FORD MOTOR COMPANY

CAN Invader BT Controller User Guide

Set-up and Klippel Interface

Bellanca, Louis (L.)

5/17/2016

User guide for the Bluetooth interface program for the CAN Invader. Purpose is to allow diagnostic control of infotainment system during automated Klippel BSR testing.

Table of Contents

Set Up CAN Invader and UD100 Bluetooth Adapter for use with Python Scripting. 2

General Usage 9

Klippel Integration 9

APPENIX: Background Information and to do: 10

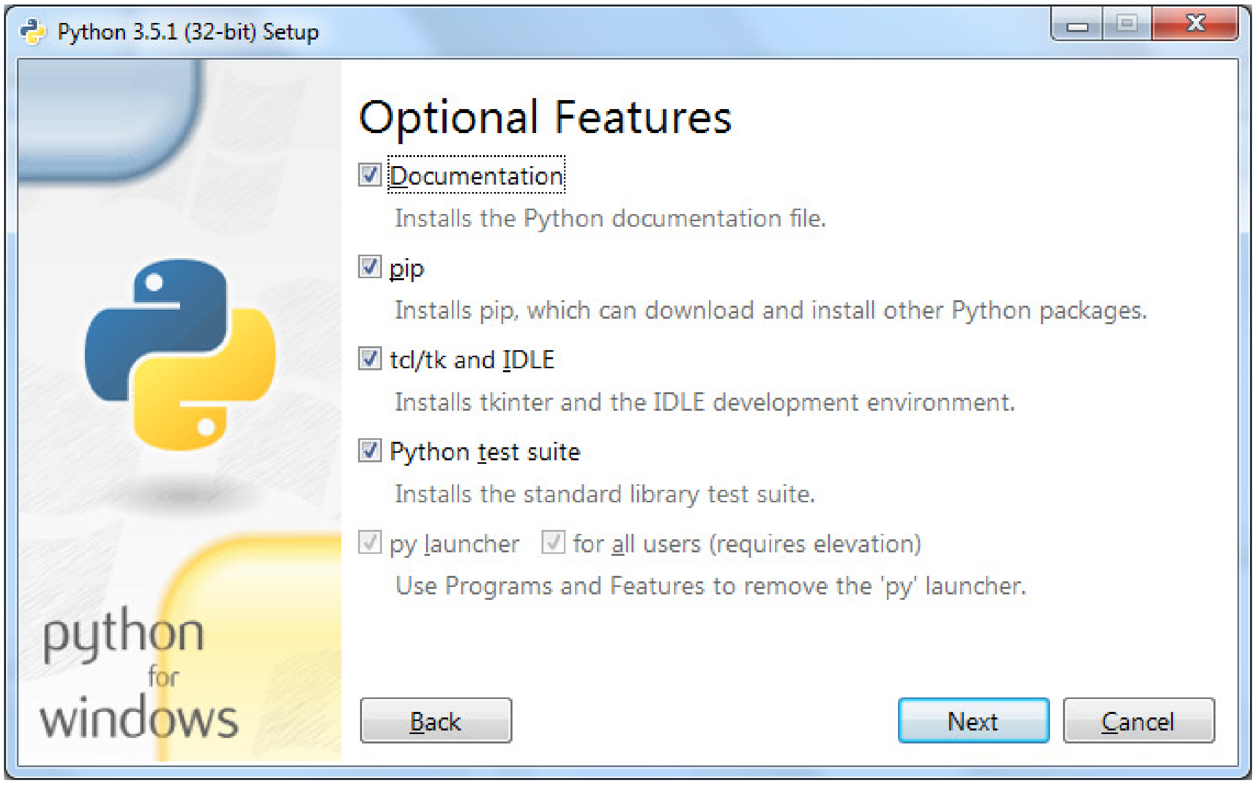
- Overview of operation 10
- Summary of current commands: 10
- OOBDesk Windows Installation 11
- OPEN QUESTIONS / TODO LIST: 12

Set Up CAN Invader and UD100 Bluetooth Adapter for use with Python Scripting.

L.Bellanca: May 17, 2016: V3

Step 1: Download all the required sw to your PC. This includes:

-Download Python35 in file "python-3.5.1.exe" msi files do not exists for 3.5 branch. Note needs to be 32 bit version – 64 bit python does not work with PyBluez!!!
-Install to C:\Python35 directory – not the default location so you need to modify this during install.
-Make sure to select option to select the following as shown below and "Add Python to environment variables" during the install setup.



Python 3.5.1 (32-bit) Setup

Optional Features

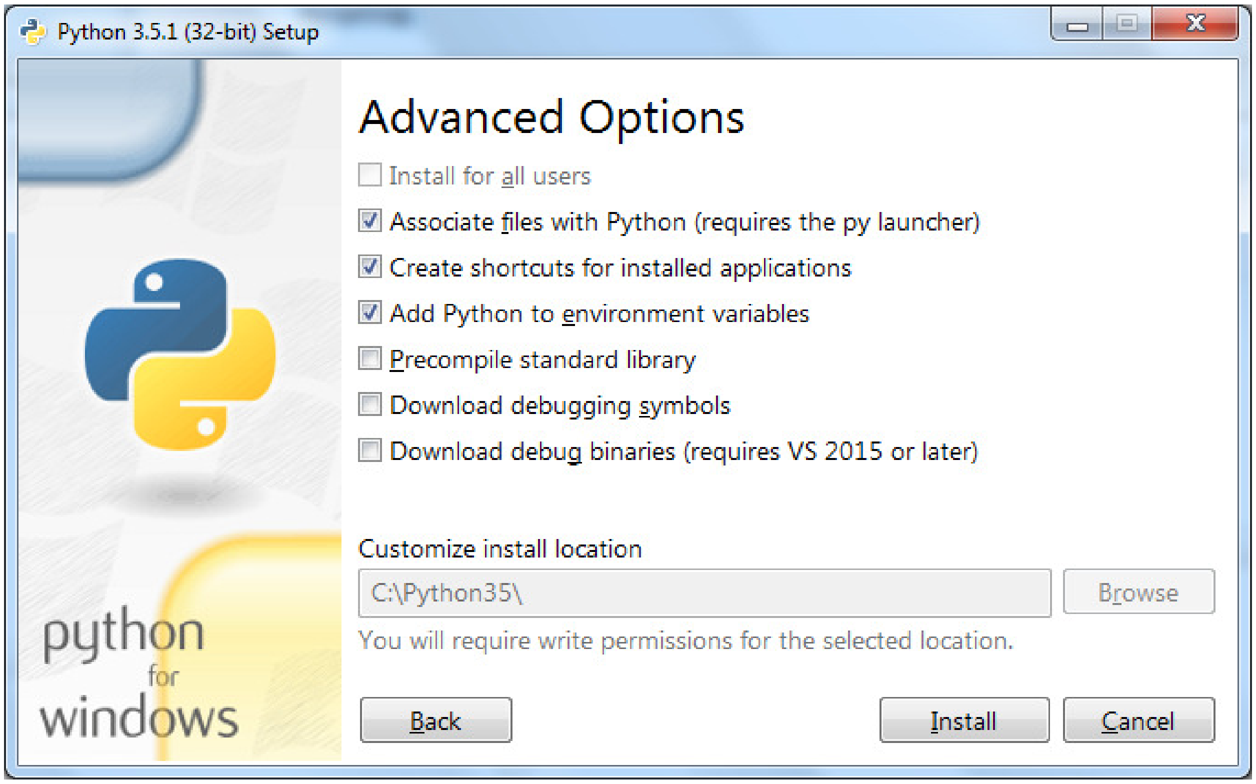
[✓] Documentation
Installs the Python documentation file.

[✓] pip
Installs pip, which can download and install other Python packages.

[✓] tcl/tk and IDLE
Installs tkinter and the IDLE development environment.

[✓] Python test suite
Installs the standard library test suite.

[✓] py launcher [✓] for all users (requires elevation)
Use Programs and Features to remove the 'py' launcher.


Python 3.5.1 (32-bit) Setup

Advanced Options

- [ ] Install for all users
- [✓] Associate files with Python (requires the py launcher)
- [✓] Create shortcuts for installed applications
- [✓] Add Python to environment variables
- [ ] Precompile standard library
- [ ] Download debugging symbols
- [ ] Download debug binaries (requires VS 2015 or later)

Customize install location

C:\Python35\

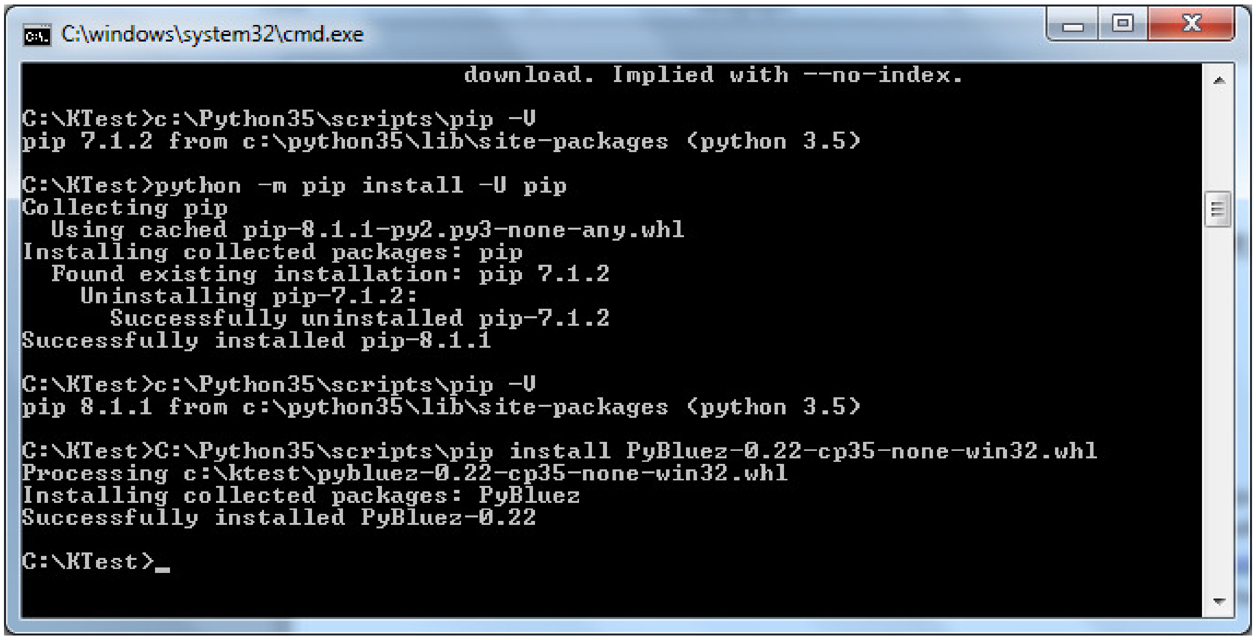
You will require write permissions for the selected location.

-Then you need to download the correct PyBluez installer for 3.5 : "PyBluez-0.22-cp35-none-win32.whl"

```
-You may need to upgrade pip first : python -m pip install -U pip
```

-CD to directory where you downloaded the Pybluez whl file.

```
-To install, enter a command shell and type: "C:\Python35\scripts\pip install
PyBluez-0.22-cp35-none-win32.whl
```



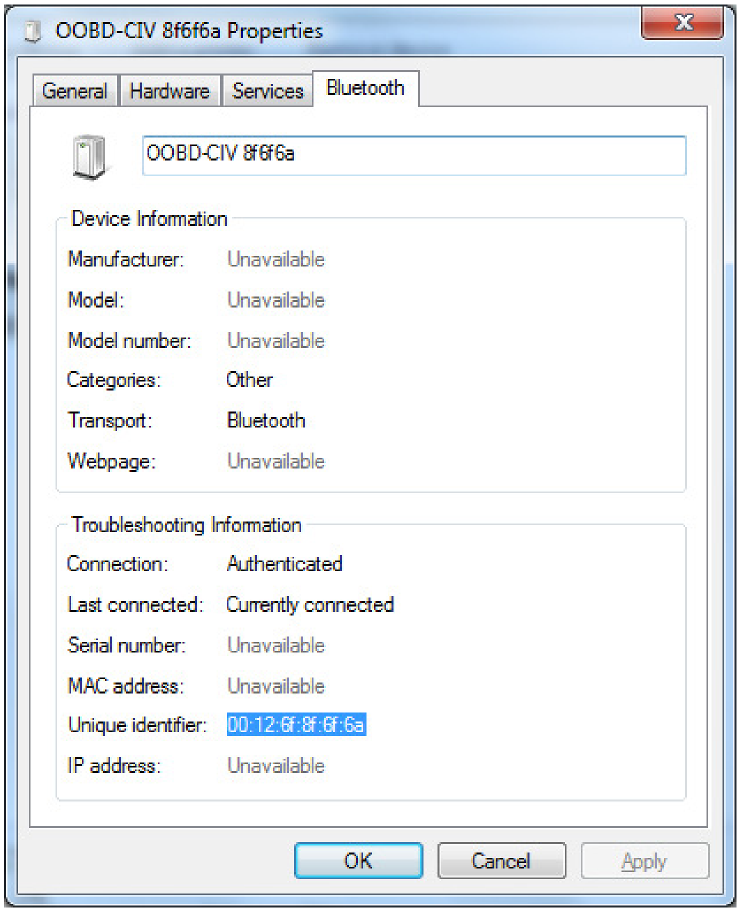
```
                              download. Implied with --no-index.

C:\KTest>c:\Python35\scripts\pip -V
pip 7.1.2 from c:\python35\lib\site-packages (python 3.5)

C:\KTest>python -m pip install -U pip
Collecting pip
  Using cached pip-8.1.1-py2.py3-none-any.whl
Installing collected packages: pip
  Found existing installation: pip 7.1.2
    Uninstalling pip-7.1.2:
      Successfully uninstalled pip-7.1.2
Successfully installed pip-8.1.1

C:\KTest>c:\Python35\scripts\pip -V
pip 8.1.1 from c:\python35\lib\site-packages (python 3.5)

C:\KTest>C:\Python35\scripts\pip install PyBluez-0.22-cp35-none-win32.whl
Processing c:\ktest\pybluez-0.22-cp35-none-win32.whl
Installing collected packages: PyBluez
Successfully installed PyBluez-0.22

C:\KTest>_
```

Step 2 – Make Connections and configure files

-Connect the UD100 to your PC USB port and observe a solid blue light. Ensure device installs without any errors as it should work with WIN7 with no other installs or downloads needed.
-Connect the CAN Invader to a powered diagnostic J1962 connector (aka vehicle or test bench)- observe the solid green light and a blinking blue light on unit.
-Refresh your PC Bluetooth device list and look for a device called "OOBD – CIV – xxxx" and right click for properties.
-You need to pair to the device with the PC first for the program to work. Enter the PIN code "1234" - note 1234 worked for me but 0000 did not.
-Note that you may need to remove/delete device for list if you had it previously paired but then allowed another PC to pair with it. So try deleting it if you are having trouble paring the device and you have used the device before.

-Open the file "oobd_control.ini" and edit the MAC address in the file to match what shows on your PC BT device properties for the unique identifier field:

-To set the default config file that is loaded on program launch, edit the file "ConfigFile.py" and ONLY change the index number where indicated.



OOBD-CIV 8f6f6a

Device Information

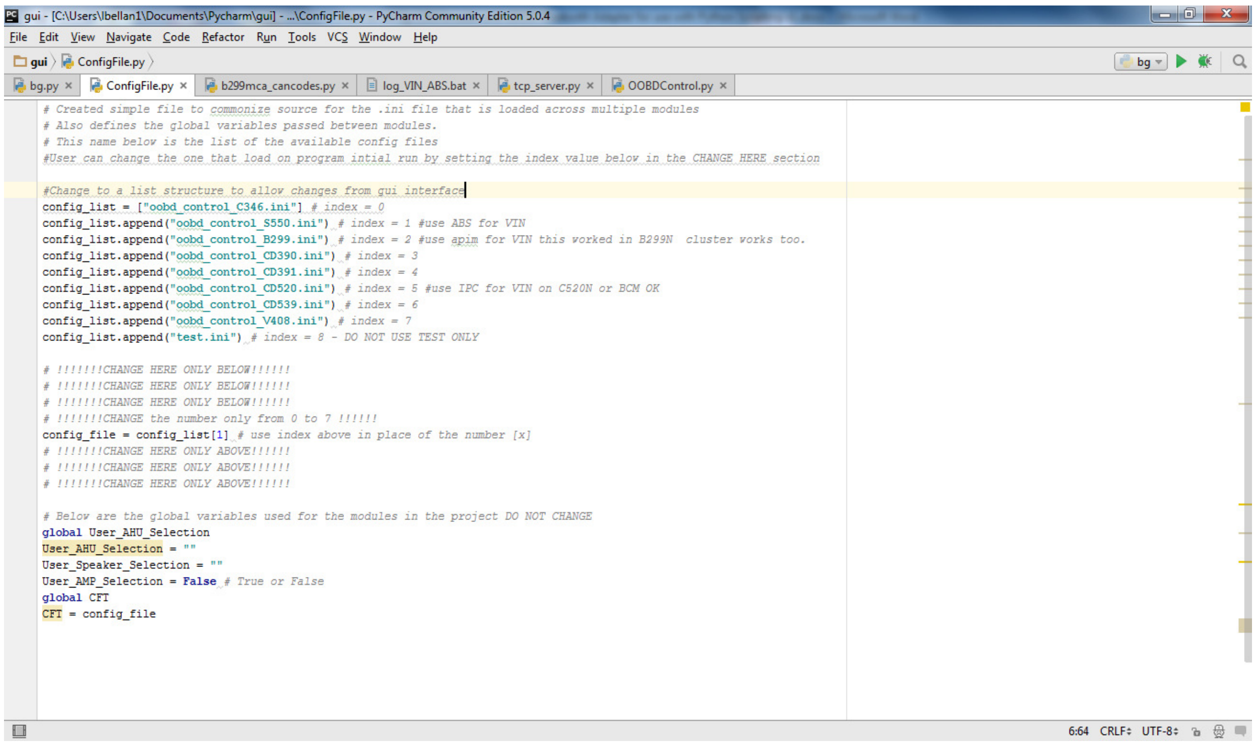
Manufacturer:	Unavailable
Model:	Unavailable
Model number:	Unavailable
Categories:	Other
Transport:	Bluetooth
Webpage:	Unavailable

Troubleshooting Information

Connection:	Authenticated
Last connected:	Currently connected
Serial number:	Unavailable
MAC address:	Unavailable
Unique identifier:	00:12:6f:8f:6f:6a
IP address:	Unavailable

Step3 – Configuration and run program.

-Set the initial configuration file that is loaded when program runs by modifying the ConfigFIle.py file in a text editor. Refer to file for instructions as this may change slightly in various releases:

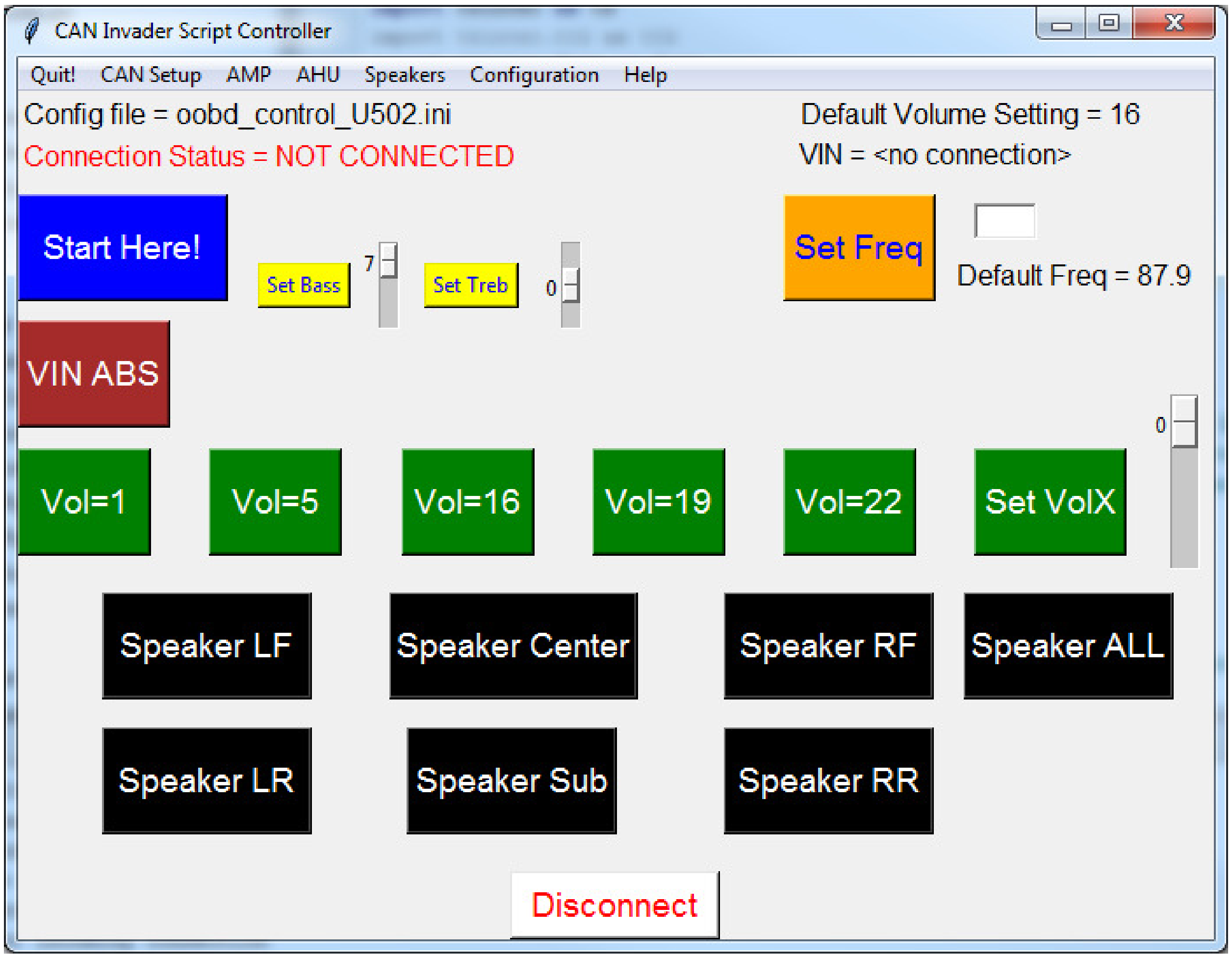

gui - [C:\Users\lbellan1\Documents\Pycharm\gui] - ...\ConfigFile.py - PyCharm Community Edition 5.0.4
File Edit View Navigate Code Refactor Run Tools VCS Window Help
gui ConfigFile.py
bg.py ConfigFile.py b299mca_cancodes.py log_VIN_ABS.bat tcp_server.py OOBDControl.py
bg
6:64 CRLF UTF-8

```python
# Created simple file to commonize source for the .ini file that is loaded across multiple modules
# Also defines the global variables passed between modules.
# This name below is the list of the available config files
#User can change the one that load on program intial run by setting the index value below in the CHANGE HERE section

#Change to a list structure to allow changes from gui interface
config_list = ["oobd_control_C346.ini"] # index = 0
config_list.append("oobd_control_S550.ini") # index = 1 #use ABS for VIN
config_list.append("oobd_control_B299.ini") # index = 2 #use apim for VIN this worked in B299N  cluster works too.
config_list.append("oobd_control_CD390.ini") # index = 3
config_list.append("oobd_control_CD391.ini") # index = 4
config_list.append("oobd_control_CD520.ini") # index = 5 #use IPC for VIN on C520N or BCM OK
config_list.append("oobd_control_CD539.ini") # index = 6
config_list.append("oobd_control_V408.ini") # index = 7
config_list.append("test.ini") # index = 8 - DO NOT USE TEST ONLY

# !!!!!!!CHANGE HERE ONLY BELOW!!!!!!
# !!!!!!!CHANGE HERE ONLY BELOW!!!!!!
# !!!!!!!CHANGE HERE ONLY BELOW!!!!!!
# !!!!!!!CHANGE the number only from 0 to 7 !!!!!!
config_file = config_list[1] # use index above in place of the number [x]
# !!!!!!!CHANGE HERE ONLY ABOVE!!!!!!
# !!!!!!!CHANGE HERE ONLY ABOVE!!!!!!
# !!!!!!!CHANGE HERE ONLY ABOVE!!!!!!

# Below are the global variables used for the modules in the project DO NOT CHANGE
global User_AHU_Selection
User_AHU_Selection = ""
User_Speaker_Selection = ""
User_AMP_Selection = False # True or False
global CFT
CFT = config_file
```

-Verify the information in the xxxxx.ini file you selected. Refer to the example.ini file for information if you need to modify anything.

-To launch program you double click on the file bg.py

-For beginner user, you can press the “Start Here” button to connect and configure the CAN interface with 1 button press.


Config file = oobd_control_U502.ini

Connection Status = NOT CONNECTED

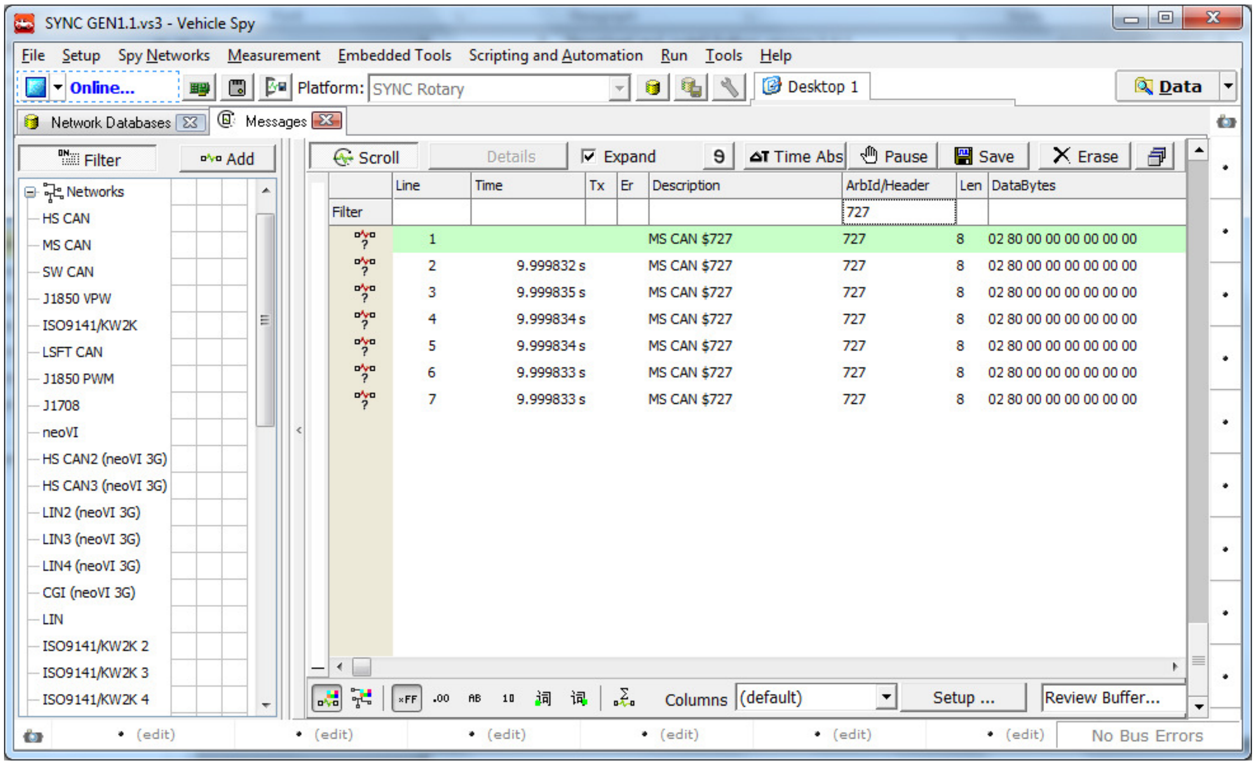
Default Volume Setting = 16

VIN = <no connection>

Default Freq = 87.9

-For the advanced engineering mode, press "shift e" (capital E) to show the engineering interface.

-You should then be able to verify the TP messages are being sent and other commands as requested if you have access to a CAN bus monitor ex Vehicle Spy or CANAlyzer



Line	Time	Tx	Er	Description	ArbId/Header	Len	DataBytes
Filter					727		
1				MS CAN $727	727	8	02 80 00 00 00 00 00 00
2	9.999832 s			MS CAN $727	727	8	02 80 00 00 00 00 00 00
3	9.999835 s			MS CAN $727	727	8	02 80 00 00 00 00 00 00
4	9.999834 s			MS CAN $727	727	8	02 80 00 00 00 00 00 00
5	9.999834 s			MS CAN $727	727	8	02 80 00 00 00 00 00 00
6	9.999833 s			MS CAN $727	727	8	02 80 00 00 00 00 00 00
7	9.999833 s			MS CAN $727	727	8	02 80 00 00 00 00 00 00

General Usage

After BT dongle is connected to PC and CAN Invader is connected to vehicle, make sure the correct configuration file is loaded. This can be chosen under the menu item “Configuration”.

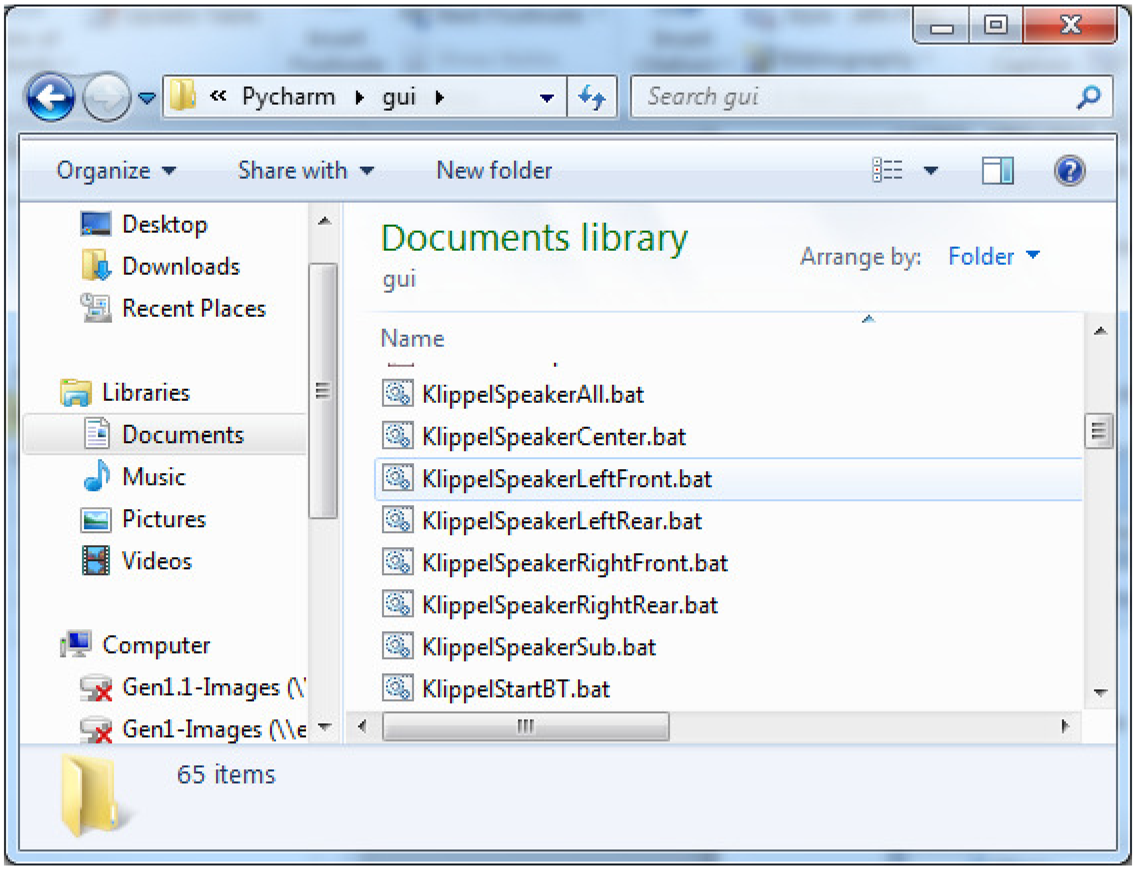
User can also hard code the default configuration by editing the “Config.py” file in a text editor. File has instructions for this.

1-Press “Start here” button – this will make connection via BT and set the default frequency, bass, treble, check for AMP presence, and volume.

2-If a user wants to enable certain speakers, then press the corresponding label “Speaker xx”.

Klippel Integration

The gui panel can be controlled by running certain bath files. These files are:



KlippelSpeakerAll.bat
KlippelSpeakerCenter.bat
KlippelSpeakerLeftFront.bat
KlippelSpeakerLeftRear.bat
KlippelSpeakerRightFront.bat
KlippelSpeakerRightRear.bat
KlippelSpeakerSub.bat
KlippelStartBT.bat

The KlippelStartBT.bat file will invoke the “Start here” function.
The 7 speaker files will control the speakers as indication in title.

Possible need to add volume control?

APPENIX: Background Information and to do:

Overview of operation

The program bg.py has 4 primary functions:
1-Show graphical panel to user to allow triggering of functions, selecting of configurations, and general system status information.
2-Start the server (tcpserver) to monitor for CAN requests over the BT interface. Initiate outgoing requests to the server using pynetcat based on user actions.
3-Start and monitor another socket server thread that communicates over another port that is only used for communicating with the Klippel program via batch file runs.
4-Define all of the communication functions such as the CAN messages needed to perform the tasks of volume control, speaker control, VIN reading, bass/treble control, frequency setting and BT connecting.

Summary of current commands:

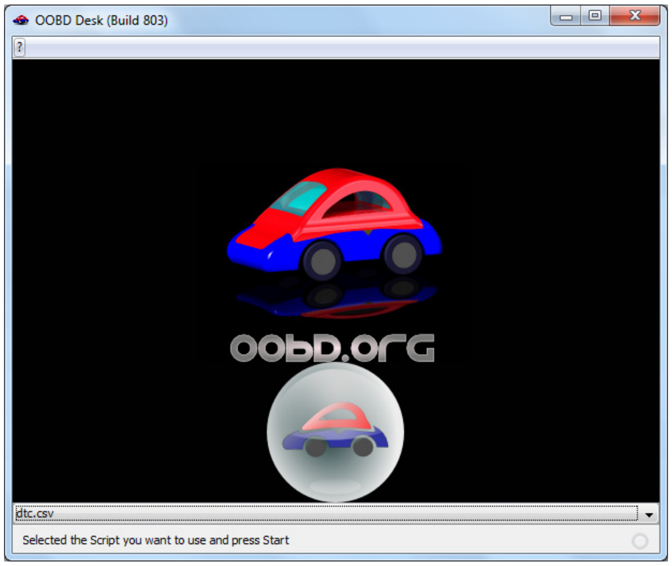
```
# MASTER DICTIONARY DEFINE
=====================================================================================
========

    canFunctionSets = {"setFreq": set_freq,
                       "setFreqX": set_freq_x,  # x in freq x 10 ex 983 = 98.3
                       "setVolumeFront": set_volume_front,
                       "setVolumeRear": set_volume_rear,
                       "setBassVisteon": set_bass_visteon,
                       "setTrebVisteon": set_treb_visteon,
                       "setBassX": set_bass_x_pana,  # x in hex
                       "setTrebX": set_treb_x_pana,  # x in hex
                       "AMPsetBassX": AMP_set_bass_x,  # x in hex
                       "AMPsetTrebX": AMP_set_treb_x,  # x in hex
                       "readVIN": read_vin_ahu,
                       "readVINrcm": read_vin_rcm,
                       "readVINsync": read_vin_sync,  # sends VIN command to SYNC
                       "readVINabs": read_vin_abs,  # sends VIN command to abs
                       "readVINbcm": read_vin_bcm,  # sends VIN command to bcm
                       "readVINpcm": read_vin_pcm,  # sends VIN command to pcm
                       "readVINipc": read_vin_ipc,  # sends VIN command to cluster
                       "speakerEnableRF": set_rf_on,
                       "speakerEnableRFtwt": set_rf_on_twt,
                       "speakerEnableLF": set_lf_on,
                       "speakerEnableLFtwt": set_lf_on_twt,
                       "AMPspeakerEnableLFtwt": AMP_set_lf_on_twt_4,
                       "speakerEnableRR": set_rr_on,
                       "speakerEnableLR": set_lr_on,
                       "speakerEnableCntr": set_cntr_on,
                       "speakerEnableCntrtwt": set_cntr_on_twt,
                       "speakerEnableSub": set_subwoofer_on,
                       "speakerEnableRF4": set_rf_on_4,  # for clarion
                       "speakerEnableRFtwt4": set_rf_on_twt_4,
                       "AMPspeakerEnableRFtwt4": AMP_set_rf_on_twt_4,
                       "speakerEnableLF4": set_lf_on_4,
                       "speakerEnableLFtwt4": set_lf_on_twt_4,
                       "speakerEnableRR4": set_rr_on_4,
                       "AMPspeakerEnableRRtwt4": AMP_set_rr_on_twt_4,
                       "speakerEnableLR4": set_lr_on_4,
                       "AMPspeakerEnableLRtwt4": AMP_set_lr_on_twt_4,
                       "speakerEnableCntr4": set_cntr_on_4,
                       "speakerEnableCntrtwt4": set_cntr_on_twt_4,
                       "speakerEnableSub4": set_subwoofer_on_4,
                       "AMPspeakerEnableSub4": AMP_set_subwoofer_on_4,
                       "speakerEnableAllOn4": set_all_on_4,
                       "AMPspeakerEnableAllOn4": AMP_set_all_on_4,
                       "speakerEnableAllOn": set_all_on,
                       "radioOn": radio_on,  # sends command to MFD
                       "radioOnSYNC": radio_on_sync,  # sends command to MFD
                       "radioOnahu": radio_on_ahu,  # GEN3 SYNC?
                       "AMPsetVolumeX": AMP_set_volume_x,  # x in hex 00 to 1D
                       "setVolumeX": set_volume_x}  # x in hex 00 to 1D
```

OOBDesk Windows Installation

Install these files from :
https://01e98c1964c442ed78c1d88c8682cb3d6da46598.googledrive.com/host/0B795A63vSunRZU90QWVPUnowMmM/

Java18_portable_unlimited_policy.zip
OOBDesk_Setup_R803.zip

This will open a java program used to interface with the CAN INVADER.
Press the Question Mark “?” on top header of program to set up the device COM port and script directory.



Selected the Script you want to use and press Start

OPEN QUESTIONS / TODO LIST:

1-Still not clear how the commands like “setVolume” and “SpeakerEnable” are translated into the respective DID settings. I cannot find this library setting - must be in the OODB database somewhere.

ANSWER – these commands are defined in the file “b299mca_cancodes.py”

2-It would be desireable to have a way to change these settings:

1. FM channel - always 87.90MHz?
2. Volume step is at 18~22 marks
3. Bass is at MAX
4. Treble is at Nominal.
5. Speaker Enable L r F R
6. Module ID for messages (x727 for AHU, x?DSP/AMP, x?ANC?)
7.

Sending Message example:
Need to call : def sendCanData(self, seq, reqId=None, checkAnswer=False)

Step 1 – set the CAN ID:
self.sendCtrlSeq(["p 6 5 $"+reqId])

Step 2-send the message data
currentAnswer = self.sendRawData(data=command+"\r").decode("ascii")

Done with

oobd.sendCanData([command[3:]], checkAnswer=True)

Example From Comamnd line:

```
C:\python34\python.exe "C:\KTest\Klippel_Niehl_Final\OOBD_scripts\pynetcat.py" localhost 50000 raw12345678
```

Sends default ID 04 12 34 56 78 00 00 00

```
C:\python34\python.exe "C:\KTest\Klippel_Niehl_Final\OOBD_scripts\pynetcat.py" localhost 50000 raw12345678
```

Modified the tcp_server.py file for function “command[:3] == "raw": where you can now change the ID in raw message format as follows:

raw12345678,7XX – will send message with new ID 7XX
raw12345678 = will send message with the default ID = 727 (or defined in ini file)
raw1003 = sends 727 03 10 03 00 00 00
raw1003,7DF = sends 7DF 03 10 03 00 00 00

TO DO LIST:
Make get VIN command pass the ECU ID also – just added read_vin_sync-DONE
Make set bass and set treble pass the data parameter-done
See about using different CAN output on the CAN Invader for c346 pins- CAN invader supports 6/14 and 3/11 AN pins only. Total 2 channels.
Set radio on in the script for AHU – DONE
Add addition speaker control functions – DONE
Design and make Diagnostic jumper for c346 and CAN Invader - DONE
Check interface on a Mustang for CAN Invader. –DONE!
Re-do command definition interface to use data item and uncouple the parameter and access items. – DONE.
Add a window resize feature to store this to ini file – DONE!
Add better feedback on the diag commands-? –DONE! Popup and red screen now used
Add a way to toggle engineering mode – Done by press of “E”
Define the non engineering mode operation - ??? – add single button press!! – DONE!
Add B299 Config file and reverse engineer messages – In progress – DONE!

Add mid range setting for AHU?? – need to find it
Increase time for message response or add as a setting to ini file! – not sure needed now…
Add method for sending addition TP messages to ini file (like 7DF requiredfor B car) – DONE!
Add method to select different config file other than the default one – DONE!
Check interface on Transit Connect DONE!
Check interface on Escape –DONE!
Check interface on Explorer – Done
Check interface on F150 - DOne
Check Interface on Focus - ?
Check Interface on EDGE- ?

Get extra CAN INVADER from EU team as Yu needs to use the device now in plant. – Eddie to coordinate.
Improve exit feature of program to kill the server window – DONE!
Check on Amplifier system- MKT - amp ID is x783 -DONE! In process of updating software with new requirements

To run program type “python bg.py”

Find keep alive for the GEN3 system to prevent transport mode shutdown. 0xFD52 or 0xFD51??

Requirements for the BT CAN setup interface:
1-User needs to choose a setup/vehicle type using a PC screen
2-Once chosen, the correct .bat and .ini files will be selected for the script files.

Operation:
All possible files .ini and .bat files could exist in a \storage directory.
Once a vehicle type is selected, then the correct files are copied into the working directory for the correct sequence of files to use.

Oobd_control.ini should be common by architecture – aka no plant changing SHOULD be needed.
Bat file could be dependent on the audio configuration and trim level and supplier of AHU.

Example seq:
<user selects the vehicle type>
0-BT interface self configures based on user input
1-Startserver for BT connection
2-Connect to BT/ configure CAN / Tester present ON
3-Radio On
4-Set Treb
5-Set Bass
6-Set Freq

7-Set Volume
Run Klippel Test
If fail, then
8-SetspeakerFR
Run Klippel Test
9-SetspeakerFL
Run Klippel Test
10-SetspeakerRR
Run Klippel Test
11-SetspeakerRL
12-Log VIN of DUT