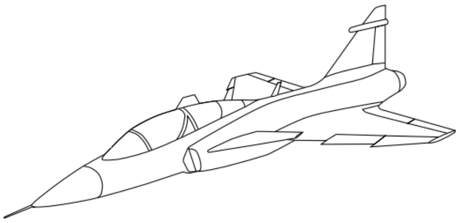
User Manual
Future Aircraft Energy Management Systems

Version 0.1

Author: ISY Group
Date: December 7, 2021



Status

Reviewed	ReviewerName	Date1
Approved	ApproverName	Date2

Course name:	Reglerteknisk projektkurs	E-mail:	emibo804@student.liu.se
Project group:	ISY group	Document responsible:	ISY Group
Course code:	TSRT10	Author's E-mail:	
Project:	Future Aircraft Energy Management Systems	Document name:	usermanual.pdf

Project Identity

Group E-mail:	emibo804@student.liu.se
Homepage:	https://gitlab.liu.se/chrre97/student-project-energy-management
Orderer:	Alessandro Dell'Amico, Saab Aeronautics **Phone:** +46 ordernr, **E-mail:** alessandro.dellamico@liu.se
Customer:	Alessandro Dell'Amico, Linköping University, IEI. **Phone:** +46 customernr , **E-mail:** alessandro.dellamico@liu.se
Course Responsible:	Anders Hansson, Linköping University **Phone:** +46 13 281681, **E-mail:** hansson@isy.liu.se
Project Manager:	Emil Boström
Supervisor:	Kristoffer Ekberg, Linköping University, ISY. **Phone:** +46 advisornr , **E-mail:** kristoffer.ekberg@liu.se

Group Members

Name	Responsibility	E-mail (@student.liu.se)
Emil Boström	Project Leader	emibo804
Petrus Eriksson	Hardware Manager	peter792
Hugo Lundeberg	Document Manager	huglu683
Erik Börjesson	Test Leader	eribo610
Robin Helsing	Design Manager	robhe093
Emil Brunberg	Software Manager	emibr702

Document History

Version	Date	Changes made	Sign	Reviewer
0.1	date1	First draft.	Sign1	Name1
0.2	date2	First revision	Sign2	Name2

Contents

1 Introduction

This document is the user manual for the content created in the project *Future Aircraft Energy Management System* in the course *TSRT10, Reglerteknisk projektkurs, CDIO* given at Linköping University. The user manual contains information and instructions regarding how to run the complete system for future work. The document will consist of a short description of the system. This is followed by instructions on how to run and operate the system.

2 System description

The system consists of six parts which all can be find in the simulink file: The D.T of the producer, D.T of the consumer, communication with keysight, Pilot model, JSBSim and a block which communicates with Iron Bird.

1. **D.T Producer -** This block contains the D.T of the producing system. It consists of models for Main engine, Constant speed drive, Generator, Inverter, Converter, SSPC and battery.
2. **D.T Consumer -** This block contains the D.T of the consuming system. It consists of model for EMA, SHA, Pump and Motor.
3. **Keysight Communication -** This block consists of different TCP clients enabling communication with the Keysight boxes from simulink.
4. **JSBSim -** This block contains the flight simulator used to simulate different parameters for the aircraft.
5. **Pilot -** This block is used to simulate a flight mission for the JSBSim.
6. **Iron Bird -** This block enables communication between simulink and the Iron Bird.

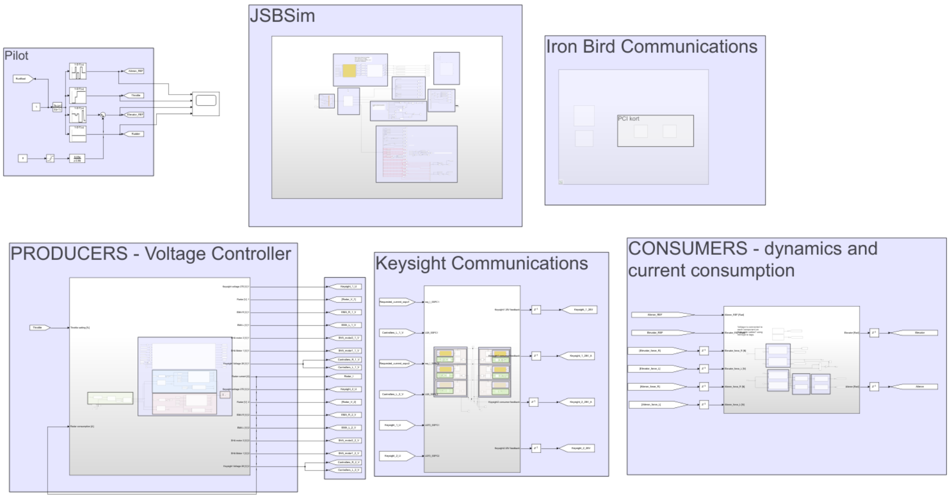

JSBSim
Pilot
Iron Bird Communications
PCI kort
PRODUCERS - Voltage Controller
Keysight Communications
CONSUMERS - dynamics and current consumption

Figure 1: Complete system in Simulink

3 Installation Guide

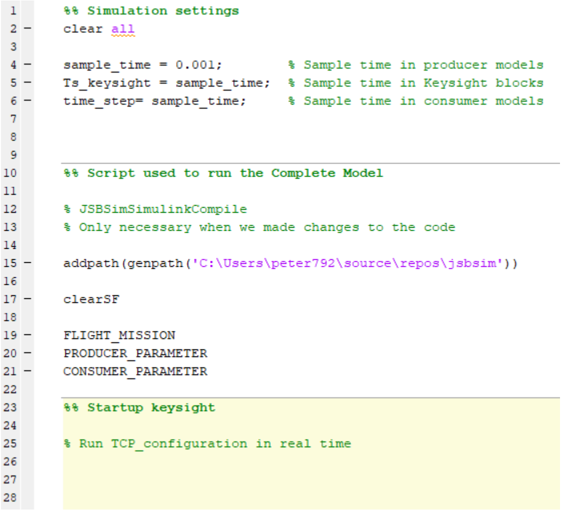
There are some software that needs to be installed in order to be able to run a simulation. First, a standard installation of MATLAB (R2020b) and Simulink is needed. In addition to this there are some software which needs installation.

1. Simulink real-time library
2. JSBSim

For the JSBSim installation, an existing guide already explains the installation process in detail. This guide can be found in the git repository of the project.

4 User Guide

Below follows a guide for how to start and operate the system.
First and foremost, the matlab file *Run_script* is the main script which will be run before each new simulation, see figure 3. The *Run_script* in turn defines the sample time for simulink and executes six different commands that is needed for the simulink model to run. The script can be seen in figure 3.



```
%% Simulation settings
clear all

sample_time = 0.001;        % Sample time in producer models
Ts_keysight = sample_time;  % Sample time in Keysight blocks
time_step= sample_time;     % Sample time in consumer models



%% Script used to run the Complete Model

% JSBSimSimulinkCompile
% Only necessary when we made changes to the code

addpath(genpath('C:\Users\peter792\source\repos\jsbsim'))

clearSF

FLIGHT_MISSION
PRODUCER_PARAMETER
CONSUMER_PARAMETER

%% Startup keysight

% Run TCP_configuration in real time
```

Figure 2: Run_script file in matlab

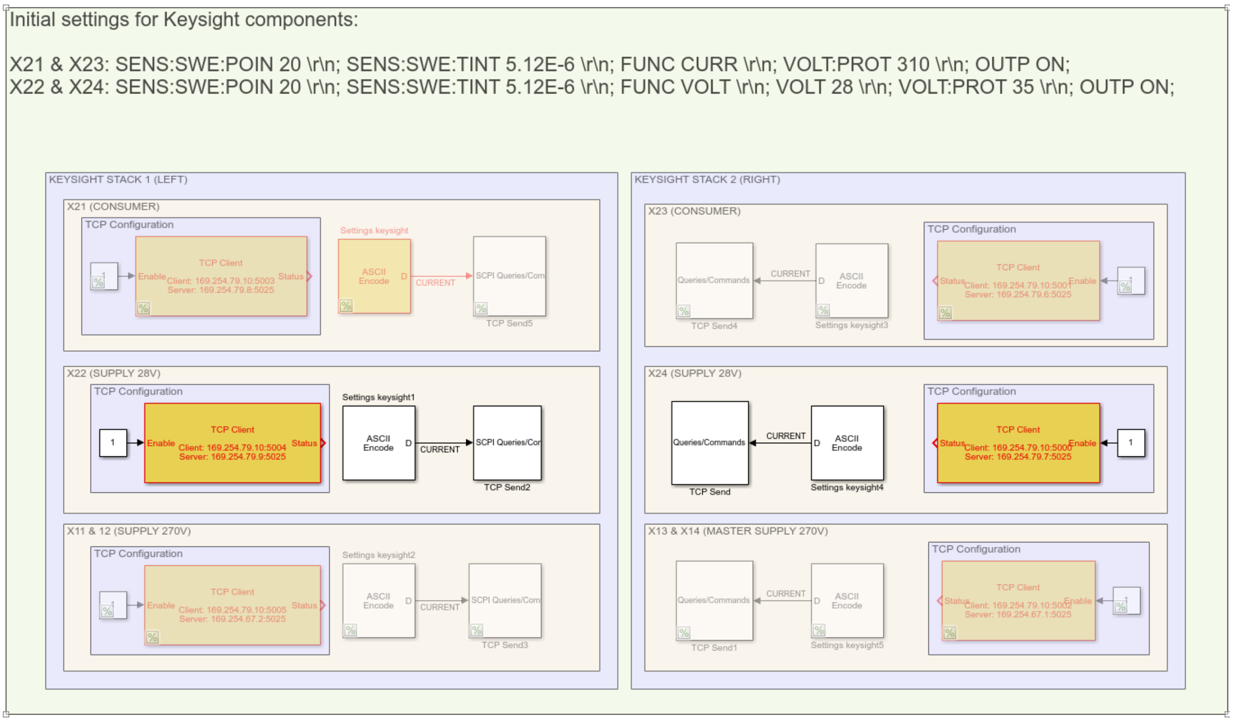
Starting from the top of the code in *Run_script* the different commands will be explained.

1. **Simulation settings -** In the simulation settings section the sample times used in the simulation are set.
2. **Script used to run the Complete Model -** This sections consists of six different executions which has to be ran every time a new simulation is done.
   - **JSBSimSimulinkCompile -** This line compiles the JSBSim C++ code into runnable matlab code. Note that this line only has to be ran one time since it creates a mex file which is placed in the right folder. However if there are any changes made to the JSBSim C++ code the *JSBSimSimulinkCompile* must be ran to create a new .mex file.

- **addpath** - This line makes sure that the correct files are in the path.
- **clearSF** - Resets all the parameters used in the JSBSim.This is an important command which has to be executed every time before a simulation. The clearSF command resets and sets initial condition used by the flight simulator JSBSim.
- **FLIGHT_MISSION** - Creates vectors that define the flight mission used in JSBSim.
- **PRODUCER_PARAMETER** - Defines each variable needed for the producers D.T.
- **CONSUMER_PARAMETER** - Defines each variable needed for the consumers D.T.

3. **Startup keysight** - Sets the parameters used for the keysight hardware when starting the simulations.

The section **Startup Keysight** in the matlab script will be explained more comprehensively since it is vital to the keysight connection.



Initial settings for Keysight components:

X21 & X23: SENS:SWE:POIN 20 \r\n; SENS:SWE:TINT 5.12E-6 \r\n; FUNC CURR \r\n; VOLT:PROT 310 \r\n; OUTP ON;
X22 & X24: SENS:SWE:POIN 20 \r\n; SENS:SWE:TINT 5.12E-6 \r\n; FUNC VOLT \r\n; VOLT 28 \r\n; VOLT:PROT 35 \r\n; OUTP ON;

Figure 3: Run_script file in matlab

The section **Startup Keysight** refers to a simulink file *TCP_configuration*, which is a real-time simulink file. This real-time file has to be executed every time a new simulation is done. The *TCP_configuration* resets parameters from previous simulation and also sets initial condition for the next simulation.
TCP_configuration consists of six different parts, one for each of the different keysight boxes. Each part has a TCP client block, ASCII encode block and a TCP send block.

1. **TCP client block** - This block is used to establish connection between the speedgoat and a specific keysight. It takes in IP-address and port number for both the

speedgoat and the keysight hardware. The block can then be enabled by sending a constant 1 into it.

2. **ASCII-encode block -** The ASCII-encode block is used to encode the message into ASCII code. This is done since the Keysight is able to be programmed by using ASCII code.
   The following line of ASCII-code is an example of how the input to one of these ASCII-encode blocks can look:

   ```
   SENS:SWE:POIN 20 \r\n; SENS:SWE:TINT 5.12E-6 \r\n; FUNC VOLT \r\n;
   VOLT 28 \r\n; VOLT:PROT 35 \r\n; OUTP ON;
   ```

   In short terms what this line of code does is that it sets the measurement setting for the keysight. The measurement interval is set to 5.12μS and it takes the average of 20 points. VOLT 28 sets the initial voltage of the system and VOLT:PROT sets a voltage limit for the Keysight. FUNC VOLT sets the keysight to a constant voltage mode and OUTP ON enables this setting.
   An exact explanation of the ASCII-code can be found in the Keysight manual which is located in the project git repository .

3. **TCP send block -** This block is used for sending the information, which has previously been encoded from the ASCII encode block.

Number 2 in the list above is where the important settings are set and will therefore be further explained in order to simplify the understanding.

5 Running a simulation

This section will step by step go through how to run a simulation. This guide is based on that previous steps of installation such as JSBSim and complementary simulink libaries are installed.

5.1 Step 1 - Power on

Start by switching on the fuses in the fuse box. See figure 4. Note that the red marked fuses are the ones which has been used through out this project. The green marked fuses are connected to the four keysight boxes in the bottom of the rack. These four boxes have not been connected when the work has been performed. However, in the future these fuses will also be relevant.

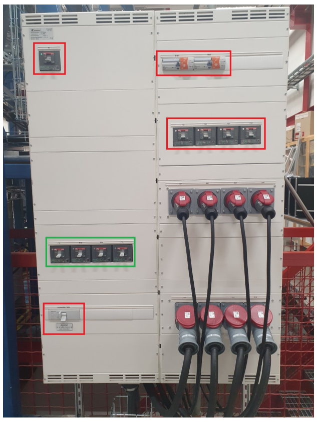


Figure 4: Fuse box for the Keysight hardware.

To be able to run the system in real-time the speedgoat must be switched on. See figure 5 where the blue button is the power button.

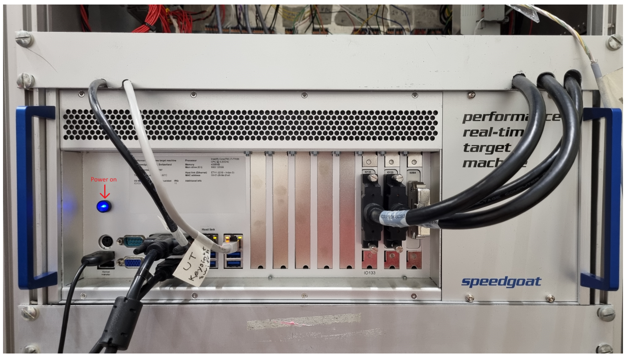


Figure 5: The speedgoat computer.

5.2 Step 2 - Run MATlAB/Simulink

Open MATLAB R2020b and make sure that you are in the right path for JSBSim. Open and run the script *Run_script*, this will set all necessary parameters used in the simulink files. In order to set the initial parameters on the keysight components open the TCP_configuration, connect the real-time computer and run the model. In order to see if the speedgoat is connected see figure 6.

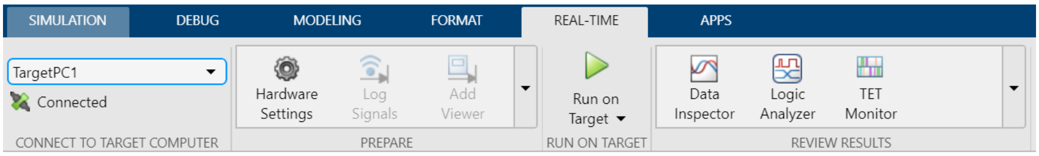


Figure 6: Connect real-time computer.

When the *TCP_configuration* file has been ran and successfully completed you should now be able to run the main simulink file, *jsb_sim*.. see figure 7.

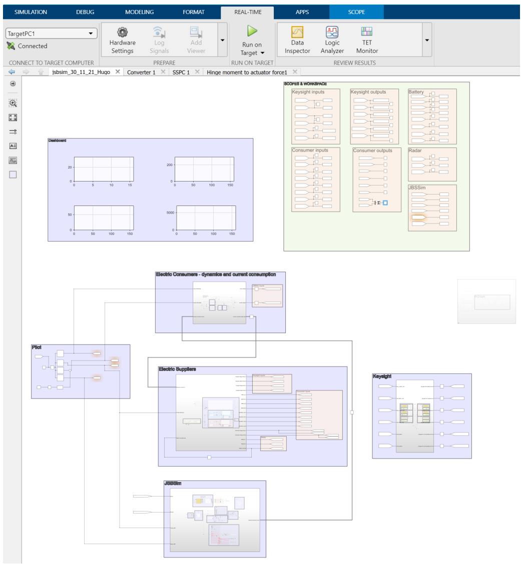


Figure 7: The main simulink model.

6 NOTES

Below are some notes that might be of interest for the user.

- To change the length of the flight mission, change the variable t in FLIGHT_MISSION and make sure it correlates with the time in jsb_sim.
- When running JSBSim in real time each look up table used in the simulink file will need its own time vector. Therefore right now there are six time vectors in the flight mission. However, for future development this is worth mentioning since using the same vector leads to problem executing the file.