



Apply stimuli from your VHDL testbench with the GP-22050 pattern generator

This application note describes how to use a VHDL testbench to generate 'real-world' stimuli for the GP-22050 device set in pattern generation mode and apply them directly to a real system under test.

1 Introduction

Producing the adequate set of real-world stimuli to test a new microelectronic system can be very tricky, especially when not everything is available yet to implement what the future system environment will be. While it is generally advised to use 'real source' environment with a prototype – in order to best approach what the future system will be, not everything is always available to configure, trigger or stimulate custom interfaces – especially if partial system only is implemented in the prototype. In other words, there can be a need to generate essential logic vector at some inputs of a prototype system.

At the same time, a system test bench is very often available, with cycle-accurate models of the system stimuli. In order to improve the designer's productivity during prototype testing, it might come very handy to automatically generate real-world stimuli on some ports directly from the VHDL test bench. Using the GP-22050 to do so can also be much simpler than looking for a standard test/demo board and generate a synthesisable test bench in it that generate the stimuli. The GP-22050 really implements a link between your simulation environment and the system under test.

This application notes describes how to reuse VHDL test vectors and apply them directly onto a prototype system with the GP-22050 used as a pattern generator.

2 General flow

Figure 1 sketches the overall flow.

The 'GP-22050 VHDL to stimuli generator' is inserted in the available VHDL testbench as a specific VHDL instance dedicated to GP-22050 stimulus generation. This instance is connected to a set of the input lines that stimulate the system under test (SUT).

A system simulation is run within the chosen simulation environment. The 'GP-22050 VHDL to stimuli generator' module automatically samples the simulation input stimuli and automatically generates a set of input files, formatted according to the GP-22050 source files formats.

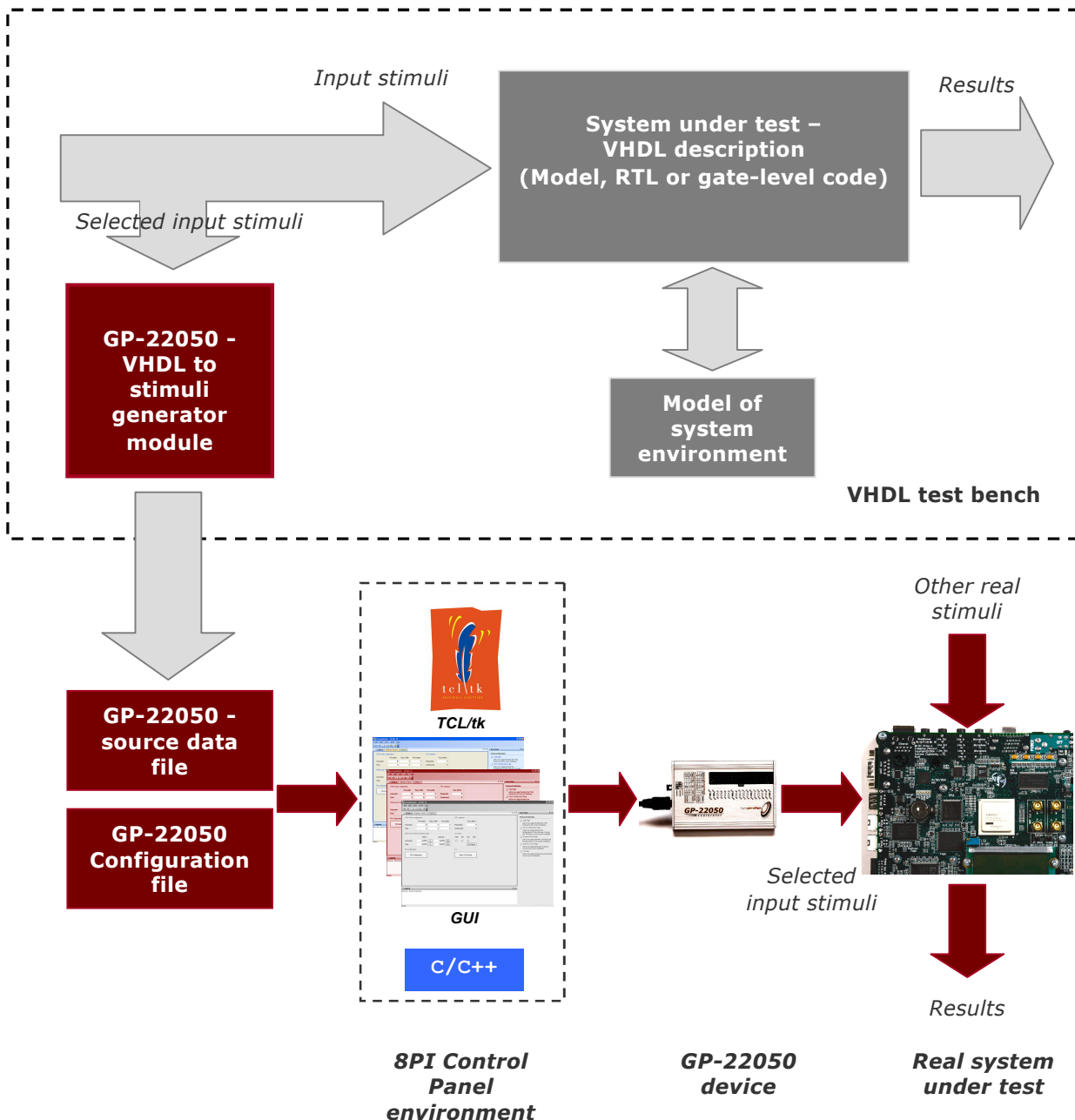
A source data file and a GP-22050 configuration file are provided to the GP-22050 device with the 8PI Control Panel host software environment, using one of the provided software interfaces. As described in the rest of this document, this process can be highly automated by means of the TCL/tk interface, for instance. There is no obligation, though, to choose this interface, for the 8PI Control Panel source data file and configuration file can be used with the GUI or the C/C++ interfaces as well.

The GP-22050 device is set in ADWG mode (or pattern generator mode). With this mode, cycle-accurate digital stimuli can be applied to a 'real' SUT, such as a system implemented in a board with FPGA.

This flow helps the design engineer start early with the system test phase on prototype. It is especially useful when it is hard to fully modelize a system from a test bench environment and/or many design constraints make simulation long and almost impracticable. In this case, it is often preferred to go on prototype, using the real future system environment as a source for the stimuli. Even in this case, some other stimuli sources may be missing, because not everything in the system is designed yet.

The GP-22050 device and its programming interface offer a flexible yet powerful solution to complement the prototyping environment.

Figure 1: VHDL to GP-22050 stimuli – General flow





3 Go step-by-step

3.1 Instantiate the VHDL to stimuli generator

3.1.1 Getting the source file

Access Byte Paradigm download page www.byteparadigm.com/Download.html and download archive 'VHDL2Stim.zip'. Unpack the archive in your work directory. It contains the file VHDL2Stim.vhd.

3.1.2 VHDL to stimuli generator description

3.1.2.1 Principle

This module basically samples the incoming data with the connected input clock and generates an output data file with the data formatted as hexadecimal numbers and a configuration file. The data are stored into the file 'VHDL2Stim.dat' which is referenced from the GP-22050 configuration file 'VHDL2Stim.cfg'.

3.1.2.2 Libraries

The following libraries are used by the modules included in the VHDL2Stim.vhd file.

Table 1 : Used libraries

Library name	Comment
std_logic_1164	Standard IEEE library – not provided
std_logic_arith	Standard IEEE library – not provided
textio	Standard STD library – not provided
txt_util	Additional library – provided – must be compiled in the WORK library

3.1.2.3 VHDL2Stim.vhd

3.1.2.3.1 VHDL2StimCtrlData entity declaration

```
entity VHDL2StimCtrlData is
generic(
    RiseNFallEdge      : boolean;
    CtrlSeqLength      : integer range 0 to 250; -- Length of the ctrl sequence
    ClockFreq          : natural;              -- Clock frequency in Hz
    NrCtrlLines        : integer range 1 to 6;
    NrDataLines        : integer range 1 to 16;
    DataFileName       : string := "VHDL2Stim.dat";
    ConfigFileName     : string := "VHDL2Stim.cfg"
);
port (
    RecordEn           : in std_logic;
    SampleClock        : in std_logic;
    CtrlVector         : in std_logic_vector(NrCtrlLines - 1 downto 0);
    DataVector         : in std_logic_vector(NrDataLines - 1 downto 0)
);
end entity;
```



Generic list:

- **RiseNFallEdge** : boolean setting on which edge of the SampleClock input the DataVector input must be sampled. 'True' for rising edge; 'False' for falling edge.
- **CtrlSeqLength** : integer value specifying the length of the sequence length in number of clock (SampleClock) cycles.
- **ClockFreq** : natural specifying the frequency of the GP-22050 clock in Hz.
- **NrCtrlLines** : integer specifying the number of used control lines. Valid values range from 1 to 6. Control lines (according to the GP-22050 definition of 'control line') will be mapped onto the selected number of LSBs on the GP-22050 connector.
- **NrDataLines** : integer specifying the number of used data lines. Valid values range from 1 to 16. The input stimuli will be mapped onto the selected number of LSBs on the GP-22050 connector.
- **DataFileName** : string specifying the name of the data file – default value: "VHDL2Stim.dat".
- **ConfigFileName** : string specifying the name of the configuration file – default value: "VHDL2Stim.cfg".

Port list:

- **RecordEn** : when set to '1', enables the input stimuli recording. Synchronous with the SampleClock input clock.
- **SampleClock** : module clock input; this clock will be used to sample the input stimuli.
- **CtrlVector** : std_logic vector connected to the selected input control lines from the testbench. The vector length is specified with the NrCtrlLines generic parameter.
- **DataVector** : std_logic vector connected to the selected input stimuli from the test bench. The vector length is specified with the NrDataLines generic parameter.

3.1.2.3.2 VHDL2StimCtrlData operation

This module can be used with or without control sequence – according to the definition of the 'GP-22050 control sequence'. In case a control sequence is used, a repetitive pattern is played onto the control lines. During the execution of the repetitive pattern, the output data lines are held at a constant value. For instance, this can be the case when accessing a memory interface, where write and enable control must be generated for each data. Once programmed, a control sequence is entirely managed by the GP-22050 and does not require any USB bandwidth to be sent over and over to the system-under-test. This is why it is often preferred to use the control lines instead of the data lines for repetitive data /patterns (please refer to [1] and [2] for more information about the GP-22050 control sequences).

Figure 2 shows an example where a control sequence of 4 clock cycles is executed for each data sample. The 'VHDL2StimCtrlData' module records the control sequence once and samples the data every 4 clock cycles.

The data are stored into the file specified with the 'DataFileName' generic.

The control sequence is used to generate the configuration file specified with the 'ConfigFileName' generic. Figure 4 shows an example of such a configuration file.

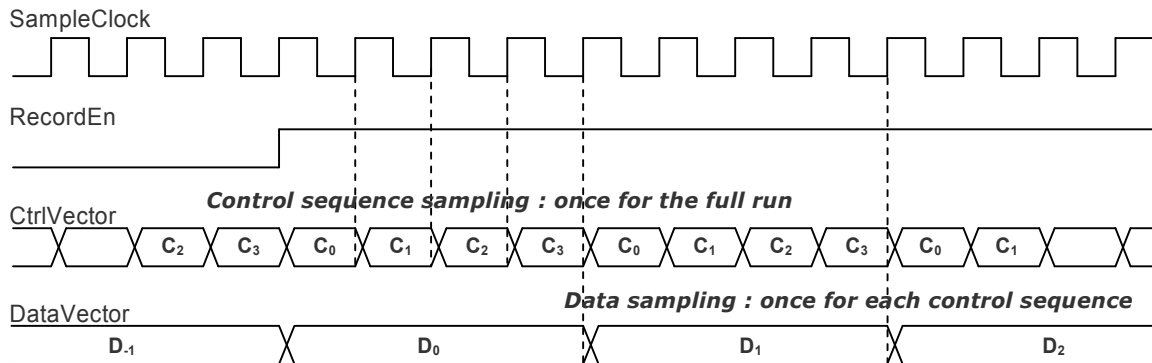
In the provided version of VHDL2Stim.vhd, most configuration options are set to default (e.g. the clock output is enabled by default). The parameters that can be configured with the VHDL2StimCtrlData module generics are summarised in

Table 2.



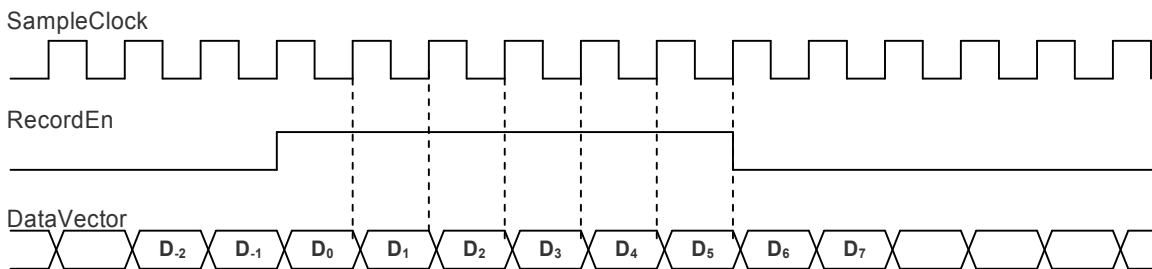
To apply a constant value onto the control lines set the generic parameter 'CtrlSeqLength' to 1.

Figure 2: VHDL2StimCtrlData module operation example



To operate **without** control sequence, the generic parameter 'CtrlSeqLength' must be set to 0. In this case, there is no need to connect the 'CtrlVector' port when creating the module instance.

Figure 3: VHDL2StimCtrlData module operation example without ctrl sequence



RiseNFallEdge = true → SampleClock rising edge is used.
D₀ to D₅ will be stored into the output file.

Table 2 : VHDL2StimCtrlData module generics and configuration file

Parameter	Generic	Configuration file setting
Clock frequency	ClockFreq	@freq <ClockFreq>
Length of the control sequence	CtrlSeqLength	<Sets the '@ctrl' commands in the control section of the configuration file. If set to 0, no control section will be inserted into the configuration file>.
Active control lines	NrCtrlLines	<Specifies the '@header' of the configuration file control section>.
Active data lines	NrDataLines	<Specifies the '@header' of the configuration file data section>.

During a simulation run, this module performs the following tasks:

- it samples the input 'DataVector' at the selected 'SampleClock' input edge if the input 'RecordEn' is active AND once for each control sequence. For example, if the control sequence is 4 clock cycles long, data will be sampled every 4 clock cycles, at the last cycle of the control sequence. Data is stored as hexadecimal strings into the file specified with the



- 'DataFileName' generic string. When the control sequence length is set to 0, the 'DataVector' is sampled at each of the selected clock cycle.
- it samples the input 'CtrlVector' one time for the full simulation run. For that purpose, it uses the selected clock input edge, the control sequence length entered as input generic parameter and checks if the 'RecordEn' is active. This control sequence will be automatically inserted in the configuration file, in the control section ('@ctrl' keyword).
 - it creates the 2 following files:
 - VHDL2Stim.cfg : configuration file to be used as input file for the GP-22050 device with the 8PI Control Panel software.
 - VHDL2Stim.dat : this file holds the data samples only, coded as hexadecimal strings; this file is called by the configuration file.

Figure 4 : Example of configuration file generated with the 'VHDL2StimCtrlData' module

```
# ADWG config file - generated from VHDL2Stim
@transfer optimised true
@transfer continuous true
@transfer unsupervised false
@transfer infinite false
@freq 50000000
@clock source internal
@clock pll enable
@clock output enable
@clock outputratio 1
@clock type continuous
@clock polarity positive
@trigger type internal
@ctrl_begin
# pin      543210
  @header  --0000
  @ctrl    001111
  @ctrl    001110
  @ctrl    001101
  @ctrl    001100
@ctrl_end
@data_begin
# pin      FEDCBA9876543210
  @header  -----000000000000
  @file txt_rawhex "VHDL2Stim.dat"
@data_end
```

3.1.3 Insert modules instances in the test bench

Figure 5 shows an instance example of the VHDL2StimCtrlData module.

In this example, the generic are passed with constants defined in the testbench. Please refer to the source code for data types.

Remark: if the generic parameter 'CtrlSeqLength' is set to 0, the 'CtrlVector' does not need to be connected.

Figure 5 : Example of VHDL2StimCtrlData module instance

```
i01_VHDL2StimCtrlData : entity work.VHDL2StimCtrlData (GenStim)
generic map (
    RiseNFallEdge      => c_SamplingEdge,
    CtrlSeqLength      => c_CtrlSeqLength,
    ClockFreq          => c_ClockFreqHz,
    NrCtrlLine         => c_NrCtrl,
    NrDataLines        => c_NrData,
)
port map (
    RecordEn          => RecordEn,
    SampleClock       => MainClock,
    CtrlVector        => CtrlVectorStd,
    DataVector        => DataVectorStd
);
```

Figure 6 : Example of VHDL2StimCtrlData module instance without control sequence

```
constant c_CtrlSeqLength : integer := 0;

<...>

i01_VHDL2StimCtrlData : entity work.VHDL2StimCtrlData (GenStim)
generic map (
    RiseNFallEdge      => c_SamplingEdge,
    CtrlSeqLength      => c_CtrlSeqLength,
    ClockFreq          => c_ClockFreqHz,
    NrDataLines        => c_NrData,
)
port map (
    RecordEn          => RecordEn,
    SampleClock       => MainClock,
    DataVector        => DataVectorStd
);
```

3.2 Use the files with the 8PI Control Panel

3.2.1 8PI Control Panel source files summary

In the framework of this application note, the 8PI Control Panel always works with 2 source files:

- A configuration file: this file holds the GP-22050 device settings, the description of the control sequence and the source data file that must be sent with the pattern generator.
- A source data file containing the data samples to be sent on the GP-22050 data lines.

3.2.2 Use the files with the 8PI Control Panel GUI

Start the 8PI Control Panel GUI.

Click on the 'Load' button in the 'Configuration group' of the 'Arbitrary Wave sheet', 'Configuration tab' (Figure 7). A dialog box opens, asking to select the configuration file. Select **VHDL2Stim.cfg** (or another one if you generated the configuration file under a different file name).

The GP-22050 has now loaded its configuration.



To run the recorded pattern set, switch to the operation tab, select 'file mode' and click on the 'start' button (Figure 8).

Figure 7 : Load configuration file with the 8PI Control Panel GUI

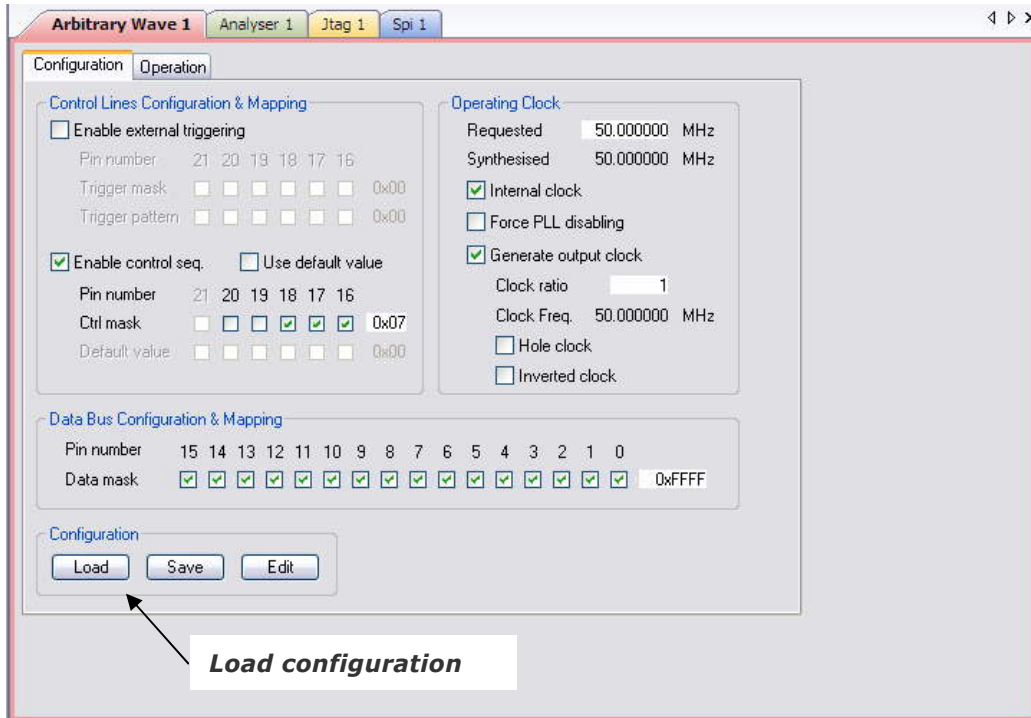
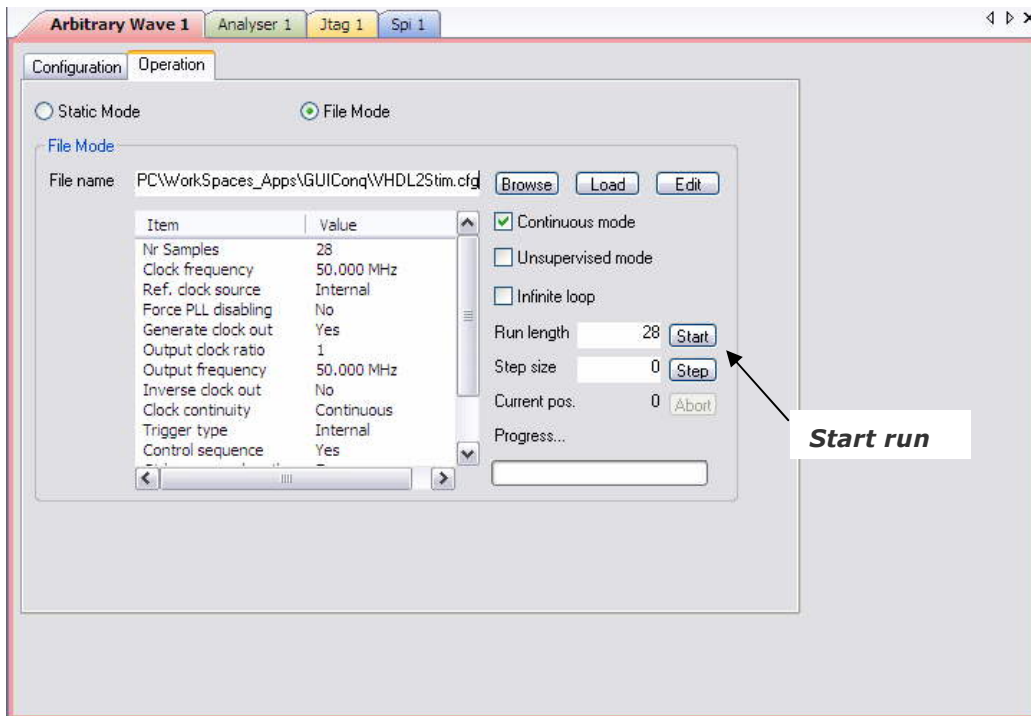


Figure 8 : Run the pattern generator with the 8PI Control Panel GUI





3.2.3 Use the files with 8PI Control Panel TCL/tk interface

Start a TCL/tk shell.

Execute the script 'VHDL2StimRun.tcl' :

```
# Load the ADWG TCL library and start the smart router
source ADWGTclLib.tcl
# Load the configuration file generated with VHDL2Stim
ReadConfAndDataFile VHDL2Stim.cfg
# Extract the total number of data samples
set NSamples [GetNrSamples]
# Run NSamples in ADWG mode
AdwgRun $NSamples
```

4 References

- [1] 8PI Control Panel user's guide : ug_GP22050_8PIControlPanel.pdf
- [2] GP-22050 data sheet : ds_GP22050.pdf

All references are available from: www.byteparadigm.com/Documentation.html.