORCERIGHTYAXIS = 3286  
Global Const PEP\_bALLOWPOINTSPLUSLINE = 3287  
Global Const PEP\_bALLOWPOINTSPLUSSPLINE = 3288  
Global Const PEP\_nSYMBOLFREQUENCY = 3289

Global Const PEP\_bSHOWANNOTATIONS = 3290  
Global Const PEP\_bCSHOWANNOTATIONS = 3202  
Global Const PEP\_dwANNOTATIONCOLOR = 3203  
Global Const PEP\_dwCANNOTATIONCOLOR = 3204  
Global Const PEP\_faGRAPHANNOTATIONX = 3291  
Global Const PEP\_faGRAPHANNOTATIONY = 3292  
Global Const PEP\_szaGRAPHANNOTATIONTEXT = 3293  
Global Const PEP\_nMAXAXISANNOTATIONCLUSTER = 3296  
Global Const PEP\_faXAXISANNOTATION = 3297  
Global Const PEP\_szaXAXISANNOTATIONTEXT = 3298  
Global Const PEP\_faYAXISANNOTATION = 3299  
Global Const PEP\_szaYAXISANNOTATIONTEXT = 3201  
Global Const PEP\_bANNOTATIONSINFRONT = 3208  
Global Const PEP\_nCURSORPAGEAMOUNT = 3211  
Global Const PEP\_fLINEGAPTHRESHOLD = 3212

Global Const PEP\_faHORZLINEANNOTATION = 3213  
Global Const PEP\_szaHORZLINEANNOTATIONTEXT = 3214  
Global Const PEP\_naHORZLINEANNOTATIONTYPE = 3216  
Global Const PEP\_dwaHORZLINEANNOTATIONCOLOR = 3217  
Global Const PEP\_faVERTLINEANNOTATION = 3218  
Global Const PEP\_szaVERTLINEANNOTATIONTEXT = 3219  
Global Const PEP\_naVERTLINEANNOTATIONTYPE = 3221  
Global Const PEP\_dwaVERTLINEANNOTATIONCOLOR = 3222  
Global Const PEP\_bSHOWGRAPHANNOTATIONS = 3223  
Global Const PEP\_bSHOWXAXISANNOTATIONS = 3224

Global Const PEP\_bSHOWYAXISANNOTATIONS = 3226  
Global Const PEP\_bSHOWHORZLINEANNOTATIONS = 3227  
Global Const PEP\_bSHOWVERTLINEANNOTATIONS = 3228  
Global Const PEP\_bALLOWGRAPHANNOTHOTSPOTS = 3229  
Global Const PEP\_bALLOWXAXISANNOTHOTSPOTS = 3231  
Global Const PEP\_bALLOWYAXISANNOTHOTSPOTS = 3232  
Global Const PEP\_bALLOWHORZLINEANNOTHOTSPOTS = 3233  
Global Const PEP\_bALLOWVERTLINEANNOTHOTSPOTS = 3234  
Global Const PEP\_dwaGRAPHANNOTATIONCOLOR = 3236  
Global Const PEP\_dwaXAXISANNOTATIONCOLOR = 3237  
Global Const PEP\_dwaYAXISANNOTATIONCOLOR = 3238  
Global Const PEP\_nGRAPHANNOTATIONTEXTSIZE = 3242  
Global Const PEP\_nAXESANNOTATIONTEXTSIZE = 3243  
Global Const PEP\_nLINEANNOTATIONTEXTSIZE = 3244  
Global Const PEP\_naGRAPHANNOTATIONTYPE = 3246  
Global Const PEP\_nZOOMINTERFACEONLY = 3247  
Global Const PEP\_fZOOMMINX = 3248  
Global Const PEP\_fZOOMMAXX = 3249  
Global Const PEP\_nDATAHOTSPOTLIMIT = 3251  
Global Const PEP\_nHOURGLASSTHRESHOLD = 3252  
Global Const PEP\_nHORIZSCROLLPOS = 3253  
Global Const PEP\_bALLOWAREA = 3254  
Global Const PEP\_bVERTORIENT90DEGREES = 3257  
Global Const PEP\_dwaPOINTCOLORS = 3258

Global Const PEP\_naMULTIAXESSUBSETS = 3001  
Global Const PEP\_naGRAPHANNOTATIONAXIS = 3002  
Global Const PEP\_naHORZLINEANNOTATIONAXIS = 3003  
Global Const PEP\_naYAXISANNOTATIONAXIS = 3004  
Global Const PEP\_nWORKINGAXIS = 3006  
Global Const PEP\_faMULTIAXESPROPORTIONS = 3007  
Global Const PEP\_naLEGENDANNOTATIONAXIS = 3008

Global Const PEP\_bLOGSCALEEXPLABELS = 3009  
Global Const PEP\_nPLOTINGMETHODII = 3011  
Global Const PEP\_nCPLOTINGMETHODII = 3012  
Global Const PEP\_faXDATAII = 3013  
Global Const PEP\_faYDATAII = 3014  
Global Const PEP\_bUSINGXDATAII = 3016  
Global Const PEP\_bUSINGYDATAII = 3017  
Global Const PEP\_nDATETIMEMODE = 3018  
Global Const PEP\_fBARWIDTH = 3019

Global Const PEP\_nSPECIFICPLOTMODE = 3021  
Global Const PEP\_bALLOWBAR = 3022  
Global Const PEP\_structGRAPHLOC = 3023  
Global Const PEP\_faAPPENDYDATAII = 3024  
Global Const PEP\_bYAXISONRIGHT = 3026

Global Const PEP\_nSHOWYAXIS = 3027  
Global Const PEP\_nSHOWRYAXIS = 3028  
Global Const PEP\_nSHOWXAXIS = 3029  
Global Const PEP\_nGRIDSTYLE = 3032  
Global Const PEP\_bINVERTEDYAXIS = 3033  
Global Const PEP\_bINVERTEDRYAXIS = 3034  
Global Const PEP\_dwYAXISCOLOR = 3036

Global Const PEP\_dwRYAXISCOLOR = 3037  
Global Const PEP\_dwXAXISCOLOR = 3038  
Global Const PEP\_fFONTSIZEAXISCNTL = 3041  
Global Const PEP\_fFONTSIZELEGENDCNTL = 3042

Global Const PEP\_bYAXISLONGTICKS = 3043  
Global Const PEP\_bRYAXISLONGTICKS = 3044  
Global Const PEP\_nMULTIAXESSEPARATORS = 3046  
Global Const PEP\_nZOOMMINAXIS = 3047  
Global Const PEP\_nZOOMMAXAXIS = 3048

Global Const PEP\_rectGRAPH = 3049  
Global Const PEP\_rectAXIS = 3051  
Global Const PEP\_szLEFTMARGIN = 3052  
Global Const PEP\_szTOPMARGIN = 3053  
Global Const PEP\_szRIGHTMARGIN = 3054  
Global Const PEP\_szBOTTOMMARGIN = 3056

Global Const PEP\_bAUTOSCALEDATA = 3057  
Global Const PEP\_faBESTFITCOEFFS = 3058  
Global Const PEP\_naOVERLAPMULTIAXES = 3059  
Global Const PEP\_bNOHIDDENLINESINAREA = 3061  
Global Const PEP\_bSPECIFICPLOTMODECOLOR = 3062  
Global Const PEP\_nAUTOMINMAXPADDING = 3063

Global Const PEP\_nLOGICALLIMIT = 3064  
Global Const PEP\_bNULLDATAGAPS = 3066  
Global Const PEP\_bALLOWSTEP = 3067  
Global Const PEP\_naSUBSETDEGREE = 3068  
Global Const PEP\_bSCROLLINGVERTZOOM = 3069

Global Const PEP\_szAXISFORMATY = 3071  
Global Const PEP\_szAXISFORMATRY = 3072

Global Const PEP\_fZOOMMINRY = 3073  
Global Const PEP\_fZOOMMAXRY = 3074

Global Const PEP\_n3DTHRESHOLD = 3076

Global Const PEP\_bXAXISLONGTICKS = 3078  
Global Const PEP\_bTXAXISLONGTICKS = 3079

// LATEST  
Global Const PEP\_nHOTSPOTSIZE = 3081  
Global Const PEP\_nLEGENDLOCATION = 3082  
Global Const PEP\_bALLOWAXISLABELHOTSPOTS = 3083  
Global Const PEP\_bALLOWAXISHOTSPOTS = 3084  
Global Const PEP\_bAPPENDWITHNOUPDATE = 3086  
Global Const PEP\_bBESTFITFIX = 3087  
Global Const PEP\_dwBOXPLOTCOLOR = 3088  
Global Const PEP\_naGRAPHANNOTATIONHOTSPOT = 3089  
Global Const PEP\_bALLOWRIBBON = 3091  
Global Const PEP\_bNOGRIDLINEMULTIPLES = 3092  
Global Const PEP\_nSPECIALSCALINGY = 3093  
Global Const PEP\_nSPECIALSCALINGRY = 3094  
Global Const PEP\_nDELTA = 3096

Global Const PEP\_nDELTASPERDAY = 3097  
Global Const PEP\_fSTARTTIME = 3098  
Global Const PEP\_fENDTIME = 3099  
Global Const PEP\_nLOGTICKTHRESHOLD = 3101  
Global Const PEP\_nMINIMUMPOINTSIZ = 3102

Global Const PEP\_naAUTOSTATSUBSETS = 3300  
Global Const PEP\_bNOSTACKEDDATA = 3305  
Global Const PEP\_nPOINTSTOGRAPHINIT = 3310

Global Const PEP\_nPOINTSTOGRAPHVERSION = 3315  
Global Const PEP\_nCPOINTSTOGRAPHVERSION = 3320

Global Const PEP\_nPOINTSTOGRAPH = 3325  
Global Const PEP\_nCPOINTSTOGRAPH = 3330

Global Const PEP\_naRANDOMPOINTSTOGRAPH = 3335  
Global Const PEP\_naCRANDOMPOINTSTOGRAPH = 3340

Global Const PEP\_nFORCEVERTICALPOINTS = 3345  
Global Const PEP\_nCFORCEVERTICALPOINTS = 3350

Global Const PEP\_nGRAPHPLUSTABLE = 3355  
Global Const PEP\_nCGRAPHPLUSTABLE = 3360

Global Const PEP\_nTABLEWHAT = 3365  
Global Const PEP\_nCTABLEWHAT = 3370

Global Const PEP\_bALLOWTABLEHOTSPOTS = 3400  
Global Const PEP\_bALLOWHISTOGRAM = 3401  
Global Const PEP\_naALTFREQUENCIES = 3403  
Global Const PEP\_nTARGETPOINTSTOTABLE = 3404  
Global Const PEP\_nALTFREQTHRESHOLD = 3405  
Global Const PEP\_fMANUALSTACKEDMAXY = 3406  
Global Const PEP\_nMAXPOINTSTOGRAPH = 3407  
Global Const PEP\_bNORANDOMPOINTSTOGRAPH = 3408  
Global Const PEP\_szMANUALMAXPOINTLABEL = 3409  
Global Const PEP\_szMANUALMAXDATASTRING = 3410  
Global Const PEP\_bALLOWBESTFITLINEII = 3413  
Global Const PEP\_bALLOWBESTFITCURVEII = 3414  
Global Const PEP\_bAPPENDTOEND = 3415  
Global Const PEP\_bALLOWHORIZONTALBAR = 3416

Global Const PEP\_nFIRSTPTLABELOFFSET = 3417  
Global Const PEP\_fMANUALSTACKEDMINY = 3418  
Global Const PEP\_bALLOWHORZBARSTACKED = 3419

Global Const PEP\_bTABLECOMPARISONSUBSETS = 3420

// LATEST  
Global Const PEP\_bFORMATTABLE = 3421  
Global Const PEP\_bALLOWTABLE = 3422  
Global Const PEP\_nAUTOXDATA = 3423

Global Const PEP\_nINITIALSCALEFORXDATA = 3600

Global Const PEP\_nSCALEFORXDATA = 3605  
Global Const PEP\_nXAXISSCALECONTROL = 3610

Global Const PEP\_nMANUALSCALECONTROLX = 3615  
Global Const PEP\_fMANUALMINX = 3620  
Global Const PEP\_fMANUALMAXX = 3625

Global Const PEP\_bGRAPHDATALABELS = 3630  
Global Const PEP\_bCGRAPHDATALABELS = 3635

Global Const PEP\_bALLOWBUBBLE = 3640  
Global Const PEP\_nBUBBLESIZE = 3641  
Global Const PEP\_nALLOWDATALABELS = 3642  
Global Const PEP\_szaDATAPOINTLABELS = 3643  
Global Const PEP\_bMANUALXAXISTICKNLINE = 3644  
Global Const PEP\_fMANUALXAXISTICK = 3645  
Global Const PEP\_fMANUALXAXISLINE = 3646  
Global Const PEP\_bALLOWSTICK = 3648  
Global Const PEP\_bSCROLLINGHORZZOOM = 3652  
Global Const PEP\_bNORANDOMPOINTSTOEXPORT = 3653  
Global Const PEP\_bXAXISVERTNUMBERING = 3654  
Global Const PEP\_bENGSTATIONFORMAT = 3655  
Global Const PEP\_fNULLDATAVALUEX = 3656  
Global Const PEP\_bASSUMESEQDATA = 3657  
Global Const PEP\_faAPPENDXDATA = 3658  
Global Const PEP\_faAPPENDXDATAII = 3659

Global Const PEP\_nTXAXISCOMPARISONSUBSETS = 3661  
Global Const PEP\_nTXAXISSCALECONTROL = 3662  
Global Const PEP\_nINITIALSCALEFORTXDATA = 3663  
Global Const PEP\_nMANUALSCALECONTROLTX = 3664  
Global Const PEP\_fMANUALMINTX = 3665  
Global Const PEP\_fMANUALMAXTX = 3666  
Global Const PEP\_szTXAXISLABEL = 3667  
Global Const PEP\_nSCALEFORTXDATA = 3668  
Global Const PEP\_bMANUALTXAXISTICKNLINE = 3669  
Global Const PEP\_fMANUALTXAXISTICK = 3670  
Global Const PEP\_fMANUALTXAXISLINE = 3671  
Global Const PEP\_bFORCETOPXAXIS = 3672  
Global Const PEP\_bXAXISONTOP = 3673  
Global Const PEP\_bINVERTEDXAXIS = 3674  
Global Const PEP\_bINVERTEDTXAXIS = 3675  
Global Const PEP\_nSHOWTXAXIS = 3676  
Global Const PEP\_dwTXAXISCOLOR = 3677

Global Const PEP\_szAXISFORMATX = 3678  
Global Const PEP\_szAXISFORMATTX = 3679

// LATEST

Global Const PEP\_bALLOWCONTOURLINES = 3680  
Global Const PEP\_bALLOWCONTOURCOLORS = 3681  
Global Const PEP\_fMANUALMINZ = 3682  
Global Const PEP\_fMANUALMAXZ = 3683  
Global Const PEP\_nMANUALSCALECONTROLZ = 3684  
Global Const PEP\_fMANUALZAXISLINE = 3685  
Global Const PEP\_nCONTOURLINELABELDENSITY = 3686

Global Const PEP\_bSPECIALDATETIMEMODE = 3687

Global Const PEP\_bSMITHCHART = 3800

Global Const PEP\_nSMITHCHART = 3800

Global Const PEP\_bSMARTGRID = 3801

Global Const PEP\_nSHOWPOLARGRID = 3802

Global Const PEP\_nZERODEGREEOFFSET = 3803

Global Const PEP\_nGROUPINGPERCENT = 3900

Global Const PEP\_nCGROUPINGPERCENT = 3905

Global Const PEP\_nDATA LABELTYPE = 3910

Global Const PEP\_nCDATA LABELTYPE = 3915

Global Const PEP\_nAUTOEXPLODE = 3920

Global Const PEP\_szZAXIS LABEL = 4000

Global Const PEP\_nDEGREE OF ROTATION = 4001

Global Const PEP\_bALLOW ROTATION = 4002

Global Const PEP\_bAUTOROTATION = 4003

Global Const PEP\_nROTATION INCREMENT = 4004

Global Const PEP\_nROTATION DETAIL = 4005

Global Const PEP\_bALLOW HORZ SCROLLBAR = 4006

Global Const PEP\_bALLOW VERT SCROLLBAR = 4007

Global Const PEP\_nVIEWING HEIGHT = 4008

Global Const PEP\_bSURFACE POLYGON BORDERS = 4009

Global Const PEP\_bNOSURFACE POLYGON BORDERS = 4009

Global Const PEP\_nSHOW BOUNDING BOX = 4010

Global Const PEP\_nROTATION SPEED = 4011

Global Const PEP\_bADD SKIRTS = 4012

Global Const PEP\_nPOLYMODE = 4013

Global Const PEP\_struct POLYDATA = 4014

Global Const PEP\_dwXZBACKCOLOR = 4015

Global Const PEP\_dwYBACKCOLOR = 4016

Global Const PEP\_dwZAXIS COLOR = 4017

Global Const PEP\_nSHOW ZAXIS = 4018

Global Const PEP\_bMANUAL ZAXIS TICK NLINE = 4019

Global Const PEP\_fMANUAL ZAXIS TICK = 4020

Global Const PEP\_bZAXIS LONG TICKS = 4021

Global Const PEP\_fZDISTANCE = 4022

Global Const PEP\_bINVERTED ZAXIS = 4023

Global Const PEP\_nSHOW CONTOUR = 4024

Global Const PEP\_bALLOW CONTOUR CONTROL = 4025

Global Const PEP\_bSHOW CONTOUR LEGENDS = 4026

Global Const PEP\_fMANUAL CONTOUR LINE = 4027

Global Const PEP\_fMANUAL CONTOUR MIN = 4028

Global Const PEP\_fMANUAL CONTOUR MAX = 4029

Global Const PEP\_nMANUAL CONTOUR SCALE CONTROL = 4030

Global Const PEP\_nSHADING STYLE = 4031

Global Const PEP\_bRESET GDICACHE = 4032

Global Const PEP\_bSHOW ZAXIS LINE ANNOTATIONS = 4035

Global Const PEP\_faZAXIS LINE ANNOTATION = 4036

Global Const PEP\_szaZAXISLINEANNOTATIONTEXT = 4037  
Global Const PEP\_naZAXISLINEANNOTATIONTYPE = 4038  
Global Const PEP\_dwaZAXISLINEANNOTATIONCOLOR = 4039  
Global Const PEP\_faGRAPHANNOTATIONZ = 4040

Global Const PEP\_bANNOTATIONSONSURFACES = 4041

Global Const PEP\_bALLOWWIREFRAME = 4042  
Global Const PEP\_bALLOWSURFACE = 4043  
Global Const PEP\_bALLOWSURFACESHADING = 4044  
Global Const PEP\_bALLOWSURFACECONTOUR = 4045  
Global Const PEP\_bALLOWSURFACEPIXEL = 4046

Global Const PEP\_bUSINGZDATAII = 4047  
Global Const PEP\_faZDATAII = 4048  
Global Const PEP\_faAPPENDZDATA = 4049  
Global Const PEP\_fNULLDATAVALUEZ = 4050

Global Const PEP\_nINITIALSCALEFORZDATA = 4051  
Global Const PEP\_nSCALEFORZDATA = 4052  
Global Const PEP\_faAPPENDZDATAII = 4053  
Global Const PEP\_bDEGREEPROMPTING = 4054

Type POINT3D  
  x As Single  
  y As Single  
  Z As Single  
End Type

Type POLYGONDATA  
  Vertices(0 To 3) As POINT3D  
  NumberOfVertices As Long  
  PolyColor As Long  
  dwReserved1 As Long  
  dwReserved2 As Long  
End Type

Type Rect  
  left As Long  
  top As Long  
  right As Long  
  bottom As Long  
End Type

Type GLOBALPROPERTIES  
  nObjectType As Long  
  szMainTitle As String \* 48  
  szSubTitle As String \* 48  
  nSubsets As Long  
  nPoints As Long  
  bMonoWithSymbols As Long  
  nDefOrientation As Long  
  nPrepareImages As Long  
  b3dDialogs As Long  
  bDataShadows As Long  
  bAllowCustomization As Long

bAllowExporting As Long  
bAllowMaximization As Long  
bAllowPopup As Long  
nPageWidth As Long  
nPageHeight As Long  
rectLogicalLoc As Rect  
bCustom As Long  
nViewingStyle As Long  
nCViewingStyle As Long  
dwMonoDeskColor As Long  
dwMonoTextColor As Long  
dwMonoShadowColor As Long  
dwMonoGraphForeColor As Long  
dwMonoGraphBackColor As Long  
dwMonoTableForeColor As Long  
dwMonoTableBackColor As Long  
dwCMonoDeskColor As Long  
dwCMonoTextColor As Long  
dwCMonoShadowColor As Long  
dwCMonoGraphForeColor As Long  
dwCMonoGraphBackColor As Long  
dwCMonoTableForeColor As Long  
dwCMonoTableBackColor As Long  
dwDeskColor As Long  
dwTextColor As Long  
dwShadowColor As Long  
dwGraphForeColor As Long  
dwGraphBackColor As Long  
dwTableForeColor As Long  
dwTableBackColor As Long  
dwCDeskColor As Long  
dwCTextColor As Long  
dwCShadowColor As Long  
dwCGraphForeColor As Long  
dwCGraphBackColor As Long  
dwCTableForeColor As Long  
dwCTableBackColor As Long  
nDataPrecision As Long  
nCDataPrecision As Long  
nFontSize As Long  
nCFontSize As Long  
szMainTitleFont As String \* 48  
bMainTitleBold As Long  
bMainTitleItalic As Long  
bMainTitleUnderline As Long  
szCMainTitleFont As String \* 48  
bCMainTitleBold As Long  
bCMainTitleItalic As Long  
bCMainTitleUnderline As Long  
szSubTitleFont As String \* 48  
bSubTitleBold As Long  
bSubTitleItalic As Long  
bSubTitleUnderline As Long  
szCSubTitleFont As String \* 48  
bCSubTitleBold As Long  
bCSubTitleItalic As Long

```
bCSubTitleUnderline As Long
szLabelFont As String * 48
bLabelBold As Long
bLabelItalic As Long
bLabelUnderline As Long
szCLabelFont As String * 48
bCLabelBold As Long
bCLabelItalic As Long
bCLabelUnderline As Long
szTableFont As String * 48
szCTableFont As String * 48
bAllowSubsetHotSpots As Long
bAllowPointHotSpots As Long
End Type
```

```
Type POINTSTRUCT
    x As Long
    y As Long
End Type
```

```
Type PEFILEHDR
    nMajVersion As Long '// ProEssentials version number
    nMinVersion As Long
    nObjectType As Long
    dwSize As Long
    extra(0 To 7) As Long
End Type
```

```
Type SCROLLPARMS
    nVmin As Long '// vertical scrollbar minimum
    nVmax As Long '// vertical scrollbar maximum
    nVpos As Long '// vertical scrollbar position
    nHmin As Long '// horizontal scrollbar minimum
    nHmax As Long '// horizontal scrollbar maximum
    nHpos As Long '// horizontal scrollbar position
End Type
```

```
Type HOTSPOTDATA
    HotSpotL As Long
    HotSpotT As Long
    HotSpotR As Long
    HotSpotB As Long
    nHotSpotType As Long
    n1 As Long
    n2 As Long
End Type
```

```
Type KEYDOWNDATA
    nChar As Integer
    nRepCnt As Integer
    nFlags As Integer
End Type
```

```
Type GRAPHLOC
    nAxis As Long
    fXval As Double
```

```
fYval As Double
End Type
```

```
Type TM
  nMonth As Long
  nDay As Long
  nYear As Long
  nHour As Long
  nMinute As Long
  nSecond As Long
  nWeekDay As Long
  nYearDay As Long
End Type
```

```
Type EXTRAAXIS
  nSize As Long
  fMin As Double
  fMax As Double
  szLabel As String * 64
  fManualLine As Double
  fManualTick As Double
  szFormat As String * 16
  nShowAxis As Long
  nShowTickMark As Long
  bInvertedAxis As Integer
  bLogScale As Integer
  dwColor As Long
End Type
```

```
Type CUSTOMGRIDNUMBERS
  nAxisType As Long ' // 0=Y, 1=RIGHT Y, 2=X, 3=TOP X
  nAxisIndex As Long ' // only used for y and ry axes, index number relates to PEP_nWORKINGAXIS
  dNumber As Double ' // number to format
  szData As String * 48 ' // With PEvget, default format string ... With PEvset, completed formatted string
End Type
```

```
//////// API FUNCTIONS //////////
```

```
Declare Function PEsetglobal Lib "PEGRP32C.DLL" (ByVal hObject&, lpdata As GLOBALPROPERTIES)
As Long
Declare Function PEgetglobal Lib "PEGRP32C.DLL" (ByVal hObject&, lpdata As GLOBALPROPERTIES)
As Long
Declare Function PEvset Lib "PEGRP32C.DLL" (ByVal hObject&, ByVal nProperty&, lpvData As Any,
ByVal nItems&) As Long
Declare Function PEvget Lib "PEGRP32C.DLL" (ByVal hObject&, ByVal nProperty&, lpvDest As Any) As
Long
Declare Function PEvsetcell Lib "PEGRP32C.DLL" (ByVal hObject&, ByVal nProperty&, ByVal nCell&,
lpvData As Any) As Long
Declare Function PEvgetcell Lib "PEGRP32C.DLL" (ByVal hObject&, ByVal nProperty&, ByVal nCell&,
lpvDest As Any) As Long
Declare Function PESzset Lib "PEGRP32C.DLL" (ByVal hObject&, ByVal nProperty&, ByVal szData$) As
Long
Declare Function PESzget Lib "PEGRP32C.DLL" (ByVal hObject&, ByVal nProperty&, ByVal szData$) As
Long
Declare Function PENset Lib "PEGRP32C.DLL" (ByVal hObject&, ByVal nProperty&, ByVal nData&) As
Long
Declare Function PENget Lib "PEGRP32C.DLL" (ByVal hObject&, ByVal nProperty&) As Long
```

Declare Function PEset Lib "PEGRP32C.DLL" (ByVal hObject&, ByVal nProperty&, ByVal nData&) As Long  
 Declare Function PEget Lib "PEGRP32C.DLL" (ByVal hObject&, ByVal nProperty&) As Long  
 Declare Function PCreate Lib "PEGRP32C.DLL" (ByVal nObjectType&, ByVal dwStyle&, lpRect As Rect, ByVal hObject&, ByVal nId&) As Long  
 Declare Function PEdestroy Lib "PEGRP32C.DLL" (ByVal hObject&) As Long  
 Declare Function PEload Lib "PEGRP32C.DLL" (ByVal hObject&, lphGbl As Any) As Long  
 Declare Function PEstore Lib "PEGRP32C.DLL" (ByVal hObject&, lphGbl As Any, lpdwSize As Any) As Long  
 Declare Function PEloadpartial Lib "PEGRP32C.DLL" (ByVal hObject&, lphGbl As Any) As Long  
 Declare Function PEstorepartial Lib "PEGRP32C.DLL" (ByVal hObject&, lphGbl As Any, lpdwSize As Any) As Long  
 Declare Function PEgetmeta Lib "PEGRP32C.DLL" (ByVal hObject&) As Long  
 Declare Function PEResetimage Lib "PEGRP32C.DLL" (ByVal hObject&, ByVal nLength&, ByVal nHeight&) As Long  
 Declare Function PElaunchcustomize Lib "PEGRP32C.DLL" (ByVal hObject&) As Long  
 Declare Function PElaunchexport Lib "PEGRP32C.DLL" (ByVal hObject&) As Long  
 Declare Function PElaunchmaximize Lib "PEGRP32C.DLL" (ByVal hObject&) As Long  
 Declare Function PElaunchtextexport Lib "PEGRP32C.DLL" (ByVal hObject&, ByVal bToFile&, ByVal lpszFilename\$) As Long  
 Declare Function PElaunchprintdialog Lib "PEGRP32C.DLL" (ByVal hObject&, ByVal bFullPage&, lpPoint As POINTSTRUCT) As Long  
 Declare Function PElaunchcolordialog Lib "PEGRP32C.DLL" (ByVal hObject&) As Long  
 Declare Function PElaunchfontdialog Lib "PEGRP32C.DLL" (ByVal hObject&) As Long  
 Declare Function PElaunchpopupmenu Lib "PEGRP32C.DLL" (ByVal hObject&, lpPoint As POINTSTRUCT) As Long  
 Declare Function PEReinitialize Lib "PEGRP32C.DLL" (ByVal hObject&) As Long  
 Declare Function PEReinitializcustoms Lib "PEGRP32C.DLL" (ByVal hObject&) As Long  
 Declare Function PEgethelpcontext Lib "PEGRP32C.DLL" (ByVal hWnd&) As Long  
 Declare Function PECopymetatoclipboard Lib "PEGRP32C.DLL" (ByVal hObject&, lpPoint As POINTSTRUCT) As Long  
 Declare Function PECopymetatofile Lib "PEGRP32C.DLL" (ByVal hObject&, lpPoint As POINTSTRUCT, ByVal lpszFilename\$) As Long  
 Declare Function PECopybitmaptoclipboard Lib "PEGRP32C.DLL" (ByVal hObject&, lpPoint As POINTSTRUCT) As Long  
 Declare Function PECopybitmaptofile Lib "PEGRP32C.DLL" (ByVal hObject&, lpPoint As POINTSTRUCT, ByVal lpszFilename\$) As Long  
 Declare Function PECopyoletoclipboard Lib "PEGRP32C.DLL" (ByVal hObject&, lpPoint As POINTSTRUCT) As Long  
 Declare Function PEprintgraph Lib "PEGRP32C.DLL" (ByVal hObject&, ByVal hdc&, ByVal nWidth&, ByVal nHeight&, ByVal nOrient&) As Long  
 Declare Function PEconvpixeltograph Lib "PEGRP32C.DLL" (ByVal hObject&, ByRef nAxis&, ByRef nX&, ByRef nY&, ByRef fX#, ByRef fY#, ByVal bRight&, ByVal bTop&, ByVal bVV&) As Long  
 Declare Function PEReset Lib "PEGRP32C.DLL" (ByVal hObject&) As Long  
  
 Declare Function PEgethotspot Lib "PEGRP32C.DLL" (ByVal hObject&, ByVal nX&, ByVal nY&) As Long  
 Declare Function PEvsetEx Lib "PEGRP32C.DLL" (ByVal hObject&, ByVal property&, ByVal nStartingCell&, ByVal nCellCount&, lpdata As Any, lpMemSetValue As Any) As Long  
 Declare Function PEvgetEx Lib "PEGRP32C.DLL" (ByVal hObject&, ByVal property&, ByVal nStartingCell&, ByVal nCellCount&, lpdata As Any) As Long  
 Declare Function PEpartialresetimage Lib "PEGRP32C.DLL" (ByVal hObject&, ByVal nStartPoint&, ByVal nPointsToAdd&) As Long  
 Declare Function PESavetofile Lib "PEGRP32C.DLL" (ByVal hObject&, ByVal lpFileName\$) As Long  
 Declare Function PEloadfromfile Lib "PEGRP32C.DLL" (ByVal hObject&, ByVal lpFileName\$) As Long  
 Declare Function PCreatefromfile Lib "PEGRP32C.DLL" (ByVal lpFileName\$, ByVal hObject&, lpRect As Rect, ByVal nId&) As Long

```

Declare Function PEcopyjpegtoclipboard Lib "PEGRP32C.DLL" (ByVal hObject&, lpPoint As POINTSTRUCT) As Long
Declare Function PEcopyjpegtofile Lib "PEGRP32C.DLL" (ByVal hObject&, lpPoint As POINTSTRUCT, ByVal lpszFilename$) As Long

Declare Function PEResetimageEx Lib "PEGRP32C.DLL" (ByVal hObject&, ByVal nExtX, ByVal nExtY, ByVal nOrgX, ByVal nOrgY) As Long
Declare Function PElaunchcustomizeEx Lib "PEGRP32C.DLL" (ByVal hObject&, ByVal nPageID) As Long
Declare Function PECopypngtoclipboard Lib "PEGRP32C.DLL" (ByVal hObject&, lpPoint As POINTSTRUCT) As Long
Declare Function PECopypngtofile Lib "PEGRP32C.DLL" (ByVal hObject&, lpPoint As POINTSTRUCT, ByVal lpFileName$) As Long
Declare Function PEcreateserialdate Lib "PEGRP32C.DLL" (pfSerial As Double, dt As TM, ByVal nType) As Long
Declare Function PEdecipherserialdate Lib "PEGRP32C.DLL" (pfSerial As Double, dt As TM, ByVal nType) As Long
Declare Function PEserializefile Lib "PEGRP32C.DLL" (ByVal hObject&, ByVal lpFileName$) As Long
Declare Function PEVsetcellEx Lib "PEGRP32C.DLL" (ByVal hObject&, ByVal nProperty&, ByVal nSub&, ByVal nPt&, lpvData As Any) As Long
Declare Function PEVgetcellEx Lib "PEGRP32C.DLL" (ByVal hObject&, ByVal nProperty&, ByVal nSub&, ByVal nPt&, lpvDest As Any) As Long
Declare Function GetClientRect Lib "user32" (ByVal hWnd As Long, lpRect As Rect) As Long
Declare Function GlobalLock Lib "Kernel32" (ByVal HGLOBAL&) As Long
Declare Function GlobalUnlock Lib "Kernel32" (ByVal HGLOBAL&) As Long
Declare Function GlobalAlloc Lib "Kernel32" (ByVal nHowTo&, ByVal dwSize As Long) As Long
Declare Function GlobalFree Lib "Kernel32" (ByVal HGLOBAL&) As Long
Declare Function hwrite Lib "Kernel32" Alias "_hwrite" (ByVal HGLOBAL&, lpdata As Any, ByVal dwSize As Long) As Long
Declare Function hread Lib "Kernel32" Alias "_hread" (ByVal HGLOBAL&, lpdata As Any, ByVal dwSize As Long) As Long
Declare Function OpenFile Lib "Kernel32" (ByVal lpszFilename$, lpOFstruct As Any, ByVal nAccess&) As Long
Declare Function lclose Lib "Kernel32" Alias "_lclose" (ByVal hFile&) As Long
Declare Function SetMapMode Lib "gdi32" (ByVal hdc&, ByVal Mode&) As Long
Declare Function SetViewportExtEx Lib "gdi32" (ByVal hdc&, ByVal x&, ByVal y&, lpPoint As Any) As Long
Declare Function SetViewportOrgEx Lib "gdi32" (ByVal hdc&, ByVal x&, ByVal y&, lpPoint As Any) As Long
Declare Function PlayMetaFile Lib "gdi32" (ByVal hdc As Long, ByVal hMF As Long) As Long

Declare Function UpdateWindow Lib "USER32.DLL" (ByVal hObject&) As Long
Declare Function MoveWindow Lib "USER32.DLL" (ByVal hObject&, ByVal nX&, ByVal nY&, ByVal nWidth&, ByVal nHeight&, ByVal bPaint&) As Long
Declare Function InvalidateRect Lib "USER32.DLL" (ByVal hWnd&, lpRect As Any, ByVal bRepaint&) As Long

Declare Function CreateRectRgn Lib "gdi32" (ByVal X1 As Long, ByVal Y1 As Long, ByVal X2 As Long, ByVal Y2 As Long) As Long
Declare Function SelectClipRgn Lib "gdi32" (ByVal hdc As Long, ByVal hRgn As Long) As Long
Declare Function DeleteObject Lib "gdi32" (ByVal hObject As Long) As Long

```

-----  
' QTMClientVB Module

' Author:  
' Wayne Pearson  
' Software Engineering Group, IOT  
-----

### Option Explicit

```
/* Frame formats */  
#define QTM_FrameTypeError      0x01 /* Error frame,          type 0 */  
#define QTM_FrameTypeCalibration 0x02 /* Calibration frame,      type 1 */  
#define QTM_FrameTypeVerification 0x04 /* Verification frame,    type 2 */  
#define QTM_FrameTypeUnidentified3D 0x08 /* Frame with unidentified 3D markers, type 3 */  
#define QTM_FrameTypeIdentified3D 0x10 /* Frame with identified 3D markers, type 4 */  
#define QTM_FrameType6DOF      0x20 /* Frame with 6DOF data,   type 5 */  
#define QTM_FrameTypeRotationMatrix 0x40 /* Frame with rotation matrix, type 6,*/
```

```
Public Const QTM_FrameTypeError As Byte = &H1  
Public Const QTM_FrameTypeCalibration As Byte = &H2  
Public Const QTM_FrameTypeVerification As Byte = &H4  
Public Const QTM_FrameTypeUnidentified3D As Byte = &H8  
Public Const QTM_FrameTypeIdentified3D As Byte = &H10  
Public Const QTM_FrameType6DOF As Byte = &H20  
Public Const QTM_FrameTypeRotationMatrix As Byte = &H40
```

### Type Conversion

'VB	C++	Size	Offset	Multiple
'Single	float	4	4	
'Long	pointer types, int, DWORD		4	4
'Integer	WORD	2	2	
'Byte	char, unsigned char, bool, BYTE	1	1	1

'Note that Dim frameBody() As QTM\_TFrameCalibrationBody allocates 4 bytes (Long) for a pointer to  
'a SafeArray that in turn points to an array of QTM\_TFrameCalibrationBody structures

'Note in VB6 that passing a parameter ByRef means that a 4 byte pointer (Long)  
'to the parameter, AddressOf(parameter), is passed to the function

'Note in C++ that Byte bArray[8] specifies an 8 element byte array with  
'lowest index value 0 and highest index value 7

'Note in C++ that Byte bArray[3, 4] specifies a 2 dimensional 12 element byte array with  
'lowest index value 0 and highest index value 2 for the first dimension and with  
'lowest index value 0 and highest index value 3 for the second dimension

'Note in VB6 that Dim bArray(8) As Byte specifies a 9 element byte array with  
'lowest index value 0 and highest index value 8

'Note in VB6 that Dim bArray(3, 4) As Byte specifies a 2 dimensional 20 element byte array with  
'lowest index value 0 and highest index value 3 for the first dimension and with  
'lowest index value 0 and highest index value 4 for the second dimension

'Note in C++ that, for default aligned structures, structure members are located at offsets  
 'which are multiples of their respective sizes  
 'The offset of any 4 byte structure member is a multiple of 4  
 'The offset of any 2 byte structure member is a multiple of 2  
 'The offset of any 1 byte structure member is a multiple of 1  
 ',  
 'Padding is used where necessary to maintain this alignment of structure members  
 ',  
 'This behavior can be modified by using the #pragma pack directive  
 ',  
 'In VB6 all members are aligned on boundaries which are multiples of 1  
 'This means that all structure members are jammed together with no padding between them  
 ',  
 'This structure alignment can be specified by using a #pragma pack (1) directive in C++  
 ',  
 'If we wish VB structures to adhere to the same alignment rules as C++ structures,  
 'we must either specify the #pragma pack(1) option in our C++ files or, if this option  
 'is unavailable, we must insert padding in our VB structures to ensure proper alignment  
 'of structure members  
 ',

```

'*****
'* Frame content bodies
'*****/
',
'*****
'* Type.....: QTM_TFrameCalibrationBody
'* Description..: Contains calibration data for one camera
'* Remarks.....: See "Command protocol for Evmtrk - PPU frame type 1"
'*****/
'typedef struct
' {
',
'   char   id[8];
'   WORD   markerCount;
'   float  meanResidual;
',
'} QTM_TFrameCalibrationBody;

```

```

Public Type QTM_TFrameCalibrationBody
    id(7) As Byte      ' 0
    markerCount As Integer ' 8
    padding00(1) As Byte ' 10 (2 bytes of padding required to align meanResidual)
    meanResidual As Single ' 12
End Type

```

```

',
',
',
'*****
'* Type.....: QTM_TFrameVerificationBody
'* Description..: Contains verification data for one camera
'* Remarks.....: See "Command protocol for Evmtrk - PPU frame type 2"

```

```

/*****/
typedef struct
{
' char id[8];
' WORD markerCount;
' WORD markerCountAtCalibration;
' float xDiff, yDiff, xRange, yRange;
'
} QTM_TFrameVerificationBody;

```

```

Public Type QTM_TFrameVerificationBody
id(7) As Byte ' 0
markerCount As Integer ' 8
markerCountAtCalibration As Integer ' 10
xDiff As Single ' 12
yDiff As Single ' 16
xRange As Single ' 20
yRange As Single ' 24
End Type

```

```

'
'
'

```

```

/*****
* Type.....: QTM_TFrameUnidentified3DBody
* Description..: Contains 3D data for one marker.
* Remarks.....: See "Command protocol for Evmtrk - PPU frame type 3"
*****/

```

```

typedef struct
{
' float x, y, z, residual;
'
} QTM_TFrameUnidentified3DBody;

```

```

Public Type QTM_TFrameUnidentified3DBody
x As Single ' 0
y As Single ' 4
z As Single ' 8
residual As Single ' 12
End Type

```

```

'
'
'

```

```

/*****
* Type.....: QTM_TFrameIdentified3DBody
* Description..: Contains body matched 3D data and for one marker.
* Remarks.....: See "Command protocol for Evmtrk - PPU frame type 4"
*****/

```

```

typedef struct

```

```
{
,
' BYTE bodyID;          /* Which body this marker belongs to. 0 if unidentified */
' float x, y, z, residual;
,
'} QTM_TFrameIdentified3DBody;
```

```
Public Type QTM_TFrameIdentified3DBody
    bodyID As Byte      ' 0
    padding00(2) As Byte ' 1 (3 bytes of padding required to align x)
    x As Single         ' 4
    y As Single         ' 8
    Z As Single         ' 12
    residual As Single  ' 16
End Type
```

```
,
,
,
/*****
* Type.....: QTM_TFrame6DOFBody
* Description...: Contains 6DOF data on one body.
* Remarks.....: See "Command protocol for Evmtrk - PPU frame type 5"
*****/
'typedef struct
'{
,
' float x, y, z, roll, pitch, yaw, residual;
' DWORD events;
,
'} QTM_TFrame6DOFBody;
```

```
Public Type QTM_TFrame6DOFBody
    x As Single      ' 0
    y As Single      ' 4
    Z As Single      ' 8
    roll As Single   ' 12
    pitch As Single  ' 16
    yaw As Single    ' 20
    residual As Single ' 24
    events As Long   ' 28
End Type
```

```
,
,
,
/*****
* Type.....: QTM_TFrameRotationBody
* Description...: Contains rotation matrix for one body.
* Remarks.....: See "Addition to command protocol for Evmtrk - PPU frame type 6"
*****/
'typedef struct
```

```
{
,
' float x, y, z, residual;
' float r[3][3];
' DWORD events;
,
'} QTM_TFrameRotationBody;
```

```
Public Type QTM_TFrameRotationBody
    x As Single      ' 0
    y As Single      ' 4
    Z As Single      ' 8
    residual As Single ' 12
    r(2, 2) As Single ' 16
    events As Long   ' 52
End Type
```

```
,
,
,
/******
'* Frame contents formats
*****/
,
/******
'* Type.....: QTM_TFrameError
'* Description..: Contains an error frame, frame type 0
'* Remarks.....: Error codes corresponds to the errors listed in
'*               "Command protocol for Evmtrk", rev. 9, p. 9
*****/
'typedef struct
'{
,
' BYTE  errorCode;
' char  description[80];
,
'} QTM_TFrameError;
```

```
Public Type QTM_TFrameError
    errorCode As Byte   ' 0
    description(79) As Byte ' 1
End Type
```

```
,
,
,
/******
'* Type.....: QTM_TFrameCalibration
'* Description..: Contains a calibration frame, frame type 1
'* Remarks.....: See "Command protocol for Evmtrk - PPU frame type 1
*****/
'typedef struct
```

```

{
,
' /* Header */
' BYTE          numberOfCameras;
' char          dateAndTime[20];
,
' /* Body, contains <numberOfCameras> bodies */
' QTM_TFrameCalibrationBody *frameBody;
,
} QTM_TFrameCalibration;

```

```

Public Type QTM_TFrameCalibration
    numberOfCameras As Byte          ' 0
    dateAndTime(19) As Byte          ' 1
    padding00(2) As Byte              ' 21 (3 bytes of padding required to align frameBody)
    frameBody() As QTM_TFrameCalibrationBody ' 24
End Type

```

```

,
,
,
'/******
* Type.....: QTM_TFrameVerification
* Description..: Contains a verification frame, frame type 2
* Remarks.....: See "Command protocol for Evmtrk - PPU frame type 2
*****'/
'typedef struct
'{
,
' /* Header */
' BYTE          numberOfCameras;
,
' /* Body, contains <numberOfCameras> bodies */
' QTM_TFrameVerificationBody *frameBody;
,
} QTM_TFrameVerification;

```

```

Public Type QTM_TFrameVerification
    numberOfCameras As Byte          ' 0
    padding00(2) As Byte              ' 1 (3 bytes of padding required to align frameBody)
    frameBody() As QTM_TFrameVerificationBody ' 4
End Type

```

```

,
,
,
'/******
* Type.....: QTM_TFrameUnidentified3D
* Description..: Contains a frame with 3D marker data, frame type 3
* Remarks.....: See "Command protocol for Evmtrk - PPU frame type 3"
*****'/
'typedef struct

```

```

{
'
' /* Header */
' DWORD      frameCount; /* The frame number */
' BYTE       numberOfMarkers; /* Number of markers in the frame */
'
'           /* i.e. number of frame bodies that follows */
'
' /* Body, contains <numberOfMarkers> bodies */
' QTM_TFrameUnidentified3DBody *frameBody;
'
} QTM_TFrameUnidentified3D;

Public Type QTM_TFrameUnidentified3D
    frameCount As Long          ' 0
    numberOfMarkers As Byte     ' 4
    padding00(2) As Byte       ' 5 (3 bytes of padding required to align frameBody)
    frameBody() As QTM_TFrameUnidentified3DBody ' 8
End Type

'
'
'
'*****
' Type.....: QTM_TFrameIdentified3D
' Description..: Contains a frame with 3D marker data that is matched to it's
'               corresponding rigid body, frame type 4
' Remarks.....: See "Command protocol for Evmtrk - PPU frame type 4
'*****/
'typedef struct
'{
'
' /* Header */
' DWORD      frameCount; /* The frame number */
' BYTE       numberOfMarkers; /* Number of markers in the frame */
'
'           /* i.e. number of frame bodies that follows */
'
' /* Body, contains <numberOfMarkers> bodies */
' QTM_TFrameIdentified3DBody *frameBody;
'
} QTM_TFrameIdentified3D;

Public Type QTM_TFrameIdentified3D
    frameCount As Long          ' 0
    numberOfMarkers As Byte     ' 4
    padding00(2) As Byte       ' 5 (3 bytes of padding required to align frameBody)
    frameBody() As QTM_TFrameIdentified3DBody ' 8
End Type

'
'
'
'*****

```

```

/* Type.....: QTM_TFrame6DOF
/* Description..: Contains a frame with 6DOF data, frame type 5
/* Remarks.....: See "Command protocol for Evmtrk - PPU frame type 5
*****/
typedef struct
{
    /* Header */
    DWORD    frameCount;    /* The frame number */
    BYTE     numberOfBodies; /* Number of rigid bodies in the frame */
                          /* i.e. number of frame bodies that follows */

    /* Body, contains <numberOfBodies> bodies */
    QTM_TFrame6DOFBody *frameBody;
} QTM_TFrame6DOF;

```

```

Public Type QTM_TFrame6DOF
    frameCount As Long    ' 0
    numberOfBodies As Byte    ' 4
    padding00(2) As Byte    ' 5 (3 bytes of padding required to align frameBody)
    frameBody() As QTM_TFrame6DOFBody ' 8
End Type

```

```

,
,
,
/******
/* Type.....: QTM_TFrameRotationMatrix
/* Description..: Contains a frame with rotation matrix information, frame type 6
/* Remarks.....: -
*****/
typedef struct
{
    /* Header */
    DWORD    frameCount;    /* The frame number */
    BYTE     numberOfBodies; /* Number of rigid bodies in the frame */
                          /* i.e. number of frame bodies that follows */

    /* Body, contains <numberOfBodies> bodies */
    QTM_TFrameRotationBody *frameBody;
} QTM_TFrameRotationMatrix;

```

```

Public Type QTM_TFrameRotationMatrix
    frameCount As Long    ' 0
    numberOfBodies As Byte    ' 4
    padding00(2) As Byte    ' 5 (3 bytes of padding required to align frameBody)
    frameBody() As QTM_TFrameRotationBody ' 8
End Type

```



```

Public Type QTM_TPPUFrame
    soh As Byte
    size As Integer
    frameType As Byte
    frameError As Long      ' Add later
    frameCalibration As Long ' Add later
    frameVerification As Long ' Add later
    frameUnidentified3D As QTM_TFrameUnidentified3D
    frameIdentified3D As QTM_TFrameIdentified3D
    frame6DOF As QTM_TFrame6DOF
    frameRotation As QTM_TFrameRotationMatrix
    checksum As Byte
    terminator As Byte
End Type

```

'Note that QTMclient.h boolean is defined as unsigned char  
'This data type is equivalent to the VB Byte data type

```

/*****
'* Function.....: QTM_OpenTCPIPConnection
'* Description....: Opens a TCP connection to the QTM RT tracker server
'* Arguments.....:
'*   strDestination: The address to the QTM server as a host name or an
'*                   IP-address in dot-format
'*   strServiceName: The port where the QTM server listens. Can be either a
'*                   service name or a port number
'* Returns.....: TRUE if successfull and FALSE if unsuccessful.
'*               Call QTM_GetLastError for details.
*****/
'QTMCLIENT_API boolean QTMCLIENT_CALL QTM_OpenTCPIPConnection(
'   LPCTSTR strDestination,
'   LPCTSTR strServiceName
'   );
Public Declare Function QTM_OpenTCPIPConnection Lib "QTMClient.dll" _
    (ByVal strDestination As String, _
    ByVal strServiceName As String) As Byte

/*****
'* Function...: QTM_SetFrameTypes
'* Description: Set the frame type(s) to be returned by QTM_GetFrame.
'*             Uses the QTM_FrameTypeXXX flags.
'* Arguments...:
'*   nFrameType: Frame type flags
'* Returns.....: TRUE if successfull and FALSE if unsuccessful.
'*             Call QTM_GetLastError for details.
*****/
'QTMCLIENT_API boolean QTMCLIENT_CALL QTM_SetFrameTypes( BYTE nFrameType );
Public Declare Function QTM_SetFrameTypes Lib "QTMClient.dll" (ByVal nFrameType As Byte) As Byte

/*****
'* Function...: QTM_AllocateFrameMemoryUnidentified3D
'* Description: Allocates memory to store an Unidentified3D frame
'* Arguments...:
'*   oPPUFrame: The PPUFrame object to hold the Unidentified3D frame.

```

```

'*      oPPUFrame.frameUnidentified3D must be NULL
'*  numberOfMarkers: The number of markers to allocate memory for
'*      Data returned by QTM_GetFrame will contain this number of markers
'* Returns....: TRUE if successfull and FALSE if unsuccessful.
'*      Call QTM_GetLastError for details.
'* Remark.....: If memory previously was allocated for Unidentified3D data,
'*      call QTM_FreeFrameMemoryUnidentified3D first.
'*****/
'QTMCLIENT_API boolean QTMCLIENT_CALL QTM_AllocateFrameMemoryUnidentified3D(
'      QTM_TPPUFrame *oPPUFrame,
'      BYTE      nNumberOfMarkers
'      );
Public Declare Function QTM_AllocateFrameMemoryUnidentified3D Lib "QTMClient.dll" _
    (ByRef oPPUFrame As QTM_TPPUFrameC, _
    ByVal nNumberOfMarkers As Byte) As Byte

'*****
'* Function...: QTM_AllocateFrameMemoryIdentified3D
'* Description: Allocates memory to store an Identified3D frame
'* Arguments...:
'*  oPPUFrame:   The PPUFrame object to hold the Identified3D frame.
'*      oPPUFrame.frameIdentified3D must be NULL
'*  numberOfMarkers: The number of markers to allocate memory for
'*      Data returned by QTM_GetFrame will contain this number of markers
'* Returns....: TRUE if successfull and FALSE if unsuccessful.
'*      Call QTM_GetLastError for details.
'* Remark.....: If memory previously was allocated for Unidentified3D data,
'*      call QTM_FreeFrameMemoryUnidentified3D first.
'*****/
'QTMCLIENT_API boolean QTMCLIENT_CALL QTM_AllocateFrameMemoryIdentified3D(
'      QTM_TPPUFrame *oPPUFrame,
'      BYTE      nNumberOfMarkers
'      );
Public Declare Function QTM_AllocateFrameMemoryIdentified3D Lib "QTMClient.dll" _
    (ByRef oPPUFrame As QTM_TPPUFrameC, _
    ByVal nNumberOfMarkers As Byte) As Byte

'*****
'* Function...: QTM_AllocateFrameMemory6DOF
'* Description: Allocates memory to store a 6DOF frame
'* Arguments...:
'*  oPPUFrame:   The PPUFrame object to hold the 6DOF frame.
'*      oPPUFrame.frame6DOF must be NULL
'*  numberOfBodies: The number of markers to allocate memory for.
'*      Data returned by QTM_GetFrame will contain this number of bodies
'* Returns....: TRUE if successfull and FALSE if unsuccessful.
'*      Call QTM_GetLastError for details.
'* Remark.....: If memory previously was allocated for 6DOF data,
'*      call QTM_FreeFrameMemory6DOF first.
'*****/
'QTMCLIENT_API boolean QTMCLIENT_CALL QTM_AllocateFrameMemory6DOF(
'      QTM_TPPUFrame *oPPUFrame,
'      BYTE      nNumberOfBodies
'      );
Public Declare Function QTM_AllocateFrameMemory6DOF Lib "QTMClient.dll" _
    (ByRef oPPUFrame As QTM_TPPUFrameC, _

```

ByVal nNumberOfBodies As Byte) As Byte

/\*\*\*\*\*\*

\* Function...: QTM\_AllocateFrameMemoryRotationMatrix  
\* Description: Allocates memory to store a RotationMatrix frame  
\* Arguments...:  
\* oPPUFrame: The PPUFrame object to hold the 6DOF frame.  
\* oPPUFrame.frameRotationMatrix must be NULL  
\* nNumberOfBodies: The number of markers to allocate memory for  
\* Data returned by QTM\_GetFrame will contain this number of bodies  
\* Returns....: TRUE if successfull and FALSE if unsuccessful.  
\* Call QTM\_GetLastError for details.  
\* Remark.....: If memory previously was allocated for RotationMatrix data,  
\* call QTM\_FreeFrameMemoryRotationMatrix first.

\*\*\*\*\*/

```
'QTMCLIENT_API boolean QTMCLIENT_CALL QTM_AllocateFrameMemoryRotationMatrix(  
'      QTM_TPPUFrame *oPPUFrame,  
'      BYTE      nNumberOfBodies  
'      );
```

```
Public Declare Function QTM_AllocateFrameMemoryRotationMatrix Lib "QTMClient.dll" _  
    (ByRef oPPUFrame As QTM_TPPUFrameC, _  
    ByVal nNumberOfBodies As Byte) As Byte
```

/\*\*\*\*\*\*

\* Function...: QTM\_ContinuousTransmission  
\* Description: Set whether the data should be transmitted to the QTMClient as soon  
\* as it's available in QTM, or if QTM should send data only as  
\* response to a QTM\_RetrieveOneFrame command  
\* Arguments...:  
\* bContinuousTransmission: True if QTM should send data as soon as it's available

\*\*\*\*\*/

```
'QTMCLIENT_API boolean QTMCLIENT_CALL QTM_ContinuousTransmission( bool  
bContinuousTransmission = TRUE );
```

```
Public Declare Function QTM_ContinuousTransmission Lib "QTMClient.dll" (ByVal  
bContinuousTransmission As Byte) As Byte
```

/\*\*\*\*\*\*

\* Function...: QTM\_HasNewData  
\* Description: Checks for new data  
\* Returns....: True if new data has been received since the last call to QTM\_GetFrame

\*\*\*\*\*/

```
'QTMCLIENT_API boolean QTMCLIENT_CALL QTM_HasNewData( void );  
Public Declare Function QTM_HasNewData Lib "QTMClient.dll" () As Byte
```

/\*\*\*\*\*\*

\* Function...: QTM\_GetFrame  
\* Description: Get data frame containing the types of frames set by QTM\_SetFrameType  
\* The QTM\_HasNewData-flag will be cleared.  
\* Arguments...:  
\* poPPUFrame: Pointer to an array of QTM\_TPPUFrame structs.  
\* poPPUFrame should point to a memory area that was previously  
\* allocated by QTM\_AllocateFrameMemoryXXX. The number of bodies/markers/cameras  
\* in the returned structure was previously set by QTM\_AllocateFrameMemory.  
\* The QTM\_TFrameXXX fields that's not in use should be set to NULL by caller.  
\* Returns....: TRUE if successfull and FALSE if unsuccessful.  
\* Call QTM\_GetLastError for details.

```

/*****/
'QTMCLIENT_API boolean QTMCLIENT_CALL QTM_GetFrame( QTM_TPPUFrame *poPPUFrame );
Public Declare Function QTM_GetFrame Lib "QTMClient.dll" (ByRef oPPUFrame As QTM_TPPUFrameC)
As Byte

```

```

/*****
'* Function...: QTM_FreeFrameMemoryRotationMatrix
'* Description: Releases any memory previously allocated for RotationMatrix data
'* Arguments...:
'*   oPPUFrame: The PPUFrame object that holds the RotationMatrix data.
'* Returns....: TRUE if successfull and FALSE if unsuccessful.
'*       Call QTM_GetLastError for details.
*****/

```

```

'QTMCLIENT_API boolean QTMCLIENT_CALL QTM_FreeFrameMemoryRotationMatrix(
QTM_TPPUFrame *oPPUFrame);
Public Declare Function QTM_FreeFrameMemoryRotationMatrix Lib "QTMClient.dll" (ByRef oPPUFrame
As QTM_TPPUFrameC) As Byte

```

```

/*****
'* Function...: QTM_FreeFrameMemory6DOF
'* Description: Releases any memory previously allocated for 6DOF data
'* Arguments...:
'*   oPPUFrame: The PPUFrame object that holds the 6DOF data.
'* Returns....: TRUE if successfull and FALSE if unsuccessful.
'*       Call QTM_GetLastError for details.
*****/

```

```

'QTMCLIENT_API boolean QTMCLIENT_CALL QTM_FreeFrameMemory6DOF( QTM_TPPUFrame
*oPPUFrame);
Public Declare Function QTM_FreeFrameMemory6DOF Lib "QTMClient.dll" (ByRef oPPUFrame As
QTM_TPPUFrameC) As Byte

```

```

/*****
'* Function...: QTM_FreeFrameMemoryIdentified3D
'* Description: Releases any memory previously allocated for Identified3D data
'* Arguments...:
'*   oPPUFrame: The PPUFrame object that holds the Identified3D data.
'* Returns....: TRUE if successfull and FALSE if unsuccessful.
'*       Call QTM_GetLastError for details.
*****/

```

```

'QTMCLIENT_API boolean QTMCLIENT_CALL QTM_FreeFrameMemoryIdentified3D( QTM_TPPUFrame
*oPPUFrame);
Public Declare Function QTM_FreeFrameMemoryIdentified3D Lib "QTMClient.dll" (ByRef oPPUFrame As
QTM_TPPUFrameC) As Byte

```

```

/*****
'* Function...: QTM_FreeFrameMemoryUnidentified3D
'* Description: Releases any memory previously allocated for Unidentified3D data
'* Arguments...:
'*   oPPUFrame: The PPUFrame object that holds the Unidentified3D data.
'* Returns....: TRUE if successfull and FALSE if unsuccessful.
'*       Call QTM_GetLastError for details.
*****/

```

```

'QTMCLIENT_API boolean QTMCLIENT_CALL QTM_FreeFrameMemoryUnidentified3D(
QTM_TPPUFrame *oPPUFrame);
Public Declare Function QTM_FreeFrameMemoryUnidentified3D Lib "QTMClient.dll" (ByRef oPPUFrame
As QTM_TPPUFrameC) As Byte

```

```

'/******
'* Function...: QTM_CloseConnection
'* Description: Close any currently open connections
'* Returns....: TRUE if successfull and FALSE if unsuccessfull.
'*           Call QTM_GetLastError for details.
'*****/
'QTMCLIENT_API boolean QTMCLIENT_CALL QTM_CloseConnection( void );
Public Declare Function QTM_CloseConnection Lib "QTMClient.dll" () As Byte

Public Declare Sub Sleep Lib "Kernel32" (ByVal dwMilliseconds As Long)

Public Declare Sub CopyMemory Lib "Kernel32" Alias "RtlMoveMemory" (ByRef Destination As Any, ByRef
Source As Any, ByVal Length As Long)

'Note that ZeroMemory is NOT declared properly in "C:\Program Files\Microsoft Visual
Studio\Common\Tools\Winapi\WIN32API.TXT"
'ZeroMemory is declared as
'Declare Sub ZeroMemory Lib "KERNEL32" Alias "RtlMoveMemory" (dest As Any, ByVal numBytes As
Long)
'Note that the Alias "RtlMoveMemory" SHOULD be Alias "RtlZeroMemory"
Public Declare Sub ZeroMemory Lib "kernel32.dll" Alias "RtlZeroMemory" (ByRef Destination As Any,
ByVal Length As Long)

Public Declare Function VarPtrArray Lib "msvbvm60.DLL" Alias "VarPtr" (ByRef Ptr() As Any) As Long

Public Function QTM_GetFrameVB(ByRef moPPUFrameVB As QTM_TPPUFrame, ByRef moPPUFrame As
QTM_TPPUFrameC) As Byte
    ZeroMemory moPPUFrameVB, LenB(moPPUFrameVB)

    QTM_GetFrameVB = QTM_GetFrame(moPPUFrame)

    If (QTM_GetFrameVB = 1) Then
        moPPUFrameVB.soh = moPPUFrame.soh
        moPPUFrameVB.size = moPPUFrame.size
        moPPUFrameVB.frameType = moPPUFrame.frameType
        moPPUFrameVB.frameError =
        moPPUFrameVB.frameCalibration =
        moPPUFrameVB.frameVerification =
        moPPUFrameVB.frameUnidentified3D = GetQTM_TFrameUnidentified3D(moPPUFrame)
        moPPUFrameVB.frameIdentified3D = GetQTM_TFrameIdentified3D(moPPUFrame)
        moPPUFrameVB.frame6DOF = GetQTM_TFrame6DOF(moPPUFrame)
        moPPUFrameVB.frameRotation = GetQTM_TFrameRotationMatrix(moPPUFrame)
        moPPUFrameVB.checksum = moPPUFrame.checksum
        moPPUFrameVB.terminator = moPPUFrame.terminator
    End If
End Function

Public Function GetQTM_TFrameUnidentified3D(ByRef moPPUFrame As QTM_TPPUFrameC) As
QTM_TFrameUnidentified3D
    '
    'Get the QTM_TFrameUnidentified3D information
    '
    'VB6 does not have a sizeof operator. However it does have a Len function.
    'The Len function operates on variables and NOT data types, therefore we allocate

```

'a variable, frameUnidentified3DBody, of type QTM\_TFrameUnidentified3DBody so that  
'we can use the Len function on it to determine the size of the QTM\_TFrameUnidentified3DBody  
'structure  
Dim frameUnidentified3DBody As QTM\_TFrameUnidentified3DBody

'The dataAddress variable is used to temporarily store the address of the marker data  
Dim dataAddress As Long

'Copy the QTM\_TGetQTM\_TFrameUnidentified3D structure into GetQTM\_TFrameUnidentified3D  
,

'Note that CopyMemory's Destination and Source are ByRef parameters  
'For ByRef parameters the address of the parameter, AddressOf(parameter), is  
'passed to the function or sub. It's ok to pass GetQTM\_TFrameUnidentified3D by reference  
'since we want to copy directly into GetQTM\_TFrameUnidentified3D - basically the  
'Destination Address is AddressOf(GetQTM\_TFrameUnidentified3D). However,  
'moPPUFrame.GetQTM\_TFrameUnidentified3D is the "actual" Source address. We do NOT want to  
'copy the contents of AddressOf(moPPUFrame.GetQTM\_TFrameUnidentified3D) to  
'GetQTM\_TFrameUnidentified3D - we want to copy the contents of  
'moPPUFrame.GetQTM\_TFrameUnidentified3D. Therefore we pass  
'moPPUFrame.GetQTM\_TFrameUnidentified3D by value  
CopyMemory GetQTM\_TFrameUnidentified3D, ByVal moPPUFrame.frameUnidentified3D,  
LenB(GetQTM\_TFrameUnidentified3D)

'Save the address of the marker data  
,

'Since we copied the QTM\_TGetQTM\_TFrameUnidentified3D structure into  
GetQTM\_TFrameUnidentified3D,  
'the address of the marker data is stored at GetQTM\_TFrameUnidentified3D.frameBody()  
'We are copying directly into dataAddress so pass it by reference, AddressOf(dataAddress)  
'The value located at GetQTM\_TFrameUnidentified3D.frameBody() is the address of the \*actual\* data  
'so we pass VarPtrArray(GetQTM\_TFrameUnidentified3D.frameBody()) by value - we do NOT want to  
copy  
'the contents of AddressOf(VarPtrArray(GetQTM\_TFrameUnidentified3D.frameBody())) to dataAddress,  
'we want to copy the contents of VarPtrArray(GetQTM\_TFrameUnidentified3D.frameBody()) to  
dataAddress,  
'so we pass VarPtrArray(GetQTM\_TFrameUnidentified3D.frameBody()) by value  
CopyMemory dataAddress, ByVal VarPtrArray(GetQTM\_TFrameUnidentified3D.frameBody()), 4

'GetQTM\_TFrameUnidentified3D.frameBody() contains the address of the marker data. However,  
'in order for GetQTM\_TFrameUnidentified3D.frameBody() to be a proper VB6 array, it should point  
'to a SafeArray that, in turn, points to the marker data. The next couple of statements adjusts  
'GetQTM\_TFrameUnidentified3D.frameBody() to ensure that it's a proper VB6 array

'Zero GetQTM\_TFrameUnidentified3D.frameBody() prior to the following ReDim operation.  
,

'This will prevent any nasty side effects, normally when you ReDim a variable size array, the  
'original data is destroyed. Since GetQTM\_TFrameUnidentified3D.frameBody() does not "own" any data  
'it should not destroy any data on a ReDim. To ensure this is the case we "zero"  
'GetQTM\_TFrameUnidentified3D.frameBody() prior to the ReDim. Since we wish to zero the  
'contents located at memory address VarPtrArray(GetQTM\_TFrameUnidentified3D.frameBody()), and  
'NOT at AddressOf(VarPtrArray(GetQTM\_TFrameUnidentified3D.frameBody())), we pass  
'VarPtrArray(GetQTM\_TFrameUnidentified3D.frameBody()) by value  
ZeroMemory ByVal VarPtrArray(GetQTM\_TFrameUnidentified3D.frameBody()), 4

If (GetQTM\_TFrameUnidentified3D.numberofMarkers > 0) Then

```

'ReDim GetQTM_TFrameUnidentified3D.frameBody large enough to hold the available data
,
'The ReDim statement allocates a SafeArray structure and enough space to hold the specified
'amount of data. It then sets GetQTM_TFrameUnidentified3D.frameBody() to point to the allocated
'SafeArray structure, thus ensuring that GetQTM_TFrameUnidentified3D.frameBody() is a
'proper VB6 array
ReDim
GetQTM_TFrameUnidentified3D.frameBody(GetQTM_TFrameUnidentified3D.numberofMarkers - 1)

'Copy the marker data into the space just allocated
,
'Note that the address of GetQTM_TFrameUnidentified3D.frameBody(0) is the address of the
'data section of the previously allocated space, therefore we pass the Destination address,
'GetQTM_TFrameUnidentified3D.frameBody(0), by value. The Source address is contained in the
'dataAddress variable. We do NOT want to copy from AddressOf(dataAddress), we want to copy from
'dataAddress, therefore we pass dataAddress by value
CopyMemory GetQTM_TFrameUnidentified3D.frameBody(0), ByVal dataAddress,
(GetQTM_TFrameUnidentified3D.numberofMarkers * LenB(frameUnidentified3DBody))
End If
End Function

Public Function GetQTM_TFrameIdentified3D(ByRef moPPUFrame As QTM_TPPUFrameC) As
QTM_TFrameIdentified3D
,
'Get the QTM_TFrameIdentified3D information
,

'VB6 does not have a sizeof operator. However it does have a Len function.
'The Len function operates on variables and NOT data types, therefore we allocate
'a variable, frameIdentified3DBody, of type QTM_TFrameIdentified3DBody so that
'we can use the Len function on it to determine the size of the QTM_TFrameIdentified3DBody
'structure
Dim frameIdentified3DBody As QTM_TFrameIdentified3DBody

'The dataAddress variable is used to temporarily store the address of the marker data
Dim dataAddress As Long

'Copy the QTM_TGetQTM_TFrameIdentified3D structure into GetQTM_TFrameIdentified3D
,
'Note that CopyMemory's Destination and Source are ByVal parameters
'For ByVal parameters the address of the parameter, AddressOf(parameter), is
'passed to the function or sub. It's ok to pass GetQTM_TFrameIdentified3D by reference
'since we want to copy directly into GetQTM_TFrameIdentified3D - basically the
'Destination Address is AddressOf(GetQTM_TFrameIdentified3D). However,
'moPPUFrame.GetQTM_TFrameIdentified3D is the "actual" Source address. We do NOT want to
'copy the contents of AddressOf(moPPUFrame.GetQTM_TFrameIdentified3D) to
'GetQTM_TFrameIdentified3D - we want to copy the contents of
'moPPUFrame.GetQTM_TFrameIdentified3D. Therefore we pass
'moPPUFrame.GetQTM_TFrameIdentified3D by value
CopyMemory GetQTM_TFrameIdentified3D, ByVal moPPUFrame.frameIdentified3D,
LenB(GetQTM_TFrameIdentified3D)

'Save the address of the marker data
,
'Since we copied the QTM_TGetQTM_TFrameIdentified3D structure into GetQTM_TFrameIdentified3D,

```

'the address of the marker data is stored at GetQTM\_TFrameIdentified3D.frameBody()  
'We are copying directly into dataAddress so pass it by reference, AddressOf(dataAddress)  
'The value located at GetQTM\_TFrameIdentified3D.frameBody() is the address of the \*actual\* data  
'so we pass VarPtrArray(GetQTM\_TFrameIdentified3D.frameBody()) by value - we do NOT want to copy  
'the contents of AddressOf(VarPtrArray(GetQTM\_TFrameIdentified3D.frameBody())) to dataAddress,  
'we want to copy the contents of VarPtrArray(GetQTM\_TFrameIdentified3D.frameBody()) to dataAddress,  
'so we pass VarPtrArray(GetQTM\_TFrameIdentified3D.frameBody()) by value  
CopyMemory dataAddress, ByVal VarPtrArray(GetQTM\_TFrameIdentified3D.frameBody()), 4

'GetQTM\_TFrameIdentified3D.frameBody() contains the address of the marker data. However,  
'in order for GetQTM\_TFrameIdentified3D.frameBody() to be a proper VB6 array, it should point  
'to a SafeArray that, in turn, points to the marker data. The next couple of statements adjusts  
'GetQTM\_TFrameIdentified3D.frameBody() to ensure that it's a proper VB6 array

'Zero GetQTM\_TFrameIdentified3D.frameBody() prior to the following ReDim operation.  
,

'This will prevent any nasty side effects, normally when you ReDim a variable size array, the  
'original data is destroyed. Since GetQTM\_TFrameIdentified3D.frameBody() does not "own" any data  
'it should not destroy any data on a ReDim. To ensure this is the case we "zero"  
'GetQTM\_TFrameIdentified3D.frameBody() prior to the ReDim. Since we wish to zero the  
'contents located at memory address VarPtrArray(GetQTM\_TFrameIdentified3D.frameBody()), and  
'NOT at AddressOf(VarPtrArray(GetQTM\_TFrameIdentified3D.frameBody())), we pass  
'VarPtrArray(GetQTM\_TFrameIdentified3D.frameBody()) by value  
ZeroMemory ByVal VarPtrArray(GetQTM\_TFrameIdentified3D.frameBody()), 4

If (GetQTM\_TFrameIdentified3D.numberofMarkers > 0) Then

    'ReDim GetQTM\_TFrameIdentified3D.frameBody large enough to hold the available data  
    ,

    'The ReDim statement allocates a SafeArray structure and enough space to hold the specified  
    'amount of data. It then sets GetQTM\_TFrameIdentified3D.frameBody() to point to the allocated  
    'SafeArray structure, thus ensuring that GetQTM\_TFrameIdentified3D.frameBody() is a  
    'proper VB6 array

    ReDim GetQTM\_TFrameIdentified3D.frameBody(GetQTM\_TFrameIdentified3D.numberofMarkers - 1)

    'Copy the marker data into the space just allocated  
    ,

    'Note that the address of GetQTM\_TFrameIdentified3D.frameBody(0) is the address of the  
    'data section of the previously allocated space, therefore we pass the Destination address,  
    'GetQTM\_TFrameIdentified3D.frameBody(0), by value. The Source address is contained in the  
    'dataAddress variable. We do NOT want to copy from AddressOf(dataAddress), we want to copy from  
    'dataAddress, therefore we pass dataAddress by value

        CopyMemory GetQTM\_TFrameIdentified3D.frameBody(0), ByVal dataAddress,  
(GetQTM\_TFrameIdentified3D.numberofMarkers \* LenB(frameIdentified3DBody))

    End If

End Function

Public Function GetQTM\_TFrame6DOF(ByRef moPPUFrame As QTM\_TPPUFrameC) As  
QTM\_TFrame6DOF

    ,

    'Get the QTM\_TFrame6DOF information  
    ,

'VB6 does not have a sizeof operator. However it does have a Len function.

'The Len function operates on variables and NOT data types, therefore we allocate  
'a variable, frame6DOFBody, of type QTM\_TFrame6DOFBody so that

'we can use the Len function on it to determine the size of the QTM\_TFrame6DOFBody

```
'structure
Dim frame6DOFBody As QTM_TFrame6DOFBody
```

```
'The dataAddress variable is used to temporarily store the address of the marker data
Dim dataAddress As Long
```

```
'Copy the QTM_TGetQTM_TFrame6DOF structure into GetQTM_TFrame6DOF
'
```

```
'Note that CopyMemory's Destination and Source are ByVal parameters
'For ByVal parameters the address of the parameter, AddressOf(parameter), is
'passed to the function or sub. It's ok to pass GetQTM_TFrame6DOF by reference
'since we want to copy directly into GetQTM_TFrame6DOF - basically the
'Destination Address is AddressOf(GetQTM_TFrame6DOF). However,
'moPPUFrame.GetQTM_TFrame6DOF is the "actual" Source address. We do NOT want to
'copy the contents of AddressOf(moPPUFrame.GetQTM_TFrame6DOF) to
'GetQTM_TFrame6DOF - we want to copy the contents of
'moPPUFrame.GetQTM_TFrame6DOF. Therefore we pass
'moPPUFrame.GetQTM_TFrame6DOF by value
CopyMemory GetQTM_TFrame6DOF, ByVal moPPUFrame.frame6DOF, LenB(GetQTM_TFrame6DOF)
```

```
'Save the address of the marker data
'
```

```
'Since we copied the QTM_TGetQTM_TFrame6DOF structure into GetQTM_TFrame6DOF,
'the address of the marker data is stored at GetQTM_TFrame6DOF.frameBody()
'We are copying directly into dataAddress so pass it by reference, AddressOf(dataAddress)
'The value located at GetQTM_TFrame6DOF.frameBody() is the address of the *actual* data
'so we pass ByVal VarPtrArray(GetQTM_TFrame6DOF.frameBody()) by value - we do NOT want to copy
'the contents of AddressOf(VarPtrArray(GetQTM_TFrame6DOF.frameBody())) to dataAddress,
'we want to copy the contents of VarPtrArray(GetQTM_TFrame6DOF.frameBody()) to dataAddress,
'so we pass ByVal VarPtrArray(GetQTM_TFrame6DOF.frameBody()) by value
CopyMemory dataAddress, ByVal VarPtrArray(GetQTM_TFrame6DOF.frameBody()), 4
```

```
'GetQTM_TFrame6DOF.frameBody() contains the address of the marker data. However,
'in order for GetQTM_TFrame6DOF.frameBody() to be a proper VB6 array, it should point
'to a SafeArray that, in turn, points to the marker data. The next couple of statements adjusts
'GetQTM_TFrame6DOF.frameBody() to ensure that it's a proper VB6 array
```

```
'Zero GetQTM_TFrame6DOF.frameBody() prior to the following ReDim operation.
'
```

```
'This will prevent any nasty side effects, normally when you ReDim a variable size array, the
'original data is destroyed. Since GetQTM_TFrame6DOF.frameBody() does not "own" any data
'it should not destroy any data on a ReDim. To ensure this is the case we "zero"
'GetQTM_TFrame6DOF.frameBody() prior to the ReDim. Since we wish to zero the
'contents located at memory address VarPtrArray(GetQTM_TFrame6DOF.frameBody()), and
'NOT at AddressOf(VarPtrArray(GetQTM_TFrame6DOF.frameBody())), we pass
'VarPtrArray(GetQTM_TFrame6DOF.frameBody()) by value
ZeroMemory ByVal VarPtrArray(GetQTM_TFrame6DOF.frameBody()), 4
```

```
If (GetQTM_TFrame6DOF.numberofBodies > 0) Then
'ReDim GetQTM_TFrame6DOF.frameBody large enough to hold the available data
'
```

```
'The ReDim statement allocates a SafeArray structure and enough space to hold the specified
'amount of data. It then sets GetQTM_TFrame6DOF.frameBody() to point to the allocated
'SafeArray structure, thus ensuring that GetQTM_TFrame6DOF.frameBody() is a
'proper VB6 array
```

```

ReDim GetQTM_TFrame6DOF.frameBody(GetQTM_TFrame6DOF.numberOfBodies - 1)

'Copy the marker data into the space just allocated
'
'Note that the address of GetQTM_TFrame6DOF.frameBody(0) is the address of the
'data section of the previously allocated space, therefore we pass the Destination address,
'GetQTM_TFrame6DOF.frameBody(0), by value. The Source address is contained in the
'dataAddress variable. We do NOT want to copy from AddressOf(dataAddress), we want to copy from
'dataAddress, therefore we pass dataAddress by value
CopyMemory GetQTM_TFrame6DOF.frameBody(0), ByVal dataAddress,
(GetQTM_TFrame6DOF.numberOfBodies * LenB(frame6DOFBody))
End If
End Function

Public Function GetQTM_TFrameRotationMatrix(ByRef moPPUFrame As QTM_TPPUFrameC) As
QTM_TFrameRotationMatrix
'
'Get the QTM_TFrameRotationMatrix information
'

'VB6 does not have a sizeof operator. However it does have a Len function.
'The Len function operates on variables and NOT data types, therefore we allocate
'a variable, frameRotationBody, of type QTM_TFrameRotationBody so that
'we can use the Len function on it to determine the size of the QTM_TFrameRotationBody
'structure
Dim frameRotationBody As QTM_TFrameRotationBody

'The dataAddress variable is used to temporarily store the address of the marker data
Dim dataAddress As Long

'Copy the QTM_TGetQTM_TFrameRotationMatrix structure into GetQTM_TFrameRotationMatrix
'
'Note that CopyMemory's Destination and Source are ByRef parameters
'For ByRef parameters the address of the parameter, AddressOf(parameter), is
'passed to the function or sub. It's ok to pass GetQTM_TFrameRotationMatrix by reference
'since we want to copy directly into GetQTM_TFrameRotationMatrix - basically the
'Destination Address is AddressOf(GetQTM_TFrameRotationMatrix). However,
'moPPUFrame.GetQTM_TFrameRotationMatrix is the "actual" Source address. We do NOT want to
'copy the contents of AddressOf(moPPUFrame.GetQTM_TFrameRotationMatrix) to
'GetQTM_TFrameRotationMatrix - we want to copy the contents of
'moPPUFrame.GetQTM_TFrameRotationMatrix. Therefore we pass
'moPPUFrame.GetQTM_TFrameRotationMatrix by value
CopyMemory GetQTM_TFrameRotationMatrix, ByVal moPPUFrame.frameRotation,
LenB(GetQTM_TFrameRotationMatrix)

'Save the address of the marker data
'
'Since we copied the QTM_TGetQTM_TFrameRotationMatrix structure into
GetQTM_TFrameRotationMatrix,
'the address of the marker data is stored at GetQTM_TFrameRotationMatrix.frameBody()
'We are copying directly into dataAddress so pass it by reference, AddressOf(dataAddress)
'The value located at GetQTM_TFrameRotationMatrix.frameBody() is the address of the *actual* data
'so we pass VarPtrArray(GetQTM_TFrameRotationMatrix.frameBody()) by value - we do NOT want to copy
'the contents of AddressOf(VarPtrArray(GetQTM_TFrameRotationMatrix.frameBody())) to dataAddress,

```

```

'we want to copy the contents of VarPtrArray(GetQTM_TFrameRotationMatrix.frameBody()) to
dataAddress,
'so we pass VarPtrArray(GetQTM_TFrameRotationMatrix.frameBody()) by value
CopyMemory dataAddress, ByVal VarPtrArray(GetQTM_TFrameRotationMatrix.frameBody()), 4

'GetQTM_TFrameRotationMatrix.frameBody() contains the address of the marker data. However,
'in order for GetQTM_TFrameRotationMatrix.frameBody() to be a proper VB6 array, it should point
'to a SafeArray that, in turn, points to the marker data. The next couple of statements adjusts
'GetQTM_TFrameRotationMatrix.frameBody() to ensure that it's a proper VB6 array

'Zero GetQTM_TFrameRotationMatrix.frameBody() prior to the following ReDim operation.
,

'This will prevent any nasty side effects, normally when you ReDim a variable size array, the
'original data is destroyed. Since GetQTM_TFrameRotationMatrix.frameBody() does not "own" any data
'it should not destroy any data on a ReDim. To ensure this is the case we "zero"
'GetQTM_TFrameRotationMatrix.frameBody() prior to the ReDim. Since we wish to zero the
'contents located at memory address VarPtrArray(GetQTM_TFrameRotationMatrix.frameBody()), and
'NOT at AddressOf(VarPtrArray(GetQTM_TFrameRotationMatrix.frameBody())), we pass
'VarPtrArray(GetQTM_TFrameRotationMatrix.frameBody()) by value
ZeroMemory ByVal VarPtrArray(GetQTM_TFrameRotationMatrix.frameBody()), 4

If (GetQTM_TFrameRotationMatrix.numberofBodies > 0) Then
'ReDim GetQTM_TFrameRotationMatrix.frameBody large enough to hold the available data
,

'The ReDim statement allocates a SafeArray structure and enough space to hold the specified
'amount of data. It then sets GetQTM_TFrameRotationMatrix.frameBody() to point to the allocated
'SafeArray structure, thus ensuring that GetQTM_TFrameRotationMatrix.frameBody() is a
'proper VB6 array
ReDim GetQTM_TFrameRotationMatrix.frameBody(GetQTM_TFrameRotationMatrix.numberofBodies -
1)

'Copy the marker data into the space just allocated
,

'Note that the address of GetQTM_TFrameRotationMatrix.frameBody(0) is the address of the
'data section of the previously allocated space, therefore we pass the Destination address,
'GetQTM_TFrameRotationMatrix.frameBody(0), by value. The Source address is contained in the
'dataAddress variable. We do NOT want to copy from AddressOf(dataAddress), we want to copy from
'dataAddress, therefore we pass dataAddress by value
CopyMemory GetQTM_TFrameRotationMatrix.frameBody(0), ByVal dataAddress,
(GetQTM_TFrameRotationMatrix.numberofBodies * LenB(frameRotationBody))
End If
End Function

```

'This module comes from the SeaMac software  
DefInt A-Z

' Public variables  
Public PortIsOpen As Boolean ' Echo On/Off flag.  
Public CancelSend ' Flag to stop sending a text file.  
Public Handle As Long

Public RxBuffer As String ' holds each buffer  
Public DrvType As Integer

Public Const PORTS\_PER\_DRIVER As Byte = 4  
Public Const NUM\_DRIVERS As Byte = 4

\*\*\*\*\*

' Information structures

\*\*\*\*\*

' Must pass string parameters as Byte arrays. See "Calling DLL  
' Procedures" on MSDN. DLL's have problems with VB string arrays

Type PORT\_DATA  
szPortKeyName(29) As Byte  
dwPresent As Long  
#If Win32 Then  
hPort As Long  
#Else  
hPort As Integer  
#End If  
End Type

Type DRIVER\_INFO  
wDriverType As Integer  
szDrvKeyName(29) As Byte  
dwPresent As Long  
PortData(PORTS\_PER\_DRIVER - 1) As PORT\_DATA  
wNumPorts As Integer  
End Type

Type SEAMAC\_INFO  
wNumDrivers As Integer  
DrvInfo(NUM\_DRIVERS - 1) As DRIVER\_INFO  
End Type

Type PORT\_INFO  
' ... Future  
dwReserved0 As Long ' Reserved double words \*/  
dwReserved1 As Long  
dwReserved2 As Long  
dwReserved3 As Long

' State capabilities  
dwStateCapabilities As Long

' ...system resources  
wBaseIO As Integer ' I/O address  
wMemoryBase As Integer ' Store for DeviceInfo Call

```

wCH_TxDMA As Integer      ' Transmitter DMA channel
wCH_RxDMA As Integer      ' Receiver DMA channel
wIRQ As Integer           ' Hardware interrupt

' ... Buffer information
wTxFrameSize As Integer   ' Size of each Tx frame buffer
wRxFrameSize As Integer   ' Size of each Rx frame buffer
wTotalTxFrames As Integer ' Total Tx Frames
wTotalRxFrames As Integer ' Total Rx Frames

' ... Adapter setup
wCardNumber As Integer    ' Model of adapter
wChannel As Integer       ' A, B, etc.
wElecInterface As Integer ' Electrical Interface
wDPLLMode As Integer      ' DPLL mode of operation
wFlags As Integer         ' Driver flags
wNodeAddress As Integer   ' SDLC Address Reg
wCRCMode As Integer       ' type of CRC
wRunmode As Integer       ' Full dup., Half dup., Interrupt
wChip As Integer          ' SCC, ESCC, other
'szOscillator As String * 10 ' Informational
szOscillator(9) As Byte   ' Informational
End Type

Type PORT_STATE
' ... Future
dwReserved0 As Long      ' Reserved double words */
dwReserved1 As Long
dwReserved2 As Long
dwReserved3 As Long

'Communication Settings
dwRxTimeout As Long
dwTxTimeout As Long
dwBlockOnRead As Long
dwBlockOnWrite As Long
wDivisor As Integer
wClockx As Byte
wDataBits As Byte
wStopBits As Byte
wParity As Byte
bNodeAddress As Byte
bCRCPreset As Byte      ' CRC Preset Option - runtime
bDPLLMode As Byte

' Adapter Setup
wTxClock As Byte
wRxClock As Byte
wBRGSource As Byte
wLoopBack As Byte
wRS485 As Byte

' ... operational status
dwTxFrameCount As Long   ' Total Tx Frames sent
dwRxFrameCount As Long   ' Total Rx Frames received
dwTxPacketCount As Long  ' Packets can span multiple

```

```
dwRxPacketCount As Long ' frames on some products
dwCRCErrors As Long ' Total CRC Errors
dwOverrunErrors As Long ' Total overrun errors
dwParityErrors As Long ' Total parity errors
dwOptions As Long ' Misc Init options from init sequence
dwStatus As Long ' Misc Status
End Type
```

```
*****
```

```
' Clocking Options
```

```
*****/
```

```
Public Const RXC As Byte = 0
Public Const TXC As Byte = 1
Public Const BRG As Byte = 2
Public Const DPLL As Byte = 3
```

```
*****
```

```
' Baud Rate Generator Clock Source Options
```

```
*****/
```

```
Public Const BRG_PCLK As Byte = 0
Public Const BRG_RTxC As Byte = 1
```

```
*****
```

```
' Electrical Interface Options
```

```
*****/
```

```
Public Const EIA_530 As Byte = 1
Public Const RS_232 As Byte = 2
Public Const V35 As Byte = 4
Public Const MIL_188 As Byte = 8
```

```
' DPLL Modes
```

```
Public Const DPLL_None As Byte = &H0
Public Const DPLL_NRZ As Byte = &H1
Public Const DPLL_NRZI As Byte = &H2
Public Const DPLL_FM0 As Byte = &H4
Public Const DPLL_FM1 As Byte = &H8
Public Const DPLL_Src_Brg As Byte = &H10
Public Const DPLL_Src_RTxc As Byte = &H20
Public Const DPLL_Manchester As Byte = &H40
```

```
' CRC Modes
```

```
Public Const CRC_Pre0 As Byte = &H1
Public Const CRC_pre1 As Byte = &H2
Public Const CRC_CCITT As Byte = &H4
```

```
*****
```

```
' Chip Options
```

```
*****/
```

```
Public Const SCC As Byte = 0
Public Const ESCC As Byte = 1
```

```
*****
```

```
' LoopBack Options
```

```
*****/
```

```
Public Const LoopBackON As Byte = 1
```

```

Public Const LoopBackOFF As Byte = 0

'*****
' RS485 Options
'*****/
Public Const RS485ON As Byte = 1
Public Const RS485OFF As Byte = 0

'*****
' Parity Options
'*****/
Public Const NO_PARITY As Byte = 0
Public Const EVEN_PARITY As Byte = 3
Public Const ODD_PARITY As Byte = 1

'*****
' Card Type Options
'*****/
Public Const ACB_III As Byte = 0
Public Const ACB_IV As Byte = 1
Public Const ACB_V As Byte = 2
Public Const ACB_VI As Byte = 3
Public Const ACB_56 As Byte = 4
Public Const ACB_530 As Byte = 5
Public Const ACB_104 As Byte = 6
Public Const Route_56 As Byte = 7
Public Const ACB_II As Byte = 8
Public Const C3_104 As Byte = 9
Public Const CARD_LAST As Byte = 10

'*****
' Return Status Codes from API
' These codes are also used to report errors to the API interface.
'*****/
Public Const SEAMAC_ERROR_NONE As Long = 0 ' No Error
Public Const SEAMAC_ERROR_TXBUFF As Long = 1 ' Tx Buffer Error
Public Const SEAMAC_ERROR_RXBUFF As Long = 2 ' Rx Buffer Error
Public Const SEAMAC_ERROR_CRC As Long = 3 ' CRC Error
Public Const SEAMAC_ERROR_PARITY As Long = 4 ' Parity Error
Public Const SEAMAC_ERROR_OVERRUN As Long = 5 ' Overrun Error
Public Const SEAMAC_ERROR_ISR_LOOP As Long = 6 ' Internal isr error
Public Const SEAMAC_ERROR_SP_RX_INT As Long = 7
Public Const SEAMAC_ERROR_RX_INT As Long = 8
Public Const SEAMAC_ERROR_TX_INT As Long = 9
Public Const SEAMAC_ERROR_EXT_STAT_INT As Long = 10
Public Const SEAMAC_ERROR_PARAM As Long = 11 ' invalid parameter passed to driver
Public Const SEAMAC_ERROR_DMA_FRAME As Long = 12 ' DMA Frame Buffer overflow
Public Const SEAMAC_ERROR_NOT_SUPPORTED As Long = 13 ' Feature not supported for this driver
Public Const SEAMAC_ERROR_PORT_NOT_OPEN As Long = 14 ' Invalid port handle in DLL
structures
Public Const SEAMAC_ERROR_INVALID_NAME As Long = 15 ' OS does not recognize device name
Public Const SEAMAC_ERROR_SHARING_VIOLATION As Long = 16 ' Someone else has the device
open
Public Const SEAMAC_ERROR_PORT_NUM As Long = 17
Public Const SEAMAC_ERROR_DRIVER_TYPE As Long = 18
Public Const SEAMAC_ERROR_PROCESS_ID As Long = 19

```

```
Public Const SEAMAC_ERROR_BUFFER As Long = 20
Public Const SEAMAC_ERROR_INCOMPLETE As Long = 21
Public Const SEAMAC_ERROR_GENERAL As Long = 22
Public Const SEAMAC_ERROR_INTERNAL As Long = 23
Public Const SEAMAC_ERROR_LAST As Long = 24
```

```
*****
```

```
' Modem Control Options
```

```
*****
```

```
Public Const DTR_ON As Integer = &H1
Public Const RTS_ON As Integer = &H2
Public Const LL_ON As Integer = &H4
Public Const RL_ON As Integer = &H8
Public Const DSR_ON As Integer = &H10
Public Const DCD_ON As Integer = &H20
Public Const CTS_ON As Integer = &H40
Public Const RI_ON As Integer = &H80
Public Const TMA_ON As Integer = &H100
```

```
*****
```

```
' Function Codes for API
```

```
*****
```

```
'Public Const ACB_DIOC_OPEN As Byte = 0
'Public Const ACB_DIOC_GETVERSION As Byte = 1
'Public Const ACB_DIOC_GETINFO As Byte = 2
'Public Const ACB_DIOC_SETNODEADDRESS As Byte = 3
'Public Const ACB_DIOC_GETBUFFERCOUNT As Byte = 4
'Public Const ACB_DIOC_GETDATA As Byte = 5
'Public Const ACB_DIOC_PUTDATA As Byte = 6
'Public Const ACB_DIOC_RESETBUFFER As Byte = 7
'Public Const ACB_DIOC_CLOSE As Byte = 8
'Public Const ACB_DIOC_LAST As Byte = 9
```

```
*****
```

```
' Misc Const
```

```
*****
```

```
Public Const RESET_RX_BUFFER As Byte = 1
Public Const RESET_TX_BUFFER As Byte = 2
```

```
*****
```

```
' SEAMAC Driver Types
```

```
*****
```

```
Public Const ASYNC530 As Integer = &H0
Public Const HDLC530 As Integer = &H1
Public Const MSBS530 As Integer = &H2
Public Const HDLCC32 As Integer = &H3
Public Const DRIVER_LAST As Integer = &H4
```

```
*****
```

```
' SEAMAC Communication Channels
```

```
*****
```

```
Public Const CHANNEL_A As Integer = &H0
Public Const CHANNEL_B As Integer = &H1
Public Const CHANNEL_LAST As Integer = &H2
```

```
*****
```

```

' Run-Time Options - wOptions return in CARD_INFO
*****
Public Const RTOP_Z85230 As Integer = &H1      ' ESCC 85230 detected on init
Public Const RTOP_Z8530 As Integer = &H0      ' SCC 8530 detected on init
Public Const RTOP_RS485 As Integer = &H2      ' RS-485 selected in init file
Public Const RTOP_ALTDPLLSRC As Integer = &H4  ' RTXC Pin = DPLL Source
Public Const RTOP_ALTBGRSRC As Integer = &H8   ' RTXC Pin = BRG Source

Public PortInfo As PORT_INFO
Public PortState As PORT_STATE
Public SeamacInfo As SEAMAC_INFO

*****
' Function Declarations for DLL exports
*****
#If Win32 Then
    Declare Sub SetWindowPos Lib "user32" (ByVal hWnd As Long, ByVal hWndInsertAfter As Long, ByVal x As Long, ByVal y As Long, ByVal cx As Long, ByVal cy As Long, ByVal wFlags As Long)
#Else
    Declare Sub SetWindowPos Lib "c:\win95\USER" (ByVal hWnd%, ByVal hWndInsertAfter%, ByVal x%, ByVal y%, ByVal cx%, ByVal cy%, ByVal wFlags%)
#End If

' Note: All pointers passed from Visual Basic to an external function are 32-bit far pointers.
' The above applies to the fpbuffer string pointers in the declares for GetData and PutData
' See "Extending Visual Basic with Microsoft Windows DLLs" on MSDN CD
' NOTE: Use Sub vs Function if C return type is void (i.e. has no return value)

#If Win32 Then
    Declare Function SEAMAC_GetSystemInfo Lib "seamac32.dll" (pStruct As SEAMAC_INFO) As Long
    Declare Function SEAMAC_PortOpen Lib "seamac32.dll" (ByVal wDriverType As Integer, ByVal wPortNumber As Integer, pHandle As Long) As Long
    Declare Function SEAMAC_PortClose Lib "seamac32.dll" (ByVal Handle As Long) As Long
    Declare Function SEAMAC_SetNodeAddress Lib "seamac32.dll" (ByVal Handle As Long, ByVal bAddress As Byte) As Long
    Declare Function SEAMAC_SetModemControl Lib "seamac32.dll" (ByVal Handle As Long, ByVal wMask As Integer) As Long
    Declare Function SEAMAC_GetModemControl Lib "seamac32.dll" (ByVal Handle As Long, wStatus As Integer) As Long
    Declare Function SEAMAC_GetBufferCount Lib "seamac32.dll" (ByVal Handle As Long, wRxCount, wTxCount) As Long
    Declare Function SEAMAC_GetPortInfo Lib "seamac32.dll" (ByVal Handle, pStruct As PORT_INFO) As Long
    Declare Function SEAMAC_GetPortState Lib "seamac32.dll" (ByVal Handle, pStruct As PORT_STATE) As Long
    Declare Function SEAMAC_SetPortState Lib "seamac32.dll" (ByVal Handle, pStruct As PORT_STATE) As Long
    Declare Function SEAMAC_GetData Lib "seamac32.dll" (ByVal Handle As Long, ByVal fpBuffer As String, ByVal dwLength As Long, dwRetLength As Long) As Long
    Declare Function SEAMAC_PutData Lib "seamac32.dll" (ByVal Handle As Long, ByVal fpBuffer As String, ByVal dwLength As Long) As Long
    Declare Function SEAMAC_TransmitCommChar Lib "seamac32.dll" (ByVal Handle As Long, ByVal bBuffer As Byte) As Long
    Declare Function SEAMAC_ResetBuffers Lib "seamac32.dll" (ByVal Handle As Long, ByVal bBuffer As Byte) As Long
#Else

```

```
    Declare Function SEAMAC_GetSystemInfo Lib "seamac16.dll" (pStruct As SEAMAC_INFO) As Long
    Declare Function SEAMAC_PortOpen Lib "seamac16.dll" (ByVal wDriverType As Integer, ByVal
wPortNumber As Integer, Handle As Long) As Long
    Declare Function SEAMAC_PortClose Lib "seamac16.dll" (ByVal Handle As Long) As Long
    Declare Function SEAMAC_SetNodeAddress Lib "seamac16.dll" (ByVal Handle As Long, ByVal bAddress
As Byte) As Long
    Declare Function SEAMAC_SetModemControl Lib "seamac16.dll" (ByVal Handle As Long, ByVal wMask
As Integer) As Long
    Declare Function SEAMAC_GetModemControl Lib "seamac16.dll" (ByVal Handle As Long, wStatus As
Integer) As Long
    Declare Function SEAMAC_GetBufferCount Lib "seamac16.dll" (ByVal Handle As Long, wRxCount,
wTxCount) As Long
    Declare Function SEAMAC_GetPortInfo Lib "seamac16.dll" (ByVal Handle As Long, pStruct As
PORT_INFO) As Long
    Declare Function SEAMAC_GetPortState Lib "seamac16.dll" (ByVal Handle As Long, pStruct As
PORT_STATE) As Long
    Declare Function SEAMAC_SetPortState Lib "seamac16.dll" (ByVal Handle As Long, pStruct As
PORT_STATE) As Long
    Declare Function SEAMAC_GetData Lib "seamac16.dll" (ByVal Handle As Long, ByVal fpBuffer As
String, ByVal dwLength As Long, dwRetLength As Long) As Long
    Declare Function SEAMAC_PutData Lib "seamac16.dll" (ByVal Handle As Long, ByVal fpBuffer As
String, ByVal dwLength As Long) As Long
    Declare Function SEAMAC_TransmitCommChar Lib "seamac16.dll" (ByVal Handle As Long, ByVal
bBuffer As Byte) As Long
    Declare Function SEAMAC_ResetBuffers Lib "seamac16.dll" (ByVal Handle As Long, ByVal bBuffer As
Byte) As Long
#End If
```

```
*****
' End of file
*****
```

```

Public Declare Function TrackSetModel Lib "Tracking.dll" (ByVal numMarkers As Integer, ByRef x_array As
Double, ByRef y_array As Double, ByRef z_array As Double) As Long
Public Declare Function TrackInit Lib "Tracking.dll" () As Long
Public Declare Function TrackProcessMarkers Lib "Tracking.dll" ( _
    ByVal numMarkers As Integer, _
    ByRef mx_array As Double, _
    ByRef my_array As Double, _
    ByRef mz_array As Double, _
    ByRef merr_array As Double, _
    ByRef numMatched As Integer, _
    ByRef body_array As Double, _
    ByRef fitx_array As Double, _
    ByRef fity_array As Double, _
    ByRef fitz_array As Double, _
    ByRef fiterr_array As Double, _
    ByRef fitAvg As Double _
) As Long

Public Function LoadBodyFile(filename As String)
    Dim F As Integer

    LoadBodyFile = False
    On Error Resume Next

    F = FreeFile

    Open filename For Input As #F
    If Err.Number = 0 Then
        Dim sArray() As String
        Dim S As String
        Dim numMarkers As Integer
        Dim xCoord(0 To 25) As Double
        Dim yCoord(0 To 25) As Double
        Dim zCoord(0 To 25) As Double

        numMarkers = 0
        Do While Not EOF(F)
            Line Input #F, S
            sArray = split(Pack(S))
            If UBound(sArray) = 3 Then
                ' We have 3 words on the line, assume it's a valid body file
                xCoord(numMarkers) = val(sArray(1))
                yCoord(numMarkers) = val(sArray(2))
                zCoord(numMarkers) = val(sArray(3))
                numMarkers = numMarkers + 1
            End If
        Loop

        If numMarkers >= 3 Then
            TrackSetModel numMarkers, xCoord(0), yCoord(0), zCoord(0)
            LoadBodyFile = True
        End If

        numModelMarkers = numMarkers
    End If

```

```
Close #F  
End Function
```

```
Private Function Pack(str As String) As String  
    Dim words As Variant  
    Dim x As Long  
    Dim temp As String  
  
    words = split(str, " ")  
    For x = LBound(words) To UBound(words)  
        If words(x) <> "" Then  
            temp = temp & " " & words(x)  
        End If  
    Next x  
    Pack = temp  
End Function
```

'DAC board function declarations

' access32.dll is a general purpose port access library

Public Declare Function OutPort Lib "ACCES32" Alias "VBOutPort" (ByVal Port As Long, ByVal Value As Long) As Integer

Public Declare Function InPortB Lib "ACCES32" Alias "VBInPortB" (ByVal Port As Long) As Integer

'Process priority calls

Public Enum PRIORITY\_CLASS

NORMAL\_PRIORITY\_CLASS = &H20

IDLE\_PRIORITY\_CLASS = &H40

HIGH\_PRIORITY\_CLASS = &H80

REALTIME\_PRIORITY\_CLASS = &H100

End Enum

Public Const PROCESS\_DUP\_HANDLE = &H40

Public Const PROCESS\_SET\_INFORMATION = &H200

Public Declare Function OpenProcess Lib "Kernel32" (ByVal dwDesiredAccess As Long, ByVal bInheritHandle As Long, ByVal dwProcessId As Long) As Long

Public Declare Function CloseHandle Lib "Kernel32" (ByVal hObject As Long) As Long

Public Declare Function GetCurrentProcessId Lib "Kernel32" () As Long

Public Declare Function SetPriorityClass Lib "Kernel32" (ByVal hProcess As Long, ByVal dwPriorityClass As Long) As Long

Public Declare Function GetLastError Lib "Kernel32" () As Long

'Timer function declarations

Public Declare Function QueryPerformanceCounter Lib "Kernel32" (lpPerformanceCount As Currency) As Boolean

Public Declare Function QueryPerformanceFrequency Lib "Kernel32" (lpPerformanceFreq As Currency) As Boolean

Public curStartTime As Currency

Public curEndTime As Currency

Public curFreq As Currency

Public curCurTime As Currency

Public curLastCall As Currency

Public curNow As Currency

Public Const defaultConfig As String = \_

0 & vbTab & "#Output Selection (0 = Body, 1 = Markers)" & vbCrLf & \_

768 & vbTab & "#DAC board base address (decimal)" & vbCrLf & \_

-25000 & vbTab & "#Max X value" & vbCrLf & \_

-35000 & vbTab & "#Min X value" & vbCrLf & \_

18000 & vbTab & "#Max Y value" & vbCrLf & \_

9000 & vbTab & "#Min Y value" & vbCrLf & \_

2000 & vbTab & "#Max Z value" & vbCrLf & \_

-1000 & vbTab & "#Min Z value" & vbCrLf & \_

190 & vbTab & "#Max Roll value" & vbCrLf & \_

-190 & vbTab & "#Min Roll value" & vbCrLf & \_

190 & vbTab & "#Max Pitch value" & vbCrLf & \_

-190 & vbTab & "#Min Pitch value" & vbCrLf & \_

190 & vbTab & "#Max Yaw value" & vbCrLf & \_

-190 & vbTab & "#Min Yaw value" & vbCrLf & \_

200 & vbTab & "#Max RMS value" & vbCrLf & \_

-10 & vbTab & "#Min RMS value" & vbCrLf & \_

50 & vbTab & "#RMS error value" & vbCrLf & \_

10 & vbTab & "#Angle Hysteresis size" & vbCrLf & \_

25 & vbTab & "#Graph max value" & vbCrLf & \_  
-10 & vbTab & "#Graph min value" & vbCrLf & \_  
300 & vbTab & "#Number of Data Points in Graph" & vbCrLf

Public logString As String

'DAC board addresses

'768 = 300 Hex

Public BASE As Integer 'Base address of DAC card

Public AUTO As Integer 'Turn off simultaneous mode

Public DAC0 As Integer

Public DAC1 As Integer

Public DAC2 As Integer

Public DAC3 As Integer

Public DAC4 As Integer

Public DAC5 As Integer

Public DAC6 As Integer

Public DAC7 As Integer

Public DAC8 As Integer

Public DAC9 As Integer

Public DAC10 As Integer

Public DAC11 As Integer

Public DAC12 As Integer

Public DAC13 As Integer

Public DAC14 As Integer

Public DAC15 As Integer

'Port B variables

Public errB As Long

Public handleB As Long

Public RxBufferB As String

Public PortInfoB As PORT\_INFO

'File I/O

Public openedFilename As String 'filename

Public fileObj As New FileSystemObject

Public fileStream As Scripting.TextStream

Public fileOpen As Boolean 'status of log file

Public fitFileStream As Scripting.TextStream

Public fitFileOpen As Boolean

Public numModelMarkers As Integer

'Error Checking

Public chkSumCount As Long 'number of checksum errors encountered

Public droppedFrames As Long 'number of bad frames received

'max/min values for analog output

Public MaxX As Long

Public MinX As Long

Public MaxY As Long

Public MinY As Long

Public MaxZ As Long

Public MinZ As Long

```
Public MaxR As Long
Public MinR As Long
```

```
Public MaxP As Long
Public MinP As Long
```

```
Public MaxYw As Long
Public MinYw As Long
```

```
Public MaxRes As Long
Public MinRes As Long
```

```
Public ErrRMS As Long
Dim moPPUFrameVB As QTMClientVB.QTM_TPPUFrame
```

```
'used in radian to degree conversion
' 180/pi
Public Const radToDeg As Single = 57.295
```

```
'used in angle hysteresis
Public hysteresis As Integer
```

```
'Graph variables
Public m_nRealTimeCounter As Long
Public GraphMin As Long
Public GraphMax As Long
Public GraphPoints As Long
```

```
'Output selection
Public outputBody As Boolean
'Processing Selection
Public selspot As Boolean
```

```
'Changes the priority of the process
Public Function ChangePriority(dwPriorityClass As PRIORITY_CLASS) As Boolean
    Dim hProcess As Long
    Dim ret As Long, pid As Long
    pid = GetCurrentProcessId()
    hProcess = OpenProcess(PROCESS_SET_INFORMATION, True, pid)
    If hProcess = 0 Then Exit Function
    ret = SetPriorityClass(hProcess, dwPriorityClass)
    Call CloseHandle(hProcess)
    ChangePriority = ret <> 0
End Function
```

```
'-----
'verifyChkSum
' Takes frame data and verifys the checksum value
'-----
Public Function verifyChkSum(ByRef data() As Byte, ByVal Length As Integer) As Boolean

    Dim chkSum As Integer
    Dim valid As Boolean
```

```
'calculate checksum value of frame
```

```

chkSum = 0
For i = 0 To Length - 3
    chkSum = chkSum Xor data(i)
Next

'verify checksum value
If chkSum = data(Length - 2) Then
    valid = True
Else
    valid = False
End If

    verifyChkSum = valid
End Function

'-----
'toSingle
' Converts 4 bytes to 32-bit single precision number
' Based on IEEE-754
' value = (sign)(-1) * 2^(exponent - 127) * 1.(mantissa)
'-----
Public Function toSingle(ByVal a As Byte, ByVal b As Byte, ByVal c As Byte, ByVal d As Byte) As Single
    Dim sign As Long    'sign
    Dim exp As Long    'exponent
    Dim temp As Long
    Dim mant As Single  'mantissa
    Dim val As Single   'calculated FP number

    If a > 127 Then
        sign = 1
    Else:
        sign = 0
    End If

    exp = (((a And 127) * 2) Or ((b And 128) / 128)) - 127

    temp = d + 2 ^ 8 * c + 2 ^ 16 * (b And 127)

    mant = 1 + temp / (2 ^ 23)

    If sign = 1 Then
        val = -1 * mant * 2 ^ exp
    Else: val = mant * 2 ^ exp
    End If

    toSingle = val
End Function

'-----
'limit
' limits DAC output to 0 -> 65535
'-----
Public Function limit(ByVal b As Long) As Long
    If b > 65535 Then
        b = 65535
    ElseIf b < 0 Then

```

```
    b = 0
End If
limit = b
```

```
End Function
```

```
-----
'startGraph
' Initializes graph parameters
' and enables graph timer
-----
```

```
Public Sub startGraph()
```

```
Dim dwinindex As Single
```

```
    Main.pltRMS.Visible = True
```

```
    ** Chart fills 100 points but x axis is initially
```

```
    ** manually scaled. Once 100 point have been passed,
```

```
    ** the chart switches to autoscaling the x axis.
```

```
    Main.pltRMS.Subsets = 2
```

```
    Main.pltRMS.Points = GraphPoints
```

```
    Main.pltRMS.SubsetLineTypes(0) = PELT_MEDIUM_SOLID
```

```
    Main.pltRMS.SubsetColors(0) = RGB(255, 0, 0)
```

```
    Main.pltRMS.SubsetLineTypes(1) = PELT_MEDIUM_SOLID
```

```
    Main.pltRMS.SubsetColors(1) = RGB(0, 255, 0)
```

```
' Main.pltRMS.MultiAxesSubsets(0) = 1
```

```
' Main.pltRMS.MultiAxesSubsets(1) = 1
```

```
' Main.pltRMS.MultiAxisStyle = PEMAS_SEPARATE_AXES
```

```
    ** Set Auto Y scale **'
```

```
' Main.pltRMS.WorkingAxis = 0
```

```
' Main.pltRMS.ManualScaleControlY = PEMSC_MINMAX
```

```
' Main.pltRMS.ManualMinY = GraphMin
```

```
' Main.pltRMS.ManualMaxY = GraphMax
```

```
' Main.pltRMS.WorkingAxis = 1
```

```
    Main.pltRMS.ManualScaleControlY = PEMSC_MINMAX
```

```
    Main.pltRMS.ManualMinY = GraphMin
```

```
    Main.pltRMS.ManualMaxY = GraphMax
```

```
    ** Set Manual X scale **'
```

```
' Main.pltRMS.ManualScaleControlX = Graph.PEMSC_MINMAX
```

```
' Main.pltRMS.ManualMinX = 1
```

```
' Main.pltRMS.ManualMaxX = 100
```

```
    ** Clear out default data **'
```

```
    Main.pltRMS.XData(0, 0) = 0
```

```
    Main.pltRMS.XData(0, 1) = 0
```

```
    Main.pltRMS.XData(0, 2) = 0
```

```
    Main.pltRMS.XData(0, 3) = 0
```

```
    Main.pltRMS.YData(0, 0) = 0
```

```
    Main.pltRMS.YData(0, 1) = 0
```

```
    Main.pltRMS.YData(0, 2) = 0
```

```
    Main.pltRMS.YData(0, 3) = 0
```

```

*** Set various properties ***
Main.pltRMS.MainTitle = "Body RMS Error"
Main.pltRMS.SubTitle = vbNullString
Main.pltRMS.XAxisLabel = "Frames"
Main.pltRMS.YAxisLabel = "Residual"
Main.pltRMS.NoRandomPointsToExport = True
Main.pltRMS.FocalRect = False
Main.pltRMS.AllowBar = False
Main.pltRMS.AllowPopup = False
Main.pltRMS.DeskColor = RGB(192, 192, 192)

Main.pltRMS.PEactions = 0

' Initialize Counters and Timer '
m_nRealTimeCounter = 1
m_nSinCounter = 1

End Sub
Public Sub ClearGraphData()
    Main.pltRMS.XData(0, -1) = 0
    Main.pltRMS.YData(0, -1) = 0

    Main.pltRMS.PEnarg1 = 0
    Main.pltRMS.PEnarg2 = 0
    Main.pltRMS.PEactions = REINITIALIZE_RESETIMAGE
End Sub

Public Sub ClearGridData()
    With Main.vbalGridBody
        .Clear

        .AddRow

        .CellDetails 1, 1, "Frame Count"
        SetRowColors 1, HEADER_FOREGROUND_COLOR, HEADER_BACKGROUND_COLOR
    End With
End Sub

'-----
'graphData
' append new value to strip chart
'-----
Public Sub graphData(ByVal fRMS00 As Single, ByVal fRMS01 As Single)
    Dim newy(0 To 1) As Single
    Dim newx(0 To 1) As Single
    Dim t As Long

    *** New y value and x value ***
    newy(0) = fRMS00
    newx(0) = m_nRealTimeCounter
    newy(1) = fRMS01
    newx(1) = m_nRealTimeCounter

    *** Append new values ***
    t = PE5API.PEvset(Main.pltRMS, PE5API.PEP_faAPPENDYDATA, newy(0), 1)

```

```
t = PE5API.PEvset(Main.pltRMS, PE5API.PEP_faAPPENDXDATA, newx(0), 1)

'** Increment counter **'
m_nRealTimeCounter = m_nRealTimeCounter + 1

'** Switch to AutoScaling x axis after receiving 100 data points **'
'If m_nRealTimeCounter = 100 Then Main.pltRMS.ManualScaleControlX = Graph.PEMSC_NONE

Main.pltRMS.PEnarg1 = 0
Main.pltRMS.PEnarg2 = 0
'Main.pltRMS.PEactions = 0 '// Calls PErresetimage
Main.pltRMS.PEactions = REINITIALIZE
Main.pltRMS.PEactions = RESET_IMAGE
Main.pltRMS.PEactions = INVALIDATE_RECT
End Sub
```

'Frame 3

```
Private frameCount As Currency 'the current frame number
Private numBytes As Long 'number of bytes contained in frame
Private numMarkers As Long 'number of markers defined in frame
```

```
Private outStr As String 'screen output string
```

```
'Arrays to hold marker values
```

```
Private xCoords() As Double
```

```
Private yCoords() As Double
```

```
Private zCoords() As Double
```

```
Private residual() As Double
```

```
'Arrays to hold raw byte data
```

```
Private x() As Byte
```

```
Private y() As Byte
```

```
Private Z() As Byte
```

```
Private res() As Byte
```

```
'arrays to hold Dac output values
```

```
Private DACx As Long
```

```
Private DACy As Long
```

```
Private DACz As Long
```

```
Private DACres As Long
```

```
'-----
```

```
'passData
```

```
' takes byte array and extracts/converts frame
```

```
' values
```

```
'-----
```

```
Public Function passDataFromFile(ByRef line As String) As Long
```

```
Dim sArray() As String
```

```
Dim c As Integer
```

```
sArray = split(line, vbTab)
```

```
If UBound(sArray) < 6 Then
```

```
passDataFromFile = -2
```

```
Exit Function
```

```
End If
```

```
frameCount = val(sArray(0))
```

```
numMarkers = val(sArray(1))
```

```
If (numMarkers > 20) Then
```

```
passDataFromFile = -2
```

```
Exit Function
```

```
End If
```

```
ReDim xCoords(numMarkers) As Double
```

```
ReDim yCoords(numMarkers) As Double
```

```
ReDim zCoords(numMarkers) As Double
```

```
ReDim residual(numMarkers) As Double
```

```
For c = 0 To numMarkers - 1
```

```

    xCoords(c) = val(sArray(c * 4 + 2))
    yCoords(c) = val(sArray(c * 4 + 3))
    zCoords(c) = val(sArray(c * 4 + 4))
    residual(c) = val(sArray(c * 4 + 5))
Next c

    outputFrame
    passDataFromFile = numMarkers
Exit Function

End Function

'-----
'passData
' takes byte array and extracts/converts frame
' values
'-----
Public Function passData(ByRef inByte() As Byte) As Long
    Dim count As Long
    Dim c As Long
    Dim j As Long

    'Handle Errors
    On Error GoTo BadFrame

    If inByte(0) <> 1 Then
        passData = -1
        Exit Function
    End If

    If inByte(3) <> 3 Then
        passData = -1
        Exit Function
    End If

    'find number of bytes in frame
    numBytes = inByte(1) + 2 ^ 8 * inByte(2)

    'verify checksum
    If Not verifyChkSum(inByte, numBytes) Then
        passData = -1
        Exit Function
    End If

    count = 0

    frameCount = inByte(4) + 2 ^ 8 * inByte(5) + 2 ^ 16 * inByte(6) + 2 ^ 24 * inByte(7)

    numMarkers = inByte(8)

    If (numMarkers > 20) Then
        passData = -2
        Exit Function
    End If

```

```
ReDim xCoords(numMarkers) As Double
ReDim yCoords(numMarkers) As Double
ReDim zCoords(numMarkers) As Double
ReDim residual(numMarkers) As Double
```

```
ReDim x(4 * numMarkers) As Byte
ReDim y(4 * numMarkers) As Byte
ReDim Z(4 * numMarkers) As Byte
ReDim res(4 * numMarkers) As Byte
```

```
count = 9
```

```
'this section involves stepping through the incoming raw data byte-by-byte
'and extracting the information into a VB friendly form
'the details of this conversion can be found in the MarineTrak Users manual
For c = 0 To numMarkers - 1
```

```
  For j = 0 To 3
    x(4 * c + j) = inByte(count)
    count = count + 1
  Next
```

```
  xCoords(c) = toSingle(x(4 * c + 3), x(4 * c + 2), x(4 * c + 1), x(4 * c + 0))
```

```
  For j = 0 To 3
    y(4 * c + j) = inByte(count)
    count = count + 1
  Next
```

```
  yCoords(c) = -1 * toSingle(y(4 * c + 3), y(4 * c + 2), y(4 * c + 1), y(4 * c + 0))
```

```
  For j = 0 To 3
    Z(4 * c + j) = inByte(count)
    count = count + 1
  Next
```

```
  zCoords(c) = -1 * toSingle(Z(4 * c + 3), Z(4 * c + 2), Z(4 * c + 1), Z(4 * c + 0))
```

```
  For j = 0 To 3
    res(4 * c + j) = inByte(count)
    count = count + 1
  Next
```

```
  residual(c) = toSingle(res(4 * c + 3), res(4 * c + 2), res(4 * c + 1), res(4 * c + 0))
```

```
Next
```

```
count = count + 1
```

```
If inByte(count) <> 4 Then
  passData = -1
  Exit Function
End If
```

```
outputFrame
passData = numBytes
```

```

Exit Function
'Errors treated as bad frame
BadFrame:
    passData = -1
End Function

```

```

'-----
'outputFrame
' produces output for:
'   Screen
'   Analog channels
'   Log File if selected
'-----

```

```

Private Sub outputFrame()
    Dim q As Long

    'outStr = CStr(frameCount) + vbCrLf
    outStr = ""
    logString = logString + CStr(frameCount) + vbTab + CStr(numMarkers) + vbTab
    For q = 0 To numMarkers - 1
        outStr = outStr + _
            right$(" " & CStr(q + 1), 2) & "    " & _
            right$(" " & Format(xCoords(q), "00000.0"), 8) & " " & _
            right$(" " & Format(yCoords(q), "00000.0"), 8) & " " & _
            right$(" " & Format(zCoords(q), "00000.0"), 8) & " " & _
            right$(" " & Format(residual(q), "000.000"), 8) & vbCrLf
        logString = logString + Format(xCoords(q), "00000.0") & vbTab & _
            Format(yCoords(q), "00000.0") & vbTab & _
            Format(zCoords(q), "00000.0") & vbTab & _
            Format(residual(q), "000.000") & vbTab
    Next

```

```

If selspot Then
    Dim body(0 To 6) As Double
    ReDim fitx(0 To numMarkers - 1) As Double
    ReDim fity(0 To numMarkers - 1) As Double
    ReDim fitz(0 To numMarkers - 1) As Double
    ReDim fiterr(0 To numMarkers - 1) As Double
    Dim fitAvg As Double
    Dim numMatched As Integer
    Dim str As String

```

```

    TrackProcessMarkers numMarkers, xCoords(0), yCoords(0), zCoords(0), residual(0), numMatched,
    body(0), fitx(0), fity(0), fitz(0), fiterr(0), fitAvg

```

```

    logString = logString & "1" & vbTab & body(0) & vbTab & body(1) & vbTab & body(2) & vbTab &
    body(3) & vbTab & body(4) & vbTab & body(5) & vbTab & body(6)

```

```

    str = Format(frameCount, "0000000000") & "," & _
        right$(" " & Format(body(0), "00000"), 6) & "," & _
        right$(" " & Format(body(1), "00000"), 6) & "," & _
        right$(" " & Format(body(2), "0000"), 5) & "," & _
        right$(" " & Format(body(3) * radToDeg, "000.0"), 6) & "," & _
        right$(" " & Format(body(4) * radToDeg, "000.0"), 6) & "," & _
        right$(" " & Format(body(5) * radToDeg, "000.0"), 6) & "," & _
        right$(" " & Format(body(6), "00.000"), 7)

```

```
'Main.vbalGridBody.Caption = "frame    x    y    z    roll    pitch    yaw    residual" & vbCrLf & _  
' Replace(str, ",", " ") & vbCrLf
```

```
'graphData body(6)
```

```
' Manage the fit file log
```

```
If numMatched = numModelMarkers Then
```

```
    Dim fitString As String
```

```
    fitString = ""
```

```
    For q = 0 To numMatched - 1
```

```
        fitString = fitString & _
```

```
            fitx(q) & vbCrLf & _
```

```
            fity(q) & vbCrLf & _
```

```
            fitz(q) & vbCrLf & _
```

```
            fiterr(q) & vbCrLf
```

```
    Next q
```

```
    fitString = fitString & fitAvg
```

```
    If fitFileOpen Then
```

```
        fitFileStream.WriteLine fitString
```

```
    End If
```

```
End If
```

```
If outputBody And body(6) <> 50 Then
```

```
    Dim DACx As Long
```

```
    Dim DACy As Long
```

```
    Dim DACz As Long
```

```
    Dim DACr As Long
```

```
    Dim DACp As Long
```

```
    Dim DACyw As Long
```

```
    Dim DACres As Long
```

```
    DACx = ((body(0) - MinX) / (MaxX - MinX)) * 65535
```

```
    DACy = ((body(1) - MinY) / (MaxY - MinY)) * 65535
```

```
    DACz = ((body(2) - MinZ) / (MaxZ - MinZ)) * 65535
```

```
    DACr = ((body(3) * radToDeg - MinR) / (MaxR - MinR)) * 65535
```

```
    DACp = ((body(4) * radToDeg - MinP) / (MaxP - MinP)) * 65535
```

```
    DACyw = ((body(5) * radToDeg - MinYw) / (MaxYw - MinYw)) * 65535
```

```
    DACres = ((body(6) - MinRes) / (MaxRes - MinRes)) * 65535
```

```
    OutPort DAC0, limit(DACx)
```

```
    OutPort DAC1, limit(DACy)
```

```
    OutPort DAC2, limit(DACz)
```

```
    OutPort DAC3, limit(DACr)
```

```
    OutPort DAC4, limit(DACp)
```

```
    OutPort DAC5, limit(DACyw)
```

```
    OutPort DAC6, limit(DACres)
```

```
ElseIf residual(0) = -1 Then
```

```
    DACres = ((ErrRMS - MinRes) / (MaxRes - MinRes)) * 65535
```

```
    OutPort DAC6, limit(DACres)
```

```
End If
```

```
End If
```

```
'terminal output
```

```

Main.vbalGridInfo.Text = _
' left$(CStr(frameCount) & "      ", 10) & " " & _
' "x      " & _
' "y      " & _
' "z      " & _
' "residual" & vbCrLf & _
'outStr

'log file
'If fileOpen And Main.optMarker.Value Then
' fileStream.WriteLine CStr(frameCount) & vbCrLf & outStr
'End If

'If Main.optMarker.Value And residual(0) <> -1 Then
If Not outputBody And residual(0) <> -1 Then
    DACx = ((xCoords(0) - MinX) / (MaxX - MinX)) * 65535
    DACy = ((yCoords(0) - MinY) / (MaxY - MinY)) * 65535
    DACz = ((zCoords(0) - MinZ) / (MaxZ - MinZ)) * 65535

    OutPort DAC0, limit(DACx)
    OutPort DAC1, limit(DACy)
    OutPort DAC2, limit(DACz)

    If numMarkers > 2 Then
        DACx = ((xCoords(1) - MinX) / (MaxX - MinX)) * 65535
        DACy = ((yCoords(1) - MinY) / (MaxY - MinY)) * 65535
        DACz = ((zCoords(1) - MinZ) / (MaxZ - MinZ)) * 65535
        DACres = (((residual(1) + residual(0)) / 2 - MinRes) / (MaxRes - MinRes)) * 65535

        OutPort DAC3, limit(DACx)
        OutPort DAC4, limit(DACy)
        OutPort DAC5, limit(DACz)
        OutPort DAC6, limit(DACres)
    End If

ElseIf residual(0) = -1 Then
    DACres = ((ErrRMS - MinRes) / (MaxRes - MinRes)) * 65535
    OutPort DAC6, limit(DACres)
End If
End Sub

```

'Frame 5

```
Private frameCount As Currency    'the current frame number
Private numBytes As Long          'number of bytes contained in frame
Private numBodies As Long        'number of bodies defined in frame
```

```
Private WDcount As Long          'used to track frame count changes
Private prevFrameCount As Long
Private str As String            'general purpost string
Private fromPositive As Boolean  'used in angle hysteresis
```

```
'Arrays to hold body values
Private xCoords(2) As Single
Private yCoords(2) As Single
Private zCoords(2) As Single
Private roll(2) As Single
Private pitch(2) As Single
Private yaw(2) As Single
Private residual(2) As Single
```

```
'arrays to hold raw body data
Private x(8) As Byte
Private y(8) As Byte
Private Z(8) As Byte
Private r(8) As Byte
Private p(8) As Byte
Private yw(8) As Byte
Private res(8) As Byte
```

```
'DAC output values
Private DACx As Long
Private DACy As Long
Private DACz As Long
Private DACr As Long
Private DACp As Long
Private DACyw As Long
Private DACres As Long
```

```
Public Function passDataFromFile(ByRef line As String) As Long
    Dim sArray() As String
    Dim c As Integer
    Dim numMarkers As Integer
```

```
    sArray = split(line, vbTab)
    If UBound(sArray) < 6 Then
        passDataFromFile = -2
        Exit Function
    End If
```

```
'Grab time for Samples/sec. calculation
curLastCall = curNow
QueryPerformanceCounter curNow
Main.lblSamp.Caption = CLng(1 / ((curNow - curLastCall) / curFreq))
'Running time
Main.lblTime = Format((curNow - curStartTime) / curFreq, "0000000000")
```

```

prevFrameCount = frameCount
frameCount = val(sArray(0))
WDcount = WDcount + 1

numMarkers = val(sArray(1))
numBodies = val(sArray(numMarkers * 4 + 2))

For c = 0 To numBodies
    xCoords(c) = val(sArray(numMarkers * 4 + c * 7 + 3))
    yCoords(c) = val(sArray(numMarkers * 4 + c * 7 + 4))
    zCoords(c) = val(sArray(numMarkers * 4 + c * 7 + 5))
    roll(c) = val(sArray(numMarkers * 4 + c * 7 + 6))
    pitch(c) = val(sArray(numMarkers * 4 + c * 7 + 7))
    yaw(c) = val(sArray(numMarkers * 4 + c * 7 + 8))
    residual(c) = val(sArray(numMarkers * 4 + c * 7 + 9))
Next c

outputFrame
passDataFromFile = numBodies

End Function

'-----
'passData
' takes byte array and extracts/converts frame
' values
'-----
Public Function passData(ByRef inByte() As Byte) As Long
    If selspot Then
        passData = 1
        Exit Function
    End If

    Dim count As Long
    Dim c As Long
    Dim j As Long

    'Error handler
    On Error GoTo BadFrame

    'Grab time for Samples/sec. calculation
    curLastCall = curNow
    QueryPerformanceCounter curNow
    Main.lblSamp.Caption = CLng(1 / ((curNow - curLastCall) / curFreq))
    'Running time
    Main.lblTime = Format((curNow - curStartTime) / curFreq, "0000000000")

    'check for SOH byte
    If inByte(0) <> 1 Then
        passData = -1
        Exit Function
    End If

    'check for frame type
    If inByte(3) <> 5 Then

```

```

    passData = -1
    Exit Function
End If

'find number of bytes in frame
numBytes = inByte(1) + 2 ^ 8 * inByte(2)

'verify checksum
If Not verifyChkSum(inByte, numBytes) Then
    passData = -1
    Exit Function
End If

prevFrameCount = frameCount
frameCount = inByte(4) + 2 ^ 8 * inByte(5) + 2 ^ 16 * inByte(6) + 2 ^ 24 * inByte(7)
WDcount = WDcount + 1

numBodies = inByte(8)

count = 9

'this section involves stepping through the incoming raw data byte-by-byte
'and extracting the information into a VB friendly form
'the details of this conversion can be found in the MarineTrak Users manual
For c = 0 To numBodies - 1
    For j = 0 To 3
        x(4 * c + j) = inByte(count)
        count = count + 1
    Next

    xCoords(c) = toSingle(x(4 * c + 3), x(4 * c + 2), x(4 * c + 1), x(4 * c + 0))

    For j = 0 To 3
        y(4 * c + j) = inByte(count)
        count = count + 1
    Next

    yCoords(c) = -1 * toSingle(y(4 * c + 3), y(4 * c + 2), y(4 * c + 1), y(4 * c + 0))

    For j = 0 To 3
        Z(4 * c + j) = inByte(count)
        count = count + 1
    Next

    zCoords(c) = -1 * toSingle(Z(4 * c + 3), Z(4 * c + 2), Z(4 * c + 1), Z(4 * c + 0))

    For j = 0 To 3
        r(4 * c + j) = inByte(count)
        count = count + 1
    Next

    roll(c) = angHyst(toSingle(r(4 * c + 3), r(4 * c + 2), r(4 * c + 1), r(4 * c + 0)))

    For j = 0 To 3
        p(4 * c + j) = inByte(count)
        count = count + 1
    Next

```

Next

pitch(c) = angHyst(-1 \* toSingle(p(4 \* c + 3), p(4 \* c + 2), p(4 \* c + 1), p(4 \* c + 0)))

For j = 0 To 3

    yw(4 \* c + j) = inByte(count)

    count = count + 1

Next

yaw(c) = angHyst(-1 \* toSingle(yw(4 \* c + 3), yw(4 \* c + 2), yw(4 \* c + 1), yw(4 \* c + 0)))

For j = 0 To 3

    res(4 \* c + j) = inByte(count)

    count = count + 1

Next

residual(c) = toSingle(res(4 \* c + 3), res(4 \* c + 2), res(4 \* c + 1), res(4 \* c + 0))

Next

count = count + 1

If inByte(count) <> 4 Then

    passData = -1

    Exit Function

End If

outputFrame

passData = numBytes

Exit Function

'Errors treated as bad frame

BadFrame:

    passData = -1

End Function

'-----

'outputFrame

' produces output for:

' Screen

' RS-422 output

' Analog channels

' Log File if selected

'-----

Private Sub outputFrame()

Dim q As Long

logString = logString + CStr(numBodies) + vbCrLf

str = ""

For q = 0 To numBodies - 1

    str = str + Format(frameCount, "0000000000") & "," & \_  
        right\$(" " & Format(xCoords(q), "00000"), 6) & "," & \_

```

right$(" " & Format(yCoords(q), "00000"), 6) & "," & _
right$(" " & Format(zCoords(q), "0000"), 5) & "," & _
right$(" " & Format(roll(q) * radToDeg, "000.0"), 6) & "," & _
right$(" " & Format(pitch(q) * radToDeg, "000.0"), 6) & "," & _
right$(" " & Format(yaw(q) * radToDeg, "000.0"), 6) & "," & _
right$(" " & Format(residual(q), "00.000"), 7) & vbCr
logString = logString + _
Format(xCoords(q), "00000") & vbTab & _
Format(yCoords(q), "00000") & vbTab & _
Format(zCoords(q), "0000") & vbTab & _
Format(roll(q) * radToDeg, "000.0") & vbTab & _
Format(pitch(q) * radToDeg, "000.0") & vbTab & _
Format(yaw(q) * radToDeg, "000.0") & vbTab & _
Format(residual(q), "00.000")

'log file
'If fileOpen And Main.optBody.Value Then
' fileStream.WriteLine Replace(str, ",", " ") & vbLf
'End If

Next q

'Display data to screen
'Main.lblBody.Caption = "frame    x    y    z    roll    pitch    yaw    residual" & vbLf & _
'Replace(str, ",", " ") & vbLf

'Transmit data on portB
errB = SEAMAC_PutData(handleB, str, Len(str))

'graphData getRMS

graphData residual(0), residual(1)

'If outputBody And residual(0) <> -1 Then
If (outputBody = True) Then
If (residual(0) > 0 And residual(1) < ErrRMS) Then
DACx = ((xCoords(0) - MinX) / (MaxX - MinX)) * 65535
DACy = ((yCoords(0) - MinY) / (MaxY - MinY)) * 65535
DACz = ((zCoords(0) - MinZ) / (MaxZ - MinZ)) * 65535
DACr = ((roll(0) * radToDeg - MinR) / (MaxR - MinR)) * 65535
DACp = ((pitch(0) * radToDeg - MinP) / (MaxP - MinP)) * 65535
DACyw = ((yaw(0) * radToDeg - MinYw) / (MaxYw - MinYw)) * 65535
DACres = ((residual(0) - MinRes) / (MaxRes - MinRes)) * 65535

OutPort DAC0, limit(DACx)
OutPort DAC1, limit(DACy)
OutPort DAC2, limit(DACz)
OutPort DAC3, limit(DACr)
OutPort DAC4, limit(DACp)
OutPort DAC5, limit(DACyw)
OutPort DAC7, limit(DACres)
Else
DACres = ((ErrRMS - MinRes) / (MaxRes - MinRes)) * 65535
OutPort DAC6, limit(DACres)
End If

```

```

If (residual(1) > 0 And residual(1) < ErrRMS) Then

    DACx = ((xCoords(1) - MinX) / (MaxX - MinX)) * 65535
    DACy = ((yCoords(1) - MinY) / (MaxY - MinY)) * 65535
    DACz = ((zCoords(1) - MinZ) / (MaxZ - MinZ)) * 65535
    DACr = ((roll(1) * radToDeg - MinR) / (MaxR - MinR)) * 65535
    DACp = ((pitch(1) * radToDeg - MinP) / (MaxP - MinP)) * 65535
    DACyw = ((yaw(1) * radToDeg - MinYw) / (MaxYw - MinYw)) * 65535
    DACres = ((residual(1) - MinRes) / (MaxRes - MinRes)) * 65535

    OutPort DAC8, limit(DACx)
    OutPort DAC9, limit(DACy)
    OutPort DAC10, limit(DACz)
    OutPort DAC11, limit(DACr)
    OutPort DAC12, limit(DACp)
    OutPort DAC13, limit(DACyw)
    OutPort DAC15, limit(DACres)
Else
    DACres = ((ErrRMS - MinRes) / (MaxRes - MinRes)) * 65535
    OutPort DAC14, limit(DACres)
End If

End If

```

End Sub

```

'-----
'getFrameCount
' provides watchdog timer with the change in
' frame count since last call
'-----
Public Function getFrameCount() As Long
    getFrameCount = WDcount
    WDcount = 0
End Function

```

```

'-----
'getRMS
' returns current RMS value
'-----
Public Function getRMS() As Single
    getRMS = residual(0)
End Function

```

```

'-----
'angHyst
' performs hysteresis calculation of angle values
'-----
Private Function angHyst(a As Single) As Single
    Dim ah As Single

    'if -170 < a < 170
    'then return a and set previous state
    If a < (180 - hysteresis) And a >= 0 Then
        fromPositive = True
    ElseIf a > -(180 - hysteresis) And a < 0 Then

```

```
    fromPositive = False  
End If
```

```
If fromPositive Then  
    If a > 0 Then  
        ah = a  
    Else  
        ah = 180 + (180 - Abs(a))  
    End If  
Else  
    If a < 0 Then  
        ah = a  
    Else  
        ah = -180 - (180 - Abs(a))  
    End If  
End If
```

```
angHyst = ah
```

```
End Function
```

```
'-----  
'setHysteresis  
' set the size of the angle hysteresis  
'-----  
Public Sub setHysteresis(h As Integer)  
    hysteresis = h  
End Sub
```