Università degli Studi di Padova

FACOLTÀ DI INGEGNERIA

Corso di Laurea Magistrale in Ingegneria Elettronica

Tesi di laurea Magistrale

LabVIEW development of a debug GUI for automated tests on voltage regulators

Relatore

Prof.ssa Giada Giorgi

Correlatore

Dott. Ing. Daniele Dario

Laureando

Riccardo Ciatto

Padova - 9 Dicembre 2014

Anno accademico 2013/2014

CONTENTS

LIST OF FIGURES

INTRODUZIONE

Il presente lavoro di tesi si è sviluppato nel contesto di un'esperienza di stage da me svolta presso Infineon Technologies Italy (PD), all'interno della Standard Power Business Line.

Infineon Technologies è un'azienda tedesca che lavora nell'ambito dei semiconduttori, occupandosi in particolare di sistemi per automotive, componenti e sistemi per il controllo delle alte potenze, transistor e sistemi di pilotaggio per applicazioni industriali, microelettronica per sistemi di pagamento, comunicazione e identificazione personale. L'area di Standard Power si occupa nello specifico del settore Automotive, spaziando su una vasta gamma di prodotti, come Linear Voltage Regulators, FlexRay, CAN e LIN Transceivers, Switching Voltage Regulators (DC/DC Converters) e Industrial Standard.

L'interesse cruciale delle aziende che, come Infineon Technologies, si occupano di semiconduttori, è quello di sviluppare prodotti e tecnologie che siano in grado di sostenere l'elevato livello di competitività che caratterizza questo settore del mercato. Uno degli step cruciali, all'interno dello sviluppo di questi prodotti, è la caratterizzazione di laboratorio, fase necessaria alla validazione delle funzionalità elettriche di ogni dispositivo. Le modalità di esecuzione di tale fase devono essere tali da garantire la qualità della valutazione e la conformità del prodotto alle relative specifiche. Per ogni nuovo componente che deve essere caratterizzato, infatti, è molto importante essere in grado di assicurare con precisione che tale prodotto funzioni correttamente all'interno delle specifiche che sono segnalate nel relativo datasheet.

La Figura 1.1 mostra, a titolo di esempio, il datasheet di un voltage regulator, documento utile a chiunque sia interessato all'acquisto e all'utilizzo del componente, dal momento che ne fornisce una descrizione dettagliata e tutte le informazioni necessarie (caratteristiche, comportamento elettrico, ...) ad effettuarne una corretta applicazione.

Durante la fase di caratterizzazione di un nuovo prodotto, prima del suo rilascio, devono essere testati tutti i parametri richiesti nel datasheet dello specifico dispositivo, e i relativi risultati vengono poi inseriti all'interno di tale scheda tecnica.

Chiaramente la fase di caratterizzazione di un componente richiede un tempo di esecuzione maggiore, se il metodo applicato per la realizzazione di tutti i test è quello manuale, e l'obiettivo generale perseguito in questo ambito è quello di automatizzare il più possibile le varie attività di test.

L'obiettivo principale del progetto che ho seguito all'interno di Infineon è stato quello di realizzare una Graphical User Interface da associare ad una preesistente piattaforma hardware, sviluppata dal Team dei Product Engineer della sede di Padova, pensata allo scopo di supportare il lavoro degli ingegneri nelle valutazioni di laboratorio, in fase di caratterizzazione dei dispositivi. Tale piattaforma hardware è stata progettata come la combinazione di due diverse schede: una custom-made test board, chiamata GPB (General Purpose Board) o motherboard, e una board di dimensioni minori, definita PCB (Printed Circuit Board) o doughterboard. Dalla combinazione di queste due diverse schede è possibile effettuare in modo semplificato tutti i test necessari a ricoprire l'intera gamma di parametri richiesti all'interno del datasheet di un dispositivo. Nello specifico, tali schede e l'interfaccia software da me realizzata sono state sviluppate per supportare la fase di test della famiglia dei dispositivi Voltage Regulator, in tutti i loro packages.

L'idea è stata quindi di realizzare un supporto hardware/software che permettesse di automatizzare la fase di caratterizzazione dei dispositivi, in modo da ridurre i tempi necessari alle misure senza però comprometterne la qualità in termini di accuratezza.

L'hardware della GPB è composto da un set di connettori esterni, ai quali sono connessi gli strumenti di misura, e da una matrice di 48 relè la cui funzione è quella di interruttori. Al centro della GPB sarà collegata la daughterboard con il DUT.

Uno strumento di misura sarà connesso ad uno specifico pin del DUT se il relè necessario al collegamento viene attivato; altrimenti lo strumento rimane flottante o n.c. (relè off). In questo modo, variando lo stato dei relè (on/off) è possibile attivare/disattivare il collegamento degli strumenti ai pin del DUT. Ad ogni setup

di misura (configurazione di strumenti utilizzata per effettuare una misura sul DUT) sarà quindi associata una configurazione di relè della board. L'attivazione/disattivazione dei relè è possibile usando i dip-switch della GPB, oppure inviando un array di 48 bit (ad ogni bit è associato la stato di un relè nella board) via SPI.

La GPB è sempre accompagnata/integrata da una PCB (daughterboard) specifica per ogni prodotto. La daughterboard è una scheda pensata e progettata sul dispositivo (voltage regulator) da testare e sul set di misure richieste dal datasheet per la fase di caratterizzazione. Infatti, il set di misure di test richieste, varia (seppur di poco) da dispositivo a dispositivo, e per ogni tipo di misura il dispositivo necessita di componenti esterni (resistenze, condensatori) che saranno integrati nella board.

La daughterboard sarà formata da resistenze e condensatori necessari alle misure, e da una matrice di n relè come per la GPB. Variando lo stato dei relè, è possibile attivare/disattivare il collegamento dei componenti passivi ai pin del dispositivo.

L'integrazione tra matherboard e daughterboard è quindi in grado di dare pieno supporto agli ingegneri per la fase di caratterizzazione, rendendo possibile l'automazione nelle misure. Inviando tramite interfaccia SPI un array di 48 + n bits, è possibile impostare un setup di misura completo: i primi 48 bit gestiscono il collegamento degli strumenti ai pin del DUT, i successivi n bit attivano/disattivano la connessione di resistenze e condensatori esterni al DUT. Pre-impostando una matrice di 48*n bits, dove ogni riga corrisponde ad un differente setup di misura, è possibile generare una matrice di setup che copra tutte le misure richieste dal datasheet per il dispositivo in esame.

Il lavoro da me svolto è stato quello di realizzare una GUI in LabVIEW che consenta di gestire da remoto tutta la fase di setup delle misure e di comunicare con gli strumenti di misura tramite interfacce pop-up dedicate. Il setup delle misure è gestibile inviando comandi SPI alle boards, mentre, è possibile aprire/chiudere l'interfaccia pop-up di uno strumento dalla GUI principale tramite un pulsante dedicato.

La Figura 1.2 illustra come tools hardware e software sono messi in comunicazione tra loro. In particolare, il software LabVIEW comunica via USB con il microcontrollore Arduino Due, il quale a sua volta, invia comandi SPI alle boards. Diversamente, la comunicazione tra software LabVIEW e strumenti di misura è ottenuta usando l'interfaccia GPIB.

Nel primo capitolo di questa tesi sono introdotti i tools hardware/software e le interfacce di comunicazione utilizzate nello sviluppo della GUI. Il secondo capitolo

descrive poi le principali funzionalità messe a disposizione dalla GUI, riportando, per ogni funzionalità implementata, esempi di sviluppo di codice LabVIEW. Il terzo capitolo propone quindi una breve descrizione teorica degli strumenti di misura utilizzati, e per ogni strumento, illustra parte del codice LabVIEW sviluppato per la realizzazione delle interfacce pop-up degli strumenti di misura.

CHAPTER

1

INTRODUCTION

The present work has been developed within Infineon Technologies Italy (PD) in the context of an internship carried out in the Automotive (ATV) area, in the Standard Power Business Line.

Infineon Technologies offers semiconductors and systems for automotive, industrial and multimarket sectors, as well as chipcard and security products. The department of Standard Power, in particular, offers a broad portfolio on the Automotive market for standard power products, such as Linear Voltage Regulators, FlexRay, CAN and LIN Transceivers, Switching Voltage Regulators (DC/DC Converters) and Industrial Standard.

The core of semiconductor companies, such as Infineon Technologies, is product and technology development to bear the competitiveness. One of the criteria in product development is electrical lab characterization which is crucial in validating the device's electrical functionality without jeopardizing the evaluation quality and ensuring that it is within the product specifications.

For a new product characterization, it is significant to ensure that the product is functioning well within the datasheet specification. Figure 1.1 illustrates an example of a voltage regulator product datasheet which is a useful document for users. They will be able to obtain as much information as possible for instances

the product information, application setup and device's electrical behavior in the datasheet. During a new product characterization, all datasheet parameters will be tested and results are compiled to finalize the datasheet before its release. Typical lab characterization takes longer time to complete since it is deploying the manual measurement method, and the overall objective in this area is to automate as much as possible the various testing activities.

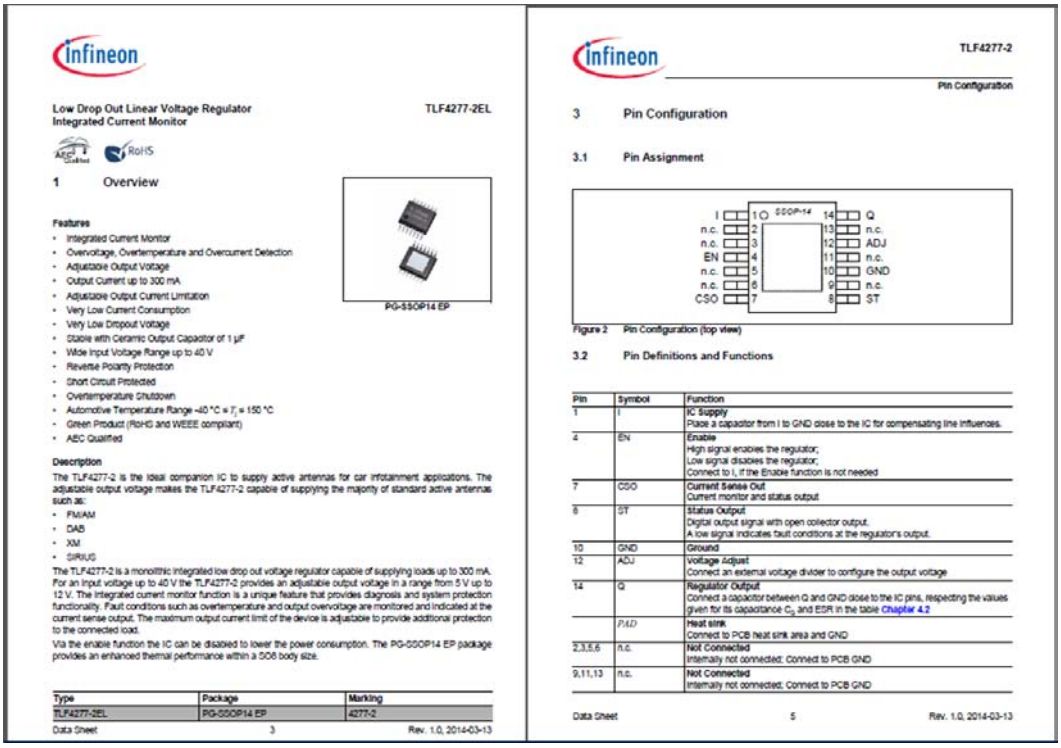

Low Drop Out Linear Voltage Regulator
Integrated Current Monitor
TLF4277-2EL
1 Overview
Features
Description
PG-SSOP14 EP
Type | Package | Marking
TLF4277-2EL | PG-SSOP14 EP | 4277-2
Data Sheet 3 Rev. 1.0, 2014-03-13
TLF4277-2
Pin Configuration
3 Pin Configuration
3.1 Pin Assignment
Figure 2 Pin Configuration (top view)
3.2 Pin Definitions and Functions
Pin | Symbol | Function
Data Sheet 5 Rev. 1.0, 2014-03-13

Figure 1.1: Datasheet example

The target of the main project in which i have worked has been to realize the Graphical User Interface of a new hardware platform, developed by the Product engineers Team of Infineon Padua, conceived to support engineers in lab evaluations for the characterization phase. This hardware platform has been designed as the combination between a custom-made test board, named general purpose board (GPB) or motherboard, and a smaller dimension of printed circuit board (PCB), called daughterboard. From this combination derives the capability to fulfill the entire coverage of datasheet test parameters for lab characterization. Specifically, these two hardware boards support characterization for voltage regulators which come in various device packages.

The idea was to realize a hardware/software support that would allow to automate the devices characterization phase, in order to reduce the time required to

perform measures, without compromising their quality, in terms of accuracy.

The hardware of the GPB is composed by a set of external sockets, to which is possible to connect instruments to perform measures, and by an array of 48 relays that work as switches. In the central area of the GPB is possible to connect the daughterboard with the DUT mounted above.

A measuring instrument is considered as connected to a specific pin of the DUT, if the relay on its connection is activated; otherwise the instrument is considered as floating or N.C. (relay off). In this way, by changing the status of the relay (on/off), you can enable/disable the connection of instruments to the DUT pins.

Each measurement setup (configuration of instruments used to perform a measurement on the DUT) is associated with a specific configuration of the relays of the board. The activation/deactivation of the relays is possible by using the dip-switches of the GPB, or by sending an array of 48 bits via SPI (to each bit is associated the status of a relay on the board).

The GPB is always accompanied/integrated by a PCB (daughterboard) that is different for each specific product to be tested. The daughterboard is a board conceived and designed in order to test a specific device (voltage regulator), that is paying attention to its special features, and the specific set of measures required by its datasheet for the characterization phase. In fact, the requested set of measures, varies (albeit slightly) from device to device, and for each type of measurement the device need external components (resistors, capacitors) that have to be integrated in the board.

The daughterboard is provided with resistors and capacitors, that are necessary to perform measurements, and with a matrix of n relays. By changing the status of relays, you can enable/disable the connection of passive components to the pins of the device.

The integration between matherboard and daughterboard is therefore able to give full support for the characterization phase, making possible the automation in measurements. Sending via the SPI interface an array of 48-bit plus n bit, you can set a complete measurement setup: first 48-bits manage the connection of instruments to the pins of the DUT, next n bits activate/deactivate the connection of external resistors and capacitors to DUT.

The work I have developed has been to develop, using LabVIEW, a GUI (Graphical User Interface) that allows to remotely manage the setup of the measures and to communicate with instruments via dedicated pop-ups. The setup of the measures is manageable by sending SPI commands to the board, while, it is possible to

open/close interface pop-ups of an instrument from the main GUI via a dedicated button.

Figure 1.2 illustrates how hardware and software tools are put in communication with each other. In particular, LabVIEW software communicates via USB with the Arduino due microcontroller, which sends SPI commands to the boards. The communication between LabVIEW software and the instruments is otherwise obtained using the GPIB interface.

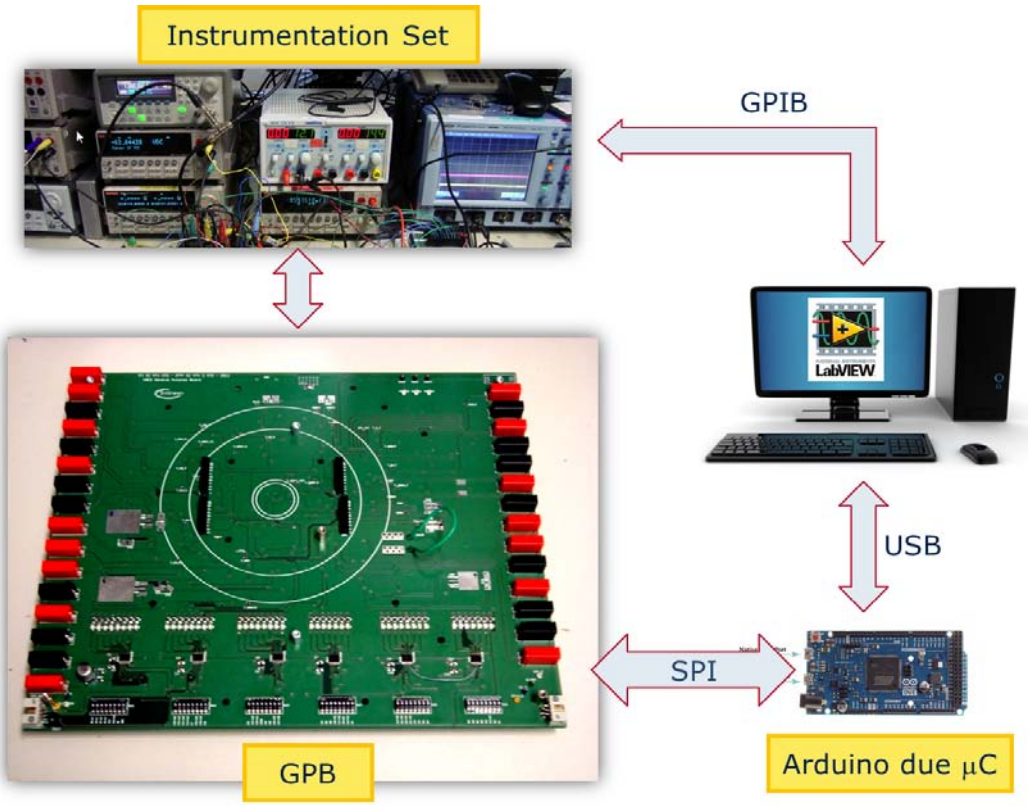


Figure 1.2: Measurement bench

The second chapter of this work introduces the hardware/software tools and the communication interfaces used in the project.

The third chapter describes the main features provided by the GUI, highlighting, for each function implemented, examples of LabVIEW code developed.

The fourth chapter is a brief theoretical description of instruments used in association with the GPB, and for each instrument, it shows the LabVIEW code developed for the realization of instrument pop-ups.

CHAPTER

2

HARDWARE/SOFTWARE AND COMMUNICATION PROTOCOLS

This chapter presents hardware/software and communication protocols involved in this project, listed below:

1. GPB (General Purpose Board);
    - SPI (Serial Peripheral Interface);
    - MAX482x;
    - Relays;
    - Daughterboard;
2. Arduino due microcontroller;
3. LabVIEW software;
4. GPIB (General Purpose Interface Bus);
5. SCPI (Standard Commands for Programmable Instruments);

2.1 GPB (General Purpose Board)

As introduced in the previous chapter, the GPB hardware is a custom-made test board designed to support the characterization for voltage regulators which come in various device packages. The GPB, also called motherboard, is always completed by the integration of another board: the daughterboard. The combination between motherboard and daughterboard allows to fulfill the entire coverage of datasheet test parameters for lab characterization.

The GPB has been thought to connect instruments to device pins, and this is possible by activating/deactivating the relays of the board. The relays could be viewed as switches, which enable the physical connection between instrument and device pins. The Figure 2.1 shows, for example, a portion of the PCB (Printed Circuit Board) for the GPB.

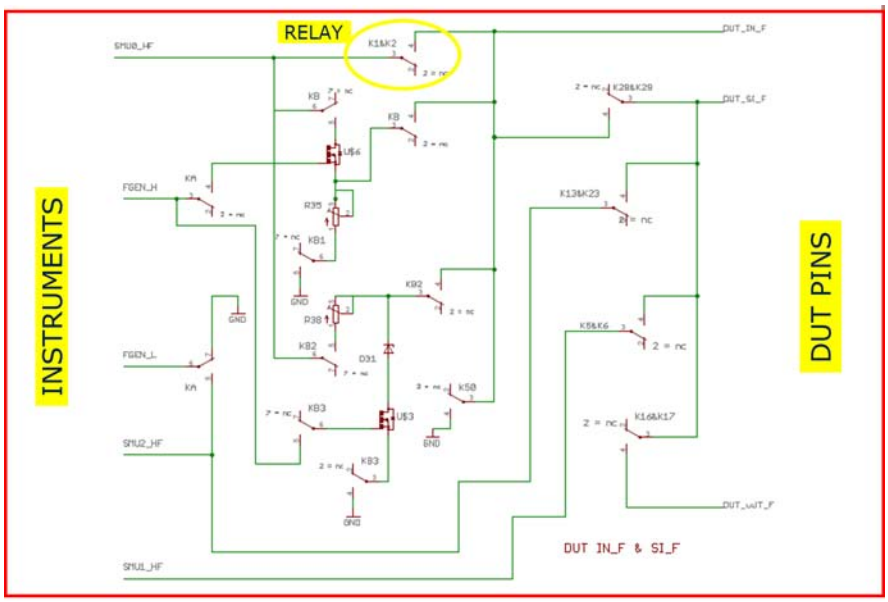
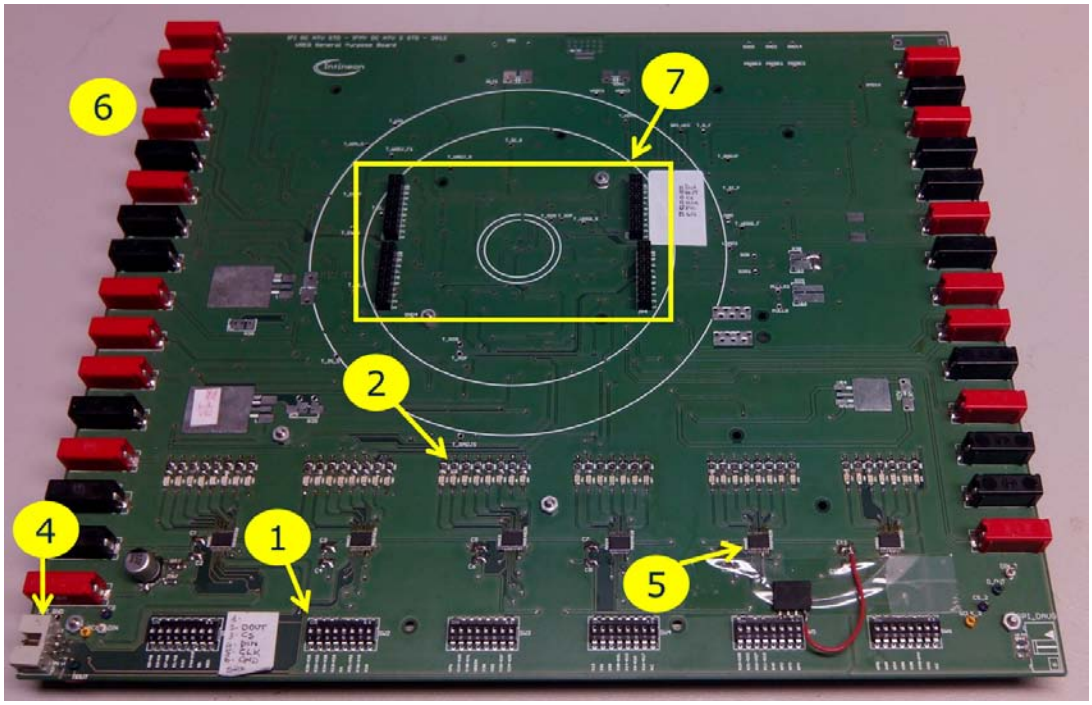


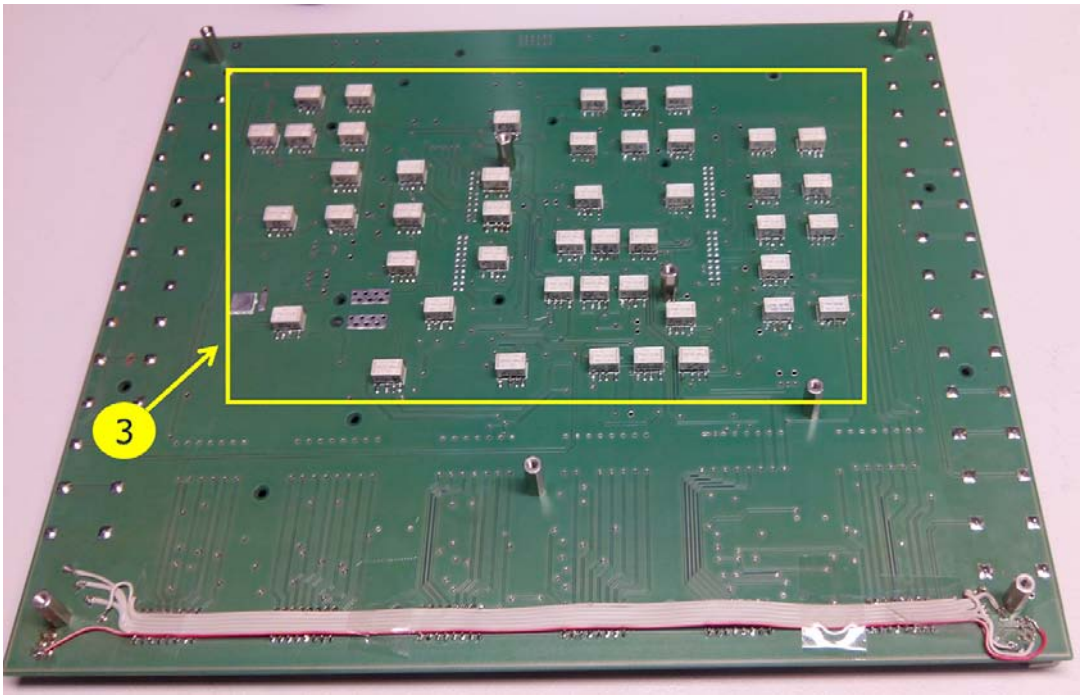
Figure 2.1: Portion of the Printed Circuit Board (PCB)

The GPB is composed by 48 relays, which are directly controlled using dip-switches or eventually through SPI interface. Each relay has a LED, which indicates if a specific relay is active. The physical connection between instrument and GPB is realized using banana connectors positioned into the two sides of the board. Besides, the daughterboard could be positioned in the center of the GPB, where there are 4 adaptors to join the boards.

Figure 2.2 shows the GPB hardware, where we can see all the components mentioned above:



(a) top of the GPB



(b) bottom of the GPB

Figure 2.2: General Purpose Board (GPB) top and bottom view

1. Dip-switches;
2. Led indicators;
3. Relays;
4. SPI (Serial Peripheral Interface);
5. MAX4820 relay driver;
6. Banana connectors;
7. Daughterboard socket;

In order to design the GPB, Eagle PCB design software from CadSoft is used as the printed circuit board (PCB) design tool since it is a freeware which can be obtained easily from any open source. Figure 2.3 elucidates the PCB designing flow in EAGLE PCB design software. It begins with design and prototype, capturing the schematic once prototype is available then followed by physical layout where traces checking are done and finally testing the hardware once it is complete. The hardware testing stage wraps up the PCB design process.

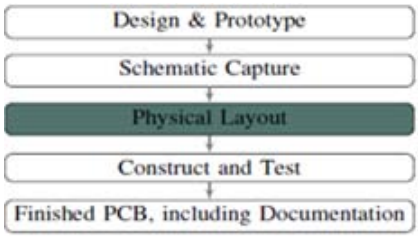


Figure 2.3: Eagle design flow

2.1.1 SPI (Serial Peripheral Interface)

Serial peripheral interface (SPI) is a synchronous serial interface where an 8-bit byte data can be shifted in and out 1 bit at a time. It can be used to communicate with a serial peripheral device or with a different microcontroller that has an embedded SPI interface.

SPI is a synchronous data bus, which means that it uses separate lines for data and a "clock" that keeps both sides in perfect sync. The clock is an oscillating signal that tells the receiver exactly when to sample the bits on the data line. This could be the rising (low to high) or falling (high to low) edge of the clock signal. When

the receiver detects that edge, it will immediately look at the data line to read the next bit.

In SPI, only one side generates the clock signal (usually called CLK or SCK for Serial ClocK). The side that generates the clock is called the master, and the other side is called the slave. There is always only one master, but there can be multiple slaves (more on this in a bit).

When data is sent from the master to a slave, it is sent on a data line called MOSI, for "Master Out/Slave In". If the slave needs to send a response back to the master, the master will continue to generate a prearranged number of clock cycles, and the slave will put the data onto a third data line called MISO, for "Master In/Slave Out". Notice we said "prearranged" in the above description. Because the master always generates the clock signal, it must know in advance when a slave needs to return data and how much data will be returned. This is very different than asynchronous serial, where random amounts of data can be sent in either direction at any time. Note that SPI is "full duplex" (has separate send and receive lines), and, thus, in certain situations, you can transmit and receive data at the same time (for example, requesting a new sensor reading while retrieving the data from the previous one).

There is one last line you should be aware of, called SS for Slave Select. This tells the slave that it should wake up and receive/send data and is also used when multiple slaves are present to select the one you would like to talk to.

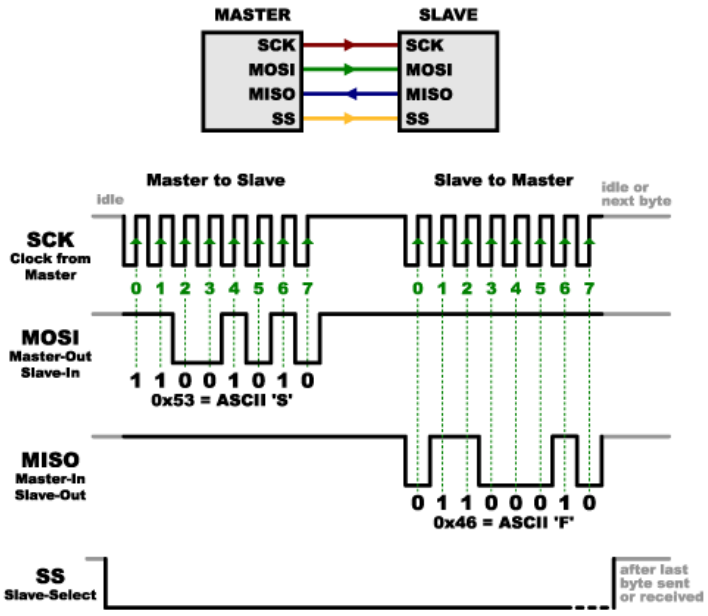


Figure 2.4: SPI (Serial Peripheral Interface)

There are two ways of connecting multiple slaves to an SPI bus:

- In general, each slave will need a separate SS line (Figure 2.5). To talk to a particular slave, you will make that slave's SS line low and keep the rest of them high (you do not want two slaves activated at the same time, or they may both try to talk on the same MISO line resulting in garbled data). Lots of slaves will require lots of SS lines; if you are running low on outputs, there are binary decoder chips that can multiply your SS outputs.

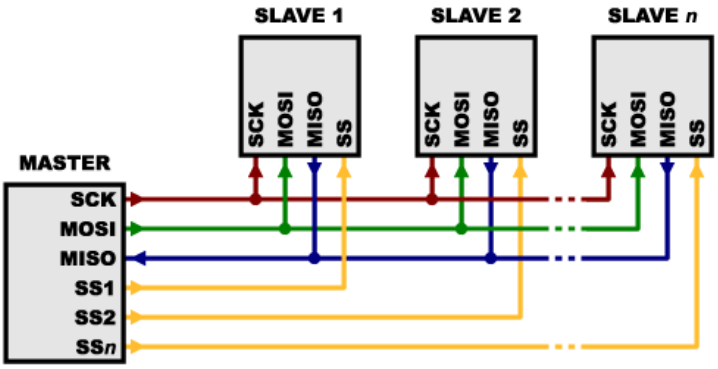


Figure 2.5: SPI separate SS line structure

- On the other hand, some parts prefer to be daisy-chained together, with the MISO (output) of one going to the MOSI (input) of the next (Figure 2.6). In this case, a single SS line goes to all the slaves. Once all the data is sent, the SS line is raised, which causes all the chips to be activated simultaneously. This is often used for daisy-chained shift registers and addressable LED drivers.

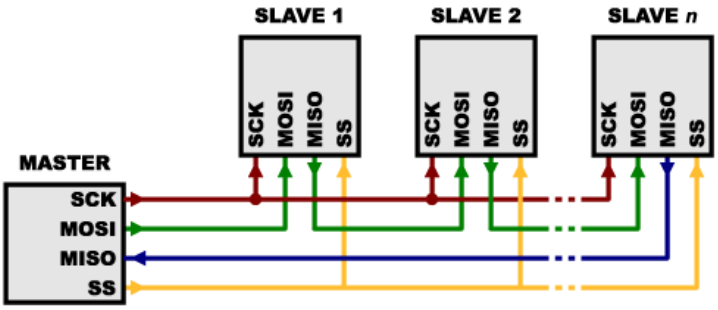


Figure 2.6: SPI daisy chain structure

This type of layout is typically used in output-only situations, such as driving LEDs where you do not need to receive any data back. In these cases you can leave the master's MISO line disconnected. However, if data does need to be returned to

the master, you can do this by closing the daisy-chain loop (blue wire in the above diagram).

2.1.2 MAX482x

The serial peripheral interface (SPI) is used to activate relay matrix for each test parameter setup. A special dedicated circuit is required to drive the relays according to the test setup. MAX482x is a relay driver which is used in the GPB to control the relays on the motherboard. Figure 2.7 is an example of a MAX482x device.

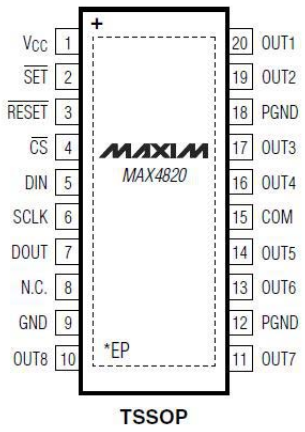


Figure 2.7: MAX4820 device

MAX482x relay driver can be connected in a *daisy chain* which provides flexibility to the users to have more relays as well as relay drivers with only one supply. In addition to the advantages, MAX482x could also be easily controlled using an SPI device which is among the advantages of utilizing a MAX relay driver in the setup.

Any DC power supply is sufficient to power up the relay drivers but an SPI device plays a very important role which is to control relays based on users' input. *Arduino due* microcontroller can be used to generate SPI signal, and it is programmable using a dedicated programming language.

2.1.3 Relays

The relays used in the GPB are DPDT relays. DPDT stands for double pole double throw relay. Relay is an electromagnetic device used to separate two circuits electrically and connect them magnetically. They are very useful devices and allow one circuit to switch another one while they are completely separate.

The DPDT relay (Figure 2.8) has a structure with two sections, input and output. The input section consists of a coil with two pins which are connected to the ground and the input signal. The output section consists of contactors which connect or disconnect mechanically. The output section consists of six contactors with two sets. Each set has three changeover contacts, namely, normally open (NO), normally closed (NC) and common (COM). When no supply is given the COM is connected to NC. When the operating voltage is applied the relay coil gets energized and the COM changes contact to NO.

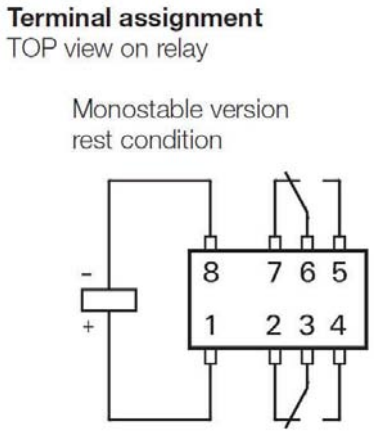


Figure 2.8: DPDT relay

The DPDT relays in the GPB lead FORCE and SENSE signals. To reach more precision in measurement, it is used to make measures with four wires.

Four-Wire Resistance Measurements

Due to the limitations of the two-wire method, a different approach is used for low resistance measurements that reduce the effect of test lead resistance. For measuring DUTs with resistances equal to or less than $1K\Omega$, you may use the four-wire connection shown in Figure 2.10.

Because the voltage is measured at the DUT, voltage drop in the test leads is eliminated (this voltage could be significant when measuring low-resistance devices).

With this configuration, the test current (I) is forced through the test resistance (R) via one set of test leads, while the voltage (V_M) across the DUT is measured through a second set of leads (sense leads). Although some small current (typically less than $100pA$) may flow through the sense leads, it is usually negligible and can generally be ignored for all practical purposes.

Therefore the voltage measured by the meter (V_M) is essentially the same as the voltage (V_R) across the resistance (R). As a result, the resistance value can be determined much more accurately than with the two-wire method. The voltage-sensing leads should be connected as close to the resistor under test as possible to avoid including part of the resistance of the test leads in the measurement.

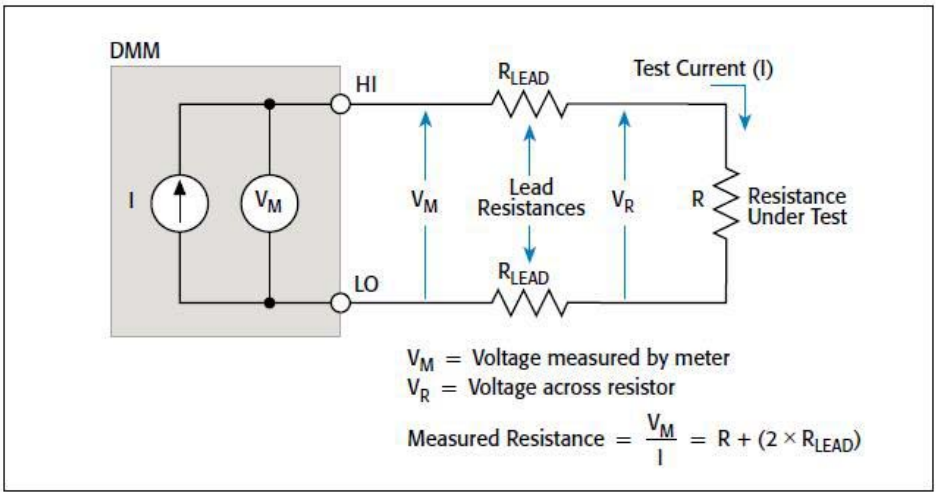


Figure 2.9: Two-wire measure

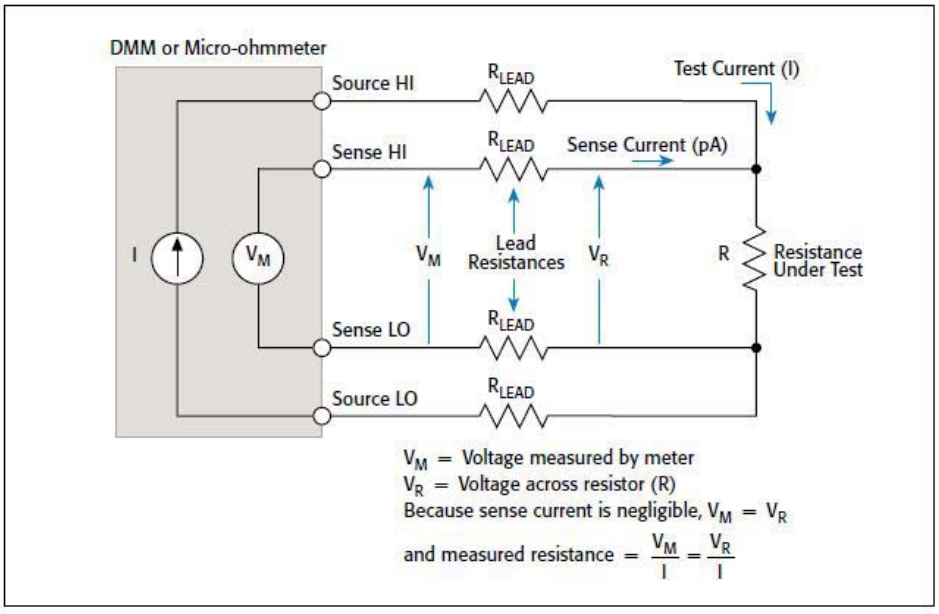


Figure 2.10: Four-wire measure

2.1.4 Daughterboard

The GPB functioning is always completed by the integration of another board, that is specific for each product: the daughterboard. This second board adds flexibility to the GPB, since it is precisely designed for the characterization phase of specific devices. Every type of measure, which is required by the datasheet, needs a series of external components (resistors or capacitors) to be realized. In the same way of the GPB, also the daughterboard is realized from a series of relays, whose function is to activate/deactivate the connection of resistors or capacitors to the device pins.

The Motherboard and Daughterboard relays are managed by an SPI signal of 48 + n bits (n are the number of relays on the daughterboard). MAX4820 relay drivers are mounted on the boards and connected in a daisy-chain structure.

In order to design the daughterboard, *Altium Designer* software is used as the printed circuit board (PCB) design tool. Figure 2.11 shows a 3D image of the PCB daughterbord realized.

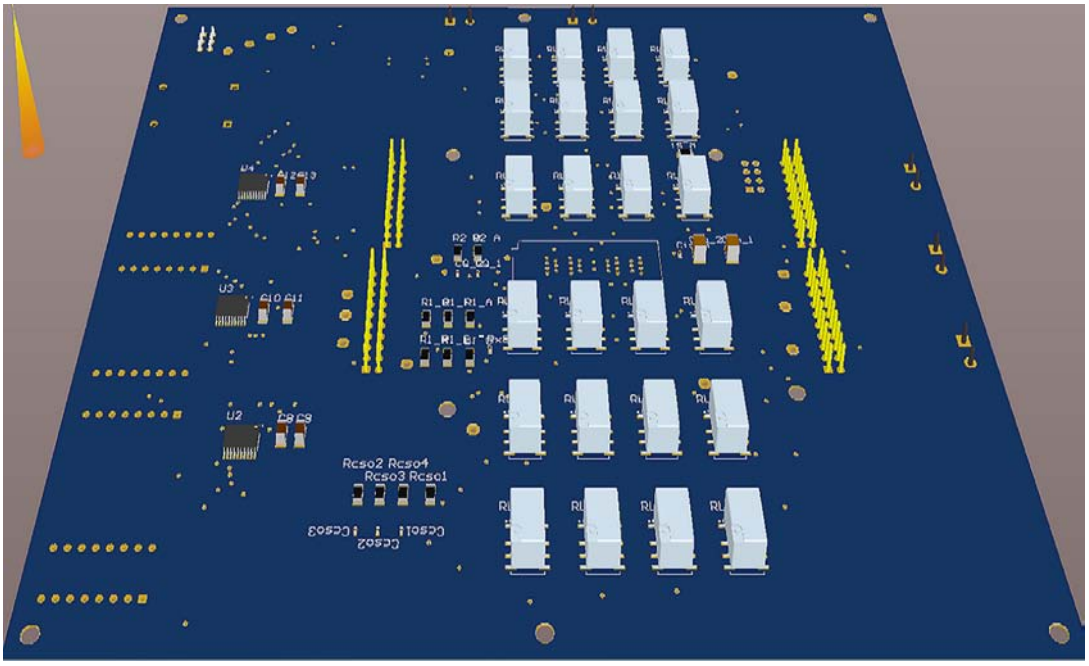


Figure 2.11: Altium 3D view of Daughterboard PCB

2.2 Arduino Due microcontroller

Arduino is an open-source electronics prototyping platform based on flexible, easy to use hardware and software. Arduino can sense the environment by receiving in-

put from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language and the *Arduino development environment*. Arduino projects can be stand-alone or they can communicate with software on running on a computer.

To our scope, *Arduino due* has been used to transmit the SPI commands in order to turn on/turn off the relays of the GPB. It is a USB compatible device which is shown in Figure 2.12.

The *Arduino due* is a microcontroller board based on the *Atmel SAM3X8E ARM Cortex-M3 CPU*. It is the first Arduino board based on a 32 bit ARM core microcontroller. It has a micro USB cable for communicating with a computer, and it also supports SPI communication.

The *Programming port* is connected to an *ATmega16U2 microcontroller*, which provides a virtual COM port to software on a connected. The *ATmega16U2* is also connected to the *SAM3X* hardware *UART (Universal Asynchronous Receiver-Transmitter)* which provides serial to USB communication for programming the board.

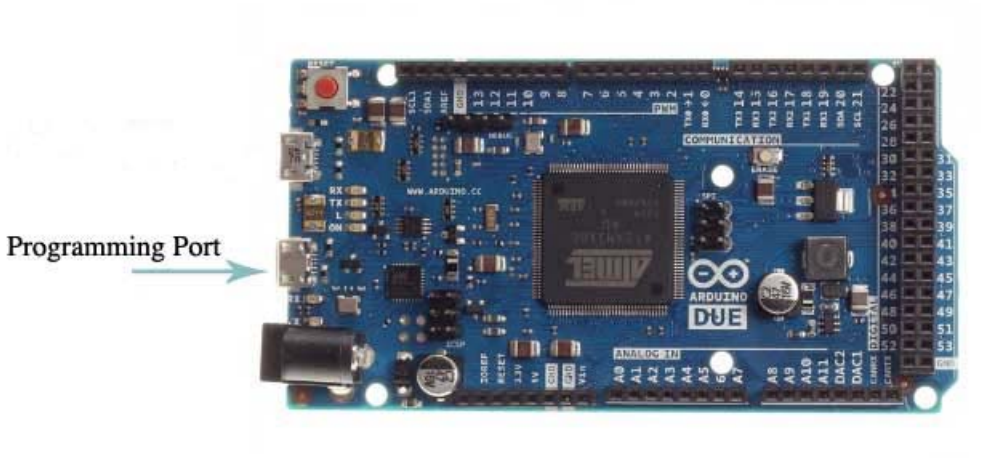


Figure 2.12: Arduino Due microcontroller

To write code and upload it to the I/O board the open-source *Arduino development environment* could be used. It connects to the Arduino hardware to upload programs and communicate with them. The Arduino environment contains a text editor for writing code, a message area and a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the *ATmega16U2* chip and USB

connection to the computer.

2.3 LabVIEW software

In this project, LabVIEW software has been chosen to develop a GUI (Graphical User Interface) to manage measure setups, sending them to the GPB, and to manage instruments for measurements. NI LabVIEW is a graphical programming language designed to develop tests, control, and measurement applications. LabVIEW gives the flexibility of a powerful programming language without the complexity of traditional development environments. LabVIEW delivers extensive acquisition, analysis, and presentation capabilities in a single environment.

2.4 GPIB (General Purpose Interface Bus)

In this project, to communicate between the instrument interfaces realized in LabVIEW and the real instrument have been used the *GPIB General Purpose Interface Bus*.

Hewlett-Packard designed the Hewlett-Packard Interface Bus (HP-IB) to connect their line of programmable instruments to their computers. Today, the name General Purpose Interface Bus (GPIB) is more widely used than HP-IB, and thousands of different instruments are equipped with this interface. The GPIB bus is standardized by the IEEE (Institute of Electrical and Electronics Engineers). The IEEE Standard 488-1975 defined the electrical and mechanical specifications, then the ANSI/IEEE 488.2-1987 standard defined precisely how controllers and instruments should communicate. *Standard Commands for Programmable Instruments (SCPI)* took the command structures defined in IEEE 488.2 and created a single, comprehensive programming command set that is used with any SCPI instrument.

Typically, a measurement setup comprises a system controller (usually a PC or workstation) and at least one device (instrument). The system controller has to have a GPIB controller card.

GPIB is an 8-bit, electrically parallel bus. The bus employs sixteen signal lines eight used for bidirectional data transfer, three for handshake, and five for bus management plus eight ground return lines. Every device on the bus has a unique 5-bit primary address, in the range from 0 to 30 (31 total possible addresses). The standard allows up to 15 devices to share a single physical bus. The physical topology can be linear or star (Figure 2.13). The maximum data rate is about one megabyte

per second. The later HS-488 extension allowing up to 8 Mbyte/s. The slowest participating device determines the speed of the bus.

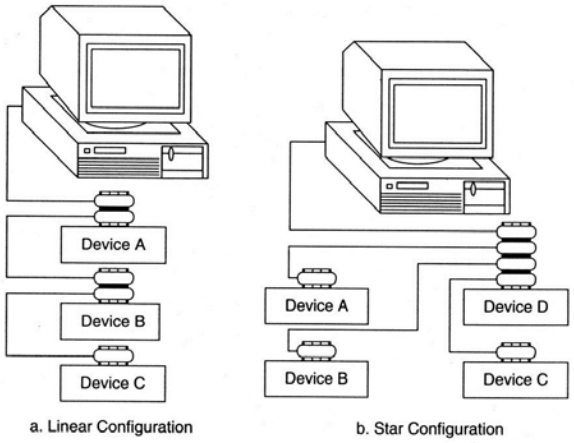


Figure 2.13: GPIB configurations

2.5 SCPI (Standard Commands For Programmable Instruments)

SCPI defines a common command set for programming instruments. Before SCPI, each instrument manufacturer developed its own command sets for its programmable instruments. This lack of standardization forced developers to learn a number of different command sets and instrument-specific parameters for the various instruments used in an application. By defining a standard programming command set, SCPI decreases development time and increases the readability of test programs and the ability to interchange instruments.

The standard specifies a common syntax, command structure, and data formats, to be used with all instruments. It introduced generic commands (such as CONFigure and MEASure) that could be used with any instrument.

SCPI commands are ASCII textual strings, which are sent to the instrument over the physical layer. The physical communications link is not defined by SCPI. While originally created for IEEE-488 (GPIB), it can also be used with RS-232, Ethernet, USB, ecc. Commands are a series of one or more keywords, many of which take parameters. Responses to query commands are typically ASCII strings.

SCPI offers numerous advantages. One of these is that SCPI provides a comprehensive set of programming functions covering all the major functions of an

instrument. This standard command set ensures a higher degree of instrument interchangeability and minimizes the effort involved in designing new test systems. The SCPI command set is hierarchical, so adding commands for more specific or newer functionality is easily accommodated.

Command Syntax

SCPI commands to an instrument may either perform a **set** operation (e.g. switching a power supply on) or a **query** operation (e.g. reading a voltage). Queries are issued to an instrument by appending a question-mark to the end of a command. Some commands can be used for both setting and querying an instrument. For example, the data-acquisition mode of an instrument could be set by using the ACQuire:MODe command or it could be queried by using the ACQuire:MODe? command. Some commands can both set and query an instrument at once. For example, the *CAL? command runs a self-calibration routine on some equipment, and then returns the results of the calibration.

Similar commands are grouped into a hierarchy or "tree" structure. For example, any instruction to read a measurement from an instrument will begin with MEASure. Specific sub-commands within the hierarchy are nested with a colon (:) character. For example, the command to "Measure a DC voltage" would take the form MEASure:VOLTage:DC?, and the command to "Measure an AC current" would take the form MEASure:CURRent:AC?.

Some commands require an *additional argument*. Arguments are given after the command, and are separated by a space. For example, the command to set the trigger mode of an instrument to "normal" may be given as TRIGger:MODe NORMal. Here, the word NORMal is used as the argument to the TRIGger:MODe command.

Multiple commands can be issued to an instrument in a single string. Each command must be separated by a semicolon character (;). Additionally, all commands except the first must be prefixed by a colon (unless they already begin with an asterisk). For example, the command to "Measure a DC voltage then measure an AC current" would be issued as MEASure:VOLTage:DC?;:MEASure:CURRent:AC?.

The command syntax shows some characters in a mixture of upper & lower case. Abbreviating the command to only sending the upper case has the same mean-

ing as sending the upper & lower case command. For example, the command SYSTem:COMMunicate:SERial:BAUD 2400 could also alternatively be abbreviated SYST:COMM:SER:BAUD 2400.

Example

The following command programs a digital multimeter (DMM) to configure itself to make an AC voltage measurement on a signal of 20 V with a 0.001 V resolution.

:MEASure:VOLTage:AC? 20, 0.001

- The leading colon indicates a new command is coming;
- The keywords MEASure:VOLTage:AC instruct the DMM to take an AC voltage measurement;
- The ? instructs the DMM to return its measurement to the computer/controller;
- The $20, 0.001$ specifies the range ($20V$) and resolution ($.001V$) of the measurement;

CHAPTER

3

GRAPHICAL USER INTERFACE FOR MEASURE SETUP

This chapter presents the GUI (Graphical User interface) realized in the first part of this work. This Interface allows the user to manage the measure setup, giving a clear vision of how the GPB (General Purpose Board) is connected to instruments. It also gives the possibility to change setups in real-time and to store settings into an Excel file, so that you can load them whenever you want.

The interface has been realized using LabVIEW. The main VI structure is an *event structure*, which waits until an event occurs, then executes the appropriate case to handle that event. The *event structure* has one or more sub diagrams, or event cases, that are executed in a mutually exclusive manner in accordance with the occurence of external events.

The Figure 3.1 shows a representative image of the developed interface, which is structured by a *tab control*. Tab controls are employed to overlap front panel controls and indicators in a smaller area. A tab control consists of pages and tabs. It is possible to add front panel objects to the pages of the tab control, and to use the tab as the selector to display each page.

In each page of the tab control there is a portion of the PCB (Printed Circuit

Board) of the General Purpose Board. It is populated with relays that control the connections between instruments and device pins.

The first page (I/O connection) is a reproduction of the GPB. In this section it is possible to edit the name of all the instruments connected to the board, and of the device pins. On the right side of the page there are a series of buttons that enhance the functionalities of the GUI.

The main functionalities of the Graphical User Interface are listed below:

1. Check relays status;
2. Load/Save setup measures;
3. Load/Save settings for connectors and pins;
4. Send setups to GPB
5. Quick *Instruments→Device* setup;
6. Select instruments and open pop-ups
7. Check instruments connections at daughterboard level

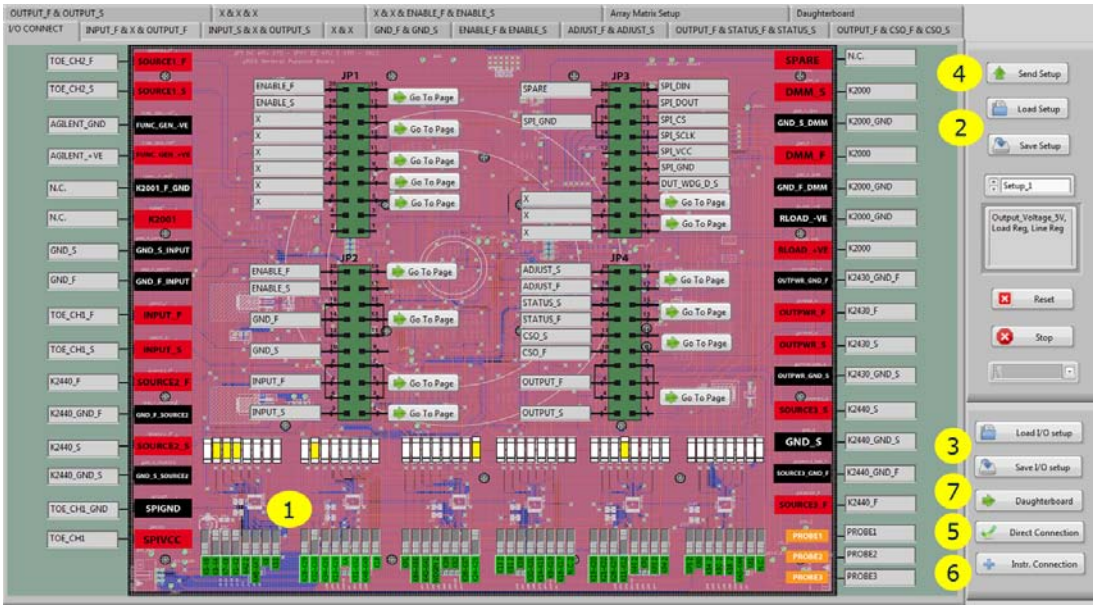


Figure 3.1: Graphical User Interface

3.1 Control of relays status

The GUI, from a graphical point of view, is realized with a tab-control containing the sections of the printed circuit board, which are called with the name of the DUT pins. Each relay is a Boolean control: connections between instruments and device pins can be activated or deactivated by switching the relay state.

For example, as it is shown in the Figure 3.2, the k1&k2 relay is active (green color) and we can see a direct connection between the instrument (TOE_CH1_F) and the device pin (INPUT_F).

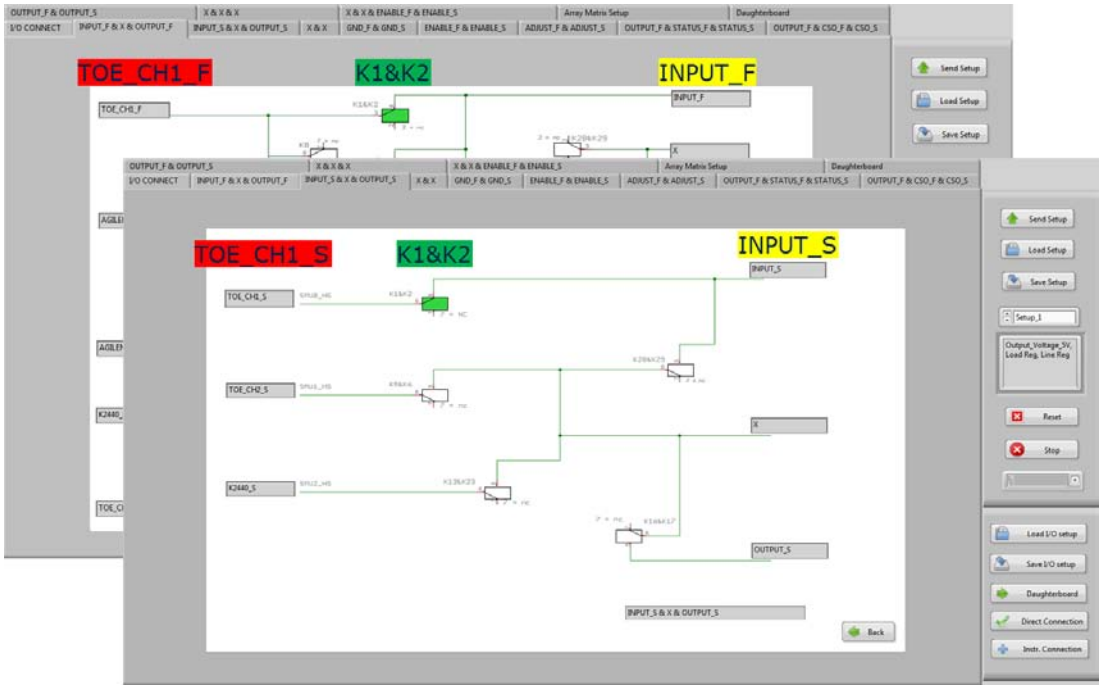


Figure 3.2: Relay k1&k2 active

The GPB has N instruments connected through banana connectors, and for each type of measure, a different configuration of instruments has to be activated. The connection of different configurations of instruments is possible because physical connections between instruments and the device under test are regulated by switching-on/switching-off the relays. For each relays configuration, we will have a different setup of measure, that is a different type of configuration of instruments to realize a specific measure.

Since the board is composed of 48 relays, and their activation/deactivation can be selected by changing the value of a boolean variable, each measurement setup will be described by an array of 48 bits. A "1" logic is equivalent to switch-on a relay and "0" logic to switch-off it. In this implementation we have chosen DPDT

relays (Double Pole Double Throw) therefore it has been necessary to realize two switches, strictly correlated, for each relay. The activation/deactivation of one of these switches will drive the activation/deactivation of the other one, and vice versa.

Usually, a DPDT relay is used to bring FORCE and SENSE signals. To reach more measurement accuracy, it is used to make measures with four wires. Figure 3.2 shows two sections of the GUI, where there are two buttons that represent a single relay, used for the force and sense wires of TOE CH1.

In the interface (Figure 3.1) also dip-switch buttons have been reproduced. The user could manage relay status in two ways: from relays controls in the various printed circuit board sections, or directly from dip-switch controls.

In addition to dip-switch controls also a yellow led is provided for each relay: the led will be ON when the respective relay will be active.

LabVIEW development

To understand how to implement the *Control of relays status* functionality, it is necessary to explain how the measurement setup is kept in memory and how it is updated when a relay changes in value. The idea is to save the setups of measure in a matrix using a FGV (Functional Global Variable). The matrix is composed by n*48 bits, where n is the number of setups and 48 are the bits for each setup.

Functional global variables (Figure 3.3) are VIs that use loops with uninitialized shift registers to hold global data. They are realized by a case structure inside a while loop where the conditional terminal is always true, and every time you call the VI, the code in the loop runs exactly once.

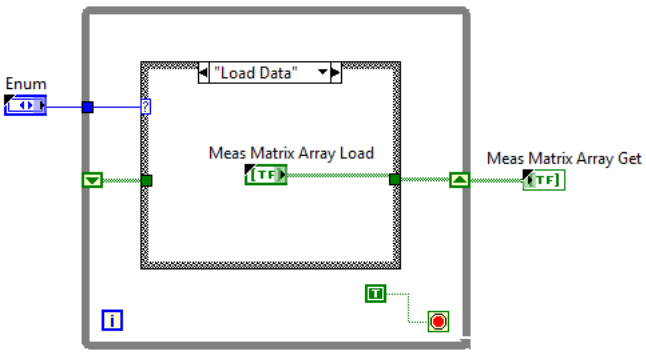


Figure 3.3: Functional Global Variable structure

A functional global variable has an action input parameter (called *Enum* in Figure 3.3) that specifies which task the VI performs. In our case we have three

sections: *Init* to initialize the matrix to zero, *Load Data* to update the matrix with new values and *Get Data* to extract the matrix saved.

When a relay change the value, it will be necessary to update the FGV. The Figure 3.4 shows the event generated by the relay kA_A. The relay kA is represented of two controls strictly related, kA_A and kA_B, which have always the same Boolean value because they represent the same physical relay.

The LabVIEW code in the Figure 3.4 can be described as follow:

1. *Get Data* extracts the current matrix saved in the FGV;

2. *Enum* selects the row of the matrix to identify the measurement setup;

3. *Index Array:* Returns the element or sub array of n-dimension array at index;

4. *Replace Array Subset:* updates the bit associated to the relay pressed. It replaces an element or sub array in an array at the point you specify in index.

5. *Load Data:* updates the matrix saved in the FGV;

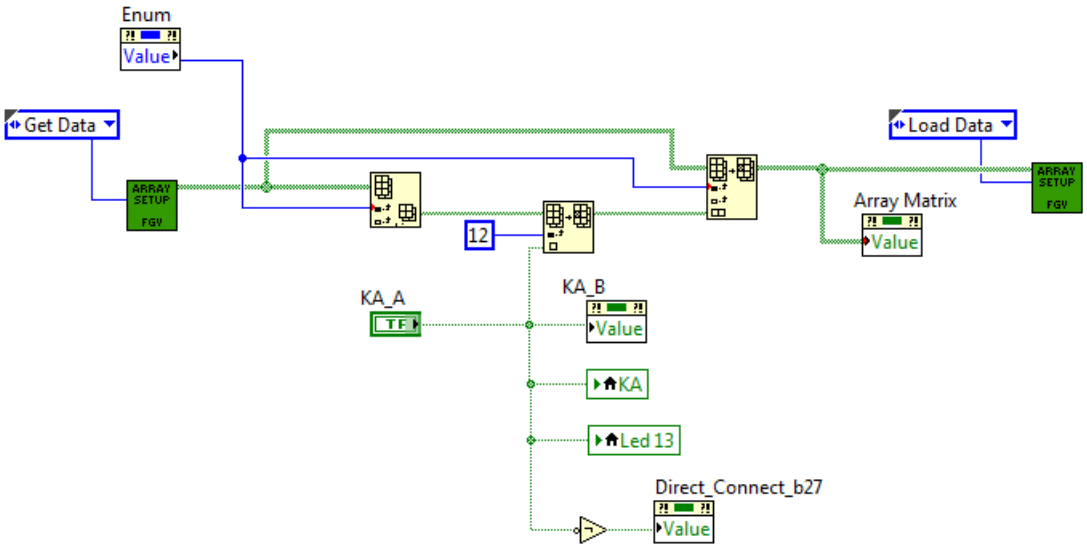


Figure 3.4: LabVIEW code for updating matrix setup

As already seen, the GPB has 48 relays, and the main VI has about 80 events in the event structure, because there are almost two control buttons for each relay. These events have the same structure, as shown in the Figure 3.4.

3.2 Load/Save setup measures

The functionality of Load/Save setup, allows you to load from an Excel file a matrix of measurement setup, including a brief description for each setup (e.g. Output Voltage 5V, Load Reg, Line Reg). The Figure 3.5 shows the section of the GUI that summarizes the matrix of loaded setup.

Through the interface, it will be possible to change the setup quickly, using relay or dip-switch controls. Moving from one setup to another, the state of controls, relays, dip-switches and LED indicators will change.

After loading setups, it will be possible to modify them, and to save them in an Excel file with the save setup button. It is also possible to modify and to save new measurement setup directly from the Excel file.

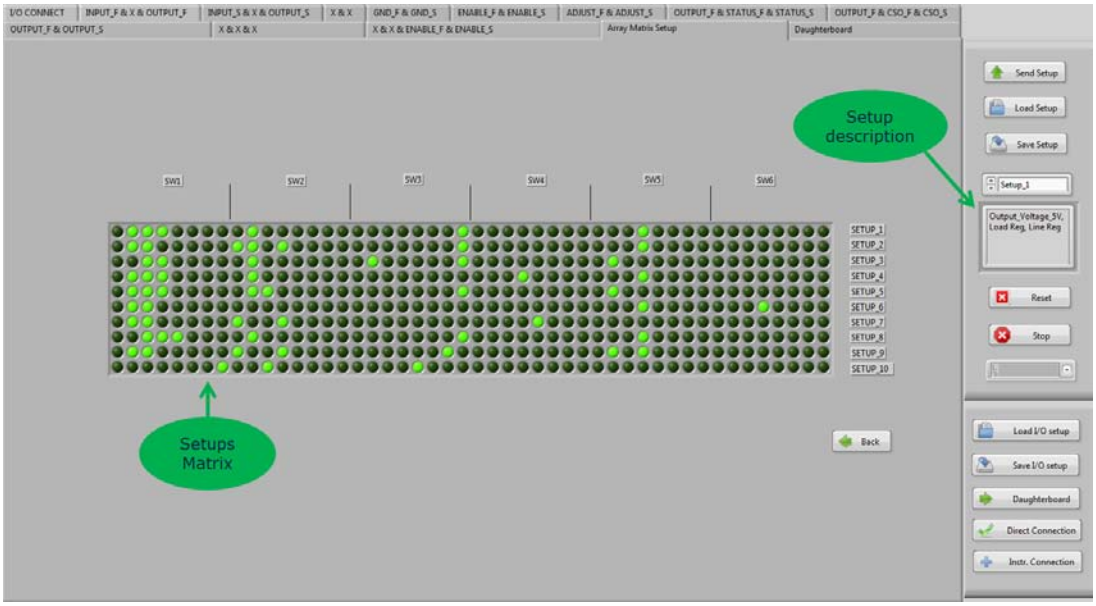


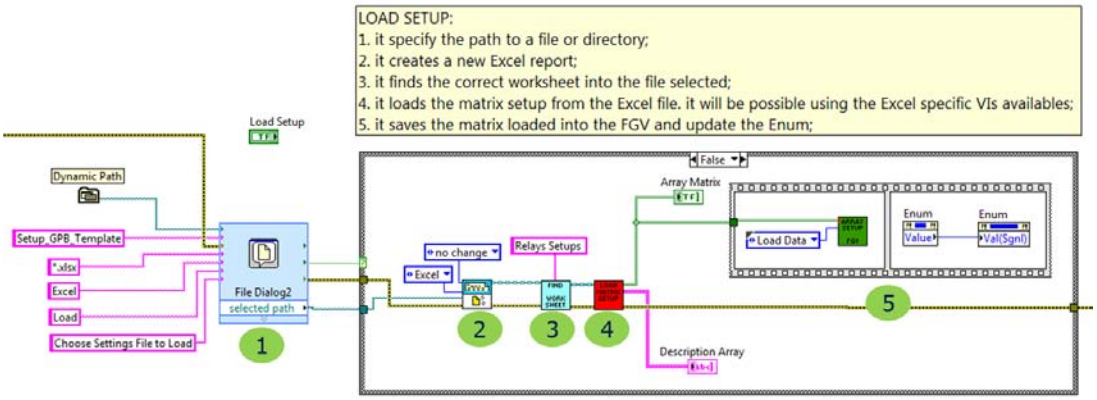
Figure 3.5: GUI section that represent the Matrix of setups

LabVIEW development

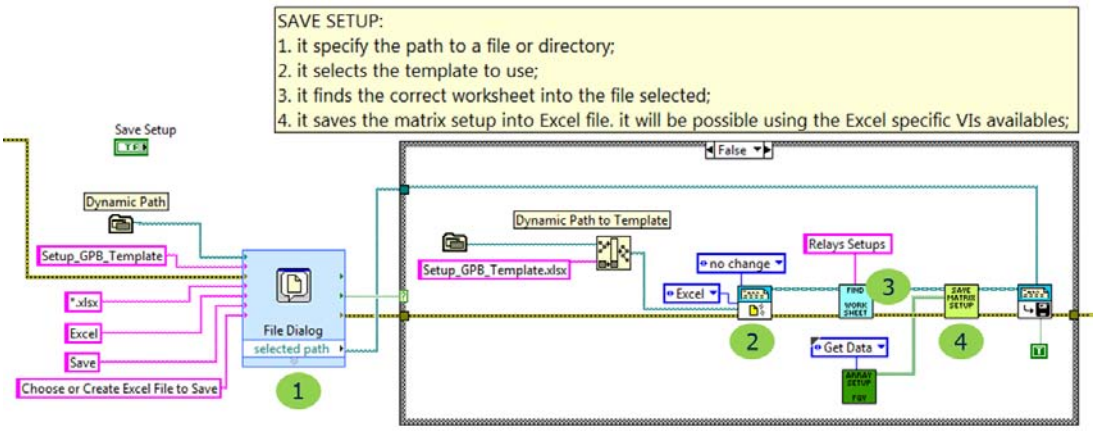
To implement this functionality, a customized Excel file with two worksheets has been created, one for I/O connectors and pins and the other one for matrix setup.

LabVIEW 2013 gives the possibility to use the *Excel specific VIs* to incorporate Microsoft Excel features into LabVIEW reports.

The Figure 3.6 shows the code structure of the *Load/Save Setup* function, accompanied by a brief description. The corresponding VIs are described below:



(a) Load setup



(b) Save setup

Figure 3.6: LabVIEW code for Load/Save setup measures

- *File Dialog:* This VI displays a dialog box by which you can specify the path of a file or directory. You can use this dialog box to select existing file or directories or to select a location and a name for a new file or directory;
- *Find the worksheet:* this VI is realized to find the correct worksheet (Connectors&Pin or Relays Setups);
- *Load Matrix Setup:* this VI returns the matrix of setups, included a description for each setup;
- *Save Matrix Setup:* this VI receives the matrix of setups and save it into the Excel file;

To complete this topic, it is necessary to explain in which way the created sub-VI are realized, and which VI can be used among the available ones, in order to interact with the Excel file.

Figure 3.7 shows the code and a description for the sub-VI *Find Worksheet*. Excel specific VIs are described below:

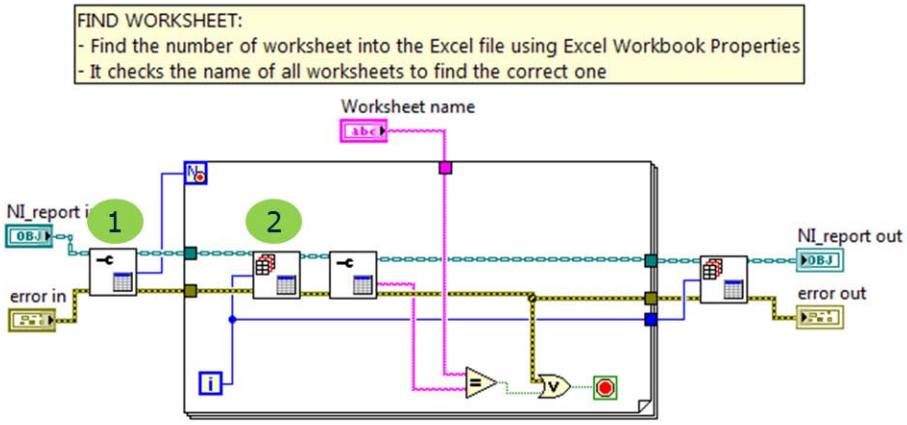
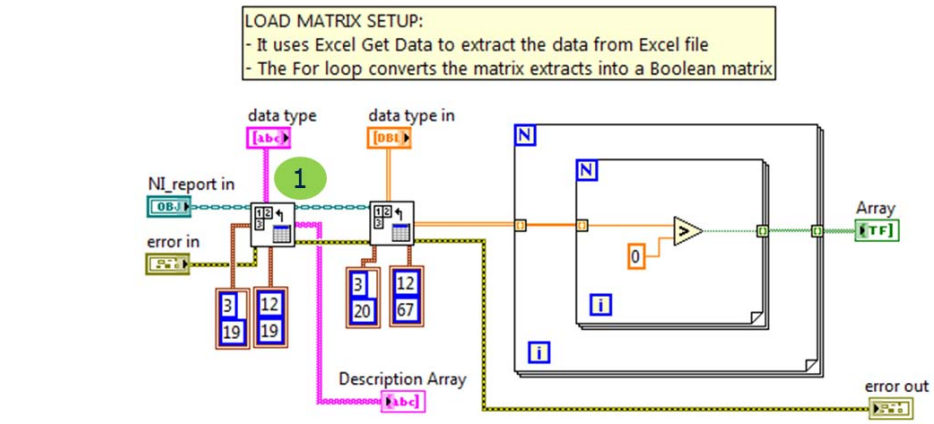
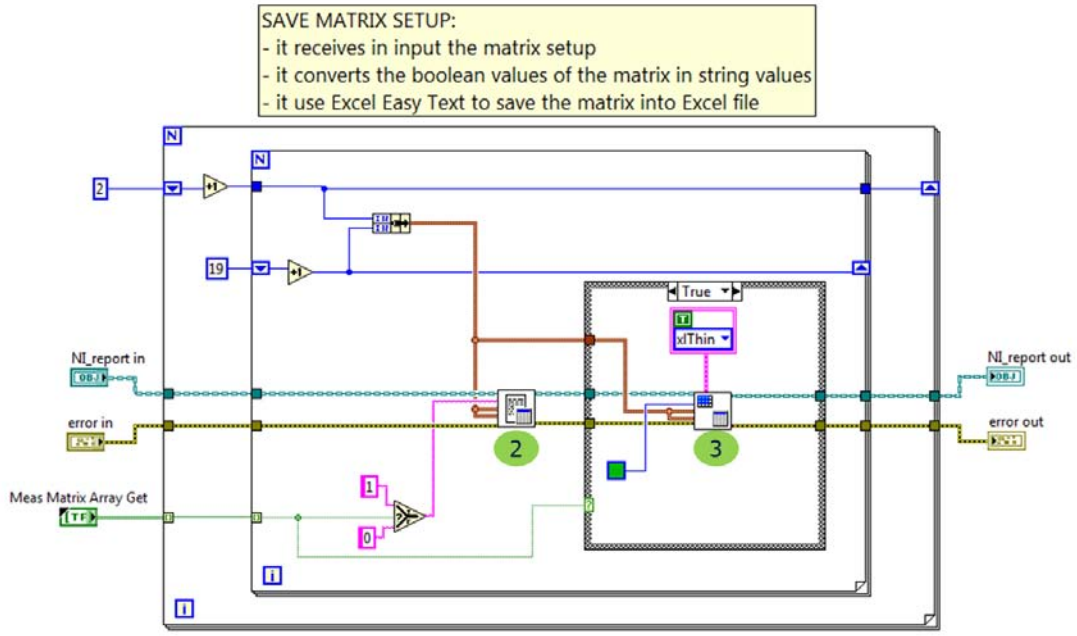


Figure 3.7: LabVIEW code for Find Worksheet sub-VI

1. *Excel Workbook Properties:* this VI sets the properties of the current workbook, such as author, title, and subject. You also can use this VI to return the number of worksheets and the worksheet name in the current workbook.
2. *Excel Get Worksheet:* this VI makes a specified worksheet the current worksheet. It is possible to use the worksheet index or name parameter to specify the worksheet you want to set as current.



(a) Load matrix setup



(b) Save matrix setup

Figure 3.8: LabVIEW codes sub-VI Load/Save matrix setup

Figure 3.8 shows the *Load Matrix Setup* and *Save Matrix Setup* VIs.
Excel specific VIs are described below:

1. *Excel Get Data:* this VI retrieves data from the current worksheet. The data type you wire to the data type input determines the polymorphic instance to use. If you want to return a numeric value, you have to use the start cluster or the name input to specify the cell from which you want to return the value. If you want to return an array, you have to wire both the start and end clusters or the name input to specify the range from which you want to return the array. If the *start* and *end* clusters remain unwired, the VI returns the entire used section of the current worksheet.

2. *Excel Easy Text:* it is possible to use the font parameter to format the text. The VI merges a range of cells specified by start and end inputs and inserts the text into the merged cell.

3. *Set Cell Color and Border:* Sets the borders and background color of the cells specified by the start and end cluster or by a named range.

3.3 Load/Save settings for connectors and pins

As shown in the Figure 3.9, the GUI gives a clear representation of which instruments are connected to the General Purpose Board. The interface allows users to edit the name of instruments connected to the board and also to label the device pins. Labels associated to each instrument and to each pin will be automatically updated in each section of the GUI. In this way, when a relay is activated, it will be immediately clear which is the connection (es: Toellner Ch1 to Input pin). An example of this functionality is shown in Figure 3.9, which illustrates the relationship between relay controls, dip switches and led indicators.

A user is able to load settings from an Excel file, edit and save new settings or modify existing setting directly from the Excel file, as previously illustrated for the measurement setup function.

LabVIEW development

To implement the *Load I/O setup* and the *Save I/O setup* functionalities, the same approach used for the *Load/Save of setup measure* has been adopted. In this case the *Load In Out Setup VI* and *Save In Out Setup VI* have been realized.

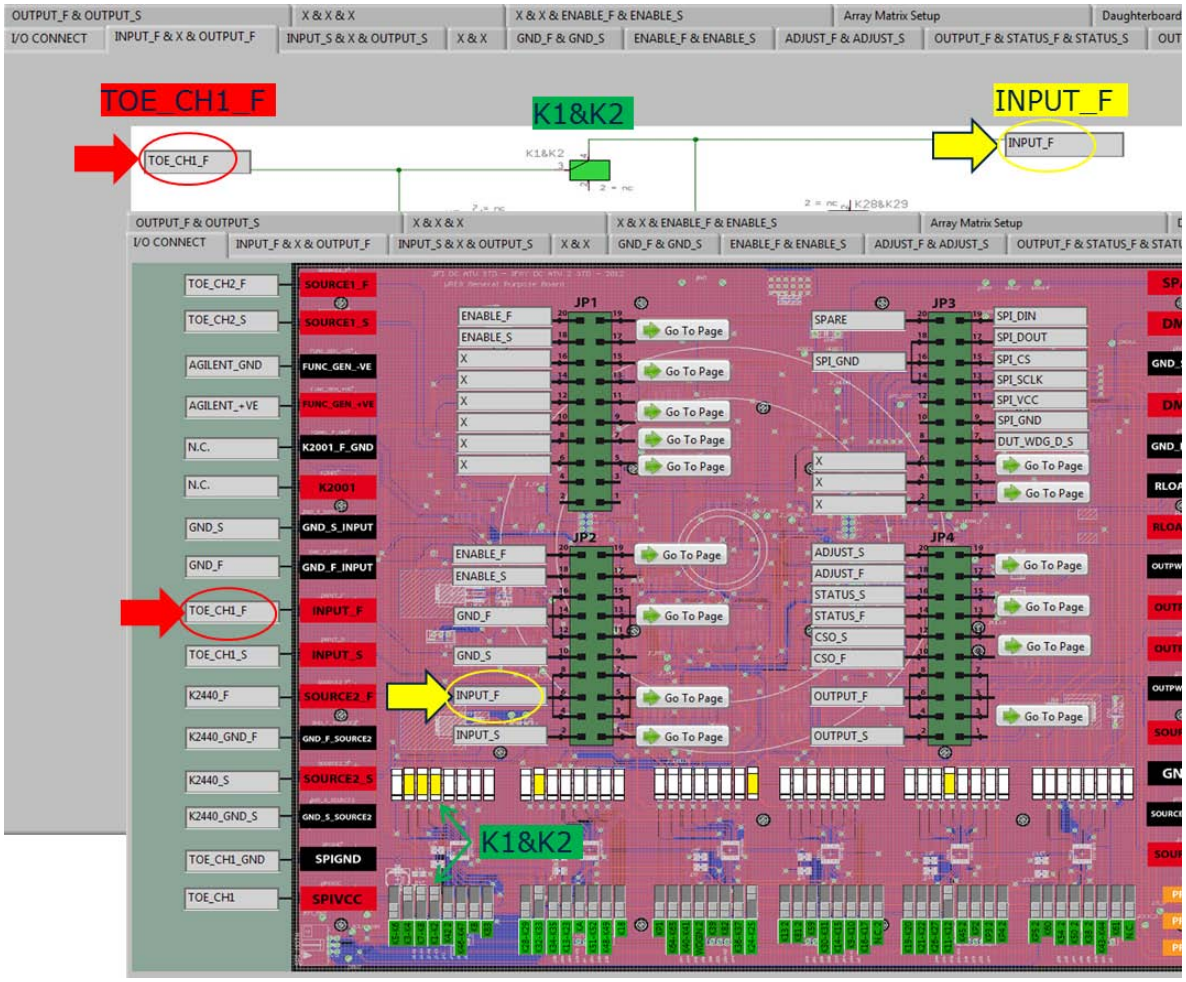
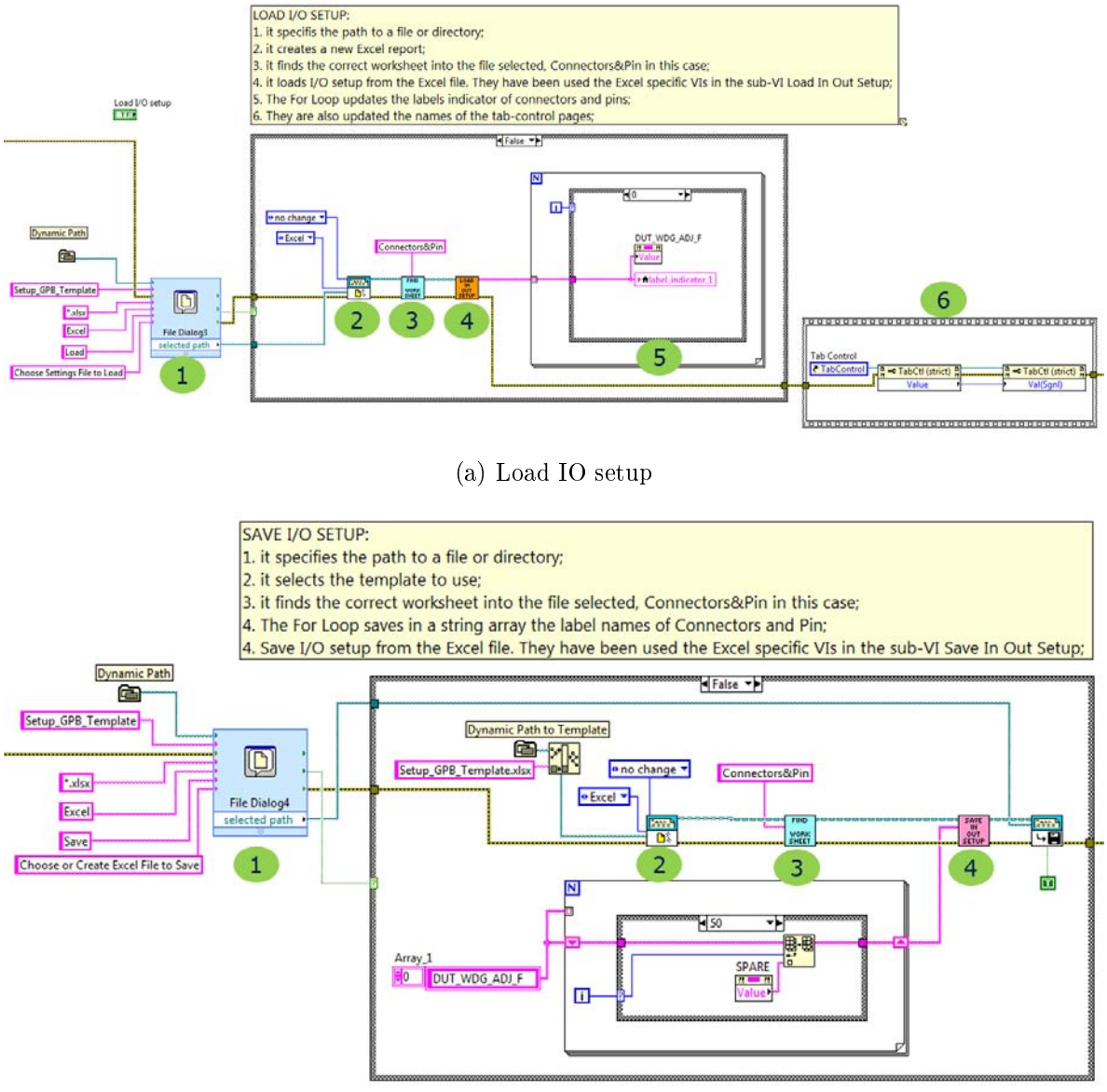


Figure 3.9: Connectors&Pins functionality

Below (Figure 3.10) it is reported the LabVIEW code developed for the implementation of this functionality. Excel VIs are the same described in the previous section.



(a) Load IO setup

(b) Save IO setup

Figure 3.10: LabVIEW codes sub-VI Load/Save IO setup

3.4 Send setups to GPB

The *Send Setup* button sends the setup to the boards (GPB + Daughterboard) as an array of (48 + n) bits. In this process, LabVIEW program converts the array

into a string for the USB interface. The USB is directly connected to the *Arduino due microcontroller*, which communicates with the GPB by an SPI interface.

The Figure 3.11, shows the VI used to convert the array of bits into a string.

1. The size of *Array in* is divided by 8 and the result is a counter for the *For Loop*;

2. In each iteration of the loop, 8 bits are processed and converted into a number using the *Boolean Array To Number* VI. It converts a Boolean array to an integer or a fixed-point number by interpreting the array as the binary representation of the number. If the number is signed, LabVIEW interprets the array as the two's complement representation of the number. The first element of the array corresponds to the least significant bit in the number.

3. At the end of the For Loop, the *Byte Array To String* VI converts the array of unsigned bytes representing ASCII characters into a string.

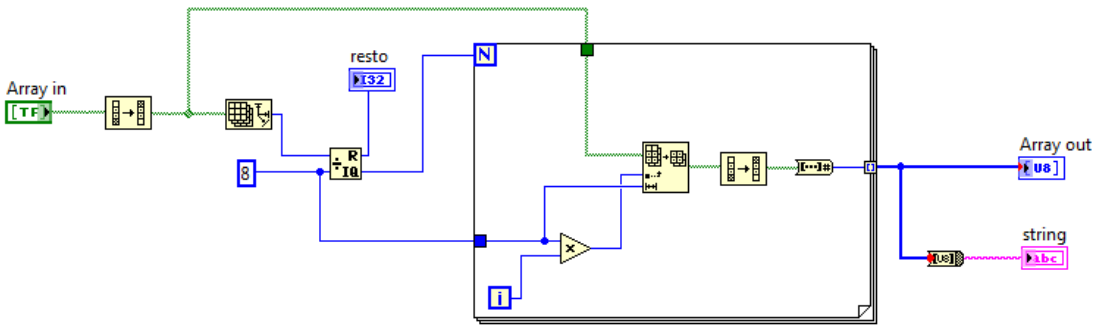


Figure 3.11: LabVIEW code to convert array bits into string

3.5 Quick Instruments → Device setup

The GUI is provided also with a special button, the *Direct Connection* button, dedicated to quickly activate/deactivate connections between instruments and device pins. When you push this button, next to the device pins, the interface will populate of coloured buttons, and the labels associated with the instruments will take a different color for each instrument (Figure 3.12).

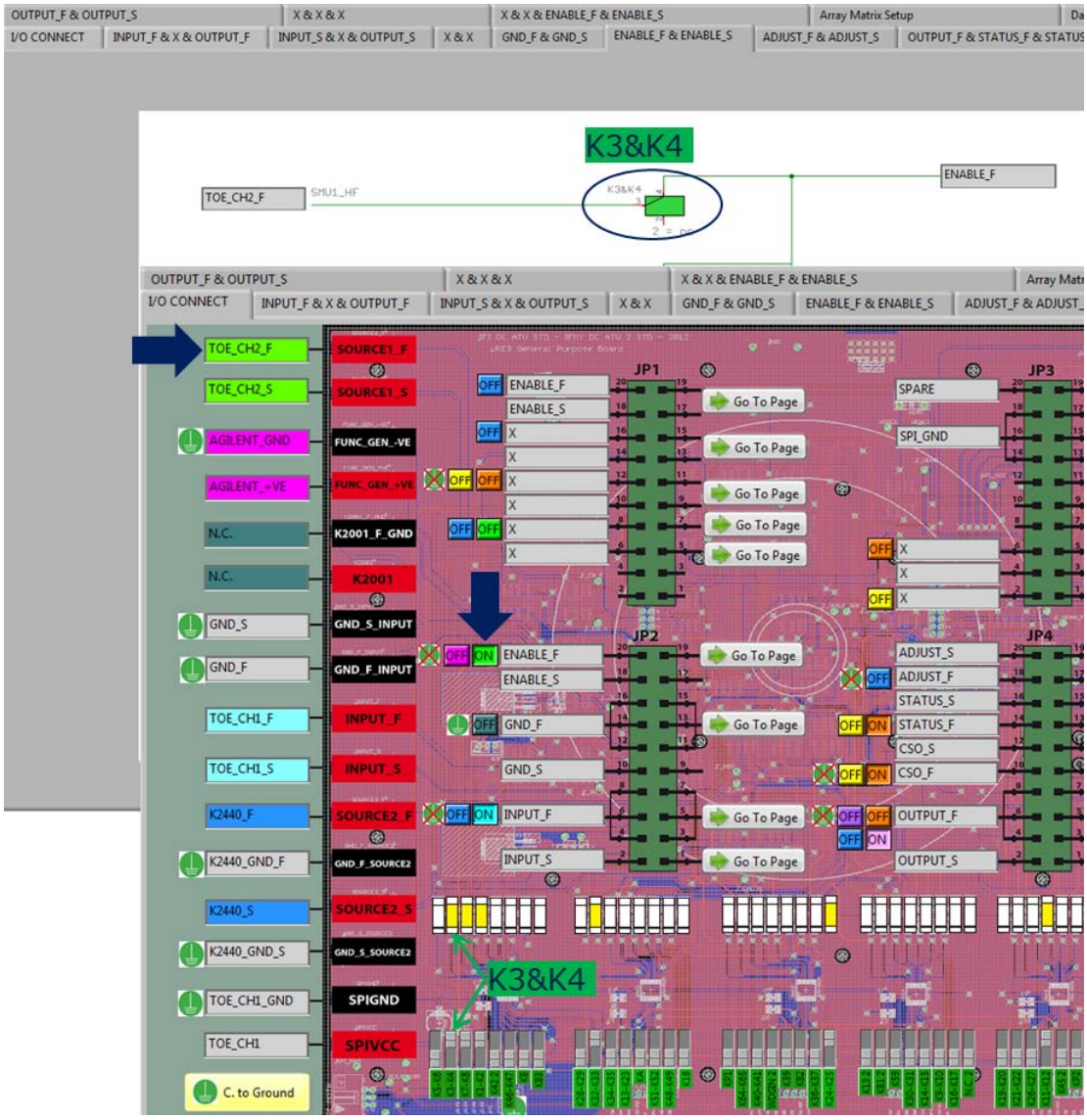


Figure 3.12: Direct Connection functionality

Colours are associated with the possible connections between instruments and device pins. Activating (ON) a coloured button, relays that need to be used for performing the connection will be switched-on. This functionality allows to see immediately if there is a direct link between instruments and device pins and to activate/deactivate connections using the coloured buttons mentioned above. The *Connect to Ground* button, switches-on the relay necessary to link the ground pin of the device to the ground plane of the board.

The blue arrow in the Figure 3.12 shows that TOE_CH2 is connected to the ENABLE pin by relay k3&k4. This connection is reached activating the green on/off button, or, as already seen, using relays or dip-switch controls.

LabVIEW development

The *Direct Connection* functionality is realized in LabVIEW by using the *Property Node* function. It gets/reads and/or sets/writes properties of a reference. It is possible to use the Property Node to get or set properties and methods on local or remote application instances, VI, and objects.

When the *Direct Connection* button will be pressed, using the Property Node functionalities (Value, Visible, TextColor, Value(Sgnl)), it will be possible to make visible or invisible the coloured buttons and to paint the labels of instruments. These coloured buttons are strictly correlated with relays and dip-switch controls, in fact, they allows to activate/deactivate the connections between instruments and device pins.

If a coloured button is pressed, it will be activated/deactivated the relay used for the connection. This is done using the Value(Sgnl) property. It sets the value of the control and generates a value change event. The Figure 3.13 shows a portion of LabVIEW code developed for this task.

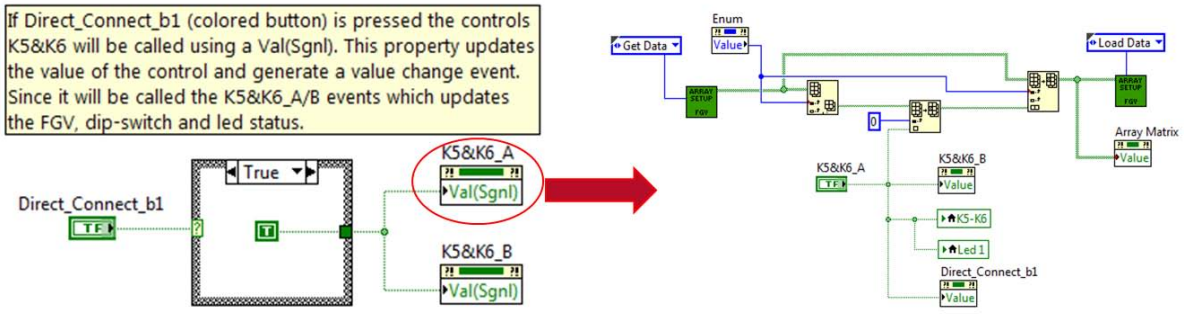


Figure 3.13: Example of LabVIEW code for Direct Connection functionality

3.6 Select instruments and open pop-ups

Instrument selections and their corresponding setup by pop-up menu is allowed by the activation of the *Instr. Connection* button. When it is *on*, as shown in the Figure 3.14, the interface will be populated with a series of drop-down menu and buttons, next to instruments labels. In this way, it will be possible to select from the drop-down menu the instrument we want to connect to a specific *banana connectors* of the GPB, and by the + button associated to each drop-down menu, to open the selected instrument pop-up. This functionality allows the realization of the setup of instruments to be used for all the measurement needed.

Figure 3.14 shows the drop down menu where it is possible to select instruments, and an example of the Toellner pop-up, opened using the + button. The *TOE* interface allows a complete communication with the instrument. It is also possible to close the pop-up and to open again it to change settings, if necessary.

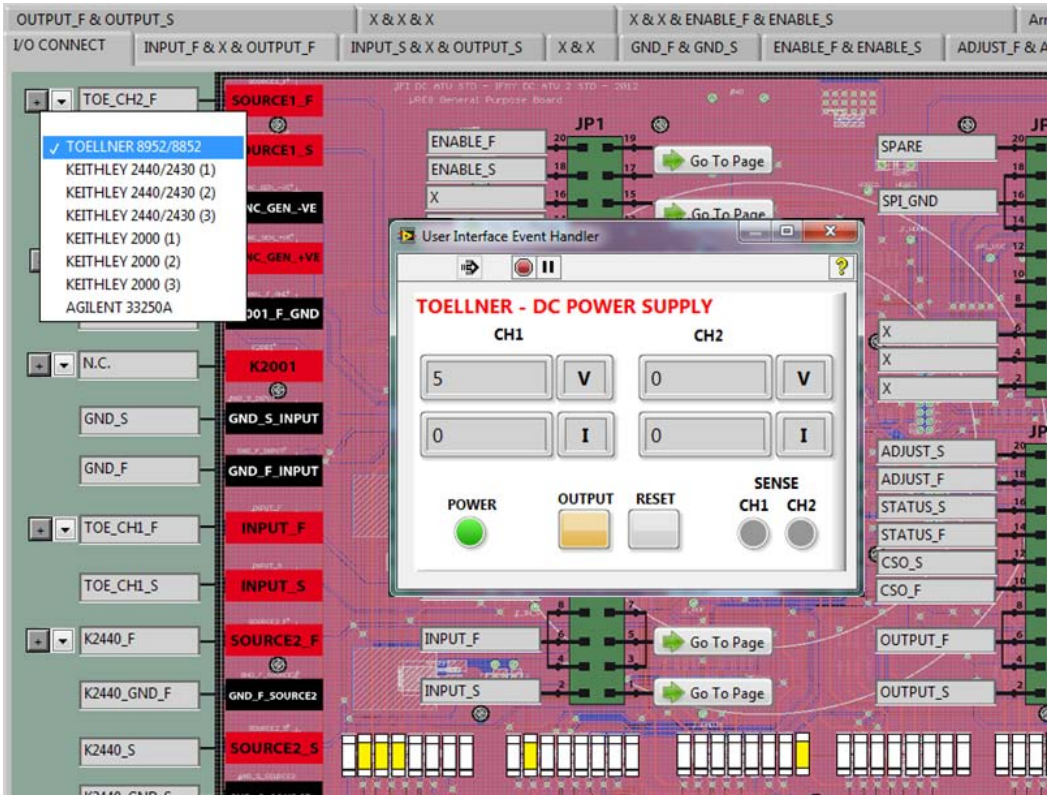


Figure 3.14: Select instrument and open pop-up

LabVIEW development

The part concerning instruments pop-ups will be more exhaustively described, in the next chapter. The Figure 3.15 shows a portion of LabVIEW code to select instruments and open the related pop-ups.

Select_Inst_1 is a drop down menu control which contains the name of instruments that can be selected. When *Button_Istr_C1* (+ button in the GUI) is pressed, an event will be generated and the instrument pop-up is opened.

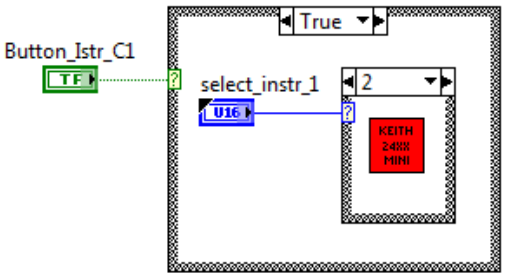


Figure 3.15: Example of LabVIEW code to select and to open instrument pop-ups

As shown in the figure below, it is necessary to select, from the category pull-down menu, the *Window Appearance* option , and then the *Custom* option. Clicking on *customize* button a new window will appear, that allows to customize the windows appearance for VIs. It is possible to use the options of this window to change how the user interacts with the application by restricting access to LabVIEW features and by changing the way the window looks and behaves. These windows make possible to open another VI, for example an instrument pop-up, when the main VI is already working.

As shown in the Figure 3.16, it is necessary to select, from the category menu, the *Window Appearance*, and then the *Custom* option. In this way it is possible to open another VI, for example an instrument pop-up, when the main VI is already working.

3.7 Check instrument connections at daughterboard level

As already seen, the GPB functioning is always completed by the integration of another board, that is specific for each product: the daughterboard. This second

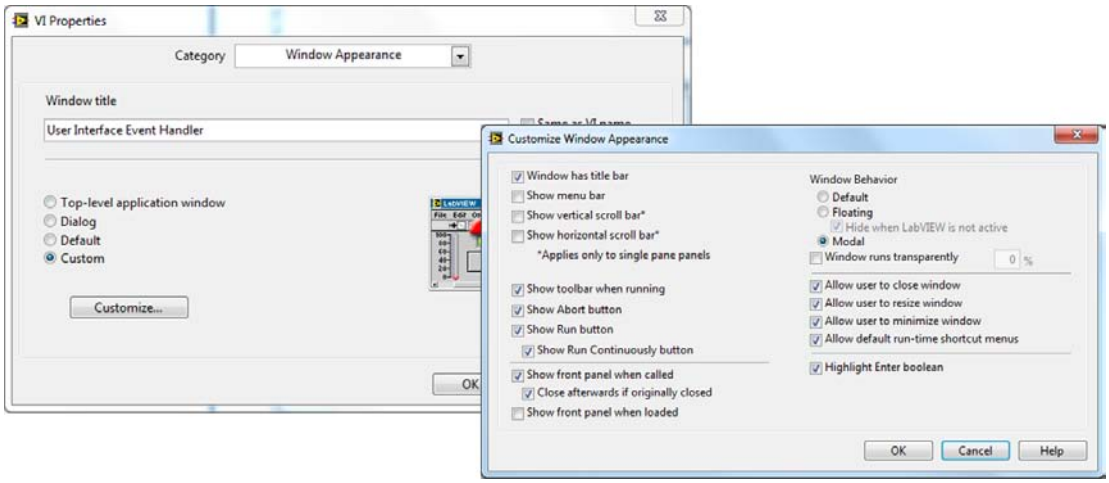


Figure 3.16: VI properties for Windows Apperance

board adds flexibility to the GPB, since it is precisely designed for the characterization phase of specific device (voltage regulators).

The GUI includes a specific section dedicated to the daughterboard. As shown in the Figure 3.17, this section is provided with an image of the PCB, and, for each selected measurement setup, it is possible to see clearly the connections between instruments and device pins. If you change measure setups, or modify the state of some relays, the visualized configuration of connected instruments will be changed. It is also possible to recognize conflicts between instruments wrongly connected to the same pin, thanks to pop-ups that warn the user.

The interface is also provided with a + button next to each instrument indicator, which allows to open instruments pop-ups.

LabVIEW development

To implement this functionality it is required to understand how the code finds the instruments connected to the GPB. When the *Daughterboard* button is pressed, it generates an event. The program searches for instruments connected to the board, analyzing the measurement setup. As already discussed, each daughterboard is designed for a specific product/device and, for this reason, it is possible in advance to know exactly the possible connections *instruments* $\rightarrow$ *device pins*. It has been necessary to develop an algorithm to find the configuration for each selected setup.

The code is realized with a For Loop, which analyzes the possible instruments connected to the device pins, specifically, analysing one pin for each iteration of the loop.

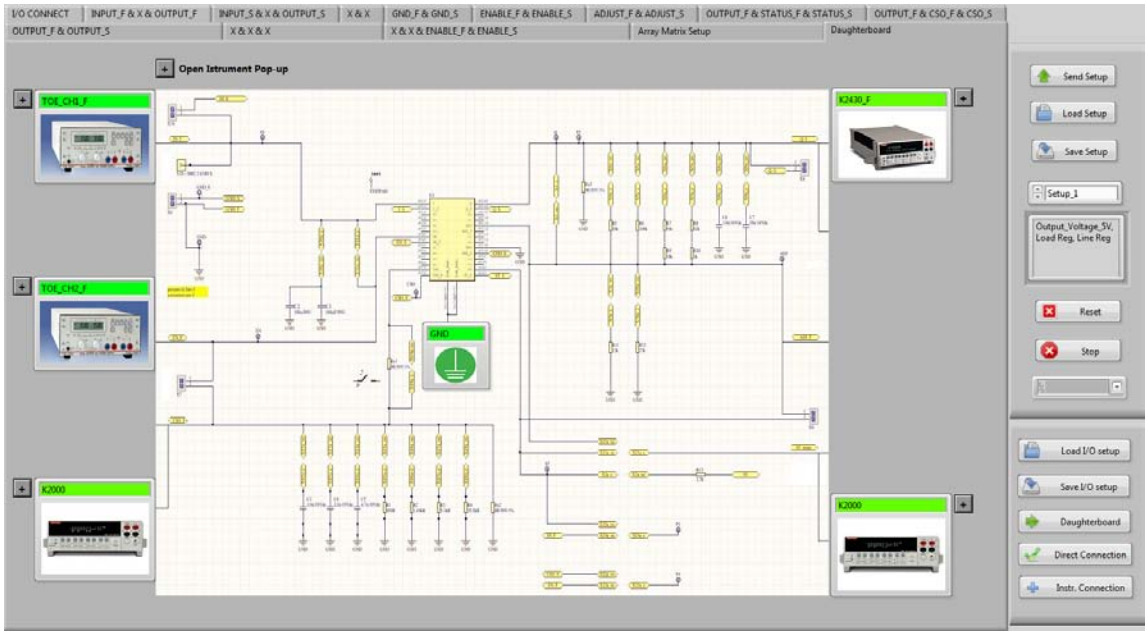


Figure 3.17: Example of *measure setup* viewed from Daughterboard section

For example, the Figure 3.18 shows the code developed for the *Input Pin*. The algorithm used is the same for each pin.

The *Input Pin* could be connected to two instruments (*Toellner* or *Keithley 2440*) or, eventually, to the *gnd*. When a specific set up is selected, in order to find which is the connection between those above mentioned, a For Loop looks for which relays (between instruments and the input pin) are activated. This Loop creates a Boolean array which will be converted into a integer, using the *Boolean Array To Number* VI. As shown in the table below, one of the instruments, the ground (GND), or an error message is associated to each integer. When the case structure receives an integer, it will select the corresponding case that will be visualized on the daughterboard section of the GUI. If there are no instruments connected to the *Input Pin*, no instruments indicators will be represented on the interface.

GND	TOE	K2440	Integer	Visualization
0	0	0	0	No Istr.
0	0	1	1	K2440
0	1	0	2	TOE
0	1	1	3	Conflict pop-up
1	0	0	4	GND
1	0	1	5	Conflict pop-up
1	1	0	6	Conflict pop-up
1	1	1	7	Conflict pop-up

3.8 Excel file settings

As already discussed in section *Load/Save of setup measures* and *Load/Save of settings for Input/Output connectors and pins*, it is possible to save and load setups into/from an Excel file.

A customized Excel file with two worksheets has been created, one for the management of the I/O connectors and pins, and the other one for the management of the setup measures matrix.

The user could modify the settings directly from the Excel file and also could load them, while the VI is working. Vice versa it is possible to modify the settings using the GUI (using editable labels for I/O connectors and pins, and relay controls for the matrix setup), and to save new settings by the Save buttons. Figure 3.19 and Figure 3.20, shows Excel worksheets.

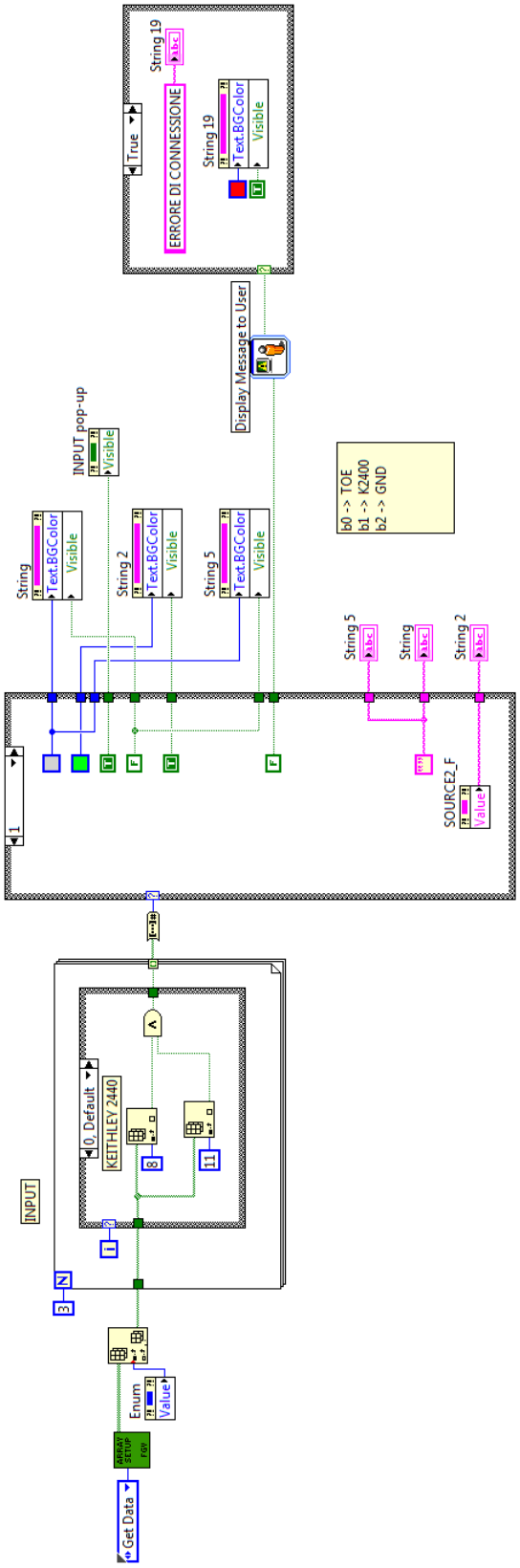


Figure 3.18: Portion of LabVIEW code used to find the *mesure setup* at daughterboard level

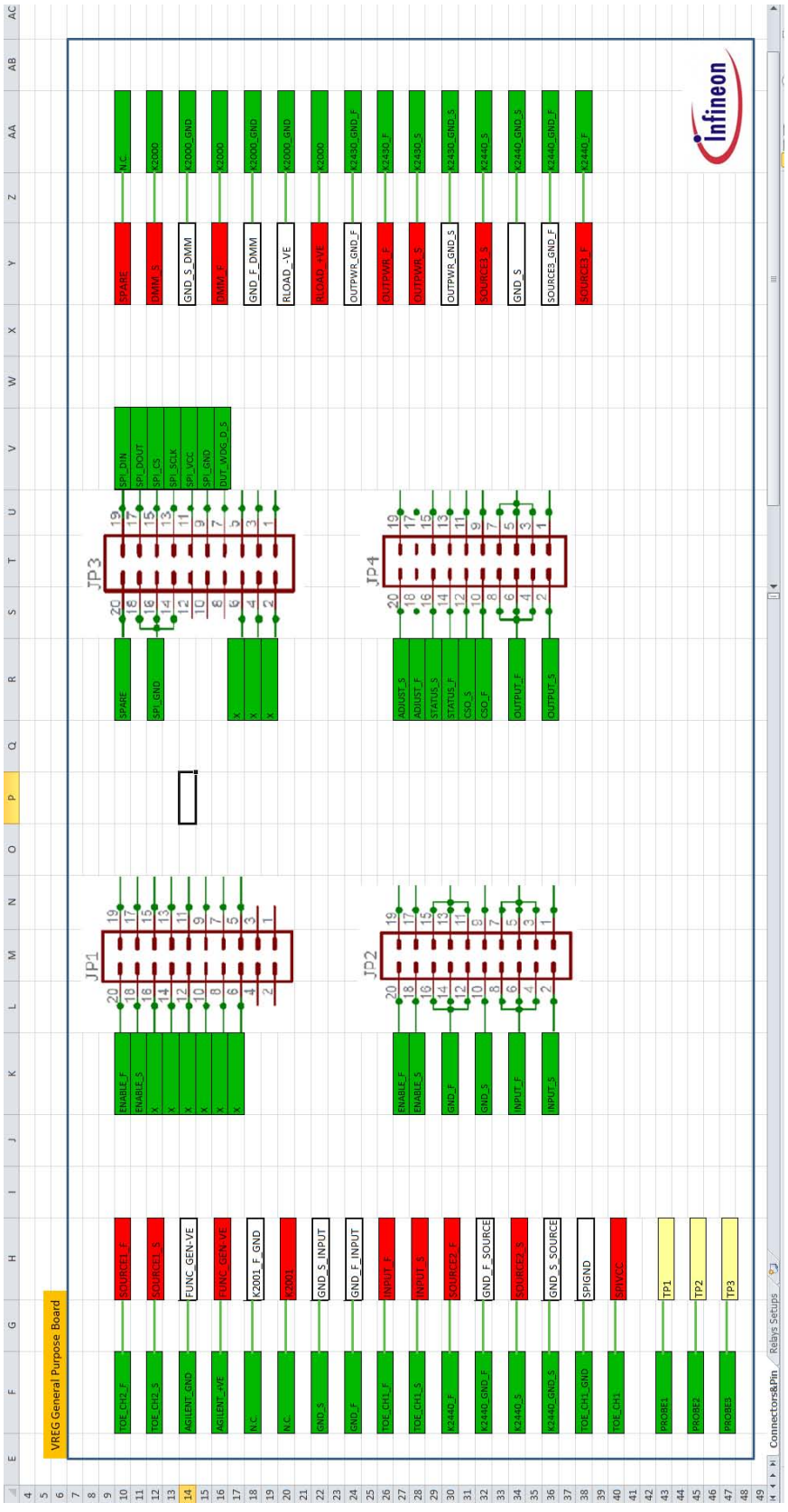


Figure 3.19: Excel worksheet to manage Input/Output connectors and pins

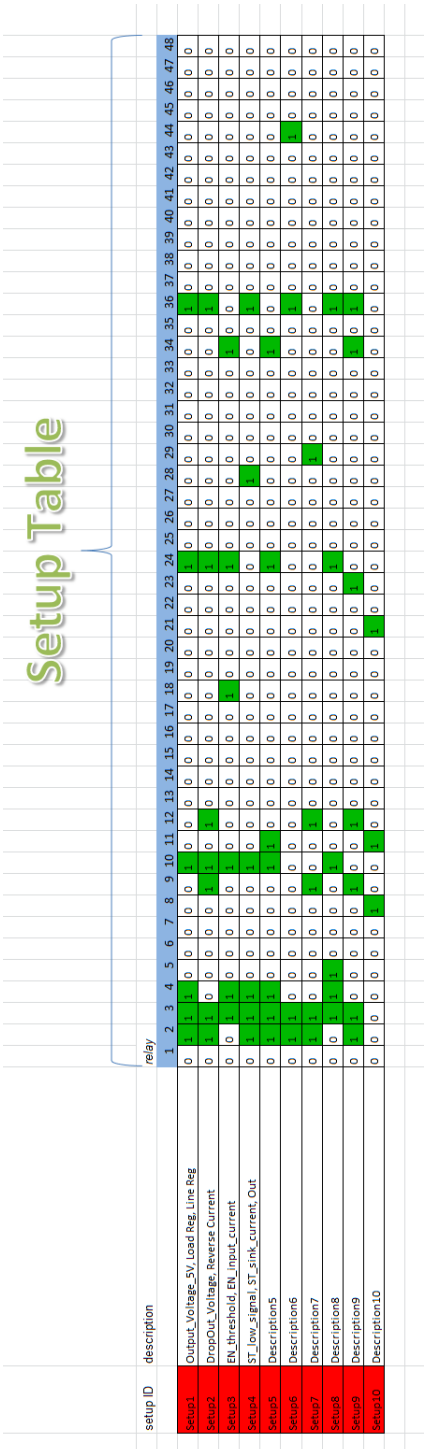


Figure 3.20: Excel worksheet to manage measurement setup matrix

CHAPTER

4

INSTRUMENTS POP-UPS

This chapter deals with the second part of the project concerning the development of GUIs for instruments utilized during the characterization of device under test. The goal of these GUIs is mainly to provide a means for remotely controlling and setting up instruments connected to the GPB.. This opportunity results to be very useful, for example, during a debug phase.

As already discussed in the previous chapter, the GPB GUI has the possibility to open instruments pop-ups, which can control and communicate with real instruments. Instruments pop-ups realized are:

1. Toellner 8952/8852 – DC Power Supplies
2. Keithley 2430/2440 – Source Meter
3. Keithley 2000 - Multimeter
4. Agilent 33250A – Waveform generator

The bus used for connecting pc to instruments was the GPIB (General Purpose Interface Bus), that is standardized by the IEEE. Commands used for communicating with instruments are provided by ASCII strings, conforming the standard

SCPI (Standard Commands for Programmable Instruments). The SCPI defines a standard for syntax and commands to use in controlling programmable test and measurement devices.

To develop instruments pop-ups in LabVIEW, the approach used has been equivalent for each instrument. The code has been structured in a modular way. The main VI structure is an event structure, which calls the FGV VI (Functional Global Variable VI) when an event occurs, to send a specific command to the interested instrument (for example *set V* for a *Toellner*). The FGV is used to manage the communication and to keep in memory the values utilized during the phase of connection of the instrument interface to the real instrument (for example: GPIB address).

The FGV structure is shown in the Figure 4.1. It uses loops with uninitialized shift registers to hold global data. This is realized by a case structure inside a while loop, where the conditional terminal is always true, and every time a VI is run, the block diagram in the loop is executed exactly once. A *functional global variable* has an action input parameter (called Operators in the Figure 4.1) that specifies which task the VI performs.

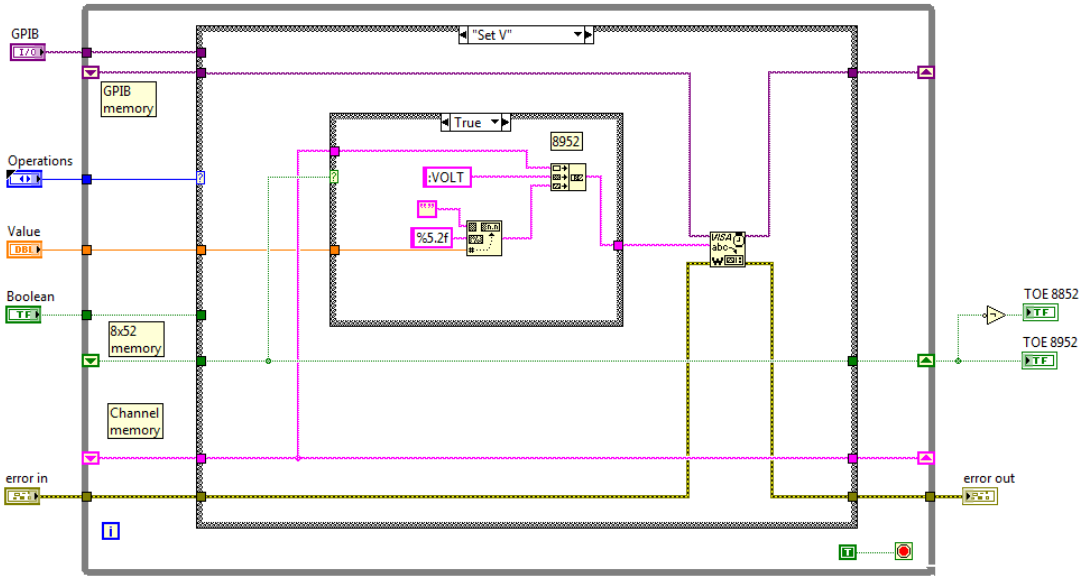


Figure 4.1: Example of FGV structure

Inside each case, SCPI commands are sent and received throuh the VISA (Virtual Instrumentation Software Architecture) interface. Anyway, in some cases, National Instruments gives the possibility to use high level *instrument driver* to communicate.

An *instrument driver* is a set of software routines that control a programmable instrument. Each routine corresponds to a programmatic operation such as config-

uring, reading from, writing to, and triggering the instrument. *Instrument drivers* simplify instrument control and reduce test program development time by eliminating the need to learn the programming protocol for each instrument. National Instruments provides *instrument drivers* for a wide variety of instruments and the architecture use VISA to provide bus and platform independent instrument communication.

Figure 4.2 shows an example of utilization of the k24xx instrument drivers to realize a measure of voltage.

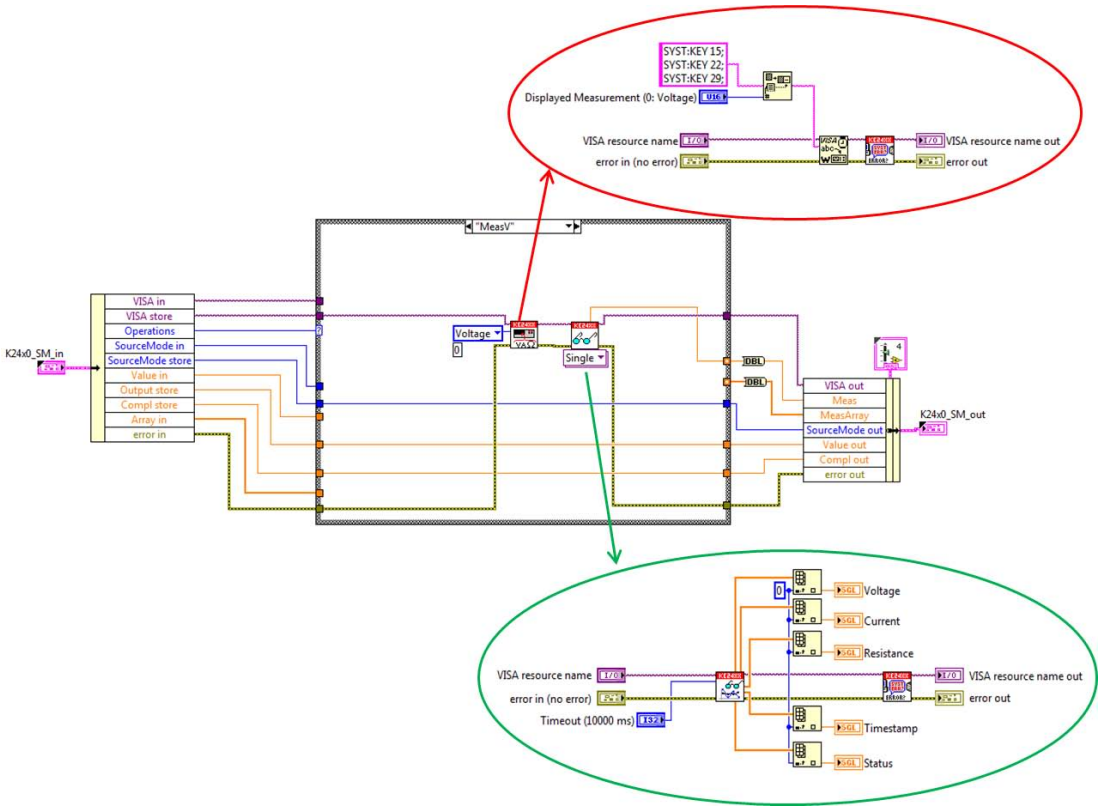


Figure 4.2: Example of using instrument driver

In some cases, the case structure inside the FGV is grouped in a sub-VI called FSM (Final State Machine), which only controls the communication with instruments. In this case, the FGV keeps in memory the values used to communicate, instead FSM, controls the dialog with the instrument. The *Operations* control is wired to the selector terminal of the case structure; each case contains a cascade of VISA blocks with SCPI commands, or, *instrument drivers*.

The VISA is a comprehensive package for configuring, programming, and troubleshooting instrumentation systems comprised of VXI, PXI, GPIB, TCP/IP, USB, and/or serial interfaces. It provides a common foundation for the development, de-

livery, and interoperability of high-level multi-vendor system software components, such as *instrument drivers*, soft front panels, and application software.

Figure 4.3 shows the modular structure FGV → FSM previously described. In this case, all input and output values to FSM are arranged in a cluster structure.

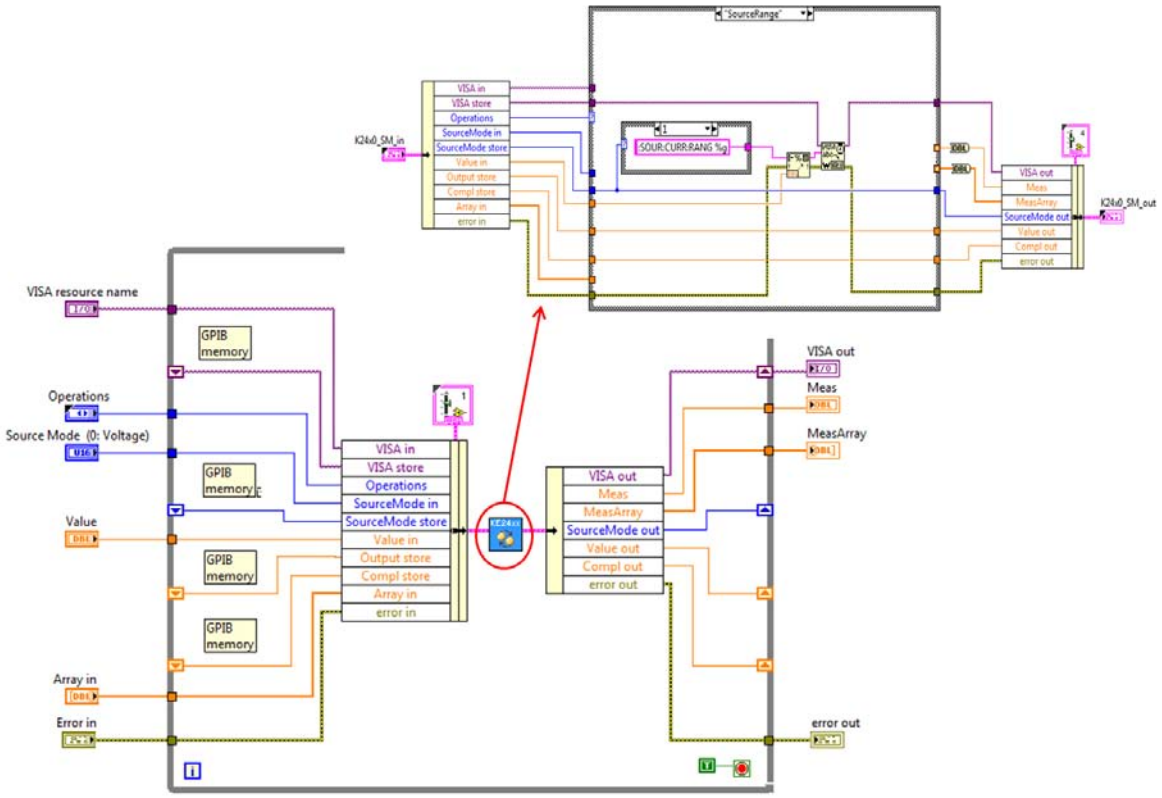


Figure 4.3: Example of modular structure FGV → FSM

In the first part of this chapter we have exposed the structure implemented to realize the GUI for instrument pop-ups. Now we show the *GPIB entry* VI, which allows to select the GPIB address of an instrument when the instrument pop-up is called. Figure 4.4 shows the Front Panel and the Block Diagram of this VI.

The GPIB entry VI, which is used to set the GPIB address of the instrument, will be opened only the first time that the main VI of the instrument pop-up is called. This is possible thanks to the LabVIEW *First Call?* function. The *First Call?* function indicates that a sub-VI or section of a block diagram is running for the first time. This function returns a true boolean value only the first time you call it, after you click the run button. You can place the *First Call?* function in multiple locations within a VI.

Figure 4.5 shows the *CHECK 8852 or 8952 VI*, which allows to identify the

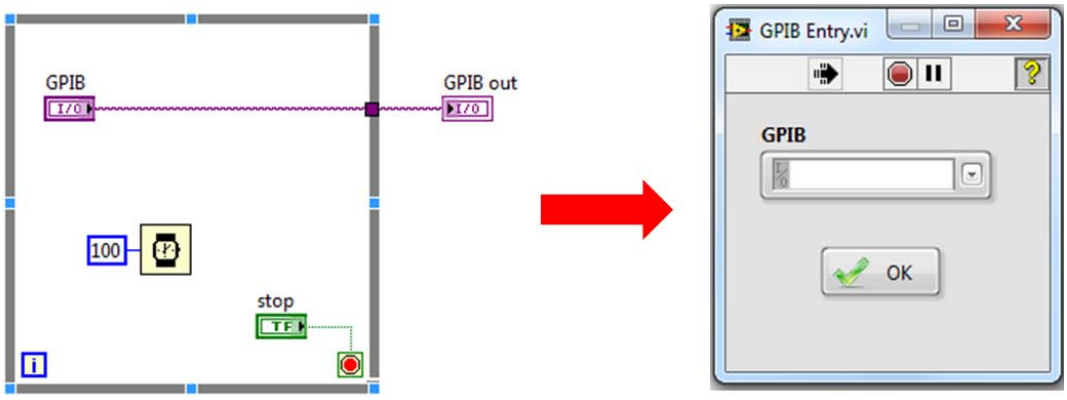


Figure 4.4: GPIB entry: Front Panel and Block Diagram

model of the *Toellner* that we want to use. This approach is also used for Keithley 2430/2440 to identify the correct instrument. For example, the LabVIEW code developed for the *Toellner* is described below:

1. The *Visa write* VI sends an SCPI command (*IDN?) which asks the IDN of the instrument;
2. The *Visa read* VI returns the string requested;
3. *Match Pattern* VI searches a string in the IDN received. It returns -1 if the string is not found, which means that the instrument connected is *TOE 8952*.

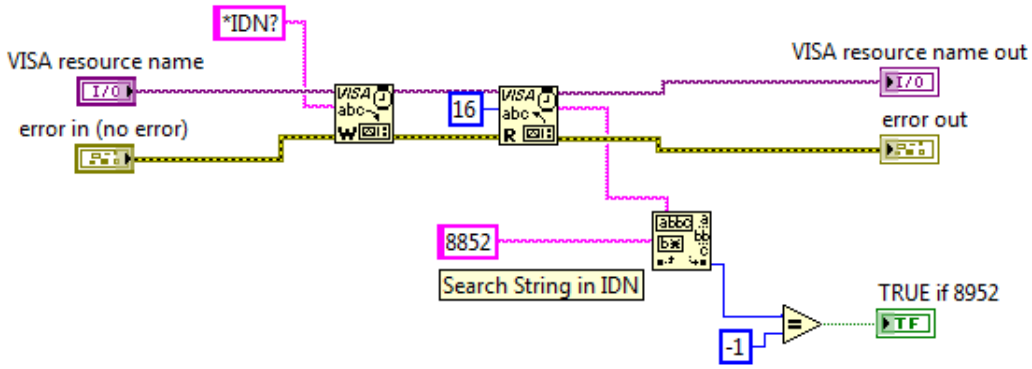


Figure 4.5: LabVIEW code to check Toellner 8852/8952 models

4.1 Toellner 8952/8852 - DC Power Supplies

Figure 4.6: Toellner 8952 DC Power Supply

The principal features of DC Power Supply Toellner 8952/8852 instruments are:

- Automatic setting to the existing line voltage: $115V$ or $230V$, 47 to $63Hz$;
- Autoranging;
- RS 232 and analog interfaces included as standard;
- USB, LAN, and GPIB interfaces optional;
- Outputs at front and rear as standard;
- On/off switching of the output;
- Sensing;
- Free LabVIEW driver;
- Can be used as constant voltage, constant current and constant power source (CV/CC/CP);

The power supplies of the TOE 8950 series deliver a total output power of $400W$, and are available in many different versions. These power supplies operate with a high efficiency of $> 80\%$ at full load, and feature a low residual ripple with voltage and current regulation. Depending on the set values for voltage and current, and also on the load conditions, the power supplies can be used either as a voltage or current source or with a previously set power limit. A further significant feature is the *sense* mode (four-wire connection), that enables direct measurement on the

load of the output voltage which is decisive for voltage control. The voltage on the load is then not influenced by the voltage drops caused by the load current on the load feeders. These voltage drops are compensated in that the voltage at the output sockets is automatically increased by an appropriate amount.

Power supplies with *Autoranging* can output their rated power over a wide and stepless range of voltage and current combinations.

For example, the TOE 8951-40 model, with maximum values of $40V$ and $20A$, permits a current of $20A$ in the voltage range from 0 to $20V$ at its rated power of $400W$. In the voltage range from 20 to $40V$, the available current is $400W$ divided by the actual voltage, e.g.:

- With $30V$ output voltage: $400W : 30V = 13.33A$
- With $40V$ output voltage: $400W : 40V = 10A$

Autoranging power supplies from *Toellner* have a significantly larger operating range than standard power supplies with the same output power.

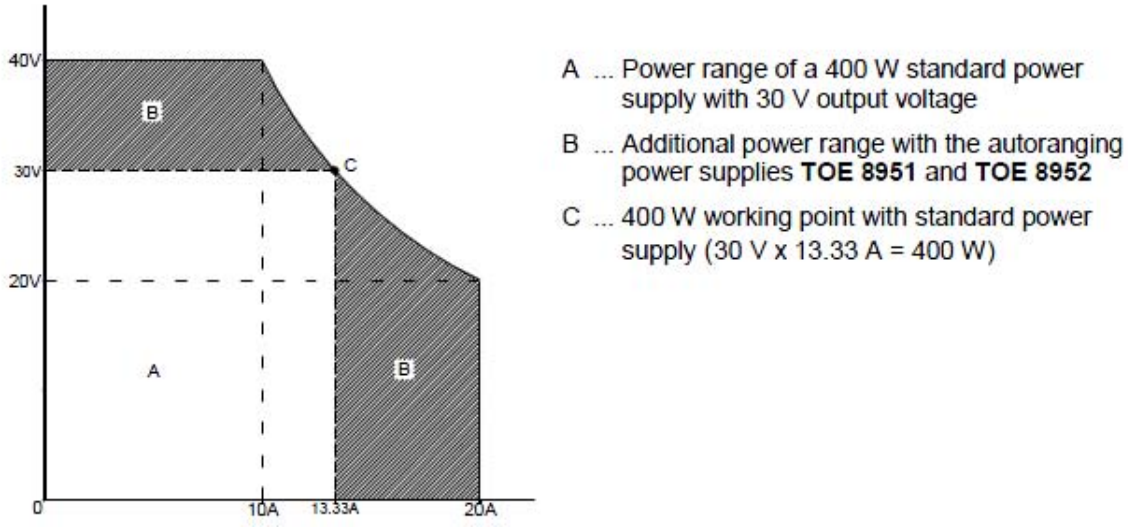


Figure 4.7: Voltage/current diagram with autoranging

Using the standard RS-232 interface, all models can be used as remote-controlled power supplies in computer-based automatic measuring and test systems. GPIB or USB interfaces are available as options. The command syntax is designed in line with the IEEE 488.2 standard. The command set can be switched over between compatible Toellner commands and the SCPI commands standardized for measuring equipment. In remote control mode, the measured values can be read by the

controller at a rate of up to 20 measurements per second, meaning that it is usually unnecessary to use additional measuring instruments.

The Toellner 8850 series is very similar to the Toellner 8950 series presented above but with a total output power of 160W (dual channel) or 320W (single channel).

LabVIEW developmet

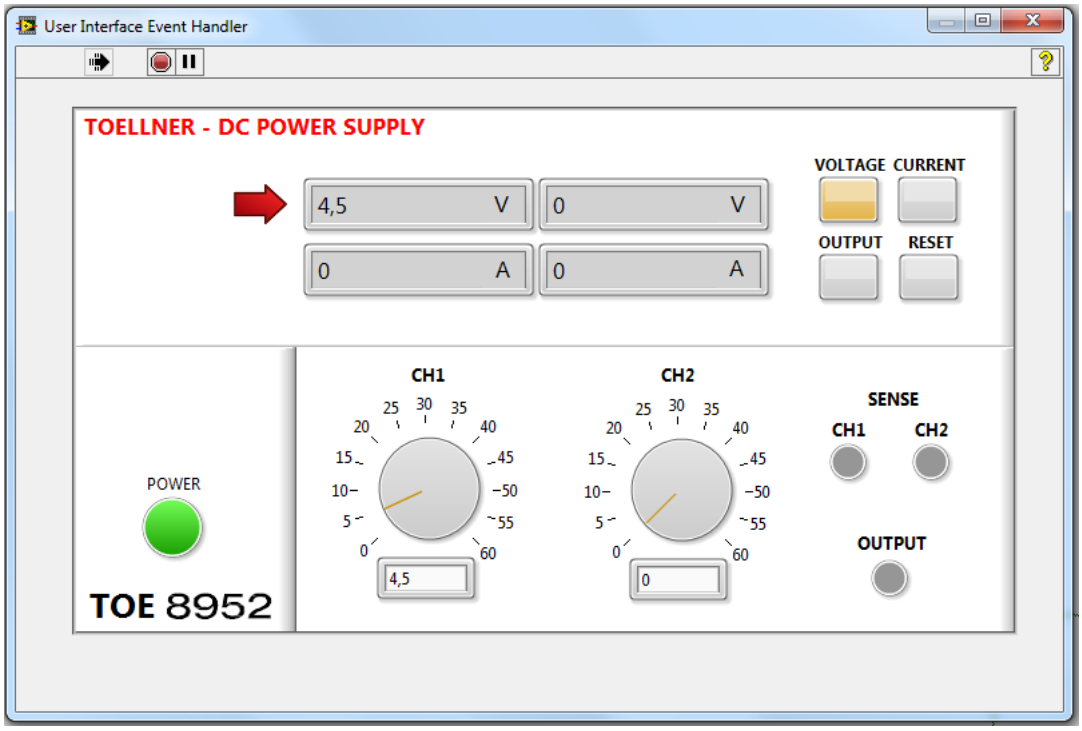


Figure 4.8: Toellner 89528852 LabVIEW interface

Figure 4.8 shows the interface realized for the *Toellner 8952/8852*. When the VI is initialized, a check to understand the model that we are using, *Toellner 8952 or 8852*, will be done.

The interface implemented offers the following functionalities:

- The setting of the voltage of ch1 and ch2;
- The setting of the current limits (compliance) of ch1 and ch2;
- The activation/deactivation of the sensing mode for ch1 and ch2;
- The activation/deactivation of the Output;
- The sending of a *Reset* command to the instrument;

From a LabVIEW point of view, as already discussed, the TOE GUI is realized with an events structure, which calls the sub-VI FGV to send a specific command to the instrument. In this case, we do not have the FSM (Final State Machine), but the case structure is inside the FGV. The communication does not use *instrument driver*, but it is totally done with VISA blocks and SCPI commands. The Figure 4.9(b) shows, by way of example, the internal structure of the FGV.

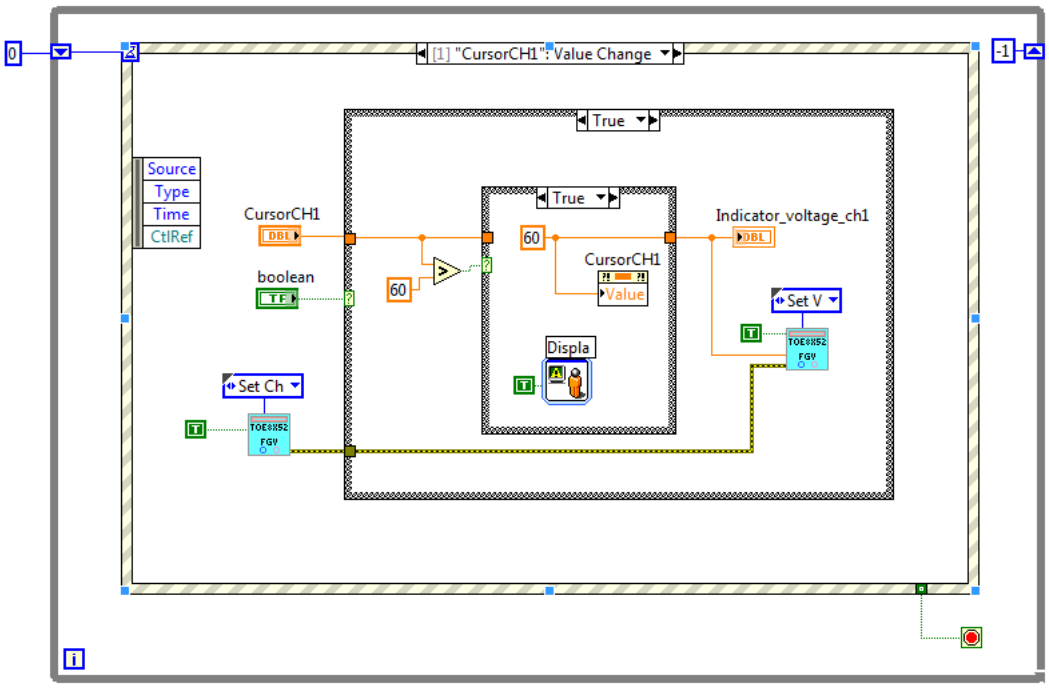
The Figure 4.9(a) shows a portion of LabVIEW code developed to realize the TOE VI. The event structure waits until an event occurs, then executes the appropriate case to handle that event. The code is described here below:

1. The event case will be generated when *CursorCH1* changes in value;
2. The *Boolean* control makes possible to set the voltage or to set the current limit (compliance) of the instrument, *Set V* in this figure;
3. The internal check in the *case structure* verifies if the inserted voltage is not greater than $60V$ (limit of the instrument);
4. The new value inserted will be sent to the instrument using the sub-VI FGV, which receives *Set V* as *Operations* input and *CursorCH1* as *Value* input.

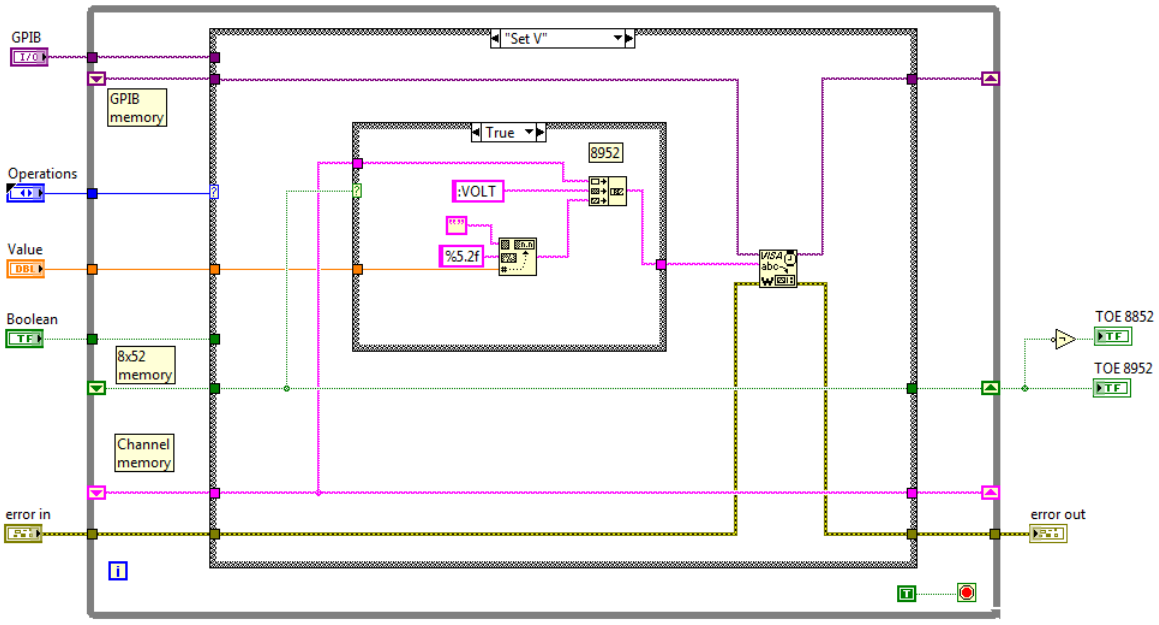
The Operations input of the TOE FGV receives an index that selects a section of the case structure, which contains the functions/commands to be sent to the instrument (Initialize, on/off, Sense on/off, Set Channel, Set V, Set I, Reset, and Close). Each function/command contains blocks and SCPI commands to communicate with the instrument.

Figure 4.9(b) shows the internal structure of the TOE FGV for the case (Set V) selected, which allows to send a new setting of voltage to the instrument. In detail:

1. The string SCPI to be sent for setting the voltage to $5V$ is: VOLT 5
2. The *Format Value* function converts a number (*CursorCH1* value in this case) into a regular string according to the format specified in *format string*;
3. The *format string* used is %5.2f. This format is floating point, where "5" specifies a width of 5, and "2" specifies the number of digits to the right of decimal, or precision;
4. The *Visa Write* function writes the data from *write buffer* to the device or interface specified by VISA resource name.



(a) Portion of LabVIEW code of Toellner for *Set V* function



(b) Toellner FGV with *Set V* as case selected

Figure 4.9: Portion of LabVIEW code for Toellner VI

4.2 Keithley 2430/2440 - Source Meter

Figure 4.10: Keithley 2440 SourceMeter

Keithley's SourceMeter family is designed specifically for test applications that demand sourcing and measurement. All SourceMeter models provide precision voltage and current sourcing as well as measurement capabilities. Each SourceMeter instrument is both a highly stable DC power source and a true instrument-grade multimeter. The power source characteristics include low noise, precision, and readback.

The principal features of this instrument are:

- Five instruments in one (IV Source, IVR Measure);
- Source and sink (4-quadrant) operation;
- 2, 4, and 6-wire remote V-source and measure sensing;
- 1700 readings/second at $4 - 1/2$ digits via GPIB;
- Standard SCPI GPIB, RS-232, and Keithley Trigger Link interfaces;
- Pass/Fail comparator for fast sorting/binning;

Following, the source-measure capabilities of the Keithley 2430 and Keithley 2440 models, which have been used in this project, are listed:

Model 2430

- Source DC or pulse voltage from $5\mu V$ to $105V$; measure voltage from $1\mu V$ to $105.5V$;

- Source DC current from $500pA$ to $3.15A$; measure DC current from $100pA$ to $3.165A$;
- Source pulse current from $500pA$ to $10.5A$; measure pulse current from $100pA$ to $10.55A$;
- Measure resistance from $10uA$ ($< 10uA$ in manual ohms) to $21.1M\Omega$;
- Maximum DC source power is $110W$;
- Maximum pulse source power is $1.1KW$;

Model 2440

- Source voltage from $5\mu V$ to $42V$; measure voltage from $1\mu V$ to $42V$
- Source current from $500pA$ to $5.25A$; measure current from $100pA$ to $5.25A$.
- Measure resistance from $10\mu A$ ($< 10\mu A$ in manual ohms) to $21.1M\Omega$;
- Maximum source power is $55W$;

In operation, these instruments can act as a voltage source, a current source, a voltage meter, a current meter, and an ohmmeter. They also allow measuring with 2-wire or 4-wires. In particular, use 4-wire remote sensing for the following source-measure conditions:

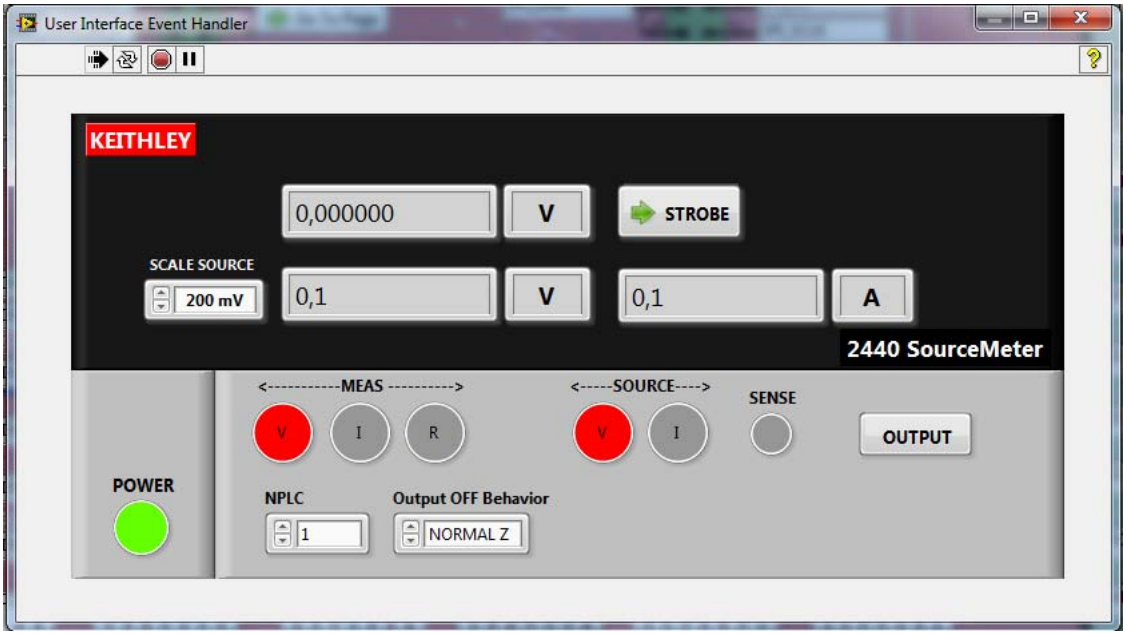
- Test circuit impedance is $< 1K\Omega$
- Optimum ohms, V-source, andor V-measure accuracy is required

All SourceMeter instruments provide four-quadrant operation. In the first and third quadrants, they operate as a source, delivering power to a load. In the second and fourth quadrants, they operate as a sink, dissipating power internally. Voltage, current, and resistance can be measured during source or sink operation.

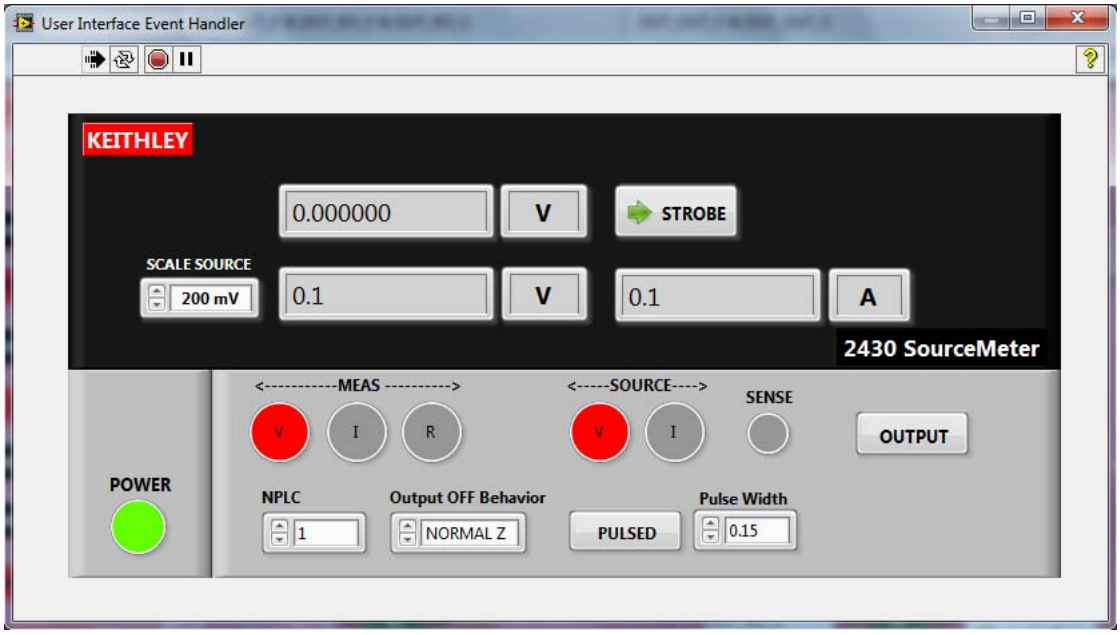
LabVIEW development

Figure 4.11 shows the interfaces realized for the *Keithley 2430/2440*. When the VI is initialized, it will be done a check to understand the model that we are using, Keithley 2430 or 2440.

The interface implemented offers these functionalities:



(a) Keithley 2440 SourceMeter



(b) Keithley 2430 SourceMeter

Figure 4.11: Keithley 2430/2440 LabVIEW interfaces

- The selection of the type of source, V or I, and of the type of measure: V, I, R;
- The setting of the voltage and current limit for V source;
- The setting of the current and voltage limit for I source;
- The setting of the source scale and the measure scale;
- The Activation/deactivation of the *sensing* mode;
- The Activation/deactivation of the Output;
- The setting of *NPLC* and of *Output OFF behavior*;
- the use of the Pulse mode for 2430 model;
- The Acquisition of measure using the STROBE button;

The main VI structure is an events structure, and for each event that occurs, in order to send a specific command to the instrument, it will be called the k24XX FGV sub-VI. In this case, there is another sub-VI inside the FGV, called FSM (Final State Machine), that controls the communications with the instrument. The Figure 4.3, as already shown in the first section of this chapter, illustrates the modular structure FGV $\rightarrow$ FSM for the k24XX.

It is possible to create more instances of the same instrument (e.g. k24XX(1), k24XX(2)) using different copies of the FGV sub-VI (e.g. k24XX FGV(1), k24XX FGV(2)). This is useful because, in some cases, it will be necessary to use more *Keithley 2440 2430* at the same time, during the execution of a measurement in the characterization phase.

The Figure 4.12, for example, shows a portion of the LabVIEW code developed in order to set/change the value of the current source of the instrument. The event structure waits until an event occurs, then executes the appropriate case to handle that event. The code is described as follow:

- The event case will be generated when *Source Control I source* change in value;
- The first case structure extracts the range of current selected and compare it with the new *I source* value;

- If the *I source* value is bigger than the scale selected, it will be automatically searched a new scale for the *I source* value inserted. Otherwise, the scale remains unchanged and the *I source* value is passed directly to the FGV;
- The FGV, with *SetOut* as *Operations* input, is called to send the *I source* value to the instrument.
- The FGV is also called to set the type of source, the current limit (compliance) and the scale selected;

Figure 4.13 shows the internal structure of k24xx FSM, with *SetOut* as case selected in the case structure, and below shows the internal structure of the instrument driver *Set Output*. The *Set Output.vi instrument driver* configures whether the output is in terms of current or voltage, the amplitude of the output, and the compliance settings.

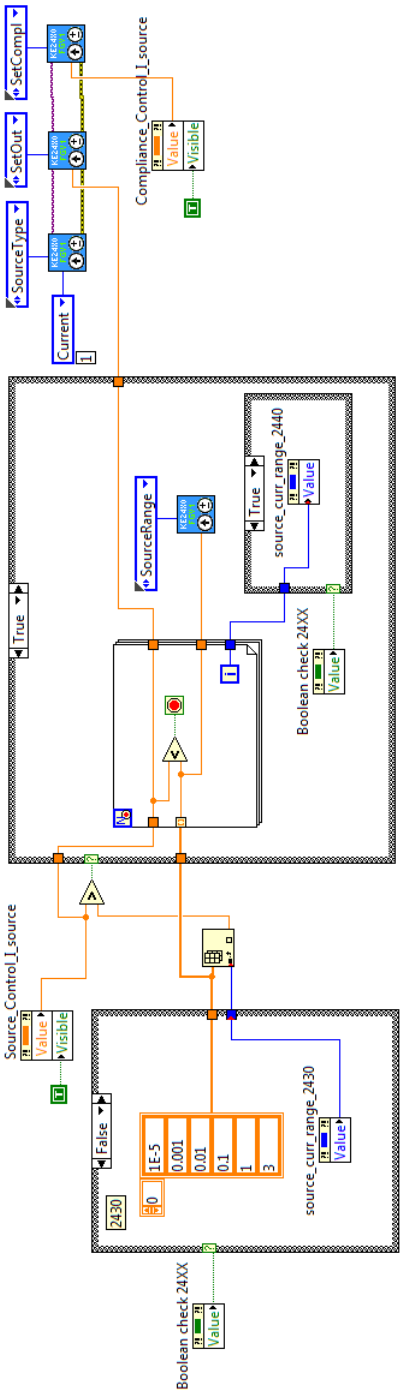
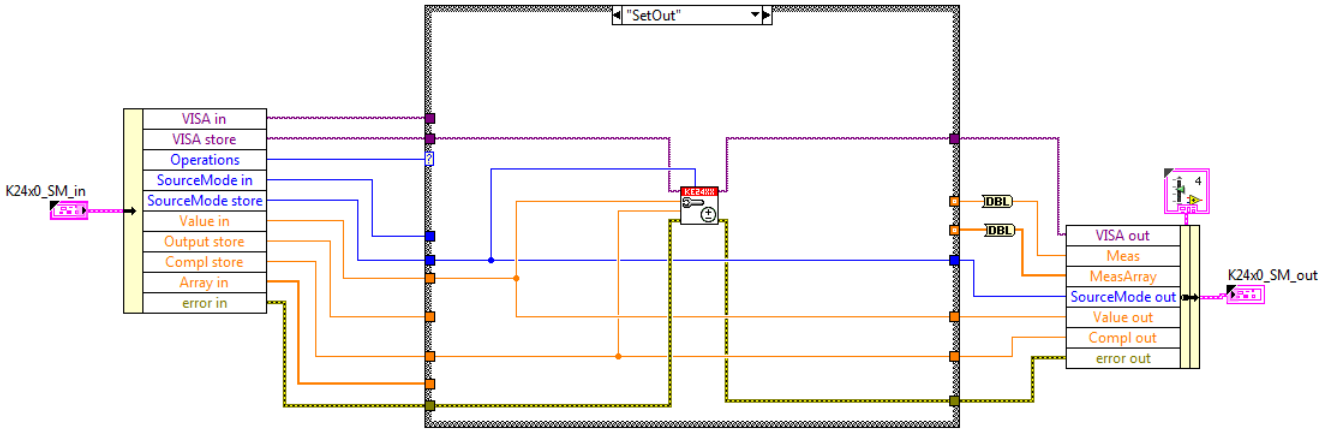
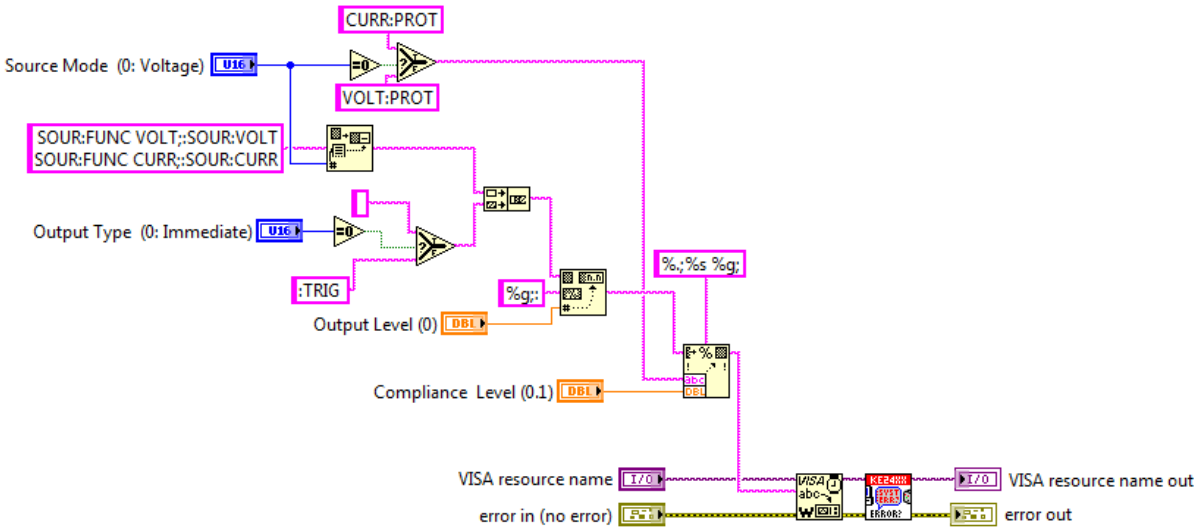


Figure 4.12: Example of LabVIEW code for k24xx interface



(a) FSM of k24XX with *SetOut* as case selected



(b) LabVIEW code for *SetOutput instrument driver*

Figure 4.13: FSM structure of k24XX and *SetOutput* instrument driver

4.3 Keithley 2000 - Multimeter

Figure 4.14: Keithley 2000 Multimeter

Model 2000 (Figure 4.14) is a $6-1/2$ digit digital multimeter, which has broad measurement ranges:

- DC voltage from $0.1\mu V$ to $1000V$;
- AC (RMS) voltage from $0.1\mu V$ to $750V$, $1000V$ peak;
- DC current from $10nA$ to $3A$;
- AC (RMS) current from $1\mu A$ to $3A$;
- Two and four-wire resistance from $100\mu\Omega$ to $120M\Omega$;
- Frequency from $3Hz$ to $500kHz$;
- Thermocouple temperature from -200 to +1372 Celsius;

Some additional capabilities of the Model 2000 include:

- In addition to those listed above, the Model 2000 functions include period, dB, dBm, continuity, diode testing, mX+b, and percent;
- It offers three programming language choices (SCPI, Keithley Models 196199, and Fluke 8840A/8842A) and two remote interface ports (IEEE-488/GPIB and RS-232C);

Another *multimeter* functionality whose presentation is useful for our scopes are:

NPLC: The Rate operation sets the integration time of the A/D converter, the period of time the input signal is measured (also known as aperture). The integration time is specified in parameters based on a *number of power line cycles (NPLC)*, where 1 PLC for $60Hz$ is $16.67ms$ and 1 PLC for $50Hz$ and $400Hz$ is $20ms$.

$$AquisitionTime = NPLC/NetworkFrequency$$

LabVIEW development

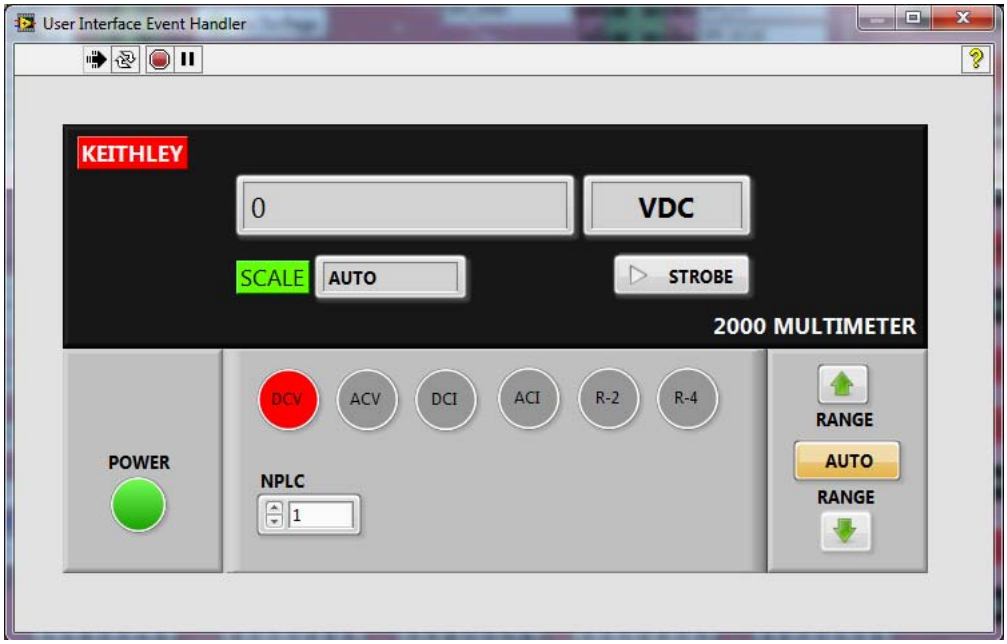


Figure 4.15: K2000 interface

Figure 4.15 illustrates the interface realized for the Keithley 2000.
The interface implemented offers these functionalities:

- Selection of the type of measure: DCV, ACV, DCI, ACI, R-2, R-4;
- Setting of the SCALE of measure;
- Setting of NPLC;
- Acquisition of a measure using STROBE button;

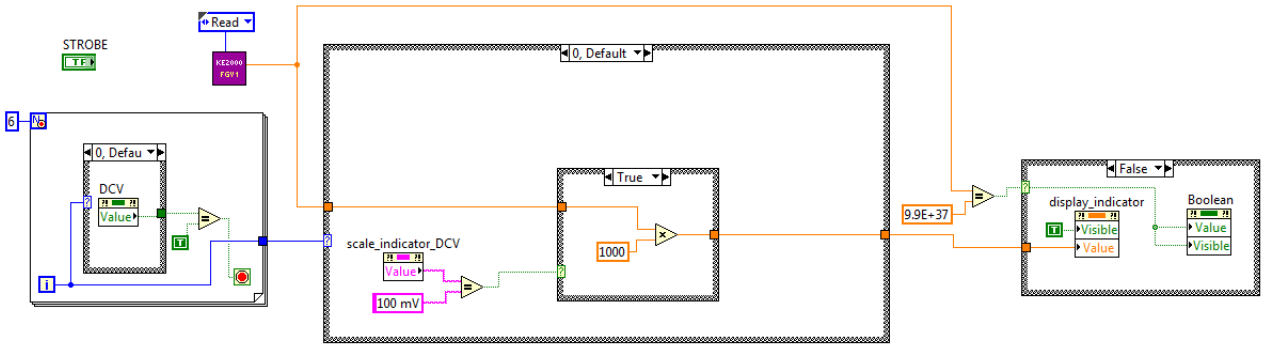
The main VI structure is an events structure, and for each event that occurs, in order to send a specific command to the instrument, the K2000 FGV sub-VI will be called. Also in this case, the FGV VI contains a FSM sub-VI to manage the communication with the instrument.

Figure 4.16(a), for example, shows a portion of the LabVIEW code developed for performing a measurement reading. The event structure waits until an event occurs, then executes the appropriate case to handle that event. The code is described as follow:

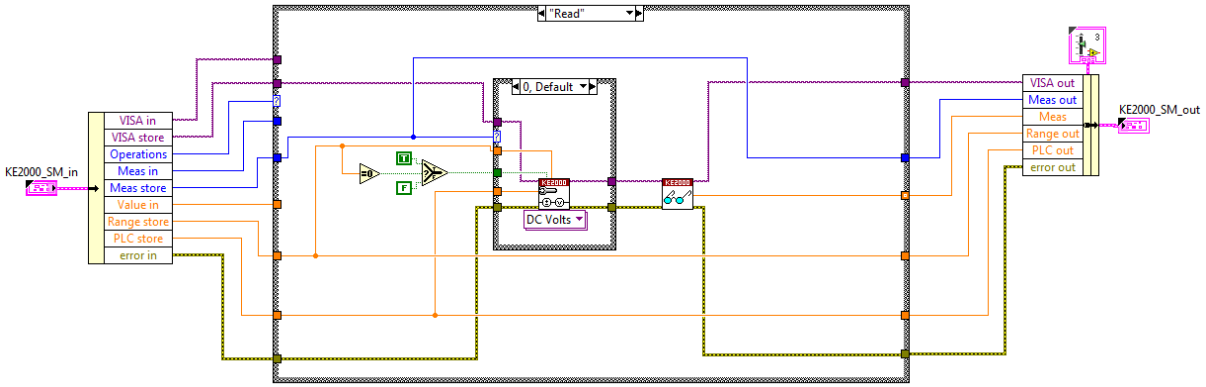
1. The event case will be generated when the STROBE button is pressed;
2. The For Loop searches the type of measure selected (DCV, ACV, DCI, ACI, R-2, R-4);
3. The FGV K2000 with *Read* as *Operations* input asks the instrument to read measure and to return the read value;
4. The value returned is converted in accordance to the scale selected. The example in the Figure 4.16 converts the value from V to mV;
5. If the value returned is equal to $9,9E+37$ it means that there has been an OVERFLOW;

Following, the Figure 4.16(b) shows the internal structure of K2000 FSM for the case (Read) selected, which is used to ask the instrument to read a type of measurement. This section calls two instrument drivers, more specifically:

1. *Configure DC Volts:* The VI configures the instrument to measure DC Volts;
2. *Data Read Single:* This VI reads a measurements or calculation from the instrument. It also returns the channel number and units of the measurements (if applicable);



(a) Portion of LabVIEW code of k2000 for *Read* function



(b) k2000 FSM with *Read* as case selected

Figure 4.16: Portion of LabVIEW code for k2000 VI

4.4 Agilent 33250A - Waveform Generator

Figure 4.17: Agilent 33250a Waveform Generator

The *Agilent Technologies 33250A Function/Arbitrary Waveform Generator* (Figure 4.17) uses direct digital-synthesis techniques to create a stable, accurate output on all waveforms, down to $1\mu Hz$ frequency resolution.

The *function generator* can output five standard waveforms including sine, square, ramp, pulse, and noise. You can also select one of five built-in arbitrary waveforms or create your own custom waveforms. You can internally modulate any of the standard waveforms (except pulse and noise) and also arbitrary waveforms using AM, FM, or FSK. The output frequency range depends on the function currently selected. The *default* frequency is 1 kHz for all functions.

Others function generator functionalities whose presentation is useful for our scopes are:

Burst: You can generate a burst waveform using any of the standard waveforms and also arbitrary waveforms. The Burst mode is used to output a waveform with a specified number of cycles, called a burst.

Trigger: Applies to sweep and burst only. You can issue triggers for sweeps or bursts using internal triggering, external triggering, or manual triggering. Internal or "automatic" triggering is enabled when you turn on the function generator. In this mode, the function generator outputs continuously when the sweep or burst mode is selected.

Remote control: For system applications, both GPIB and RS-232 interfaces are

standard, and support full programmability using SCPI commands.

In the Front Panel, the knob or numeric keypad can be used to adjust frequency, amplitude and offset. You can even enter voltage values directly in Vpp, Vrms, dBm, or high/low levels. Timing parameters can be entered in hertz (Hz) or seconds.

LabVIEW development

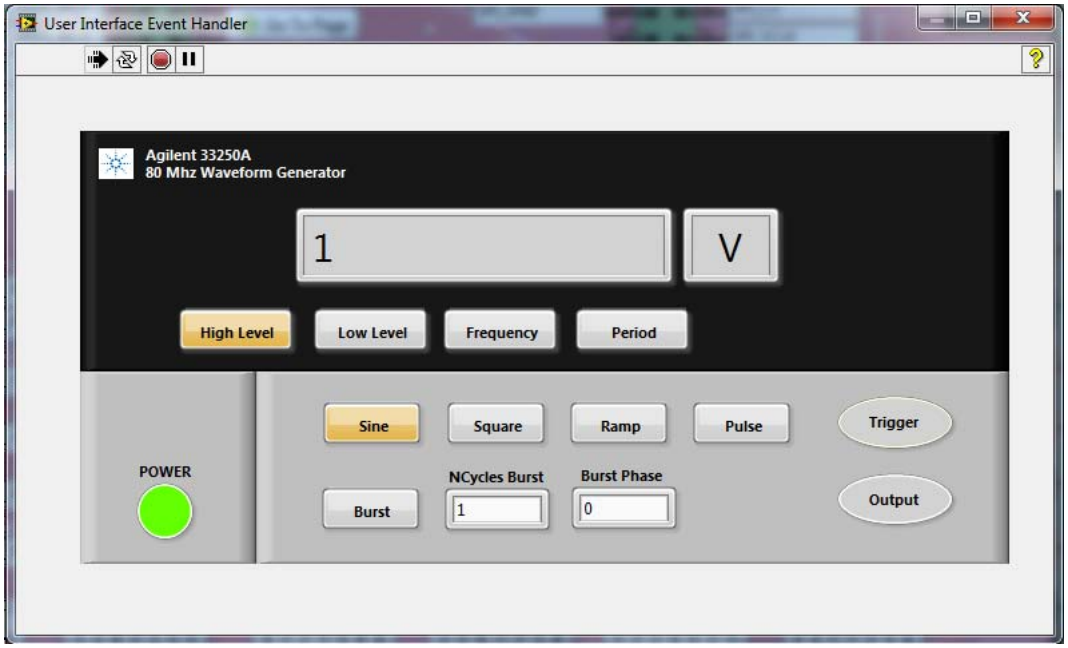


Figure 4.18: Agilent 33250a LabVIEW interface

Figure 4.18 illustrates the interface realized for the Waveform Generator Agilent 33250A.

This interface offers these functionalities:

1. The possibility to select one of these types of waveforms: Sine, Square, Ramp and Pulse;

2. For the Sine waveform the possibility to set: high level, low level, frequency and period;

3. For the Square waveform the possibility to set: high level, low level, frequency, period and duty cycle;

4. For the Ramp waveform the possibility to set: high level, low level, frequency and period;

5. For the Pulse waveform the possibility to set: high level, low level, frequency, period, pulse width and edge time;

6. The selection of the *Burst* mode and set *NCycles Burst* and *Burst Phase*;

7. The Activation/deactivation of *Trigger* pulse;

8. The Activation/deactivation of *Output*;

The main VI structure is an events structure, and to send a specific command to the instrument, the corresponding Agilent FGV sub-VI will be called. The FGV contains also the FSM sub-VI which manages the communication with the instrument.

The Figure 4.19(a), for example, shows a portion of LabVIEW code extracted by the main VI developed for the Agilent interface. The code allows to set the parameters of a Sine waveform. In details:

1. An event case will be called when the Sine button is pressed or when one of the high level sine, low level sine, set frequency sine controls changes in value;

2. The Agilent FGV, which manages the communication with the instrument, receives the new value as Value input, and the type of the command/function to be sent or of the parameter to be set, as Operations input;

Figure 4.19(b) shows the FSM sub-VI with Set Frequency as case selected. The LabVIEW code uses Visa write to send SCPI commands, through GPIB, to set the frequency, and uses Format Into String function to convert the input value (Set Frequency Sine in this case) into string.

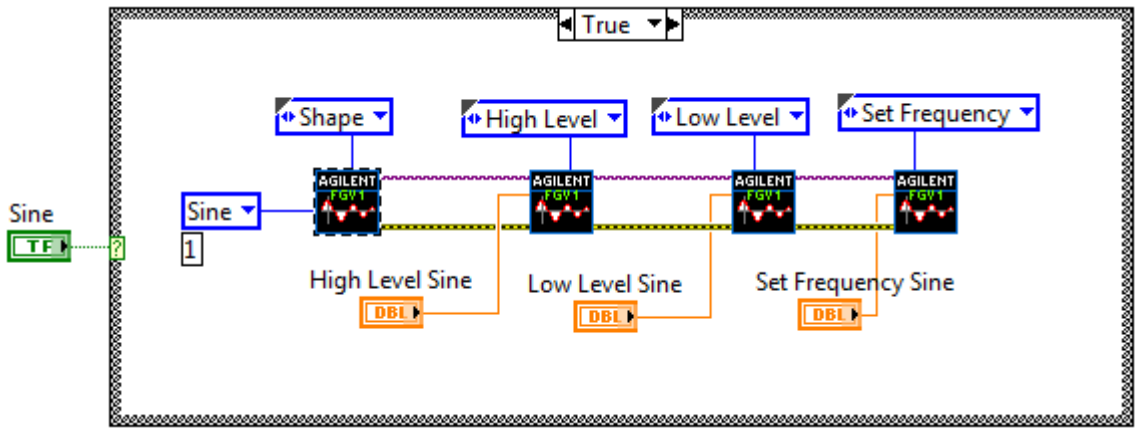
By way of example, the following command strings sent from your computer will output a 3 Vpp sine wave at 5 KHz with a -2,5 volt offset:

1. FUNC SIN Select sine wave function;

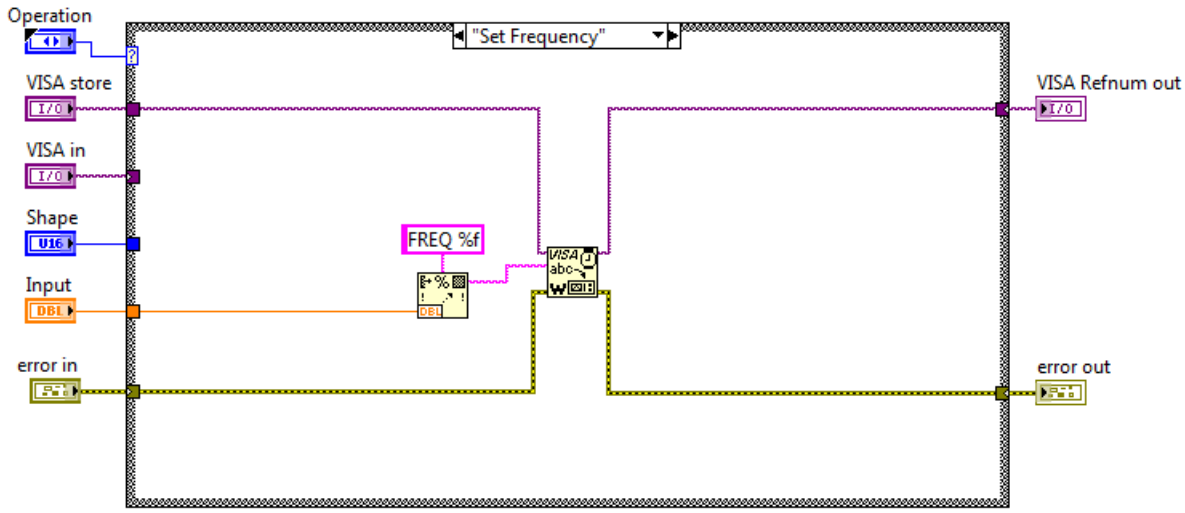
2. FREQ 5000 Select Frequency to 5 KHz;

3. VOLT 3.0 Select Amplitude to 3 Vpp;

4. VOLT:OFFS -2.5 Set offset to -2.5 Vdc;



(a) Portion of LabVIEW code for Agilent interface



(b) FSM of Agilent with SetFreq as case selected

Figure 4.19: Example of LabVIEW code for Agilent interface

CHAPTER

5

CONCLUSIONS

The Electrical-lab-characterization is a crucial phase in Semiconductor product development. Thus it is fundamental to find innovative solutions in the characterization phase of devices and in this sense, the automation of measurements brings numerous advantages and represents the main strategy to address this issue.

The Product Engineers team of Infineon Padua, to automate the characterization of voltage regulators devices, designed a dedicated hardware with the aim of achieving the goal of obtaining an automated measurement bench. To integrate this part of the project, the work I have done has been to develop a graphical user interface which allows to communicate with the hardware, allowing to manage the measurement setup and to control the instruments connected to the GPB by dedicated pop-up interfaces .

- Reduction in characterization cycle time: reduces the time taken for the execution of the measures, allowing you to test a larger number of samples;
- Increase Evaluation Quality: More samples tested involve statistics much more accurate;
- Greater Reliability: the likelihood of errors in the measurements caused by "human errors" is eliminated;

- Remote control of measures setup: possibility to manage the measurement setup from the terminal by exploiting the capabilities of the GUI;
- Full support to the measuring bench: the graphical interface accurately reproduces the physical GPB, allowing you to have a clear visual support that represent the test bench in an orderly manner (thanks to indications on associated instruments, on relays status, on links instruments → DUT etc.)
- Real-time debugging: The possibility to communicate in real-time with instruments, with the opportunity to use dedicated graphical interfaces, allows you to have full support for debugging;
- Reusability: Hardware and software developed can be used for the whole family of Voltage Regulators, only changing the hardware of the daughterboard;

The graphical interface I have developed, will be integrated as a complement/-support for a GUI realized by the Product Engineers team of Infineon Padua. The automated test bench, described in this work, is already in use in the laboratories of Infineon Padua for the characterization of the device TLF4277 (Low Drop Out Linear Voltage Regulator) which is the ideal companion IC to supply active antennas for car infotainment applications.

Starting from a platform that is usable for our purposes, possible future developments to obtain a workbench optimized are listed below:

- Optimize the functionality of strobing of the measures. The GUI developed integrates the ability to communicate with instruments and to capture measurements in real-time. The future goal is to optimize this functionality by implementing the possibility to acquire/display waveforms in real-time;
- Realize a dedicated hardware in order to perform an hardware check on the relays of the board;
- Starting from the one already implemented, develop a software and hardware platform, complete and optimized, to be exploitable with the whole Voltage Regulator Family. For this purpose, the goal is to search for possible improvements in hardware (GPB + daughterboard), and implement a complete software architecture (Measures setup, Sequencer measures, real-time debugging);

APPENDIX

A

GRAPHICAL USER INTERFACE FIGURES

This Appendix contains a series of images of the various sections of the GUI. It is a useful support for chapter 3 (Graphical User Interface for measure setup). Here the various interface features are described in detail.

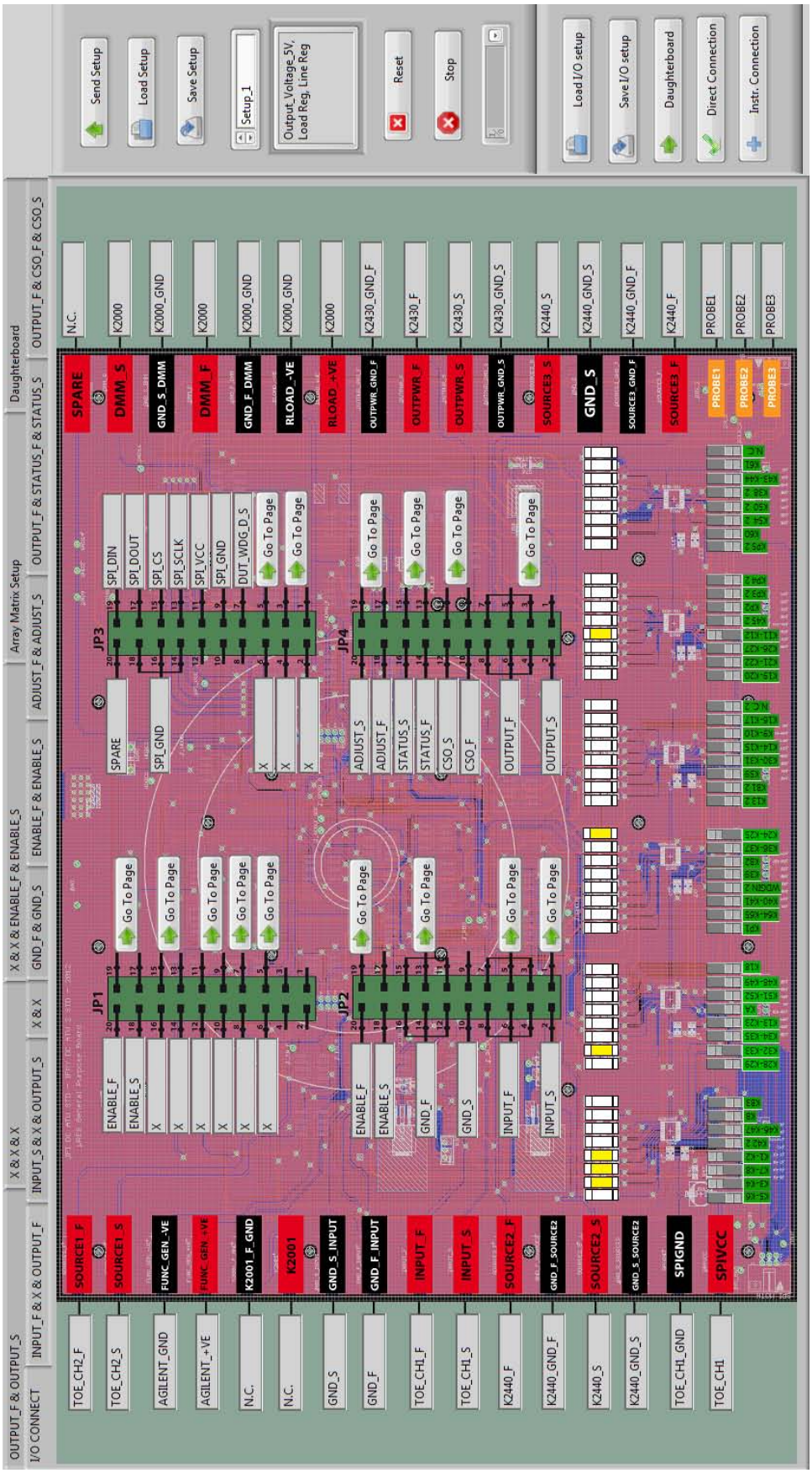


Figure A.1: I/O Connections section of the GUI

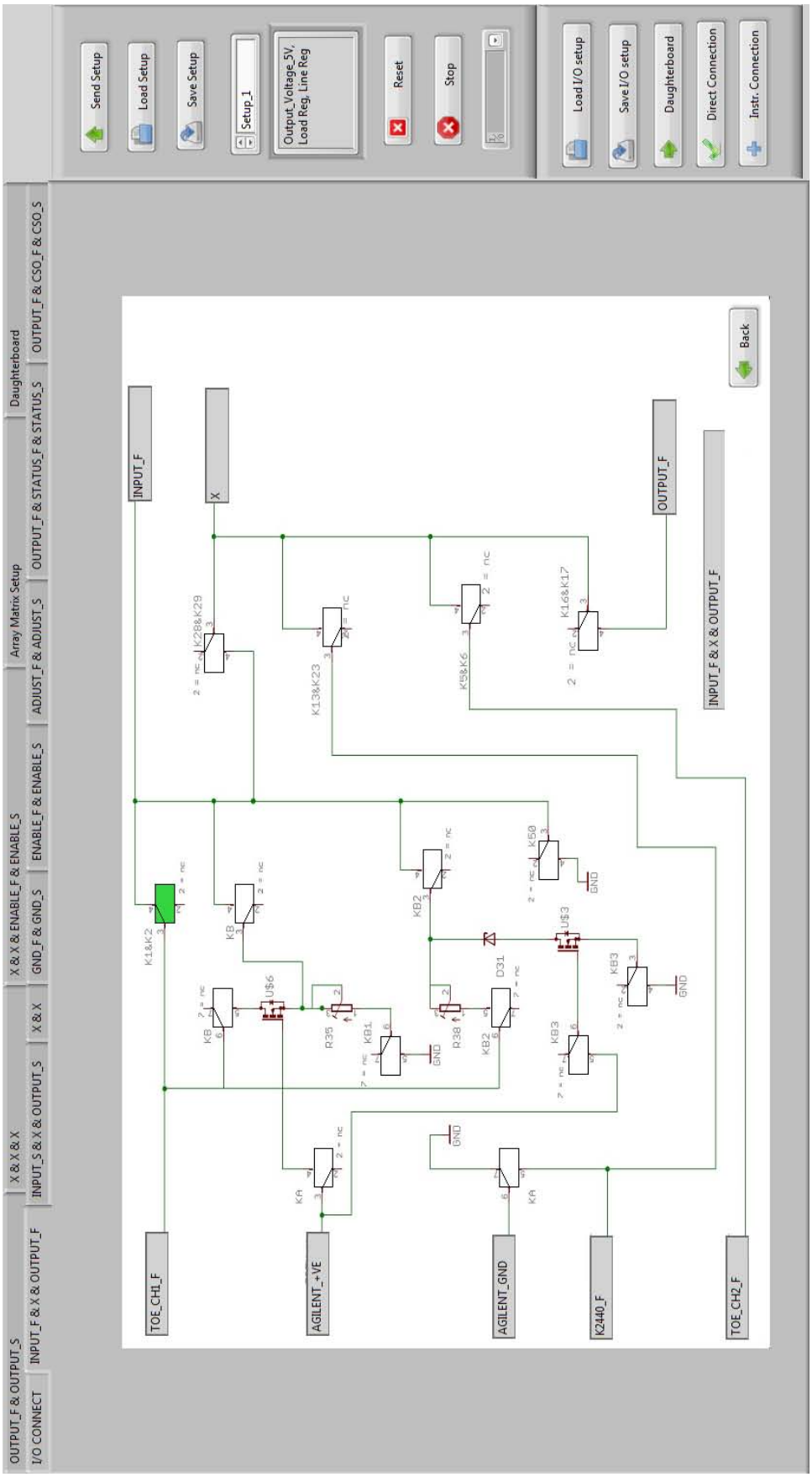


Figure A.2: PCB section 1

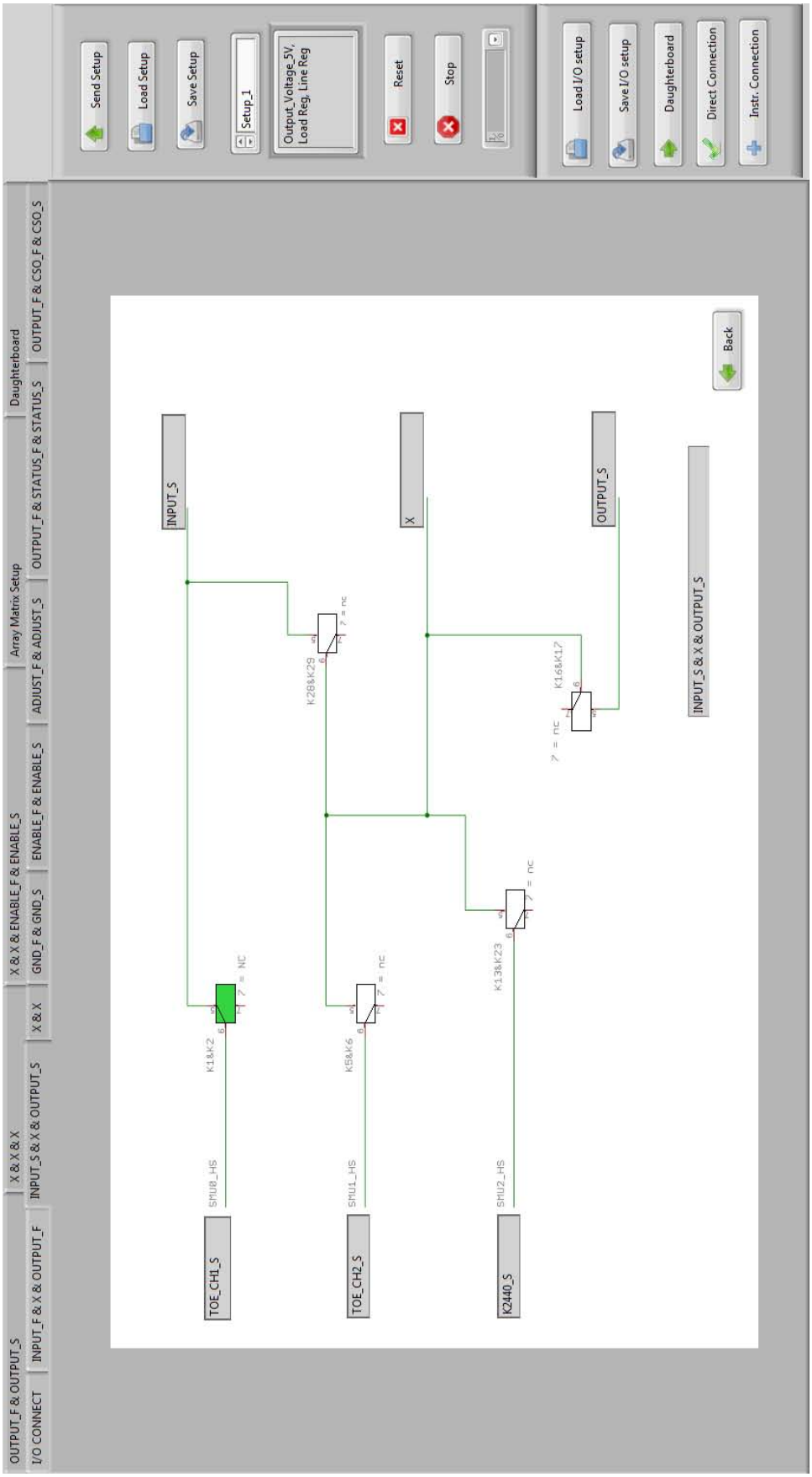


Figure A.3: PCB section 2

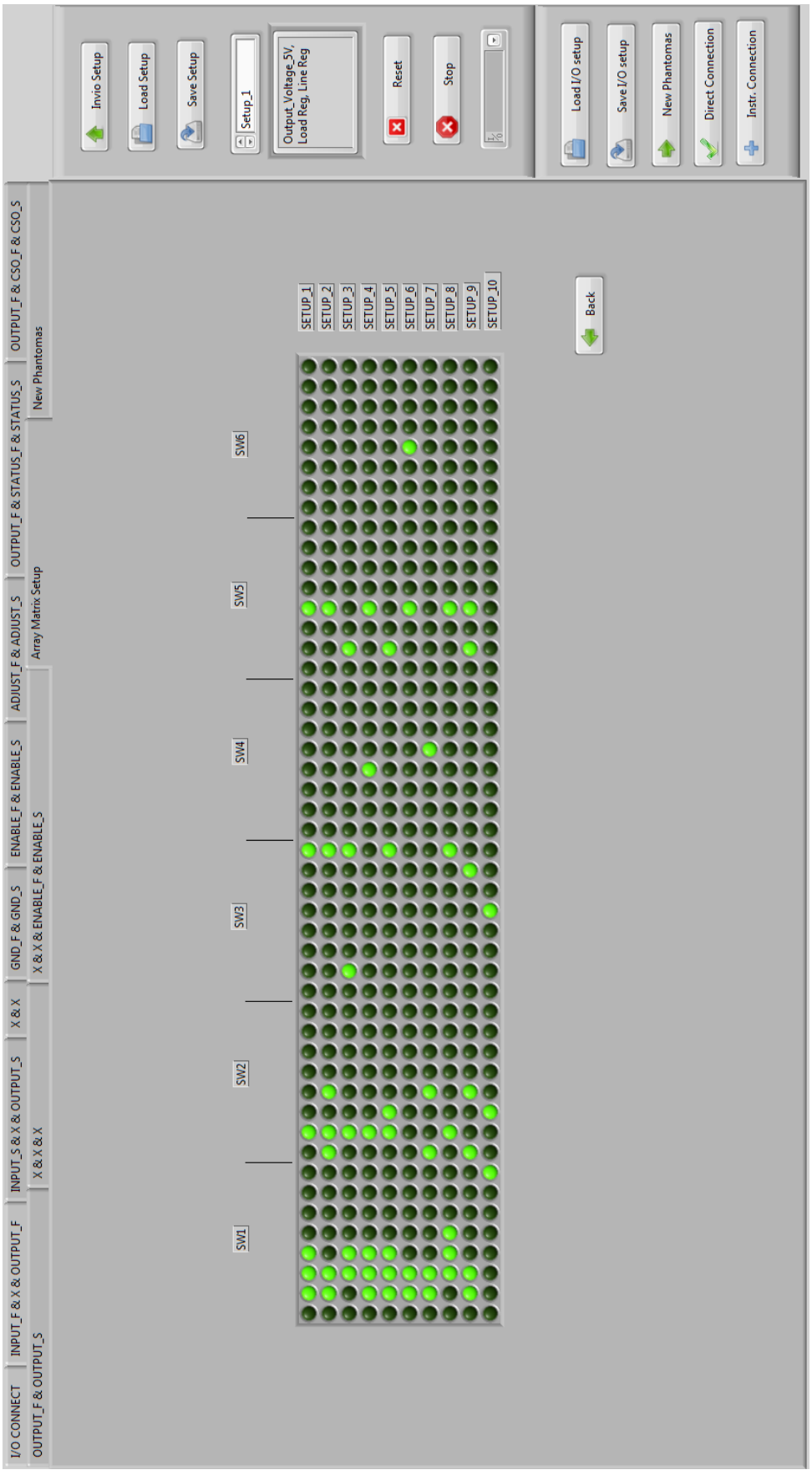


Figure A.4: Setups Matrix section

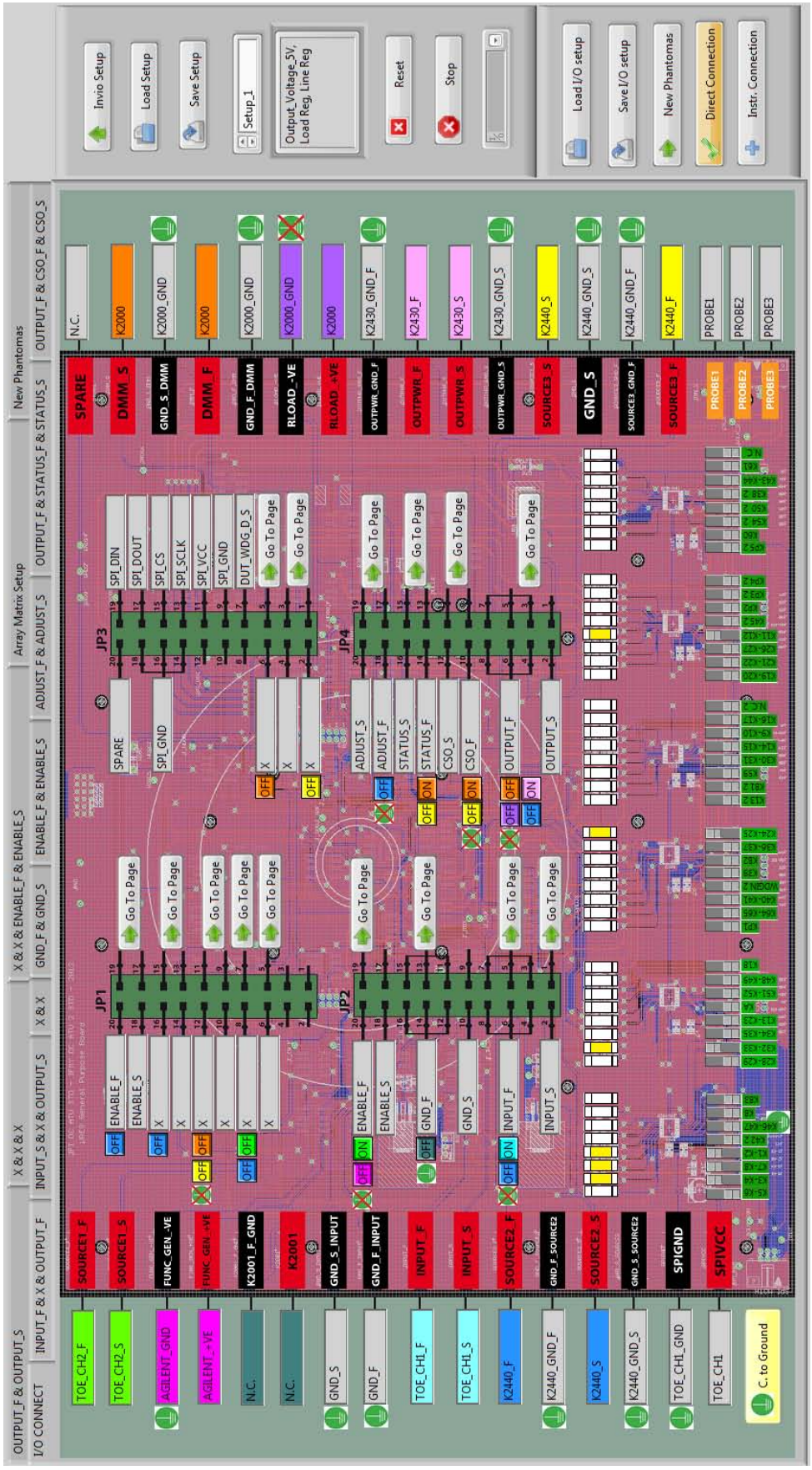


Figure A.5: Direct Connection functionality

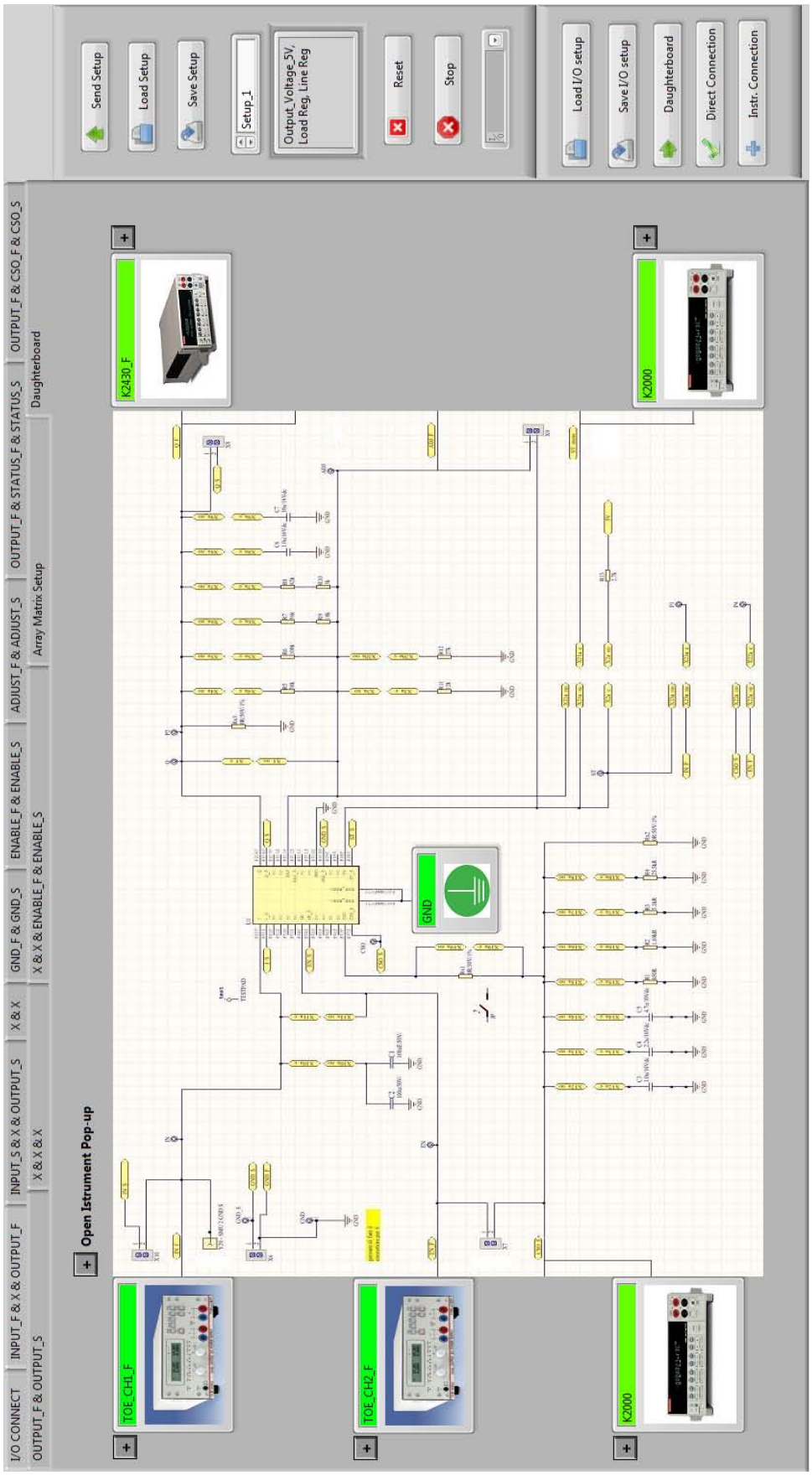


Figure A.6: Daughterboard section

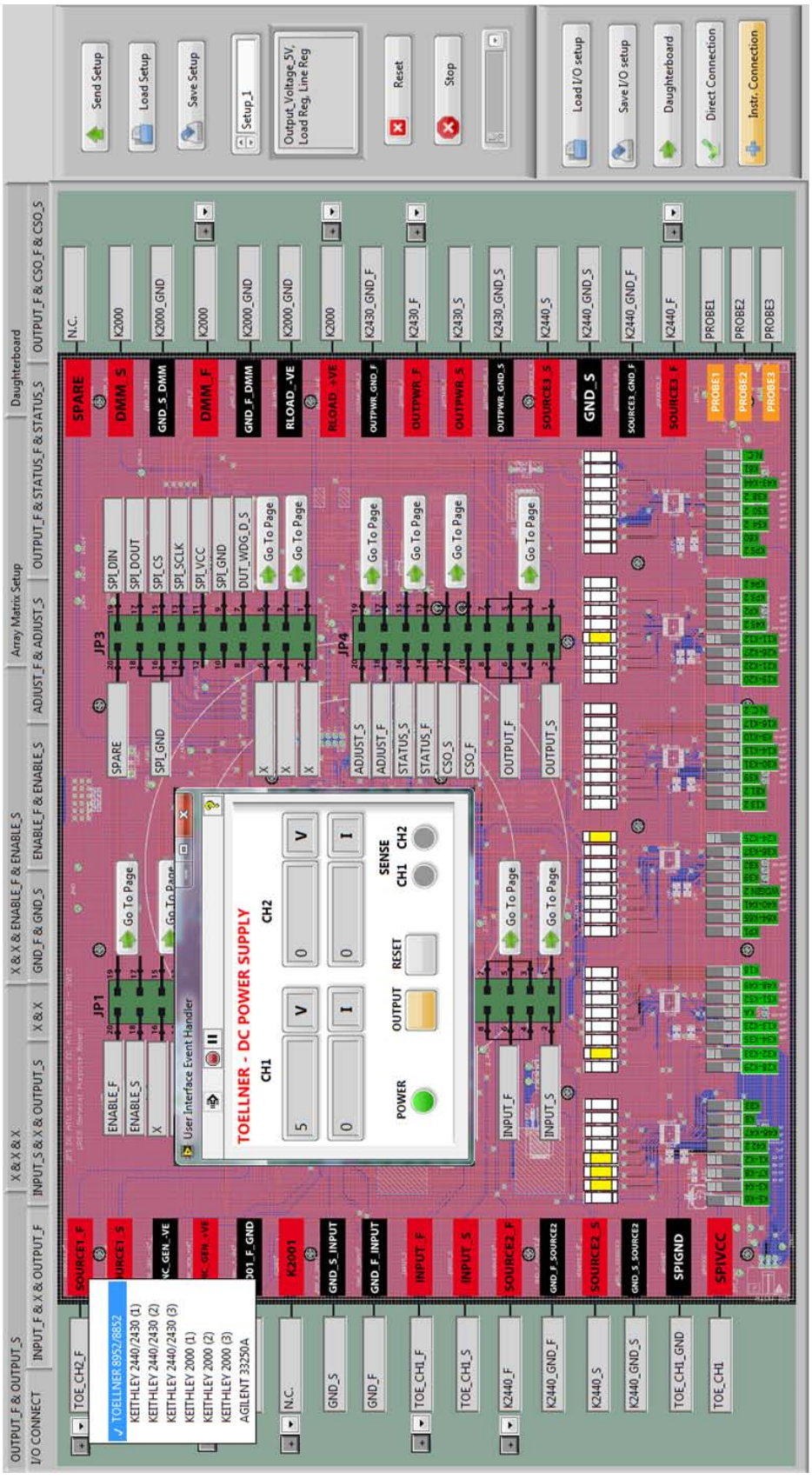


Figure A.7: Instr. Connection functionality with instr. pop-up opened

BIBLIOGRAPHY

[1] S. Rajamanickam, M. Piccinelli, M. Giarin and N. Jamil: Time-Effective Lab Characterization on Voltage Regulators Without Compromising the Evaluation Quality, IEEE, June 2013;

[2] Model 2000 Multimeter, User's Manual, Rev. J / August 2010;

[3] Toellner 8950E Instruction Manual - January 2008;

[4] 2400 Series SourceMeter Quick Results Guide, May 2002;

[5] Agilent 33250A 80 MHz Waveform Generator, User's Guide, Edition 2, March 2003;

[6] http://arduino.cc/en/Tutorial/HomePage

[7] M. Piccinelli - /Projects_Padova/PRE_PRS/.../GPB_roadmap_v2;

[8] M. Piccinelli - /Projects_Padova_PRE_PRS/temp/.../Doc/LDO_GPB_Mario_v2.sch;

[9] TLF4277 LDO Linear Voltage Regulator Data Sheet Rev. 1.0, 2014-03-13;

[10] D. Dario, M. Lenz: DOPL-VREG-Introduction - June 2007;

[11] ni.com/training - LabVIEW core 1, core 2, core 3;

[12] NI-488.2 User Manual - February 2005;