

**VECTO-AAUX Software Developer Guide**

**Content**

[1. Introduction 3](#_Toc426703005)

[2. General Implementation and Approach 3](#_Toc426703006)

[3. Architecture 3](#_Toc426703007)

[4. VECTO UI Intervention 4](#_Toc426703008)

[5. VECTO Code intervention 4](#_Toc426703009)

[6. New file types 5](#_Toc426703010)

[7. Auxiliary Events 5](#_Toc426703011)

[8. Unit Testing 6](#_Toc426703012)

[9. Code Documentation 6](#_Toc426703013)

[10. Schema and Model Versions 6](#_Toc426703014)

[11. VECTO tool Version Update 6](#_Toc426703015)

[12. Further Reading 6](#_Toc426703016)

[Appendix – Advanced Auxiliaries Schema Version 13 7](#_Toc426703017)

# Introduction

VECTO is a tool which has been developed to test the efficiency of heavy commercial Vehicles in terms of Fuel consumption and CO2 Emissions; this was designed by the Technical University of Graz (TUG), Austria.

A requirement was identified for Bus Auxiliaries (an Advanced Auxiliary Model) in order to more accurately, and with greater configurability, model Auxiliary demands along with the vehicle itself. This document is intended to give the prospective developer an overview at high level, and looking at the code should cement that understanding.

A Model was also developed by Ricardo Simulation team. The software was designed to support and integrate the model into VECTO.

# General Implementation and Approach

The VECTO ‘CLASSIC’ (as it was before Advanced Auxiliaries) has been written as a VB.NET Winforms application. The general architecture is not easily augmentable and is sensitive to input variations. For this reason, it was decided to implement the model designed by Ricardo simulation team and ratified by the commission in as non-intrusive a way as possible; this meant almost minimal intervention in existing code.

# Architecture

Figure 1: Schematic of the integration of the Advance Auxiliaries module into VECTO



This system composes of three main parts; VECTO proper can access one of a number of Advanced Auxiliary DLL’s available from building one or more specific advanced utility types. In order to effectively detach and remotely work with the DLL’s the ADVANCED Auxiliary interfaces must be referenced by both VECTO and Advanced Auxiliaries, this way the common interface can be used by both. From a VECTO perspective the object is cast as an Interface and then signals are presented to it and power and fuelling information is used during VECTO’s calculation cycle. That is the extent of any intervention in VECTO.

# VECTO UI Intervention

The only UI Intervention in VECTO is on the VECTO Form itself. In this respect, the user may choose from CLASSIC (Unchanged VECTO) or one of the selected Advanced Auxiliary assemblies which much be present in the executing directory of VECTO in order to be discovered.

Figure : Initial UI intervention of Advanced Auxiliaries into VECTO



You can see from the image above, the dropdown box (CLASSIC or in this example BusAuxiliaries) can be chosen. When selecting Auxiliary Type of an AdvancedAuxiliary type, the Advanced Aux File .AAUX, can be entered in the box, browsed, or a new name can be inserted from here the file can be chosen and configured.

# VECTO Code intervention

VECTO code intervention is minimal. Wherever an intervention has been done, it is commented with ‘**AA-TB**‘, search in files should reveal all the altered code points.

**mAAUX\_GLOBAL** contains global variables which are used to present signals to the AA Model and also other helper functions like discovering resident DLL’s available for AA and several other functions.

**Pre-Calc Loop**

Decides if CLASSIC or AA has been selected and sets up signals for Declaration Mode and WHTC correction for the AA Model.

**Calc Loop**

During the loop sets signals in **mAAUX\_GLOBAL** to be used when the following function is called.

Public Function fPaux(ByVal t As Integer, ByVal nU As Single) As Single

Similar setups exist in a two or three other places. In this function a decision is made to either use the classic formulae to return the Auxiliary power to the crank from the Auxiliaries, or return the value provided by the AA Model.

The main entry point for the advanced auxiliaries is the AdvancedAuxiliaries class. This class represents slide number 17 titled ***Calculations of Cycle FC accounting for Smart Auxiliaries.***  The input signals are received in the Signals class. This class generates an instance of each module numbered 0 to 14, and connects them together via constructors. After executing all connected modules this class generates 2 main outputs which are: TotalFuelGRAMS and TotalFuelLITRES. Apart from these outputs the are other outputs which are used for diagnostic purposes.

# New file types

Several new file types are now defined to support Advanced Auxiliaries; these may need to be added to if further AA Types are added in the future.

Table 1: New file types defined to support the new Advanced Auxiliaries module in VECTO

|  |  |  |
| --- | --- | --- |
| **File EXT NAME** | **Storage Type** | **Description** |
| .AAUX | JSON | Overall configuration information for Electrical, Pneumatic and HVAC. Top of the tree for Advanced Auxiliaries |
| .AALT | CSV | Advanced Combined Alternators. Contains combined map plus source maps. |
| .ACMP | CSV | Advanced Compressor Map. |
| .APAC | CSV | Pneumatic Actuations Map. Stores number of actuations per cycle |
| .AHSM | JSON | Stores Steady State Model results, and also the configuration which resulted in the final result. UI to calculate various heat/cool/ventilate properties resulting in Electrical and Mechanical Power as well as cooling based on environmental conditions. |
| .ABDB | CSV | Bus Database carrying basic values such as Length, Width, Floor Type etc. Used by the HVAC Steady State Model. |
| .AENV | CSV | Stores a number of environmental conditions to be used by HVAC model when in batch-mode. |

# Auxiliary Events

The following events could be subscribed to in VECTO and a delegate signature method would need to be written in order to do something with the data. Currently this is unused, but has been included in case messages need to be passed back.

Care needs to be taken with this because the nature of AAModel means thousands of calculations may be done which could result in event being generated. However, the facility is there should it be needed.

Event AuxiliaryEvent( ByRef sender As Object, byval message As String, ByVal messageType As AdvancedAuxiliaryMessageType )

# Unit Testing

Unit testing is done with NUNIT and MOQ Mocking framework. Although not completely exhaustive, around 340 Unit tests have been written to service approximately seventeen or so classes, fifteen of which comprise the Bus Auxiliaries Mode; this has given a reasonable level of confidence having been modelled in MS Excel before code was written for the tests.

# Code Documentation

XML Comments have been applied mainly to the interface specification which is inherited by those classes implementing the interface. Comments appear where needed in the concrete implementation of the classes.

The Model Schema produced by Ricardo Simulation in most classes over module 5 are annotated with staging calculation markers like S1, S2 (Sum 1, Sum2) etc. so that the schema can easily be tracked in the code base. This is useful for understanding and also for fault finding.

# Schema and Model Versions

The currently implemented schema version for the bus auxiliary module in the code base is version 13 which is implemented based on the schematics in file: BusAux\_Schematic\_V13.pptx; the relevant flow charts are also replicated in the Appendix of this VECTO-AAUX Software Developer Guide document.

The currently implemented Steady State HVAC model is version 7 which is implemented based on the model in file: SSHVAC\_V07\_final.xlsm.

The bus auxiliary module generates a set of default values when created a new configuration. The default values that it generates are based on the default values defined in the updated User Manual.

# VECTO tool Version Update

To update the version number of a VECTO release, navigate to the class VECTO\_Global and modify the VECTOvers constant.

# Further Reading

Further explanation can be found from reading the Model schema (see Appendix) and relating it to the code. This should be self-evident, at least from the Advanced Auxiliaries side of this project.

For the wider VECTO model considerations, explanations are provided in the previous/existing VECTO Developer Guide.

# Appendix – Advanced Auxiliaries Schema Version 13

Figure 3: Advanced Auxiliaries – Implemented Schema for Modules M0: Non-Smart Alternator Set Efficiency

 

Figure 4: Advanced Auxiliaries – Implemented Schema for Module M0.5: Smart Alternator Set Efficiency

 

Figure 5: Advanced Auxiliaries – Implemented Schema for Module M1: Average HVAC Load Demand

 

Figure 6: Advanced Auxiliaries – Implemented Schema for Module M2: Average Electrical Load Demand

 

Figure 7: Advanced Auxiliaries – Implemented Schema for Module M3: Average Pneumatic Load Demand

 

Figure 8: Advanced Auxiliaries – Implemented Schema for Module M4: Air Compressor

 

Figure 9: Advanced Auxiliaries – Implemented Schema for Module M5: Smart Alternator Set Generation

 

Figure 10: Advanced Auxiliaries – Implemented Schema for Module M6: OVER-RUN smart and non-smart systems alternator and air compressor load calculations





Figure 11: Advanced Auxiliaries – Implemented Schema for Module M7: FULL CYCLE definition of alternator and air compressor loads for smart systems





Figure 12: Advanced Auxiliaries – Implemented Schema for Module M8: FULL assignment of auxiliary loads



Figure 13: Advanced Auxiliaries – Implemented Schema for Pneumatic System, Module M9: Air delivery and FC calculations for smart and non-smart systems over the cycle





Figure 14: Advanced Auxiliaries – Implemented Schema for Pneumatic System, Module M10: Actual FC calculation for true air delivery for smart and non-smart systems over the cycle





Figure 15: Advanced Auxiliaries – Implemented Schema for Electrical System, Module M11: Electrical energy and FC calculations for smart and non-smart systems over the cycle





Figure 16: Advanced Auxiliaries – Implemented Schema for Electrical System, Module M12: Actual FC calculation for true electrical demand for smart & non-smart systems over the cycle





Figure 17: Advanced Auxiliaries – Implemented Schema for Module M13: WHTC Corrected FC calculation over the cycle





Figure 18: Advanced Auxiliaries – Implemented Schema for Module M14: HVAC Fuelling Accounting





Figure : Advanced Auxiliaries – Implemented Schema for Calculation of auxiliary power in Pre-run





Figure 20: Advanced Auxiliaries – Calculation of Cycle FC accounting for Smart Auxiliaries





Figure 21: Advanced Auxiliaries – Implemented Schema for Combined Alternators Module





Figure 22: Advanced Auxiliaries – Implemented Schema for HVAC Module



Table 2: Tabulated List of Reported Logged Signals from Schema

|  |  |  |  |
| --- | --- | --- | --- |
| **Module Origin** | **Model Signal Names In Schematics** | **Code Level Signal Names** | **Public Signals as Reported** |
| M0 | Alternators Efficiency | AlternatorsEfficiency | AA\_NonSmartAlternatorsEfficiency |
| M0.5 | Smart Idle Current (A) | SmartIdleCurrent | AA\_SmartIdleCurrent\_Amps |
| M0.5 | Alternators Efficiency (Idle – Result Card) | AlternatorsEfficiencyIdleResultsCard | AA\_SmartIdleAlternatorsEfficiency |
| M0.5 | Smart Traction Current (A) | SmartTractionCurrent | AA\_SmartTractionCurrent\_Amps |
| M0.5 | Alternators Efficiency (Traction – Result Card) | AlternatorsEfficiencyTractionONResultsCard | AA\_SmartTractionAlternatorEfficiency |
| M0.5 | Smart Overrun Current (A) | SmartOverrunCurrent | AA\_SmartOverrunCurrent\_Amps |
| M0.5 | Alternators Efficiency (Overrun – Result Card) | AlternatorsEfficiencyOverrunResultsCard | AA\_SmartOverrunAlternatorEfficiency |
| M4 | Compressor Flow Rate (l/s) | GetFlowRate | AA\_CompressorFlowRate\_LitrePerSec |
| M6 | Over-run Flag | OverrunFlag | AA\_OverrunFlag |
| M7 | Engine Idle | INTL\_EngineIdleFlag | AA\_EngineIdleFlag |
| M8 | Compressor Flag | CompressorFlag | AA\_CompressorFlag |
| M13 | Total cycle FC (Grams) | TotalCycleFuelConsumptionGrams | AA\_TotalCycleFC\_Grams |
| M13 | Total cycle FC ( LITRES ) | TotalCycleFuelConsumptionLitres | AA\_TotalCycleFC\_Litres |