**Quick Users Guide for Coolsim,**

**NREL’s Automotive A/C System Model V84**

**Quasi-Transient Beta Version**

**September 30, 2015**

**Introduction**

CoolSim software is provided free of charge. NREL does not provide user support for CoolSim. This quick guide is intended to provide the minimum instructions needed to run the software and report results. The software includes a model of a generalized HVAC system called CoolSim\_v84 built in the MATLAB/Simulink® environment. The model includes C-written Simulink S-functions distributed in binaries. Working knowledge of Matlab/Simulink is assumed. The model has been tested on 32-bit and 64-bit Windows 7 platforms with MATLAB/Simulink release 2014 and Simulink V8.4.

To enable faster executions the model needs to be compiled with an accelerator. That requires a C++ compiler to be installed on the user system. For 32-bit systems, the compiler that is included with the MATLAB installation is appropriate. For 64-bit systems, a MATLAB compatible 64-bit C++ compiler must be installed. It is advised to read the MathWorks support page at the following link before installing a 64-bit C++ compiler. <http://www.mathworks.com/support/compilers/R2014a/index.html>

If a compiler cannot be installed, the accelerator option should be disabled in the Simulink environment for the model to execute correctly without an accelerator. To disable or enable the accelerator the user can follow a procedure described at the following link:

<http://www.mathworks.com/company/newsletters/articles/improving-simulation-performance-in-simulink.html>

**A script to run the model is provided with the distribution**. The MATLAB script, called ‘run\_mc.m’ runs a mechanical compressor version of the model for 10 seconds of simulated time (specified by trun=10.0) and plots the results. The value for the ‘trun’ parameter can be increased for longer simulations. Similarly, a MATLAB script called run\_ec.m runs the model with an electrically driven compressor.

**Description of the Simulation Model**

The provided model is intended to simulate transient processes typical for light-duty automotive air conditioning systems. The model consists of three main sub-models: cooling circuit, compressor, and cabin. Electronic controls are also implemented. The model can be configured for two different configurations by executing one of the two available versions of the initialization file. These two model configurations include a mechanical drive compressor and an electric drive compressor with their associated electronic controls. The CoolSim\_84 model is of a ‘Quasi-Transient’ type. A brief overview of the modeling method with references is given below.

**Cooling Circuit Sub-model**

The cooling circuit sub-model uses refrigerant zero-dimensional control volume simulation blocks (0-D volume blocks) and one-dimensional refrigerant pipe simulation blocks (1-D pipe blocks). Equations for conservation of mass and energy are implemented in these simulation blocks. All refrigerant thermodynamic and material properties are obtained from two-dimensional lookup tables based on specific enthalpy and pressure. The files related to refrigerant property calculations are located in the ‘R134a\_files’ subfolder. The heat exchangers are modeled with the 1-D pipe simulation blocks. For the wall-to-refrigerant heat transfer, the Dittus-Boelter correlation [1] is used in the condenser, and the Dittus-Boelter and the Chen correlations [1-2] are used in the evaporator. For the air-to-wall heat transfer, the Chang correlation [3] is used in both the condenser and evaporator. Fin efficiency calculations are incorporated. The effects of air humidity and condensation of water from air on the evaporator surfaces are also modeled. Thermal masses of heat exchanger walls are accounted for; therefore, the hysteresis that they cause is modeled. The axial heat transfer (heat transfer in the direction of refrigerant flow) in the walls is also taken into account. Temperature drop across the wall material from the refrigerant to the air-side, however, is assumed negligible. A single 1-D pipe simulation block can be used for modeling multiple parallel channels of refrigerant to represent flat tubes typically used in automotive compact condensers. More details on the simulation method for the refrigerant circuit can be found in [4].

The expansion device implemented in the model is an externally balanced thermostatic expansion valve. Delayed response of the bulb temperature to changes in the evaporator exit temperature is achieved with a first order delay simulation block. The bulb pressure is the saturated refrigerant pressure at the bulb temperature. This response is the only dynamically modeled detail of the expansion device. Otherwise, the position of the valve ball is determined from a static force balance. Once the flow area through the valve is known, the refrigerant flow rate is calculated using the two-phase orifice flow equations.

**The Compressor Sub-model**

The model includes options for simulating mechanically and electrically driven compressors. The choice of the compressor is determined by the input parameter called ‘ifmech’. A setting of ‘ifmech = 1.0’ enables the mechanical compressor and its controls while a setting of ‘ifmech = 0.0’ enables the electric drive compressor.

The compressors are modeled as constant volume variable speed displacement devices. The rotational speed and the displacement per revolution (an input parameter) determine the ideal forwarded volume per second. The actual forwarded volume per second is obtained by taking into account the compressor volumetric efficiency. Compressor upstream conditions and downstream pressure are obtained from the upstream and downstream simulation blocks. The downstream enthalpy is calculated using the isentropic efficiency. Both the volumetric and the isentropic efficiencies are functions of the compressor rpm and the downstream-to-upstream pressure ratio. These efficiency tables are a part of the model input data and are different for the mechanical and electric drive compressor versions. For the mechanical compressor, efficiency tables typical for piston compressor types are used, while for electric drive compressors, efficiency tables typical of scroll compressor types are used. The model controls include logic for compressor cycling. Such controls are different for mechanical and electric drive compressors. The provided model includes examples of control implementation. Details of the compressor model can be found in [5].

**The Cabin Sub-model**

The provided cabin sub-model is a zero-dimensional lump-sum air/water vapor mix volume model that tracks pressure, temperature and humidity. The shell and interior thermal masses as well as heat transfer between the thermal masses and the ambient are accounted for. Solar energy is absorbed by the cabin shell and it’s interior. Although simplified, the cabin model provides relevant and reasonable boundary conditions for the cooling circuit sub-model. Details of the cabin modeling method can be found in [5].

**Controls**

Basic electronic controls are implemented for both mechanical and electric drive compressor models. These common controls include cycling-off the compressor due to downstream pressure exceeding a high limit, or upstream pressure exceeding a low limit. Limiting the compressor off period duration after each of shut-downs is also supported by the model.

For a mechanical drive compressor configuration, the compressor is cycled on and off depending on whether the cabin temperature is below or above the target temperature. An additional control cycles off the compressor if evaporator exit air temperature is below a preset target value (typically 2-3 degrees Celsius) to prevent freezing of any water condensed on the surface of the evaporator. A temperature dead-band is also implemented for both cabin and evaporator-out air temperatures to reduce the frequency at which the cycling occurs. The evaporator blower speed is set to one of three available settings at the beginning of a simulation and the setting does not change throughout the simulation. The user may add a control that adjusts the blower flow rate based, for example to minimize its consumed power.

For an electric drive compressor configuration, the compressor speed is independent from engine speed and the engine rpm input parameter is ignored. The compressor speed is adjusted based on whether the evaporator wall temperature is above or below a target temperature in a refrigerant path just upstream of where superheat is expected to occur. With this kind of compressor speed control, compressor cycling is only needed to prevent freezing or to prevent the compressor from operating at inefficiently low rpms. When the compressor restarts, it does so at a preset minimum speed. The evaporator blower speed is automatically adjusted based on the cabin temperature relative to the preset cabin target temperature for flow rate modulation.

The above controls can be easily modified and additional control algorithms can be added by the user familiar with the block simulation environment of Simulink.

**Model Preparation and Execution**

There are 20 Simulink binary (mex) files contained in the distribution folder, 10 for a 32-bit system and 10 for a 64-bit system. For a 32-bit system the files are: airmix\_volume04.mexw32, airmixout01.mexw32, evap\_airout01.mexw32, lookup\_s\_vs\_urho.mexw32, orifice13.mexw32, R134a\_line77.mexw32, R134a\_line78c.mexw32, R134a\_volume11.mexw32, tempr\_from\_phw.mexw32 and reverse\_signal01.mexw32. For a 64-bit system the file names are the same, except the file extensions that are mexw64.

The basic pre-processing, running, and post-processing of a simulation is the same regardless of mechanical or electric drive compressor selection. In this section, the entire process is shown for an example of the mechanical drive compressor setup; however, all the steps shown in this section are valid for an electric drive compressor setup as well, with only the ‘\_mc’ qualifiers in the file names to be replaced with the ‘\_ec’ qualifiers.

**The simplest way to run the model is to execute the ‘run\_mc.m’ MATLAB script**. This will initialize the model, run it for 10 seconds of simulated time, and plot results. The ‘run\_mc.m’ script first executes the ‘point\_mc01.m’ file. Next, ‘point\_mc01.m’ sets up the input parameters and performs initialization. The model is then run by issuing a ‘sim model\_mc’ command in the Matlab command window. Finally, the results are plotted with the included three plotting scripts. The ‘plots\_basic\_mc.m’ script is used to plot results that are relevant to the cooling circuit model, the ‘plots\_cabin\_mc.m’ script is used to plot results that are relevant to the cabin sub-model, and the ‘plots\_cycle\_mc.m’ script shows the thermodynamic cycle in the pressure-enthalpy space as of the end of the simulation run.

The ‘run\_mc.m’ script calls an operating point definition script that is called ‘point\_mc01.m’. The ‘point\_mc01.m’ script first calls a generic initialization file, called ‘CoolSim\_v84\_mc\_init.m’. This file sets default values for all of the input parameters and boundary conditions of a mechanical drive compressor and its controls. The user can override these defaults by issuing new assignments at the end of the ‘point\_mc01.m’ file. It is recommended that for different cycles or operating conditions, new files named ‘point\_mc02.m’, ‘point\_mc03.m’… etc. are created and used in place of the original ‘point\_mc01.m’ file. The ‘CoolSim\_v84\_mc\_init.m’ or ‘CoolSim\_v84\_ec\_init.m’ file should not be modified by the user since they guarantee that all the needed parameters have default values. All model input parameters that typically need to be changed to define a driving cycle or operating condition are available in the ‘point\_mc01.m’ file and some of them override the defaults specified in the CoolSim\_v84\_ec\_init.m’ or ‘CoolSim\_v84\_mc\_init.m’ files .

An additional script called ‘run\_mc\_restart.m’ is included with the distribution to demonstrate how simulations can be restarted if they are too long to be executed in a single run or if a simulation needs to be extended at a later time. If the ‘reinit.m’ script is executed after a completion of a simulation, a re-initialization of all simulation state variables is done based on the values stored in a binary file called ‘newinitstates.mat’ that is created after each successful run. These provisions in the code guarantee that a re-started simulation will proceed from the point where a previous run finished. The user can also manually load the ‘newinitstates.mat’ file into the Matlab workspace and initiate a continued simulation. The ‘newinitstates.mat’ files can also be used for initialization of entirely new simulations (as initial “guesses”). In such cases a ‘newinitstates.mat’ file should be loaded into the Matlab workspace and re-saved with a setting of “time0” = 0.0 to prevent the new simulation from starting from a nonzero time that may affect lookup values. As a final note, the re-initialization is achieved with the data saved in Simulink ‘scope’ blocks. These blocks by default save the last 100 time step values of the corresponding state variables. The default is set by the ‘nsave’ input parameter. Note that with typical time steps on the order of 0.0002 and simulation times of tens of minutes setting nsave to a low value will result in saving data for a large number of time steps and may saturate the computer memory.

Data of the entire simulation is saved in a series of ‘To Workspace’ blocks. Such data is sampled at a low rate to reduce memory demand. The frequency of data sampling is set by the ‘out\_decim’ input parameter, which has a default value of 50. This default setting means that data of every 50th time step is saved. As an example, with a typical computational time step of 2x10-4 seconds and a decimation of 50, this rate of the data recording will make 100 samples per second.

To save simulation results for plotting or a long simulation executed in several steps, a script called ‘record\_segrun.m’ is provided with the distribution. When executed, this script will write all the variables currently available on the MATLAB workspace (the ones saved by the ‘To Workspace’ Simulink blocks) into a file called ‘segmentedrun.mat’. These files can be renamed according to a specific simulation need (for example ‘segmentedrun\_mc01.mat’ for ‘point\_mc01.m’), and can later be used for plotting without having to rerun the simulation (although a ‘point\_mc01.m’ script still has to be executed before loading the ‘segmentedrun\_mc01.mat’ file for plotting purposes). If a continuation run is completed while the ‘segmentedrun.mat’ file of the original is present in the simulation folder, the ‘record\_segrun.m’ file will append the results of the continuation run to the results of the original run. This concatenates the data and makes it available for the entire length of the original and continuation runs, both on the MATLAB workspace as well as in an updated ‘segmentedrun.mat’ file. Use-cases of these re-initialization /restart methods are demonstrated in file called ‘run\_mc\_restart.m’

Note that to use the ‘plots\_cycle.m’ plotting file, a simulation must first be completed and the variables that are saved in the ‘scope’ blocks must be in the workspace. The information needed for a p-h cycle plotting is not included in the ‘segmentedrun.mat’ file, so that only loading the ‘segmentedrun.mat’ file and executing the ‘plots\_cycle.m’ script afterwards will result in an error.

The last line in the ‘CoolSim\_v84\_mc\_init.m’ file initializes the simulation state variables across the entire system by loading the ‘initstates.mat’ file. This enables initialization of refrigerant variables in accordance to typical system conditions. The user should not adjust the refrigerant initialization by assigning new values to input parameters affecting initial refrigerant conditions. Only the original ‘initstates.mat’ initialization file or a newly generated ‘newinitstates.mat’ initialization file should be used for new refrigerant circuit initializations. Other simulation state variables included in the ‘initstates.mat’ file can be overwritten in the operating point files (such as ‘point\_mc01.m’ file).

The Simulink simulation time step (an input parameter ‘deltat’ specified in the CoolSim\_v84\_ec\_init.m file) should be set to 2x10-4 seconds, and the integration method should be the fixed-step first-order solver Euler method (these are set as default). Higher time steps may lead to inaccuracies, excessive oscillations, or crashes. Using smaller timesteps will extend the execution time. The model will run, with the same timestep, using the Runge-Kutta 4th order integration method instead of the Euler method, however, it will be at least three times slower with nearly identical results. Variable time steps (ODE23) solvers may also be used, but have been not shown to produce quicker or more accurate results then the Euler method. The model runs at approximately real-time speed on a Windows 7 64-bit computer with a Core™ i7-4800MQ 2.7 GHz processor and 8 GB of memory.

**List of Key Input Parameters:**

There are a large number of input parameters that are needed for a simulation. All of these parameters are provided in the ‘CoolSim\_v84\_mc\_init.m’ file for a mechanical compressor configuration and in ‘CoolSim\_v84\_ec\_init.m’ file for an electrical compressor configuration. Only a small subset of these parameters, however, should be modified by a user with limited experience. These input parameters aid in specifying the system conditions, but they do not make changes to the system definition itself. These adjustable parameters are included in the ‘point\_mc01.m’ file for the mechanical compressor configuration, and in ‘point\_ec01.m’ file for the electrical compressor configuration. These adjustable parameters are also listed below. The user should not typically modify the input parameters related to the refrigerant circuit, including the header volumes, lengths and diameters of the pipes, and the number of segments of the pipes. If any of these values are modified, the original initialization or any new initialization created by the user with the ‘reinit.m’ will not work. If the Simulink model is modified by adding passes in the evaporator or condenser, then the input parameters for the refrigerant circuit will change and thus the initialization file should also be updated. Only users with advanced knowledge of the code should consider this type of modification.

The mechanical compressor setup uses vehicle and engine speeds typical for a mid-size sedan traversing the SC03 drive cycle. For electrical compressor setup, the model only uses vehicle speed since the engine speed is not needed. The engine speed for the SC03 cycle is specified in a file called “rpmvstimeSCO3.mat” followed by a pulley ratio specified by a parameter ratio\_compr in the CoolSim\_v84\_ec\_init.m file. The vehicle speed for the SC03 cycle is specified in a file called “velvstimeSC03.mat”.

If the A/C model provided in this distribution is used with vehicle simulation models (such as Autonomie), the lookup tables of the top-level model can be replaced with signals provided by the vehicle model. The lookup tables can also be overwritten with data from ‘m’ or ‘mat’ files. An example included is the ‘velvstimeSC03.mat’ file which is loaded in the last line of the ‘point\_ec01’ script overwrites the previously defined vehicle velocity lookup table.

The input parameters that can be safely modified are included in the point\_mc01.m and ‘point\_ec01.m’ files and also are listed below. For definitions of all the other input parameters, see the master initialization files ‘CoolSim\_v84\_mc\_init.m’ and ‘CoolSim\_v84\_ec\_init.m’. Most of the input parameter units are metric, powers are in Watts, speeds in rpm, temperatures in Kelvin, and pressures in Pascals. The parameters that do not follow the rule are specifically marked in the initialization files.

setting\_recirc: Air mass flow rate going to the evaporator from the cabin divided by total air mass flow rate going to the evaporator including that from ambient.

setting\_blower: For mechanical drive compressor only, this setting can be 1, 2 or 3

setting\_blend: Fraction of the maximum capacity of the heater applied to the air exiting the evaporator

time\_ambvstime: Time vector for ambient conditions vs. time lookup tables [sec]

p\_ambvstime: Ambient pressure vs. time lookup vector [Pa].

t\_ambvstime: Ambient temperature vs. time lookup vector [K].

rh\_ambvstime: Ambient relative humidity vs. time lookup vector [%].

time\_rpmvstime: Time vector for vehicle engine RPM vs. time lookup table [sec]

rpm\_rpmvstime: Engine RPM vs. time lookup vector [m/s]

time\_velvstime: Time vector for vehicle velocity vs. time lookup table [sec]

vel\_velvstime: Vehicle velocity vs. time lookup vector [m/s]

time\_solarintvstime: Time vector for rate of solar radiation absorbed by the cabin interior vs. time lookup table [sec]

htin\_solarintvstime : Rate of solar radiation absorbed by the cabin interior vector [W].

time\_solarshellvstime: Time vector for rate of solar radiation absorbed by the cabin interior vs. time lookup table [sec]

htin\_solarshellvstime: Rate of solar radiation absorbed by the cabin interior vs. time lookup vector [W]

htin\_enginecomp: Heat addition rate to cabin air from engine compartment [W]

tairin\_cond: Condenser incoming air temperature

time\_fanonoffvstime: Time vector for condenser fan on/off signal vs. time lookup table [sec]

onoff\_fanonoffvstime: On/off signal vector for condenser fan on/off signal vs. time lookup table

pinit\_cabin: Cabin initial pressure [Pa].

tinit\_cabin: The cabin air initial temperature [K].

rhinit\_cabin: Cabin air initial relative humidity [%]

tinit\_cabinint: Initial temperature of cabin interior [K]

tinit\_cabinshell: Initial temperature of cabin shell [K]

ttarg\_cabin: The target temperature for the cabin [K]. Used for determining if A/C is on or off.

out\_decim: Adjusts the frequency of saving output data in ‘To Workspace’ blocks as described above.

trun: Length of simulation time [sec]

**Model Outputs:**

The main outputs that would be most relevant for a vehicle model are collected in Simulink ‘To Workspace’ blocks on the right side of the top level simulation model. The frequency of recording the output data is determined by the input parameter ‘out\_decim’. Other ‘To Workspace’ blocks that save the data for the entire simulation run for the plotting files are found in various lower levels of the simulation models. The units for all output parameters are metric, powers are in Watts, speed in RPM, temperatures in Kelvins, pressures in Pascals. All variables saved in Simulink ‘To Workspace’ blocks used for the plotting are listed below.

time1: Simulation time. This time vector can be used with the vectors saved in other ‘To Workspace’ blocks to plot the results, like ‘plots\_basic\_ec’ and ‘plots\_cabin\_ec’ do.

rpm\_compr: Compressor speed, revolutions per minute

rpm\_drive1: The compressor drive speed coming from lookup table or from a vehicle model. It represents engine speed or electric motor speed if drive is electric. Although it is also input, it is saved at the same rate as other outputs for post processing purposes.

power\_compr: Compressor shaft power

power\_condfan: Power to drive the condenser fan. This is a simple estimation approach for the portion of total fan power used to drive the air through the condenser. The user is encouraged to add his own calculation.

power\_blower: Power to drive the evaporator blower. Simple model, users are encouraged to add their own calculation.

mdaout\_evap: Mass flow rate (column 1), temperature (column2), and relative humidity (column 3) of the humid air exiting the evaporator.

onoff\_evap1: Control signal to compressor based on evaporator-out air temperature, if value is 1.0, compressor cycles on, if value is zero, compressor cycles off.

qwr\_cond: The heat transfer rate to the refrigerant in the condenser

qwr\_evap: The heat transfer rate to the refrigerant in the evaporator

pdcompr: Pressure downstream of the compressor. This is the system high pressure used to cycle off compressor when it rises too high.

systplow: Pressure upstream of the compressor. This is the system low pressure used to shut down compressor when it falls too low.

velvehicle: Vehicle velocity. Although it is also input, it is saved at the same rate as other outputs for post processing purposes.

afr\_txv: TXV flow area divided by the TXV maximum flow area

tevapwall: Evaporator wall temperature where the sensor is assumed. Used for compressor RPM control.

pcabin: Cabin variables including pressure (column 1), temperature (column 2), density (column 3), specific enthalpy (column 4), H2O mass (column 5), relative humidity (column 6) and absolute humidity (column 7)

ambient: The ambient pressure, temperature and relative humidity. Although it is also input, it is saved at the same rate as other outputs for post processing purposes.

tcabinint: Temperature of the cabin interior (seats, dashboard, etc.)

tcabinshell: Temperature of the cabin shell

onoff\_cabin1: For mechanical compressor drive only, control signal to compressor based on cabin temperature, if value is 1.0, compressor cycles on, if value is zero, compressor cycles off.

htinsolar: Rate of solar energy absorption by the cabin shell (column 1) and cabin interior (column 2). Although it is also input, it is saved at the same rate as other outputs for post processing purposes.

hx\_cabinair: Heat transfer rates for cabin air including from cabin shell (column1) to cabin interior (column2) and from engine compartment (column2).

recirc: Cabin air recirculation rate: air mass flow rate to evaporator from cabin divided by the total air mass flow rate to evaporator

**References**

1. Incropera, F., P., and DeWitt, D. P., “Fundamentals of Heat and Mass Transfer,” 2nd edition, 1985, John Wiley and Sons. Inc.
2. Chen, J. C., “Correlation for Boiling Heat Transfer to Saturated Fluids in Convective Flow,” I&EC Process Design and Development, Vol. 5. No. 3 July 1966 pp.322-329
3. Chang, Y., and Wang, C., “A Generalized Heat Transfer Correlation for Louver Fin Geometry,” Int.J. Heat and Mass Transfer, vol. 40, No. 3, pp. 533-544, 1997
4. Kiss, T. and Lustbader, J., “Comparison of the Accuracy and Speed of Transient Mobile A/C System Simulation Models,” SAE Int. J. Passeng. Cars – Mech. Syst. August 2014 7:739-754; doi:10.4271/2014-01-0669. <http://www.nrel.gov/docs/fy14osti/61868.pdf>
5. Kiss, T., Chaney, L., and Meyer, J., “A New Automotive Air Conditioning System Simulation Tool Developed in MATLAB/Simulink,” SAE Int. J. Passeng. Cars – Mech. Syst. 6(2):2013, doi:10.4271/2013-01-0850. <http://www.nrel.gov/docs/fy13osti/57488.pdf>