

# SQLC – SQL Control for the Orbiter Space Flight Simulator

Pablo Edronkin, 2013 - 2019

<https://orcid.org/0000-0001-8690-7030>

## Abstract

SQLC is a multi functional display (MFD) designed as an interface between the user and the artificial engine of the DGIIAI spacecraft.

DGIIAI artificial intelligence engine is designed around a relational database that uses SQL as means both to control the database as well as to execute the rules of its expert system.

A significant number of those rules are enforced automatically depending on the conditions that the ship encounters during a flight session, but since everything is expressed in terms of SQL snippets, by sending SQL queries to the expert system it is possible to influence its behavior.

That is what SQL was designed for, and it can do so using pure SQL snippets, mode and program activation and deactivation instructions as well as passing parameters directly to the knowledge base.

## Keywords

AI, simulation, artificial intelligence, expert systems, databases

# Acknowledgements

This project could not have taken place without the work done by:

- Dr. Martin Schweiger for developing Orbiter.
- The developers of Sqlite3.
- The Orbiter community of users and developers.

## License and info for contributors

Please read the following files included with this project:

- README.md: contains info on setting up your system and installing the files related to this module.
- CONTRIBUTING.md: if you want to contribute to this project.
- COPYING: for license information.
- Also note that this MFD would work only in conjunction with DGIIIAI or a ship designed around a compatible AI system. It is not intended for use with any other sort of OSFS ship.

# Table of Contents

Abstract.....	1
Keywords.....	1
Acknowledgements.....	2
License and info for contributors.....	2
1. Introduction.....	5
2. Conventions, caveats and base conditions of development.....	6
3. Development.....	7
3.1. Use of SQLC with any RDBMS.....	7
3.2. Use of SQLC with other SQL compatible MFD modules.....	7
3.3. Additional AIE functions for SQLC.....	8
4. Expert system.....	9
5. Use.....	10
5.1. Installation.....	10
5.2. Functions.....	10
5.2.1. [DB].....	10
5.2.2. [CON].....	10
5.2.3. [DIS].....	10
5.2.4. [SQL].....	11
5.2.5. [MON].....	11
5.2.6. [MOF].....	11
5.2.7. [PON].....	11
5.2.8. [POF].....	12
5.2.9. [ITM].....	12
5.2.10. [VAL].....	12
5.2.11. [EXE].....	13
5.3. Procedures.....	13
5.3.1. Connect to a database.....	13
5.3.2. Disconnect.....	14

5.3.3. Send a raw SQL query.....14

6. Sources.....16

7. Alphabetical Index.....17

# 1. Introduction

SQLC is an MFD module made specifically to serve as an interface between the OSFS user and the onboard expert system and artificial intelligence engine of the DGIHAI ship.

It uses a limited number of buttons that allow to connect and disconnect to a database, and operate with the expert system embedded into the code of DGIHAI.

This expert system operates essentially based on facts characterized by a fact name or *Item* field, and a *Value*. Each fact is contained in a record of a database table called *sde\_facts*.

Operations take place of the *Value* field of each record based on data that the Orbiter API sends to the system, MFD or module input, and the set of rules that the expert system has. Those rules are contained per operation in a table called *sde\_rules*.

The artificial intelligence engine or AIE cyclically reads the contents of each table of the knowledge base (a name given to a database that contains expert knowledge of some type, in this case, how to fly a spaceship), the external modules such as this MFD, and applies the appropriate rules, according to the data it has at a given moment.

Then, once the rules are applied and the contents of the Value field of each fact record have been transacted, those values are returned to the OAPI and the ship does hat is being told to do like increase thrust, or angle off attack, change its heading, etc.

This, in essence, is the context in which SQLC operates. For more information on how the AIE operates and the characteristics of the KB of DGIHAI, please read [4.] and [5.] in this document and review the DGIHAI.pdf document that accompanies the installation files for the DGIHAI project.

## **2. Conventions, caveats and base conditions of development**

For the SQLC project, all considerations regarding these points are the same that are applicable for the DGIHAI project.

Terms and abbreviations used in this document are also defined in section [2.] of DGIHAI.pdf. This document is available amongst the installation files for project DGIHAI.

## **3. Development**

SQLC provides a basic interface to DHIHAI's AIE and there is still plenty of room for improvement. At the time of this writing some aspects of this project can still be developed, like:

### **3.1. Use of SQLC with any RDBMS.**

Orbiter ships never made use of database systems. In fact, storage facilities and procedures have always been limited to flat files, which is fine for low - volume reads and writes. But AI and other complex systems might require more advanced data management.

Considering the volume of data that an OSFS session might handle, relational databases could be well -suited for almost any need while providing safe storage and data management.

There are many RDBMS systems, and some of those are free, including Sqlite3, PostgreSQL, MySQL and MariaDB. These all use SQL with varying degrees of compliance with standards, and each one of them has advantages and disadvantages. This topic is fully discussed in DGHAI.pd but suffice to say for now that Sqlite3 was the choice for this project after evaluating the systems mentioned in this paragraph.

This does not mean that those other RDBMS are useless for Orbiter or the DGHAI project; quite on the contrary, it would be interesting to develop a layer of compatibility with those systems and DGHAI + SQLC.

But if you are now thinking about ODBC to achieve such goals, I would recommend against following that route due to the need for speed. It would be much better to develop C or C++ based interfaces.

### **3.2. Use of SQLC with other SQL compatible MFD modules.**

It is perfectly possible to develop new MFD modules to deal with things like cargo, environmental

systems, etc. outside of the DGIIAI AIE realm, and share data thorough the same RDBMS used by SQLC.

This system proves that it is possible to use a relational database in conjunction with OSFS modules; now the sky is the limit as to what can be achieved, and the best part is that using SQL any table of any database used in suh a fashion can be lined to any other using just SQL queries, with no need for other interface methods.

### **3.3. Additional AIE functions for SQLC.**

So far SQLC provides basic access and functionality to interact with DGIIAI. There is nothing that forbids putting more specialized function on the current MFD.



## 4. Expert system

SQLC was designed to work as an onboard interface to DGIIAI's expert system; it does not work as an independent MFD.

It is not the scope of this document to describe the characteristics of the AIE of the intelligent Delta Glider, so for details on that matter you should read the file `~/docs/DGIIAI/DGIIAI.pdf` and it is recommendable that you become at least acquainted with that document before attempting to use this module.

What SQLC allows the user to do is to send commands to the AIE in different ways like raw SQL instructions, specialized calls to `%mode%` facts, program loading and unloading and calls to facts using the fact Item and Value fields.

In this way, it should be regarded as the user interface for the AIE of the DGIIAI project, being essentially an interface to Sqlite3 databases and specifically, those used on DGIIAI.

You can also access the databases used by DGIIAI and SQLC by means of any Sqlite3 compatible database manager or editor, but SQLC provides some convenience functions to do so from within an OSFS session.

## 5. Use

### 5.1. Installation

SQLC installs in a similar way that other similar modules to ~/modules/plugin, but in addition to the file SQLC.dll and its corresponding docs, it also installs a database in ~/databases/SQLC. It also requires that ~/databases/DGIIIAI/DGIIIAI.db is installed.

### 5.2. Functions

#### 5.2.1. [DB]

Database. Lets the user connect to a specific database. With the **[DB]** function the user can switch between DGIIIAI and SQLC databases.

#### 5.2.2. [CON]

Connect. Execute the connection to the database whose name was entered with the **[DB]** function and dialog.

#### 5.2.3. [DIS]

Disconnect. Disconnects the MFD from the currently connected database.

#### 5.2.4. [SQL]

SQL query. Allows the user to enter a full SQL query as a string using a dialog.

#### 5.2.5. [MON]

Mode on. Sets a mode – say, for the sake of an example, *mode\_crs* - specified by the user on, which equates to sending to the AIE the SQL query *UPDATE sde\_facts SET Value = 1 WHERE Item = 'mode\_crs'*

Once you press [MON], a dialog appears and you have to enter the mode name that you want to activate, just by entering its name without the first part comprised of the substring '*mode\_*'. For example, instead of entering '*mode\_crs*' you will have to input '*crs*'.

#### 5.2.6. [MOF]

Mode off. This performs the same query as [MON] except that it sends *Value = 0* instead of *Value = 1*. Once you press [MOF], a dialog appears and you have to enter the mode name that you want to deactivate, just by entering its name without the first part comprised of the substring '*mode\_*'. For example, instead of entering '*mode\_crs*' you will have to input '*crs*'.

Notice that [MOF] and other similar activation or deactivation functions allow you to send a full SQL query just by pressing the corresponding button, writing an abbreviated version of the mode name (with no '*mode\_*' part), saving time and trouble, but you can send the same query using [SQL] if you like.

#### 5.2.7. [PON]

Program on. [PON] and its companion [POF] work in a similar fashion to [MON] and [MOF] but

instead of activating and deactivating modes, they load and unload programs from table *sde\_prg\_rules* to *sde\_rules*.

Normally this process is performed by the AIE itself, but in some circumstances you may want to control that process yourself.

When you press **[MON]** a dialog appears and you have to enter only the numbers that correspond to the program number and version that you want to load, without the string '*prg*'. For example, if you want to load program '*prg43.2*' you only have to input '*43.2*'.

### 5.2.8. [POF]

Program off. Performs the opposite operation to that of **[PON]**.

### 5.2.9. [ITM]

Item. This function opens a dialog that lets you enter the *Item* string that identifies a fact you want to operate with. For example, if you want to set manually a specific target heading for your ship during atmospheric flight, you will press **[ITM]**, enter the string '*tgt\_hdg*' and later use **[VAL]** (see below) to enter the actual heading you want.

Always remember that **[ITM]** and **[VAL]** work together. You need to enter values using both functions in order to alter a fact.

### 5.2.10. [VAL]

Value. After entering the name of a fact that you want to operate with directly using **[ITM]**, you need to give that fact a Value. This is what **[VAL]** is for.

Once the dialog opens, you need to enter a number. Following with the example presented in [5.2.9.], once you entered 'tgt\_hdg', if you want your ship to turn to a 90 deg heading, you should press [VAL] and enter '90'.

### 5.2.11. [EXE]

Execute. This is used to actually send data to the AIE. Whenever you use, for example, [ITM] and [VAL], that fact – changing data is not sent away to the AIE right away but is kept in a buffer.

This gives you time to enter data correctly without doing things in a hurry and confirming your intentions by pressing [EXE]. This sends the data and effectively clears the buffer.

Bear in mind that whenever you press [EXE] you are sending something to the AIE. What that something is depends on what is in the buffer at the time. So don't press [EXE] unless you mean it.

## 5.3. Procedures

These are some procedures that should be performed using the functions described in [5.2.]

### 5.3.1. Connect to a database.

5.3.1.1. Press [DB] and wait for the dialog box to open.

5.3.1.2. On the dialog box write 'DGIIAI' to connect to the DGIIAI.db database or 'SQLC' to connect to SQLC database.

**5.3.1.3.** Press **[CON]** to connect.

After calling this MFD for the first time in your Orbiter session, you should connect to DGIHAI.db with this procedure.

If you want to switch databases, first disconnect from the current database you are connected to following **[5.3.2.]** and then perform **[5.3.1.]**.

## **5.3.2. Disconnect**

**5.3.2.1.** Press **[DIS]**.

**5.3.2.2.** Wait until all pending DB operations are finished. This might take a second or so.

Before leaving a database it is recommendable to disconnect or close it. Otherwise Sqlite3 might leave some connections open and when you finish your OSFS session things might crash or even data might get corrupted.

After disconnection, you will have to perform **[5.3.1.]** to connect to a database again.

## **5.3.3. Send a raw SQL query**

**5.3.3.1.** Press **[SQL]** and wait for the dialog box to open.

**5.3.3.2.** Write a full SQL query on the dialog.

**5.3.3.3.** Press **[EXE]**.

## 6. Sources

6.1. Edronkin, P. (2019). DGIIAI - Intelligent Delta Glider. [online] DGIIAI. Available at: <https://peschoenberg.github.io/DGIIAI/> [Accessed 26 Aug. 2019].

6.2. Sqlite.org. (2000). SQLite Home Page. [online] Available at: <https://www.sqlite.org/index.html> [Accessed 26 Aug. 2019].

6.3. Schweiger, M. (2000). Orbiter 2016 Space Flight Simulator. [online] Orbit.medphys.ucl.ac.uk. Available at: <http://orbit.medphys.ucl.ac.uk/> [Accessed 26 Aug. 2019].



## 7. Alphabetical Index

### A

AI.....	1
AIE.....	3, 5, 7pp., 11pp.
artificial.....	1

### B

base.....	1
-----------	---

### C

CON.....	3, 10, 14
connect.....	3, 5, 10, 13p.
Connect.....	3, 10, 13
contribute.....	2
CONTRIBUTING.....	2
COPYING.....	2

### D

data.....	1
database.....	1
DB.....	3, 10, 13p.
developer.....	2

DGIIAI.....	1p., 5pp., 13p., 16
DIS.....	3, 10, 14
disconnect.....	5, 14
Disconnect.....	3, 10, 14
display.....	1

## E

Edronkin.....	1, 16
es.....	1pp.
EXE.....	3, 13, 15
expert.....	1, 5, 9
Expert.....	3, 9

## F

file.....	2
flight.....	1, 12
Flight.....	1, 16
format.....	2
function.....	1, 3, 8pp.
Function.....	3, 10

## I

input.....	5, 11p.
install.....	2

intelligence.....	1
Item.....	5, 9, 11p.
ITM.....	3, 12p.

## K

KB.....	5
knowledge.....	2

## L

lat.....	1
license.....	2
License.....	2p.

## M

MFD.....	1pp., 5, 7pp., 14
MOF.....	3, 11
MON.....	3, 11p.
multi.....	1

## O

Orbit.....	2
Orbiter.....	1p., 5, 7, 14, 16

orcid.....1

## P

plug.....10

POF.....3, 11p.

PON.....3, 11p.

procedure.....7, 13p.

Procedure.....3, 13

program.....1, 9, 12

Program.....11p.

## R

README.....2

## S

Schweiger.....2, 16

send.....1, 5, 9, 11, 13

Send.....4, 14

setting.....2

Simulator.....1, 16

space.....1, 5

Space.....1, 16

sql.....16

Sql.....2

SQL.....	1, 3p., 7pp., 11, 14p.
SQLC.....	1, 3, 5pp., 13
Sqlite3.....	2
system.....	1pp., 5, 7pp.

## V

VAL.....	3, 12p.
value.....	5, 12
Value.....	5, 9, 11p.