



**SYNTH300-TRIG,
SYNTH300-TRIG-LVDSX2,
And SYNTH300-TRIG-HS**

Dual 100 kHz – 300 MHz Direct Digital Synthesizer

Product Specification and Manual – v1.05

May 26, 2008

Covers Boards With Firmware rev 1.00 (5/2/05)
For Windows Vista™, XP™ and Windows 2000™

Furaxa, Inc.

34 Canyon View, Orinda, CA 94563

(925) 253-2969

Fax (925) 253-4894

e-mail : support@furaxa.com

URL : www.furaxa.com

copyright c 2005, 2006, 2007, 2008 Furaxa, Inc.

TABLE OF CONTENTS

1. WARRANTY..... 4

2. MODEL DESCRIPTIONS..... 5

 2.1 MODEL SYNTH300-TRIG..... 5

 2.2 MODEL SYNTH300-TRIG (OPTION LVDSX2)..... 5

 2.3 MODEL SYNTH300-TRIG (OPTION LVDSX2+TTL)..... 5

 2.4 MODEL SYNTH300-TRIG-HS..... 5

3. SPECIFICATIONS 6

 3.1 SYNTHESIZER OUTPUTS (SYNTH300-TRIG)..... 6

 3.2 FREQUENCY STABILITY..... 6

 3.3 PHASE RESOLUTION..... 6

 3.4 TRIG/MODA AND TRIG/MODB INPUT/OUTPUT..... 7

 3.5 LVDSB+ AND LVDSB- OUTPUTS..... 7

 3.6 EXT CLK INPUT OR 1GHZ CLK OUTPUT (SEE FIG 3.1)..... 7

 3.7 GENERAL..... 7

 3.8 PHYSICAL..... 7

4. HARDWARE ARCHITECTURE 9

 4.1 SYNTHESIZER A & B OUTPUTS (DDS A OUT, DDS B OUT)..... 9

 4.2 LVDS OUTPUTS LVDSB+, LVDSB-..... 9

 4.3 TRIG/MODA AND TRIG/MODB I/O LINES..... 9

 4.4 LED INDICATORS..... 10

 4.4.1 Ext Clk LED..... 10

 4.4.2 ARM Sweep LED..... 10

 4.4.3 Synth On LED..... 10

5. HARDWARE INSTALLATION AND SETUP..... 11

6. SOFTWARE INSTALLATION AND SETUP..... 12

 6.1 SOFTWARE PACKAGE CONTENTS..... 12

 6.1.1 Driver..... 12

 6.1.2 The Example Program..... 12

 6.1.3 The Source Code..... 12

 6.2 SOFTWARE INSTALLATION FOR WINDOWS 2000 OR WINDOWS XP TM..... 12

 6.2.1 Windows XP TM..... 12

 6.2.2 Windows 2000 TM..... 13

 6.2.3 Continuing with the software installation in 2000 and XP..... 13

7. RUNNING SYNTH300 PROGRAMS..... 14

 7.1 USING THE SYNTH300 SERIES UNDER WINDOWS 2000/XP..... 14

8. USING THE SYNTH300 GUI UNDER WINDOWS XP OR WINDOWS 2000..... 15

 8.1 EXAMPLE #1 – TWO INDEPENDENT SINUSOIDS..... 16

 8.2 EXAMPLE #2 – PHASE-LOCKED SINUSOIDS..... 17

 8.3 EXAMPLE #3 – TWO FULLY INDEPENDENT SWEPT SINUSOIDS..... 18

 8.4 EXAMPLE #4 – EXTERNAL MODULATION EXAMPLE..... 19

9. CUSTOM APPLICATION PROGRAMMING OF THE SYNTH300..... 20

 9.1 USING THE CSYNTHAPI OBJECT 21

FURAXA

9.2 INSTANTIATING A CSYNTHAPI OBJECT.....	21
9.3 CSYNTHAPI OBJECT MEMBER FUNCTIONS.....	22
10. LOW LEVEL SOFTWARE INTERFACE (MOST USERS NEED NOT READ).....	26
10.1 PCI CONFIGURATION HEADER.....	26
10.2 SYNTH300 CONTROL REGISTER.....	27
10.2.1 Unreset (write only).....	27
10.2.2 Output TTL_125 (write only).....	27
10.2.3 Update (write only).....	27
10.2.4 Synthesizer A Profile Select (write only).....	28
10.2.5 Synthesizer B Profile Select (write only).....	28
10.2.6 External Modulation Mode (write only).....	28
10.2.7 Enable External Modulation (write only).....	29
10.2.8 Read_Synth_Select (write only).....	29
10.2.9 Arm_Write (write only).....	29
10.2.10 External_clk (write only).....	29
10.2.11 Write is to Synth B, Write is to Synth A (write only).....	29
10.2.12 Synthesizer IC Register Address (write only).....	29
10.2.13 Sample IC Data Word (write only).....	29
10.3 AUXILIARY REGISTER.....	31
10.4 THE AD9858.....	31



1. Warranty

Furaxa, Inc. hardware, software and firmware products are warranted against defects in materials and workmanship for a period of two (2) years from the date of shipment of the product. During the warranty period, Furaxa, Inc. shall, at its option, either repair or replace hardware, software or firmware products which prove to be defective. This limited warranty does not cover damage caused by misuse or abuse by customer, and specifically excludes damage caused by the application of excessive voltages to the inputs and/or outputs of PCI boards. The limited warranty additionally excludes damage caused by overheating due to installation of the product in systems that do not have direct forced air flow over the PCI bus slots.

While Furaxa, Inc. hardware, software and firmware products are designed to function in a reliable manner, Furaxa, Inc. does not warrant that the operation of the hardware, software or firmware will be uninterrupted or error free. Furaxa products are not intended to be used as critical components in life support systems, aircraft, military systems or other systems whose failure to perform can reasonably be expected to cause significant injury to humans. Furaxa expressly disclaims liability for loss of profits and other consequential damages caused by the failure of any product, and recommends that customer purchase spare units for applications in which the failure of any product would cause interruption of work or loss of profits, such as industrial, shipboard or military equipment.

THIS LIMITED WARRANTY IS IN LIEU OF ALL OTHER WARRANTIES EXPRESSED OR IMPLIED. THE WARRANTIES PROVIDED HEREIN ARE BUYER'S SOLE REMEDIES. IN NO EVENT SHALL FURAXA, INC. BE LIABLE FOR DIRECT, SPECIAL, INDIRECT, INCIDENTAL OR CONSEQUENTIAL DAMAGES SUFFERED OR INCURRED AS A RESULT OF THE USE OF, OR INABILITY TO USE THESE PRODUCTS. THIS LIMITATION OF LIABILITY REMAINS IN FORCE EVEN IF FURAXA, INC IS INFORMED OF THE POSSIBILITY OF SUCH DAMAGES.

Some states do not allow the exclusion or limitation of incidental or consequential damages, so the above limitation and exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

2. Model Descriptions

The SYNTH300 series of Direct Digital Synthesizer (DDS) boards are complete low noise precision synthesizers on a single PCI bus card. Designed for low jitter operation in communications, industrial and scientific applications these boards function in PCI bus systems using supplied drivers for Windows Vista™, or XP™ or 2000™. In addition to dual synthesizers, the SYNTH300-TRIG board has two TTL inputs which can be used to dynamically switch between up to four frequency profiles, for a variety of dynamic waveform synthesis applications (frequency/phase modulation, triggered quadrature frequency sweeping, etc.).

All models support a single LVDS output channel (option LVDSX2 supports two such channels), driven by the filtered output(s) of the DDSs, and include additional firmware to allow advanced functionality such as externally triggered frequency sweeping capability by use of the ModA and ModB TTL inputs. Some models can accept an external system clock (nominally 1GHz) in addition to the on-board oscillator. Further information for each model is given below.

2.1 MODEL SYNTH300-TRIG

Model Synth300-TRIG is a Dual-channel DDS board with two independent Direct Digital Synthesizers, each separately programmable for any frequency between 300kHz and 300MHz in increments of approximately 0.233 Hz. Two TTL inputs (ModA and ModB) are used to select up to four frequency/phase profiles by hardware control. A single LVDS channel is supported on the LVDSB output connectors.

2.2 MODEL SYNTH300-TRIG (option LVDSX2)

Model Synth300-TRIG-LVDSX2 is a Dual-channel DDS board with two independent Direct Digital Synthesizers, each separately programmable for any frequency between 300kHz and 300MHz in increments of approximately 0.233 Hz. Instead of the two TTL inputs (ModA and ModB) supported under the standard version, the LVDSX2 option supports a second LVDS channel (LVDSA).

2.3 MODEL SYNTH300-TRIG (option LVDSX2+TTL)

Model Synth300-TRIG-LVDSX2 is a Dual-channel DDS board with two independent Direct Digital Synthesizers, each separately programmable for any frequency between 300kHz and 300MHz in increments of approximately 0.233 Hz. Instead of the two TTL inputs (ModA and ModB) supported under the standard version, the LVDSX2 option supports a second LVDS channel (LVDSA). However, the TTL inputs can be used by altering the default configuration as discussed in section 4. Additionally option LVDSX2+TTL has a TTL output that runs at the same frequency as the DDS A output (default) or alternately the 125MHz output from DDS A, controlled by a software function call. **The option LVDSX2+TTL board does not have the external clock input/output.**

2.4 MODEL SYNTH300-TRIG-HS

Model Synth300-TRIG-HS is similar to the Synth300-TRIG model, but incorporates a high stability SAW stabilized oscillator. This a Dual-channel DDS board with two independent Direct Digital Synthesizers, each separately programmable for any frequency between 300kHz and 300MHz in increments of approximately 0.233 Hz. **The option HS board does not support the external clock input.**

3. Specifications

3.1 Synthesizer outputs (Synth300-TRIG, Synth300-TRIG-HS)

Number of Output Channels:	2 independently programmable analog outputs
Analog output voltage (into 50 ohms):	446mV RMS (6dBm) +/-10% from 2-200MHz 446mV RMS +/-20% from 100kHz-300MHz
Output impedance:	50 ohms 10pF
Output connectors:	SMA connectors.

Analog Output Specifications:

Wideband SFDR (DC to Nyquist):	
40MHz Fout	TBD
100MHz Fout	TBD
180MHz Fout	TBD
Narrowband SFDR:	
40MHz Fout (+- 1 MHz)	TBD
40MHz Fout (+- 50 kHz)	TBD
100MHz Fout (+- 1 MHz)	TBD
100MHz Fout (+- 50 kHz)	TBD
180MHz Fout (+- 1 MHz)	TBD
180MHz Fout (+- 50 kHz)	TBD
Output Phase Noise (103MHz lout):	
@ 1kHz Offset	TBD
@ 10kHz Offset	TBD
@ 100kHz Offset	TBD

3.2 Frequency stability

Absolute frequency accuracy	+/-0.02% of specified frequency +/-0.233Hz
-----------------------------	--

3.3 Phase Resolution

Each DDS has 14-bit phase resolution, allowing fine phase adjustment of each DDS core. Further, the Synth300-TRIG boards incorporate dedicated circuitry and firmware to synchronize the two DDS cores at 1GHz, allowing advanced functionality such as synchronous frequency sweeping of two sinusoids with a fixed phase relationship (e.g. quadrature).

3.4 TRIG/MODA and TRIG/MODB Input/Output

The TRIG/MODA and TRIG/MODB SMA connectors can be configured as profile selection inputs (Synth300-TRIG model), or as outputs (Synth300-TRIG-LVDSX2), depending on the jumper configuration described in section 4 of this document.

Connectors: 2 SMA connectors.

When MODA/MODB are configured as input ports (MODA/MODB function):

Number of TTL Input lines: 2, Standard TTL ($V_{OI} < 0.8V$, $V_{ih} > 2.4V$)

When MODA/MODB are configured as an LVDS output port (LVDSX2 option only):

Number of LVDS Output ports: 1, Standard differential LVDS
 LVDS output driver type: TI SN65LVDS100DGK

When MODB is configured as a 125 MHz or SynthA output:

Number of TTL Output lines: 1, Standard TTL ($V_{OI} < 0.8V$, $V_{ih} > 2.4V$)

3.5 LVDSB+ and LVDSB- Outputs

Output connectors: 2 SMA connectors.
 Number of LVDS Output ports: 1, Standard differential LVDS
 LVDS output driver type: TI SN65LVDS100DGK

3.6 EXT CLK Input or 1GHz CLK Output (see Fig 3.1)

Connectors: SMA connector.

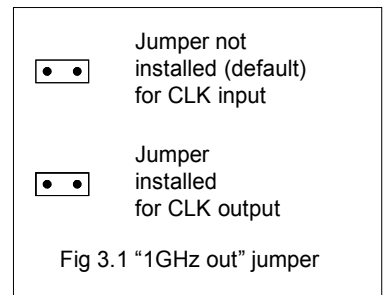
When configured as a CLK input (External CLK mode)

Number of Input lines: 1, ($0.2V_{p-p} < V_{clk} < 1.1V_{p-p}$)
 ($200MHz < f_{clk} < 1GHz$)

When configured as a CLK output (On-board 1GHz clock)

Number of Output lines: 1, (@200mVp-p)

NOTE: EXT CLK Input / 1GHz CLK Output is not available on models Synth300-TRIG-LVDSX2+TTL and Synth300-TRIG-HS.



3.7 General

Operating Temperature Range: 0 to +55 Degrees Celsius

Storage Temperature Range: -25 to +85 Degrees Celsius

Power Requirements:

SYNTH300-TRIG: +5V +/-5% at 0.3A Maximum
 +3.3V +/-5% at 1.5A Maximum

3.8 Physical

SYNTH300 boards are half size 32-bit PCI bus boards, which will operate in either 64-bit or 32-bit 33MHz or 66MHz PCI slots with either 5V or 3.3V signalling environment. The figure below shows the locations of the SMA signal output and digital I/O connectors, and LED indicators.

FURAXA

To avoid overheating, all SYNTH300 boards **must be installed a in well-cooled workstation, PC or server chassis.**

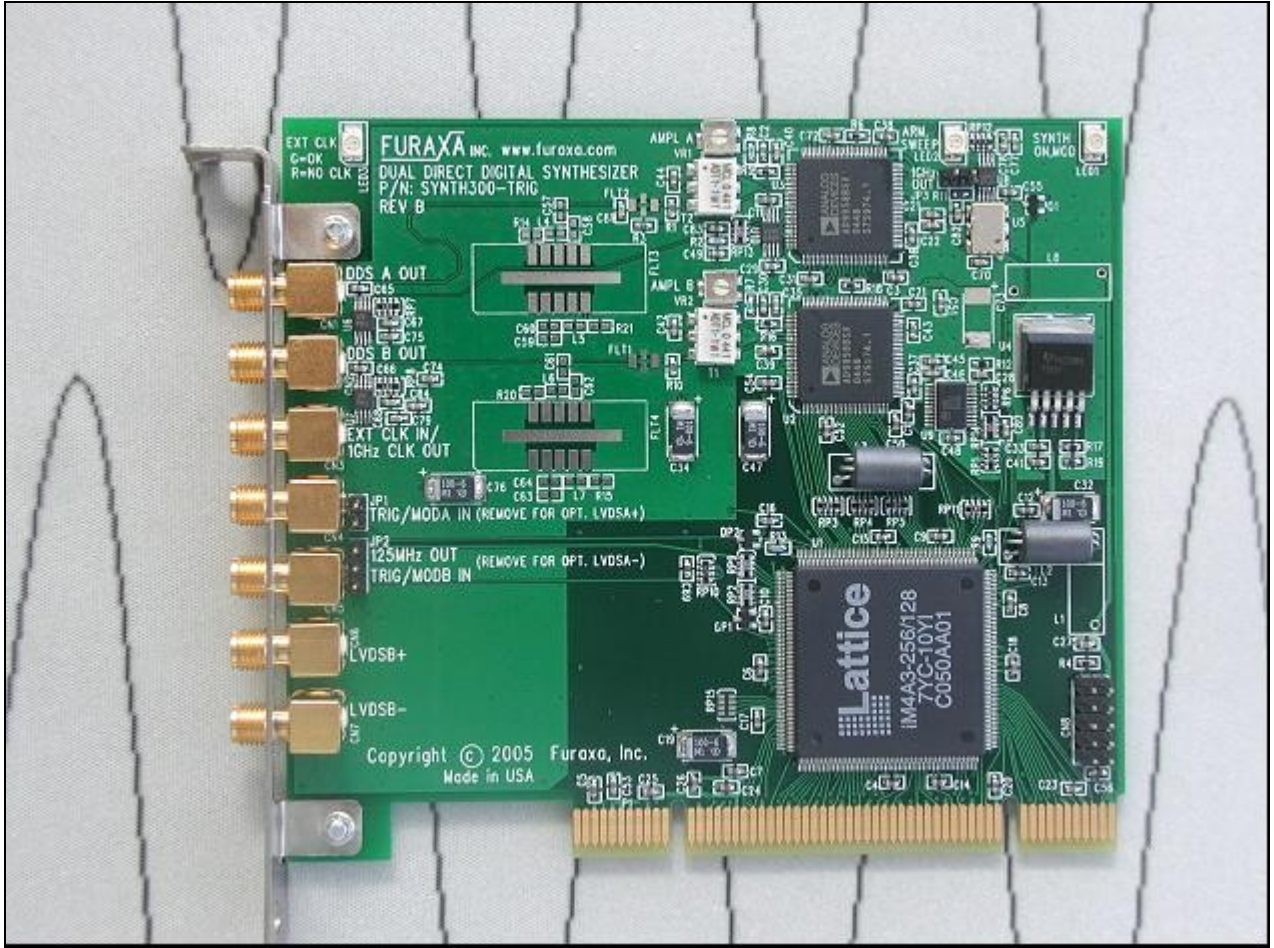


Figure 3.2 SYNTH300-TRIG PCI Board (LVDSX2 option pictured)

4. Hardware Architecture

SYNTH300 series boards are comprised of a digital section and an analog section. The digital section includes a high speed programmable logic device which implements the PCI interface and embedded firmware for controlling the DDS cores of the analog section. The analog section contains a 1GHz low-jitter clock, and two single-chip Direct Digital Synthesizer (DDS) ICs. The current outputs of each DDS are fed to a broadband RF transformer and 300MHz low-pass filter.

4.1 Synthesizer A & B Outputs (DDS A OUT, DDS B OUT)

The two analog synthesizer outputs are provided on SMA connectors with amplitude (nominally 200mV RMS into 50 ohms). The frequency, phase, frequency sweep rate (if specified), and other parameters are software programmable. A wide range of 50 ohm coaxial attenuators, such as those available from Mini-Circuits (Brooklyn, NY) may be used in conjunction with the board.

4.2 LVDS Outputs LVDSB+, LVDSB-

The two LVDSB output lines are driven by a TI SN65LVDS100DGK LVDS driver, which receives the output generated by DDSB, after filtering.

4.3 TRIG/MODA and TRIG/MODB I/O Lines

The MODA and MODB jacks are used as profile selection inputs, or ARM/SWEEP control inputs, in standard operation (boards without the LVDSX2 option). If the MODB input is not required for profile selection, it may be alternatively jumpered so as to provide a 125MHz clock output or a TTL version of the SynthA output, as shown below.

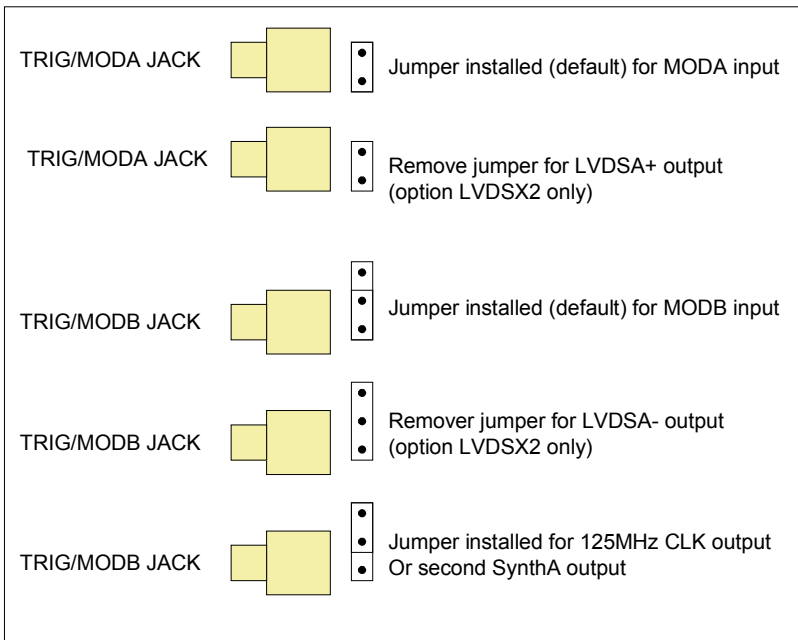


Figure 4.1 ModA and ModB Input / Output jack jumper configuration

In boards with the LVDSX2 option, the TTL TRIG/MODA & TRIG/MODB jumpers are removed, and the TRIG/MODA and TRIG/MODB jacks are used for the LVDSA+ and LVDSA- outputs, respectively. This removes the ability of these jacks to be used for external modulation capability. If the user desires to use either one of the TTL modulation or ARM/SWEEP inputs on an LVDSX2 device the user should:

FURAXA

- 1) Connect the internal push-on connector (connected to C79, in the case of boards with the HS clock option, in which an external clock is not used) to the bottom pin (the pin closest to the PCI interface) on the header for either MODA or MODB (JP1 or JP2), depending on which modulation input is desired (see modulation input section). Boards with both the HS option and the LVDSX2 option are factory-default-configured with this push-on connector connected to the TRIG/MODB jumper, thereby enabling the “EXT CLK IN” SMA jack to control the TRIG/MODB input.
- 2) Connect the TTL modulation input to the “EXT CLK IN” SMA jack. In this case, jumper JP3 (1GHz clock out) **must NOT** be installed.
- 3) If the second TTL modulation input is also required, it may be connected to the bottom pin of the other of jumper JP1 or JP2.

Alternatively, the LVDSX2 option may be removed completely, if not needed, by completely unsoldering IC U6, and installing the appropriate jumpers on JP1 and JP2, to allow the TRIG/MODA and TRIG/MODB jacks to be used as TTL modulation inputs.

4.4 LED Indicators

There are three Red/Green LEDs on the top edge of the SYNTH300 board that are useful during system integration for monitoring the board status. The functions of the LEDs are outlined below.

4.4.1 Ext Clk LED

For board models in which an external clock input is available, when the board is configured for an external system clock (see section 9.3) the Ext Clk LED illuminates, and is used to indicate the status of the externally supplied clock, as follows:

RED: Indicates insufficient clock frequency. Clock frequency is < 200MHz
GREEN: Indicates healthy clock frequency. Clock frequency is > 235MHz

If the clock frequency is between 200 and 235 MHz, the LED will partially light GREEN and partially light RED, appearing as Orange or Yellow, indicating marginal clock frequency.

4.4.2 ARM Sweep LED

This LED is useful when using externally controlled frequency sweeping operation, with the set_arm_write function enabled. Refer to the supplied example program (example.cpp) for further information regarding this application. If external modulation is selected and the Set Arm Write (refer to section 9.3) capability is enabled, the LED will indicate the following:

GREEN: The board is ARMED for frequency sweeping (sweep not active)
RED: The board is performing a frequency sweep (sweep active)
OFF: The board is in a post-sweep dwell (sweep not active)
Mixed (yellow): The board is in a pre-sweep dwell (sweep not active)

NOTE: ARM/Sweep operation is not available when CN4 and CN5 are used for LVDSA outputs.

4.4.3 Synth On LED

This LED is useful to determine overall board operation/configuration, as follows:

GREEN: Board is powered but not configured
RED: Board is configured using software profile selection (EXT_MOD=FALSE)
Mixed (yellow): Board is in EXTERNAL MODULATION mode and MODA input != MODB input
OFF: Board is in EXTERNAL MODULATION mode and MODA input = MODB input

5. Hardware Installation and Setup

Before you begin, **be sure your system has at least 256MB installed RAM**. To avoid overheating, the SYNTH300 **must be installed in a well-cooled workstation or server chassis, or alternatively in an industrial chassis PC**. Installation in a standard desktop PC **without fans at the front end of the card cage may cause the SYNTH300 to overheat**, and resulting damage is not covered by warranty.

1. Use the shutdown command on your system and then turn OFF the power to the system.

BEFORE REMOVING THE COMPUTER SYSTEM COVER OR REMOVING ANY BOARD, BE SURE THAT THE POWER TO THE COMPUTER, AS WELL AS TO ALL PERIPHERAL DEVICES IS OFF. WEAR A STATIC-DISSIPATING WRISTBAND WHICH IS GROUNDED TO THE SYSTEM CHASSIS WHILE OPENING OR WORKING ON YOUR SYSTEM.

2. Remove any screws that attach the computer system cover and remove the cover.
3. Remove the filler bracket from the PCI bus slot into which you wish to install your SYNTH300 board. If a mixture of 5V and 3.3V slots are available in the system, **choose a 32-bit 5V slot as your first preference**. If that is not available, install it in a 32 or 64-bit 3.3V slot. For details, refer to the hardware manual for your computer system.
4. Hold the SYNTH300 board by the top of the metal PCI bracket. Then hook the tab on the bottom edge of the SYNTH300's metal bracket into the corresponding slot in the computer's rear panel. Carefully push the SYNTH300 down so its PCI bus connector mates with the PCI bus connector on the motherboard. Be sure that the SYNTH300 is seated firmly into the motherboard PCI bus connector. Check that no other PCI boards have become unseated when the SYNTH300 was installed, as motherboards may flex slightly when installing PCI boards.
5. Plug coaxial I/O cables for the analog outputs into the appropriate SMA connectors on the SYNTH300's rear bracket at the rear of the system. Please refer to the diagram on page 8 of this manual.
6. We recommend that synthesizer output channels A and B (the 1st and 2nd SMA connectors from the bracket top) initially be connected to an oscilloscope set for 200mV/div., so you can observe the two synthesized waveforms operation when running the example software.
7. Replace the computer system cover, installing all screws you had removed. Reconnect the power cables to the system and peripherals.
8. Power up and reboot the system. The system will then be ready for software installation.

6. Software Installation and Setup

6.1 Software Package Contents

The software package shipped with each Synth300 series board includes the necessary driver, a ready-to-use GUI for simple synthesizer applications, and a Microsoft Visual Studio v6.0 project containing all the code necessary, to allow rapid development of powerful custom applications.

6.1.1 Driver

The driver shipped with the software package is included in the root of the distribution media. This driver will work for any Win32 application, whether it is a console application or not. Installation of the driver is facilitated by the associated INF file.

6.1.2 The Example Program

The source code and project files for the example program are contained in the folder titled "Example Program". The project files are associated with Microsoft Visual Studio version 6.0. The project builds a simple Win32 console application (example.exe) that performs basic configuration of an attached Synth300 device and initiates single tone generation. Modification of this program can be achieved by editing example.cpp and rebuilding the application. A discussion of this example program and the procedure for programming a Synth300 device is included in section 9. Additional information is contained in the C code comments.

6.1.3 The Source Code

The source code contained in the "API Object Source Code" folder contains the code necessary for developing a Synth300 custom application in a development environment other than Microsoft Visual Studio. An explanation of the object-oriented API is included in section 9. Additional information is contained in the C code comments.

6.2 Software Installation for Windows 2000 or Windows XP™

When you reboot your system for the first time with the Synth300 board properly installed, you may see a "Found New Hardware Wizard" prompt. This Wizard will copy the driver for the device onto the user's hard disk and register the device with the Windows Device Manager.

6.2.1 Windows Vista™ or XP™

The first step of the "Found New Hardware Wizard" prompts the user to select from one of two methods for installing the drivers for the newly found "PCI Data Acquisition and Signal Processing Controller". The two options are as follows:

- Install the Software Automatically (recommended)
- Install from List or Specific Location (advanced)

Insert the diskette distributed with the Synth300 device. The default option in the Wizard is to "Install the Software Automatically", leave this as the selected option and press "NEXT". The wizard will search for an appropriate driver for the Synth300 (this may take several minutes).

FURAXA

Once the Wizard has located the driver it will prompt the user that the driver has not passed the Windows Logo Test, and is not digitally signed. Select "Continue Anyway" to proceed with configuration of your new hardware. Once the Wizard has finished the configuration process the user will be prompted to select "Finish" to complete the installation. Pressing "Finish" will complete the software installation for the Synth300 device.

In some instances the user's copy of Windows XP™ may be configured to never allow the installation of unsigned drivers. In order change this configuration option follow these steps:

1. Click on "Start" on the taskbar.
2. Right Click on "My Computer".
3. Select "Properties" from the pop-up list.
4. The "System Properties" dialog box will appear. Select the "Hardware" tab.
5. In the "Drivers" section select the "Driver Signing" button. The "Driver Signing Options" dialog box will appear.
6. Change the default action to "Ignore" and click "OK".

Having made this change, restart the machine and repeat the installation.

6.2.2 Windows 2000™

The first step prompts the user to install a device driver for a hardware device. Insert the diskette distributed with the Synth300 device and hit "next" to continue. The next screen prompts the user to select from one of two methods for searching for the driver. Choose the default option, "Search For a Suitable Driver" and hit "Next". At the next screen (options for search) hit "Next". Allow the search to complete. Once the Wizard has located the driver it will prompt the user that the driver has not passed the Windows Logo Test, and is not "Digitally Signed". Select "Continue Anyway" to proceed with configuration of your new hardware. Once the Wizard has finished the configuration process the user will be prompted to select "Finish" to complete the installation. Pressing "Finish" will complete the software installation for the Synth300 device.

6.2.3 Continuing with the software installation in Vista, 2000 and XP

Once the driver has been successfully installed, the GUI distributed with the Synth300 (Synth300.exe) can be executed from anywhere on the user's hard-disk to control the configuration and operation of the device. Initially this file can be executed directly from the root of the installation diskette.

7. Running Synth300 programs

The Synth300 series is supported under Windows 2000/XP. A GUI program is provided (Synth300.exe) for Windows 2000/XP users as well.

7.1 Using the Synth300 series Under Windows 2000/XP

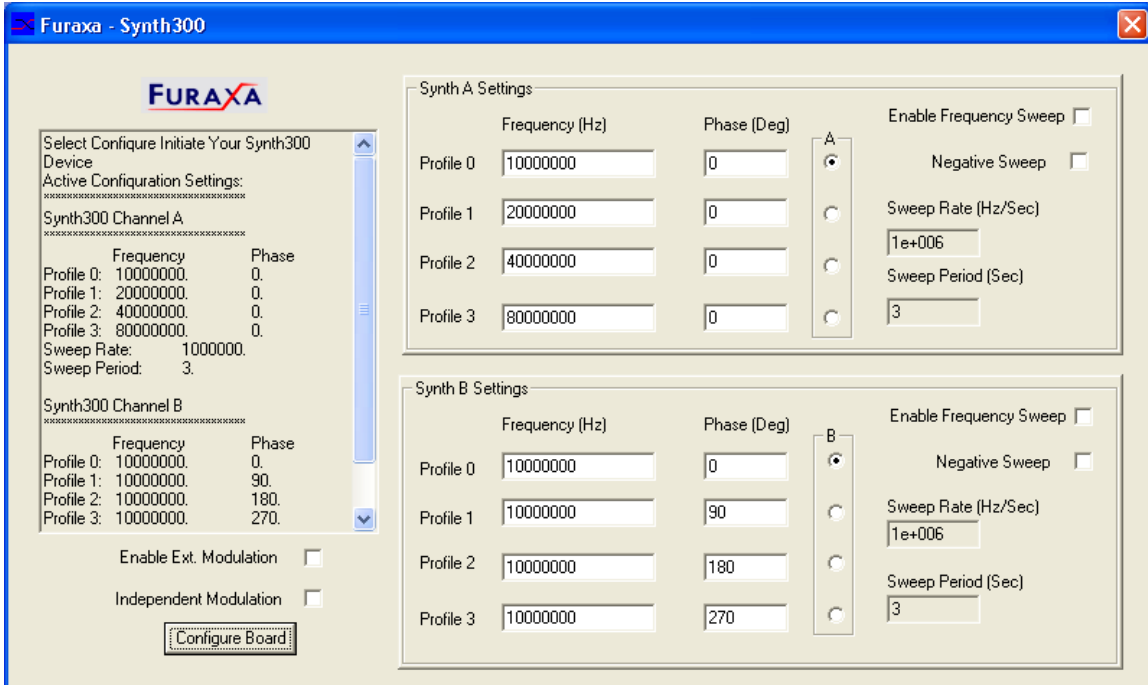
Running the Synth300 board under the Windows 2000/XP environment allows the use of a highly functional Graphical User Interface (GUI) program, Synth300.exe. This program, outlined further in section 8 of this document allows the user to specify each of the frequency and phase profiles for each synthesizer, and select the active profile easily. Additionally, frequency sweeping capabilities can be enabled and specified, and the board can be set for the two modes of external modulation. This program will allow the user to access most, but not all, of the features of the Synth300 board.

After selecting fields and entering the settings, the user simply clicks on the “Configure Board” button to program the Synth300. ***It is important to remember that no changes take effect until the user selects “Configure Board”.***

If a higher level of control is desired, the user may modify the C code provided in the example program directory (example.dsw is a MS Visual Studio project containing all files necessary to create custom user programs). The user library is provided in simple to use function calls, and the compiled driver is supplied. Please refer to section 9 of this document for further detail.

8. Using the Synth300 GUI under Windows XP or Windows 2000

Synth300.exe is a simple-to-use User Interface program for use under Windows XP and Windows 2000. Double-clicking on the Synth300.exe icon will bring up the Furaxa Synth300 window below:



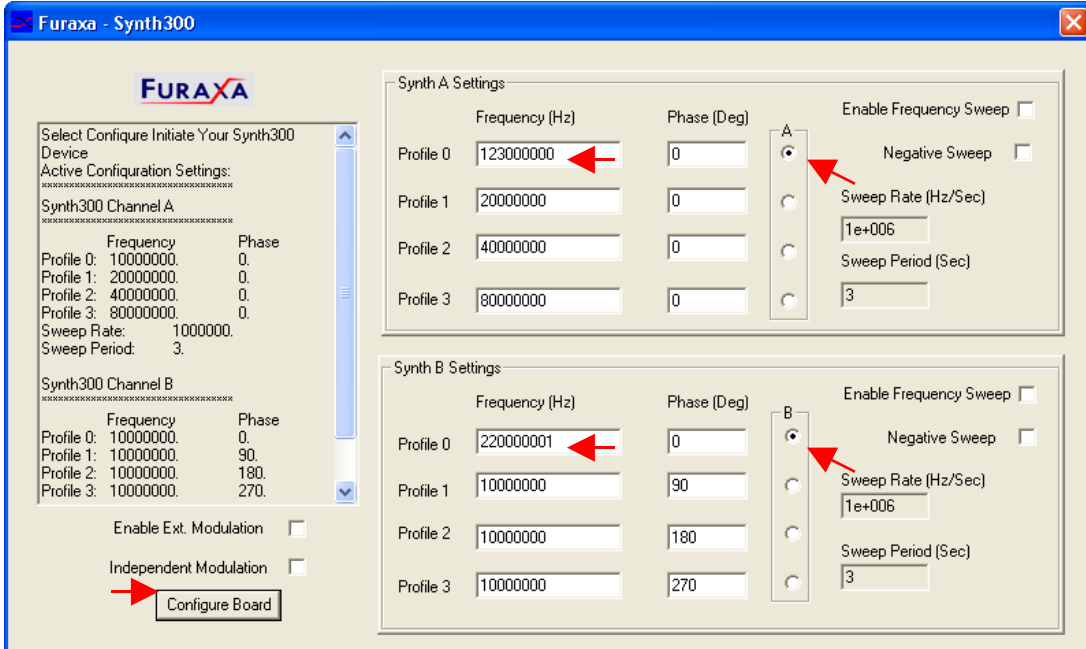
The window has four sections. The upper right section controls the settings for Synthesizer A. The lower right section controls the settings for Synthesizer B. The upper left hand window shows the configuration that has most recently been written to the synthesizer board initiating the current settings and resulting outputs. Finally, the two check boxes at the bottom left control external modulation choices.

For a given use, the user simply enters the desired frequencies and phases for each synthesizer, (and frequency sweeping and/or external modulation if desired) and then clicks on “Configure Board”. ***It is important to note that no change will occur until the user clicks on the “Configure Board” button (or hits “return”).***

A few examples of common applications follow:

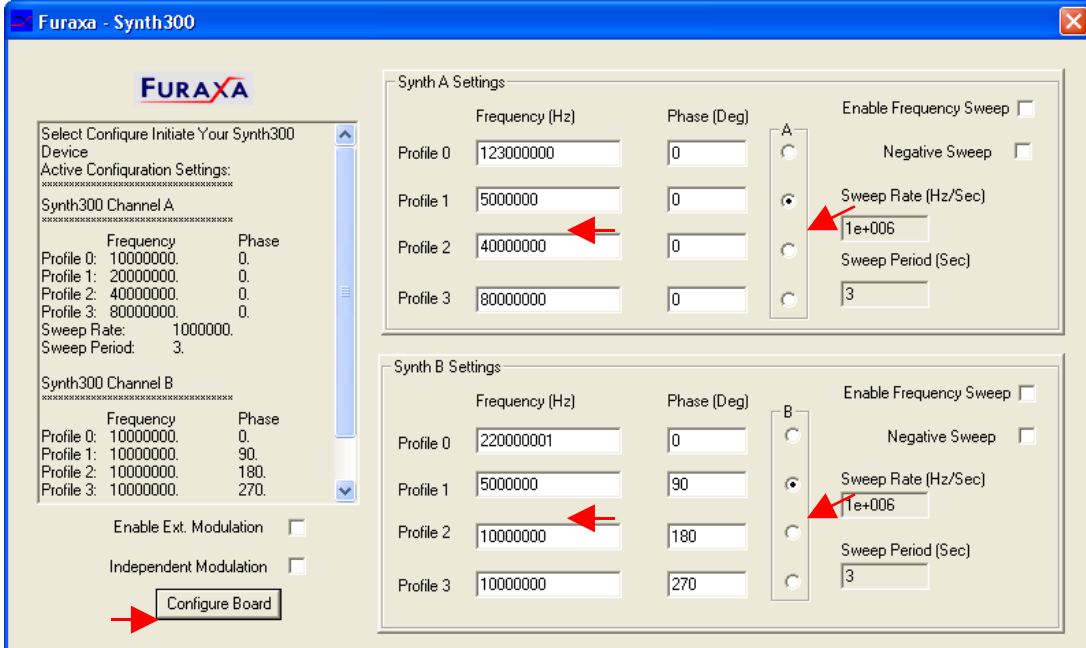
8.1 Example #1 – Two independent sinusoids

123.000000 MHz is generated on Synth A, and 220.000001 MHz on Synth B. Notice that “Profile 0” is selected on each synthesizer by use of the bullets to the right of the frequency and phase settings. Press “Configure Board” to accept these settings.



8.2 Example #2 – Phase-locked sinusoids

5.000000 MHz is generated on Synth A, and 5.000000MHz on Synth B with a 90 degree phase offset. Profile 1 was used for both synthesizers, while retaining the Profile 0 settings. Each synthesizer has four independent frequency and phase profiles.

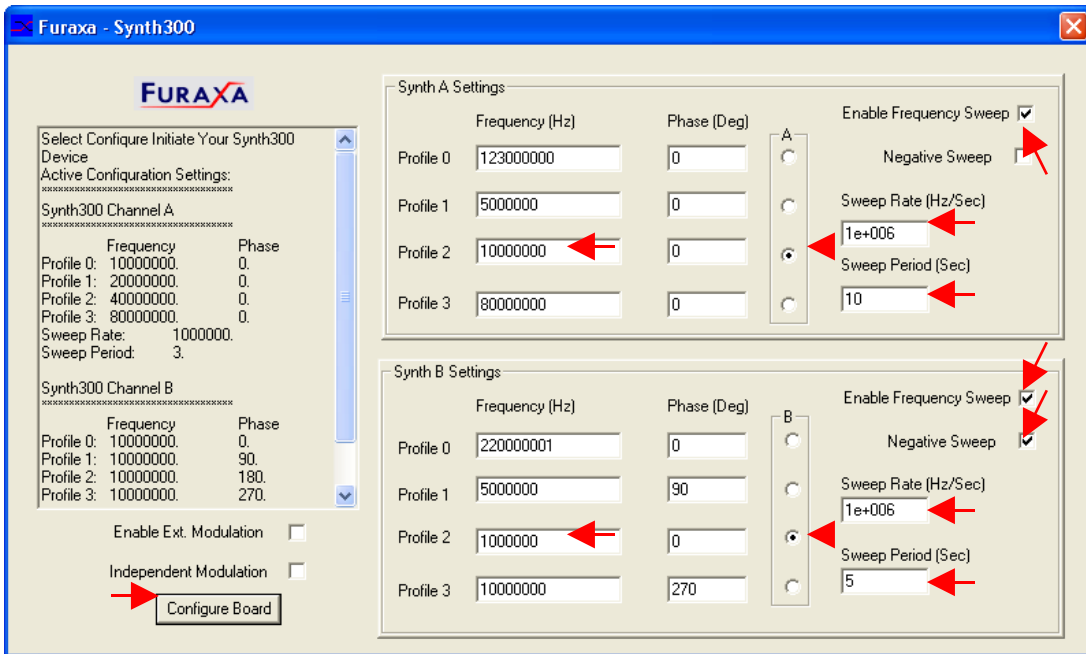


8.3 Example #3 – Two fully independent swept sinusoids

Synthesize two swept sinusoids, one with increasing frequency and the other with decreasing frequency.

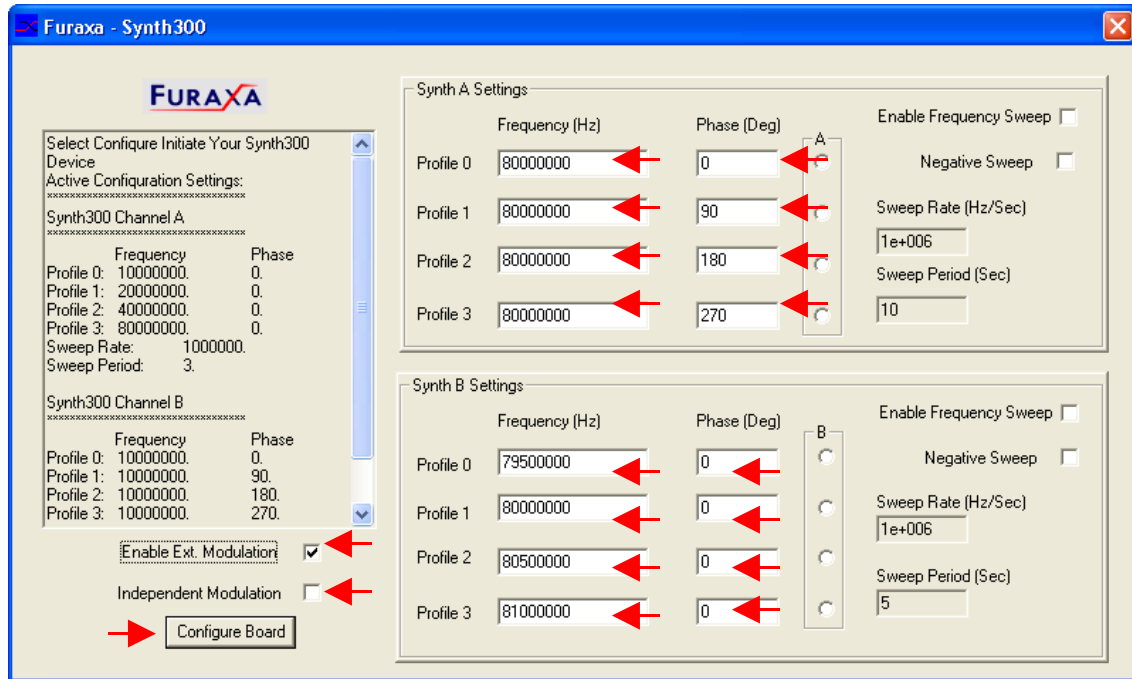
- 10.000000 MHz from Synth A, increasing at 1MHz/sec (Sweep Rate = 1e+006 hz/sec) for 10 seconds (Sweep Period = 10 Sec). Then repeating
- 10.000000MHz from Synth B, decreasing (Negative Sweep Checked) at 1MHz/sec (Sweep Rate = 1e+006 hz/sec) for 5 seconds (Sweep Period = 5 Sec) Then repeating

Notice the “Enable Frequency Sweep” box is checked for each synthesizer, and the corresponding settings configured.



8.4 Example #4 – External Modulation example

Externally modulated QPSK (4-PSK Phase Shift Key) signal (IF = 80MHz) on Synthesizer A. Externally modulated 4-FSK signal (f = 79.5, 80, 80.5, and 81MHz) on Synthesizer B. In this case the four profiles are set for the corresponding desired frequency/phase relationships for the two synthesizers.



Notice that **enabling** external modulation **disables** the user control of the profile selection, as profile selection is now controlled by the externally supplied TTL signals on Trigger Modulation Control A and Trigger Modulation Control B (Refer to Figure 3.2 in Section 3 for physical location of ModA and ModB input jacks).

If External Modulation is checked the TTL signals on Trigger Modulation Control A and Trigger Modulation Control B (synchronized to the on-board 125 MHz clock) will control the selection of frequency/phase profiles for each of the synthesizers (positive logic: TTL_High = logic '1', TTL_Low = logic '0'). The state of the "Independent Modulation" box governs the profile selection as given in Tables 8.1 and 8.2.

Table 8.1 JOINT mode ("Independent modulation" box left unchecked)

Trigger Modulation Control A	Trigger Modulation Control B	Synth A Profile	Synth B Profile
0	0	0	0
0	1	1	1
1	0	2	2
1	1	3	3

Table 8.2 INDEPENDENT mode ("Independent modulation" box checked)

Trigger Modulation Control A	Trigger Modulation Control B	Synth A Profile	Synth B Profile
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	1

9. Custom Application Programming of the Synth300

Powerful custom applications (beyond the basic functions incorporated in the GUI) for the Synth300 device can be quickly programmed with source code provided on the software installation diskette. The easiest method for creating a custom application is to modify the supplied software project, example.dsw, which is contained in the “Example Program” folder. This project is a Microsoft Visual Studio v6.0 project and can be immediately built using Microsoft Visual C++. The main example program, example.cpp, uses the CSynthAPI object (described below) in order to configure a Synth300 device, resulting in the executable example.exe. Modifications to this program are easily made by calling the API functions discussed in section 9.3.

In the event that the user wishes to use a development environment other than Microsoft Visual Studio, the files included in the “API Object Source Code” folder should be included into their project as explained in section 9.2.

The suggested procedure for creating a custom application is to open the example program project (example.dsw) in Microsoft Visual Studio, examine example.cpp and build the application using the functions discussed in section 9.3. Note that for convenience the generic code at the beginning and end of example.cpp is identified (in comments) and need not be modified. The application specific code is also identified and the programmer would modify this section as required.

Synth300 Operational Overview:

The Synth300 device uses a combination of AD9858 DDS-core implemented and firmware (CPLD-implemented) functionality. This combination results in an augmentation of the basic DDS core functions to enable capability such as triggered frequency hopping, frequency dwell/sweep/dwell capability, and other advanced capability.

In programming the board, one must remember an important difference between the CPLD and DDS core functional partition. Any CPLD-implemented function takes effect immediately upon the corresponding write to the board. However, DDS-core implemented functions take effect only after a qualifying “update event” as described below. This allows the user to configure a number of parameters in the DDS cores before an “update event” occurs, to allow event-initiated behavior such as triggered frequency/phase-hopping, dwell/sweep/dwell sinusoids, and other functionality. **To differentiate between these two types of function behavior, functions in SynthAPI.cpp that require a subsequent “update event” begin with “Enter_” whereas functions that do not require a subsequent “update event” start with something other than “Enter_” such as “Set_” .**

“Update Event”:

As described above, the DDS cores of the Synth300 device require an “update event” in order for any programmed changes following the most recent “update event” to take effect. Thus a programmer can configure the DDS cores for one type of operation while the current configuration is active and running. This is particularly important for advanced operation, such as use of the Set_Arm_Write function described below.

The DDS cores recognize the following as qualifying “update events”, some of which depend on the state of the CPLD-supported External_Modulation bit (described below).

- 1) Calling the Update() function (described below) will cause an “update event” for both DDSs, regardless of the state of the Internal/External Modulation bit.

FURAXA

- 2) In internal (Ext_Mod bit set to FALSE) modulation (profile select) changing to a new profile will cause an “update event”. Note, in the software function Set_Active_Profile an explicit call to the Update() function is included to avoid confusion caused by changes that may not cause a profile change of one DDS.
- 3) In External Modulation (Ext_Mod bit set to TRUE) any change to either of the 2 hardware (TTL-level) inputs will cause a qualifying update event for the DDS(s) that are affected by that/those bits.

9.1 Using the CSynthAPI Object

The API used to configure a Synth300 device is defined in SynthAPI.h as a C++ object called CSynthAPI. The CSynthAPI object provides a useful abstraction to the programmer. The collection of functions that define the API allow full access to the capabilities of a Synth300 device without the need to understand the low level software interface to the control registers on the Synth300. An explanation of the low level software interface is included in section 10 for completeness. However, most programmers will have no reason to read section 10.

9.2 Instantiating a CSynthAPI Object

In order to exploit the functionality provided by the CSynthAPI object the programmer must include the appropriate files into their project and instantiate a CSynthAPI object. The files that must be added to the programmer’s project are located in the folder “Synth300 API Object Source Code”. All the files in this folder must be placed in the project directory and the files must be added to the project. (If the programmer is using the MS Visual Studio v6.0 project “example.dsw” these files are already added to the project). These files are as follows:

Devioctl.H
Synth300_Driver.h
Synth300_IO.h
SynthAPI.cpp
SynthAPI.h
SynthDefines.h
Guids.h
DeviceList.h
DeviceList.cpp

Further description of the function of each file is included as comments inside each file. Once the files are added to the project, include "SynthAPI.h" to the main source code file for the project and instantiate a CSynthAPI object. An example of this is as follows:

```
#include "SynthAPI.h"  
CSynthAPI m_SynthAPI;
```

9.3 CSynthAPI Object Member Functions

As mentioned above the member functions of the CSynthAPI object provide the means for connecting to the driver service and configuring a Synth300 device. The following is a brief explanation of these functions. Most functions include a BOOLEAN “verbose_flag” variable which will cause information to be output to the screen when set to TRUE. This is useful during software integration and can be suppressed (set to FALSE) if desired for release versions. Further explanation of each of the member functions is given within the code itself (example.cpp and SynthAPI.cpp).

The definitions of the two functions used to connect to the driver service are shown below. The functions return FALSE if there is an error opening or closing the handle associated with connection to the driver. OpenSynth300Board should be the first function called in an application. Similarly, CloseSynth300Board should be the last function called, as it releases the handle.

BOOLEAN OpenSynth300Board(BOOLEAN verbose_flag);
BOOLEAN CloseSynth300Board(BOOLEAN verbose_flag);

ResetSynth300Board function prepares the hardware to accept configuration information and initializes variables. Further, the synch_flag variable will cause a the DDS synchronization procedure (at 1GHz) to be called. For initiation settings read the code and commenting in SynthAPI.cpp. *This function should be called immediately after the clock source is chosen (Set_Ext_Clk function).*

void ResetSynth300Board(BOOLEAN synch_flag, BOOLEAN verbose_flag);

A valid “update event” is required for any function starting with “Enter_” to take effect. One method of generating a valid “update event” is through use of the Update function. For further information refer to the Update Event at the beginning of section 9 of this document and the AD9858 data sheet.

void Update();

The primary configuration of a Synth300 device is facilitated by the Enter_Synth_Profile function. Each DDS has 4 separate freq/phase profiles, and this function sets the corresponding profile in the corresponding Synth (DDS) with the given frequency and phase passed to the function. SynthA is targeted by setting the first parameter to FALSE, and SynthB is targeted by setting the first parameter to TRUE. A profile within the targeted synth is targeted by setting the second argument to either 0, 1, 2, or 3. The frequency for the corresponding synth profile is specified as a decimal number (frequencies less than 300KHz or greater than 300Mhz can be programmed, but are not supported by the analog output hardware). The phase for the corresponding synth profile is specified as a decimal number between 0.0 and 360.0 degrees. See the implementation of this function in SynthAPI.cpp for a more detailed look at how it is implemented.

void Enter_Synth_Profile(BOOLEAN SynthB_flag, int Profile, double Freq, double Phase, BOOLEAN verbose_flag);

FURAXA

If configured for Internal Modulation (Ext_Mod bit set to FALSE) the Active profile for each synth is determined by the parameters passed to the Set_Active_Profiles function shown below. Each of the arguments in the function is set to 0,1,2 or 3. Note, this call will only set the internal profile selection, and if the External Modulation bit is currently TRUE these internal settings will not take effect until the Ext_Mod bit is set to FALSE.

void Set_Active_Profiles(UINT ProfileA, UINT ProfileB, BOOLEAN verbose_flag);

At Reset, the board is set to Internal Modulation (External_Modulation bit set to FALSE). Enabling External Modulation is accomplished by setting the External_Modulation bit to TRUE. External Modulation is discussed in section 8.4. A value of FALSE will disable the external modulation capability. Note that changing the Ext_Mod bit can cause an “update event” if the external TTL bits (profile) are set differently than the internal (software controlled) bits (profile). This (usually) unwanted behavior can be avoided by ensuring that the TTL bits are set to the same profile selection as the internal bits before the Ext_Mod bit is changed.

void Set_Ext_Mod(BOOLEAN Ext_Mod, BOOLEAN verbose_flag);

Each DDS has the ability to autoclear any combination of frequency and phase accumulators upon an “update event”. By setting the corresponding frequency or phase autoclear bit, the programmer can ensure frequency and phase accumulator zeroing upon receipt of an “update event”. Setting the corresponding BOOLEAN to TRUE will cause an autoclear upon an “update event” for (respectively) SynthA_Freq, SynthA_Phase, SynthB_Freq, SynthB_phase accumulators.

void Enter_Synth_AutoClear(BOOLEAN A_f_auto, BOOLEAN A_p_auto, BOOLEAN B_f_auto, BOOLEAN B_p_auto, BOOLEAN verbose_flag);

Frequency sweeping capability involves two parts. The first part of setting up a frequency sweep is to enter the desired sweep rate parameters to the corresponding DDS. This is done by use of the Enter_Sweep_Rate function. The first argument identifies which DDS is targeted (FALSE = SynthA, TRUE = SynthB).

The second and third arguments specify the sweep rate to the DDS core. Care must be taken to design the sweep properly, as there are tradeoffs in the sweep design. The second argument specifies the increment to the frequency accumulator. It is a signed number, allowing for negative sweeping. The Third variable FreqRampRate specifies the number of 125MHz clock cycles between each update to the frequency accumulator. It is an unsigned number, with a minimum value of 1. In general, one desires to use the minimum FreqRampRate word possible while satisfying the sweep rate in order to have the least granularity in the resulting swept waveform. Further information and guidance is incorporated in the code SynthAPI.cpp and in the AD9858 datasheet.

Note that a separate function call to Enter_Synth_Sweeping is required to enable the frequency sweep.

void Enter_Sweep_Rate(BOOLEAN SynthB_flag, UINT FreqStep, UINT FreqRampRate, BOOLEAN verbose_flag);

The second part of setting up a frequency sweep is to enable the frequency sweeping capability on the DDS. The programmer can, if desired, set up a frequency sweep using the above

FURAXA

Enter_Sweep_Rate function, and wait to enable it by leaving the corresponding DDS frequency sweep disabled (such an approach is often used in conjunction with the Set_Arm_Write command for dwell_f1/sweep/dwell_f2 applications). In order to enable sweeping, the frequency sweep enable bit must be set in the corresponding DDS core. This is accomplished through the use of the Enter_Synth_Sweeping function. Setting an argument to TRUE will cause the corresponding DDS to start a frequency sweep upon the next “update event” with the sweep rate settings entered by the preceding Enter_Sweep_Rate function.

void Enter_Synth_Sweeping(BOOLEAN SweepA, BOOLEAN SweepB, BOOLEAN verbose_flag);

Some applications require very fast setup of the DDS cores, specifically the frequency sweep bit (address 0x01, bit 7 of DDS cores, see Section 10.2.9 table) and the frequency and phase autoclear bits (address 0x02 bit 7 and bit 6 of DDS cores, see Section 10.2.9 table). To allow additional capability, the Synth300 CPLD incorporates firmware to do the following, but only when External Modulation (Ext_Mod bit set to TRUE).

Given both Ext_Mod bit and arm_bit are set to TRUE, upon a change one of the external TTL signals, the CPLD will perform a write to the corresponding address of both DDSs as given below (see table in section 10.2.9 for address and word definitions). These writes will not conflict, so that both TTL signals on ModA and ModB input jacks can change simultaneously.
Trig/Mod B (TTL input) Low -> High transition: CPLD writes 0x80 to address 0x01
Trig/Mod B (TTL input) High -> Low transition: CPLD writes 0x00 to address 0x01
Trig/Mod A (TTL input) Low -> High transition: CPLD writes 0xC0 to address 0x02
Trig/Mod A (TTL input) High -> Low transition: CPLD writes 0x00 to address 0x02

Upon examination, the user will see that a TTL transition effectively acts as both the update event for the settings prior to the transition and the signal to “prime” the corresponding function (frequency sweep enable/disable or autoclear enable/disable) to take effect upon the next update event. The “primed” functionality DOES NOT take effect at the TTL transition that caused the priming, but rather at the following update event (often the next TTL transition). Though a bit conceptually cumbersome, this allows very fast triggered behavior (125 MHz trigger synchronization). Note that the DDS profile selection will also change when either of the TTL signals changes, so that care must be exercised in setting up the corresponding profiles (in software) so that the desired behavior is given at all transitions.

This function should only be used once the software has “handed off” profile selection to hardware control (Ext_Mod bit set TRUE), and usually is the last function call before the CloseSynth300Board call.

For example, if one wished to create an application in which the board started at 100MHz and dwelled until a certain event, then swept downward at a given rate until another event, then dwelled at the lower frequency until another event, with the possibility of repeating the sequence, one would in software:

Open the Synth300 board
Select Internal or External clock
Reset and Synch the Synth300 board
Enter all profiles to 100MHz, 0 phase.
Enter Synth Autoclears all FALSE.
Enter desired Sweep parameters
Enter Synth_Sweeping as FALSE
Set Active Profiles at (0,0).
Ensure TTL inputs are both Low (for glitchless internal to external profile handoff).
Set_External_Modulation to TRUE

FURAXA

Set_Arm_Write to TRUE
Close the Synth300 Board

Now the TTL signals would traverse the states (ModA TTL input, ModB TTL input):
0, 0: Entry state, receives profile control from software when Ext_Mod set to TRUE
0, 1: DDS primed to start sweeping, DDS uses profile 1 settings.
0, 0: DDS begins sweeping, DDS primed to stop sweeping, DDS uses profile 0 settings.
1, 0: DDS stops sweeping, DDS primed for autoclear, DDS uses profile 2 settings
0, 0: DDS autocleared, DDS primed for no autoclear, DDS uses profile 0 settings (Initial state)

void Set_Arm_Write(BOOLEAN arm_bit, BOOLEAN verbose_flag);

The user can select to use the on-board 1GHz clock (default condition at reset) or, on certain models, and externally supplied clock. By setting ext_clk to TRUE, the external clock input will be used to clock both DDS cores. Refer to section 4.4.1 regarding LED indications of external clock condition reporting. *This function is usually called immediately after opening the board.* On models Synth300-TRIG-HS and Synth300-TRIG-LVDSX2+TTL, which do not support an external clock feature, Set_Ext_Clk must never be set to specify an external clock

void Set_Ext_Clk (BOOLEAN ext_clk, BOOLEAN verbose_flag);

Given proper jumper configuration of the MODB output jack (see figure 4.1) the user can select to output the 125 MHz clock or the SynthA output. By setting TTL_125Select to TRUE, the output will be the 125 MHz clock. By setting the TTL_125Select bit to FALSE (default) the output will be the SynthA output.

void Set_TTL125_Output (BOOLEAN TTL_125Select, BOOLEAN verbose_flag);

COMMUNICATION PRIMITIVES

The primitive function used in reading from the Synth300 board is given below. Most users will not use this primitive as the envisioned functionality for the Synth300 has been incorporated in the SynthAPI functions. It is listed here for completeness and for advanced users who may wish to modify or augment the existing SynthAPI.cpp code.

ReadSynth300BoardRegister(PULONG pRegValue)

The primitive function used in communicating to the Synth300 board is given below. Most users will not use this primitive as the envisioned functionality for the Synth300 has been incorporated in the SynthAPI functions. Upon examination of the member functions in SynthAPI.cpp, the programmer will notice this function used extensively. It is listed here for completeness and for advanced users who may wish to modify or augment the existing SynthAPI.cpp code.

BOOLEAN SetSynth300BoardRegister(IN HANDLE fileHandle, IN WORD which_parameter, IN WORD setting);

10. Low Level Software Interface (Most Users need not read)

The SYNTH300 board is easy to communicate with. In most cases, this section may be skipped, as the driver supplied with the board will automatically handle all communication with the board registers. **The best way to develop your own custom software is simply to modify the included example program, example.cpp, using the functions from SynthAPI.cpp and then rebuild the application.** However, the following section gives an overview of how the driver calls control the board.

The software interface consists of a PCI type-00 Configuration Header and a Control Register..

Accesses to Control and Auxiliary registers must be made as 32-bit transfers.

10.1 PCI Configuration Header

SYNTH300 series boards support a PCI Configuration Header, whose map is shown below..

Double Word Address	byte 3	byte 2	byte 1	byte 0
00 H	Device ID		Vendor ID	
04 H	Status		Command	
08 H	Class Code			Revision ID
0C H		Header Type	Latency Timer	
10 H	Base Address for memory-mapped control register			
14 H				
18 H				
1C H				
20 H				
24 H				
28 H				
2C H				
30 H				
34 H				
38 H				
3C H	Max Lat (=01 H)	Min Gnt (=01 H)	Interrupt Pin	Interrupt Line

The board control register is mapped at **configuration space address 80H**, as well as in **memory space at Base Address + 1FFFFFFC**. The Auxiliary register is mapped at **configuration space address 8CH**, and in **memory space at Base Address + 1FFFFFF4**. Accessing either place will read or write these registers. .

Double Word Address	byte 3	byte 2	byte 1	byte 0
80 H	SYNTH300 Control Register			
84 H	Auxiliary Register			

10.2 SYNTH300 Control Register

The SYNTH300 Control Register is used to configure the synthesizer frequencies and modulation options and to start and stop the synthesis process. The table below shows the usage of the SYNTH300 Control Register, and these bits' functions are outlined in the sections which follow. The first table shows the function of the Control Register during a **write**. The register will not contain meaningful data when read back. The Control Register is never directly written by user programs, but is modified by calls to the driver, which are each summarized in the discussion of the respective bit.

Bit	Function
31	Reserved for future use
30	Reserved for future use
29	Reserved for future use
28	Unreset (unresets synthesizer ICs)
27	Output TTL_125
26	Update both Synths
25	Synthesizer A
24	Profile Select [0,1,2, or 3]
23	Synthesizer B
22	Profile Select [0,1,2,or 3]
21	Reserved for future use
20	External Modulation Mode
19	Enable External modulation
18	Reserved for future use
17	Arm_Write
16	External_Clk
15	Write is to Synth B
14	Write is to Synth A
13..8	Synthesizer IC register address[5..0]
7..0	Synthesizer IC data word [7..0]

Function of Control Register bits during **write**.

10.2.1 Unreset (write only)

The Unreset bit must be set to 1 before writing to the registers on the synthesizer ICs.

When this bit is driven to 0, all registers in the two synthesizer ICs are set to their default values of 0x00, with the exception of address 0x00 which is set to 0x18.

10.2.2 Output TTL_125 (write only)

With the appropriate jumper selection (see section 3.4), when this bit is set to 1, the output will be a TTL signal derived from the 125MHz clock of Synthesizer A. When set to a 0, the output will be a TTL signal derived from the Synthesizer A output.

10.2.3 Update (write only)

This bit is used to transfer the internal buffered register contents to the register memory of the synthesizers. Thus, after a series of writes to the respective addresses, setting this bit from a 0 to a 1 (remember to set it back to 0 for any further updates) will load the data into the working

registers of the synthesizers. Changing the state of the a synthesizer profile will also cause an update for the respective synthesizer.

10.2.4 Synthesizer A Profile Select (write only)

If the **Enable External Modulation** bit (described below) is 0, this 2-bit number is used to select the profile for Synthesizer A. The user selects the profile by a single write to parameter ULTRAD_SYNTH_A_PROFILE. If the Enable External Modulation bit is set to a 1, refer to table 5.2 for explanation of the profile selection for Synthesizer A.

10.2.5 Synthesizer B Profile Select (write only)

If the **Enable External Modulation** bit (described below) is 0, this 2-bit number is used to select the profile for Synthesizer B. The user selects the profile by a single write to parameter ULTRAD_SYNTH_B_PROFILE. If the Enable External Modulation bit is set to a 1, refer to table 5.2 for explanation of the profile selection for Synthesizer B.

10.2.6 External Modulation Mode (write only)

If external modulation is enabled, this bit selects between two external modulation modes. In the first mode (bit set to 0), named **JOINT** modulation, the 2 external TTL signals determine the 2-bit profile selection for each synthesizer. In the second mode (bit set to 1), named **INDEPENDENT** modulation, each of the 2 external TTL signals controls the LSB of each synthesizer's profile selection, with the second bit controlled by software. If external modulation is disabled, this bit is ignored.

If External Modulation is enabled, and this bit is set to a 0, the TTL signals (synchronized to the on-board 125 MHz clock) on Trigger Modulation Control A and Trigger Modulation Control B will control the selection of frequency/phase profiles for each of the synthesizers (JOINT mode). Trigger Modulation Control A will determine the MSB of the 2-bit profile number, and Trigger Modulation Control B will determine the LSB of the 2-bit profile number (positive logic: TTL_High = logic '1', TTL_Low = logic '0') for each synthesizer as shown below in Table 9.2.5.1

Table 9.2.5.1 JOINT mode ("Independent modulation" box left unchecked)

Trigger Modulation Control A	Trigger Modulation Control B	Synth A Profile	Synth B Profile
0	0	0	0
0	1	1	1
1	0	2	2
1	1	3	3

If this bit is set to a 1, the MSB for the profile selection for each synthesizer should be set to 0. The TTL signal on MOD_A will control the LSB of the 2-bit profile selection for Synthesizer A, while the TTL signal on MOD_B will control the LSB of the 2-bit profile selection for Synthesizer B (INDEPENDENT mode). With this setup the modulation will be as shown in Table 9.2.5.2

Table 9.2.5.2 INDEPENDENT mode ("Independent modulation" box checked)

Trigger Modulation Control A	Trigger Modulation Control B	Synth A Profile	Synth B Profile
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	1

10.2.7 Enable External Modulation (write only)

If this bit is set to 0, the values in bits 25 through 22 (Synthesizer A Profile Select and Synthesizer B Profile Select respectively) will control the selection of frequency/phase profiles for the synthesizers, regardless of the state of the External modulation inputs.

If this bit is set to 1, the TTL signals (synchronized to the on-board 125 MHz clock) on MOD_A and MOD_B will control the selection of frequency/phase profiles for the synthesizers, in the manner consistent with the setting of the External Modulation Mode bit described above.

10.2.8 Read_Synth_Select (write only)

This bit is currently not supported.

10.2.9 Arm_Write (write only)

When this bit is set to 1 (and the Ext_Mod bit is set to 1) upon a transition on either of the TTL input signals, the CPLD will write to both DDSs as follows:

Trig/Mod B (TTL input) Low -> High transition: CPLD writes 0x80 to address 0x01

Trig/Mod B (TTL input) High -> Low transition: CPLD writes 0x00 to address 0x01

Trig/Mod A (TTL input) Low -> High transition: CPLD writes 0xC0 to address 0x02

Trig/Mod A (TTL input) High -> Low transition: CPLD writes 0x00 to address 0x02

Writes to the two addresses, if any, are accomplished on opposite edges of the PCI clock to avoid data/address conflict.

10.2.10 External_clk (write only)

For board models in which an external clock input is available, when this bit is set to 1, the board will use the externally supplied clock. When this bit is set to 0 the board will use the on board 1GHz clock. On Model Synth300-TRIG-HS and Synth300-TRIG-LVDSX2+TTL, the External_clk bit must never be programmed to 1.

10.2.11 Write is to Synth B, Write is to Synth A (write only)

When the **Write is to Synth B** bit is a 1, then the **Sample IC Data Word** specified below will be written to synthesizer B. When the **Write is to Synth A** bit is a 1, then the **Sample IC Data Word** specified below will be written to synthesizer A. When both of these two bits are high, then the data will be written simultaneously to both synthesizer ICs. In any of these cases, the data in **Sample IC Data Word** will be written to the register specified by **Synthesizer IC Register Address** (discussed below). If both of these bits are 0, the data will not be written to either synthesizer.

10.2.12 Synthesizer IC Register Address (write only)

The value written to **Synthesizer IC Register Address [5..0]** specifies which of each synthesizer's 32 registers is to be written to with the value specified by the 8-bit data word **Synthesizer IC Data Word [7..0]** discussed below.

10.2.13 Sample IC Data Word (write only)

This 8-bit value is written to the register specified in **IC Register Address [5..0]**, to the synthesizer(s) enabled for the write by bits 15 and 14 (**Write to Synthesizer B** and **Write to Synthesizer A**, respectively). The function of the registers on each synthesizer is given by the table on the next page.

Register Name	Address	(MSB) Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	(LSB) Bit0	Default Value
Control Function Register 0	0x00	Not Used	Not Used	Leave 0	Leave 1	Leave 1	Power Down	Leave 0	Leave 0	0x18
Control Function Register 1	0x01	Freq Swp Enable	Leave 0	Leave 0	Leave 0	Leave 0	Leave 0	Leave 0	Leave 0	0x00
Control Function Register 2	0x02	<i>Freq Accum Autoclear</i>	<i>Phase Accum Autoclear</i>	Leave 0	Leave 0	Leave 0	Not Used	Leave 0	Leave 0	0x00
Control Function Register 3	0x03	Leave 0	Leave 0	Leave 0	Leave 0	Leave 0	Leave 0	Leave 0	Leave 0	0x00
Δf Tuning Word	0x04	Delta Frequency Word [7:0]								-
	0x05	Delta Frequency Word [15:8]								-
	0x06	Delta Frequency Word [23:16]								-
	0x07	Delta Frequency Word [31:24]								-
Δf Ramp Rate	0x08	Delta Frequency Word [7:0]								-
	0x09	Delta Frequency Word [15:8]								-
Frequency Tuning Word 0	0x0A	Frequency Tuning Word for Profile_0 [7:0]								-
	0x0B	Frequency Tuning Word for Profile_0 [15:8]								0x00
	0x0C	Frequency Tuning Word for Profile_0 [23:16]								0x00
	0x0D	Frequency Tuning Word for Profile_0 [31:24]								0x00
Phase Offset Word 0	0x0E	Phase Offset Word for Profile_0 [7:0]								0x00
	0x0F	Phase Offset Word for Profile_0 [13:8] (2 MSBs Not Used)								0x00
Frequency Tuning Word 1	0x10	Frequency Tuning Word for Profile_1 [7:0]								-
	0x11	Frequency Tuning Word for Profile_1 [15:8]								-
	0x12	Frequency Tuning Word for Profile_1 [23:16]								-
	0x13	Frequency Tuning Word for Profile_1 [31:24]								-
Phase Offset Word 1	0x14	Phase Offset Word for Profile_1 [7:0]								-
	0x15	Phase Offset Word for Profile_1 [13:8] (2 MSBs Not Used)								-
Frequency Tuning Word 2	0x16	Frequency Tuning Word for Profile_2 [7:0]								-
	0x17	Frequency Tuning Word for Profile_2 [15:8]								-
	0x18	Frequency Tuning Word for Profile_2 [23:16]								-
	0x19	Frequency Tuning Word for Profile_2 [31:24]								-
Phase Offset Word 2	0x1A	Phase Offset Word for Profile_2 [7:0]								-
	0x1B	Phase Offset Word for Profile_2 [13:8] (2 MSBs Not Used)								-
Frequency Tuning Word 3	0x1C	Frequency Tuning Word for Profile_3 [7:0]								-
	0x1D	Frequency Tuning Word for Profile_3 [15:8]								-
	0x1E	Frequency Tuning Word for Profile_3 [23:16]								-
	0x1F	Frequency Tuning Word for Profile_3 [31:24]								-
Phase Offset Word 3	0x20	Phase Offset Word for Profile_3 [7:0]								-
	0x21	Phase Offset Word for Profile_3 [13:8] (2 MSBs Not Used)								-
Reserved	0x22	Reserved Do Not Write; Leave at 0xFF								0xFF
	0x23	Reserved Do Not Write; Leave at 0xFF								0xFF

10.3 Auxiliary register

Reserved for future use.

10.4 The AD9858

The analog outputs are created by two Analog Devices AD9858 synthesizer devices. In the event that the user would like to learn more about this device refer to the datasheet on the Analog Devices website.