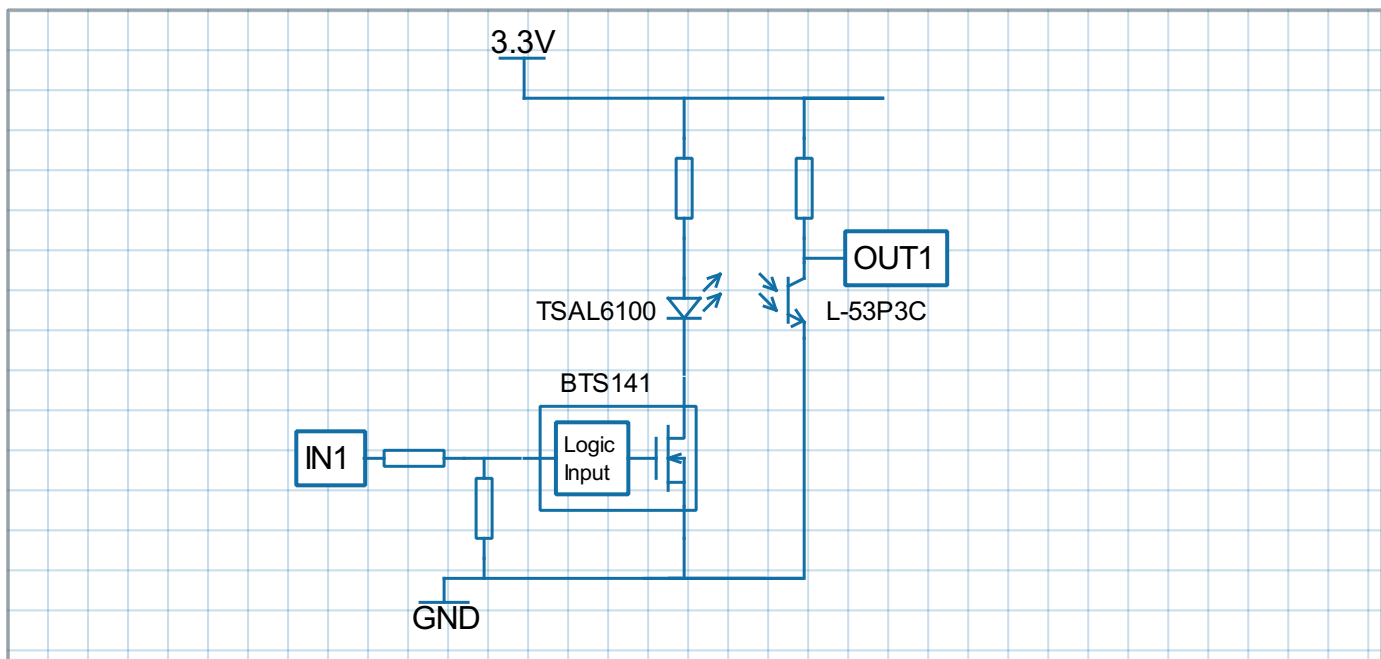


Optical communications system realized using LabJack T7 and MatDeck - GUI Configuration

In this document, we will illustrate an example of an electronic circuit which also functions as a basic optical communications system. The optical communication system is based on a light emitting diode (LED) which is used as a transmitter, and a photo diode which is used as an optical receiver.

Schematics of the electronic circuits

The schematics of the described optical system are displayed below. It should be pointed out that the schematics are created in MatDeck, which is a suitable platform for various professional drawings.



Functionality

- IN1 and OUT1 demonstrate IR communication with DSP filtering

Demo board schematic pin descriptions

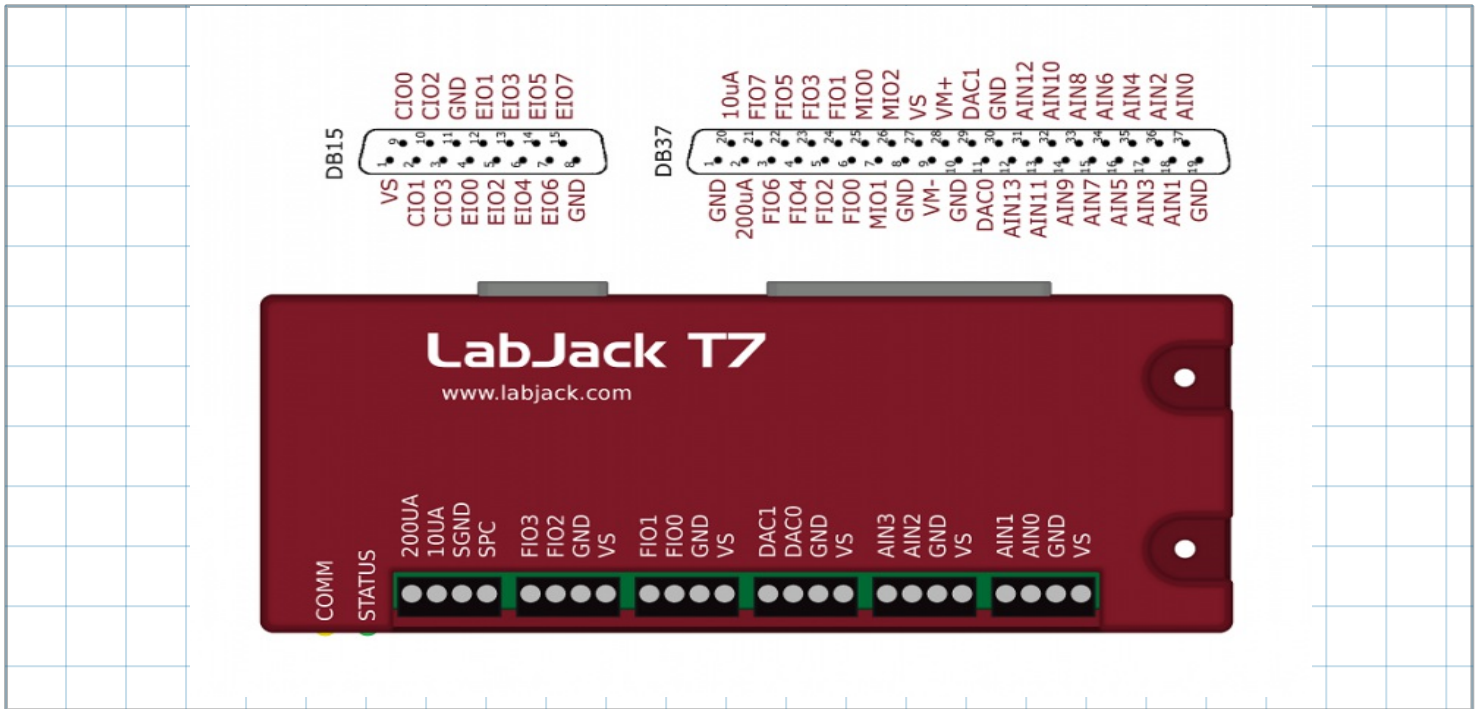
- IN1 - input for IR LED driver
- OUT1 - output from IR sensor

Connection to ADC unit

- IN1 is connected to the PWM output.
- OUT1 is connected to Analog inputs 1

Parts:

- For power driver is used BTS 141 which is logic level low side driver.
- Photo-transistor L-53P3C NPN 5MM, 940nm
- IR emitter TSAL6100 20° 940nm



Configuration of LabJack T7 using GUI

The LabJack T7 device, shown above, is used in the experiment for several roles. First, the pulse width modulation (PWM) is used to produce a input signal, denoted as IN1 on the schematics above. The IN1 signal is generated using the DIO extended features, PWM output at DIO0(FIO0) and a high-speed counter at the DIO18 output. These extended features are explained in detail in the following links:

<https://labjack.com/support/datasheets/t7/digital-io/extended-features>

<https://labjack.com/support/datasheets/t7/digital-io/extended-features/pwm-out>

<https://labjack.com/support/datasheets/t7/digital-io/extended-features/high-speed-counter>

Second, the output signal is measured using the LabJack T7 device, stream reading at AIN0. Finally, the noise in the communications is simulated using DAC0 so that a signal close to the output is added.

MatDeck supports LabJack functions which can be used directly inside the MatDeck script to configure LabJack devices, and to generate and acquire signals from electronic circuits as described above. However, MatDeck also provides Graphical User Interface (GUI) plug-ins for simple, effective configuration. There are three plug-ins for three different groups of pins: `ljdio_config_form()` is used to configure DIOs, `ljaio_config_form()` is used to configure AINs and `ljdac_config_form()` is used to configure DACs. Here, details about the configuration of the selected features in this experiment are explained.

GUI for DIO Configuration

GUI form for DIO configuration can be started using `ljdio_config_form()`, as follows. The form is embedded within the canvas and used for the configuration of DIOs.

```
1 f := ljdioT7_config_form(0, "Form0")
```

PWM Out configuration in MatDeck script

PWM Out at FIO0(DIO0) requires the clock source, thus the clock is first configured. There are three parameters to select for the configuration: the clock source, the clock divisor and the roll value for the given clock. There are three different clocks supported by T7, the most common is clock0 whose frequency is 80MHz. The clock divisor can be any power of two from 1, 2, up to 256. In this example, we use divisor

value equal to one. The roll value is determined according to the desired frequency of the PWM Out signal. For example, if the desired frequency is 1kHz, the roll value is $80\text{Hz}/\text{Divisor}/1\text{kHz}=80000$. In the GUI, it is possible to choose and set the desired frequency or desired roll value.

PWM out at FIO0 (DIO0) is configured by selecting the appropriate option from the drop down menu. In the GUI, it is possible to set the desired value of the duty cycle directly to 50%.

Configuration of High Speed Counter

High speed counter at DIO18 will be used to check the obtained PWM out frequency. It is done by selecting the appropriate option from the drop down menu at DIO18.

LabJack - DIO Configuration Form

Device Type: T7
Connection Type: ANY
Device ID: ANY

DIO(0:7) FIO(0:7) | DIO(8:15) EIO(0:7) | DIO(16:19) CIO(0:3) | DIO(20:22) MIO(0:2) | EF Clock Source

Clock

Clock Source: Clock0
Clock Divisor: 1

Roll Value or Frequency

Roll Value or Frequency: Roll Value
Roll Value: 80000

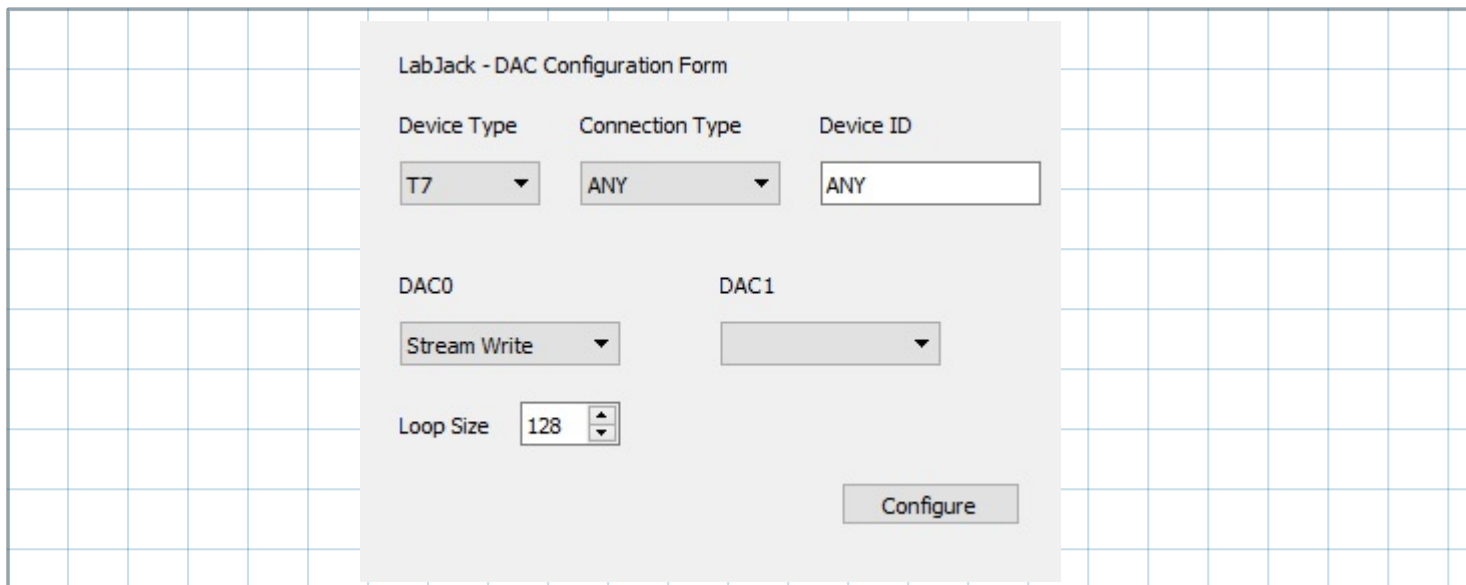
For following EFs:
Frequency In
Pulse Width In
Line-to-Line In

Configure

GUI for DAC Configuration

The interference is simulated using an analog output which is produced using DAC0. The output of DAC0 is generated using the stream output, which is configured here. We use the GUI form for the DAC configuration `ljdac_config_form()`, which is enabled in a similar manner as the previous form. The form is embedded within the canvas and used for the configuration of DACs.

```
3 f1 := ljdac_config_form(0, "Form_DAC")
```



The screenshot shows a GUI window titled "LabJack - DAC Configuration Form" overlaid on a grid. The form contains the following fields:

- Device Type:** A dropdown menu with "T7" selected.
- Connection Type:** A dropdown menu with "ANY" selected.
- Device ID:** A text input field containing "ANY".
- DAC0:** A dropdown menu with "Stream Write" selected.
- DAC1:** A dropdown menu that is currently empty.
- Loop Size:** A spin box with the value "128" displayed.
- Configure:** A button located at the bottom right of the form.

```
4 ljdac_config_form_configure(f1)
```

GUI for AIN configuration

The signal at the output, OUT1 on the schematics, is read by AIN0 in the stream mode. The stream mode is set by writing in the appropriate registers. Analog inputs have four values to set: resolution index, settling time, range, and single-ended or differential reading. In the stream mode, resolution index and settling time are set for all streamed analog inputs, while range and negative channel are configured separately. MatDeck provides `ljaiio_config_form()` which can be used to set all the parameters graphically, which is very convenient for the user. In the next segment, there is an illustration on how to use the form for AIN configuration. At the beginning, the form is evoked by calling `ljaiio_config_form()`. The form is embedded within the canvas and used for the configuration of AINs.

```
5 f2 := ljainT7_config_form(0, "Form AIN")
```

LabJack - AIN Configuration Form

Device Type: T7
 Connection Type: ANY
 Device ID: ANY

AI(0:3) AI(4:7) AI(8:11) AI(12:13) Stream All

AIN Configuration:

AIN	AIN0	AIN1	AIN2	AIN3
AIN0	Stream			
Range	-10:10			
Negative Ch.	GND			

Configure

```
6 ljainT7_config_form_configure(f2)
```

Use of Configured LabJack T7

In order to use the configuration and use the device, the LabJack T7 device should be opened in the document:

```
7 dev := ljdevice_open("any", "any", "any")
```

Enable Extended Features

Configured Extended Features are enabled, however, if some features are time sensitive they should be restarted by performing the additional enable:

```
8 //Enable and run counter at DI018
9 ljdevice_write(dev, "DI018_EF_ENABLE", 1)
10 a := ljdevice_last_error("s") //Check for errors in configuration
```

```
a = "LJ_SUCCESS"
```

Check PWM Out Frequency

The frequency of the PWM Out signal can be checked by reading the value of the counter after one second.

```

12 // Wait for one second
13 sleep(1000)
14 // Read the counter
15 counter := ljdevice_read(dev, "DI018_EF_READ_A")

```

counter = 2158 Hz

Adding interference

The interference is simulated using an analog output which is produced using DAC0. The output of DAC0 is generated using the stream output, which is configured using `ljdock_config_form`, and used here.

```

16 STREAM_OUT0_LOOP_VALUES := [0, 5, 0, 5, 0, 5, 0, 5, 0, 5]
17 // Write stream-out
18 ljdevice_write(dev, "STREAM_OUT0_LOOP_SIZE", 10)
19 ljdevice_awrite(dev, "STREAM_OUT0_BUFFER_F32", STREAM_OUT0_LOOP_VALUES)
20 ljdevice_write(dev, "STREAM_OUT0_SET_LOOP", 1)

```

Reading values in the stream mode

The function `ljdevice_stream_start()` is used to prepare the device for reading, the user has to specify scans per read, and scan rate. The function `ljdevice_stream_read()` is used to obtain the data. If multiple channels are streamed, all the data is stored in a single vector, thus `reshape` is used to separate the channels.

```

21 ljdevice_stream_start(dev, 10000, ["DIO0"; "AIN0"; "STREAM_OUT0"], 10000)
22 pwm := ljdevice_stream_read(dev, 20000)
23 ljdevice_stream_stop(dev)
24 ch := mat_reshape(pwm, [size(pwm) / 2, 2])

```

Closing the T7 Device

Finally, at the end, all the extended features should be disabled and the device should be closed.

```

25 //Disable clock0, high speed counter, PWM out and close the device
26 ljdevice_write(dev, "DIO_EF_CLOCK0_ENABLE", 0)
27 ljdevice_write(dev, "DI018_EF_ENABLE", 0)
28 ljdevice_write(dev, "DIO0_EF_ENABLE", 0)
29 ljdevice_close(dev)

```

pwm = 20000 elements vector

Signal Processing and Visualization

The obtained signal is visualized using a MatDeck graph. It is evident from observing the graph that the signal is corrupted with interference.

```

30 ttime := ynodes(x, 0, 9999 / 10000, 10000)
31 IN1 := subset(ch, 0, 0, size(ch) / 2 - 1, 0)
32 OUT1 := subset(ch, 0, 1, size(ch) / 2 - 1, 1)
33 graph := join_mat_cols(ttime, IN1)
34 graph1 := join_mat_cols(ttime, OUT1)

```

35

```
36 graph_test := subset(graph1, 0, 0, 100, 1)
```



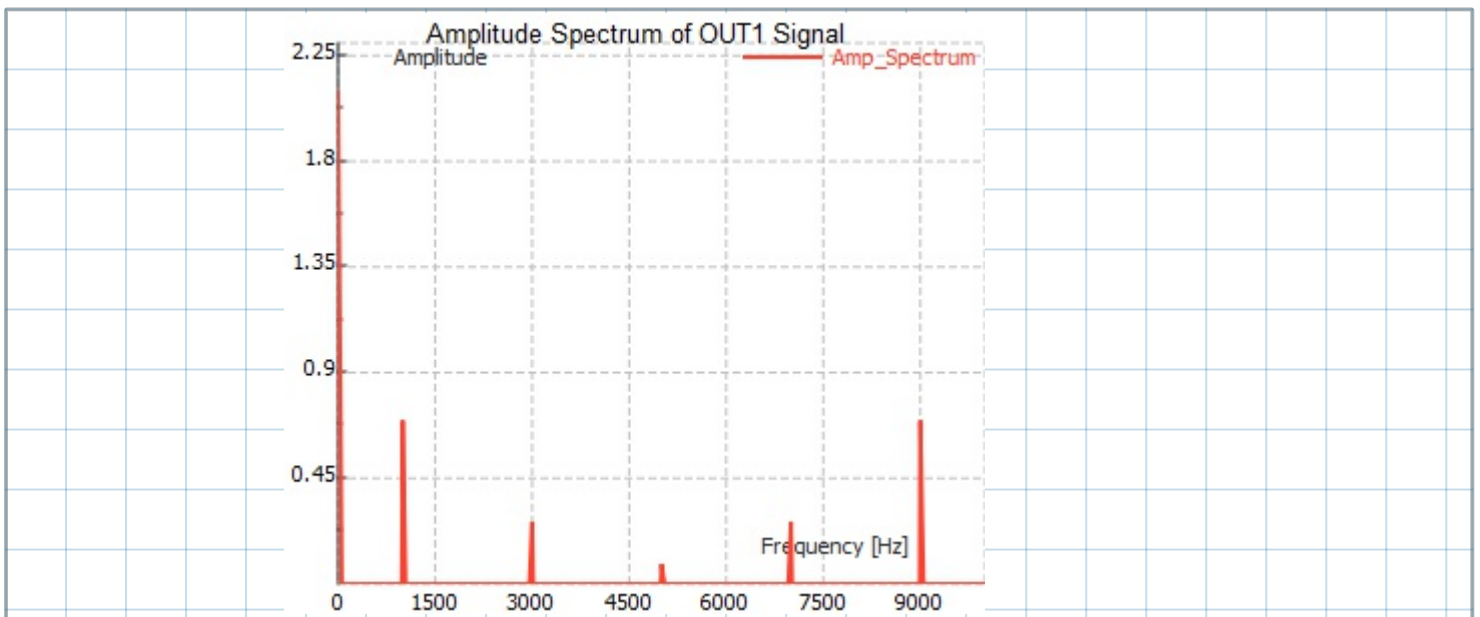
Signal Processing using MatDeck Digital Filter

It is evident from the time domain graph, that it is corrupted with high frequency interference. However, it is also possible to check the amplitude spectrum of the signal using the MatDeck function, `fft()`. The amplitude spectrum shows the frequency of the interference in Hz. The information is used to design the filter in order to remove interference from the signal.

```
37 Amp_Spectrum := abs(fft1(OUT1)) / 10000
```

```
38 Fre := ynodes(x, 0, 9999, 10000)
```

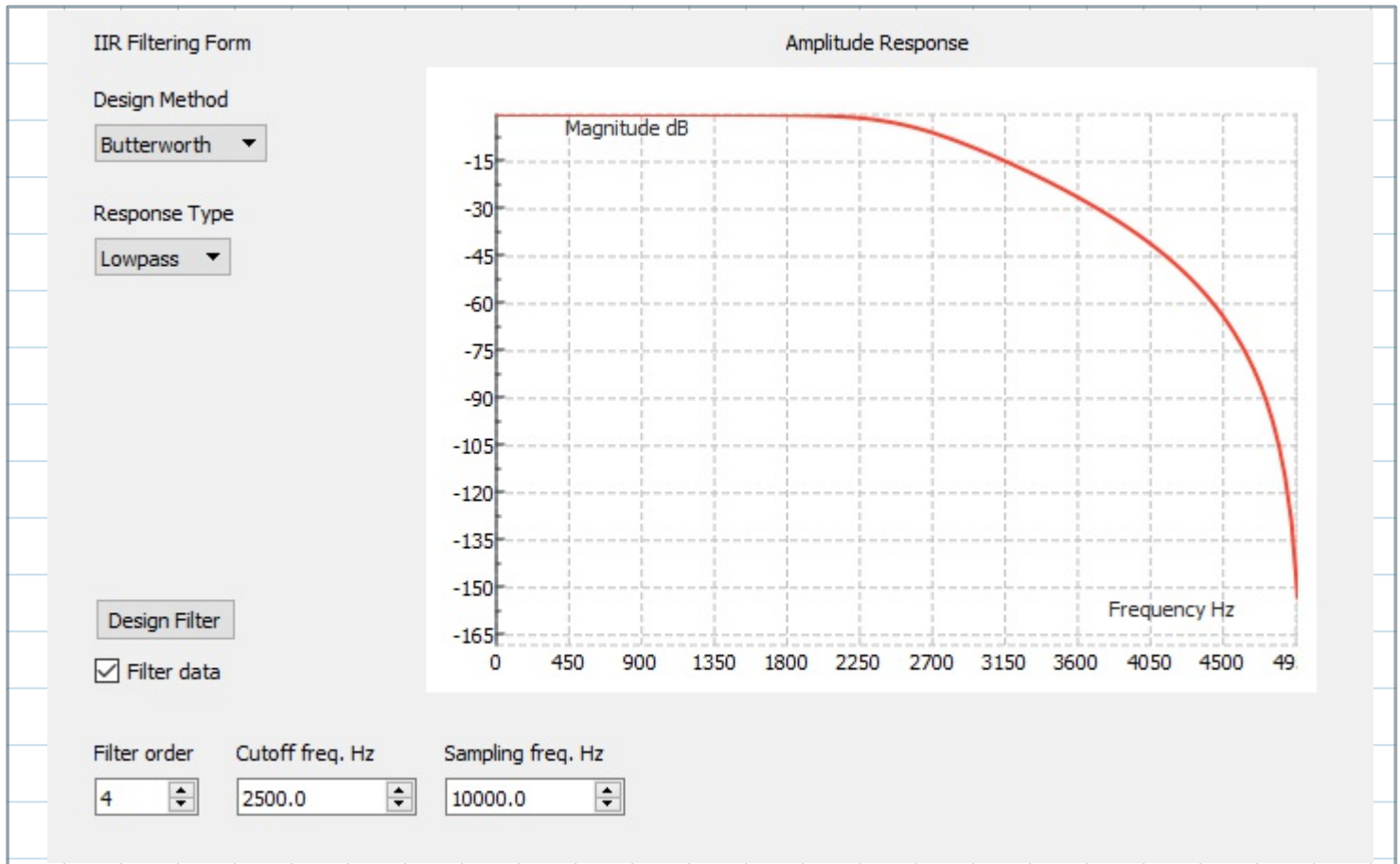
```
39 Amp_Sgraph := join_mat_cols(Fre, Amp_Spectrum)
```



The idea is to use a simple low pass Butterworth filter of the fourth order, with the cutoff frequency equal to 2500Hz. In MatDeck, we can use the IIR form to design the aforementioned filter and to apply the filter to the signal. In order to use the form, the user has to create the form using the function `f:=iirfilteringform(0,"form1")`. Here, the first argument is the widget parent or 0, and the second argument is the string form name. The function returns the identification of the widget form. The created IIR Filtering

Form is placed in the Canvas at the position where the function `embed_widget()` is called with the appropriate widget identity. As explained earlier, the IIR filtering form can be used to filter the OUT1 signal. In order to perform the filtering, the Filter data check box has to be checked. The results are derived by calling the functions `iirfilteringresult(w1,OUT1)`, where `f` is the identification of the widget, `x` is the input signal and `y` is the output signal.

```
40 w1 :=iirfilteringform(0, "FormIIR2")
```



At the end, we just have to visualize the time domain and amplitude spectrum of the filtered OUT1 signal.

```
41 OUT1_filtered :=iirfilteringresult(w1, OUT1)
42 Amp_Spectrum1 := abs(fft1(OUT1_filtered)) / 10000
43 Amp_Sgraph1 := join_mat_cols(Fre, Amp_Spectrum1)
44 Time_OUT1f := join_mat_cols(ttime, OUT1_filtered)
45 graph_test1 := subset(Time_OUT1f, 0, 0, 100, 1)
```