PhD30:
Measurements prep ATF May26

Laura Karina Pedraza, Nuria Fuster, Daniel Esperante, Benito Gimeno

PhD Journal, 16/04/2026

laura.pedraza@ific.uv.es

PhD30: Measurements prep ATF May26

LabRF meeting, 17/04/26

INSTITUT DE FÍSICA CORPUSCULAR

Introduction

I. Acquisition and processing scripts
   A) Acquisition identification
   B) Digital signal processing
   C) Calibration
   D) Resolution results

II. Tasks to perform for ATF measurements
   A) Before measurements
   B) Checklist for packing
   C) During the shifts

Conclusion

I. Acquisition and processing scripts

A) Acquisition identification

shift number (1,2,3,4,5,6)

axis: X or Y

`S3_res_Y_001`

número de la medida

Identificación de la adquisición:

res: resolution
cal: calibration

Datos adicionales que se agregan al excel al momento de tomar las medidas:

fecha y hora de adquisición

intensidad y valores de los kickers

datos del osciloscopio

waveforms

calibración para esa medida

	A	B	C	D	E	F	G	H	I	J	K	L	M
1		ACQUISITION									OSCILLOSCOPE		
2	run_id	comment	shift	axis	time_stamp	N_wavefroms	I_bunch	I_ZH2P	I_ZV2P	cal_id	sampling_rat	time_windc	points_wave
3							(nC/bunch	(A)	(A)		(GS/s)	(ns)	
4	S3_res_Y_001	2DC	3	Y	16/12/2025 03:07	1000	0.544	-0.05	1.3	S3_cal_Y_001	2	50	100
5	S3_res_Y_002	2DC	3	Y	16/12/2025 03:53	1000	0.656	-0.05	1.3	S3_cal_Y_001	2	50	100
6	S3_res_Y_003	2DC	3	Y	16/12/2025 04:22	1000	0.88	-0.05	1.3	S3_cal_Y_001	2	50	100

I. Acquisition and processing scripts

A) Acquisition identification

shift number (1,2,3,4,5,6)

axis: X or Y

Valor kicker en sub-adquisición

`S3_cal_Y_001_Kv=1.00`

número de la medida

res: resolution
cal: calibration

Identificación de la adquisición:

Datos adicionales que se agregan al excel al momento de tomar las medidas:

fecha y hora de adquisición

waveforms

intensidad y valores de los kickers

datos del osciloscopio

	A	B	C	D	E	F	G	H	I	J	K	L
1					ACQUISITION					OSCILLOSCOPE		
2	run_id	comment	shift	axis	time_stamp	N_wavefroms	I_bunch	I_ZH2P	I_ZV2P	sampling_rat	time_windo	points_wavef
3							(nC/bunch	(A)	(A)	(GS/s)	(ns)	
4	S3_cal_Y_001	2DC	3	Y	16/12/2025 03:07	50	0.656	-0.05		2	50	100
5	S3_cal_Y_001_K_v=1.00								1			
6	S3_cal_Y_001_K_v=2.20								2.2			
7	S3_cal_Y_001_K_v=2.80								2.8			

I. Acquisition and processing scripts

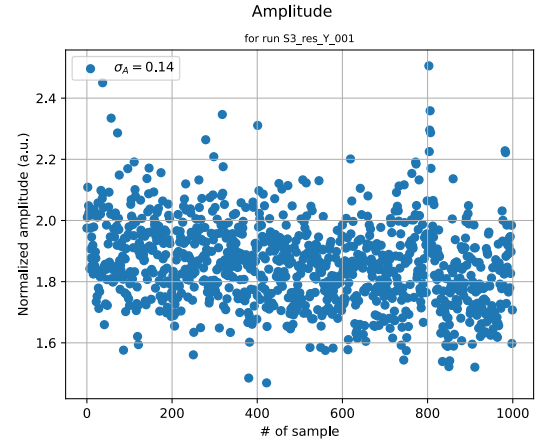
B) Digital signal processing

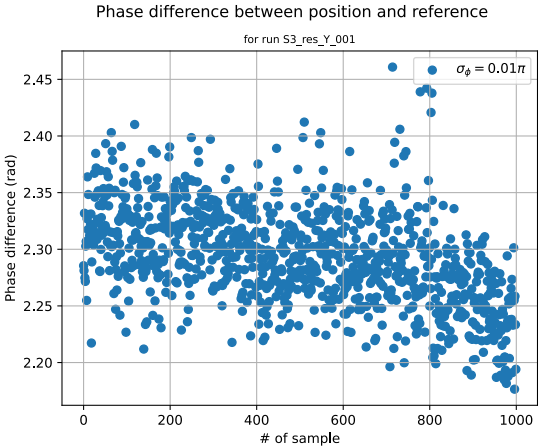
	A	N	O	P	Q	R
1		DDC PARAMETERS				
2	run_id	f_IF_pos	f_IF_ref	sigma_DDC	sigma_A	sigma_phi
3		(Hz)	(Hz)	(Hz)	(a.u.)	(rad.π)
4	S3_res_Y_001	6.00E+07	4.00E+07	3.00E+07	0.14	0.01
5	S3_res_Y_002	6.00E+07	4.00E+07	3.00E+07	1.38	0.03
6	S3_res_Y_003	6.00E+07	4.00E+07	3.00E+07	0.07	0.97

frecuencia de oscilación de señales de posición y referencia en IF

σ de filtro gaussiano aplicado en DDC

stdv de Amplitud y fase determinada para el set de medidas





I. Acquisition and processing scripts

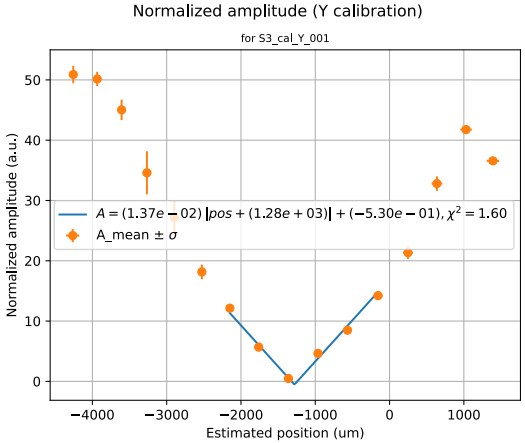
C) Calibration

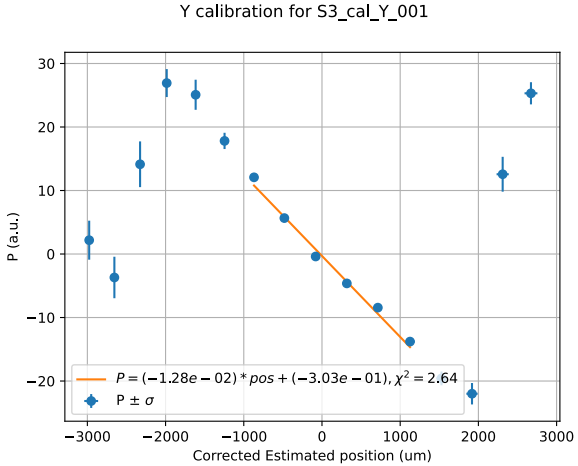
To insert when doing the calibration:

	A	R	S
1		RANGE	
2	run_id	d_bound	u_bound
3		(um)	(um)
4	S3_cal_Y_001	-2500	0
5	S3_cal_Y_001_K_v=1.00		
6	S3_cal_Y_001_K_v=2.20		
7	S3_cal_Y_001_K_v=2.80		

Given calibration results:

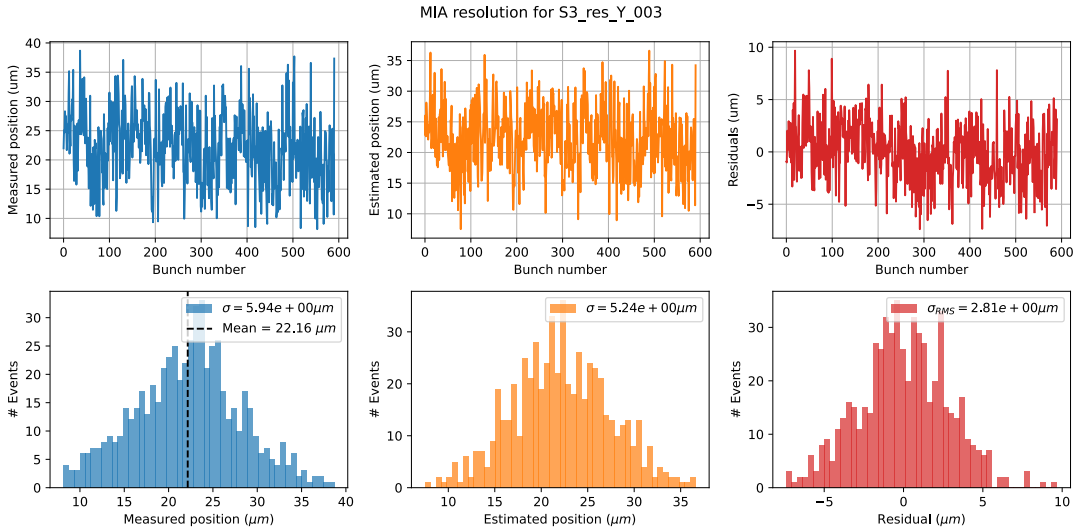
	A	T	U	V	W	X	Y	Z
1		RESULTS						
2	run_id	dyn_range	static_offset	slope	slope_error	intercept	intercept_er	chi2
3		(um)	(um)	(1/um)	(1/um)			
4	S3_cal_Y_001	1994.674	1280.614	-0.013	0.001	-0.303	0.42	2.635
5	S3_cal_Y_001_K_v=1.00							
6	S3_cal_Y_001_K_v=2.20							
7	S3_cal_Y_001_K_v=2.80							





I. Acquisition and processing scripts

D) Resolution results



	A	S	T	U	V	W	X	Y
1		SYNC	RESULTS					
2	run_id	window_sync	N_bunches	num_strips	mean_pos	std_dev	resolution	res_err
3		(s)			(um)	(um)	(um)	(um)
4	S3_res_Y_001	0.21	582	10	99.74	10.393	**8.844**	
5	S3_res_Y_002	0.21	500	10	82.068	6.403	**4.523**	
6	S3_res_Y_003	0.21	591	10	22.157	5.937	**2.813**	
7								

PhD30: Measurements prep ATF May26

LabRF meeting, 17/04/26

INSTITUT DE FÍSICA CORPUSCULAR

Introduction

I. Acquisition and processing scripts
   A) Acquisition identification
   B) Digital signal processing
   C) Calibration
   D) Resolution results

II. Tasks to perform for ATF measurements
   A) Before measurements
   B) Checklist for packing
   C) During the shifts

Conclusion

II. Tasks for ATF measurements

A) Before measurements (IFIC and KEK)

Pedidos:

- confirmar que todos los pedidos estén hechos
- todo llega a tiempo excepto filtros de X-microwave (Mouser) para los cuales se tiene alternativa de mini-circuits

Phase shifters + medidas componentes:

- tune the phase shifters with the hybrid couplers
  - VNA: generar cambio de 180 grados a la frecuencia del dipolo
  - bloquear los phase-shifters en la fase deseada
  - medir los S-parameters de los componentes de down-conv.

Envío de material:

- lista de material que debe ser enviado en las maletas de la electrónica
- empacado del BPM
- formalización de los envíos

Electrónica:

- 6 líneas de down-conversion:
  - guardar los S-parameters de todos los componentes
  - ensamblaje y prueba de respuesta (signal analyser o VNA)

Electrónica:

- 4 líneas de local-oscillator:
  - montaje
  - ensamblar, probar y montar la PCB que conecta los PLLs, atenuadores y el M5
  - La PCB ya está fabricada, pero falta que lleguen los componentes
- crates y ensamblaje:
  - diseño y construcción de crates
  - Crate LO montaje en IFIC. Crate del mixing se monta en ATF.
  - test de funcionamiento después de ensamblaje

Adquisición de datos oscilloscopio:

- integrar código con excel y algoritmo de adquisición
- optimización de la adquisición:
  - time-stamps para identificar pasos demorados
  - influencia del # de puntos de la waveform
  - extraer # trigger?
  - adquisición de varios canales simultaneamente
- preparación de la adquisición:
  - implementar en nuevo oscillo
  - implementar adquisición para 2 BPMs

II. Tasks for ATF measurements

B) Checklist for packing

Caja BPM:
- caja rígida / cartón grueso
- BPM
- gaskets (grandes y pequeños)
- acolchonamiento

Maleta 1:
- 2 osciloscopios
- cables de alimentacion
- adaptadores para Japón?
- cables de probe?

Maleta 2:
- crate con local oscillator, PLL y μ-controladores
- componentes mini-circuit down-conversion
- placa down-conversion X-microwave
- cables SMA
- Herramientas: llaves, etiquetadora

II. Tasks for ATF measurements

C) During the shifts

PhD30: Measurements prep ATF May26

LabRF meeting, 17/04/26

INSTITUT DE FÍSICA CORPUSCULAR

Introduction

I. Acquisition and processing scripts
   A) Acquisition identification
   B) Digital signal processing
   C) Calibration
   D) Resolution results

II. Tasks to perform for ATF measurements
   A) Before measurements
   B) Checklist for packing
   C) During the shifts

Conclusion

III. Oscilloscope acquisition

A) Code

1. Sets the trigger to single
2. Checks if the trigger hits Stop every 0.05 seconds
3. Switches to position channel
4. Acquires waveform from position channel
5. Switches to reference channel
6. Acquires waveform from reference channel
7. Saves waveforms in dictionary

```
# Wait for adquisition to complete
for i in range(1,N_waveforms+1):

    # Set trigger mode: Single
    scope.write('TRIGger:SWEep SINGle')
    # time.sleep(0.01)
    time_diff = time.time() - time_stamp
    time_stamp = time.time()
    print(f"Time to SWEep SINGle: {time_diff}")

    # Verify that the trigger has already triggered anything and it's stopped
    while scope.query(':TRIG:STAT?').strip() != 'STOP':
        # print("Trigger not detecting signal")
        time.sleep(0.05)
    # As soon as triggered, get time stamp
    time_diff = time.time() - time_stamp
    time_stamp = time.time() # THIS IS THE TIME-STAMP THAT REMAINS
    print(f"The time to detect a new trigger: {time_diff}")

    # # Extract at the same time BPM and kickers data from EPICS
    # updated_epics_data = f_adquire_epics(waveform_storage["epics_data"])
    # waveform_storage["epics_data"] = updated_epics_data
```

```
    # Extract information and generate np array
    scope.write('WAVeform:SOURce '+pos_chan) # CHANNEL FOR POSITION DATA
    time_diff = time.time() - time_stamp
    time_stamp = time.time()
    print(f"Time to change the channel to position: {time_diff}")
    # waveform_pos = scope.query(':WAVeform:DATA?')
    waveform_pos = scope.query_binary_values(':WAV:DATA?', datatype='h', container=np.array)
    time_diff = time.time() - time_stamp
    time_stamp = time.time()
    print(f"Time to acquire position waveform: {time_diff}")
    scope.write('WAVeform:SOURce '+ref_chan) # CHANNEL FOR REFERENCE DATA
    time_diff = time.time() - time_stamp
    time_stamp = time.time()
    print(f"Time to change tha channel to reference: {time_diff}")
    # waveform_ref = scope.query(':WAVeform:DATA?')
    waveform_ref = scope.query_binary_values(':WAV:DATA?', datatype='h', container=np.array)
    time_diff = time.time() - time_stamp
    time_stamp = time.time()
    print(f"Time to acquire reference waveform: {time_diff}")

    # Store waveforms into dictionary
    dict_byte["position_signal"]["Waveform"].append(waveform_pos)
    dict_byte["position_signal"]["time_stamp"].append(time_stamp)
    dict_byte["reference_signal"]["Waveform"].append(waveform_ref)
    dict_byte["reference_signal"]["time_stamp"].append(time_stamp)
    time_diff = time.time() - time_stamp
    time_stamp = time.time()
    print(f"Time to add recorded waveforms to the dictionnary: {time_diff}")

    print(f"Waveform {i} added to dictionary \n \n")
```

III. Oscilloscope acquisition

B) Time per step

With a trigger of 3 MHz, and acquiring 100 points in binary


```
Time to restart the loop: 5.125999450683594e-05
Time to SWEep SINGle: 0.0009799003601074219
The time to detect a new trigger: 0.017473936080932617
Time to change the channel to position: 0.000982046127319336
Time to acquire position waveform: 0.27859997749328613
Time to change tha channel to reference: 0.0012557506561279297
Time to acquire reference waveform: 0.00683903694152832
Time to add recorded waveforms to the dictionnary: 4.291534423828125e-05

Waveform 8 added to dictionary
Aquisition performed in a total of 0.3062260150909424 s


Time to restart the loop: 3.1948089599609375e-05
Time to SWEep SINGle: 0.0009479522705078125
The time to detect a new trigger: 0.3237948417663574
Time to change the channel to position: 0.0009868144989013672
Time to acquire position waveform: 0.007429838180541992
Time to change tha channel to reference: 0.0007472038269042969
Time to acquire reference waveform: 0.008801937103271484
Time to add recorded waveforms to the dictionnary: 5.1021575927734375e-05

Waveform 9 added to dictionary
Aquisition performed in a total of 0.3428080081939697 s


Time to restart the loop: 3.910064697265625e-05
Time to SWEep SINGle: 0.0007450580596923828
The time to detect a new trigger: 0.010344982147216797
Time to change the channel to position: 0.0009219646453857422
Time to acquire position waveform: 0.5484340190887451
Time to change tha channel to reference: 0.00115203857421875
Time to acquire reference waveform: 0.0070841312408447266
Time to add recorded waveforms to the dictionnary: 4.8160552978515625e-05

Waveform 10 added to dictionary
Aquisition performed in a total of 0.5687730312347412 s


Time to restart the loop: 5.1021575927734375e-05
Time to SWEep SINGle: 0.0010938644409179688
The time to detect a new trigger: 0.010306119918823242
Time to change the channel to position: 0.0008471012115478516
Time to acquire position waveform: 0.2773470878601074
Time to change tha channel to reference: 0.0011758804321289062
Time to acquire reference waveform: 0.006653785705566406
Time to add recorded waveforms to the dictionnary: 5.2928924560546875e-05

Waveform 11 added to dictionary
Aquisition performed in a total of 0.29753708839416504 s
```

III. Oscilloscope acquisition

C) Zig-zag channel change:

```
# For zig-zag iteration, we start with position channel
scope.write('WAVeform:SOURce '+pos_chan) # CHANNEL FOR POSITION DATA

# Wait for adquisition to complete
for i in range(1,N_waveforms+1):

    # to restart the loop:
    time_diff = time.time() - time_stamp_prov
    time_stamp_prov = time.time()
    print(f"Time to restart the loop: {time_diff}")

    # Set trigger mode: Single
    scope.write('TRIGger:SWEep SINGle')
    # time.sleep(0.01)
    time_diff = time.time() - time_stamp_prov
    time_stamp_prov = time.time()
    print(f"Time to SWEep SINGle: {time_diff}")

    # Verify that the trigger has already triggered anything and it's stopped
    while scope.query(':TRIG:STAT?').strip() != 'STOP':
        # print("Trigger not detecting signal")
        time.sleep(0.05)
    # As soon as triggered, get time stamp
    # time_stamp = time.time()
    time_diff = time.time() - time_stamp_prov
    time_stamp_prov = time.time()
    print(f"The time to detect a new trigger: {time_diff}")

    # # Extract at the same time BPM and kickers data from EPICS
    # updated_epics_data = f_adquire_epics(waveform_storage["epics_data"])
    # waveform_storage["epics_data"] = updated_epics_data
```

```
    # Perform acquisition with zig-zag in channel change:
    if i % 2: # for odd iterations:
        # we start with the position acquisition
        waveform_pos = scope.query_binary_values(':WAV:DATA?', datatype='h', container=np.array)
        # we switch to the reference channel and acquire then:
        scope.write('WAVeform:SOURce '+ref_chan) # CHANNEL FOR REFERENCE DATA
        waveform_ref = scope.query_binary_values(':WAV:DATA?', datatype='h', container=np.array)
    else: # for even iterations:
        # we start with the reference acquisition
        waveform_ref = scope.query_binary_values(':WAV:DATA?', datatype='h', container=np.array)
        # we switch to the position channel and acquire then:
        scope.write('WAVeform:SOURce '+pos_chan) # CHANNEL FOR POSITION DATA
        waveform_pos = scope.query_binary_values(':WAV:DATA?', datatype='h', container=np.array)

    # Store waveforms into dictionary
    dict_byte["position_signal"]["Waveform"].append(waveform_pos)
    dict_byte["position_signal"]["time_stamp"].append(time_stamp)
    dict_byte["reference_signal"]["Waveform"].append(waveform_ref)
    dict_byte["reference_signal"]["time_stamp"].append(time_stamp)
    time_diff = time.time() - time_stamp_prov
    time_stamp_prov = time.time()
    print(f"Time to add recorded waveforms to the dictionnary: {time_diff}\n")

    time_diff_acq = time.time() - time_stamp
    time_stamp = time.time()
    print(f"Waveform {i} added to dictionary")
    print(f"Aquisition performed in a total of {time_diff_acq} s \n \n")
```

III. Oscilloscope acquisition

C) Zig-zag channel change:

With a trigger of 3 MHz, and acquiring 100 points in binary


```
Time to restart the loop: 0.0005619525909423828
Time to SWEep SINGle: 0.00092315673828125
The time to detect a new trigger: 0.2551140785217285
Time to acquire position waveform: 0.004681825637817383
Time to change the channel to reference: 0.0010738372802734375
Time to acquire reference waveform: 0.008845806121826172
Time to add recorded waveforms to the dictionnary: 0.00011110305786132812

Waveform 1 added to dictionary
Aquisition performed in a total of 0.27133703231811523 s


Time to restart the loop: 3.2901763916015625e-05
Time to SWEep SINGle: 0.0008490085601806641
The time to detect a new trigger: 0.2539341449737549
Time to acquire reference waveform: 0.005342960357666016
Time to change the channel to position: 0.0008997917175292969
Time to acquire position waveform: 0.009305000305175781
Time to add recorded waveforms to the dictionnary: 5.602836608886719e-05

Waveform 2 added to dictionary
Aquisition performed in a total of 0.27042698860168457 s


Time to restart the loop: 2.8848648071289062e-05
Time to SWEep SINGle: 0.0008361339569091797
The time to detect a new trigger: 0.010057926177978516
Time to acquire position waveform: 0.5784430503845215
Time to change the channel to reference: 0.0015561580657958984
Time to acquire reference waveform: 0.009119033813476562
Time to add recorded waveforms to the dictionnary: 5.316734313964844e-05

Waveform 3 added to dictionary
Aquisition performed in a total of 0.6001110076904297 s


Time to restart the loop: 5.2928924560546875e-05
Time to SWEep SINGle: 0.0008878707885742188
The time to detect a new trigger: 0.3117389678955078
Time to acquire reference waveform: 0.004457950592041016
Time to change the channel to position: 0.0006952285766601562
Time to acquire position waveform: 0.006609916687011719
Time to add recorded waveforms to the dictionnary: 1.5020370483398438e-05

Waveform 4 added to dictionary
Aquisition performed in a total of 0.32444095611572266 s
```

III. Oscilloscope acquisition

D) Record mode

1. Sets the oscilloscope to ultra-acquire mode
2. Indicates the number of Waveforms to acquire
3. Sets the oscilloscope to run
4. Checks if the oscilloscope has stopped
5. Checks the number of waveforms acquired
6. Changes to channel POS and asks for the waveforms recorded.
7. Changes to channel REF and asks for the waveforms recorded.
8. Translates to Volts and records them in the dictionary

```
# 1. Enable UltraAcquire and set frame count
scope.write(':ACQ:UACQ:STAT ON')
scope.write(f':ACQ:UACQ:FRAM {N_waveforms}')  # Set to record 50 frames per trigger event

# 2. Configure Trigger (ensure it's in NORMAL mode)
scope.write(':TRIG:SWE NORM')

scope.write(':RUN')
print(f"Waiting for {N_waveforms} frames to trigger...")

# Wait until the scope stops automatically
while True:
    status = scope.query(':TRIG:STAT?').strip()
    if status == 'STOP':
        print("Capture Complete!")
        break
    time.sleep(0.5)

# Now check the recorded frames
total_recorded = int(scope.query(':ACQ:UACQ:FRAM?'))
print(f"Total frames stored in UltraAcquire: {total_recorded}")
```

```
# Download All Frames for Channel 1
scope.write(f':WAV:SOUR {pos_chan}')
for i in range(1, N_waveforms + 1):
    scope.write(f':WAV:SEGment {i}') # Navigate to the specific frame
    raw_storage_pos.append(scope.query_binary_values(':WAV:DATA?', datatype='h', container=np.array))

# Download All Frames for Channel 2
scope.write(f':WAV:SOUR {ref_chan}')
for i in range(1, N_waveforms + 1):
    scope.write(f':WAV:SEGment {i}')
    raw_storage_ref.append(scope.query_binary_values(':WAV:DATA?', datatype='h', container=np.array))

for raw in raw_storage_pos:
    volts = (raw - meta_pos["yref"]) * meta_pos["yinc"] + meta_pos["yor"]
    waveform_storage["position_signal"]["Waveform"]["pos"].append(np.array(volts))

for raw in raw_storage_ref:
    volts = (raw - meta_ref["yref"]) * meta_ref["yinc"] + meta_ref["yor"]
    waveform_storage["reference_signal"]["Waveform"].append(np.array(volts))
```

SCPI commands don't work for this!

Conclusion

Thank you for your attention

We gratefully acknowledge the ATF staff for their assistance during the installation and BPM measurements. Special thanks to Toshiyuki Okugi, Alex Aryshev, and Konstantin Popov for their support. We also thank Toshihiro Matsumoto and Hiroshi Kaji for providing the necessary equipment for the measurements.

This work has the support of "Plan de Recuperación, Transformación y Resiliencia (PRTR) 2022 (ASFAE/2022/013)", funded by Conselleria d'Innovació, Universitats, Ciència i Societat Digital from Generalitat Valenciana, and NextGenerationEU from European Union.
This work was partially supported by the European Union's Horizon Europe Marie Sklodowska-Curie Staff Exchanges programme under grant agreement no. 101086276.

laura.pedraza@ific.uv.es

Back-up slides