CAEN
Tools for Discovery

Electronic Instrumentation

PRELIMINARY

User Manual UM7047

i-Spector Digital SDK

Software Development Kit

Rev. 3 - July 12th, 2024

Purpose of this Guide

This Manual contains a brief description of the Software Development Kit for i-Spector Digital (S2570) and i-Spector PSD (S2590)

Change Document Record

Date	Revision	Changes
July 4th, 2019	00	Initial release
February 17th, 2020	01	Added Chapter Rad Cloud
September 23rd, 2020	02	Revised ordering options
July 12th, 2024	03	Removed Chapter Rad Cloud

Symbols, abbreviated terms and notation

ADC	Analog to Digital Converter
API	Application Programming Interface
FPGA	Field Programmable Gate Array
MCA	Multichannel Analyzer
OS	Operating system
ROI	Region of Interest
LoRa	Long Range Communication Protocol
IoT	Internet of Things

Reference Document

[RD1] GD6745 – i-Spector family Quick Start Guide

CAEN S.p.A.
Via Vetraia, 11 55049 Viareggio (LU) - ITALY
Tel. +39.0584.388.398 Fax +39.0584.388.959
info@caen.it
www.caen.it

Index

List of Tables

1 Introduction

i-Spector Digital S2570 and S2590 models can be managed via remote control using a specific **Software Development Kit** (SDK) based on HTTP API, Python or C# coding. The SDK is available for free download.

Models compatible with SDK are listed below:

Unit	Description	Product Code
S2570E	i-Spector Digital 1" (24x24 mm^2) -OEM	WS2570EXOAAA
S2570F	i-Spector Digital 30x30 mm^2 -OEM	WS2570FXOAAA
S2570E	i-Spector Digital 1" (24x24 mm^2) - CsI ASSEMBLY	WS2570EXAAAA
S2570F	i-Spector Digital 30x30 mm^2 - CsI ASSEMBLY	WS2570FXAAAA
S2590C	i-Spector PSD 24x24 mm^2 – ASSEMBLY	WS2590CXAAAA
S25X0	Assembly kit and service for i-Spector OEM – new version	WS25X0ASSBXA

Table 1.1: table of available models and accessories.

2 API usage

The i-Spector is controlled through **JSON-based REST APIs**. HTTP API allows to perform all operations supported by the web-based GUI from a remote host.

i-Spector Digital uses GET/POST HTTP call to both **retrieve spectrum/wave data** and **configure the instrument**. HTTP protocol, the same used to transport web page, has the big advantage of not requiring any special configuration on the network.

Several endpoints are available, as listed in the table below.

Endpoint (WEB PAGE)	Operation	Description
/set_config.cgi	POST	Set configuration of HV/MCA/PSD
/status.cgi	GET	Read the status of the instruments and of all available processing channels (MCA/PSD)
/spectrum.cgi	GET	Get spectrum data
/wavedump.cgi	GET	Get waveform data
/psd.cgi	GET	Get PSD data
/get_mca_config.cgi	GET	Readback MCA/PSD configuration
/resetspectrum.cgi	GET	Reset spectrum
/mca_run.cgi	GET	Start MCA acquisition
/mca_stop.cgi	GET	Stop MCA acquisition
/fb_settings.cgi	GET	Retrieve fabric configuration
/get_sysx.cgi	GET	Retrieve firmware version and installed options

Table 2.1: table of the HTTP GET/POST endpoints available for i-Spector Digital.

In order to access to the endpoint, it is sufficient to perform HTTP GET/POST to the unit address, for example *<i-spector_ip>/set_config.cgi*.

Default i-Spector IP: **192.168.50.2**

i-Spector API support multiple user simultaneously connected to the instruments. Meanwhile you are developing, keep your browser open on the i-Spector Web Interface page, in order see in live the performed operation. Please remind that, if you change parameters from API, you need to refresh browser page in order to see parameters changing also in the GUI.

Set Configuration

In order to configure the i-Spector, the ***/set_config.cgi*** endpoint is available. This endpoint is a POST service which must be used performing a RAW POST operation, sending in the body of the POST the configuration JSON. It is not possible to use form-data or x-www-form-urlencoded because they are not supported by i-Spector.
There are two different possible configurations to be sent to i-Spector:

- High Voltage configuration
- MCA configuration

High Voltage Configuration

In order to configure the High Voltage, the following JSON format can be used in the body of the HTTP POST.

{"command" : "SET_CHANNEL_CONFIG","channel_config" :[{"id" : 0,"HV_STATUS" : true,"HV_VOLTAGE" : 41.5,"MaxV" : 46,"MaxI" : 5, "RAMP" : 20,"TCoeff" : -34,"HV_MODE" : "temperature","HV_PWRON" : true}],"store_flash" : false}

All parameters are case-sensitive.

PARAMETERS	VALID VALUES	FUNCTION
HV_STATUS	true/false	Enable/Disable HV
HV_VOLTAGE	22..80	HV voltage. Please pay attention to do not destroy the sensor setting too high voltage
MaxV	22..80	Max output voltage
MaxI	0..9	Trip Current in mA
RAMP	1..100	Ramp speed of HV
TCoeff	-1000 .. 1000	[mV/°C] Temperature compensation coefficient
MODE	“digital”/”temperature”	Enable / Disable temperature compensation on HV
HV_PWRON	true/false	Power on/off the HV on instrument boot

Table 2.2: table of the High Voltage configurable parameters.

MCA Configuration

In order to configure the MCA, the following JSON format can be used in the body of the HTTP POST.

{"command" : "SET_CHANNEL_CONFIG","mca_config" :[{"id" : 0,"trigger_thrs" : 28,"trigger_inib" : 300,"int_pre" : 300,"int_val" : 10,"int_gain" : 80,"pileup_inib" : 30,"pileup_pen" : 30,"baseline_inib" : 24,"baseline_len" : 256,"taget_run" : 0,"taget_value" : 0,"reset_on_apply" : true}],"store_flash" : false}}

All parameters are case-sensitive.

PARAMETERS	VALID VALUES	FUNCTION
trigger_thrs	10..1000 [int]	(LSB) Trigger threshold
trigger_inib	10..1000 [int]	(ns) Trigger inhibit after a trigger events. (set in in order to avoid double triggers)
int_pre	0..1000 [int]	(ns) Charge integrator pre-trigger integration extension
int_val	0..100 [float]	(us) Charge integrator integration time
int_gain	0..1000 [int]	Charge integrator GAIN
pileup_inib	0..100 [float]	(us) Pileup inhibition after a trigger
pileup_pen	0..100 [float]	(us) Pileup penalty if a pileup event occurs
baseline_inib	0..100 [float]	(us) Baseline inhibition after a trigger
baseline_len	1024,512,256, 128,64,32,16	Length is samples of the moving average used to calculate the baseline
taget_run	0,1,2	Acquisition run mode 0 – FREE 1 – TIME CONTRAINED (ms) 2 – TOTAL COUNTS ON SPECTRUM
taget_value	[int]	Referring to taget_run parameters, this field specify the run limit. For example, to run for 10 seconds set taget_run=1 and taget_value=10000
reset_on_apply	true/false	Reset spectrum when one or more configuration parameters are changed

Table 2.3: table of the MCA configurable parameters.

Get Instrument Status

In order to get the status of the i-Spector, it is possible to perform a HTTP GET to the ***/status.cgi*** endpoint. In the following, the structure returned by the endpoint.


```
{
  "command":"GET_SYSTEM_STATUS",
  "Result":"ok",
  "ErrorCode":0,
  "Reason":"",
  "current_status":{
    "system_status":{
      "temperature":0,
      "eth_status":0,
      "eth_ip":"192.168.50.2",
      "last_user_interact":-1,
      "power":"wall",
      "battery":false,
      "battery_life":0,
      "battery_charge":0,
      "battery_in_charge":false,
      "remaining_time":0,
      "battery_voltage":0,
      "battery_current":0,
      "battery_temperature":0,
      "alarm":0,
      "httpcloud":0,
      "loracloud":0
    },
    "channels":[
      {
        "id":0,
        "HV_STATUS":true,
        "HV_VOLTAGE":41.5,
        "HV_MODE":"temperature",
        "COMPL_V":false,
        "COMPL_I":false,
        "Vout":42.38652,
        "Vref":1.954313,
        "Iout":0.3540874,
        "IoutRAW":0.050250001,
        "Temp":50.199402,
        "SetPoint":42.356781,
        "ICR":1294,
        "OCR":1254,
        "runtime":4542,
        "livetime":4540,
        "sattime":0,
        "incnt":5856370,
        "outcnt":5646006,
        "live":0.969088,
        "dead":0.030912,
        "mca_running":1,
        "mca_status":0
      }
    ]
  }
}
```

Get Spectrum

In order to get the spectrum measured by the i-Spector, it is possible to perform a HTTP GET to the ***/spectrum.cgi*** endpoint. In the following, the structure returned by the endpoint.


```
{
  "command":"GET_SPECTRUM",
  "Result":"ok",
  "ErrorCode":0,
  "Reason":"",
  "data":[
    3,
    2,
    4,
    7,
    ..,
    1883
  ]
}
```


The data field in the JSON body contains the spectrum data in a 4096 bin array.

Get Waveform

In order to get the waveform measured by the i-Spector, it is possible to perform a HTTP GET to the ***/wavedump.cgi*** endpoint. In the following, the structure returned by the endpoint.


```
{
  "command":"GET_WAVEDUMP",
  "Result":"ok",
  "ErrorCode":0,
  "Reason":"",
  "data":[
    [
      3421,
      0,
      0,
      0,
      1,
      0,
      0
    ],
    [
      3425,
      0,
      0,
      0,
      1,
      0,
      0
    ],
    [
      3425,
      0,
      0,
      0,
      1,
      0,
      0
    ],
```

```
[
    3423,
     0,
     0,
     0,
     1,
     0,
     0
   ],
  ....
 ]
}
```


The data field in the JSON body contains the waveform data.

Each data element is and array of 7 elements:

- Analog data
- Trigger pulse
- Charge Integration Window
- PSD Tail integration Window
- Baseline restorer status
- Pile up rejector discard
- Pile up inhibition

Get MCA configuration

In order to get the status of the i-Spector MCA, it is possible to perform a HTTP GET to the **/get_mca_config.cgi** endpoint. In the following, the structure returned by the endpoint.


```
{
  "command":"GET_CHANNEL_CONFIGURATION",
  "Result":"ok",
  "ErrorCode":0,
  "Reason":"",
  "mca_config":[
    {
      "id":0,
      "trigger_thrs":28,
      "trigger_inib":300.000000,
      "int_pre":300.000000,
      "int_val":10.000000,
      "int_gain":80.000000,
      "pileup_inib":30.000000,
      "pileup_pen":30.000000,
      "baseline_inib":24.000000,
      "baseline_len":256,
      "rebinnig":4096,
      "reset_on_apply":true,
      "taget_run":0,
      "taget_value":0,
      "psd_gain":1.000000,
      "psd_delay":0.500000,
      "psd_int":0.800000,
      "scaleTimeWave":0
    }
  ]
}
```

3 Testing API

It is suggested to use CURL to POST/GET data. Below, some example of API usage.

- EXAMPLE: Send configuration for MCA

```
curl -d '{"command" : "SET_CHANNEL_CONFIG","mca_config" :[{"id" : 0,"trigger_thrs" : 100,"trigger_inib" : 100,"int_pre" : 300,"int_val" : 10,"int_gain" : 100,"pileup_inib" : 10,"pileup_pen" : 10,"baseline_inib" : 10,"rebinnig" : 4096,"baseline_len" : 512,"taget_run" : 0,"taget_value" : 0,"reset_on_apply" : true}],"store_flash" : false}}' -H "Content-Type: application/json" -X POST http://192.168.50.2/set_config.cgi
```

- EXAMPLE: Read system status

```
curl -X POST http://192.168.50.2/status.cgi
```

- EXAMPLE: Download spectrum

```
curl -X POST http://192.168.50.2/spectrum.cgi
```

4 Python SDK

A SDK for Python language is also available. The SDK uses HTTP API communication to interface with the i-Spector and requires Python >3.2. It works on x86/ia64/ARM processor. It could be used on a standard PC as well as on a Raspberry PI.

Required Modules

The SDK requires the following modules:

- requests [pip install requests]
- enum [pip install enum]

The Example file requires the following modules:

- pprint [pip install pprint]
- numpy [pip install numpy]
- matplotlib [pip install matplotlib]

Download the SDK

Python SDK files can be download from Nuclear Instruments Github:

https://github.com/NuclearInstruments/IspectorSDK-Python

or cloned with git:

```
git clone https://github.com/NuclearInstruments/IspectorSDK-Python.git
```

The SDK includes the library (inspector_sdk.py) and an example file (test_ispector_sdk.py)

Library Usage

In order to use the library, import ispector_sdk and open a connection creating a new ispector_sdk object:

```
from ispector_sdk import ispector_sdk
I1 = ispector_sdk("192.168.50.2")
```

The following functions are available:

- **set_hv_basic(self, hv_on, hv_voltage):**

Parameter	Type	Description
hv_on	bool	Enable/Disable HV
hv_voltage	float	HV voltage value (V)

Return: NONE

- **def set_hv_compensation(self, mode, temp_coeff):**

Parameter	Type	Description
mode	HVCompensation	DISABLE_COMPENSATION: no active temperature compensation ENABLE_COMPENSATION: active temperature compensation
temp_coeff	int	SiPM temperature compensation in mV/°C

Return: NONE

- **set_hv_cfg(self, ramp, maxI, maxV, on_starup):**

Parameter	Type	Description
ramp	int	[V/s] HV ramp speed
maxI	int	[mA] HV trip current
maxV	int	[V] Protection maximum voltage
on_starup	bool	Power on/off the HV on instrument boot

Return: NONE

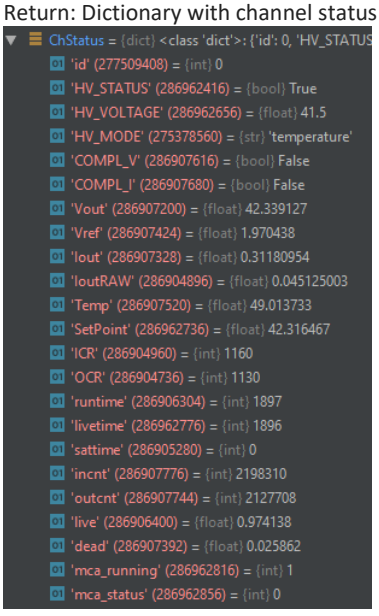
- **configureMCA(self, trigger_threshold, trigger_inibit, pre_int_time, int_time, int_gain, pileup_inib, pileup_penality, baseline_inib, baseline_len, target_run, target_value):**

Parameter	Type	Description
trigger_threshold	10..1000 [int]	(LSB) Trigger threshold
trigger_inibit	10..1000 [int]	(ns) Trigger inhibit after a trigger events. (set in in order to avoid double triggers)
pre_int_time	0..1000 [int]	(ns) Charge integrator pre-trigger integration extension
int_time	0..100 [float]	(us) Charge integrator integration time
int_gain	0..1000 [int]	Charge integrator GAIN
pileup_inib	0..100 [float]	(us) Pileup inhibition after a trigger
pileup_pen	0..100 [float]	(us) Pileup penalty if a pileup event occurs
baseline_inib	0..100 [float]	(us) Baseline inhibition after a trigger
baseline_len	[BaselineLength]	Length is samples of the moving average used to calculate the baseline
target_run	[RunMode]	Acquisition run mode 0 – FREE 1 – TIME CONTRAINED (ms) 2 – TOTAL COUNTS ON SPECTRUM
target_value	[int]	Referring to taget_run parameters, this field specify the run limit. For example to run for 10 seconds set taget_run=1 and taget_value=10000

Return: NONE

- **getChannelStatus(self):**

Return: Dictionary with channel status information, as shown below



In order to read a particular value using the getChannelStatus function:

```
I1 = ispector_sdk("192.168.50.2")

ChStatus = I1.getChannelStatus()
print(ChStatus["ICR"])
```

- **getSystemStatus(self):**

Return: Dictionary with system status

- **getWave(self):**

Return list of array of array containing the waveform information.
In order to extract a column of the matrix, it is possible to use a numpy matrix and select one column (column 0 contains analog values)

```
WaveMatrix = I1.getWave()

A = np.array(WaveMatrix)
wave_track = A[:,0]
```

Each data element is and array of 7 columns:

  - [0] Analog data
  - [1] Trigger pulse
  - [2] Charge Integration Window
  - [3] PSD Tail integration Window
  - [4] Baseline restorer status
  - [5] Pile up rejector discard
  - [6] Pile up inhibition

- **getSpectrum(self):**

Return: Array with 4096 spectrum bins

- **resetSpectrum(self):**

Return: NONE

- **def runSpectrum(self):**

Start spectrum acquisition
Return: NONE

- **def stopSpectrum(self):**

Stop spectrum acquisition
Return: NONE

5 C# SDK

A SDK for C# language is also available. The SDK uses HTTP API communication to interface with the i-Spector.

Required Modules

The SDK requires the following module:

- Newtonsoft JSON

It will be automatically downloaded from NuGet at compiling time.

The Example file requires the following modules:

- pprint [pip install pprint]
- numpy [pip install numpy]
- matplotlib [pip install matplotlib]

Download the SDK

C# SDK files can be download from Nuclear Instruments Github:

https://github.com/NuclearInstruments/IspectorSDK-CSHARP

or cloned with git:

```
git clone https://github.com/NuclearInstruments/IspectorSDK-CSHARP.git
```

The SDK includes a DLL library and a C# example. The DLL can be imported in any programming language supporting C# (.NET) DLL, including VB.NET, Labview, Matlab.

Library Usage

The function in C# SDK are the same as Python SDK. Refer to **Python SDK** for usage guide.

6 Technical Support

CAEN makes available the technical support of its specialists for request concerning the software and the hardware.
Use the support form available at the following link:

https://www.caen.it/support-services/support-form/

CAEN SpA is acknowledged as the only company in the world providing a complete range of High/Low Voltage Power Supply systems and Front-End/Data Acquisition modules which meet IEEE Standards for Nuclear and Particle Physics. Extensive Research and Development capabilities have allowed CAEN SpA to play an important, long term role in this field. Our activities have always been at the forefront of technology, thanks to years of intensive collaborations with the most important Research Centres of the world. Our products appeal to a wide range of customers including engineers, scientists and technical professionals who all trust them to help achieve their goals faster and more effectively.

CAEN S.p.A.
Via Vetraia, 11
55049 Viareggio
Italy
Tel. +39.0584.388.398
Fax +39.0584.388.959
info@caen.it
www.caen.it

CAEN GmbH
Klingenstraße 108
D-42651 Solingen
Germany
Tel. +49 (0)212 254 4077
Mobile +49 (0)151 16 548 484
Fax +49 (0)212 25 44079
info@caen-de.com
www.caen-de.com
CAEN GmbH

CAEN Technologies, Inc.
1140 Bay Street - Suite 2 C
Staten Island, NY 10305
USA
Tel. +1.718.981.0401
Fax +1.718.556.9185
info@caentechnologies.com
www.caentechnologies.com

CAEN
Tools for Discovery
Electronic Instrumentation

UM7047 - i-Spector Digital SDK User Manual Rev. 3 - July 12th, 2024
00000-00-S2570-MUTX