BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

VHDL-Based System Design of a Cognitive Sensorimotor Loop (CSL) for Haptic Human-Machine Interaction (HMI)

Bachelor Thesis

Pablo de Miguel Morales

Matrikelnummer: 821784

SS 2015

Betreuer BHT: *Prof*. *Dr*. Hild

Gutachter: *Prof.* Kersten

Beginn Datum: 03.08.2015

Ende Datum: 03.11.2015

Abgabe Datum: 10.08.2015

Contents

1. Introduction

In the growing field of automation, a large amount of workforce is dedicated to improve, adapt and design motor controllers for a wide variety of applications. In the specific field of robotics or other machinery designed to interact with humans or their environment, new needs and technological solutions are often being discovered due to the existing, relatively unexplored and new scenario it is.

The haptic interaction between an automated machine and its surroundings presents a large number of challenges as several objectives have to be achieved simultaneously. For example, a robot hand has to be able not only to detect and identify an object through touching, but also to be able to grab it with the appropriate force. The complexity and variety of the haptic interaction for an automated system requires semi-autonomous solutions which can handle these tasks with only a minimal surveillance of a centralized control.

To approach this problem, a *Cognitive Sensorimotor Loop* (CSL) based (Developed by the *Neurobotics Research Laboratory*)[1] system is to be developed since this guarantees the implementation of several autonomous motor systems acting simultaneously without a centralized control structure. Moreover, the lack of sensors drastically reduces the maintenance and calibration requirements of a complex system with multiple motor controls. This CSL system is to be implemented in VHDL to allow an easy escalation and the possibility to implement it in any FPGA/CPLD system based on the original CSL *Verilog* code created by *Prof. Dr.* Hild[2].

The principal aim of this document is to describe in detail all the systems developed and their analysis. The 2 developed systems (*CSL Stay in Touch* and *CSL Fingerlike Mechanism* systems) are totally described and analyzed independently. Additionally, all the graphic modules used to monitor the behavior of the system are also described. A haptic perception experiment is designed and its results described. Finally, a conclusion is written.

This document is written by Pablo de Miguel Morales, *Electronics Engineering* student at the *Universidad Politécnica de Madrid* (UPM Madrid, Spain) during an *Erasmus+* Exchange Program at the *Beuth Hochschule* für *Technik* (BHT Berlin, Germany). The tutor of this project is *Dr. Prof.* Hild. This project has been developed inside the *Neurobotics Research Laboratory* (NRL) in close collaboration with Benjamin Panreck, a member of the NRL, and another exchange student from the UPM Pablo Gabriel Lezcano.

1.1 Motivation

Since the beginning of my engineering studies, I have become increasingly interested in the field of robotics and automation. It is in my opinion, apart from biomedical engineering and spacecraft technology, the only noble path of an engineer in the industrial society. In this sense, I have devoted time and effort outside my university studies to develop small projects adequate for my knowledge which aim was to learn the use of a variety of technologies regarding my future objectives.

Another thing I find extremely interesting about robotics is the vast variety of disciplines needed to develop good products. Unlike other dark fields, a robotic project needs the approach of many different professionals, both technical (engineers, mathematicians, etc.) and non-technical (designers, psychologists, artists, etc.).

---

[1] *Neurobotics Research Laboratory Main Page*(2015), http://www.neurorobotik.de/index_en.php

[2] *Defying Gravity – A Minimal Cognitive Sensorimotor Lopp Which Makes Robots With Arbitrary Morphologies Stand Up* (2013), Dr. Manfred Hild, DEMI 2013 Paper, http://www.neurorobotik.de/downloads/publications/2013%20Hild%20-%20Defying%20Gravity.pdf

For this reason I chose to finish my Bachelor degree in the *Neurorobotics Research Laboratory* (NRL) as it was a unique opportunity considering these kind of research laboratories are not present in many universities.

This project suited my expectations in many ways as it allowed me to have a lot of freedom to develop my system through very few guidelines and an objective based success. It also made me feel part of a more complex system in which team mechanisms and regular meetings played an important role, showing me how a real research laboratory works and how to deal with teammates and shared timelines.

Even though I do not enjoy VHDL design very much (Compared to microcontroller programming code), this project added other elements as the use of real hardware and a physical working system. Finally, the possibility of basing future systems in my work makes me consider myself to have developed something valuable and not just another thesis to be forgotten in the university warehouses.

"When a society is rich, its people don't need to work with their hands; they can devote themselves to activities of the spirit. We have more and more universities and more and more students. If students are going to earn degrees, they've got to come up with dissertation topics. And since dissertations can be written about everything under the sun, the number of topics is infinite. Sheets of paper covered with words pile up in archives sadder than cemeteries, because no one ever visits them, not even on All Souls' Day. Culture is perishing in overproduction, in an avalanche of words, in the madness of quantity."[3]

2. Objectives

The original objectives of this project were:

1. To understand the *Cognitive Sensorimotor Loop* (CSL)
2. Development of graphic interfaces based on inherited modules
3. Development of a basic one joint Stay in Touch System
4. Development of a complex one joint Stay in Touch System
5. Development of a two joint Fingerlike Mechanism System [Optional]

3. Attached DVD Files

Attached to this document, a DVD is provided containing crucial files that will be mentioned throughout the work and are vital for the understanding of the systems. The files inside this DVD are organized as following:

- CSL_FingerlikeMechanism_System (Contains all the functional files of the system)
  - FM_Schematics (Contains the Schematics of the system)
  - FM_Modules (Contains the VHDL files and constraint file needed to build uo the system)
- CSL_FingerlikeMechanism_Videos(Contains the videos of the system performance)
- CSL_StayInTouch_System (Contains all the functional files of the system)
  - SIT_Schematics (Contains the Schematics of the system)
  - SIT_Modules (Contains the VHDL files and constraint file needed to build up the system)
  - SIT_SystemAnalysis (Contains the DRC reports of the system)
- CSL_StayInTouch_Videos (Contains the videos of the system performance)
- HapticExperiment_Videos (Contains two examples of the experiment for both Groups)
- Bachelorthesis (This same document is also included in the DVD)

[3] *The unbereable lightness of being*, Milan Kundera (1984)

4. Stay in Touch System (SIT)

4.1. System Hardware Overview

The appearance of the system and the names that are shown in this overview are the ones to be used all through the document.

4.1.1. Hardware Used

This section contains a description of the hardware used and the connections. It also labels the specific control interface.

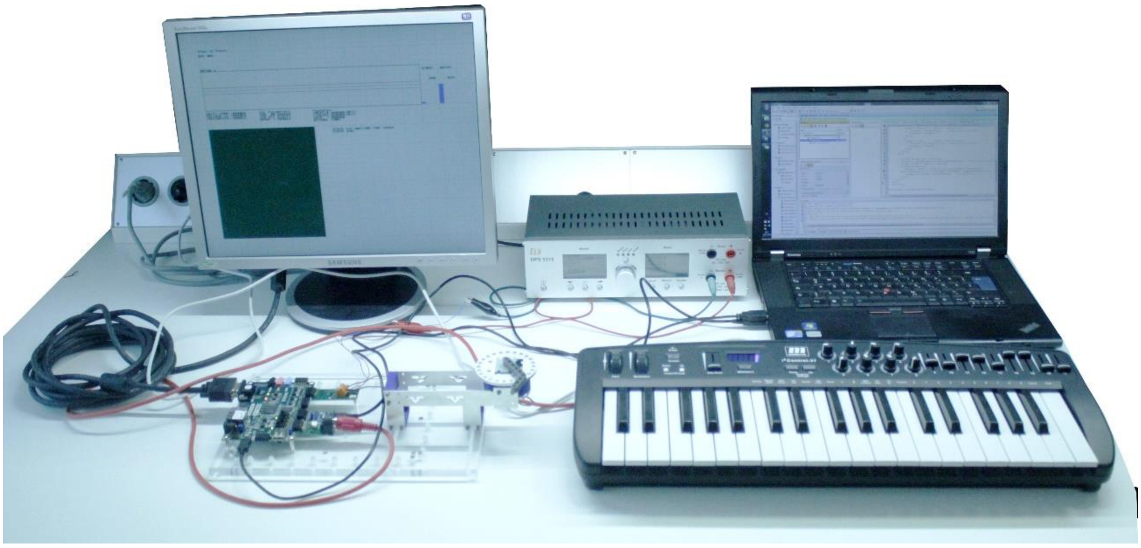


Figure 1: System Desktop Overview

4.1.1.1. Items

This is a list of all the items present in the system.

FPGA Development Platform
Model: Zynq ZYBO 7000 Development Board
Technical information:
https://www.digilentinc.com/Products/Detail.cfm?Prod=ZYBO

Figure 2: ZYBO Board

Motor Driver & ADC Sensor for CSL Pmod
Model: Laboratory design with the TI ADS1203
Technical information:
http://www.ti.com.cn/cn/lit/ds/symlink/ads1203.pdf

Figure 3: Pmod Motor Drive

Measure platform adapter Pmod
Model: Laboratory Design
Technical information: Irrelevant

Figure 4: Pmod Measure

MIDI Input adapter Pmod
Model: Laboratory Design
Technical information: Irrelevant

Figure 5: Pmod MIDI Input

UART USB adapter Pmod
Model: PmodUSBUART
Technical information:
https://www.digilentinc.com/Products/Detail.cfm?NavPath=2,401,928&Prod=PMOD-USB-UART

Figure 6: Pmod UART USB

MIDI controller keyboard
Model: Miditech I2 Control-37 Black Edition
Technical information: Irrelevant

Figure 7: MIDI Keyboard

VGA Screen
Model: Irrelevant
Technical information: Irrelevant

Figure 8: VGA Screen

Power Supply
Model: ELV DPS 5315
Technical information: http://www.elv.de/dual-power-supply-dps-5315-fertiggeraet.html

Figure 9: Power Supply

MIDI wire
Model: Irrelevant
Technical information: Irrelevant

Figure 10: MIDI Wire

Supply wire
Model: Irrelevant
Technical information: Irrelevant

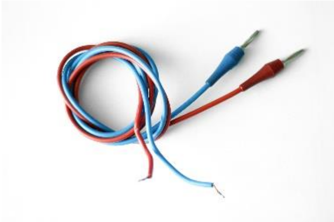


Figure 11: Supply Wire

USB-MicroUSB wire
Model: Irrelevant
Technical information: Irrelevant

Figure 12: MicroUSB Wire

System Platform
Model: Irrelevant
Technical information: Irrelevant

Figure 13: System Platform

Motor Structure
Model: Irrelevant
Technical information: Irrelevant

Figure 14: Motor Structure

LEGO DC Motor
Model: LEGO 71427
Technical information:
https://alpha.bricklink.com/pages/clone/catalogitem.page?P=71427c01#T=C

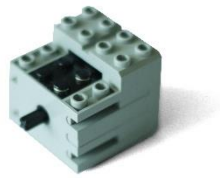


Figure 15: LEGO Motor

PC Computer
Model: ThinkPad type 4349-BL1
Technical information: Irrelevant

Figure 16: PC

4.1.1.2. System connection

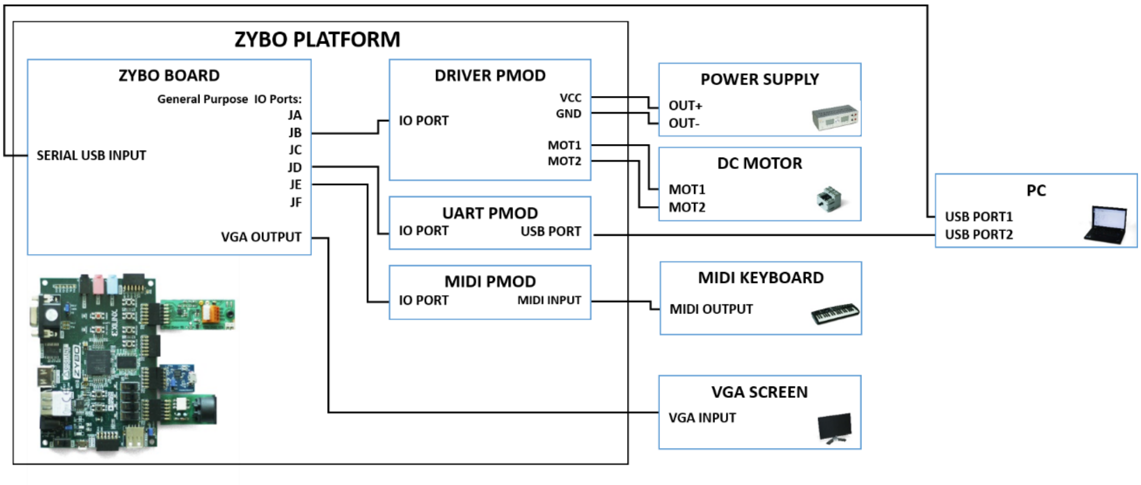


Figure 17: Stay In Touch System Connection Diagram

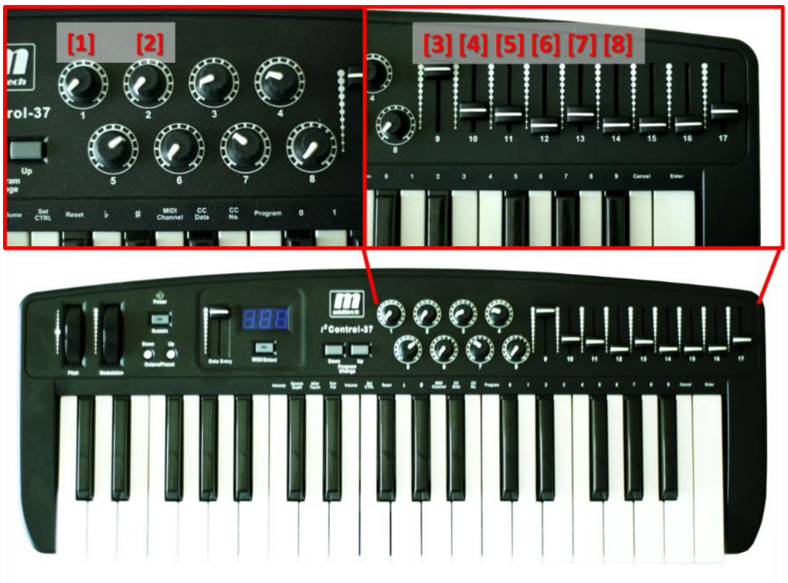
The core of the system is the ZYBO platform composed by the ZYBO Board and the connected *Pmods*. The MIDI Keyboard and the VGA Screen are *User Interface* elements used to set parameters and monitored the process. The ZYBO board is powered by the PC through the USB connection and the *Driver Pmod* is powered by the *Power Supply* with a voltage of 8V (Mentioned in the configuration section).

4.1.1.3. User interface

This section contain a detailed description of the *User Interface* elements of the system.

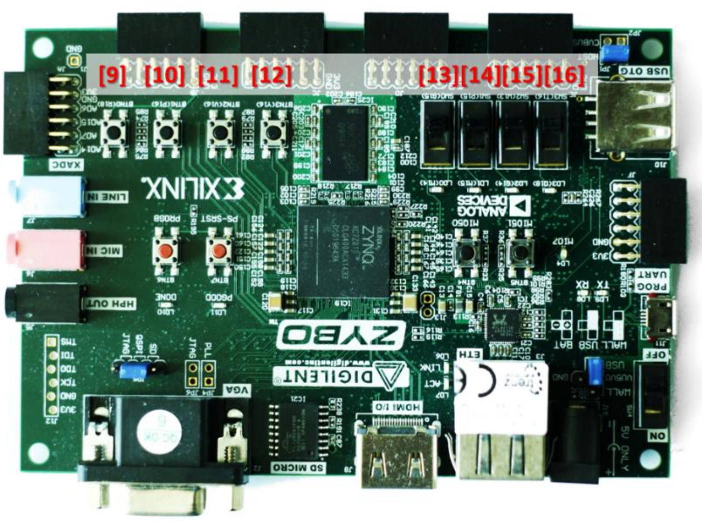
4.1.1.3.1. Parameter Interface

The parameter Interface is both done through the MIDI keyboard and the *ZYBO Board*. The parameter controls of the system are:



MIDI Keyboard Parameters:
[1] Drive Control
[2] Sense Control
[3] Brake Control
[4] Threshold ON
[5] Threshold OFF
[6] Inertia Control
[7] Timeout Control
[8] Search Control

Figure 18: Stay In Touch Keyboard Parameter Reference

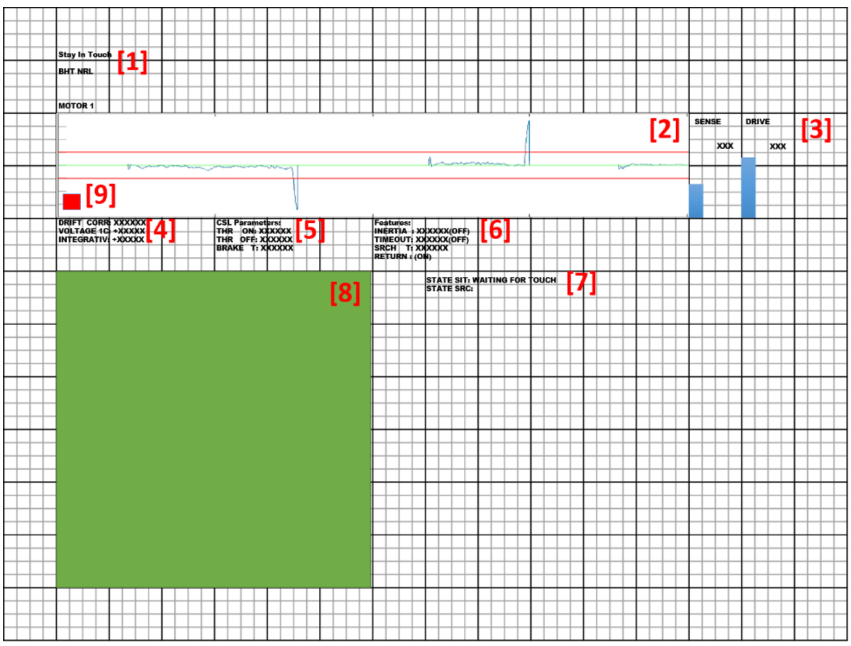


MIDI Keyboard Parameters:
[9] Graphic Chart Pause
[10] None
[11] None
[12] None
[13] Motor ON/OFF
[14] Search ON/OFF
[16] Inertia ON/OFF
[16] Return ON/OFF

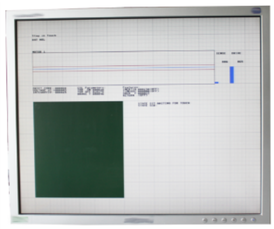
Figure 19: Stay In Touch ZYBO Parameter Reference

4.1.1.3.2. VGA Interface

The different parameters and settings of the system are displayed for monitoring in a VGA showing the following information:



Screen Interface Elements:
[1] Title
[2] Voltage & Threshold Graphic
[3] DS Parameters
[4] Sense Measurements
[5] SIT Basic Parameters
[6] Mode Parameters
[7] SIT&SRC State
[8] State Color Display
[9] Graphic STOP signal



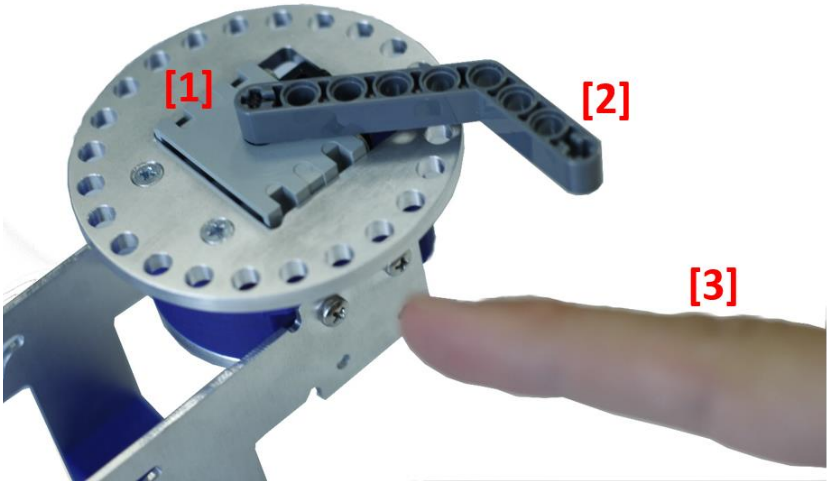
VGA Screen Photograph

Figure 20: Stay In Touch VGA Interface Schematic

A schematic image has been used due to the difficulty of photographing a VGA Screen. However a small photograph is attached to show the resemblance between both representations.

4.1.2. System Elements

This section contains different labels given to different elements to identify them in the context of the document.



System Elements:
[1] Motor Joint
[2] Sensing Platform
[3] External Object

Figure 21: Stay In Touch System Elements

4.2. System Functionality Analysis & Settings

This section contains a description of the System Functionality through its behavior.

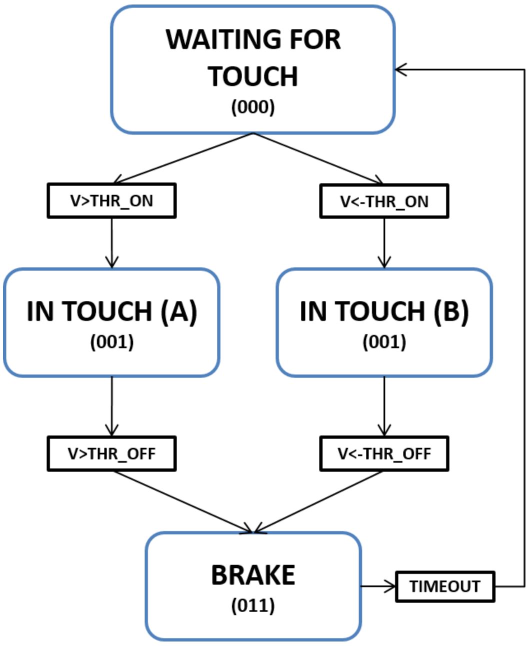
4.2.1. Basic System

The *Basic Configuration* consist in a simple Bi-directional *CSL Stay In Touch (SIT)* system in which the parameters that can be set are:

- **CSL Parameters**
  - ***Sense Period Time:*** Establishes the *Sense* period duration of the *CSL State Machine* (in ms).
  - ***Drive Period Time:*** Establishes the *Drive* period duration of the *CSL State Machine* (in 0,512≈0,5ms).
- **SIT Parameters**
  - ***Threshold ON:*** Establishes the threshold the voltage measurement of one measurement cycle (*Sense*) has to surpass in order to activate the *In Touch* state.
  - ***Threshold OFF:*** Establishes the threshold the voltage measurement of one measurement cycle (*Sense*) has to surpass in order to deactivate the *In Touch* state.
  - ***Brake Time:*** Establishes the number of *Sense* periods the motor brakes before a new measurement is done (To avoid invalid readings during the motor relax period).

The *Basic SIT Stay in Touch* system implements a simple state machine to drive the motor. Its algorithm consists in a state machine with 3 states (*Waiting for Touch, In Touch* and *Brake*) whose aim is to keep contact with any external object that contacts the *Sensing Platform*. Once the *Sensing Platform* has been contacted, the system drives the motor with a fixed voltage in the direction the E*xternal Object* has touched it (*In Touch*).

Once the systems detects the *external object* is no longer in contact with the mechanism, the system brakes the motor (*Brake*) and returns to an idle state (*Waiting For Touch*) awaiting a new contact. The system works symmetrically in both directions, which will be labelled as directions A and B.



State Description:

WAITING FOR TOUCH: The motor is stopped and the Drive time is 0.

- IN1 <= 0
- IN2 <= 0

IN TOUCH: The motor is driven with a constant voltage during a *Drive* time set by *Drive Period Time.*

- Direction A: IN1<=0; IN2<= 1
- Direction B: IN1<=1; IN2<= 0

BRAKE: The motor is stopped by a voltage shortcut both in the *Sense* time. Drive time is 0.

- IN1 <= 1
- IN2 <= 1

Figure 22: Basic CSL Stay In Touch State Machine

4.2.1.1 Configurations

Seven different relevant configurations have been studied and recorded in order to analyze different possibilities or detect wrong behaviors. The configurations studied do not cover all the possible setting combinations the system can have but a selection of relevant configurations that lead to interesting results. All of these configurations are shown in the video *CSL_SIT_BasicBehavior.avi*. The different configurations are properly labelled in the video with the same references shown in this document.

The value of the configuration parameters is shown both in a qualitatively and quantitative way. The qualitatively parameters aim to establish a guidance of the implementation of this system in other hardware or software supports while the quantitative parameters aim to reproduce the configurations. The qualitative parameters are written in a scale: *VERY LOW, LOW, MEDIUM, HIGH* and *VERY HIGH*. All this configurations have been implemented powering the *ADC DAC Pmod* with an 8V voltage.

The Voltage unit used all over the document is not measured in Volts(V) but the proportional measurement done by the TI ADS1203 present in the *Driver Pmod*. This ADC acts as a single *delta-sigma* converter that outputs a stream of HIGH and LOW levels proportional to the Input voltage.[4]

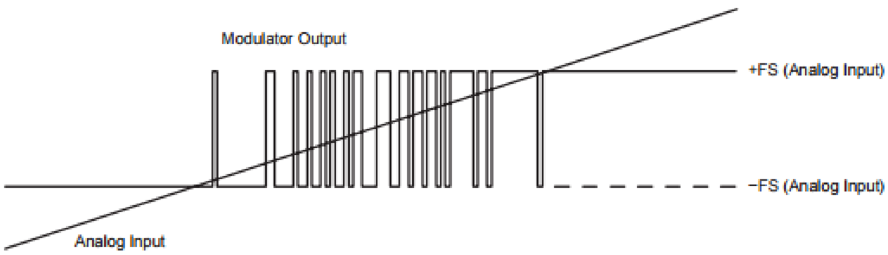


Figure 23: TI ADS1203 ADC Converter Behavior[4]

[4] ADS1203 AD Converter Datasheet (nd), Texas Instruments, http://www.ti.com.cn/cn/lit/ds/symlink/ads1203.pdf

Therefore, the voltage units present in the graphics are not meant to be a proper voltage measurement but a qualitative value of the differential voltage generated by the motor during a *Sense* cycle.

The configurations studied are:

Configuration 1 (Optimal):

This configuration presents the optimal behavior of the *CSL Stay In Touch* system. It is considered the best configuration first due to its high sensibility, secondly to its proper haptic force appliance, thirdly to its low brake angular derivation and finally to its short brake dead-zone.

Configuration 1	Optimal	
CSL Parameters:		
Sense Time	VERY LOW	006/127
Drive Time	MEDIUM	065/127
SIT Parameters:		
Threshold ON	VERY LOW	48/762
Threshold OFF	HIGH	10176/12192
Brake Time	LOW	46/211

The voltage (one *Sense* cycle) measurements of this configuration are shown in this graphic with the thresholds properly labelled.

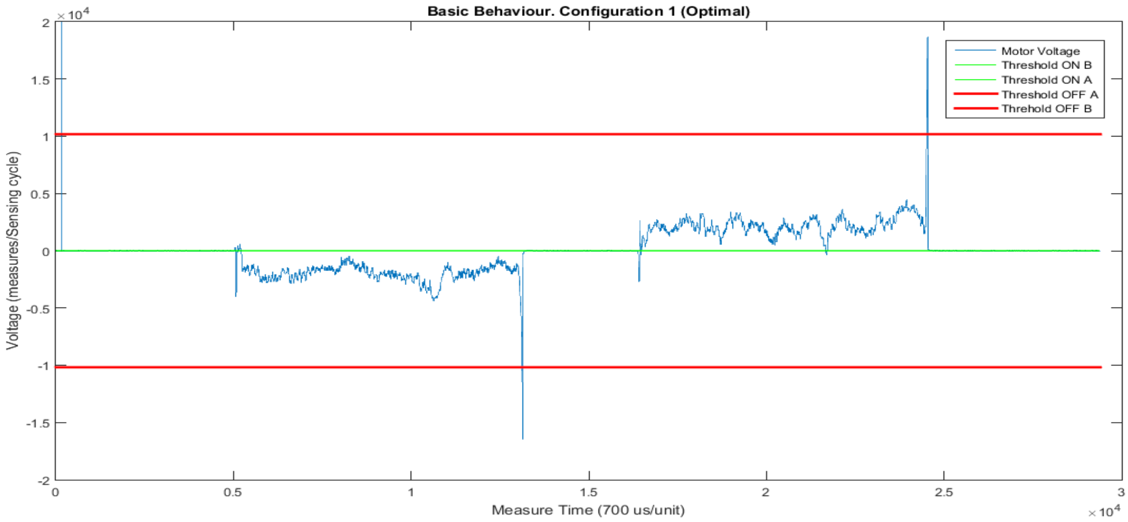


Figure 24: Stay In Touch Basic Behavior Configuration 1

In the graphic it can be seen through the measurements that the voltages of a normal *CSL Stay In Touch* interaction are clearly under the *Threshold OFF*, allowing comfortable haptic resistance to the *Motor Joint* movement without triggering the brake (0,5 to 1,3 and 1,7 to 2,5 in the time axis).

Configuration 2 (Low Drive):

This configuration presents a valid behavior of the *CSL Stay In Touch* system. It has a low drive time which mostly affects the haptic force applied by the motor. In particular conditions where the *External Object* handled is fragile or extremely soft, this configuration is preferred to *Configuration 1*.

The sensibility is high, and the angular brake derivation superior to *Configuration* 1 but still low.

Configuration 2	Optimal Low Drive	
CSL Parameters:		
Sense Time	VERY LOW	006/127
Drive Time	VERY LOW	009/127
SIT Parameters:		
Threshold ON	VERY LOW	48/762
Threshold OFF	HIGH	10176/12192
Brake Time	LOW	46/211

The voltage (one *Sense* cycle) measurements of this configuration are shown in this graphic with the thresholds properly labelled.

Figure 25: Stay In Touch Basic Behavior Configuration 2

This graphic, very similar to *Configuration 1* shows a comfortable area where the *CSL Stay In Touch* interaction can be developed way under the *Threshold OFF* (0,25 to 1,3 and 1,7 to 2,7 in the time axis). The release peaks are smaller to the ones in *Configuration 1* due to the low drive and therefore lower motor angular moment after the release. Peaks are also represented less immediate.

Configuration 3 (Low Threshold OFF):

This configuration presents an incorrect behavior of the *CSL Stay In Touch*. It is a modified *Configuration 1* with a lower *Threshold OFF*. Due to this modification, the system finishes its *In Touch* state prematurely failing to follow the external object.

Configuration 3	Low Threshold OFF	
CSL Parameters:		
Sense Time	VERY LOW	006/127
Drive Time	MEDIUM	065/127
SIT Parameters:		
Threshold ON	VERY LOW	48/762
Threshold OFF	LOW	4128/12192
Brake Time	LOW	46/211

The voltage (one *Sense* cycle) measurements of this configuration are shown in this graphic with the thresholds properly labelled.

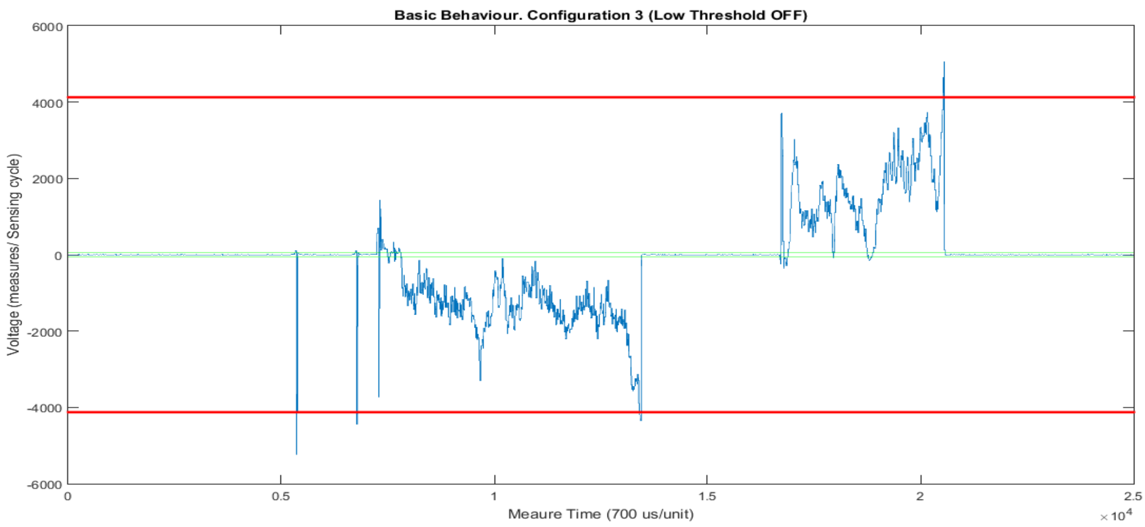


Figure 26: Stay In Touch Basic Behavior Configuration 3

This graphic shows a little range for the *CSL Stay In Touch* interaction (Range between Threshold ON and Threshold OFF) that results in undesired trigger of the motor brake (For example in 0,6 and 0,7 in the time axis).

Configuration 4 (High Threshold ON):

This configuration presents a valid behavior of the *CSL Stay In Touch*. It is a modified *Configuration 1* with a higher *Threshold ON*. Due to this modification, the system triggers its *In Touch* state with a lower sensibility requiring a higher push during the *Waiting For Touch* state rather than just contact.

Configuration 4	High Threshold ON	
CSL Parameters:		
Sense Time	VERY LOW	006/127
Drive Time	MEDIUM	065/127
SIT Parameters:		
Threshold ON	VERY HIGH	762/762
Threshold OFF	HIGH	10176/12192
Brake Time	LOW	46/211

The voltage (one *Sense* cycle) measurements of this configuration are shown in this graphic with the thresholds properly labelled.

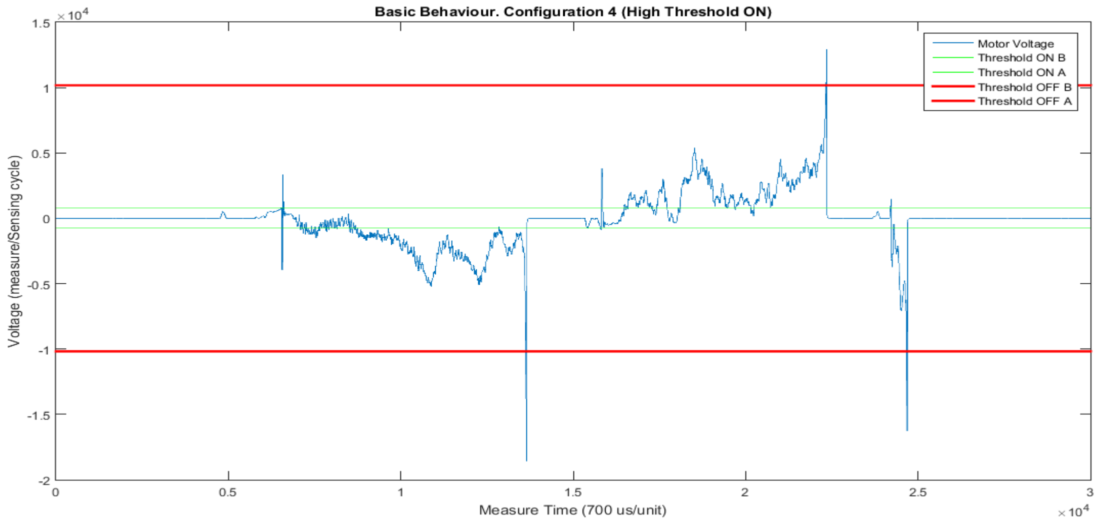


Figure 27: Stay In Touch Basic Behavior Configuration 4

The graphic shows a wide *Waiting For Touch* area that allows the motor to receive contact without triggering the *In Touch* state (For example in the region form 0,5 to 0,6 or 1,4 to 1,6 in the time axis).

Configuration 5 (High Sense):

This configuration presents an incorrect behavior of the *CSL Stay In Touch*. It sets a high *Sense* period time resulting in a non-continuous motor movement that provides not only an erratic haptic contact towards the external object but also incorrect readings that can trigger the *In Touch* state to finish unexpectedly.

Furthermore, a longer *Sense* state does not provide any measuring advantages as the voltage parameters (*Threshold ON*, *Threshold OFF*) have their values adjusted proportionally to the *Sense Time* parameter.

Configuration 5	High Sense	
CSL Parameters:		
Sense Time	MEDIUM	048/127
Drive Time	MEDIUM	065/127
SIT Parameters:		
Threshold ON	VERY LOW	96/6096
Threshold OFF	HIGH	71424/97536
Brake Time	LOW	08/26

This graphic has not been imported due to the resolution of the serial module, that has an upper limit to reduce the transmission data time and therefore high sense values would trigger an overflow in the serial communication.

Configuration 6 (Low Brake):

Configuration 6 presents an incorrect behavior of the CSL *Stay In Touch*. It sets a very low *Brake Time* resulting in an unstable behavior due to voltage measurements during the motor relax period. As a consequence, this erratic measures trigger the *In Touch* state without any external object present and returning immediately to the cyclic incorrect behavior.

Configuration 6	Low Brake	
CSL Parameters:		
Sense Time	VERY LOW	006/127
Drive Time	MEDIUM	065/127
SIT Parameters:		
Threshold ON	VERY LOW	48/762
Threshold OFF	HIGH	10176/12192
Brake Time	VERY LOW	10/211

The voltage (one *Sense* cycle) measurements of this configuration are shown in this graphic with the thresholds properly labelled.

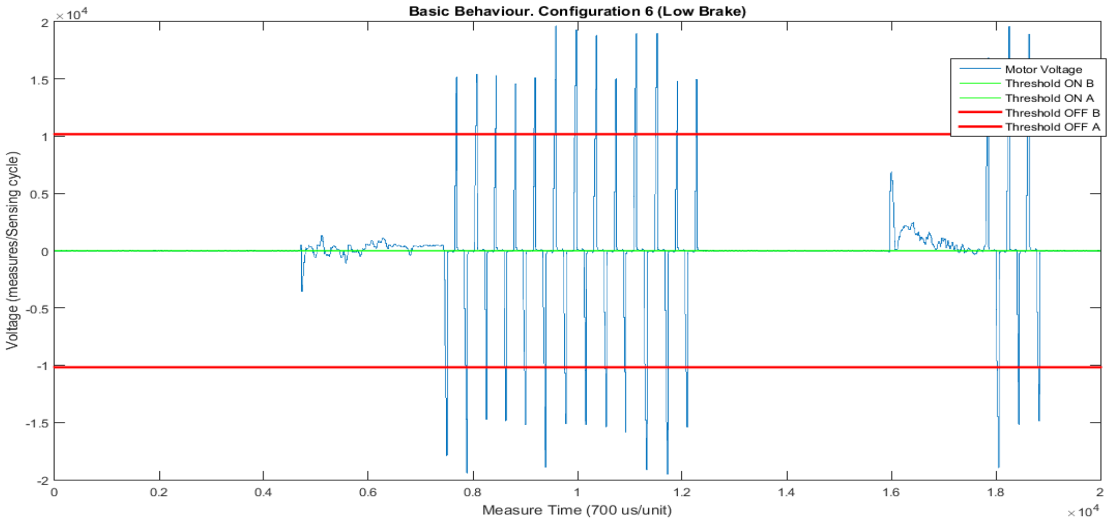


Figure 28: Stay In Touch Basic Behavior Configuration 6

The graphic shows the instability caused by the low brake and the iteration between the *IN TOUCH A* and *B* state.

Configuration 7 (High Brake)

Configuration 7 presents a valid behavior of the CSL *Stay In Touch*. It is a modified *Configuration 1* with a very high *Brake Time* resulting in a long brake dead-zone that is unnecessary. However, this can be preferred in systems in which the motor controls a bigger mass as the time needed to bring back the motor to relax is longer.

Configuration 7	HIGH BRAKE	
CSL Parameters:		
Sense Time	VERY LOW	006/127
Drive Time	MEDIUM	065/127
SIT Parameters:		
Threshold ON	VERY LOW	48/762
Threshold OFF	HIGH	10176/12192
Brake Time	VERY HIGH	211/211

The voltage (one *Sense* cycle) measurements of this configuration are shown in this graphic with the thresholds properly labelled.

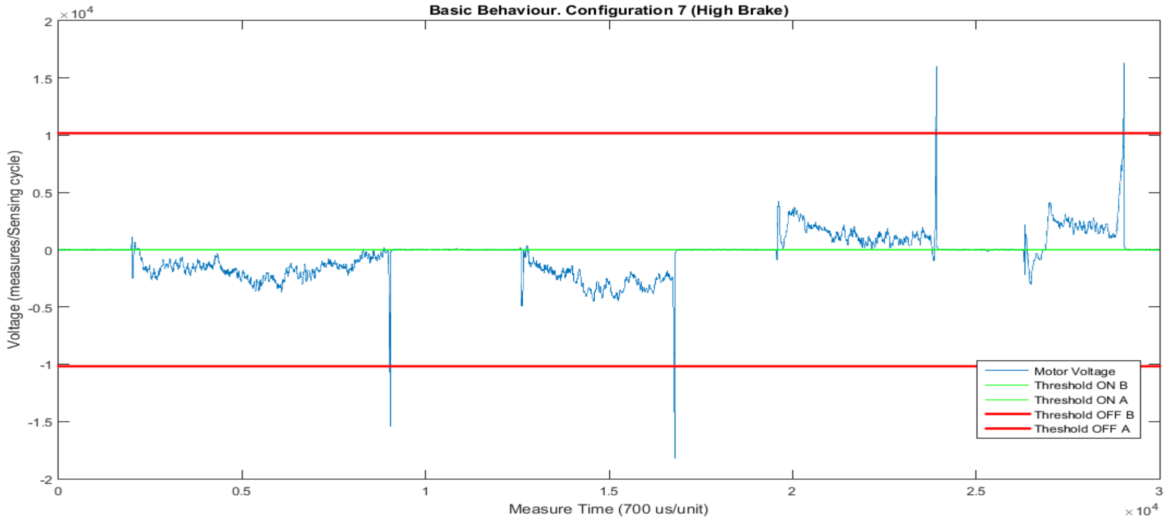


Figure 29: Stay In Touch Basic Behavior Configuration 7

This graphic shows the long dead zones in which the motor is braked and unresponsive as the *CSL Stay In Touch* cannot be triggered.

4.2.2. Additional Features

In order to improve and enhance the experience and to add other functionalities to the CSL *Stay In Touch* system, three modular functionalities had been added:

- *Inertia Mode*
- *Search Mode*
- *Return Mode*

These modes can both be activated individually or simultaneously through the *ZYBO Board* switches (SW1 for Inertia, SW2 for Search and SW3 for Return) as described in section *4.1.1.3.1. Parameter Interface*.

4.2.2.1. Inertia Mode

The objective of the *Inertia Mode* is to continue the *In Touch* state for a fixed time trying to re-contact the external object. The *In Touch Inertia* state is placed in between the *In Touch* and the *Brake* state. If the external object is contacted before the timeout, the state machine switches back to *In Touch* state. If not, the state machine switches to *Brake* to stop the motor and prepare a new contact. Its only parameter is:

- Inertia Mode Parameter
    - ***Inertia Time:*** Establishes the duration of the *Inertia Mode* measurements in Drive cycles.

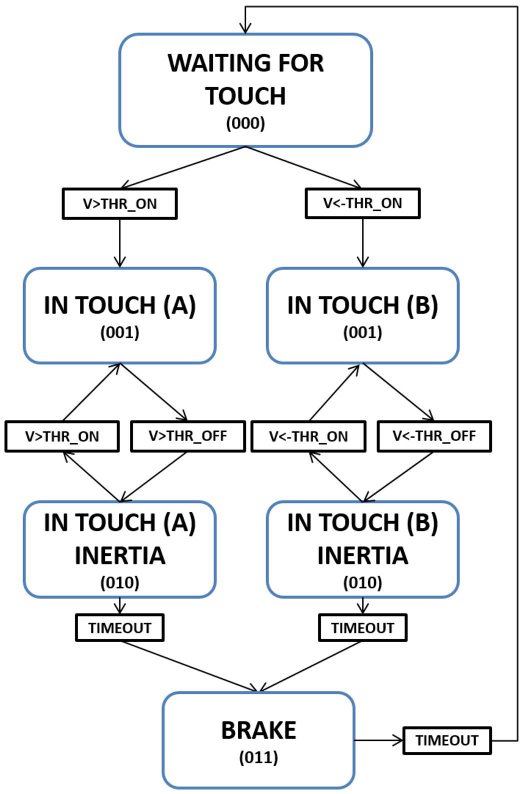


Figure 30: CSL SIT Stay In Figure 2: Touch Inertia Mode State Machine

State Description:

WAITING FOR TOUCH: The motor is stopped and the Drive time is 0.

- IN1 <= 0
- IN2 <= 0

IN TOUCH: The motor is driven with a constant voltage during a Drive time set by Drive Period Time.

- Direction A: IN1<=0; IN2<= 1
- Direction B: IN1<=1; IN2<= 0

IN TOUCH INERTIA: The motor is driven with a constant voltage during a Drive time set by Drive Period Time.

- Direction A: IN1<=0; IN2<= 1
- Direction B: IN1<=1; IN2<= 0

BRAKE: The motor is stopped by a voltage shortcut both in the Sense time. Drive time is 0.

- IN1 <= 1
- IN2 <= 1

4.2.2.1.1. Configurations

Two different configurations have been studied in order to analyze different possibilities or detect wrong behaviors.

The configurations studied do not cover all the possible setting combinations the system can have but a selection of relevant configurations which lead to interesting results. All of these configurations are shown in the video *SIT_InertiaModeBehavior.avi*. The different configurations are properly labelled in the video with the same references shown in this document.

As in the previous system the parameters have been displayed both in a qualitative and quantitative way.

Configuration 1 (Optimal):

This configuration presents the optimal behavior of the *CSL Inertia Mode*. It is considered the best configuration because of its low angular derivation and its capability of easily gaining contact with the external object if this has lost its contact due to a high movement speed in the direction of the motor.

Configuration 1	Optimal	
CSL Parameters:		
Sense Time	VERY LOW	006/127
Drive Time	MEDIUM	065/127
SIT Parameters:		
Threshold ON	VERY LOW	48/762
Threshold OFF	HIGH	10176/12192
Brake Time	LOW	46/211
Inertia Parameters:		
Inertia Time	VERY LOW	06/211

The voltage (one *Sense* cycle) measurements of this configuration are shown in this graphic with the thresholds properly labelled.

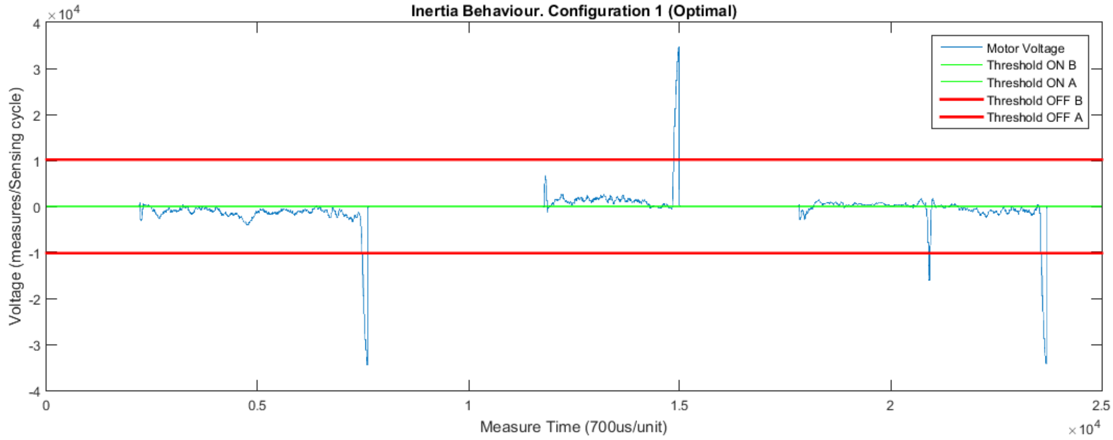


Figure 31: Stay In Touch Inertia Mode Configuration 1

This graphic shows a *Inertia Mode* in two first routines which do not succeed in returning to contact with the external object (0,75 and 1,5 in the time axis) and a third attempt that interrupts the *Inertia* routine and returns to the *In Touch A* state (2,2 in the time axis).

Configuration 2 (High Inertia Time):

This configuration presents a valid behavior of the *CSL Inertia* mode. It has a high *Inertia Time* which, as a consequence, presents a very high angular derivation (of even several whole motor rotations). However, the optimal *Inertia Time* needed is a characteristic dependent of the motor torque.

Configuration 2	High Inertia Time	
CSL Parameters:		
Sense Time	VERY LOW	006/127
Drive Time	MEDIUM	065/127
SIT Parameters:		
Threshold ON	VERY LOW	48/762
Threshold OFF	HIGH	10176/12192
Brake Time	LOW	46/211
Inertia Parameters:		
Inertia Time	HIGH	150/211

The voltage (one *Sense* cycle) measurements of this configuration are shown in this graphic with the thresholds properly labelled.

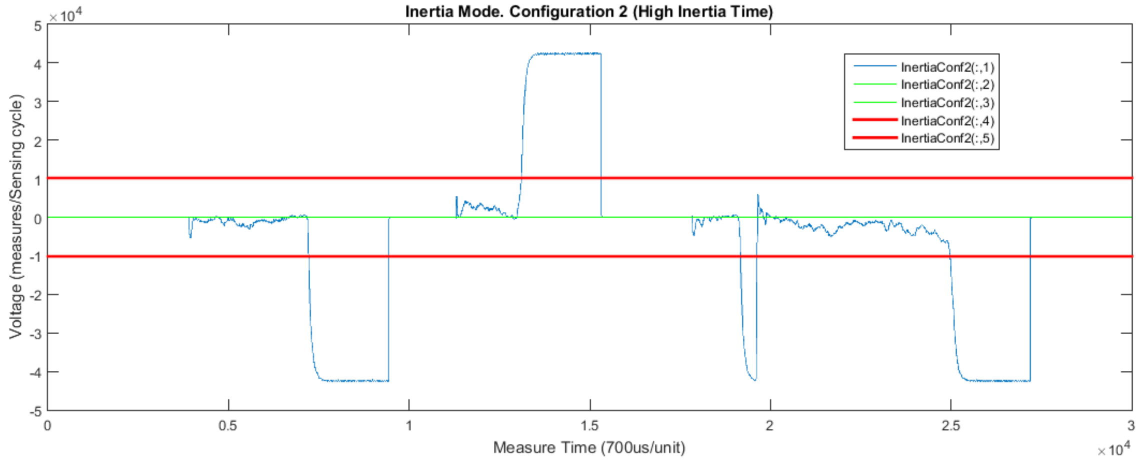


Figure 32: Stay In Touch Inertia Mode Configuration 2

This graphic shows the *Inertia* routine clearly. At the end of each *In Touch* state, once the external object ends the contact with the sensing platform, the motor continues its motion accelerating until a constant speed as in the first two routines (0,7 to 0,9 and 1,3 to 1,6 in the time axis). In the third routine, the *Inertia* state is triggered but interrupted by a new contact with the external object (1,9 in the time axis) which triggers the return to the *In Touch A* state.

4.2.2.2. Search Mode

The objective of the *Search Mode* is to perform a periodic search for external objects next to the system *sensing extension* by exploring the surrounding area. It is triggered from the *Waiting For Touch* state after a set timeout and then enters the *Search* state.

The *Search* state implements a new state machine which controls the motor drive. It has 3 states (*Search A, Search B* and *Search Pause*). Its parameters of this state are:

- Search Mode Parameters
  - ***Timeout:*** Establishes the amount of time that the system stays in *Waiting For Touch* state before entering *Search* state. It is measured in seconds with a range from 0 to 127.
  - ***Search Time:*** Establishes the duration of the different *Search A, Search B* states. It is measured in *Drive* cycles.

When the *SIT Search* state is triggered, the first *Search A* drive the motor on direction A for 1/3 of the *Search Time*. During this state, if an *external object* is contacted, the *Search* state terminates abruptly and the *In Touch A* state is triggered. If not, the *In Touch A* ends and a *Search Pause* state begins. The objective of this *Search Pause* state is to relax the motor and avoid false readings during the next state. The *Search Pause* state brakes the motor during a *Brake Time* duration.

Once the first *Search Pause* state finishes, the *Search B* state is triggered and drives the motor in direction B during 1/2 of the *Search Time*. It has a similar performing as the *Search A* but in the opposite direction. Once the *Search B* state finishes, another *Search Pause* state occurs and then another *Search A* during 1/3 of the *Search Time*.

The objective of performing the *Search A* state twice is to return the sensing extension back to the original position prior to the *SIT Search* state. This, however, has not been achieved with much accuracy.

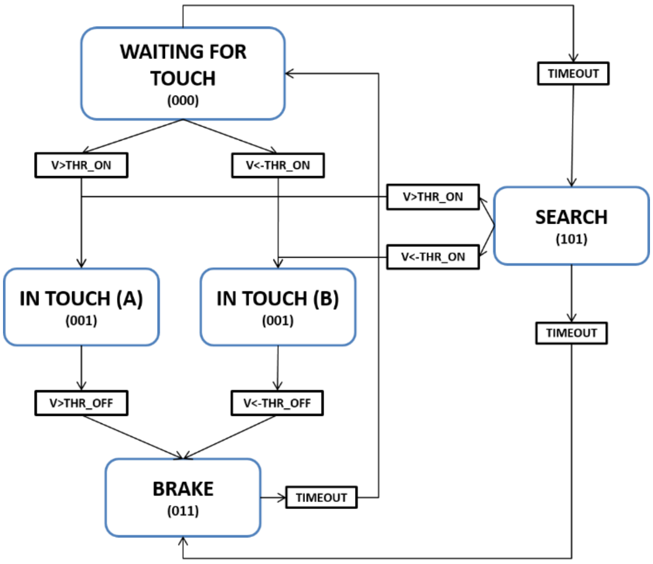


Figure 33: Search Mode CSL Stay In Touch State Machine

State Description:

WAITING FOR TOUCH: The motor is stopped and the *Drive* time is 0.

- IN1 <= 0
- IN2 <= 0

IN TOUCH: The motor is driven with a constant voltage during a *Drive* time set by *Drive Period Time*.

- Direction A: IN1<=0; IN2<= 1
- Direction B: IN1<=1; IN2<= 0

SEARCH: The motor is driven by the *SEARCH* state machine states.

BRAKE: The motor is stopped by a voltage shortcut both in the *Sense* time. *Drive* time is 0.

- IN1 <= 1
- IN2 <= 1

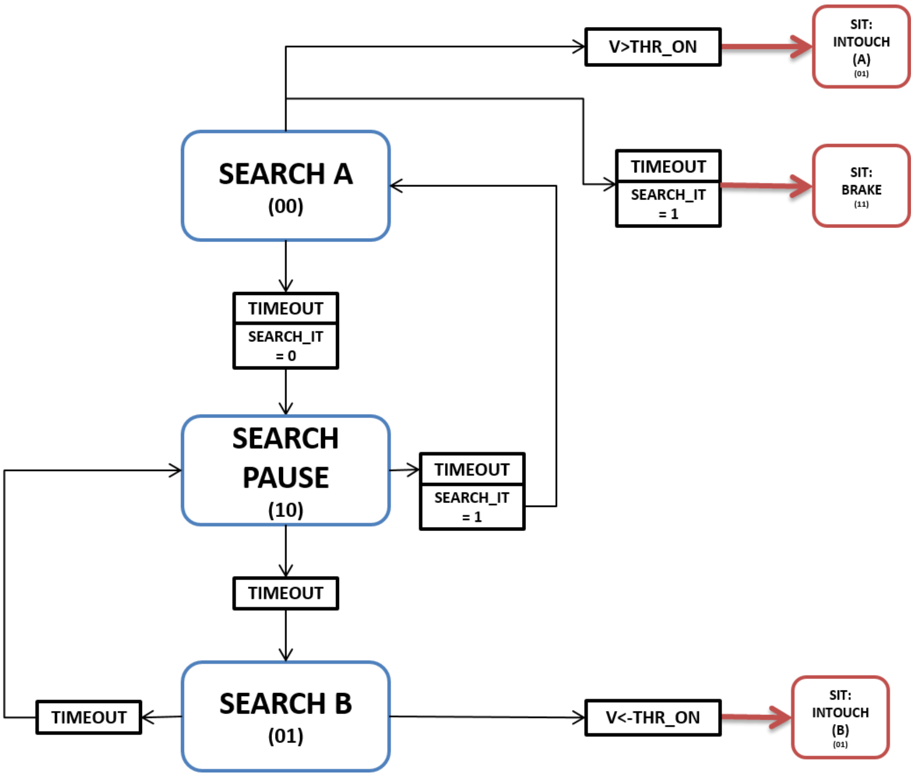


Figure 34: Search State Machine

4.2.2.2.1. Configurations

Two different configurations have been studied in order to analyze different possibilities or detect wrong behaviors.

The configurations studied do not cover all the possible setting combinations the system can have but a selection of relevant configurations that lead to interesting results. All of these configurations are shown in the video *SIT_SearchModeBehavior.avi*. The different configurations are properly labelled in the video with the same references shown in this document.

As in the previous system the parameters have been displayed both in a qualitative and quantitative way.

The *Timeout* has not been studied in the configurations as its effects are obvious.

Configuration 1(Optimal):

This configuration presents an optimal behavior of the *Search Mode*. It has a small angular exploration and an accurate return to the original position of the *Sensing Extension*.

Configuration 1	Optimal	
CSL Parameters:		
Sense Time	VERY LOW	006/127
Drive Time	MEDIUM	065/127
SIT Parameters:		
Threshold ON	VERY LOW	48/762
Threshold OFF	HIGH	10176/12192
Brake Time	LOW	46/211
Search Parameters:		
Search Time	LOW	10/84
Timeout	IRRELEVANT	-

The voltage (one *Sense* cycle) measurements of this configuration are shown in this graphic with the thresholds properly labelled.

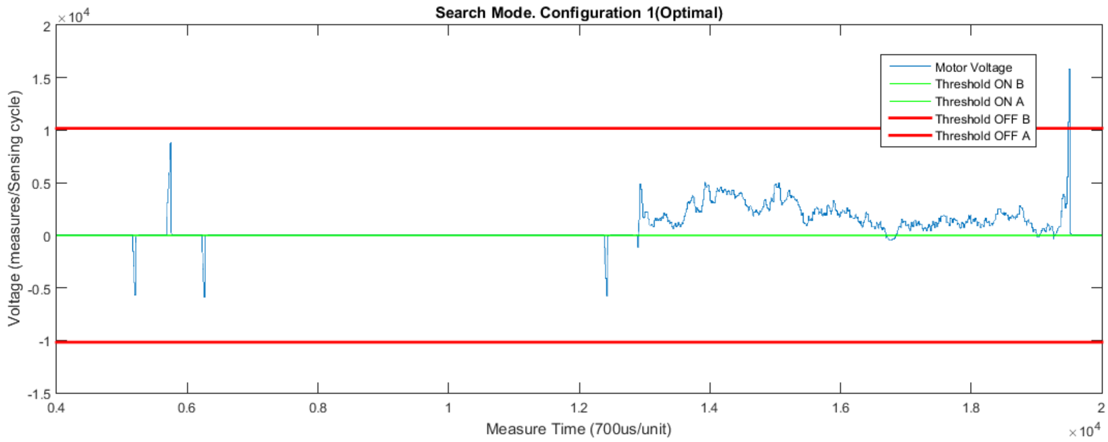


Figure 35: Stay In Touch Search Mode Configuration 1

The graphic shows a complete search process at first without success contacting an external object (0,5 to 0,7 in the time axis). It is seen a very short acceleration of the motor in both directions (due to the low *Search Time*) without it achieving a constant speed. During the second *Search* routine, an external object is contacted during the *Search B* state, exiting the *SIT Search* state machine and entering the *Search B* state (1,3 in the time axis) for a haptic interaction until the external object is not contacted anymore and the motor brakes.

Configuration 2 (High Search Time):

This configuration presents a valid behavior of the *Search* mode. It has a high angular exploration and a very inaccurate return to the original position of the *Sensing Extension*.

Configuration 2	High Search Time	
CSL Parameters:		
Sense Time	VERY LOW	006/127
Drive Time	MEDIUM	065/127
SIT Parameters:		
Threshold ON	VERY LOW	48/762
Threshold OFF	HIGH	10176/12192
Brake Time	LOW	46/211
Search Parameters:		
Search Time	VERY HIGH	84/84
Timeout	IRRELEVANT	-

The voltage (one *Sense* cycle) measurements of this configuration are shown in this graphic with the thresholds and states properly labelled.

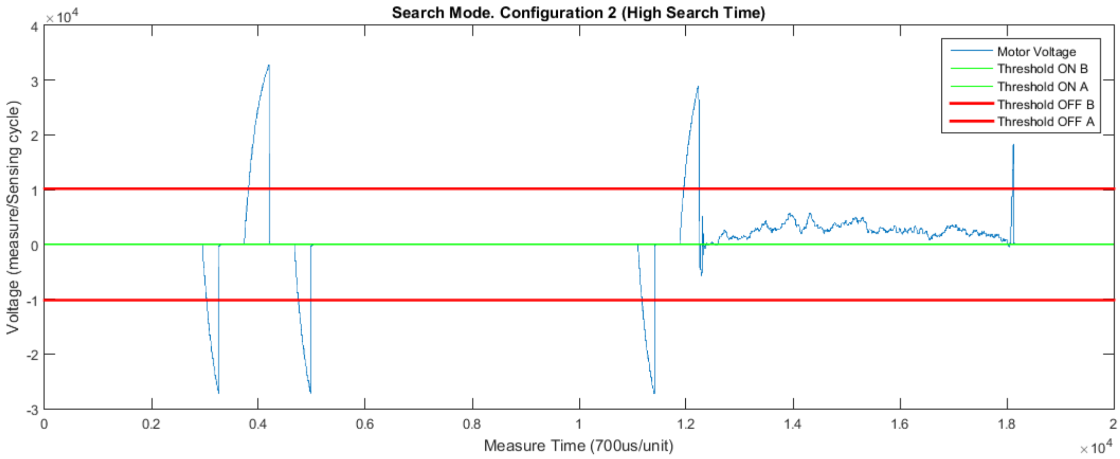


Figure 36: Stay In Touch Search Mode Configuration 2

The graphic shows a complete search process at first without success contacting an external object (0,3 to 0,5 in the time axis). It is seen an acceleration of the motor in both directions without it achieving a constant speed. During the second *Search* routine, an external object is contacted during the *Search B* state, exiting the *Search* state machine and entering the *In Touch B* state (1,2 in the time axis).

This configurations shows clearly the aim and process of the *Search* mode, however it is not optimal due to the motor torque, which works better with low *Search Time (Configuration 1)*.

4.2.2.3. Return Mode

The objective of the *Return* mode is to perform a returning movement after the *In Touch* to compensate the brake angular derivation and move the *Sensing Extension* to the position in which the contact with the external object was lost. It has no parameters since it is proportional to the *Drive Time* and it is implemented through a *Return* state inside the *SIT* state machine.

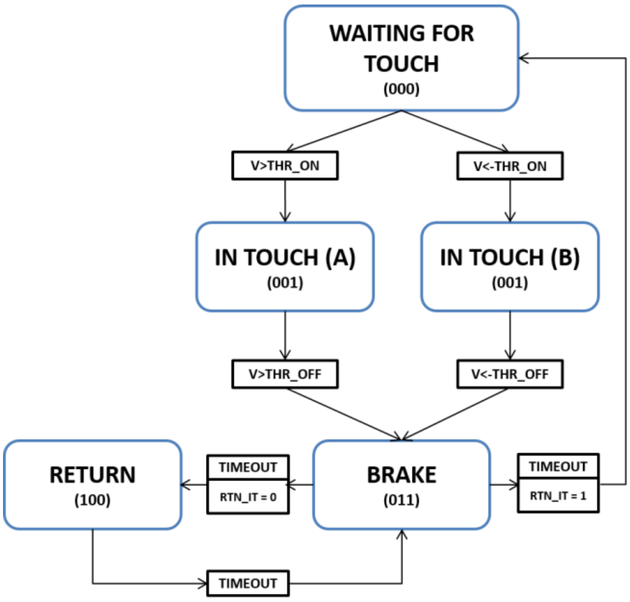


Figure 37: Search Mode CSL Stay In Touch State Machine

State Description:

WAITING FOR TOUCH: The motor is stopped and the *Drive* time is 0.

- IN1 <= 0
- IN2 <= 0

IN TOUCH: The motor is driven with a constant voltage during a *Drive* time set by *Drive Period Time*.

- Direction A: IN1<=0; IN2<= 1
- Direction B: IN1<=1; IN2<= 0

BRAKE: The motor is stopped by a voltage shortcut both in the *Sense* time. *Drive* time is 0.

- IN1 <= 1
- IN2 <= 1

RETURN: The motor is driven with a constant voltage during a *Drive* time set by a fraction of the *Drive Period Time*.

- Direction A: IN1<=1; IN2<= 0
- Direction B: IN1<=0; IN2<= 1

4.2.2.2.1. Configurations

Only one configuration has been studied in order to show the behavior of the feature. This configuration is shown in the video *SIT_ReturnModeBehavior.avi*.

Configuration 1 (Optimal):

This is a simple optimal configuration of the basic CSL *Stay In Touch* system but with the *Return* mode activated. For this configuration no graphic has been made because it is considered irrelevant.

4.2.2.4. Combination of Modes

All the modes listed can be activated or deactivated simultaneously, increasing the features the system has and its possibilities.

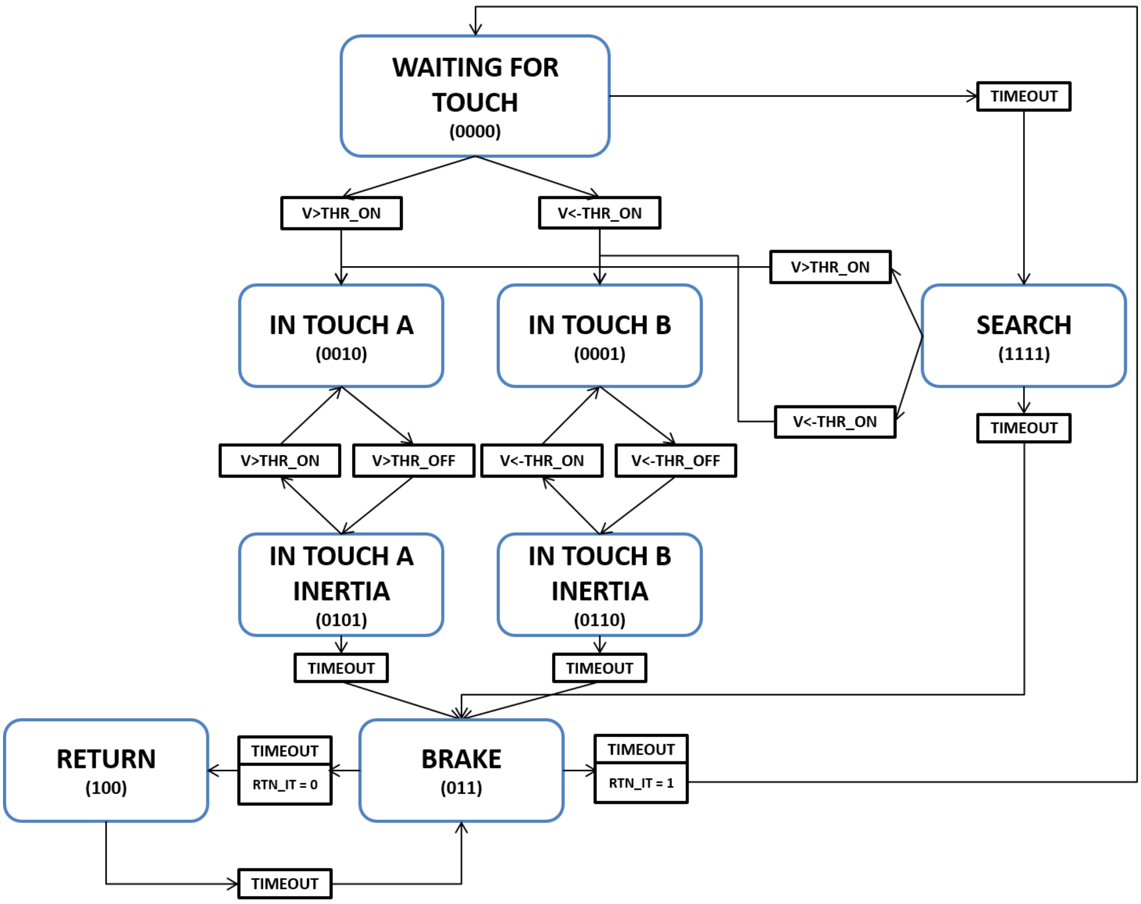


Figure 38: Complete CSL Stay In Touch State Machine

4.3. Module Description

This section contains a detailed description of the different modules that conform the system. The modules that are to be displayed are not all original, as some are ready-made and directly implemented or modified and adapted from previous designs. All the modules will be labelled as *ORIGINAL*, *MODIFIED* or *INHERITED*. Some of the modules are not described due to their low importance in the system behavior.

All the Block Diagrams showed can be seen in further detail in the *.pdf* files attached to the document.

4.3.1. Global System

The *CSL Stay In Touch* design is organized in functional blocks that separate the various tasks the system performs. These organizational modules are:

- **ClockTree:** Generates various *Clock* signals through the PLL pending from the system Clock (125MHz).
- **MIDIInterface:** Controls the MIDI Input and Output numerical values used as parameters by the other modules.
- **CSLControl:** Controls all the CSL process and the *Stay In Touch* system, as well as the motor driver outputs.
- **VGACanvas:** Controls the canvas displayed in the VGA Screen.
- **GridPaper:** Controls the canvas matrix displayed in the VGA Screen.
- **VGA1024:** Controls the VGA signal and the VGA_X and VGA_Y used by other internal modules (VGA_ADDR).
- **Serial_Com:** Controls the serial stream output of a specific value.

The signal buses simplified in this schematic (*CSL RAW PARAMETERS*, *MODE PARAMETERS*) can be analyzed in further detail in the module interfaces present in the module descriptions.

This simplified schematic shows the dependencies between modules. For a more detailed schematic refer to the *SIT_Global_SCH.pdf* attached to the document.

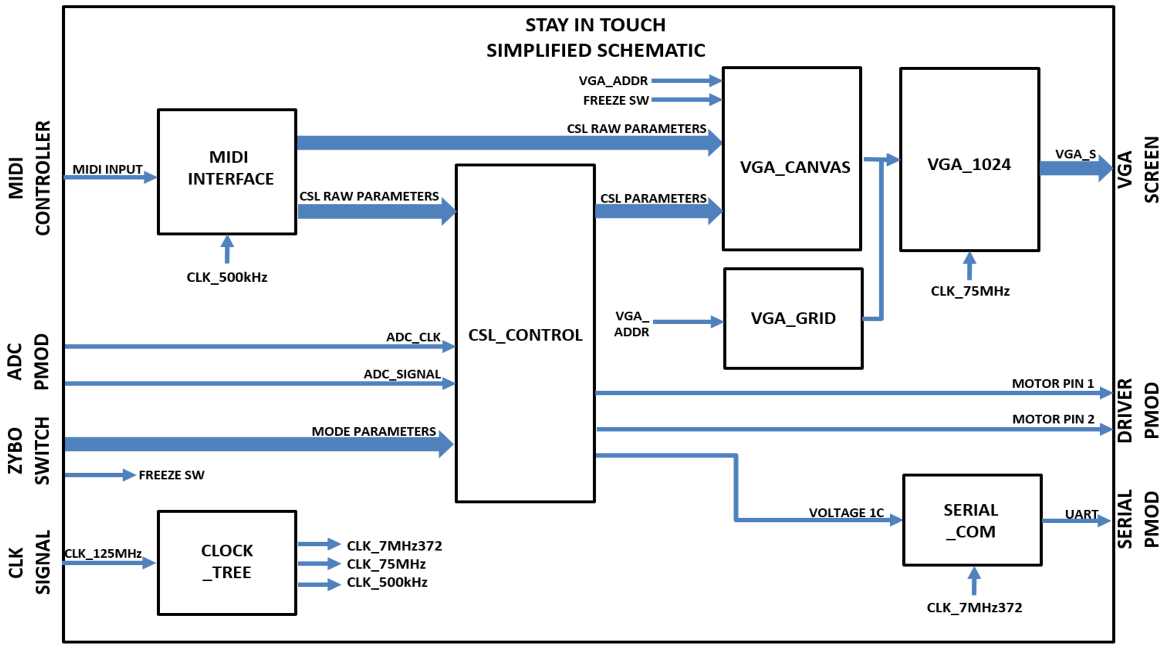


Figure 39: Stay In Touch Main Simplified Schematic

The core of the system is the *CSL_Control* module whose parameters are introduced rather directly from the ZYBO switches or through the *MIDIInterface*. The parameters introduced through the MIDI Interface are then adapted to the system values and outputted to the VGA Canvas for visualization. The ZYBO switches control the different modes (as described in section in the *User Interface* section).

4.3.2. CSL_Control [ORIGINAL]

The *CSL_Control* is built up by several modules. This simplified schematic shows the dependencies between modules. For a more detailed schematic refer to the *SIT_CSLControl_SCH.pdf* attached to the document.

The *CSL_Control* design is organized in functional blocks that separate the various tasks the sub-system performs. These modules are:

- **CSL_StayInTouch:** Controls the main behavior of the system through the SIT (*Stay In Touch*) state machine, the SRC (*Search*) state machine and the DS (*Drive-Sense*) state machine.
- **CSL_Sense:** Controls the ADC motor voltage input during the *Sense* period of the DS state machine.
- **Drift_Corrector:** Generates a drift correction for the *CSL_Sense* module.
- **StandByClock:** Controls a simple 1sec timer for the *Search* mode.

The signal buses simplified in this schematic (*CSL RAW PARAMETERS*, *MODE PARAMETERS*) can be analyzed in further detail in the module interfaces present in the module descriptions.

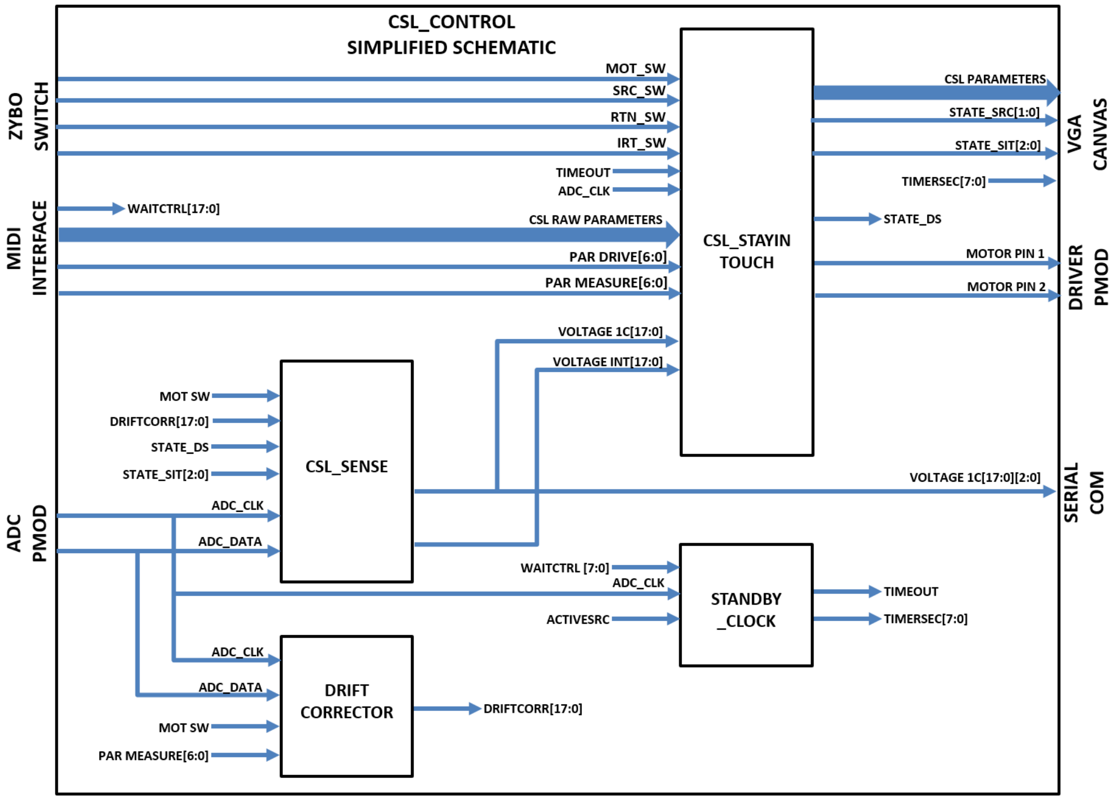


Figure 40: Stay In Touch CSL Control Simplified Schematic

4.3.2.1. CSL_StayInTouch

The *CSL_StayInTouch* module is the most complex of all the system and controls all the state machines as well as the direct motor drive signals. Its interface has the following described signals:

INPUT SIGNALS	OUTPUT SIGNALS
mclk: ADC Pmod Clock signal 10MHz. Behaves as the Clock signal of all the processes. **MOTsw:** Motor enable switch (ON/OFF) **RTNsw:** *Return Mode* switch (ON/OFF) **SRCsw:** *Search Mode* switch (ON/OFF) **IRTsw:** *Inertia Mode* switch (ON/OFF) **ParMeasure[6:0]:** Establishes the *Sense* period duration of the *CSL State Machine* (in ms). **ParDrive[6:0]** Establishes the Sense period duration of the *CSL State Machine* (in ms). **BrakeCtrl[6:0]:** Input RAW value of the *Brake Time* (0-127). **InertiaCtrl[6:0]:** Input RAW value of the *Inertia Time* (0-127). **ThrCtrl_0[6:0]:** Input RAW value of the *Threshold ON* (0-127). **ThrCtrl_1[6:0]:** Input RAW value of *the Threshold OFF* (0-127). **SrcCtrl[6:0]:** Input RAW value of the *Search Time* (0-127). **Voltage[17:0]:** Value of the voltage measurement in one *Sense* cycle **VoltageInt[17:0]:** Value if the voltage integrated measurement. **Timeout:** Timeout flag signal for *Search Mode*.	**IN1**: Motor pin 1 signal. **IN2:** Motor pin 2 signal. **M0:** Configuration Pins for ADC Pmod. **M1:** Configuration Pin for ADC Pmod. **Sleep:** Energy saving configuration pin. **StateDS**: *Drive-Sense* state. **StateSIT[2:0]:** *Stay In Touch* state. **StateSRC[1:0]:** *Search* state **Direction:** Motor rotation direction (A for 0, B for 1). **ThrCtrl_ON[17:0]:** Threshold ON voltage. **ThrCtrl_OFF[17:0]:** Threshold OFF voltage. **BrakeCount[24:0]:** *Brake Time* period (In complete DS cycles). **InertiaCount[24:0]:** *Inertia Time* period (In complete DS cycles). **SrcCount[24:0]:** *Search Time* period (In complete DS cycles.

The different functionalities this module features are going to be described separately even though they are all merged in the same VHDL module.

SIT Parameter dependencies

For the system to perform a similar behavior regardless of the *Sense* and *Drive* parameters, the SIT parameters depend both on the raw value input from the *MIDI Interface* and the DS values.

As the DS values control the state duration, the duration parameters of the SIT modes have to adapt to the new conditions to last the same amount of time as in previous conditions (As their duration is measured in *Drive* and *Sense* complete cycles).

To achieve this depends its adaptation, a simple linear transformation is done to define the SIT final parameters.

```
BrakeCount  <= 11d"0" & std_logic_vector((unsigned(BrakeCtrl)*10) / unsigned(ParMeasure));
InertiaCount  <= 11d"0" & std_logic_vector((unsigned(InertiaCtrl)*10) / unsigned(ParMeasure));
SrcCount <= 11d"0" & std_logic_vector((unsigned(SrcCtrl))*4 / unsigned(ParMeasure));
ThrCtrl_ON  <= std_logic_vector(unsigned(ThrCtrl_0) * unsigned(ParMeasure));
ThrCtrl_OFF <= std_logic_vector(unsigned(ThrCtrl_1) * unsigned(ParMeasure)) & "0000";
```

VHDL Code 1: Stay In Touch Parameter Dependencies

This way, each parameter can be set only once at the beginning of the setting process.

DS (Drive-Sense) State machine

The core *CSL Drive/Sense* state machine performs the main structure in the system as the other state machines are allocated inside the *Sense* state.

It is scheduled through the DS parameters, as the raw input MIDI signals (0-127) are transformed into a timer value in 0,5ms for *Drive* and 1ms for *Sense*. However, to reduce the resources needed, this is done through bit sifting resulting in not an accurate timing (0,512 for *Drive*). The clock used is the cock signal provided by the *ADC PMOD* (*mclk*) whose frequency is of 10MHz in mode 0.

```
SpeedTouch  <= (9d"0") & unsigned(ParDrive)& (9d"0");
timer <= (18d"9999" * unsigned(ParMeasure)) + 25d"9999";
```

VHDL Code 2: Stay In Touch DS State Machine cycle duration

As the *Sense* performance is done in another module (*CSL_Sense*), the state DS signal (*StateDS*) is outputted and therefore the *Drive* state does not add any other functionality than timing through the decrease of the timing signal and establishing the conditions of the Sense cycle.

```
when '1' => -- drive
   timer <= timer - 1;
   if (timer = 25d"0") then -- end of drive phase
      timer <= (18d"9999" * unsigned(ParMeasure)) + 25d"9999";
      if (StateSIT="011" or ((StateSIT="101")and((StateSRC="10")))) then
          IN1 <= '1';
          IN2 <= '1';
      else
          IN1 <= '0';
          IN2 <= '0';
      end if;
      StateDS <= '0';
   end if;
end case;
```

VHDL Code 3: Stay In Touch DS State Machine Drive Cycle

SIT (Stay In Touch) State machine

The SIT state machine is the core of the system and it is mainly allocated inside the DS *Sense* state. It performs the state machine described in the *SIT Basic Behavior* and therefore the code is not included in this document and should be rather studied directly in the VHDL files.

The SIT state signal (*StateSIT*) does not make use of two states for both directions. The direction is marked by a flag signal (*Direction*) which indicates the direction of movement and it is established during the transition from the *Waiting for Touch* state. This way, not only the amount of states required are reduced but it is also simpler to analyze and makes it easier for the *Return* state to establish the direction for the return movement (As the direction is registered until a new *Waiting for Touch – In Touch* transition)

```
Direction    : out std_logic;
```

VHDL Code 4: Stay In Touch Direction Signal Output

```
when "000" => -- WAITING FOR TOUCH
   if (abs(signed(Voltage)) > signed("0" & ThrCtrl_ON(13 downto 0))) then
      StateSIT <= "001"; -- go to IN TOUCH
      if Voltage(17) then
         Direction <= '1'; -- Direction B
       else
         Direction <= '0'; -- Direction A
       end if;
       ReturnFlag <= ('1' and RTNsw); -- Activate ReturnFlag
```

VHDL Code 5: Stay In Touch Direction Flag Setting

```
when "001" => -- IN TOUCH
  IN1 <= not(Direction); -- sign for rot. direction
  IN2 <= Direction;       -- sign for rot. direction
  timer <= SpeedTouch;

  if (abs(signed(Voltage)) >  signed("0" & ThrCtrl_OFF(13 downto 0))) then
     if (Voltage(17) xor Direction) then
        if (IRTsw = '1') then -- IN TOUCH INERTIA enabled
        StateSIT <= "010"; -- go to IN TOUCH INERTIA
        timerInertia <= unsigned(InertiaCount); -- Run timecontrol in ms
     else
        StateSIT <= "011"; -- go to BRAKE
        timerBrake <= unsigned(BrakeCount); -- Pause timecontrol in ms
     end if;
  end if;
end if;
```

VHDL Code 6: Stay In Touch In Touch Direction

The direction is used both to set the values of the motor pin outputs and to validate the trigger of the *Threshold OFF*. Therefore, the *Threshold OFF* can be overpassed but it will continue the transition to *Brake* only if it does it in the right direction.

SRC (Search) State machine

The SRC state machine is allocated inside the *Search SIT* state and controls the *Search* behavior. It has 4 states due to its static routine, so contrary to the In *Touch SIT* state, has one state for each direction (A, B). It also has a *Search Pause* state that is redundant with the *Brake SIT* state but it has been created to simplify the transition between the SIT and SRC states.

Instead of using another bit for the state signal, a flag signal has been used (*SearchIterator*) in order to reduce the code extension and make it more understandable (as it does not save any resources). This flag determinates if the *Search A* state is performed for the first time in the particular *Search* cycle or for second time.

```
when "00" =>-- SEARCH A
  IN1 <= '1'; -- sign for rot. direction
  IN2 <= '0'; -- sign for rot. direction
  timer <= (SpeedTouch/4);
  timerSearch <= timerSearch - 1;

   if ((abs(signed(Voltage)) > signed("000" & ThrCtrl_ON(13 downto 0))) and (Voltage(17)='0'))
then
     StateSIT <= "001"; -- go to IN TOUCH
     Direction <= '0'; -- Direction A
   elsif (timerSearch = 25d"0") then
     if(SearchIterator = '0')then
```

```
         StateSRC <= "10"; -- Go to SEARCH PAUSE
         timerBrake  <= unsigned(BrakeCount);
      else
         StateSIT <= "011"; -- Go to BRAKE
         timerBrake  <= unsigned(BrakeCount);
      end if;
   end if;
```

VHDL Code 7: Stay In Touch Search Mode State A

The initial conditions of the *Search* state are set in the transition from *Waiting for Touch* - *Search* states in the SIT state machine.

```
when "000" => -- WAITING FOR TOUCH
   if (abs(signed(Voltage)) > signed("0" & ThrCtrl_ON(13 downto 0))) then
StateSIT <= "001"; -- go to IN TOUCH
   if Voltage(17) then
      Direction <= '1'; -- Direction B
   else
      Direction <= '0'; -- Direction A
   end if;
ReturnFlag <= ('1' and RTNsw); -- Activate ReturnFlag
elsif((Timeout = '1')and (SRCsw = '1')) then -- Timeout Enabled
   StateSIT <="101"; -- Start SEARCH
   StateSRC <= "00"; -- go to SEARCH A
   SearchIterator <= '0';
   timerSearch <= unsigned(SrcCount)/3;
end if;
   timer <= 25d"0"; -- drive-time NULL
```

VHDL Code 8: Stay In Touch Search Mode Initial Conditions

The drive times of the *Search* states (*SrcCount/3* for *Search A* and *SrcCount/2* for *Search B*) has been established for the sensing platform to return to the original position after the process (if it fails to contact the external object). However, as explained in the *Search Mode Configurations*, this inaccurate method works better with low *Search Time* parameters.

A possible improvement to the *Search* mode could be for the behavior not to perform always the same direction steps (A-B-A) but to start the search in the last *In Touch* state direction has been. This has not been implemented due to the lack of time.

Return Mode State

The *Return* behavior is included as a state from the main SIT state machine. The transition to this state is dependent on the *ZYBO* switch signal (*RTNsw*) that triggers a flag (*ReturnFlag*) that is set in the *Waiting for Touch* state.

```
when "000" => -- WAITING FOR TOUCH
   if (abs(signed(Voltage)) > signed("0" & ThrCtrl_ON(13 downto 0))) then
StateSIT <= "001"; -- go to IN TOUCH
   if Voltage(17) then
      Direction <= '1'; -- Direction B
   else
      Direction <= '0'; -- Direction A
   end if;
ReturnFlag <= ('1' and RTNsw); -- Activate ReturnFlag
elsif((Timeout = '1')and (SRCsw = '1')) then -- Timeout Enabled
   StateSIT <="101"; -- Start SEARCH
   StateSRC <= "00"; -- go to SEARCH A
   SearchIterator <= '0';
   timerSearch <= unsigned(SrcCount)/3;
end if;
   timer <= 25d"0"; -- drive-time NULL
```

VHDL Code 9: Stay In Touch Return Mode Waiting For Touch

```
when "011" => -- BRAKE
    IN1 <= '1'; -- sign for rot. direction
    IN2 <= '1'; -- sign for rot. direction
    timer <=  25d"0"; -- drive-time NULL
    timerBrake <= timerBrake - 1; --drive-Time Decrement
    if (timerBrake = 25d"0") then
```

```
         if (ReturnFlag = '1') then
              StateSIT <= "100";  -- go to RETURN
              timerReturn <= 25d"1";
          else
              StateSIT <= "000";  -- go to WAITING FOR TOUCH
              timerBrake <= 25d"1";
          end if;
       end if;
```

VHDL Code 10: Stay In Touch Return Mode Brake

The *Return* mode itself makes use of the direction flag (*Direction*) to set the returning movement direction. The *Return* movement is applied only during one DS cycle, as this has been tested on to perform the best in returning to the position in which the sensing platform loses contact with the external object.

```
when "100" => -- RETURN
   IN1 <= Direction;      -- sign for rot. direction
   IN2 <= not(Direction); -- sign for rot. direction
   timer <= SpeedTouch;

   timerReturn <= timerReturn - 1;
   if (timerReturn = 25d"0") then
      StateSIT <= "011"; -- go to BRAKE
      timerBrake <= unsigned(BrakeCount); -- Pause timecontrol in ms
      ReturnFlag <= '0'; -- Disable ReturnFlag
   end if;
```

VHDL Code 11: Stay In Touch Return Mode State

Motor Drive Time

The motor *Drive* time is constant and does not rely on the *Sense* values (therefore these are only involved in the state transition triggering). The drive is constant regardless of the resistance to the movement as the objective of the system is to keep in touch and not to work against the external object.

Brake Mechanism

In order to properly brake the motor to reduce the time it needs to stop, both motor pins are set to a HIGH level to shortcut the motor. If the SIT states are only active during the *Drive DS* state, the brake is applied interrupted by idle motor state. To solve this issue, during the *Drive* - *Sense* transition, an exception is made for the *Brake SIT* state and the *Pause SRC* state.

```
when '1' => -- drive
   timer <= timer - 1;
   if (timer = 25d"0") then -- end of drive phase
      timer <= (18d"9999" * unsigned(ParMeasure)) + 25d"9999";
      if (StateSIT="011" or ((StateSIT="101")and((StateSRC="10")))) then
          IN1 <= '1';
          IN2 <= '1';
      else
          IN1 <= '0';
          IN2 <= '0';
      end if;
      StateDS <= '0';
   end if;
end case;
```

VHDL Code 12: Stay In Touch Brake

If not, the motor pin outputs are set to LOW (idle) not to interfere the movement of the motor.

4.3.2.2. CSL Sense

The *CSL_Sense* module performs the register of the incoming measurements from the *ADC Pmod*. It is separated but dependent from the CSL DS state machine to make the system more functional and in order to reuse this module in other systems.

INPUT SIGNALS	OUTPUT SIGNALS
mclk: ADC Pmod Clock signal 10MHz. Behaves as the Clock signal of all the processes. **mdat:** ADC Pmod Data signal. **StateDS**: *Drive-Sense* state. **StateSIT[2:0]:** *Stay In Touch* state. **DriftCorr[17:0]:** ADC Drift correction value.	**VoltageOut[17:0]:** Voltage measure of one *Sense* cycle (Often referred in this document as Voltage 1C). **VoltageIntOut[17:0]:** Voltage Integration value.

The dependencies regarding the DS state machine consist in the timing of the measure period. This measurement is only done during the *Sense* cycle and it is commanded externally, not having the *CSL_Sense* module to control any state machine or timing parameters and simplifying the system.

The module has security limits in both polarities of the Voltage signal (As it is a signed C2 value) to avoid overflow. The values of *Voltage* and *VoltageInt* are outputted only once the Sense cycle is finished due to the need for the drift correction. The *VoltageInt* signal is set to 0 during the *Waiting for Touch* state to avoid a low derivation. In any case, in this system the voltage integration is not used and therefore not needed, but it has been included in order to reuse this module.

```
ADCMeasure: process(mclk) begin
if mclk'event and mclk = '1' then
   if StateDS='0' then -- Sense
      if mdat then
         voltage <= voltage + 1; -- Immediate voltage cant saturate
         if ( voltageInt < 18d"131071") then     -- prohibit positive saturation
           voltageInt <= voltageInt + 1;
         end if;
       else
         voltage <= voltage - 1; -- Immediate voltage cant saturate
         if ( voltageInt > "1"&16d"0"&"1") then   -- prohibit negative saturation
           voltageInt <= voltageInt - 1;
         end if;
       end if;
    end if;

    if (not(Q(1)) and Q(0)) then -- End of Sense
       VoltageOut <= std_logic_vector((voltage)- signed(DriftCorr));
       voltage <= 18d"0"; -- Immediate voltage reset
       if (StateSIT="000") then
         voltageInt <= 18d"0"; -- VoltageInt reset during the WAITING FOR TOUCH state
       elsif ((voltageInt(17)='1') or ((voltageInt < (18d"131070" - signed(DriftCorr))))) then
          voltageInt <= voltageInt - signed(DriftCorr); -- Drift Correction
       end if;
    end if;
end if;
end process;
```

VHDL Code 13: Stay In Touch CSL Sense Main

The detection of the end of the *Sense* cycle is done through a pulse conformer that detects the end of the *Sense* cycle.

```
-- Pulseconformer todetect the end of the SENSE StateDS period
Conformer: process(mclk) begin
    if mclk'event and mclk = '1' then
        Q(1) <= Q(0);
        Q(0) <= StateDS;
    end if;
end process;
```

VHDL Code 14: Stay In Touch CSL Sense Conformer

4.3.2.3. Drift Corrector

The *Drift_Corrector* module registers the ADC Pmod voltage drift error and adapts it dynamically for the *CSL_Sense* module to correct its measures.

INPUT SIGNALS	OUTPUT SIGNALS
mclk: ADC Pmod Clock signal 10MHz. Behaves as the Clock signal of all the processes. **mdat:** ADC Pmod Data signal. **SWactive**: Motor switch. **ParMeasure[6:0]:** Establishes the *Sense* period duration of the *CSL State Machine* (in ms).	**DriftCorr[17:0]:** ADC Drift correction value.

The mechanism to register the voltage drift is to make a 300ms measurement repeatedly when the habilitation is enabled (*SWactive*) and register the last value for the 300ms window. As the measurements are proportional to the *Sense Time*, through this 300ms measurement window (highly above the 12,7ms MAX *Sense Time*), the error voltage measurement values can be calculated for every *Sense Time* (*ParMeasure*). This is done through a simple proportional operation:

$$DriftCorrection = \frac{Measure300ms * ParMeasure}{300ms}$$

```
-- Drift Value dinaic adaptation depending on the ParMeasure value
DriftCorr_27 <= std_logic_vector((driftCycle * (signed("0" & ParMeasure)+1))/(signed("0" &
measureTime)/9999));
DriftCorr <= DriftCorr_27(17 downto 0);
```

VHDL Code 15: Stay In Touch Drift Correction

The *DriftCorr_27* is an auxiliary signal needed for the operations in VHDL due to the potential overflow of the signal during the operation.

The *measureTime* (originaly 300ms) can be set through a generic parameter.

```
Generic ( measureTime : unsigned(24 downto 0) := 25d"2999700"); -- Generic value for 300ms
```

VHDL Code 16: Stay In Touch Drift Correction Parameter

The motor has to be stopped when the measurements are done (to measure the drift error and not a motion voltage), for this reason the *SWactive* signal is connected to the *MTRsw*. The switch has to be turned OFF at some point every system restart. This does not guarantee that the motor is relaxed during the period but disables the control over its movement (IN1 = 0, IN2 = 0).

4.3.2.4. StadnByClock

The *StandByClock* is a simple module that controls a 1 second timeout clock to control the trigger of the *Search* mode if habilitated.

INPUT SIGNALS	OUTPUT SIGNALS
mclk: ADC Pmod Clock signal 10MHz. Behaves as the Clock signal of all the processes. **WaitCtrl[6:0]:** MIDI Input that controls the *Wait Time* in seconds (0-127) **Active**: Habilitation signal for the timer reset	**TimerSec[7:0]:** Time remaining for next *Timeout* (in sec) **Timeout:** End of countdown signal

The measurement is easily done due to the proportional 10MHz (*mclk*) clock signal. The habilitation signal that resets the timer (*Active*) is controlled directly in the *CSLControl* structural module during the *Waiting for Touch SIT* state.

```
StandBy <= '1' when StateSIT = "000" else '0';
```

VHDL Code 17: Stay In Touch StandByClock Reset

As in previous modules, the *TimerSec* (only outputted for monitoring purposes) is calculated through an auxiliary signal (*TimerSecVector*).

```
architecture Behavioral of StandByClock is
   signal timerStart : unsigned(31 downto 0) := unsigned(WaitCtrl) * 25d"9999999";
   signal timer : unsigned(31 downto 0) := 32d"0";
   signal timerSecVector : unsigned(31 downto 0) := 32d"0";
begin

DriftMeasure: process(mclk) begin
   if mclk'event and mclk = '1' then
       if Active = '1' then
          if(timer > 32d"0") then
             timer <= timer -1;
          end if;
       else
          timer <= timerStart;
       end if;
   end if;
end process;

timerSecVector <= timer / 25d"9999999";
TimerSec <= std_logic_vector(timerSecVector(7 downto 0));
Timeout <= '1' when timer = 0 else '0';
```

VHDL Code 18: Stay In Touch StandByClock Frequency Divider

4.3.3. ClockTree [INHERITED]

The *ClockTree* module generates clock signals used by the other modules. All the clock signals are derived from the 125MHz Input Clock Signal from the *Ethernet PHY*. These new signal are generated rather through the PLL or through counters (frequency dividers). The clock signals generated are:

- 75 MHz (Used by the VGA Interface)
- 12.288 MHz
- 7.372 MHz (Used by the Serial Interface)
- 3.072 MHz
- 500 kHz (Used by the MIDI Interface)
- 250 kHz
- 48 kHz

The use of standard *std_signals* for clock signals is not recommended in a general basis due to the existence of Clock Buses in the FPGAs, however in this system this has not been taken into account.

4.3.4. VGA_Canvas [ORIGINAL]

This section contains all the information about the VHDL modules developed for monitoring and graphic purposes. Therefore, these modules do not perform critical tasks in the system but provide tools for the user to control and analyze the different parameters involved.

The *VGA_Canvas* is built up by several modules. This simplified schematic shows the dependencies between modules. For a more detailed schematic refer to the *SIT_VGACanvas_SCH.pdf* attached to the document. All the modules made with double low bar represent a group of similar modules rather than just one.

The *VGA_Canvas* design is organized in functional blocks that separate the various tasks the sub-system performs. These modules are:

- **ASCII_Canvas:** Organizational module that contains all the ASCII individual modules.
- **ShowVBar**: Graphic bar for the display of 1 to 127 values.
- **WriteBCD:** Graphic display of up to 6 digits of a binary value in BCD.
- **WriteSigned:** Graphic display of up to 5 digits of a signed binary (C2) in BCD.
- **ShowScope:** Graphic display of a signal in a time chart.
- **DrawState:** Graphic color display of the SIT states.

The signal buses simplified in this schematic (*CSL PARAMETERS*, CSL *MODES*) can be analyzed in further detail in the module interfaces present in the module descriptions.

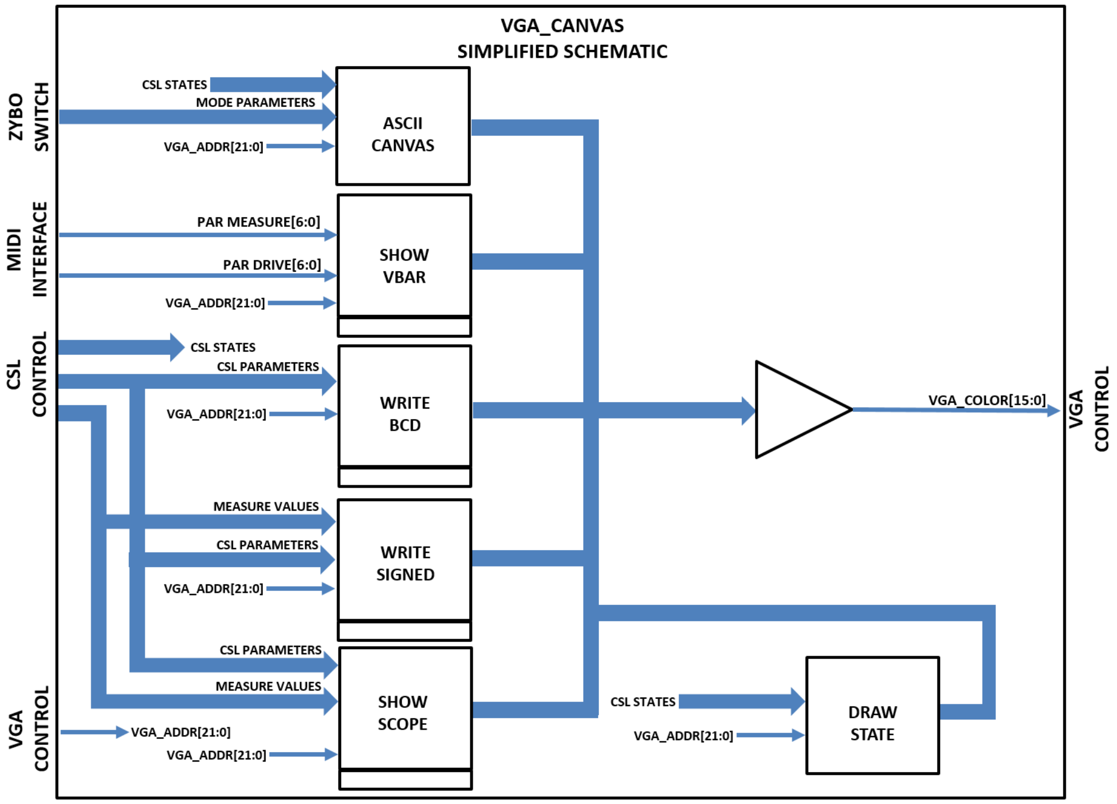


Figure 41: Stay In Touch VGA Canvas Simplified Schematic

4.3.4.1. ASCII_Canvas [ORIGINAL]

The *ASCII_Canvas* contains all the ASCII characters of the screen display through the *ASCII_sign* module.

INPUT SIGNALS	OUTPUT SIGNALS
VGA_Addr[21:0]: VGA current X and Y address **StateSIT[2:0]:** Stay In Touch state. **StateSRC[1:0]:** Search state. **MOTsw:** Motor enable switch (ON/OFF) **RTNsw:** *Return Mode* switch (ON/OFF) **SRCsw:** *Search Mode* switch (ON/OFF) **IRTsw:** *Inertia Mode* switch (ON/OFF) **DDir:** Motor rotation direction (A for 0, B for 1).	**Color_out [15:0]:** Pixel color output.

ASCII_Sign implementation

The *ASCII_Sign* module places a ASCII character of a fixed dimension in a fixed position set in pixels. This is done through an index matrix in which all the characters are graphically divided in pixel rows.

```
-- Word "BHT NRL"
BN_s0: entity work.ASCII_sign  generic map ( XPOS => 8, YPOS => 9)
                                  port map ( VGA_Addr, 7d"66" ,BN_0);
BN_s1: entity work.ASCII_sign  generic map ( XPOS => 9, YPOS => 9)
                                  port map ( VGA_Addr, 7d"72" ,BN_1);
BN_s2: entity work.ASCII_sign  generic map ( XPOS => 10, YPOS => 9)
                                  port map ( VGA_Addr, 7d"84" ,BN_2);
BN_s3: entity work.ASCII_sign  generic map ( XPOS => 12, YPOS => 9)
                                  port map ( VGA_Addr, 7d"78" ,BN_3);
BN_s4: entity work.ASCII_sign  generic map ( XPOS => 13, YPOS => 9)
                                  port map ( VGA_Addr, 7d"82" ,BN_4);
BN_s5: entity work.ASCII_sign  generic map ( XPOS => 14, YPOS => 9)
```

```
port map ( VGA_Addr, 7d"76" ,BN_5);
```

VHDL Code 19: Saty In Touch ASCII_sign organization

This system, based on *Hex_sign*, is efficient in terms of resources once implemented but extremely costly to design due to the needed previous planning and the need to add each letter separately.

ASCII_Sign output signal merge

To merge all the pixel activation outputs in a single color signal (*Color_out*) a *OR* structure is needed. However, a simple *OR* structure would be literally interpreted by the implementation of the development software (*Vivado*) resulting in a very inefficient circuit.

```
Color_out <= Color when ((((MOT1_0 or MOT1_1) or (MOT1_2 or MOT1_3)) or ((MOT1_4 or MOT1_5) or
                        (BN_0 or BN_1))) or (((BN_2 or BN_3) or (BN_4 or BN_5)))) or
                        (((DRV_0 or DRV_1) or (DRV_2 or DRV_3)) or DRV_4) or -- "DRIVE"
                        (((SNS_0 or SNS_1) or (SNS_2 or SNS_3)) or SNS_4) or -- "SENSE"
                        ((((SIT_0 or SIT_1) or (SIT_2 or SIT_3)) or ((SIT_4 or SIT_5) or
                        (SIT_6 or SIT_7))) or (((SIT_8 or SIT_9) or (SIT_10)))
```

VHDL Code 20: Stay In Touch ASCII Canvas Signal OR Merge

To avoid this situation, the elements have been grouped in coupled *OR* structures. The result is a much simpler schematic that ensures a better timing performance. A great delay in this module could generate interface errors in the VGA Screen display, showing old values instead of the new ones (Even though this is not relevant due to the great speed of the FPGA compared to the VGA refresh rate and the human eye perception).

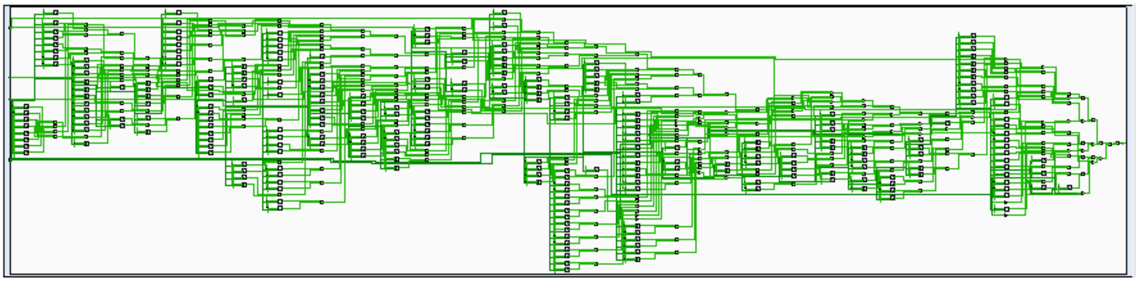


Figure 42: Stay In Touch ASCII Canvas Schematic

This schematic is not meant to be read but only to show the vast complexity of this module and the amount of simple *OR* or *AND* operators.

ASCII_Sign Placement

The ASCII_Sign placement of the ASCII character is done through static generic parameters.

```
entity ASCII_sign is
    generic (XPOS, YPOS    : integer := 0);
    Port    (VGA_Addr       : in std_logic_vector(21 downto 0);
             ASCII          : in std_logic_vector(6 downto 0);
             Pixel          : out STD_LOGIC);
end ASCII_sign;
```

VHDL Code 21: Stay In Touch ASCII_Sign Generic Parameters

ASCII_Sign Character chose

The *ASCII_Sign* index matrix does not cover all the ASCII characters as some especial ones (such as the STR, ACK or other system characters are not included). Nevertheless, the ASCII numeration remains the same through a simple subtraction.

```
ASCII_cor <= std_logic_vector(unsigned(ASCII)-32); -- (Subtraction correction for the missing
31 first values)
```

VHDL Code 22: Stay In Touch ASCII_Sign Missing Character Avoidance

This makes the implementation easier as it fits the standardized ASCII values.

4.3.4.2. WriteBCD and WriteSigned [ORIGINAL]

The *WriteBCD* and *WriteSigned* are two twin modules very similar in performance. They both transform and display a binary value into a BCD value displayed in the VGA Screen.

INPUT SIGNALS	OUTPUT SIGNALS
VGA_Addr[21:0]: VGA current X and Y address **Bin_in[18:0]:** Input binary value	**Color_out [15:0]:** Pixel color output.

The 19 bit input value was selected by Benjamin Panreck in order to fit the voltage values handled.

BIN to BCD Conversion

To convert the binary values into BCD digits, a *double dabble* algorithm was implemented through a state machine.

```
Iterator: process(CLK_75MHz) begin
   if CLK_75MHz'event and CLK_75MHz = '1' then
    if(Iteration < 19) then -- COMPARE&ADD3 SEQUENCE  UNIT 10^0
        if ((BCD(3 downto 0) > "0100")and(OP0 = '0')) then
            BCD(3 downto 0) <= (BCD(3 downto 0) + 3);
            OP0 <= '1';
        elsif ((BCD(7 downto 4) > "0100")and(OP1 = '0')) then -- DIGIT 10^1
            BCD(7 downto 4) <= (BCD(7 downto 4) + 3);
            OP1 <= '1';
        elsif ((BCD(11 downto 8) > "0100")and(OP2 = '0')) then -- DIGIT 10^2
            BCD(11 downto 8) <= (BCD(11 downto 8) + 3);
            OP2 <= '1';
        elsif ((BCD(15 downto 12) > "0100")and(OP3 = '0')) then -- DIGIT 10^3
            BCD(15 downto 12) <= (BCD(15 downto 12) + 3);
            OP3 <= '1';
        elsif ((BCD(19 downto 16) > "0100")and(OP4 = '0')) then -- DIGIT 10^4
            BCD(19 downto 16) <= (BCD(19 downto 16) + 3);
            OP4 <= '1';
        elsif ((BCD(23 downto 20) > "0100")and(OP5 = '0')) then -- DIGIT 10^5
            BCD(23 downto 20) <= (BCD(23 downto 20) + 3);
            OP5 <= '1';
        else
            Iteration <= Iteration + 1;
            BCD(23 downto 0) <= (BCD(22 downto 0) & BIN(18)); -- SHIFT CONTROL
            BIN(18 downto 0) <= (BIN(17 downto 0) & '0');
            OP0 <= '0'; -- OPX SYNCHRONOUS RESET
            OP1 <= '0';
            OP2 <= '0';
            OP3 <= '0';
            OP4 <= '0';
            OP5 <= '0';
        end if;
     else
        Iteration <= (others => '0');
        BIN <= BIN_in;
        BCD_out <= (STD_LOGIC_VECTOR(BCD));
        BCD <= (others => '0');
     end if;
   end if;
   end process;
```

The conversion is tested and proved to work correctly. The main problem of the *double dabble* algorithm is that the duration is not fixed and therefore may change depending on the input values. However this does not affect the system as it is used for the much slower VGA Screen.

WriteBCD Digit Display

The *WriteBCD* allows the designer to choose the amount of digits to be displayed through generic parameters.

```
entity WriteBCD is
    generic (  PosX     : INTEGER := 0
               PosY     : INTEGER := 0;
```

```
                Color   : STD_LOGIC_VECTOR(15 downto 0) := 16d"1";
                Digit   : UNSIGNED(2 downto 0) := 3d"6");
     Port ( BIN_in         : in STD_LOGIC_VECTOR(18 downto 0);
            VGA_Addr       : in STD_LOGIC_VECTOR(21 downto 0);
            Color_out      : out STD_LOGIC_VECTOR(15 downto 0));
end WriteBCD;
```

VHDL Code 23: Stay In Touch WriteBCD Generic Parameters

Then, these are multiplexed in the output signals.

```
Color_out <= Color when (((Hex_0='1')and(Digit>0)) or
                         ((Hex_1='1')and(Digit>1)) or
                         ((Hex_2='1')and(Digit>2)) or
                         ((Hex_3='1')and(Digit>3)) or
                         ((Hex_4='1')and(Digit>4)) or
                         ((Hex_5='1')and(Digit>5)))
                   else(16d"0");
```

VHDL Code 24: Stay In Touch WriteBCD Digit Output

Even if not displayed, the digits are calculated.

WriteSigned Characteristics

The *WriteSigned* makes use of the *WriteBCD* but previously it reverses the signed C2 binary if needed, as well as placing a ASCII sign (+ or -) in the VGA Screen.

```
WriteBCD_3:   entity work.WriteBCD  generic map ( PosX => PosX, PosY => PosY, Digit => "110")
                                    port map    ((2d"0" & BIN(16 downto 0)), VGA_Addr, BCD);
Sign_Pos:  entity work.ASCII_sign  generic map ( XPOS => (PosX-6), YPOS =>  PosY)
                                   port map(VGA_Addr, 7d"43" ,POS);
Sign_Neg:  entity work.ASCII_sign  generic map ( XPOS => (PosX-6), YPOS =>  PosY)
                                   port map(VGA_Addr, 7d"45" ,NEG);

BIN <= BIN_in when not(BIN_in(17)) else
       std_logic_vector(unsigned(not(BIN_in))+1);

Color_Sign <= Color when (NEG and (BIN_in(17))) or (POS and (not(BIN_in(17))))
              else (16d"0");
```

VHDL Code 25: Stay In Touch WriteSigned

Contrary to the *WriteBCD*, the *WriteSigned* module does not allow the designer to choose the number of digits displayed.

4.3.4.3. ShowScope [MODIFIED]

The *ShowScope* is a module designed to display a time Chart of a signal.

INPUT SIGNALS	OUTPUT SIGNALS
VGA_Addr[21:0]: VGA current X and Y address **Value[6:0]:** Input binary value **StopSW:** *Freeze Mode* switch (ON/OFF)	**VGA_Color[15:0]:** Pixel color output.

This module has been inherited from previous designs and modified to allow a freeze of the chart. This has been done by creating a mirrored signal that remains not displayed until the freeze option is activated. It also displays a small red square as a Freeze signal. The freeze signal is controlled by a conformed *ZYBO* switch.

```
-- Drawing of the REAL TIME waveform
if  VGA_X >= std_logic_vector(PosX) and VGA_X < std_logic_vector(PosX + Width) and
VGA_Y =  std_logic_vector(PosY - unsigned(WAVE(Width - to_integer(unsigned(VGA_X) - PosX))))
and (STOP = '0')
  then VGA_Color <= Color;
-- Drawing of the FREEZED waveform
elsif VGA_X >= std_logic_vector(PosX) and VGA_X < std_logic_vector(PosX + Width) and
  VGA_Y =  std_logic_vector(PosY - unsigned(FREEZ(Width - to_integer(unsigned(VGA_X) -
PosX)))) and (STOP = '1') then
  VGA_Color <= Color;
-- Drawing of the STOP signal
```

```
elsif VGA_X >= std_logic_vector(PosX + 2) and VGA_X < std_logic_vector(PosX + 12) and
VGA_Y >= std_logic_vector(PosY - 10) and VGA_Y < std_logic_vector(PosY) and (STOP = '1') then
   VGA_Color <= "1111100000000000";
-- Background Color
else
   VGA_Color <= 16d"0";
         end if;
```

VHDL Code 26: Stay In Touch ShowScope Freeze Mechanism

This solution is very costly in terms of FPGA resources because the RAM signal is very heavy as it makes use of 1024 8bit signals.

```
architecture Behavioral of ShowScope is
    signal Counter : unsigned(Speed downto 0);
    type RAM is array (Width downto 0) of std_logic_vector(7 downto 0);
    signal WAVE         : RAM;
    signal FREEZ        : RAM;
    signal Clk_75MHz    : std_logic;
    signal VGA_X        : std_logic_vector(10 downto 0);
    signal VGA_Y        : std_logic_vector(9 downto 0);
    signal Q0           : std_logic_vector(1 downto 0);        -- 2bit Shift Register
    signal STOP         : std_logic;
```

VHDL Code 27: Stay In Touch ShowScope RAM Signal

The signal displayed is only 7 bit wide, for this reason, a proper section of the original signal (18 bits) is selected in *VGA_Canvas*.

```
VoltageScope <= std_logic_vector(7d"64" + unsigned(Voltage(16 downto 10)));
ThresholdON_0 <= std_logic_vector(7d"64" + unsigned("0" & ThrCtrl_ON(13 downto 8)));
ThresholdON_1 <= std_logic_vector(7d"64" - unsigned("0" & ThrCtrl_ON(13 downto 8)));
ThresholdOFF_0 <= std_logic_vector(7d"64" + unsigned(ThrCtrl_OFF(16 downto 10)));
ThresholdOFF_1 <= std_logic_vector(7d"64" - unsigned(ThrCtrl_OFF(16 downto 10)));
```

VHDL Code 28: Stay In Touch VGA Canvas Scope Signal

4.3.4.4. ShowScopeThreshold [ORIGINAL]

The *ShowScopeThreshold* displays the thresholds of the system in the chart.

INPUT SIGNALS	OUTPUT SIGNALS
VGA_Addr[21:0]: VGA current X and Y address **Value0[6:0]:** Input binary value 0 **Value1[6:0]:** Input binary value 1 **StopSW:** *Freeze Mode* switch (ON/OFF)	**VGA_Color[15:0]:** Pixel color output.

Contrary to *ShowScope*, it is very little resource spending as it does not use a RAM signal but a single static value (Same value in every point of the chart, not time sensible).

```
-- Drawing of the REAL TIME waveform
if  ((VGA_X >= std_logic_vector(PosX) and VGA_X < std_logic_vector(PosX + Width)) and
((VGA_Y =  std_logic_vector(PosY - unsigned(Value1))))) then
   VGA_Color <= Color;
elsif  ((VGA_X >= std_logic_vector(PosX) and VGA_X < std_logic_vector(PosX + Width)) and
((VGA_Y =  std_logic_vector(PosY - unsigned(Value0))))) then
   VGA_Color <= Color;
-- Background Color
else
   VGA_Color <= 16d"0";
end if;
```

VHDL Code 29: Stay In Touch Scope Threshold

4.3.4.5. DrawState[ORIGINAL]

The *DrawState* displays a big color square in the VGA Screen that indicate the SIT state.

INPUT SIGNALS	OUTPUT SIGNALS
VGA_Addr[21:0]: VGA current X and Y address **StateSIT[2:0]:** *Stay In Touch* state.	**VGA_Color[15:0]:** Pixel color output.

The colors and dimensions of the square can be set through static generic parameters.

```
entity DrawState is
    generic ( PosX : integer := 0; -- 0 <= x <= 63
              PosY : integer := 0; -- 0 <= y <= 5
              Color_0 : std_logic_vector(15 downto 0) := 5d"0" & 6d"15" & 5d"5";
              Color_1 : std_logic_vector(15 downto 0) := 5d"0" & 6d"0" & 5d"20";
              Color_2 : std_logic_vector(15 downto 0) := 5d"20" & 6d"20" & 5d"20";
              Color_3 : std_logic_vector(15 downto 0) := 5d"20" & 6d"0" & 5d"0";
              Color_4 : std_logic_vector(15 downto 0) := 5d"10" & 6d"10" & 5d"0";
              Color_5 : std_logic_vector(15 downto 0) := 5d"0" & 6d"30" & 5d"30";
              BackColor : std_logic_vector(15 downto 0) := 16d"0";
              DimX     : integer := 384;
              DimY     : integer := 384);                -- transparent
    port (  VGA_Addr : in std_logic_vector(21 downto 0);
            VGA_Color : out std_logic_vector(15 downto 0);
            StateSIT : in std_logic_vector(2 downto 0) );
end DrawState;
```

VHDL Code 30: Stay In Touch DrawState Generic Parameters

4.3.5. Serial_Com [MODIFIED]

The *Serial_Com* is a structural module for the UART module created by Benjamin Panreck that implements a serial communication with the computer.

INPUT SIGNALS	OUTPUT SIGNALS
Clk125MHz: System Clock signal. **Clk7MHz372:** Clock signal for the UART communication. **ONsw:** *Freeze Mode* switch (ON/OFF) **DataIN[18:0]:** Data input stream	**Txd:** data output stream.

The *ONsw* allows the designer to make the transfer dependent on a particular condition. In this system it is always set to HIGH. The *Serial_Com* makes the UART work at maximum speed by ordering a new transfer immediately after the previous has finished.

```
NewVal <= '1' when (Active='0'and ONsw='1') else '0';
```

VHDL Code 31: Stay In Touch Serial Communication

UART [MODIFIED]

The *UART* module implements the UART output stream for the USB Pmod with 115200 Baudrate. It sends ASCII characters and therefore it uses a *double dabble* algorithm similar to the one used for *WriteSigned*.

The modification made to this module is that it has been for it to allow the transfer of signed BCD to directly export the data into a *.txt* file through *Realterm* software and then edit and plot a graphic with *Matlab*.

```
if position = 0 then – First Byte to send
   if(Data(17)='1')then
     byte <= 8d"45"; -- Sign -
   else
     byte <= 8d"43"; -- Sign +
   end if;
elsif position < 6 then
```

VHDL Code 32: Stay In Touch UART Signed

4.4. System Analysis

This section contains the analysis of the system regarding the reports generated by *Vivado* on the synthesized design. Two systems will be analyzed: the complete *CSL Stay In Touch* system studied in this document and then another system without the graphic interface (*CSL Stay In Touch Light*).

4.4.1. Complete Stay In Touch Analysis

4.4.1.1. Global Resource Use

This sections shows the FPGA resources used by the system in a percentage of the total capability possible.

Name	Slice LUTs	Slice Registers	F7 Muxes	Slice	LUT as Logic	LUT Flip Flop Pairs	Block RAM Tile	DSPs
testbench_1_bpa	41.09 %	35.08 %	0.63 %	82.06 %	41.09 %	71.61 %	41.66 %	3.75 %
ClockTree	0.02 %	0.01 %	0.00 %	0.04 %	0.02 %	0.02 %	0.00 %	0.00 %
CSLControl	7.02 %	0.75 %	0.01 %	9.40 %	7.02 %	7.02 %	0.00 %	3.75 %
CSL_Sense	1.53 %	0.15 %	0.00 %	2.02 %	1.53 %	1.53 %	0.00 %	0.00 %
CSL_StayInTouch	1.71 %	0.38 %	0.01 %	2.56 %	1.71 %	1.72 %	0.00 %	1.25 %
Drift_Corrector	2.50 %	0.12 %	0.00 %	3.61 %	2.50 %	2.50 %	0.00 %	1.25 %
StandByClock	1.27 %	0.09 %	0.00 %	1.84 %	1.27 %	1.46 %	0.00 %	1.25 %
MIDIInterface	2.33 %	0.44 %	0.00 %	4.31 %	2.33 %	2.82 %	0.00 %	0.00 %
Serial_Com	1.26 %	0.44 %	0.00 %	1.63 %	1.26 %	1.42 %	0.00 %	0.00 %
VGA1024	2.48 %	0.36 %	0.03 %	4.61 %	2.48 %	2.63 %	0.00 %	0.00 %
VGACanvas	27.36 %	33.05 %	0.59 %	68.97 %	27.36 %	58.63 %	41.66 %	0.00 %

Table 1: Stay In Touch Complete System Resources

The parameters considered in this analysis are:

- **Slice LUTs:** Group unit of LUTs.
- **Slice Registers:** Group unit of registers (Group of Flip Flops).
- **Muxes:** Group unit of multiplexers.
- **LUT (Look Up Table):** Basic unit of a FPGA in which an output value can be set through its input values. It is programmed through a RAM indexed blocks.
- **Block RAM Tile:** RAM memory of the FPGA.

The *CSL Stay In Touch* system makes use of up to 41% of the FPGA LUT Slices (Main functional module of a FPGA). However, the system core (*CSLControl*, *ClockTree* and *MIDIInterface*) uses less than 10% of the FPGA resources making it possible to implement the system in much less powerful hardware or a CLP.

The *VGACanvas* is the only module in the whole system to make use of the *RAM Tile* through the *ShowScope* module that needs up to 1024bytes for the displayed signal (depending on the width of the Scope).

4.4.1.2. Power Analysis

This section makes a prediction on the system power consumption of the FPGA (Not regarding the other elements such as the *Pmods – DAC ADC Pmod* and *Serial Pmod* are supplied externally – or the MIDI Keyboard and the VGA Screen).

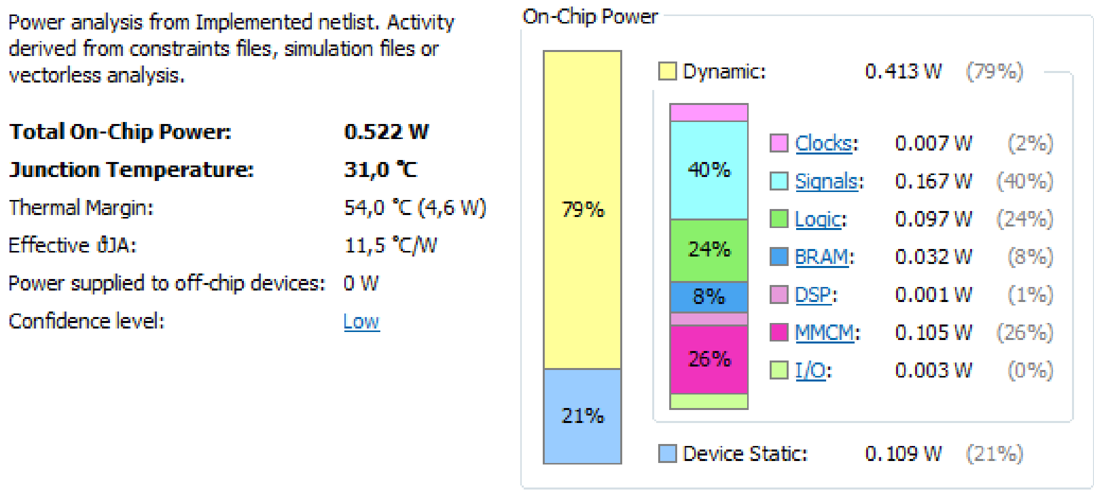


Figure 43: Stay In Touch System Power Consumption

As seen, the system is not very power demanding. Its consumption, however, depends on the specific performance, as the static device consumption is of 0,1W while the unpredictable maximum dynamic consumption is of 0,413W.

With the configurations tried (8V from the power supply to the *DAC ADC Pmod*), the power consumption of the motor is of 0,48W.

4.4.2. Light Stay In Touch Analysis

This analysis aims to give an idea of the real *CSL Stay In Touch* system once implemented in a bigger system, as it lacks all the serial and graphic interface. Also, by comparison with the complete system, it gives an idea of the amount of resources dedicated to monitoring.

4.4.2.1. Global Resource Use

This sections shows the FPGA resources used by the system in a percentage of the total capability possible.

Name	Slice LUTs	Slice Registers	Slice	LUT as Logic	LUT Flip Flop Pairs	DSPs	Bonded IOB	BUFGCTRL
testbench_1_bpa	7.31 %	1.16 %	9.54 %	7.31 %	7.52 %	3.75 %	17.00 %	6.25 %
ClockTree	0.02 %	0.01 %	0.04 %	0.02 %	0.02 %	0.00 %	0.00 %	3.12 %
CSLControl	4.46 %	0.70 %	5.88 %	4.46 %	4.47 %	3.75 %	0.00 %	0.00 %
CSL_Sense	0.59 %	0.10 %	1.04 %	0.59 %	0.69 %	0.00 %	0.00 %	0.00 %
CSL_StayInTouch	1.69 %	0.38 %	2.45 %	1.69 %	1.71 %	1.25 %	0.00 %	0.00 %
Drift_Corrector	1.95 %	0.12 %	2.45 %	1.95 %	1.95 %	1.25 %	0.00 %	0.00 %
StandByClock	0.22 %	0.09 %	0.54 %	0.22 %	0.40 %	1.25 %	0.00 %	0.00 %
MIDIInterface	2.26 %	0.44 %	3.97 %	2.26 %	2.60 %	0.00 %	0.00 %	0.00 %

Table 2: Stay In Touch Light System Resource

Contrary to the complete system, the light version makes uses of far less resources of the FPGA. Due to the reduction of the connections between the core modules and the graphic modules, the *CSLControl* module uses 4,46% of the resources and not 7,02% (As the system is more efficiently built up by *Vivado*). Additionally, the light system does not use any RAM resources.

4.4.2.2. Power Analysis

This section makes a prediction on the system power consumption of the FPGA (Not regarding the other elements such as the *Pmods – DAC ADC Pmod* and *Serial Pmod* are supplied externally – or the MIDI Keyboard).

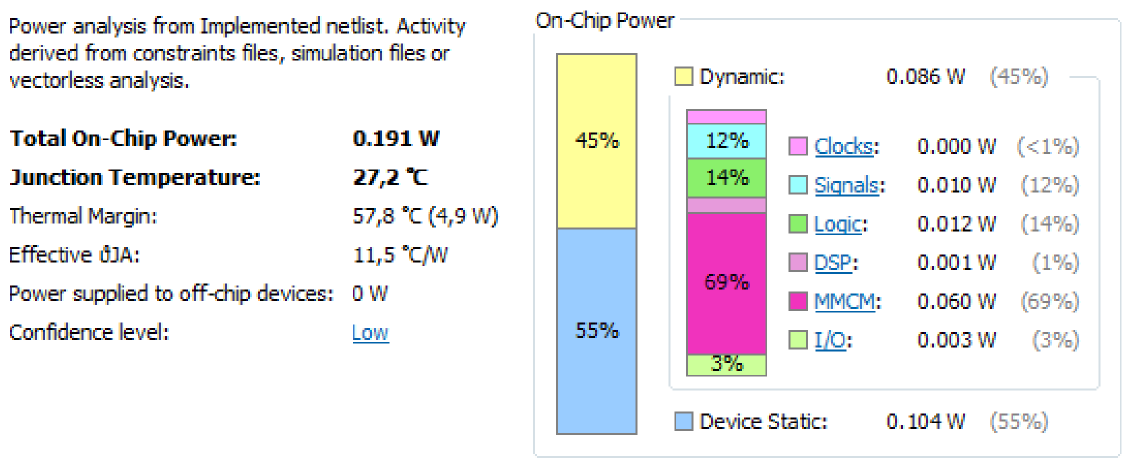


Figure 44: Stay In Touch Light System Power Consumption

As seen in the graphic, the power consumption is a lot lower than in the complete system. The VGA Screen wouldn't be present in this system reducing the consumption even more. The *DAC ADC Pmod* consumption would remain the same.

5. Haptic Perception Study

This section contains all the study of the Human Haptic Perception that has been done to correctly develop the *CSL Fingerlike Mechanism* system.

It content is not meant to be a deep study into the huge field of haptic perception but an introductory approach.

5.1. Preliminary Study

5.1.1. The importance of Haptic Interaction

"Haptic perception is touch perception in which both the cutaneous and kinesthesis convey significant information about distal objects and event and body parts."[5]

The need of designing proper Human-Machine interfacing systems (HMI) is extremely challenging for the engineering community due to the great importance this kind of interaction has for the humans.

In human social relations, haptic interaction plays a fundamental role as it is the main sense to transmit affection and love. It is basic to build up confidence towards another individual and represent an important factor in the human expression.[6]

Through a comforting haptic interaction, the neuronal indicators of pleasure increase both in the giver and the receiver. It causes a decrease of the stress-related adrenal hormones and an increase of the oxytocin.

It is also basic in the development of the emotional, cognitive and physical development as studies show how a lack of haptic interaction in early stages of childhood can result in great further handicaps in the adult life.

Harlow, for instance, developed an experiment involving baby monkeys that were raised isolated from other primates and their only maternal figures were two dummies. One contained milk (food) while the other was just soft and warm. The result was that the baby monkeys preferred the second one and only draw on the first one when they were hungry. The conclusion of this experiment was that for primates,

[5] Martin Grunwald (2008) Appendix. Human Haptic Perception: Basis and applications: 162. Martin Grunwald(Ed)
[6] Martin Grunwald (2008) Haptic behavior in social interaction. Human Haptic Perception: Basis and applications: 155-162. Martin Grunwald(Ed)

haptic interaction and care represents not a secondary need but a primordial condition in early development.[7]

Other researches summarized by Montagu[8] and Spitz[9][10] show how in orphanages of 19th and early 20th century well feed and protected children grew weak and sickly due to the lack of haptic caring, resulting in high mortality rates.

Regardless of the final application, the automated haptic HMI has to be taken into account. Through the development of both hardware and software, valid and realistic haptic devices have to be made in order to credibly replicate the human haptic complexity. The applications in which these technologies can be applied are countless. Nursing machines or robots designed to take care of children, elder or physically handicapped need to create a comfortable ambience and a caring relation. Domestic robots must be regarded with empathy by the user rather than just a helping machine. Androids or sex-oriented robots need to replicate in a realistic way the feel of interacting with a real human being.

"In short, haptic behavior is the sine qua non of interpersonal interaction in all close relationships and perhaps the most basic and fundamental form of human communication." [11]

5.1.2. Haptic object perception

To correctly achieve develop a fingerlike system, first an insight into how does the human use their haptic perception has to be done.

This insight focuses only in active perception, this is, according to Gibson[12][13], the haptic perception that involves not only the cutaneous and kinesthesis perception of the human being but also includes the different active strategies it uses to maximize the amount of information received.

Lederman and Klatzy[14][15] developed an experiment to explore and define this strategies used to obtain different haptic information about objects. They concluded the existence of stereotyped exploratory procedures (EPs). These EPs are specific hand movement humans use to extract a certain type of information out of the haptic perception (texture, hardness, temperature, weight, volume or shape). Other conclusions where that material properties were processed by haptic perception better than geometric dimensions (size, symmetry, curvature).

The existence of EPs and them to be used equally by different subjects means that this strategies (rather cultural or merely biological) are to be imitated by the automated HMI.

Other researches done by Lederman, Klatzty and Metzger[16] pointed out that the haptic recognition of ordinary object familiar to the subject is much more successful. Their experiment involved blindfolded subjects which had to recognize different objects in a short time. The success rate for common objects was of 99% and the time needed very low (around 2 seconds).

This experiments show the importance of implementing data bases from which the HMI can extract models to fast identify and compare its own information. If for example a cloud servers allowed

---

[7] Harlow HR, Harlow MK, Hansen EW (1963) The maternal affectional system of rhesus monkeys. In HL Rhinegold (ed): Maternal behaviors in mamals. Wiley, New York, 254-281
[8] Montagu A (1978) Touching: The human significance of the skin. Harper & Row, New York
[9] Spitz RA (1945) Hospitalism: An inquiry into the genesis of the psychiatric conditions in early childhood. Psychoanalytic Study of the Child 1: 53-74
[10] Spitz RA (1946) Hospitalism: A follow-up report on investigation described in Volume I, 1945. Psychoanalytic Study of the Child I: 113-117
[11] Martin Grunwald (2008) Haptic behavior in social interaction: Summary. Human Haptic Perception: Basis and applications: 162. Martin Grunwald(Ed)
[12] Gibson JJ (1962) Observations on active touch. *Pshych Rev* 69:477-491
[13] Gibson JJ (1966) The senses considered as perceptual systems. Houghton-Mifflin, Boston
[14] Lederman SJ, Klatzky RL (1987) Hand-movements: A Windows into haptic object recognition. *Cog Psych 19*: 342-368
[15] Lederman SJ, Klatzky RL (1990) Haptic classification of common objects: Knowledge-driven exploration. *Cog Psych* 22: 421-459
[16] Klatzky RL, Lederman SJ, Metzger V (1985) Identifying objects by touch: An expert system. *Perc & Psych 37:* 299-302

similar HMI both to extract and add new models, a big collection of valuable data could be gathered fast and trustworthily.

5.2. Haptic perception experiment

This section contains both the description and the result of a haptic perception experiment.

5.2.1. Experiment Description

5.2.1.1. Purpose

The purpose of this experiment is to explore the human haptic perception of objects in order to design algorithms for a robotic system to interact with its environment in the same way. The aimed robotic system is a 2 joint fingerlike mechanism controlled by a CSL system. Therefore, the aimed systems lacks almost all of the human haptic sensors such as temperature, texture or light perception. The system is only able to detect the resistance of the environment to its contact and movement.

According to Thus Jansson and Monaci[17], the object recognition is highly limited when only one finder is used rather than the whole hand or several finger. Therefore, the results expected are not of high object identification or sensibility.

In this scenario, to correctly observe the way a human would interact, it is necessary to limit the senses of the studied subjects to work alike the robotic system. For this study, 2 groups have been designed, each under different conditions but all with the same objective.

5.2.1.2. Control Groups

This section describes the different control groups. Each group is intended to be of 4-6 subjects.

Group 1:

This group has only its sight and hearing suppressed but maintains all the haptic senses. The purpose of the study of this group is to determine the exclusive use of these senses not present in the other groups.

Sense Limitation:

The subjects of *Group 1* are blindfolded and wearing a sound suppresser headphones. The hand is let free but they are told no to use the lateral movement of their index finger and to maintain all the other fingers folded.

Figure 45: Haptic Experiment Sense Limitation Group 1 and 2

[17] Jansson G, Monci L (2004) Haptic identification of objects with different numbers of finger. In: S Ballesteros, MA Heller (eds): Touch, blindness and neuroscience. UNED Press, Madrid, Spain, 209-219

Group 2:

This group has the sight, the hearing and their haptic senses suppressed. They also have one the last finger articulation blocked in order to simulate a 2 joint system alike the aimed robotic system.

Sense Limitation:

The subjects of *Group 2* are blindfolded and wearing a sound suppresser headphones. They also have a solid stick attached to the upper part of their fingers (index) that is to be used to interact with the environment during the experiments (not interfering with the finger folding). The hand is let free but they are told no to use the lateral movement of their index finger and to maintain all the other fingers folded.

Figure 46: Haptic Experiment Sense Limitation Group 2

5.2.1.3. Experiments:

This section covers the different experiment to be carried. All the groups perform the same experiments.

5.2.1.3.1. Experiment 1: Object identification

Once the sense limitation has been applied, the subject stand in front of 4 objects. The objective is for the subject to identify, one by one, the nature of this objects and, if not identifying the object, guess some characteristics (texture, material, etc).

The 4 objects proposed are:

Figure 47: Haptic Experiment Object 1 Lighter

Figure 48: Haptic Experiment Object 2 Rubber

Figure 49: Haptic Experiment Object 3 Sponge

Figure 50: Haptic Experiment Object 4 Glue Stick

During the experiment, the subjects are able to perform movements with their hands (As the final system is to be implemented into a more complex system). The time is limited to 60 seconds for each object.

5.2.1.3.2. Experiment 2: Shape identification

Once the sense limitation has been applied, the subjects stand in front of a LEGO figure built in a random shape. The objective is for the subject to form a mental shape of the figure and afterwards, being liberated from the sense limitation and unable to see or touch the figure again, for the subject to build the shape detected with the same LEGO pieces as the original one has been created.

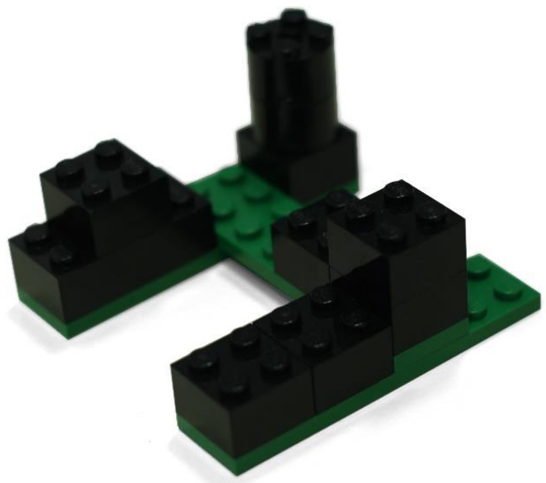


Figure 51: Haptic Experiment Structure

5.2.1.3.3. Experiment 3: Environment exploration

Once the sense limitation has been applied, and immobilizing the hand into a fixed surface, the subjects have to move the finger in order to find and hold a stick hold by the investigator within the range of the subject.

5.2.1.4. Objectives:

Objective 1: Study the recognition of objects with limited haptic perception through the contrast between Group 1 and Group 2.

Objective 2: Study the shape exploring of figures with limited haptic perception through the contrast between Group 1 and Group 2.

Objective 3: Study the potential EPs present in the subjects in their haptic exploration using one finger (index).

5.2.1.5. Investigation:

All the experiments will be recorded to perform further study. An observation code will be designed to correctly analyze the result with binary conditions.

Binary Condition Sheet for Experiment 1:

Experiment 1	**Result Conditions**											
Subject	**Object 1**			**Object 2**			**Object 3**			**Object 4**		
	Detects Object	Detects Material	Fast	Detects Object	Detects Material	Fast	Detects Object	Detects Material	Fast	Detects Object	Detects Material	Fast
Subject 1	Y/N	Y/N	Y/N	Y/N	Y/N	Y/N	Y/N	Y/N	Y/N	Y/N	Y/N	Y/N

Binary Condition Sheet for Experiment 2:

Experiment 2	Result Conditions											
Subject	Global Performing			Surface Dimension			Height Dimension			Distribution		
	Bad	Medium	Good	Bad	Medium	Good	Bad	Medium	Good	Bad	Medium	Good
Subject 1	Y/N	Y/N	Y/N	Y/N	Y/N	Y/N	Y/N	Y/N	Y/N	Y/N	Y/N	Y/N

Binary Condition Sheet for Experiment 3:

Experiment 3	Result Conditions						
Subject	Touch 2º Segment		Touch 1º Segment		Keeps contact until grab	Explore Joint	
	Forward	Back	Forward	Back		1º Joint	2º Joint
Subject 1	Y/N	Y/N	Y/N	Y/N	Y/N	Y/N	Y/N

Additionally to the *Binary Condition Sheet*, all the records will be taken into account and a example published with this document under the name *HapticExperiment_GroupX.avi* so that they can be further analyzed and the results checked. In the final document, only one iconic video of each group will be included.

5.2.2. Experiment Results

This section contains the result sheets of the experiment. Two example videos are present in the attached DVD with the names of: *HapticExperiment_Group1.avi* and *HapticExperiment_Group2.avi*.

5.2.2.1. Group 1 Results

Experiment 1	Result Conditions											
Subject	Object 1			Object 2			Object 3			Object 4		
	Detects Object	Detects Material	Fast	Detects Object	Detects Material	Fast	Detects Object	Detects Material	Fast	Detects Object	Detects Material	Fast
S1(R.Arn)												
S2(M.Mlk)												
S3(D.Srr)												
S4(M.Rod)												
S5(P.Lez)												

Binary Condition Sheet for Experiment 2:

Experiment 2	Result Conditions											
Subject	Global Performing			Surface Dimension			Height Dimension			Distribution		
	Bad	Medium	Good	Bad	Medium	Good	Bad	Medium	Good	Bad	Medium	Good
S1(R.Arn)												
S2(M.Mlk)												
S3(D.Srr)												
S4(M.Rod)												
S5(P.Lez)												

Binary Condition Sheet for Experiment 3:

Experiment 3	Result Conditions

Subject	Touch 2º Segment		Touch 1º Segment		Keeps contact until grab	Explore Joint	
	Forward	Back	Forward	Back		1º Joint	2º Joint
S1(R.Arn)							
S2(M.Mlk)							
S3(D.Srr)							
S4(M.Rod)							
S5(P.Lez)							

5.2.2.2. Group 2 Results

Experiment 1	Result Conditions											
Subject	Object 1			Object 2			Object 3			Object 4		
	Detects Object	Detects Material	Fast	Detects Object	Detects Material	Fast	Detects Object	Detects Material	Fast	Detects Object	Detects Material	Fast
S1(Mar)												
S2(B.Pan)												
S3(Chri)												
S4(Petr)												
S5(Benj)												

Binary Condition Sheet for Experiment 2:

Experiment 2	Result Conditions											
Subject	Global Performing			Surface Dimension			Height Dimension			Distribution		
	Bad	Medium	Good	Bad	Medium	Good	Bad	Medium	Good	Bad	Medium	Good
S1(Mar)												
S2(B.Pan)												
S3(Chri)												
S4(Petr)												
S5(Benj)												

Binary Condition Sheet for Experiment 3:

Experiment 3	Result Conditions						
Subject	Touch 2º Segment		Touch 1º Segment		Keeps contact until grab	Explore Joint	
	Forward	Back	Forward	Back		1º Joint	2º Joint
S1(Mar)							
S2(B.Pan)							
S3(Chri)							
S4(Petr)							
S5(Benj)							

5.2.3. Experiment Conclusion

This section contains the valuable conclusions of the experiment based on the contrast between both control groups and the comments registered during the experiment.

5.2.3.1. Experiment 1

Dimension and shape perception

Despite not being able to identify the objects in *Experiment 1*, all the subjects described the dimensions of the object with an acceptable accuracy. The *Group 2* subjects were able to define the longitude and with of the object as well as an estimation of the height.

Group 2 also was normally able to detect characteristics of the edges and the round corners, as well as detecting complex areas such as the upper part of the lighter (*Object 1*), the lower part of the glue stick (*Object 2*) but without achieving any further conclusions.

Texture perception

The texture and material perception were two characteristics of the objects that the *Group 1* subject (fully haptic capable, where obviously able to describe and identify. Nevertheless, *Group 2* subjects were able to describe simple things about the texture of the objects especially for the sponge (*Object 3*) and the rubber (*Object 4*).

The sponge was described as soft (even mistaken with food) and the rubber as rough and resistant to the slide of the sensing extension. However, the subjects did not risk stopping the identification (Except *Subject 5*) and did not feel sure enough to identify the material.

The lighter and the glue stick were only defined as hard, without any other recognizable characteristics.

Weigh perception

Due to the objects being glued to the ground (and despite this information being provided to the subjects), the weight of the different objects was not something that could be detected in this experiment.

Expected objects

The subjects of *Group 1* did not have any problem recognizing the objects as everyday objects. On the contrary, some of the *Group 2* subjects complained about not expecting the particular objects to be in the experiment and advised to choose objects present in the laboratory or the household.

This defines a tendency for the subjects to try and guess the objects in a wider range when they have wider information about it.

5.2.3.2. Experiment 2

The results of *Experiment 2* were not very different between the two groups due to the complexity of analyzing and memorizing the complex and not rational (meaning rational as known before) figure.

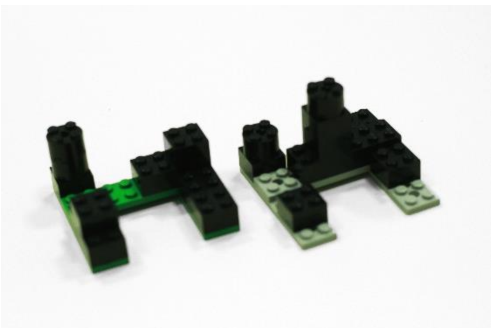


Figure 52: Haptic Experiment 2 Group 1 Result Example

Figure 53: Haptic Experiment 2 Group 2 Result Example

In the opinion of the author, this is relevant as it shows the difficulty of learning new figures and connecting them to other characteristics (such as use or utility) by the haptic perception alone. A robot or autonomous mechanism would also have to make also of other perceptions (visual, by instance) to define these objects and store them as valuable knowledge.

5.2.3.3. Experiment 3

The purpose of *Experiment 3* was to define different EPs (exploratory procedures) in order to apply the to the *CSL Fingerlike Mechanism* system (not finally developed). This experiment reported similar simple results with both groups depending on the external object touching the back part of the finger or the front part. An example of these EPs is described with photographs:

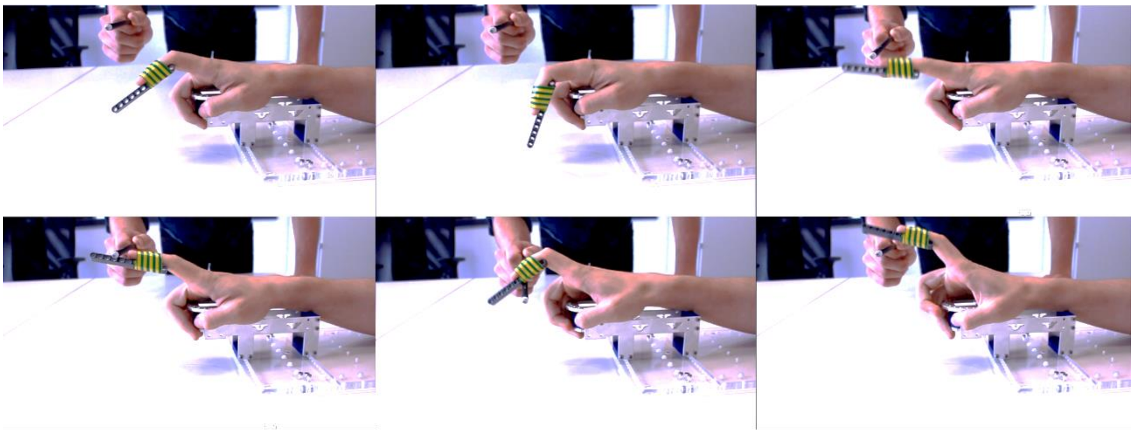


Figure 54: Haptic Exploratory Procedure Example

6. Fingerlike Mechanism System (FM) [Outlook]

The *CSL Fingerlike Mechanism* (FM) system is the second system partially developed during this Bachelor thesis. It has not been fully finished and tested on due to the lack of time and the limited extent of a Bachelor thesis. The aim of this section is to provide a proper starting point for further development of the system.

Some sections like the *Global Resource Use* or the *Power Analysis* are not included as in the previous system as it is irrelevance regarding the system is incomplete.

6.1. System Hardware Overview

The appearance of the system and the names that are shown in this overview are the ones to be used all through the document.

6.1.1. Hardware Added

The Hardware used for this second system is the same as the one used for the *CSL Stay In Touch* system, so instead of repeating the whole section, only the added Hardware is included.

6.1.1.1. Items

LEGO DC Motor
Model: LEGO 71427
Technical information:
https://alpha.bricklink.com/pages/clone/catalogitem.page?P=71427c01#T=C

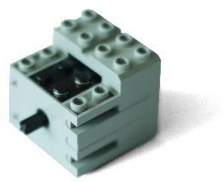


Figure 55: LEGO Motor

Motor Driver & ADC Sensor for CSL Pmod
Model: Laboratory design with the TI ADS1203
Technical information:
http://www.ti.com.cn/cn/lit/ds/symlink/ads1203.pdf

Figure 56: Pmod Motor Drive

System Platform
Model: Irrelevant
Technical information: Irrelevant

Figure 57: System Platform

Motor Structure
Model: Irrelevant
Technical information: Irrelevant

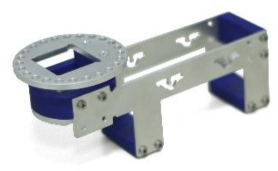


Figure 58: Motor Structure

6.1.1.2. System Connection

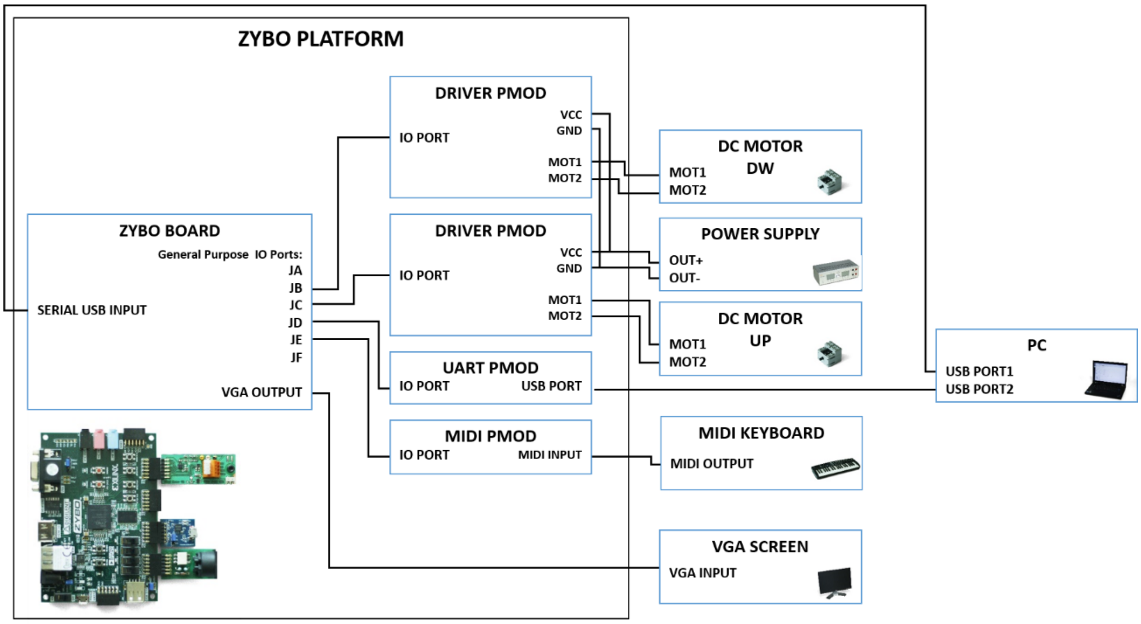


Figure 59: Fingerlike Mechanism System Connection Diagram

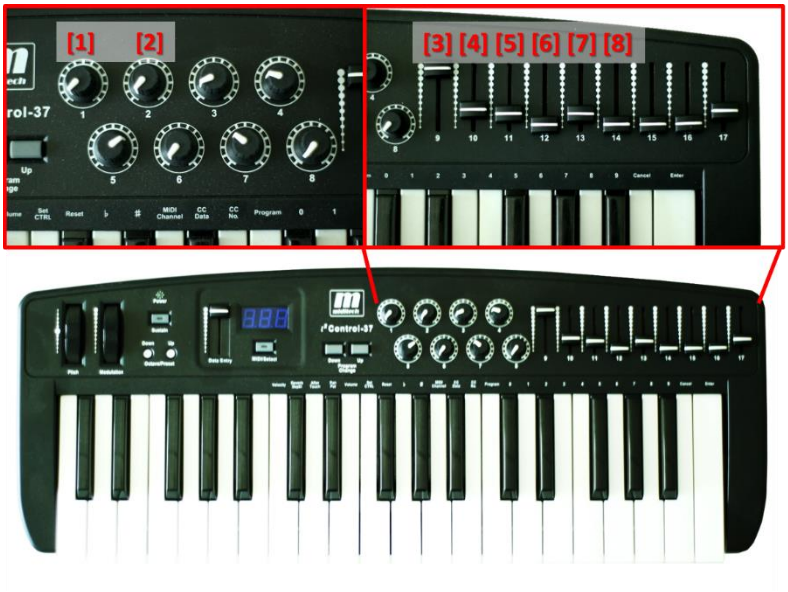
As in the previous *CSL Stay In Touch* system, the core of the system is the ZYBO platform composed by the ZYBO Board and the connected *Pmods*. The MIDI Keyboard and the VGA Screen are *User Interface* elements used to set parameters and monitored the process. The ZYBO board is powered by the PC through the USB connection and the *Driver Pmod* is powered by the *Power Supply* with a voltage of 8V (Mentioned in the configuration section).

6.1.1.3. *User Interface*

This section contains a detailed description of the *User Interface* elements of the system.

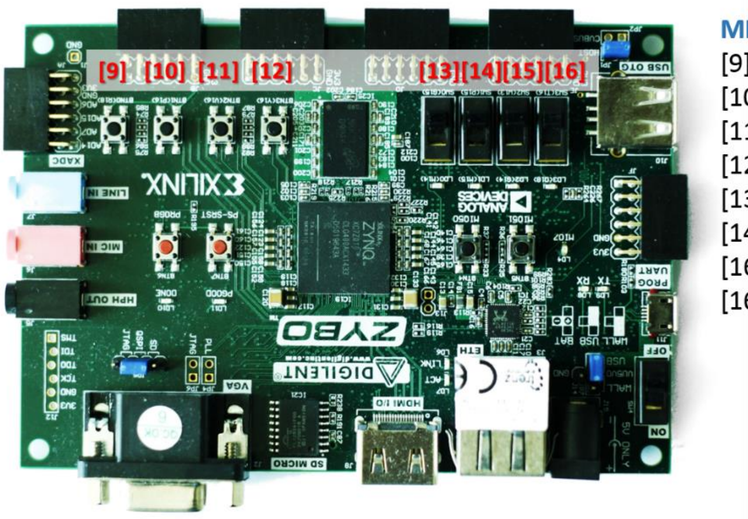
6.1.1.3.1. Parameter Interface

The parameter Interface is both done through the MIDI keyboard and the *ZYBO* Board. The parameter controls of the system are:



MIDI Keyboard Parameters:
[1] Drive Control
[2] Sense Control
[3] Brake Control
[4] Threshold ON
[5] Threshold OFF
[6] None
[7] None
[8] None

Figure 60: Fingerlike Mechanism Keyboard Parameter Reference

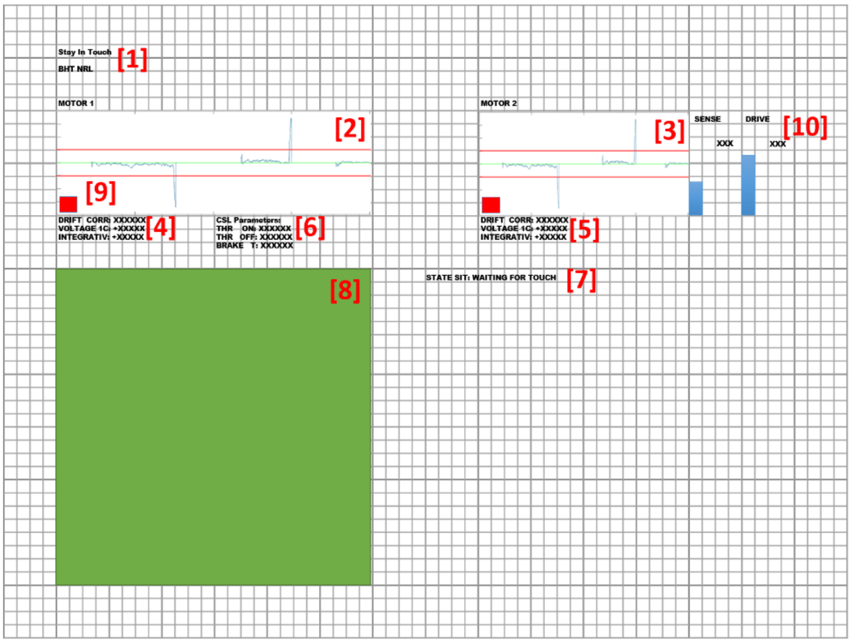


MIDI Keyboard Parameters:
[9] Graphic Chart Pause
[10] None
[11] None
[12] None
[13] Motor UP ON/OFF
[14] Motor DW ON/OFF
[16] None
[16] None

Figure 61: Fingerlike Mechanism ZYBO Board Parameter Reference

6.1.1.3.2. VGA Interface

The different parameters and settings of the system are displayed for monitoring in a VGA Screen showing the following information:



Screen Interface Elements:
[1] Title
[2] Voltage & Threshold Graphic UP Motor
[3] VoltageInt Graphic DW Motor
[4] Sense Measurements UP Motor
[5] Sense Measurements DE Motor
[6] SIT Basic Parameters
[7] SIT State
[8] State Color Display
[9] Graphic STOP signal
[10] DS Parameters

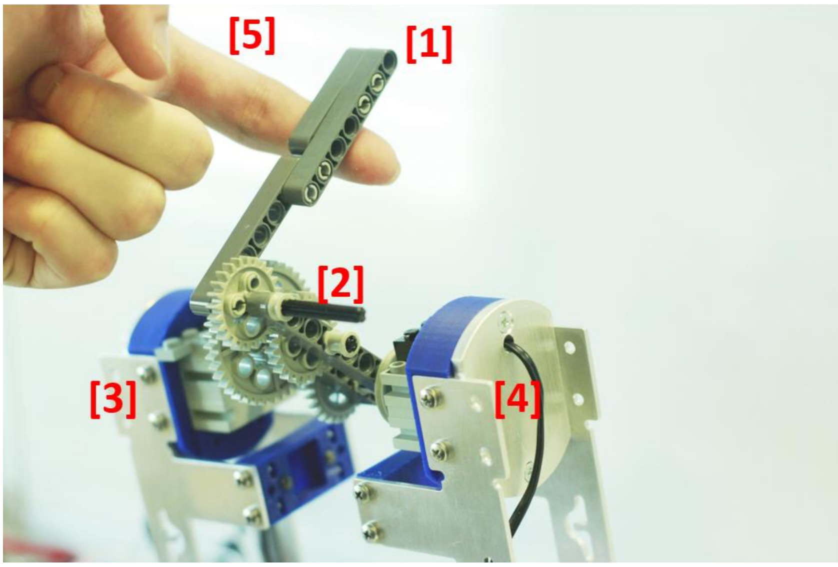
VGA Screen Photograph

Figure 62: Fingerlike Mechanism VGA Interface Schematic

A schematic image has been used due to the difficulty of photographing a VGA Screen. However a small photograph is attached to show the resemblance between both representations.

6.1.2. System Elements

This section contains different labels given to different elements to identify them in the context of the document.



System Elements:
[1] UP Finger Part
[2] DW Finger Part
[3] UP Motor Joint
[4] DW Motor Joint
[5] External Object

Figure 63: Fingerlike Mechanism System Elements

6.2. System Functionality Analysis & Settings

This section contains a description of the System Functionality through its behavior.

6.2.1. Basic System

The *CSL Fingerlike Mechanism* system consist in a 2 joint robotic finger run by 2 LEGO motors. The aim of this system is to try and implement a *CSL Stay In Touch* system in a more complex system dealing with the numerous problems of the coordination of both CSLs. Therefore it makes use of previous modules and parameters with significant modifications.

- **CSL Parameters**
    - ***Sense Period Time:*** Establishes the Sense period duration of the CSL State Machine (in 0,862ms).
    - ***Drive Period Time:*** Establishes the Drive period duration of the CSL State Machine (in 0,431ms).
- **SIT Parameters**
    - ***Threshold ON:*** Establishes the Threshold the voltage measurement of one measurement cycle (Sense) has to surpass in order to activate the IN TOUCH state of the *UP Finger Part*.
    - ***Threshold OFF:*** Establishes the Threshold the voltage measurement of one measurement cycle (Sense) has to surpass in order to deactivate the IN TOUCH state of the *UP Finger Part.*
    - ***Brake Time:*** Establishes the number of Sense periods the motor brakes before a new measurement is done (To avoid invalid readings during the motor relax period) of the *UP Finger Part*.

The basic *CSL Fingerlike Mechanism* system implements only one functionality that consist in finger mechanism which keeps contact with an external object when contacted and once it loses its contact return to the original position (erected) waiting for a new contact. The state machine for the *UP Finger* part of the finger is:

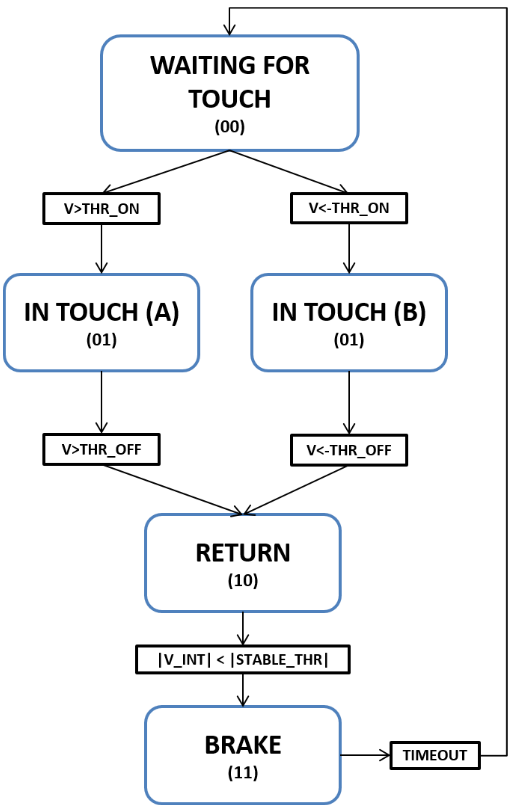


Figure 64: Basic CSL Fingerlike Mechanism State Machine

State Description:

WAITING FOR TOUCH: The motor is stopped and the Drive time is 0.

- IN1 <= 0
- IN2 <= 0

IN TOUCH: The motor is driven with a constant voltage during a *Drive* time set by *Drive Period Time.*

- Direction A: IN1<=0; IN2<= 1
- Direction B: IN1<=1; IN2<= 0

RETURN: The motor is driven to return to an erected position through a CSL integrative voltage run, seeking a stable position.

- IN1 <= not(*VoltageInt*(17))
- IN2 <= *VoltageInt*(17)

BRAKE: The motor is stopped by a voltage shortcut both in the *Sense* time. Drive time is 0.

- IN1 <= 1
- IN2 <= 1

The *DW Finger Part* is driven with a classic CSL stability system through the Voltage Integrator which only aims to reach an equilibrium position. This has two objectives: The first one is to return to the original position once a contact has finished, the second is to oppose the *In Touch* upper finger state and ensure the structure remains as erected as possible. Both motor are run by the same *Sense* and *Drive* parameters.

6.2.1.1. Configurations

Even though the system does not work properly, two different configurations have been studied and recorded in order to analyze different possibilities and illustrate the possible problems this system might have to solve if further developed by some other student in the future.

The configurations studied do not cover all the possible setting combinations the system can have. All of these configurations are shown in the video *FM_BasicBehavior.avi*. The different configurations are properly labelled in the video with the same references shown in this document.

The value of the configuration parameters is shown both in a qualitatively and quantitative way as in the *CSL Stay In Touch* study. All this configurations have been implemented powering the *ADC DAC Pmod* with a 6V voltage.

The configurations studied are:

Configuration 1 (Low Instability):

This configuration presents a low instability behavior of the *CSL Fingerlike Mechanism* system. It present a right haptic interaction and contact but lacks a proper stability after the *Return* state, presenting a constant oscillation of the *DW Finger Part.*

Configuration 1	Low Instability	
CSL Parameters:		
Sense Time	VERY LOW	004/127
Drive Time	MEDIUM	064/127
SIT Parameters:		
Threshold ON	MEDIUM	360/508
Threshold OFF	VERY HIGH	8128/8128
Brake Time	MEDIUM	140/317

The medium *Threshold ON* is set to avoid the *UP Finger Part* to trigger a *In Touch* state because of the oscillations of the *DW Finger Part* during the stabilization process. Even with this precaution, this situation often happens making the system unstable.

Configuration 2 (High Instability):

This configuration presents a high instability behavior of the *CSL Fingerlike Mechanism* system. Due to the low *Threshold ON*, the *DW Finger Part CSL* system triggers the *In Touch* state constantly entering an instable behavior not reaching nor an erected position nor a *Waiting For Touch* state.

Configuration 2	High Instability	
CSL Parameters:		
Sense Time	VERY LOW	004/127
Drive Time	MEDIUM	064/127
SIT Parameters:		
Threshold ON	LOW	32/508
Threshold OFF	VERY HIGH	8128/8128
Brake Time	MEDIUM	140/317

Configuration 3 (Low DW Drive):

This configuration presents an invalid behavior of the *CSL Fingerlike Mechanism* system. Due to the low *Drive Time*, the *DW Finger Part CSL* system is not able to properly move the whole mechanism

Configuration 3	Low Drive	
CSL Parameters:		
Sense Time	VERY LOW	004/127
Drive Time	LOW	015127
SIT Parameters:		
Threshold ON	MEDIUM	360/508
Threshold OFF	VERY HIGH	8128/8128
Brake Time	MEDIUM	140/317

6.3. Module Description

This section contains a detailed description of the different modules that conform the system. The modules that are to be displayed are not all original, as some are ready-made and directly implemented or modified and adapted from previous designs or the *CSL Stay In Touch* system. All the modules will be labelled as *ORIGINAL*, *MODIFIED* or *INHERITED*. Some of the modules are not described due to their low importance in the system behavior.

All the Block Diagrams shown can be seen in further detail in the *.pdf* files attached to the document.

6.3.1. Global System

The Main system of the *CSL Fingerlike Mechanism* system is very similar and heavily based on the *CSL Stay In Touch* system and therefore the description of its main characteristics is not described here to avoid redundancy of content. For a more detailed schematic refer to the *FM_Global_SCH.pdf* attached to the document.

The signal buses simplified in this schematic (*CSL RAW PARAMETERS*, *MODE PARAMETERS*) can be analyzed in further detail in the module interfaces present in the module descriptions.

The main difference in the global system is only the existence of another motor.

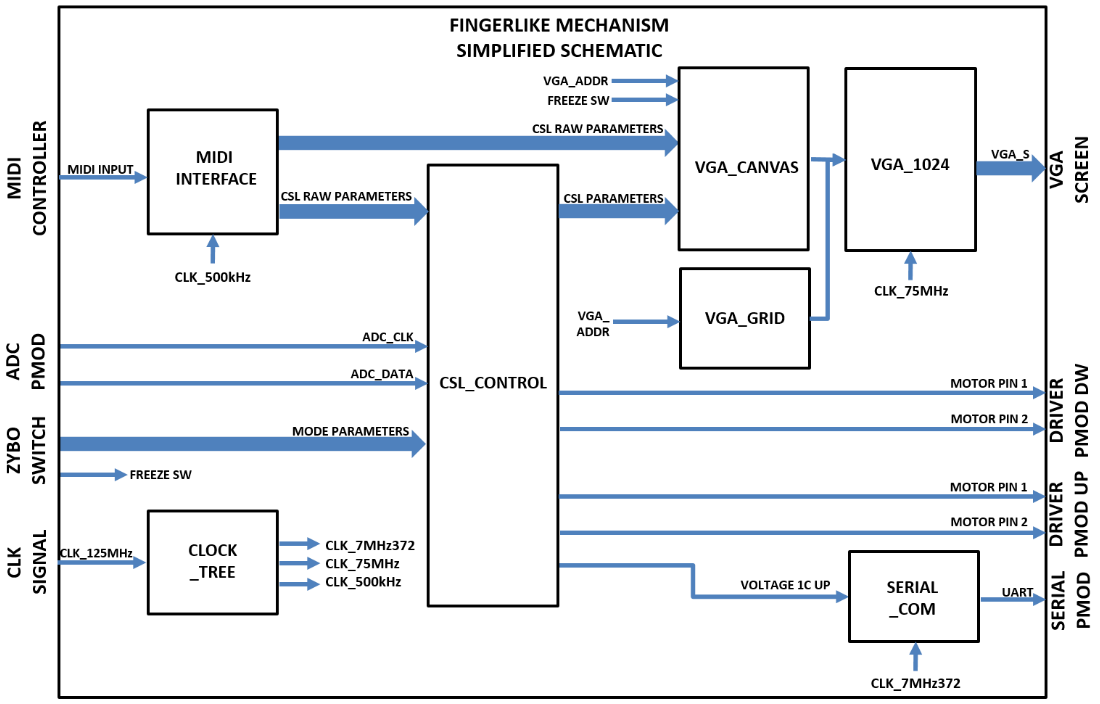


Figure 65: Fingerlike Mechanism Main Simplified Schematic

6.3.2. CSL Control [ORIGINAL]

As in the previous design, the *CSL_Control* is the core of the system. The *CSL_Control* is built up by several modules. This simplified schematic shows the dependencies between modules. For a more detailed schematic refer to the *FM_CSLControl_SCH.pdf* attached to the document.

The *CSL_Control* design is organized in functional blocks that separate the various tasks the sub-system performs. These modules are:

- **CSL_UpperFinger:** Controls the behavior of the *UP Finger Part* system through the SIT (*Stay In Touch*) state machine and the DS (*Drive-Sense*) state machine.
- **CSL_LowerFinger:** Controls the main behavior of the *DW Finger Part* system through the DS (*Drive-Sense*) state machine.
- **CSL_Sense:** Controls the ADC Motor Voltage input during the *Sense* period of the DS state machine.
- **Drift_Corrector:** Generates a Drift Correction for the *CSL_Sense* module.

The signal buses simplified in this schematic (*CSL RAW PARAMETERS*, *MODE PARAMETERS*) can be analyzed in further detail in the module interfaces present in the module descriptions.

All the modules made with double low bar represent a group of similar modules rather than just one.

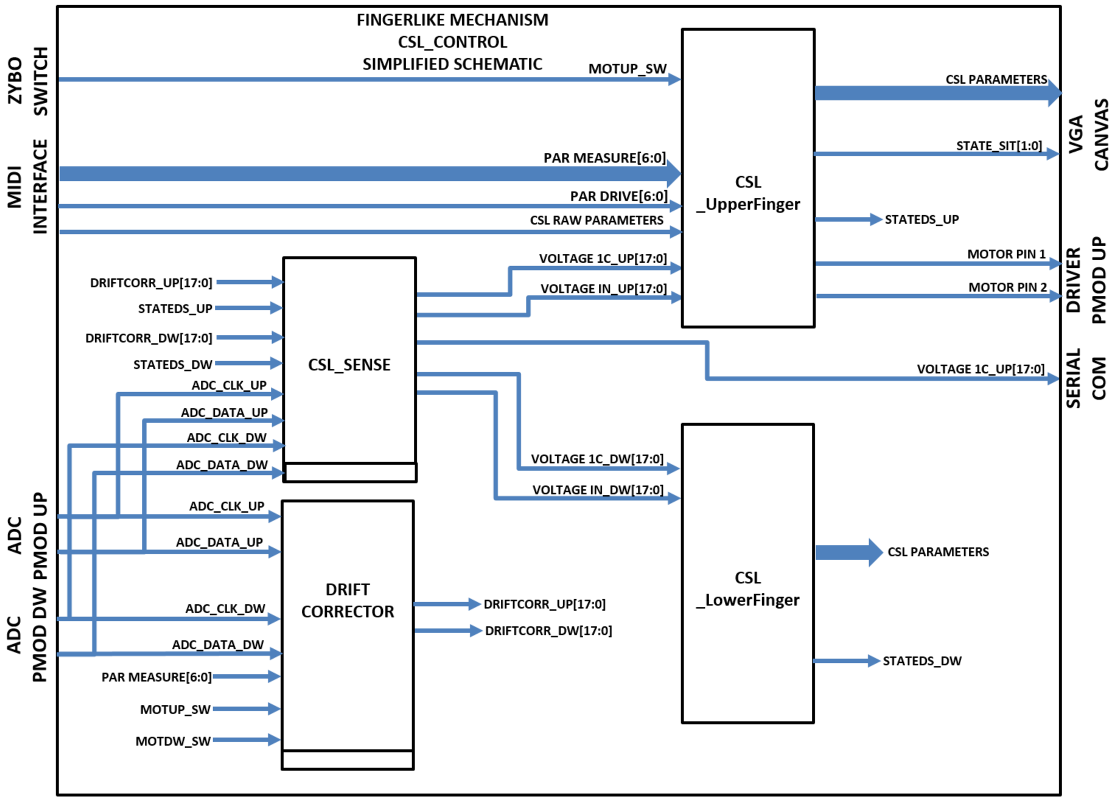


Figure 66: Fingerlike Mechanism CSL Control Simplified Schematic

The *DriftCorreactor* module and the *CSL_Sense* are similar to the *CSL Stay In Touch* system with little changes in the SIT state dependencies.

6.3.2.1. CSL_UpperFinger

The *CSL_Upperfinger* module is the most complex of all the system and controls all the state machines as well as the direct motor drive signals for the *UP Finger Part*. Its interface has the following described signals:

INPUT SIGNALS	OUTPUT SIGNALS
mclk: ADC Pmod Clock signal 10MHz. Behaves as the Clock signal of all the processes. **MOTsw:** Motor enable switch (ON/OFF) **ParMeasure[6:0]:** Establishes the *Sense* period duration of the *CSL State Machine* (in ms). **ParDrive[6:0]** Establishes the Sense period duration of the *CSL State Machine* (in ms). **BrakeCtrl[6:0]:** Input RAW value of the *Brake Time* (0-127). **ThrCtrl_0[6:0]:** Input RAW value of the *Threshold ON* (0-127). **ThrCtrl_1[6:0]:** Input RAW value of *the Threshold OFF* (0-127). **Voltage[17:0]:** Value of the voltage measurement in one *Sense* cycle **VoltageInt[17:0]:** Value if the voltage integrated measurement.	**IN1**: Motor pin 1 signal. **IN2:** Motor pin 2 signal. **M0:** Configuration Pins for ADC Pmod. **M1:** Configuration Pin for ADC Pmod. **Sleep:** Energy saving configuration pin. **StateDS**: Drive-Sense state. **StateSIT[1:0]:** Stay In Touch state. **Direction:** Motor rotation direction (A for 0, B for 1). **ThrCtrl_ON[17:0]:** Threshold ON voltage. **ThrCtrl_OFF[17:0]:** Threshold OFF voltage. **BrakeCount[24:0]:** Brake time period (In complete DS cycles).

The only functionality that is going to be described is the new *Return* state as is the only main difference from the *CSL_StayInTouch* module from the previous system.

Return state

The Return state is radically different from the one in the previous system as its aim is to return the system to an erected position and therefore makes use of the CSL delta-sigma converted voltage integration (*VoltageInt*) to drive the motor until the system reaches a stable position defined by a threshold not parameterized (Not dependent of the user preferences).

Once this stability threshold has been reached, the SIT state machine turns to the *Waiting For Touch* state ready for a new haptic interaction.

```
when others => -- RETURN with CSL VoltageInt
   IN1 <= not(VoltageInt(17));      -- sign for rot. direction
   IN2 <= (VoltageInt(17)); -- sign for rot. direction
   timerReturn <= timerReturn - 1;
   timer <= "000" & (unsigned(abs(signed(VoltageInt(17 downto 3)))))*unsigned("00" &
ParDrive(6 downto 2));
   if ((abs(signed(VoltageInt(17 downto 7))) = 15d"0")) then
       StateSIT <= "11"; -- go to BRAKE
       timerBrake <= unsigned(BrakeCount); -- Pause timecontrol in ms
   end if;
end case;
```
VHDL Code 33:Fingerlike Mechanism Upper Finger Return State

The main problem of this stabilization process is that, due to the mechanism, the stability of the *UP Finger Part* can be reached without the mechanism being erected.

6.3.2.2. CSL_LowerFinger

The *CSL_Lowerfinger* module controls the *DW Finger Part*. Its interface has the following described signals:

INPUT SIGNALS	OUTPUT SIGNALS
mclk: ADC Pmod Clock signal 10MHz. Behaves as the Clock signal of all the processes. **MOTsw:** Motor enable switch (ON/OFF) **ParMeasure[6:0]:** Establishes the *Sense* period duration of the *CSL State Machine* (in ms). **ParDrive[6:0]** Establishes the Sense period duration of the *CSL State Machine* (in ms). **Voltage[17:0]:** Value of the voltage measurement in one *Sense* cycle **VoltageInt[17:0]:** Value if the voltage integrated measurement.	**IN1**: Motor pin 1 signal. **IN2:** Motor pin 2 signal. **M0:** Configuration Pins for ADC Pmod. **M1:** Configuration Pin for ADC Pmod. **Sleep:** Energy saving configuration pin. **StateDS**: Drive-Sense state.

The functionality of this module is implemented by a CSL delta-sigma converted voltage integration (*VoltageInt*) control. The stable position is defined by a threshold not parameterized (Not dependent of the user preferences).

```
CSL: process(mclk) begin
    if mclk'event and mclk = '1' then
    case(StateDS) is
        when '0' => -- measure
            timer <= timer - 1;
            if (timer = 25d"0") then -- END of Mesure
                IN1 <= not(VoltageInt(17));      -- sign for rot. direction
                IN2 <= (VoltageInt(17)); -- sign for rot. direction
                timer <= "000000" & (unsigned(abs(signed(VoltageInt(17 downto
8)))))*unsigned("00" & ParDrive(6 downto 0));
                StateDS <= '1'; -- Turn to Drive
             end if;
       when '1' => -- drive
                timer <= timer - 1;
                if (timer = 25d"0") then -- end of drive phase
                    timer <= (18d"9999" * unsigned(ParMeasure)) + 25d"9999"; -- prepare
Measure-time, fix value: 18d"999" -> 0,1 ms, 0,1 ms minimum time
                    IN1 <= '0';      -- sign for rot. direction
                    IN2 <= '0'; -- sign for rot. direction
                    StateDS <= '0'; -- Turn to Sense
                end if;
     end case;
     end if;
end process;
```

VHDL Code 34: Fingerlike Mechanism DW Finger Part Control

6.3.3. ClockTree [MODIFIED]

In order to use two *Pmods* working at the same time, the previous use of the ADS1203 internal oscillator was not valid due to the incoordination of both *Pmods*. The solution given to this problem was to use the ADS1203 in *Mode 3*.

Mode 3 defines the ADC clock externally with the clock signal being an input in the *mclk* port. This signal had to be generated then in the *CSL Fingerlike Mechanism* system. The input clock signal defines the ADC clock signal in a 2:1 relation with a range of 1MHz to 32MHz.

The solution to this problem was to generate two signals with the *ClockTree* module with simple counters to generate a 20,84Mhz and a 10,42Mhz signals from the 125MHz system clock.

```
    BUFG_1: BUFG port map ( Clk_75MHz, ClkU_75MHz );
    BUFG_2: BUFG port map ( Clk_12_288MHz, ClkU_12_288MHz );
    BUFG_3: BUFG port map ( Clk_3_072MHz, ClkU_3_072MHz );
    BUFG_4: BUFG port map ( Clk_500kHz, Counter2(0) );
    BUFG_5: BUFG port map ( Clk_250kHz, Counter2(1) );
    BUFG_6: BUFG port map ( Clk_48kHz, ClkU_48kHz );
    BUFG_7: BUFG port map ( Clk_7MHz372, ClkU_7MHz372 );
    BUFG_8: BUFG port map ( Clk_20_83MHz, ClkU_20_83MHz);
    BUFG_9: BUFG port map ( Clk_10_42MHz, ClkU_10_42MHz);
```

```
end Behavioral;
```
VHDL Code 35: Fingerlike Mechanism ADC Clocksignal Generation

The *Mode 3* is set in the main module.

```
-- ADS1203 Mode3 Clock
    m0_1 <= '1';
    m1_1 <= '1';
    m0_2 <= '1';
    m1_2 <= '1';
    mclk_1 <= Clk_20_83MHz;
    mclk_2 <= Clk_20_83MHz;
```
VHDL Code 36: Fingerlike Mechanism ADC Mode 3

6.4. System Improvements

This section contains all the main improvements which according to the author thinks should be first applied to this incomplete system in order to be developed ahead in the system.

6.4.1. Mechanism Improvement

The mechanism built for this system is weak and unstable. It was done quickly and only with the LEGO pieces available in the laboratory.

The main problem here is the lack of sensibility the motors have. Some haptic interactions do not transfer any movement to the motor and the motor lacks control of the mechanism. The mechanism is also very weak and not able to perform high forces need to return the mechanism to an erected position in some configurations.

Being the author of this document a telecommunications students with little mechanical knowledge I was not able to design and build a proper mechanism. It seems clear that the design of a far better mechanism should be the starting point of a future system using this piece of research.

6.4.2. Stability Improvement

The return behavior could not be fully developed due to the lack of time. A way to improve the instability situation and the failures in returning the mechanism to an erected position is to create dependencies between both motors states as in this system they act independently.

7. Conclusion

This section contains various conclusions of the system done and the work in the Bachelor thesis.

7.1. Main Difficulties

Throughout the development of the system, a number of difficulties were encountered.

7.1.1. Working Tools

During the first weeks of my work in the Bachelor thesis, I met some difficulties with the installation of the *Vivado* environment as the *Webpack License* did not work in my personal laptop as it caused a certain problem with the *Zybo Board* drivers not allowing me to test my designs in hardware. Additionally, I had problems with the VHDL test environment of the V*ivado* due to the license.

The problem was solved by the *Fachbereich VII* of the *Beuth Hoschschule* (BHT) by providing me with a PC laptop so that I could continue the work on my thesis. This way, the driver problem was solved but the *Vivado License* problem with testing continued. As this system worked with Hardware, most modules were not suitable to be tested through simulations.

7.1.2. CSL Understanding

The first difficulty I faced was to grasp how the CSL system worked and understand how to modify the parameters in order to achieve the desired performance. Despite all the documentation at hand, this was mainly carried out through the analysis of inherited modules from previous designs and punctual doubts solved rather by Benjammin Panreck or *Dr. Prof.* Hild.

Once this difficulty was solved, the systems were developed a lot faster as the comprehension of the system core was complete and I could center on other parts of the system.

7.1.3. Synthesis Time Duration

A big problem during the project was the long time *Vivado* took to synthesize the designs particularly when the VGA functionality was included (in all systems except the *CSL Stay In Touch Light* system). This long dead periods made the whole process a lot slower and obstaculized a proper working rhythm and concentration.

7.1.4. Video and Photograph Edition

The edition of the material for the Bachelor thesis needed a large amount of work to achieve the level required. This meant the edition of every single photograph to remove its background and the rendering of videos which included graphs or extra information.

Additionally, these videos had to be edited several times to include the recommendations done by the tutor or other NRL member in terms of aesthetics.

7.2. Main Conclusion

The objectives of this Bachelor thesis set in section 2 (Objectives) were to develop two systems (*CSL Stay In Touch* and *CSL Fingerlike Mechanism*) using the *CSL Control loop*. The *CSL Stay In Touch* system has been totally developed and documented while the *CSL Fingerlike Mechanism* system (Optional) has only been partially started due to the lack of time and the limited dimension of a Bachelor thesis.

During my work with the CSL algorithm, I have concluded that it has great potential in the field of human-haptic interfaces but it cannot be used all alone by an autonomous machine to retrieve haptic information of the environment. In my opinion, haptic perception based only on the resistance to force (motor spin) can only retrieve information about shapes and dimensions and very partially about textures, being needed other sensors such as temperature or position sensors. On the contrary, if implemented, for example, in a robot hand, it could provide an optimal force and haptic sensation to grab or touch both objects and sensitive surfaces (skin, tactile screens, etc). Due to the characteristics of the CSL control loop, a number of decentralized systems could work autonomously with a very good performance.

This Bachelor thesis has suited all my expectations and I cannot imagine a better field to finish my degree than robotics. I have improved my VHDL knowledge and my abstract capability to deal with complex systems. Also, the need to study the haptic perception has open me a window to the complexity of developing systems meant to interact with human beings and the need to take into account a number of non-technical but crucial parameters.

8. Acknowledgments

I would greatly like to thank *Dr. Prof.* Hild for giving me the unique opportunity to develop my Bachelor thesis in the *Neurobotics Research Laboratory*. Also to all the members of the laboratory (Markus Janz, Christian Thiele, Peter Hirschfeld, Jörg Meier and Stefan Bethge) for creating a superb working environment and being always willing to help me and solve any doubts as well as making me feel integrated in the team and to *Dr. Prof.* Gober for his help in many issues.

Special thanks to Benjamin Panreck for his friendly guidance and attention during all the semester.

Thanks, too, to all the people that performed the haptic experiment for their patience and thrust. I would also like to express my gratitude to Maria Rodriguez and Marta Fernandez for their advice during the design of the haptic experiment procedures.

Special dedications to my IEEE Student Branch AETEL for all the great times it has provided me with during my last years and hopefully the years to come.

I would also like to mention my gratefulness to Pablo Lezcano with whom I have fought side by side in the trenches during all the exchange year.

This thesis is dedicated to my parents for their loving support during all my studies.

9. Figure Index

10. Table Index

11. VHDL Code Index

12. Statement of Authorship

I declare that I completed this thesis in my own and that information which has been directly or indirectly taken from other sources has been noted as such. Neither this nor a similar work has been presented to an examination committee.

Berlin, August 7, 2015 ……………………………………………..