

FREQUENCY SYNTHESIZER

LNO-6xM

Operating Manual

Rev. 1.2

Advantex LLC

March 4, 2016

Russian Federation, 111250, Moscow,
Krasnokazarmennaya st., 13/1
tel. +7 (495) 721-47-74, +7(495) 728-08-03
info@advantex.ru
<http://advantex-rf.com>, www.advantex.ru



Document Revisions

Rev.	Date	Description
1.0	February 3, 2015	LNO Frequency Synthesizer Manual (starting from LNO-62M-RF)
1.1	March 31, 2015	Revised section 5.3
1.2	March 04, 2016	Revised table 1 (power consumption)

Contents

1	Getting Started	7
1.1	Interfaces and Connectors	7
1.2	LNO-6xM-KIT and LNO-SS6xM-KIT Evaluation Boards	9
1.3	RFCTL Evaluation Board	14
2	Remote Control	16
2.1	Quick Start	17
2.2	SCPI Commands	17
2.2.1	SCPI Compliance	17
2.2.2	SCPI Summary	19
2.3	SCPI Command List	20
2.3.1	*CLS	20
2.3.2	*IDN?	20
2.3.3	*RST	21
2.3.4	*OPC?	21
2.3.5	SYSTem:ERRor[:NEXT]?	21
2.3.6	OUTPut[:STATe]	21
2.3.7	OUTPut:ROSCillator[:STATe]	22
2.3.8	[SOURce]:FREQUency[:CW]	22
2.3.9	[SOURce]:POWer[:LEVel][:IMMediate][:AMPLitude]	22
2.3.10	[SOURce]:PHASe[:ADJust]	23
2.3.11	[SOURce]:ROSCillator:SOURce	23
2.3.12	[SOURce]:ROSCillator:EXTernal:FREQUency	23
2.3.13	MEASure[:SCALAR]:TEMPerature?	24
2.3.14	STATus:QUESTionable:CONDition?	24
2.3.15	STATus:QUESTionable[:EVENT]?	24
3	Evaluation Board Firmware Update	25
4	LNO-6xM SPI Registers and Commands	28
4.1	General Description	28
4.2	Format of SPI Commands	30
4.3	SPI Commands	31
4.3.1	Func register writing C[7:0]=0x01	31

4.3.2	Func register reading C[7:0]=0x81	31
4.3.3	Divider buffer register writing C[7:0]=0x02	31
4.3.4	Divider buffer register reading C[7:0]=0x82	31
4.3.5	Gain buffer register writing C[7:0]=0x03	31
4.3.6	Gain buffer register reading C[7:0]=0x83	34
4.3.7	DigPotWP register writing C[7:0]=0x04	34
4.3.8	DigPotWP register reading C[7:0]=0x84	34
4.3.9	MuxOut register writing C[7:0]=0x05	34
4.3.10	MuxOut register reading C[7:0]=0x85	36
4.3.11	DDS_SPI Access C[7:0]=0x10	36
4.3.12	Toggling DDS_IOUPDATE C[7:0]=0x11	36
4.3.13	Divider register updating C[7:0]=1x12	36
4.3.14	Gain register updating C[7:0]=1x13	36
4.3.15	All register updating and DDS_IOUPDATE toggling C[7:0]=1x1F	36
4.3.16	TEMP_SPI Access C[7:0]=0x30	37
4.3.17	ADC_SPI Access C[7:0]=0x40	37
4.3.18	DIGPOT_SPI Access C[7:0]=0x50	37
4.3.19	FLASH_SPI Access C[7:0]=0x70	37
5	LNO-6xM SPI Programming	37
5.1	Parameter Calculations	37
5.2	Initialization	40
5.3	Setting Frequency, Phase and Level	41
5.4	Reading internal temperature	42
5.5	Flash Memory	43
5.5.1	FLASH_CMD_READ (0x03)	43
5.5.2	FLASH_CMD_WRITE (0x02)	44
5.5.3	FLASH_CMD_WREN (0x06)	44
5.5.4	FLASH_CMD_WRDI (0x04)	45
5.5.5	FLASH_CMD_RDSR (0x05)	45
5.5.6	FLASH_CMD_WRSR (0x01)	46
5.5.7	FLASH_CMD_PE (0x42)	46
5.5.8	FLASH_CMD_SE (0xD8)	46
5.5.9	FLASH_CMD_CE (0xC7)	46
5.5.10	FLASH_CMD_RDID (0xAB)	46
5.5.11	FLASH_CMD_PDP (0xB9)	47
5.6	Memory address and data mapping	47
5.6.1	Configuration Block	50
5.6.2	Data block	51

List of Figures

1	RF Out Connector	7
2	REF In and SPI Connectors	7
3	SPI Control and Power Supply interface	8

4	RS2SPI Evaluation Board	10
5	COM-port number	11
6	Select SPI-connector number	11
7	COM-port configuration	12
8	LNO tab	13
9	LNO reference frequency source control	14
10	Drain current setting of output amplifier	15
11	RFCTL Evaluation Board	16
12	Example of the remote control using <i>HyperTerminal</i> application	18
13	COM-port settings	18
14	Instrument control and command processing model	19
15	Command structure	20
16	<i>Device Manager</i> window (My Computer▷Manage)	26
17	Firmware update application – XMI Programmer	27
18	LNO synthesizer block diagram	29
19	LNO SPI writing cycle diagram	30
20	Command and data bytes	30
21	LNO SPI reading cycle diagram	30
22	Writing 1 data byte	31
23	Input and output data for LNO programming	38
24	Bilinear interpolation	40
25	Structure of flash-memory data transfer	43
26	Flash-memory data read command	43
27	Flash-memory data write command	44
28	Flash-memory write enable command	44
29	Flash-memory write/erase disable command	45
30	Flash-memory status register reading	45
31	Flash-memory write to status register	46
32	Flash-memory page erase command	46
33	Flash-memory byte erase command	47
34	Flash-memory clear all command	47
35	Flash-memory power on and read ID command	47
36	Flash-memory power off command	51

List of Tables

1	SPI Control and Power Supply pinout	8
2	LNO SPI Commands C[7:0]	32
3	Func register	33
4	Divider register	33
5	Gain register	34
6	DigPotWP register	34
7	MuxOut register	35
8	LNO input parameters	39
9	Intermediate variables and output parameters	39

10	Turning on the internal power supply system at initialization process	41
11	Turning on the DDS power supply at initialization process	41
12	Flash-memory commands	43
13	Flash-memory status register	45
14	Flash-memory block protection	46
15	Memory address and data mapping	47
16	Type of Data Table (CTYPE)	51
17	X,Y and Z-data format (XVALUE, YVALUE, ZVALUE)	52

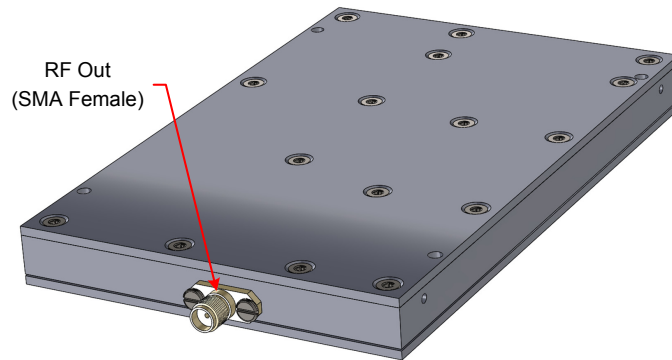


Figure 1: RF Out Connector

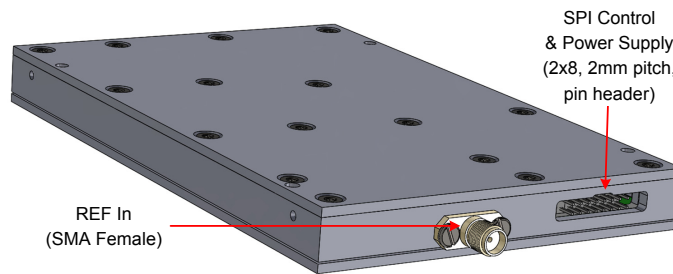


Figure 2: REF In and SPI Connectors

1 Getting Started

1.1 Interfaces and Connectors

Figures 1, 2, 3 show external interfaces and connectors of LNO-6xM-RF synthesizer. The module has the following connectors:

RF Out – RF signal output, 100 MHz to 12 GHz frequency range with less than 0.001 Hz step, -14 to +15 dBm level range with 0.5 dB step, connector type – SMA, female;

REF In – input of required external reference frequency signal, rated level 0 dBm. Signal of any frequency can be applied in 100 to 200 MHz range;

SPI Control and Power Supply – SPI interface designed to control LNO module, LVTTTL 3.3V levels, max. clock rate is 10 MHz. Connector type: 2 row, 2 mm pitch, 16-pin holder.

Table 1 shows pinout of SPI Control and Power Supply interface.

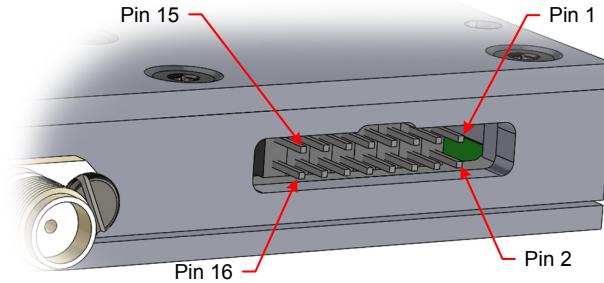


Figure 3: SPI Control and Power Supply interface

Table 1: SPI Control and Power Supply pinout

Pin #	Name	Direction (relative to module)	Description
1	LNO_MOSI	In	SPI master output, slave input
3	LNO_SS#	In	SPI select signal, data are latched only if this signal is active ("0" state)
5	LNO_SCK	In	SPI clock signal, data are latched by rising edge of LNO_SCK signal
7	LNO_MISO	Out	SPI master input, slave output
8	LNO_AUX	Out	LNO_AUX signal can be programmed as PLL lock status output, fixed low or high state output, and as the output of internal update signal of the loaded data (see section 4.3.9 for more details). This pin is connected to CPLD via 1 kOhm resistor, so if not used you can tie it to the GND
2, 10, 12, 14, 16	GND	-	Ground pins, internally connected to the case body
4, 6	-	-	AC ground. Internally connected via 1 uF capacitors to GND
9, 11	+5V	In	Positive power supply, +5.0 to +5.5V allowed. Rated current 0.18A
13, 15	+9V	In	Positive power supply, +9 to +12V allowed, but to reduce overheating and power consumption it's better not to exceed +10V. Rated current 0.78A

1.2 LNO-6xM-KIT and LNO-SS6xM-KIT Evaluation Boards

LNO-6xM-KIT includes LNO Frequency Synthesizer and RS2SPI evaluation board (fig. 4). LNO-SS6xM-KIT also includes DSG-3xM-RF frequency synthesizer as the reference frequency source which is used for additional spur suppression and enhanced synchronization capability (see section 1.3). It is designed for quick start and helps to understand SPI protocol command set and algorithms used to control LNO synthesizer. All SPI commands which sent to the module are displayed in log window of the application. So you can easily check yourself while debugging of your own control system of the synthesizer. For remote control of LNO (LNO+DSG) module follow the steps below.

1. Connect LNO module to the RS2SPI board (*Connector 1* located on the top side of the board, see fig. 4) via SPI cable (16-pin, 2-mm pitch, IDC flat cable).
2. Connect DSG module to the RS2SPI board (*Connector 2* located on the bottom side of the board) via SPI cable (16-pin, 2-mm pitch, IDC flat cable).
3. Connect RS2SPI board to PC via USB or RS-232 cable (see jumper positions for USB/RS-232 use on fig. 4)
4. Connect RS2SPI board to **+12 VDC** power supply. Max current of the power supply source should be set to **3A** for the DC-DC converters of RS2SPI board to start. Turn on the power. Power supply consumption will be about 0.1A (for non-initialized LNO and DSG modules).
5. When using USB cable find out the COM-port number in **Computer Management** window (see fig. 5) You can specify another COM-port number (**Properties**▷**Port Settings**▷**Advanced**▷**COM Port Number**).
6. Launch *RF Debug Application* (double click on `advantex.exe` or `advantex.tcl` if you have Tcl/Tk installed).
7. Select radio-boxes as shown on the fig. 6 (specify the connector of the RS2SPI board to which LNO is connected) and click OK.
8. Select menu **Setup**▷**COM_configure** and specify COM-port number (fig. 7) as it was determined in fig. 5. Click OK.
9. To start working with LNO-module press **Init** button (fig. 8, Item 1), then you can load frequency or output level without **Init** button. To load frequency you need to set value in MHz (Item 2) and press **Load** button (Item 3). Output level adjustment is implemented by loading a gain value (Item 4) which is in range 0 to 31.5 dB with 0.5 dB step. It corresponds to the digital attenuator setting, where 0 - maximum attenuation, 31.5 - minimum attenuation. You can read the internal temperature of the block and the output stage drain current by pressing **Read** button (Item

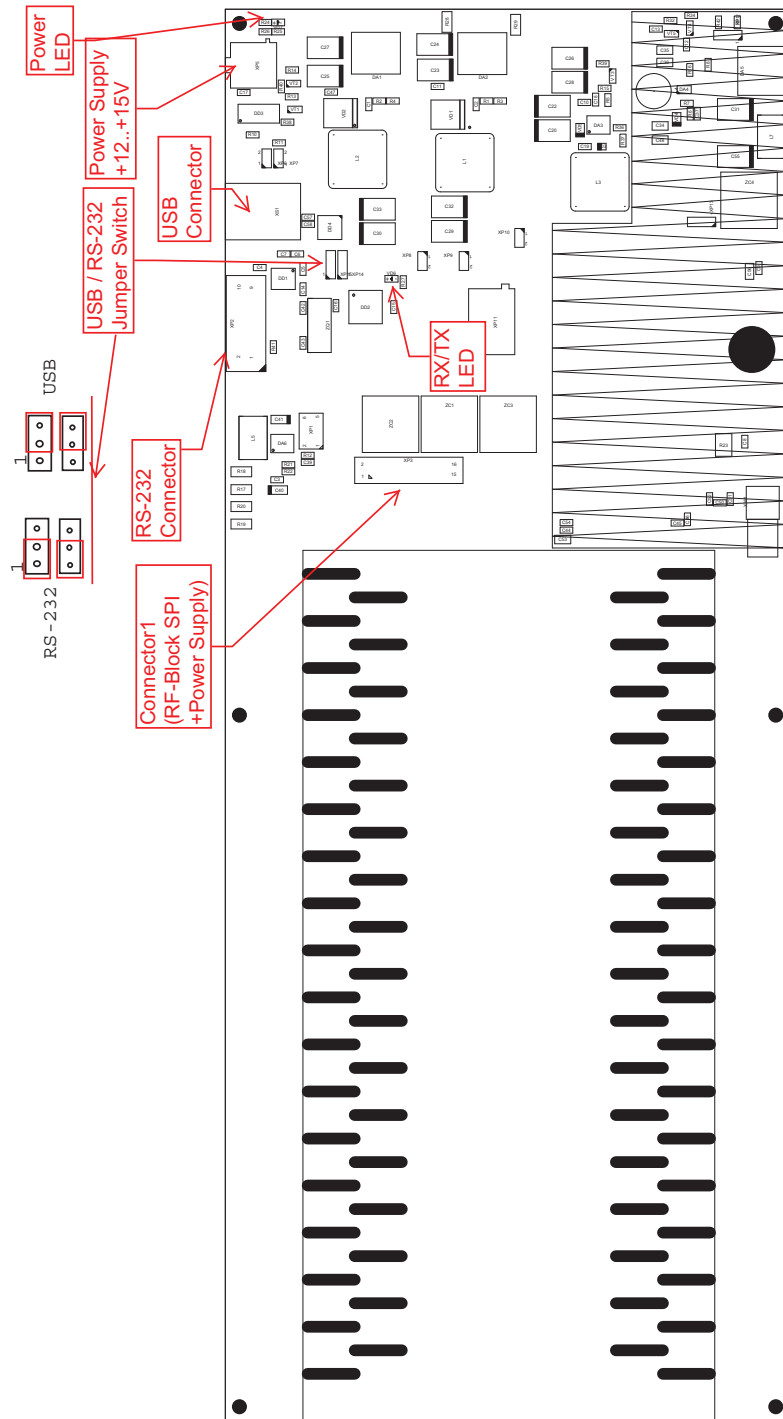


Figure 4: RS2SPI Evaluation Board

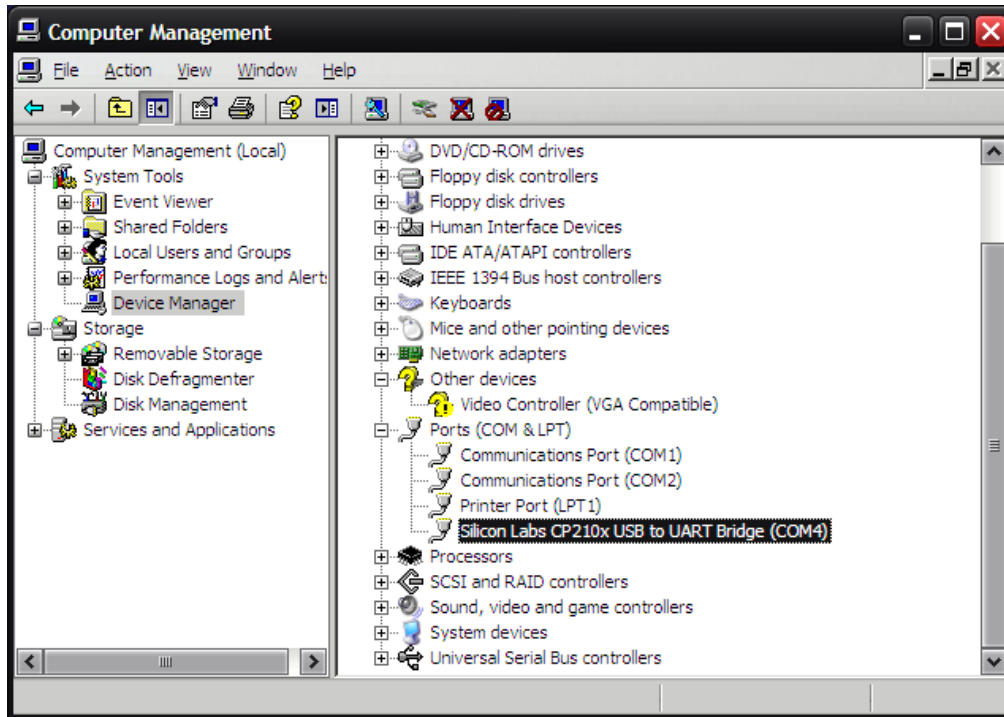


Figure 5: COM-port number



Figure 6: Select SPI-connector number

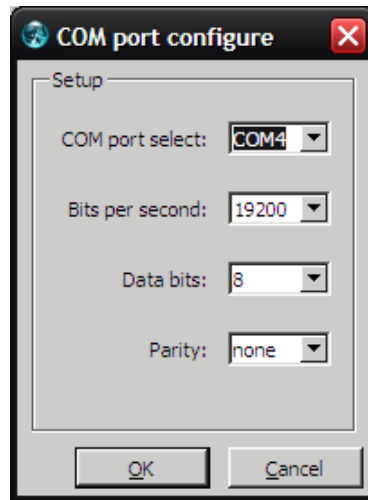



Figure 7: COM-port configuration

6). For LNO+DSG configuration (LNO-SS6xM-KIT) set the LNO-62 & DSG-03 couple checkbox and press Init button. After pressing Init button power consumption will increase up to 0.8A for LNO-6xM-KIT, and 1.2A for LNO-SS6xM-KIT (LNO+DSG configuration).

10. In LNO+DSG configuration the DSG synthesizer is used as the reference frequency source for the high frequency LNO synthesizer. Choosing the right LNO reference frequency value defines the quality of LNO RF Out signal (in terms of spurs). You can set this frequency manually by setting the Enable manual reference checkbox or automatically (the checkbox is unset). In this case new reference frequency is calculated each time you press Load Freq button using the frequency table located in `_export_spur_list.dat` data file. If you need to use DSG external reference, please check the box Enable DSG external reference (fig. 9), set the DSG reference frequency value in range 1 to 250 MHz and press Init button (fig. 8, Item 1). Each change of any checkbox setting requires the pressing of Init button to load new settings to the hardware.

11. The POT tab described below (fig. 10) is used only in the factory adjustment of the drain current of output amplifier. After adjustment the setting is stored in FLASH memory of digital potentiometer and is applied automatically after power on of the LNO module. To adjust drain current follow these steps.

 Dig POT tab settings are not used in normal operation.

- (a) Load the Digital Potentiometer DAC value (in range 0 to 1023) using Dig POT field (Item 1) and Write(DAC) button (Item 2). 0 corresponds to the maximum current, 1023 corresponds to the minimum

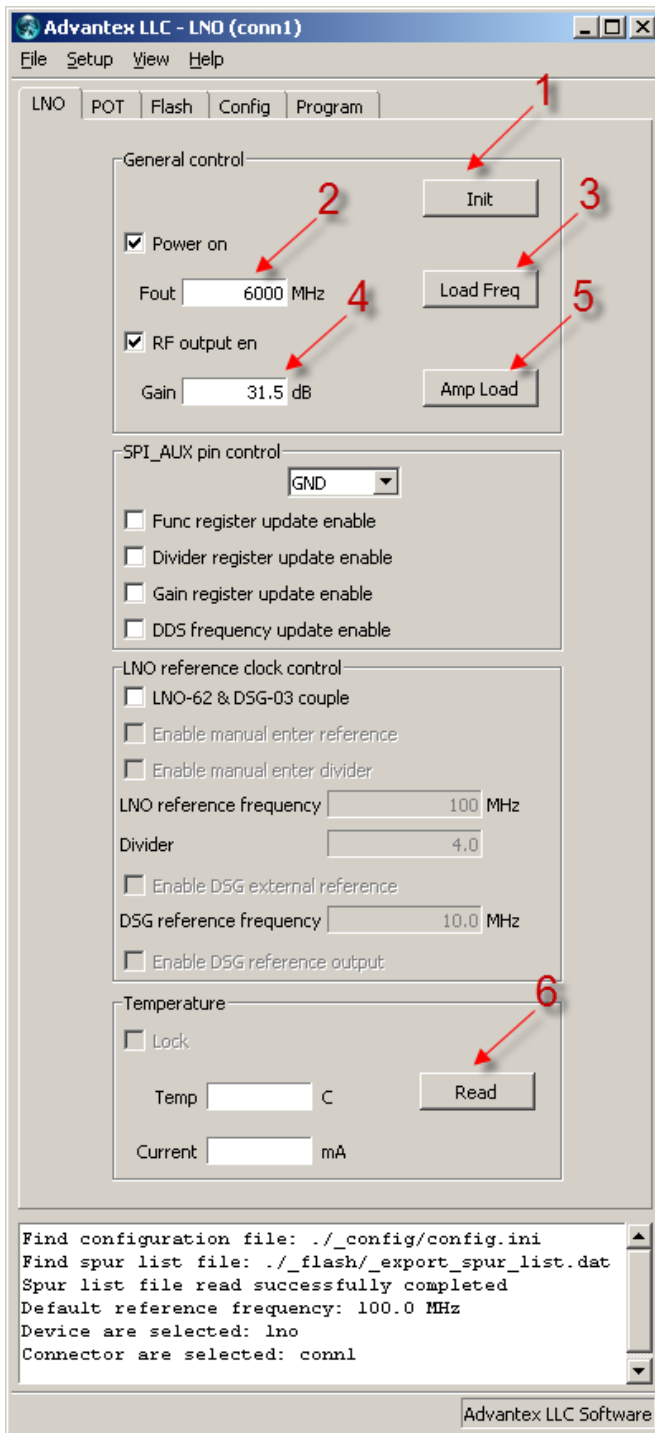


Figure 8: LNO tab

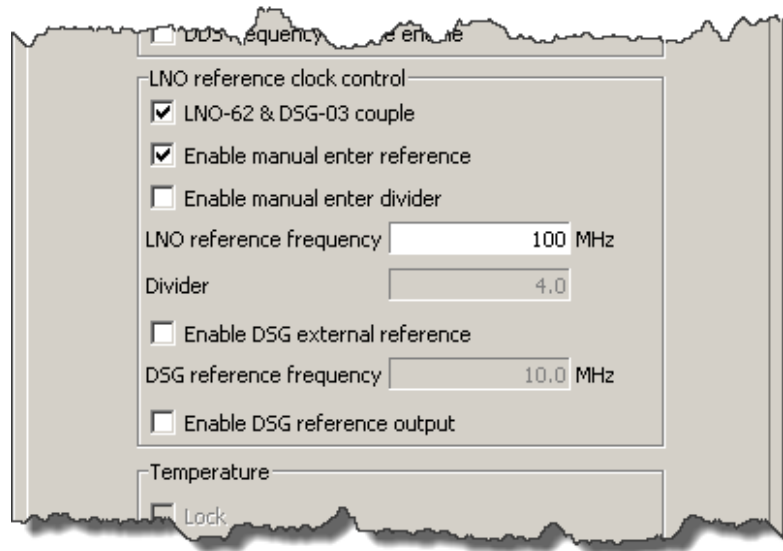


Figure 9: LNO reference frequency source control

current. For not to exceed maximum current ratings, please start with 1023.

- (b) Read the drain current value by pressing Read(curr) button (Item 4).
- (c) Repeat these steps until the current is set about 75 mA.
- (d) Write the final value to the digital potentiometer FLASH by pressing Store(FLASH) button.

1.3 RFCTL Evaluation Board

RFCTL-1xM-PCB evaluation board can be used in two ways: (a) as the RS2SPI replacement or (b) as the RF-module control PCB. In this case RF-modules (LNO and DSG) are controlled using SCPI command set via USB, RS-232 or UART interfaces. All calculations are performed by MCU of RFCTL board. Particular implementation depends on the firmware loaded to MCU located on the RFCTL board.

For remote control of LNO (LNO+DSG) module follow the steps below.

1. Connect LNO module to the RFCTL board to SPI3 (*Connector 1* in terms of RS2SPI, see fig. 11) via SPI cable (16-pin, 2-mm pitch, IDC flat cable).
2. Connect DSG module to the RFCTL board to SPI2 (*Connector 2* in terms of RS2SPI) via SPI cable (16-pin, 2-mm pitch, IDC flat cable).

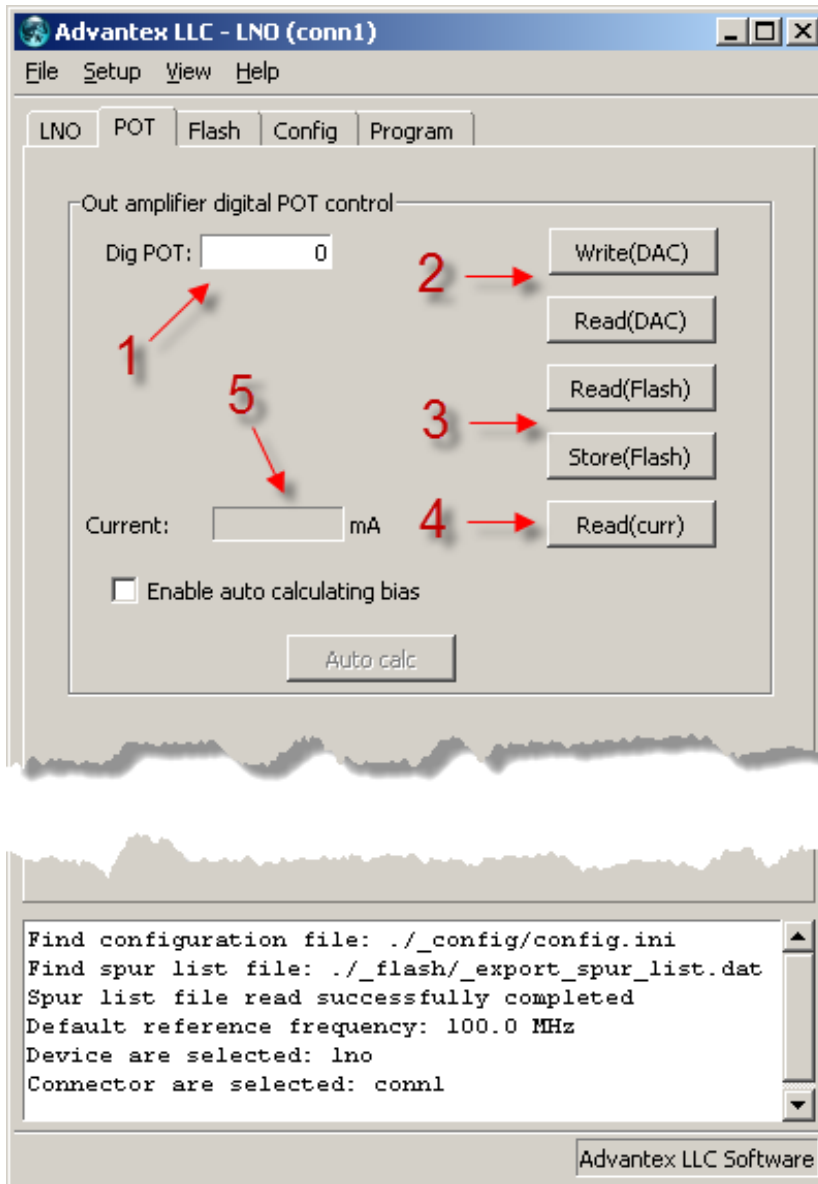


Figure 10: Drain current setting of output amplifier

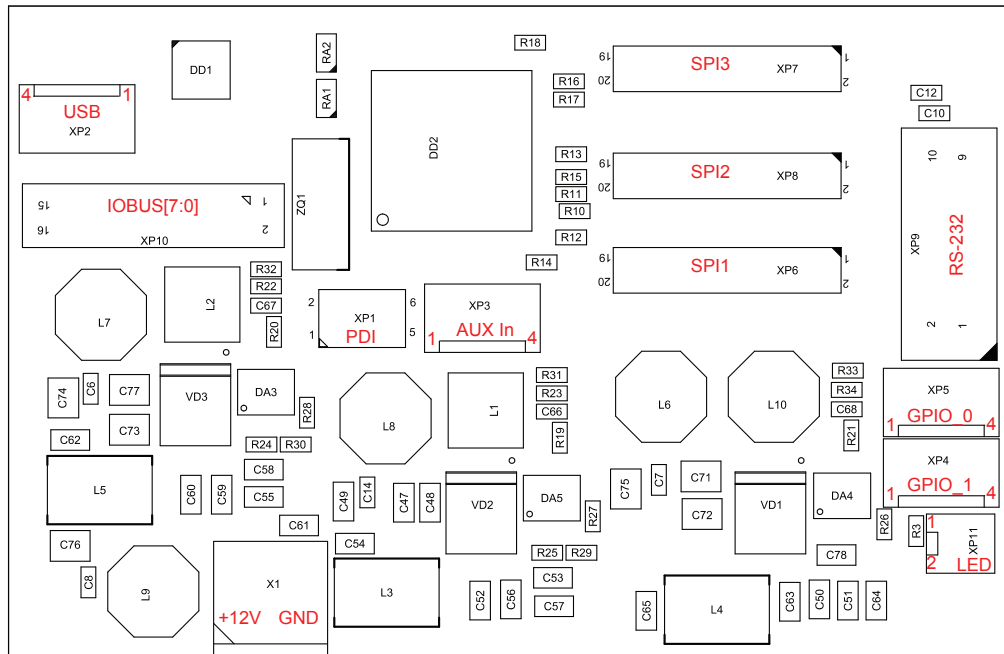


Figure 11: RFCTL Evaluation Board

3. Connect RFCTL board to PC via USB (*XP2* Connector: 1: VBUS, 2: D-, 3: D+, 4:GND) or RS-232 cable (*XP9* connector, 10-pin, 2.54-mm pitch, IDC flat cable). *XP9* connector pinout is the following. Pins 1, 2 and 7 (DCD, DSR, DTR) are internally connected and left floated. Pins 4 and 6 (RTS, CTS) are internally connected and left floated. Pin 8 is not connected. Pin 9 – GND. Pin 3 – TX (output), pin 5 – RX (input). This connector can be connected directly to the standard DB-9 female connector via flat ribbon cable.
4. Connect RFCTL board to +12 VDC power supply (*X1* terminal block).

In variant (a) the remaining steps are identical to described in section 1.2. In (b) variant don't launch *RF Debug Application*. You can use any application which can send chars to COM-port (e.g. standard Windows Hyper Terminal application) to control RF-modules with aid of standard SCPI commands.

2 Remote Control

In (b) variant the remote control of the RFCTL Evaluation Board is based on the SCPI (Standard Commands for Programmable Instruments) protocol. It

is implemented via RS-232 and USB interfaces located on the top side of the board, fig. 11, (USB looks in OS like COM-port), so it can be easily managed by any software which has access to the COM-port of the PC. Only one port can be active, the choice is automatic and remains the same until the power off. The choice of the active port occurs when any byte comes to the one of the ports (not necessary a valid command). The connection via USB is implemented with aid of USB to UART bridge, so from the view point of PC software it looks like a COM-port, fig. 16. This method requires driver installation¹ (integral circuit CP2102). The driver is available for the following OSes: Win2K/XP/2K3, Vista, Windows 7, Mac OS, Linux 3.1.

2.1 Quick Start

As a simple remote control application you can use the standard Windows application – *HyperTerminal* (Start▷Programs▷Accessories▷Communications▷HyperTerminal) to send the SCPI commands to the instrument, fig. 12. COM-port settings are the following: 115200 bps, 8 data bits, parity none, 1 stop bit, flow control none, fig. 13. For more convenient use of the *HyperTerminal* it's recommended to configure the settings (File▷Properties, Settings tab, ASCII Setup... button) “Echo typed characters locally” and “Send line ends with line feeds”. For example, the following typed sequence in *HyperTerminal* application

```
*rst  
freq 100MHz  
pow -1dBm
```

will reset the instrument, set the output frequency to 100 MHz, and output RF level to –1 dBm.

2.2 SCPI Commands

Figure 14 shows the model of SCPI command processing.

2.2.1 SCPI Compliance

Instrument control command set is based on the SCPI v. 1999.0, but it's not fully compatible. The differences are listed below:

- not all required commands are implemented (see the command list);
- command parser recognizes only one command in single string, and the string size shall not exceed 64 symbols;
- command buffer length equals two, i.e. it is possible to send second command after the first one immediately (not waiting for completion of the first command), but not more;
- status information is not fully supported;

¹The driver (its last version) can be downloaded from CP2102 bridge vendor site [Silicon Labs](http://SiliconLabs.com)

i In applications where one command follows after another immediately without any time delay, for proper SCPI operation it is recommended to use *OPC? command

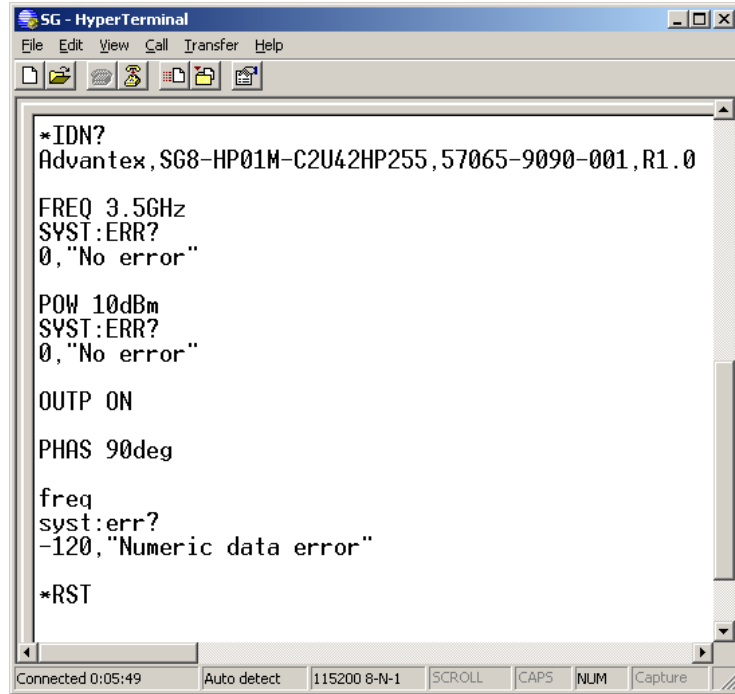


Figure 12: Example of the remote control using *HyperTerminal* application

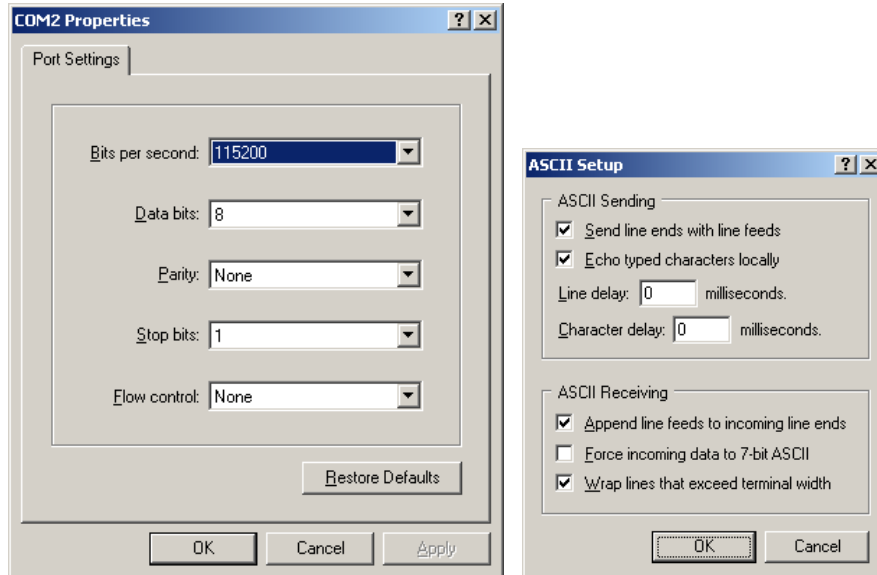


Figure 13: COM-port settings

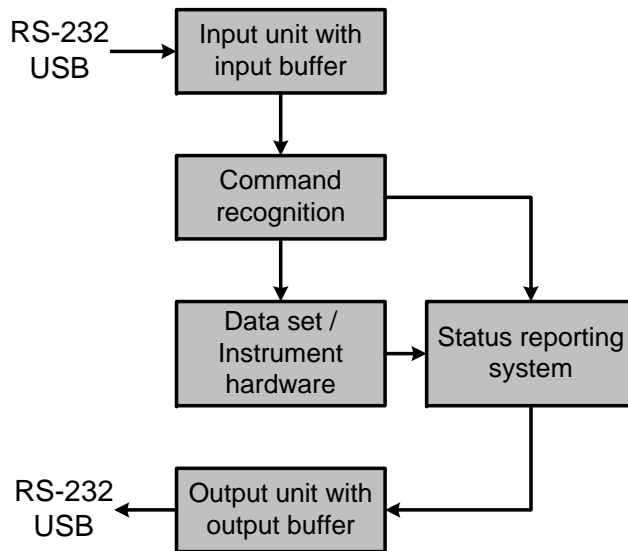


Figure 14: Instrument control and command processing model

- not all data formats are supported;
- documentation doesn't meet all the requirements of the standard.

2.2.2 SCPI Summary

Some notes listed below can help the newcomers to start the work with SCPI instrument.

- SCPI-commands are not case sensitive, i.e. commands `*RST` and `*rst` are identical.
- The first part of the command name, which is written in capitals, corresponds to a short form of the command. For example, to set frequency you can write `FREQ`, as well as `FREQuency`.
- Commands placed to the brackets are optional. For example the command `[SOURce:]FREQuency[:CW]` have the same effect when written in following forms `SOURce:FREQuency:CW`,
- `FREQuency:CW`, `SOURce:FREQuency`, `FREQuency`.
- If command requires the numerical parameter expressed in some units then the default unit exists. For example commands `FREQ 1GHz`, `FREQ 1E9Hz` and `FREQ 1000000000` sets the same frequency value 1 GHz, and the default unit is one Hz.

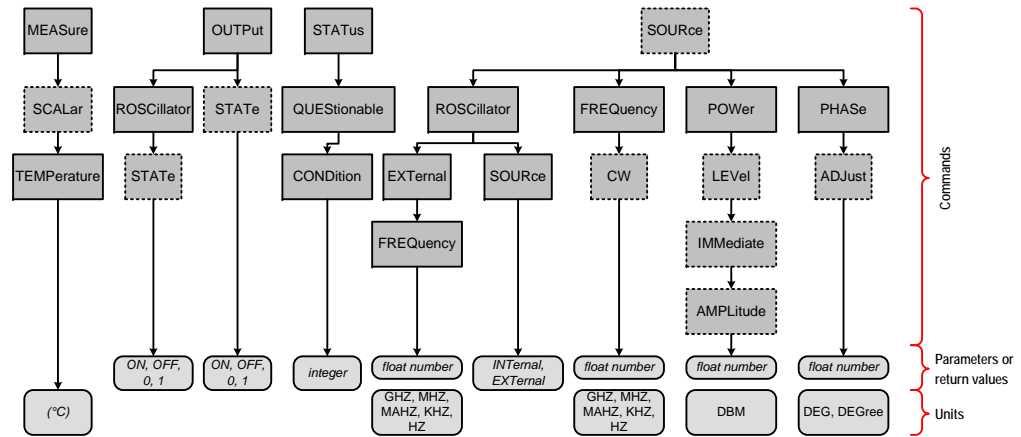


Figure 15: Command structure

- Commands ending with a question mark, are the queries and return a response. For example the *IDN? command returns an instrument ID. Many commands that set some value, also have a query form to read the current value, e.g. FREQ? – returns the current value of frequency.
- Commands that provide input of numerical parameters, can also take the following: MINimum – minimum value, MAXimum – maximum value, DEFault – default value. For example FREQ MAX command sets the maximum frequency (8 GHz), and FREQ DEF sets 1 GHz (default value).
- If the entered command didn't produce the desired effect, it's worth checking the error buffer by means of request SYSTem:ERRor:[NEXT]?. If it were not an error, the return value will be the following string 0,'No error', otherwise it returns error code and short description.

2.3 SCPI Command List

Figure 15 represents the available command tree (except standard commands). Optional parts of commands are marked with dotted line.

2.3.1 *CLS

*CLS command clears error buffer.

2.3.2 *IDN?

This query command returns the string containing the information about the instrument in the following format: *manufacturer,part number,serial number,firmware info.*

2.3.3 *RST

This command presets the instrument: frequency 1 GHz, level 0 dBm, phase 0 degrees, RF output is off.

It's a good practice to start the remote control process with the *RST command.

2.3.4 *OPC?

This query command returns 1 after its completion. It means that all previous commands are completed. It's good practice to use this command instead of fixed time delay between commands.

Examples: If you need to send several commands following immediately one after another, for example to set frequency and level, it's recommended to do it this way:

```
freq 100 mhz
*opc?
1
pow 1 dbm
*opc?
1
```

2.3.5 SYSTem:ERRor[:NEXT]?

This query command returns the string containing the error code and its description from the error buffer. If the buffer is empty then the returned string contains 0, 'No error'.

Error buffer is organized in form of FIFO (First In First Out). If input command doesn't meet the requirements of parser, or for whatever reason can not be executed, the corresponding message is placed to the error buffer. The buffer can contain 2 messages. There are two ways to clear the buffer: by reading the errors with aid of SYST:ERR? command one by another, or using *CLS command. When the buffer is already full, and one more error message comes, then the last error message will be rewritten by the following -350, "Queue overflow".

2.3.6 OUTPut[:STATe]

The command turns on or off the RF output as the hardware RF OUT ON/OFF button located at the front panel do.

Parameters: To turn on the output: 1 or ON, to turn off – 0 or OFF.

In query form the command returns 0 – if RF output is off, and 1 – if it's on.

Examples:

```
output on
outp off
outp:state 1
OUTPUT 0
OUTP:STAT?
```

2.3.7 OUTPut:ROSCillator[:STATe]

The command turns on and off the REF OUT output located on the rear panel of the instrument.

Parameters: To turn on the output: 1 or ON, to turn off – 0 or OFF.

In query form the command returns 0 – if REF OUT output is off, and 1 – if it's on.

Examples:

```
output:rosc on
outp:rosc off
outp:rosc:state 1
```

2.3.8 [SOURce:]FREQuency[:CW]

The command sets output frequency

Parameters: Parameter has the following form

[+|-]float_num[E[+|-]int_num] [GHZ|MHZ|MAHZ|KHZ|HZ] .

The default unit is HZ. Input value is rounded with accuracy of 10^{-4} . If the specified value is out of valid range, then the nearest limit value is applied, error message will not be formed.

Query form of the command returns current frequency in Hz. The result has the following form: [+|-]float_num.

Examples:

```
freq 2.1GHZ
frequency 21e-1ghz
sour:freq:cw 21E8
freq max
```

2.3.9 [SOURce:]POWer[:LEVel][:IMMediate][:AMPLitude]

The command sets RF output level.

Parameters: Parameter has the following form `[+|-]float_num[E[+|-]int_num]` [DBM]. The default unit is DBM. Input value is rounded with accuracy of 10^{-2} . If the specified value is out of valid range, then the nearest limit value is applied, error message will not be formed.

Query form of the command returns current output level in dBm. The result has the following form: `[+|-]float_num`.

Examples:

```
pow 5.1dbm
source:power 1.23
POWER 123E-2DBM
POW MAX
```

2.3.10 [SOURCE:]PHASe[:ADJust]

The command sets RF signal phase offset.

Parameters: Parameter has the following form `[+|-]float_num[E[+|-]int_num]` [DEGREE]. The default unit is DEGREE. Input value is rounded with accuracy of 10^{-2} . If the specified value is out of valid range, then the nearest limit value is applied, error message will not be formed.

Query form of the command returns current phase offset in degrees. The result has the following form: `[+|-]float_num`.

Example:

```
phas 90deg
PHASE 90DEG
phase:adj 90.1e-1
```

2.3.11 [SOURCE:]ROSCillator:SOURce

The command turns on and off the REF IN input, i.e. switches between internal and external reference frequency source.

Parameters: Input parameter value may be INTernal – internal reference source is used, or EXTernal – external reference source is used.

Query form of the command returns current source in use: INT or EXT.

Examples:

```
rosc:source INT
rocs:sour ext
```

2.3.12 [SOURCE:]ROSCillator:EXTernal:FREQuency

The command sets the value of external reference frequency at REF IN input.

Parameters: Input parameter has the following form:

`[+|-]float_num[E[+|-]int_num] [GHZ|MHZ|MAHZ|KHZ|HZ]` .

The default unit is HZ. Input value is rounded with accuracy of 10^{-4} . If the specified value is out of valid range, then the nearest limit value is applied, error message will not be formed.

Query form of the command returns current reference frequency value in Hz. The result has the following form: `[+|-]float_num`.

Examples:

```
rosc:ext:freq 100MHZ
SOURCE:ROSC:EXTERNAL:FREQUENCY 32MHz
rosc:ext:freq DEF
```

2.3.13 MEASure[:SCALar]:TEMPerature?

The command reads the internal temperature of RF synthesizer block in °C. The result has the following form: `[+|-]float_num`.

Examples:

```
meas:scal:temp?
meas:temp?
```

2.3.14 STATus:QUEStionable:CONDition?

The command returns current status of the instrument. The result has the following form: `integer_num`. Zero value means that all is OK, "1" in 4-th bit means that power which was set to the instrument is outside of the calibrated area, "1" in 6-th bit means that PLL is not locked.

Examples:

```
STAT:QUES:COND?
8
```

2.3.15 STATus:QUEStionable[:EVENT]?

The command returns the value of the Questionable Status Event Register. The result has the following form: `integer_num`. The command clears Questionable Status Event Register. The 6-th bit of the register is set to "1" if the PLL unlock event happens. Next command will return this value, and then register is cleared until the next event. Zero value means that all is OK, "1" in 6-th bit means that PLL unlock event happend.

Examples:

```
STAT:QUES:EVEN?
32
```


3 Evaluation Board Firmware Update

As it was mentioned above, RFCTL-1xM-PCB evaluation board can be used in two ways depending on the loaded firmware: (a) as the RS2SPI replacement or (b) as the RF-module control PCB. In (a) variant MCU works only as a bridge to translate data from USB/RS-232 to SPI. All calculations are performed by the PC. In (b) variant MCU works with high level SCPI commands. The content of all registers of RF-module is calculated by MCU of the evaluation board. You can load (a) or (b) variant of the firmware to the evaluation board by yourself using the procedure below.

MCU has two types of nonvolatile memory: Flash that contains program code, and EEPROM which stores all information about the device, including information about current firmware version. Thus the firmware update consists of two files in Intel Hex format, one – for Flash, and the other – for EEPROM. Firmware update is carried out through RS-232 or USB interface, connected to the PC. The connection via USB is implemented with aid of USB to UART bridge, so from the view point of PC software it looks like a COM-port, fig. 16. This method requires driver installation ² (integral circuit CP2102).

After driver installation and when the instrument is connected to the PC, the COM-port appears in the hardware list. Its number is indicated at the end of the string (fig. 16).

When powering up the evaluation board, it enters firmware update mode and waits for the appropriate command from the PC. If it doesn't receive the command within a half of a second, it exits the update mode and no longer responds to the read/write Flash or EEPROM commands, since in regular mode RS-232 and USB interfaces are used for remote control. For the firmware updating the XMI Programmer (XMEGA Instrument Programmer)³ application is used, fig. 17. The sequence of actions is the following:

1. Turn off the instrument;
2. Connect it to the PC via RS-232 or USB cable;
3. Launch the XMI Programmer software;
4. Select the port, then press **Connect** button. If you chose the wrong port, then press the **Stop** button, select the right port, then press **Connect**;
5. Turn on the instrument. The connection should be established within a second (you will see the string **Connecting... Ok!**). If it's not happened then probably wrong port was selected. In such a case press the **Stop** button, turn off the instrument, change the port in drop-down list, press **Connect** and then turn on the instrument. The string **Connecting... Ok!** should be displayed;

²The driver (its last version) can be downloaded from CP2102 bridge vendor site [Silicon Labs](http://www.siliconlabs.com)

³XMI Programmer software supports the following OSes: WinXP, Vista, Windows 7.

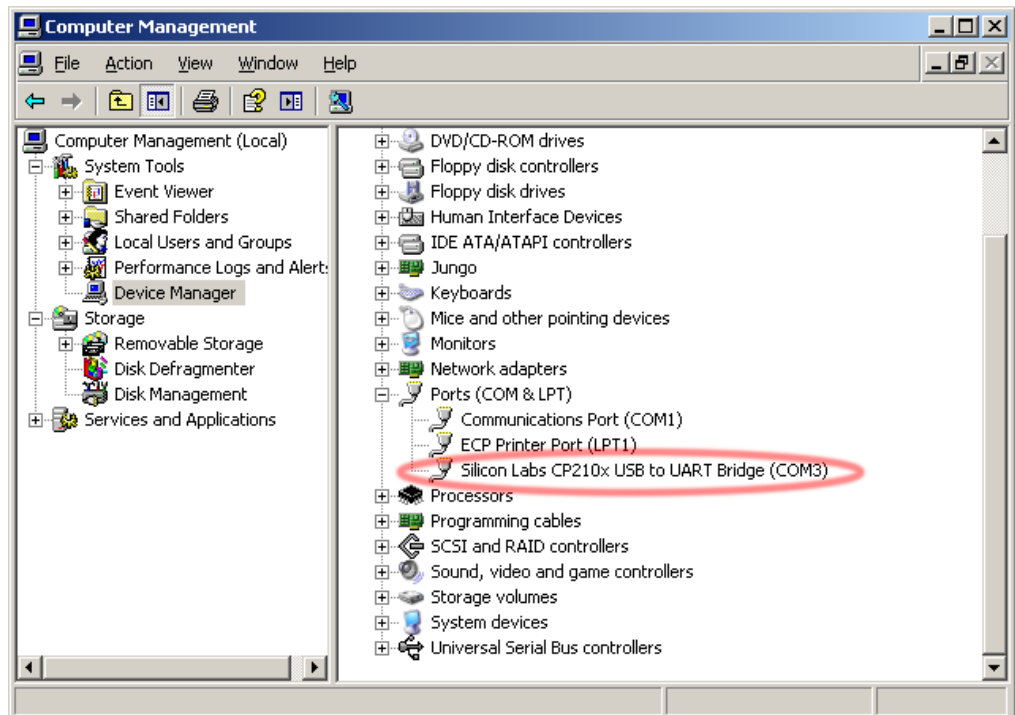


Figure 16: *Device Manager* window (My Computer▷Manage)

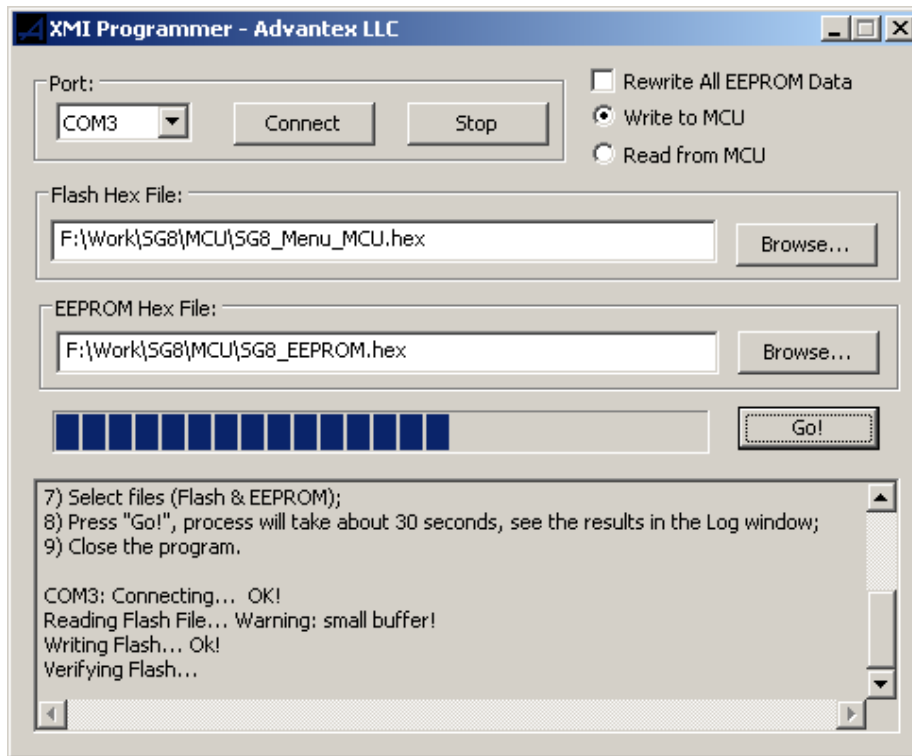


Figure 17: Firmware update application – XMI Programmer

6. Select the Write to MCU radio-box, the Rewrite All EEPROM Data flag should remain inactive⁴;
7. Select the files for Flash and EEPROM;
8. Press Go! button. The firmware update process takes about 30 seconds and comprises the data integrity check, the results of the operations are displayed in the log window;
9. After successful verifying the written data you may close the application.

You can save the current firmware and EEPROM data by selecting Read from MCU radio-button at 6-th step.

⁴If you select this flag, then all EEPROM memory will be re written, not only the part, that contains the information about current firmware revision. That is the personal data of the instrument (serial number, operation time, power-on count) will be erased, which is not desirable

4 LNO-6xM SPI Registers and Commands

4.1 General Description

LNO synthesizer does not include any MCU that would make all low-level calculations for you, just simple CPLD that works as SPI multiplexer 1-to-4 channels and contains some static registers (fig. 18). This CPLD does not use any clock signal except external SPI LNO_SCK line that changes its state only when loading new data to the LNO module. The reason of this approach is to avoid interference of MCU clock signal to analog lines and to make the response time of LNO module as fast as possible.

Figure 18 shows block diagram of LNO-6xM-RF synthesizer with internal control lines (colored in blue). There are two types of control lines: SPI channels and static registers. There is also one special line DDS_IOUPDATE which corresponds to DDS update command which toggles the corresponding DDS pin (IO_UPDATE pin of AD9912) to make new loaded in DDS data valid.

There are four internal SPI channels:

DDS_SPI is connected to DDS (AD9912) that is placed in PLL loop. It is used to control frequency and phase of VCO output signal.

DIGPOT_SPI is connected to Digital Potentiometer (AD5231) that is responsible for the adjustment of the gate bias voltage. The gate bias voltage sets the drain bias current of the output amplifier.

ADC_SPI is connected to ADC (AD7478) which is used for reading the drain bias current of the output amplifier.

FLASH_SPI is connected to Flash memory (25LC1024) which stores device ID data (signature, part number, serial number, date), calibration data for digital attenuator, etc.

TEMP_SPI is connected to internal temperature sensor (AD7814) which can be used to retrieve temperature of the module.

There are three static registers:

Func register is used to control internal power supply system and for reading PLL lock status.

Divider register is used to control additional dividers which extend the frequency range of the signal at VCO output.

Gain register is used to control digital attenuator which adjusts the output level of the RF signal.

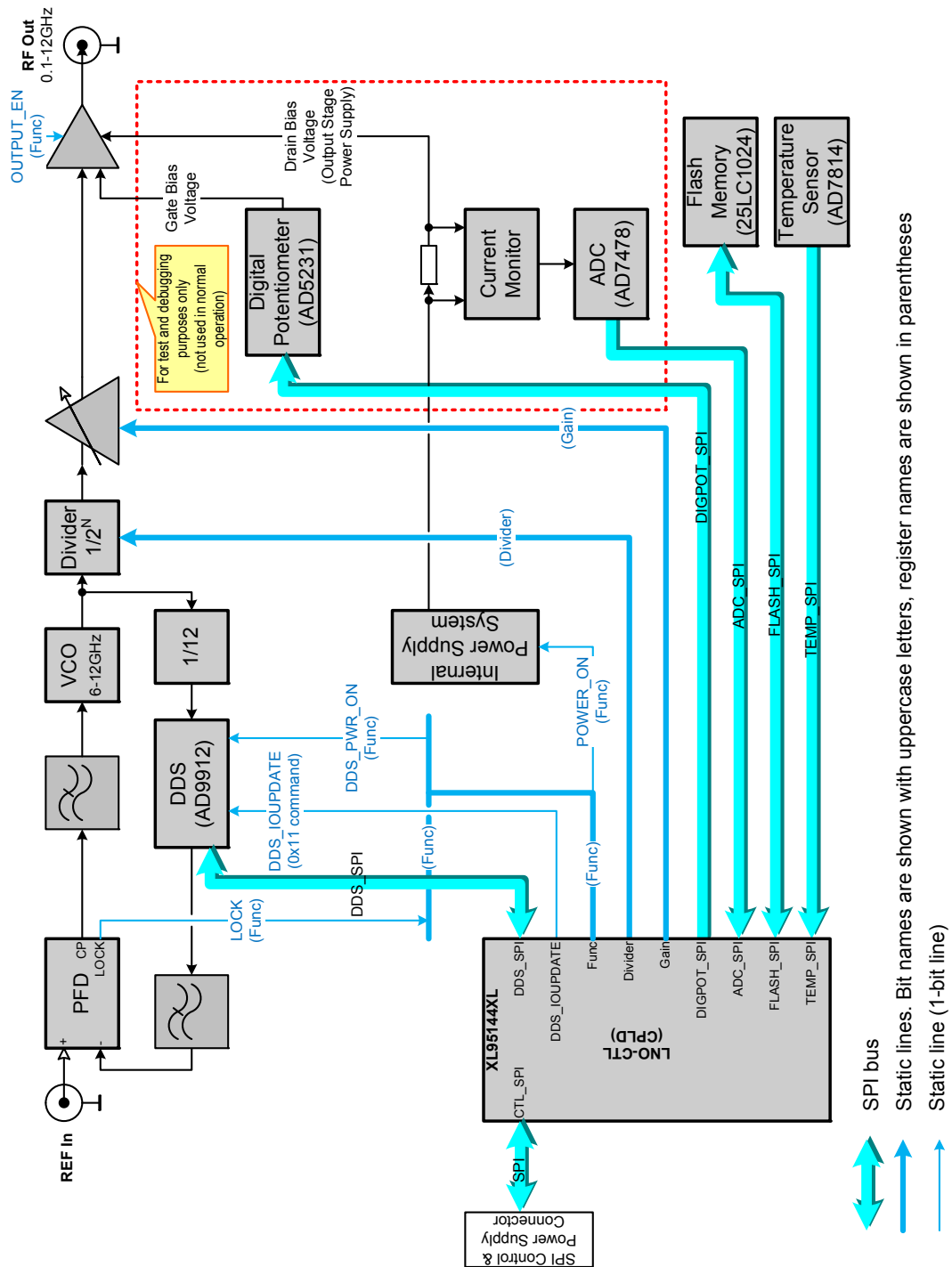


Figure 18: LNO synthesizer block diagram

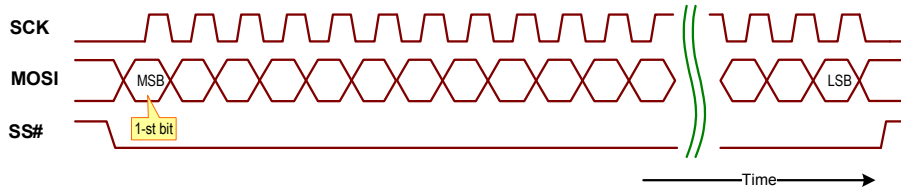


Figure 19: LNO SPI writing cycle diagram

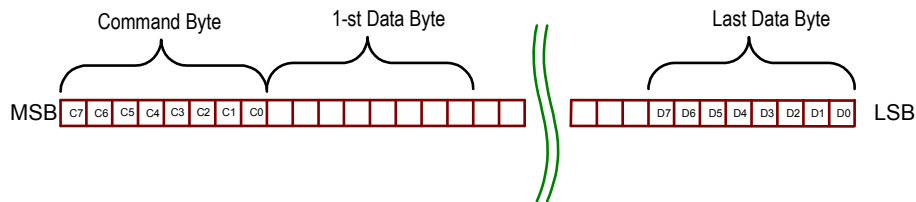


Figure 20: Command and data bytes

4.2 Format of SPI Commands

LNO SPI commands consist of one command byte $C[7:0]$ following by N data bytes. Number of data bytes (N) depends on the particular command. Data on LNO_MOSI signal line are latched on rising edge of LNO_SCK signal as shown on figure 19. MSB (Most Significant Bit) is loaded first, LSB (Least Significant Bit) – last. The first is the command byte $C[7:0]$ all other bytes are data (fig. 20). The command byte works as the address for multiplexer implemented in CPLD and defines the destination where to transfer following data – to one of the SPI channels or to one of the static registers, and the type of operation – writing or reading.

At reading cycle data on LNO_MISO line are switched on the falling edge of LNO_SCK signal and should be latched by master on rising edge of LNO_SCK signal accordingly as shown in figure 21.

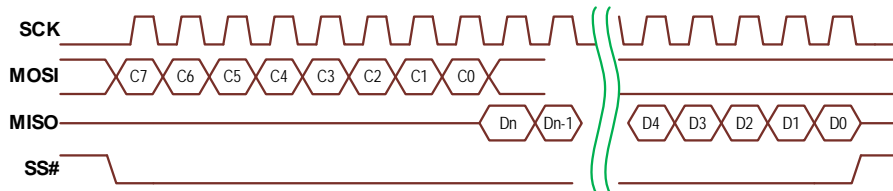


Figure 21: LNO SPI reading cycle diagram

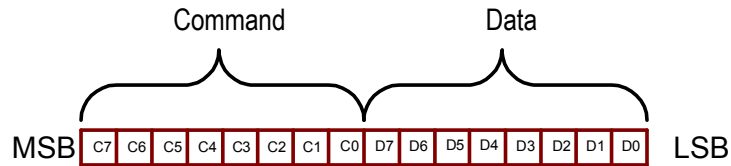


Figure 22: Writing 1 data byte

4.3 SPI Commands

Table 2 shows commands C[7:0] used to control LNO module.

4.3.1 Func register writing C[7:0]=0x01

This command writes 1 data byte D[7:0] to Func register (fig. 22). Table 3 shows the meaning of these data bits.

4.3.2 Func register reading C[7:0]=0x81

This command reads the content of Func register. See table 3 and figure 21 for more details.

4.3.3 Divider buffer register writing C[7:0]=0x02

This command writes 1 data byte D[7:0] to Divider buffer register (fig. 22). It defines the division factor of VCO output signal. VCO frequency range is from 6 to 12 GHz, so to obtain any other frequency which is lower than 6 GHz you need to divide VCO frequency. To cover all range from 100 MHz to 12 GHz output dividers with 2^N division factors are used (fig. 18). Table 4 shows the meaning of data bits controlling these output dividers.

4.3.4 Divider buffer register reading C[7:0]=0x82

This command reads the content of Divider register. See table 4 and figure 21 for more details.

4.3.5 Gain buffer register writing C[7:0]=0x03

This command writes 1 data byte D[7:0] to Gain buffer register (fig. 22). It is used to control digital attenuator which adjusts the output level of the RF signal. Approximately the output level in dBm is defined by the following: *Output Level = 15 - Attenuation*. Minimum attenuation corresponds to maximum output level. Table 5 shows register values and corresponding attenuation.

Table 2: LNO SPI Commands C[7:0]

	C[7:0]	# of bytes	Description
1	0x01	1	Writing to Func register. This command changes the states of static lines
2	0x81	1	Reading from Func register
3	0x02	1	Writing to Divider buffer register (without changes of the states of static lines)
4	0x82	1	Reading from Divider buffer register
5	0x03	1	Writing to Gain buffer register (without changes of the states of static lines)
6	0x83	1	Reading from Gain buffer register
7	0x04	1	Writing to DigPotWP register. This command changes the states of static lines
8	0x84	1	Reading from DigPotWP register
9	0x05	1	Writing to internal MuxOut register
10	0x85	1	Reading from internal MuxOut register
11	0x10	–	Access to DDS_SPI (DDS AD9912)
12	0x11	1	Toggling DDS IO_UPDATE signal (DDS AD9912)
13	0x12	1	Updating Divider register. This command changes the states of static lines according to the buffer register content loaded by 0x02 command
14	0x13	1	Updating Gain register. This command changes the states of static lines according to the buffer register content loaded by 0x02 command
15	0x1F	1	Toggling DDS IO_UPDATE signal (DDS AD9912) and updating Divider and Gain registers at one time
16	0x30	2	Access to TEMP_SPI (Temperature Sensor AD7814)
17	0x40	2	Access to ADC_SPI – current sensor ADC (AD7478)
18	0x50	3	Access to DIGPOT_SPI – digital potentiometer (AD5231)
19	0x70	–	Access to FLASH_SPI (Flash Memory 25LC1024)

Table 3: Func register

D7	D6	D5	D4	D3	D2	D1	D0	Description
LOCK			DDS_PWR_ON	OUTPUT_EN			POWER_ON	Bit Name
-	0	0	0	0	0	0	0	Default Values
x	x	x	x	x	x	x	0	Internal Power Supply System OFF
x	x	x	x	x	x	x	1	Internal Power Supply System ON
x	x	x	x	0	x	x	x	RF Out output is OFF
x	x	x	x	1	x	x	x	RF Out output is ON
x	x	x	0	x	x	x	x	DDS power is OFF. This also resets the DDS settings, so you need to init it again after power on
x	x	x	1	x	x	x	x	DDS power is ON
0	x	x	x	x	x	x	x	PLL is not locked (read only)
1	x	x	x	x	x	x	x	PLL is locked (read only)

Table 4: Divider register

D7	D6	D5	D4	D3	D2	D1	D0	Divider factor
					DIVIDER2	DIVIDER1	DIVIDER0	Bit Name
0	0	0	0	0	0	0	0	Default Values
x	x	x	x	x	0	0	0	1
x	x	x	x	x	0	0	1	2
x	x	x	x	x	0	1	0	4
x	x	x	x	x	0	1	1	8
x	x	x	x	x	1	0	0	16
x	x	x	x	x	1	0	1	32
x	x	x	x	x	1	1	x	64

Table 5: Gain register

D7	D6	D5	D4	D3	D2	D1	D0	Attenuation, dB
		GAIN5	GAIN4	GAIN3	GAIN2	GAIN1	GAIN0	Bit Name
0	0	0	0	0	0	0	0	Default Values
x	x	0	0	0	0	0	0	31.5
x	x	0	0	0	0	0	1	31
x	x	0	0	0	0	1	0	30
...								
x	x	1	1	1	1	1	0	0.5
x	x	1	1	1	1	1	1	0

Table 6: DigPotWP register

D7	D6	D5	D4	D3	D2	D1	D0	Attenuation, dB
							WP#	Bit Name
0	0	0	0	0	0	0	0	Default Values
x	x	x	x	x	x	x	0	Write operation protected
x	x	x	x	x	x	x	1	Write operation is allowed

4.3.6 Gain buffer register reading C[7:0]=0x83

This command reads the content of Gain buffer register. See table 5 and figure 21 for more details.

4.3.7 DigPotWP register writing C[7:0]=0x04

This command writes 1 data byte D[7:0] to DigPotWP register (fig. 22). It allows or denies the write operation to the digital potentiometer (table 6) which sets the gate bias voltage of the output amplifier.

4.3.8 DigPotWP register reading C[7:0]=0x84

This command reads the content of DigPotWP register. See table 6 and figure 21 for more details.

4.3.9 MuxOut register writing C[7:0]=0x05

This command writes 1 data byte D[7:0] to MuxOut register (fig. 22). This register defines the behaviour of the LNO_AUX output signal (table 7).

Table 7: MuxOut register

D7	D6	D5	D4	D3	D2	D1	D0	Attenuation, dB
FUNC_INT_EN	DIVN_INT_EN	GAIN_INT_EN	DDS_UPD_INT_EN			MUX1	MUX0	Bit Name
0	0	0	0	0	0	0	0	Default Values
x	x	x	x	x	x	0	0	LNO_AUX<='0'
x	x	x	x	x	x	0	1	LNO_AUX<=LOCK
x	x	x	x	x	x	1	0	LNO_AUX<=UPDATE
x	x	x	x	x	x	1	1	LNO_AUX<='1'
0	x	x	x	x	x	1	0	Disable the toggling of UPDATE signal when Func register is loaded
1	x	x	x	x	x	1	0	Enable the toggling of UPDATE signal when Func register is loaded with new data
x	0	x	x	x	x	1	0	Disable the toggling of UPDATE signal when Divider static lines are updated
x	1	x	x	x	x	1	0	Enable the toggling of UPDATE signal when Divider static lines are updated
x	x	0	x	x	x	1	0	Disable the toggling of UPDATE signal when Gain static lines are updated
x	x	1	x	x	x	1	0	Enable the toggling of UPDATE signal when Gain static lines are updated
x	x	x	0	x	x	1	0	Disable the toggling of UPDATE signal when DDS_IOUPDATE signal is toggling
x	x	x	1	x	x	1	0	Enable the toggling of UPDATE signal when DDS_IOUPDATE signal is toggling

D[1:0]="01" LNO_AUX output signal is connected to internal LOCK signal which indicates PLL lock status.

D[1:0]="10" LNO_AUX output signal is connected to internal UPDATE signal which indicates the update process of the static lines of Func, Divider, Gain registers and also it can indicate the toggling of DDS_IOUPDATE signal. The behaviour of UPDATE signal depends on D[7:4] flag bits. UPDATE signal can be used to observe transient process at RF output just after loading new data to the synthesizer. For example it can be used when measuring the frequency or level switching time. UPDATE signal switches from low to high state at rising edge of LNO_SCK signal at the same time as the static control lines (Func, Divider, Gain etc.) switch to their new states. UPDATE signal switches from high to low at rising edge of LNO_CS# signal.

4.3.10 MuxOut register reading C[7:0]=0x85

This command reads the content of MuxOut register. See table 7 and figure 21 for more details.

4.3.11 DDS_SPI Access C[7:0]=0x10

This command is used to access the DDS (AD9912) controlling phase and frequency of VCO output signal. Number of data bytes can be different but not less than 3 bytes. Actual length of data is defined in first 2 data bytes (Long Instruction). For more information refer to section 5 and AD9912 data sheet.

4.3.12 Toggling DDS_IOUPDATE C[7:0]=0x11

After the loading new data to DDS you need to toggle DDS_IOUPDATE signal by sending this command to SPI. It is required for DDS proper operation (see AD9912 data sheet).

4.3.13 Divider register updating C[7:0]=1x12

This command updates the control static signals which correspond to Divider buffer register.

4.3.14 Gain register updating C[7:0]=1x13

This command updates the control static signals which correspond to Gain buffer register.

4.3.15 All register updating and DDS_IOUPDATE toggling C[7:0]=1x1F

This command updates the control static signals which correspond to Divider and Gain buffer registers and toggles DDS_IOUPDATE signal.

4.3.16 TEMP_SPI Access C[7:0]=0x30

This command is used to access the temperature detector (AD7814) located near DDS IC. This command uses 2 data bytes D[15:0]. For more information refer to section 5 and AD7814 data sheet.

4.3.17 ADC_SPI Access C[7:0]=0x40

This command is used to access the ADC (AD7478) to read the drain bias current value. Rated drain bias current is 75mA. ADC reference voltage is 3.3V. Measured drain current I_D is converted to the input ADC voltage V_{ADC} as follows: $V_{ADC} = 20 \cdot I_D \cdot R$, where $R=1$ Ohm. Thus ADC code in normal operation should be about 1862 (full scale corresponds to 2^{12}). This command uses 2 data bytes D[15:0]. For more information refer to section 5 and AD7478 data sheet.

4.3.18 DIGPOT_SPI Access C[7:0]=0x50

This command is used to access to the digital potentiometer (AD5231) to adjust gate bias voltage which in turn defines drain bias current of the output amplifier. The digital potentiometer has 0 to 1023 range, 0 corresponds to the maximum drain current, 1023 corresponds to the minimum current. After adjusting the current, write the adjusted digital potentiometer value to its flash memory. This code will be loaded at power on automatically, so you won't need to access to ADC or Digital Potentiometer anymore. This command uses 3 data bytes D[23:0]. For more information refer to section 5 and AD5231 data sheet.

4.3.19 FLASH_SPI Access C[7:0]=0x70

This command is used to access to the flash memory of LNO module which stores product ID, serial number, calibration data etc. For more information refer to section 5 and 25LC1024 data sheet.

5 LNO-6xM SPI Programming

5.1 Parameter Calculations

Working with synthesizer implies some calculation procedure that should be implemented on the user side. It includes algorithms of DDS FTW register value and Divider register value calculations based on the input frequency. For more precise output level setting you can also use Gain register value calculations based on the calibration data stored in the flash memory of the module.

Thus we have some input parameters in user friendly format (e.g. frequency, phase, level as float numbers), some calibration data stored in flash memory embedded to the module, and as a result of calculation algorithms – output data in low-level format (i.e. register values), see figure 23. This section describes how to find these output parameters.

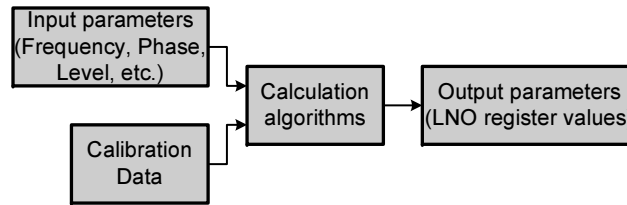


Figure 23: Input and output data for LNO programming

Table 8 shows input parameters. Some of them are in ready-to-use format, like `power_on`, `outamp_en`, other can not be loaded to LNO directly. Table 9 shows intermediate variables used in calculation algorithms, table shows output parameters which are downloaded to LNO registers.

n_pow

$$n_pow = \text{floor}(\log(6000.0/fr_out)/\log(2))+1$$

divider

$$\text{divider} = 2^{n_pow} \text{ (2 to the } n_power \text{)}$$

fr_vco

$$fr_vco = fr_out * \text{divider}$$

ftw

$$ftw = \text{round}((3 * 2^{50}) * fr_ref / fr_vco)$$

ptw

$$ptw = \text{round}((2^{16}) * \text{phase} * fr_ref / (2\pi * fr_out))$$

poutbits

For the approximate output level setting you can use the following equation.

$$poutbits = \text{round}(2 * (p_out + 16))$$

For more precise output level setting you can also use Gain register value calculations based on the calibration data stored in the flash memory of the module. The calculation of `poutbits` is based on the calibration values (Gain values) residing on 2-dimensional rectilinear grid (frequency and level). To find Gain value (i.e. `poutbits` value) for the arbitrary frequency-level point, bilinear interpolation can be used. Calibration data is retrieved from the flash memory – table `CTYPE=8`, for more details see section 5.5. Suppose that we have four

Table 8: LNO input parameters

Parameter	Type	Corresponding register bits	Description
power_on	1 bit	POWER_ON (Func), DDS_PWR_ON (Func)	Turns ON/OFF LNO internal power supply system. “0” – power system is OFF (standby mode), “1” – power is ON (normal operation)
outamp_en	1 bit	OUTPUT_EN (Func)	Turns ON/OFF RF Out output. “0” – output stage is OFF, “1” – ON
fr_out	float	–	Frequency of output signal (at RF Out output) in MHz, $93.75 \leq fr_out \leq 12000$
p_out	float	–	Level of output signal (at RF Out output) in dBm
fr_ref	float	–	Frequency of reference signal in MHz, $20 \leq fr_ref \leq 200^*$
phase	float	–	Relative phase shift of output signal in radians. For example, if you synchronize two LNO synthesizers from the same reference source, and set the same output frequency for both of them, you can shift phase of one signal relative to another in 360 degree range

* Reference frequency below 100 MHz may result in phase noise degradation of the RF output signal. When working with high reference signal (about 200 MHz), to prevent PLL locking problem it's recommended to initialize LNO module with 100 - 150 MHz reference frequency. Then you can switch reference frequency to 200 MHz.

Table 9: Intermediate variables and output parameters

Parameter/variable	Type	Description
Intermediate		
fr_vco	float	VCO output frequency ($6000 \leq fr_vco \leq 12000$)
divider	integer	Output division factor, multiple to 2, $1 \leq divider \leq 64$
Output		
ftw	48 bit	DDS Frequency Tuning Word
ptw	16 bit	DDS Phase Tuning Word
n_pow	3 bit	Division exponent ($divider=2^n_pow$), Divider register value
poutbits	6 bit	Data value for Gain register (i.e. RF path attenuator register value), corresponds to output signal level

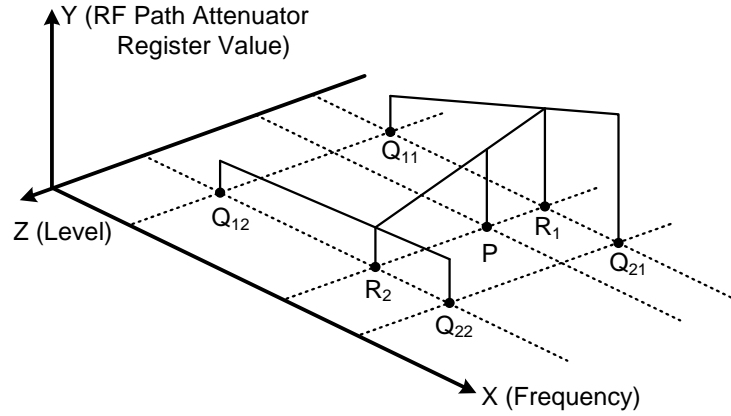


Figure 24: Bilinear interpolation

calibration points (Q_{11} , Q_{12} , Q_{21} , Q_{22}) around the point which Gain value we need to find (P), see fig 24. X axis means frequency in MHz, Z axis – output level in dBm, Y – Gain value. So we have $Q_{11} = (x_1, z_1)$, $Q_{12} = (x_1, z_2)$, $Q_{21} = (x_2, z_1)$, $Q_{22} = (x_2, z_2)$, $P = (x, z)$, $Y(Q_{11})$, $Y(Q_{12})$, $Y(Q_{21})$, $Y(Q_{22})$ – are calibration values, and we need to find $Y(P)$.

First we need to find Y values of auxiliary points R_1 and R_2 , where $R_1 = (x, z_1)$ and $R_2 = (x, z_2)$:

$$Y(R_1) \approx \frac{x_2 - x}{x_2 - x_1} Y(Q_{11}) + \frac{x - x_1}{x_2 - x_1} Y(Q_{21})$$

$$Y(R_2) \approx \frac{x_2 - x}{x_2 - x_1} Y(Q_{12}) + \frac{x - x_1}{x_2 - x_1} Y(Q_{22})$$

Then we need to find interpolated value between auxiliary points R_1 and R_2 :

$$Y(P) = \frac{z_2 - z}{z_2 - z_1} Y(R_1) + \frac{z - z_1}{z_2 - z_1} Y(R_2).$$

Gain register maximum value (0x3F) corresponds to maximum output level, value 0x00 – corresponds to minimum signal level at RF Out output, LSB of Gain register is about 0.5 dB.

5.2 Initialization

After power on the LNO synthesizer is in standby mode, i.e. internal power supply system is off, and DDS, Gain, Divider and other registers are in their default state. So just after power on you need to initialize it by the following procedure:

1. Set minimum output signal level by loading to SPI 0x0300.

Table 10: Turning on the internal power supply system at initialization process

C[7:0]	D7	D6	D5	D4	D3	D2	D1	D0
				dds_pwr_on	output_en			power_on
0x01	0	0	0	0	1	0	0	1

Table 11: Turning on the DDS power supply at initialization process

C[7:0]	D7	D6	D5	D4	D3	D2	D1	D0
				dds_pwr_on	output_en			power_on
0x01	0	0	0	1	1	0	0	1

2. Turn on the internal power supply by loading to SPI value shown in table 10.
3. Turn on the DDS power supply by loading to SPI value shown in table 11. Turning on the DDS power supply as separate operation is required for all power supply voltages were settled before DDS power-on.
4. Reset the DDS – 0x10001201
5. Update DDS – 0x1100 (toggle DDS_IOUPDATE signal)
6. Initialize DDS:
 - (a) 0x10000080 (write to DDS at address 0x0000)
 - (b) 0x10001090 (write to DDS at address 0x0010)
 - (c) 0x10040BFF (write to DDS at address 0x040B)
 - (d) 0x10040C03 (write to DDS at address 0x040C)
7. Update DDS and Gain register – 0x1F00

5.3 Setting Frequency, Phase and Level

To set level and frequency at one time use the following command sequence:

1. 0x1061AB[ftw] (setting frequency)

2. 0x020[0][n_pow] (Divider register)
3. 0x03[00][poutbits] (setting output level)
4. 0x1F00 (update all control static lines and toggle DDS_IOUPDATE)

To set only level use the following command sequence:

1. 0x03[00][poutbits] (setting output level)
2. 0x1300 (update Gain static lines)

To set only frequency (without output level correction) use the following command sequence:

1. 0x1061AB[ftw] (setting frequency)
2. 0x020[0][n_pow] (Divider register)
3. 0x1F00 (update all control static lines and toggle DDS_IOUPDATE)

To set new phase of output signal you need to load the following:

1. 0x1061AD[ptw]
2. 0x1100 (toggle DDS_IOUPDATE)

For the PLL to remain in locked condition, it's good idea to set new phase by several small phase shifts. Phase difference between old and new consecutive values should meet the following condition:

$$\frac{\Delta ptw}{2^{16}} \ll 1.$$

5.4 Reading internal temperature

Temperature detector (AD7814) embedded to LNO calculates temperature each 350 us. Other times it resides in standby mode. If reading temperature rate is faster than 350 us then detector is not able to calculate new temperature value, and you will read the same value. Thus it is recommended to read temperature value not faster than one time per 500 us.

To read the temperature proceed as follows:

1. 0x300000 (sets normal operation of Temperature Detector)
2. Wait for 500 us
3. 0x30FFFF (sets Temperature Detector to power down mode and reads temperature code in format b00D[9:0]0000, where D is 10-bit signed integer). Temperature value in °C can be found by the following equation $T = 0.25 \cdot D$.

For more details about Temperature Detector operation see AD7814 datasheet.

i It should be noted that when you set new frequency, the output level can be slightly different from the level at previous frequency. For large frequency step it's recommended to recalculate *poutbits* for new frequency for output level correction



Figure 25: Structure of flash-memory data transfer

Table 12: Flash-memory commands

Command	Byte	Description
FLASH_CMD_READ	0x03	Read data
FLASH_CMD_WRITE	0x02	Write data
FLASH_CMD_WREN	0x06	Write data enable
FLASH_CMD_WRDI	0x04	Write data disable
FLASH_CMD_RDSR	0x05	Read status register
FLASH_CMD_WRSR	0x01	Write status register
FLASH_CMD_PE	0x42	Page erase
FLASH_CMD_SE	0xD8	Byte erase
FLASH_CMD_CE	0xC7	Erase all
FLASH_CMD_RDID	0xAB	Read ID and Turn on flash-memory Power Supply
FLASH_CMD_PDP	0xB9	Turn off flash-memory Power Supply

5.5 Flash Memory

LNO synthesizer has 1 Mbit (131072 bytes) Flash-memory. Its address space is within 0x00000 to 0x1FFFF range. Memory space is divided into pages, 256 bytes each. It is possible to work with separate bytes and with pages. Pages start at addresses 0xXXX00, where XXX is in range 000 to 1FF. The number of data bytes in packet can be different and depends on the actual memory command. General structure of the packet is shown in figure 25. First command byte, FLASH_ADR (0x70 – Flash-memory access, see table 2), is for CPLD multiplexer, second byte is command for the 25LC1024 memory, followed by data bytes if required. Flash-memory commands are listed in table 12.

5.5.1 FLASH_CMD_READ (0x03)

Data read command, see figure 26. First byte – FLASH_SPI access command



Figure 26: Flash-memory data read command



Figure 27: Flash-memory data write command

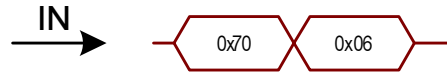


Figure 28: Flash-memory write enable command

for CPLD multiplexer (0x70), second – flash data read command FLASH_CMD_READ (0x03), then three bytes which contains data address ADR[2:0]. Actually it's start address, so if you hold SS# signal in active position ("0") you will get data at next address for each CLK cycle. For example if we have the following data (3 bytes) 0xAA, 0xBB, 0xCC which are located at addresses 0x000006, 0x000007 and 0x000008 respectively. To retrieve these three bytes at once we need to send the following command: 0x0703000006000000. Last three zero bytes in this command are used to generate CLK cycles to retrieve three consecutive data bytes. As a response on MISO line we will obtain the following: 0xAABBCC.

5.5.2 FLASH_CMD_WRITE (0x02)

Data write command, see figure 27. First byte – FLASH_SPI access command for CPLD multiplexer (0x70), second – flash data write command FLASH_CMD_WRITE (0x02), then three address bytes ADR[2:0] and data bytes. First data byte will be written at ADR[2:0] address, second at ADR[2:0]+1 and so on, but all written data should reside inside the same page. This way maximum number of data bytes is 256 if ADR[2:0] points to the start of page. Otherwise if address ADR[2:0] plus number of bytes exceeds the address of end of page, some data of this page will be overwritten. This will result in lost of some data.

5.5.3 FLASH_CMD_WREN (0x06)

Write enable command, see figure 28. This command sets WEL bit to "1" in flash-memory status register. To enable any write or erase operation, this bit should be set to "1", otherwise these operations won't be performed. WEL bit is reset automatically to "0" after performing the following commands:

- FLASH_CMD_WRDI
- FLASH_CMD_WRSR
- FLASH_CMD_WRITE
- FLASH_CMD_PE

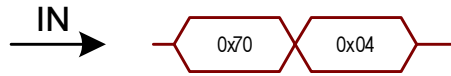


Figure 29: Flash-memory write/erase disable command

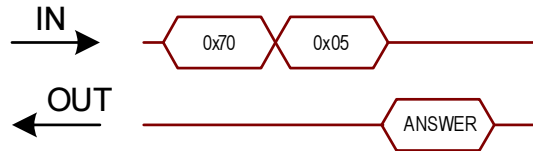


Figure 30: Flash-memory status register reading

- FLASH_CMD_SE
- FLASH_CMD_CE

After power on WEL bit is also in “0” state.

5.5.4 FLASH_CMD_WRDI (0x04)

This command disables write and erase operations by setting WEL bit to “0” in status register of flash-memory. Command is shown in figure 29.

5.5.5 FLASH_CMD_RDSR (0x05)

This command reads flash-memory status register, see figure 30. Table 13 shows the content of status register.

WIP – flash-memory busy. “1” – write operation in progress (is not finished).
 “0” – all operations are finished.

WEL – write enable. “0” – write/erase operation disabled, “1” – enabled.

BP1, BP0 – memory block write protection. All address space is divided into 4 blocks and these bits can be used to control protection for write/erase operations for these blocks, see table 14.

To perform reading operation you need to send 0x700500.

Table 13: Flash-memory status register

	R7	R6	R5	R4	R3	R2	R1	R0
Read/Write	–	–	–	–	R/W	R/W	R	R
	X	X	X	X	BP1	BP0	WEL	WIP

Table 14: Flash-memory block protection

BP1	BP0	Protected area	Unprotected area
0	0	–	All address area (00000h-1FFFFh)
0	1	Upper 1/4 address space (18000h-1FFFFh)	Lower 3/4 address space (00000h-17FFFh)
1	0	Upper 1/2 address space (10000h-1FFFFh)	Lower 1/2 address space (00000h-0FFFFh)
1	1	All address space (00000h-1FFFFh)	–



Figure 31: Flash-memory write to status register

5.5.6 FLASH_CMD_WRSR (0x01)

Write to status register, figure 31. Actually this command is used to set BP0 and BP1 bits, since WEL bit can be controlled via FLASH_CMD_WREN and FLASH_CMD_WRDI commands.

5.5.7 FLASH_CMD_PE (0x42)

Command erases content of a page which address is inserted in command, see figure 32.

5.5.8 FLASH_CMD_SE (0xD8)

Clears one byte, figure 33.

5.5.9 FLASH_CMD_CE (0xC7)

Clears all memory, figure 34.

5.5.10 FLASH_CMD_RDID (0xAB)

This command turns on the memory power supply and reads ID number of the chip (ID=0x29), figure 35. To perform reading you should send 0x70AB00, and



Figure 32: Flash-memory page erase command



Figure 33: Flash-memory byte erase command

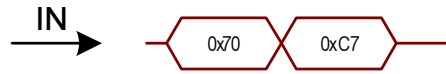


Figure 34: Flash-memory clear all command

you get 0x29 in response.

5.5.11 FLASH_CMD_PDP (0xB9)

Turns off the memory chip power supply, see figure 36.

5.6 Memory address and data mapping

Address and data mapping is shown in table 15.

Table 15: Memory address and data mapping

Address	Variable/Value	Description
0x00	0xAA	CONFIGBLKSIG0
0x01	0xBB	CONFIGBLKSIG1
0x02	0xCC	CONFIGBLKSIG2
0x03	0xDD	CONFIGBLKSIG3
0x04	PID0	Product ID (LSB)
0x05	PID1	Product ID (MSB)
0x06	SID0	Software ID (LSB)
0x07	SID1	Software ID (MSB)
0x08	SN0	Serial Number (LSB)
0x09	SN1	Serial Number (MSB)
0x0A	LOT	Lot
0x0B	DY	Year of production

(continued on next page)

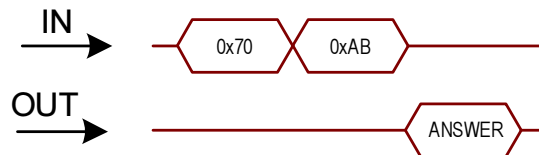


Figure 35: Flash-memory power on and read ID command

(continued Table 15, beginning on page 47)

Address	Variable/Value	Description
0x0C	DM	Month of of production
0x0D	DD	Day of production
0x10	REF_FR0	Reference frequency in Hz (LSB)
0x11	REF_FR1	Reference frequency in Hz
0x12	REF_FR2	Reference frequency in Hz
0x13	REF_FR3	Reference frequency in Hz (MSB)
0x14	DATA_SIZE0	Size of Data Block (LSB)
0x15	DATA_SIZE1	Size of Data Block
0x16	DATA_SIZE2	Size of Data Block
0x17	DATA_SIZE3	Size of Data Block (MSB)
0x18	FLASH_SIZE0	Flash-memory size (LSB) = 0x00
0x19	FLASH_SIZE1	Flash-memory size = 0x00
0x1A	FLASH_SIZE2	Flash-memory size = 0x00
0x1B	FLASH_SIZE3	Flash-memory size (MSB) = 0x02
0x00FE	CRC0	CRC 16-bit (LSB)
0x00FF	CRC1	CRC 16-bit (MSB)
0x0100 (0xXXX00)	0x99	TABLESIG0
0x0101	0x88	TABLESIG1
0x0102	0x77	TABLESIG2
0x0103	0x66	TABLESIG3
0x0104	CTYPE	Table type (characteristic type)
0x0105	XVALUE	Type of X axis values
0x0106	YVALUE	Type of Y axis values
0x0107	ZVALUE	Type of Z axis values
0x0108	ZCOUNT0	Number of points on Z axis (LSB)
0x0109	ZCOUNT1	Number of points on Z axis
0x010A	ZCOUNT2	Number of points on Z axis
0x010B	ZCOUNT3	Number of points on Z axis (MSB)
0x010C	XYCOUNT0	Number of points on X axis (LSB)
0x010D	XYCOUNT1	Number of points on X axis
0x010E	XYCOUNT2	Number of points on X axis
0x010F	XYCOUNT3	Number of points on X axis (MSB)
0x0110	0x33	XROWSIG0
0x0111	0x22	XROWSIG1
0x0112	X_MULT	X multiplier
0x0113	-	Not used
0x0114	X_L0	X grid value 0 (LSB)

(continued on next page)

(continued Table 15, beginning on page 47)

Address	Variable/Value	Description
0x0115	X_H0	X grid value 0 (MSB)
0x0116	X_L1	X grid value 1 (LSB)
0x0117	X_H1	X grid value 1 (MSB)
0x0118	X_L2	X grid value 2 (LSB)
0x0119	X_H2	X grid value 2 (MSB)
0x011A	X_L3	X grid value 3 (LSB)
0x011B	X_H3	X grid value 3 (MSB)
0x011C	X_L4	X grid value 4 (LSB)
0x011D	X_H4	X grid value 4 (MSB)
	0x55	ZROWSIG0
	0x44	ZROWSIG1
	Z_L0	Z grid value 0 (LSB)
	Z_H0	Z grid value 0 (MSB)
	Y_L0	Y value for X grid value 0 and Z grid value 0 (LSB)
	Y_H0	Y value for X grid value 0 and Z grid value 0 (MSB)
	Y_L1	Y value for X grid value 1 and Z grid value 0 (LSB)
	Y_H1	Y value for X grid value 1 and Z grid value 0 (MSB)
	Y_L2	Y value for X grid value 2 and Z grid value 0 (LSB)
	Y_H2	Y value for X grid value 2 and Z grid value 0 (MSB)
	0x55	ZROWSIG0
	0x44	ZROWSIG1
	Z_L1	Z grid value 1 (LSB)
	Z_H1	Z grid value 1 (MSB)
	Y_L0	Y value for X grid value 0 and Z grid value 1 (LSB)
	Y_H0	Y value for X grid value 0 and Z grid value 1 (MSB)
	Y_L1	Y value for X grid value 1 and Z grid value 1 (LSB)
	Y_H1	Y value for X grid value 1 and Z grid value 1 (MSB)
	Y_L2	Y value for X grid value 2 and Z grid value 1 (LSB)

(continued on next page)

(continued Table 15, beginning on page 47)

Address	Variable/Value	Description
	Y_H2	Y value for X grid value 2 and Z grid value 1 (MSB)
0xFFFFE	CRC0	CRC 16-bit (LSB)
0xFFFFF	CRC1	CRC 16-bit (MSB)

Memory space is divided into two blocks: configuration block (00000h-000FFh) and data block. Data of each block is supplied with check sum (CRC). CRC for configuration block is placed at 000FEh-000FFh, CRC for data block is placed at the end of last page of data block. The address of the CRC word for data block can be found as $DATASIZE[3:0]+0x100$. CRC calculation algorithm is standard CCITT, 16-bit, polynomial A001h, starting with FFFFh. CRC is calculated for all data (even for unused space) that is multiple to memory page, i.e. for configuration block – from 0h to FD, and for data block – from 100h to $DATASIZE[3:0]+0xFF$ inclusively.

5.6.1 Configuration Block

Configuration block occupies 256 bytes, from 00000h to 000FFh. First four bytes are signature of the configuration block 0xDDCCBBAA.

PID[1:0] – product ID (unsigned integer 0 to 65535), first (5-digit in DEC) number in full serial number of particular device. This ID corresponds to the part number of the device. For example for the following partnumber LNO-HP35M-RF the ID will be 04608, while full serial number can be 04608-3021-014.

SID[1:0] – software ID, it can be treated as version of set of calibration data tables in data block.

SN[1:0] – serial number (unsigned integer 0 to 999), last number (3-digit in DEC) in full serial number of particular device. It's 014 for the example above.

LOT – lot number (unsigned integer 0 to 9).

DY – year of production starting from 1970 (unsigned integer). Thus real year is $DY+1970$. Last digit of the year is coded as first digit in second digit group of full serial number, i.e. 3 in the example above.

DD – day of production (unsigned integer 1 to 31).

FR_REF[3:0] – reference frequency value expressed in Hz (unsigned integer, normally 147 000 000). Please note, that before using it in calculations

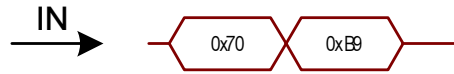


Figure 36: Flash-memory power off command

Table 16: Type of Data Table (CTYPE)

CTYPE	Description
0	Type is undefined
0x08	Calibration data for precise setting of output level

you need to express it in MHz, i.e. you should to change its type to float and then divide this value by 1 000 000.

DATA_SIZE[3:0] – size of data block excluding its CRC (2 bytes).

FLASH_SIZE[3:0] – size of embedded flash-memory in bytes.

Unused bytes in configuration block are filled with zeroes.

5.6.2 Data block

Data block may contain one or more calibration data tables. Each table starts with new page. The structure of the table is described below.

First 4 bytes (TABLESIG[3:0]) of the table are signature 0x66778899.

Type of data table (CTYPE) The signature is followed by type of data table (CTYPE), the description is shown in table 16.

Tables with different types can be placed in memory in any order. Table with CTYPE=0x08 is optional.

For CTYPE=0x08 table X-data are frequency grid, normally it's not regular: 10 to 100 MHz with 1 MHz step, 100 MHz to 1 GHz with 10 MHz step, and 1 to 8 GHz with 25 MHz step. X-data values are expressed in MHz. Z-data are power level grid values, normally it's regular: from -10 to +14 dBm with 2 dB step. Values are in dBm. Y-data are calibrated Gain register values (2-byte unsigned integer) which correspond to the appropriate frequency (X-data value in MHz) and level (Z-data value in dBm). Y-data values should be decoded as follows:

- 0xFFFF – calibration point is not valid, it can't be used in interpolation algorithm.
- 0xFFFF > Y-data > 0x7FFF – calibration point can be used in interpolation algorithm, but its precision is not guaranteed.
- Y-data ≤ 0x7FFF – calibration point is valid and can be used in interpolation algorithm.

Table 17: X,Y and Z-data format (XVALUE, YVALUE, ZVALUE)

{X Y Z}VALUE	Description
0	Type is not defined
1	2-byte integer
2	2D.2 - fixed point (2 digits after point) value, 2-byte. To convert it to float you should divide it by 100.0

X,Y,Z-data type (XVALUE, YVALUE, ZVALUE) Table 17 shows types of X, Y and Z-data values.

Number of Z-data points (ZCOUNT[3:0]) 4-byte integer. It is the number of Z-axis grid points.

Number of X and Y-data points per row (XYCOUNT[3:0]) 4-byte integer. It is the number of X-axis grid points, and it is equal to number of Y-data points per row (i.e. per one Z-data value).

X-data value multiplier (X_MULT) Actually it is commonly used for frequency data. If this value equals 6 then X-data (frequency) is expressed in MHz, if 3 – in kHz, 0 – in Hz.

X_H[XYCOUNT-1:0] and X_L[XYCOUNT-1:0] MSB and LSB bytes of X-axis grid values. Usually they are frequency values.

ZROWSIG[1:0] Each data row starts with signature ZROWSIG[1:0]=0x4455.

Z_H[ZCOUNT-1:0] and Z_L[ZCOUNT-1:0] Z_H[N] and Z_L[N] are MSB and LSB bytes of Z-value for the following by Y-data Y_H[XYCOUNT-1:0,N] and Y_L[XYCOUNT-1:0,N].

Y_H[XYCOUNT-1:0] and Y_L[XYCOUNT-1:0] Z-value is followed by Y-data row.