

MATLAB Demo for 3DGuidance Tracker

User's Guide

Rev. 5.0 March 21, 2013

1 Introduction

The included script and DLL files take advantage of the MATLAB ability (MATLAB supports dynamic linking) to load and interface with a DLL. This DLL is called “ATC3DG.DLL” (or ATC3DG64.DLL for 64-bit platforms). To understand the details of the low level I/O, one needs to become familiar with the 3DGuidance driveBAY or 3DGuidance trakSTAR *Installation and Operation Guide*. However, familiarity with the low level I/O is not necessary in use of most of the 3DGuidance features. This documentation assumes that the reader is familiar with MATLAB.

2 Requirements and Limitations

- The attached code has been tested on the following platforms:
 - MATLAB version 7.5 (R2007b) running on Windows XP (32 bit)
 - MATLAB version 7.9 (R2009b) running on Windows Vista (32 bit)
 - MATLAB version 8.0 (R2012b) running on Windows 8 (64 bit)
- The following API versions have been tested:
 - ATC3DG.dll version 32.0.7.41 for driveBAY/trakSTAR
 - ATC3DG.dll version 36.0.19.8 for driveBAY2/trakSTAR2
 - ATC3DG64.dll version 36.0.19.8 for driveBAY2/trakSTAR2

- The MATLAB demo is designed to detect the platform and automatically select the 32- or 64-bit versions of the API.
- The included version of ATC3DG.h has minor modifications for MATLAB compatibility.
- Dipole transmitters are supported on all versions. 16 sensors (Multi-Unit Sync) are supported for version 32.0.7.41 on driveBAY/trakSTAR units.
- Nondipole transmitters are supported for driveBAY2/trakSTAR2 units. Autoconfig for more than 4 sensors is not supported for MATLAB.
- Sensor hot-swapping is not monitored in this demo code but the user can implement this behavior via the API.

3 Quick Setup

1. Connect all 3DGuidance tracker cabling as shown in the *Installation and Operation Guide* for the tracker.
2. All the necessary script files are provided in the .zip file provided by Ascension. Extract the files included in this Zip file to a folder on the host machine.
3. Run MATLAB and change the MATLAB current directory to the extracted folder.
4. From the MATLAB command line, type: **run_tracker**. The demo menu appears:

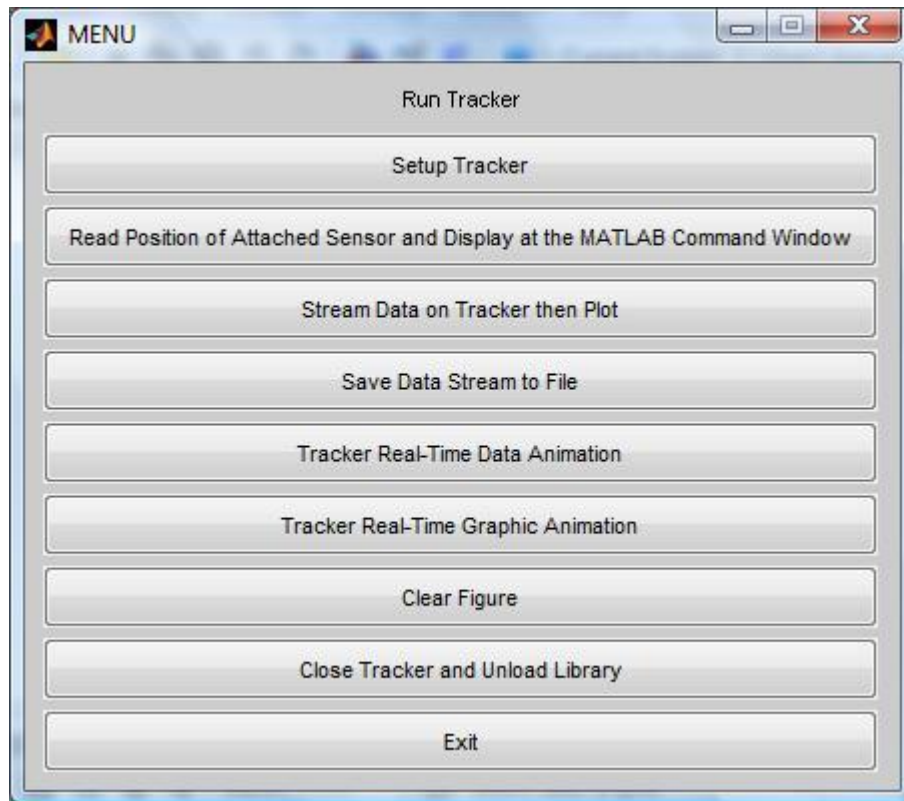


Figure 1: Demo main menu.

5. Select the **Setup Tracker** option on the GUI window to initialize the system. NOTE: Execute this option the first time *Run Tracker* is executed, and any time after *Close Tracker and Unload Library* option is executed. This might take a few minutes.
6. Select any of the following four options from the GUI window to plot and display data:

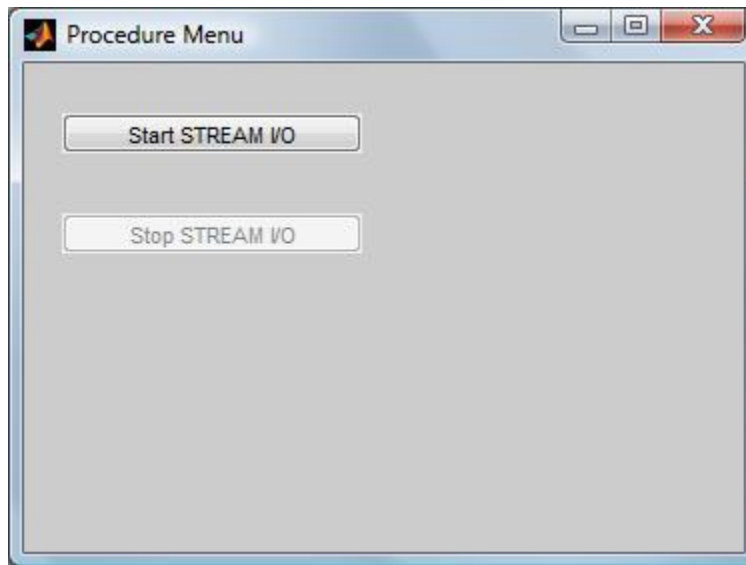
- a. **Read Position of Attached Sensor(s) and Display at the MATLAB Command Window**

Use this to display position, angle and quality data records from the attached sensors in the MATLAB command window. Data is collected from the tracker using the *GetAsynchronousRecord* data collection method.

- b. **Stream Data from the Tracker and then Plot**

This option collects data from the tracker using the *GetSynchronousRecord* data collection method. After selecting this option, a second GUI window will open providing a mechanism for starting and stopping the data collection. Upon STOP,

a figure will be generated showing the data.

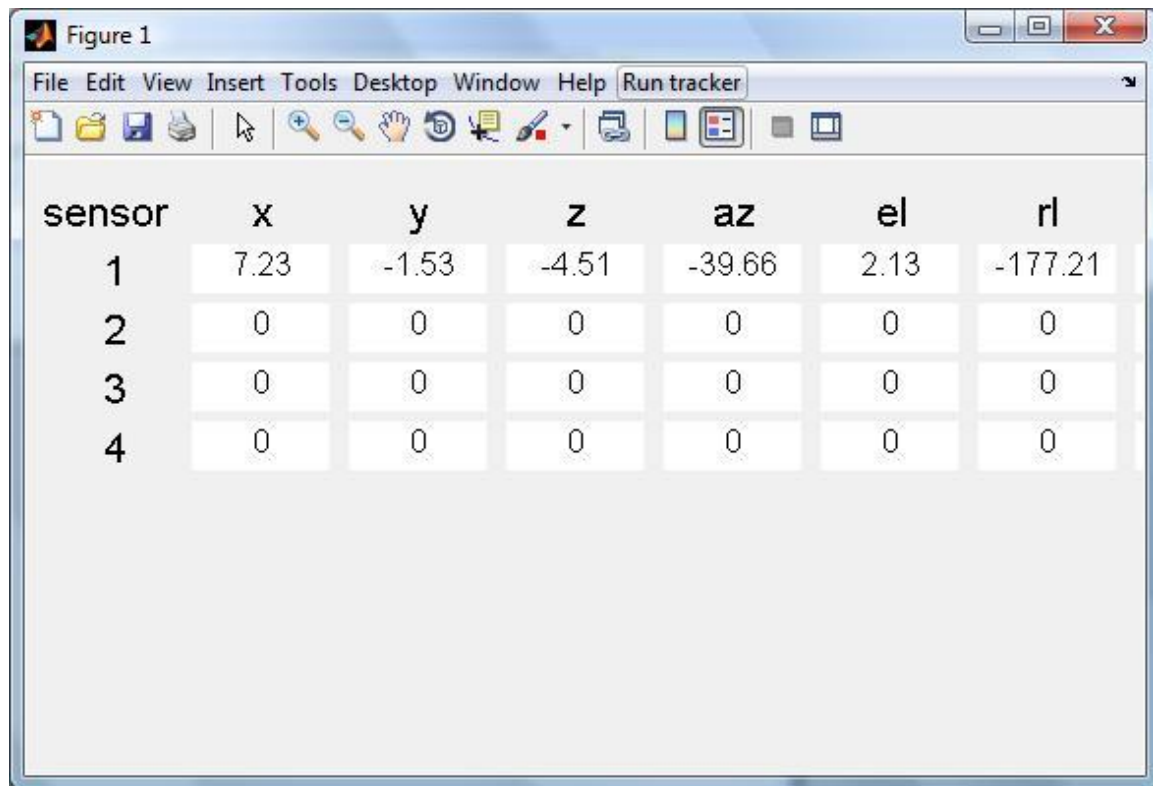


c. Save Stream Data to File

This option is identical to the previous, but provides a means of saving the collected data to file. Files are saved in the MATLAB working directory (scripts directory). A similar GUI window is again used to start and stop the data collection. Upon STOP, the text file can be opened using an editor or word processor.

d. Tracker Real-Time Data Animation

Use this option to produce a real-time textual display of the data as the sensor is moved around in the motion volume. The mechanism for starting and stopping this data collection is via selection under the drop-down menu called **Run Tracker**.



e. Tracker Real-Time Graphic Animation

Use this option to get a graphical representation of the data as the sensor is moved around in the motion volume. The mechanism for starting and stopping this data collection is via selection under the drop-down menu called **Run Tracker**.

7. Click *Close Tracker and Unload Library* to release the tracker interface, for example, to use another application to interface with the tracker.
8. Click *Exit* to close the MATLAB tracker GUI.
9. If a connection error is encountered during any of the operations, try:
 - a. *Close Tracker and Unload Library*
 - b. *Exit*

Then restart by typing **run_tracker** on the MATLAB command line.

4 MATLAB Code

To get started writing your own MATLAB code for the tracker, examine the code in **run_tracker.m**.

4.1 Function Calls

Some of the key function calls are:

- ***tracker_setup***: This sets up the tracker for data acquisition. It loads the library ATC3DG.DLL (or ATC3DG64.DLL), initializes the system and gets the transmitter and the attached sensor configuration.
- ***tracker_position_angles_Quality***: Displays attached sensors positions, angles and quality data at the MATLAB command window using the asynchronous mode of data collection.
- ***tracker_stream_plot***: is designed to place the tracker in so-called 'stream' or synchronous mode. It plots the sensors position after the data is acquired.
- ***save_Stream_Data***: Writes position and angle information from attached sensors to files. The number of files is equal to the number of attached sensors. For example: if sensors 1 and 4 are attached, then the files *Data_Sensor_1.dat* and *Data_Sensor_4.dat* will be created and the data will be saved accordingly.
- ***real_time_Animation_Text***: implements real-time data acquisition and textual display. Data collection is initiated from the drop-down menu **Run tracker**.
- ***real_time_Animation_Graphic***: implements real-time data acquisition and graphical display. Data collection is initiated from the drop-down menu **Run tracker**.
- ***tracker_close***: turns-off the transmitter and unloads the library

4.2 Key Variables

- **Pos**: Position vector of the tracker sensor. Columns are x, y, z values in inches; rows are by tracker number. These variables are computed inside the ***tracker_stream_plot.m*** for stream mode. To access these variables in asynchronous mode use the structure **Record1**

inside the **tracker_position_angles.m** code.

- **Ang:** Angle vector of the tracker sensor. Columns are in degrees; rows are by tracker number. Similar comments for accessing these variables as for **Pos** and **quality**.
- **TimStam:** is the timestamp for the sensor data record, and is returned as a double value. The integer portion of the variable represents the number of elapsed seconds since midnight, Jan 1, 1970, UTC, and is the standard way of representing time and date. This variable is computed in *save_Stream_Data.m* script.

4.3 Key Library Functions

For more information about the following functions, check the *3DGuidance trakSTAR* or *3DGuidance driveBAY Installation and Operation Guide*.

- **GetAsynchronousRecord:** allows an application to acquire a position and orientation data record from an individual sensor or all possible sensors.
- **GetSynchronousRecord:** allows an application to acquire unique position and orientation data records for a given sensor (or from all possible sensors), only as they are computed by the tracker and become available once per data acquisition cycle.

For both methods of continuous data acquisition, a MATLAB GUI is used to start and stop the I/O. MATLAB does not have a function to monitor the keyboard (such as kbhit in C). Accordingly, the command line does not provide a convenient way to stop a loop with a keyboard input. The GUI approach opens a figure with a pushbutton commands. Implementing the `drawnow` MATLAB command within the loop, forces MATLAB to monitor the GUI for mouse and button activity.