General Note on the DDS card and on the BUS driver

Etienne Maréchal and Fabrice Wiotte
Laboratoire de Physique des Lasers, UMR 7538 CNRS,
Université Paris Nord, 99 Avenue J.-B. Clément, 93430 Villetaneuse, France
(Dated: May 15, 2014)

This note is a memo on a new electronic card that use DDS components, and that is controlled by computer using a data BUS. These programmable RF sources will be implemented on the new cold atoms experiment to drive the AOMs of the experiment. These RF sources are connected to a single computer using a data BUS, and are programmed using the BUS. To control the cards, we have also developed a simple Labview interface and we will also present briefly this simple interface. After explaining the motivations, we present the BUS architecture, then the DDS and its different functions. We then give details on the card schematic and on the PCB. We will give some test on the system that has been realized and the one that we have implemented and that we have tested now. We try also to review future possible extensions to other cards. The main motivation on this project is to construct a control system for the experiment (cold atoms or complex experiments) that allow to integrate many functions with a user friendly interface. With this memo, we will try to identify future improvements, and also which new devices could be also implemented based on this BUS architecture.

INTRODUCTION AND MOTIVATIONS

The control of the Chromium experiment is mainly done with two kinds of electronics cards : Digital cards that control TTL levels, and Analog Output cards that allow to tune the voltage controlling different devices. (On the Chromium experiment, the Digital Card is the DIO 64 card by View Point System, providing 64 digital output with $1/(10MHz) = 100ns$ maximum resolution, and we use also two National Instrument cards having 8 analog outputs, with 12 bits resolution , with +10/-10 maximum amplitude). These cards allow to control with a good synchronization the different devices of the experiments like RF switch, currents sources, Shutters, AOMs (frequency, and RF amplitude, RF fast switch)...

The most used device on the experiment are AOMs modulators. AOMs are corner stones for all the cold atoms experiments, and they are often very numerous and have different purpose. They have to be controlled with good RF driver sources. The frequency, the amplitude of the drivers is very often changed during the experimental sequence. On the chromium experiment, the majority of the build RF drivers are analog RF sources, using a VCO for the control of the frequency and the amplitude (using a variable attenuator?). These sources are easy to use and are robust, but they have a few drawbacks : the possibility to tune the frequency using an external voltage is also a source of frequency fluctuation as any variation of the voltage control will change the frequency of the AOM. The VCO frequency itself is compared with a PLL with a local quartz oscillator whose frequency can also change with time. In addition, as the RF amplitude is controlled with an attenuator, for a total switching off of the RF, a RF switch has to be added. So, each AOM driver has to be controlled with three lines : one voltage control of the amplitude, one voltage control for the frequency, and one TTL to switch off quickly the RF ("RF switch"), so that the number of connections becomes very important, adding some complexity to the set-up.It gives to the system a low level of integration, and increase the probability to have a failure.

A more recent possibility to realize stable and versatile RF sources is to use a DDS component as the RF synthesizer. This component allows to numerically synthetize a precise RF frequency with a stable and programmable amplitude. Each DDS is controlled by an external reference clock, whose stability is transfered to the DDS. The functions seems the same that one of a VCO but the DDS has to be programmed, so that a single connection to the DDS allow to integrate the different functions. In addition, some DDS comes with many options, like frequency sweeps, and phase control. Modulation of the frequency or of the phase of the DDS is also often available for the component. DDS offer a modern way to realize very stable and versatile RF sources: their output can be very fast reprogrammed at a fix of variable frequency, with a given amplitude. Two DDS, if clocked by the same clock source, will be naturally coherent sources, which can be useful in some cases. (The core of a DDS is a counter (up to 48 digits), whose value is increment at each clock transition. The value of the counter is then transformed using an integrated sinus table, into an output voltage with a sinusoidal shape.)

What are the drawbacks of the DDS. The first drawback is that they have to be programmed, so that they need to be interfaced with a micro controller or with a computer connection. This is a first difficulty. For example, a servo loop on the frequency will be complicated to implement with a DDS, as the error signal has to be digitalized and

then the frequency has to be reprogrammed. A second drawback is that, as the signal is numerically generated, some numerical noise is added(seen on frequency analyzer as different unwanted side bands). For some applications, these spurious side bands can be a problem.

As the number of AOMs can be as large as 30 AOM on an experiment, it seems a good idea to be able to program the different DDS with a single computer. For that, we have decided to developed a BUS on which all the DDS are connected. Each DDS card has a specific address and is connected to the BUS, and can be program at any time (but not at the same time if they have a different address). This allow to have a very high level of integration of all the DDS of the experiment: a single connector will connect all the DDS, allowing to integrate many connections on a single BUS. This work is largely inspired by the work of Florian Schreck and Todd Meyra, that have developed an experiment control system relying on a BUS architecture that controls DDS drivers but also digital outputs and Analog Output card, all connected to the same BUS, and having different address. Florian Schreck has also shared the schematics of the devices that were developed for this purpose, on his group web page.

BUS ARCHITECTURE

Here we present the BUS architecture. We have used the same architecture than in the Florian Schreck group: the BUS is a parallel BUS, that has 25 lines that are transfered with a flat ribbon cable (this cable is also now existing in round section "normal" cable). The cable has 50 lines, so that ground lines and signal lines are alternated. This arrangement limits the cross talk between lines and also is a way to have 25 independent twisted pairs. The maximum frequency of such a cable is limited to maybe 20 MHz on a short length of typically one meter. For our application, we will use a frequency of 1 MHz for the data rate which should be enough for our need, and cable length of ten meters should work. In fact, this speed limitation comes here not from the BUS, bus from the computer card whose rate is limited to a few MHz, as explain bellow. With a short cable (50 cm) and simple instructions (up to 6 bytes for a new frequency programmation) on a single device, we have been able to push the BUS clock to 10 MHz. (It could be quite easily be measured.). To reach higher frequencies, an other method should be used. In that case, serial communications can allow faster rate.

The BUS architecture is the following: the 16 first lines will be interpreted as Data. The 8 following lines correspond to address lines meaning that in principle the BUS is able to address 256 devices. Finally the last line is the clock line, that control the data rate that are transmitted on the BUS. This signal a copy of the BUS clock signal that updates the data and the address on the BUS. This clock signal has to be present on the BUS: as will be expained in the card description, this signal is used to to generate a strobe signal that validate new data .

Before describing the DDS componet and the DDS card, we present the interface between the BUS and the computer.

DIO 32 HS computer card and interface card

The data on the BUS is controlled by a computer. The main task to be done is to be able to control, with an externally clocked rate, the 25 lines of the BUS. Different possibilities are possible.

We have chosen to use a National Instrument card , DIO 32 HS, that we had used on the Cesium experiment. This card is a little bit old but give enough lines (32) to control the 25 lines of the BUS. We have used this card for different reasons: we had it, this solution is the same one that the one develop in Florian Group, and this card from NI is easily programmed using Labview or other languages. Another possibility would be to use also 25 lines of the DIO 64 of viewpoint systems. It will remain enough logical lines to control the whole experiment, as the BUS function will replace many lines that were used. Other possibilities would be to use a microcontroler with a buffer which would be fed at the beginning of the experiment,with a USB or an Ethernet connection. This solution would be very interesting to developp, as in that case the BUS and the cards could be used by all the team of the lab, without having to have the DIO 32 card and a specific Labview programm.

For this first version, the 25 lines are controlled by a NI DIO 32 HS card. The maximum transfer rate of this card depends on the amount of data that are generated. For a continuous generation, I have test with a simple Labview program that the external clock could be 1 MHz Maximum. For short amount of data, the frequency clock can be up to 20MHz (verifier). Note that, the DIO 32 is able to generate 32 lines, that are grouped into 4 ports of 8 lines : PORT A, B, C and D. For the BUS, only A, B and C are used as explained below. Port D could be also used for something else. In the Florian group, this last address is used to add more address bus to the systems so that the clock signal can be direct on one BUS (corresponding to one 50 line Flat ribbon cable), or another cable).

The NI card comes with a 1 meter, very particular connector and can not be directly connected to the BUS. For that; we have developped a simple interface card that connect the DIO32 connector to a the BUS connector. To increase the signal quality we have used tri state buffers, that should increase the output quality of the DIO 32 HS card. A schematic and a picture of the BUFFER card is in annexe. A similar card was already developed in the Florian Schreck group.

A common part for all the cards: the adress recognition and the data validation

Lets speak first of the general architecture of the BUS, independently of the device that will be addressed by the 25 lines of the BUS. As already explained, the 25 lines are cut in 3 parts: the 16 first lines correspond to data lines (corresponding to PORT A and B if we use the DIO 32). The 8 following lines are used for the address recognition for the BUS (corresponding to PORT C of the DIO). And the last line number 25 corresponds is a copy of the clock signal, that updates the data of the BUS at each clock rising edge (in the case of the DIO 32 the A, B, C ports are update with the next value present in the BUFFER. Note that, if the BUFFER is empty, the PORT values are unchanged);

The main idea of the BUS is the following: all the devices are connected to the BUS, and if the address that is present on the BUS is equal to the device address, then the data will be accepted by the device and interpreted : the effect of the data will completely depend on the device. For a DDS for example, the frequency value can be change. It is very important that the device is affected only in the case that the address is correct: for that, we need to generate a strobe signal, that is used to tell that the data are stable and valid and can be read. The first task is then the address comparison between the device and the BUS. Once that the address is recognized by a device present on the BUS, data can be validate (so, a little bit after) and the device state is updated (the result will depend on the device). So a validation signal has to be generated, that is commonly referred in electronics as the strobe signal.

A first idea is simply to use the clock signal as strobe signal. The problem is that this signal is a little bit in advance with the data, as it is used to updated the data. In addition, after a few meters of travel, the logical front are distorted. So the idea is to wait a little bit before to look at the data value: for that, using the clock signal, a retarded signal , referred as a strobe signal is generated. The address comparison is performed only when the strobe signal is high, and if it is the case, the data are validated and passed to the component.

We will give detail of this function in the card description, but we want to stress here that these function; the address recognition and data validation; will be rather identical on all the different cards, not only on the DDS cards.

The AD9852 DDS component

The chosen component is the AD9852 by analog device (Same as the one used by Schreck, and also very much used in Syrtes). This device is quite impressive, as it allow many different possibiliies: It allow to have an external frequency clock of up to 300 MHZ. For DDS, it is recommended to perform a maximum frequency of maximum 40% of the clock frequency giving a frequency of 120 MHz. Note here that for our applications with cold atoms, it is common to use a AOM frequency higher than 120 MHZ. It is in that case possible to add a doubling or a quadrupling stage at the output of the DDS (with filters and also a small preampliication stage before the final RF amplifier. We will test this possibility later, using standard components manufactured by Minicircuits. Another possibility is to use component working at higher frequency, like the AD9858 or the more recent AD9912, supporting a clock frequency of 1 GHZ. The different devices have all some drawback and some advantages. We will test an other card using the AD9858 wich has an architecture very similar to the AD9852, so that the developpement done for the AD9852 can be reutilize.

Here we focus on the AD9852 device. This device is almost perfect for our application except that, the maximum frequency will be about 150 MHz, and that the power consumption can be as 4W (?), a very well designed heat sink is important for the component integrity. Particularly, the resolutions of all the parameters of this device is extremely high. The AD9852 can be programmed with a serial connection or with a parrallel connection using 8 bits data line and 5 bits for the register lines . This is this mode (parallel programmation) that we use to programm the DDS. If you look at a list a Analog device component, you will see that the majority of the DDS chips have only a serial programmation. Such a parallel programation allow a higher programation rate. The AD9852 comes with two output: one, the Cosine DAC is the normal output, consisting of a Sinus wave form which totally controlled parameters (Frequency, amplitude and phase) The second output, that we have not yet used and that is not directly available with the actual design, is a DAC output whose value can be change by programmation. As the device can

be programmable at 100 MHz rate, this output could be used as programmable arbitrary waveform generation. We will give more details on the first output, the Cosine DAC output: First, the frequency can be programmed with a resolution of 48 bits. The output frequency of the DAC output is given by the simple equation $f8out = M * f_{clock}/2^{48}$ where M is the frequency tuning word using 48 bits: at each clock tranistion, the 48 bits counter is increase by M. If the clock is perfectly stable, the resolution is equal to $f_{clock}/2^{48} = 1\mu Hz$ for a 300 MHz frequency clock. Second, the amplitude of the sinus is controlled with a 12 bits resolution (4096 levels). Finally, the phase of the sinus is also controlled with the same accuracy, 14 bits. Finally, the DDS can be programmed in different modes (5 modes)that can be useful.

- The first mode is the single tone mode. In this mode, the frequency, the amplitude and the phase can be programmed. In this mode, change of all these parameters can be done by programation. The maximum change rate is limited by the refreshing at a maximum rate of 100 MHz using the 8 byte parallel port (Only 10 MHz max with using the serial port)
- The second mode is the unramped FSK mode, FSK meaning Frequency Shift Keying. For this mode, two frequency f_1 and f_2(the two are programmed with two 48 bytes frequency word) have to be defined. The AD9852 allow to go from f_1 to f_2 : an abrupt jump (phase is always conserved), controlled by a single TTL signal on the FSK pin, that is available as an external SMA connector on the developed card.
- The third mode is also a FSK mode but a ramped FSK. As in the unramped FSK mode, f_1 and f_2 are defined. In addition, a signed delta frequency world is also defined, using also 48 bits. The delta frequency world is the frequency step, used to go from $f1$ to $f2$. In this mode, linear ramp up and down, with a programmable period can de programed. The time spend at each intermediate frequency is controlled with a 20 bits register. The frequency will go from F1 to F2 by controlling the FSK pin. Another very usefull possibility is a continuous triangle ramp between the two defined frequencies can be also programmed, without FSK control.
- Another control is the OSK pin, that allow to control the change of the amplitude by programmable steps. This can be use full if we need to switch on or off the RF output with a controllable time.

Another very nice feature of this device is that a clock multiplier is integrated, with value going from 4 to 20. Using a 15 MHz external clock we can then run the DDS at its 300 MHz maximal frequency clock. Note also that the DDS can be also over clocked. In that case, power dissipation is very critical and it is important to have a very efficient power dissipation. The data sheet of the device can be downloaded on the Analog Device site. The main characteristics, are in fact summarize in the register table that we also put in annexe of this note. These registers contains all the data that are used by the DDS. The programation of the DDS consist in addressing these registers with the desired value.

Description of the DDS card

We have given in Annexe details of the DDS electonic card that is connected to the Data BUS : the schematic, the Altium PCB and some pictures. The card is also mount in a standard instrumentation Rack that can receive up to 6 DDS cards (see also pictures). The schematic show that the BUS is connected to the card via BUFFER lines. Then

CPLD part

To perform this BUS task, we have decided to use CPLDs, that are programmable logical components instead of discrete standard BUS components like Buffers and logical components and counters. The idea is that this way of doing is very flexible so that, the connections between the device and the BUS can be programmed and easily optimized to guaranty a good transfer rate and a good immunity to noises. They also can be reprogrammed at will if certain options are need. The used component is a CPLD component. (complex programmable logic device). We use the following device: a small CPLD by Xilynx, XC2C64A. We will give details on that in the following. This component has 44 I/O that can be programmed as input or output. Two ones are necessary for the 25 lines of the bus.

A first CPLD is used to control the validation of the address and generates the strobe that will then validate the 16 data lines. In the particular case of the DDS, we use two bits of the adress bits A0 and A1 (see table on the

schematic) also allow to implement the different functions of the DDS. Note that due to these two bytes options, the real number of limited to 256/4=64 devices: It seems rather still enough for a complex experiment.

The DDS programmation is done by addressing the Ports with the Data. The bits data adress are D0 to D7, corresonding to the 8 first lines of the BUS. The ports adress (see table in annexe) goes from &00 to &27, (6 bits are necessary) and are programmed with ports A0 to A5 connections, correspondig of lines 8 to 13 . Two more lines are therefore unused. We have connected them to the FSK or to the OSK pin (line 14 and 15 of the BUS). The transfer between the BUS line and the DDS pins is done by the second CPLD which mainly acts as a three state buffer : the port and the data will be utpdated if and only if a strobe signal is recieved by the CPLD. If not, the value previous values are hold. The programation of the DDS, using the parallel connection, consist in updating the port adress line, and the data adress line and writing it to the component using the WR pin of the component (in fact a low to high transition is need for this validation). The DDS, once the date received may or may not change its output as two mode are available : we can use the IO update pin (using a low to high transition) to update the outputs or is is internally done. The advantage of using the IOupdate with an external connection is that the timing constaints are welle controled. For example after a reset, the frequency (48 bits) and amplitude can be written to the corresponding ports. Then, a single IO update will make the DDS change its output. This connection is possible ThenThis update is done by the second CPLD The DDS itself can work in two different mode : the most simple one is that the outputs will be update 'autimaticcaly' if the ports are changes. An other solution is to

We describe here the different options that can be used :

- If A0A1=00 then a master reset is send on the reset pin.for example if A0A1=00, a master reset is done, that allow to switch off in 1 clock cycle the RF output.
- If A1A1=01 then data will be written in the register adress, using the WR pin. The data will then be validate or not depending of the DDS selected option, that can be update with the IO update connection or using the internal clock (see block diagramm on the fist page of the AD9852 data sheet).
- If A0A1=10, the IO update pin is used to validate the data

In annexe, we give the CPLD code that is used and implement on the two CPL component. This code is adapted to the

DDS electronic

The DDS 9852 can have a consumption as high as 4W so that the power dissipation is critical.

In annexe, we give the CPLD code that is used and implement on the two CPL component. This code is adapted to the

Labview interface

We have programmed a simple labview interface that allow to programm a DDS card, using the BUS connected to the NI DIO 32 HS card. A picture of the programm is shown in annexe. Using this programm, we have verified that the DDS can be programmed with a clock BUS up to 10 MHz. This simple interface allows to control almost all the parameters of the DDS. We have not yest implemented the triangle option and the OSK possibilities.

Future developments

Using the same bus architechture, others card can be developped: we will try to test an other RF card using the AD9858 component. This component can be clock at 1GHz, giving the possibility to have generate souces up to 400 MHz. The connections are very close to the one of the 9852 (same parallel programmations). One difficulty is that this component has no integrated frequency multiplier so that a 1GHZ external clock has to be used as external clock. The functions and the resolutions are also generally not si good than the AD9852 (no triangle, amplitide resolution of 10 bits, no reset function....)

It seems also very interesting to develop a Digital to Analogous interface card, as these card are also very much used on the experiment. Using the address bus, a 16 bits resolution should be possible. For this Task, the CPLD

architecture could be easily reprogrammed. For example, we can imagine that we will use 3 bits of the address to control which of the 8 outputs phase to be update with a 16 bits resolution.

CMOS 300 MSPS Complete DDS

AD9852

FEATURES

- 300 MHz internal clock rate
- FSK, BPSK, PSK, chirp, AM operation
- Dual integrated 12-bit D/A converters
- Ultrahigh speed comparator, 3 ps rms jitter
- Excellent dynamic performance
  - 80 dB SFDR at 100 MHz (±1 MHz) A_{OUT}
- 4× to 20× programmable reference clock multiplier
- Dual 48-bit programmable frequency registers
- Dual 14-bit programmable phase offset registers
- 12-bit programmable amplitude modulation and on/off output shaped keying function
- Single-pin FSK and BPSK data interfaces
- PSK capability via I/O interface
- Linear or nonlinear FM chirp functions with single pin frequency hold function
- Frequency ramped FSK
- <25 ps rms total jitter in clock generator mode
- Automatic bidirectional frequency sweeping
- Sin(x)/x correction
- Simplified control interface
  - 10 MHz serial 2-wire or 3-wire SPI-compatible
  - 100 MHz parallel 8-bit programming
- 3.3 V single supply
- Multiple power-down functions
- Single-ended or differential input reference clock
- Small, 80-lead LQFP or TQFP with exposed pad

APPLICATIONS

- Agile LO frequency synthesis
- Programmable clock generator
- FM chirp source for radar and scanning systems
- Test and measurement equipment
- Commercial and amateur RF exciter

FUNCTIONAL BLOCK DIAGRAM

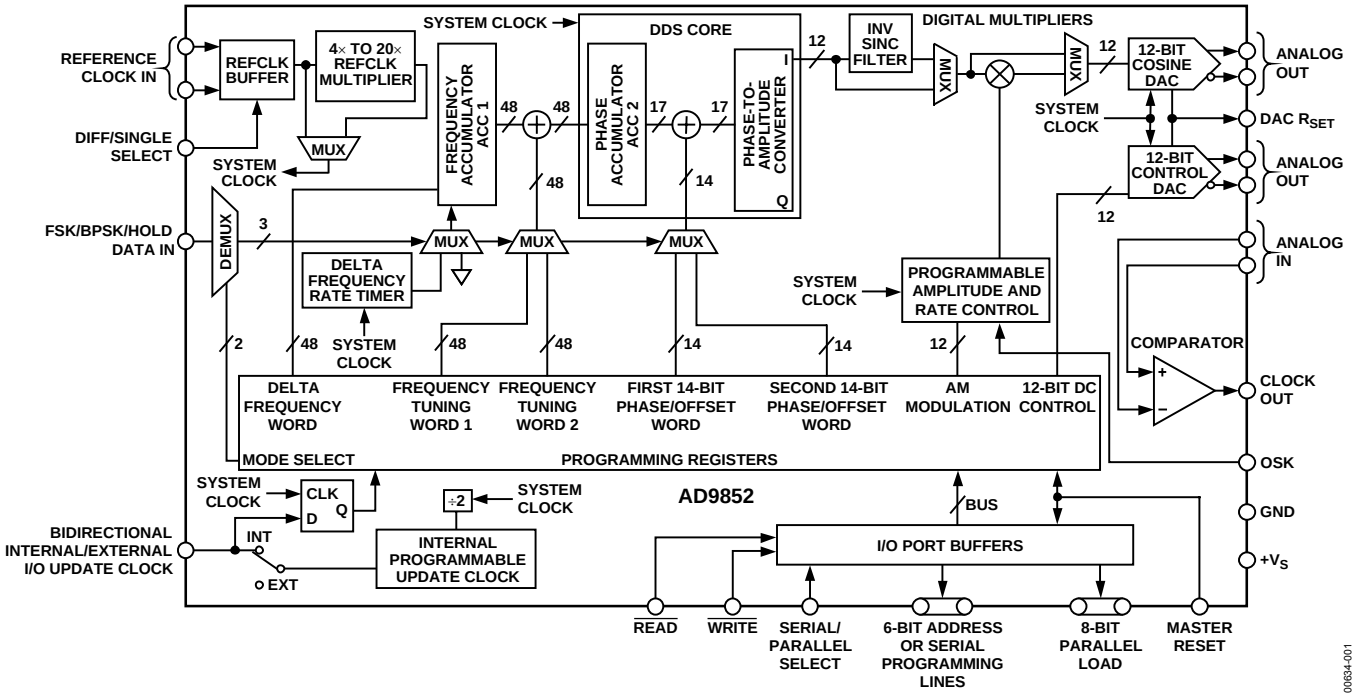


Figure 1.

Rev. E

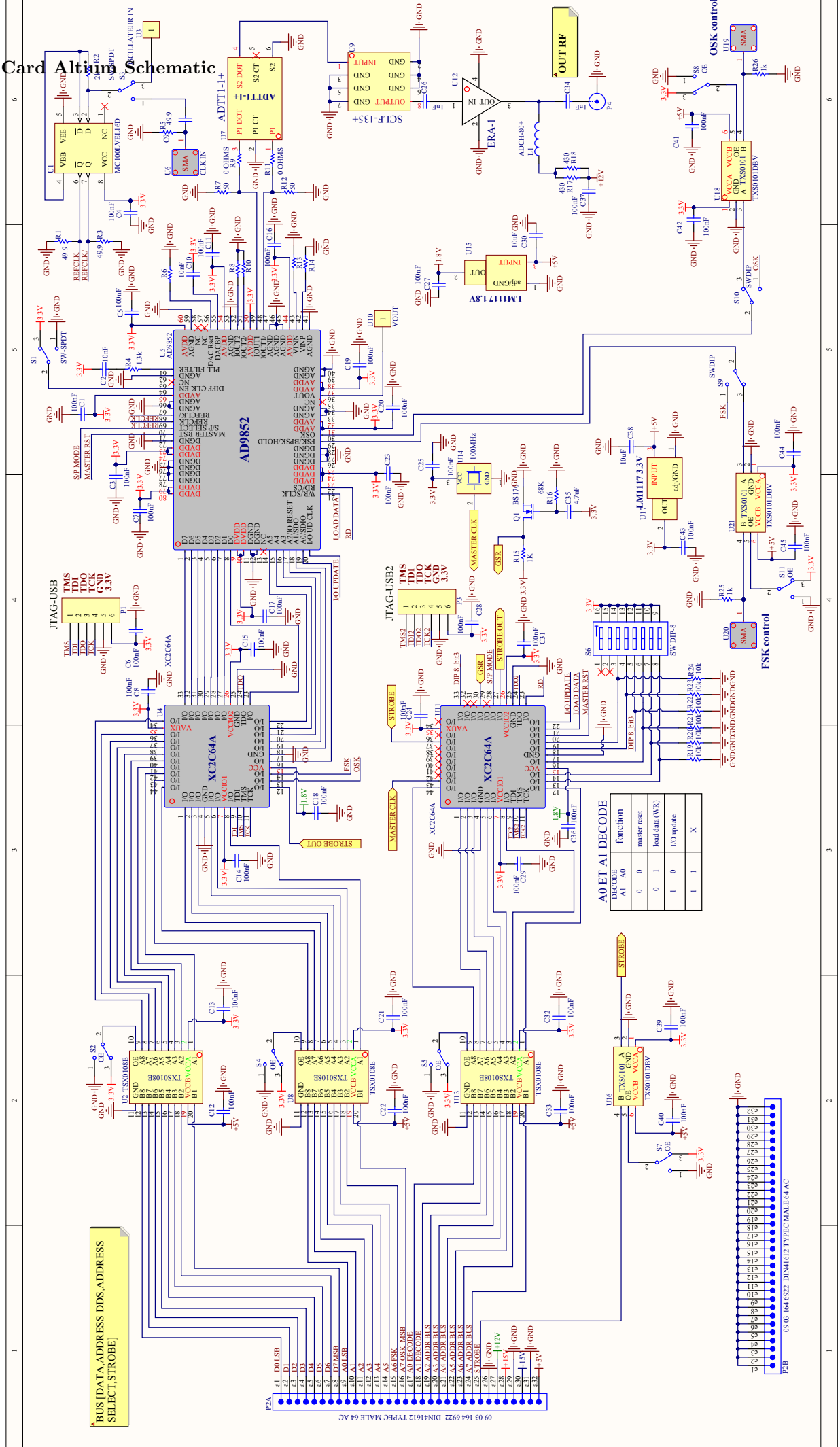
One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781.329.4700 www.analog.com
Fax: 781.461.3113

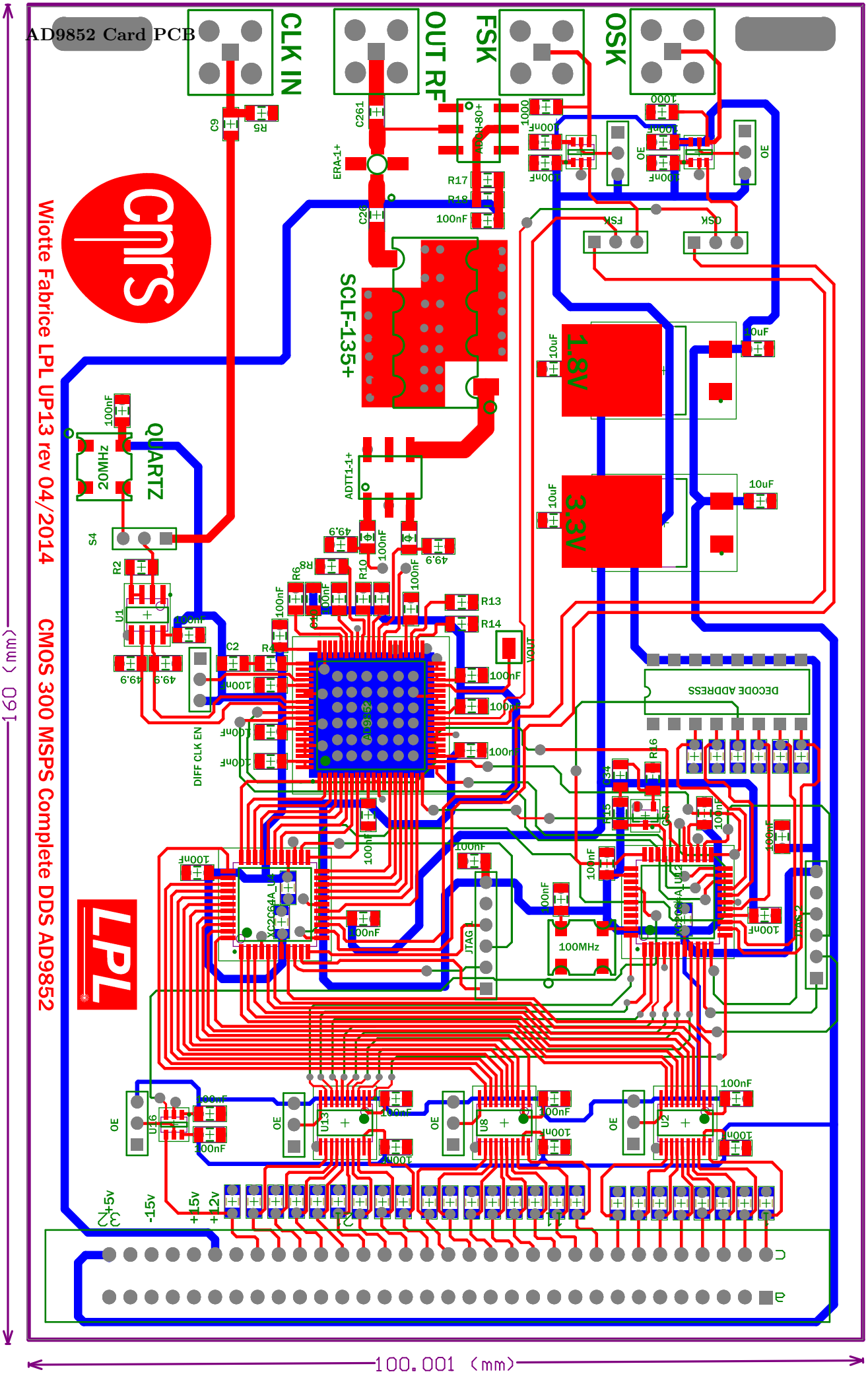
Table 9. Register Layout[1]

Parallel Address (Hex)	Serial Address (Hex)	AD9852 Register Layout								Default Value (Hex)
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
00	0	Phase Adjust Register 1 <13:8> (Bits 15, 14 don't care)					Phase 1			00
01		Phase Adjust Register 1 <7:0>								00
02	1	Phase Adjust Register 2 <13:8> (Bits 15, 14 don't care)					Phase 2			00
03		Phase Adjust Register 2 <7:0>								00
04	2	Frequency Tuning Word 1 <47:40>					Frequency 1			00
05		Frequency Tuning Word 1 <39:32>								00
06		Frequency Tuning Word 1 <31:24>								00
07		Frequency Tuning Word 1 <23:16>								00
08		Frequency Tuning Word 1 <15:8>								00
09		Frequency Tuning Word 1 <7:0>								00
0A	3	Frequency Tuning Word 2 <47:40>					Frequency 2			00
0B		Frequency Tuning Word 2 <39:32>								00
0C		Frequency Tuning Word 2 <31:24>								00
0D		Frequency Tuning Word 2 <23:16>								00
0E		Frequency Tuning Word 2 <15:8>								00
0F		Frequency Tuning Word 2 <7:0>								00
10		Delta frequency word <47:40>								00
11		Delta frequency word <39:32>								00
12		Delta frequency word <31:24>								00
13		Delta frequency word <23:16>								00
14		Delta frequency word <15:8>								00
15		Delta frequency word <7:0>								00
16	5	Update clock <31:24>								00
17		Update clock <23:16>								00
18		Update clock <15:8>								00
19		Update clock <7:0>								00
1A	6	Ramp rate clock <19:16> (Bits 23, 22, 21, 20, don't care)								00
1B		Ramp rate clock <15:8>								00
1C		Ramp rate clock <7:0>								00
1D	7	Don't care CR [31]	Don't care	Don't care	Comp PD	Reserved, always low	Control DAC PD	DAC PD	DIG PD	10
1E		Don't care	PLL range	Bypass PLL	Ref Mult 4	Ref Mult 3	Ref Mult 2	Ref Mult 1	Ref Mult 0	64
1F		CLR ACC1	CLR ACC2	Triangle	Don't care	Mode 2	Mode 1	Mode 0	Int/Ext update clock	01
20		Don't care	Bypass inv sinc	OSK EN	OSK INT	Don't care	Don't care	LSB first	SDO active CR [0]	20
21	8	Output shaped keying multiplier <11:8> (Bits 15, 14, 13, 12 don't care)								00
22		Output shaped keying multiplier <7:0>								00
23	9	Don't care								00
24		Don't care								00
25	A	Output shaped keying ramp rate <7:0>								80
26	B	Control DAC <11:8> (Bits 15, 14, 13, 12 don't care)								00
27		Control DAC <7:0> (Data is required to be in twos complement format)								00

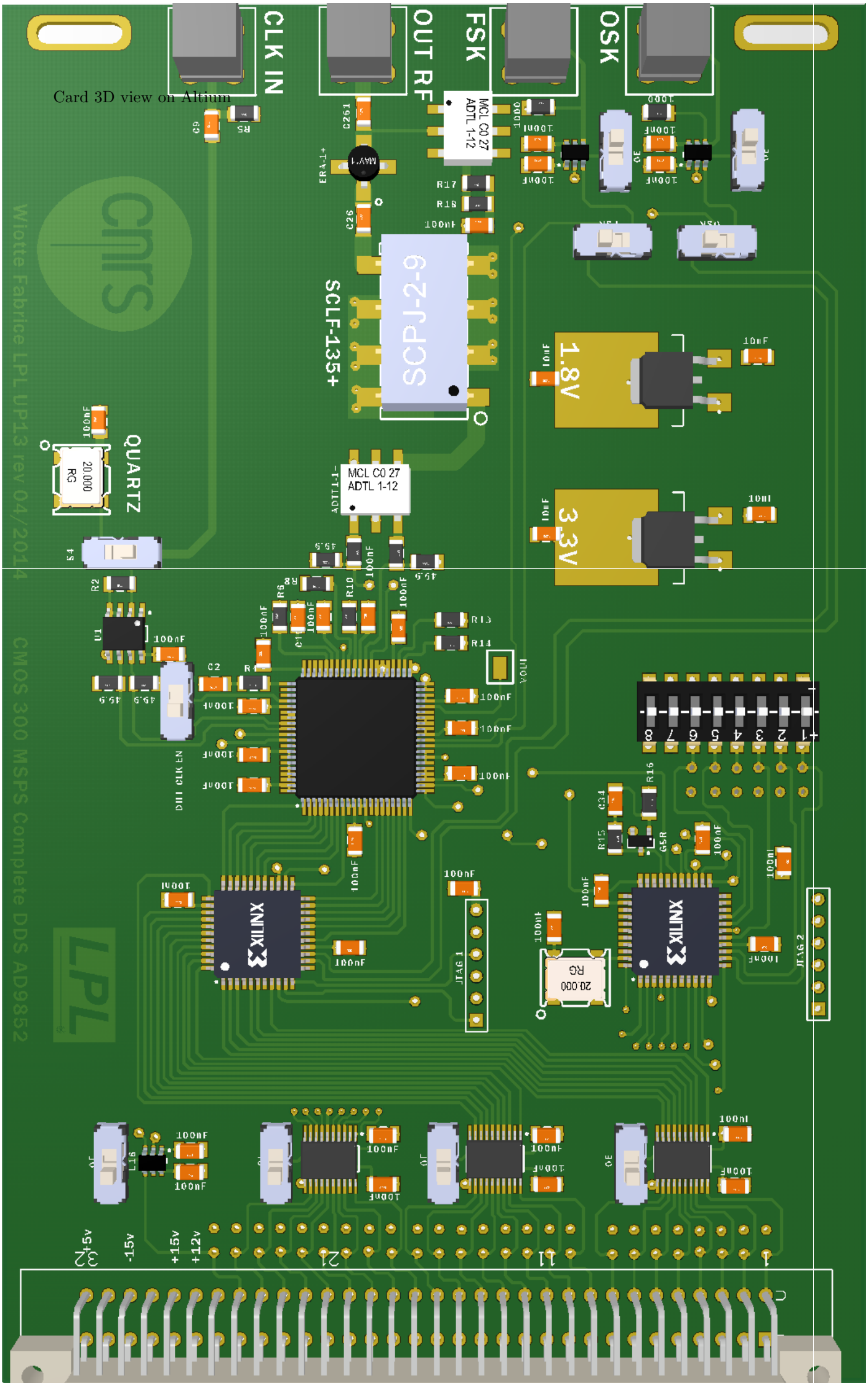
[1] The shaded sections comprise the control register.

Card Altium Schematic

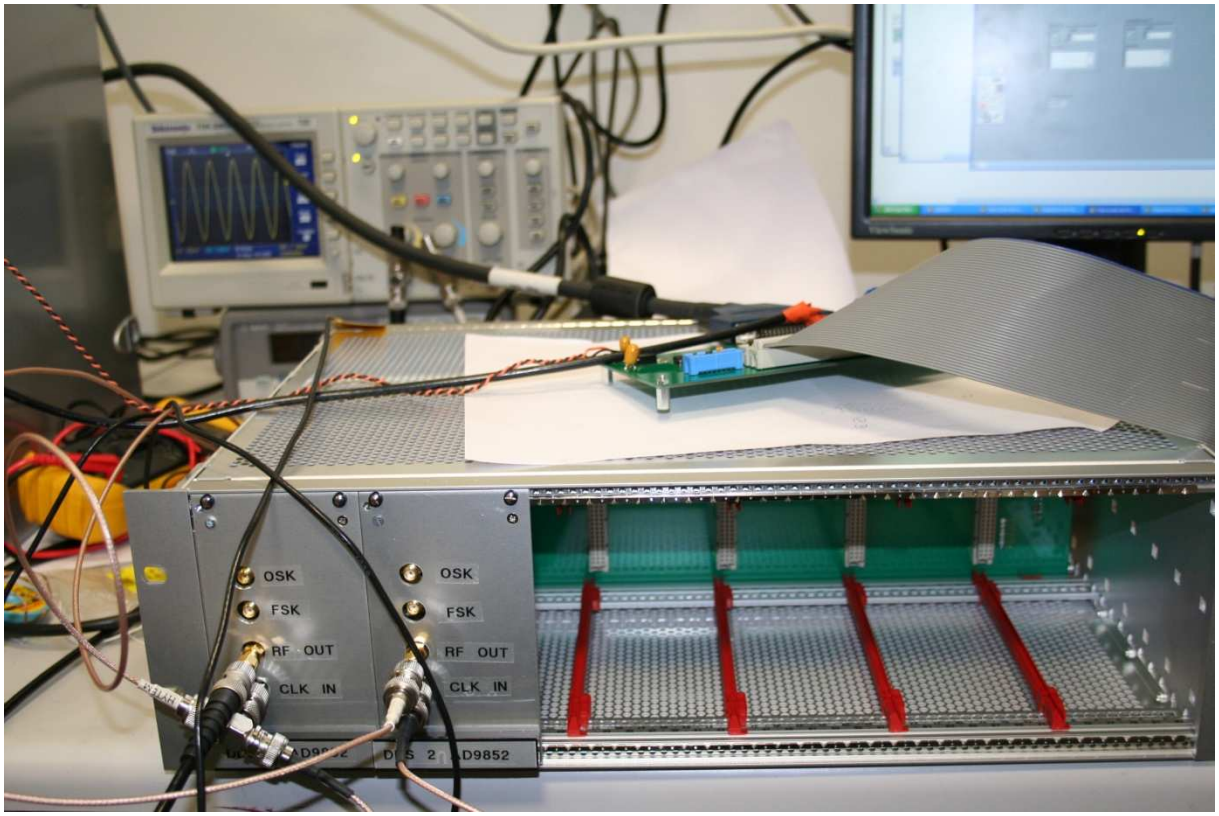


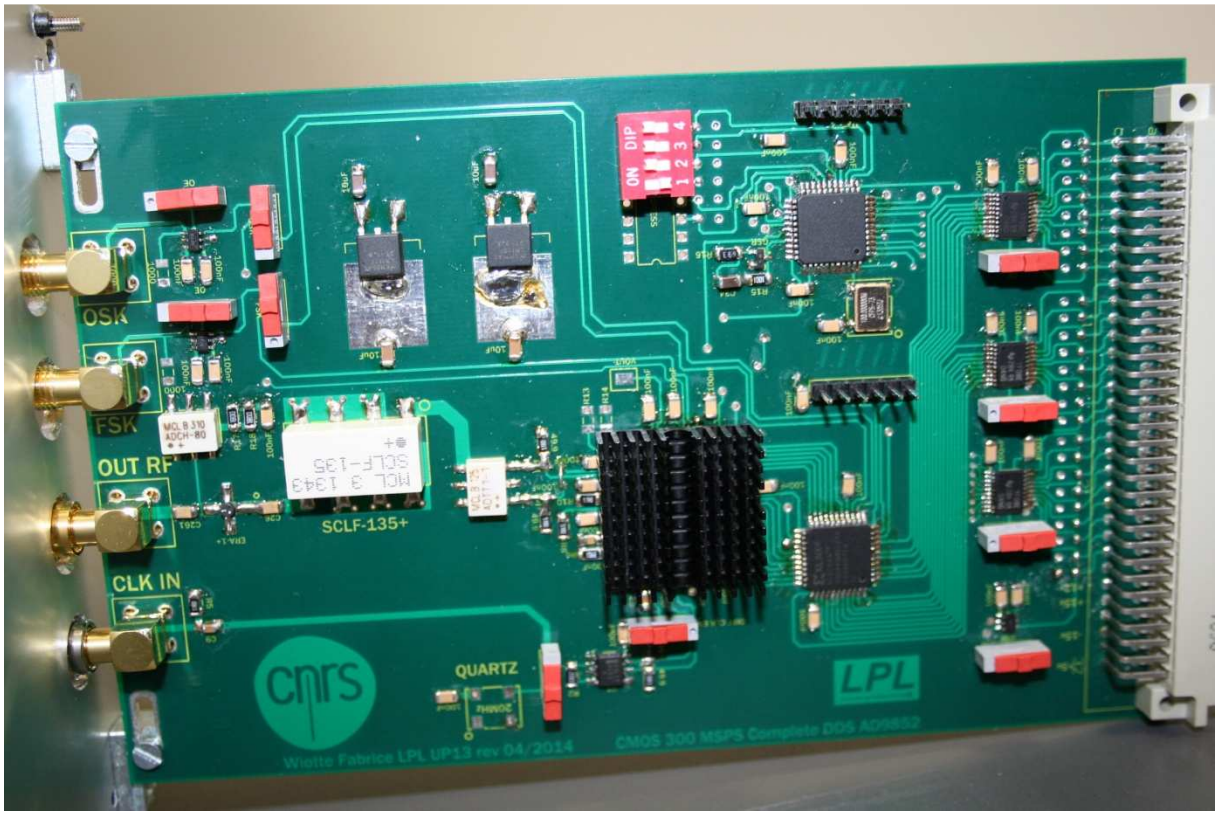

AD9852 Card PCB
CLK IN
OUT RF
FSK
OSK
Wiotte Fabrice LPL UP13 rev 04/2014
CMOS 300 MSPS Complete DDS AD9852
QUARTZ
20MHz
SCLF-135+
ADTT1-1+
ERA-1+
ZX60-83LN-S+
1.8V
3.3V
10uF
100nF
49.9
DECODE ADDRESS
DIFF CLK EN
JTAG
100MHz
160 (mm)
100.001 (mm)

Card 3D view on Altium

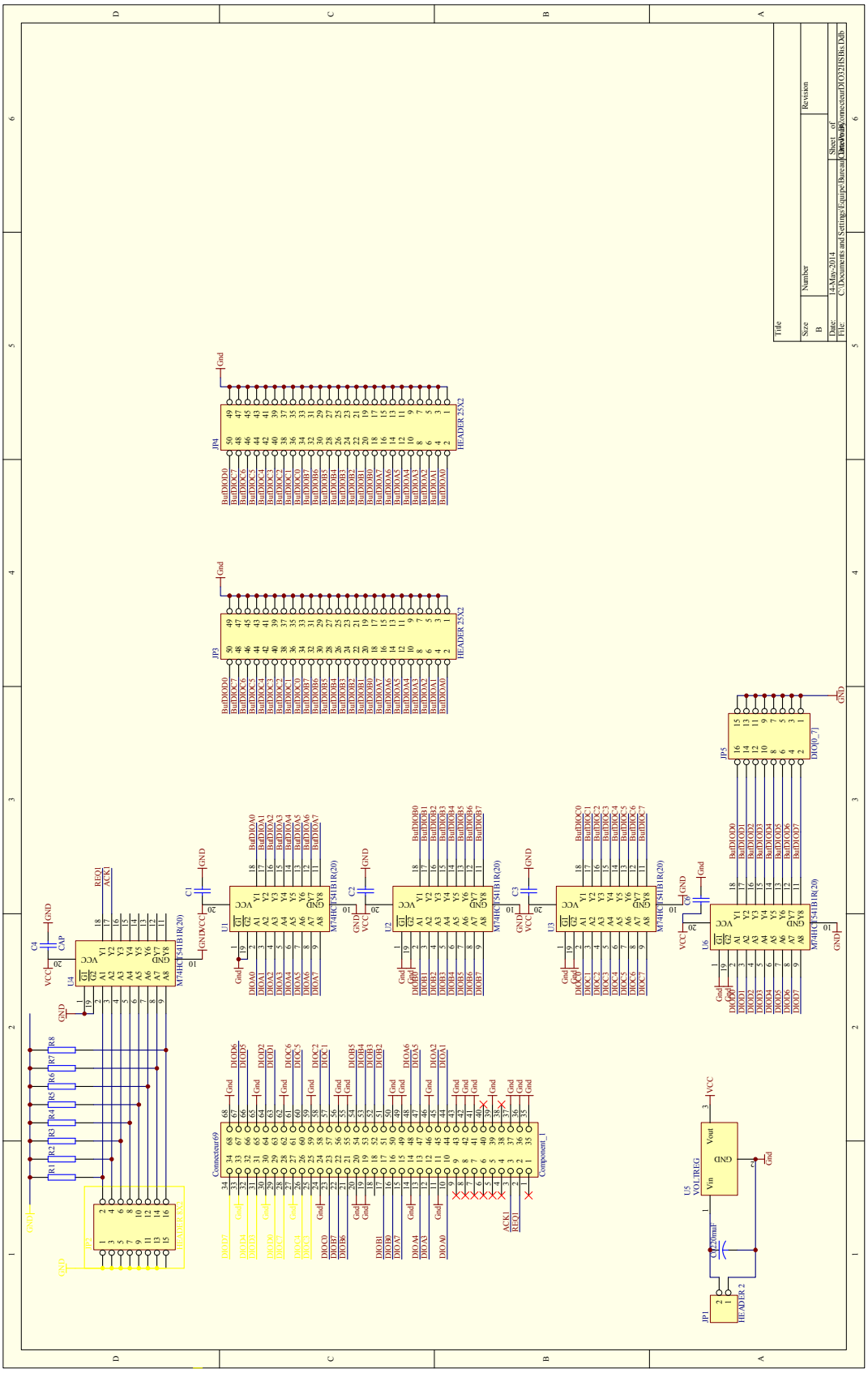


Photos : DDS and Rack





Annexe : Schematic of the Interface card between the NI card DIO32 HS and the BUS



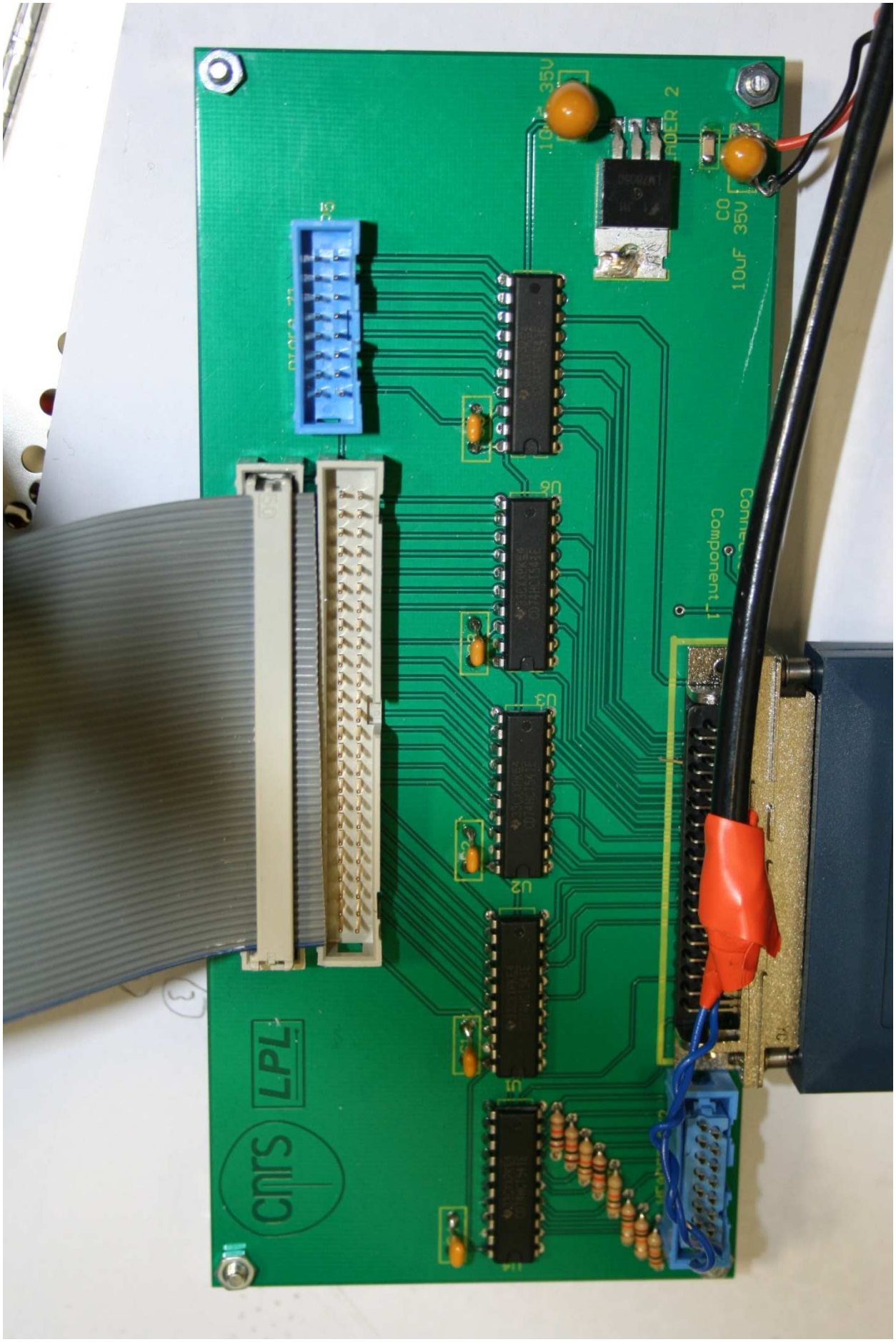
Annexe : Screen capture of the simple labwiew interface

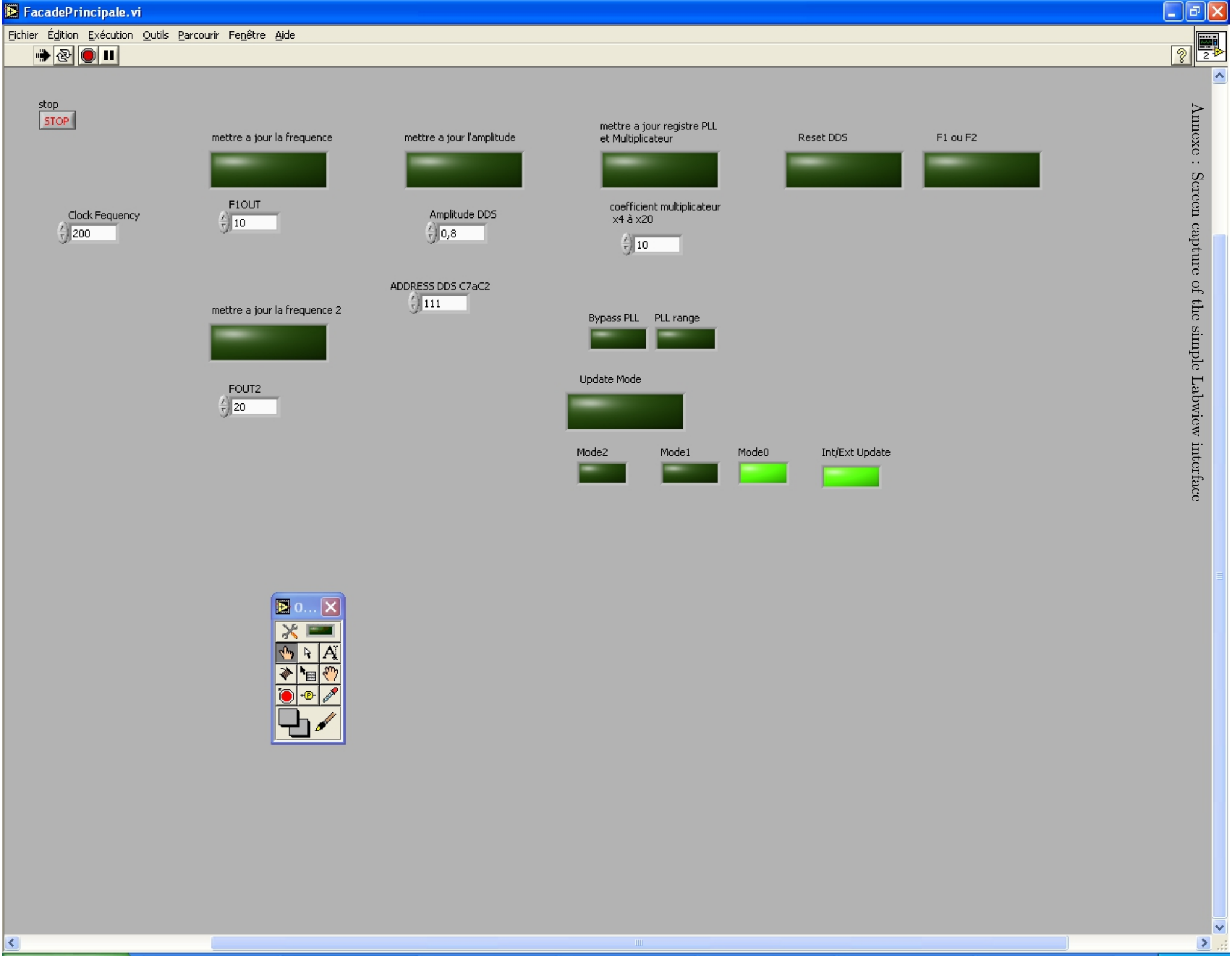
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;


entity addressbus_and_decode is
    GENERIC (n: INTEGER := 20);
    Port (
        ADDR_BUS_in : in  STD_LOGIC_VECTOR (5 downto 0);
     A0_A1_address: in  STD_LOGIC_VECTOR (1 downto 0);
     STROBE : in  STD_LOGIC;
          MASTER_CLOCK : in  STD_LOGIC;
          STROBE_OUT : out  STD_LOGIC;
          --IO_UPDATE : out  STD_LOGIC;
          RD : out  STD_LOGIC;
          LOAD_DATA : out  STD_LOGIC;
          MASTER_RESET : out  STD_LOGIC;
          SP_MODE : out  STD_LOGIC;
          CONTROL_DIP8 : in  STD_LOGIC_VECTOR (5 downto 0)
    );

end addressbus_and_decode;


architecture Behavioral of addressbus_and_decode is

SIGNAL internal: STD_LOGIC_VECTOR (n-1 DOWNTO 0);
SIGNAL STROBE_OUT_bis : STD_LOGIC;
SIGNAL SEL0 : STD_LOGIC;
SIGNAL SEL1 : STD_LOGIC;
SIGNAL SEL2 : STD_LOGIC;
SIGNAL SEL3 : STD_LOGIC;

begin

SP_MODE <='1';
RD <='1';
--MASTER_RESET <='0';

--registre a décalage Tempo STROBE
PROCESS(MASTER_CLOCK,STROBE)
BEGIN
IF (STROBE ='0') THEN
 internal <= (OTHERS => '0');
ELSIF (MASTER_CLOCK'EVENT AND MASTER_CLOCK='1') THEN
 internal <= STROBE & internal(internal'LEFT DOWNTO 1);
END IF;
END PROCESS;

-- 2 bits address A0 et A1 decoder 2 to 4 and control ADDR=CONTROL_dip8
process(internal(15),A0_A1_address)
begin
if  internal(15) = '1' and (ADDR_BUS_in = CONTROL_DIP8)  then
case A0_A1_address is
when "00"=> SEL0 <='1'; SEL1<='0'; SEL2<='0'; SEL3<='0';
when "01"=> SEL0 <='0'; SEL1<='1'; SEL2<='0'; SEL3<='0';
when "10"=> SEL0 <='0'; SEL1<='0'; SEL2<='1'; SEL3<='0';
when "11"=> SEL0 <='0'; SEL1<='0'; SEL2<='0'; SEL3<='1';
```

```
when others => SEL0 <='Z'; SEL1<='Z'; SEL2 <='Z'; SEL3 <='Z';
end case;
else
SEL0 <='Z'; SEL1<='Z'; SEL2 <='Z'; SEL3 <='Z';
end if;
end process;

STROBE_OUT <= STROBE_OUT_bis;

--declenchement STROBE out
process(internal(15),ADDR_BUS_in,CONTROL_DIP8)
begin
if internal(15) = '1' and (ADDR_BUS_in = CONTROL_DIP8) then
   STROBE_OUT_bis <='1';
else
   STROBE_OUT_bis <='0';
end if;
end process;

-- control bit MASTER_RESET
process(internal(10),SEL0,ADDR_BUS_in,CONTROL_DIP8)
begin
if internal(10) = '1' and SEL0 ='1' and (ADDR_BUS_in = CONTROL_DIP8) then
   MASTER_RESET <= '1';
else
   MASTER_RESET <= '0';
end if;
end process;

-- control bit Load data into IO buffer(WR)
process(internal(0),SEL1,ADDR_BUS_in,CONTROL_DIP8)
begin
if internal(0) = '1' and SEL1 ='1' and (ADDR_BUS_in = CONTROL_DIP8) then
   LOAD_DATA <= '0';
else
   LOAD_DATA <= '1';
end if;
end process;

-- control bit I/O update output register
--process(internal(0),SEL2)
--begin
--if internal(0) = '1'  then
--   IO_UPDATE <= '1';
--else
 --  IO_UPDATE <= '0';
--end if;
--end process;

-- no additional strobe
process(internal(0),SEL3)
begin
if internal(0) = '1' and SEL3 ='1' then
null;
end if;
end process;
```

```
end Behavioral;
```

Annexe : Photos of the Interface card between the NI card DIO32 HS and the BUS




FacadePrincipale.vi
Fichier Édition Exécution Outils Parcourir Fenêtre Aide
stop
STOP
mettre a jour la frequence
mettre a jour l'amplitude
mettre a jour registre PLL
et Multiplicateur
Reset DDS
F1 ou F2
Clock Fequency
200
F1OUT
10
Amplitude DDS
0,8
coefficient multiplicateur
x4 à x20
10
ADDRESS DDS C7aC2
111
mettre a jour la frequence 2
Bypass PLL
PLL range
Update Mode
FOUT2
20
Mode2
Mode1
Mode0
Int/Ext Update
Annexe : Screen capture of the simple Labwiew interface

blbla

Acknowledgements: LPL is Unité Mixte (UMR 7538) of CNRS and of Université Paris Nord. We acknowledge financial support from Conseil Régional d'Ile-de-France, Ministère de l'Enseignement Supérieur et de la Recherche and IFRAF.